**INFO 243 Course Project**

**Auckland Transport Vehicle Tracking Web Application**

**Project Report: Description and Reflection**

1. **Description**

**1.a. Purpose of Web Application (What does it do?)**

Our website combines many different programming languages such as PHP, MySQL, JavaScript, jQuery, CSS and HTML as well as techniques and methods including AJAX, JSON and APIs.

The purpose of the website is to display a list of routes for the user to select, once a route is selected the google map API will display (using markers on the map) the locations of all the vehicles that are on this route. Then if a marker of a vehicle is clicked on, additional information about this particular vehicle is displayed. Additionally, the google map API is able to resize each time a new route is selected so that all vehicle markers can be seen at the same time on the map, and it also refreshes every 30 seconds, showing the vehicles moving along their routes, as indicated by the markers on the map.

**1.b. How our web application works & why we have done it this way?**

In a more detailed explanation, the dropdown list is populated using the attribute from the “routes” table called route\_short\_name (which is a short, mainly 3 digit, code or abbreviation of the name of the particular bus route). This “routes” table is one of three (3) tables from the “akl\_transport” database.

This is done by using PHP to connect to the database (by creating a connection object, and using some parameters from the config.php file which is required). Then the SQL query (which requires no user input, and therefore no sanitizing) is performed, using this connection object. Then the results are retrieved, and using a “for loop” in PHP and HTML <option> and <select> tags, we are able to display each of the route\_short\_name’s in an HTML dropdown menu. This is the most appropriate way to display a list/array/grouping of options of this size. Then we close the connection to avoid running out of memory, since we don’t need it anymore. This populating of the dropdown list is all done on the server-side, as it uses the database and is querying it using the SQL.

Then when a selection is chosen, on the client-side, it is recognised using JavaScript, at which point a function is triggered, using the “onchange” property of the <option> <select> dropdown list. This function stores the selection as a variable (on the client side), then sends this variable to a PHP file (called getTripId.php) using an AJAX post in jQuery. Then on getTripId.php, there is a check to see that a selection was actually made and posted to this file. Then assuming a selected route\_short\_name was received from this AJAX post; it is sanitized to avoid malicious injection. Then, on the server-side, the route\_short\_name that was selected is then used in a prepared statement to query the database using SQL.

This query is a join query, using the route\_id on the “routes” table to join it with the “trips” table, as the route\_id, which is the primary key of the “routes” table, is also a foreign key on the “trips” table (a one-to-many relationship). This makes sense, as it was explained in the instructions of this project that a route would have two or more trips. The point of joining these two tables is that the Auckland Transport API requires parameters of trip\_id’s to then find and produce vehicle position information (such as latitude and longitude). Therefore, since the trip\_id is only on the “trips” table, and the route\_short\_name is only on the “routes” table, we must join these tables, using their one-to-many relationship, which is the route\_id being an attribute on both tables.

Assuming this is successful and produces results, these results of trip\_ids from the “trips” table are returned and these are stored into a numerical array. This is then converted to an associative array, which along with the API key for Auckland transport (defined in config.php file which is required by the getTripId.php file) and the URL for vehicle locations are passed through the requests function which puts this data together and uses it to retrieve the corresponding vehicle location information from the Auckland transit website in JSON form.

Then this JSON object is decrypted using PHP, and is sorted through to find where vehicles belong to the trip\_ids and then important information such as their latitude, longitude and other additional information is saved and then a JSON object is created with this information which is then sent to the client-side.

Then using JavaScript, on the client-side, the information in the JSON object is used to create new markers on the google map API to display each vehicle belonging to the route which was selected on the dropdown menu. Then other tasks relating to the map are performed, including removing any markers from the previously selected route, resizing the map to fit in all of the markers from this selected route, and refreshing the map every 30 seconds so that as the vehicles move, the map is able to accurately show their position as they move. Additionally, if there are no vehicles belonging to the selected route, then a relevant error message/feedback will be displayed for the user, so that they know what has happened and so it helps their choices be successful in the future.

Finally, we have styled up our webpage very nicely, making the dropdown menu look much better than the HTML default, we are also displaying the full names of the routes, to help make things easier for the users, we have clear and useful feedback by using error messages, which even have colour-coding using jQuery. Additionally, we have used CSS to customise the information windows that pop up when a vehicle markers is clicked on.

1. **Reflection**

**2.a. How we communicated with each other**

Our development group’s primary means of communication was done using Facebook’s Messenger app, since we all had this already, it was very easy to add each other (if we weren’t already connected). This way it was much easier and faster, and less formal, to put quick messages up about availability, or unavailability, regarding the organising of meetings or sharing ideas.

We had our first meeting as a group in the first week of the holidays, and then has subsequent meetings once a week, except the odd week when we were sick, out of Christchurch or had other urgent commitments (such as tests in other courses). This was a good chance for us to share what we had picked up and learnt through the week from our own research online, talking to class mates or others or just from tutorials or Dion.

**2.b. Managing and Sharing of Code**

As a development group, we managed and shared our code with each other using GitHub. This was something that was new to all of us before this course and this project. However, after watching the helpful online tutorial and explanation videos it became quite a comfortable and simple thing to use. It was great how we could have an integrated set of our files and they were very easy to change and commit, as well as attaching quick messages about what you changed or why.

**2.c. Separation of tasks between team members**

We, as a development group tried to keep our separation and delegation of tasks as even as we could between team members, but obviously in reality it will never be exactly even due to other courses, other commitments, availability on certain days or weeks and other factors including proficiency in the topic and languages. Therefore, while we always left a group meeting with different ideas or things to try out in our own time, so we could all come back to our next meeting with ideas on how to solve the problem we had reached. In terms of actually typing our code, we tended to do this during meetings so we could discuss the logic before just trying stuff randomly, and then also some of us would ask Dion specific questions at our lab tutorials and add in new lines of code and commit onto our repository so everyone else could see the change and had this when they next went to use our files.

**2.d. What went well in this project**

As a development group, we communicated well and so we were always away of potential issues relating to absences to group meetings well ahead of time. We shared our different tasks well and assigned things to the most suitable team member when we split up tasks. We collaborated well as a team, and we had a good cooperative and friendly environment. We made sure that we thought things through and had a good understanding of the bigger picture and what we were trying to achieve for each step of the way while programming the website. Also, we made sure we kept solving problems, getting things working and ticking off tasks, requirements and “to dos” each meeting and each lab tutorial, and by doing this we were able to complete everything to a very high standard, and in a reasonable amount of time, by submitting this roughly one week before the final due date.

**2.e. What didn’t go well in this project**

One thing that perhaps didn’t go well in this project, however was out of our control for most of the time, as that we had separate tutorial times, and it didn’t really suit any of us to change our slot to join up with other group members, due to other commitments (mainly other courses at uni). Because of this, it meant that we couldn’t collaborate with our team members at lab tutorials and that when we learnt something through the exercises or from some helpful advice or ideas from Dion, that only one of us was there for it, not several, so it was harder for us to pass information on to each other than it would have been for other groups who shared the same lab tutorial slots.

**2.f. What we would do differently next time**

Overall, we are very pleased at what we have achieved during this group project, both for the end result we have produced, but also for our teamwork, cooperation and hard work that we have maintained and continued along the course of this group project. We could have potentially tried to meet up more often, and made more of an effort to be at the same labs in order to make our communications more efficient and to ensure that the entire group was “on the same page” more often. However, we still think that we controlled this well enough and were still able to get the optimal result in the end.