Design Document – Algorithm’s Group Project

Aislinn Smyth - Myles McCarthy - Sean Lysaght - Aaryan Gupta

**Part 1 – Aislinn**

I did part one of the project, which was calculating the shortest paths between two inputted bus stops, returning the route taken and the cost of this bus route. I decided to adapt my code heavily from the Dijkstra Graph Algorithm from the Algorithms, Fourth Edition book written by Robert Sedgewick and Kevin Wayne. The reason I chose to implement the Dijkstra algorithm was because this algorithm finds the shortest path from ever node to ever other reachable node instead of finding the shortest path from one starting node to another starting node. Since we had to take in user input, we did not know which bus stop the user was going to input, therefore that is why I picked Dijkstra as my algorithm of choice. Another big advantage I saw with using Dijkstra algorithm was its low complexity, it was a lot easier to implement than a lot of other shortest path algorithms in my opinion. Also, I picked Dijkstra as its specialty is to calculate the shortest path in weighted graphs, which ours was. I was only stuck between one other algorithm, Floyd-Warshall and I began implementing it before I realised that Dijkstra was a better fit. The reason I turned away from Floyd-Warshall was upon research I found out that for a higher number of nodes, the Dijkstra’s algorithm was better and more efficient, also Floyd-Warshall had a higher time complexity. Overall, based off my research I decided that Dijkstra’s Algorithm was the better fit for me.

**Part 2- Myles**

I did part 2 of the project, which consisted of taking user input of searching a bus stop name and returning all of the matching bus stop names and their corresponding stop information. I did this using a ternary search tree, in doing so, I made a recursive search and insert method which searched for the bus stop name and also insert which I used to create the ternary search tree. Then I had to adjust the names of the bus stops from the stops.txt file, which contained all the bus stop names and information. For example, WB ‘HASTINGS ST FS AVE-‘ was changed to ‘HASTINGS ST FS AVE- WB’. This was to make the search more meaningful as many of the bus stop names began with flag stop or wb/sb/nb/eb. Also I changed my program so that it is no longer case sensitive, which we see in the main when we take user input.

**Part 3 - Sean**

I did part 3 of the project and my task was to read in the file and allow then user to search for a specific arrival time. The arrival times would then be sorted by Trip ID. I considered several ways to go about searching for the arrival times but I eventually settled on using a 2d array. This was the best option because it allowed me to single out the arrival times easily and use a for loop to do so to keep the code compact. I used patterns and matching to remove any lines that contained times that were invalid. I compared the user inputed time to the arrival times using .equals. I then used an Arraylist to store the tripIDs as I didn’t know initially how large it would need to be. I then compared the tripIDs in the array list to the file and printed and lines that matched. I also used a while loop of booleans to allow the program to run again after the outputs

**Part 4 - Aaryan**

I did part 4 of the project that was making the user interface. I decided to use command line interface as I felt it was suitable to our project. I used switch cases to make the user menu which allowed the user to type in numbers corresponding to the different functionalities incorporated in the code. The only thing that blocked my progress was building the menu as I had to go over the codes of my group members to understand the methods and provide appropriate functionality.