Test Documentation

Group Link-list

1 Unit tests for API（Automated testing by using Newman）

1.1 Test Brief

We implemented extensive unit testing on the API during the development process to ensure that it can return appropriate results to the user's request. Initially, we used postman's visual interface to manually test the API, and at the same time, we used this to design our automated test points. Then, based on the automatic test points, we use postman to export the json file and use newman for automated testing.

Through unit testing, we simulate user input information (including extreme input data) as much as possible to ensure 100% accuracy, and at the same time, it is convenient for us to find out some problems of programs.

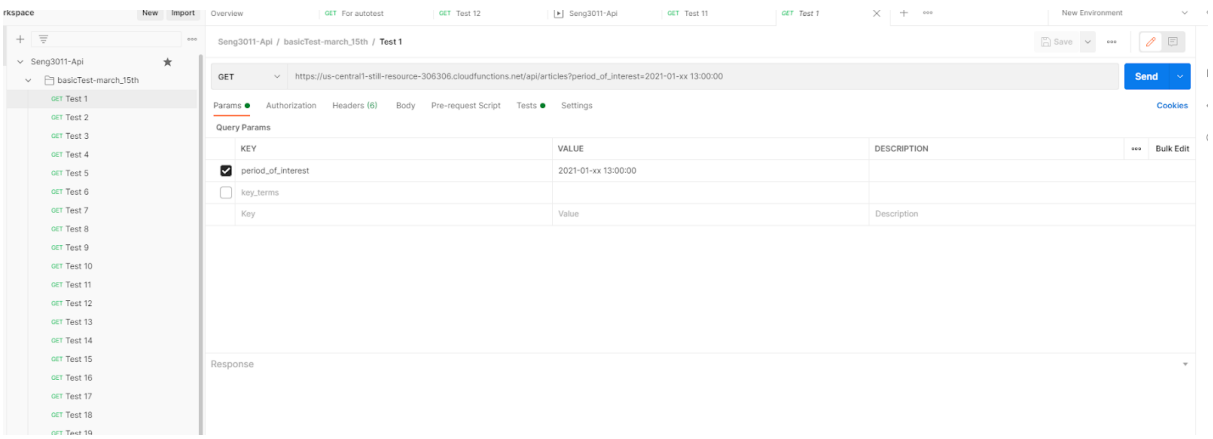
Test environment

All our unit tests are executed on postman or newman. It directly makes a request to the real-time API to ensure that the test results are consistent with the user experience.

Manual test

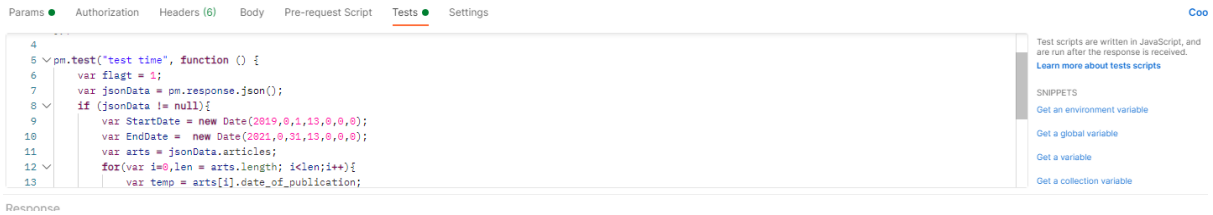
The reason why we do manual testing is that we need to understand the return value of our API and write appropriate automatic testing samples accordingly. At the same time, manual testing can help us to eliminate some problems that are difficult to deal with in automatic testing.

Our manual tests are mainly carried out by postman. It allows us to freely define URLs, request patterns and query parameters. At the same time, it provides script functions for automatic testing and data returned by multiple clock views. We can view the return value of the API from the page and file to simply query the expected output of JSON. Using it, we can simply create one or more requests and specify the corresponding test set in the corresponding test.



Automated test

For automated testing, we use postman's automated support library-newman. By running the json file exported by postman, newman can detect the returned information, just like manual testing. At the same time, since the scraper is running all the time, we can't exactly match the returned information with the expected information like in the local test. In this regard, we only made a simple test on the returned information.



Refer to TestScript/Readme.rd for details on how to run the test

1.2 Test Detail

Test ID	Test point	Test input	Test analysis
1	Basic test-corresponding to a single timestamp	period_of_interest=2021-01-xx xx:xx:xx	Success
2	Corresponding to two timestamps	period_of_interest=2021-01-xx xx:xx:xx to 2021-02-xx xx:xx:xx	Success
3	Corresponds to 1 precise time	period_of_interest=2021-01-02 13:00:00	Success
4	Corresponding to 2 precise time	period_of_interest=2019-01-02 13:00:00 to 2021-03-17	Success
5	Corresponding to 2 precise times (standard)	period_of_interest=2019-01-02 13:00:00 to 2021-03-17 13:00:00	Success
6	Corresponding to 2 precise times (reverse order)	period_of_interest=2021-01-02 13:00:00 to 2019-03-17 13:00:00	Success it return empty
7	Corresponds to 2 precise times, but the time is wrong	period_of_interest=2019-02-30 13:00:00 to 2021-04-30 13:00:00	Return 200,but need 400,it will be fixed later
8	Corresponds to 2 precise times, the first time is wrong	period_of_interest=2019-01-32 to 2021-02-20	Success

9	Corresponds to 2 precise times, the second time is wrong	period_of_interest=2020-01-30 14:00:00 to 2021-02-32 14:00:00	Success
10	Corresponds to 2 precise times, ignoring hours	period_of_interest=2020-01-30 to 2021-03-15	Success
11	Check Extremely close time	period_of_interest=2019-01-11 01:32:15 to 2019-01-11 01:32:17	Success
12	Check Same time	articles?period_of_interest=2019-01-11 01:32:16 to 2019-01-11 01:32:16	Success
13	Add the normal Test -flu of key_term(Fuzzy words)	period_of_interest=2020-01-16 13:00:00 to 2021-01-16 13:00:00&key_terms=flu	Success
14	Abnormal Test-Haemorrhagic Fever with key_term added	period_of_interest=2020-01-16 13:00:00 to 2021-01-16 13:00:00&key_terms=Haemorrhagic Fever	Success
15	Add the normal Test location=China,!Influenza-like illness of key_term	period_of_interest=2020-01-16 13:00:00 to 2021-01-16 13:00:00&location=China,!Influenza-like illness	Success,it return emty
16	Add key_term's normal Test Fever of unknown Origin	period_of_interest=2020-01-16 13:00:00 to 2021-01-16 13:00:00&key_terms=Fever of unknown Origin	Success
17	Normal Test fever with key_term added	period_of_interest=2020-01-16 13:00:00 to 2021-01-16 13:00:00&key_terms=fever	Success

18	Add key_term abnormality Test silence wench(Wrong key_term)	period_of_interest=2020-01-16 13:00:00 to 2021-01-16 13:00:00&key_terms=slience wench	Success,it returns empty
19	Can the test recognize similar diseases -covid-19	period_of_interest=2020-01-16 13:00:00 to 2021-01-16 13:00:00&key_terms=covid-19	Success,It seems that there is no such function before,but now we fix the bug
20	Test multiple keys_terms	period_of_interest=2020-01-16 13:00:00 to 2021-01-16 13:00:00&key_terms=China,fever	Success
21	Test with location	period_of_interest=2020-01-16 13:00:00 to 2021-01-16 13:00:00&location=China	Success

The data of Unit test in Phase_1/TestScripts

Only one bug was found in the final test. Our API may not be able to distinguish the special date of February 30 from the illegal time, but it does not affect the use.

2 Test for Scraper (Manual test)

We test whether our scraper runs successfully by giving the specified web page. First, we find a web page whose information has been manually determined (the amount of information that can be captured is less than 20, so we can simply check it), and then we execute our scraper. When we run our program, it will automatically upload articles and corresponding reports to the cloud server, namely firestore, and then we will screen the data by calling the functions inside and manually, and finally determine whether our data is correct.

Test ID	Test reason	Test analysis

1	Standard CDC web page	Success
2	Standard CDC webpage with accurate event time	Success ,Returned the correct time
3	Standard CDC webpage with accurate event time in the future	Success, The default value is returned

3.Performance Test

We use Stoplight and JMeter for load test. We analyze the load of our program by simulating two scenarios: multiple different requests and multiple user requests at the same time.

L1 When testing the impact on our API based on multiple different requests,

I chose stoplight as my testing software. Through its own preprocessing function, I wrote a program to automatically generate random query time and key terms Script for terms. The lexicon of key terms comes from the disease collection in spec. The test calls randomly generated queries many times.

```
var random = GetRandomNum(0, 85);
var query = locationwords[random];


var period_of_interest = firstTime + " to "+ SencondTime

pm.environment.set('period_of_interest',period_of_interest)

function GetRandomNum(Min, Max) {
    var Range = Max - Min;
    var Rand = Math.random();
    return(Min + Math.round(Rand * Range));
}

var locationwords = ["China","Japan","Us","UK","Asia","Australia","German","India","Haemorrhagic Fever" ,
"Acute Flacid Paralysis" ,
"Acute gastroenteritis" ,
"Acute respiratory syndrome" ,
```

Here is the test information for using the US server.

Test ID	Test Input	Number of requests	Avg Time	Max time

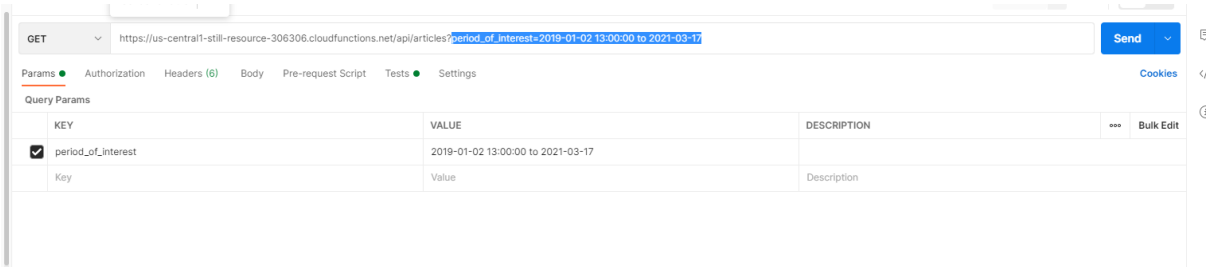
1	Random time period	10	302ms	560ms
2	Random time period	50	310ms	736ms
3	Random time period	150	320ms	1200ms
4	Random time period with random key	10	293ms	461ms
5	Random time period with random key	50	310ms	600ms
6	Random time period with random key	150	311ms	810ms
7	Random time period with random key	200	340ms(have error)	1297ms

Using AU server, a good test time of about 60ms is obtained. However, the Au based server has some small bugs, so it is not completely tested.

As far as testing is concerned, large-scale queries in turn in a short period of time will not affect the running speed of the API too much. Queries less than **150 times** are about the normal load capacity of the API.When the number of requests exceeds 200, some errors will be generated.

P1 Test how long it takes to query the entire database

Input	Time
period_of_interest=2019-01-02 13:00:00 to 2021-03-17	2.26s

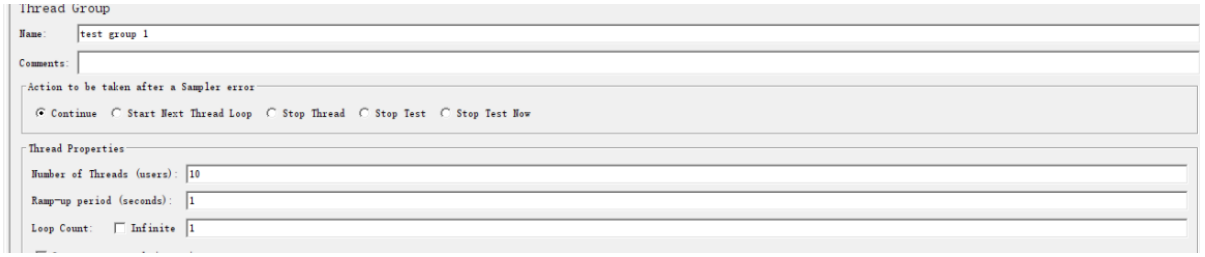


P2 Test how long does it take to query one thing

Input	Time
period_of_interest=2020-01-16 13:00:00 to 2021-01-16 13:00:00&key_terms=China,fever	300ms

L2 Performance under multi person processing.

Since postman only supports single threaded queries, I use JMeter as a tool for multi-threaded queries.In this test, I tested the performance of the API when multiple users access the API at the same time in a short time.However, since the test is conducted overseas, the delay may be large.The actual situation will be much better (almost one in five).



Input	UserNumber	Average time (Including transfer and load)	Error rate
period_of_interest=2019-01-10%2013:00:00%20to%202021-05-20%2005:44:23	10	2393ms	0%
	50	3421ms	0%
	150	4498ms	0%
	180	5600ms	2.2%
	200	7579ms(due to error)	3%
	500	8710ms(due to error)	65%

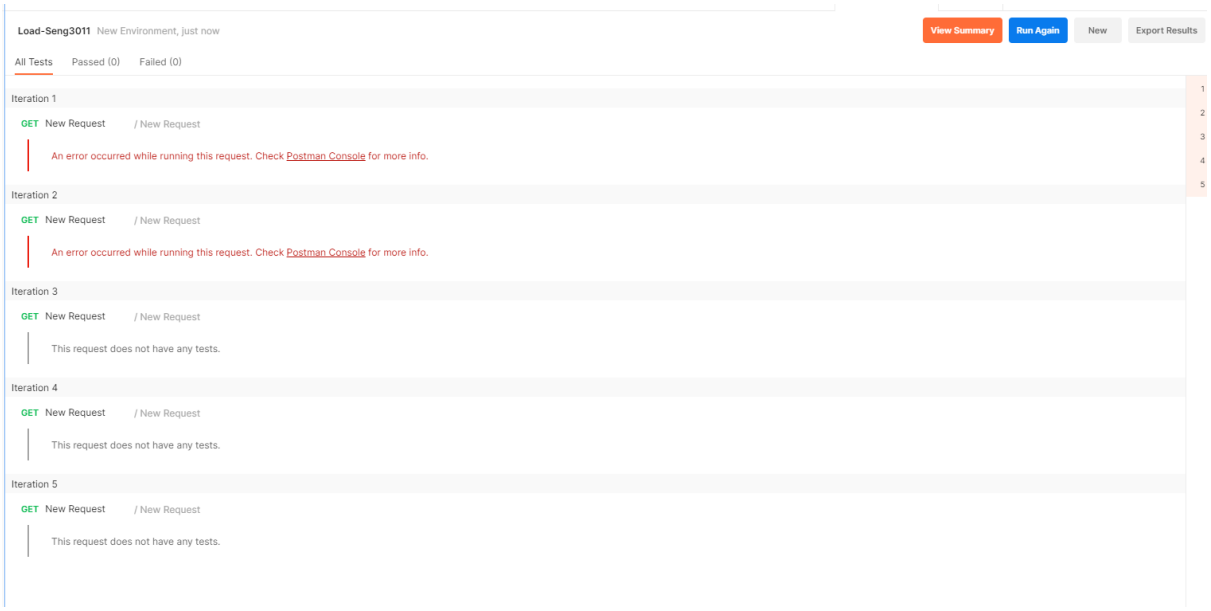
According to the data, the maximum load of API is about **150 users,** and the return speed will decrease with the increase of users.

4.Security testing

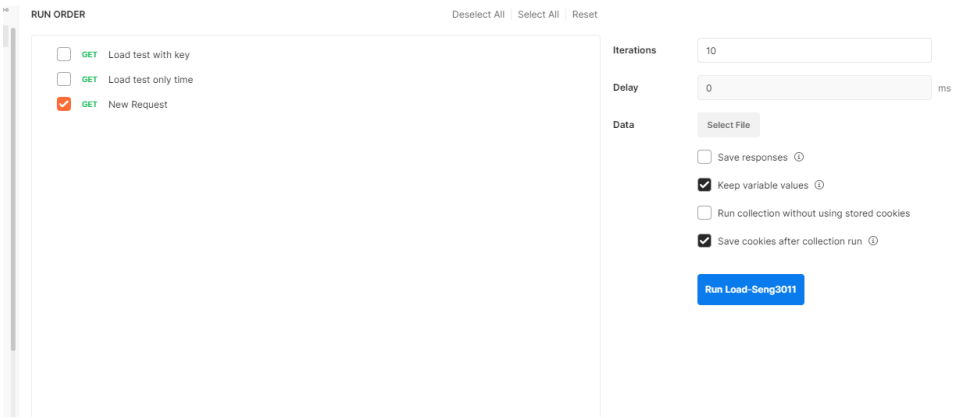
We test API security, on the one hand, to make our users get the right information, on the other hand, make our API stable enough.

Because our API only provides the query service to the database, and does not save any user information in the API, we only test the stability of the server..

Google server has a simple performance against DDoS attack. When I use postman or JMeter to make 10 requests to the remote server in 0.1ms, the server rejected my request.



This can minimize the possibility of DDoS attacks on the server and ensure the stability of the API



150 request

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
Http Request	50	12193	4213	14124	2725.50	0.00%	3.5/sec	4685.50	0.75	1382960.5
TOTAL	50	12193	4213	14124	2725.50	0.00%	3.5/sec	4685.50	0.75	1382960.5

180 request

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
Http Request	180	41769	29051	59240	11097.57	1.11%	3.0/sec	3992.68	0.64	1367628.9
TOTAL	180	41769	29051	59240	11097.57	1.11%	3.0/sec	3992.68	0.64	1367628.9

400 request

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
Http Request	400	40434	14724	85589	20605.46	52.75%	4.7/sec	2975.22	0.47	655115.5
TOTAL	400	40434	14724	85589	20605.46	52.75%	4.7/sec	2975.22	0.47	655115.5

In addition, the data based on the load test shows that when the API receives a large number of visits from the same IP in a short period of time (about 180), the system will automatically reject some requests from this IP.

5.Limitation

First,Our automatic test code does not run automatically with API iteration, and it only provides the test of current functions. If we add new functions, we need to rewrite the automatic test code and run it.

Detecting Scraper is a very troublesome thing, because it is difficult for us to make sure that our crawling information is accurate every time based on CDC's website structure. We can only ensure the accuracy of the information by manually checking the database. At the same time, if we want to ensure that it is accurate, we need a better scraper to provide accurate information, which is like a paradox.

In addition, for security testing, we only tested the API response measures under large-scale traffic attacks. We have no way to detect the attack method of implanting Trojan horses to obtain user information, because our API is only an API for obtaining epidemic information. It does not contain any user information and private information.

The performance test may be biased, because each person may have different physical distances and transmission speeds based on different people's network speeds or different regions, resulting in different delays. We can only try to estimate the internal running time. This gives an expected value.

This test is based on our test server in the United States, but when users use it, we will provide a version based on Australian server in the near future, which will run a little faster.