

Testing Documentation



Epidemic Eagle

Testing Documentation

Prepared for SENG3011

Created by Aaron Shek, Sahibpreet Bassi, Daniel Steyn, Liam Treavors and Mihail Georgiev

- [Epidemic Eagle](#)
 - [Testing Documentation](#)
- [Context](#)
- [Manual Testing](#)
 - [Example OpenAPI route test for /api/articles.](#)
- [Automated Testing](#)
 - [GitHub Pipeline](#)
 - [Automatic Deployment](#)
 - [Overview of Tests](#)
 - [Unit Testing](#)
 - [Parameter Testing](#)
 - [Acceptance Testing](#)
 - [Limitations of Testing](#)
 - [Web Scraper \(Scrapy\)](#)
 - [Log Files](#)
 - [Benefit of Logs](#)

Context

Testing our API ensures that the API is correctly implemented according to standards set by the user. Without testing our API, then the client would be dissatisfied with the product, as well as reducing the reputation of the API.

There are two ways our API is tested: manual testing and automated testing.

Manual testing is done to test the user experience via human observation of how data is sent to the user, using tools such as Swagger.

Automated testing is also included as an aspect of our API, to inform our team when an error occurs in the development environment. Another feature of automated testing is the ease of repeated running of tests to reduce human error and labor. In our API, we will be using GitHub Actions as a pipeline as well as another check in Heroku for the website content.

Manual Testing

We will be using OpenAPI (formerly known as Swagger) to test the functionality of our API routes from the user experience. Since we are using FastAPI, the OpenAPI is automatically generated for us, however it will still require manual checking to ensure the behavior of routes is correct.

For the hosting of our OpenAPI, it is located at <http://epidemic-eagle.herokuapp.com/docs>.

OpenAPI is a user-friendly and open-source tool to test the JSON responses from our routes.

One benefit of Swagger is that it shows which parameters are required in order to get a successful response from the route.

GET /api/articles List All Articles With Params

Lists all the articles specified within the parameters: start_date to end_date, key_terms and location. page_number can be specified to go to the corresponding page.

Parameters

Name	Description
key_terms * required string (query)	<input type="text" value="covid"/>
location * required string (query)	<input type="text" value="sydney"/>
start_date * required string (query) pattern: ^(\d{4})-(\d{d xx})-(\d{d xx})\$: (\d{d xx}) : (\d{d xx})\$	<input type="text" value="start_date"/>
end_date * required string (query) pattern: ^(\d{4})-(\d{d xx})-(\d{d xx})\$: (\d{d xx}) : (\d{d xx})\$	<input type="text" value="end_date"/>
page_number integer (query)	<input type="text" value="page_number"/>

Execute

Additionally the response body and headers are also shown, to provide extra information to the client.

`{\d{4}} - (\d{d}|xx) - (\d{d}|xx) :`
`{\d{d}|xx} - (\d{d}|xx) :`
`{\d{d}|xx} :`
`{\d{d}|xx} :`
`{\d{d}|xx} :`

end_date * required
string
(query)
pattern: ^(\d{4}) - (\d{d}|xx) - (\d{d}|xx) : (\d{d}|xx) - (\d{d}|xx) : (\d{d}|xx) :

page_number
integer
(query)

2019-xx-xx xx:xx:xx

page_number

Execute Clear

Responses

Curl

```
curl -X 'GET' \
'https://epidemic-eagle.herokuapp.com/api/articles?key_terms=covid&location=sydney&start_date=2018-xx-xx%20xx%3Axx%3Axx&end_date=2019-xx-xx%20xx%3Axx%3Axx' \
-H 'accept: application/json'
```

Request URL

https://epidemic-eagle.herokuapp.com/api/articles?key_terms=covid&location=sydney&start_date=2018-xx-xx%20xx%3Axx%3Axx&end_date=2019-xx-xx%20xx%3Axx%3Axx

Server response

Code	Details
200	<p>Response body</p> <pre>{ "num_pages": 1, "page_number": 1, "articles": [] }</pre> <p>Response headers</p> <pre>connection: keep-alive content-length: 45 content-type: application/json date: Thu, 17 Mar 2022 11:46:27 GMT server: uvicorn via: 1.1 vegur</pre>

Example OpenAPI route test for /api/articles.

First of all, to generate a GET request we use the url “/api/articles” and press the “Try it out” button.

api

GET /api/articles List All Articles With Params

Lists all the articles specified within the parameters: start_date to end_date, key_terms and location. page_number can be specified to go to the corresponding page.

Parameters Try it out

Name	Description
key_terms * required string (query)	<input type="text" value="key_terms"/>
location * required string (query)	<input type="text" value="location"/>
start_date * required string (query) pattern: ^(\d{4})-(\d{2} \d{1,2})-(\d{2} \d{1,2})\$	<input type="text" value="start_date"/>
end_date * required string (query) pattern: ^(\d{4})-(\d{2} \d{1,2})-(\d{2} \d{1,2})\$	<input type="text" value="end_date"/>
page_number integer (query)	<input type="text" value="page_number"/>

Responses

Firstly we enter the required parameters, in the format specified.

GET /api/articles List All Articles With Params

Lists all the articles specified within the parameters: start_date to end_date, key_terms and location. page_number can be specified to go to the corresponding page.

Parameters Cancel

Name	Description
key_terms * required string (query)	<input type="text" value="covid"/>
location * required string (query)	<input type="text" value="sydney"/>
start_date * required string (query) pattern: ^(\d{4})-(\d{2} \d{1,2})-(\d{2} \d{1,2})\$	<input type="text" value="2018-xx-xx xx:xx:xx"/>
end_date * required string (query) pattern: ^(\d{4})-(\d{2} \d{1,2})-(\d{2} \d{1,2})\$	<input type="text" value="2019-xx-xx xx:xx:xx"/>
page_number integer (query)	<input type="text" value="page_number"/>

Execute Clear

There are two types of status codes, a 422 Error for invalid parameters or a 200 successful response with a list of articles with the parameters.

Responses

Code	Description	Links
200	<div>Successful Response</div> <div>Media type application/json</div> <div>Controls Accept header</div> <div>Example Value Schema</div> <div><pre>{ "num_pages": 0, "page_number": 0, "articles": [{ "url": "string", "date of publication": "string", "headline": "string", "main text": "string", "reports": [{ "diseases": ["string"], "syndromes": ["string"], "event_date": "string", "locations": [{ "country": "string", "location": "string" }] }] }] }</pre></div>	No links
422	<div>Validation Error</div> <div>Media type application/json</div> <div>Example Value Schema</div> <div><pre>{ "detail": { { "loc": ["string"], "msg": "string", "type": "string" }] }</pre></div>	No links

Then we can push the execute button.

end_date * required
string
(query)
pattern: ^(\d{4})-(\d{2})-(\d{2})\$

2019-xx-xx xx:xx:xx

page_number
integer
(query)

page_number

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'https://epidemic-eagle.herokuapp.com/api/articles?key_terms=covid&location=sydney&start_date=2018-xx-xx%20xx%3Axx%3Axx&end_date=2019-xx-xx%20xx%3Axx%3Axx' \
  -H 'accept: application/json'
```

Request URL

```
https://epidemic-eagle.herokuapp.com/api/articles?key_terms=covid&location=sydney&start_date=2018-xx-xx%20xx%3Axx%3Axx&end_date=2019-xx-xx%20xx%3Axx%3Axx
```

Server response

Code	Details
200	<p>Response body</p> <pre>{ "num_pages": 1, "page_number": 1, "articles": [] }</pre> <p>Response headers</p> <pre>connection: keep-alive content-length: 45 content-type: application/json date: Thu, 17 Mar 2022 11:46:27 GMT server: uvicorn via: 1.1 vegur</pre>

In this instance, it responds with a passed result and a response code 200.

By using manual testing, we can get a sense of how a user would interact with our API.

And by using OpenAPI, we can visualize how a client uses our routes and update them if they are not user friendly to enter. It is an effective tool to test user experience and to quickly change and send parameters.

Automated Testing

To ensure that the software we are deploying is usable and functional code, we have implemented a GitHub pipeline. This will run multiple tests every time we push our code to main, and if these some of these tests fail then the code will not be pushed to the website. By using a pipeline,

GitHub Pipeline

First, the pipeline ensures that the Operating System being used when running the tests is Ubuntu. This is to make sure that the tests have consistent results to the web host, since we are using a linux based system. The pipeline then checks that it is in the correct branch, so that the tests run on the correct version of our software.

```
13 # A workflow run is made up of one or more jobs that can run sequentially or in parallel
14 jobs:
15   build:
16     strategy:
17       matrix:
18         python-version: [3.10.2]
19         runs-on: ubuntu-latest
20
```

Using Ubuntu, the pipeline then installs python version 3.10.2, our desired version for all tests and software. Different versions provide slightly different functionality, so this also maintains consistency.

```

21     # Steps represent a sequence of tasks that will be executed as part of the job
22     steps:
23     - name: Checkout
24       uses: actions/checkout@v2
25       with:
26         fetch-depth: 0
27
28     - name: Switch to Current Branch
29       run: git checkout ${ env.BRANCH }
30
31     - name: Set up Python ${ matrix.python-version }
32       uses: actions/setup-python@v1
33       with:
34         python-version: ${ matrix.python-version }

```

We then install a dependency named pip, which is a helpful tool to manage and download packages or libraries. A list of all of these is found in requirements.txt, and pip is used to make sure the correct version of these are all installed. Pip is also used to install pytest, which is how the tests will finally be run after the setup stage.

```

36     - name: Install dependencies
37       run: |
38         python -m pip install --upgrade pip
39         pip install -r requirements.txt
40         pip install pytest
41     - name: Run unit tests
42       run: python -m pytest --import-mode=append PHASE_1/

```

Now that the correct environment is installed, the pytest command is used to run all of the tests found in python files with the prefix 'test_'. If all of these tests pass, the pipeline will succeed and will show a green tick on GitHub. If even one of these tests fail, then the pipeline will fail and will show a red cross on GitHub, and the code will not be pushed to deployment. The pipeline can be checked to see which test caused the failure, so that a fix can be implemented.



Search or jump to...



[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)

EpidemicEagle / SENG3011_EpidemicEagle Public

[Code](#) [Issues](#) [Pull requests](#) 1 [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

[pipeline](#) [5 branches](#) [0 tags](#)

[Go to file](#)

[Add file](#)

[Code](#)

This branch is 17 commits ahead of main.

[#2](#)

Aaron Shek Renamed files

✓ 145ec0 2 days ago 24 commits

github/workflows	Fixed files	2 days ago
PHASE_1	Renamed files	2 days ago
PHASE_2	Repository Setup	14 days ago
Reports	Renamed files	2 days ago
README.md	Renamed files	2 days ago
setup.sh	Renamed files	2 days ago

[Code](#) [Issues](#) [Pull requests](#) 1 [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)

Files pushed to create pipeline CI #11

[Re-run all jobs](#)



Summary

Jobs

build (3.10.2)

build (3.10.2)

succeeded 2 days ago in 13s

[Search logs](#)



> Set up job	1s
> Checkout	2s
> Switch to Current Branch	1s
> Set up Python 3.10.2	1s
> Install dependencies	9s
> Run unit tests	1s
<pre>1 ▶ Run python -m pytest --import-mode=append PHASE_1/ 2 ===== test session starts ===== 3 platform linux -- Python 3.10.2, pytest-7.0.1, pluggy-1.0.0 4 rootdir: /home/runner/work/SENG3011_EpidemicEagle/SENG3011_EpidemicEagle 5 plugins: anyio-3.5.0 6 collected 4 items 7 8 PHASE_1/API_SourceCode/backend/src/test_old.py .. [50%] 9 PHASE_1/testscripts/test_api.py .. [100%] 10 11 ===== 4 passed in 0.30s =====</pre>	
> Post Checkout	1s
> Complete job	0s

Open

Files pushed to create pipeline #2
aashek wants to merge 41 commits into `main` from `pipeline`

Added command to access logs

0cfec35

aashek deployed to epidemic-eagle 34 minutes ago

View deployment

Last commit before push to main

3c7e323

Add more commits by pushing to the `pipeline` branch on EpidemicEagle/SENG3011_EpidemicEagle.

This branch was successfully deployed

1 active (outdated) deployment

All checks have passed

1 successful check

Show all checks

This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request

You can also open [this in GitHub Desktop](#) or view [command line instructions](#).

Successful Pipeline

Search or jump to... Pull requests Issues Marketplace Explore

EpidemicEagle / SENG3011_EpidemicEagle Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Files pushed to create pipeline CI #26 Re-run jobs

Summary

Jobs

build (3.10.2) failed 3 days ago in 27s

Search logs

- Set up job 2s
- Checkout 2s
- Switch to Current Branch 0s
- Set up Python 3.10.2 0s
- Install dependencies 21s
- Run unit tests 0s

```

1 ▶ Run python -m pytest --import-mode=append PHASE_1/
2 ===== test session starts =====
3 platform linux -- Python 3.10.2, pytest-7.1.0, pluggy-1.0.0
4 rootdir: /home/runner/work/SENG3011_EpidemicEagle/SENG3011_EpidemicEagle, configfile: pytest.ini
5 plugins: anyio-3.5.0
6 collected 3 items
7
8 PHASE_1/API_SourceCode/backend/src/test_old.py F. [ 66%]
9 PHASE_1/TestScripts/test_api.py F [100%]
10
11 ===== FAILURES =====
12 test_search
13
14 def test_search():
15     > response = client.get("/api/search", params={"start_date": "2018-xx-xx xx:xx:xx", "end_date": "2018-xx-xx xx:xx:xx",
16       "key_terms": "wow,no", "location": "somewhere"})
17
18 PHASE_1/API_SourceCode/backend/src/test_old.py:21:
19
20 /opt/hostedtoolcache/Python/3.10.2/x64/lib/python3.10/site-packages/requests/sessions.py:542: in get

```

EpidemicEagle / SENG3011_EpidemicEagle Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

pipeline SENG3011_EpidemicEagle / requirements.txt Go to file

aashek Fixed pytest errors. Latest commit 94a4ee9 3 days ago History

1 contributor

31 lines (31 sloc) 1.08 KB

Raw Blame

```

1 anyio==3.5.0
2 asgiref==3.5.0
3 certifi==2021.10.8
4 charset-normalizer==2.0.12
5 click==8.0.4
6 colorama==0.4.4
7 dnspython==2.2.1
8 email-validator==1.1.3
9 fastapi==0.75.0
10 gunicorn==20.1.0
11 h11==0.13.0
12 httpx==0.2.0

```

Unsuccessful Pipeline

Automatic Deployment

When the GitHub pipeline is successful, then Heroku will automatically restart and update the website with the new code. This was established to reduce manual labor required to restart the server due to new code changes.

Salesforce Platform

HEROKU

Jump to Favorites, Apps, Pipelines, Spaces...

Deployment method

Heroku Git Use Heroku CLI

GitHub **Connected**

Container Registry Use Heroku CLI

App connected to GitHub

Code diffs, manual and auto deploys are available for this app.

Connected to EpidemicEagle/SENG3011_EpidemicEagle by Saheeb

Disconnect...

Releases in the [activity feed](#) link to GitHub to view commit diffs

Automatically deploys from pipeline

Automatic deploys

Enables a chosen branch to be automatically deployed to this app.

You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions [here](#).

Automatic deploys from pipeline are enabled

Every push to pipeline will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch in GitHub is always in a deployable state and any tests have passed before you push. [Learn more](#).

☒ Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

Disable Automatic Deploys

Manual deploy

Deploy the current state of a branch to this app.

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. [Learn more](#).

Choose a branch to deploy

pipeline

Deploy Branch

heroku.com Blogs Careers Documentation **Support**

Terms of Service Privacy Cookies © 2022 Salesforce.com

Insights Settings

main 5 branches 0 tags

Go to file Add file Code

About

No description, website, or topics provided.

Readme

0 stars

0 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

Environments 1

epidemic-eagle Active

Languages

Python 90.0% HTML 7.4%

Shell 1.7% Other 0.9%

aashek Merge pull request #2 from EpidemicEagle/pipeline

8e5e851 19 hours ago 49 commits

.github/workflows	pytest fix	3 days ago
PHASE_1	Last commit before push to main	19 hours ago
PHASE_2	Repository Setup	18 days ago
Reports	Renamed files	6 days ago
static	Fake response added. Files moved.	3 days ago
templates	Fake response added. Files moved.	3 days ago
README.md	Last commit before push to main	19 hours ago
phase1.sh	Fake response added.	3 days ago
pytest.ini	Fixed pytest errors.	3 days ago
requirements.txt	Fixed pytest errors.	3 days ago
setup.sh	Restructured files	4 days ago

README.md

EpidemicEagle

SETUP

Overview of Tests

Testing helps provide proof of correctness for software. Although our team has in-depth knowledge about setting an environment for automatic and manual testing, we ran out of time to write testing functions for our API. However some sample ones are provided below.

Unit Testing

Unit testing is one of the most important parts of testing. By ensuring that each small part of our software is able to work independently on a consistent basis, we can be more confident that they will work together in our system to achieve the desired result. If we were simply to test the system all at once, it would be difficult to pinpoint any errors that may occur. Unit testing means that we know exactly what parts of the software are breaking, saving time and resources when searching for problems.

Parameter Testing

One of the tests is a parameter test, meaning we test the parameters and whether they are the right type. This is necessary as we cannot guarantee that the users will put in the correct parameters. If they manage to put incorrect parameters in, our software needs to be able to return an error to the user, with an explanation of what caused the error, helping protect our code from breaking.

For example, a call to `/api/articles/"1"` would return a 422 Error for Invalid Type.

A helper function is described below.

```
def arg_test(url):
    params={"start_date":"2018-xx-xx xx:xx:xx", "end_date":"2019-xx-xx xx:xx:xx", "key_terms":"covid, measles", "location":"Shanghai"}

    response = client.get(url, params = {"start_date":"2018-xx-xx xx:xx:xx", "end_date":"2019-xx-xx xx:xx:xx", "key_terms":"covid, measles"})
    assert response.status_code == 422

    response = client.get(url, params = {"start_date":"2018-xx-xx xx:xx:xx", "end_date":"2019-xx-xx xx:xx:xx", "location":"Shanghai"})
    assert response.status_code == 422

    response = client.get(url, params = {"end_date":"2019-xx-xx xx:xx:xx", "key_terms":"covid, measles", "location":"Shanghai"})
    assert response.status_code == 422

    response = client.get(url, params = {"start_date":"2018-xx-xx xx:xx:xx", "key_terms":"covid, measles", "location":"Shanghai"})
    assert response.status_code == 422

    response = client.get(url, params = {**params, 'start_date': 'two days from now'})
    assert response.status_code == 422

    response = client.get(url, params = {**params, 'end_date': 'today'})
    assert response.status_code == 422

    response = client.get(url, params = {**params, 'start_date': '2019-20-20'})
    assert response.status_code == 422

    response = client.get(url, params = {**params, 'end_date': '2019-20-20'})
    assert response.status_code == 422
```

Then this helper function is called for the three routes with these required parameters.

```
29  def test_wrong_args():
30      arg_test("/api/search")
31      arg_test("/api/articles")
32      arg_test("/api/reports")
33
```

Acceptance Testing

Another type of testing the acceptance tests, which ensures that correct parameters input achieve the desired output. This means that when users are calling the API, they find all the information they are looking for, be it multiple reports on a common disease over the past few years or a single report on a new disease in the last month. By comprehensively testing with correct parameters with results we know to be true, we ensure that our users are satisfied, thus accepting that the API works.

Limitations of Testing

Some parts of our API cannot be tested, due to the variance in results. This is namely when new ProMed articles are released, as they won't be in our test suite, and the new articles may change the results of previous tests. Thus, it is difficult to test new material that ProMed releases.

Web Scraper (Scrapy)

Currently we cannot run automated tests on our web scraper. It would be too difficult to test daily results, since we do not know how many posts were created on a single day. We can do some limited manual testing by checking our database each day to see if new entries are being made, though this is not guaranteed to be correct as we won't be able to ensure that all reports and articles are being generated. Manual testing also takes lots of time, which is inefficient and often unviable as the team already has limited time to work on the project.

Log Files

Logs are an extremely helpful feedback tool when deploying an API. They can keep track of requests made to the API and store information about each request.

Heroku offers an inbuilt logging function whenever calls are made to the API it is hosting, saving the logs in Heroku. Using the Heroku CLI, we can read these logs and store them in a text file. By typing in "Heroku logs" into the CLI, Heroku will display the 100 most recent logs. A "-n 200" tag after the command will display the 200 most recent logs and can be changed to the desired number of logs. Heroku logs store information containing the timestamp, source, and message, which includes status code and message length.

```
38 2022-03-15T03:22:28.472954+00:00 app[web.1]: [2022-03-15 03:22:28 +0000] [9] [INFO] Started server process [9]
39 2022-03-15T03:22:28.472954+00:00 app[web.1]: [2022-03-15 03:22:28 +0000] [12] [INFO] Started server process [12]
40 2022-03-15T03:22:28.472954+00:00 app[web.1]: [2022-03-15 03:22:28 +0000] [9] [INFO] Waiting for application startup.
41 2022-03-15T03:22:28.473016+00:00 app[web.1]: [2022-03-15 03:22:28 +0000] [12] [INFO] Waiting for application startup.
42 2022-03-15T03:22:28.473202+00:00 app[web.1]: [2022-03-15 03:22:28 +0000] [9] [INFO] Application startup complete.
43 2022-03-15T03:22:28.473243+00:00 app[web.1]: [2022-03-15 03:22:28 +0000] [12] [INFO] Application startup complete.
44 2022-03-15T03:22:49.992102+00:00 app[web.1]: 129.94.8.207:0 - "GET /docs HTTP/1.1" 200
45 2022-03-15T03:22:49.993585+00:00 heroku[router]: at=info method=GET path="/docs" host=epidemic-eagle.herokuapp.com request_id=8b97cd91-fe18-459f-9dc4-856d2fe84575 fwd="129.94.8.207" dyno=web.1 connect=
46 2022-03-15T03:22:50.529795+00:00 app[web.1]: 129.94.8.207:0 - "GET /api/v1/openapi.json HTTP/1.1" 200
47 2022-03-15T03:22:50.531376+00:00 heroku[router]: at=info method=GET path="/api/v1/openapi.json" host=epidemic-eagle.herokuapp.com request_id=abe02bee-584e-48ae-ba13-223ac9c126ed fwd="129.94.8.207" dyno=web.1 connect=
48 2022-03-15T03:23:46.184133+00:00 app[web.1]: 129.94.8.207:0 - "GET /api/search?key_terms=covid&location=sydney&start_date=2018-xx-xx%20xx%3Axx%3Axx&end_date=2019-xx-xx%20xx%3Axx%3Axx HTTP/1.1" 200
49 2022-03-15T03:23:46.185648+00:00 heroku[router]: at=info method=GET path="/api/search?key_terms=covid&location=sydney&start_date=2018-xx-xx%20xx%3Axx%3Axx&end_date=2019-xx-xx%20xx%3Axx%3Axx" host=epidemic-eagle.herokuapp.com request_id=039b507f-33a3-45ca-9868-338b913cb335 fwd="129.94.8.207" dyno=web.1 connect=
50 2022-03-15T03:23:56.030735+00:00 app[web.1]: 129.94.8.207:0 - "GET /api/search?key_terms=measles&location=sydney&start_date=2018-xx-xx%20xx%3Axx%3Axx&end_date=2019-xx-xx%20xx%3Axx%3Axx HTTP/1.1" 200
51 2022-03-15T03:23:56.032279+00:00 heroku[router]: at=info method=GET path="/api/search?key_terms=measles&location=sydney&start_date=2018-xx-xx%20xx%3Axx%3Axx&end_date=2019-xx-xx%20xx%3Axx%3Axx" host=epidemic-eagle.herokuapp.com request_id=204839ce-5995-42d7-8418-a8a39f118662 fwd="129.94.8.207" dyno=web.1 connect=
52 2022-03-15T03:43:29.453940+00:00 heroku[router]: at=info method=GET path="/docs" host=epidemic-eagle.herokuapp.com request_id=039b507f-33a3-45ca-9868-338b913cb335 fwd="129.94.8.207" dyno=web.1 connect=
53 2022-03-15T03:43:29.455616+00:00 app[web.1]: 129.94.8.207:0 - "GET /docs HTTP/1.1" 200
54 2022-03-15T03:43:30.077956+00:00 heroku[router]: at=info method=GET path="/api/v1/openapi.json" host=epidemic-eagle.herokuapp.com request_id=039b507f-33a3-45ca-9868-338b913cb335 fwd="129.94.8.207" dyno=web.1 connect=
55 2022-03-15T03:43:30.078645+00:00 app[web.1]: 129.94.8.207:0 - "GET /api/v1/openapi.json HTTP/1.1" 200
56 2022-03-15T03:43:40.300847+00:00 app[web.1]: 52.114.32.28:0 - "GET /docs HTTP/1.1" 200
57 2022-03-15T03:43:40.301094+00:00 heroku[router]: at=info method=GET path="/docs" host=epidemic-eagle.herokuapp.com request_id=afb99749-1ead-412e-860f-2d7c6d9d6fb6 fwd="52.114.32.28" dyno=web.1 connect=
58 2022-03-15T04:07:50.264324+00:00 app[web.1]: 120.151.212.251:0 - "GET /docs HTTP/1.1" 200
59 2022-03-15T04:07:50.265574+00:00 heroku[router]: at=info method=GET path="/docs" host=epidemic-eagle.herokuapp.com request_id=234a5bdd-6bc2-45c7-9438-298979d7dcf2 fwd="120.151.212.251" dyno=web.1 connect=
60 2022-03-15T04:07:50.774786+00:00 app[web.1]: 120.151.212.251:0 - "GET /api/v1/openapi.json HTTP/1.1" 200
61 2022-03-15T04:07:50.776043+00:00 heroku[router]: at=info method=GET path="/api/v1/openapi.json" host=epidemic-eagle.herokuapp.com request_id=7890b25c-bb49-44f1-b63e-0fb7e576c456 fwd="120.151.212.251" dyno=web.1 connect=
62 2022-03-15T04:07:59.256430+00:00 app[web.1]: 120.151.212.251:0 - "GET /?item=1 HTTP/1.1" 200
63 2022-03-15T04:07:59.257573+00:00 heroku[router]: at=info method=GET path="/?item=1" host=epidemic-eagle.herokuapp.com request_id=2d5f6cc6-c5cb-4b3e-854b-d2c910139681 fwd="120.151.212.251" dyno=web.1 connect=
64 2022-03-15T04:41:48.171373+00:00 heroku[web.1]: Idling
65 2022-03-15T04:41:48.207979+00:00 heroku[web.1]: State changed from up to down
66 2022-03-15T04:41:51.000740+00:00 heroku[web.1]: Stopping all processes with SIGTERM
67 2022-03-15T04:41:51.079339+00:00 app[web.1]: [2022-03-15 04:41:51 +0000] [4] [INFO] Handling signal: term
```

Benefit of Logs

Logs are useful as they contain runtime information for users. By analyzing this information, we can determine how to optimize our software so that users have a more enjoyable experience. This could be anything from how much traffic we are receiving at certain times, how fast each request is being processed, which requests are being called the most and if processes are returning errors.

Analyzing the logs will allow us to see what routes are called the most, and allows us to further improve the usability of those routes. It can also show us what routes are not being used, and determines whether or not the functionality of a route is viable to a client.