API Testing

OVERVIEW

ENVIRONMENT

Our testing is mainly conducted using Mac OS, except for the testing tools, other client end tests are using both Safari browser and Chrome browser.

Tools

Postman

Postman is a scalable and powerful API testing tool, which can be used to simplify API testing and make the whole process less manually. It can compare the json output with expectation automatically. It allows users to generate a series of API calls and organise them to collections that are convenient to use later. We use postman for both integration tests and load tests.

Pytest

Pytest is a python built in framework for unit tests. It provides a rich set of tools for constructing and running tests, which includes aggregation of tests. We mainly use this tool to test our basic functions.

LoadView

LoadView is a cloud based real browsers load testing tool. It can simulate common user page interactions and allows us to Define the number of users, user behavior, and duration through custom-built scenarios. We use LoadView to perform load testing and speed testing.

Process

We tested our API during the entire process of development. Before our scraper actually works, we tested each helper function used during scrapping. And then we individually test the scrapper based on a single page that we already generated the result manually. In the meantime the cloud functions are also separately tested before integration. After the whole API works, we use postman to test both normal case and edge case to ensure we covered all the cases. Finally we did the performance tests and security tests.

Limitations

The major limitation of our testing is the uncertainty of the results. Since we scripted the pages automatically and perically, we were unclear of how much results or what results should be returned for each query. It could be more exact if we use a seperate database for testing, but due to time limitation, we finally choose to manually check the returned results are correct, and also compare the result with the query of the database to ensure the completeness of the result.

In addition, since the entire CDC website is huge, it is hard to manually check all the page we needed is scripted. We assume that all the reports in CDC have been scripted and stored in the database.

Correctness Test Unit Test

Helper Function Test

We use Pytest to test the helper functions used by the scraper, which includes functions to extract location(geonames), date, diseases, syndromes for the paragraph of text.

```
~/3011/SENG3011_200OK/PHASE_1/API_SourceCode(combined*) » pytest                    QAQ@bogon
============================= test session starts ==============================
platform darwin -- Python 3.7.3, pytest-6.1.1, py-1.9.0, pluggy-0.13.1
rootdir: /Users/QAQ/3011/SENG3011_200OK/PHASE_1/API_SourceCode
plugins: hypothesis-5.5.4, arraydiff-0.3, remotedata-0.3.2, openfiles-0.4.0, doctestplus-0.5.0,
astropy-header-0.1.2
collected 4 items

helper_test.py ....                                                      [100%]

============================== 4 passed in 4.12s ===============================
```

```python
import helper

main = "As of November 20, 2020, Fever of unknown Origin 18 people infe
main2 = "As of November 20, 2020, 18 people infected crimean-congo haer

def test_get_date():…

def test_get_location():
    result1 = helper.get_locations("asdfas asdfasdfa sdf Shanghai sdfs
    result2 = helper.get_locations(main)
    result3 = helper.get_locations("will continue to work with state pu
    result4 = helper.get_locations("ted. 6 ill people were children int

    assert result1[0] == 1796236
    assert result2[0] == 4801859
    print(result3)
    assert result3[0] == 1796236
    assert len(result3) == 1   # test multiple same location only retu
    assert result4[0] == 1814906
    assert result4[1] == 1796236

def test_get_disease():…

def test_get_syndromes():…
```

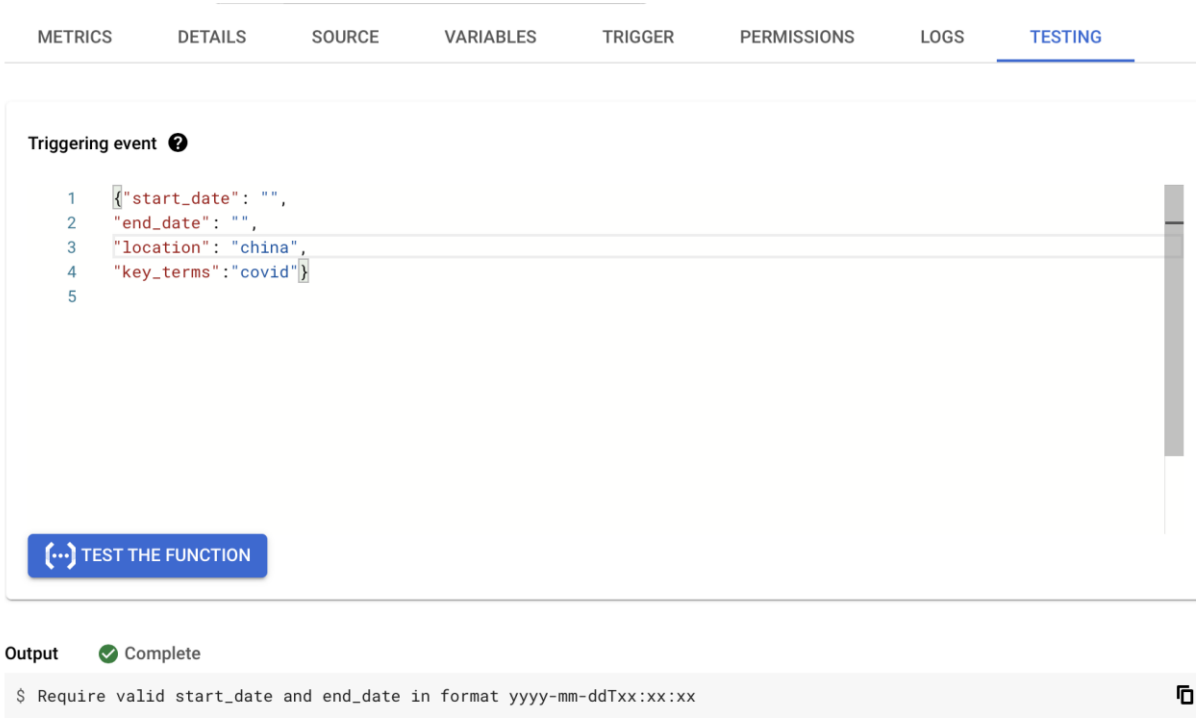
Scraper Test

To test our scraper, we manually create articles for the certain web pages, scrap that page to verify if the result matches what we want.

Cloud Function and Database Test

We also tested our cloud function and the database before integrating the entire API. However, both cloud function and database are tested manually in the Google Cloud Platform Console.

For the cloud function, we use the testing function in Cloud Function to manually verify the Output.



For the database, we use functions to write in or delete data from the schema and check it in the console. And also do some basic query using function and compare the result with the content in the schema.

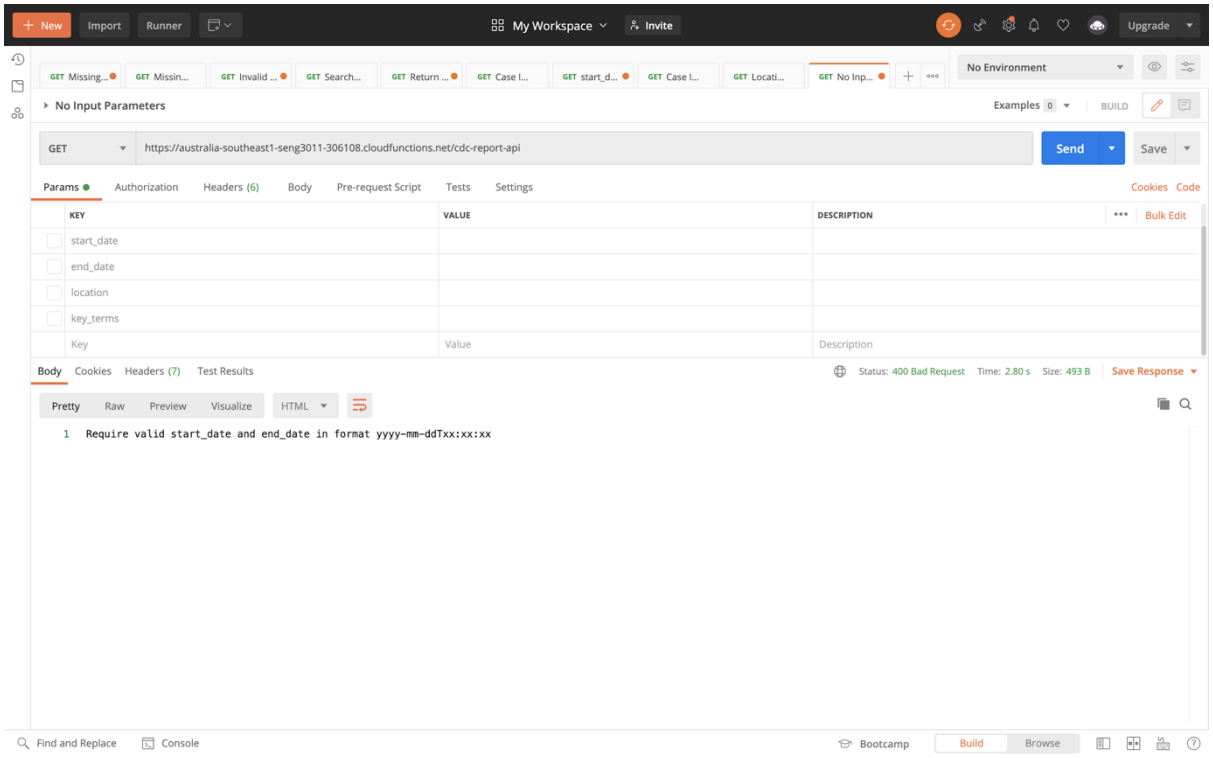
.

Integration Test

For the integration test, we send requests to the API and check the response returned, including status code and response body. It is the black box functional test. Equivalence partitioning, Error guessing, Use Case Analysis and other methods are used for generating these test cases.

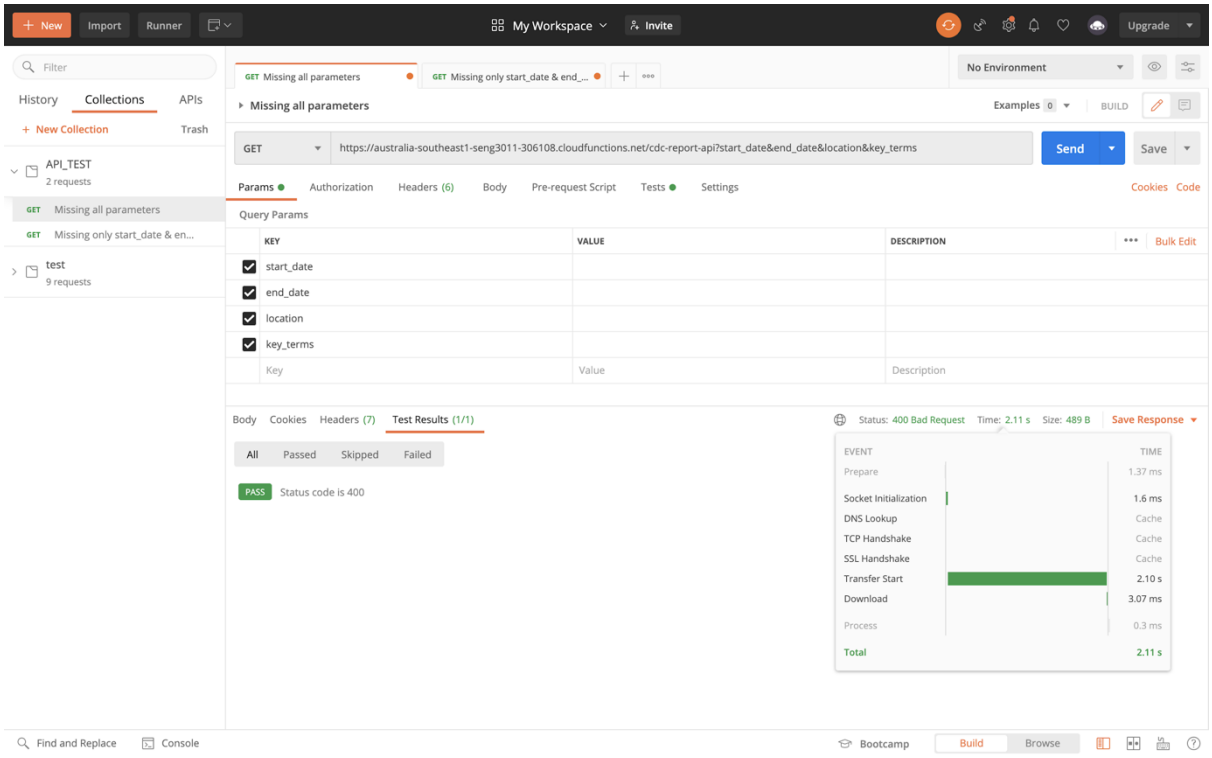
Missing all input parameters

Request Parameters	None
Expected output	The status code returned should be 400 and an error message should also be returned to inform the user
Response time	~2000ms



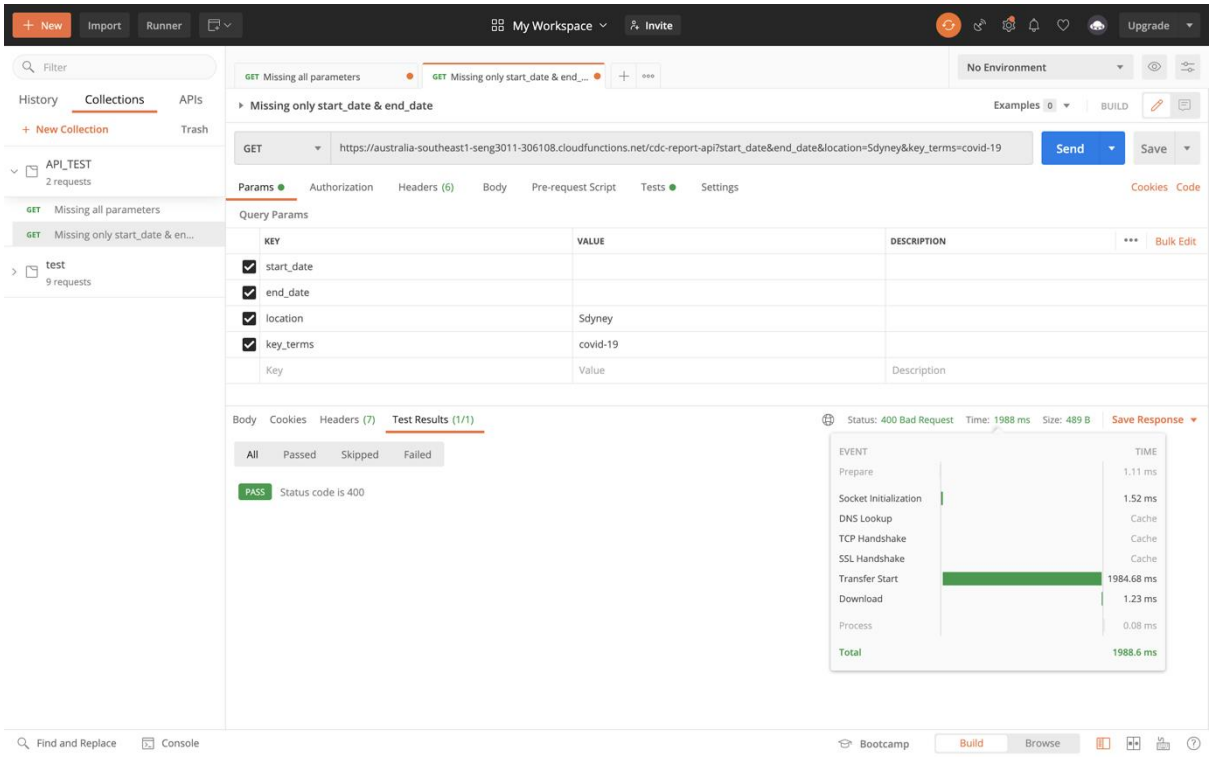
All inputs are empty

Input Parameters	• - start_date: “” • - end_date: “” • - location: “” • - key_terms: “”
Expected output	The status code returned should be 400 and an error message should also be returned to inform the user
Response time	~2000ms



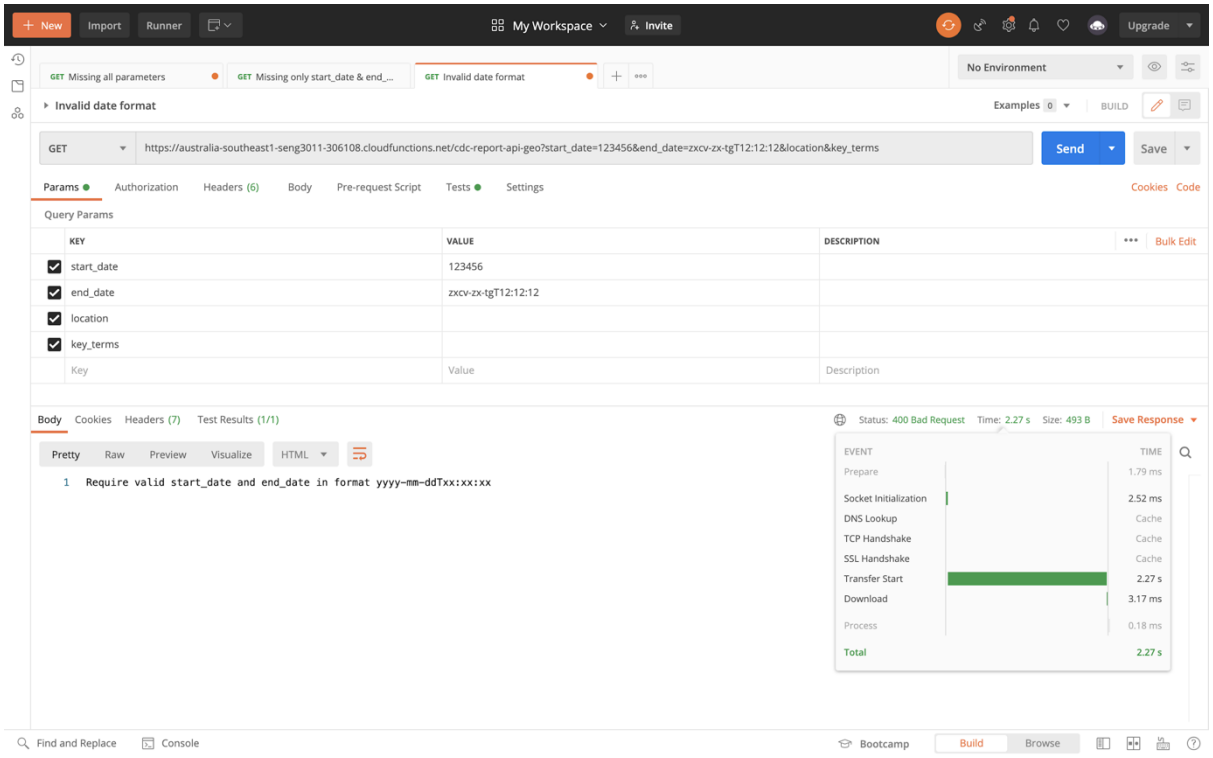
Empty start_date and end_date

Input Parameters	- start_date: - end_date: • - location: “Sydney” • - key_terms: “covid-19”
Expected Output	The status code returned should be 400 and an error message should also be returned to inform the user
Response Time	~2000ms



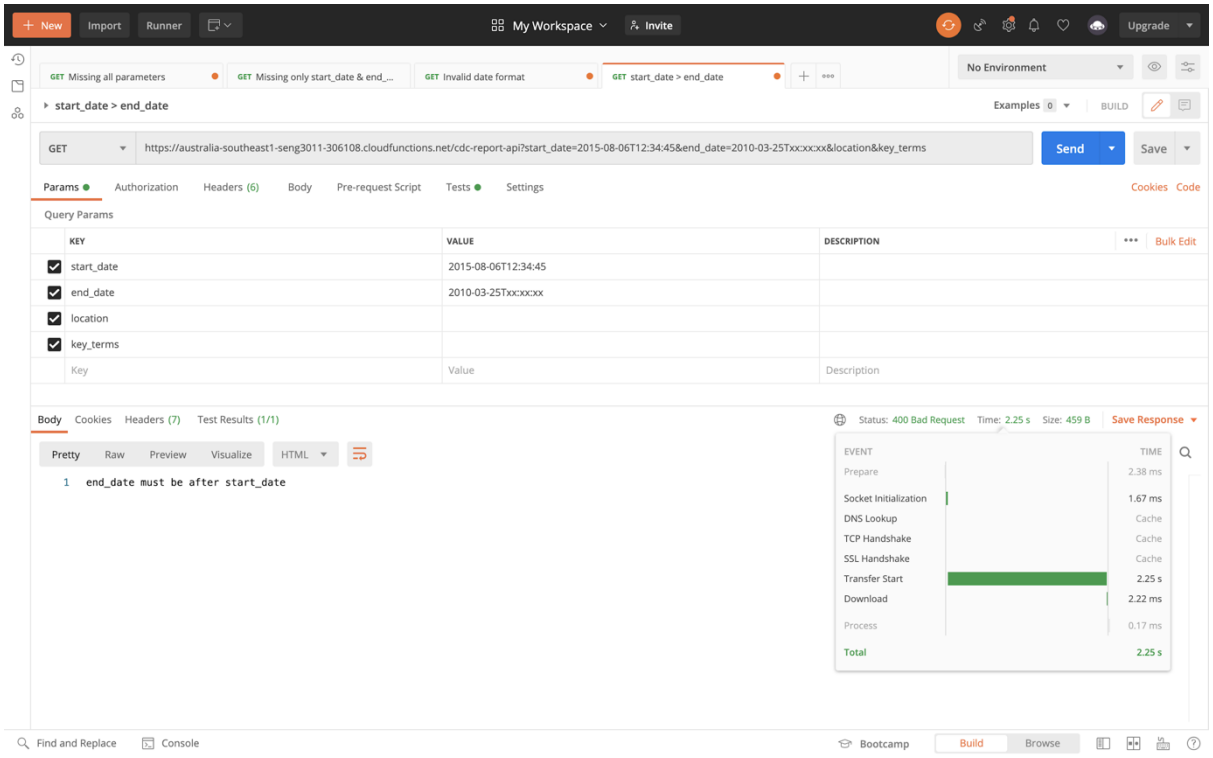
Invalid date input

Input Parameters	• - start_date: “123456” • - end_date: “zxcv-zx-tgT12:12:12” - location: - key_terms:
Expected Output	The status code should be 400 and an error message should be returned to inform the user
Response Time	~2000ms



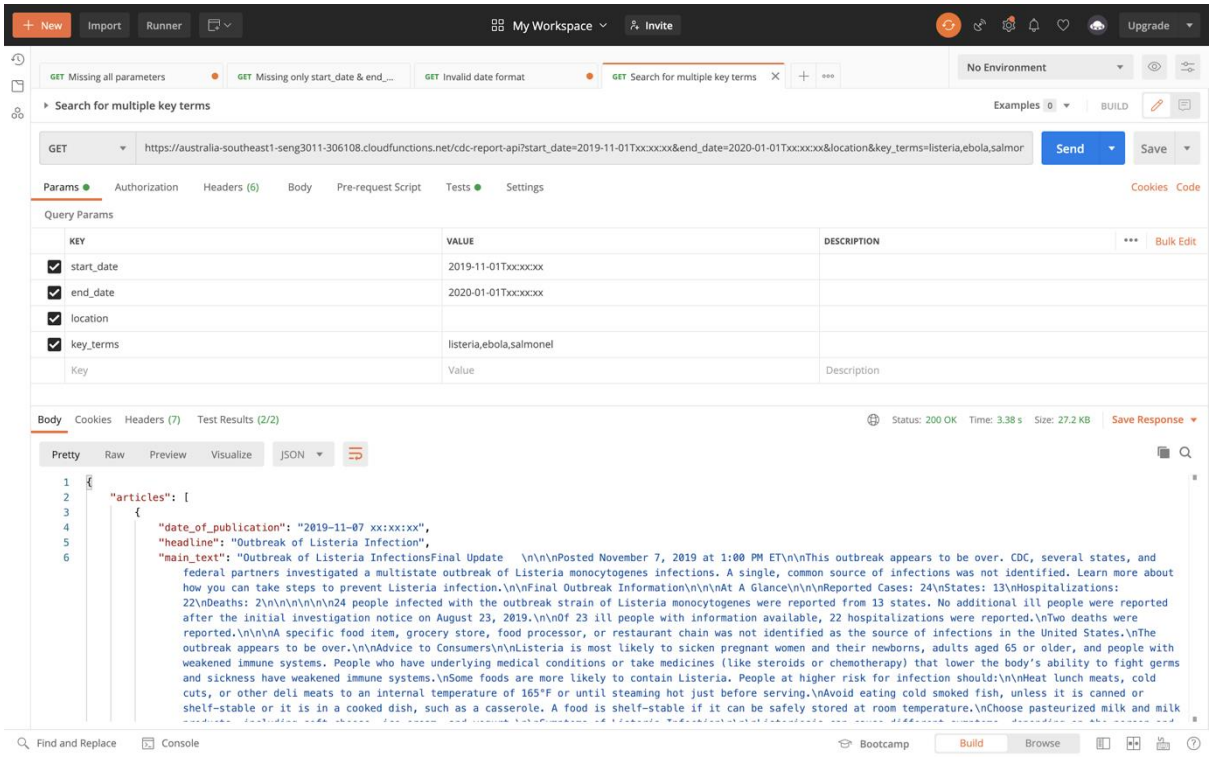
end_date is before start_end

Input Parameters	• - start_date: “2015-08-06T12:34:45” • - end_date: “2010-03-25Txx:xx:xx” - location: - key-terms:
Expected output	The status code returned should be 400 and an error message should also be returned to inform the user
Response time	~2000ms



Multiple key terms

Input Parameters	• - start_date: “2019-11-01Txx:xx:xx” • - end_date: “2020-01-01Txx:xx:xx” • - location: “” • - key_terms: “listeria,ebola,salmonella”
Expected Response	API should return a list of reports which have publication date in between start_date and end_date and contains AT LEAST one key term listed in key_terms
Response Time	~3500ms



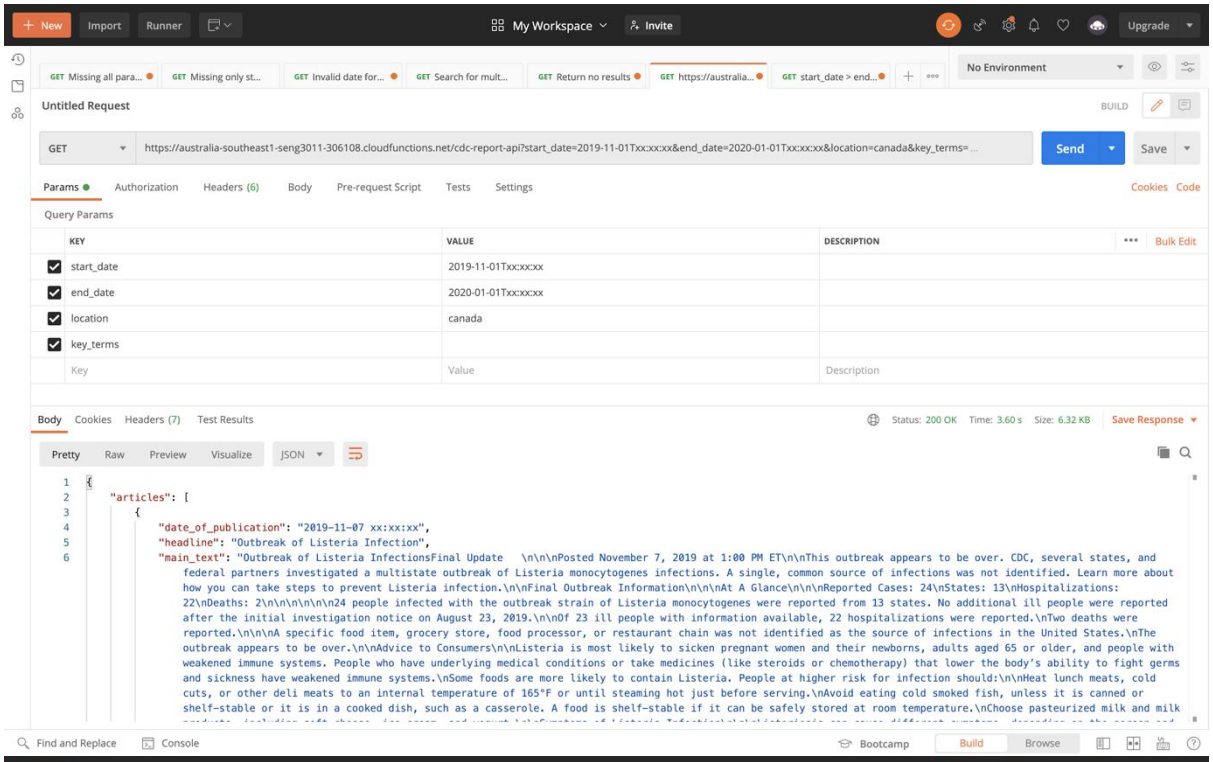
No articles returned

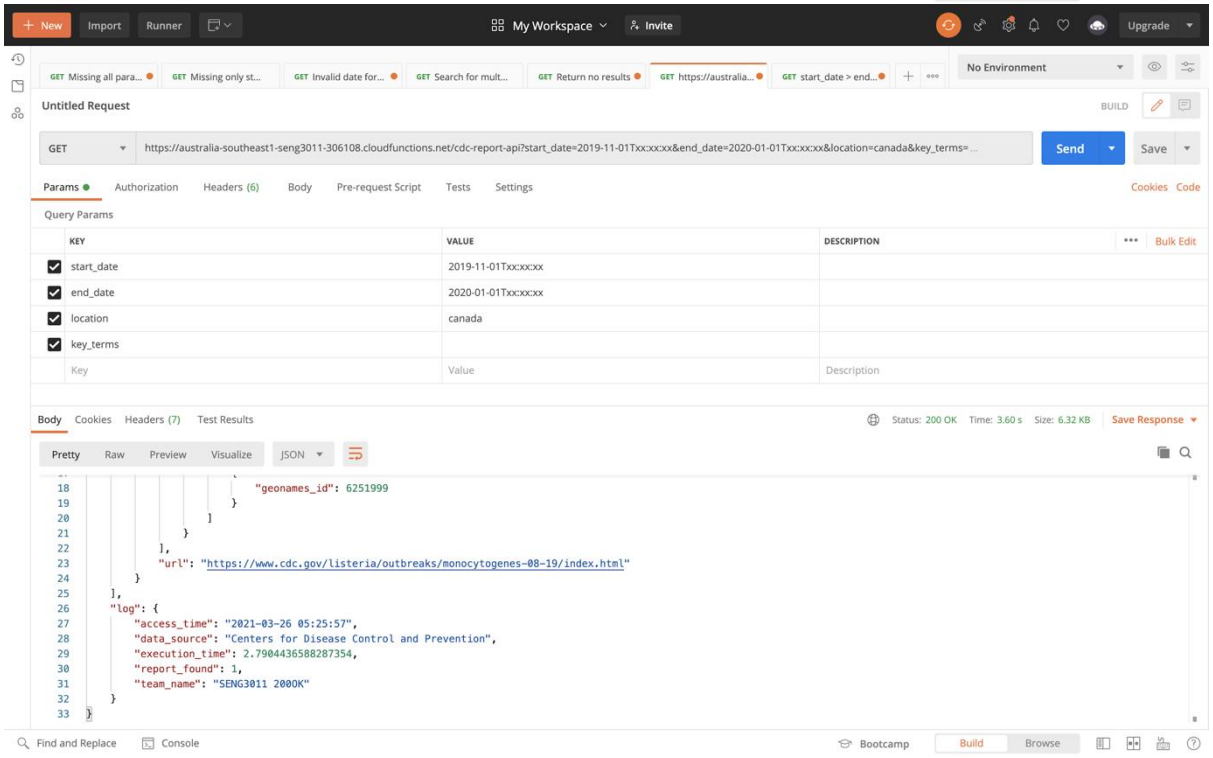
Input Parameters	• - start_date: "2019-11-01Txx:xx:xx" • - end_date: "2020-01-01Txx:xx:xx" - location:"sydney" - key_terms: “asdfsadf”
Expected Response	The status code returned should be 200 and an empty list should be returned
Response Time	

Case Insensitive

Lower Case

Input Parameters	• - start_date: "2019-11-01Txx:xx:xx" • - end_date: "2020-01-01Txx:xx:xx" • - location: "canada"
	- key_terms: “”
Expected Response	The status code should be 200 and the results should be same as query "SYDNEY"
Response Time	~3000ms

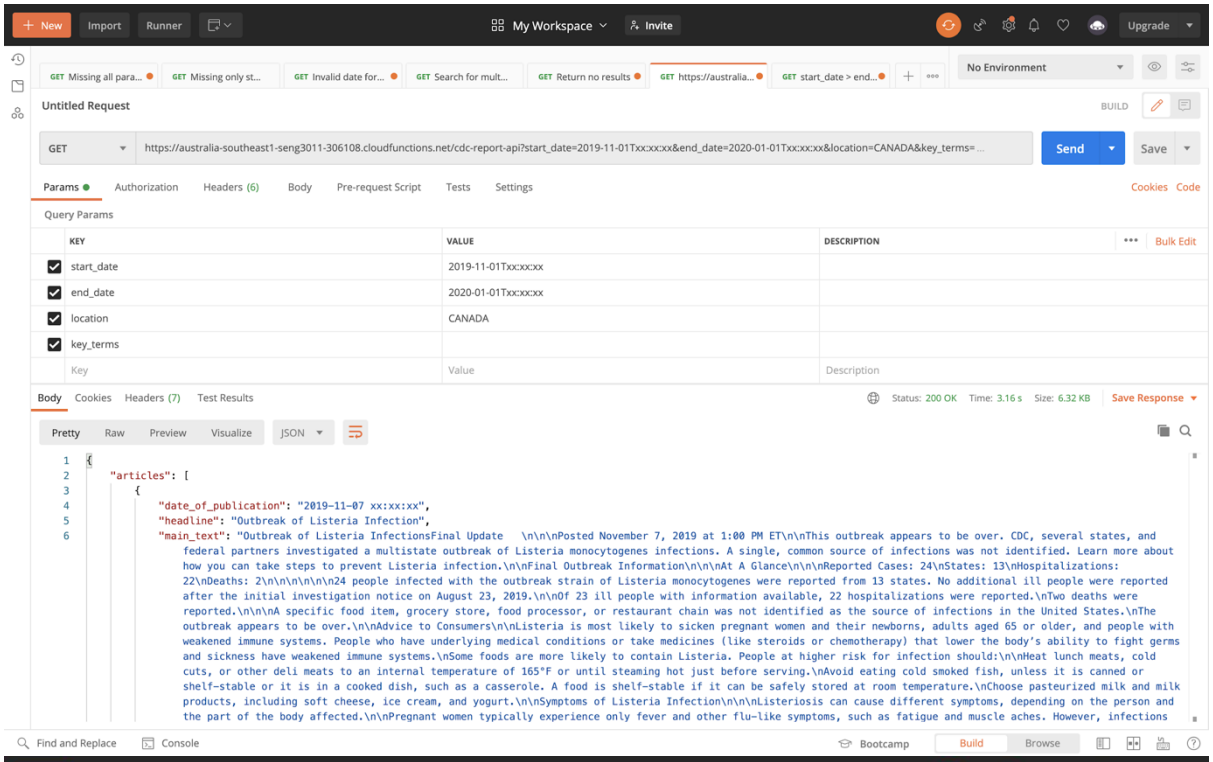


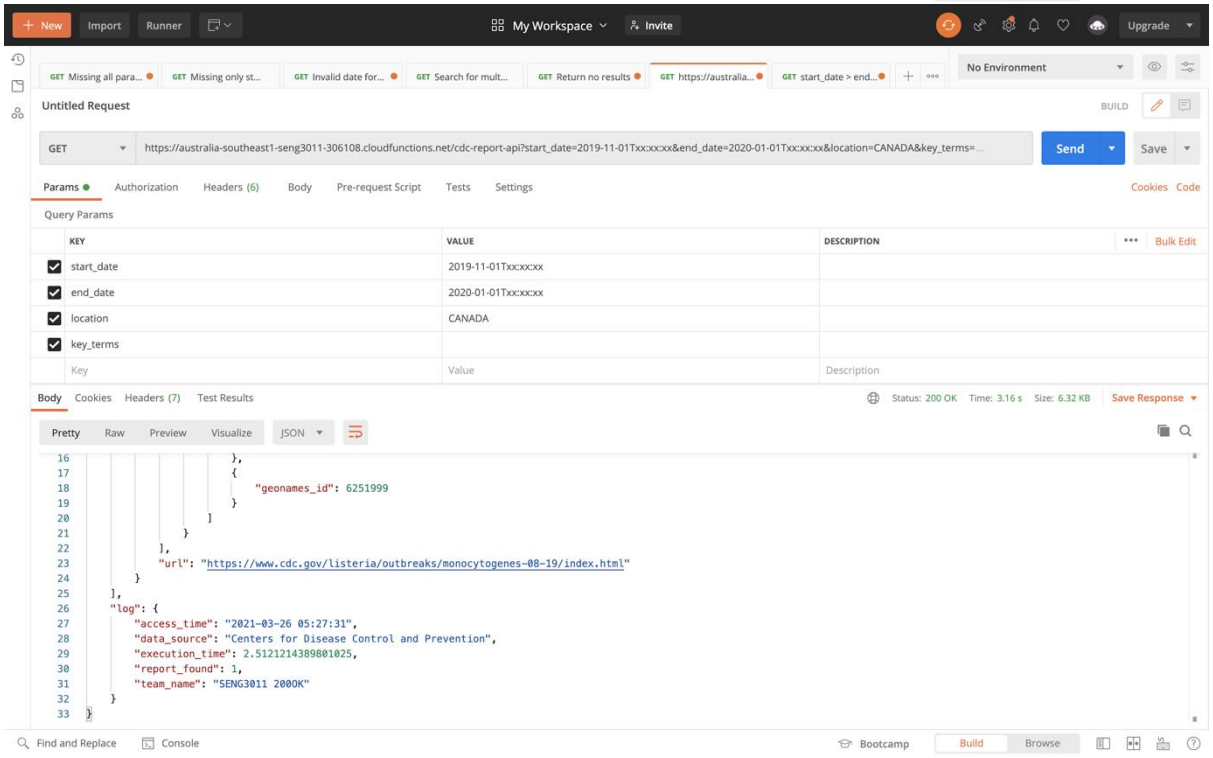


Capital

Input Parameters	• - start_date: "2019-11-01Txx:xx:xx" • - end_date: "2020-01-01Txx:xx:xx" • - location: "CANADA" • - key_terms: “”

Expected Response	The status code should be 200 and the results should be same as query "sydney"
Response Time	~3000ms

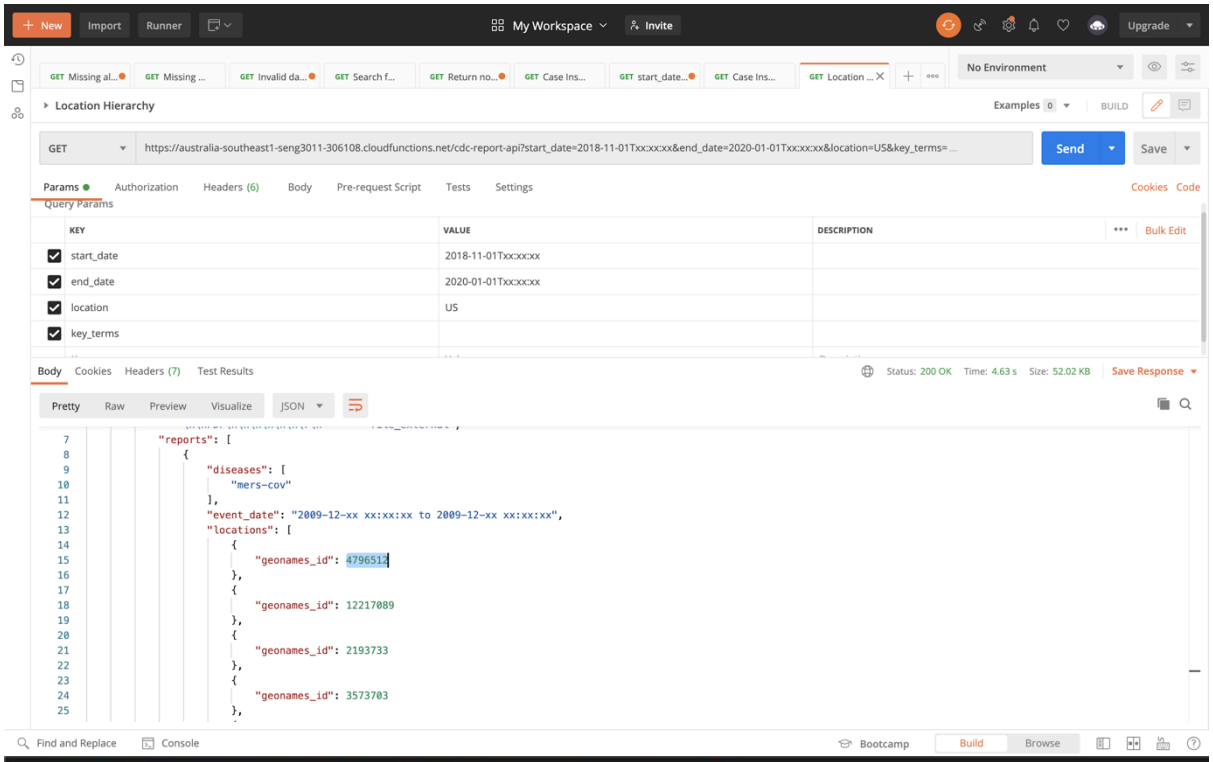




Hierarchy

Input Parameters	• - start_date: "2018-11-01Txx:xx:xx" • - end_date: "2020-01-01Txx:xx:xx" • - location: "US" • - key_terms: “”

Expected Response	The status code should be 200 and the results should contains the articles whose reports has location in the United States, eg. "New York".
Response Time	~4000ms




```
~/3011/SENG3011_200OK/PHASE_1/TestScripts(combined*) » python3 getLocation.py 4796512
St Croix
(base)
```

Saint Croix

Island in the Caribbean Sea

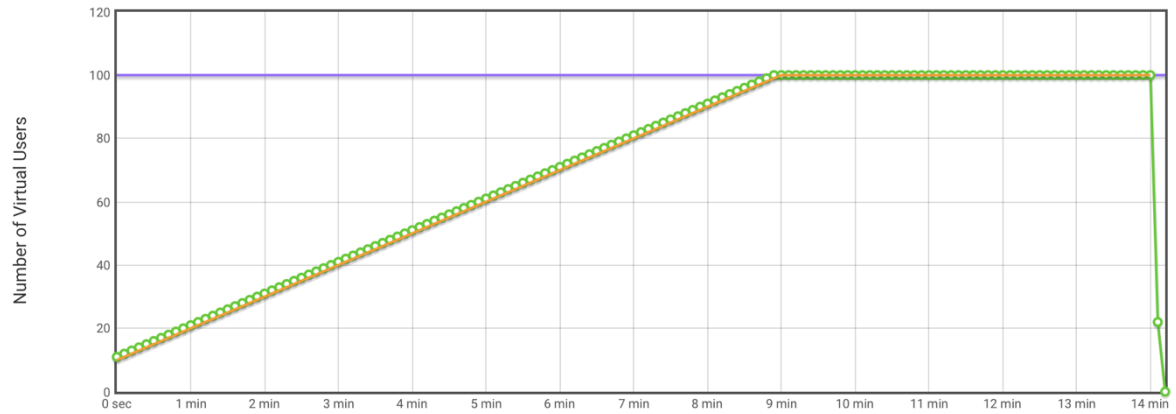
4.7 ★★★★½ 662 Google reviews

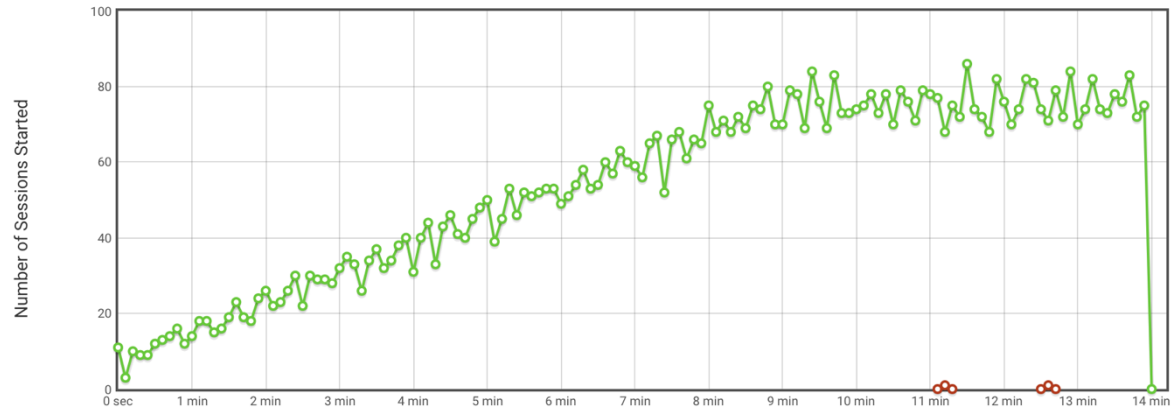
St. Croix is one of the U.S. Virgin Islands, in the Caribbean. Bright yellow Fort Christiansvaern is among Christiansted National Historic Site's Danish colonial buildings. West, St. George Village Botanical Gardens occupies an old sugar plantation. Point Udall's Millennium Monument marks the easternmost point of U.S. territory. North, Salt River Bay National Park has archaeological sites, mangroves and coral reefs. — Google

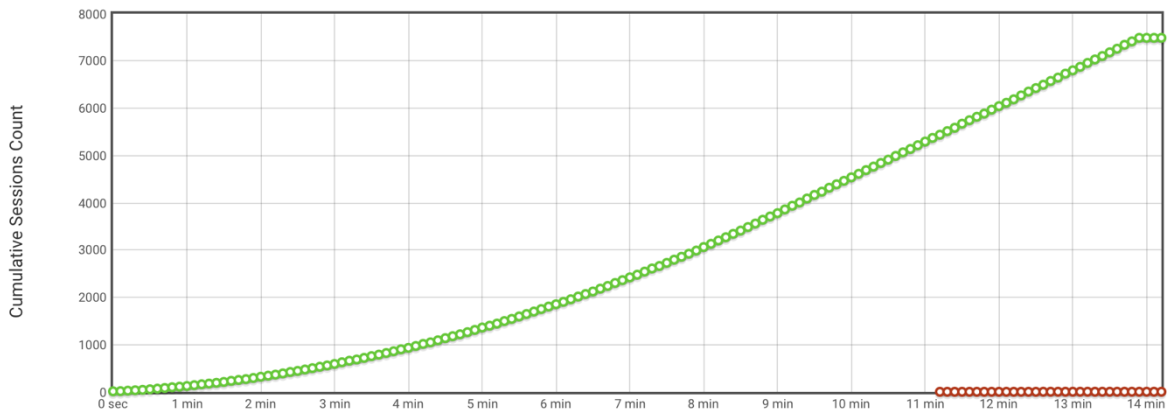
Performance Test

Load Test

We use LoadView to conduct the load test. We tried different scenarios. The initial scenario started with 5 users and raised the number of users generally up to 50. The total number of sessions are about 1200 during this test. The API works well and no error occurs. Then we started with 10 users and gradually raised the number of users to 100. And we kept 100 users for 5 mins to verify if the API can stably handle this. The total number of sessions is about 7500. The API works well overall, although there are a few errors. As a result, we claim that our API can handle at least 100 users simultaneously. Below is the result for the second scenario.







Speed Test

The speed tests are also conducted using LoadView. During the process for testing load, we also monitored the average response time. The result is shown below.

It can be seen that the average response time is around 3.5s, which is Acceptable.

Security Test

For the convenience of users, we allow both HTTP and HTTPS requests to access our resources. And we also do not require user authentication.

Besides these, all other security aspects are ensured by the third party platform. We use Google Cloud Function to deploy the API and Google Firebase to manage the data. Google Cloud Platform has a number of security services. For example, it has built-in protection to protect the application, and the transit of data is encrypted. So we trust that the API is security based on the service of Google.