🧪 Postman Project – API Testing with DummyJSON

🔹 Overview

In this project, I worked with the **DummyJSON API** to practice **API testing** using **Postman**.
The project simulates a simple **e-commerce backend** with authentication, product management, and shopping carts.

I structured the collection into **folders** to mirror real-world feature separation:

📂 **CRUD OPERATIONS**

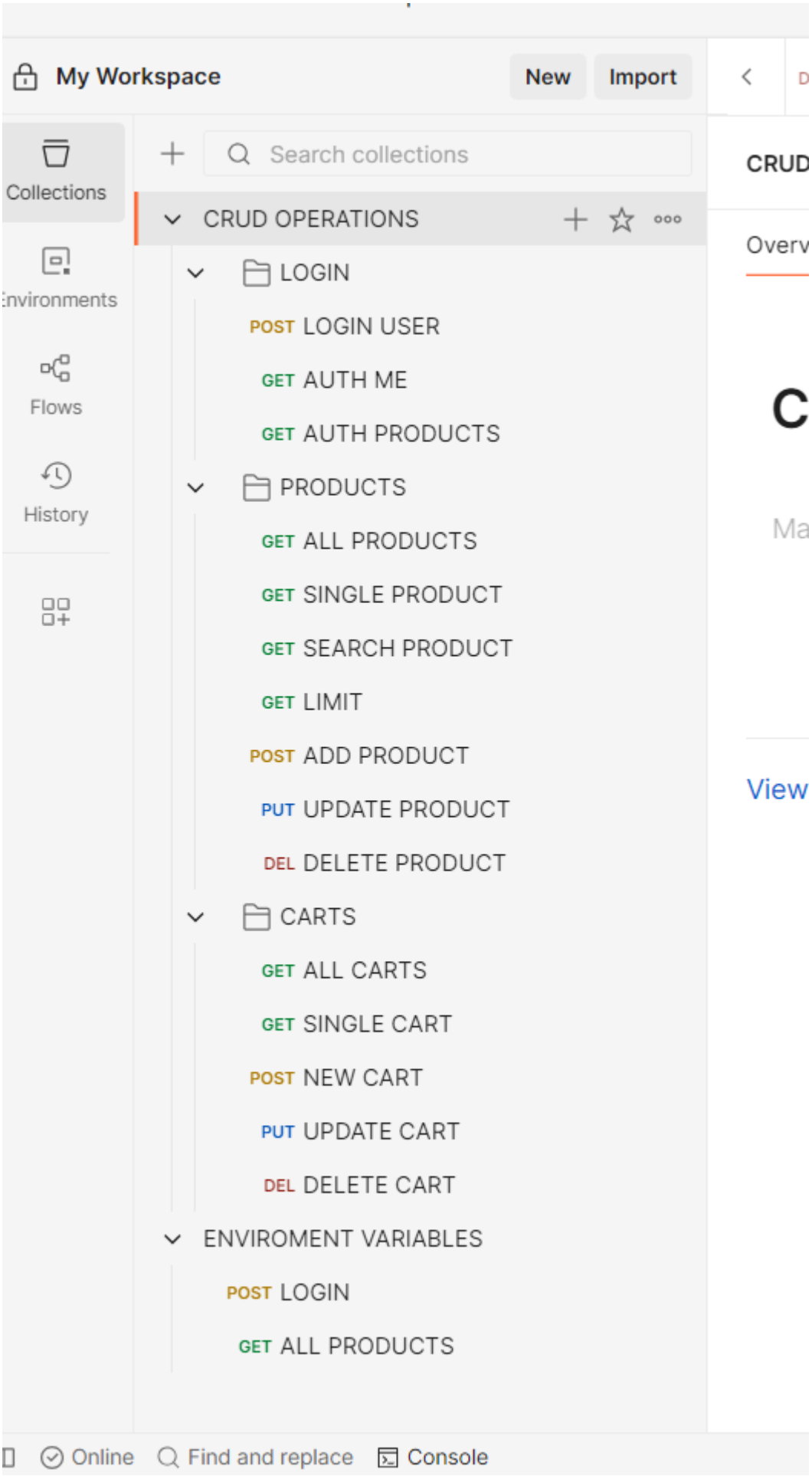
- **LOGIN**
    - POST Login User
    - GET Auth Me
    - GET Auth Products
- **PRODUCTS**
    - GET All Products
    - GET Single Product
    - GET Search Product
    - GET Limit
    - POST Add Product
    - PUT Update Product
    - DELETE Product

- **CARTS**
    - GET All Carts
    - GET Single Cart
    - POST New Cart
    - PUT Update Cart
    - DELETE Cart

📂 **ENVIRONMENT VARIABLES**

- POST Login (with variables)
- GET All Products (with validations)

This structure demonstrates how I:
✔️ Organized endpoints by feature
✔️ Applied **CRUD operations**
✔️ Implemented **environment variables**
✔️ Wrote **JavaScript test scripts** to validate API behavior


My Workspace
New
Import
Collections
Environments
Flows
History
Search collections
CRUD OPERATIONS
LOGIN
POST LOGIN USER
GET AUTH ME
GET AUTH PRODUCTS
PRODUCTS
GET ALL PRODUCTS
GET SINGLE PRODUCT
GET SEARCH PRODUCT
GET LIMIT
POST ADD PRODUCT
PUT UPDATE PRODUCT
DEL DELETE PRODUCT
CARTS
GET ALL CARTS
GET SINGLE CART
POST NEW CART
PUT UPDATE CART
DEL DELETE CART
ENVIROMENT VARIABLES
POST LOGIN
GET ALL PRODUCTS
CRUD
Overv
C
Ma
View
Online
Find and replace
Console

🔑 LOGIN

Request Body:

```
{
  "username": "emilys",
  "password": "emilyspass",
  "expiresInMins": "30"
}
```

Tests:

```
let responseJson = pm.response.json()
pm.environment.set("accessToken",
responseJson.accessToken)

pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
```

Auth Me – Tests:

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});

pm.test("Response contains user object", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("id");
    pm.expect(jsonData).to.have.property("username");
});
```

3. GET — Auth Products

Headers:

```
Authorization: Bearer {{accessToken}}
```

📦 PRODUCTS

🔍 Get Single Product (invalid ID)

```
pm.test("Status code is 404", () => {
   pm.expect(pm.response.code).to.eql(404)
})
```

➕ Add Product

Body:

```
{
  "title": "mma gloves",
  "price": "25",
  "category": "sport"
}
```

Tests:

```
pm.test("New product created successfully", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("id");
    pm.expect(jsonData.title).to.eql("mma gloves");
});
```

✏️ Update Product

Body:

```
{
  "title": "maskica 12 pro",
  "price":"10"
}
```

Tests:

```
pm.test("Product updated with new title", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.title).to.eql("maskica 12 pro");
});
```

❌ Delete Product

Tests:

```
pm.test("Product deleted successfully", function () {
    pm.response.to.have.status(200);
    var jsonData = pm.response.json();
    pm.expect(jsonData).to.be.an("object");
});
```

🛒 CARTS

🆕 New Cart

```
{
  "userId": 1,
  "products": [
    { "id": 144, "quantity": 4 },
    { "id": 98, "quantity": 1 }
  ]
}
```

✏️ Update Cart

```
{
  "merge": true,
  "products": [
    { "id": 1, "quantity": 1 }
  ]
}
```

⚙️ ENVIRONMENT VARIABLES

🔑 Login (dynamic variables)

```
{
  "username":"{{username}}",
  "password":"{{password}}",
  "expiresInMins":30
}
```

Tests:

```
let responseJson = pm.response.json()
```

```
pm.environment.set("accesstoken",
responseJson.accessToken)

pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});

pm.test("Response contains token", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("token");
    pm.expect(jsonData.token).to.not.be.empty;
});

pm.test("Response contains user object", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("id");
    pm.expect(jsonData).to.have.property("username");
});
```

📦 Validate All Products

```
let responseJson = pm.response.json()

pm.test("Status code is 200", () => {
    pm.expect(pm.response.code).to.eql(200)
})

pm.test("Verify all values are as expected", () => {
    pm.expect(responseJson.total).to.eql(194)
    pm.expect(responseJson.skip).to.eql(0)
    pm.expect(responseJson.limit).to.eql(30)
})

pm.test("Verify that products variable is an Array", ()
=> {
```

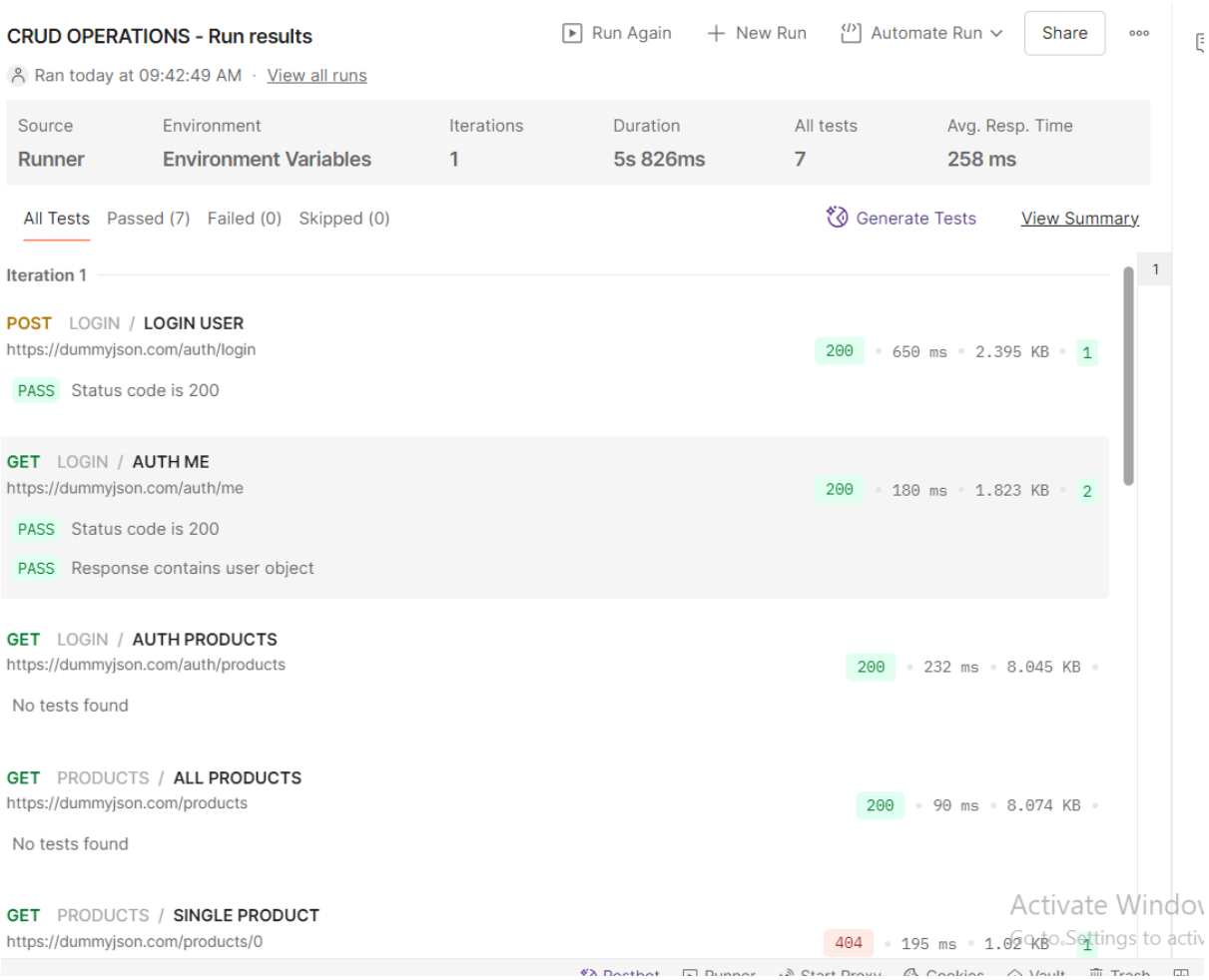
```
    pm.expect(responseJson.products).to.be.an('array')
})

pm.test("Verify the response array number", () => {
    pm.expect(responseJson.products.length).to.eql(30)
}
```

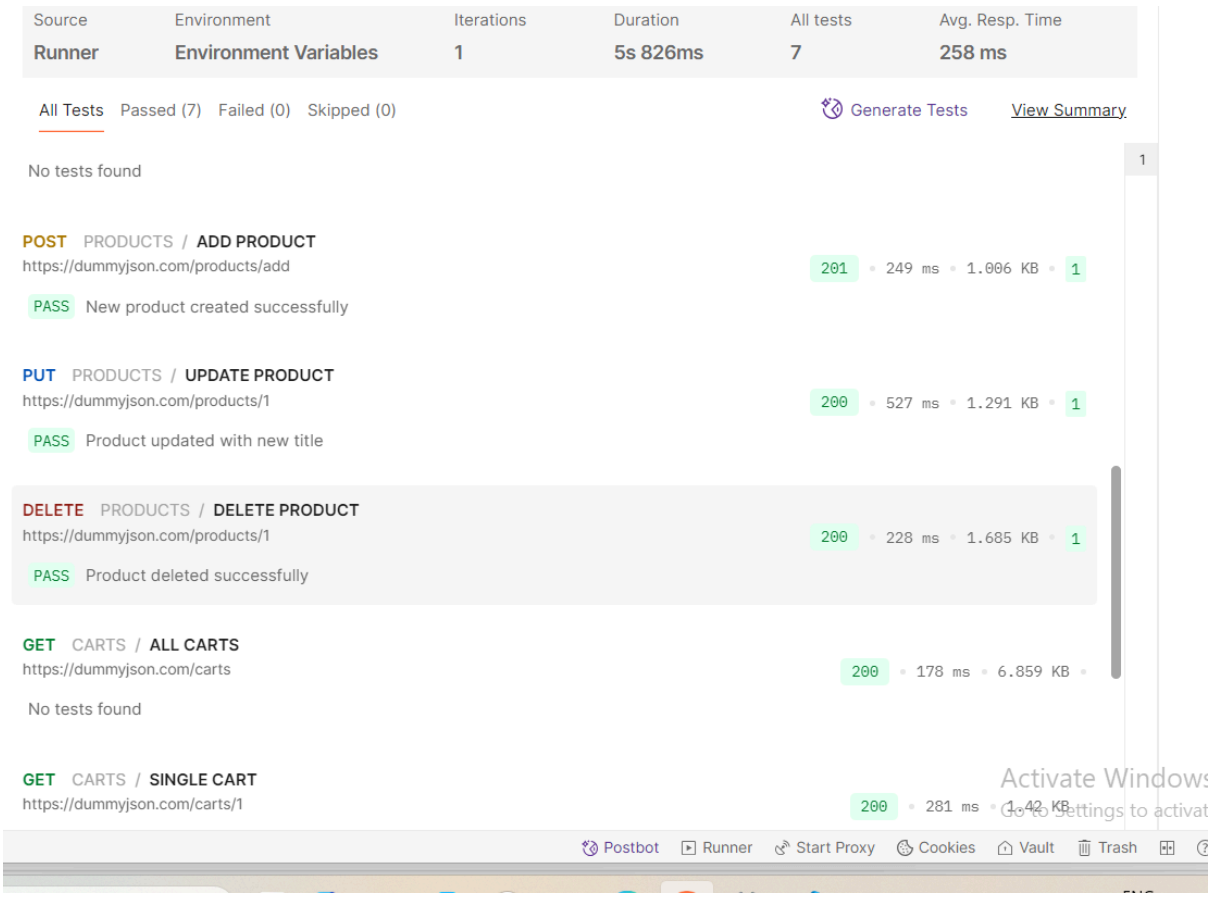
📝 Test Run Overview

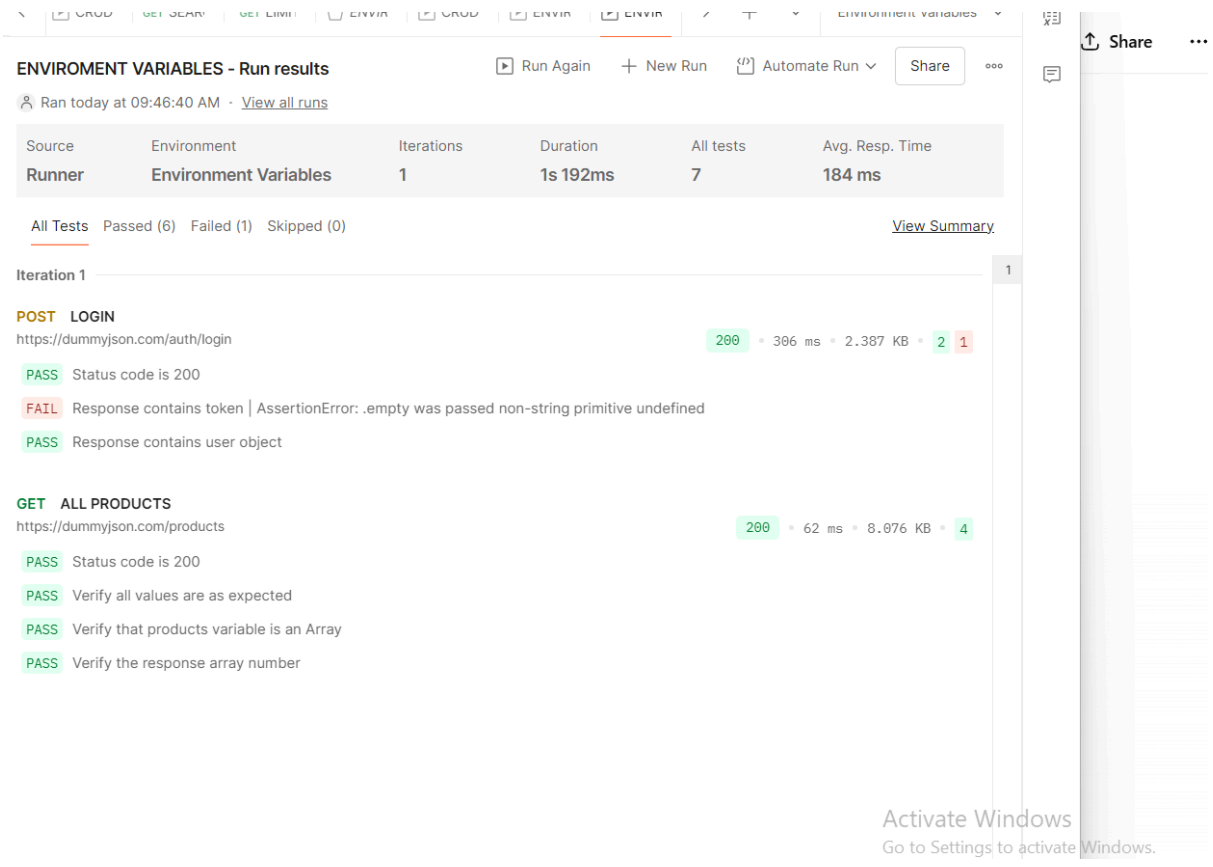
After creating and organizing all requests into folders (Login, Products, Carts), I executed the full collection in **Postman Collection Runner**.
The runner allows me to validate:

- if all requests are working correctly,
- whether the **test scripts** I wrote are passing or failing,
- and how long each request takes to respond.

This gives a clear overview of the **overall health of the API**, with a summary of passed/failed assertions.
Below are screenshots from the test run, showing the execution results for different endpoints.


CRUD OPERATIONS - Run results
Run Again
New Run
Automate Run
Share
Ran today at 09:42:49 AM · View all runs
Source
Runner
Environment
Environment Variables
Iterations
1
Duration
5s 826ms
All tests
7
Avg. Resp. Time
258 ms
All Tests
Passed (7)
Failed (0)
Skipped (0)
Generate Tests
View Summary
Iteration 1
POST LOGIN / LOGIN USER
https://dummyjson.com/auth/login
200 • 650 ms • 2.395 KB • 1
PASS Status code is 200
GET LOGIN / AUTH ME
https://dummyjson.com/auth/me
200 • 180 ms • 1.823 KB • 2
PASS Status code is 200
PASS Response contains user object
GET LOGIN / AUTH PRODUCTS
https://dummyjson.com/auth/products
200 • 232 ms • 8.045 KB •
No tests found
GET PRODUCTS / ALL PRODUCTS
https://dummyjson.com/products
200 • 90 ms • 8.074 KB •
No tests found
GET PRODUCTS / SINGLE PRODUCT
https://dummyjson.com/products/0
404 • 195 ms • 1.02 KB • 1
Activate Windo
Go to Settings to activ


Source
Runner
Environment
Environment Variables
Iterations
1
Duration
5s 826ms
All tests
7
Avg. Resp. Time
258 ms
All Tests
Passed (7)
Failed (0)
Skipped (0)
Generate Tests
View Summary
No tests found
POST PRODUCTS / ADD PRODUCT
https://dummyjson.com/products/add
201 • 249 ms • 1.006 KB • 1
PASS New product created successfully
PUT PRODUCTS / UPDATE PRODUCT
https://dummyjson.com/products/1
200 • 527 ms • 1.291 KB • 1
PASS Product updated with new title
DELETE PRODUCTS / DELETE PRODUCT
https://dummyjson.com/products/1
200 • 228 ms • 1.685 KB • 1
PASS Product deleted successfully
GET CARTS / ALL CARTS
https://dummyjson.com/carts
200 • 178 ms • 6.859 KB
No tests found
GET CARTS / SINGLE CART
https://dummyjson.com/carts/1
200 • 281 ms • 1.42 KB
Activate Windows
Go to Settings to activat
Postbot
Runner
Start Proxy
Cookies
Vault
Trash


ENVIROMENT VARIABLES - Run results
Run Again
New Run
Automate Run
Share
Ran today at 09:46:40 AM · View all runs
Source
Runner
Environment
Environment Variables
Iterations
1
Duration
1s 192ms
All tests
7
Avg. Resp. Time
184 ms
All Tests
Passed (6)
Failed (1)
Skipped (0)
View Summary
Iteration 1
POST LOGIN
https://dummyjson.com/auth/login
200 • 306 ms • 2.387 KB • 2 1
PASS Status code is 200
FAIL Response contains token | AssertionError: .empty was passed non-string primitive undefined
PASS Response contains user object
GET ALL PRODUCTS
https://dummyjson.com/products
200 • 62 ms • 8.076 KB • 4
PASS Status code is 200
PASS Verify all values are as expected
PASS Verify that products variable is an Array
PASS Verify the response array number
Share
Activate Windows
Go to Settings to activate Windows.