

Group index : 18

Project name : 新竹曾好停

Team member :

110550089周冠辰 110550083范均宏 110550032文玠敦 110550085房天越 110550117林英碩

Demo 影片連結：[Group 18 demo video](#)

1. An introduction of your application, including why you want to develop the application and the main functions of your application.

網站網址：<https://dbproject.dowfe73fujisir.amplifyapp.com/>

新竹曾好停

目前僅支援新竹市區範圍

自行輸入地點或者使用預設選項(若搜尋的地點為連鎖店請一併輸入分店名)

請選擇交通工具：☒ 汽車 ☐ 機車

輸入新竹市內地點：

預設選項：



停車場資訊 (顯示最近五個) :

停車場名稱	地址	營業時間	平日價格	假日價格	剩餘汽車停車位	剩餘機車停車位	距離(公里)
遠東巨城購物中心停車場	新竹市中央路229號	每日08:00-24:00	汽車：60元/H、機車：免費	汽車：60元/H、機車：免費	1390	0	0.0
新竹民權路停車場	新竹市東區民權路216巷2號旁空地	每日00:00-24:00(24H)	汽車：20元/H(第1小時20元)	汽車：20元/H(第1小時20元)	78	0	0.2
新竹錦華1停車場	新竹市東區錦華街13巷6號	每日00:00-24:00(24H)	汽車：40元/H(24H150元)	汽車：40元/H(24H150元)	70	0	0.2
新竹錦華2停車場	新竹市東區錦華街21巷5號	每日00:00-24:00(24H)	汽車：40元/H(24H150元)	汽車：40元/H(24H150元)	63	0	0.3
東大捷橋下(中央路)停車場	新竹市東大捷橋下(中央路-民族路)	每日00:00-24:00(24H)	汽車：20元/H、限高180cm	汽車：20元/H、限高180cm	39	0	0.3

新竹明志書院停車場

汽車剩餘停車位統計表(7日內各時段平均):



機車剩餘停車位統計表(7日內各時段平均):



我們製作的是一個可以查詢新竹市剩餘停車位的網站，主要是基於新竹市政府即時更新的停車場資訊來製作。因為我們覺得在新竹找停車位是一件非常困難的事情，無論是汽車還是機車，所以每當要出門的時候，不管是去逛街還是去辦事，都能利用這個網站查詢該地點附近的剩餘停車位，非常便利。網站的功能有二，自行輸入新竹市內地點查詢，或是利用我們提前建好的下拉式選單來選擇目的地，然後選擇汽車/機車，網站會印出有剩餘車位且離目標地點距離最近的五個停車場資訊。而我們的選單內還有熱門景點，當遭遇選擇障礙時，不妨看看其他人都選了什麼。

除此之外，我們還另外將各個停車場在過去七天各個時段的剩餘車位做成可供查詢的圖表，若是想要提前規劃行程，便可從圖表中看出哪些時段特別熱門，哪些時段則高機率有車位。

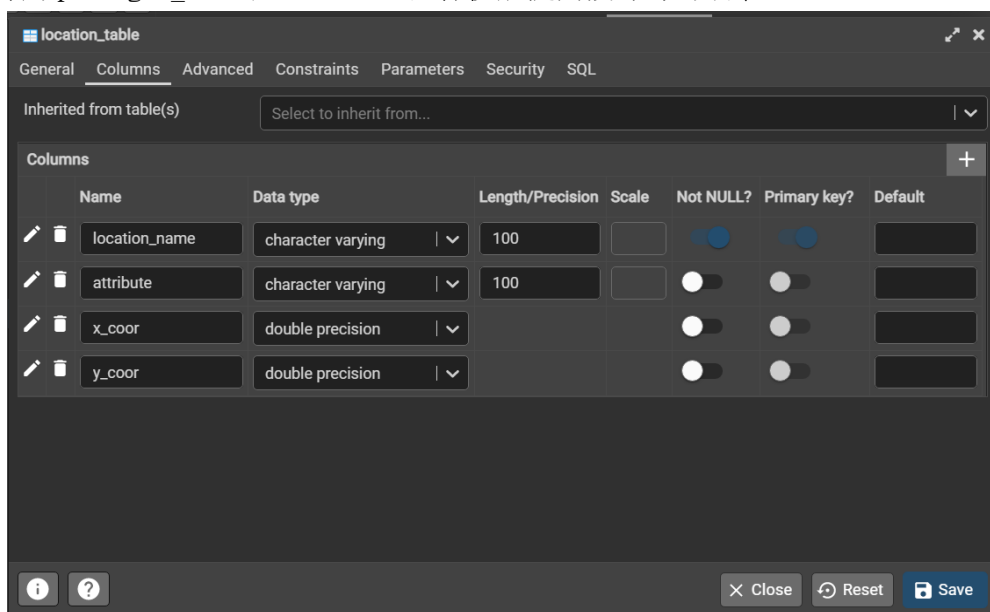
2. Database design - describe the schema of all your tables in the database, including keys and index, if applicable (why you need the keys, or why you think that adding an index is or is not helpful).

我們有八個 table，分別為：location_table, location_with_parkinglot, park_info, park_info_realtime, park_info_realtime_old, park_location, park_name, history_search。

以下分別說明各個table的內容：

- (1) location_table: location_name(pk), attribute, x_coor, y_coor
- (2) location_with_parkinglot: park_id(pk), location_name(pk), distance
- (3) park_info: park_id(pk), bussiness_hour, weekday_cost, holiday_cost, totalspace_car, totalspace_mot, x_coor, y_coor
- (4) park_info_realtime: park_id(pk), update_time, freespace_car, freespace_mot
- (5) park_info_realtime_old: park_id(pk), update_time(pk), freespace_car, freespace_mot
- (6) park_location: park_id(pk), parkinglot_location
- (7) park_name: park_id(pk), parkinglot_name(index)
- (8) history_search: location_name(pk), search_time(pk)

我們絕大多數的資料查詢都是依據 pk 或 parkinglot_name，而 pk 會自動建立 index，因此我們只額外對 parkinglot_name 建立 index。這樣便能提高搜尋時的效率。



location_with_parkinglot

General

Columns

Advanced

Constraints

Parameters

Security

SQL

Inherited from table(s)

Select to inherit from...

Columns

	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
	location_name	character varying	100				
	park_id	integer					
	distance	double precision					

Close

Reset

Save

park_info

General

Columns

Advanced

Constraints

Parameters

Security

SQL

Columns

	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
	park_id	integer					
	business_hour	character varying	100				
	weekday_cost	character varying	100				
	holiday_cost	character varying	100				
	totalspace_car	integer					
	totalspace_mot	integer					
	x_coor	double precision					
	y_coor	double precision					

Close

Reset

Save

park_info_realtime

General

Columns

Advanced

Constraints

Parameters

Security

SQL

Inherited from table(s)

Select to inherit from...

Columns

+

Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
<div><div></div><div></div><div>park_id</div></div>	<div>integer</div> <div></div>			<div></div>	<div></div>	
<div><div></div><div></div><div>freespace_car</div></div>	<div>integer</div> <div></div>			<div></div>	<div></div>	
<div><div></div><div></div><div>freespace_mot</div></div>	<div>integer</div> <div></div>			<div></div>	<div></div>	
<div><div></div><div></div><div>update_time</div></div>	<div>timestamp without tim...</div> <div></div>			<div></div>	<div></div>	

i

?

Close

Reset

Save

park_info_realtime_old

General

Columns

Advanced

Constraints

Parameters

Security

SQL

Inherited from table(s)

Select to inherit from...

Columns

+

Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
<div><div></div><div></div><div>park_id</div></div>	<div>integer</div> <div></div>			<div></div>	<div></div>	
<div><div></div><div></div><div>freespace_car</div></div>	<div>integer</div> <div></div>			<div></div>	<div></div>	
<div><div></div><div></div><div>freespace_mot</div></div>	<div>integer</div> <div></div>			<div></div>	<div></div>	
<div><div></div><div></div><div>update_time</div></div>	<div>timestamp without tim...</div> <div></div>			<div></div>	<div></div>	

i

?

Close

Reset

Save

history_search

General Columns Advanced Constraints Parameters Security SQL

Inherited from table(s) Select to inherit from... | v

Columns +

Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
location_name	character varying	100		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
search_time	timestamp without time zone			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	

i ? Close Reset Save

3. Database design - describe the normal form of all your tables. If the tables are not in BCNF, please include the reason for it (performance trade-off, etc.).

- (1) location_table: BCNF, FD: location_name \rightarrow (attribute, x_coor, y_coor)
- (2) location_with_parkinglot: BCNF, FD: (park_id, location_name) \rightarrow distance
- (3) park_info: BCNF, FD: park_id \rightarrow all
- (4) park_info_realtime: BCNF, FD: park_id \rightarrow all
- (5) park_info_realtime_old: BCNF, FD: (park_id, update_time) \rightarrow all
- (6) park_location: BCNF, FD: park_id(pk) \rightarrow parkinglot_location,
parkinglot_location(superkey) \rightarrow park_id
- (7) park_name: BCNF, FD: park_id(pk) \rightarrow parkinglot_name, parkinglot_name(superkey) \rightarrow park_id
- (8) history_search: BCNF (two columns are all pk)

所以我們設計的table皆符合BCNF的規範。

4. From the data sources to the database - describe the data source and the original format.

我們停車場的即時資訊是從政府的開源網站抓下來的，附上連結供參考：

<https://data.gov.tw/dataset/129136>

裡面為新竹市內各停車場的名稱、地址、座標、車位數以及更新日期等資訊。

<https://hispark.hccg.gov.tw/OpenData/GetParkInfo>

而我們也自己建了一張關於在新竹市有哪些地方可以去的資料，尤其是針對清交的學生。

有食物、購物、娛樂、交通等方面。下方為google 試算表的連結。

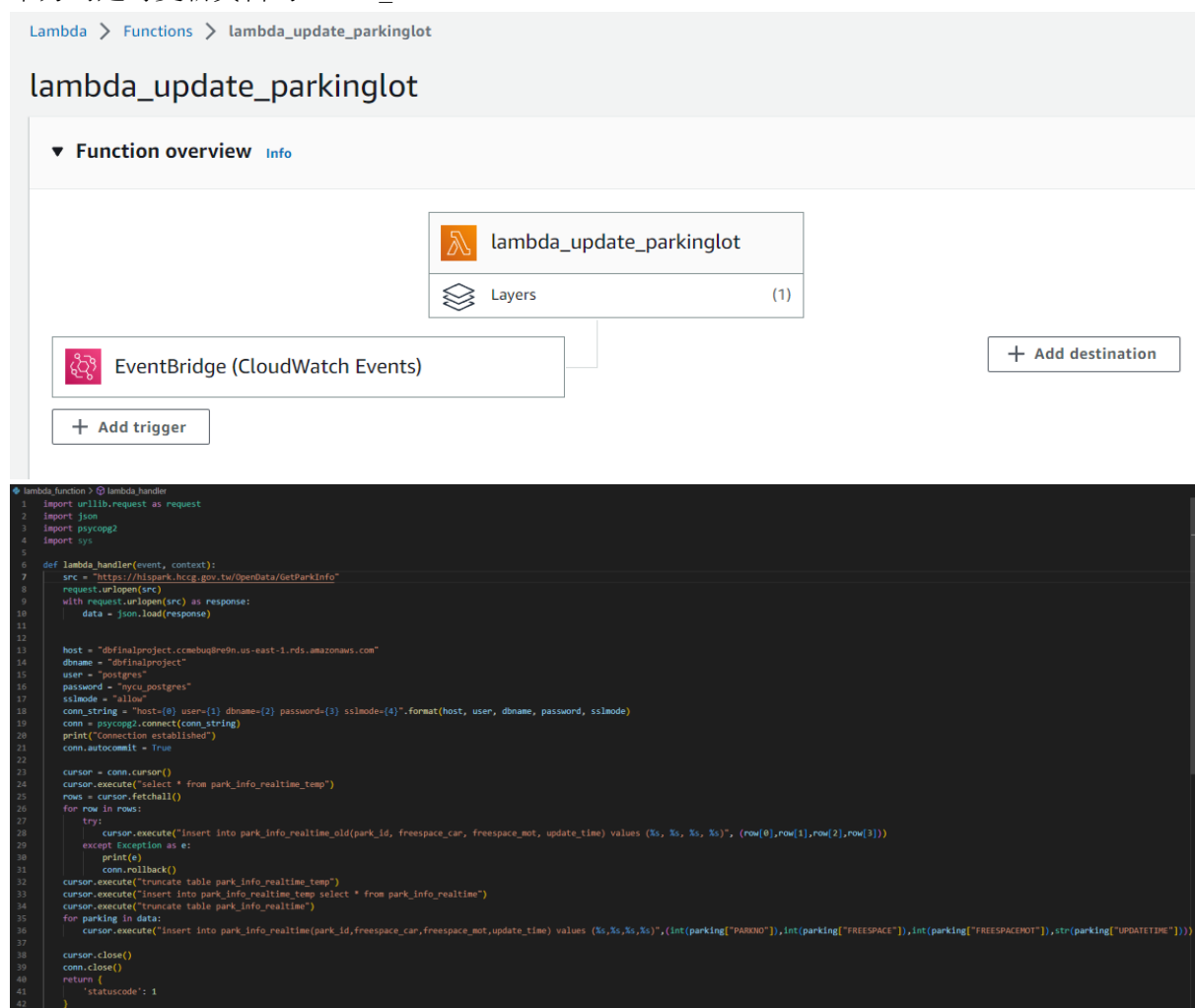
<https://docs.google.com/spreadsheets/d/1hcWg1JemjSRLghbpbEea88oJ1R0GqywQjplmm-7IseA/edit?usp=sharing>

5. From the data sources to the database - describe the methods of importing the original data to your data base and strategies for updating the data, if you have one.

我們的資料分為兩種，一種是不會隨著時間而有所變動，或是在短時間內不會變動的，如停車場的名稱、地址、經緯度、以及我們自己建立的新竹景點表等；而另一種則是隨時間變動的，如剩餘的機車和汽車停車位等。

對於前者，我們將其建表後，轉為.csv檔，匯入資料庫中；對於後者，我們則是使用 AWS Lambda Function 結合 CloudWatch，以達成定時更新資料的功能。我們每十分鐘會更新一次停車場的即時資訊，而每天則會固定一次將七天以前的舊資料給刪除。

下方為定時更新資料的lambda_function



The screenshot displays the AWS Lambda console for the function `lambda_update_parkinglot`. The function overview shows it is triggered by an EventBridge (CloudWatch Events) rule. The code is as follows:

```
1 import urllib.request as request
2 import json
3 import psycopg2
4 import sys
5
6 def lambda_handler(event, context):
7     src = "https://hispark.hccg.gov.tw/OpenData/GetParkInfo"
8     request.urlopen(src)
9     with request.urlopen(src) as response:
10         data = json.load(response)
11
12     host = "dbfinalproject.ccmbug9re9n.us-east-1.rds.amazonaws.com"
13     dbname = "dbfinalproject"
14     user = "postgres"
15     password = "mycu_postgres"
16     sslmode = "allow"
17     conn_string = "host={0} user={1} dbname={2} password={3} sslmode={4}".format(host, user, dbname, password, sslmode)
18     conn = psycopg2.connect(conn_string)
19     print("Connection established")
20     conn.autocommit = True
21
22     cursor = conn.cursor()
23     cursor.execute("select * from park_info_realtime_temp")
24     rows = cursor.fetchall()
25     for row in rows:
26         try:
27             cursor.execute("insert into park_info_realtime_old(park_id, freespace_car, freespace_mot, update_time) values (%s, %s, %s, %s)", (row[0], row[1], row[2], row[3]))
28         except Exception as e:
29             print(e)
30             conn.rollback()
31     cursor.execute("truncate table park_info_realtime_temp")
32     cursor.execute("insert into park_info_realtime select * from park_info_realtime")
33     cursor.execute("truncate table park_info_realtime")
34     for parking in data:
35         cursor.execute("insert into park_info_realtime(park_id, freespace_car, freespace_mot, update_time) values (%s, %s, %s, %s)", (int(parking["PARKNO"]), int(parking["FREESPACE"]), int(parking["FREESPACEEND"]), str(parking["UPDATETIME"])))
36
37     cursor.close()
38     conn.close()
39     return {
40         'statusCode': 1
41     }
```

(code都會附在另一個檔案)

下方為定時刪除資料的lambda_function

Lambda > Functions > lambda_delete_old_data

lambda_delete_old_data

▼ Function overview Info

lambda_delete_old_data

Layers (0)

EventBridge (CloudWatch Events)

+ Add trigger

+ Add destination

```
lambda_function > lambda_handler
1  import urllib.request as request
2  import json
3  import psycopg2
4  import sys
5  import datetime
6
7  def lambda_handler(event, context):
8      host = "dbfinalproject.ccmebuq8re9n.us-east-1.rds.amazonaws.com"
9      dbname = "dbfinalproject"
10     user = "postgres"
11     password = "nycu_postgres"
12     sslmode = "allow"
13     conn_string = "host={0} user={1} dbname={2} password={3} sslmode={4}".format(host, user, dbname, password, sslmode)
14     conn = psycopg2.connect(conn_string)
15     print("Connection established")
16     conn.autocommit = True
17     cursor = conn.cursor()
18     current_time = datetime.datetime.now()
19     weekago = current_time - datetime.timedelta(days = 7)
20     temp = weekago.strftime("%Y-%m-%d %H:%M:%S")
21     print(temp)
22     cursor.execute("delete from park_info_realtime_old where update_time < '" + temp + "'")
23     cursor.close()
24     conn.close()
25     return {
26         "statusCode": 200,
27     }
```

(code都會附在另一個檔案)

而location_with_parkinglot這個table裡面是存我們預設景點與每個停車場的距離，是另外用query建的。

```
insert into location_with_parkinglot
select t0.location_name, t1.park_id ,
(round(( 6378.138 * 2 * asin( sqrt ( pow ( sin ( ( t0.x_coor * PI() / 180 - t1.x_coor * PI() / 180 ) / 2) , 2)
+ COS(t0.x_coor * PI() / 180) * COS(t1.x_coor * PI() / 180) * POW(
SIN( ( t0.y_coor * PI() / 180 - t1.y_coor * PI() / 180 ) / 2),2
))) * 1000 ))) AS DISTANCE
from location_table t0 , park_info t1
```

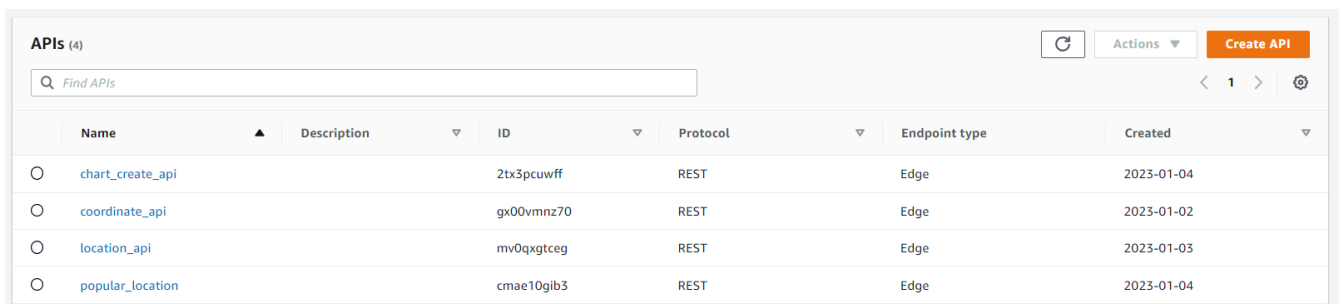

6. Application with database - explain why your application needs a database.

整個新竹市相當龐大，停車場和景點的資料更是多不勝數，因此，我們所需要的資料數量可說是相當龐大，而其中更有許多資料是會隨著時間改變的。若要隨時記錄並且更新這些資料，想必不是一件容易的事，因此這時使用資料庫便可說是一個極佳的選擇。透過資料庫的使用，我們能夠將這些繁雜的資料整理地井然有序，也能夠透過對資料庫的 query，達成在極短時間內取得所需資料的目的，另外，透過 AWS 的 Lambda 和 CloudWatch 功能，我們也能時刻更新所需的資料，並且根據這些最新的資料做出最佳的選擇。綜上所述，我們便能了解，要實踐這些構想，資料庫絕對是我們的不二之選。

利用database，我們便能將停車場資料與我們自己的資料結合，同時還能將停車場過去的資料分析並統計，讓使用者能夠預期將來的停車場空位數。

7. Application with database - includes the queries that are performed by your application, how your application performed these queries (connections between application and database), and what is the cooperating functions for your application.

我們的網站總共會利用到四個在AWS上面架設的API，每一個API各自會去呼叫一個lambda function，而每個lambda function則使用python各自連接到database上面進行query並回傳結果到網頁上。

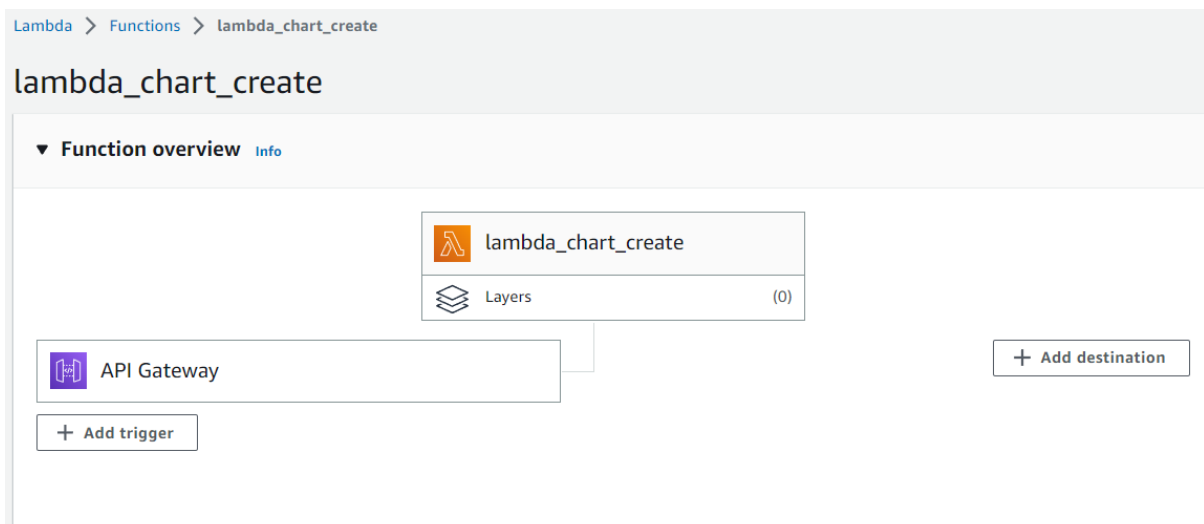


Name	Description	ID	Protocol	Endpoint type	Created
chart_create_api		2tx3pcuwff	REST	Edge	2023-01-04
coordinate_api		gx00vmnz70	REST	Edge	2023-01-02
location_api		mv0qxgtceg	REST	Edge	2023-01-03
popular_location		cmae10gib3	REST	Edge	2023-01-04

(1).將選定的停車場過去七天資料各時段的平均值傳給網站的API，讓我們的網頁依回傳的資料繪製圖表。

API: chart_create_api

lambda_function: lambda_chart_create



```

lambda_function > ...
1  import urllib.request as request
2  import json
3  import psycopg2
4  import sys
5  from datetime import datetime
6
7
8  def lambda_handler(event, context):
9      host = "dbfinalproject.ccmebuq8re9n.us-east-1.rds.amazonaws.com"
10     dbname = "dbfinalproject"
11     user = "postgres"
12     password = "nycu_postgres"
13     sslmode = "allow"
14     conn_string = "host={0} user={1} dbname={2} password={3} sslmode={4}".format(host, user, dbname, password, sslmode)
15     conn = psycopg2.connect(conn_string)
16     print("Connection established")
17     conn.autocommit = True
18     cursor = conn.cursor()
19     x = event['parkinglot']
20     str1 = "with update_hours as( "
21     str2 = "select *, extract(hour from update_time) as hours "
22     str3 = "from park_info_realtime_old), "
23     str4 = "park_avg as( "
24     str5 = "select park_id ,round(avg(freespace_car)) as avgfreecar, round(avg(freespace_mot)) as avgfreemotor ,hours "
25     str6 = "from update_hours "
26     str7 = "group by hours ,park_id ) "
27     str8 = "select avgfreecar, avgfreemotor , totalspace_car, totalspace_mot , hours "
28     str9 = "from park_avg natural join park_info natural join park_name "
29     str10 = "where park_name = '"
30     str11 = "' order by hours "
31     cursor.execute(str1 + str2 + str3 + str4 + str5 + str6 + str7 + str8 + str9 + str10 + x + str11)
32     rows = cursor.fetchall()
33     temp = [[0 for col in range(len(rows[0]))] for row in range(len(rows))]
34     for i in range(len(rows)):
35         for j in range(len(rows[0])):
36             if type(rows[i][j]) == datetime:
37                 temp[i][j] = rows[i][j].strftime("%Y-%m-%d %H:%M:%S")
38             else:
39                 temp[i][j] = rows[i][j]
40     cursor.close()
41     conn.close()
42     return {
43         "statusCode": 200,
44         "query": temp
45     }

```

實際執行的query:

```

with update_hours as(
select *, extract(hour from update_time) as hours
from park_info_realtime_old
),
park_avg as(
select park_id ,round(avg(freespace_car)) as avgfreecar, round(avg(freespace_mot)) as avgfreemotor ,hours
from update_hours
group by hours ,park_id
)
select avgfreecar, avgfreemotor , totalspace_car, totalspace_mot , hours
from park_avg natural join park_info natural join park_name
where park_name = 'THE PLACE YOU CHOOSE'
order by hours

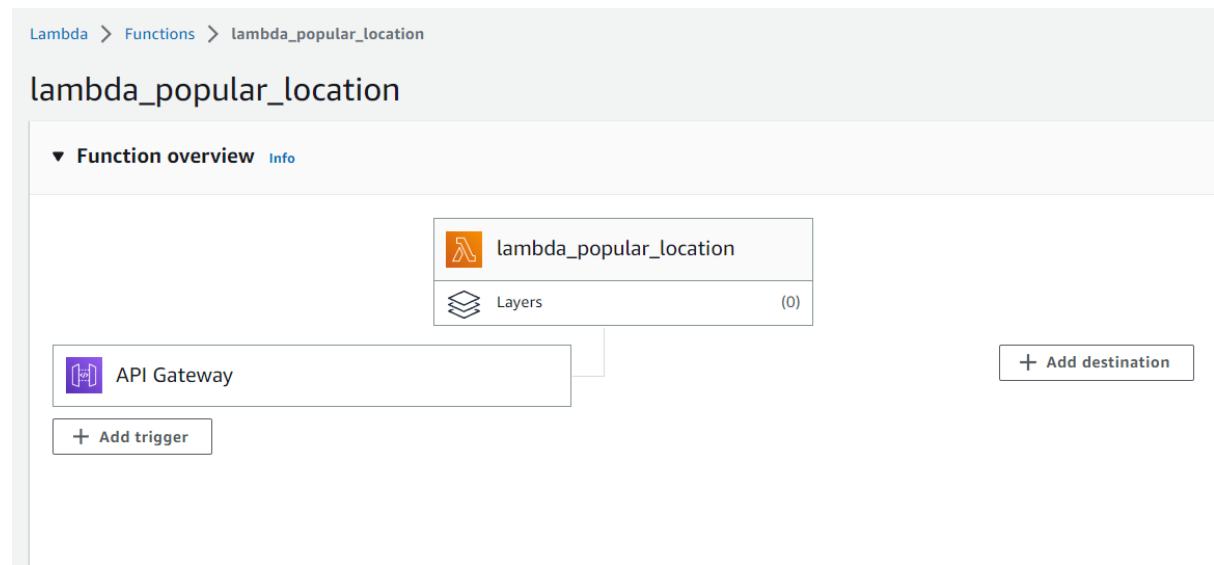
```

由於我們的database會定時將超過七天以上的停車場資料刪除，所以不需特別設定範圍。

(2).將最熱門的五項景點傳給網站的API，讓我們的網站上的熱門景點選項能正常運作。

API: popular_location

lambda_function: lambda_popular_location



```
lambda_function > ...
1  import urllib.request as request
2  import json
3  import psycopg2
4  import sys
5  from datetime import datetime
6
7
8
9  def lambda_handler(event, context):
10     host = "dbfinalproject.ccmebuq8re9n.us-east-1.rds.amazonaws.com"
11     dbname = "dbfinalproject"
12     user = "postgres"
13     password = "nycu_postgres"
14     sslmode = "allow"
15     conn_string = "host={0} user={1} dbname={2} password={3} sslmode={4}".format(host, user, dbname, password, sslmode)
16     conn = psycopg2.connect(conn_string)
17     print("Connection established")
18     conn.autocommit = True
19     cursor = conn.cursor()
20     str1 = 'select location_name ,count(location_name) as counting '
21     str2 = 'from history_search '
22     str3 = 'group by location_name '
23     str4 = 'order by counting desc '
24     str5 = 'limit 5 '
25     cursor.execute(str1 + str2 + str3 + str4 + str5)
26     rows = cursor.fetchall()
27     temp = [[0 for col in range(len(rows[0]))] for row in range(len(rows))]
28     for i in range(len(rows)):
29         for j in range(len(rows[0])):
30             if type(rows[i][j]) == datetime:
31                 temp[i][j] = rows[i][j].strftime("%Y-%m-%d %H:%M:%S")
32             else:
33                 temp[i][j] = rows[i][j]
34     cursor.close()
35     conn.close()
36     return {
37         "statusCode": 200,
38         "query": temp
39     }
```

```
select location_name ,count(location_name) as counting
from history_search
group by location_name
order by counting desc
limit 5|
```

(3).將從google map API 獲得的座標資訊傳入lambda_function，並回傳符合條件的五筆停車場資料的API，而我們的網站則將此五筆資料做成表格顯示出來。


API: coordinate_api


lambda_function: lambda_find_parkinglot_distance


Lambda > Functions > lambda_find_parkinglot_distance


lambda_find_parkinglot_distance

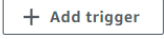
▼ Function overview Info

 lambda_find_parkinglot_distance

 Layers (0)

 API Gateway





lambda_function > lambda_handler

```
1 import urllib.request as request
2 import json
3 import psycopg2
4 import sys
5 from datetime import datetime
6
7
8 def lambda_handler(event, context):
9     host = "dbfinalproject.ccmebuq8re9n.us-east-1.rds.amazonaws.com"
10    dbname = "dbfinalproject"
11    user = "postgres"
12    password = "nycu_postgres"
13    sslmode = "allow"
14    conn_string = "host={0} user={1} dbname={2} password={3} sslmode={4}".format(host, user, dbname, password, sslmode)
15    conn = psycopg2.connect(conn_string)
16    print("Connection established")
17    conn.autocommit = True
18    cursor = conn.cursor()
19    x = event['x_coor']
20    y = event['y_coor']
21    z = event['vehicle_type']
22    str1 = ("select *,round(( 6378.138 * 2 * asin( sqrt ( pow ( sin ( ( " + str(x) + " * PI() / 180 - t1.x_coor * PI() / 180) / 2) , 2) " )
23    str2 = ("          + COS( " + str(x) + " * PI() / 180) * COS(t1.x_coor * PI() / 180) * POW( " )
24    str3 = ("          SIN( ( " + str(y) + " * PI() / 180 - t1.y_coor * PI() / 180) / 2),2 " )
25    str4 = ("          ))) * 1000 )) AS DISTANCE ")
26    str5 = ("FROM park_info as t1 natural join park_location natural join park_name natural join park_info_realtime ")
27    str6 = ("order by DISTANCE limit 5 ")
28    if z == "motor":
29        str7 = (" where freespace_mot > 0 ")
30    elif z == "car":
31        str7 = (" where freespace_car > 0 ")
32    else:
33        str7 = " "
34    print(str1 + str2 + str3 + str4 + str5 + str6)
35    cursor.execute(str1 + str2 + str3 + str4 + str5 + str7 + str6)
36    rows = cursor.fetchall()
37    temp = [[0 for col in range(len(rows[0]))] for row in range(len(rows))]
38    for i in range(len(rows)):
39        for j in range(len(rows[0])):
40            if type(rows[i][j]) == datetime:
41                temp[i][j] = rows[i][j].strftime("%Y-%m-%d %H:%M:%S")
42            else:
43                temp[i][j] = rows[i][j]
44    cursor.close()
45    conn.close()
46    return {
47        "statusCode": 200,
48        "query": temp
49    }
```

```

select * ,(round(( 6378.138 * 2 *
    asin( sqrt ( pow ( sin (
        ( INPUT_DATA) * PI() / 180 - t1.x_coor * PI() / 180) / 2) , 2)
    + COS((INPUT_DATA) * PI() / 180) * COS(t1.x_coor * PI() / 180) * POW(
        SIN( ( INPUT_DATA) * PI() / 180 - t1.y_coor * PI() / 180) / 2),2
    ))*1000 ))) / 1000 AS DISTANCE
FROM park_info as t1 natural join park_location natural join park_name natural join park_info_realtime
where freespace_car > 0
order by DISTANCE
limit 5

```

((INPUT_DATA)及(freespace_car)依輸入值變化)

(4).根據選擇好的景點回傳五筆停車場資料給網頁的API，而我們的網站則將此五筆資料做成表格顯示出來。


API: location_api

lambda_function: lambda_location_to_parkinglot

Lambda > Functions > lambda_location_to_parkinglot

lambda_location_to_parkinglot

▼ Function overview [Info](#)

 lambda_location_to_parkinglot

 Layers (0)

 API Gateway

[+ Add trigger](#)

[+ Add destination](#)

```

lambda_function > ...
1 import urllib.request as request
2 import json
3 import psycopg2
4 import sys
5 from datetime import datetime
6
7
8 def lambda_handler(event, context):
9     host = "dbfinalproject.cmebuq8re9n.us-east-1.rds.amazonaws.com"
10    dbname = "dbfinalproject"
11    user = "postgres"
12    password = "nycu_postgres"
13    sslmode = "allow"
14    conn_string = "host={0} user={1} dbname={2} password={3} sslmode={4}".format(host, user, dbname, password, sslmode)
15    conn = psycopg2.connect(conn_string)
16    print("Connection established")
17    conn.autocommit = True
18    cursor = conn.cursor()
19    x = event['location']
20    z = event["vehicle_type"]
21    current_time = datetime.now()
22    str1 = "insert into history_search(location_name,search_time) values("
23    str2 = ","
24    str3 = current_time.strftime("%Y-%m-%d %H:%M:%S")
25    str4 = ")"
26    cursor.execute(str1+x+str2+str3+str4)
27    str1 = "with parkinglot as ( "
28    str2 = "select * from park_info natural join park_location natural join park_name natural join park_info_realtime )"
29    str3 = "select park_name , park_location , distance , business_hour , weekday_cost, holiday_cost, totalspace_car, totalspace_mot, freespace_car, freespace_mot, update_time "
30    str4 = "from (location_with_parkinglot natural join location_table) inner join parkinglot "
31    str5 = "on parkinglot.park_id = location_with_parkinglot.park_id "
32    str6 = "where location_name = "
33    str7 = ""
34    str8 = " order by distance limit 5 "
35    if z == "motor":
36        str7 = (" and freespace_car > 0 ")
37    elif z == "car":
38        str7 = (" and freespace_car > 0 ")
39    else:
40        str7 = (" and freespace_car > 0 ")
41    print(str1 + str2 + str3 + str4 + str5 + str6 + x + str7 + str8)
42    cursor.execute(str1 + str2 + str3 + str4 + str5 + str6 + x + str7 + str8)
43    rows = cursor.fetchall()
44    temp = [[0 for col in range(len(rows[0]))] for row in range(len(rows))]
45    for i in range(len(rows)):
46        for j in range(len(rows[0])):
47            if type(rows[i][j]) == datetime:
48                temp[i][j] = rows[i][j].strftime("%Y-%m-%d %H:%M:%S")
49            else:
50                temp[i][j] = rows[i][j]
51    cursor.close()
52    conn.close()
53    return {
54        "statusCode": 200,
55        "query": temp
56    }

```

```

with parkinglot as
(
select *
from park_info natural join park_location natural join park_name natural join park_info_realtime
)
select park_name , park_location , distance , business_hour , weekday_cost, holiday_cost,
totalspace_car, totalspace_mot, freespace_car, freespace_mot, update_time
from (location_with_parkinglot natural join location_table) inner join parkinglot
on parkinglot.park_id = location_with_parkinglot.park_id
where location_name = 'THE PLACE YOU CHOOSE'
and freespace_car > 0
order by distance
limit 5

```

8. All the other details of your application that you want us to know.

我們的整個project基本上都是建在AWS上面的，網站主要使用 HTML、CSS、 Javascript 撰寫，利用 AWS Amplify 架設，並且利用Python 抓取資料、AWS RDS 來建構資料庫。要完成網站的功能還用到兩個重要的

API 工具：Geocoding API、Amazon API Gateway，一個是把輸入地點轉成現實經緯度座標來方便我們做距離計算，另一個則是利用寫好的 Lambda Function 來幫助我們的網站與 AWS RDS 資料庫做 Query 互動。我們還幫網站做了兩種色彩設計，所以在電腦的淺色模式跟深色模式看到的設計會不一樣。另外還有一點，如果有其他縣市即時更新的停車場資訊的話，我們的這個網站也可以拓展到其他的縣市去，這說明了這個網站蘊藏著非常大的潛力。

新竹曾好停

目前僅支援新竹市區範圍

自行輸入地點或者使用預設選項(若搜尋的地點為連鎖店請一併輸入分店名)

請選擇交通工具：☒ 汽車 ☐ 機車

輸入新竹市內地點：

預設選項：

停車場資訊 (顯示最近五個)：

停車場名稱	地址	營業時間	平日價格	假日價格	剩餘汽車停車位	剩餘機車停車位	距離(公里)
遠東巨城購物中心停車場	新竹市中央路229號	每日08:00~24:00	汽車：60元/H，機車：免費	汽車：60元/H，機車：免費	1390	0	0.0
新竹民權路停車場	新竹市東區民權路216巷2號旁空地	每日00:00~24:00(24H)	汽車：20元/H(第1小時20元)	汽車：20元/H(第1小時20元)	78	0	0.2
新竹錦華1停車場	新竹市東區錦華街13巷6號	每日00:00~24:00(24H)	汽車：40元/H(24H150元)	汽車：40元/H(24H150元)	70	0	0.2
新竹錦華2停車場	新竹市東區錦華街21巷4號	每日00:00~24:00(24H)	汽車：40元/H(24H150元)	汽車：40元/H(24H150元)	63	0	0.3
東大陸橋下(中央站)停車場	新竹市東大陸橋下(中央路-民族路)	每日00:00~24:00(24H)	汽車：20元/H，限高180cm	汽車：20元/H，限高180cm	39	0	0.3

(上圖為深色模式，保護每位使用者的眼睛)