Development & Testing Report of

COVID-19 INDIA API

BY TARUN SINGH

26th April, 2020

Contents

Note:

- **Unit Testing** reports are provided at the end of each content.
- Repository link: https://github.com/10xtarun/covid-india-api
- **Documentation** is available at: https://documenter.getpostman.com/view/10923795/SzfB16wa
- API is deployed on Heroku

Major issues faced during development:

1. Scraping - data was not in proper tags, had to filter a lot
2. Scraper updating data - the scraper has to fetch data at intervals, solved with request logger
3. Data in Hindi - came up with custom script to convert english numbers into Hindi numbered string

API for COVID-19 Data from India

Idea brief: Currently no API exists for Covid-19 data w.r.t. India. The Indian government website displays the data on it's official website. The idea is to fetch data from this website and make it available via an API.

Outcome: Currently the data is already available on the official website but making it available via an API can be useful for other people, where they can just fetch respective information instead of getting everything from scratch.

Features:

1. On **/api/v1/states** ➜ should give data of all states in India

2. On **/api/v1/state/:name** ➜ should data related to state given in the parameters (state-name)

2. On **"?lang=hindi"** or **"?lang=english"** ➜ should return the data in hindi or english language, w.r.t. query

How to do:

1. To scrap the data from the official website
2. To store and make available the data via the API

Tech-stack:

1. Web Scraping ➜ NodeJs, Cheerio, node-fetch
2. API - NodeJS, ExpressJs
3. Testing - Mocha, Chai, SuperTest

Website Link:

Official website: MoHFW | Home

The web scraper

Libraries used:-

Fetch-node and cheerio

- Fetch-node to request the website URL
- Cheerio to extract required data from fetched website

Working with cheerio:

ISSUE:

- The website being scraped had a very bad set-up of the marks, following issues were faced.
    - Table data had lot of spaces and tabs
    - After that, the children data also had lot of white space and new lines
    - Even after removing, many arrays were left, with the help of filter method, the data was cleaned

Resolve Code:

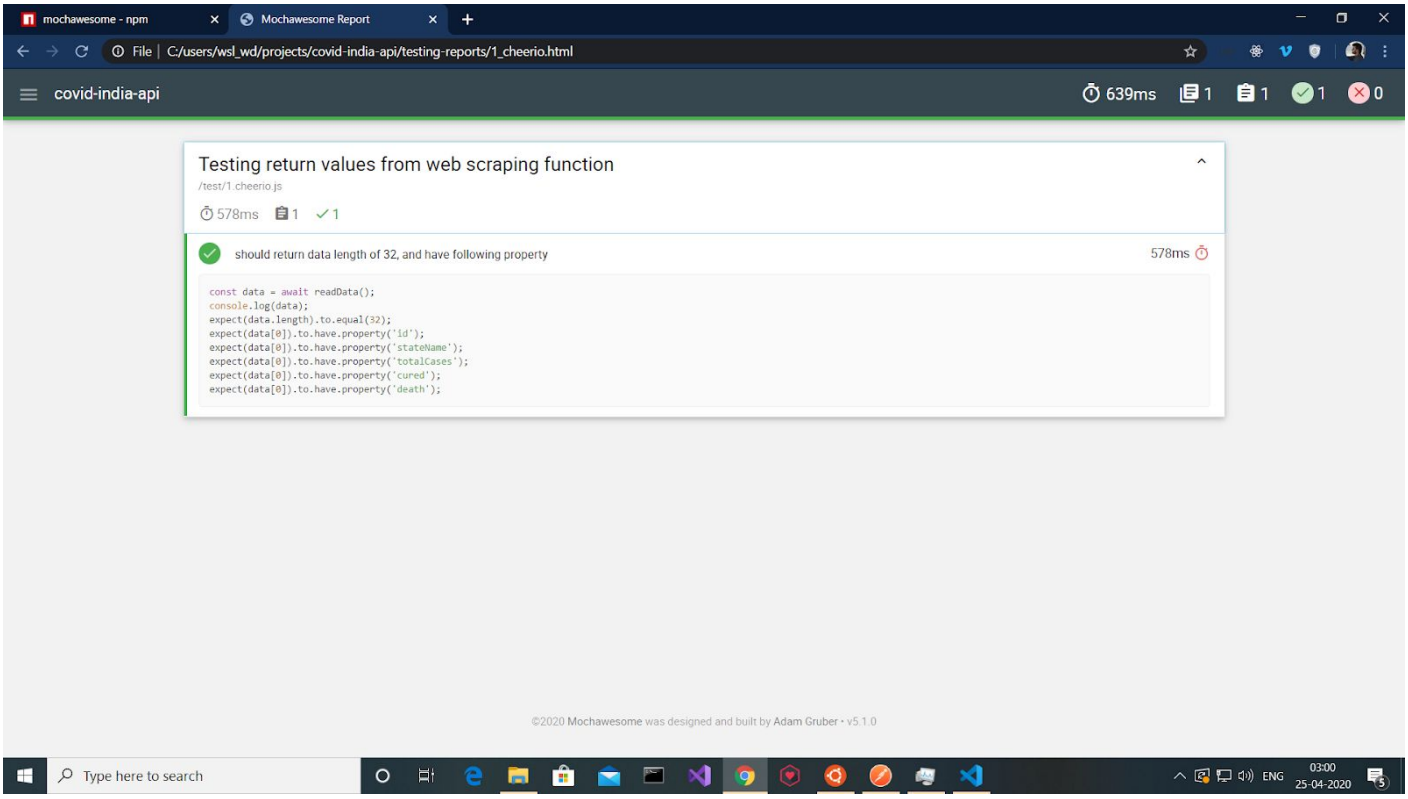
```
var stateText = stateData.contents().text().trim();
     stateText = stateText.split('\n\t\n\t');

     for (var i = 0; i < stateText.length; i++) {
       if (stateText[i] == '' || stateText[i] == '\n') {
         stateText.splice(i, 1);
       }
     }
     for (var i = 0; i < stateText.length; i++) {
       if (stateText[i] == '' || stateText[i] == '\n') {
         stateText.splice(i, 1);
       }
       temp = stateText[i].split('\n\t').filter((ele) => {
         return ele != '\n' && ele != '' && ele != "'";
       });
       jsonData.push({
         id: parseInt(temp[0]),
         stateName: temp[1],
         totalCases: parseInt(temp[2]),
         cured: parseInt(temp[3]),
         death: parseInt(temp[4]),
       });
       if (parseInt(temp[0]) == 32) {
         break;
       }
     }
```

Testing:

- To check scraper function is working properly or not
- Doing by testing the data it dumped into the folder

Testing Report:



Loop the scrapper function to update the data after intervals

Next Function:

1. The scrapper is only **triggered** when the scrapper function is called, but the covid-19 data is continuously updated on the original website, so we have to re-run the scrapper again and again to keep up with updated data.
2. One way was to use **Timers** methods provided in Node but after trying out a few experiments, this was not working as per required, this is because Node is single threaded and Timers are sync functions.
3. The other way I realised was to call the scrapper function after a certain number of successful requests from the users have been made.
4. For example:- server is running and users are making requests to the api, let say after 5 request we call the scrapper function to update the data
5. The only problem with this method is, “after how many requests the function should be called” and when the function will be called, the server will respond slowly for that particular request.

How it is done:

- We need to have logger middleware in the server to, which log the every request made by users and it will be stored in “.json” file
- Once the number of requests has met the predefined number, it will call the scrapper function and will fetch and store the updated data in the “.json” file.

A little help from 💯 Counting requests in your Node.js + Express application , to make request logging.

Request Logger

To log the successful request and store in JSON format

What type of requests are expected?

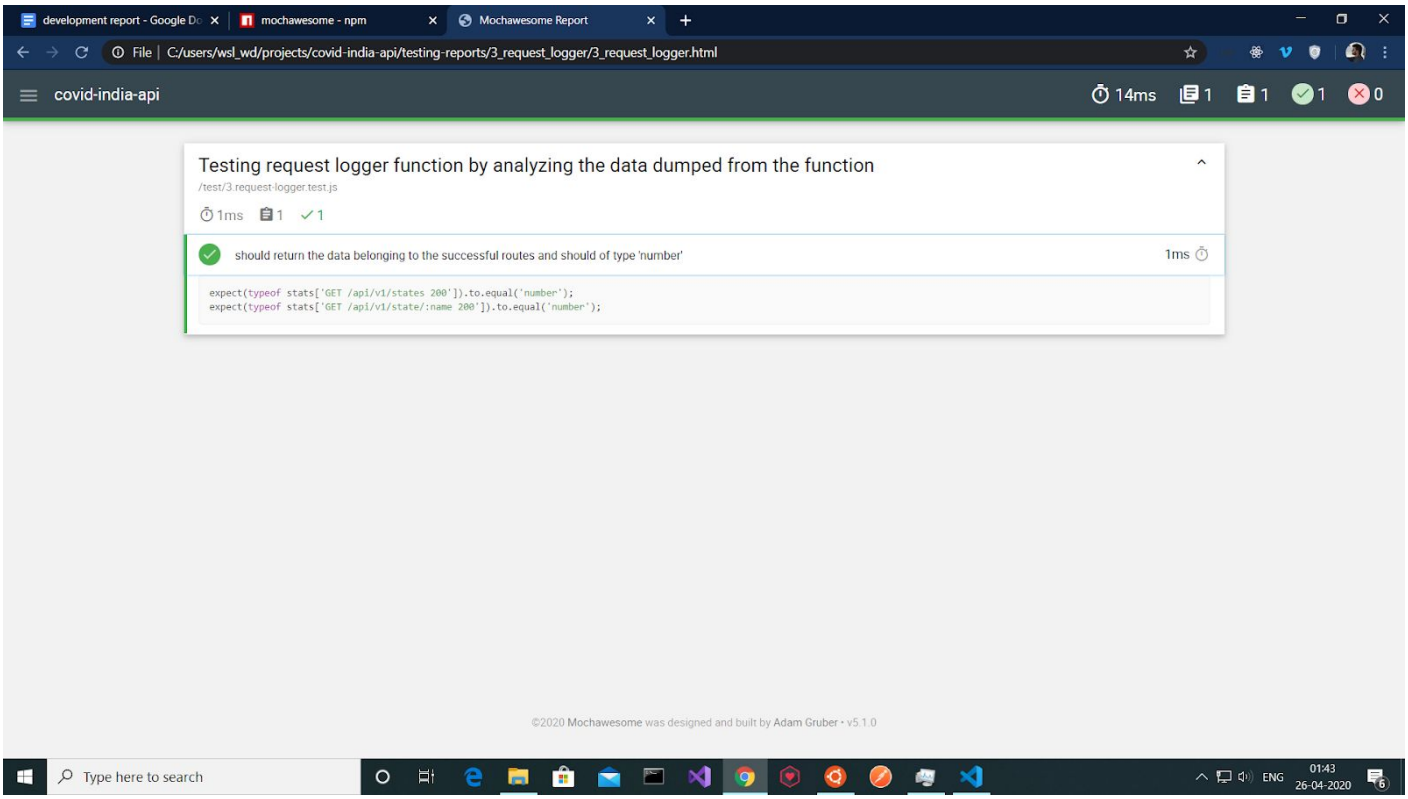
➜ These are classified on the basis of statusCode returned and requested route

- The defined routes in the beginning of the reported will be counted as success on return of status 200 OK
- These will be logged and saved in **“stats/request-logs.json”,** JSON format.

Test Condition:

1. Check logger file, what request are stored
2. How many request are stored

Testing Report:



Providing Data in Hindi

This was a bit of a challenging task, the feature of the API was to provide information of states in “Hindi” so that users can view the information hindi if required.

ISSUE:

- Since google translate API was paid but we can use free tier version
- But since the covid-19 information will be updated at intervals, which may turn into too many frequent requests
- If you request to google-translate-api too frequently it will block and will return an internal server error.

RESOLVE:

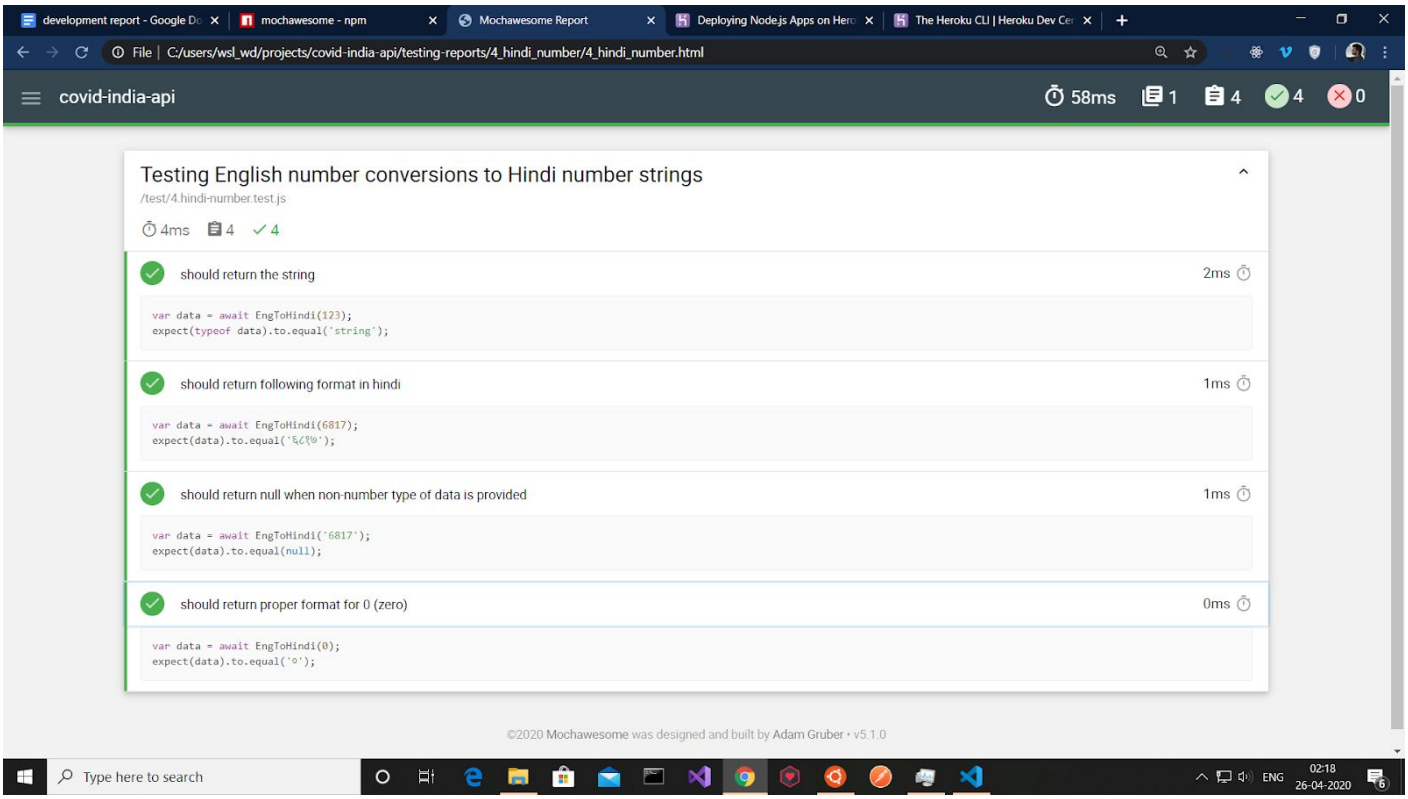
- If we observe the json extracted by scrapper, the data itself is small and only numbers are needed to be converted in hindi
- Example: totalCases = 121, cured = 12, death = 10
- These are number data, so by writing simple script, can change the number into hindi string
- Also to note the there were only 32 state names which was manually converted to hindi names

Script that converts Numbers to strings in Hindi:

```
var fs = require('fs');
function converter(num) {
  switch (num) {    case 0:
      return '०';
    case 1:
      return '१';
    case 2:
      return '२';
    case 3:
      return '३';
    case 4:
      return '४';
    case 5:
      return '५';
    case 6:
      return '६';
    case 7:
      return '७';
    case 8:
      return '८';
    case 9:
      return '९';
    default:
      return '-1';
  }
```

```
}
module.exports.EngToHindi = function (num) {
  var str = '';
  if (typeof num == 'number') {
    if (num == 0) {
      return '०';
    }
    while (num) {
      str = converter(num % 10) + str;
      num = Math.floor(num / 10);
    }
    return str;
  } else {
    return 'wrong format of number provided.';
  }
};
```

Testing Report:



API Route Testing

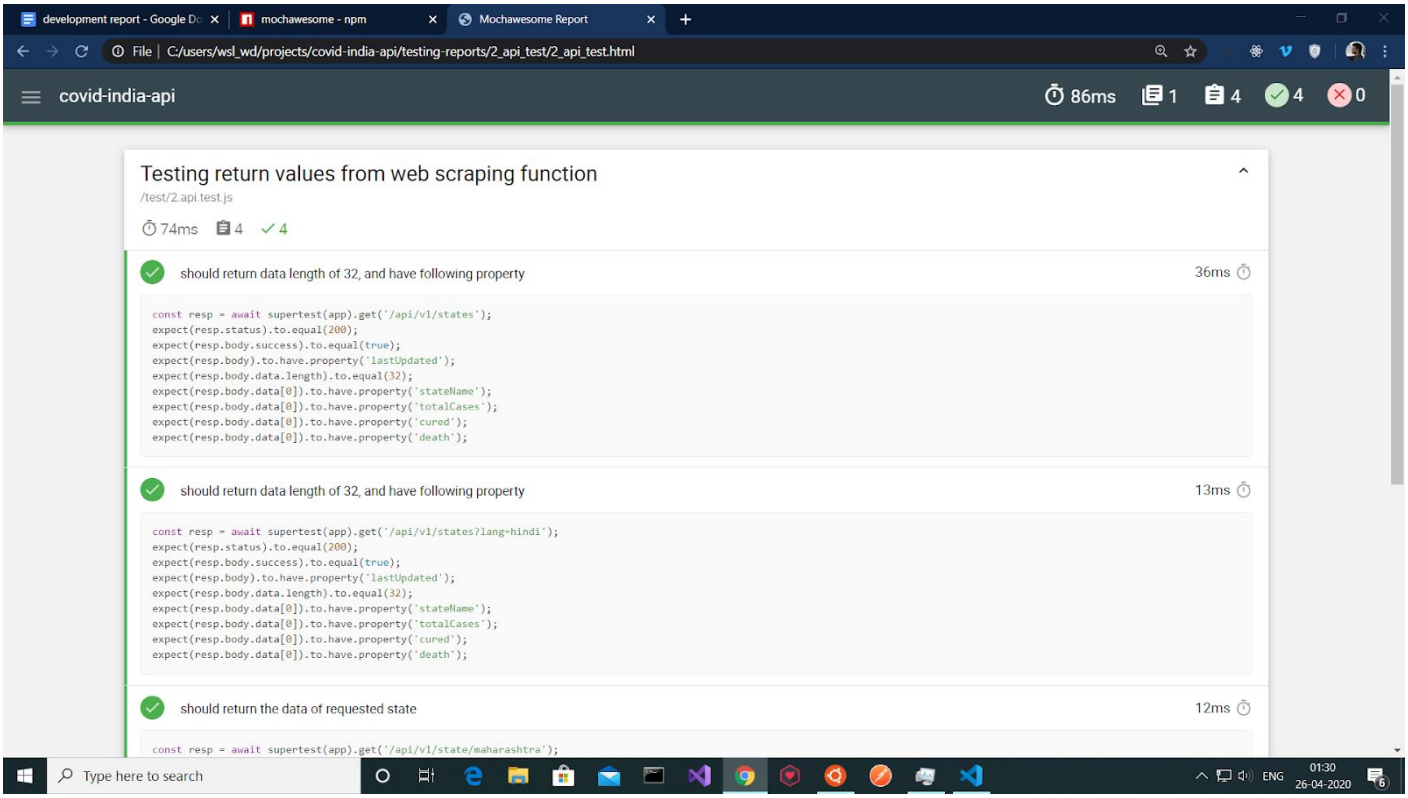
Since the testing of request logger involves use of a server (express app), wil also test the routes and data returned by our full-fledge API.

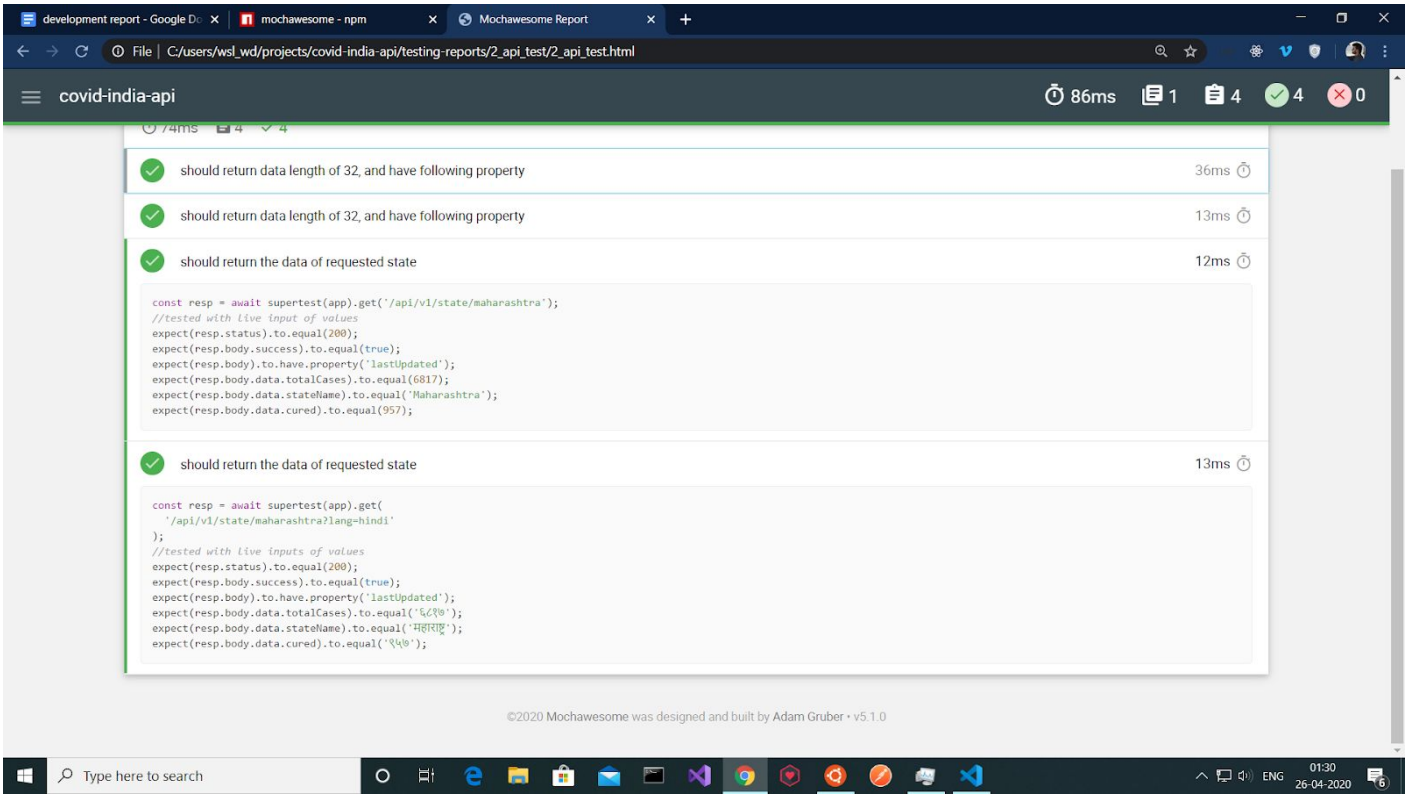
Test Conditions:

1. Fetch all states in english ➜ “/api/v1/states” or “/api/v1/states?lang=english”
    a. {200, succes: true, LastUpdatedAt, data_in_english}
2. Fetch all states in hindi ➜ “/api/v1/states?lang=hindi”
    a. {200, success: true, LastUpdatedAt, data_in_hindi}
3. Fetch specific state in english ➜ “/api/v1/state/name” or “/api/v1/state/name?lang=english”
    a. {200, success: true, LastUpdated, data_in_english}
4. Fetch specific state in hindi ➜ “/api/v1/state/name?lang=hindi”
    a. {200, success: true, LastUpdatedAt, data_in_hindi}

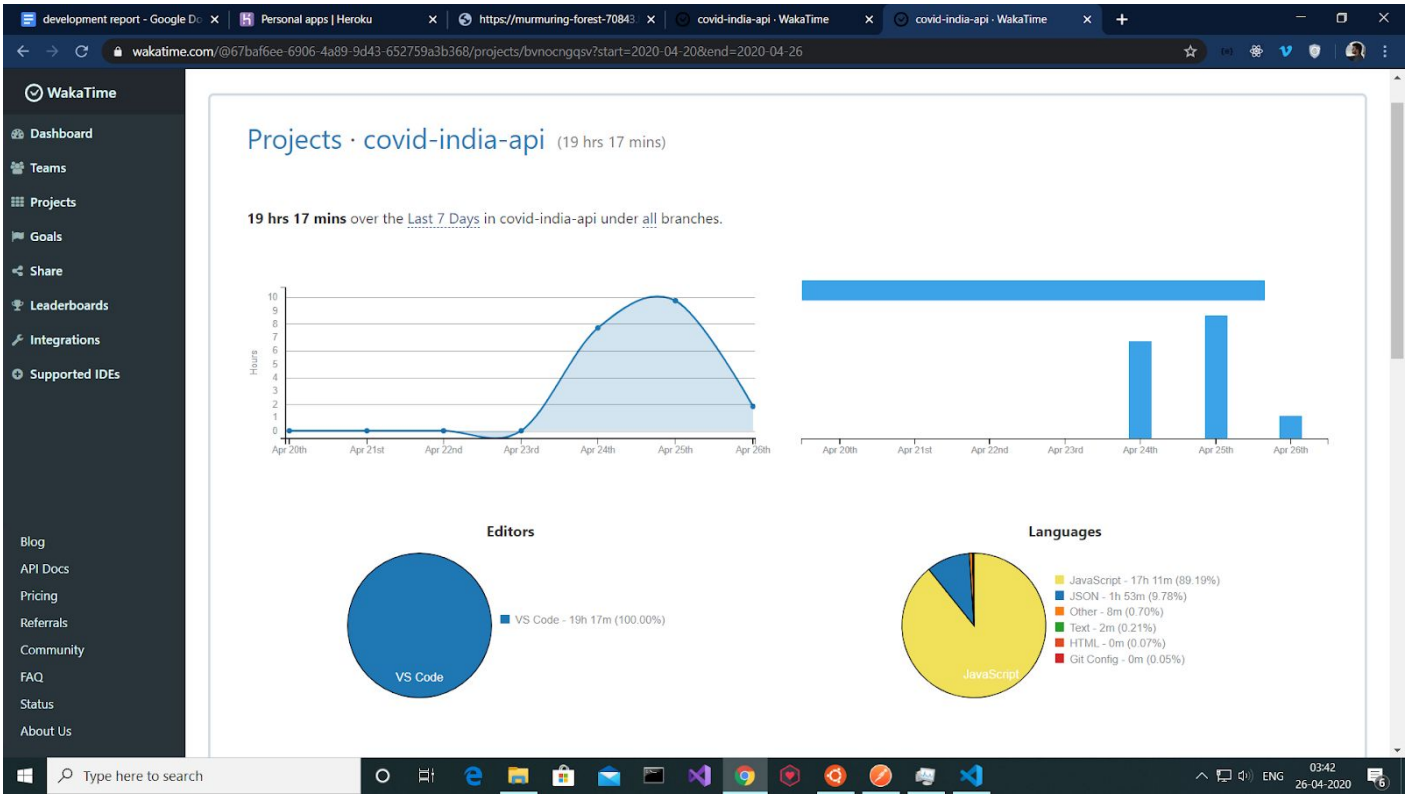
Failed routes should return➜ {404, success: false, data: null}

Testing Reports:




development report - Google Do
mochawesome - npm
Mochawesome Report
File | C:/users/wsl_wd/projects/covid-india-api/testing-reports/2_api_test/2_api_test.html
covid-india-api
86ms 1 4 4 0
should return data length of 32, and have following property 36ms
should return data length of 32, and have following property 13ms
should return the data of requested state 12ms
const resp = await supertest(app).get('/api/v1/state/maharashtra');
//tested with live input of values
expect(resp.status).to.equal(200);
expect(resp.body.success).to.equal(true);
expect(resp.body).to.have.property('lastUpdated');
expect(resp.body.data.totalCases).to.equal(6817);
expect(resp.body.data.stateName).to.equal('Maharashtra');
expect(resp.body.data.cured).to.equal(957);
should return the data of requested state 13ms
const resp = await supertest(app).get(
'/api/v1/state/maharashtra?lang=hindi'
);
//tested with live inputs of values
expect(resp.status).to.equal(200);
expect(resp.body.success).to.equal(true);
expect(resp.body).to.have.property('lastUpdated');
expect(resp.body.data.totalCases).to.equal('६८१७');
expect(resp.body.data.stateName).to.equal('महाराष्ट्र');
expect(resp.body.data.cured).to.equal('९५७');
©2020 Mochawesome was designed and built by Adam Gruber • v5.1.0
Type here to search
ENG
01:30
26-04-2020

Work-Time report for the development of API:



Conclusion

It almost took 20 coding hours to build this API. Faced a lot of issues and errors, learned and improved the code. The API is able to serve the latest covid data w.r.t. to India and with additional feature of providing information in hindi. We can say this is a mini version of the API, because currently no security features are provided for the API such as Access Token to limit or restrict malicious requests to the API.