REST API Testing with DummyJSON

Assignment Documentation

API Overview

Base URL: https://dummyjson.com

API Selected: DummyJSON – A fake REST API for testing and prototyping

Authentication Method: Token-based authentication.

Resources Used: Posts, Auth

Why I Chose DummyJSON

I selected DummyJSON as my REST API for this assignment because it's a comprehensive fake REST API that's perfect for testing and learning. It provides realistic data structures and supports all standard HTTP methods without requiring backend setup.

Something I learned is that DummyJSON is a simulation API. All operations (POST, PUT, PATCH, DELETE) return realistic responses, but data is not actually saved to any database.

Available Users

DummyJSON provides multiple test users for authentication. You can view all available users at:

Users Endpoint: https://dummyjson.com/users

User I Selected: emmaj (User ID: 5)

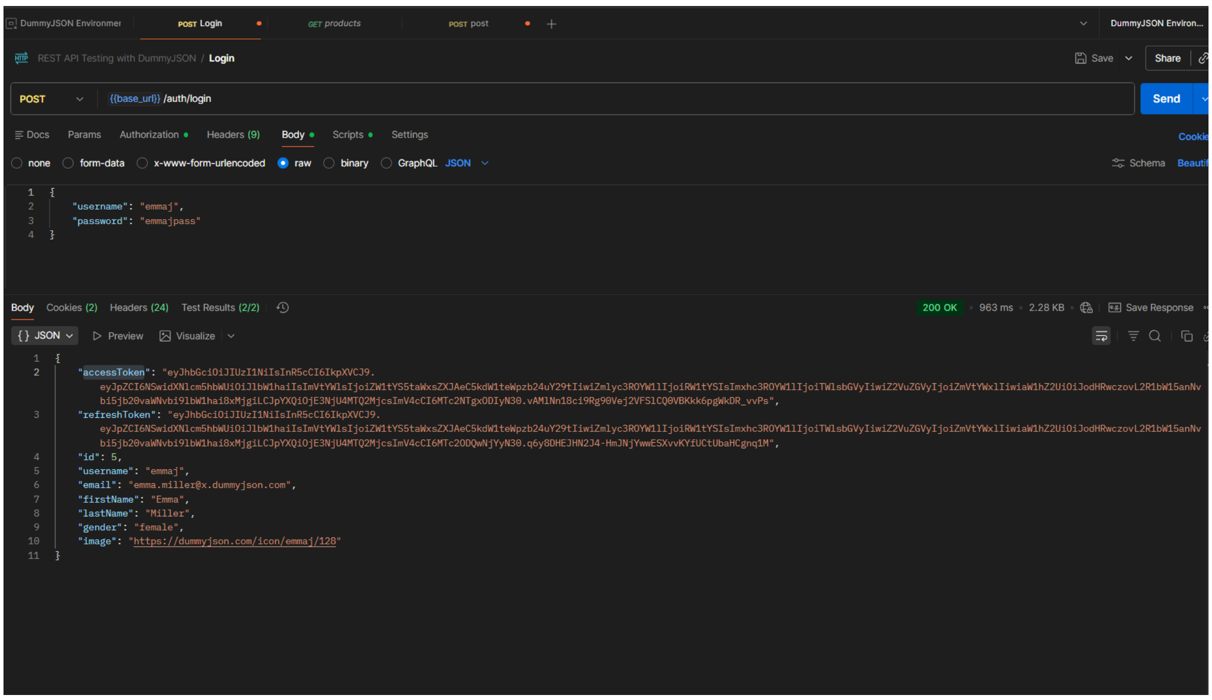
Credentials:

- **Username**: emmaj
- **Password:** emma1pass

Authentication

Login Endpoint

- **URL: {{base_url}}/auth/login**
- **Method:** POST



POST {{base_url}} /auth/login

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

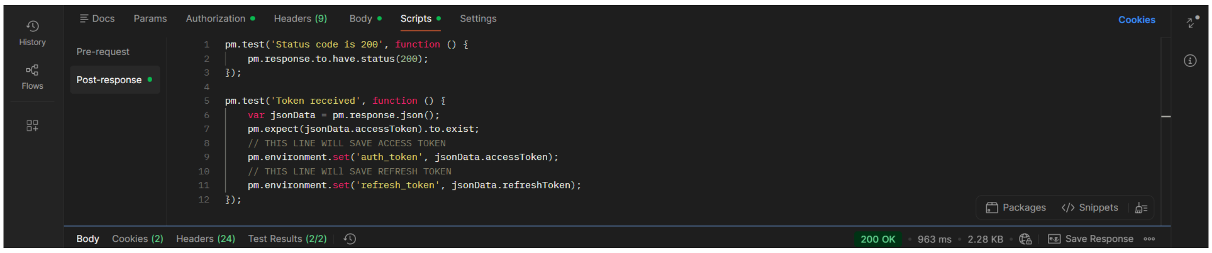
```json
{
    "username": "emmaj",
    "password": "emmajpass"
}
```

200 OK • 963 ms • 2.28 KB

```json
{
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJpZCI6NSwidXNlcm5hbWUiOiJlbW1haiIsImVtYWlsIjoiZW1tYS5taWxsZXJAeC5kdW1teWpzb24uY29tIiwiZmlyc3ROYW1lIjoiRW1tYSIsImxhc3ROYW1lIjoiTWlsbGVyIiwiZ2VuZGVyIjoiZmVtYWxlIiwiaW1hZ2UiOiJodHRwczovL2R1bW15anNv
        bi5jb20vaWNvbi9lbW1hai8xMjgiLCJpYXQiOjE3NjU4MTQ2MjcsImV4cCI6MTc2NTgxODIyN30.vAMlNn18ci9Rg90Vej2VFSlCQ0VBKkk6pgWkDR_vvPs",
    "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJpZCI6NSwidXNlcm5hbWUiOiJlbW1haiIsImVtYWlsIjoiZW1tYS5taWxsZXJAeC5kdW1teWpzb24uY29tIiwiZmlyc3ROYW1lIjoiRW1tYSIsImxhc3ROYW1lIjoiTWlsbGVyIiwiZ2VuZGVyIjoiZmVtYWxlIiwiaW1hZ2UiOiJodHRwczovL2R1bW15anNv
        bi5jb20vaWNvbi9lbW1hai8xMjgiLCJpYXQiOjE3NjU4MTQ2MjcsImV4cCI6MTc2ODQwNjYyN30.q6y8DHEJHN2J4-HmJNjYwwESXvvKYfUCtUbaHCgnq1M",
    "id": 5,
    "username": "emmaj",
    "email": "emma.miller@x.dummyjson.com",
    "firstName": "Emma",
    "lastName": "Miller",
    "gender": "female",
    "image": "https://dummyjson.com/icon/emmaj/128"
}
```

Automated Token Management

To make my workflow more efficient, I added a post-response script that automatically extracts the access token from the login response and saves it to my environment variables. This way, I don't have to manually copy-paste the token for every subsequent request!



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

pm.test('Token received', function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.accessToken).to.exist;
    // THIS LINE WILL SAVE ACCESS TOKEN
    pm.environment.set('auth_token', jsonData.accessToken);
    // THIS LINE WILl SAVE REFRESH TOKEN
    pm.environment.set('refresh_token', jsonData.refreshToken);
});
```

200 OK • 963 ms • 2.28 KB

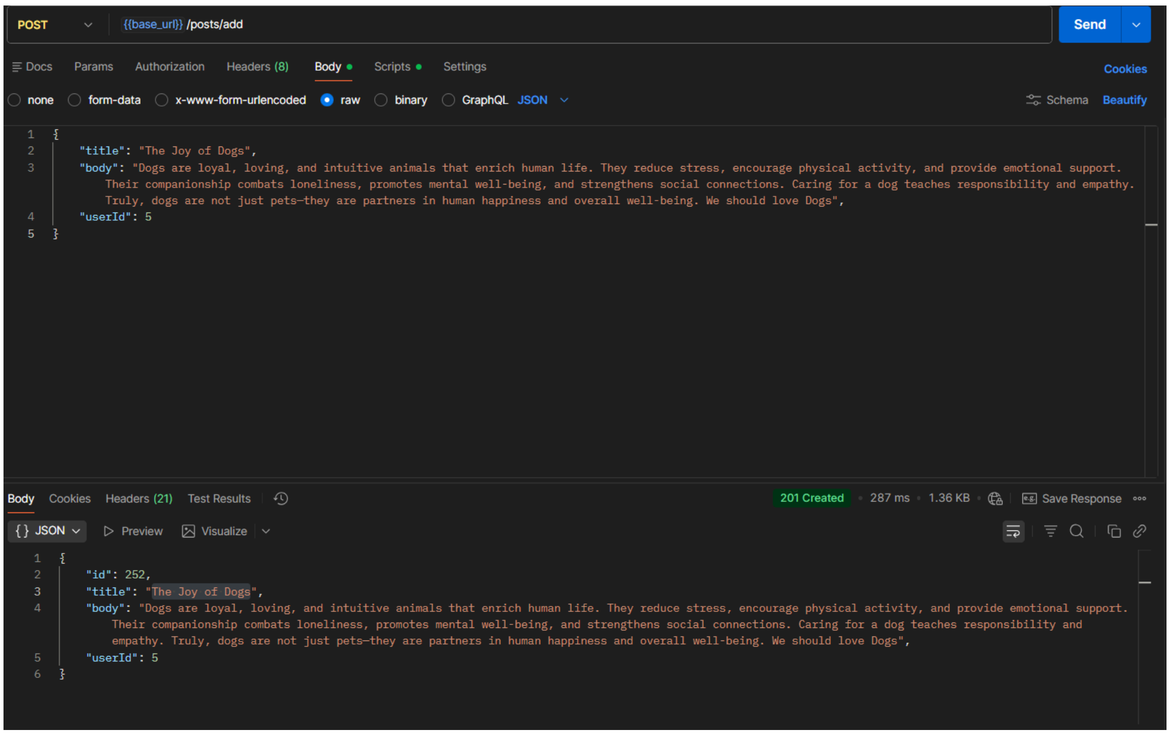
This script runs automatically after every login request,

CRUD Operations

CREATE - Add New Post

Endpoint: {{base_url}}/posts/add
Method: POST

I created a new post about dogs with userId 5 (matching my authenticated user emmaj). The API returned ID 252. This post isn't actually saved to the server, it's just simulated!



```
{{base_url}} /posts/add
```

```
{
    "title": "The Joy of Dogs",
    "body": "Dogs are loyal, loving, and intuitive animals that enrich human life. They reduce stress, encourage physical activity, and provide emotional support.
      Their companionship combats loneliness, promotes mental well-being, and strengthens social connections. Caring for a dog teaches responsibility and empathy.
      Truly, dogs are not just pets—they are partners in human happiness and overall well-being. We should love Dogs",
    "userId": 5
}
```

```
201 Created • 287 ms • 1.36 KB
```

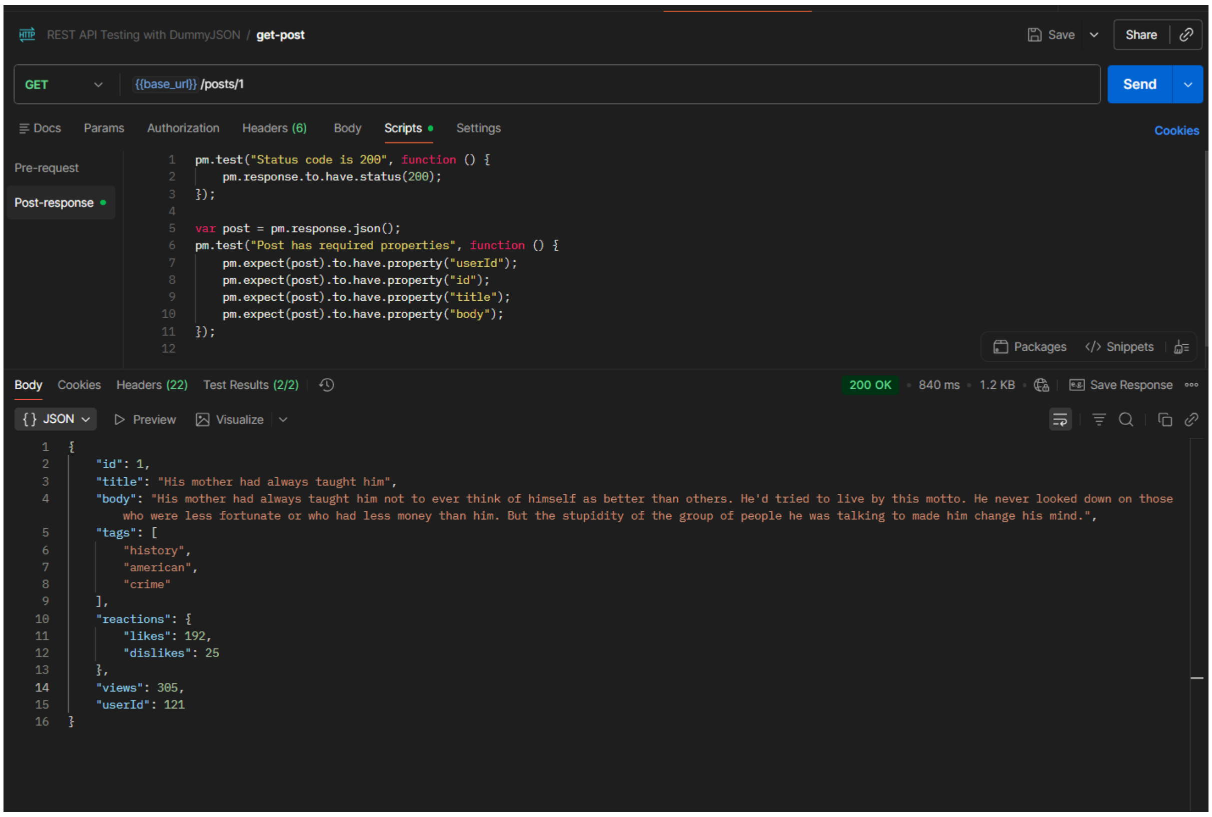
```
{
    "id": 252,
    "title": "The Joy of Dogs",
    "body": "Dogs are loyal, loving, and intuitive animals that enrich human life. They reduce stress, encourage physical activity, and provide emotional support.
      Their companionship combats loneliness, promotes mental well-being, and strengthens social connections. Caring for a dog teaches responsibility and
      empathy. Truly, dogs are not just pets—they are partners in human happiness and overall well-being. We should love Dogs",
    "userId": 5
}
```

Note: The API assigns ID 252 to this new post, but if I try to fetch or update post 252 later, it won't exist because DummyJSON doesn't actually store the data. Because its a simulation.

READ – Get Post

Endpoint: {{base_url}}/posts/1
Method: GET

Since my created post (ID 252) doesn't actually exist on the server, I fetched an existing post with ID 1 to demonstrate the READ operation. This post already exists in DummyJSON's dataset.


REST API Testing with DummyJSON / get-post
Save
Share
GET
{{base_url}} /posts/1
Send
Docs Params Authorization Headers (6) Body Scripts Settings
Cookies
Pre-request
Post-response
Packages Snippets
Body Cookies Headers (22) Test Results (2/2)
200 OK 840 ms 1.2 KB
Save Response
JSON Preview Visualize

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

var post = pm.response.json();
pm.test("Post has required properties", function () {
    pm.expect(post).to.have.property("userId");
    pm.expect(post).to.have.property("id");
    pm.expect(post).to.have.property("title");
    pm.expect(post).to.have.property("body");
});
```

```
{
    "id": 1,
    "title": "His mother had always taught him",
    "body": "His mother had always taught him not to ever think of himself as better than others. He'd tried to live by this motto. He never looked down on those
        who were less fortunate or who had less money than him. But the stupidity of the group of people he was talking to made him change his mind.",
    "tags": [
        "history",
        "american",
        "crime"
    ],
    "reactions": {
        "likes": 192,
        "dislikes": 25
    },
    "views": 305,
    "userId": 121
}
```

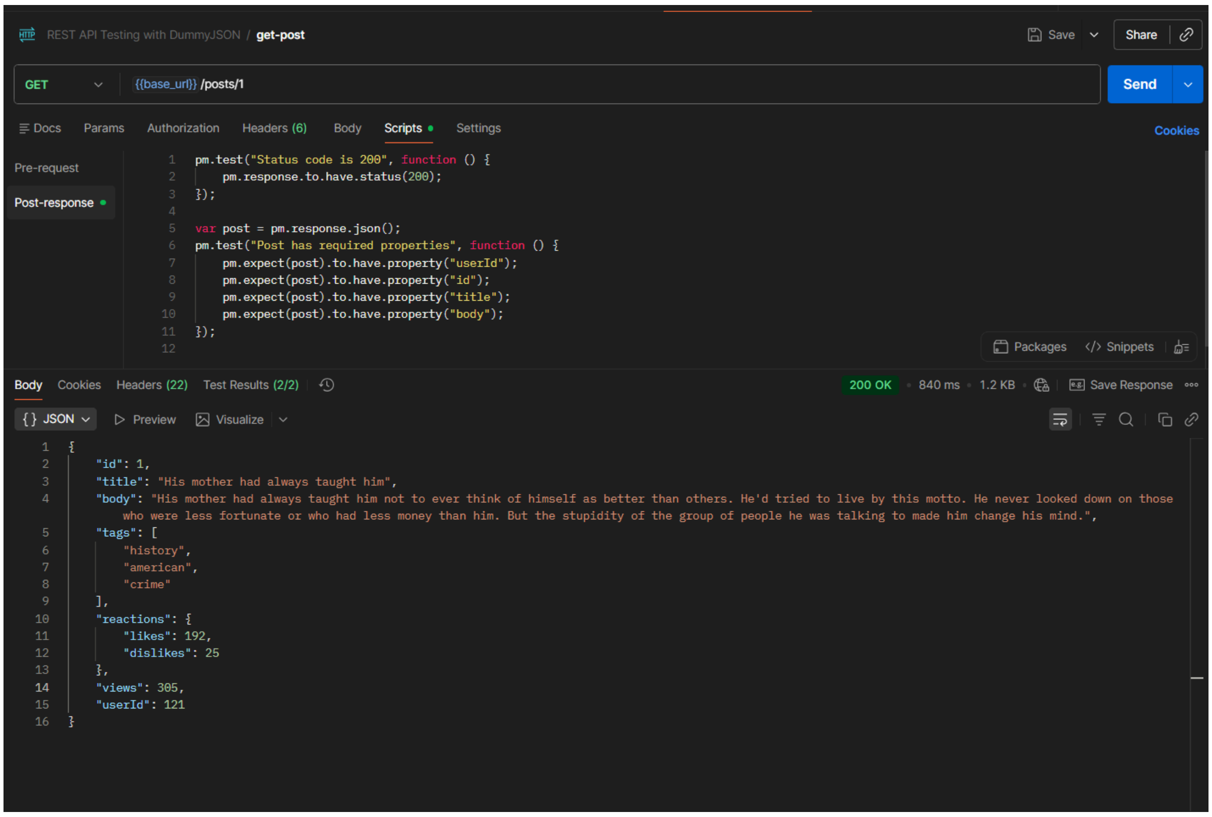
Test Scripts:

I added test scripts to verify that the response has the correct status code and contains all required properties.

UPDATE – Modify Post

Endpoint: {{base_url}}/posts/1
Method: PATCH

For the UPDATE operation, I used post ID 1 (an existing post) instead of the newly created post ID 252. Why? Because post 252 was only simulated and doesn't actually exist on the server. The API would return an error if I tried to update it.



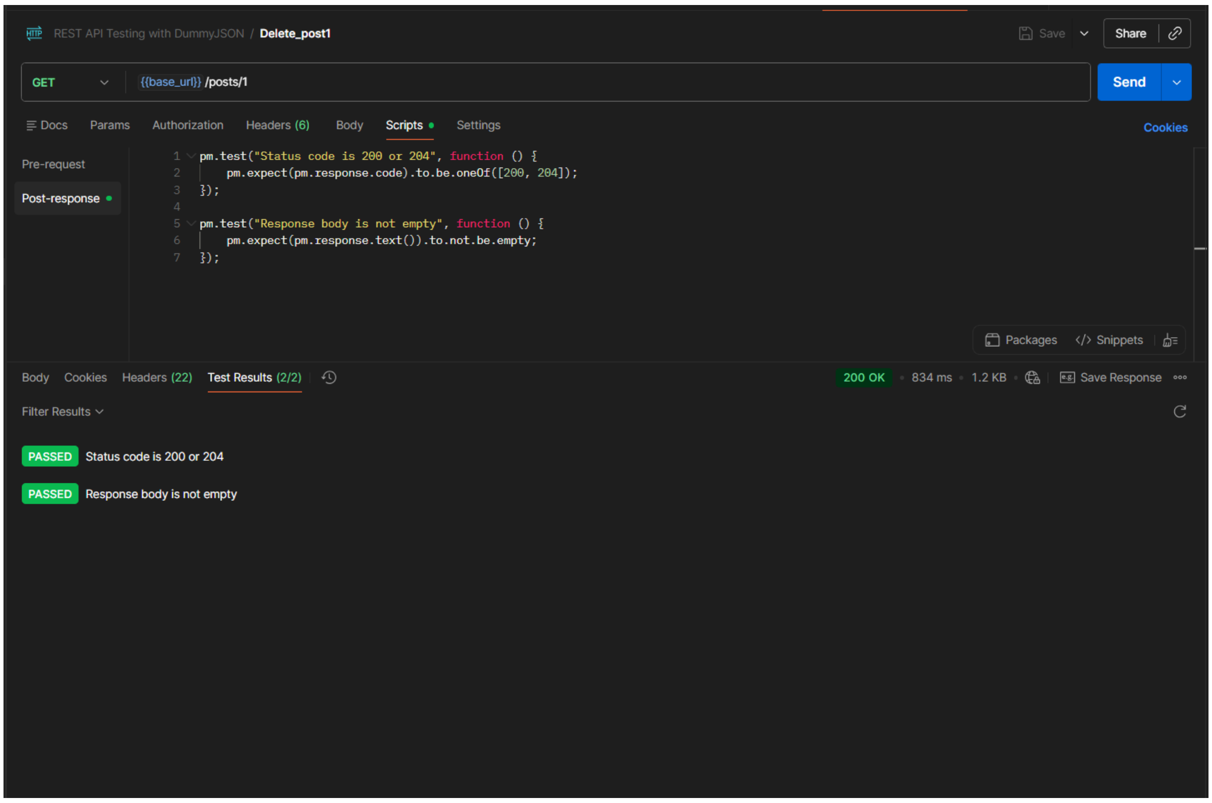
```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});

var post = pm.response.json();
pm.test("Post has required properties", function () {
    pm.expect(post).to.have.property("userId");
    pm.expect(post).to.have.property("id");
    pm.expect(post).to.have.property("title");
    pm.expect(post).to.have.property("body");
});
```

```
{
    "id": 1,
    "title": "His mother had always taught him",
    "body": "His mother had always taught him not to ever think of himself as better than others. He'd tried to live by this motto. He never looked down on those
        who were less fortunate or who had less money than him. But the stupidity of the group of people he was talking to made him change his mind.",
    "tags": [
        "history",
        "american",
        "crime"
    ],
    "reactions": {
        "likes": 192,
        "dislikes": 25
    },
    "views": 305,
    "userId": 121
}
```

DELETE – Remove Post

Endpoint: {{base_url}}/posts/1
Method: GET

For the DELETE operation, I again used post ID 1 since it's an existing post. DummyJSON uses the GET method for delete simulation, which returns the "deleted" post data as confirmation.



```
pm.test("Status code is 200 or 204", function () {
    pm.expect(pm.response.code).to.be.oneOf([200, 204]);
});

pm.test("Response body is not empty", function () {
    pm.expect(pm.response.text()).to.not.be.empty;
});
```

```
PASSED Status code is 200 or 204
PASSED Response body is not empty
```

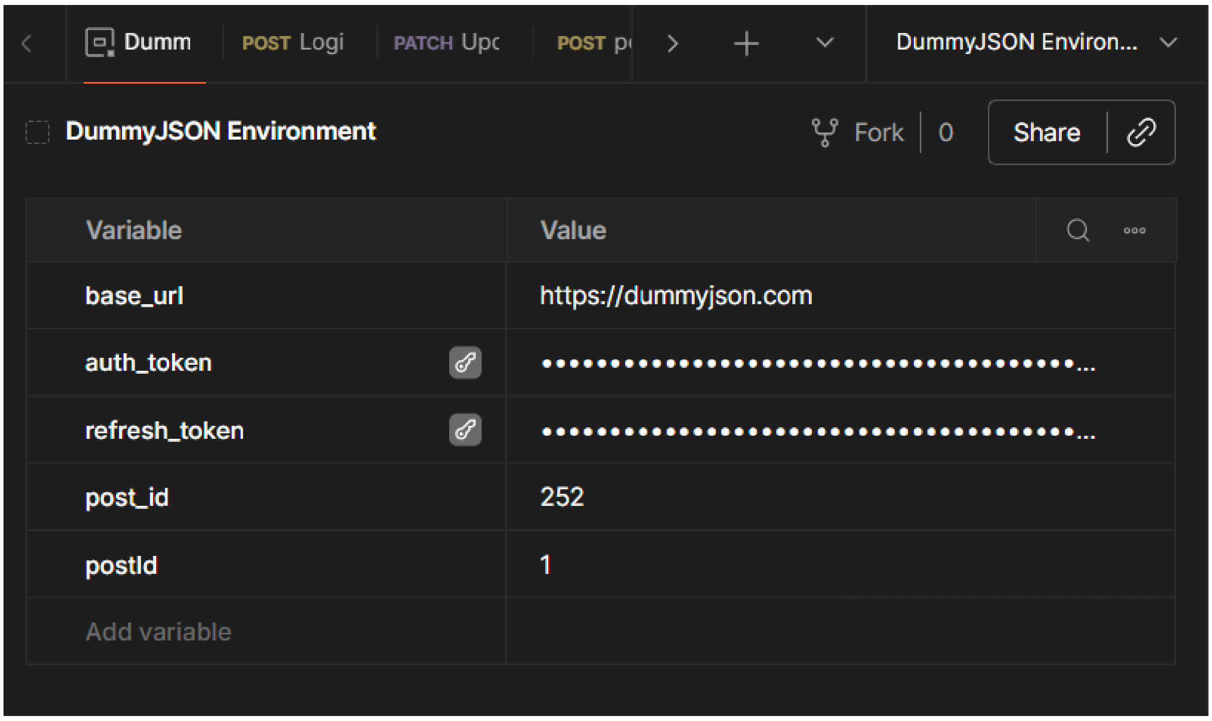
Even though the delete is simulated, the API responds as if the operation was successful.

Environment Variables

Why Use Environment Variables?

I created a DummyJSON environment where I store all reusable values. This made everything easier

For example, if the API base URL changes, I only need to update it once in my environment, and all my requests will automatically use the new URL.



Variable	Value
base_url	https://dummyjson.com
auth_token	••••••••••••••••••••••••••••••••••••••...
refresh_token	••••••••••••••••••••••••••••••••••••••...
post_id	252
postId	1
Add variable	

DummyJSON Environment Configuration I Use Variables in My Requests

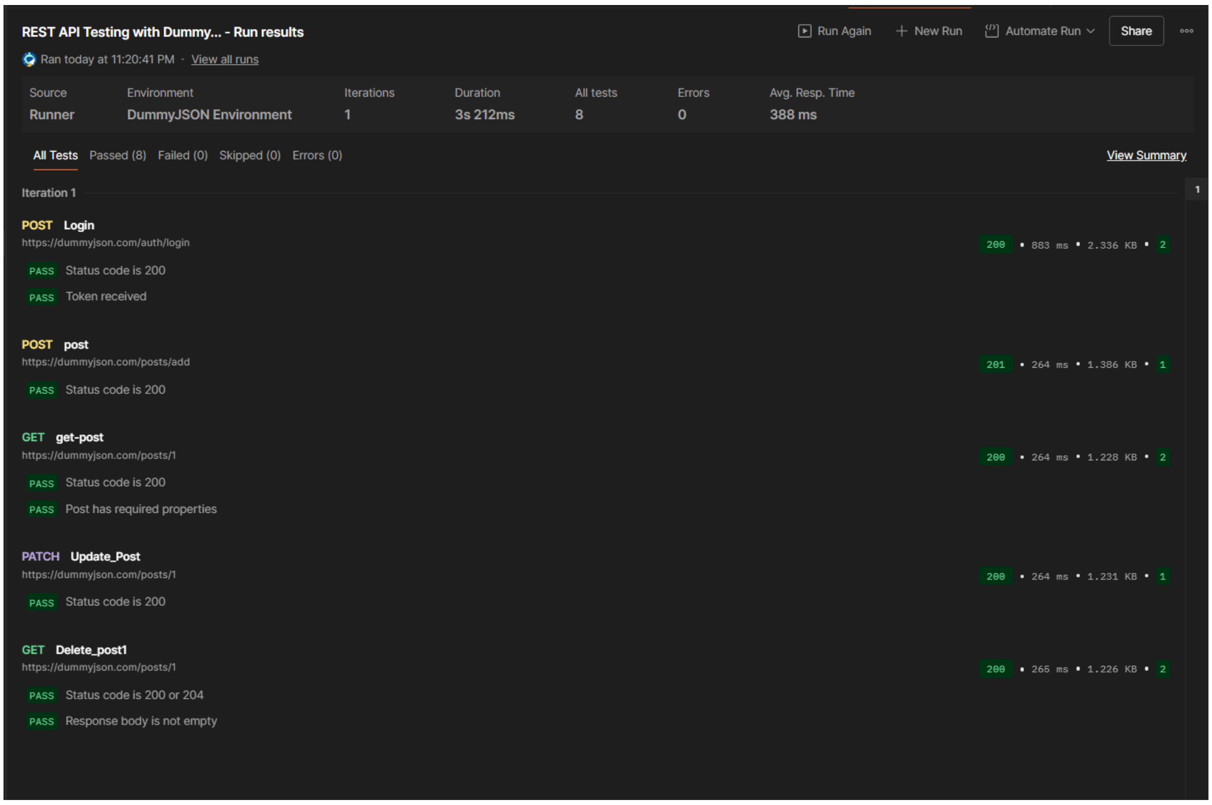
Instead of typing the full URL every time, I used the double curly braces syntax {{variable_name}} to reference my environment variables

Example: {{base_url}}/posts/1

Postman automatically replaces this with: https://dummyjson.com/posts/1

Automation & Testing

Running the Collection Runner



REST API Testing with Dummy... - Run results

Ran today at 11:20:41 PM · View all runs

Source	Environment	Iterations	Duration	All tests	Errors	Avg. Resp. Time
Runner	DummyJSON Environment	1	3s 212ms	8	0	388 ms

All Tests Passed (8) Failed (0) Skipped (0) Errors (0) View Summary

Iteration 1

POST Login
https://dummyjson.com/auth/login
`200 • 883 ms • 2.336 KB • 2`
PASS Status code is 200
PASS Token received

POST post
https://dummyjson.com/posts/add
`201 • 264 ms • 1.386 KB • 1`
PASS Status code is 200

GET get-post
https://dummyjson.com/posts/1
`200 • 264 ms • 1.228 KB • 2`
PASS Status code is 200
PASS Post has required properties

PATCH Update_Post
https://dummyjson.com/posts/1
`200 • 264 ms • 1.231 KB • 1`
PASS Status code is 200

GET Delete_post1
https://dummyjson.com/posts/1
`200 • 265 ms • 1.226 KB • 2`
PASS Status code is 200 or 204
PASS Response body is not empty

The results show that all my requests completed successfully with no errors. The authentication worked, tokens were saved automatically, and all CRUD operations responded as expected!