11.05.15 – Capstone Project Report Four - TBTB

Lots was accomplished this week as we had three weeks instead of two in order to accomplish things to write about and present. The speed issue was addressed and we talked with Shamal about it to find that there was some issue with the server that we were hosting it on, and that we are able to host it locally with no speed issue so that is the route we have decided to take. Now the speed is just as it should be and as the app is used more in a session the faster the results will be returned as SQL server does some of its own caching.

Also on the database side we have worked more on the queries we have to expand them to accept more options of what the user might want to look for such as querying for the pickup, dropoff, or even both for a certain region. A couple other queries have been worked on and are nearly done or done that Tom will talk about in his section.

The entire UI of the site has been redone from the temporary interface shown three weeks ago and is now much smaller and has more options. You may display the different boroughs and clear the entire map of all points and shapes you may have drawn. We do plan on expanding to actually query the boroughs soon but they could not fit into this iteration. Also we have gone for a very modular design approach so that everything except for the map itself and the tool bar on the top will be contained in windows that you may move about the screen as you wish. This will be much more important when we implement the visualization charts for the data.

Also on the front of the map you can chose to display the pickup, dropoff, or both points when you query for data and see them on the map. In addition to all of these things we are working on being able to implement a circle query and a line query to display on the map and will begin expanding much more in the next two weeks. Last iteration we got the functionality of the map completed and now we have the basis of the interface and can move on from here to add more bells and whistles and useful visualizations. At some point we also plan on putting more of the data into the database to prove that we can do it, however that would be something for the final presentation as it would take a lot of space to do that.

Tyler:

I concentrated originally on trying to speed up the execution of the program during this iteration by implementing various threading techniques in C# using the built in Thread and

ThreadPool classes. I tried to divide up the query into smaller units to overcome this and that did not help and threading the serialization did not help at all either. After a while Tom and I decided to test the database locally and see what the problem was and it was much faster when done locally, much faster than either of us expected. So after talking with Shamal we determined that from now on we will host the database locally on all of our machines so that the program will be as fast as it should be. I assisted in getting Brendan and Bill up to speed on this and helping them get the databases on their computers and linking them into the MVC project with Tom.

After this was out of the way my next task was to implement little popup windows for each one of the points that would display general information about the trip such as the fare, distance, and trip time. With the Kendo UI library that we are using for the styles and javascript widgets they actually have a popup window widget that is very powerful that we used for this. I originally thought I would have to store all of the data in a separate object to then look in when a point was clicked, but if you use the d3 onclick event it will have all of the data there for you. The popup also locates itself on the point you have clicked so the user does not have to go searching for anything and it will close when the user closes it or when another popup is opened. In this same vein I implemented the other popup in the bottom left with general query information which will display how many trips were returned in the most recent query and some general average statistics about them. This is the first tiny step towards actually making visualizations and charts out of the data. This can be closed as the user wishes but will reappear with each query for the moment.

I also was responsible for completely ripping out the old UI of the site that was temporary and putting in the new UI and coming up with the idea to use a modular design to go with so that the user could do what they wanted with the charts and other windows we may open with information. I scrapped the old and large grey menu for a very slim toolbar up at the top that contains all of the information and options we have at the moment and some that we have not implemented yet. Those are the Split View and the Visualizations which we will begin on this iteration. I continued using the Kendo UI here to create the UI so that the entire app looks like it belongs together and follows the same style, and the widgets are very powerful once you get the hang of using them. While I was doing this I also implemented the logic for the radio buttons to select what to query for and what to display. The logic for displaying both of the points was a bit more that I expected it to be even though it was only a few extra lines of code, but I found that you actually needed to create two separate ‘g’ elements to house the pickup and the dropoff points with d3. Once I did that however it was smooth sailing.

When I was tearing out the UI I also went in and completely organized all of the files and separated code that did not need to belong together. Our index file was massive with javascript and stuff that didn’t need to be there. I created a utilities object in another file that had all of the functions and variables we would need to use for the page and then another function in another file that would house the initialization for the interface. This way the code is better structured and when you look at an html file you only see the necessary html, and no code is repeated more than once as everything is in functions that can and should be. We started out just doing what worked to get a working product complete but now it was time to actually make the code robust and well written. This, as I have found in class and in the field, is essential to a good application. Lastly I sat down with everyone on Tuesday and integrated all of the code together to make all of the different things we did work. Originally we were just going to do a normal git merge but because I had changed so much about the structure of everything I found it to be more time effective and effective for everyone else in the group to understand what I did, if we sat down together as I went through the integration process and explained what I did.

Brendan Gray:

This iteration had multiple improvements and design implementations to the front end. Most of what I have been working on has eluded me personally; however, I feel that I am on the cusp of solving my issue. The issue I currently have is that the circle is not being created or displayed correctly, which I believe is due to different data being passed to the circle to query the area inside of it, it seems d3 doesn’t know how to handle this different data that is being passed. I’ve been trying to figure out what makes d3 not able to use this, since the only problem we are having is that when the query function is called that is when the circles are not being displayed correctly. Interestingly enough, I was able to figure this out when I removed the query function inside of the function used to create the circle. I am still trying to decipher why the query is an issue. During this iteration, I also added the Clear All Layers button. The Clear button is able to get the current layer object and remove all additional layers that were added past a certain point. I implemented this correctly by accident; when I first tried to figure this out, I was under the impression I had to keep track of all of the separate layers. However, when I tested out the first layer removal that was stored in an array, it removed all of the layers that had been additionally added to the map. I was confused by this at first but it seems the function I am using with the layer object takes care of everything in regards to it's removal. The last iteration I am working on are 3 radio buttons that allow use of querying either only drop off locations, pick up locations, or both, on the map. This functionality has yet to be seen, as I am not working on the query function used with these buttons.

<u>Bill:</u>

Over these last few weeks my main goal was to get a simple feature implemented for our map. This feature allows users to click a button and it draws a pretty detailed polygon for the layout of the five different boroughs. How this feature works is it takes the Longitude and Latitude of coordinates that I found by using the getLatlng function outputting them to an alert. Then you take those points and store them into a javascript array of objects and then you call this array into a draw polygon functions. This function is then linked to a button that when clicked calls the draw borough function in the javascript and then it draws a predefined polygon from the Latitudes and Longitudes. After creating a simple feature for just one borough Tyler implemented the rest of the boroughs in other colors with 5 different buttons. What this feature can allows us to do is highlight two borough and then see what taxis have gone through the two highlighted boroughs.

<u>Tom Taylor:</u>

During this iteration I primarily worked on two different things. The first one was getting back into data validation. I took a step back and did some research on what the columns actually provide us so that I can make an educated determination on what columns to provide in the final table for analysis. I then used what I learned to start on more data validation as we really only want to be able to query on data that represents an actual valid taxi trip. The second one was expanding on the queries/ stored procedures to be able to provide what is required regarding querying the data. Most of my work here was related to what I call the intersection query or from your data query model the beginnings of a topological query (enters, leaves, crosses). Regarding data validation I determined that there is still invalid data in regards to latitude and longitude. I ran across this while working on the intersection query. There are still start and stop points that represent a trip that are actually the same point, as in the trip did not go anywhere. Obviously we can consider these trips as not valid as well. I will remove these rows from the final table. I also now understand what medallion and hack_license really mean and queried against those columns looking for anomalies. In both cases I did not find that I could invalidate any data based on these two columns. I did determine that these columns really only have 10's of thousands of unique identifiers in them so I can easily convert these very long alphanumeric strings into simple integers making the table smaller. I also started looking at pickup_datetime – dropoff_datetime as it relates to trip_time_in_secs and trip_distance. There are definitely anomalies here as well. I am working on my boundary sets to determine what data to consider invalid and thus remove. For instance, there are almost 13,000 trips that lasted two seconds or less! On the other end of the spectrum there were almost 8000 rows with an average straight line speed of over 60 miles per hour. Hardly possible in NYC one would think! As far as the queries go, I now have four unique ones: region query on polygon, region query on circle, nearest point query (using circle) and now intersection query. Each of the first three actually contain three iterations of the query in a single stored procedure. I can return result sets, from any valid date time interval, for pickups inside the region, drop offs inside the region or both. Tyler can then also display the results as it relate to pickups, drop offs or both. The same applies to the nearest point query except that I am only returning the top 10 (descending). It is up to the front end to provide the logic of if no results returned then double the radius, etc. The most interesting part for me during this iteration was working on the intersection query. I was able to prove that I can take the trip pickup point and drop off point and dynamically create a line geography between them while processing rows and then use that geography in real time to see if that trip intersects with any geography that the query is provided. There is a lot of crunching going on here in the background so it is not as fast as some of the simpler queries but I was still surprised by how fast it executes. I would think that in a production environment database views might be created to have some of this pre-calculated. However, with our limited space this is not really an option. The other thing that I found interesting was that I could not use my already stored geographies for the pickup and drop off points to calculate the line segment geography, I actually had to create it using the original latitudes and longitudes in the same way the point geographies were created.