Deliverable 2: Testing Documentation Report
SENG3011: THUMBNAILS

Zixiang Lin z5314168
Eric Phung z5234001
Bavan Manamohan z5208542
Kaaviya Salai Manimudian z5260340
Nitharshni Chennai Kumaravel z5255563

Table of Contents

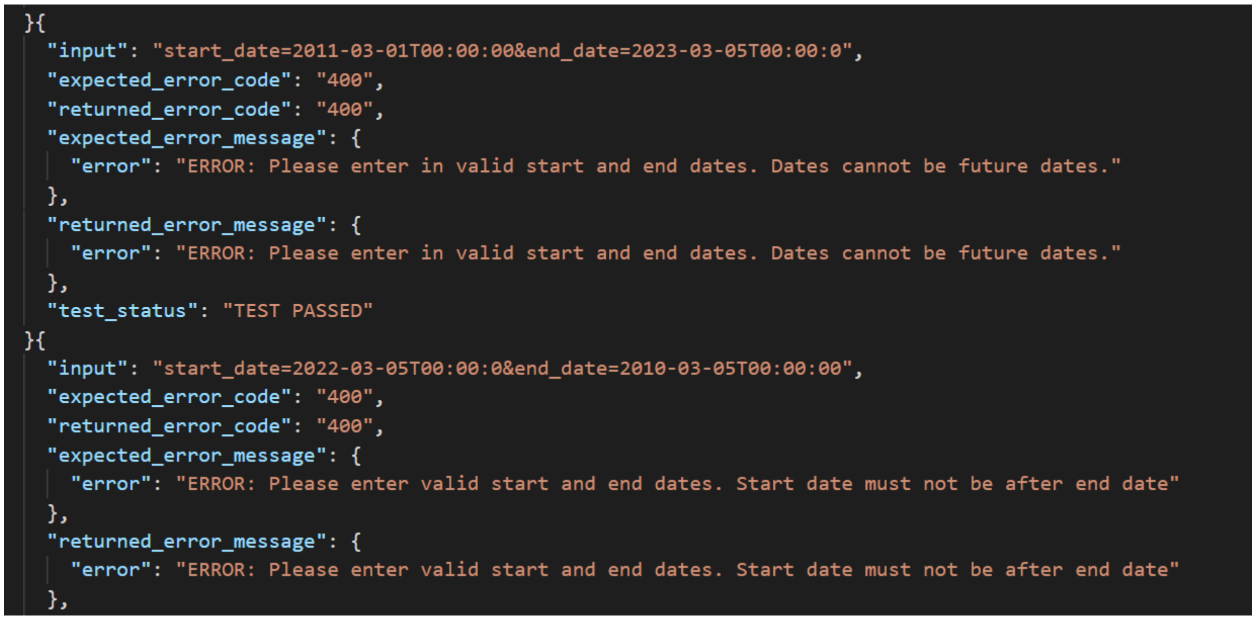
API LINK:

https://flask-service.boulmkfsb234o.us-east-1.cs.amazonlightsail.com/

Testing Methodology

The API tests were conducted using Python scripts and were categorised into two test types. The first type of tests were created to ensure that the API returns appropriate error codes and responses when a user inputs incorrect or invalid inputs. The second type of tests were conducted to verify the correctness of the API responses according to valid user inputs.

The first type of tests are contained in a script called ‘error_tests.py’. These tests are structured so they send requests to the API and examine the returned status codes and response messages, comparing them to the expected status codes and expected response messages. The test data is stored in a json file called ‘error_tests_data.json’ and is structured as shown below. Each json object also contains a ‘test_status’, to depict if the test has passed or failed. Storing expected and returned output allows for easy manual inspection, and a detailed insight into cases that failed.



```
}{
  "input": "start_date=2011-03-01T00:00:00&end_date=2023-03-05T00:00:0",
  "expected_error_code": "400",
  "returned_error_code": "400",
  "expected_error_message": {
    "error": "ERROR: Please enter in valid start and end dates. Dates cannot be future dates."
  },
  "returned_error_message": {
    "error": "ERROR: Please enter in valid start and end dates. Dates cannot be future dates."
  },
  "test_status": "TEST PASSED"
}{
  "input": "start_date=2022-03-05T00:00:0&end_date=2010-03-05T00:00:00",
  "expected_error_code": "400",
  "returned_error_code": "400",
  "expected_error_message": {
    "error": "ERROR: Please enter valid start and end dates. Start date must not be after end date"
  },
  "returned_error_message": {
    "error": "ERROR: Please enter valid start and end dates. Start date must not be after end date"
  },
```

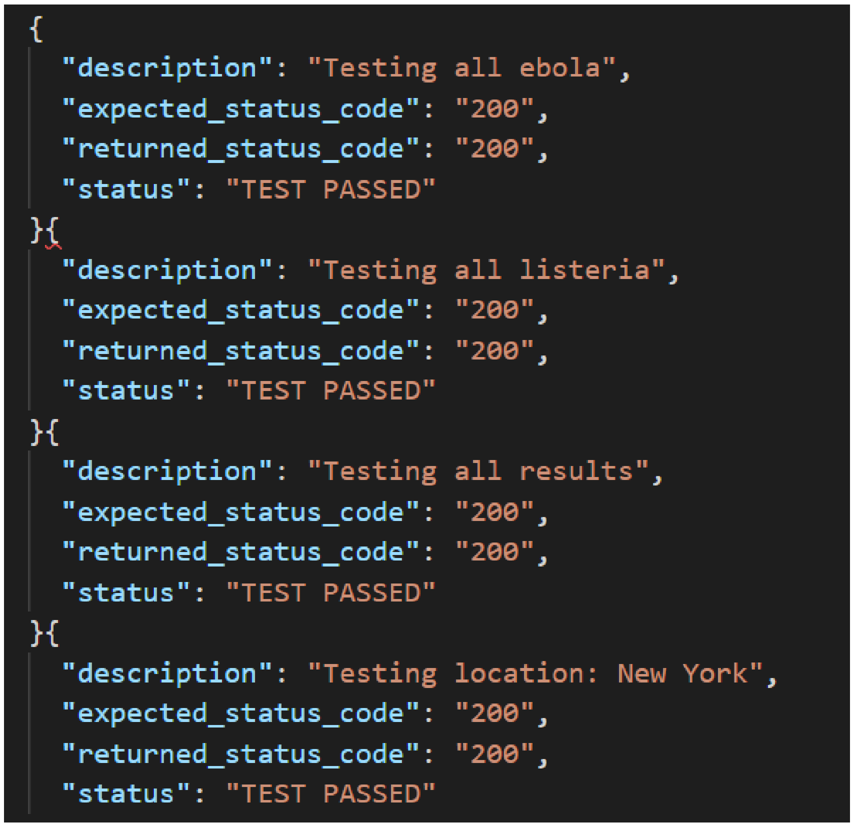
Figure 1: Displaying a snippet of the test log file

Below is a table giving an overview of the test cases, testing data and testing results for the first type of tests conducted.

Scenario	Status code returned	Error message returned	Sample test input
Dates are not in the correct format.	400 - Bad Request	{"error": "ERROR: Please enter dates in correct format: 'YYYY-MM-DDTHH:mm:ss'"}	/findstart_date=201-03-0T00:00:00&end_date=2022-03-05T00:00:0
Dates are in the future.	400 - Bad Request	{"error": "ERROR: Please enter in valid start and end dates. Dates cannot be future dates."}	/findstart_date=2011-03-01T00:00:00&end_date=2023-03-05T00:00:0
Start date is greater than the end date.	400 - Bad Request	{"error": "ERROR: Please enter valid start and end dates. Start date must not be after end date"}	/findstart_date=2022-03-05T00:00:0&end_date=2010-03-05T00:00:00
A route on the API that does not exist.	404 - Page Not Found	{"error": "404 Not Found: The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again."}	/find/info

Table 1: Overview of error test cases and results

The second type of tests that were conducted were for the success cases and the test is contained in ‘success_tests.py. These tests call particular API endpoints and compare the returned json objects to the corresponding output files that have the expected output. The test data is stored in a json file called ‘success_tests_data.json’ and is structured as shown below.



```
{
  "description": "Testing all ebola",
  "expected_status_code": "200",
  "returned_status_code": "200",
  "status": "TEST PASSED"
}{
  "description": "Testing all listeria",
  "expected_status_code": "200",
  "returned_status_code": "200",
  "status": "TEST PASSED"
}{
  "description": "Testing all results",
  "expected_status_code": "200",
  "returned_status_code": "200",
  "status": "TEST PASSED"
}{
  "description": "Testing location: New York",
  "expected_status_code": "200",
  "returned_status_code": "200",
  "status": "TEST PASSED"
```

Figure 2: Displaying a snippet of the test log file

Below is a table giving an overview of the test cases, testing data and testing results for the second type of tests conducted.

Scenario	Status code returned	Returned data	Sample test input
Only Ebola Haemorrhagic Fever	200 - success	Compared against ebola_expected.json	/findstart_date=2010-03-04T00:00:00&end_date=2022-03-05T00:00:00&keyterms=ebola
Only Listeriosis	200 - success	Compared against listeria_expected.json	/findstart_date=2010-03-04T00:00:00&end_date=2022-03-05T00:00:

			00&keyterms=listeriosis
All outputs	200 - success	Compared against all_expected.json	/findAll
Testing location "new york"	200 - success	Compared against newyork_expected.json	/findstart_date=2010-03-04T00:00:00&end_date=2022-03-05T00:00:00&location=new york
Testing keyword "deli"	200 - success	Compared against deli_expected.json	/findstart_date=2010-03-04T00:00:00&end_date=2022-1-05T00:00:00&location=new york&keyterms=deli
Testing multiple keyword - "queso", "listeria"	200 - success	Compared against multiple_keyterms_expected.json	/findstart_date=2010-03-04T00:00:00&end_date=2022-1-05T00:00:00&location=new york&keyterms=listeria,queso

Table 2: Overview of success test cases and results

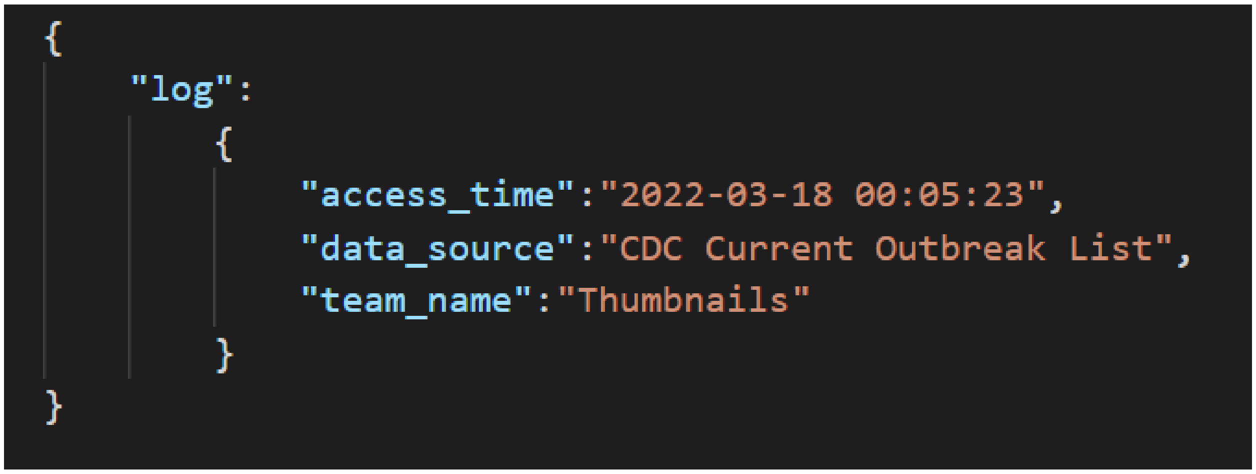
Limitations(things not tested) and Actions to Improve Test Results

In regards to testing, we encountered several limitations. The primary limitation was the lack of time which resulted in slightly rushed testing. One of the test cases we failed to address was if two date parameters were not provided to the */find* route. I.e. only *start_date* was provided as a parameter. Additionally our test cases fail to check whether the API response is empty and if the resulting message is satisfactory.

As mentioned earlier, time was the biggest limitation faced by our team in regards to testing. With additional time, we would have been able to generate many more test cases and consequently have greater coverage of all the potential results. One action which can be taken to prevent this would be to begin the development of tests prior to completion of the API, as this would allow more time to be dedicated to testing resulting in greater test coverage, greater API accuracy and faster development.

Log File

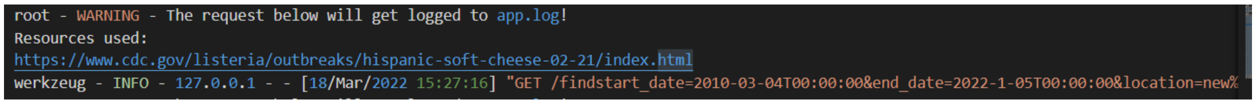
In terms of the API logs, we decided to utilise a JSON snippet which gives a response to the end user. The snippet contains the access time, data source and team name.



```
{
    "log":
        {
            "access_time":"2022-03-18 00:05:23",
            "data_source":"CDC Current Outbreak List",
            "team_name":"Thumbnails"
        }
}
```

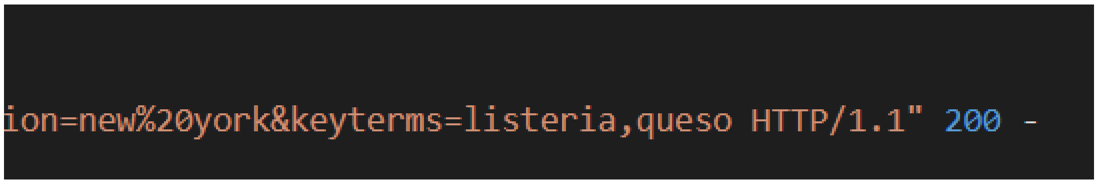
Figure 3: Sample log json snippet

Also for the backend log file, we output the API endpoint was accessed at the time, resource utilisation (urls of the web pages being scraped) and HTTP status. The log file is named app.log.



```
root - WARNING - The request below will get logged to app.log!
Resources used:
https://www.cdc.gov/listeria/outbreaks/hispanic-soft-cheese-02-21/index.html
werkzeug - INFO - 127.0.0.1 - - [18/Mar/2022 15:27:16] "GET /findstart_date=2010-03-04T00:00:00&end_date=2022-1-05T00:00:00&location=new%
```

Figure 4: Sample log file (first part)



```
ion=new%20york&keyterms=listeria,queso HTTP/1.1" 200 -
```

Figure 5: Sample log file (second part: http status)