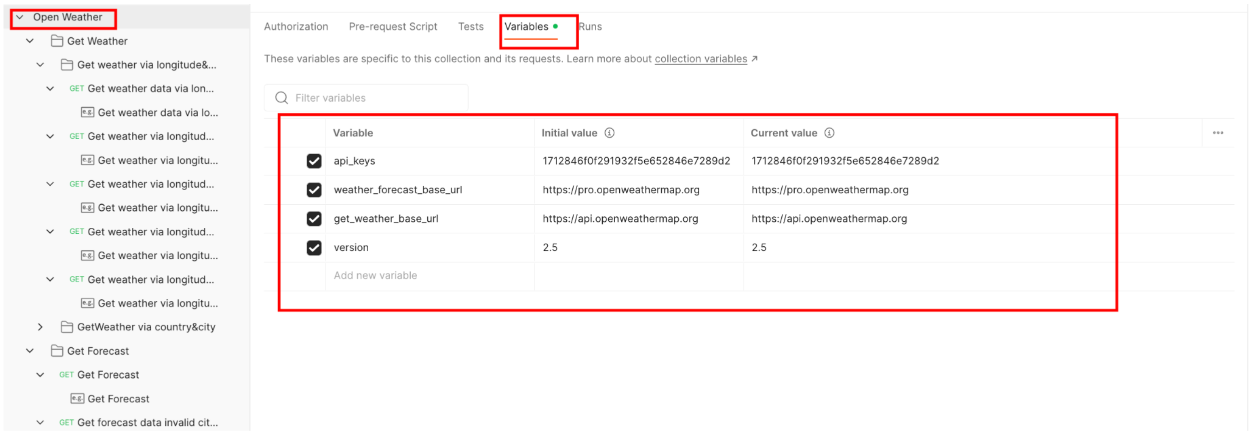
Test case documentation

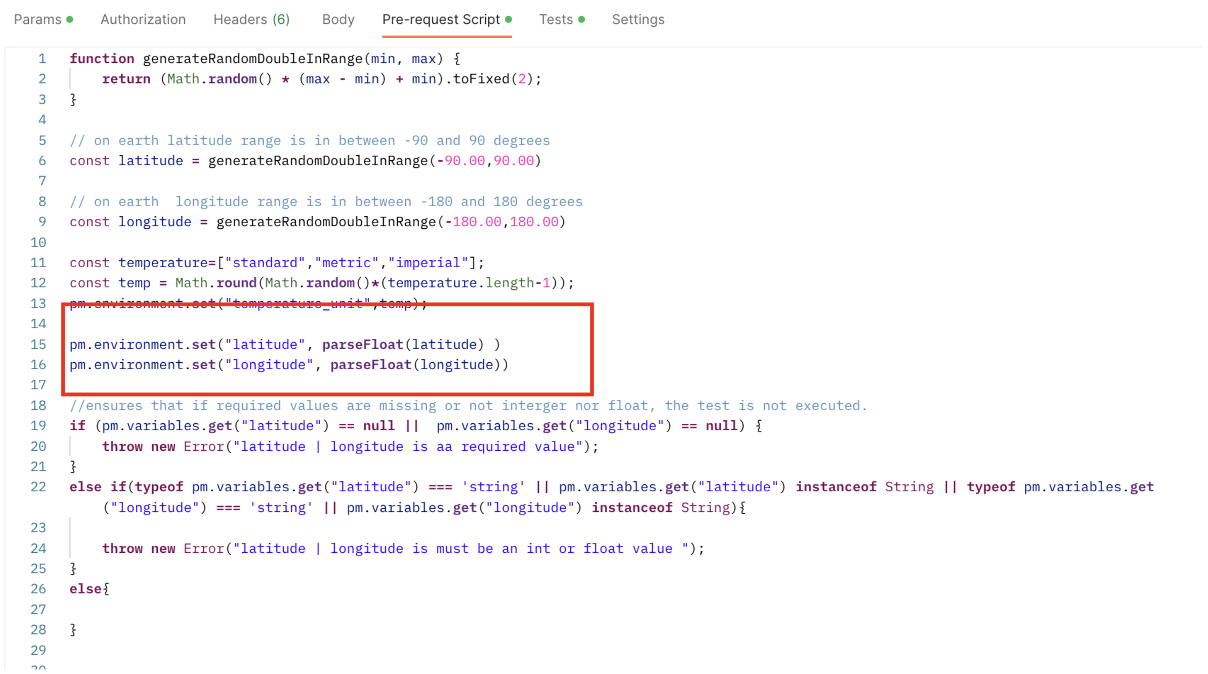
1. Each test case has environment variables:
   a. Click on the collection name > Variables. These define the base url, api version and api keys.


Open Weather
Get Weather
Get weather via longitude&...
GET Get weather data via lon...
Get weather data via lo...
GET Get weather via longitud...
Get weather via longitu...
GET Get weather via longitud...
Get weather via longitu...
GET Get weather via longitud...
Get weather via longitu...
GET Get weather via longitud...
Get weather via longitu...
GetWeather via country&city
Get Forecast
GET Get Forecast
Get Forecast
GET Get forecast data invalid cit...
Authorization
Pre-request Script
Tests
Variables
Runs
Filter variables

These variables are specific to this collection and its requests. Learn more about collection variables ↗

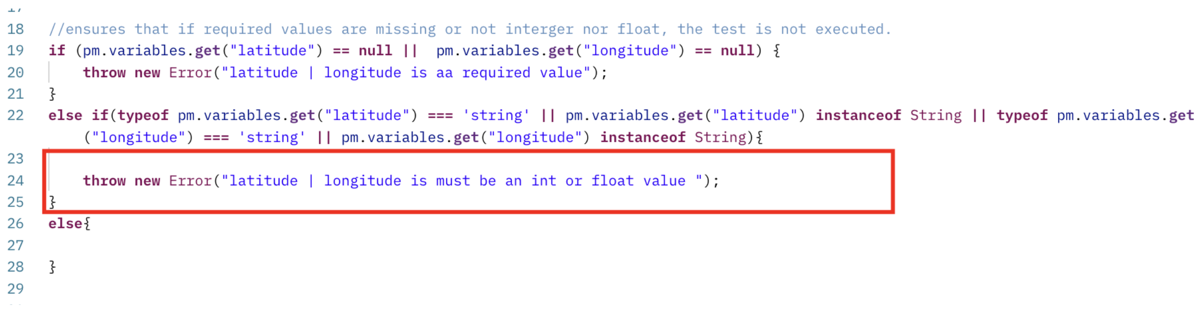
	Variable	Initial value	Current value
✓	api_keys	1712846f0f291932f5e652846e7289d2	1712846f0f291932f5e652846e7289d2
✓	weather_forecast_base_url	https://pro.openweathermap.org	https://pro.openweathermap.org
✓	get_weather_base_url	https://api.openweathermap.org	https://api.openweathermap.org
✓	version	2.5	2.5
	Add new variable		

2. Each test case has prerequisite scripts. What this does is set the variables required to run the tests. Example:
   a. This sets the latitude and longitude values generated from a random number. For each latitude and longitude value we have a minimum value.
   b. As for temperature, its picked randomly between three values: ***standard, metric and imperial***



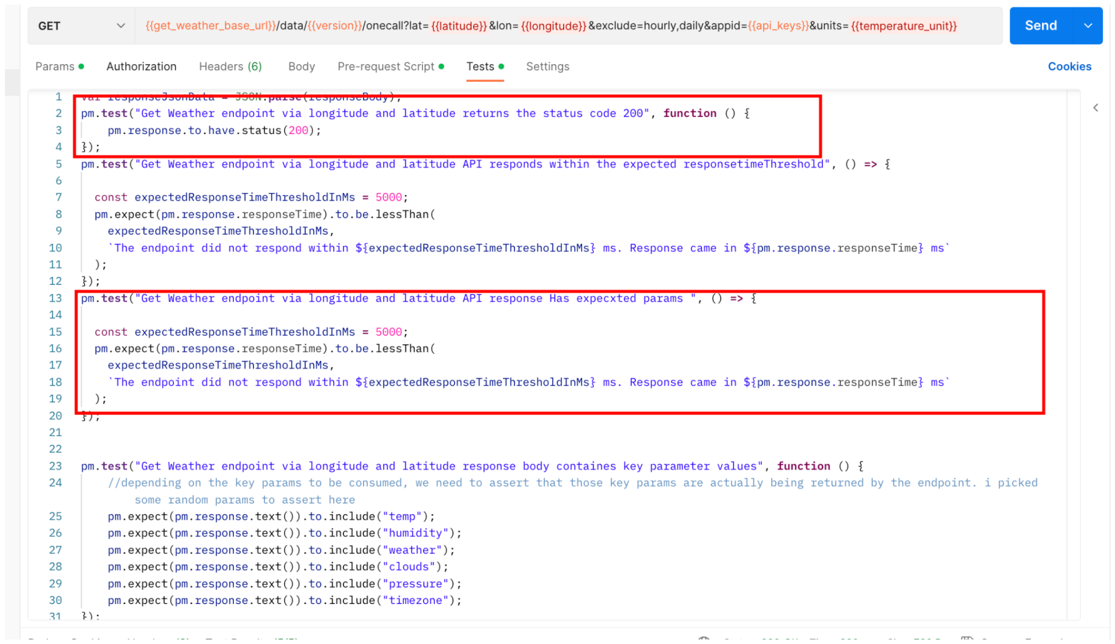
```
function generateRandomDoubleInRange(min, max) {
    return (Math.random() * (max - min) + min).toFixed(2);
}

// on earth latitude range is in between -90 and 90 degrees
const latitude = generateRandomDoubleInRange(-90.00,90.00)

// on earth  longitude range is in between -180 and 180 degrees
const longitude = generateRandomDoubleInRange(-180.00,180.00)

const temperature=["standard","metric","imperial"];
const temp = Math.round(Math.random()*(temperature.length-1));
~~pm.environment.set("temperature_unit",temp);~~

pm.environment.set("latitude", parseFloat(latitude) )
pm.environment.set("longitude", parseFloat(longitude))

//ensures that if required values are missing or not interger nor float, the test is not executed.
if (pm.variables.get("latitude") == null ||  pm.variables.get("longitude") == null) {
    throw new Error("latitude | longitude is aa required value");
}
else if(typeof pm.variables.get("latitude") === 'string' || pm.variables.get("latitude") instanceof String || typeof pm.variables.get
    ("longitude") === 'string' || pm.variables.get("longitude") instanceof String){

    throw new Error("latitude | longitude is must be an int or float value ");
}
else{

}

```

   c. As for the test: all successful test cases, we have set a precondition, if any param that is required is missing, an exception is thrown, and no tests is executed:



```
//ensures that if required values are missing or not interger nor float, the test is not executed.
if (pm.variables.get("latitude") == null ||  pm.variables.get("longitude") == null) {
    throw new Error("latitude | longitude is aa required value");
}
else if(typeof pm.variables.get("latitude") === 'string' || pm.variables.get("latitude") instanceof String || typeof pm.variables.get
    ("longitude") === 'string' || pm.variables.get("longitude") instanceof String){

    throw new Error("latitude | longitude is must be an int or float value ");
}
else{

}
```

3. All tests have the actual tests:
    a. This has a number of tests and assertions:

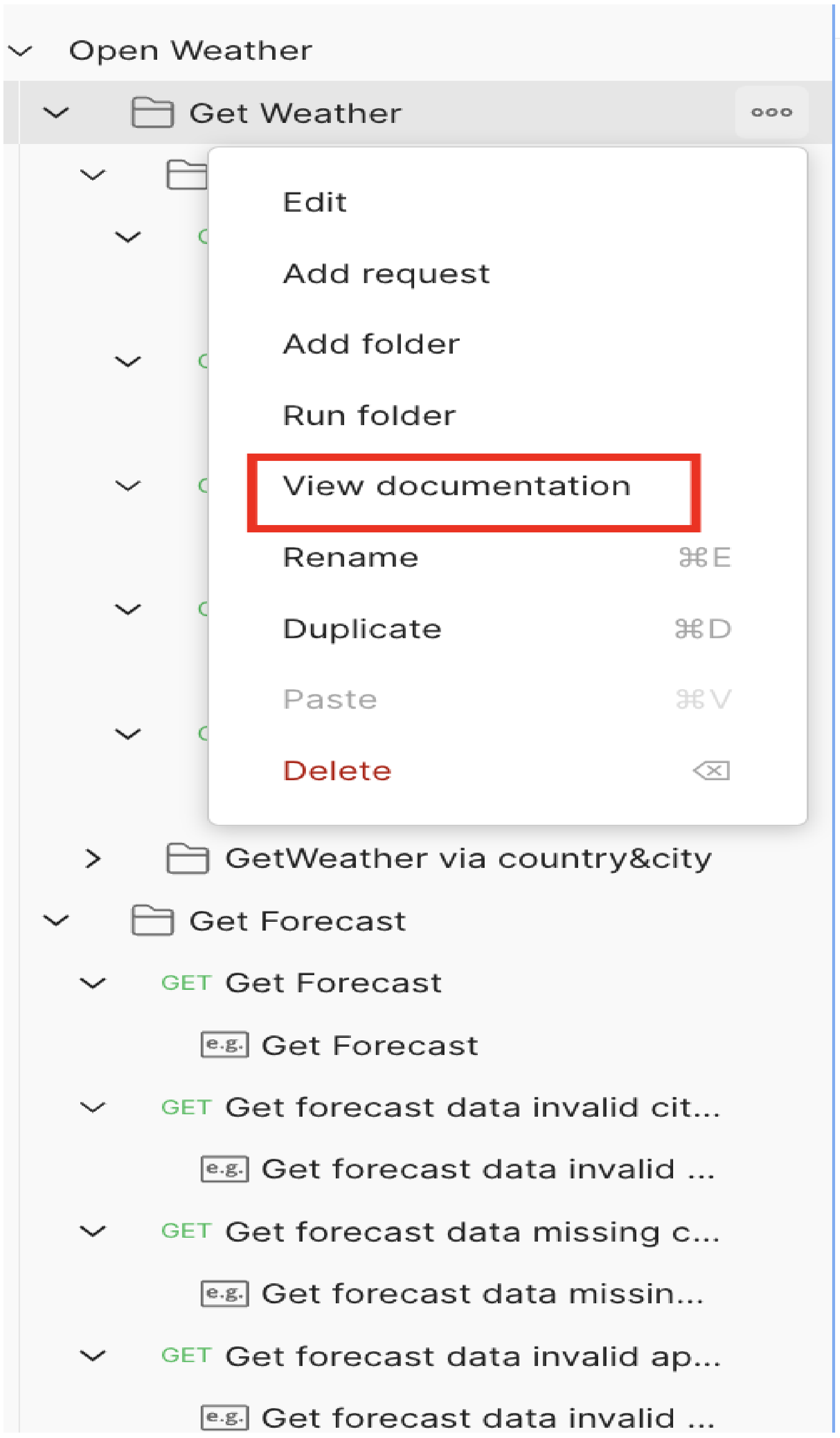


```
{{get_weather_base_url}}/data/{{version}}/onecall?lat= {{latitude}} &lon= {{longitude}} &exclude=hourly,daily&appid={{api_keys}}&units= {{temperature_unit}}
```

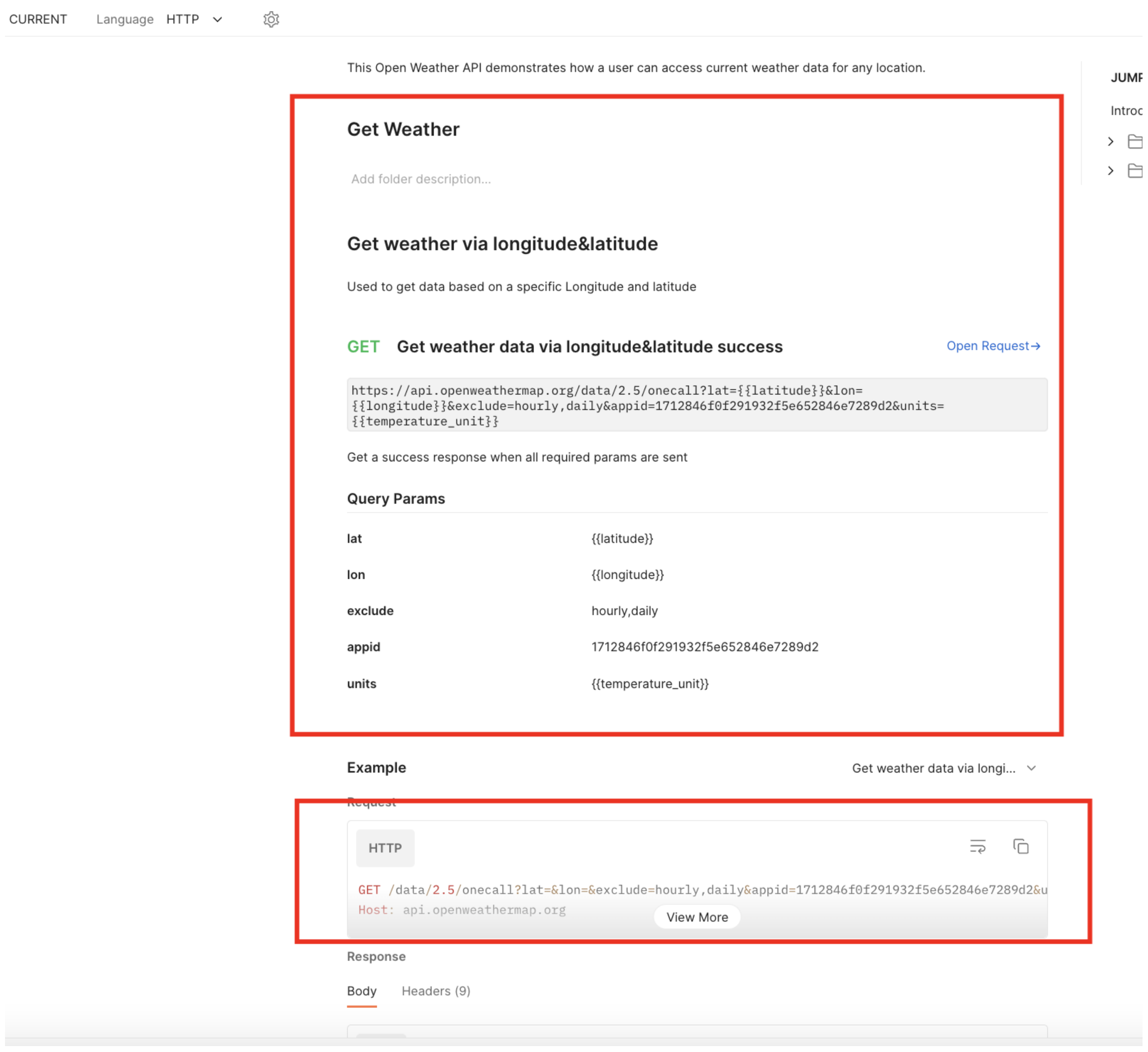
```
var responseJsonData = JSON.parse(responseBody);
pm.test("Get Weather endpoint via longitude and latitude returns the status code 200", function () {
    pm.response.to.have.status(200);
});
pm.test("Get Weather endpoint via longitude and latitude API responds within the expected responsetimeThreshold", () => {

  const expectedResponseTimeThresholdInMs = 5000;
  pm.expect(pm.response.responseTime).to.be.lessThan(
    expectedResponseTimeThresholdInMs,
    `The endpoint did not respond within ${expectedResponseTimeThresholdInMs} ms. Response came in ${pm.response.responseTime} ms`
  );
});
pm.test("Get Weather endpoint via longitude and latitude API response Has expecxted params ", () => {

  const expectedResponseTimeThresholdInMs = 5000;
  pm.expect(pm.response.responseTime).to.be.lessThan(
    expectedResponseTimeThresholdInMs,
    `The endpoint did not respond within ${expectedResponseTimeThresholdInMs} ms. Response came in ${pm.response.responseTime} ms`
  );
});


pm.test("Get Weather endpoint via longitude and latitude response body containes key parameter values", function () {
    //depending on the key params to be consumed, we need to assert that those key params are actually being returned by the endpoint. i picked
        some random params to assert here
    pm.expect(pm.response.text()).to.include("temp");
    pm.expect(pm.response.text()).to.include("humidity");
    pm.expect(pm.response.text()).to.include("weather");
    pm.expect(pm.response.text()).to.include("clouds");
    pm.expect(pm.response.text()).to.include("pressure");
    pm.expect(pm.response.text()).to.include("timezone");
});
```

Positive and negative tests have been separated to their own api endpoints since it requires a user to send different kinds of requests/responses.

4. To access the api documentation, kindly click on the collection, then click on full documentation:


Open Weather
Get Weather
Edit
Add request
Add folder
Run folder
View documentation
Rename ⌘E
Duplicate ⌘D
Paste ⌘V
Delete
GetWeather via country&city
Get Forecast
GET Get Forecast
e.g. Get Forecast
GET Get forecast data invalid cit...
e.g. Get forecast data invalid ...
GET Get forecast data missing c...
e.g. Get forecast data missin...
GET Get forecast data invalid ap...
e.g. Get forecast data invalid ...

5. You would be able to see api documentation with sample requests and responses:


CURRENT Language HTTP

This Open Weather API demonstrates how a user can access current weather data for any location.

JUMP

Introc

Get Weather

Add folder description...

Get weather via longitude&latitude

Used to get data based on a specific Longitude and latitude

GET Get weather data via longitude&latitude success

Open Request→

```
https://api.openweathermap.org/data/2.5/onecall?lat={{latitude}}&lon=
{{longitude}}&exclude=hourly,daily&appid=1712846f0f291932f5e652846e7289d2&units=
{{temperature_unit}}
```

Get a success response when all required params are sent

Query Params

lat	{{latitude}}
lon	{{longitude}}
exclude	hourly,daily
appid	1712846f0f291932f5e652846e7289d2
units	{{temperature_unit}}

Example

Get weather data via longi...

Request

HTTP

```
GET /data/2.5/onecall?lat=&lon=&exclude=hourly,daily&appid=1712846f0f291932f5e652846e7289d2&u
Host: api.openweathermap.org
```

View More

Response

Body Headers (9)

6.

NOTE 1:

One expectation has not been met:

> ***Verify that the API returns the correct temperature unit based on the parameter value passed in the request***

Reason:

- ***How I am getting the weather is via using dynamic values for latitude and longitude, hence I cant assert that the temperature received is let's say 30 degrees if I had set units as metric.***
- ***Also, on the data received, the temperature value is a double, that does not have***

unit parameter concatenated, hence the values received can’t be checked if its either of the three expected values.

NOTE 2: *Kindly review the following api automation using rest assured implementation using rest assured. This is from a different api done some years back. I have a more improved version that I need to change some things: https://github.com/Daisychepkemoi/cucumberRestApi*