Sam Boyles
Zac Knopp
Devin Davies
Shane Sword

INFO263 Assignment

Our website essentially operates as an endless loop of API calls and map updates. Initially when the website is loaded, a loading icon is displayed which represents the website querying the database and Auckland Transport API. This ensures the user knows the system is working while it is busy doing the queries, and hasn't frozen. *this query is done using SQl and PHP(server side).*

While we could have used another API call instead of querying the database it would have been slower. Once our site has received information from these sources, it is processed on the client side. The database results are compared against the API results to filter down which routes currently have buses travelling on them. This filter is done using our javascript function "loadCurrentRoutes" which also includes an ajax "get" request. We do this because the database call returns EVERY possible route a bus could take in Auckland, many of which do not have any buses on them at a given time i.e late at night. By filtering these results we no longer have empty routes displaying in our drop down box. For this reason, accessing the website at two different times may yield a different set of available routes.

All the processing done on the map.js file, including adding the bus markers and resizing based on the spread of markers is also done on the client side. When the drop down box is filled with all current routes a user may then select one of them. Upon this selection our code performs an ajax "get request" to retrieve the data associated with that route i.e Vehicle i.d, vehicle latitude and longitude . This information is then used to display our bus marker using the "showVehicles" function. Our map updates every 30 seconds, the buses also send out their location every 30 seconds. With this we needed to use another ajax "get" request each time the busses send out their location, this is in our "updateBusPositions" function. In this function we are updating only the route selected by the user.

The tasks weren't explicitly separated between each of us. We usually met up and worked through each part in a group and anything that needed fixed up was done in our own time. Meeting as a group worked well as usually one of us was searching for a solution while the other was writing and testing the code. This system was good because it ensured that each of us were understanding how the code worked. It also gave some freedom in terms of being able to work on the project at home.

We generally used facebook messenger to communicate and put comments within the code when pushing through Git to show our ideas and changes to the code. We used git to manage the sharing of our code and used Github to simplify this process. This method again let us work on small niggly things that we didn't have time for when meeting in person. It also allowed us to play around with the code without it affecting the other group members side as long as we didn't push it. The group collaborated well and we were able to work together to create the website.

The group members that are quite good at programming and problem solving were usually the ones who figured out the difficult problems. The only downside to this was the possibility of the other group members not understanding what was going on. To overcome this issue

we made sure that who ever implemented the feature clearly explained the solution to the rest of the team. Using git was a really good way of working on the same project from different places.

Something that didn’t go well was trying to get the environment set up so it was working on everyone's computer’s. This proved somewhat difficult as some of us had no issue with PhpStorm, MAMP or git but others had issues with WAMP, git and setting up PhpStorm with GitHub. This caused a few problems as some of the group members weren’t able to contribute when working from home. As this was the case, keeping the group informed via facebook messenger as to what changes were being made and where was key to ensuring everyone had a solid understanding of the code. Setting up the environment better at the start before getting into development would definitely be something we would do next time to maximise efficiency.