10.15.15 – Capstone Project Report Three - TBTB

These past two weeks we have finally gotten the project to an integrated and working state. There are some issues at the moment, however we can now strongly move forward to fix them and to add features into the application. Before in order to test speeds and accuracy we had to use our best judgement and ideas because there was no interface to feed data to the SQL database from the .NET application itself. Now a basic interface with Leaflet Draw has been built with a datetime picker from the Kendo UI Core javascript library. Kendo UI has a lot of functionality for creating charts and graphs and we plan on using that moving forward to easily generate more visual statistics on our data. That will not however, replace d3 as d3 is much more flexible to creating essentially anything we want even though it is much more complex.

D3 is responsible for placing the markers on the map when a region is queried and will display any number of points that it is given. One of the issues we are facing here, is that at the moment the application is incredibly slow. The SQL stored procedures will run in a second or less most of the time, and when the query range extends to a few days that goes up to a couple second at this point, and the rest of the code seems well optimised so our issue here lies in the Entity Framework connection we believe. One of the next steps we will take is to reevaluate that code to see if it can be improved, or maybe there is another way that will not have all of the fluff of EF, but will run much faster.

We have a very good footing to move forward from here on and are looking forward to what we may be able to accomplish in the weeks to come.

Tyler:

This was a very busy two weeks for me as in the beginning I came up with the idea to use Threads in the C# server code to make the calls to the SQL database run faster and be able to serialize the returned objects into JSON very quickly. I did lots of experimentation and tests with default C# threads, the C# ThreadPool library and threading the entire stored procedure call to just threading the serialization part. It turned out that none of this offered a viable improvement at all, and in most cases actually increased the execution time by a smaller percentage (less than 10%). This confused me greatly so I looked into how SQL server and Entity Framework handled threaded calls and I came to find that SQL server will handle the

threading on its own and that the Entity Framework connection itself can only handle a single connection at a time instead of multiple concurrent ones. Which explained exactly why it was running the same or taking long to accomplish the same tasks.

I then looked into serializing the objects returned in threads to increase on that time, and I actually found, where I thought that the serialization was taking forever, was actually taking next to no time at all to accomplish. So I became at a loss, and now I will look further into different ways to optimize EF or other simpler ways we can call stored procedures from our database.

The other thing I accomplished this sprint was I put together the very simple web interface for the project and then linked all of the parts together. I had been linking the C# server side code with the database for awhile, but this gave me a chance to actually link the client side interface with the server code to return points. I chose to use a Kendo UI datetime picker because I have had experience with Kendo UI at my current job and it is a fantastic library to use for UI elements. I then reserached some d3 code that Brendan had found earlier in creating points and got that implemented as well.

One issue I ran into with the points was interesting, as Tom stated before in our presentation and as I am sure you know, SQL server requires the ‘left hand rule’ for all region query’s with a polygon. The interesting thing here is that the toGeoJSON() function of Leaflet Draw does not care what order the points are in, and will put them either forwards or backwards from what we need depending on how the shape is drawn. For this I found a simple javascript library called GeoJSONRewind that will take a geoJSON object and rewind it to be clockwise or counter-clockwise based on your needs, and now no matter how you draw the polygon it will call the stored procedure correctly.

Brendan Gray:

Bill:

This week i was helping tyler link the interface with the SQL backend and we finally got the project to work with it. It does take 30 seconds to show up but it does work. The UI we created

was with Kendo UI core. Which allowed us to use the Calendar for picking a specific days and being able to output that points in a square using leaflet draw. Now that we have a basic output with points using D3 which brendon has been working on. Now we can start writing queries to show more complex data sets and add different features to the project. Also I am going to start working on getting charts and graphs for the data to output so we just don't have a map showing the data output. As tyler mentioned above the one of the issues we were having was with the polygons to output the points was that the coordinates go counter clockwise so we had to give the MQL data in reverse order.

Tom Taylor:

The two things that I focused on during this cycle was working on queries/stored procedures that will be needed to retrieve required data from the database as well as more data verification mostly related to pickup_datetime, dropoff_datetime, and trip_time_in_secs.

From the slide presentation given in class we need to provide the following queries in some manner: point queries, region queries, topological queries, and navigational queries. I have created successful polygon region queries so as long as I am properly provided the point coordinates used to generate triangle, square, or polygon from leaflet draw I can properly return all of the contained geography points. I can now also do the same thing as it relates to a nearest point. As long as I am provided the point of interest and the radius of the circle of interest I can provide all of the contained points as well. I am working on ways to better the performance of this query. Things like return the region that contains the circle and then sub-query that region, or query directly using latitude and longitude and then again subquery that region geo-spatially for the circle. For the topological queries I have decided that we can return some useful information without trip trajectory points. I am working on the queries to turn into stored procedures for this. All I need to be provided from leaflet draw is the endpoints of a line segment. I can generate line segments from our data by using the start and stop points on our trips. This will then allow me to do "crosses" if the two line segments intersect. I can also take the line segment provided and turn it into an appropriate sized region for "enters' and "leaves" scenarios. For the navigational queries we have trip time in seconds and trip distance so we can calculate average trip speeds. We also have passenger counts and some costs so we can glean some information on passenger movements and trip costs. I have started building these type queries as well.

As far as the data goes, I am still working on verification to ensure that the dataset we end up querying against provides accurate information. There are discrepancies between trip seconds compared to start time minus stop time. I am trying to figure out how and what I am going to consider valid data related to this. I am also trying to figure out what exactly medallion and hack license actually mean. For instance, I would think that hack would relate to a license unique to each taxi driver but seem to be unable to correlate that idea to the actual data in the database. There seem to be far too many unique hack licenses to make sense against a reasonable idea on the number of taxi drivers.