**Additional Notes**

Regarding the features that I couldn’t implement due to constraints, this is how I would have implemented them:

**Feature one**: Allow user defined input in the test suite.

* **Description**: Currently, the test suite has a feature in which the test data is generated automatically in a random manner using the fakerJS library. To complement this feature, I would have liked to add another feature where the test could also be run with user defined input and not necessarily with faker js data.
* **Implementation**: Firstly, I would have added an environment variable called fakeData If the value of fakeData is true the tests would use fakerjs random data, else, the test would load the test data from a config.json file.

In order to load the test data from the file I would have implemented some code that read the data using node fs module. This code would contain different methods so that each test can access its configuration on the config.json file.

The structure of the JSON file would contain a hierarchy similar to the one in that is used in the project to organize tests.

{

    "test-type": {

        "test-scenario": {

            "test-case": {

                "object": {

                    "value1": 87,

                    "value2": "Dante",

                    "value2": []

                },

            },

      },

}

By using this feature, the tester can also test specific test cases without needing to change of testing tool.

**This feature wasn’t executed due to a time constraint.**

**Feature two:** Allow the execution of load tests.

* **Description:** Implement a feature so that by using a different test command, the test suite triggers load tests.
* **Implementation:** To implement this feature I would have added a new node library called artillery. With artillery I would have created a new folder under the tests directory called Load. Under this directory I would have added four .yml files, each one corresponding to a test URL. In each .yml file I would have added the configuration of the load test that is going to be performed. Finally, the tests would be triggered using the artillery commands, like:

artillery run post.yml

* **Constraints:** I didn’t implement this feature for two reasons. Firstly, I was worried about swagger checking the number of requests from a determined ip and blocking my ip. Secondly, I decided to leave this feature for the last part, as I am new to artillery and I didn’t want to lose much time getting to know the technology, so I gave priority to the logic and negative tests first.