Ben Maldonado

CSE 383

Final Project

Professor Johnson

**Plan:**

* First row: Multiple tabs in index.html with a nav bar on its own row on top; each tab will replace the content in the row below
* Second row:
  + Default tab has information about assignment
    - Just store this information straight into it
  + Directions tab will take you to an implemented version of the map api from a prior lab
    - First row- HTML form with two boxes for from and to address and a submit button (in their own container)
    - Second row - Show outputs on row below that with map, elevation, and information in their own columns
    - *Ping this information after submit into the php rest server*
  + History tab will take you to a db query
    - First column
      * First row includes date and max number of lines
      * Second row is output table
    - Second column
      * Space for what they might want to click on

**What I did:**

The first thing I got working was the gitlab, which was obviously helpful because then I could build my project both locally and on the server without needing to do odd navigations to reach the necessary files. Then, locally I worked on the rest of the project:

I was able to get the nav bar working using a nav tabs div in the bootstrap framework, with each tab relocating to a new html page upon clicking. This also helps because the sections are then separated into their own HTML files (a helpful fact because each of the pages has a completely different functionality). For the rest index.html landing page, there is not much to add. I added a small description about my name, the class, and the project, but it was quite simple.

For the map.html page, I added quite a bit. First, the ajax call to the mapQuest API was working beautifully, so I grabbed all of the information there and printed it out to the page. I used a loop to put the directions in two columns going all the way to the end and then printed the final direction. Then, I called the locations API to get the LatLngCollection then use in the elevation API and get the elevation map.

Once all of that was done, I was prepped to use the database. I loaded up a new table on the sqllite server we were using so that it could hold the object we wish it to and the rest of the information to go into the history tab. I made it so that the map.js script sent all of the requests to the database as a POST containing the entire returned object, which also meant I had to edit the finak.class.php file so that it contained the proper fields for the setLookup function. Once my database started filling up, I then went and copy-pasted the code that revealed the directions into the history tab, used another ajax call to use the getLookup method, and then set the table. I also set it so that, for an onClick even for a table, it would run the function to display the map.

Finally, I used some CSS to highlight the table row once it was clicked and added some borders around each of the sections.

What didn’t:

* Everything worked!

How long it took:

* To get the initial webpage running on AWS with the Gitlab backed pages, basic bootstrap structure, and original landing page, I spent about an hour working on it.
* To get the map.html and map.js page working without database interaction took about 3 hours.
* To get the map.js to work with database interaction, and update the database to accept proper objects took about 2 hours.
* To put all of the information on the page and display the directions when clicked took about 2 hours.

What I liked:

* I think the way I built my menu was really cool, and I definitely enjoyed the database accessing because it has very useful applications.

What I didn’t like:

* I really did not like trying to figure out how to navigate the original AJAX call to the map API. It took WAY too long and I ran into WAY too many errors because I couldn’t figure out what order JS was trying to execute everything in.