158.740 Geoinformatics

Project

A web mapping application that shows how Auckland has changed over time

Coordinator:

Dr Kristin Stock

Zhihan Wang (17272381)

Jian Lu (20003617)

Da Zhang (14192018)

Meet Shah (20011945)

Wen Xu (19036937)

Abstract

The assignment identifies the process of developing web application that can show how Auckland changes over time.

Introduction

With the changes of technologies, people can see changes of their lives as well, which also makes people wonder how city has changed during this period of time. Our task in the assignment is to design and develop a web application that is available for users to see the status of Auckland at different timing, from different aspects. We build the application using pure java environment, have four main datasets as aspects implemented and available to show to the users. In the assignment, the details of process of creating and implementing the web application will be discussed.

1.0 Design process

1.1 Project Requirements

Due to the basic understanding of assignment requirement, we have a trend to investigate through some of the existing websites that have the functions and trying to build deeper understanding of the nature of the functions and designs. Websites like Goole maps, Auckland council and OpenStreetMaps really present ideas to us about the proper way of how the web application should be built in our own way. It is believed that the builders of the websites that we have looked through should have decades of experience that we could hardly have.

1.2 Functions

1.2.1 Time Series

With changing the time, the change of the appearance of the dataset will also change. For every dataset in our assignment, the function of time series had been well implemented with every layer we have. In order to see the changes happened to Auckland, we believe the best way of showing it is to show visible changes by moving the scroll bar by users and let them see the differences by their own eyes.

1.2.2 Layer search

In our web application, the users can search by layers in the map. Every route in the map has its own code. Users are able to find the route in the map by searching it in the search box. This gives more conveniences to users to locate a point in the map faster.

1.2.3 Scale

We are able to zoom in and out simply by clicking the icon of “plus” and “minus”. The icons are on left-top of the map, and the scale can be seen on left-bottom. This brings better understanding of the map to the users as fast as they see the map.

1.2.4 Layer Control

We have designed a check box in order to show multiple layers on the map customised by user.

1.3 Technologies used in the project

1.3.1 Comparisons

Several decisions have made before doing the assignment, it is clear that we will use Java as the main language during the programming part of our assignment, but the problems were: Openlayers or Leaflet? Geoserver or Mapserver? Which is more beneficial to our project, we went through some comparison and decided.

1.3.2 Leaflet and Openlayers

Comparison between Openlayers and Leaflet was rather complex, but based on our assignment environment and requirement, we believe Openlayers should be chosen for the project.

Although they are homogenous tools, Leaflet is more lightweight and more popular than Openlayers. However, Leaflet is not perfect. The coding style in the library sometimes does meet the requirement of ours, which is C-like or prototype-based language and different from what we are asking for. In addition, Leaflet has no support for web feature services which can respond to GeoJson format data that is the one of the requirement functions for our web app.

While the Openlayers have provided web feature services functions plus it supports good Object-Oriented Programming style under ECMAScript 6 standard, that makes us implement client’s requirements much easier.

1.3.3 Geoserver and mapserver

Both of them have a great work environment in Windows system, open source and both have tools within for good development. The biggest difference between the two options is that Geoserver is Java-based and Mapserver is better with PHP developers. Our choice about

backend was decided fast, since we have already decided to use Java environment, we should pick Geoserver for our assignment.

1.4 Software and environment configuration

1.4.1 OpenJDK

For the part of back-end, we have chosen OpenJDK, Tomcat and Geoserver as the tools to build Geoserver and PostGis environment. In order to build a web mapping application backend, it is necessary to install some software in the first place. First of all, we need to install OpenJDK. Since we need to use Java as the language to build the application, we installed OpenJDK for the environment of front-end and back-end.

1.4.2 Tomcat

Apache Tomcat is an open source web server and servlet container. In other words, without Tomcat, we cannot build a web server which allows java code to be run. To make a dynamic HTTP server, we need to implement servlet on our website and create the web application. HTTP server can fetch data, and Tomcat can make the environment a pure java web server environment.

1.4.3 Geoserver

In order to show and share geospatial data, we have our web application environment installed with Geoserver. After the installation of OpenJDK and Tomcat, we can have Geoserver implemented and ready to be used. Geoserver can deal with spatial datasets and its environment is java as well. It is necessary to have Geoserver installed for the analysis of geospatial dataset and layer combination.

1.4.4 HTML

HTML is a hypertext language and language, and it is also a standard markup language for creating web pages. In this language, it contains a series of elements, and this series of elements can also tell the browser how to display content. With the development of HTML, a CSS language has gradually evolved. It is a language that describes the style of HTML documents. In other words, CSS describes how HTML elements should be displayed.

1.4.5 JavaScript

JavaScript is a lightweight, interpreted or just-in-time programming language with function priority. It is a dynamic scripting language based on prototype programming, multi-paradigm, and supports object-oriented style. In our project, we use JavaScript and NodeJS to complete the function of the web page. NodeJS is also a kind of JavaScript, we will not do too much explanation here.

1.5 Multiple Layers

In order to do a better analysis with all the datasets, it is necessary to combine and mix-up layers. For example, if we mix-up the layer of waste water and water pipes, we are able to find the relationship between these two areas covered on the map and gather information from it. The more information we have, the better analysis we can make.

1.6 Change of User Interface

To show our work with the best interface, we have been working on the improvement of user interface of our web application as well.

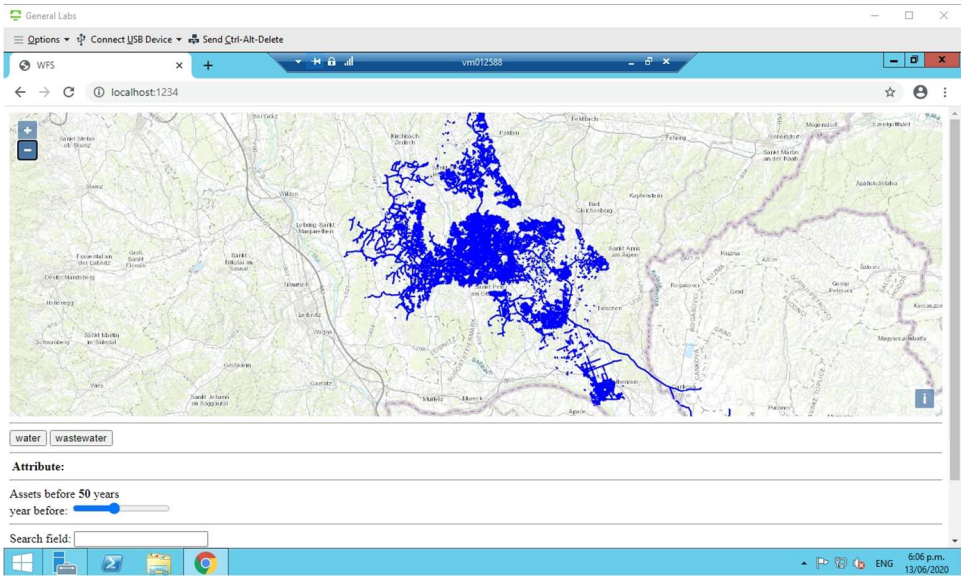


Figure 1. The picture above shows the appearance of UI we used before.

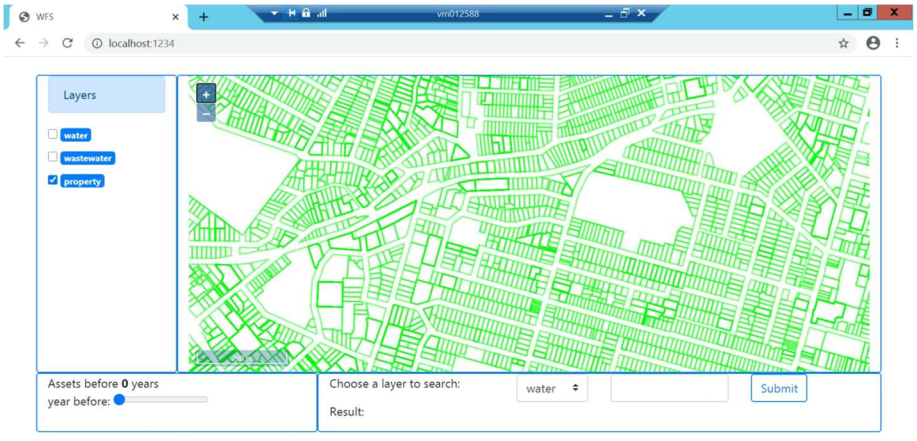


Figure 2. After changes had made to the web application, the look of it had changed for the better understanding of the map and better user experience.

2.0 Datasets Selection and Collection

2.1 Water

It is sourced from https://data-watercare.opendata.arcgis.com/datasets/water-pipe , which includes attributes such as pipe material, installed date, length etc.

To show how Auckland has changed over time, we have chosen the aspects that can best describe what improvement that Auckland have made. The water pipes are being built from time to time. It is believed that the development of the country cannot leave the process of water pipe building.

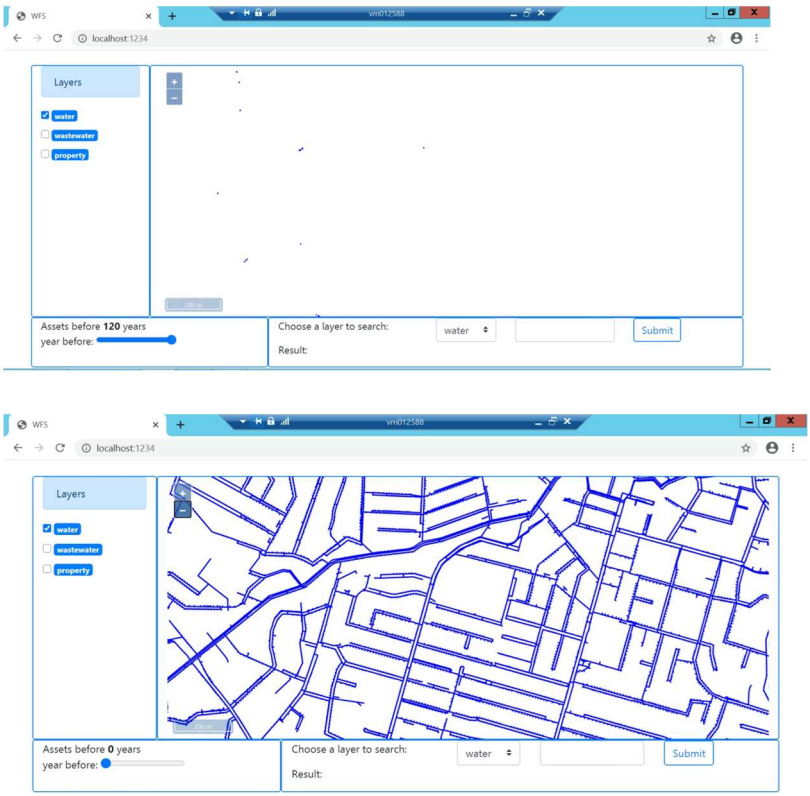


Figure 3. As shown in the two screenshots above, 120 years ago, water pipes could hardly be detected since just few of them been built, but now we have water pipes crowdedly covered the whole Auckland. The progress is visible and obvious.

2.2 Wastewater

It is sourced from https://data-watercare.opendata.arcgis.com/datasets/wastewater-pipe , which also includes attributes such as pipe material, installed date, length etc.

Similar as water pipes, the datasets of wastewater show how the trend of building wastewater pipes though time. After using water, the wastewater needs to be managed with a reliable system, and there comes the wastewater pipes. The spread of wastewater pipes shows the on-

going building process of Auckland's wastewater maintenance requirement and proper system.

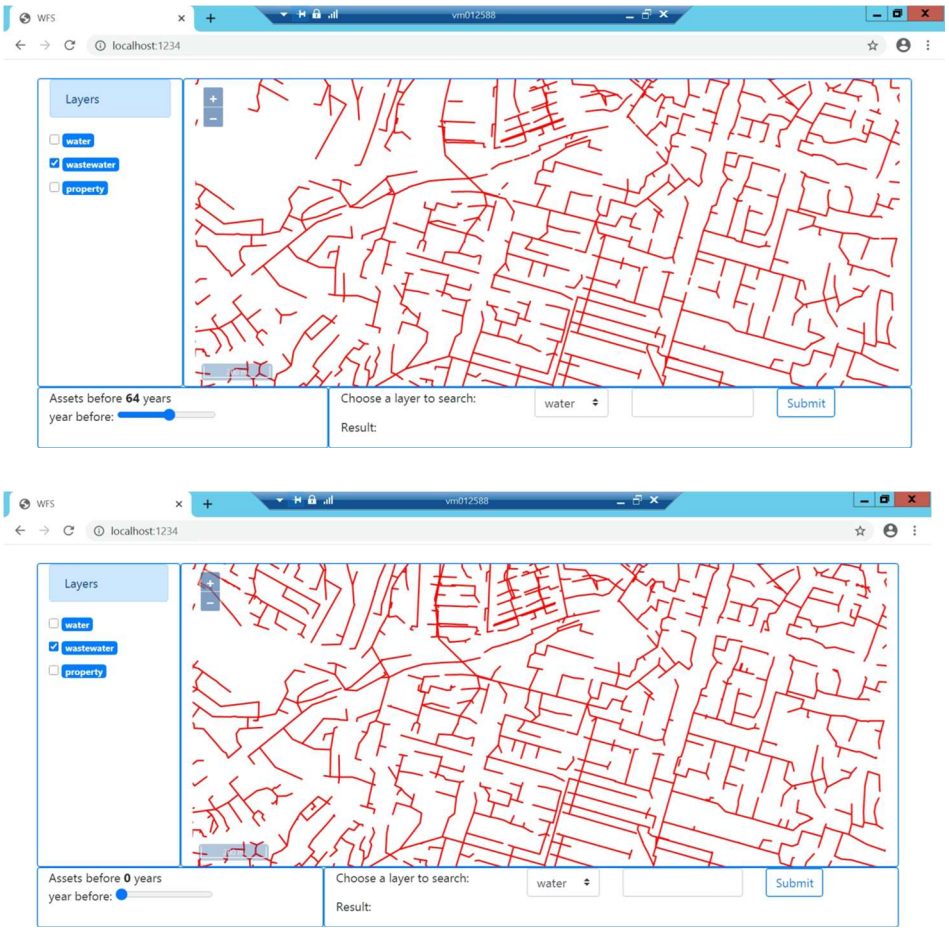


Figure 4. As the two screenshots shown above, we can see the difference between them. The dataset shows no data before 120 years, so I picked 64 years ago as a timeline to show the improvement of Auckland's wastewater pipes' statues. It is not that clear to see how it has changed, but the difference on the left-top can be found.

2.3 Property

It is sourced from https://data.linz.govt.nz/layer/50804-nz-property-titles/, which includes attributes such as title estate des, issued date, type of title etc.

The changes of properties tend to represent the process of urban spread in the country. The data of property is needed for services and mailing, and also emergency calls such as ambulance and police. In other words, the more properties have been built up, the more people we have in New Zealand. In our web application, we showed the property with green colour.

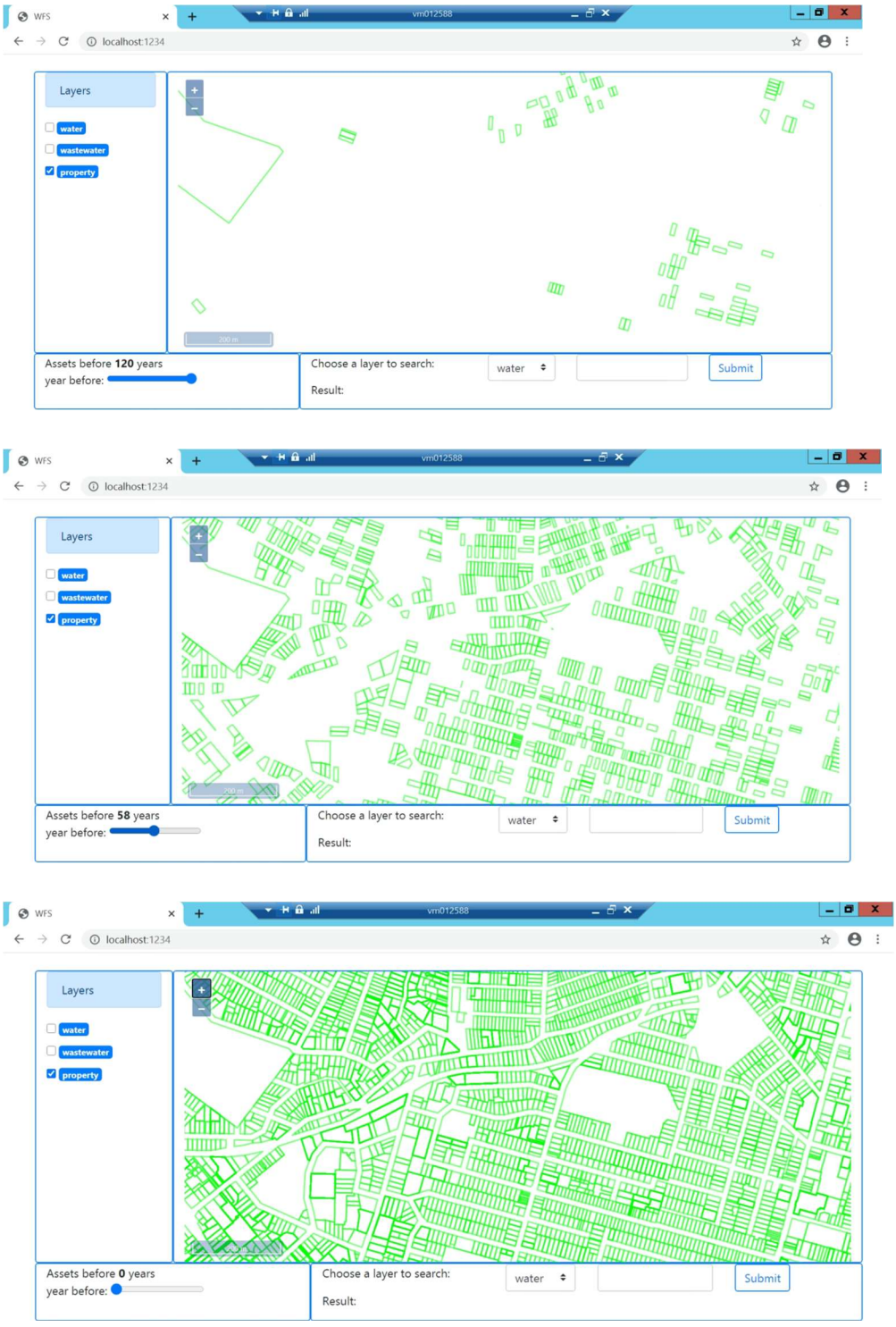


Figure 5. As you can see from the three pictures above, the properties in Auckland has been built up very fast through time. It is believed that this dataset can show the status of urban spread in Auckland well.

2.4 Bus Patronage (Through Web Scraping)

This is slightly different with previous datasets downloaded from available websites. We have used python web scraping tool so as to get all the Auckland bus stop locations from Auckland transport GIS map, https://at.govt.nz/bus-train-ferry/timetables/find-my-stop-or-station-on-a-map/ , which includes its geospatial information (x and y coordinates).

	A	B	C	D	E	F	G	H	I	J	K	L
1	X	Y	OBJECTID	STOPID	STOPCODE	STOPNAM	STOPDESC	LOCATION	STOPLAT	STOPLON	PARENTST	MODE
2	1750193	5915761	2451945	8520	8520	36 Riversdale Rd		Stop	-36.8922	174.6856	11172	Bus
3	1756648	5915264	2451946	8521	8521	847 Mt Eden Rd		Stop	-36.8956	174.7581	11422	Bus
4	1756657	5915257	2451947	8522	8522	742 Mt Eden Rd		Stop	-36.8957	174.7582	11422	Bus
5	1756580	5915022	2451948	8523	8523	909 Mt Eden Rd		Stop	-36.8978	174.7574	11423	Bus
6	1756580	5914985	2451949	8524	8524	822 Mt Eden Rd		Stop	-36.8981	174.7574	11423	Bus
7	1756518	5914827	2451950	8525	8525	931 Mt Eden Rd		Stop	-36.8996	174.7567	11424	Bus
8	1756558	5914790	2451951	8526	8526	848 Mt Eden Rd		Stop	-36.8999	174.7572	11424	Bus
9	1756667	5914645	2451952	8527	8527	953 Mt Eden Rd		Stop	-36.9012	174.7584	11425	Bus
10	1756692	5914632	2451953	8528	8528	866 Mt Eden Rd		Stop	-36.9013	174.7587	11425	Bus
11	1756721	5914216	2451954	8529	8529	Mt Eden Rd opp Kings		Stop	-36.905	174.7591	11772	Bus
12	1756725	5914173	2451955	8530	8530	924 Mt Eden Rd		Stop	-36.9054	174.7592	11427	Bus
13	1756743	5914519	2451956	8531	8531	Opp 880 Mt Eden Rd		Stop	-36.9023	174.7593	11426	Bus
14	1756687	5913888	2451957	8532	8532	Mt Eden Rd near Thre		Stop	-36.908	174.7588	11428	Bus
15	1756658	5913806	2451958	8533	8533	Mt Eden Rd Near Thre		Stop	-36.9088	174.7585	11428	Bus
16	1751503	5915505	2451959	8534	8534	Opp 9 Rosebank Rd		Stop	-36.8943	174.7003	11187	Bus
17	1764826	5916514	2451960	8535	8535	Morrin Rd near U Of A		Stop	-36.883	174.8496	31344	Bus
18	1751125	5915479	2451961	8536	8536	76 Rosebank Rd		Stop	-36.8946	174.6961	11149	Bus
19	1751484	5915510	2451962	8537	8537	11 Rosebank Rd		Stop	-36.8942	174.7001	11187	Bus
20	1760106	5916874	2451963	8538	8538	320 Remuera Rd		Stop	-36.8805	174.7965	31406	Bus
21	1751278	5915482	2451964	8539	8539	Stop C Avondale		Stop	-36.8945	174.6978	11796	Bus
22	1750487	5915797	2451965	8540	8540	158 Rosebank Rd		Stop	-36.8918	174.6889	11795	Bus
23	1758127	5918442	2451966	8541	8541	Khyber Pass Rd Opp N		Stop	-36.8667	174.774	11514	Bus
24	1750381	5915920	2451967	8542	8542	174 Rosebank Rd		Stop	-36.8907	174.6877	11153	Bus
25	1750714	5915633	2451968	8543	8543	129 Rosebank Rd		Stop	-36.8933	174.6914	11150	Bus
26	1750149	5916259	2451969	8544	8544	224 Rosebank Rd		Stop	-36.8877	174.685	11154	Bus
27	1750563	5915722	2451970	8545	8545	147 Rosebank Rd		Stop	-36.8925	174.6897	11795	Bus

Figure 6: Bus Stop from Web Scraping

However, this is insufficient with presenting change over purpose in Auckland for this project. Thus, we have also searched https://at.govt.nz/about-us/reports-publications/at-metro-patronage-report/ that showing the monthly patronage number for all the bus stops in Auckland. In addition to this, we have used Gaussian method to estimate the monthly patronage number for each bus stop in order to get its time series information, and it has been finally published in our web app as a heat map format.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1		X	Y	OBJECTID	STOPID	STOPCODE	STOPNAM	STOPDESC	LOCATION	STOPLAT	STOPLON	PARENTST	MODE	rand	month_0	month_1	month_2	month_3	month_4	month_5	month_6	month_7	month_8
2	0	1750193	5915761	2451945	8520	8520	36 Riversdale Rd		Stop	-36.8922	174.6856	11172	Bus	0.885234	1137.991	1400.806	1243.713	1198.972	1204.109	875.4082	782.2465	1097.956	1487.007
3	1	1756648	5915264	2451946	8521	8521	847 Mt Eden Rd		Stop	-36.8956	174.7581	11422	Bus	0.190525	244.9244	301.4887	267.6785	258.049	259.1546	188.4099	168.3592	236.3077	320.0415
4	2	1756657	5915257	2451947	8522	8522	742 Mt Eden Rd		Stop	-36.8957	174.7582	11422	Bus	0.543425	698.587	859.9231	763.4877	736.0219	739.1754	537.3934	480.2036	674.0103	912.8405
5	3	1756580	5915022	2451948	8523	8523	909 Mt Eden Rd		Stop	-36.8978	174.7574	11423	Bus	0.327702	421.2697	518.5603	460.4067	443.844	445.7457	324.0649	289.5776	406.4491	550.4711
6	4	1756580	5914985	2451949	8524	8524	822 Mt Eden Rd		Stop	-36.8981	174.7574	11423	Bus	0.244076	313.7663	386.2294	342.916	330.5799	331.9963	241.3671	215.6806	302.7277	409.9969
7	5	1756518	5914827	2451950	8525	8525	931 Mt Eden Rd		Stop	-36.8996	174.7567	11424	Bus	0.873752	1123.231	1382.636	1227.582	1183.421	1188.491	864.0537	772.1004	1083.715	1467.72
8	6	1756558	5914790	2451951	8526	8526	848 Mt Eden Rd		Stop	-36.8999	174.7572	11424	Bus	0.797236	1024.867	1261.556	1120.08	1079.786	1084.413	788.3869	704.4861	988.8117	1339.189
9	7	1756667	5914645	2451952	8527	8527	953 Mt Eden Rd		Stop	-36.9012	174.7584	11425	Bus	0.296511	381.1728	469.2032	416.5848	401.5985	403.3192	293.2201	262.0154	367.7629	498.0768
10	8	1756692	5914632	2451953	8528	8528	866 Mt Eden Rd		Stop	-36.9013	174.7587	11425	Bus	0.132881	170.8213	210.2718	186.691	179.975	180.7461	131.4056	117.4213	164.8116	223.2113
11	9	1756721	5914216	2451954	8529	8529	Mt Eden Rd opp Kings		Stop	-36.905	174.7591	11772	Bus	0.058613	75.34897	92.75053	82.34909	79.38666	79.72679	57.96277	51.79432	72.69814	98.45815
12	10	1756725	5914173	2451955	8530	8530	924 Mt Eden Rd		Stop	-36.9054	174.7592	11427	Bus	0.960011	1234.118	1519.133	1348.771	1300.25	1305.821	949.3547	848.3236	1190.701	1612.616
13	11	1756743	5914519	2451956	8531	8531	Opp 880 Mt Eden Rd		Stop	-36.9023	174.7593	11426	Bus	0.753791	969.0174	1192.808	1059.042	1020.944	1025.318	745.424	666.0954	934.9267	1266.211
14	12	1756687	5913888	2451957	8532	8532	Mt Eden Rd near Thre		Stop	-36.908	174.7588	11428	Bus	0.212361	272.9957	336.0431	298.3578	287.6246	288.857	210.0041	187.6553	263.3916	356.7223
15	13	1756658	5913806	2451958	8533	8533	Mt Eden Rd Near Thre		Stop	-36.9088	174.7585	11428	Bus	0.361194	464.3242	571.5581	507.4612	489.2057	491.3017	357.185	319.173	447.989	606.7303
16	14	1751503	5915505	2451959	8534	8534	Opp 9 Rosebank Rd		Stop	-36.8943	174.7003	11187	Bus	0.818313	1051.963	1294.909	1149.693	1108.334	1113.082	809.2303	723.1113	1014.954	1374.595
17	15	1764826	5916514	2451960	8535	8535	Morrin Rd near U Of A		Stop	-36.883	174.8496	31344	Bus	0.673602	865.9323	1065.916	946.3797	912.3345	916.2434	666.125	595.2354	835.4681	1131.51
18	16	1751125	5915479	2451961	8536	8536	76 Rosebank Rd		Stop	-36.8946	174.6961	11149	Bus	0.27648	355.4225	437.506	388.4422	374.4684	376.0728	273.4115	244.3148	342.9185	464.429
19	17	1751484	5915510	2451962	8537	8537	11 Rosebank Rd		Stop	-36.8942	174.7001	11187	Bus	0.419142	538.8176	663.2555	588.8752	567.6909	570.1232	414.4895	370.3792	519.8616	704.0705
20	18	1760106	5916874	2451963	8538	8538	320 Remuera Rd		Stop	-36.8805	174.7965	31406	Bus	0.232014	298.2602	367.1422	325.9694	314.2429	315.5893	229.4389	205.0218	287.7672	389.7352
21	19	1751278	5915482	2451964	8539	8539	Stop C Avondale		Stop	-36.8945	174.6978	11796	Bus	0.847104	1088.973	1340.468	1190.142	1147.328	1152.243	837.7011	748.5522	1050.662	1422.956
22	20	1750487	5915797	2451965	8540	8540	158 Rosebank Rd		Stop	-36.8918	174.6889	11795	Bus	0.997519	1282.336	1578.486	1401.468	1351.052	1356.84	986.4466	881.4682	1237.222	1675.622
23	21	1758127	5918442	2451966	8541	8541	Khyber Pass Rd Opp N		Stop	-36.8667	174.774	11514	Bus	0.859752	1105.233	1360.482	1207.912	1164.458	1169.448	850.2089	759.7289	1066.35	1444.203
24	22	1750381	5915920	2451967	8542	8542	174 Rosebank Rd		Stop	-36.8907	174.6877	11153	Bus	0.097278	125.053	153.9335	136.6707	131.7541	132.3186	96.19796	85.96049	120.6535	163.4061
25	23	1750714	5915633	2451968	8543	8543	129 Rosebank Rd		Stop	-36.8933	174.6914	11150	Bus	0.722171	928.3685	1142.772	1014.616	978.1165	982.3072	714.1545	638.1536	895.7078	1213.095
26	24	1750149	5916259	2451969	8544	8544	224 Rosebank Rd		Stop	-36.8877	174.685	11154	Bus	0.041593	53.46829	65.81659	58.43564	56.33347	56.57483	41.13089	36.75371	51.58724	69.86677
27	25	1750563	5915722	2451970	8545	8545	147 Rosebank Rd		Stop	-36.8925	174.6897	11795	Bus	0.577058	741.8222	913.1432	810.7394	781.5738	784.9225	570.6524	509.9231	715.7243	969.3356
28	26	1750400	5915910	2451971	8547	8547	185 Rosebank Rd		Stop	-36.8908	174.6879	11153	Bus	0.494442	635.6173	782.4107	694.6678	669.6778	672.547	488.9534	436.9186	613.2558	830.5582

Figure 7: Bus Stop with Monthly Patronage data

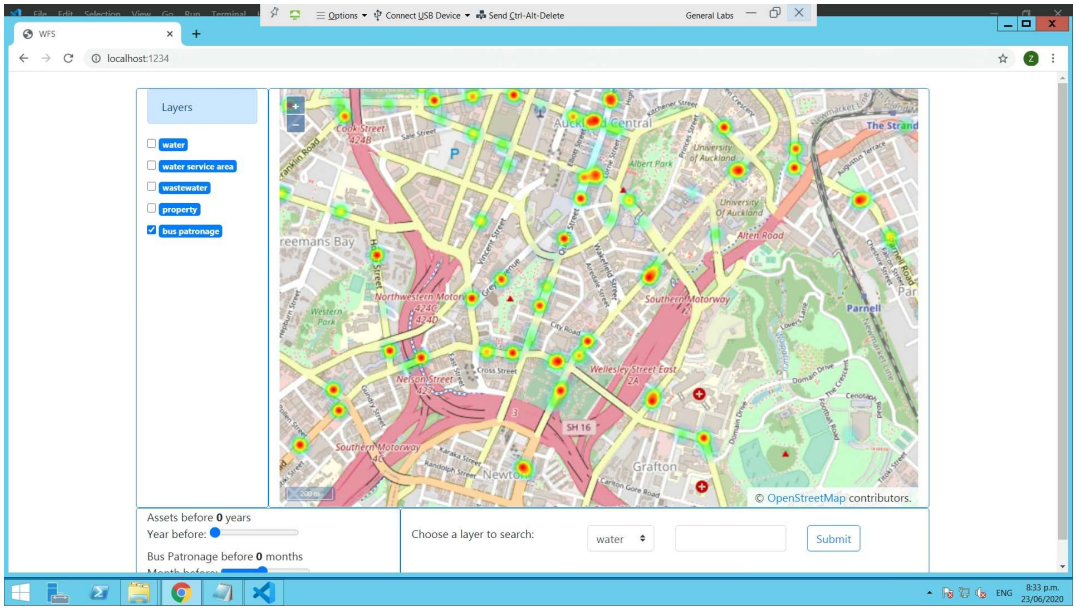


Figure 7: Web App Bus Patronage Heat Map

2.5 Water Service Area (Through QGIS network Analysis)

Since we have already presented water pipe network in previous section, there might be some requirements that people should be interested in the water service area/coverage for a specific location of water pump station.

Therefore, we have used QGIS network analysis tool to implement this approach. We select a point as the location of water pump station and then run the service area tool with certain parameters to estimate its service area coverage. Lastly, it has been published in our web app as a demonstration only as per images below.

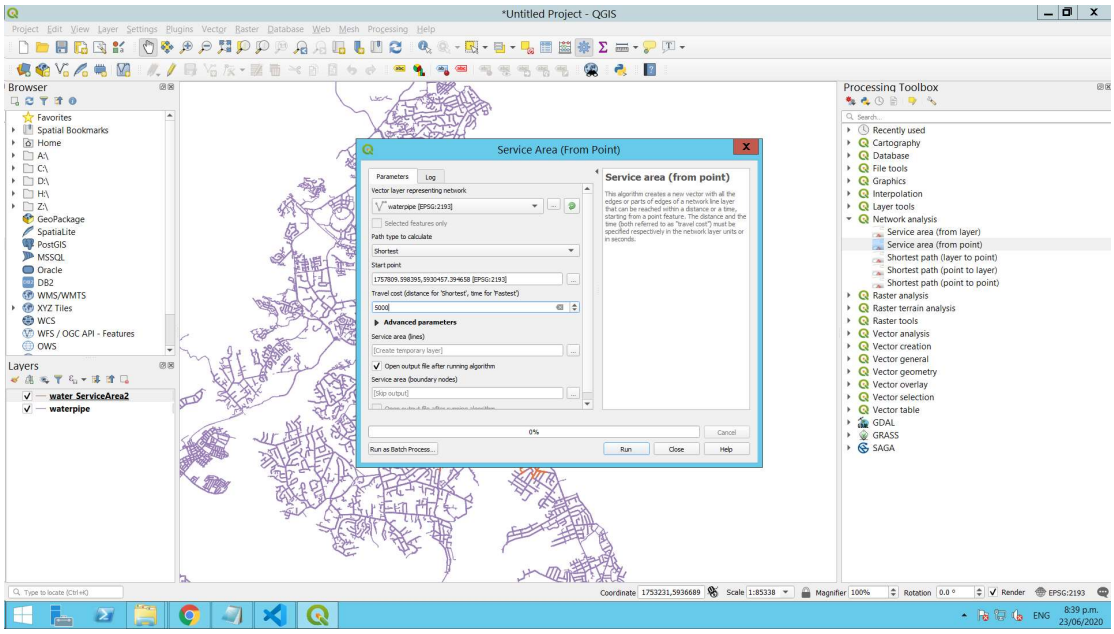


Figure 8: QGIS set up

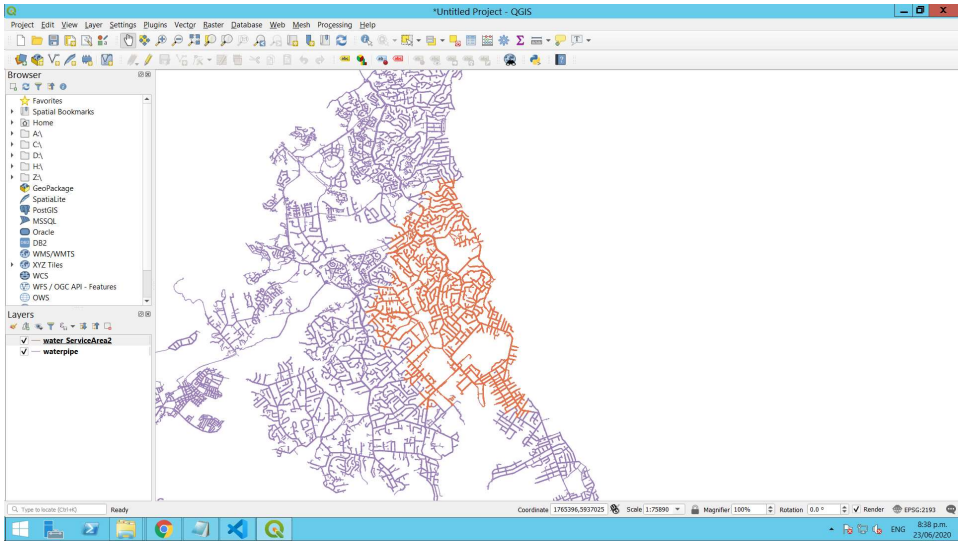


Figure 9: QGIS Network Area Analysis Result

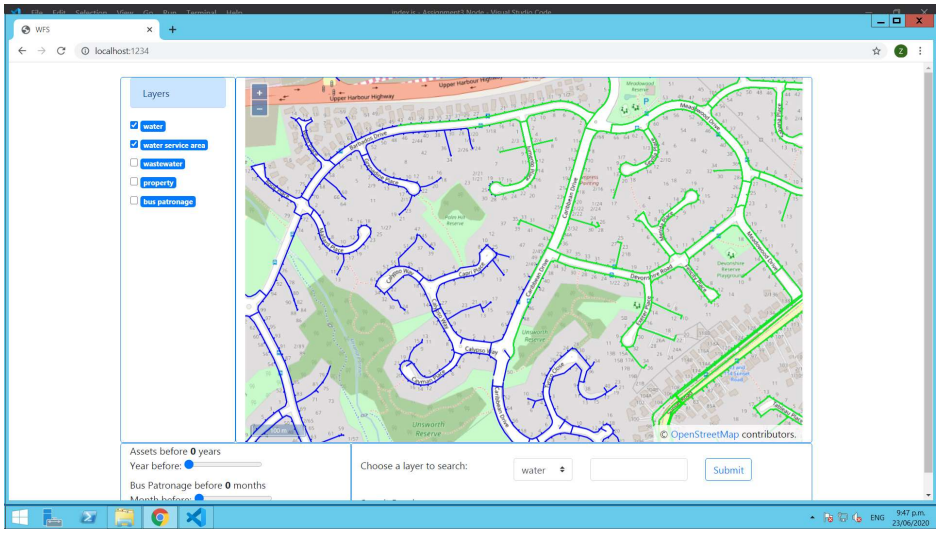


Figure 10: QGIS Network Area in web app for Water Service Area

2.6 Multiple formats for dataset presentation

In our assignment, we have tried a variety of different formats of presentation for datasets. The best format of showing the dataset depends on the nature of the datasets we have. Water pipes and wastewater pipes are classified as tend to be lines in the map, as their covered area travelled straight from one place to another. But bus patronage is different, bus patronage shows the amount of people gather and use buses, so it is considered as a steady shape on the map. We decided to use heat map to show bus patronage, in order to have the best performance of it. Last but not least, property dataset is tending to be polygon shapes in the map, since most of them are buildings. We have also tried network analysis. It is believed that the users of the web application should have a good user experience.

3.0 Front-end and back-end technologies

In our project, we need to make a front-end page to interact the geographic data in the database. In this web app, we need to realize the display of the layer, for the selection of different years and the specific query of the fixed number of streets. In order to get these functions, we intend to use HTML, JavaScript and Node.js technology to interact client and server for this web app. By the way, we put front end code for this project in **D:\Assignment3 Node** folder what in the D disk of Virtual machine.

The front end code file structure is as below, **index.html** shows the html page for web app, **index.js** is JavaScript code by using **Openlayers** tools, **package.json** and **package-lock.json** are third party libraries when building the front end projects.

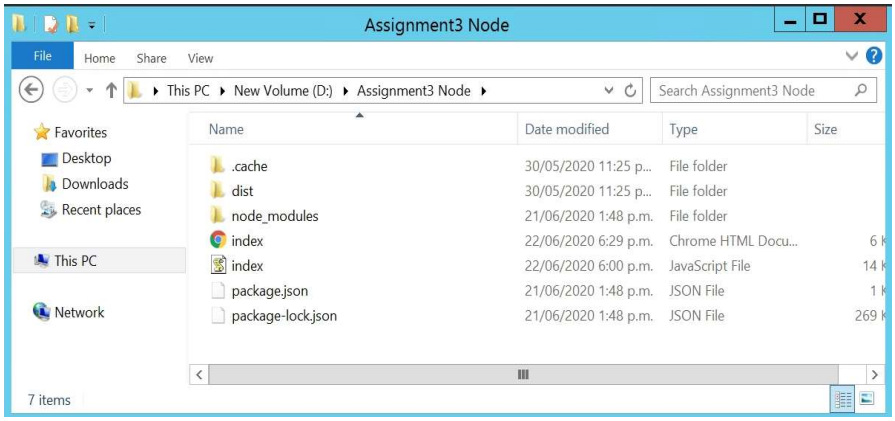


Figure 11. The path of detail page and main page.

Before starting work, we must first configure the running environment of NodeJS.

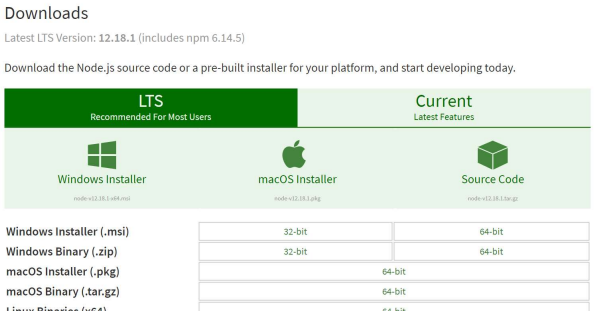


Figure 12. It's the download website. We can download a suitable version for NodeJS.

```
PATH=C:\oraclexe\app\oracle\product\10.2.0\server\bin;C:\Windows\system32;
C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;
c:\python32\python;C:\MinGW\bin;C:\Program Files\GTK2-Runtime\lib;
C:\Program Files\MySQL\MySQL Server 5.5\bin;C:\Program Files\nodejs\;
C:\Users\rg\AppData\Roaming\npm
```

Figure 13. We can test it and confirm the NodeJS has installed successfully. After following the steps of the installation package, we need to enter "path" in the command line of the CMD window.

Figure 14. Confirm the NodeJS version. We can type the code and check out the version about NodeJS.

3.1 Front End Code Interpretations

In this section, we will take some screen shots to show key structure and functions for our frond end code. All details and implementations could be referred to its source code files.

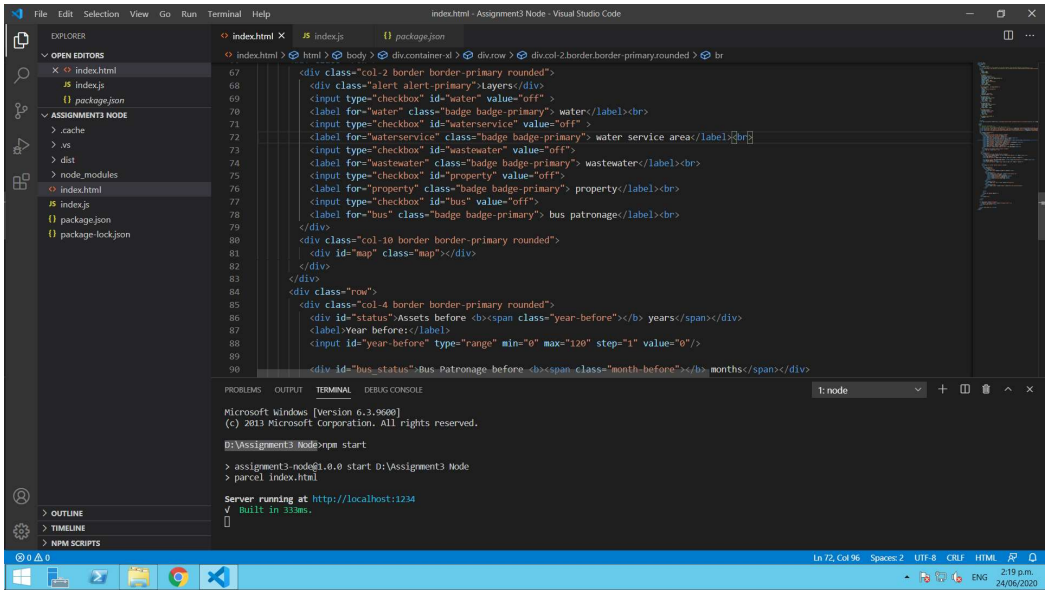


Figure 15. Main HTML page shows the web map layout in index.html

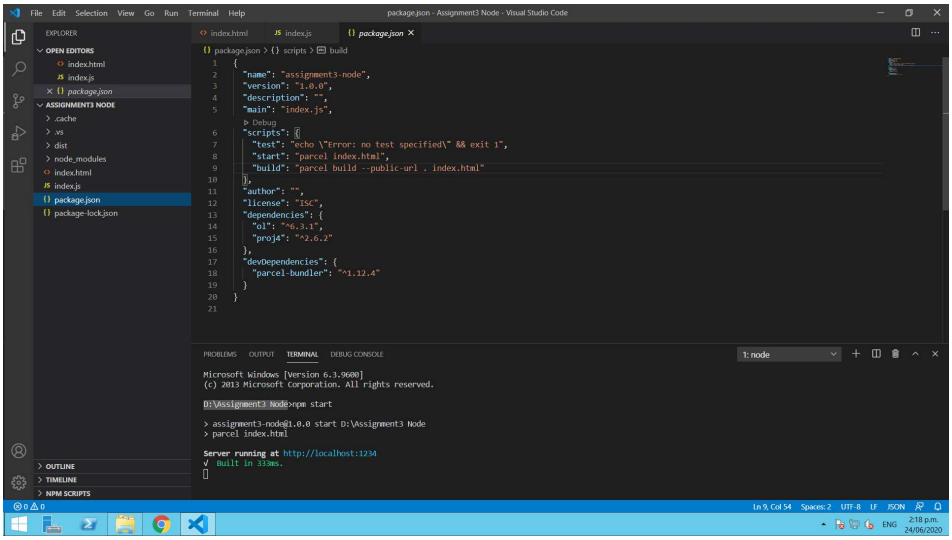


Figure 16. Dependencies configurations for package.json file

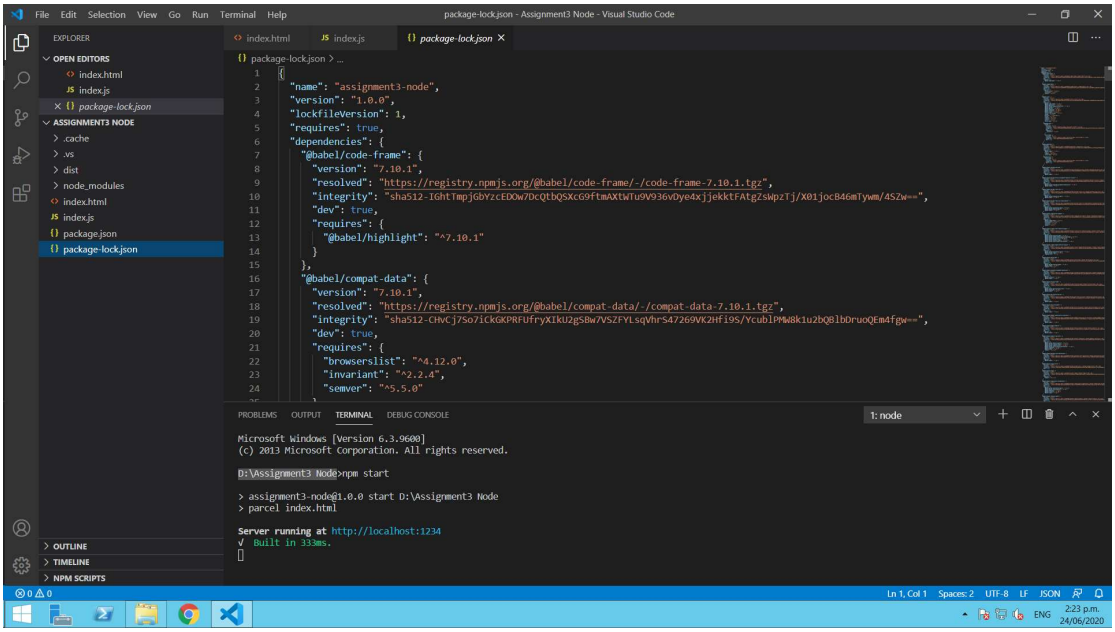


Figure 17. Third party packages in package-lock.json file

There is a question, when we view our website, how does map and layer show on page? The following figures will show the source code in index.js.

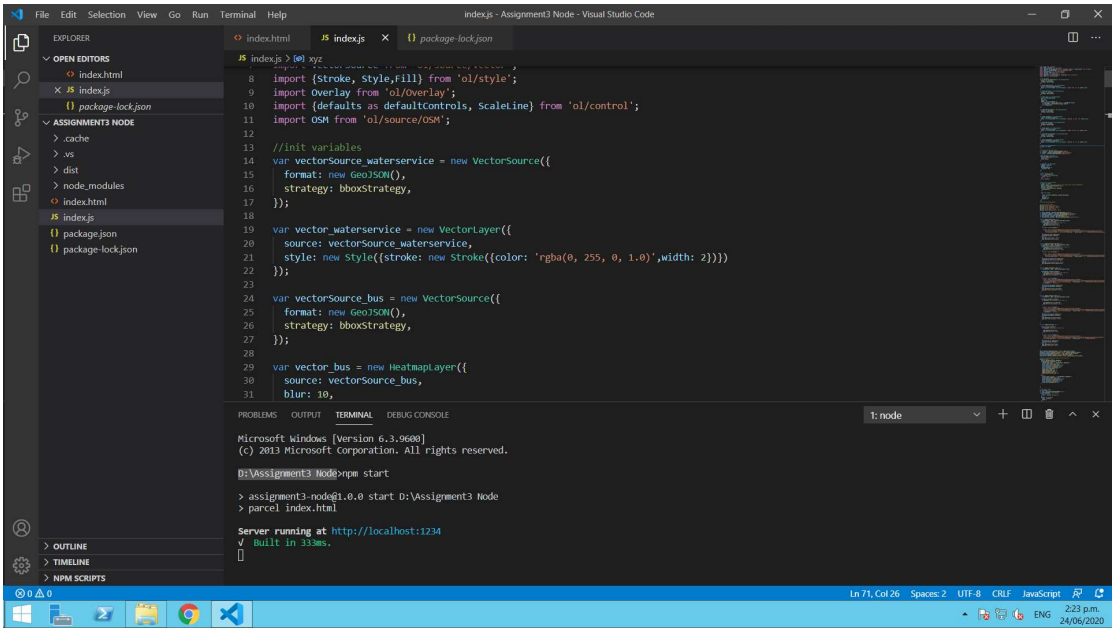


Figure 18. Initialize the interface and some objects.

Line 13 to 101 of the code, we start initializing map objects for this app.

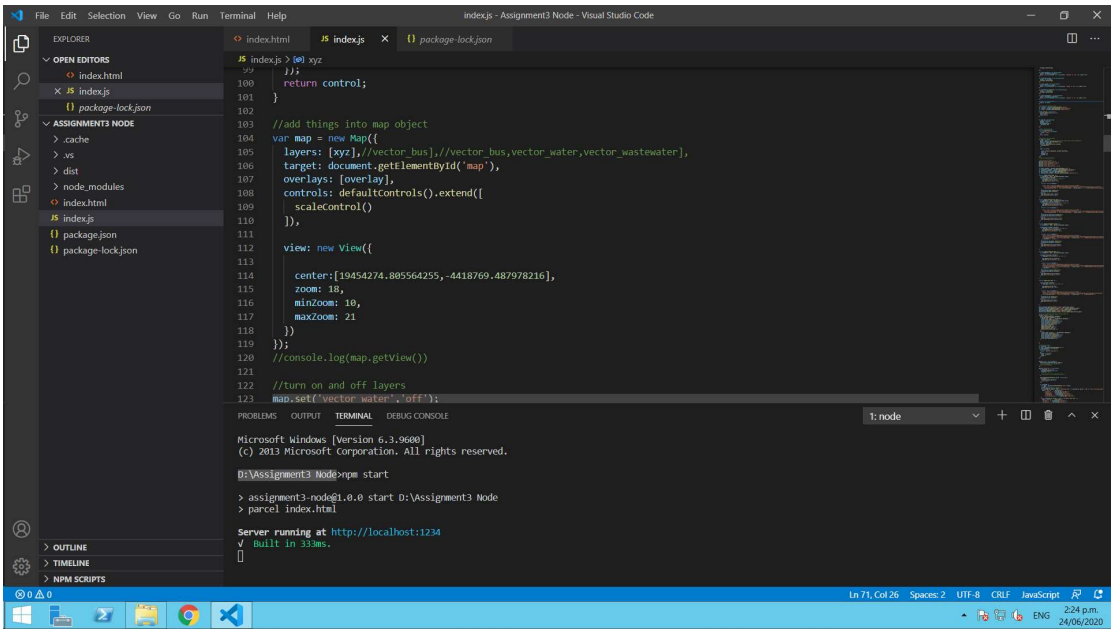


Figure 19. Map object set up.

Line 104 to 119 of the code, we have set up the map object, like setting zoom levels or center position.

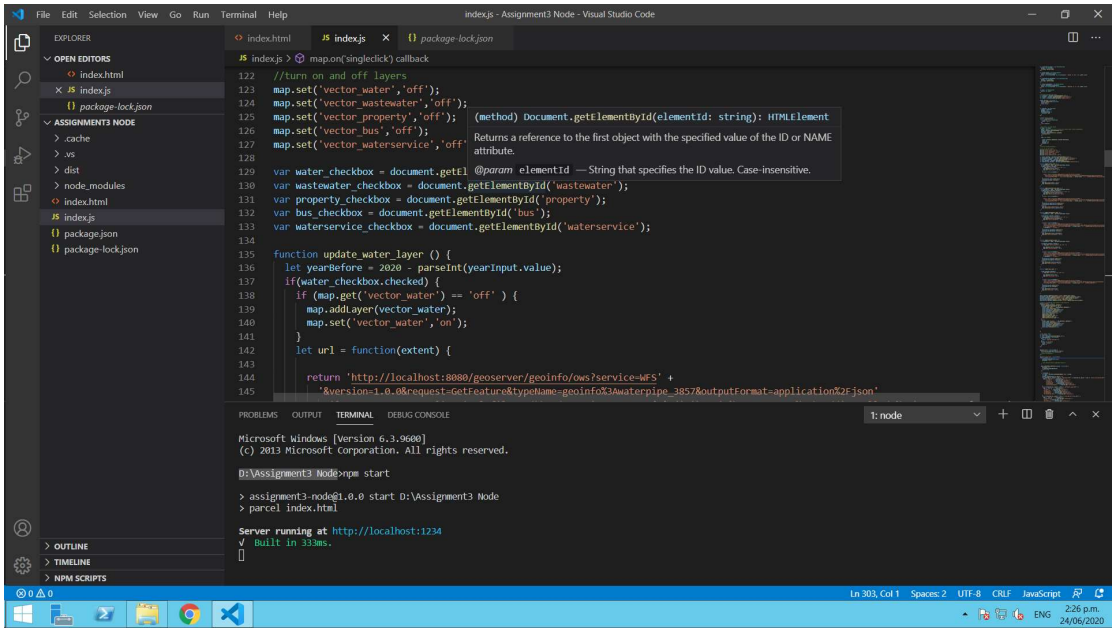


Figure 20. Layer Switch logic.

Line 122 to 147 are implementation with layer switch, on and off logic

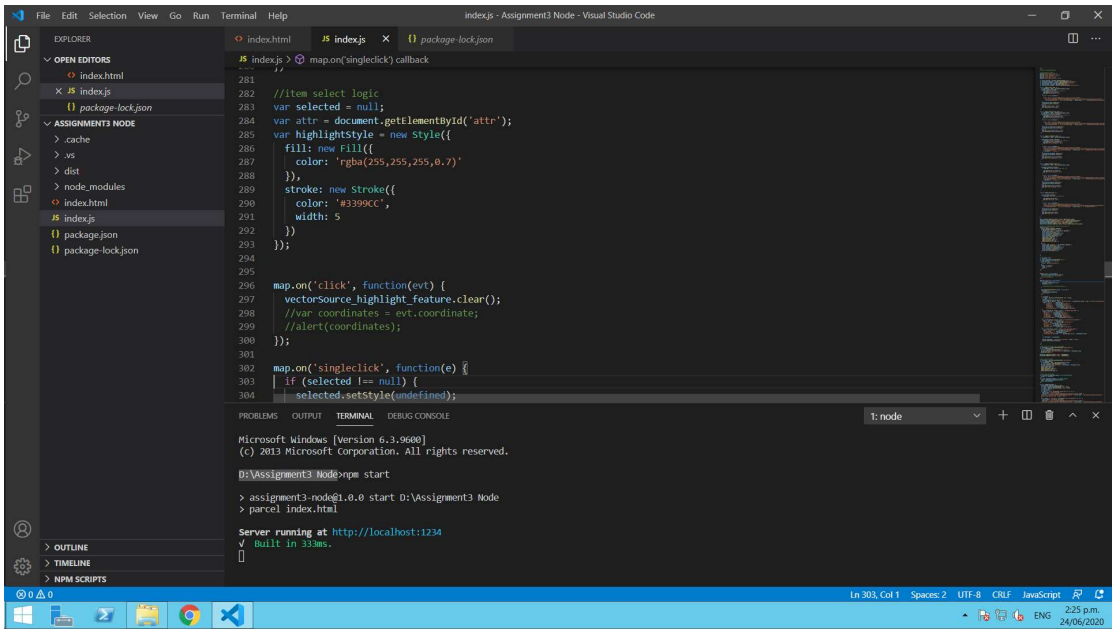


Figure 21. Item selection logic.

On lines 282 to 357, are the item selection functions for select items on the map.

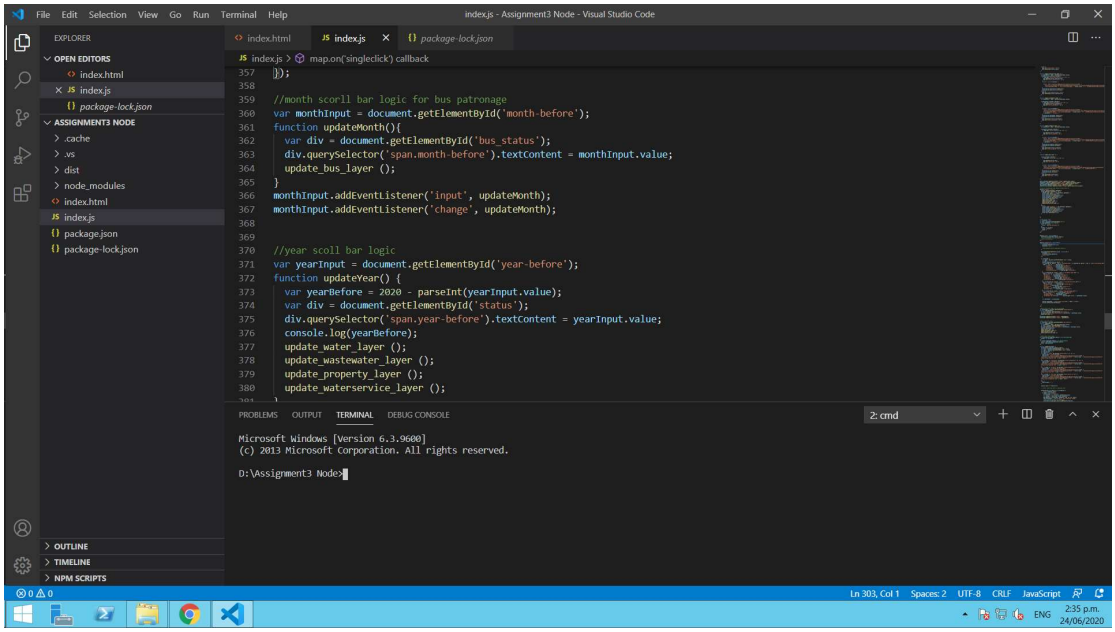


Figure 22. The year and month scroll bar logic.

In our website, we would like to show the different information or layer according to different years. So, we create a function what can select the different year and month. In order to realize the matching function of the user clicking the layer, we create a click event. In it we have implemented a click function and realized it by calling a function. The line is from 359 to 381.

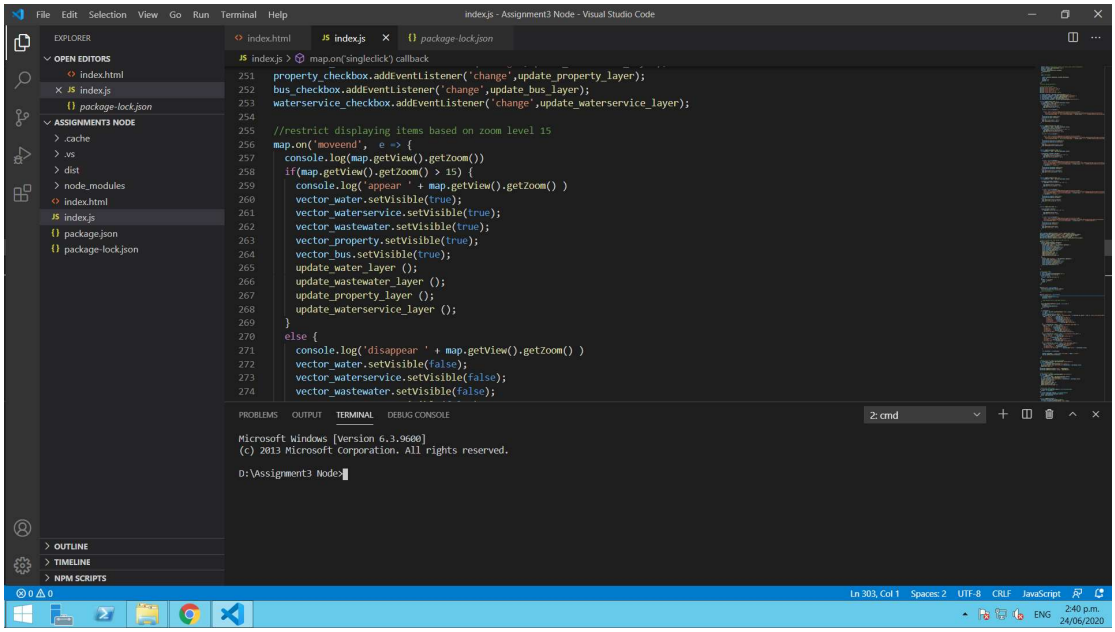


Figure 23 Restrict displaying items based on zoom level 15

We also don't want all the items to be loaded in the map at the same time, line 255-277 has implemented this in order to restrict items to be displayed above certain zoom level on the map (15 at the moment)

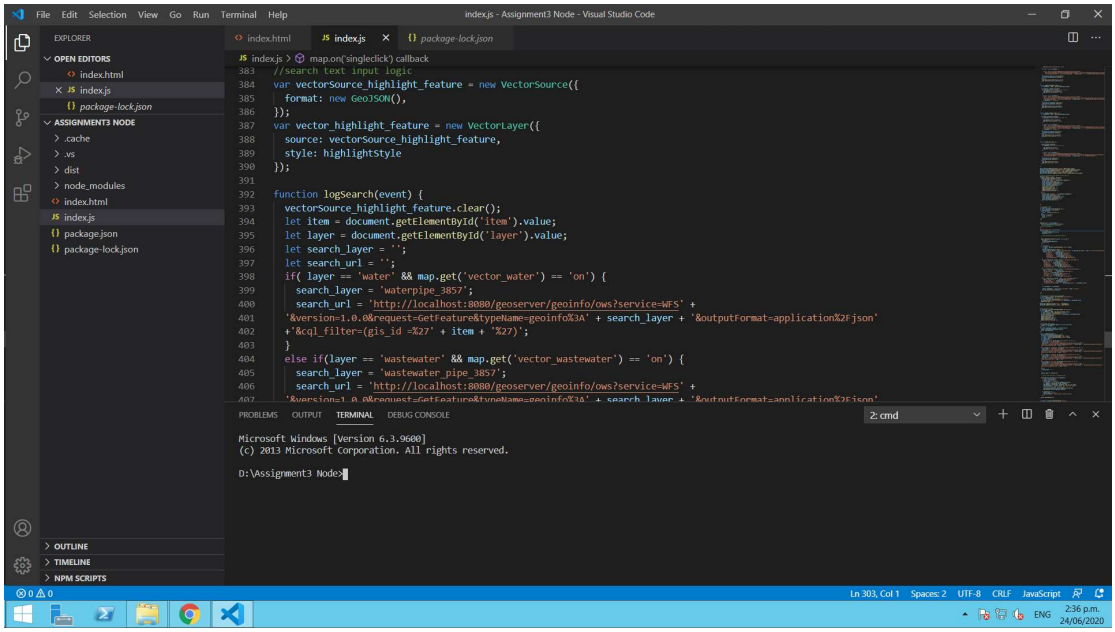


Figure 24 Searching function logic.

Lastly, we'd like to create searching function. When we type the particular ID into the map, the corresponding item detail will show on and zoomed. It is from line 383 to 448 in the source code.

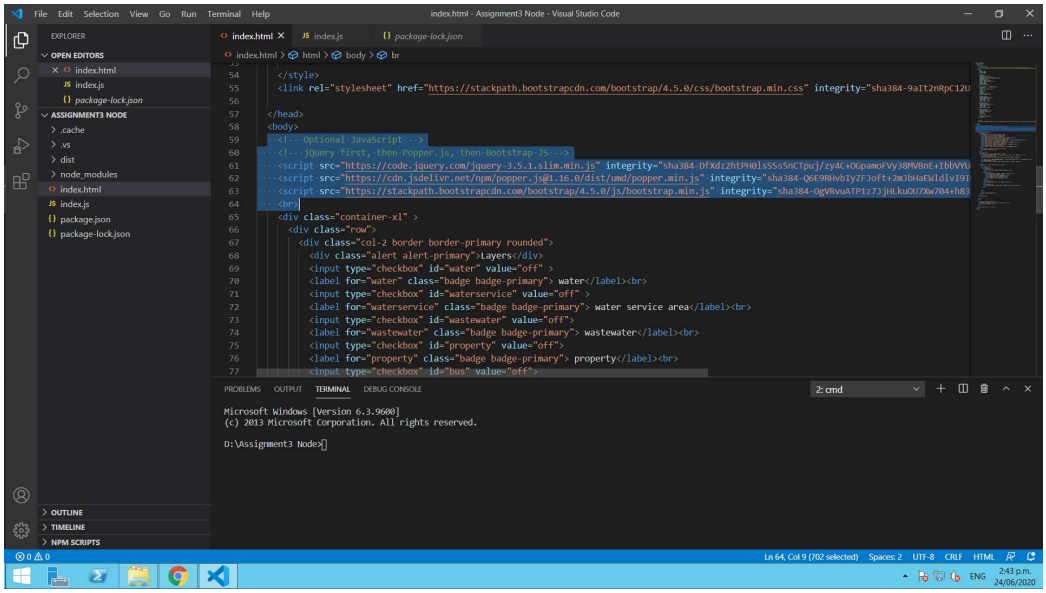


Figure 25 Quick CSS styling with BoostStrap4

We have also use BootStrap 4 add-on to quick style our web app.

3.2 The back-end Server

A Web server is a type of computer program deployed on the Internet. When the client, our browser, connects to the server and requests a file, the server accepts and processes the request, and returns the file that the user needs to the browser, and tells the browser how to view the file. The server uses HTTP to exchange information with the client's browser. To put it simply, a web server specializes in handling HTTP requests, and it can parse the HTTP protocol. But the application server provides business logic for the application through many protocols.

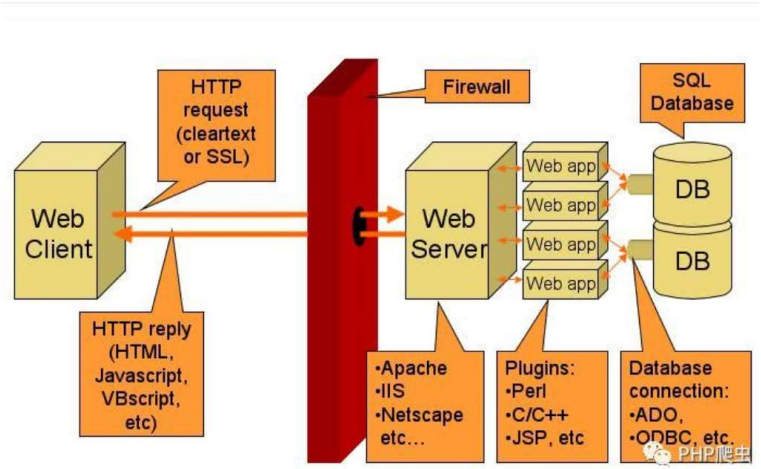


Figure 26. A working diagram of a web server that we found on the Internet.

The working principle of the Web server can be divided into four steps, the connection process, the request process, the response process, and finally closing the connection. The connection process is a connection established between the web server and its corresponding browser. If the user can find and open a socket virtual file, it proves that the connection is successful in this step. The request process is that the browser makes various requests to the corresponding server through the socket file. The response process also uses the HTTP protocol to pass the processed tasks to the web server for processing. After the request is processed, the HTTP protocol is used to transmit the processed result back to the browser. The final closed connection is that after the request is processed and the result is obtained, the browser and the server are disconnected. The web server can support multiple threads, multiple processes, and multi-thread and multi-process hybrid technologies through the above-mentioned processing procedures.

Next is the steps and detailed operations of our deployment of web application servers and web servers.

First, because Tomcat is programmed and controlled using the Java language, our first step is to deploy the Java runtime environment, which is to configure the JDK environment. Follow the website and download a jdk file.

https://adoptopenjdk.net/?variant=openjdk11&jvmVariant=hotspot

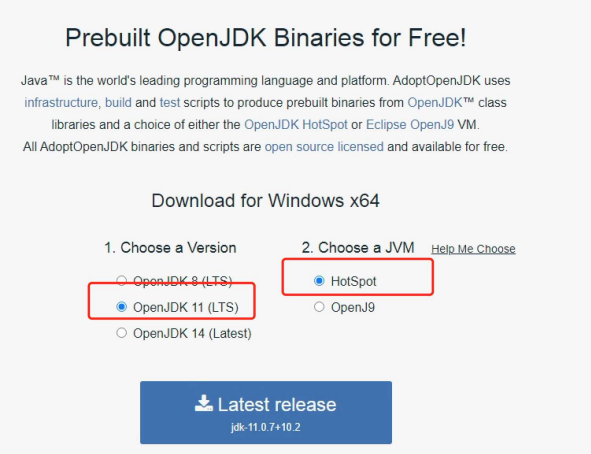


Figure 27. It's a download website.

After downloading the jdk file, we set up the environment on our virtual machine. Open "This PC - Properties - advanced system settings

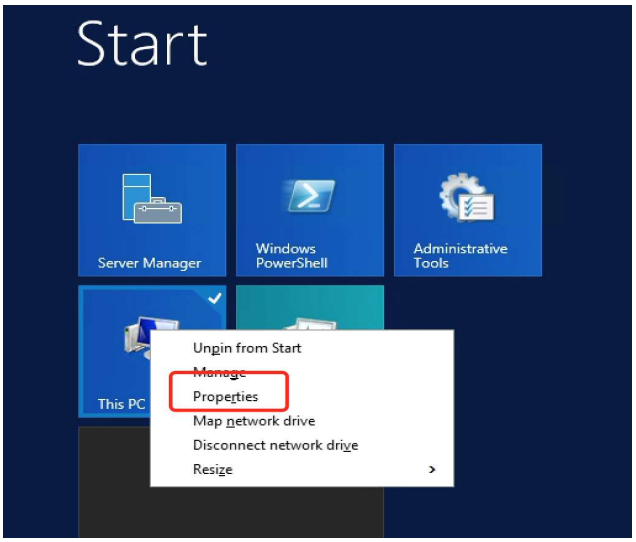


Figure 28. Click this PC and continue to click properties.

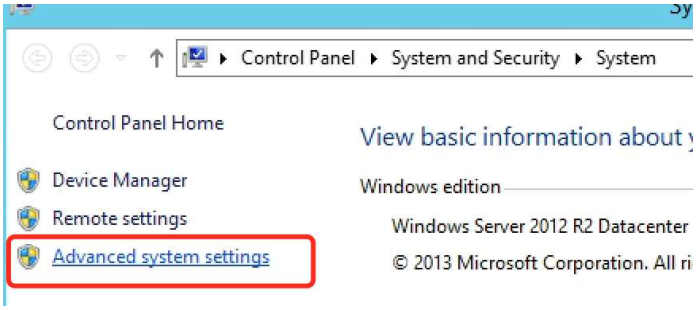


Figure 29. Set up advanced system settings.

Continue to “Advanced Environment Variable”

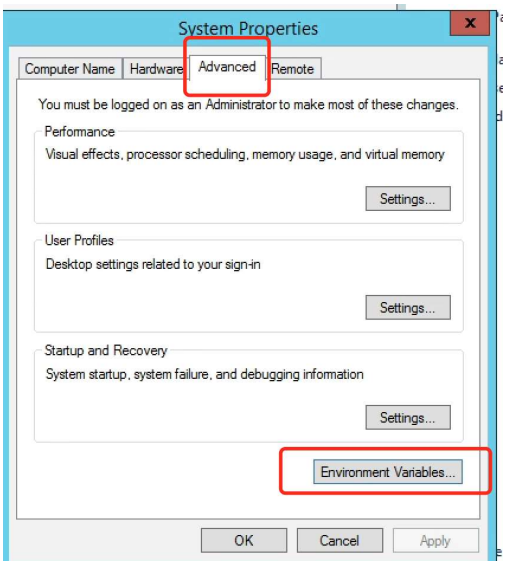


Figure 30. Continue to “Advanced Environment Variable”.

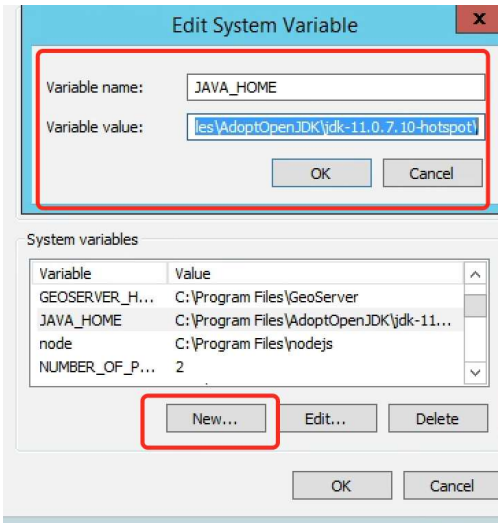


Figure 31. Build up the particular environment.

Add a “System Variables” - “JAVA_HOME”, the value is the folder (include bin folder) where you install the JDK if you use default installation setting, the following will be the right one, otherwise, find your own installation folder. **C:\Program Files\AdoptOpenJDK\jdk-11.0.7.10-hotspot** .After set up Java environment successfully, we install Tomcat. Download tomcat 64-bit windows version.

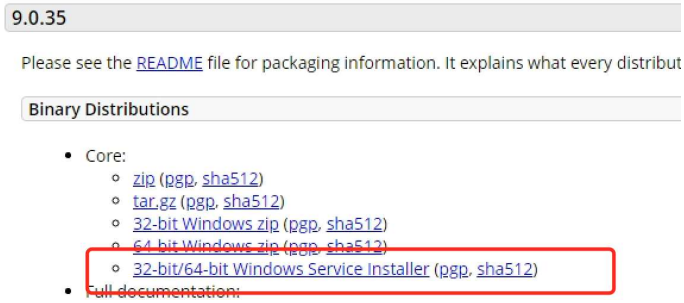


Figure 32. It’s a download website.

After downloading successfully, we set up some basic options.

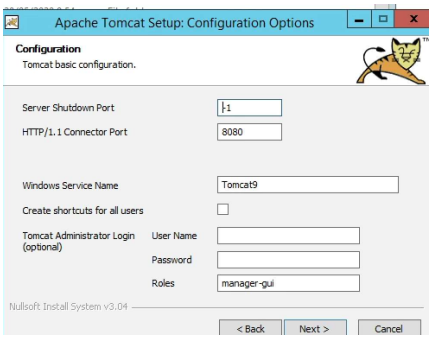


Figure 33. Install interface about tomcat.

When you come across this page, you should set your username and password, and remember two things. First, your username and password. Second, the http connector port, default is 8080, and the service name, default is Tomcat9.

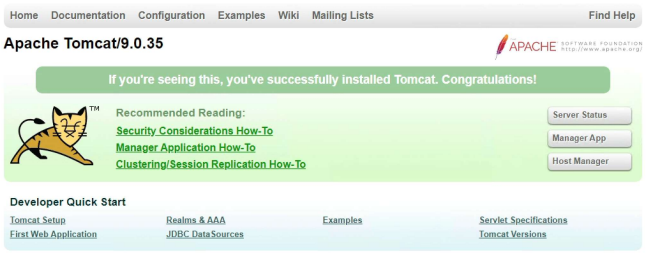


Figure 36. When we see this page, it means we install the tomcat successfully. If we install tomcat in a correct way, we can see this page after opening http://localhost:8080

Installing Geoserver ,choose a suitable version of Geoserver.

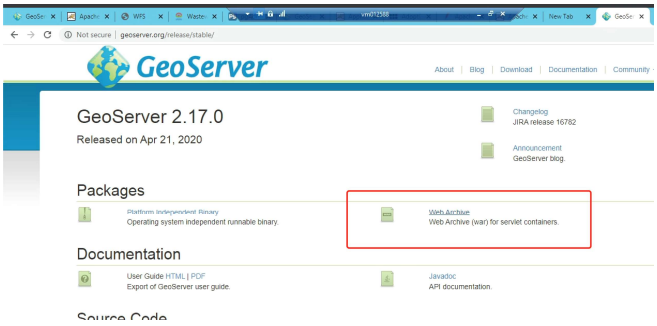


Figure 34. It's the web archive page.

In this part, we have tried to install the binary package and use jetty to edit the Geoserver but fail. And then, we change another way, download tomcat and the web Archive to control the Geoserver.

Put our download file into the tomcat's webapps folder.

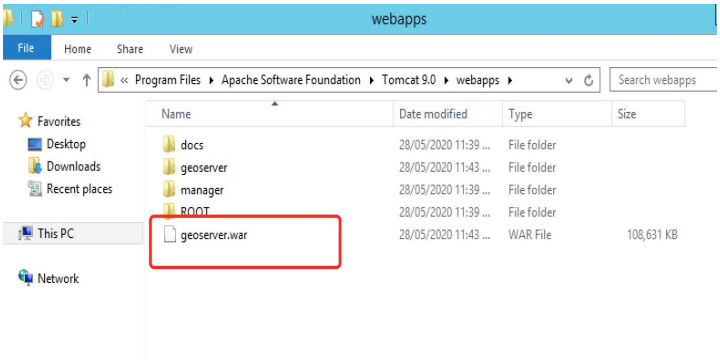


Figure 35. We put this file following the same path.

After installing them all, open the Apache tomcat and confirm it can work or not.

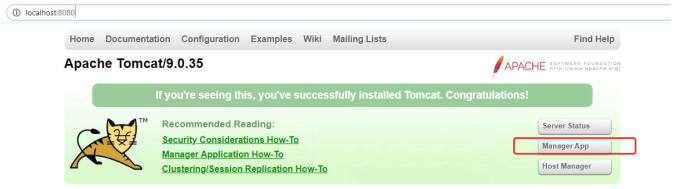


Figure 36. We should use our username and password to login into "Manager App".

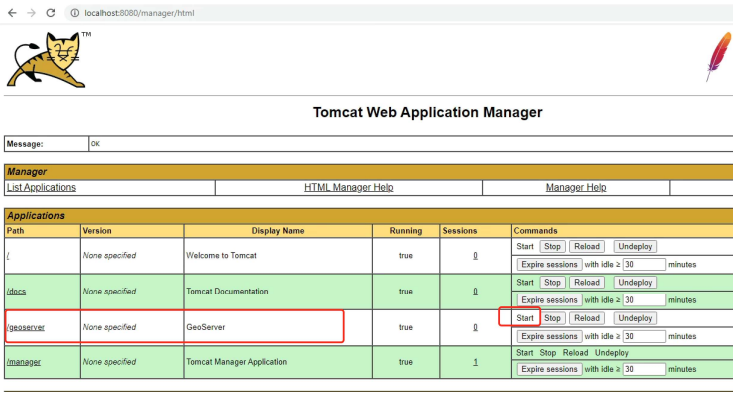


Figure 37. After seeing Geoserver are in the list, we can click "start" button.

There is a link to follow and we can explore the settings.
https://docs.geoserver.org/latest/en/user/gettingstarted/index.html

Publish our layer. There are two different method to add into our layers.The first method is adding the csv file. When we download datasets, we always get some csv files with number. We may need Postgis (database) to convert the csv or other geopackages into tables and import to Geoserver.

The second method is adding the shapefile.

a. Create a workplace.
b. Create a store.

Figure 38. We should create a new contain and add some information into it.

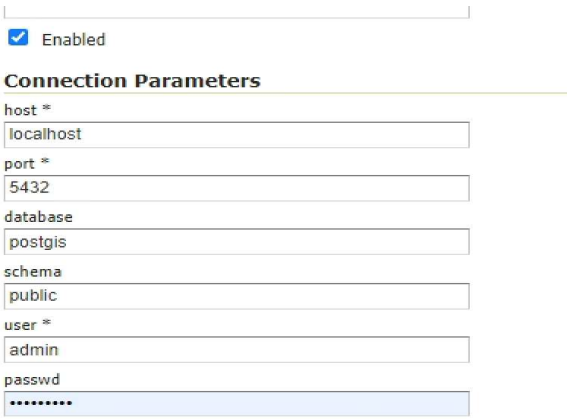


Figure 39. The screenshot about setting. The information of database should be match for my Postgis database.

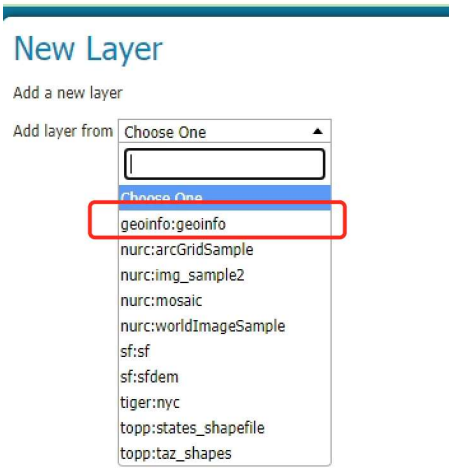


Figure 40. We should select the suitable option, so we choose geoinfo.

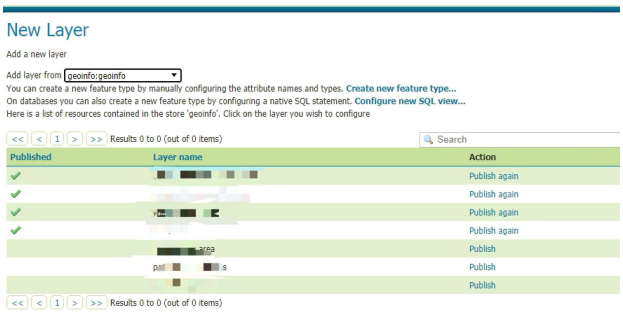


Figure 41. The list should be show on screen as long as we connect to the database.

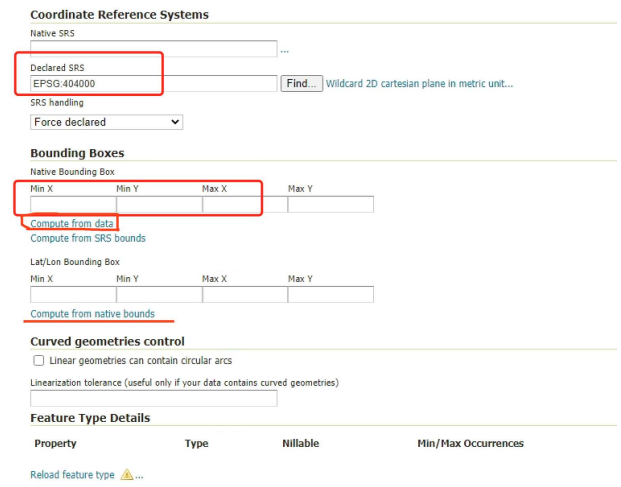


Figure 42. We set up the SRS and add some particular position into layers. The Boundary can be computing automatically, if not, you may need to figure out the coordinators.

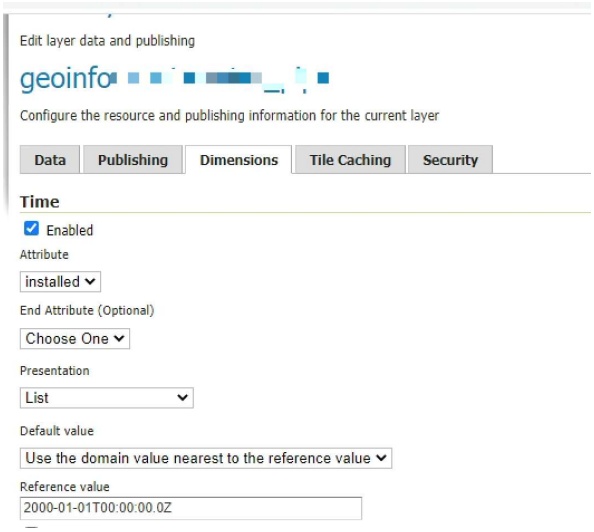


Figure 43. Set up the layer options. If the datasets have the third or fourth dimensions, like time or elevation, we can add dimensions. Since our group's project needs time as the third dimension, here we "Enable" time, and set the right attribute, presentation and default value.

Figure 44. We can add all of them as much as we like.

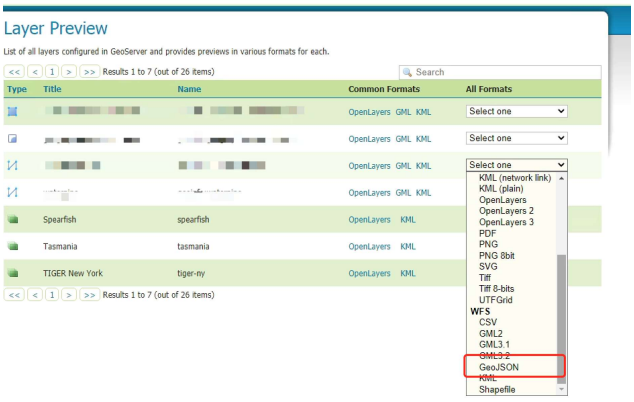


Figure 45. We choose “open layer” to see the web map. There are many options about data format. Geoserver support a bunch of choices, GeoJSON is a very useful format that JavaScript can handle.

There is a screenshot about information for login.

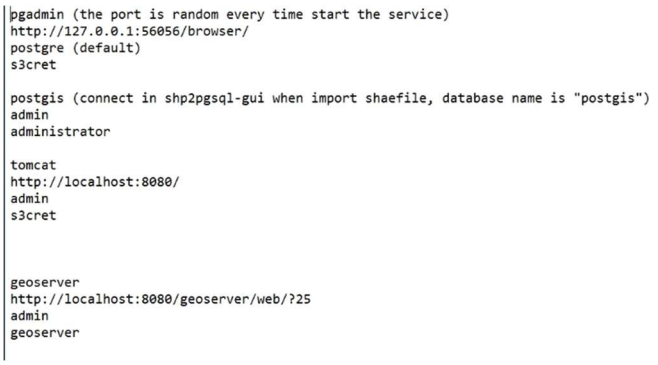


Figure 46. It’s the detail information for login. There are all details like usernames and passwords.

In accordance with all the steps above, we have set up the following 5 datasets APIs for front end code to play with:

Water:
http://localhost:8080/geoserver/geoinfo/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=geoinfo%3Awaterpipe_3857&maxFeatures=50&outputFormat=application%2Fjson

Wastewater:
http://localhost:8080/geoserver/geoinfo/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=geoinfo%3Awastewater_pipe_3857&maxFeatures=50&outputFormat=application%2Fjson

Property titles:

http://localhost:8080/geoserver/geoinfo/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=geoinfo%3Anz-property titles_3857&maxFeatures=50&outputFormat=application%2Fjson

Bus Patronage:

http://localhost:8080/geoserver/geoinfo/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=geoinfo%3Abus_patronage_3857&maxFeatures=50&outputFormat=application%2Fjson

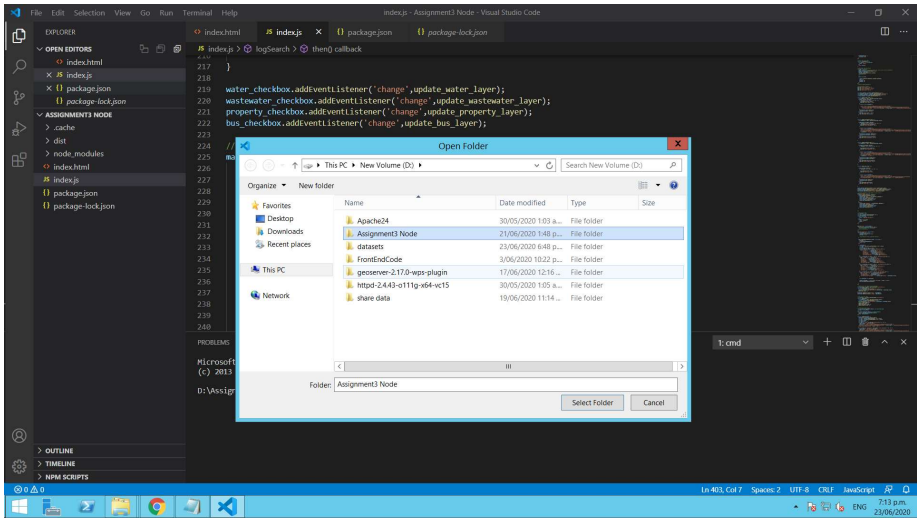
Water Service Area:

http://localhost:8080/geoserver/geoinfo/ows?service=WFS&version=1.0.0&request=GetFeature&typeName=geoinfo%3Awater_servicearea_3857&maxFeatures=50&outputFormat=application%2Fjson

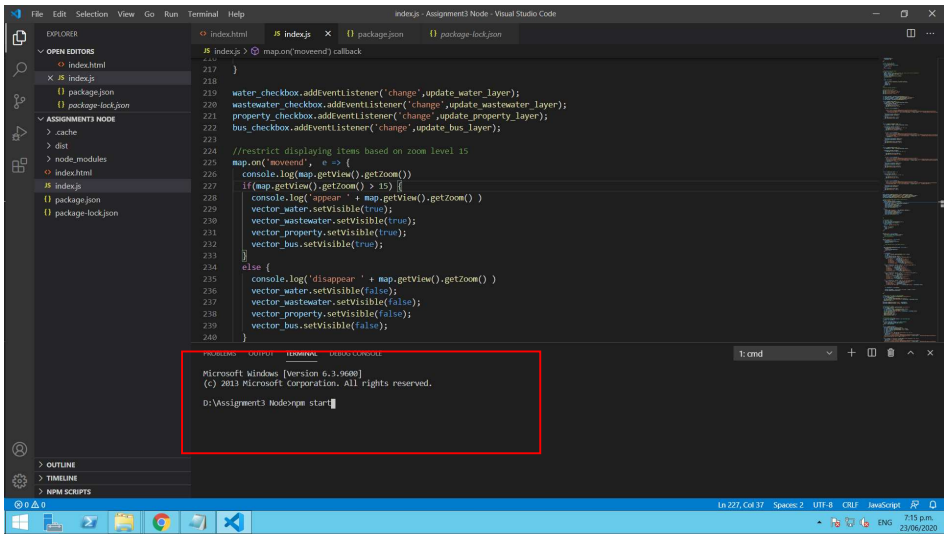
3.3 Start Web App:

Once we have set up all of our front and back end tools and codes. We can start our app in the virtual machine that being allocated to us.

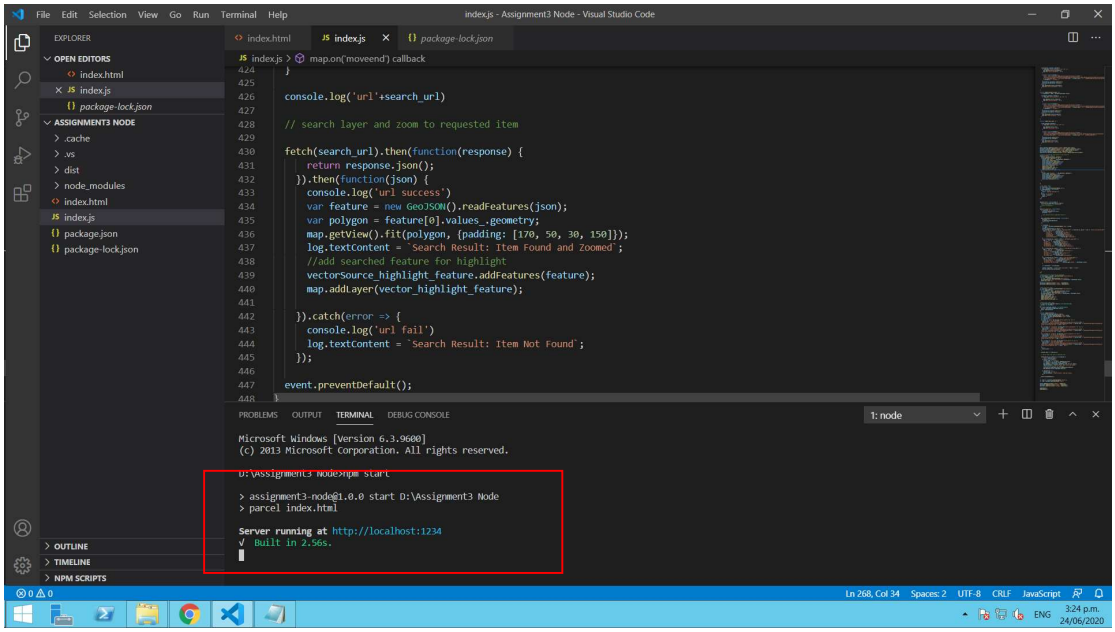
The easiest way is that, open VS code first, open **D:\Assignment3 Node** folder.



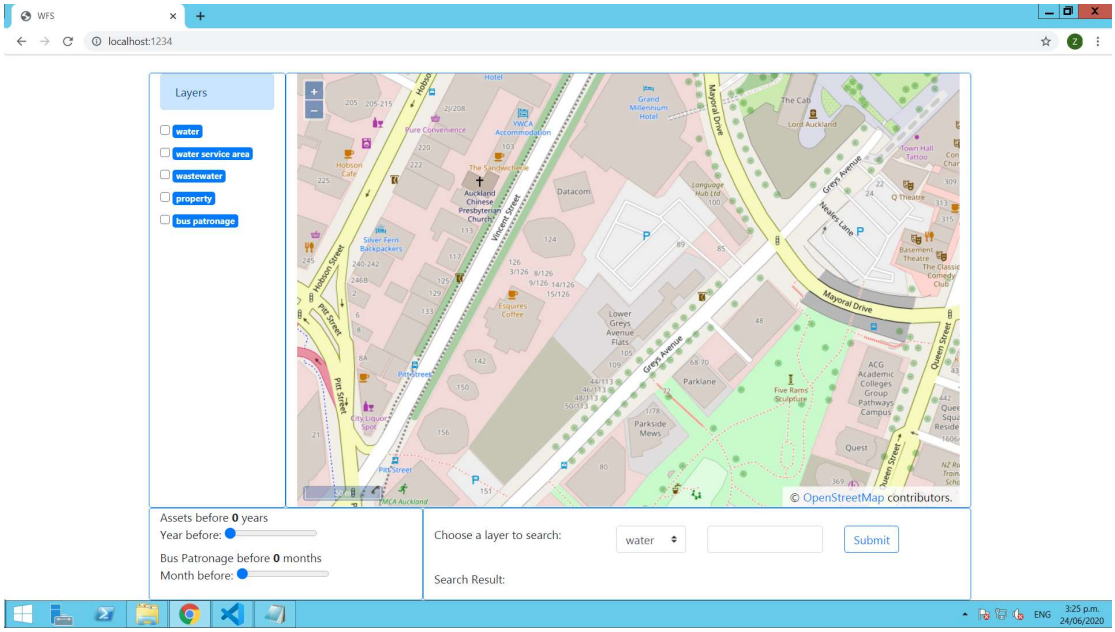
Then, type **npm start** command line at terminal panel under **D:\Assignment3 Node** path



Lastly, wait system build the package until image below, it shows that server is running at **http://localhost :1234**



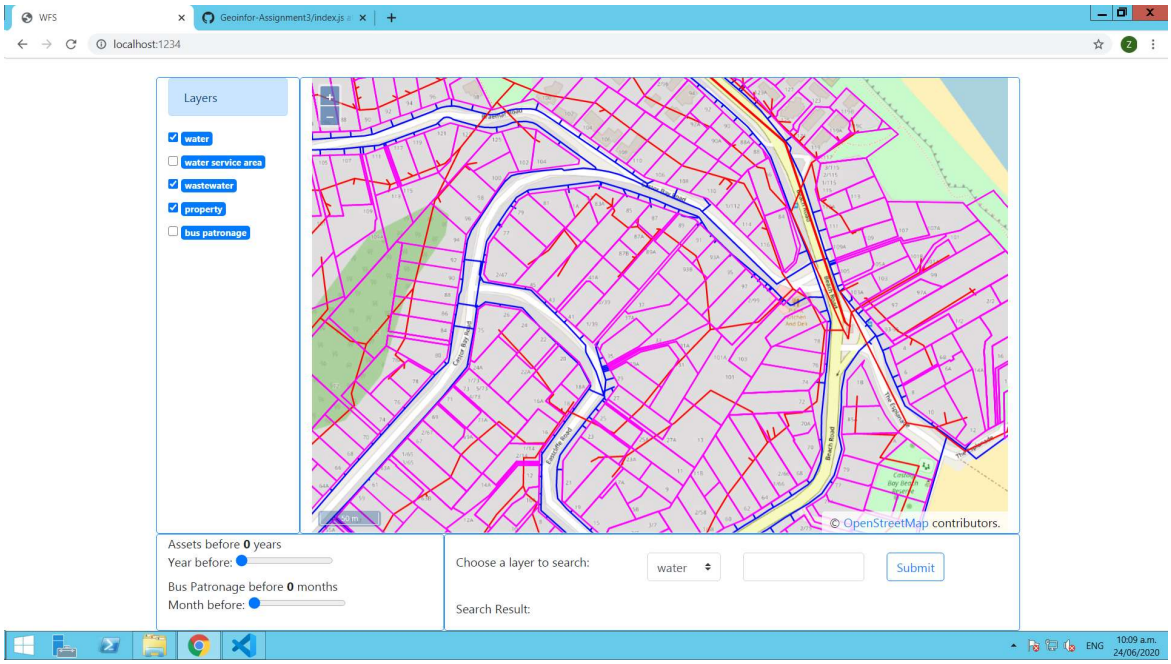
Then open the browser and type **http://localhost :1234** , you will access our web app, as we have already started this service on server, user only needs simply type **http://localhost :1234** to access it in the browser.



4.0 Functionality of Web App:

In this section, the key function of our web app will be demonstrated, which are layers switch, layers on and off, item selection and its attributes pop up , scroll bars that showing items change over time for Auckland , and search functions for different layers.

4.1 Layer Switch, On and Off

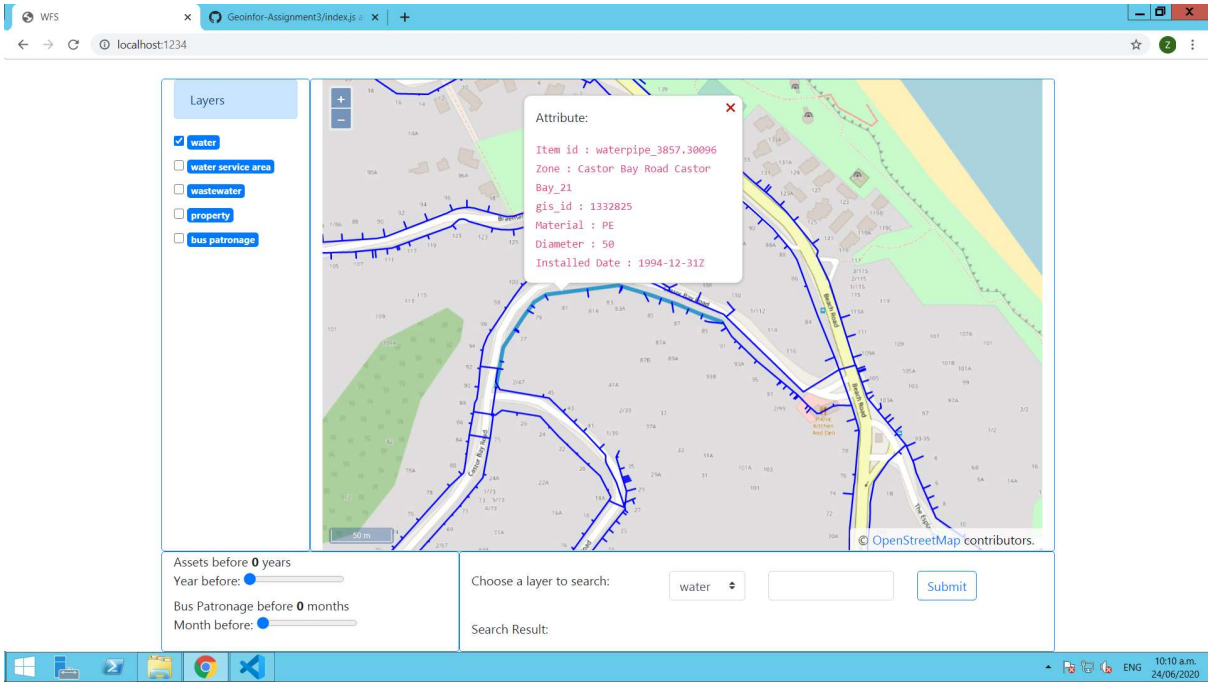


The figure above shows that various data layers as water, water service area, waste water, property, and bus patronage in this web app controlled by checkboxes. Multiple layers can be turned on and off that being displayed on the map at the same time.

If the map has been zoomed out to a certain level, all the datasets would be disappear for the performance considerations in order to prevent loading too much information on the browser.

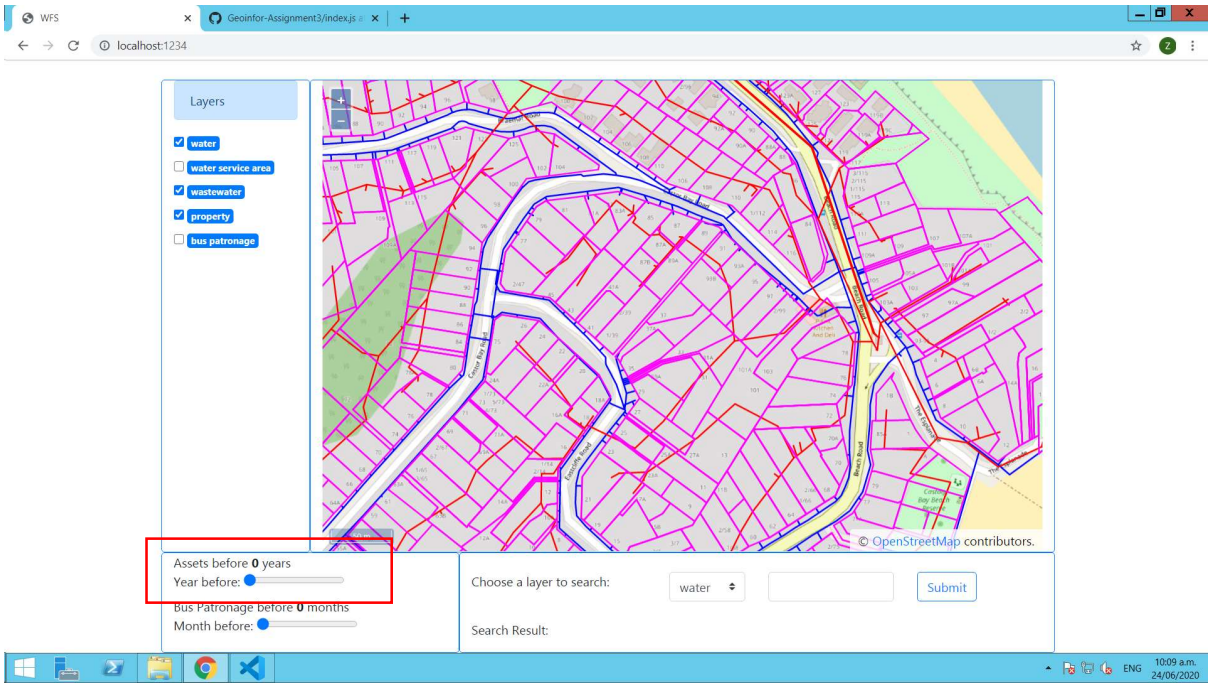
4.2 Item Select and Attributes Pop up

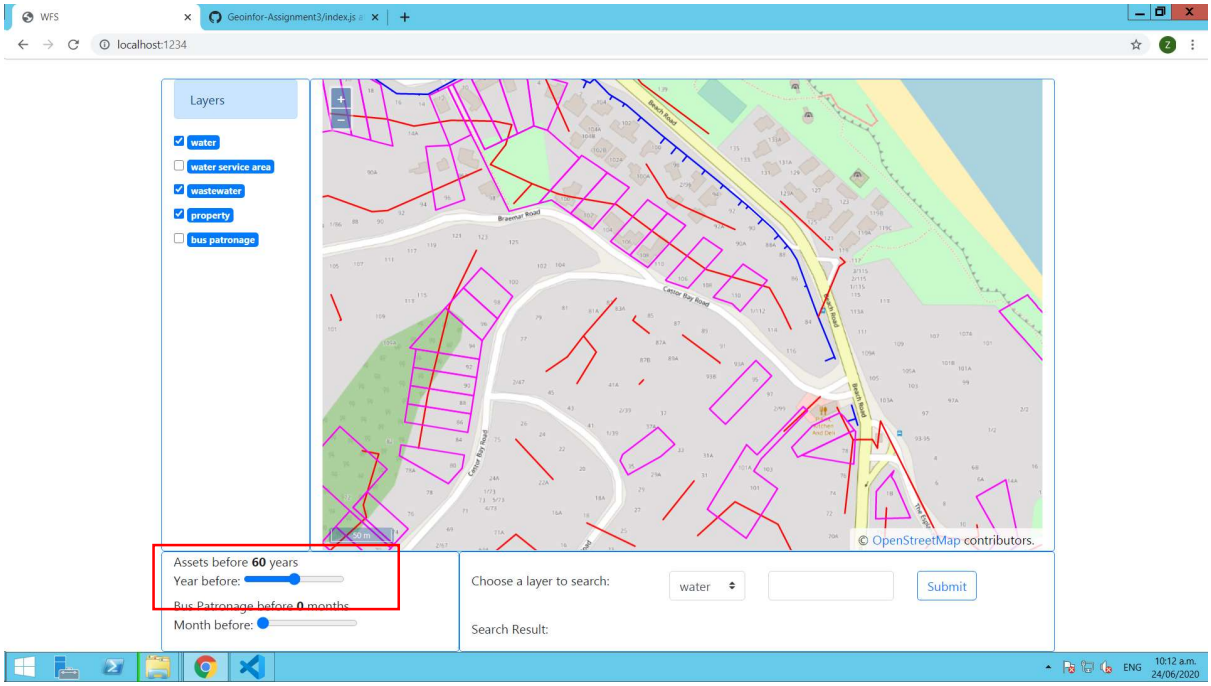
If user just clicks on one of the item in the map, an attribute table will pop out with various information that is item id, titles, type issue, installed date, data id, etc. In addition, the selected item will be highlighted as cyan colour, and all layers have the same functions. Clicking the cross icon in attribute table can close the window. In addition, click any empty area of the map will remove the highlighted feature for that item.

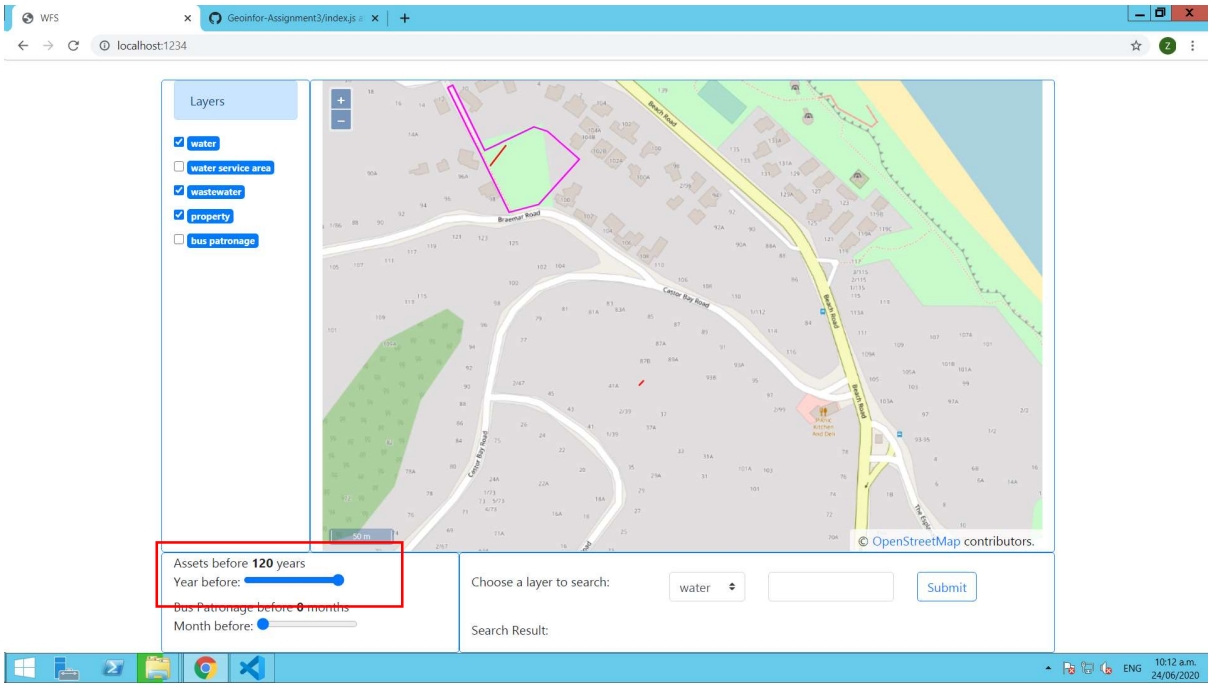


4.3 Scroll Bar to Control Change Over Time for Auckland

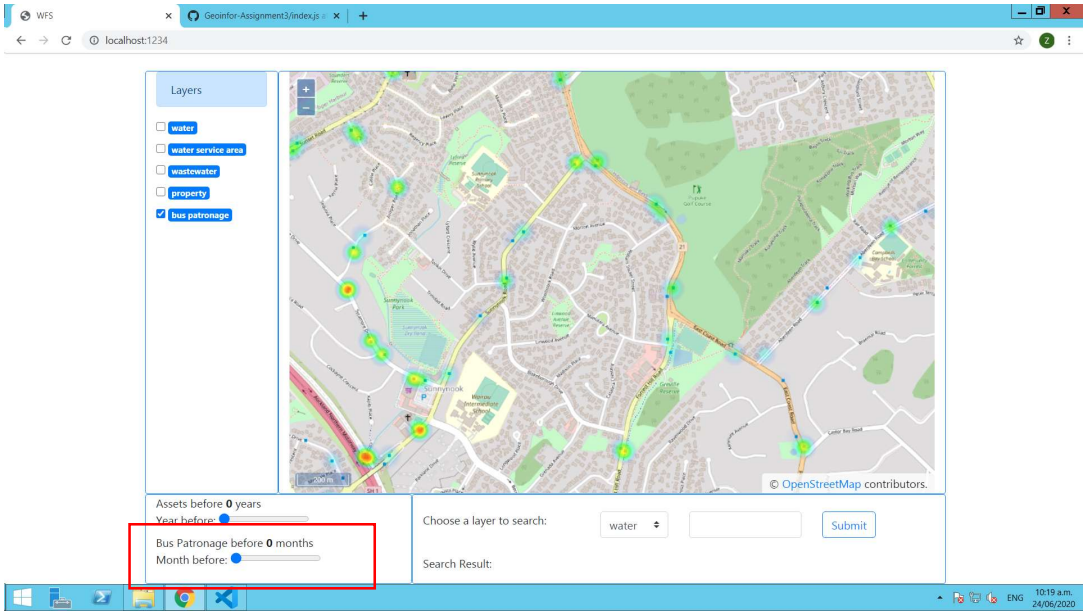
We have designed two scroll bar that controlling the datasets change over time. The scroll bar in figure below shows the spread of existing items from layers been turned on, which is controlled by the scroll bar. If pulling the scroll bar, different items have been displayed on the map represented by the 'Assets Before x years' text. The following figures have shown its effects.

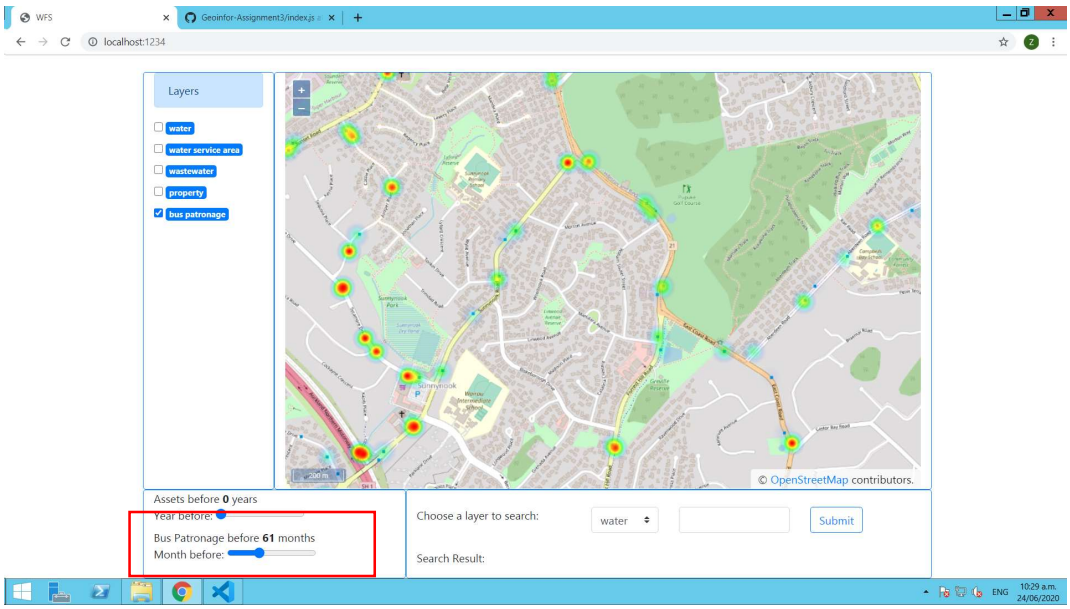


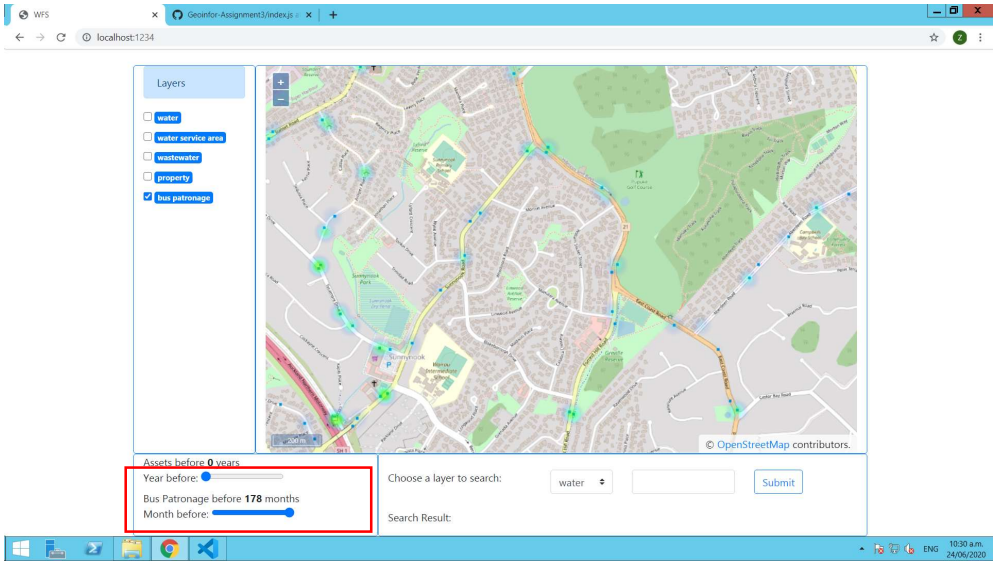




We have also designed a scroll bar for monthly bus patronage as its unit month based compared with year based unit for other layers, and it is represented as heat map. The following figures have shown its effects.

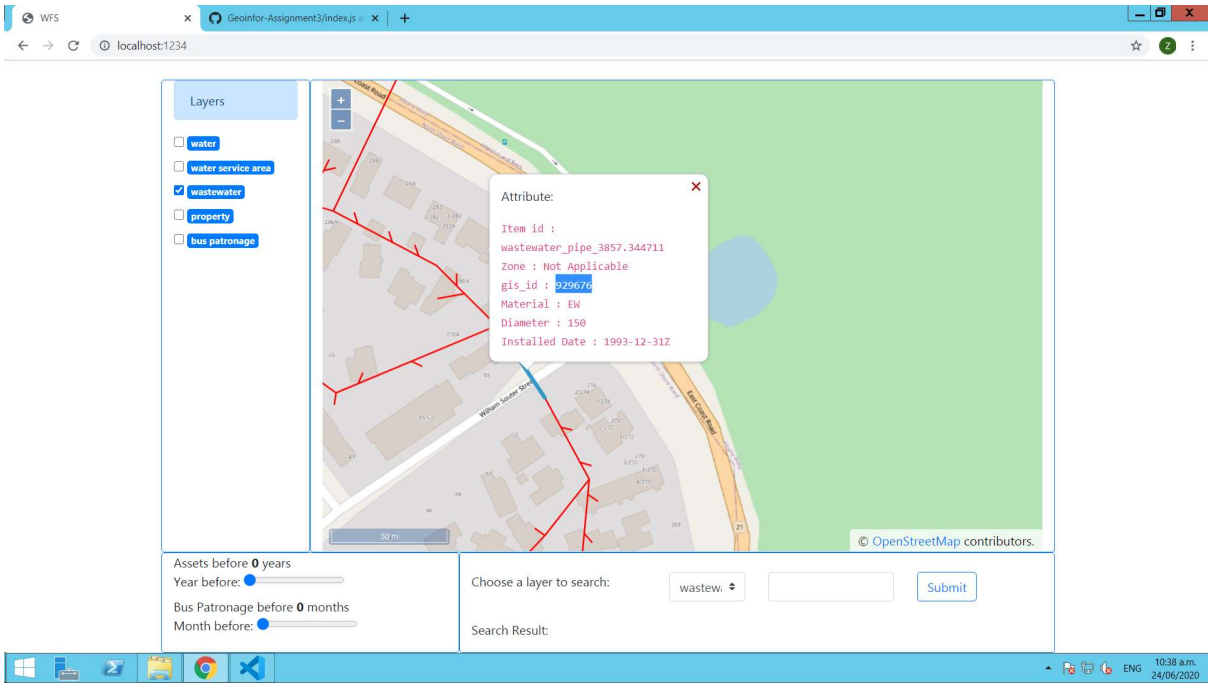

WFS
Geoinfor-Assignment3/index.js
localhost:1234
Layers
water
water service area
wastewater
property
bus patronage
© OpenStreetMap contributors.
Assets before 0 years
Year before:
Bus Patronage before 0 months
Month before:
Choose a layer to search:
water
Submit
Search Result:
ENG
10:19 a.m.
24/06/2020


WFS
Geoinfor-Assignment3/index.js
localhost:1234
Layers
water
water service area
wastewater
property
bus patronage
© OpenStreetMap contributors.
Assets before 0 years
Year before:
Bus Patronage before 61 months
Month before:
Choose a layer to search:
water
Submit
Search Result:
ENG
10:29 a.m.
24/06/2020

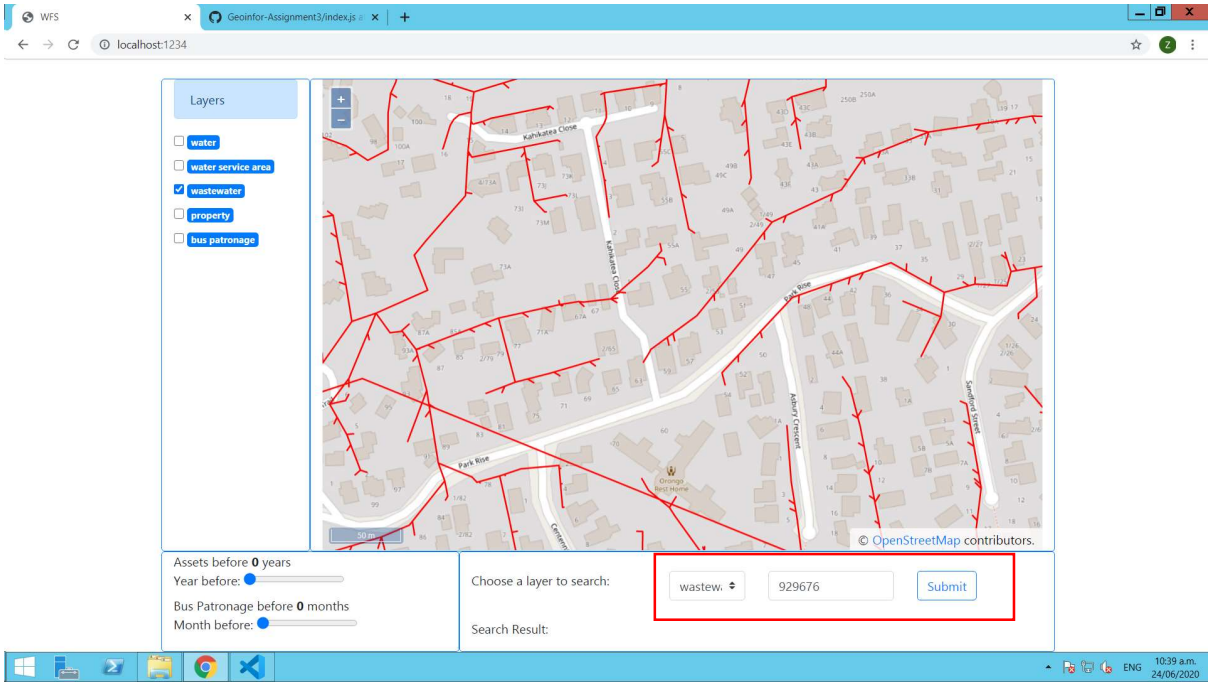

WFS
Geoinfor-Assignment3/index.js
localhost:1234
Layers
water
water service area
wastewater
property
bus patronage
© OpenStreetMap contributors.
Assets before 0 years
Year before:
Bus Patronage before 178 months
Month before:
Choose a layer to search:
water
Submit
Search Result:
ENG
10:30 a.m.
24/06/2020

4.4 Search Function for Layers

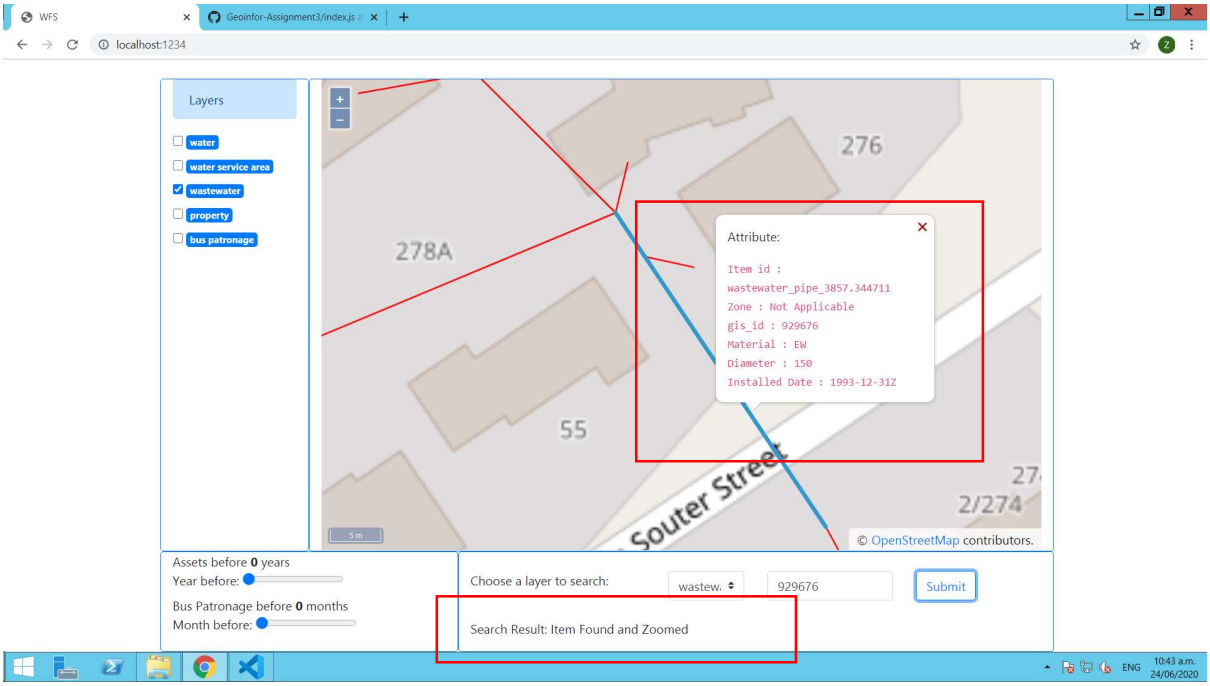
If we need to search one of item from particular layer, we should know its id first, as per example below, we find its gis_id first



Put it in the search text box below and choose wastewater layer, then click submit button



It will automatically zoom to that item, attribute table being pop up and item has been highlighted as well.



Other layers will have the same process for this search function. However, attributes required for as the input for searching is that **gis_id** for water and wastewater layers, and **id** for property layer need, **Bus Stop id** for bus patronage layer.

5.0 Conclusion

In this project, we have analysis the requirements of the GIS product from client. In order to achieve this purpose, a number of technical stacks have been researched and studied including comparison between Leaflet and Openlayers for front end tools, geomap and mapserver for back end server tools.

As a result, we have successfully produced the GIS web app that meets clients requirement, that including its report, live demonstration , deployment on the virtual server allocated to us.