Group H
Ebba Anderson, 65573347
Sarah Bates, 81767186
Harrigan Davenport, 98127669
Erika Savell, 96267748

INFO 263 Web Development & Web Design Project Report

This group report for the Auckland Transport Vehicle Tracking project contains a description of the functionality for the client- and server side of our website. We also present the process in which we made decisions as a group to succeed with this project, as well as a reflective summary in which we reflect on what went well and what we will do differently in future projects.

For both server and client side functionality, index.php was our main controller, calling the functions when required.

Server-side functionality

Our server side functionality utilises php to access the Auckland Transport API, following the format laid out in the code skeleton that was provided. The initial call to the transport API is in the function getAllRoutes(), that we had return the name and id of the route. Our function getTripInfo() then took various trip ids and called the transport API, returning information about all vehicles making that trip, including their id, latitude, longitude, bearing and time stamp so that we could accurately log their locations on the map. This would be called on 30 second intervals to give us continual updates of the vehicle locations.

After receiving this information from the API we encoded the data (currently in php form) into json, so we could then pass it into our javascript functions

Client-side functionality

We utilized javascript for our client-side functionality. After the data had been encoded into Json it could be passed to our various google maps functions saved in our map.js file. When the site is first called out initMap() function is called to initialise the map and set listeners for auto resizing. The function initialiseRoute() would take an array of coordinate objects and place current markers for them along the map. This function was called at 30 second intervals to update the location of the markers. The function drawRoute() would then take information of the route and draw it along the map, focusing the map to the center of the route.

For the client-side functionality we used JavaScript as well as HTML and CSS.

The HTML is used to give structure to our webpage on which we have put a map from the Google Maps API to present the bus routes from the Auckland Transport. We also have a navbar, a header and a footer. All of these are then styled with CSS in different colors as well

as different styles for buttons and fonts. We used some Bootstrap to style our webpage as for the buttons in the table where we present all the bus routes.

Decision-making process

We started by reading the project description and made a Facebook-group in which we could keep in contact and set up weekly group meetings throughout the assignment. Our first meeting was a moment for everyone to get to know each other and also separate the different tasks between each other. This was a decision we made together as everyone felt comfortable working remotely with the task each of us were given. Since we had weekly group meetings, we could meet and brainstorm our ideas and analyze different choices in how to write the different functionalities for the code. From the start, we made a plan of how long each task would take in order to finish the assignment and also have enough time to work on possible bugs and other tasks that could happen before submitting the code. We decided to have weekly group meetings, where we went through everyone's code for their given task and how we would continue on to the next step of the assignment. In every decision we made, we were all a part of the process and we made all decisions together.

Reflective summary

At our first meeting, we set up four milestones, each describing a list of tasks and the date it needed to be done by. We did some work on the assignment together so that everyone had an idea of how it would be set up, then allocated tasks to take us through to the next week. From there, we met up for weekly meetings where we presented the code we had been working on since the last meeting. After presenting, we assigned next week's tasks. Most meetings were followed by about an hour of co-location, and some peer programming to finish tasks that had been too difficult to complete by the person they were assigned to. This structure worked out really well. The weekly divisions kept workloads manageable, and ensured that all our code was merged regularly.

We chose to use GitHub to keep our codebase accessible for everyone. We had some very experienced Git users and some less experienced users in the team, so we often paired up when merging to ensure less confident members could become proficient in using Git, without any risk to the codebase. Each task was completed on its own branch, which was usually merged at the end of our weekly meeting. Pull requests were reviewed by at least one team member before getting accepted.

We are very happy with the way the teamwork played out. Everyone was present at meetings, and did their best to finish their tasks on time. We did not encounter any problems with members not contributing to the code. Everyone worked consistently on the project and helped out when others were struggling.

What we would do differently in the future would be to start earlier, even though we had a plan from the beginning for each task, every task needed more than time than what we first thought it would take. Therefore, in the future, we could be more realistic on how long each task will take. As we were finishing the project it became very apparent how useful Ajax would have been in making calls to the database without a complete refresh and this is something we would have built our app around if we were to write it again.