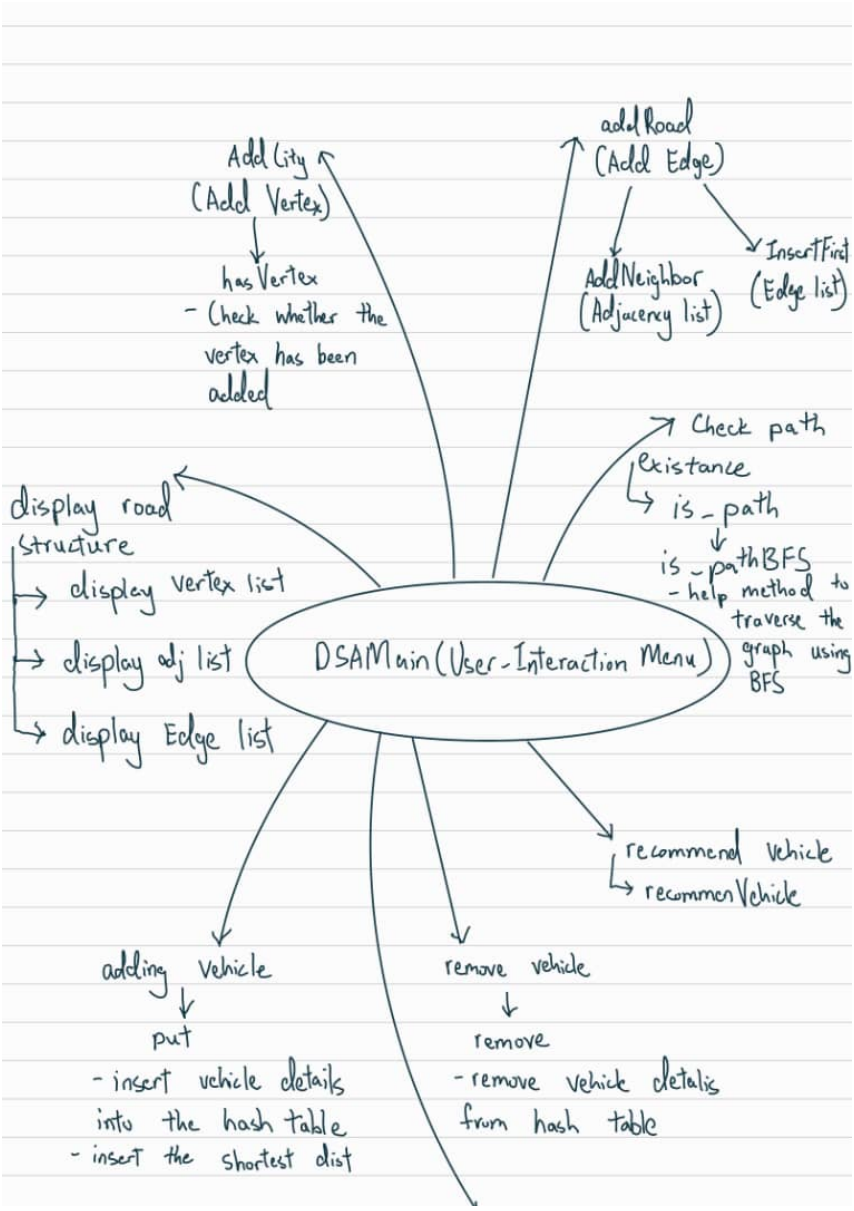
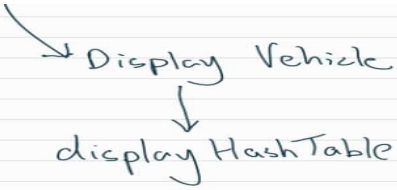
Name: Ching Mao Jin

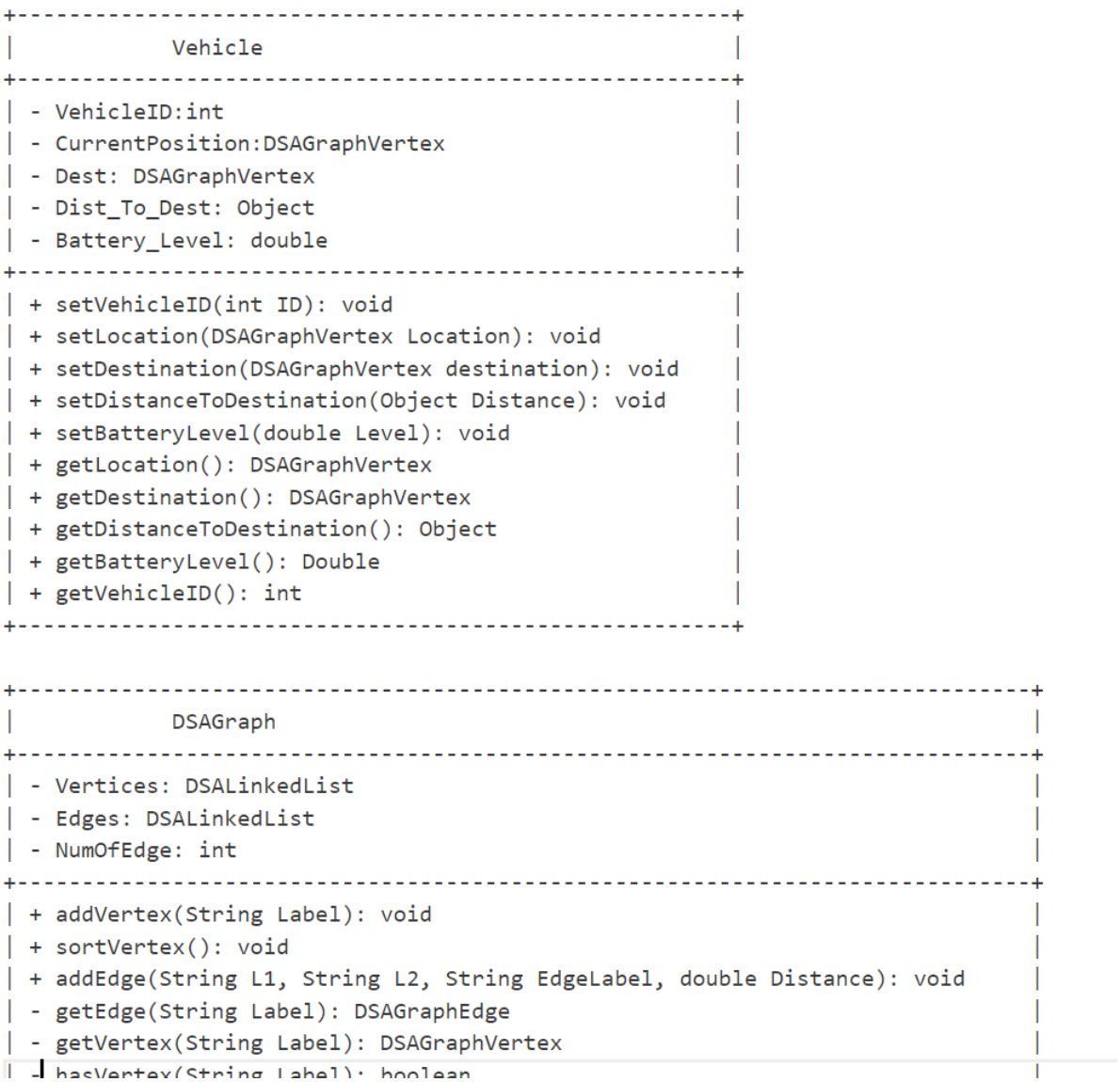
StudentID:22013213

Report

The images below shows the component interactions. The UML Model is done in a seperate file called UML.md. You will see the content below after opening the file.




Display Vehicle
display HashTable



My implementation strategy for road network is user will only interact with the DSAGraph, in this class it will call the methods it requires from another class which are DSAGraphEdge and DSAGraphVertex. There are some recursive methods in the code which requires helper method as private.

DSAGraph: use for any operation such as adding city, adding edge and display graph structure for the road network

DSAGraphVertex: used as a data structure for cities

DSAGraphEdge: Used as a data structure for roads

When it comes to adding vehicle, removing vehicle, displaying vehicle or recommending vehicle, the main function will call the corresponding methods in DSAHashTable. DSAHashTable requires a data structure for each entry of the array.

DSAHashTable: perform function requested by the user

DSAHashEntry: used as a data structure to interact with DSAHashTable

Heap array is used to for all the possible path with the distance from findPath from DSAGraph. I need to perform a heap sort on the array which requires DSAHeap and DSAHeapEntry

DSAHeap: to heap sort the array

DSAHeapEntry: Used as a data structure to interact with DSAHeap.

For sorting the battery level of vehicles, find_Vehicle_With_HighestBatteryLevel will look for all vehicles that has the same source and dest as the user input to retrieve the battery level and store in the double array for quick sort. I implemented a seperated file for quick sort which consists of the methods quickSort, partition and printArray

quickSort: recursively call itself and partition to find the pivot index for each recursive level.

Partition: set the pivot index to be the index of last element. Compare the value with low index and do swapping if required. Update the pivot index

printArray: To check if the sorted array is in correct order for debugging

Discussion: During implementation, I did face a lot of challenges. One of the biggest challenge was calculating the shortest distance from a source to dest. I had to find all the possible path from source to destination which requires a solid understanding of recursion as I used DFS for graph traversal for finding all the possible paths. In the meantime of implementation, there were a lot of logical error raised due to improper use of recursion, not clear visit of the previous node for other possible path. This caused the program to calculate the wrong result. I have to dive into the code and print out as many as statement to gain an understanding of the logic and find out which part went wrong.

Test Component	Test Data	Expected result
Graph Structure(road network)	The vertices “A,B,C,D,E,F,G,H,I,J” are added The edges are added with weight(distance) between each vertex.	Vertices and edges should be correctly added and no exception thrown. The correct output should be the following: Here's the display of the adjacent cities: The neighbor of A-> B C D The neighbor of B-> A E The neighbor of C-> A F The neighbor of D-> A E F H The neighbor of E-> B D G The neighbor of F-> C D I The neighbor of G-> E H J The neighbor of H-> D G I J The neighbor of I-> F H J The neighbor of J-> G H I Here's the display of the distance between the cities: RoadLabel:IJroad Distance:6.0. RoadLabel:HIroad Distance:5.0. RoadLabel:HJroad Distance:180.0. RoadLabel:GJroad Distance:400.0. RoadLabel:GHroad Distance:170.0. RoadLabel:FIroad Distance:4.0. RoadLabel:EGroad Distance:300.0. RoadLabel:DFroad Distance:3.0. RoadLabel:DHroad Distance:160.0. RoadLabel:DEroad Distance:150.0. RoadLabel:CFroad Distance:2.0. RoadLabel:BEroad Distance:200.0. RoadLabel:ACroad Distance:1.0. RoadLabel:ADroad Distance:100.0. RoadLabel:ABroad Distance:110.0.
Shrtest Distance	Source= ‘A’ and Dest= ‘J’	The getDistance method returns the correct shortest path distance based on the graph structure which is 13 in this case
Highest Battery Vehicle	Source= ‘A’ and Dest= ‘J’ V1=65.5% V2=70%	The find_vehicle_with_Highest batteryLevel method should return the vehicle with the highest battery level and at the same time ensure the source and dest of the vehicle tally with the user input of source and dest.
Recommended Vehicle	V1ID:1, Source: ‘A’, Dest = ‘J’ , BatLevel= 65.5% V1ID:2, Source: ‘A’, Dest = ‘J’ , BatLevel= 70%	The method recommenVehicle(“A”, “J”) will return the vehicle that has the same source and dest as the user input with the highest battery level.
HashTableDisplay	V1ID:1, Source: ‘A’, Dest = ‘J’ , BatLevel= 65.5%	DisplayHashTable() method will display the vehicle information of all the added vehicles. Here’s the

	V1ID:2, Source: ‘A’, Dest = ‘J’ , BatLevel= 70%	outcome: The VehicleID at index 1 is: 1 The battery level of the vehicle at index 1 is: 65.5 The current location of the vehicle is: A The destination of the vehicle is: J The distance to destination is: 13.0 The VehicleID at index 2 is: 2 The battery level of the vehicle at index 2 is: 70.0 The current location of the vehicle is: A The destination of the vehicle is: J The distance to destination is: 13.0