**Website of fake API**

<https://fakestoreapi.com/docs#tag/Products/operation/getAllProducts>

https://fakestoreapi.com/products/1

Website for json formatter

https://jsonformatter.org/

**PRODUCTS**

Type of API:GET

Test Scripts for Get all products

// Test case for verifying the response structure and data types

pm.test("Verify response structure and data types", () => {

    pm.response.to.have.jsonSchema({

        type: "array",

        items: {

            type: "object",

            required: ["id", "title", "price", "description", "category", "image"],

            properties: {

                id: { type: "number" },

                title: { type: "string" },

                price: { type: "number" },

                description: { type: "string" },

                category: { type: "string" },

                image: { type: "string", format: "uri" }

            }

        }

    });

});

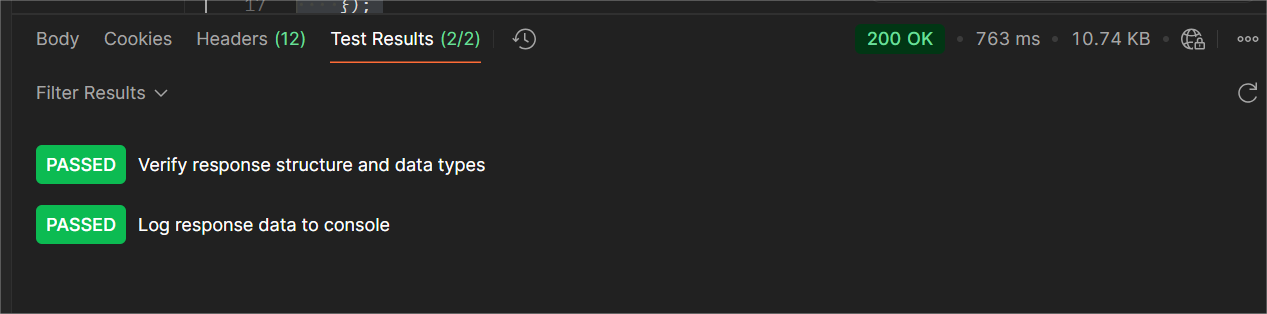
// Test case for logging the response data to the console

pm.test("Log response data to console", () => {

    console.log(pm.response.json());

});

Result of test Scripts:



Type of API: Post

Test Scripts for add a new product

pm.test("Response has the 'id' property", function () {

    pm.expect(pm.response.text()).to.not.be.empty;

    let response;

    try {

        response = pm.response.json();

    } catch (e) {

        pm.expect.fail("Response is not valid JSON");

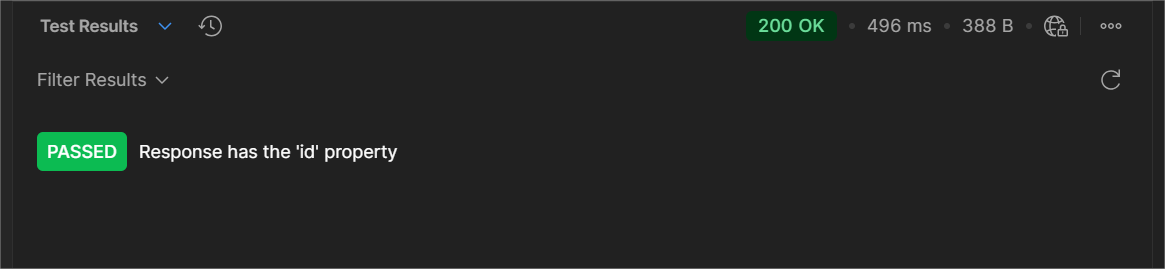
    }

    pm.expect(response).to.have.property("id");

    pm.expect(response.id).to.be.a("number");

});

Result of test Scripts:



Type of API: PUT

Test Scripts for update a product.

pm.test("Verify response after updating product", function () {

    pm.expect(pm.response.code).to.equal(200);

    const body = pm.response.json();

    // Expect only 'id' in response

    pm.expect(body).to.have.property("id");

    pm.expect(body.id).to.be.a("number");

});

Results of test Scripts:



Type of API: DEL

Test Scripts for delete a product:

pm.sendRequest({

    url: 'https://fakestoreapi.com/products/1',

    method: 'DELETE',

}, function (err, response) {

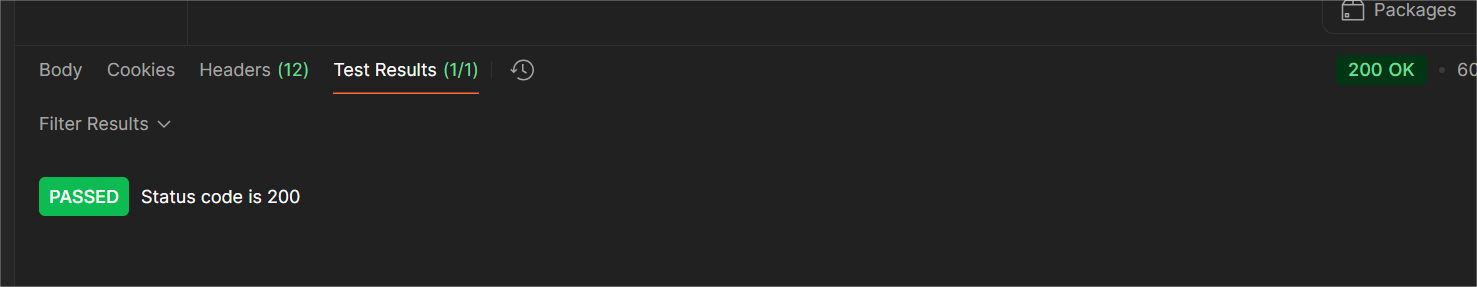
    pm.test("Status code is 200", function () {

        pm.expect(response.code).to.equal(200);

    });

});

Results of test Scripts:



**CARTS**

**Types of API: Get**

**Test scripts for Get all Carts.**

// Test to get all the carts

pm.test("Verify product details", function () {

    pm.expect(pm.response.code).to.equal(200);

    const responseBody = pm.response.json();

    responseBody.forEach(product => {

        pm.expect(product).to.have.property("id");

        pm.expect(product.id).to.be.a('number');

        pm.expect(product).to.have.property("title");

        pm.expect(product.title).to.be.a('string');

        pm.expect(product).to.have.property("price");

        pm.expect(product.price).to.be.a('number');

        pm.expect(product).to.have.property("description");

        pm.expect(product.description).to.be.a('string');

        pm.expect(product).to.have.property("category");

        pm.expect(product.category).to.be.a('string');

        pm.expect(product).to.have.property("image");

        pm.expect(product.image).to.be.a('string');

        pm.expect(product).to.have.property("rating");

        pm.expect(product.rating).to.be.an('object');

        pm.expect(product.rating).to.have.property("rate");

        pm.expect(product.rating.rate).to.be.a('number');

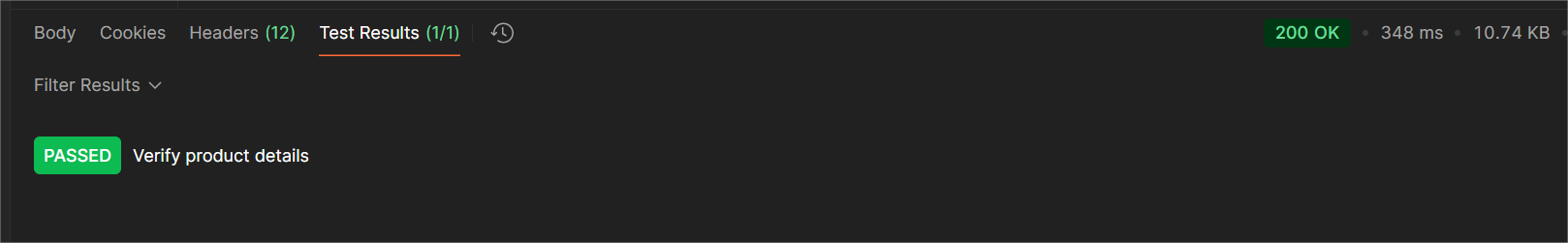
        pm.expect(product.rating).to.have.property("count");

        pm.expect(product.rating.count).to.be.a('number');

    });

});

**Result of test scripts:**

****

**Type of API:POST**

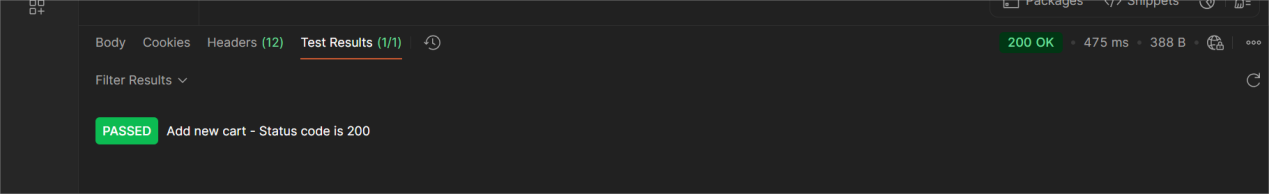
**Test scripts to add a new cart:**

pm.test("Add new cart - Status code is 200", function () {

    pm.response.to.have.status(200);

});

**Results of test Scripts:**

****

**Type of API: GET**

**Test scripts to get a single cart:**

pm.test("Status code is 200", function () {

    pm.response.to.have.status(200);

});

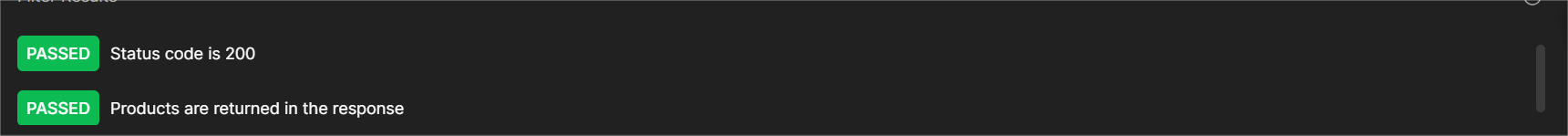
pm.test("Products are returned in the response", function () {

    const response = pm.response.json();

    pm.expect(response).to.be.an('array').that.is.not.empty;

});

**Result of Test Scripts:**

****

**Type of API: PUT**

**Test script for update cart:**

// Test to update the cart with a new product

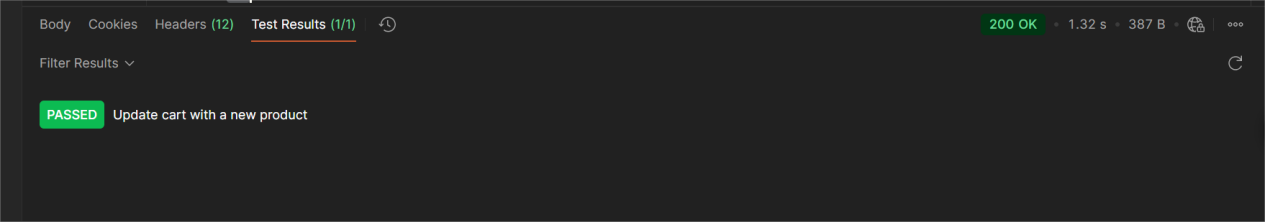
pm.test("Update cart with a new product", function () {

    const response = pm.response.json();

    pm.expect(response.id).to.eql(1);

});

**Result of Test Script:**

****

**Type of API: DELETE**

**Test script to delete a cart.**

pm.sendRequest({

    url: 'https://fakestoreapi.com/carts/1',

    method: 'DELETE',

}, function (err, response) {

    pm.test("Request completed without error", function () {

        pm.expect(err).to.be.null;

    });

    pm.test("Cart deleted successfully with 200 status", function () {

        pm.expect(response.code).to.equal(200);

        const contentType = response.headers.get('Content-Type');

        pm.expect(contentType).to.include('application/json');

        const res = response.json();

        pm.expect(res).to.have.property("id").that.is.a("number");

    });

});

**Result of test script**

****

**Users**

**Type of API: GET**

**Test scripts to get all the users**

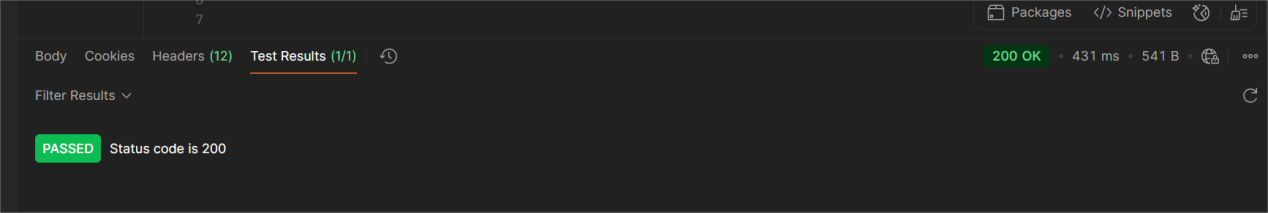
// Test to get all the users

pm.test("Status code is 200", function () {

    pm.response.to.have.status(200);

});

**Result of test script:**

****