I could not run the tests on the openweathermap api due to complexity of the code so I created my own weather application based on the weather attributes from some of the JSON files from openweathermap.org and online tutorials. This data can be found in **climate\data\index.js** or **climate\test\data\index.js** – this one is used for the tests.

My application can handle GET requests and return JSON; modify data through PUT requests. It uses the express, mocha, should, and supertest modules. The data can also be viewed through the JSON format in a browser by using, for example, <http://localhost:3000/weather/06>. 06 is the Id for the weather location.

To get the tests working , make sure you copy the climate folder to the desktop and in your GIT terminal write **cd desktop**, then **cd climate**. After this write **npm install**. I have installed all dependencies in the folder with **npm install –save-dev mocha, npm install –save-dev should,** and **npm install –save-dev supertest.**  I also installed mocha globally with **npm install –g mocha.**

The coded tests are located in **\climate\test\WeatherSpec.js**. Make sure you are in the climate directory in the GIT terminal. To run the tests, type **mocha** in the GIT terminal and you will get the returned passing tests.

**Test\_1**

The first test should return valid weather data for id:01. (The id is to keep track of weather locations). The supertest module is calling application app as an argument. The first chained call is a get request for weather id 01. Next, we are expecting an OK 200 response. Finally, we call an anonymous function as an argument which sends us an error or response. After that, I asserted what the response should be. The status object should equal 200. If the status doesn’t not equal 200, should is going to call an exception. The last thing is the call to done(); It is only called when we have a response and it equals 200.

**Test\_2**

This test should return an error for an invalid weather id (The id is to keep track of weather locations). First we pass in the application in supertest(app). Then the get request is performed for a non-existent weather location of 9999999. After this, we expect a 404 error. The .end handles the error and response. This time the response status should equal 404. Finally, we call done().

**Test\_3**

Here we want to mark weather as changed and test to make sure that the record reflects that change. So we have an http request and embed that request into the response. We start with supertest(app) and then we put weather/01/changed. After this we will expect a 200 response and end that request. Then we assert that this request was valid. The status should equal 200 and then we expect the body of the response. If these two assertions passed we test to make sure the record actually changed. We get weather/01 and expect a 200 response. Then we end with a status that should equal 200. The response body uses the endofForecast property to get the current time stamp and not equal undefined. Then we call done().