CSC 317 Final Documentation
Fall 2019

Bradley Justice
https://github.com/pambalos/csc317-final

Write-Up

Project Overview

This project involved creating a fully functional backend and front end webapp, for the purpose of data collection, display and review. In this case, it is the case of receiving car vehicle data segregated by session.

Technical Overview

I chose to go simply with the MAMP, (MacOS, Apache, MySQL, PHP) stack as it is something that I have, at this point, become intimately familiar with, and as such, knew that I could build this product with.

To start, I had to come up with a database design that would allow me to track all of the sessions and vehicles in a way that made sense and worked. I ended up coming up with 6 different objects; vehicles, sessions, wheelrecords, linerecords, echorecords, and otherrecords. This design allowed me to keep track of all of the data that I needed to, at least according to the design specification, as I understood it. Below are examples of my database:

Vehicles:

	name	width	time	created
1	bob	12.3	0	2020-05-20 17:54:18
2	jim	14.42	0	2020-05-20 17:54:18
3	pam	13.97	0	2020-05-20 17:10:17
4	rob	11.78	0	2020-05-20 17:54:30

Sessions:

	sessionId	vName	time	created	ended	active
1	73b0e874-9af7-11ea-9395-7a2444e7bcf9	pam	0	2020-05-20 17:10:17	2020-05-20 17:10:35	0
2	7e9c3f54-9af7-11ea-9395-7a2444e7bcf9	pam	0	2020-05-20 17:10:35	<null>	1
3	999ab122-9afd-11ea-9395-7a2444e7bcf9	bob	0	2020-05-20 17:54:18	<null>	1
4	99afd282-9afd-11ea-9395-7a2444e7bcf9	jim	0	2020-05-20 17:54:18	<null>	1
5	a0b918d6-9afd-11ea-9395-7a2444e7bcf9	rob	0	2020-05-20 17:54:30	<null>	1

Wheel Records:

		sessionId	leftw	rightw	time	created
9	7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	3	2	14500	2020-05-20 17:57:57
10	7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	3	2	14600	2020-05-20 17:57:57
11	7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	0	2	14700	2020-05-20 17:57:58
12	7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	3	2	14800	2020-05-20 17:57:59
13	7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	0	2	14900	2020-05-20 17:57:59
14	7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	3	2	15000	2020-05-20 17:58:00
15	7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	3	2	15100	2020-05-20 17:58:01
16	7a2444e7bcf9	7e9c3f54-9af7-11ea-9395-7a2444e7bcf9	0	0	0	2020-05-20 17:23:44
17	7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	3	2	32800	2020-05-20 18:00:52
18	7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	3	2	32800	2020-05-20 18:00:58
19	7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	3	2	32800	2020-05-20 18:01:11
20	7a2444e7bcf9	7e9c3f54-9af7-11ea-9395-7a2444e7bcf9	0	0	0	2020-05-20 17:56:11
21	7a2444e7bcf9	99afd282-9afd-11ea-9395-7a2444e7bcf9	0	0	0	2020-05-20 17:56:12
22	7a2444e7bcf9	a0b918d6-9afd-11ea-9395-7a2444e7bcf9	0	0	0	2020-05-20 17:56:12
23	7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	0	0	0	2020-05-20 17:56:12
24	7a2444e7bcf9	7e9c3f54-9af7-11ea-9395-7a2444e7bcf9	3	3	100	2020-05-20 17:57:00
25	7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	3	3	200	2020-05-20 17:57:01
26	7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	3	3	300	2020-05-20 17:57:01

Line records:

	lid	sessionId	lineData	time	crea
1	193bddd4-9afe-11ea-9395-7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	1	13720	2020-05
2	193bf864-9afe-11ea-9395-7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	1	13720	2020-05
3	193c1006-9afe-11ea-9395-7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	1	13720	2020-05
4	199f0a26-9afe-11ea-9395-7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	1	13820	2020-05
5	199f2a9c-9afe-11ea-9395-7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	1	13820	2020-05
6	199f4068-9afe-11ea-9395-7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	1	13820	2020-05
7	1a04fab6-9afe-11ea-9395-7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	1	13920	2020-05
8	1a05185c-9afe-11ea-9395-7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	1	13920	2020-05
9	1a052e82-9afe-11ea-9395-7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	1	13920	2020-05
10	1a6bd088-9afe-11ea-9395-7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	1	14020	2020-05
11	1a6c15a2-9afe-11ea-9395-7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	1	14020	2020-05
12	1a6c2ede-9afe-11ea-9395-7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	1	14020	2020-05
13	1ae34fc8-9afe-11ea-9395-7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	1	14120	2020-05
14	1ae39064-9afe-11ea-9395-7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	1	14120	2020-05
15	1ae3a734-9afe-11ea-9395-7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	1	14120	2020-05
16	1b3a44d6-9afe-11ea-9395-7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	1	14220	2020-05
17	1b3a5bf6-9afe-11ea-9395-7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	1	14220	2020-05
18	1b3a6d76-9afe-11ea-9395-7a2444e7bcf9	999ab122-9afd-11ea-9395-7a2444e7bcf9	1	14220	2020-05

Data Collection:

There were a number of database design decisions that I had to make, including what linked to what other objects, in order to get the relationship between all the data that I wanted. I came up with session records, which were linked to by many different session data points. In this way, I was able to very easily get the data that I wanted.

I built a number of different webpages/endpoints to handle the requests. These acted essentially as Get endpoints. These endpoints included:

/register?name=XXXX&width=###.###
Added a new vehicle run to the system, returns a cookie called USER=[name] that would be included for the other commands. Width is the width of the vehicle in cm. It overwrites any other “active” session for that named vehicle

/wheels?left=###.###&right=###.###
Records the speed of the left and right wheel in cm/sec for that vehicle in the current session

/echo?dist=###.###
Records the result of the echo sensor in cm for the vehicle in the current session

/line?l1=##&l2=##&l3=##
Records the result of ONE or MORE l1, l2, l3, line sensors

Here is a screenshot showcasing these endpoints:

```
bjustice-macOS:csc317-final bjustice$ curl localhost:8086/app/register.php?name=pam\&width=13.970000\&time=0

    {"USER":"pam"}bjustice-macOS:csc317-final bjustice$ curl localhost:8086/app/wheels.php?left=0.000000\&right=0.000000\&time=0 --cookie "USER=pam"

    {"SUCCESS":true}bjustice-macOS:csc317-final bjustice$ curl localhost:8086/app/echo.php?dist=9.220000\&time=10 --cookie "USER=pam"

    {"SUCCESS":true}bjustice-macOS:csc317-final bjustice$ curl localhost:8086/app/line.php?l1=1\&l2=1\&l3=1\&time=20 --cookie "USER=pam"

    {"SUCCESS":true}bjustice-macOS:csc317-final bjustice$ curl localhost:8086/app/other.php?ir=0\&time=30 --cookie "USER=pam"

    {"SUCCESS":true}bjustice-macOS:csc317-final bjustice$ curl localhost:8086/app/end.php?time=37600 --cookie "USER=pam"
```

Screenshot of running DataScript:

```
{"SUCCESS":true}
{"SUCCESS":true}
{"SUCCESS":true}
{"SUCCESS":true}
{"SUCCESS":true}
{"SUCCESS":true}
{"SUCCESS":true}
{"SUCCESS":true}
{"SUCCESS":true}
```

Backend (PHP):

I used PHP for this entire project as it was something that I have become quite familiar and comfortable using alongside html and javascript.

One thing I was experimenting with but dont think I will complete in time for the submission deadline was a routing engine. As can be seen from the source code, there is a '.htcaccess' file that sits in the root, that, when turned on, will redirect all traffic to the index.php file, which I was working on implementing as a router, but did not finish.

Instead, all of the data collection php endpoints just sit in the root folder.

Frontend (HTML, CSS & JS):

I ended up just using good old html, css and javascript to make my front end pages responsive, dynamic, and filterable. In each case, I simply had a php script that echo the json_encode of the desired data. In this way, I made my front and backend communicate.

In addition to this method of data communication, I also decided to do all of my filtering in the ***Javascript***, and not the php, that way I wouldn't have to reload the data. (granted, I could technically write endpoints that return filtered data, and have a javascript update, but I decided it made more sense and would be easier to populate the data ahead of time, then apply whatever filters I wanted via front end style manipulation (display = none).

Development Environment

I used the MAMP stack from Bitnami at https://bitnami.com/stack/mamp
This allowed me to install and run a full Apache, MySQL and PHP stack, which I was familiar with using from the storefront group project.

Setup Instructions

You can simply clone the Github folder from https://github.com/pambalos/csc317-final into the php server, then follow these steps;

MySQL:

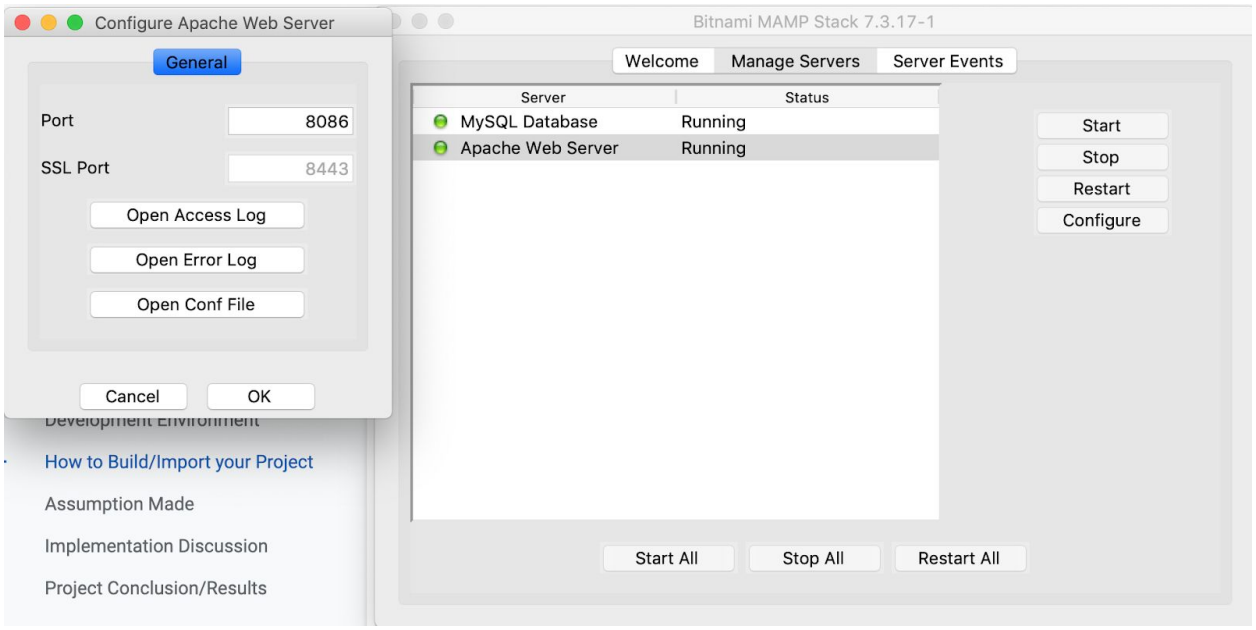
In order to set up the MySQL database with tables and the necessary user permissions, copy and paste all the MySQL statements in the './app/sql-setup.txt' file into your MySQL console and run the commands.

PHP:

In order to ensure that the php code can communicate with your MySQL server, please navigate to the ‘./app/lib/common.php’ file and adjust the database port variable as needed.

As mentioned before, this project is built to run PHP on an Apache Web server. So long as you have php 5.4 or later, have a newer MySQL installation, and have ensured the configuration variables all match up, you should be able to run this application.

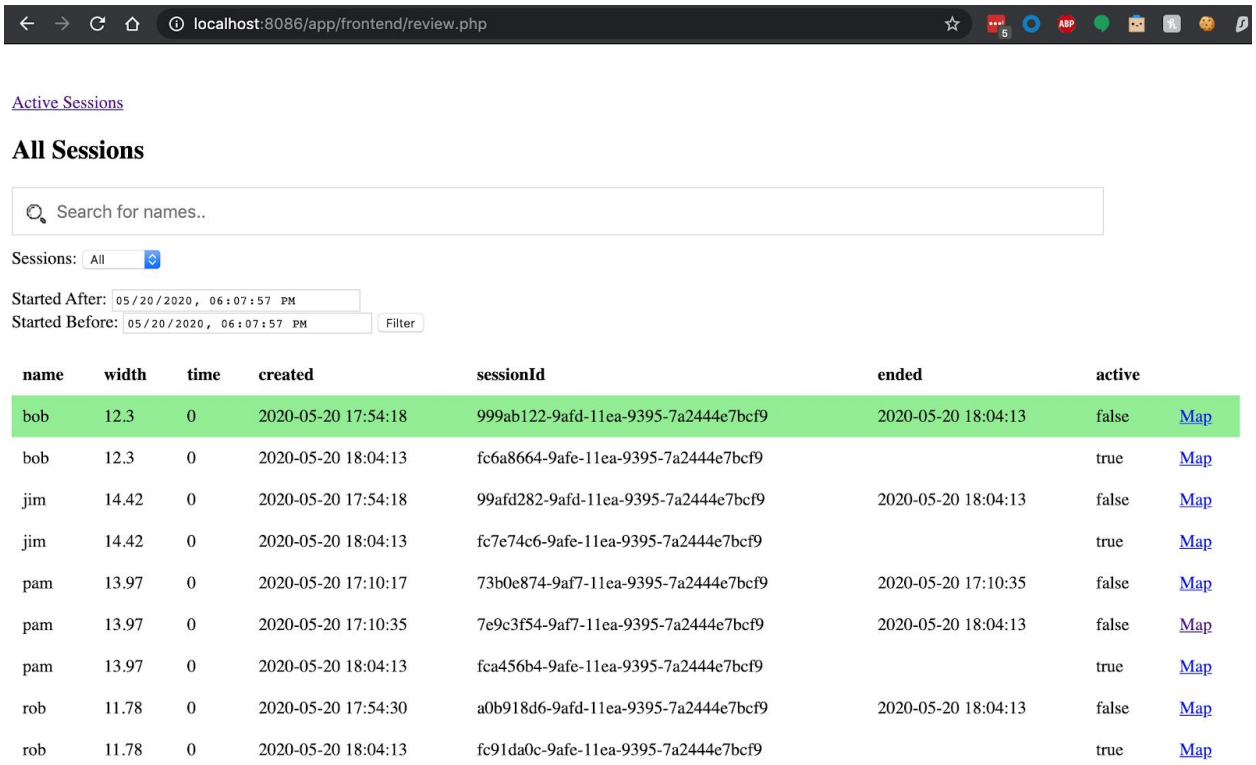
Here is a screenshot of my stack manager:



As can be seen, I simply had a MySQL database, and an Apache Web Server.

Demo

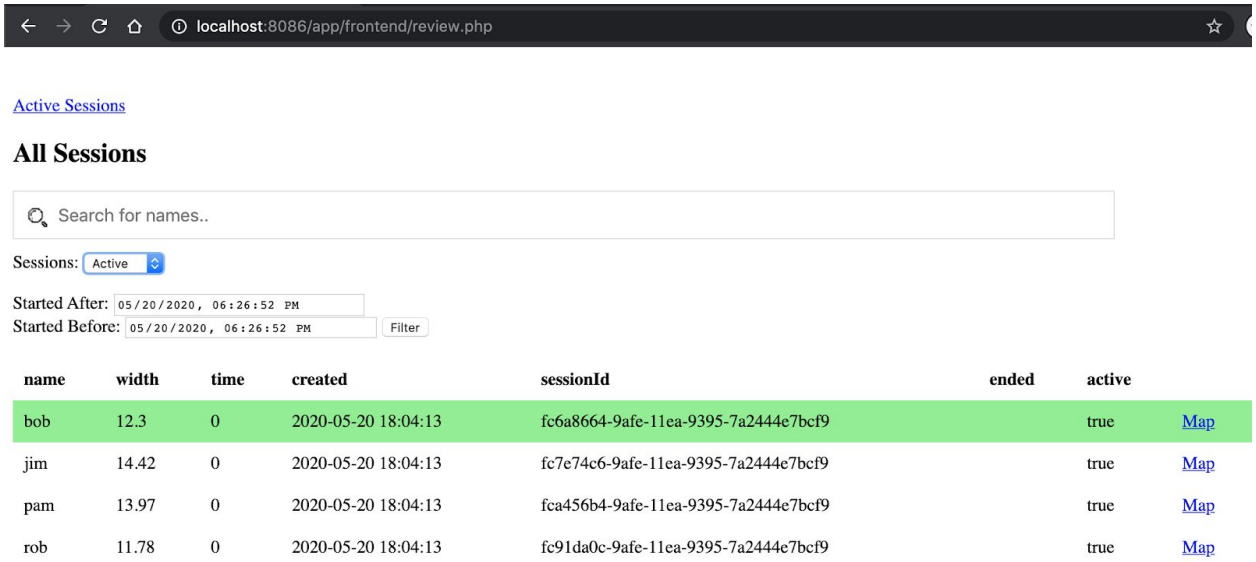
/review



name	width	time	created	sessionId	ended	active	
bob	12.3	0	2020-05-20 17:54:18	999ab122-9afd-11ea-9395-7a2444e7bcf9	2020-05-20 18:04:13	false	Map
bob	12.3	0	2020-05-20 18:04:13	fc6a8664-9afe-11ea-9395-7a2444e7bcf9		true	Map
jim	14.42	0	2020-05-20 17:54:18	99afd282-9afd-11ea-9395-7a2444e7bcf9	2020-05-20 18:04:13	false	Map
jim	14.42	0	2020-05-20 18:04:13	fc7e74c6-9afe-11ea-9395-7a2444e7bcf9		true	Map
pam	13.97	0	2020-05-20 17:10:17	73b0e874-9af7-11ea-9395-7a2444e7bcf9	2020-05-20 17:10:35	false	Map
pam	13.97	0	2020-05-20 17:10:35	7e9c3f54-9af7-11ea-9395-7a2444e7bcf9	2020-05-20 18:04:13	false	Map
pam	13.97	0	2020-05-20 18:04:13	fca456b4-9afe-11ea-9395-7a2444e7bcf9		true	Map
rob	11.78	0	2020-05-20 17:54:30	a0b918d6-9afd-11ea-9395-7a2444e7bcf9	2020-05-20 18:04:13	false	Map
rob	11.78	0	2020-05-20 18:04:13	fc91da0c-9afe-11ea-9395-7a2444e7bcf9		true	Map

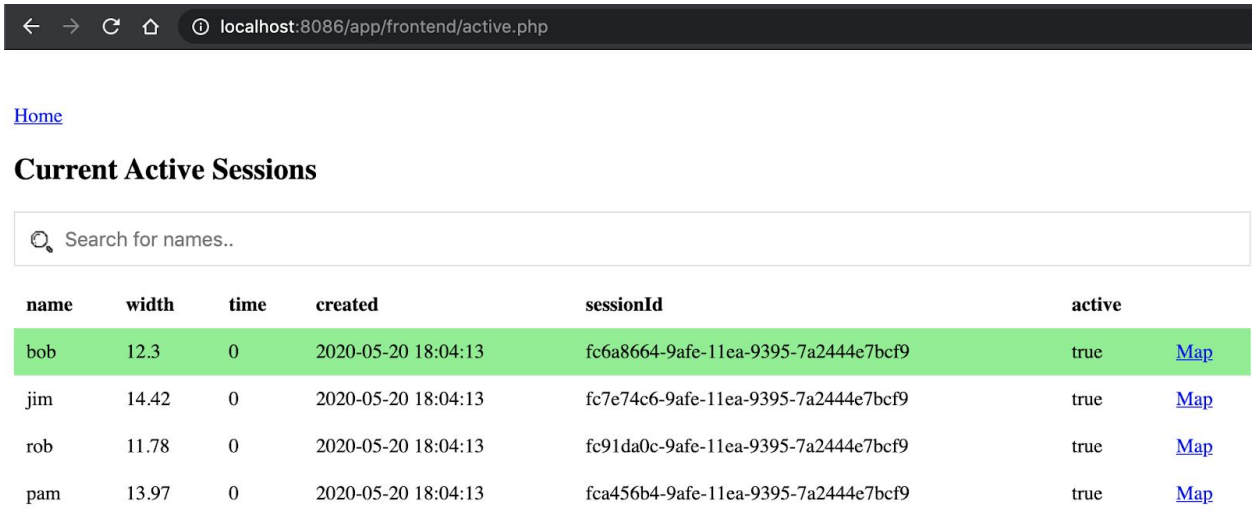
This page lists all of the sessions that are saved in the database. You can click on any one of these rows to navigate to /datareview, or /map. As can be seen, there are filter options of vehicle name, session active status, and the date filter. All of these have been tested and work. I was not able to add borders, no matter what I tried, but I increased the padding to distinguish rows.

Ie. (Active filtered)



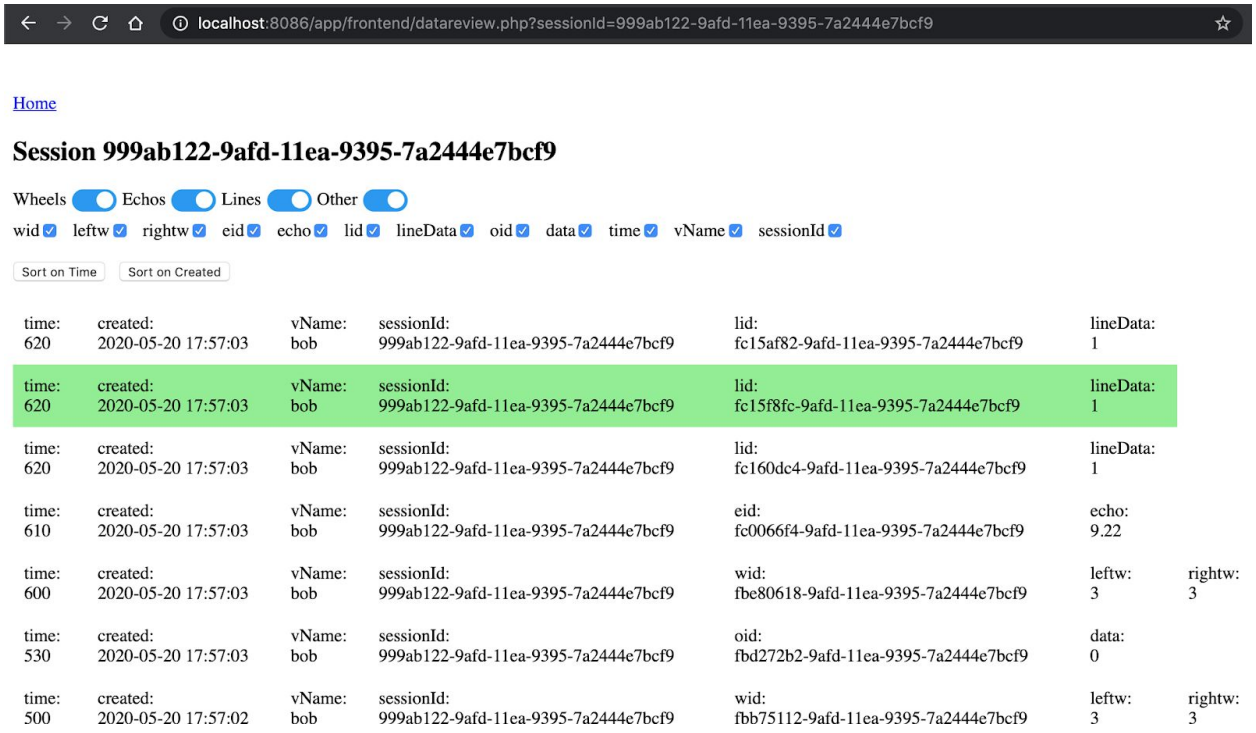
Not all of these filters can be active at the same time. I couldn't figure out the bigs in time.

/Active



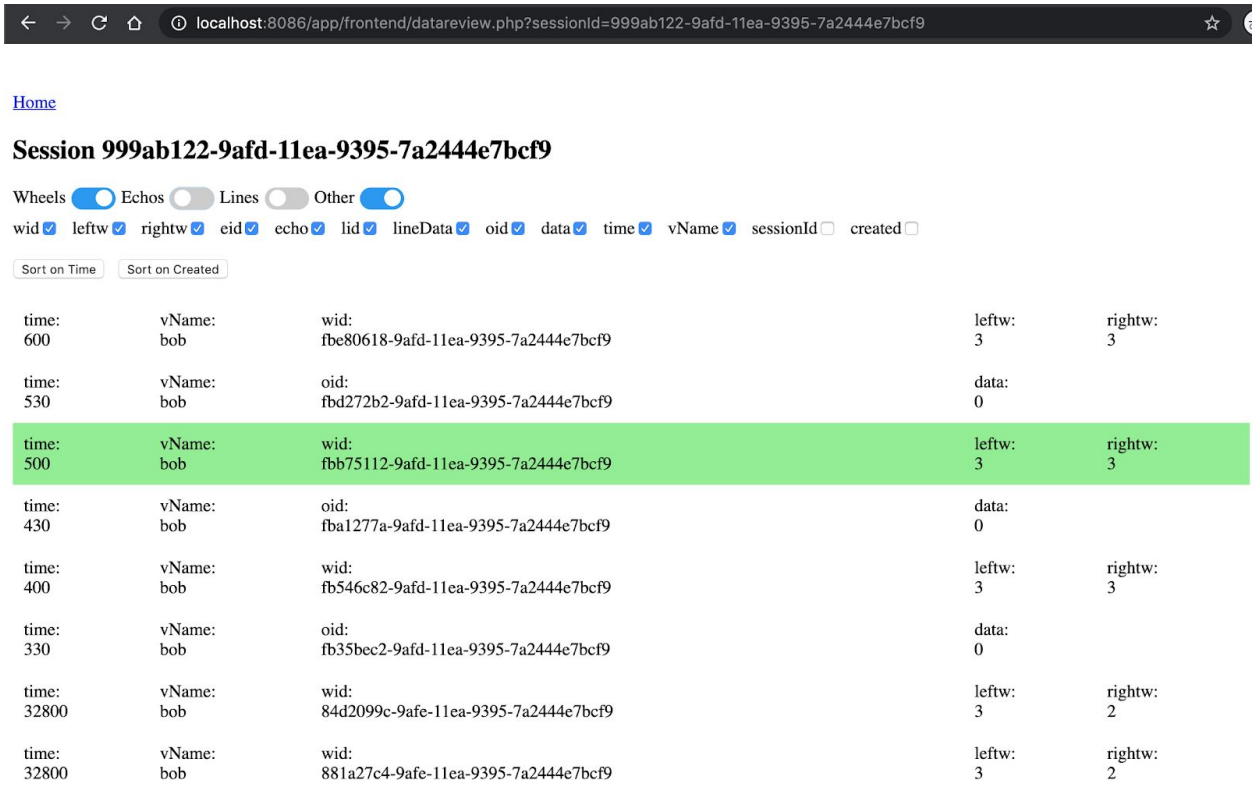
This page simply shows all of the currently active sessions.

/datareview?sessionId=####



This page takes one parameter, the sessionId, and retrieves every single record (via php) that is associated with that sessionId (wheels, echos, lines, others, dist) and compiles them into a single array that gets iterated and printed as a table via javascript. The filtering is also all done via javascript and took quite a bit of fiddling around with until I got it right. As can be seen, filtering works both on column, record type, and time:

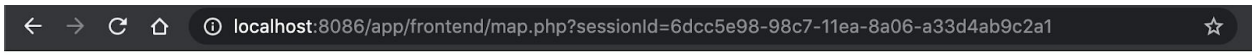
****NOTE:** Opening a large session can crash the application as it makes use of join queries.



In this screenshot, the you can see some of the filters in action.

/map?sessionId=###

Also takes on parameter, but doesnt actually do anything - I couldn't figure out the distance mapping equations given.



Project Conclusion/Results

This project was actually a very fun exercise in terms of back end development. I very much enjoyed building the ‘API’ to receive and upload data, as well as figuring out clean and efficient ways to communicate between my front end and back end. Throughout this project I feel like I really got a lot of useful practice in that regard.