Team LibNav
Midterm Progress Report
CS Senior Capstone

CS462 (Fall 2017)

Authored by
Nathan Healea, Stephen Krueger, Matthew Zakrevsky

ABSTRACT

This Progress Report contain information regarding the progress made in the CS Senior Capstone class by Team LibNav. This document will cover A recap of the project purposes and goals, a description of were the LibNav team is currently at, a description of problems that have impeded our progress and the solution to those problems, a retrospective the of fall term 2016 and the past 6 weeks of winter 2017.

CONTENTS

1 Purpose Goals

Libnav main purpose is to provide navigation to desired locations through the use of an interactive map for patrons and visitors of Oregon State University Valley Library. By using a start location and destination, Libnav will provide a navigation line through the multiple floors of the library to help guide users. Search for locations by attributes or name will help make finding a desired spaces or location an easier task

Libnavs goal is to make searching and navigating to visitor destination an easier and quick er task. Libnav will accomplish through a web application that is mobile friendly, and uncluttered by extra features. With a fully editable map, members of the library staff will be able to update locations and spaces as needed. Libnav is designed to be portable and usable with both visitors and staff in mine, allow for the application to be used for years to come.

2 Intel NUC (Nathan Healea)

2.1 Current Status

The Intel NUC has become an extra component of the Libnav project. Currently, the NUC has been rebuilt three times to fix issues regarding Grub at startup, partitions, and multiple installed operating systems. Grub became an issue since it needs human interaction to continue system startup. By doing a fresh install of CentOS 6.8, removing all partitions, and removing Grub, we can now start the NUC without interaction. At this time, user accounts have been created for all members of the team with access to sudo commands and allows connection via SSH.

Apache, Mysql, PHP and NodeJS have been installed and configured. MySQL has user accounts for all members of the team with connections from anywhere via grant permission in the database. In addition to this, a user for the application has also been added to MySQL.

Our projects latest version has been cloned, configured, and runs on the NUC. Necessary tables for Libnav have been created in MySQL and current functionality has been tested on the NUC to insure compatibility of our project.

2.2 Problems that have impeded progress

1) The problem we currently are facing is issues with connecting to the NUC via port 3000. On the local host of the NUC, the project can be reached via port 3000 but not from an outside source. This issue, so far, has been narrowed down to the firewall rules within the Kelley Engineering building.

2.3 Solutions

1) To solve the issue of connection via port 3000, we first got in contact with Kevin. Kevin has placed a request with IT in Kelley to open port 3000. As of now, the port is still not open and Kevin is working with IT on our behalf.

3 User Interface (Nathan Healea)

3.1 Current Status

Libnavs User Interface is based on a template Oregon State University uses for Drupal sites called Pine. Oregon State University offers a variant of the Drupal template for basic web pages. After acquiring this template, major structural changes to the HTML had to be made to fit the standard structure of Bootstrap. Once the structure was refactored, new CSS was written to mimic the Pine template. The new template is used for user login, location forms, Librarian Administration Dashboard views, and map navigation views. A second refactor had to be done to the Libnav template once the Bootstrap SB Admin 2 template was introduced for the Librarian Administration Dashboard and the main map navigation view needed a sidebar. To keep the design looking the same, a new CSS was written to incorporate the Pine color scheme, look, and feel to the Librarian Administration Dashboard, and map navigation views.

In the Librarian Administration Dashboard, the majority of the forms have been created and structured to follow Bootstrap standards. Users can select a floor and have the map change to reflect the users choice allowing for new locations to be drawn or selected on the map. Each field in the forms has the appropriate required field check that blocks submission if empty. For tags and attributes, input sanitization has been applied to check for alphanumeric characters including spaces and commas. Anything that violates this rule is replaced with a space. Text is separated into arrays to be inserted into corresponding tables. The above functionality is written in the form-control.js JavaScript file that also handles the Ajax call to the appropriate route method passing data by a POST method. Since this file is robust, this file can be used across all input forms in the dashboard.

In Libnavs main map navigation page, the sidebar still needs to have functionally developed that places queried information from the database in the correct nested sidebar navigation. Functionality that allows the map to be changed depending on the floor the user would like to select a location on still needs to be developed and implemented. This functionality is similar to floor selection in the forms but needs to be refactored to incorporate button clicks instead of drop down changes.

3.2 Problems that have impeded progress

1) Navigation and location selection/drawing has taken some time to get finished due to their complexity and problems the team has to design through. Since these two parts of the project are not complete, it is preventing the forms and other UI from being completed. Since the navigation and location selection/drawing is not complete, data from these features cannot be gathered, formatted, and sent to the route in order to have it saved into the database. This issue then causes another issue with the front map navigation sidebar. Since data is not stored in the database, it cannot be retrieved and displayed in the sidebar.

3.3 Solutions

1) To solve the problem, while we wait for the other portion of the project to be completed, fake data will be introduced into the database in order to move forward with the development of the user interface.

4 Data Sanitization and Processing (Nathan Healea)

4.1 Current Status

As mentioned above, data sanitization has been applied to all form fields that require it. Form-control.js utilizes a javascript library ApprovedJs to check entry field of forms. With built in functionality, ApprovedJs has eased and sped up the process of checking data. If no pre-built function contains the appropriate check, ApprovedJs allows the use of regex express to check data that has been input.

Some processing of data does take place in the routes. Once data has been checked, we have some form data that then needs a foreign key applied before being inserted into the database. Once the main location information has been inserted into the database, the ID of that row is returned to be added to tags, attributes, and data points location (soon to be phased out). This happens by calling a series of functions that have been created in a location model. The location model handles the connection to the database, running the query, and closing the database for any data regarding location information.

4.2 Problems that have impeded progress

1) The same problem that affects the user interface also affects the completion of data sanitization and process. As navigation and location selection/drawing progresses and data is being recorded, new sanitization and processes will need to be completed.

4.3 Solutions

1) The way to solve this problem is to basically wait until those features are closer to completion. Before a major refactor took place, data points were able to be inserted and retrieved from the database as explained below in a later section.

5 User Authentication and Sessions (Nathan Healea)

5.1 Current Status

Libnav user authentication is supposed to utilize Oregon State Universitys Central Access System (CAS). But due to issues with the NUC, as stated above, we cannot develop the login using CAS because our local machine cannot be hardwired into the schools network. With that said, a work around has been developed that turned into a necessary part of the project. As of now, Libnav has a setup script that allows a user to enter a master username and password along with database connection information. This data is validated using ApprovedJs like the forms and is saved into a config.json file. Before the data is saved, usernames and passwords are run through crypto using an aes-256-ctr algorithm with a salt.

When a user attempts to login, the username and password entered is encrypted using the same algorithm and salt as above, and is checked to the username and password encrypted key stored in the config.json file. Once a user is authenticated, a session is started for that user. Each route method checks for an authenticated session before allowing the user to access the view. In the case where a user tries to access a page and is not authenticated, the user is redirected to the login page.

5.2 Problems that have impeded progress

1) The biggest problem that has impeded the progress of the user authentication and sessions is the NUC. Since we cannot access the NUC through port 3000, we cannot develop and test the CAS system with our project. CAS only allows machines that are wired into the network to access the CAS system. Until this issue is resolved, we have to wait to complete this requirement.

5.3 Solutions

1) To attempt to solve the problem with user authentication and the NUC, our team has been working with Kevin and the IT department in Kelley to get the port open and check the NUC for issues.

6 Navigation (Stephen Krueger)

6.1 Current Status

The navigation system is currently functional but not complete. The navigation system produces a svg display grid made of rectangle objects. This grid is generated once the map loads. The grid is the same width and height as the map underneath. On this grid, the admin can mark and unmark squares that designate whether or not that grid square can be used in the generation of the navigation path. This grid will be completely transparent to the user when they load the map and only the generated path will be shown.

There is another grid generated for the purposes of calculating the shortest path. It uses data from the display grid to know what points are and are not walkable. This graph is only changed when the admin changes the walkable points. Otherwise, it will be pulled from the database on page load.

6.2 Problems that have impeded progress

There are two problems with the navigation currently. The first has to do with setting up the database in the correct way in order to store the grid.

The second is trying to figure out how to tie the display grid to the location on the svg map. Since they are two separate layers they do not share information. Even if they did, how would you figure out where the entrance to a large rectangular space is?

6.3 Solutions

The first solution is easy, it just requires a slight modification in the database that will create a table to store the grid.

The second solution is a little more complicated. We have decided to give the admin the ability to mark the display grid with an x y coordinate point when they create the room and enter the tags and attributes. There will be a button that will allow the admin to load the grid over the map and select the entrance location for the room. This will then be saved along with the rest of the room information. When the user wants to navigate between two rooms, the function will pull the grid location from the room info and send it to the shortest path algorithm to be calculated.

7 Path-finding (Stephen Krueger)

7.1 Current Status

The goal of the pathfinding part of the project is to make sure the path that the navigation system is returning is the shortest path between the two locations. As of now, the pathfinding algorithm works very well. It uses an A* pathfinding algorithm from the pathfinding.js library mentioned in the tech document to return the shortest path to the user. The user can see the shortest path displayed on the map after the beginning and starting location have been declared.

7.2 Problems that have impeded progress

One problem we have is finding the path between floors. We had to decide how to handle the user wanting to go from the first floor to the fifth or sixth floor, and whether or not they would take the stairs or the elevator.

7.3 Solutions

The decision actually turned out to be more of a UX solution than the pathfinding solution. We decided that if the user would change floors that wed give them the option of going either to the stairs or to the elevator. Then from the, the user would be presented with an arrow that would load the next floor.

For the pathfinding algorithm, we use the location of the stairs or elevator as the end point on the users current floor, and the start point on the users destination floor. When the new floor loads, the pathfinding algorithm is run again. The destination of the user will be saved in their session and reloaded. The pathfinding algorithm is very fast as it is just a front end script so we do not have to worry about the algorithm increasing load times. Trying to calculate the path between floors would have been complicated and ultimately unnecessary as it wouldnt have any major benefit.

8 Database (Stephen Krueger)

8.1 Current Status

The database is currently set up and running. We can write location data and user session data to the database and also query the database to return that data. The database is also currently setup to store custom drawn locations. We can get the data points back from the database and redraw those locations on the map.

8.2 Problems that have impeded progress

1) How to store the grid
2) Store points as json object

One problem we ran into was figuring out how we were going to store the navigation grid in the database. Its a series of (x,y) coordinates and we need to record both the walkable and non-walkable coordinates.

The second problem we had was figuring out how to store the location objects as a json object. The location data was originally stored as individual data points and when we went to pull the data and loop through the points it was painfully slow. We decided that we would need to refactor the database.

8.3 Solutions

1)

The solution to storing the grid was fairly simple. The set of walkable and non-walkable points are stored in arrays. We simply are going to put these arrays into a javascript object and then stringify the javascript object and store the string in the database. The tables still need to be set up, but after they are the functionality to pull that string out of the database is already there. From there we just turn that string back into a json object and pulling out the arrays

The fix for storing the points of the custom drawn locations is essentially the same. The tables that were used to store the individual data points are going to be deleted. Instead we will opt to store the data points in a json object like the grid. We will then stringify the object of points and store the string in the database. This will make it so when we pull out the points it's already in an nice iterable object format instead of a bunch of individual coordinates.

9 Framework (Matthew Zakrevsky)

9.1 Current Status

As it currently stands the web framework that we are using is the Express framework previously discussed in our design and requirements documents. This framework controls the routing of the web pages and is fully functional with the bootstrapping, mysql post and get queries. The use of Express has allowed us to use routes to move between the various web pages without error, .The framework has been meeting our needs though we have some minor problems in how we want to to pass data along to the mysql controller. This had to do with the need to use asynchronous ajax calls. Overall this major difficulty that we have had is learning the framework.

9.2 Problems that have impeded progress

1) The need to use asynchronous ajax calls to use the mysql controller

9.3 Solutions

1) The need to use asynchronous ajax calls resulted in how we need to route through the framework. We had to have separate route going from form submit and through the our form submission code.

10 SVG Rendering and Map Drawing (Matthew Zakrevsky)

10.1 Current Status

SVG rendering has been one of the primary focuses of our group when developing the LibNav project. This is because this is one of the primary features that must exist in order to make this application function. One of the features that has been developed is to be able to draw shapes through of a point and click interface that allows a library staff member to connect a series of points and create a polygon that is saved to the database.

Additionally the LibNav application can access the shapes stored in the SVG files that we were given by the library faculty. These files contain shapes that will be used to create the different locations within the library. The biggest advantage to using this rather than the point and click method as mentioned above is that this method will make the most use of the data already available to the staff. They will only have to select the spaces in question and enter the relevant form data.

Conversely we are also able to draw shapes from the database. This is done with an asynchronous ajax call which performs a callback to return the data points that match the select query, which currently returns all points from the database. Through the process of developing the application it was found that the use of using the database would take a large amount of overhead to draw all shapes, especially when all floors are fully populated

10.2 Problems that have impeded progress

1) Need to redesign how location information is stored into the database.
2) Exporting data to the database based upon rooms selected from the SVG vs drawn with the added drawing functionality.
3) Passing the necessary data on form submission based upon whether the selected shape was a rectangle, ellipse, or polygon.

10.3 Solutions

1) The solution that we came up with as a team was to refactor our code such that when we insert data into the query into the database we instead are inserting a single string of points instead of a single point. This does two things for the application.The first is that there is less overhead to iterate through each point, match that point with the correct location and then draw the shape, instead we read the needed information for polygons. The second thing this does is that we can reduce the number of tables used in the database, allowing for faster query results.
2) The problem that had to be solved was how do we differentiate between the different types of shapes being selected by the user instead of the user drawing functionality. The user could select an ellipse, rectangle, or polygon as the shape for use in the new location being added to the database. This is different than the information needed to redraw a user drawn polygon. Since Javascript does not really care about what is being passed between the router so long as it is a JSON object we were able to leverage this by saving different versions of the data object with the needed attributes to redraw the shape on the map when pulled from the database. This is handed to the AJAX call to be passed to the insert query. The goal is to have a single data object that when submitted the router, the information will be inserted without any problems.
3) Similar to the second problem that we found, when a shape is selected the issue is how do we get the correct data sent to the database. On selection the necessary attributes such as height and width for a rectangle are saved into a json object which is then passed to the mysql controller to be sent to the database

11 Search Functionality (Matthew Zakrevsky)

11.1 Current Status

This has yet to be started. This is due to the fact that the database has to be tweaked and we are still working out the kinks for data submission. We will be filling in the database with simulated data when we do begin to test. This is a problem inhibited by the UI and the pathfinding. To have this fully functional we will need to be able to load all shapes in the database and also be able to change between the currently displayed map and the map where the space is located.

11.2 Problems that have impeded progress

1) The ability to easily change between maps or load maps with dynamically without refreshing the page

11.3 Solutions

1) This problem is being solved by refactoring how the map is loaded and populated with shapes within the database. Previously there were problems with the onload functionality happening at the same time so there was a problem with having both the ability to load shapes from the database and having these shapes be selectable.. By refactoring the ability to render shapes so that these are now callable functions that can be used to draw the map as needed instead of on load, allowing for the application to load the map with the location that is being searched for.