12-13-2019

1. ­­found out how to add event of click on leaflet tiles and look at the values of the tile object being clicked (17-2-4).
2. Decided to pass everything to data base and have javascript and python read/write the database, instead of passing values b/w javascript and python.
3. Decided to use a small selection of locations in the bay area like 5 for testing and 10 for actual application, because there’s no way around calling uber api to find distance, and they only allow 1000 calls per key. And the data has 2700 locations with 2699^2 calls to make to cover all distances.

12-26-2019

1. Figured out how to use google api, needed to use the distance matrix, 'https://maps.googleapis.com/maps/api/distancematrix/json?'.
2. Decided that the uber data is not enough to do an hourly calculation for all routes, will do yearly comparison of routes instead.
3. For some reason need both css files in the html to make the map show. Also can’t use the double curly bracket to reference js to html in a different directory.
4. Found out the geojson file covers a much bigger area than the bay area, need to figure out a way to only display the bay rea.

12/27/2019

1. After some consideration, decided to manually click through the map and save the addresses on the uber movement website for San Francisco alone. Then reference that to the geojson and find the movement ids. Then use only these info as needed. This way file size will be smaller and easier to deploy. Also can practice on organizing and cleaning the data.

12/28/2019

1. Manually clicked the Uber movement website and saved a list of addresses to reference to the Geojson.

1/3/2020

1. Write an algorithm that’s able to reference to the Geojson and extract trips taken within San Francisco from the Uber dataset. It took an hour to run on my intel i7 16GB ram laptop, because it loops through 11million x 11million times to siphon the needed data.
2. Plot the Geojson with only the SF area and see if I miss anything, then added the missing blocks to the file.

1/5/2020

1. Run through 4 years of data and saved the needed trips within SF boundaries to 4 csv files. Saved these 4 csv to SQL. Then saved the Geojson within SF to mongodb.

1/8/2020

1. Fine tuned website look.
2. Added pop up and color change functionality when you click on the map layers.
3. Added click function to fill in starting and destination input field.