TAP - NMED 4990 Independent Study

Corwin Smith

Description:

Tap is a web application using a javascript stack and an Arduino to represent fluvial flow data. This application uses flow sensors for the Arduino, and a Node.js library called Johnny-five to gather flow sensor data and store this information in a MongoDB database. Node.js, jQuery, and Ajax were used to create the front end application to display the flow data in the browser.

The application in mind for this project is to use this type of technology in a tap house setting. This web app simulates a tap house, which would have these flow sensors attached to the kegs of beer, and display in real time information about these individual kegs of beer. This can be both beneficial for a customer and owner of a tap house. Customers are able to see what is on tap at any given time. The information a customer can receive includes the name of the beer, brewery, alcohol percent, description, and amount left in the keg. This can benefit tap houses which frequently rotate their taps, since a user can check their website whenever to see whats on tap so that they don't miss out on their favourite brew. Furthermore the owner can receive some very valuable detail about the flow data. On top of what the customers can see, owners are able to get information about waste, glasses poured, keg size, etc. The owners also have an interface to hookup and detach sensors from kegs as they become empty and are tapped with a new keg.

Project:

During this independent study there was a number of changes from the intial idea for the project. The outcome of the project remained the same: build a working prototype for a web application that would use flow sensing data from an Arduino, but how the stack that was going to be used to build this web application witnessed a few changes along the way.

Initially when I started planning this project I had planned to build this web application using a MERN stack. This meant that I would be building my application to use Node.js, Express, React, and MongoDB. Although I stayed fairly close to using this stack throughout, keeping Node.js, MongoDB, and Express as the core components for my backend, I had to replace the React portion of this project with jQuery and AJAX for the front end due to time constraints. Ontop of the front end needing to change, I had to go through an interative process to get my Arduino working properly with the backend of my application. Initally I this was built using Arduino. Although I was able to get a sensor working in this environment, I was not sure how to go about getting my Arduino to use multiple sensors. Upon doing some research I came accross a library called Johnny-five (which was also recommend to my by Christine). This allowed me to create the software for my Arduino using javascript. This was more natural to me, and I was able to have the sensors as objects in the software and it was easy to sense multiple sensors from that point forward.

In the beginning I spent a lot of time working on designing a layout and icons for this application, and reading about the different tools that I would be using throughout the project. This stage lasted a few weeks as I was figuring out the process that would be best to build the project.

After that it was time to begin building the backend for the project. This portion of the project took much longer since there were a number of things that I had to learn. I had to design how the objects on the database would be stored, and learn about databases as a whole. I flip flopped on how the database would be structured (between whether I should Mongoose to enforce a schema or now), but in the end just went with storing the objects since this allowed me to make changes on the fly very easily. Moving forward though this should have a bit more structure as things develop, which is when I will begin looking into Mongoose more seriously. The other beast of this stage was Node. Node took up the bulk of my time on the project. After trying to go at it head first to build the server, I was quickly put in my place and had to resort to 300 pages of tutorials before coming back for a second try. The second attempt went much smoother, but I felt behind schedule due to the time needed to go back and learn node. It was time well spent though as I was able to apply what I had learned to solve a issue very last minute dealing with the school.

After learning node I got to the part of the project where I needed to start building a frontend for the application. During this time I was constantly making small changes to the backend to accomidate what I was doing with the front end. This part went smoothly until the time to create the routes came. I had a hard time understanding the interaction between the front end and the server, and the concept of going to a route that wasnt a web page. Once I got through this barrier though I was off to the races and pulling everything together for the end of the semester went rather smoothly. Until I had gotten through that barrier I was unsure of how successful I was going to be, but the elements seemed to align after that and things pulled together nicely in the end.

There is still lots of work to be done on this moving forward, but the independent study of learning how to make a web application like this from start to finish was certainly one of the most rewarding things, if not the most rewarding project I have gotten to do during my time in school.

Final Stack:
Node.js - Node.js was used as the server for this project. It accepted routes from the front end, and was used to create an API to access the database for the frontend, and the Arduino.
Express - Express is a Node.js library that helps organize a web application to use MVC. I used it to control all my routes, and requests between all the components of my application. Express ran on the node server, and provide managment for the requests and routes send to the node server.
jQuery - A javascript library used to create all the visual components for the project. In combination with AJAX, jQuery would send and receive requests to the server and then create components with the object data that it received.

MongoDB - This was the database used for the project. It is a non-relational database which means the objects are not stored in tables and dont necisarily need to be related to eachother. This was helpful for this kind of project when developing it because I was easily able to change the objects without having to change a whole database schema.
Cors - Finding this out later into the project, I needed to use this so that I could run both my node server, and front end on my localhost. Since the front end and server were running on different ports, this was used to do a cross origin reference.
Gulp - This was my task manager for the project. It was only used to manage the front end, since there was nothing for the backend that would have utilized gulp. How I set gulp up to work was to have browser-sync (everytime I made a change to something in the front end of the project, the browser would automatically refresh). The other main thing that gulp was used for was the compile my SASS into CSS. Every time I would make a change to a SASS file, gulp would see this change and recompile my SASS into CSS.
Johnny-Five - This library was used to create the software for the Arduino. Johnny-Five utilizes firmata (which comes with Arduino and is easily uploaded onto it), and allows you to make applications for your Arduino using javascript.
SASS - I used SASS to do the styling for this project. This was my second time using SASS on a project of any scale, and have found it to be a very quick way to write up my styling for the webpages. Gulp compiles the SASS down to CSS. Coming from languages like Python I found the indentation and nesting that makes SASS so readable to be a powerful tool when developing my pages.
HTML - Although there isnt much HTML used for the webpage since the pages are largely dynamically generated by the objects on the database, there is still some basic html in this project used for the header.

Difficulties:
This project was full of challenges and difficulties to overcome. Although I had a relatively concrete structure in my head about the steps to build this application, I truly underestimated the what I was getting myself into at times. The biggest struggles encountered during this project were conceptually visualizing how the pieces of this project are working, understanding how to use the stack I had chosen, and by far the biggest difficulty was the University itself. Ive mentioned the difficulties I had having to learn my stack, or conceptually visualizing how all the moving parts work together and where. What I havent mentioned yet was the huge roadblock that working at the University was. Initially when working on the project I was able to work at the school, but as I began developing the backend of the project and set up mLab that reality soon came to an end. Since I was using a database that was remote, I was no longer able to work on my project at the school due to being blocked by the firewall. This caused many issues for all aspects of my project. I couldnt work on my backend at the school because it was interacting with the database, I couldnt work with my arduino at the school because it needed to store data on the database, and I couldnt work on the front end which needed objects from the database. Although trying to talk with IT to solve this issue, it was a more tedious task than it needed to be and I resorted to figuring out how to host my server remotely and have my components interact that way. In the end this was the only solution available for being able to present it at the

University. I was most frustrated by this component of the project because it really prevented me from working at the school on my project during my free time. I felt restricted, and this felt very unacademic as I was being blocked from persuing a project for school. It also made it diffucult to show Christine what I had worked on during the week since it was blocked. Luckily, Christine trusted that I was getting the work done and I was able to find a work around to this problem (talked about below in the tools used).

Cool Tools:

Heroku - heroku.com. With all the difficulty trying to get this application to work at the school, I needed to find a way to host my server remotely. I had no way of accessing my database while on campus if I was running a localhost on my machine, so I came accross Heroku as a tool to host my server remotely. It wasnt without its issues to get going, but it was the only way to get my application to run while I was on campus, and fairly easy to get into using. Oddly enough I could push to my heroku server though while on campus, so I was able to hack a way of working on campus by pushing my changes to the server. Although a bit hacky, and not ideal, this allowed me to make some changes to my project while on campus instead of needing to leave.

Surge - surge.sh. This was by far one of the coolest tools that got shown to me while working on this project. Its a very simple CLI that allows you to publish your website very easily. It took two command line functions to get my website live. One of them being to download the tool, and the other one to publish.

mLab - mLab.com. This was where I had my database stored. Instead of having a local copy, I decided to have my database stored remotely so that users could always access the website and see the data live. Also, I wanted to be able to hand this project in without Christine needing to setup a database locally on her machine, so the best way to go about that was to find a way to host it remotely. After doing some research about where to host a MongoDB database remotely I came accross mLab. For small test databases it is free, and you only start paying once your database has some substance to it.

Outcomes:

The biggest outcome for me was a personal one. This was the first application I built on my own that I was proud of. Not that it isnt without its flaws, but it was a product that I had a vision for and was able to build start to finish. Personally this gave me a lot of confidence moving forward as a developer. Outside of personal gain though there were a large amount of learning outcomes to come from this project. I learned about non-relational databases, and how to use these to store information for my website. This took some time as I was trying to conceptually wrap my head around how this works. Second was becoming familiar with Node. When it came to working on the project I learned I needed to spend more time than I initially though learning Node. Leaving the project though I have a better footing on how to working with it, and where to go next while I continue to learn Node. File structure for applications, and how they communicate was big for this project. I had many iterations of just structuring files, and it took talking to a developer friends who has experience with this kind of stuff to figure out a proper structure. Managing routes and connections took a while to wrap my head around, but seem

very natural now. Again I had problems conceptually wrapping my head around how the communications of these worked, but once I figured it out it became second nature. Luckily, because of all these outcome, when I had an issue the day of presenting this project I was easily able to troubleshoot and come up with a solution. I think that was one of the most rewarding things, and really showed where I had gotten to from this independent study.

Future:
Moving forward from this semester I would like to look into converting the front end over to React. That was a goal of the project that had to be comprimised in order to allow me to understand Node, so it would be good to revisit this portion of the project to achieve a part of the study that didn't get finished. I would also like to look into the pouring animations, which I ran out of time to implement due to a mad scramble at the end to fix a firewall issue with my Arduino. Lastly, I want to work at getting this to a production ready state so that I can have this tested live my July. I talked to the owner of Round Table and he was very interested in the possibilities this application has, and I would like to get it to a point where this can be installed in a business as soon as possible.