



THE UNIVERSITY OF HONG KONG
DEPARTMENT OF COMPUTER SCIENCE
COMP7503A Multimedia Technologies
Programming Assignment Report

Smart City — Facility Hunter

Group Member:

HUANG Zhuolin 30357*****

HAO Xixuan 30357*****

HUANG Chongxuan 30357*****

November 2020

1. Description of the Smart City & Our Application

Smart cities are formed by the convergence of top-down and bottom-up processes, in which market forces and strategic planning have jointly constructed broadband networks, urban operation systems, embedded systems and software, all of which have changed the city Function and life [1]. Nevertheless, bottom-up initiatives and the participation of individuals and organizations have become the main driving force of urban construction more than ever before, mainly to build solutions and smart city applications without central planning and state control [2]. The software application cluster to solve the needs of the city marks a fundamental turning point in urban construction, especially the construction of smart cities. Smart cities rely on creativity, digital skills, and learning processes to improve the capabilities of citizens [3].



Figure 1: Smart City Word Cloud

Due to this open pattern, web and smartphone applications for smart cities are becoming more and more important to the development of smart cities [4]. More and more citizens, developers, municipalities and companies are creating these applications. The application of smart cities highlights the rise of a tech-savvy pop culture and people's belief in the role of technological advancement. This social movement to create and use apps is an important milestone in the construction of smart cities. It is supported by software development kits, cloud platforms, content management systems, the compilation and reuse of existing software, and an open developer community. These methods jointly break down technical barriers, reduce entry costs, and make smart city solutions applicable In any city, urban or rural community [5].

Our smart city application is based on the data from appropriate data streams from “<https://data.gov.hk/>”. We use Public Facilities and Weather data streams to accomplish

our smart city application. As the flow in figure 2, we collect data from HK government website and store it in the MongoDB database. Through the data obtained in the database, we can process the data and then use the application to disseminate the data. Our users finally get the information they need by using the application [6].

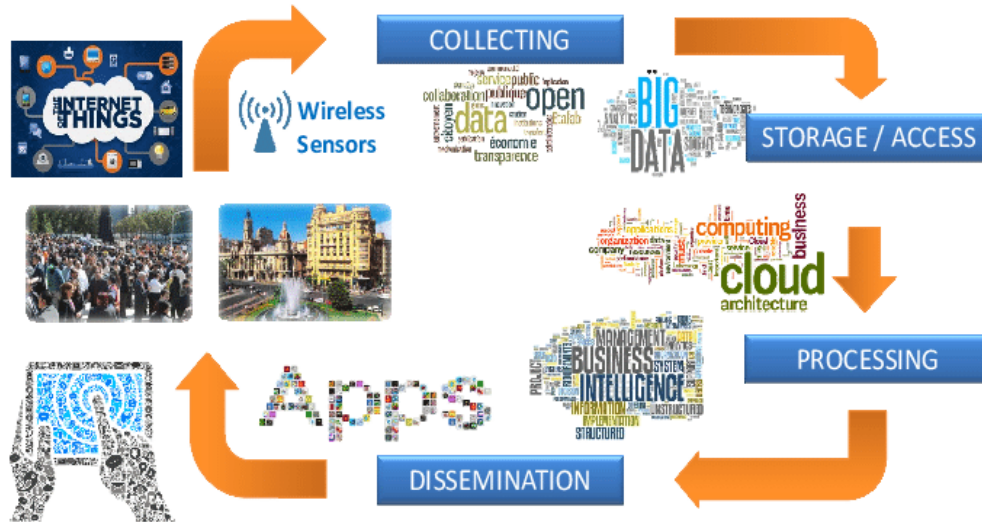


Figure 2: Information Flow Diagram using Node-RED in a Smart City

In detail, our application is very useful for citizens who need to use public facilities. For example, when citizens need to use a public facility, they need to query the geographic location of the public facility. If citizens choose to go to an outdoor public facility, they also need to know the surrounding weather conditions. According to weather conditions, citizens can choose different modes of travel and whether to use the public facility today.

By using our application, citizens can query the corresponding geographic location by the types of public facilities they want to use, such as libraries, gymnasiums, etc., and choose the public facilities that fit them best. After selecting a public facility, citizens can also obtain the weather conditions around the public facility. Through the weather conditions, citizens can finally decide whether to use the public facility today or in future. Here is a specific example. TOM wants to play basketball today, so he needs to find the nearest gym. He opened our application, selected the stadium by the type of public facilities, and found the stadium nearest to him on the map. But at this moment, he found that it was raining around the stadium, so he brought an umbrella before going out, otherwise he would be wet by the rain. “Thanks for this APP”, TOM said.

2. The Use Case & Implementation Details

2.1 Multimedia System usage instruction

First, enter the URL in the browser, localhost:1880/ui, as the following figure 3.

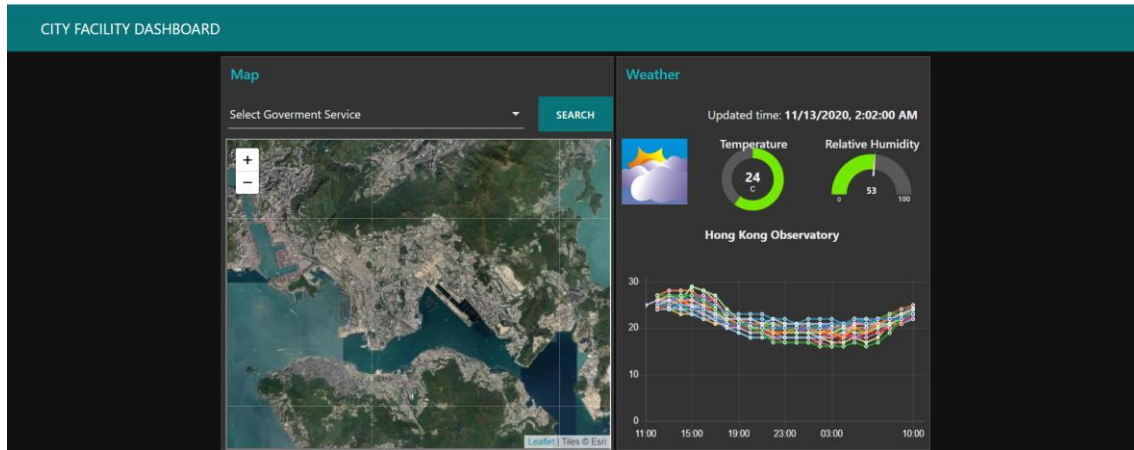


Figure3: Initial Dashboard

This is the original page. On the left side is the map interface. We can get the specific locations of different service facilities by selecting government services, and display the specific locations on the map. On the right is the weather interface, we can intuitively understand the current weather through the weather picture, and intuitively understand the specific values based on the temperature and relative humidity dashboard. The line chart below shows the fluctuation of temperature. Before selecting specific government facilities, it shows the fluctuation of temperature in all areas.

We can see that there is a drop-down menu where we can choose what we want to show on the map, such as libraries, badminton court, etc.

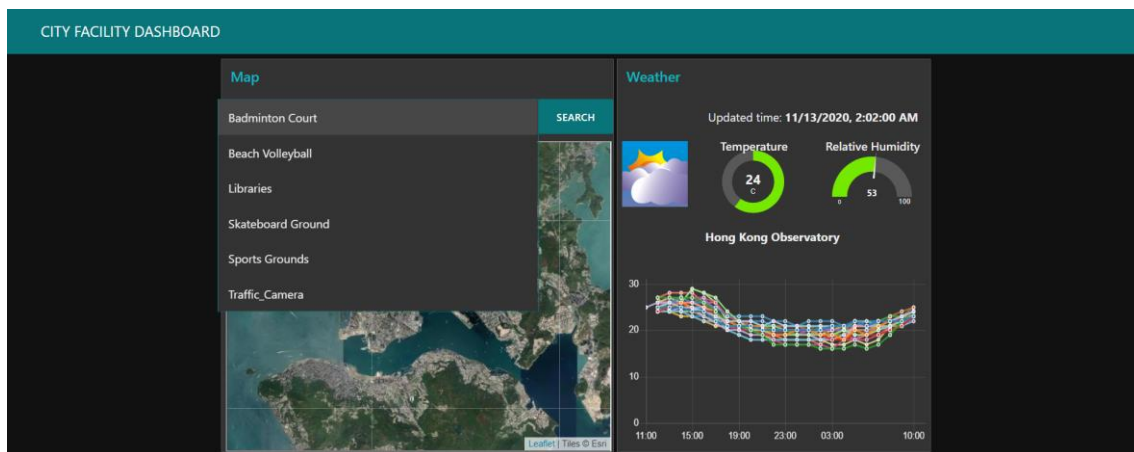


Figure 4: Selecting Government Services

For instance, if we choose Libraries, then press the SEARCH button, as figure 5, we can observe some specific location of the libraries and some gathering points that can be extended. If we select a specific location by clicking it, as figure 6, we can obtain the details about the facility, such as name, district and address. On the weather interface, the relevant weather information will be linked to the district where the facility is located. At the same time, the area where the facility is located will be displayed in the upper right

corner.

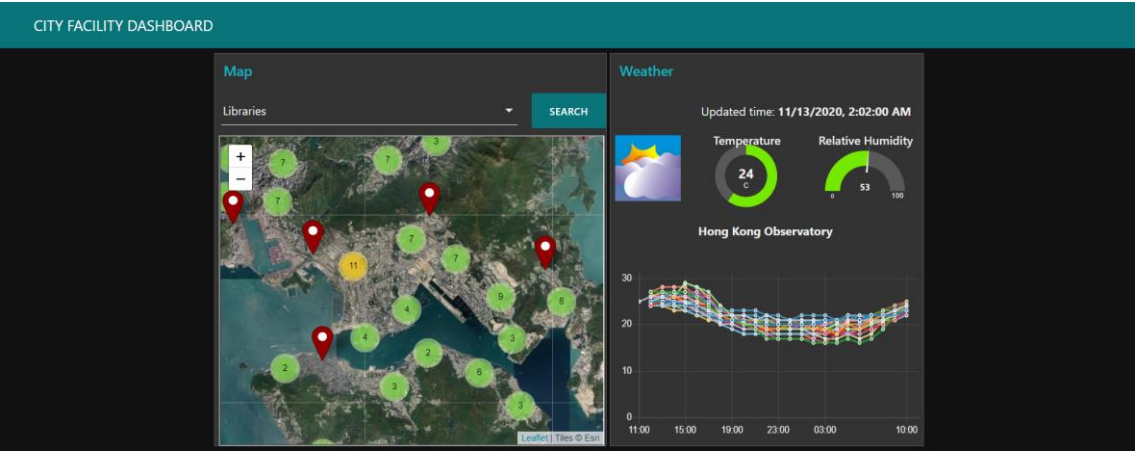


Figure 5: Selecting Government Services and SEARCH

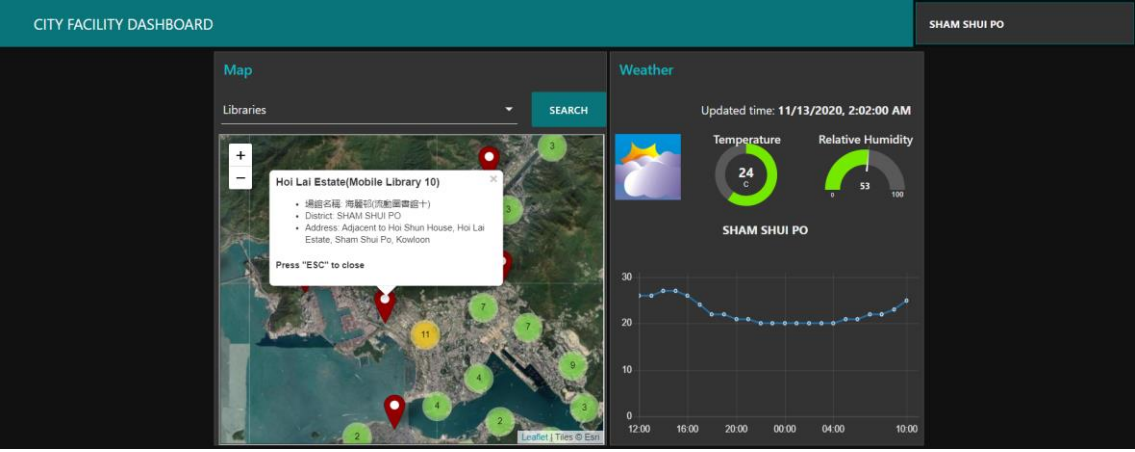


Figure 6: Selecting Specific Location

On the other hand, if we select a gathering point location by clicking it, as figure 7, we can get all the relevant facilities near the gathering point. Then, as figure 8, we can obtain the details about a specific facility, such as name, district and address. On the weather interface, the relevant weather information will be linked to the district where the facility is located. At the same time, the area where the facility is located will be displayed in the upper right corner.

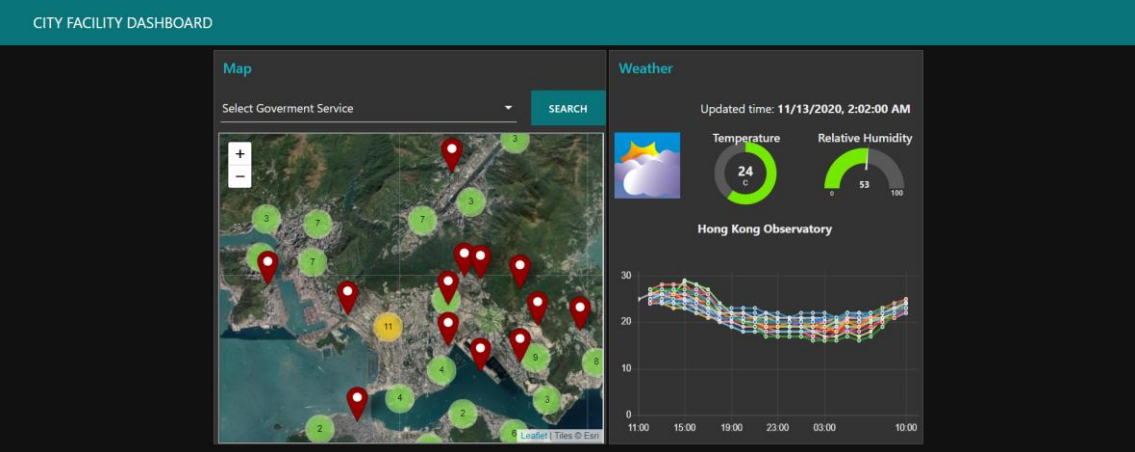


Figure 7: Expanding Gathering Point

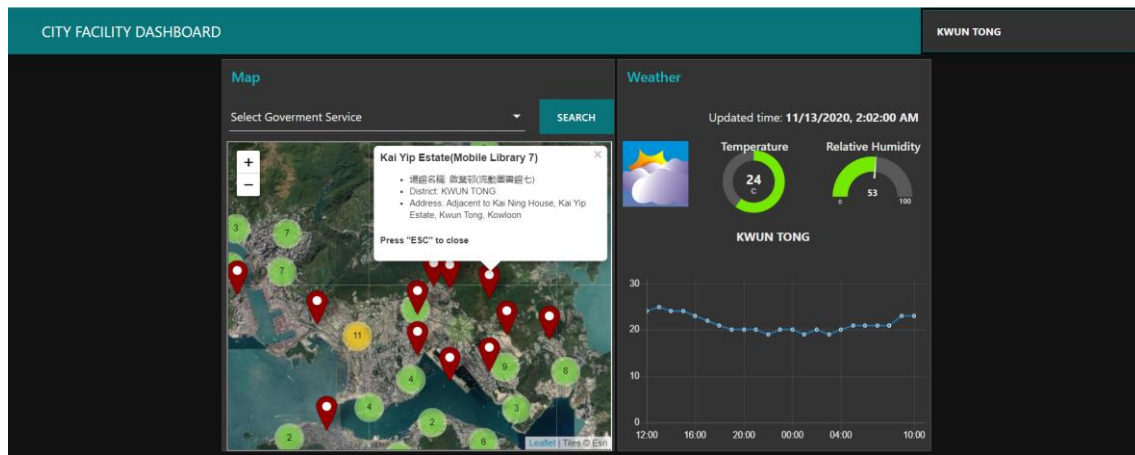


Figure 8: Selecting Specific Location From Gathering Point

2.2 The Actual Usage

Users can see the weather changes and weather in all districts of Hong Kong. Users can select different government services and see the specific location of each facility on the map by clicking the SEARCH button. By clicking a specific point or clicking a specific point expanded by the gathering point, the corresponding specific information of the facility and the corresponding weather conditions can be obtained.

As the flow in figure 9, users can get information about government services such as name, district, address about the facility and the weather information around the facility through this application. And then, they can decide whether they want to go to the facility or not.

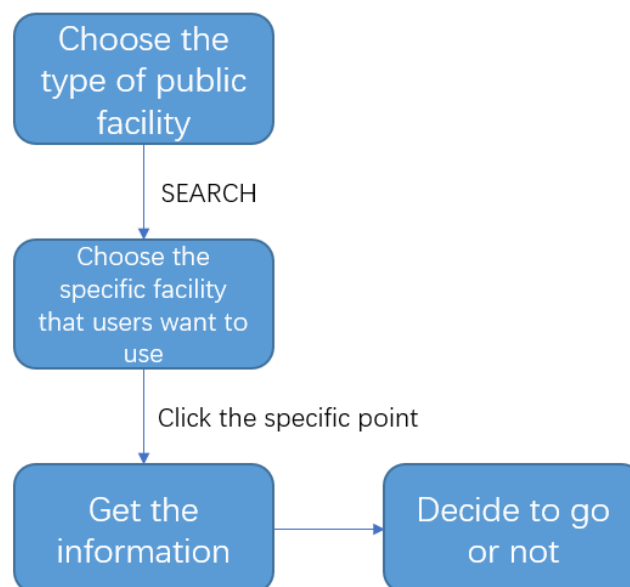


Figure 9: User's Flow

2.3 System structure and logic

2.3.1 system structure

The system comprises 2 parts, Flow 1 is in charge of process data, Flow 2 is in charge of

read and load data.

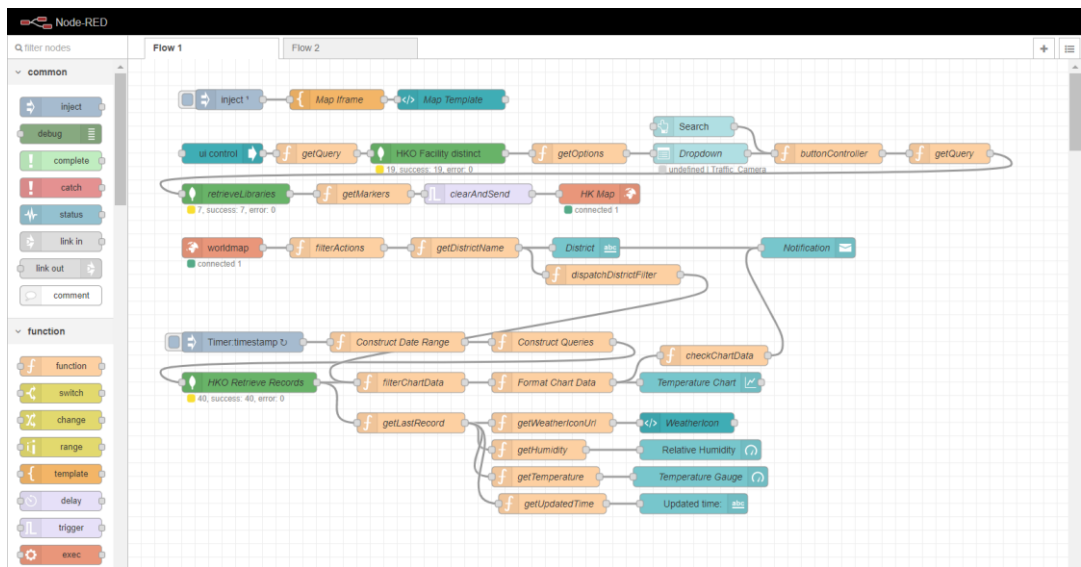


Figure 10: Flow 1

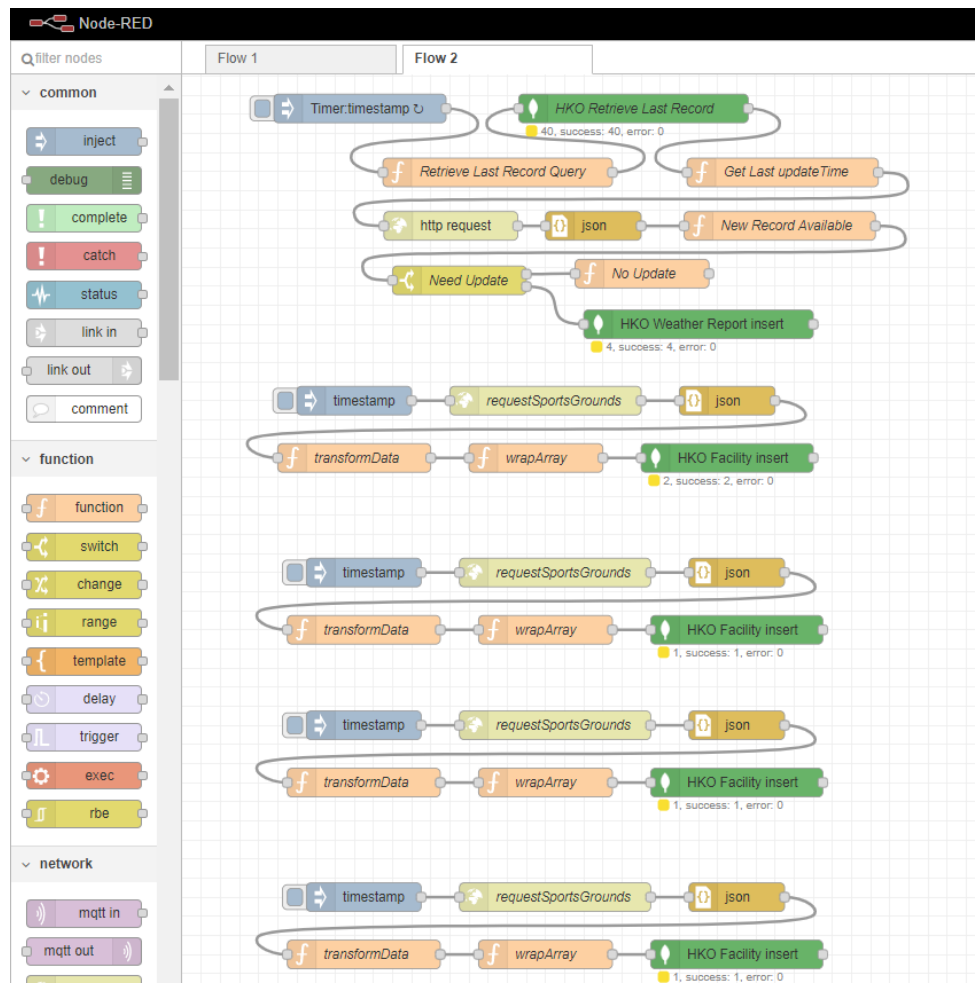


Figure 11: Flow 2

2.3.2 System logic

① Template part

Template part sets the template of the ui view.



Figure 12: template part

② Menu and data collection part

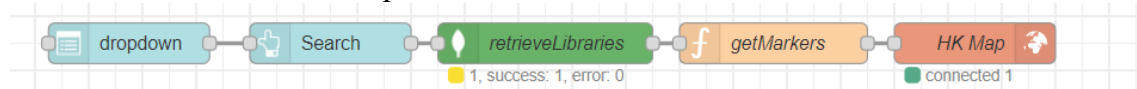


Figure 13: menu and data collection part

Menu and data collection part sets the drop-down menu, gets the libraries data from mongo and sends it to the map, which shows the data.

③ Action response part

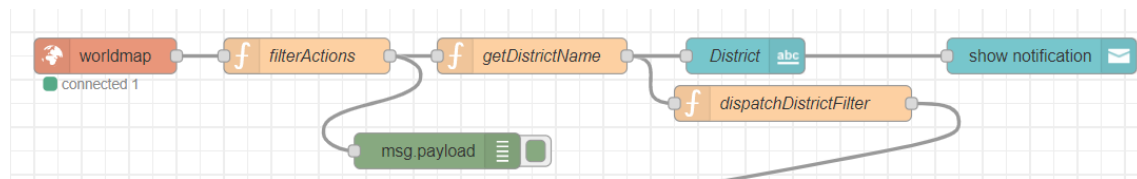


Figure 14: action response part

Action response part captures the click action from user(filters other type of action) and does relevant response.

④ Layout and weather part

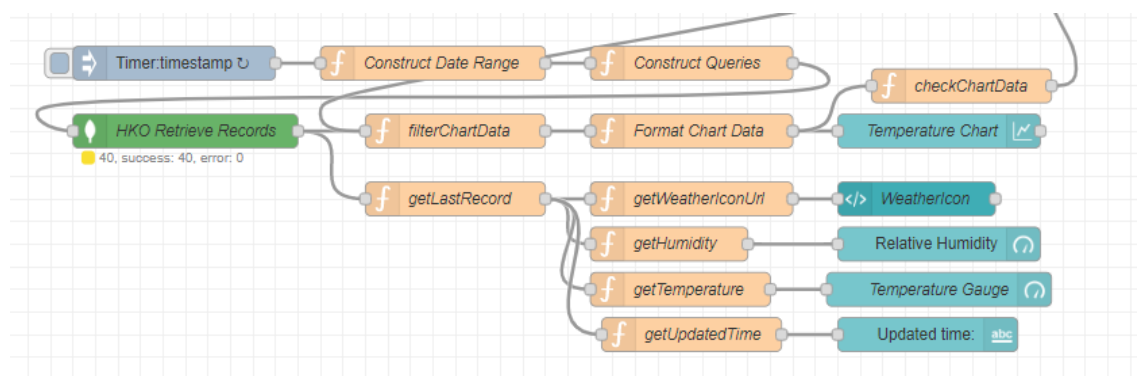


Figure 15: layout and weather part

Layout and weather part sets the layout of the top-right part, such as temperature gauge, humidity, and updates the data every 5 minutes.

3. Data Storing Approach

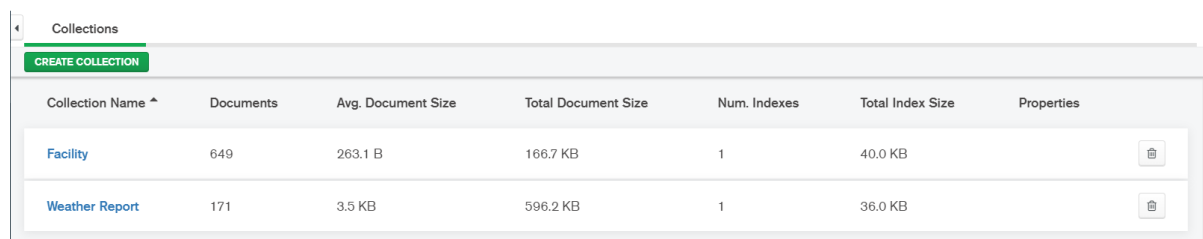
3.1 what data we use/ where the data from

In our application, we will use six different kinds of datasets, which are “Libraries”, “Sports Grounds”, “Traffic_Camera”, “Skateboard Ground”, “Beach Volleyball” as well as “Badminton Court”. These datasets are all from DATA.GOV.HK.

3.2 process and store of the facility data

we adopt two methods to process the data——weather data and facility data

As weather data and facility data have different presentation format, so we need to use different process logic to process data



Collection Name ^	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
Facility	649	263.1 B	166.7 KB	1	40.0 KB	
Weather Report	171	3.5 KB	596.2 KB	1	36.0 KB	

Figure 16: data table in mongodb

3.2.1 facility data process structure

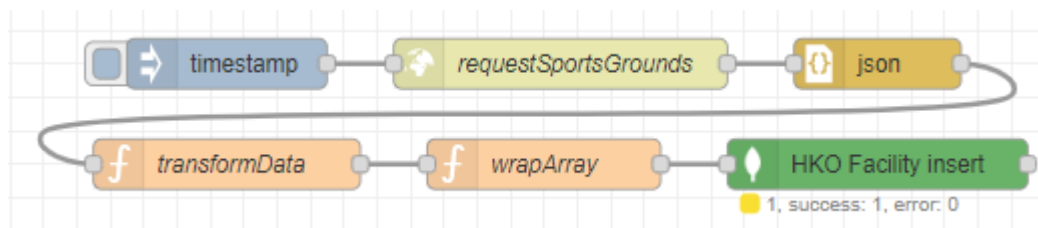


Figure 17: facility data process structure

This structure transform json data to csv format and then store the data into mongodb for different dataset, we only need to change “requestSportsGrounds” node and “transformData” node to process different dataset.

Edit http request node

Delete
Cancel
Done

Properties

Method
GET

URL
https://www.lcsd.gov.hk/datagovhk/facility/facility-s

Payload
Ignore

☐ Enable secure (SSL/TLS) connection

☐ Use authentication

☐ Enable connection keep-alive

☐ Use proxy

Return
a UTF-8 string

Name
requestSportsGrounds

Figure 18: requestSportsGrounds node content

Edit function node

Delete
Cancel
Done

Properties

Name
transformData

Setup
Function
Close

```

1 const parseLatLon = t => {
2   const segments = t.split("-");
3   const [d, m, s] = segments;
4   // return `${first}.${rest.join("")}`;
5   return parseFloat(d) + parseFloat(m) / 60 + parseFloat(s) / 3600;
6 }
7 return {
8   payload: msg.payload.map(item => ({
9     Name_en: item.Name_en,
10    Name_cn: item.Name_cn,
11    District_en: item.District_en,
12    District_cn: item.District_cn,
13    Address_en: item.Address_en,
14    Address_cn: item.Address_cn,
15    Longitude: parseLatLon(item.Longitude),
16    Latitude: parseLatLon(item.Latitude),
17    Dataset: "Sports Grounds",
18    Image: "",
19  }))
20 };

```

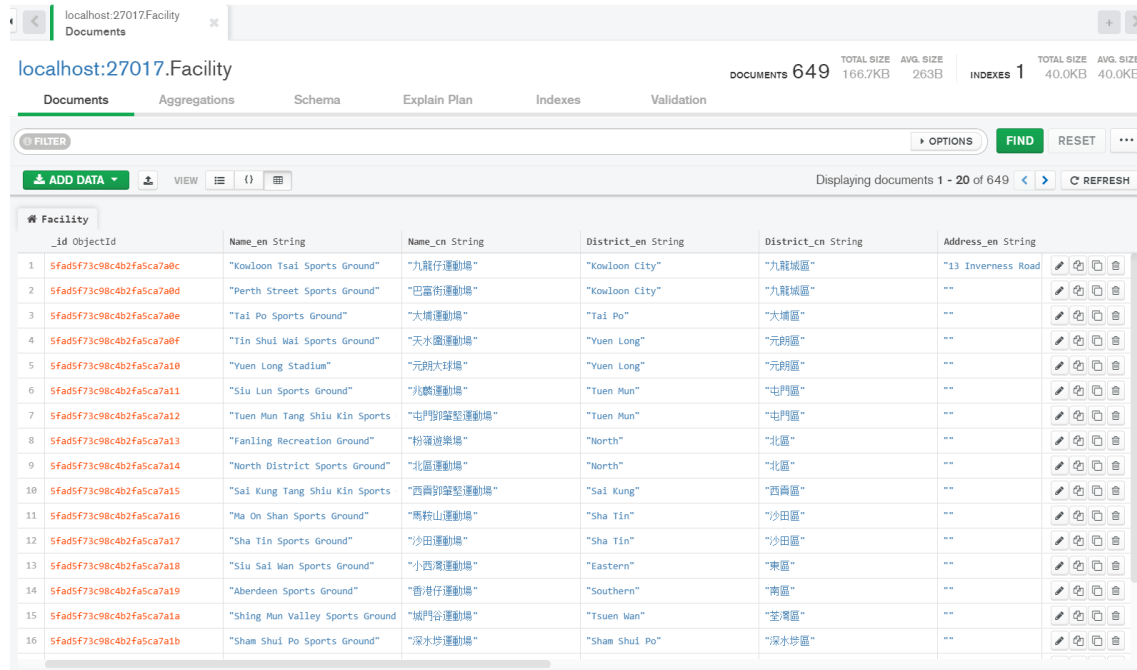
Outputs
1

☐ Enabled

Figure 19: transformData node content

3.2.2 Data extraction and integration

Actually, In every dataset there are many columns but we do not need all of them, we only need 5-6 columns to find the locations and functions. Thus, in the “transformData” node, we do data extraction, selecting the specific column that we need. Through the “Facility insert” node, we insert data into the facility table mongodb, achieving data integration.



	_id	ObjectID	Name_en String	Name_cn String	District_en String	District_cn String	Address_en String
1	5fad5f73c98c4b2fa5ca7a0c		"Kowloon Tsai Sports Ground"	"九龍仔運動場"	"Kowloon City"	"九龍城區"	"13 Inverness Road"
2	5fad5f73c98c4b2fa5ca7a0d		"Perth Street Sports Ground"	"巴拿街運動場"	"Kowloon City"	"九龍城區"	"--"
3	5fad5f73c98c4b2fa5ca7a0e		"Tai Po Sports Ground"	"大埔運動場"	"Tai Po"	"大埔區"	"--"
4	5fad5f73c98c4b2fa5ca7a0f		"Tin Shui Wai Sports Ground"	"天水圍運動場"	"Yuen Long"	"元朗區"	"--"
5	5fad5f73c98c4b2fa5ca7a10		"Yuen Long Stadium"	"元朗大球場"	"Yuen Long"	"元朗區"	"--"
6	5fad5f73c98c4b2fa5ca7a11		"Siu Lun Sports Ground"	"兆麟運動場"	"Yuen Mun"	"屯門區"	"--"
7	5fad5f73c98c4b2fa5ca7a12		"Tuen Mun Tang Shiu Kin Sports"	"屯門鄧肇堅運動場"	"Tuen Mun"	"屯門區"	"--"
8	5fad5f73c98c4b2fa5ca7a13		"Fanling Recreation Ground"	"粉嶺遊樂場"	"North"	"北區"	"--"
9	5fad5f73c98c4b2fa5ca7a14		"North District Sports Ground"	"北區運動場"	"North"	"北區"	"--"
10	5fad5f73c98c4b2fa5ca7a15		"Sai Kung Tang Shiu Kin Sports"	"西貢鄧肇堅運動場"	"Sai Kung"	"西貢區"	"--"
11	5fad5f73c98c4b2fa5ca7a16		"Ma On Shan Sports Ground"	"馬鞍山運動場"	"Sha Tin"	"沙田區"	"--"
12	5fad5f73c98c4b2fa5ca7a17		"Sha Tin Sports Ground"	"沙田運動場"	"Sha Tin"	"沙田區"	"--"
13	5fad5f73c98c4b2fa5ca7a18		"Siu Sai Wan Sports Ground"	"小西灣運動場"	"Eastern"	"東區"	"--"
14	5fad5f73c98c4b2fa5ca7a19		"Aberdeen Sports Ground"	"香港仔運動場"	"Southern"	"南區"	"--"
15	5fad5f73c98c4b2fa5ca7a1a		"Shing Mun Valley Sports Ground"	"城門谷運動場"	"Tsuen Wan"	"荃灣區"	"--"
16	5fad5f73c98c4b2fa5ca7a1b		"Sham Shui Po Sports Ground"	"深水埗運動場"	"Sham Shui Po"	"深水埗區"	"--"

Figure 20: Data In MongoDB

3.3 process and store of weather data

“Current weather report data” updates “HOURLY AND AS NECESSARY” and it contains the information of relative humidity and the air temperatures in different districts. LINK: <https://data.gov.hk/en-data/dataset/hk-hko-rss-current-weather-report>.

Weather data storing approach

As the following flow, at the beginning, timestamp repeats every 5 minutes, this is because the weather information updates once an hour, and we want to get the weather information as soon as possible. For example, if someone uses the application at 5:01, we can update the information in 5minutes, otherwise the user will get the information after a long period of time (due to the repeat period).

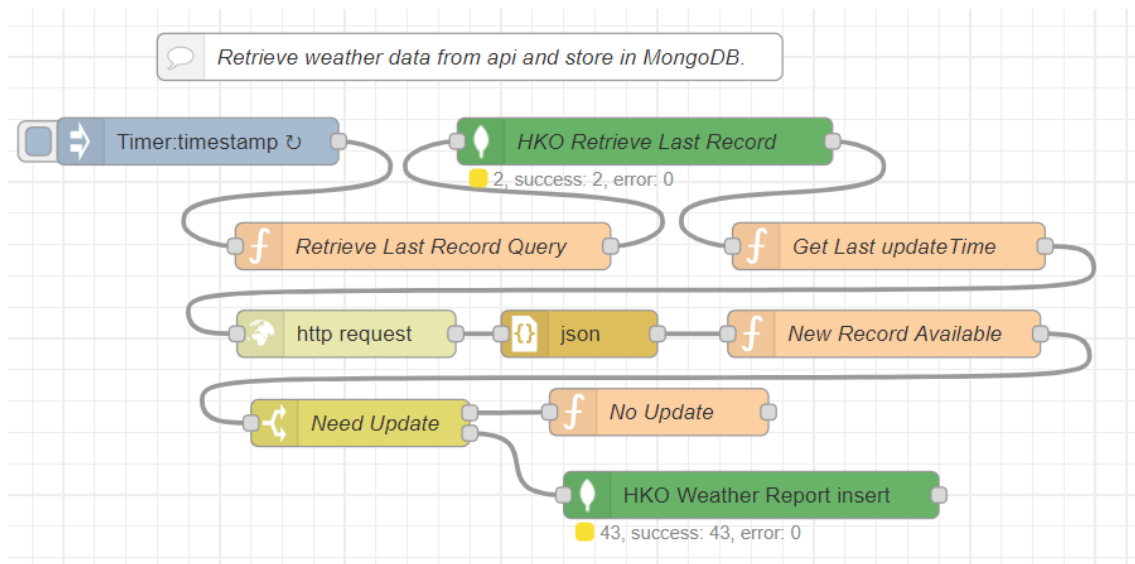


Figure 21: Weather Data Storing Approach

To finish this quickly update job, we also need to retrieve the last record query and keep it in the database. Therefore, we need to prepare a query and connect to MongoDB. In the ‘retrieve last record query’ function as figure 22, we query the latest record in the database. As figure 23, we set up MongoDB and then we can successfully get the latest record. And we can get the update time as figure 24.

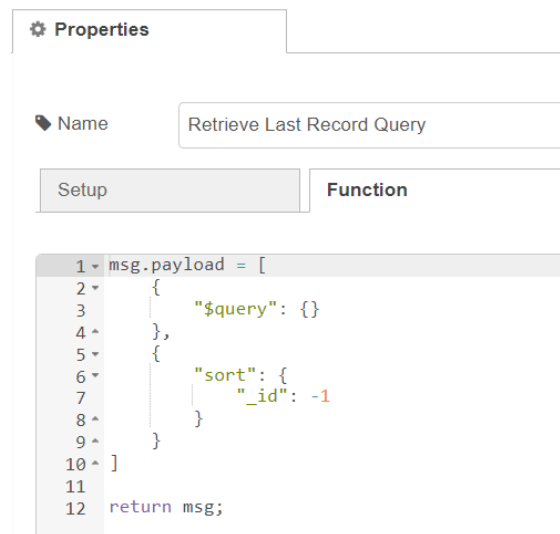


Figure 22: ‘Retrieve Last Record Query’ Function

Edit mongodb3 in node

Delete Cancel Done

Properties

Service: External service

Server: HKO

Collection: Weather Report

Operation: findOne

Name: HKO Retrieve Last Record

Figure 23: Set Up MongoDB

Name: Get Last updateTime

Setup Function Close

```

1 if(msg.payload.hasOwnProperty('updateTime')) {
2   flow.set('lastUpdateTime', msg.payload.updateTime);
3 } else {
4   flow.set('lastUpdateTime', '');
5 }
6
7 return msg;

```

Figure 24: Get Latest Update Time

After that, we need to check if there is any update for the data. Therefore, we request data through HTTP. As figure 25, we get the data from 'data.gov.hk'. Then we edit the json node and preprocess the data. As figure 26, we also need to check that if the update time greater than last update time, if yes, we update the record in MongoDB, else we do not need the update. So far, the weather data storing has finished.

Delete
Cancel
Done

⚙️ Properties

Method

GET

URL

https://data.weather.gov.hk/weatherAPI/opendata/

Payload

Ignore

☐ Enable secure (SSL/TLS) connection

☐ Use authentication

☐ Enable connection keep-alive

☐ Use proxy

Return

a UTF-8 string

Name

Name

Figure 25: Request Data Through HTTP

Name

New Record Available

📄

Setup

Function

Close

1

lastUpdateTime = flow.get('lastUpdateTime');

2

3

var updateTimeTemp = new Date(msg.payload.updateTime)

4

var updateTime = updateTimeTemp.toISOString() ;

5

6

if(updateTime > lastUpdateTime) {

7

msg.needUpdate = true ;

8

msg.payload.updateTime = updateTime ;

9

} else {

10

msg.needUpdate = false ;

11

}

12

13

return msg;

Figure 26: Check If The Data Need To Be Updated

Edit mongodb3 in node

Delete

Cancel

Done

⚙ Properties

⚙

📄

🖨

Service

External service

▼

Server

HKO

▼

✎

Collection

Weather Report

🔑 Operation

insert

▼

Name

Name

Figure 27: Insert Weather Data into MongoDB

4. Major Features

4.1 Display facility locations on Hong Kong Map

To display a map in Node-RED, the library “node-red-contrib-web-worldmap” is chosen.

1) Install the library in Node-RED. After installation, we can see this library in the Node-RED management interface.

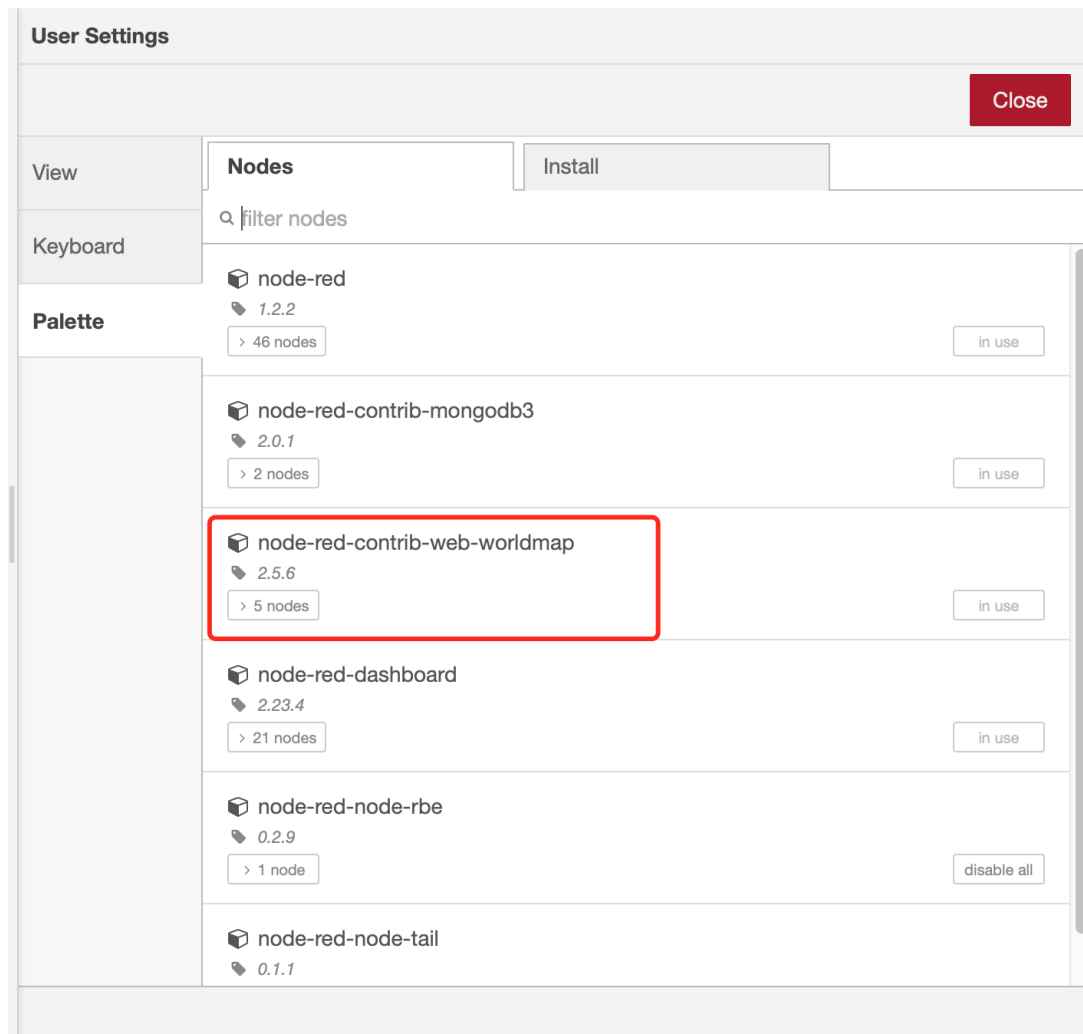


Figure 28: “Node-red-contrib-web-worldmap” in Palette

2) Integrate Map widget to Node-RED dashboard. By default, the world map can only be visited by route “localhost:1880/worldmap”, but our main application’s route is “localhost:1880/ui”. We use the iframe template provided by Node-RED dashboard to solve this problem. In the iframe template, its source is referred to the map’s route “../worldmap”. After finishing this template and adding a “worldmap” node, we can see that the map app appears in dashboard.

Iframe template configuration and the corresponding flow is shown below.

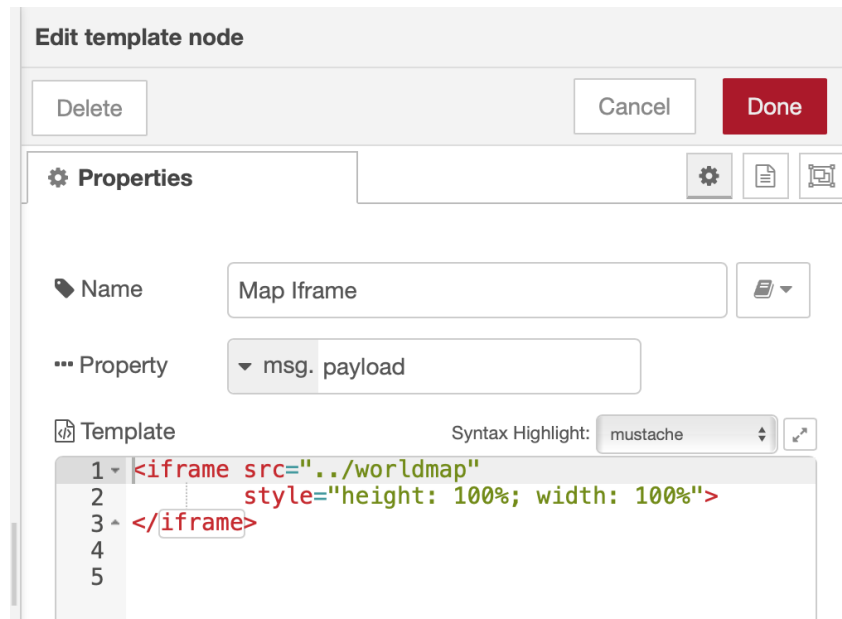


Figure 29:Map Iframe template

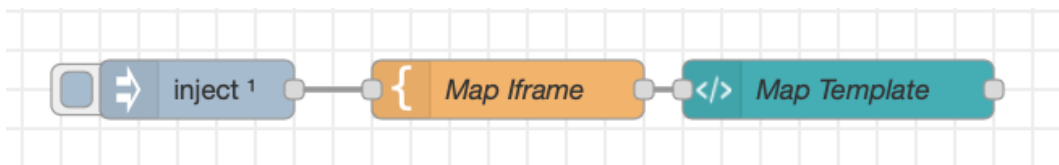


Figure 30:The template will be automatically injected once the flow is deployed.

3) In the Map node, set the initial location and default zoom.

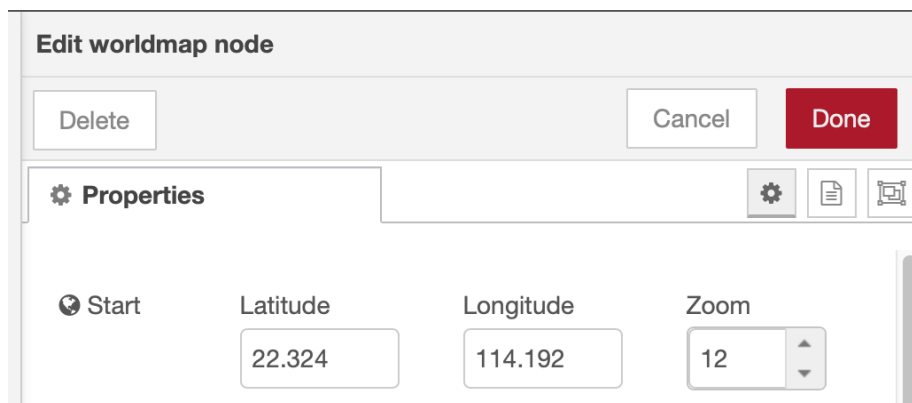


Figure 31: Map node setting

4) Add location markers on the map. According to the documentation of “node-red-contrib-web-worldmap”, we need attributes such as “name”, “lat” and “lon” to display a marker on the map. We construct a flow here to retrieve facility records from mongoDB and forward the data to the map.

5) Display information popup for a facility when a marker is clicked. We customize the popup to show the main information for a facility, and if there is any image available for the facility it will be in the popup at the same time. Here is our popup template, it will be passed as an attribute of the markers to the Map node.

```

1  const items = Object.values(msg.payload);
2  flow.set("libraries", items);
3
4  const getPopup = item => {
5    return
6    <div>
7      <ul>
8        <li>場館名稱: ${item.Name_cn ? item.Name_cn : ""}</li>
9        <li>District: ${item.District_en ? item.District_en : ""}</li>
10       <li>Address: ${item.Address_en ? item.Address_en : ""}</li>
11       ${item.Image ? `` : ""}
12     </ul>
13     <h6>Press "ESC" to close</h6>
14   </div>`
15 }
16 return {
17   payload: items.map(item => ({
18     name: item["Name_en"],
19     lat: item["Latitude"],
20     lon: item["Longitude"],
21     popup: getPopup(item),
22   })),
23 }
24

```

Figure 32: Popup template and markers construction

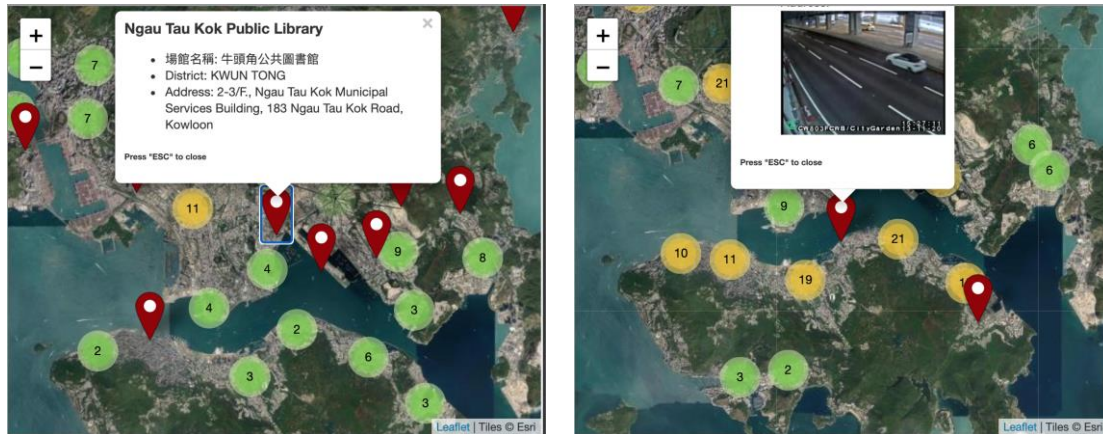


Figure 33: 2 types of popup

4.2 Multiple weather widget

To show the weather condition of a district, we provide multiple kinds of widgets to enrich user experience: a weather Icon, a temperature Pie Chart, a humidity Gauge Chart and a history temperature Line Chart.

1) Weather Icon. The image is provided by weather data from “data.gov.hk”. Here we need to create a new “img” template, and pass the image url to it.

Template type: Widget in group

Group: [CITY FACILITY DASHBOARD] Weather

Size: 2 x 2

Name: WeatherIcon

Template:

```
1 
```

Figure 34: Weather icon template configuration

- 2) Temperature Pie Chart. Here we just extract the latest weather and pass the temperature as payload to it. If the temperature is between 20°C and 25°C, the color is green indicating comfortable. If the temperature is higher than 25°C, the color will change to orange. If the temperature is lower than 20°C, the color will change to blue.
- 3) Humidity Gauge Chart. This is similar to the Pie Chart, with humidity ranging from 0 to 100. If the humidity is higher than 80, the color will turn red.
- 4) Line Chart for temperature history. This component is very similar to the tutorial demo, with some functionality extension: when the user clicks on a location marker, we will find out to which district that facility belongs, and filter out data only related to that district.

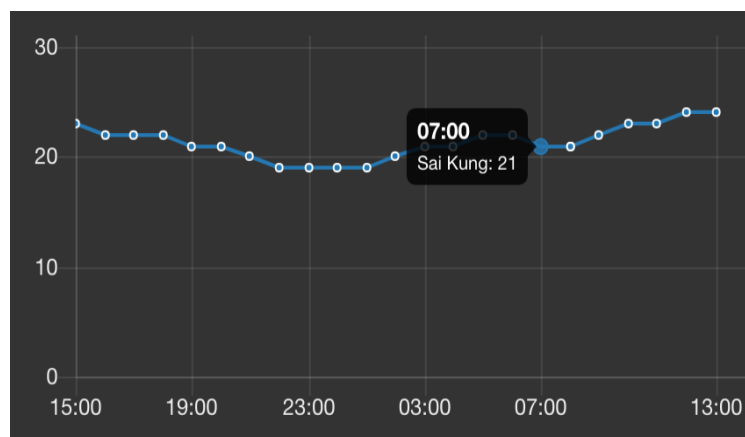


Figure 35: Historic temperature data in a single district

4.3 Multiple facility resources

- 1) Data integration. There are various data sources about different facilities, each provided by one URL. We integrate these data into one “Facility” table so a user can conveniently choose any sort of facility. Each type of facility is distinguished by the “Dataset” attribute, eg, “Libraries” and “Sports Grounds”.
- 2) Fetch select options on loading. In order to make our application flexible and scalable, we avoid hard coding the facility types. When a user loads the application, a request will

be made to mongoDB to get the existing facility types and then pass the select options to the dropdown select node dynamically. The “ui control” node can trigger an event automatically every time a user loads the application.



Figure 36: Implementation of options for dropdown

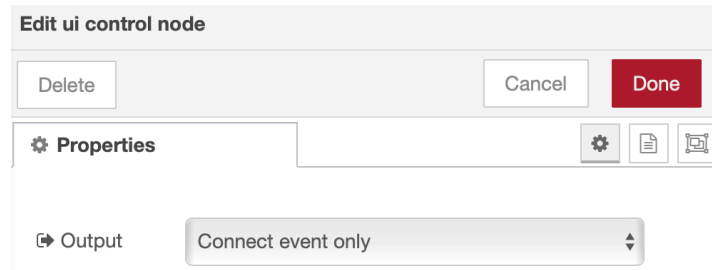


Figure 37: Configuration of ui control node

3) Button control. When a user clicks the “Search” button, it needs to know the current active facility type. However, if the user merely selects a facility type but still has not clicked the “Search” button, nothing should happen. To tackle this situation, we design the node structure below:

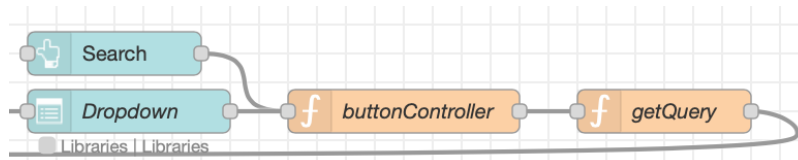


Figure 38: “Search” button node structure

```

1  const {payload, topic} = msg;
2  if (topic === "dropdown") {
3    context.set("dropdown", payload);
4  }
5  if (topic === "search") {
6    const key = context.get("dropdown");
7    if (!key) {
8      return
9    }
10   return {
11     payload: key
12   }
13 }

```

Figure 39: Code of “buttonController”

The button stores the facility option from the Dropdown, and passes the option when it receives a message from the “Search” button.

4.4 User interaction

When a user clicks a certain facility, our system will display the information of that facility on the map, meanwhile the temperature chart will display data of the district corresponding to that facility location. If any image of the clicked facility is available, it will also be shown on the map.

1) “Click” event on the map. To notify the other components to update when a user clicks a marker on the map, we need to capture the “click” event. The “node-red-contrib-web-

worldmap” library provides us with a “worldmap in” component to implement this. It can capture all the actions from the map, and pass the actions to the output node. In our case, we only need to concern 2 types of actions: “click” and “connected”. A “click” event is fired when a user clicks on a marker, and a “connected” event is fired when a user loads our application. On a “connected” event we can do some initialization logic.

2) “Dispatch” the filter to the weather data chart. Because the weather records retrieved from mongoDB consist of data from all the districts, we need to add a filter after retrieving the records from the mongoDB. Before the user selects a certain facility, the filter is deactivated. Once the weather records have been fetched, they are stored in the context of the node, so there is no need to fetch the weather data again when the filter receives a “filter” command. On the other hand, when the weather data is updated(per 5 minutes), it will check the cached filter keyword again before passing the new data through. The implementation of the filter is shown below:

```
24- if (topic === "Timer") { // Data update
25-   context.set("data", Object.values(payload));
26- }
27- if (topic === "Filter"){ // Receive a filter keyword
28-   context.set("filter", payload.toUpperCase());
29- }
30- const filter = context.get("filter");
31- let data = context.get("data");
32- if (filter){
33-   data = data.map(item => ({
34-     ...item,
35-     temperature: {
36-       ...item.temperature,
37-       data: item.temperature.data.filter(t => t.place === districtToPlace[filter])
38-     }
39-   })))
40- }
41- return {
42-   payload: data,
43- }
```

Figure 40: filter weather records

3) We provide 2 ways to close popup on the map. One is pressing the “Esc” button to close all activated popups, the other is clicking the “cross” icon on the upper right to one popup.

4.5 User notification

We use the Notification component provided by Node-RED Dashboard to give some useful feedback to the user. For example, if the user clicks on a marker but the weather data of that district is not provided by the government, we will send a notification to inform the user. Also, if the data is successfully fetched, a notification will be sent to show the update.

Conclusion

In this smart city application, we use six types of facility data (libraries, sports grounds, beach volleyball, traffic cameras, skateboard ground and badminton court) and weather data, enabling users to easily find the location and other detailed information of relevant facilities as well as the weather condition, which could make their lives more convenient.

In the UI interface, we have achieved five main function:

- Display facility locations on Hong Kong Map
- Multiple weather widget
- Multiple facility resources
- User interaction
- User notification

From the release of the programming assignment to deadline, our work lasted for five weeks, our schedule is as below:

1st week: get familiar with node-red and javascript

2nd week: find previous smart city cases and propose our own design idea

3th-4th weeks: data collection & system design

5th week: write report

In the process of completing this project, we gained new knowledge of Node-RED and javascript, as well as a deeper experience of multimedia systems.

Many thanks to Dr. Bill Luo and TA Zhang Wenhua for proving us such a fantastic programming assignment. Although it is not easy to complete the project, we have gained quite a lot and it did leave us a deep impression.

References

- [1]N. Komninos, C. Bratsas, C. Kakderi and P. Tsarchopoulos, "Smart City Ontologies: Improving the effectiveness of smart city applications", *Journal of Smart Cities*, vol. 1, no. 1, 2016. Available: 10.18063/jsc.2015.01.001.
- [2]O. Golubchikov and M. Thornbush, "Artificial Intelligence and Robotics in Smart City Strategies and Planned Smart Development", *Smart Cities*, vol. 3, no. 4, pp. 1133-1144, 2020. Available: 10.3390/smartcities3040056.
- [3]E. Farelnik and A. Stanowicka, "Smart City, Slow City and Smart Slow City as Development Models of Modern Cities", *Olsztyn Economic Journal*, vol. 11, no. 4, pp. 359-370, 2016. Available: 10.31648/oj.2938.
- [4]"IOT Based Application for Smart City Implementation", *International Journal of Modern Trends in Engineering & Research*, vol. 3, no. 9, pp. 146-149, 2016. Available: 10.21884/ijmter.2016.3057.cdtxa.
- [5]A. Bhosale, A. Zari and S. Tilekar, "A Review: Web Application for Smart City", *International Journal of Data Structures*, 2020. Available: 10.37628/ijods.v6i1.593.
- [6]L. YE, "Application of VR Agriculture Interaction Animation in Integrated Services of Smart City", *DEStech Transactions on Engineering and Technology Research*, no., 2017. Available: 10.12783/dtetr/mcee2017/15752.