API Testing Report

Testing processes

Our team first conducted some basic cases to see if the API is functioning normally. For example, we inputted a date range to see if the API returns the articles that have the publication date between that range, and we also tried out adding a key term or location to get the filtered results.
Then, we added some edge cases, such as putting the same date in the start and end dates to see if the API gets the articles from that specific date. And we tried to fill in all the parameters to narrow down the results to one specific article.

Finally, we provided some input data that would cause no articles retrieved. For example, we inputted wrong format of dates to get 400 ("Invalid parameters") and misspelled locations to get 404("Article not found"). This is to test if our API can handle these special inputs and return the error message instead of crashing itself.

During the testing, we indeed found some bugs and problems with our API. At first, when we were testing with the inputs containing locations, we could not get any articles in return. We realised there's something wrong with the scrapping and updating of the database. So, we went back to the code to fix it. After that, we rerun the test and added more cases with different locations to test out the correctness again.

By adding more edge cases, we can discover the potential problems in our codes. And we can then fix the bugs according to the test results, following by more tests to improve the correctness of the code.

Testing Environments and tools

For the API Testing, our team planned to use a software called Postman at first, as it has a clear graphic interface. We could manually enter different inputs to carry out the unit test. However, it is extremely inefficient to do so, as it takes too much time and effort to input data and take screenshots of the results. Instead, we could just write a script that run the codes for us and compare the expected results with the actual outcomes. Since test scripts need to be written in Javascript in Postman, we decided not to stick with Postman. Our script now is in Python because we're all familiar with the language and it's easier for us to test out the correctness of the API.

Testing Limitations

Since the scrapper keeps scrapping every day, the database will be updated frequently. We can't really test out if the API has got the exact search result that we want. For example, when we input the key term "Ebola", we can only see from the response state code if the articles are retrieved successfully. And by comparing the response texts of different key terms, we can see if the API is returning the same thing for different inputs. But we cannot tell if there are any articles that we're missing unless we manually search on the data source website and check for each article that contains the key term.

Overview

Overall, our test includes the correct inputs, edge cases and some invalid inputs, which tests out different responses from our API, by fixing the problems and improving our codes, the API passed all the tests and we'll try to add more test cases to improve our program and make it more stable.