**Test suite using Mocha and Chai**

**Importance of Test Suite:**

Test suite plays a significant role to make sure if the various functionalities in the application are working as expected. It aids the developer to know if the features, api endpoints in the backend application are free from flaws or defects.

* It ensures good test coverage
* Improves the quality of software
* Check if the application meets the requirements or not

**Test Cases:**

Test Cases are a set of instructions which the tester is expected to follow to achieve a specific output or step by step instructions to verify if api behaves as it is required to behave. It also guides them through the steps of the test.

There are various types of testing in the software applications. We mostly used any of them based on the requirement. In our application we have written our test cases using Unit testing.

**What is Unit Testing:**

Usually in the Unit testing, the api functionalities in an application are tested during the development phase. A function that is written in the block of code and that is tested to verify the api accuracy using various unit testing frameworks.When we perform tests as part of the development process, our code is automatically going to be designed better than if you just wrote the functions

We have written our test cases using Mocha and Chai frameworks. Mocha is a feature-rich JavaScript test framework which runs on Node.js. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases. Whereas Chai is the assertion library for node and the browser that paired with any javascript framework.

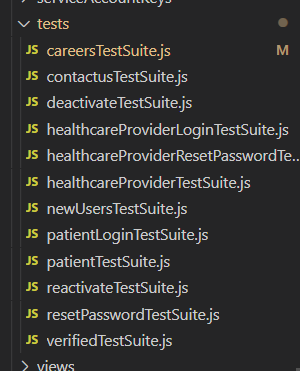
Here are the links for the Chai and Mocha documentation

<https://www.chaijs.com/>

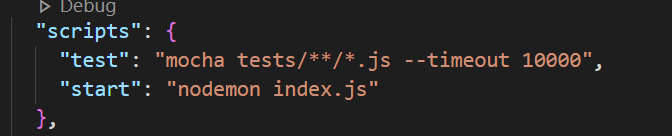
<https://mochajs.org/>

**Steps for writing Test Suite:**

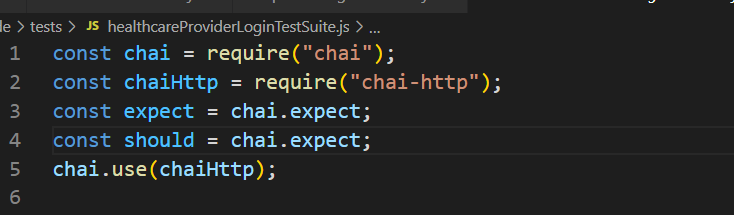
* Create a separate Test folder in the backend application
* Navigate to the node folder using the command-- cd node
* Each controller will have a separate test file as shown below. Our application has 13 controllers with all the api endpoints and the test folder will have the 13 files for these controllers.

.post('/')

* After navigating to the node folder, we need to install various libraries and dependencies.
* First install mocha using the following command------npm install mocha --save-dev
* Further install Chai using the following command ----- npm install chai --save-dev.
* In the package.json, under the scripts add the following code as shown below



* Install chai-http dependency using the command---- npm install chai-http
* We need to define all these installed libraries in each of the test files as below



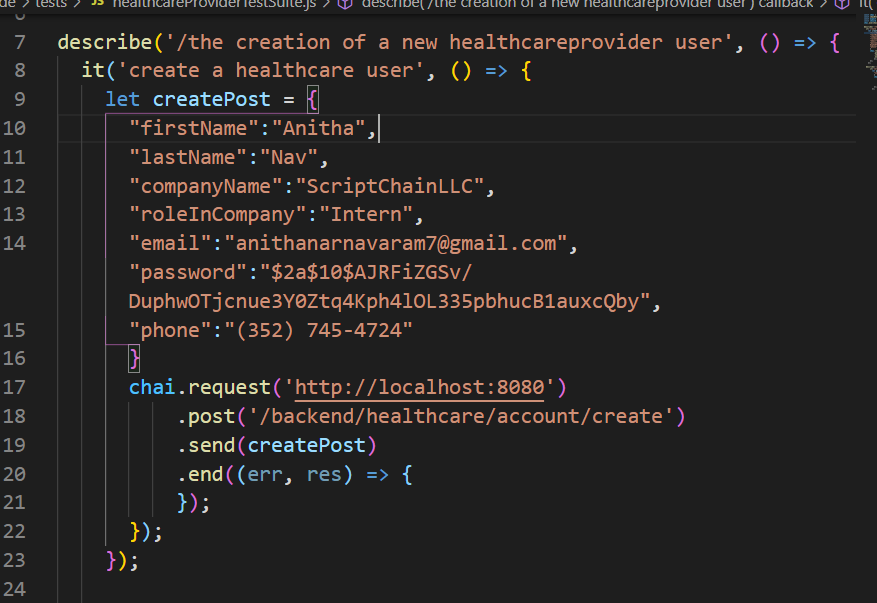
* To run all the test cases irrespective of the file run the following command----npm test

**Steps to write the test cases for each request:**

There are various approaches to write test cases based on the requirement. There is no need of being in the same format as we define below. We can use various assertion libraries like should, expect , assert.

**Test case for the POST request:**

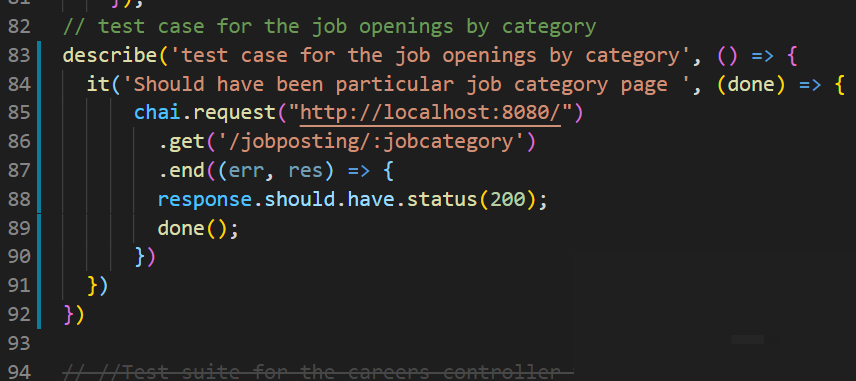
Here, we are just showing and explaining how to write a test case for a POST request and the request body will change for each endpoint. We need to first know what we are sending and testing. Also, it is upto the developer which field to check.



* Here, describe() function helps to explain what we are testing
* Whereas it() function also further provides the clarity of that particular test.
* As it is post the method, The idea is to send the data to the specified endpoint and check if that is working properly or not.
* From the above picture, we are testing the healthcare register by sending the body. The body is stored in the createPost with all the required fields along with the values.
* **Chai.request()** function will have a backend link which is the base url of the application.
* **.post()** will have the actual endpoint to where we are sending the data for healthcare registration.
* Here, we can observe that .post function has endpoint /backend/healthcare/account/create
* Under **.send()** we send the createPost(which is the data that we are sending.
* We can add various other assertions to these endpoints as below

**Test case for the GET request:**

Get test case is a bit different than the Post request as we are not sending any body to the endpoint.



* Here, we can observe that few functions like describe(), it() are same
* Under the Chai.request() function we are mentioning the base url of the application
* As it is the get method, .get() function is used for defining the specific endpoint like /jobposting/:jobcategory
* Now we need to verify if this particular endpoint is able to fetch requests from the specified get method.
* Status 200 is obtained when we try to fetch the information from the get method. So, it is significant to check the status 200. If the response is 200 then the particular endpoint is working fine.
* This particular status check is verified using response.should.have.status(200).
* There are various ways to check these status. The above one is not the only way to check the status like expect(res).to.have.status(200) and many more
* If the correct endpoint is not given then the status should be 404. So, we can also write tests for 404 status as well without restricting to only 200 status checks.