Bus Tracking Application

Group 19: NoSQL++

Joey Lim Jun Feng, 2101483@sit.singaporetech.edu.sg

Ng Wee Yang Ryan, 2101112@sit.singaporetech.edu.sg

Parris Cheng, 2101175@sit.singaporetech.edu.sg

Edwin Ng Jun Yuan, 2102597@sit.singaporetech.edu.sg

Tee Wen Jun, 2101624@sit.singaporetech.edu.sg

Foo Jieyu Danyel, 2100794@sit.singaporetech.edu.sg

Description of Application

The bus is a popular mode of transportation in Singapore, however, only a limited number of bus stops have an electronic panel to display the bus information. Therefore, this application will make sure that the bus information can be accessed easily through their mobile phones at any time. The development of the application bridges learning data management and information systems to applying these concepts in real world applications.

This application uses the data of all bus arrivals, services, and bus stop numbers in Singapore. The application will be able to read all the information of each bus when the user enters the bus stop code number and will proceed to display that information such as the timing of the first and last bus, and the bus services in that bus stop. Using the user's location, a list of suggested bus code numbers will be prompted for the user to decide.

Data

Bus Services https://datamall.lta.gov.sg/content/datamall/en/dynamic-data.html. This dataset contains information on all buses currently in operation, their first stop, last stop timing peak / off-peak frequency of dispatch. There are 700 rows.
Bus Stops https://datamall.lta.gov.sg/content/datamall/en/dynamic-data.html. This dataset contains information of the Bus Stop's code and their location coordinates. There are 5,000 rows.
Bus Routes https://datamall.lta.gov.sg/content/datamall/en/dynamic-data.html. This dataset contains information of the Bus Stop that the bus travels along and their codes. There are 25,500 rows.

Database Table

ServiceNo	*Operator*	*Direction*	*StopSequence*	*WD_FirstBus*	*WD_LastBus*
10	SBST	1	1	0500	2300

SAT_FirstBus	*SAT_Last Bus*	*SUN_First Bus*	*SUN_LastBus*	*RoadName*	*Description*	*Latitude*
0500	2300	0500	2300	Tampines Ctrl 1	*Tampines Int*	*1.354075524*
Longitude	*Category*	*OriginCode*	*DestinationCode*	*Longitude*	*Category*	*OriginCode*
103.943391	*TRUNK*	*75009*	*16009*	*103.943391*	*TRUNK*	*75009*

Figure 1: One document of the database

The table represents a document which contains the respective fields and values. The values are in string-type format.

NoSQL Migration

Prior to the migration, it was necessary to redesign the queries and amend the data structure. Through the process of denormalization, there were some attributes that were removed and functions within Excel such as “vlookup” were used to combine the data in the three tables into one. The document was merged to suit the database and converted into a JSON file type to be inserted into mongodb. Furthermore, indexes for the bus service number have been created to improve the speed and performance of the queries.

Project Management (Gantt Chart)

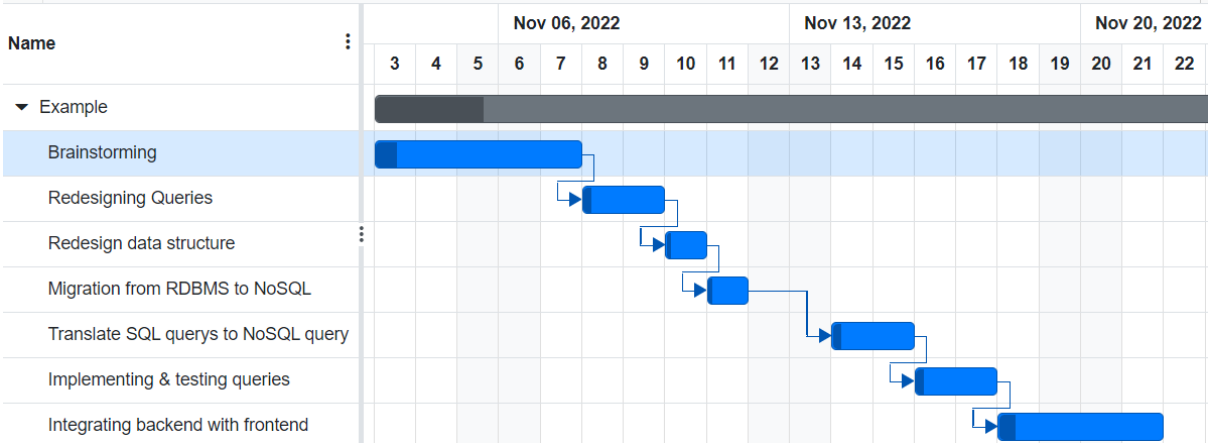

Figure 2: Gantt Chart

"By failing to prepare, you are preparing to fail", a quote by Benjamin Franklin that resonated with our group. Therefore we decided to create a Gantt Chart to have a projected timeframe for when we should complete each assignment.

Database Implementation & Demo

In this demo, we are using MongoDB for our database and we will show the output for our 5 queries that are made in MongoShell. The syntax for the queries are in NoSQL format.

Query 1: Finding the bus services at a bus stop using the bus stop code

Mongo Syntax: db.coordd.find({‘BusStopCode’: “19031”}, {“ServiceNo”:1, “Description”:1, “_id”:0})

Description: The query uses a find to search for the buses available in that bus stop

```
27017> use projecttest
switched to db projecttest
projecttest> show collections
busroutes
busservices
busstops
coord
projecttest> db.coord.find({'BusStopCode': 19031},{"ServiceNo":1, "Description":1, "_id":0})

projecttest> db.coord.find({'BusStopCode': '19031'},{"ServiceNo":1, "Description":1, "_id":0})
[
  { ServiceNo: '105', Description: 'Dover Stn Exit A' },
  { ServiceNo: '166', Description: 'Dover Stn Exit A' },
  { ServiceNo: '185', Description: 'Dover Stn Exit A' },
  { ServiceNo: '106', Description: 'Dover Stn Exit A' },
  { ServiceNo: '74', Description: 'Dover Stn Exit A' },
  { ServiceNo: '14', Description: 'Dover Stn Exit A' },
  { ServiceNo: '147', Description: 'Dover Stn Exit A' }
]
projecttest>

projecttest>
```

Figure 3a: Query 1 on MongoDb

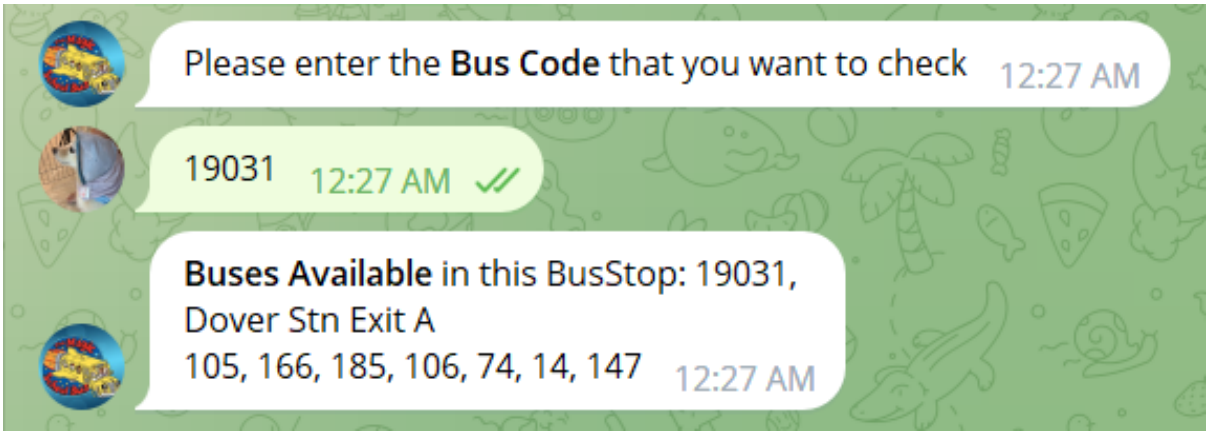

Figure 3b: Query 1 on MongoDb

Query 2: Showing the bus route for a specific bus service and shows

Mongo syntax: db.coord.find({“$and”: [{“ServiceNo”: “194”}, {“Direction”: “1”}]}, {“BusStopCode”:1, “Description”:1, “_id”:0})

Description: The query uses the “and” function to match the service number and display the bus’s stop sequence.

```
projecttest> db.coord.find( {"$and": [{ "ServiceNo": '194'},{ "Direction": '1'}]}, {"BusStopCode":1, "Description":1, "_id":0})
[
  { BusStopCode: '22009', Description: 'Boon Lay Int' },
  { BusStopCode: '22011', Description: 'Jurong Bird Pk' },
  { BusStopCode: '22021', Description: 'Givaudan' },
  { BusStopCode: '22031', Description: 'Bef Pioneer Circus' },
  { BusStopCode: '22039', Description: 'Markono Holdings' },
  { BusStopCode: '22029', Description: 'Opp Givaudan' },
  { BusStopCode: '22019', Description: 'Opp Jurong Bird Pk' },
  { BusStopCode: '22009', Description: 'Boon Lay Int' }
]
projecttest>
```

Figure 4a: Query 2 on MongoDb

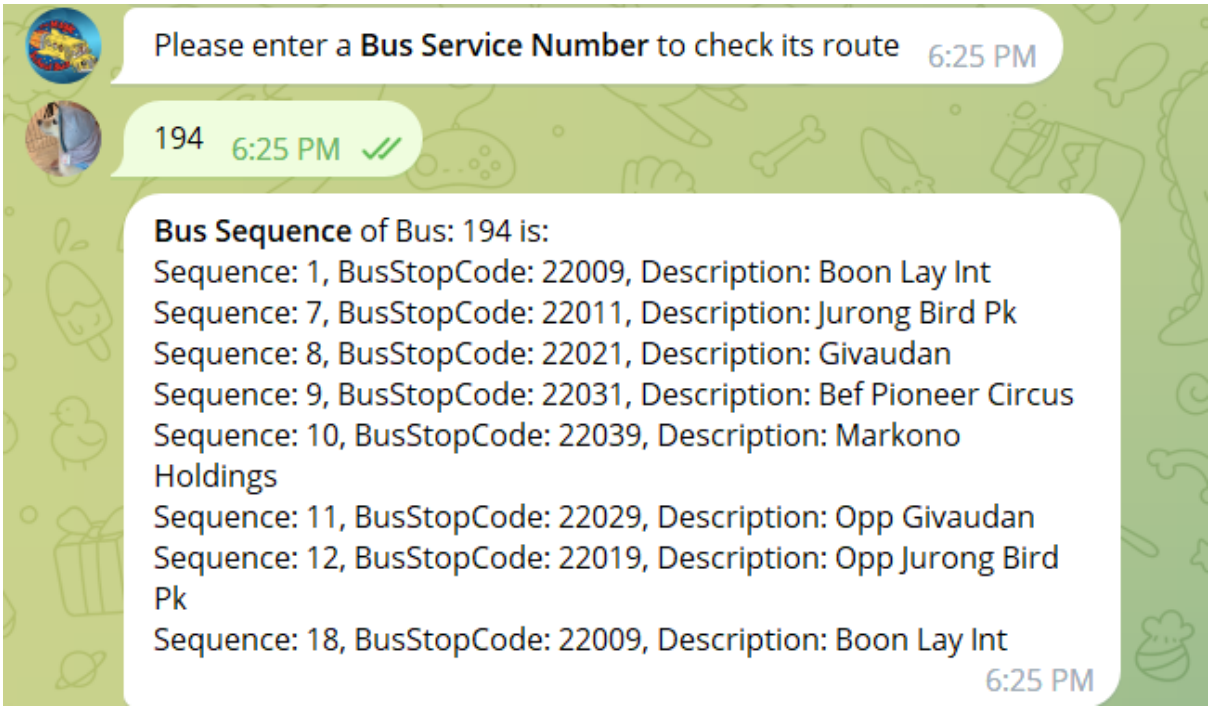

Figure 4b: Query 2 on Telegram’s Front-end

Query 3: Finding the nearby bus stops using longitude and latitude, showing bus service, bus stop code and the description of the bus stop

Mongo syntax: db.coord.find({ $and: [{ $and: [{ Latitude: { $gt: "1.3098" }, Longitude: { $gt: "103.7775" } }] }, { $and: [{ Latitude: { $lt: "1.3133" }, Longitude: { $lt: "103.781" } }] }] }, { ServiceNo: 1, BusStopCode: 1, _id: 0, Description: 1 }).sort({ Description: 1 })

Description: The query uses the coordinates of the user and adds a magic number of 0.0025 to approximate the area around it to find the bus stops within.

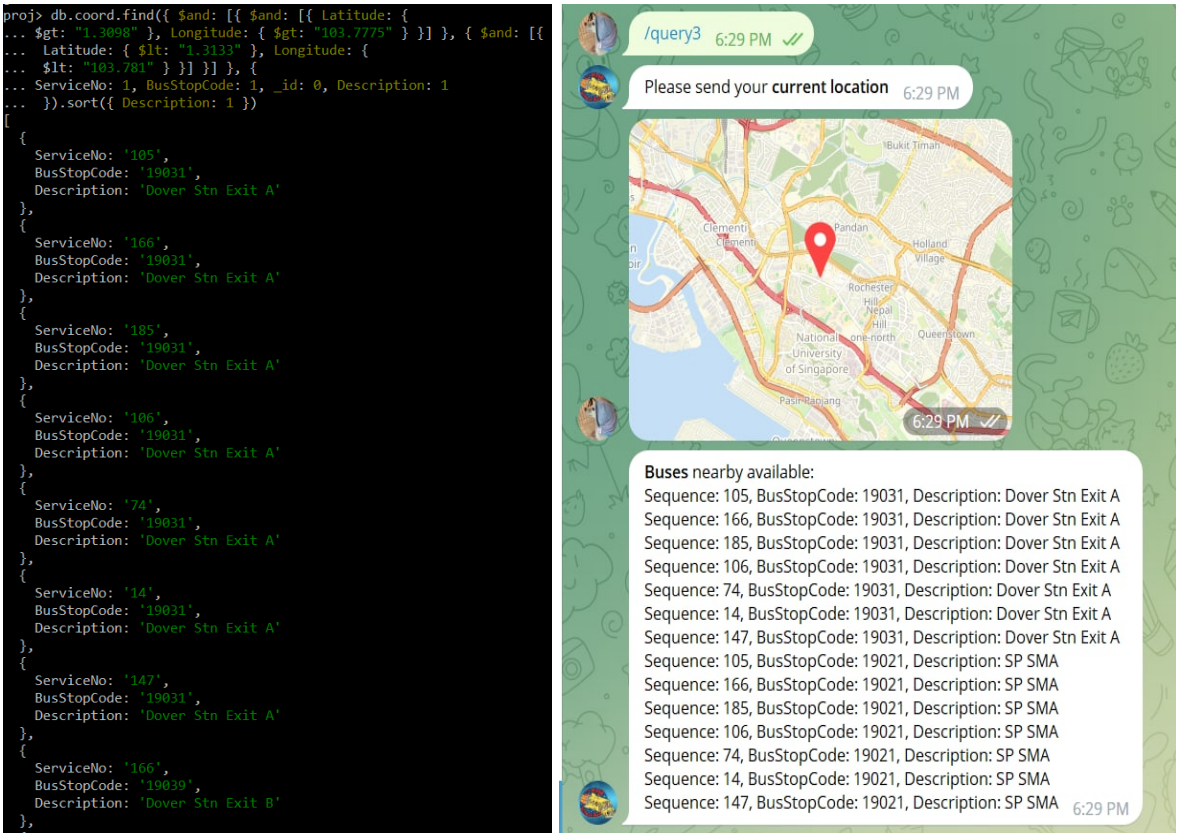

Figure 5: Query 3 on Telegram's Front-end

Query 4: Check whether the bus is in operation from the starting point for a specific bus service.

Mongo syntax: db.coord.findOne({"$and":[{"ServiceNo": '181'}, {"Direction": '1'}]}, {"ServiceNo":1, "WD_FirstBus":1, "WD_LastBus":1, "SAT_FirstBus":1, "SAT_LastBus":1, "SUN_FirstBus":1, "SUN_LastBus":1, "Category":1, "_id":0})

Description: The query returns one value of the bus's description such as the bus first and last timing.

```
projecttest> db.coord.findOne({"$and":[{"ServiceNo":'181'},{"Direction":'1'}]},  {"ServiceNo":1,"WD_FirstBus":1,"WD_LastBus":1,"SAT_FirstBus":1,"SAT_LastBus":1,"SUN_FirstBus":1,"SUN_LastBus":1,"Category":1, "_id": 0})
{
  ServiceNo: '181',
  WD_FirstBus: '600',
  WD_LastBus: '30',
  SAT_FirstBus: '600',
  SAT_LastBus: '30',
  SUN_FirstBus: '600',
  SUN_LastBus: '30',
  Category: 'TRUNK'
}
projecttest>
```

Figure 6a: Query 3 on MongoDb

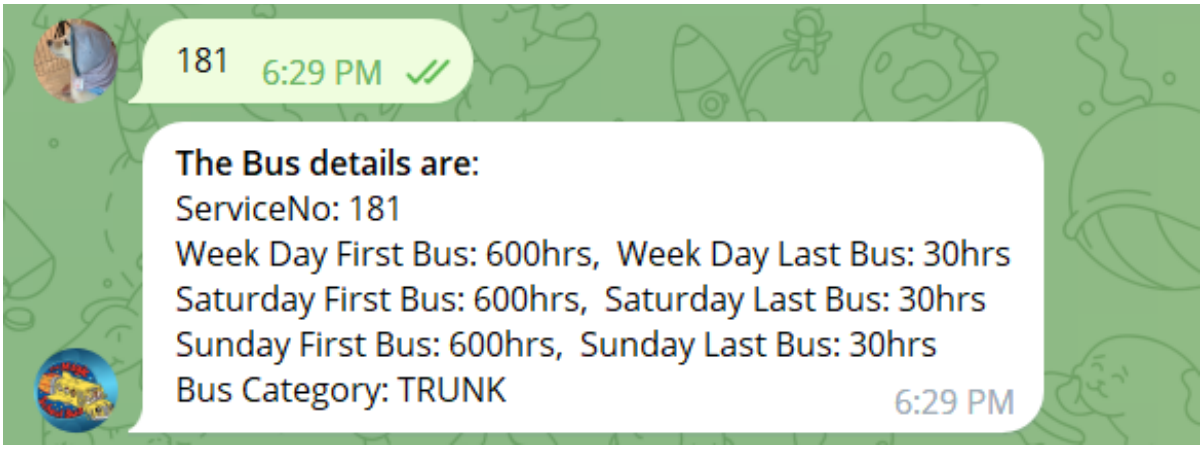

Figure 6b: Query 4 on Telegram's Front-end

Query 5: Finding the bus service available for users to take from point 1 to point 2. In this case, point 1 refers to "Jurong Bird Park" and point 2 refers to "Boon Lay Int".

Mongo syntax: db.coord.aggregate([{$match:{$or:[{Description: "Jurong Bird Pk"}, {Description: "Boon Lay Int"}]}}, {$group:{_id:{ServiceNo: "$ServiceNo"}, count:{$sum:1}}}, {$project:{count:{$gt:['$count', 2]}}}, {$match:{count:true}}])

Description:This query uses an aggregation pipeline to filter out each condition in order to print out the bus service number.

```
projecttest> db.coord.aggregate([{$match:{$or:[{Description: "Jurong Bird Pk"},{Description: "Boon Lay Int
"}]}}, {$group:{_id:{ServiceNo:"$ServiceNo"},count:{$sum:1}}},{$project:{count:{$gt:['$count',2]}}},{$matc
h:{count:true}}])
[ { _id: { ServiceNo: '194' }, count: true } ]
projecttest>
```

Figure 7a: Query 5 on MongoDb

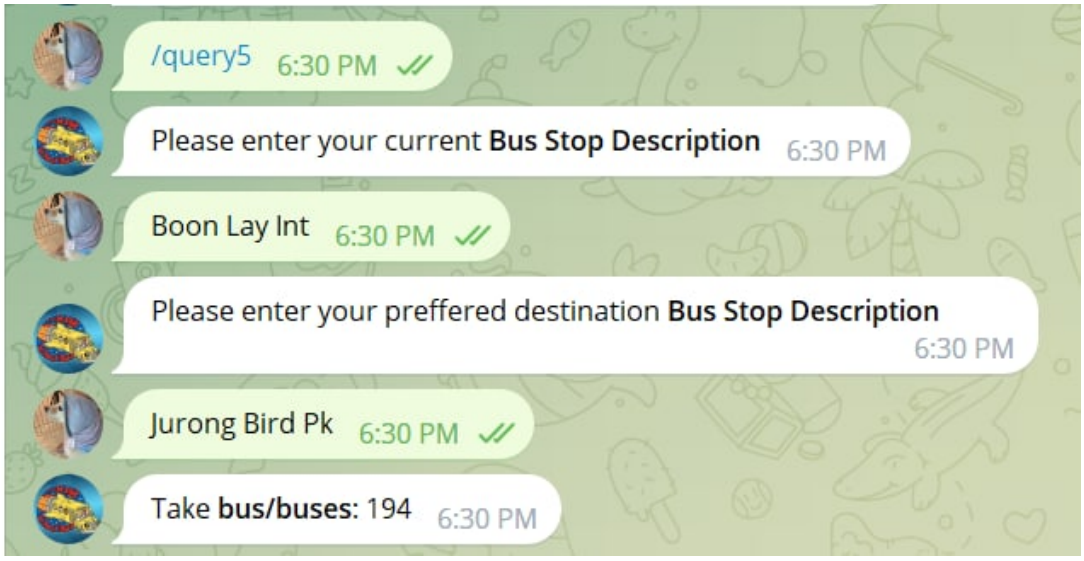

Figure 7b: Query 5 Telegram's Front-end

Data Analysis

main.py

Telegram-bot > main.py > ...

```
20  plt.figure(figsize=(11,7))
21  plt.rcParams.update({'font.size':18})
22
23  cursor = BusApp.find()
24
25  mylist = []
26  mylist2 = []
27
28  #empty counter
29  b = 0
30
31  for item in cursor:
32
33      if item["Latitude"] == '0' :
34          #print("skip")
35          b = b+1
36
37      else:
38          mylist.append(item["Latitude"])
39          mylist2.append(item["Longitude"])
40
41  print(type(float(mylist[0])))
42
43  print(len(mylist))
```

Figure 8a: Python Code from Data Analysis

```
44
45  i=0
46  while i < (len(mylist)-b):
47      plt.plot(float(mylist[i]), float(mylist2[i]), marker="o", markersize=1, markeredgecolor="red", markerfacecolor="red")
48      i = i+1
49
50  plt.xlabel('Latitude')
51  plt.ylabel('Longitude')
52
53  #Singapore main coordinates
54  plt.plot(1.3521, 103.8198, marker="x", markersize=8, markeredgecolor="black", markerfacecolor="black")
55
56  #uncomment to plot the graph
57  plt.show()
```

Figure 8b: Python Code from Data Analysis

This queries returns all the buses' coordinates and is stored into two lists, “mylist” that stores the Latitude and “mylist2” that stores all the values of the Longitude. These are values with non existing coordinates that are removed from the list through the if condition and the coordinates are plotted on the graph. The main axis which are the values of the Latitude and the Longitude using the function ‘plt.xlabel’ and ‘plt.ylabel’. The code then plots the graph using the “matplotlib.pyplot” library and displays the graph on another tab as shown below.

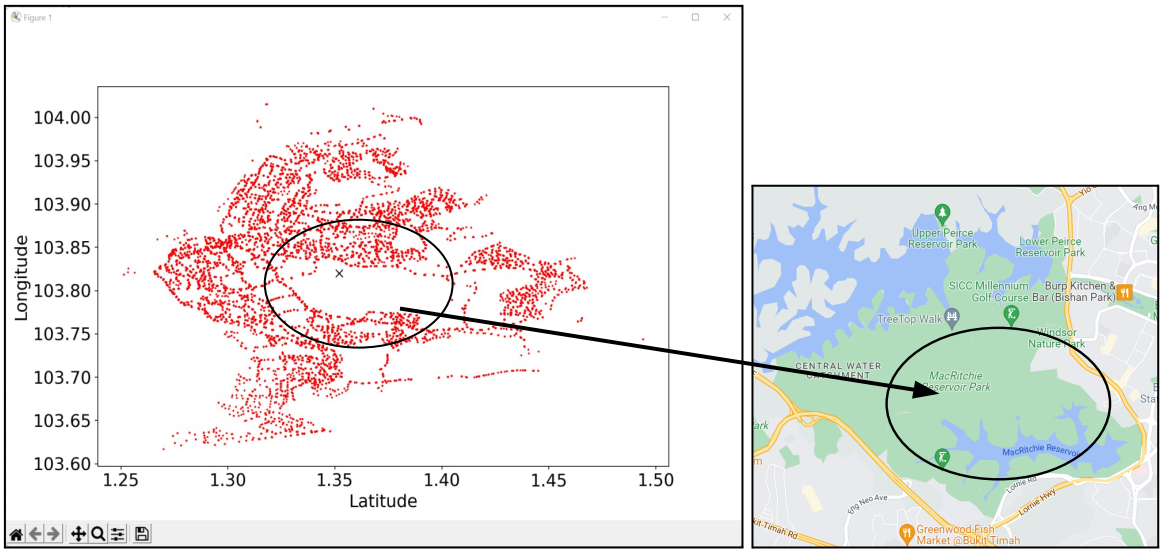

Figure 9: Bus Stops Plot and Google Map of MacRitchie

Graph Data Analysis

After the code is executed, a graphical plot of the Bus Stop concentration in Singapore is generated. One unique observation from the graph is that there is a large concentration of bus stops in the west side. This might be in relation to the west side being one of the most populated areas in Singapore with 47% of the total population living there according to “citypopulation Singapore”. Another key observation from this plot is that there are no bus stops within the centre of Singapore. This is because of MacRitchie being in the centre of Singapore as shown in Google Maps.

Discussion and Reflection

Through this project, we have gained an understanding of the benefits and drawbacks of both SQL and NoSQL. We have implemented both functions that were taught in class and interesting ones. We had used indexing to speed up our queries from 10 seconds to 8 seconds on average and even used a '$match' to match 2 objects with the same values. This project has taught us the significance of what happens in the backend and better prepares us as future data engineers.

References

Gantt Chart. Available at: https://www.onlinegantt.com/#/gantt

MongoDB query. Available at: https://www.mongodb.com/docs/manual/tutorial/query-documents/

MongoDB manual. Available at: https://www.mongodb.com/docs/manual/