TESTING DOCUMENTATION

SENG2011 Report

TELEUBBIES
Yin Huey Tan z5211373
Amanda (Xiaorui) Li z5206613
Sarah Oakman z5206178
Lavanya Sood z5208121
Yiyun Yang z5187469

1.0 API TESTING

Our team is using POSTMAN as well Pytest for our API testing.

1.1 POSTMAN

TEST CASES

We have created different collections to store our different cases for each POST,GET,PUT and DELETE requests. Test suites are written for each case to assert the status code return to ensure correct output is returned. Different test cases were written considering the many edge cases of that request. The usage of collections and splitting of requests into different collections allows us to collaborate more efficiently as well as improves clarity.

The json files for our test cases can be found in the TestScripts/TestInputScripts/Postman folder. The collections are also available for import in the public links below. To view the collection, select 'Import' button in Postman, then select 'Import From Link' and copy and paste the link below into it. (refer Figure 1)

Request	Link
GET	https://www.getpostman.com/collections/4dacc0984ce30c63dcde
DELETE	https://www.getpostman.com/collections/f25e3c0bb05c31d1cdea
POST	https://www.getpostman.com/collections/b85ff79e415aef394c30
PUT	https://www.getpostman.com/collections/4cbd9fb5d006ce069b7e

Table 1: Test case collections link

EXAMPLE:

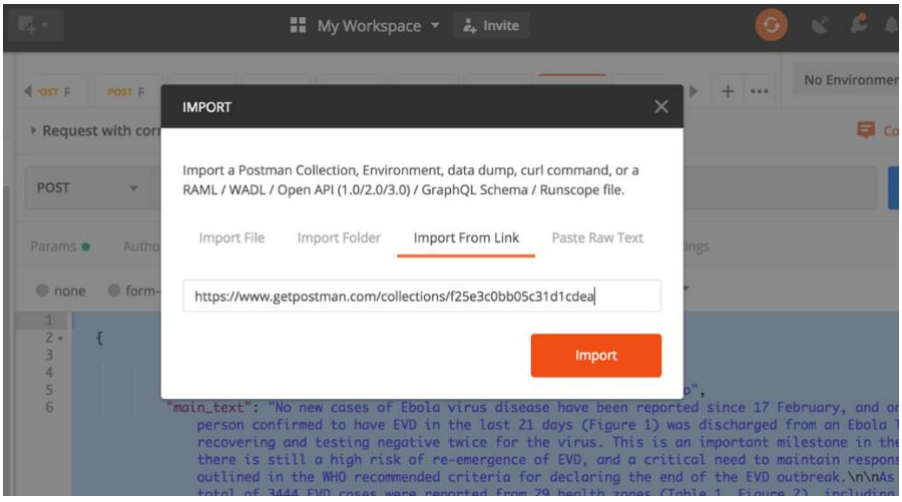

Figure 1: Importing DELETE collection

TESTING RESULTS

The json files for our test results can be found in the TestScripts/TestResults folder. Test functions are written in javascript in Postman to check the status code returned from the call. To view the test result files, select 'Runner' in Postman and then select 'Import Runs' to import the result file.

TEST RESULTS SUMMARY

EXAMPLES:

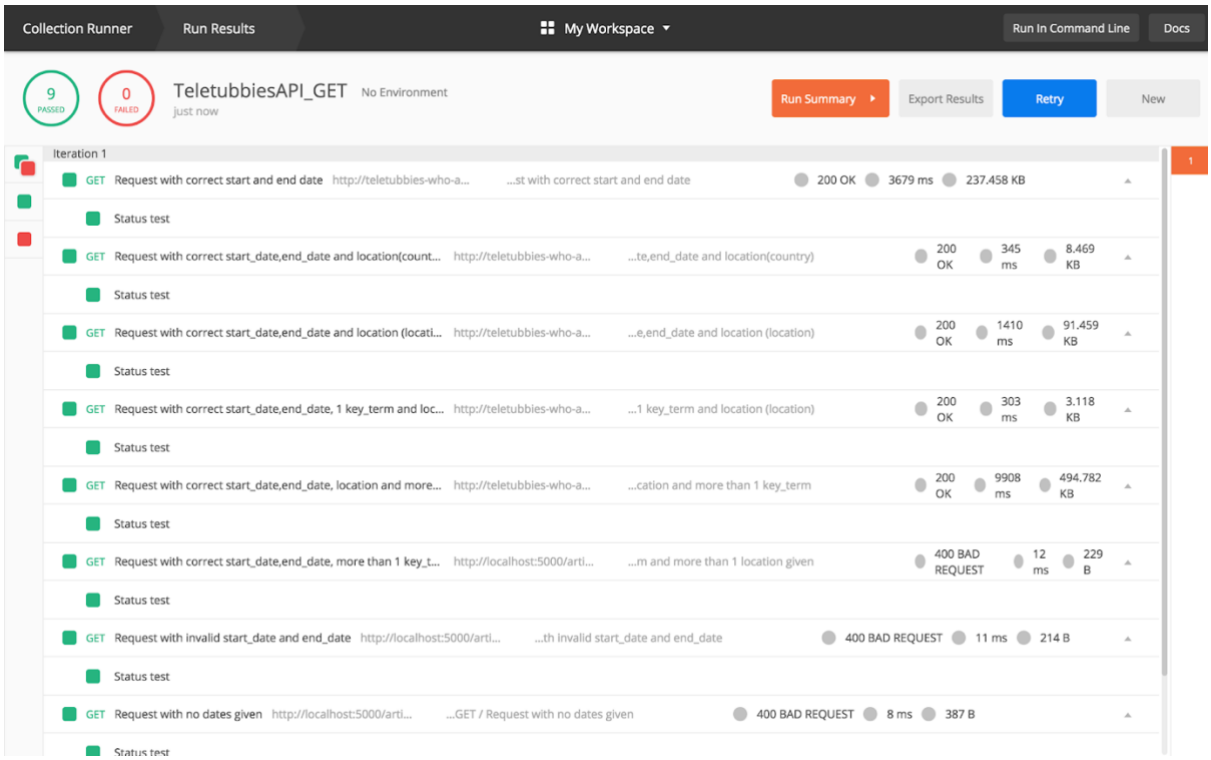

Figure 2: Test summary for GET request

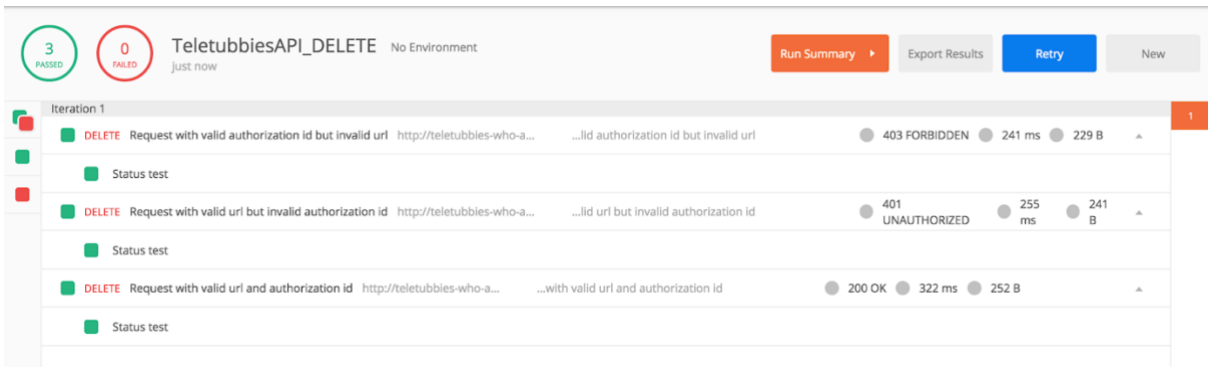

Figure 3: Test summary for DELETE request

We focused on testing the status code return from the api call to ensure that correct error code or output is returned. To improve our test results, we have considered many edge cases that users may input into our api.

1.2 PYTEST

Since we are developing our API in flask, we have also written testing functions to test using pytest in backend. Multiple test cases are written for GET, POST, PUT and DELETE requests to test the correct output, especially with various edge cases. The request functions are modified from the actual methods in api.py to simply the testing. Assertions are used to compare the actual response output with the expected one. Refer test.py file in TestScripts/TestInputScripts/Pytest folder for the testfile.

Figure 4: Pytest result

1.3 TESTING OUTPUT

Below are some example output results for GET and PUT, similarly for POST and DELETE hence we did not include them.

SOME EXAMPLE OUTPUT RESULTS FOR (GET) REQUESTS:

Test case	API call	Output result	Status code
Request with valid start and end date	http://teletubbies-who-api.herokuapp.com/article?start_date=2020-03-06T08:08:08&end_date=2020-03-08T08:08:08	{ "result": [{ "url": "https://www.who.int/csr/don/05-March-2020-ebola-drc/en/", "date_of_publication": "2020-03-06 09:41:22", "headline": "Ebola virus disease – Democratic Republic of the Congo ", "main_text": "No new cases of Ebola virus disease have been reported since 17 February, and on 3 March, the only person confirmed to have EVD in the last 21 days (Figure 1) was discharged from an Ebola Treatment Centre after recovering and testing negative twice for the virus. This is an important milestone in the outbreak. However, there is still a high risk of re-emergence of EVD, and a critical need to maintain response operations as outlined in the WHO recommended criteria for declaring the end of the EVD outbreak.", "reports": [{ "event_date": "2020-02-17 xx:xx:xx to 2020-03-03 xx:xx:xx",	200

		"locations": [
		{
		"country": "Congo",
		"location": ""
		}
		],
		"diseases": [
		"ebola haemorrhagic fever"
		],
		"syndromes": "",
		"description": [
		{
		"Source": "",
		"Cases": 3444,
		"Deaths": 2264,
		"Controls": ""
		}
		]
		}
		]
		}
		],
		"status": 200
		}	

Request with valid start and end date, location and key_terms	http://teletubbies-who-api.herokuapp.com/article?start_date=2019-12-06T08:08:08&end_date=2020-01-13T23:08:08&location=China&key_term=dengue,ebola	{
		"result": [
		{
		"url":"https://www.who.int/csr/don/12-january-2020-novel-coronavirus-china/en/",
		"date_of_publication": "2020-01-13 15:42:15",
		"headline": "Novel Coronavirus – China",
		"main_text": "On 11 and 12 January 2020, WHO received further detailed information from the National Health Commission about the outbreak.",
		"reports": [
		{
		"event_date": "2020-01-11 xx:xx:xx to 2020-01-12 xx:xx:xx",
		"locations": [
		{
		"country": "China",
		"location": "Wuhan"
		}
		],
		"diseases": [
		"COVID-19"
		],
		"syndromes": [
		"Acute respiratory syndrome"
		],
		"description": [
		{
		"Source": "",
		"Cases": 41,
		"Deaths": 1,
		"Controls": "A total of 763 close contacts including healthcare
			200

		workers, have been identified and followed up and no additional cases of infection with the novel coronavirus have been identified & The Wuhan Municipal Health Commission carried out active case finding, and retrospective investigations of the current cluster of patients have been completed & The Huanan Seafood Wholesale Market has been temporarily closed to carry out environmental sanitation and disinfection & Public risk communication activities have been carried out to improve public awareness and adoption of self-protection measures" }] }] }], "status": 200 }	

Request with invalid start date or end date	http://teletubbies-who-api.herokuapp.com/article?start_date=2000-12-0608:08:08&end_date=2020-12-0608:08:08	{
		"message": "Invalid date input",
		"status": 400
		}
			400

Request with no matching data	http://teletubbies-who-api.herokuapp.com/article?start_date=2000-12-06T08:08:08&end_date=2020-12-06T08:08:08&location=East Africa	{
		"message": "No data found",
		"status": 404
		}
			404

Table 2: Example output for GET request

SOME EXAMPLE OUTPUT RESULTS FOR (PUT) REQUESTS:

Test case	API call	Request body	Output result	Status code

Request with valid authorisation id and request body	https://teletubbies-who-api.herokuapp.com/article?id=1810051939	{
		"url": "https://www.who.int/csr/don/05-March-2020-ebola-drc/en/",
		"reports": [
		{
		"event_date": "2020-01-01 00:00:00",
		"locations": [
		{
		"country": "string",
		"location": "string"
		}
		],
		"diseases": [
			{
			'message' : "Url Successfully added",
			'status' : 200
			}
				200

		"string"
		],
		"syndromes": [
		"string"
		],
		"description": [
		{
		"source": "string",
		"cases": 0,
		"deaths": 0,
		"controls": "string"
		}
		]
		}
		]
		}		

Request with invalid authorisation id	https://teletubbies-who-api.herokuapp.com/article?id=1111111	{
		"url": "https://www.who.int/csr/don/05-March-2020-ebola-drc/en/",
		"reports": [
		{
		"event_date": "2020-01-01 00:00:00",
		"locations": [
		{
		"country": "string",
		"location": "string"
		}
		],
		"diseases": [
		"string"
		],
		"syndromes": [
		"string"
		],
		"description": [
		{
		"source": "string",
		"cases": 0,
		"deaths": 0,
		"controls": "string"
		}
		]
		}
		]
		}
			{ 'message' : "Incorrect Authorization Key", 'status' : 401 }	401

Request with invalid event date	https://teletubbies-who-api.herokuapp.com/article?id=1810051939	{
		"url": "https://www.who.int/csr/don/05-March-2020-ebola-drc/en/",
		"reports": [
		{
		"event_date": "2020-01-01",
		"locations": [
		{
		"country": "string",
		"location": "string"
		}
		],
		"diseases": [
			{ 'message' : "Invalid date input. Example input: '2020-01-01 00:00:00' or '2020-01-01 00:00:00 to 2020-02-01 00:00:00'", 'status' : 400	400

		"string"
		],
		"syndromes": [
		"string"
		],
		"description": [
		{
		"source": "string",
		"cases": 0,
		"deaths": 0,
		"controls": "string"
		}
		]
		}
		]
		}
			}	

Request with empty url	https://teletubbies-who-api.herokuapp.com/article?id=1810051939	{
		"url": "",
		"reports": [
		{
		"event_date": "2020-01-01 00:00:00",
		"locations": [
		{
		"country": "string",
		"location": "string"
		}
		],
		"diseases": [
		"string"
		],
		"syndromes": [
		"string"
		],
		"description": [
		{
		"source": "string",
		"cases": 0,
		"deaths": 0,
		"controls": "string"
		}
		]
		}
		]
		}
			{
			'message': "Url can't be empty",
			'status': 400
			}
				400

Request with invalid url	https://teletubbies-who-api.herokuapp.com/article?id=1810051939	{
		"url": "https://www.who.int/not_exist",
		"reports": [
		{
		"event_date": "2020-01-01 00:00:00",
		"locations": [
		{
		"country": "string",
		"location": "string"
		}
		],
		"diseases": [
		"string"
		],
			{
			'message': "Url does not exist",
			'status': 403
			}
				403

		"syndromes": [
		"string"
		],
		"description": [
		{
		"source": "string",
		"cases": 0,
		"deaths": 0,
		"controls": "string"
		}
		]
		}
		]
		}		

Table 3: Example output for PUT request

2.0 LIMITATIONS

There are several challenges during the API Testing process. The writing of test cases for every request takes a considerable amount of time due to the many edge cases that needed to be covered. Whenever an error is spotted, we would have to reupload our updated code for heroku and deploy again after updating in the backend, which is tolerably troublesome. For pytests, we are also unable to call the method inside the article resource class directly as the whole file will run when we call the method. Hence, we improvised by duplicating our get, post, put and delete methods into the test.py file where we write the pytests for each request.

Since timezone is a dropdown option for user to select in our swagger documentation, we assume user does not provide values other than the options eg. timezone in lowercases, unavailable timezone etc. Hence testing for invalid timezone given was not implemented.

There are also limitations in checking the updated database after Heroku is used to deploy the API module to a server allowing it to be accessed on the Web. Heroku utilises dynos, which is a controlled container framework while operating web-service applications. Therefore they can no longer be accessed and monitored after the files are pipelined from the github and deployed on the web. We don't host it locally anymore, so all the changes of the database are updated on heroku's cloud system with dynos after api requests. At the moment, we can only check the updated database in SQLiteStudio after running the requests with the localhost server.