API Smoke Test Report

Project: Dummy JSON API Testing

Prepared by: Nada Abdelrahim

Date: 21-October-2025

Contents

1. Objective

The purpose of this smoke test is to validate that the core API endpoints for user login, product retrieval, and cart management function correctly. The test verifies that responses are successful, performance is within acceptable limits, and that the system handles dynamic JWT authentication properly.

2. Tools & Environment

Tool Used	**JMeter**
Environment	https://dummyjson.com
Authentication	Bearer Token (JWT extracted dynamically)

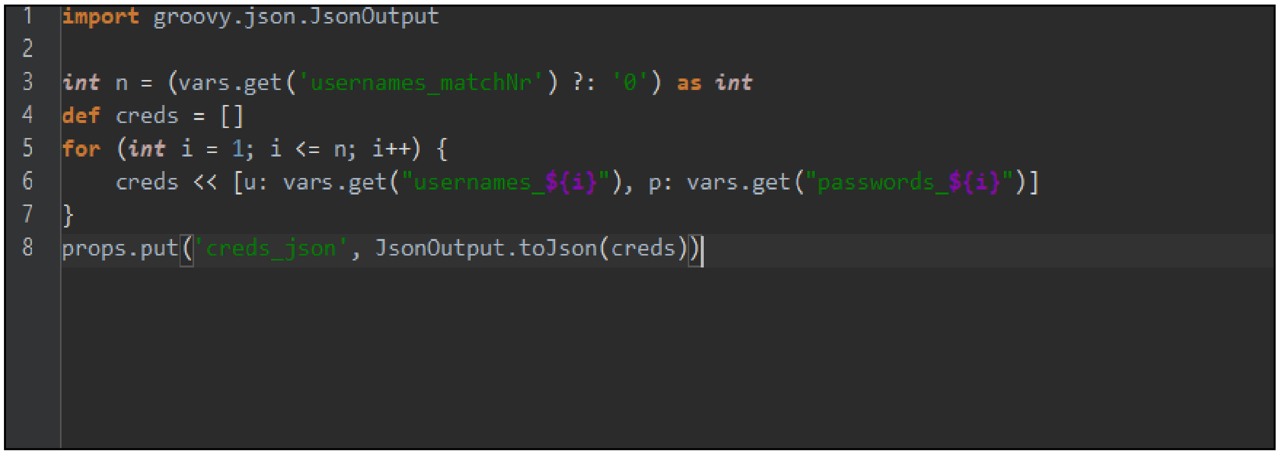
3. Data Preparations

In this step, a **JMeter JSON Extractor** was used to dynamically extract login credentials (usernames and passwords) from the API response. Two extractors were configured:

- **JSON Extractor 1**: Extracts all usernames from the /users endpoint response.
- **JSON Extractor 2**: Extracts all corresponding passwords.

After extraction, a **JSR223 Script (Groovy)** was added to store these extracted values into two separate lists:

Code Snippet – Groovy Script for Data Preparation



```
import groovy.json.JsonOutput

int n = (vars.get('usernames_matchNr') ?: '0') as int
def creds = []
for (int i = 1; i <= n; i++) {
    creds << [u: vars.get("usernames_${i}"), p: vars.get("passwords_${i}")]
}
props.put('creds_json', JsonOutput.toJson(creds))
```

4. Test Flow / Chain

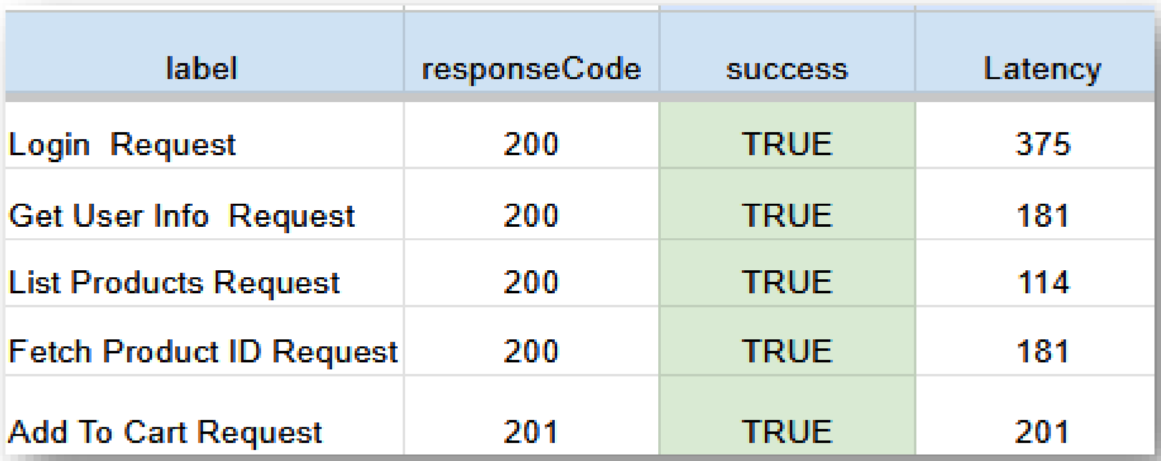
Step	API Endpoint	Method	Purpose
1	/auth/login	POST	Authenticate user and extract JWT token
2	/users/{userId}	Get	Get User details
3	/products	GET	Retrieve product list and Json Extractor used to extract random product id
4	/products/{id}	GET	Fetch details for selected product
5	/carts/add	POST	Add product to cart using token

6. Key Assertions

Assertion	Request	Expected Result	Status
Status Code	Login	200	Passed
Duration		< 800 ms	Passed
Json Assertion		User id exist	Passed
Status Code	Get User Info	200	Passed
Duration		< 800 ms	Passed
Json Assertion		User id exist	Passed
Status Code	List Products Request	200	Passed
Duration		< 800 ms	Passed
Json Assertion		Products Exist	Passed
Status Code	Fetch Product ID Request	200	Passed
Duration		< 800 ms	Passed
Json Assertion		Product Id exist	Passed
Status Code	Add To Cart Request	201	Passed
Duration		< 800 ms	Passed
Json Assertion		Total exists	Passed

7. Assertion Screenshot

Screenshot 1: Request Assertions and Latency Validation



label	responseCode	success	Latency
Login Request	200	TRUE	375
Get User Info Request	200	TRUE	181
List Products Request	200	TRUE	114
Fetch Product ID Request	200	TRUE	181
Add To Cart Request	201	TRUE	201

Description:

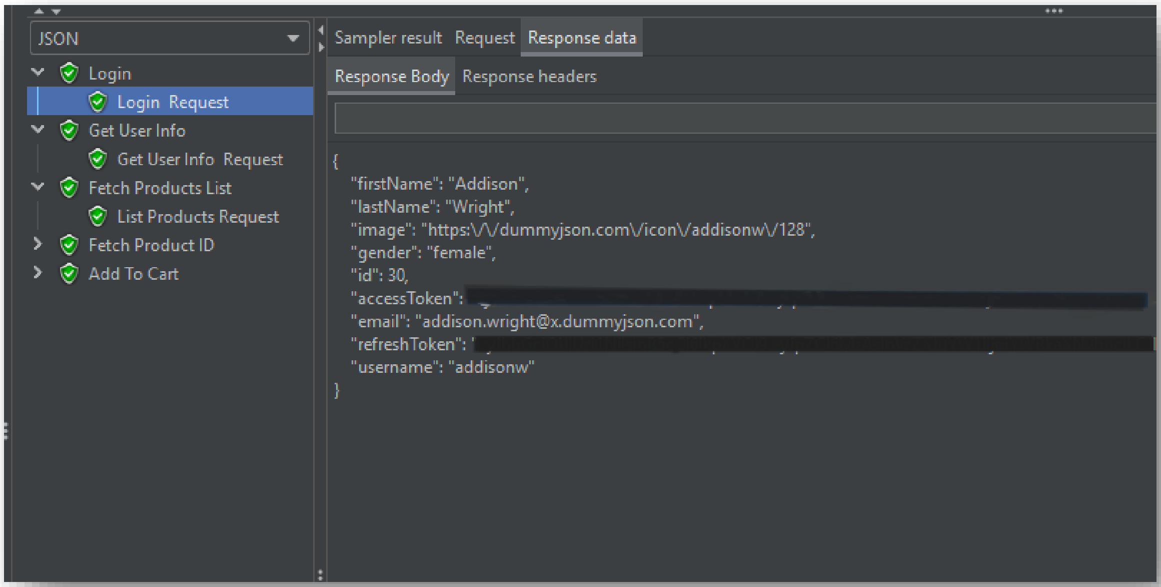
This screenshot shows the successful execution of all API endpoints during the load test.

Each request (Login, Get User Info, List Products, Fetch Product ID, and Add to Cart) returned valid **HTTP 200/201 response codes**, with all assertions passing (success = TRUE).

Observation:

- All endpoints met the Response code assertions of status **200/201**.
- Response times were **well below the 800 ms latency**, ranging from **114 ms to 375 ms**.
- All JSON fields exist (products, id, token, etc.)

Screenshot 2: Response Validation – Login API



```
{
   "firstName": "Addison",
   "lastName": "Wright",
   "image": "https:\/\/dummyjson.com\/icon\/addisonw\/128",
   "gender": "female",
   "id": 30,
   "accessToken": [redacted]
   "email": "addison.wright@x.dummyjson.com",
   "refreshToken": [redacted]
   "username": "addisonw"
}
```

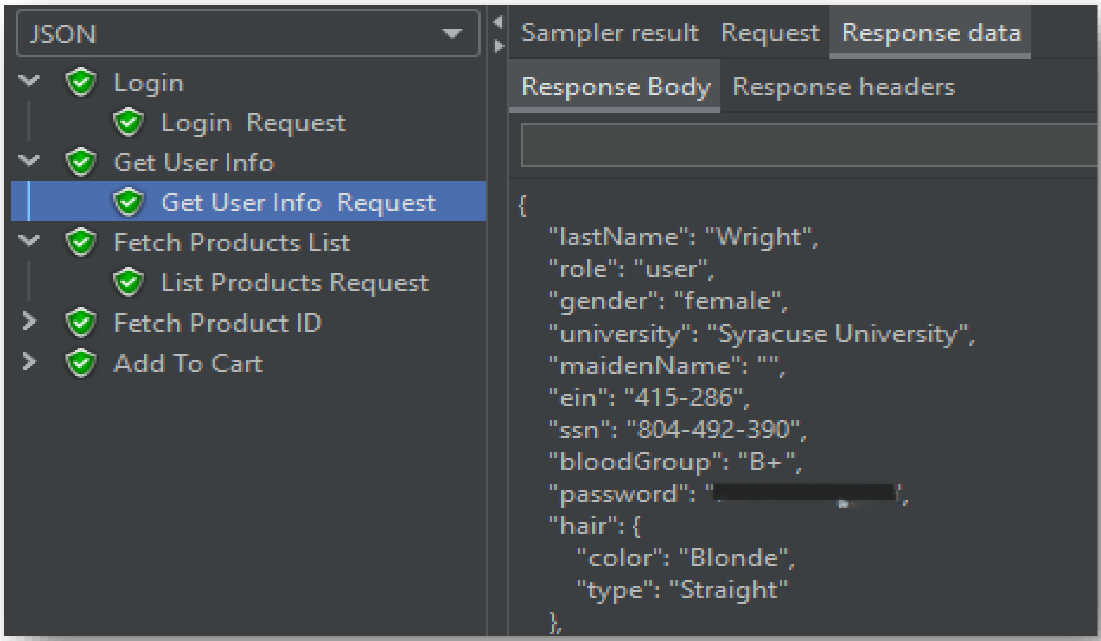
Description:

This image displays the **Response Data** tab in JMeter for the **Login** request. The JSON response includes dynamic user-name and password details (username,Password), confirming that the payload structure is valid and correctly returned by the API.

Observation:

- The response structure matches the expected schema.
- The user is authorized and the details of that user are returned.
- Assertions on JSON field existence (id) passed successfully.

Screenshot 3: Response Validation – Get User Info



```
{
  "lastName": "Wright",
  "role": "user",
  "gender": "female",
  "university": "Syracuse University",
  "maidenName": "",
  "ein": "415-286",
  "ssn": "804-492-390",
  "bloodGroup": "B+",
  "password": "[redacted]",
  "hair": {
    "color": "Blonde",
    "type": "Straight"
  },
```

Description:

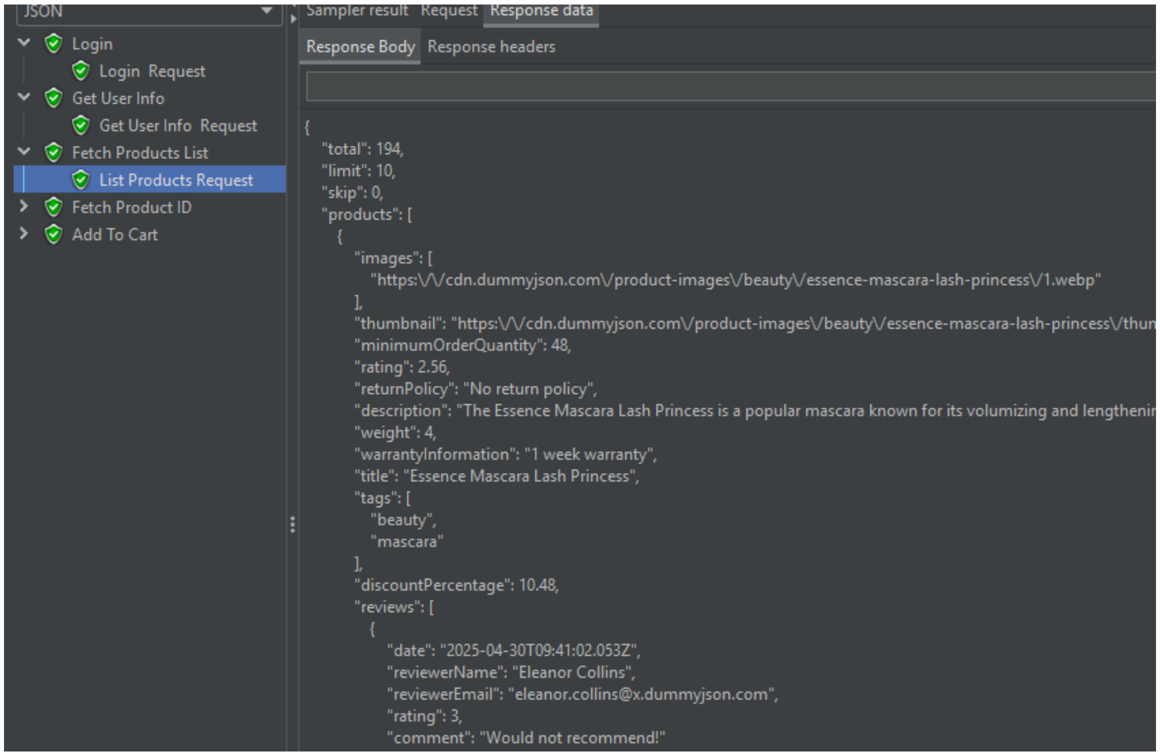
This image displays the **Response Data** tab in JMeter for the **Get User Info** request.

The JSON response includes user details (LastName, Role, Gender, University, MaidenName, etc.), confirming that the payload structure is valid and correctly returned by the API.

Observation:

- The response structure matches the expected schema.
- The information of the user was returned successfully.
- Assertions on JSON field existence (id) passed successfully.

Screenshot 4: Response Validation – Get Products List API



```
{
  "total": 194,
  "limit": 10,
  "skip": 0,
  "products": [
    {
      "images": [
        "https:\/\/cdn.dummyjson.com\/product-images\/beauty\/essence-mascara-lash-princess\/1.webp"
      ],
      "thumbnail": "https:\/\/cdn.dummyjson.com\/product-images\/beauty\/essence-mascara-lash-princess\/thun
      "minimumOrderQuantity": 48,
      "rating": 2.56,
      "returnPolicy": "No return policy",
      "description": "The Essence Mascara Lash Princess is a popular mascara known for its volumizing and lengthenir
      "weight": 4,
      "warrantyInformation": "1 week warranty",
      "title": "Essence Mascara Lash Princess",
      "tags": [
        "beauty",
        "mascara"
      ],
      "discountPercentage": 10.48,
      "reviews": [
        {
          "date": "2025-04-30T09:41:02.053Z",
          "reviewerName": "Eleanor Collins",
          "reviewerEmail": "eleanor.collins@x.dummyjson.com",
          "rating": 3,
          "comment": "Would not recommend!"
```

Description:

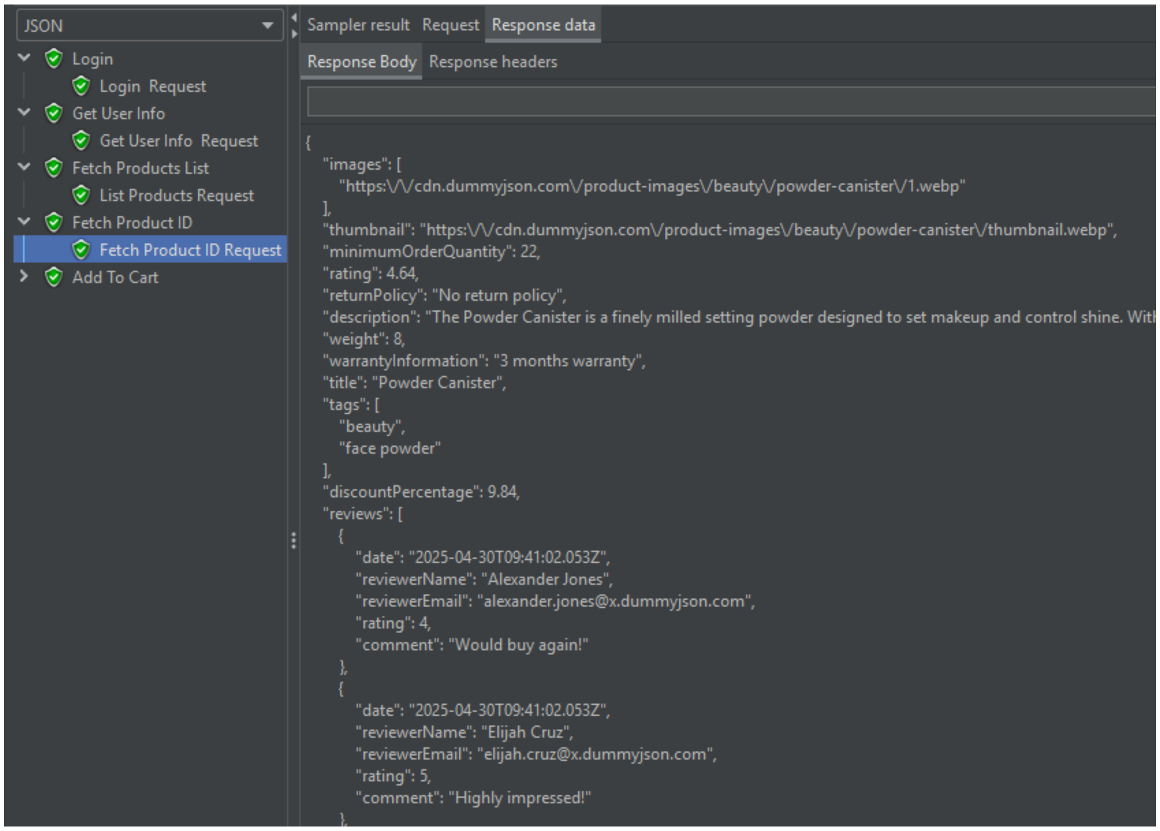
This image displays the **Response Data** tab in JMeter for the **List Products** request.

The JSON response includes a list of products(total,Limit,a list of Products), confirming that the payload structure is valid and correctly returned by the API.

Observation:

- The response structure matches the expected schema.
- It returned the product list limited to 10 product details.
- Assertions on JSON field existence (list of products) passed successfully.

Screenshot 5: Response Validation – Fetch Product ID API



```
{
  "images": [
    "https:\/\/cdn.dummyjson.com\/product-images\/beauty\/powder-canister\/1.webp"
  ],
  "thumbnail": "https:\/\/cdn.dummyjson.com\/product-images\/beauty\/powder-canister\/thumbnail.webp",
  "minimumOrderQuantity": 22,
  "rating": 4.64,
  "returnPolicy": "No return policy",
  "description": "The Powder Canister is a finely milled setting powder designed to set makeup and control shine. Wit
  "weight": 8,
  "warrantyInformation": "3 months warranty",
  "title": "Powder Canister",
  "tags": [
    "beauty",
    "face powder"
  ],
  "discountPercentage": 9.84,
  "reviews": [
    {
      "date": "2025-04-30T09:41:02.053Z",
      "reviewerName": "Alexander Jones",
      "reviewerEmail": "alexander.jones@x.dummyjson.com",
      "rating": 4,
      "comment": "Would buy again!"
    },
    {
      "date": "2025-04-30T09:41:02.053Z",
      "reviewerName": "Elijah Cruz",
      "reviewerEmail": "elijah.cruz@x.dummyjson.com",
      "rating": 5,
      "comment": "Highly impressed!"
    },
```

Description:

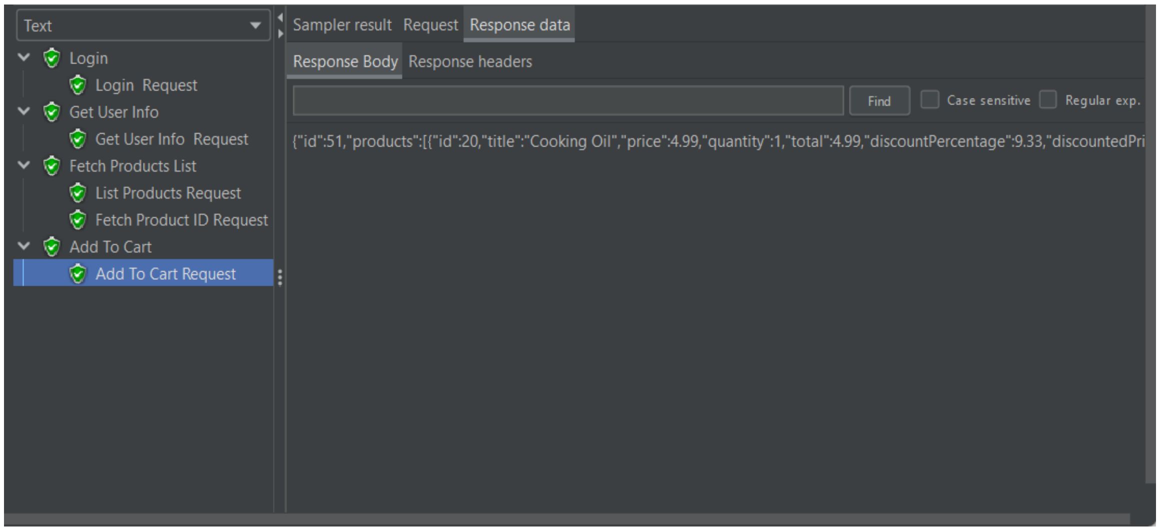
This image displays the **Response Data** tab in JMeter for the **Fetch Product Id** request.

The JSON response includes product details (images,etc), confirming that the payload structure is valid and correctly returned by the API.

Observation:

- The response structure matches the expected schema.
- It returned the product details.
- Assertions on JSON field existence (id) passed successfully.

Screenshot 6: Response Validation – Add to Cart API



```
{"id":51,"products":[{"id":20,"title":"Cooking Oil","price":4.99,"quantity":1,"total":4.99,"discountPercentage":9.33,"discountedPri
```

Description:

This image displays the **Response Data** tab in JMeter for the **Add To Cart** request.

The JSON response includes cart details (id, title, price, quantity, total, etc.), confirming that the payload structure is valid and correctly returned by the API.

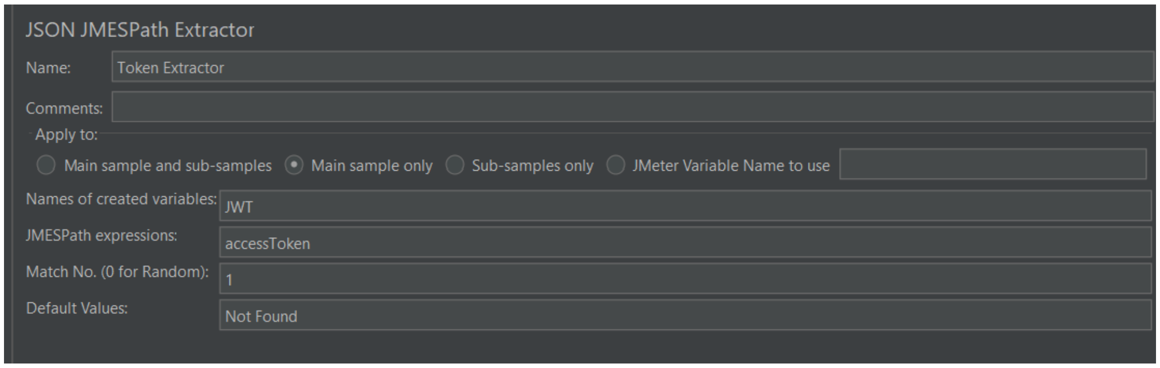
Observation:

- The response structure matches the expected schema.
- The cart creation was successful.
- Assertions on JSON field existence (total) passed successfully.

8. Token Extraction

In this step, the **JWT token** was dynamically extracted from the **Login API response** using a **JSON JMESPath Extractor**.

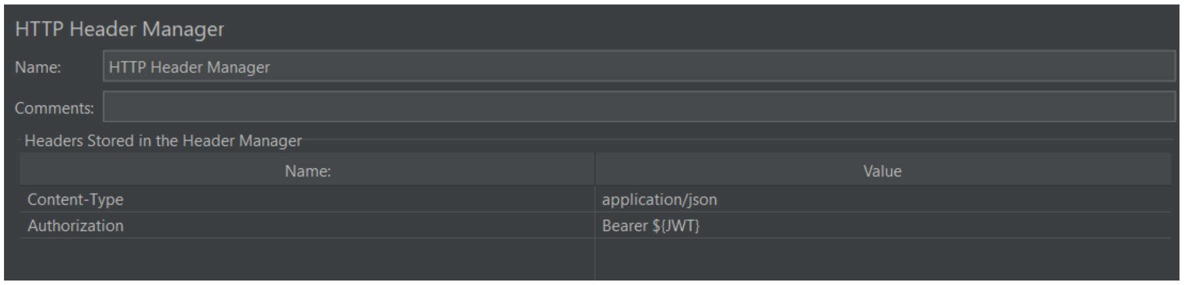
This approach implements **correlation**, allowing the test to capture runtime values and reuse them across subsequent requests.



JSON JMESPath Extractor

Name:	Token Extractor
Comments:	
Apply to:	○ Main sample and sub-samples [✓ Main sample only] ○ Sub-samples only ○ JMeter Variable Name to use
Names of created variables:	JWT
JMESPath expressions:	accessToken
Match No. (0 for Random):	1
Default Values:	Not Found

The extracted token value is stored in a JMeter variable (e.g., Bearer ${JWT}), which is then referenced within the **HTTP Header Manager**.



HTTP Header Manager

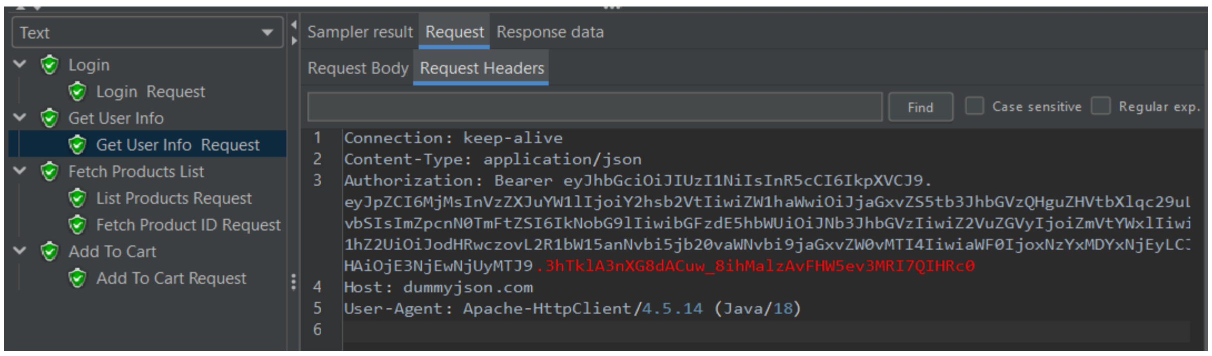
Name: HTTP Header Manager

Comments:

Headers Stored in the Header Manager

Name:	Value
Content-Type	application/json
Authorization	Bearer ${JWT}

This ensures that each request automatically includes the correct authorization.



```
Connection: keep-alive
Content-Type: application/json
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
eyJpZCI6MjMsInVzZXJuYW1lIjoiY2hsb2VtIiwiZW1haWwiOiJjaGxvZS5tb3JhbGVzQHguZHVtbXlqc29ul
vbSIsImZpcnN0TmFtZSI6IkNobG9lIiwibGFzdE5hbWUiOiJNb3JhbGVzIiwiZ2VuZGVyIjoiZmVtYWxlIiwi
1hZ2UiOiJodHRwczovL2R1bW15anNvbi5jb20vaWNvbi9jaGxvZW0vMTI4IiwiaWF0IjoxNzYxMDYxNjEyLC
HAiOjE3NjEwNjUyMTJ9.3hTklA3nXG8dACuw_8ihMalzAvFHW5ev3MRI7QIHRc0
Host: dummyjson.com
User-Agent: Apache-HttpClient/4.5.14 (Java/18)
```