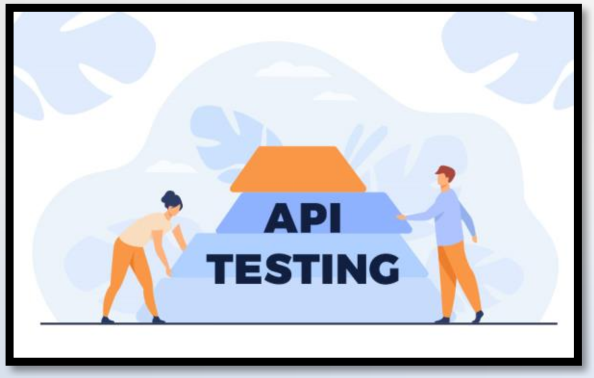


WEBSERVICES PROJECT

By ***Saurabh Kumar Singh***

ShoppersStack

USING

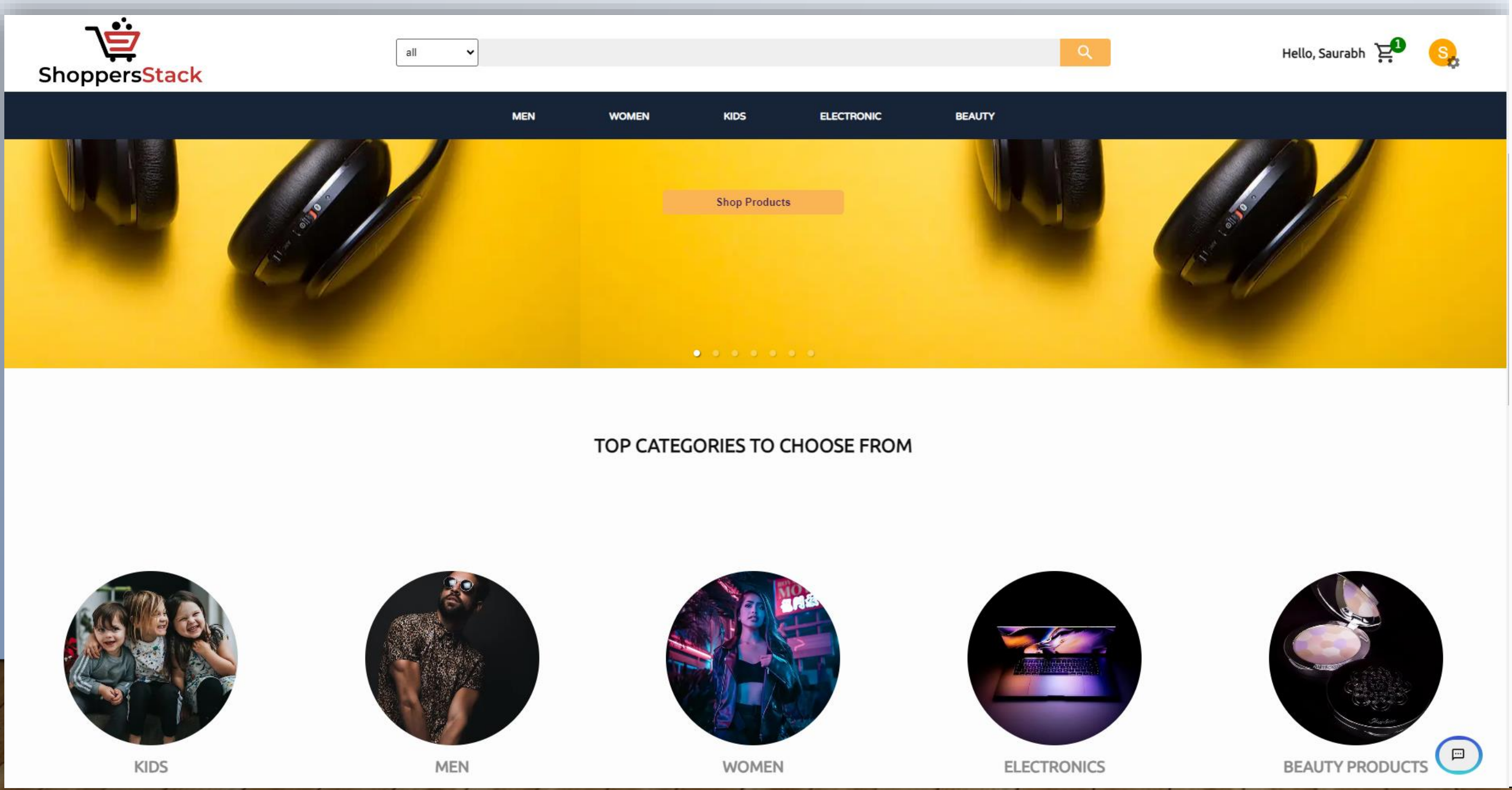
POSTMAN

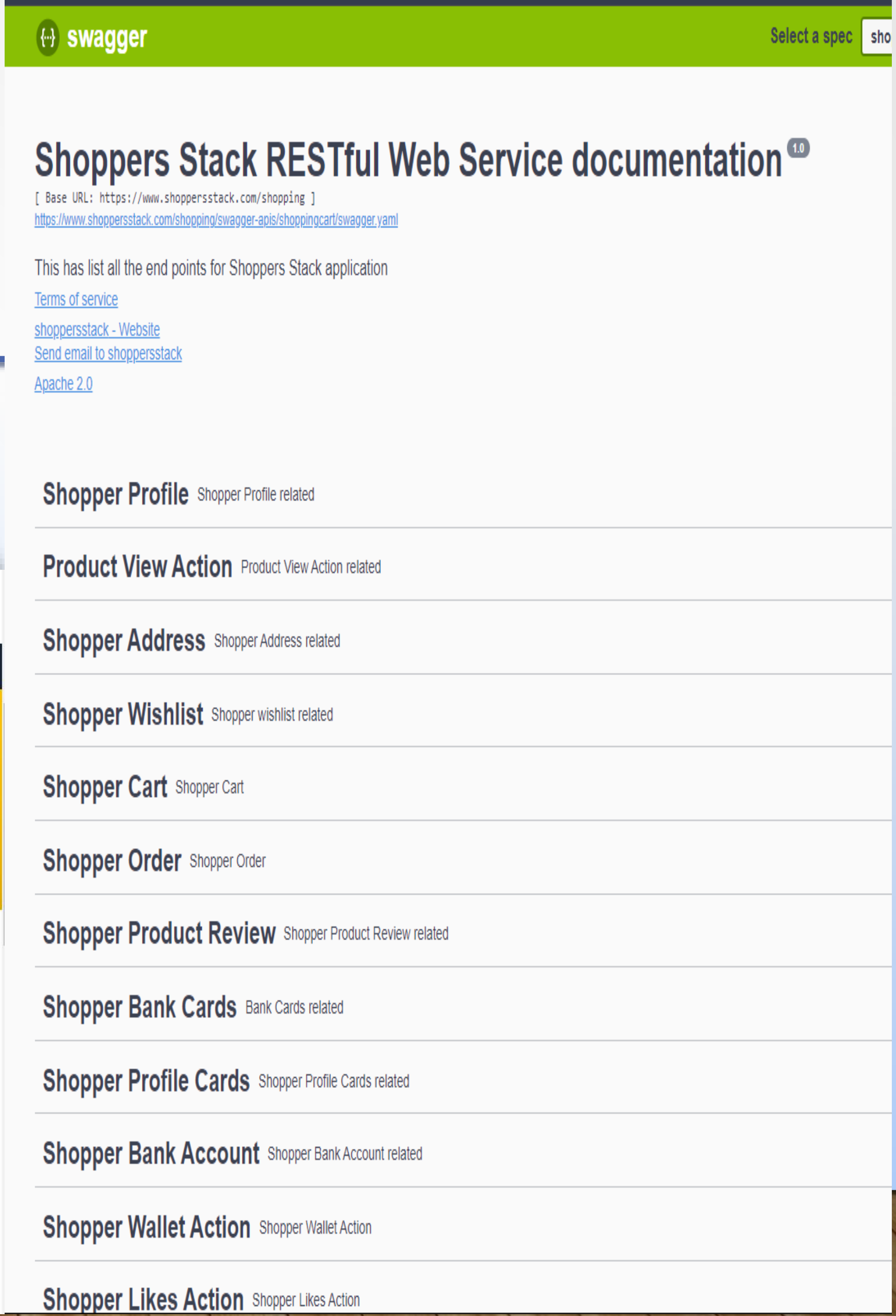
OBJECTIVE

- What : Test the functionality and performance of the ShopperStack API using Postman.
- Why : Ensure that the API functions correctly and reliably according to its specifications, provided in
  - Swagger Documentation present on the website.
- Gaining practical experience with API testing tools and methodologies.
- Understanding the process of testing and validating web services in a real-world scenario

DOMAIN

- ShoppersStack Website





Shoppers Stack RESTful Web Service documentation 1.0

[Base URL: https://www.shoppersstack.com/shopping]

https://www.shoppersstack.com/shopping/swagger-apis/shoppingcart/swagger.yaml

This has list all the end points for Shoppers Stack application

Terms of service

shoppersstack - Website

Send email to shoppersstack

Apache 2.0

Shopper Profile Shopper Profile related

Product View Action Product View Action related

Shopper Address Shopper Address related

Shopper Wishlist Shopper wishlist related

Shopper Cart Shopper Cart

Shopper Order Shopper Order

Shopper Product Review Shopper Product Review related

Shopper Bank Cards Bank Cards related

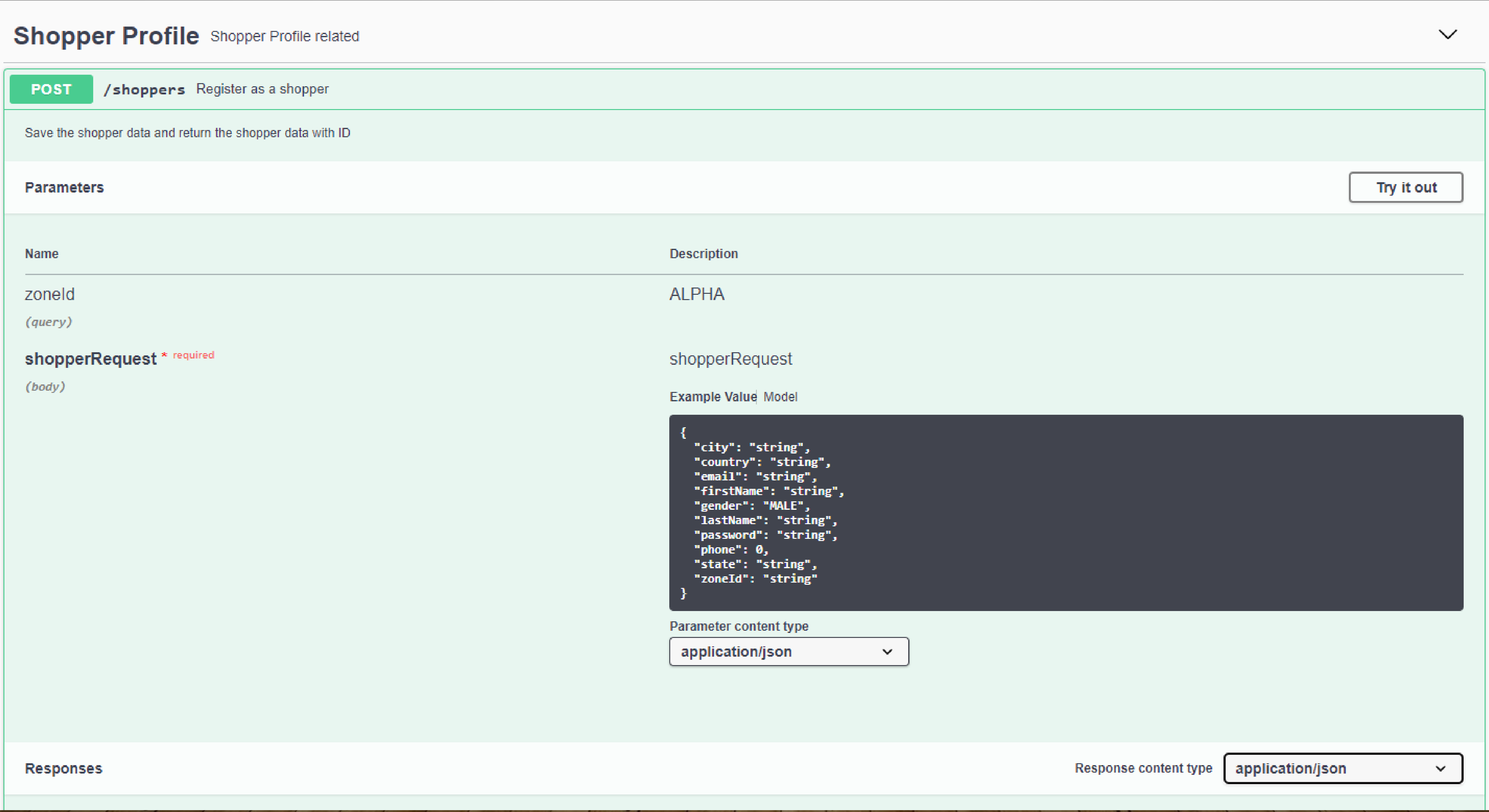
Shopper Profile Cards Shopper Profile Cards related

Shopper Bank Account Shopper Bank Account related

Shopper Wallet Action Shopper Wallet Action

Shopper Likes Action Shopper Likes Action

- Used the above data from swagger documentation to create API requests.
- Used the response data to create responses for tests.



Shopper Profile Shopper Profile related

POST /shoppers Register as a shopper

Save the shopper data and return the shopper data with ID

Parameters

Try it out

Name	Description
zoneId *(query)*	ALPHA
shopperRequest * required *(body)*	shopperRequest

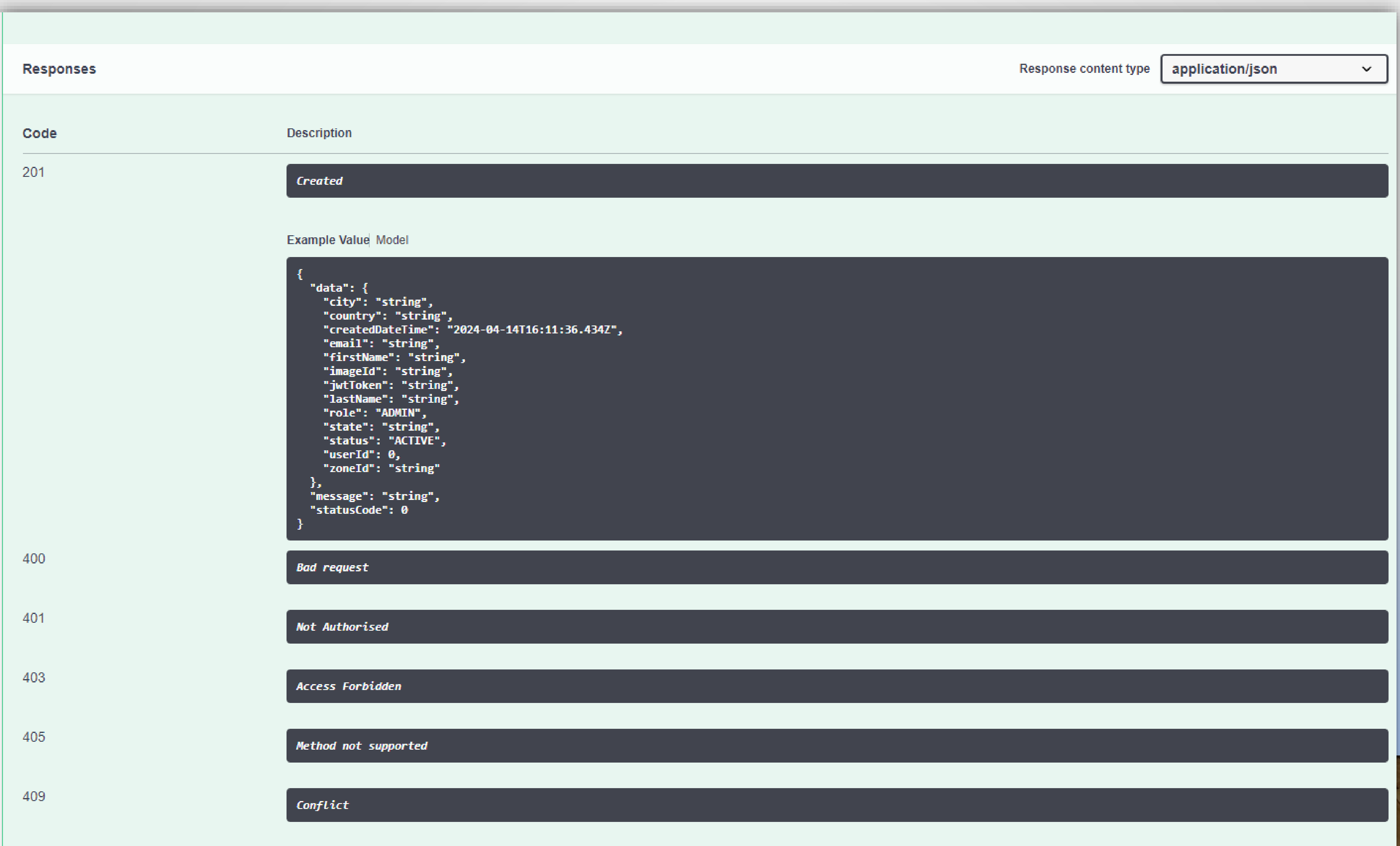
Example Value Model

```
{
  "city": "string",
  "country": "string",
  "email": "string",
  "firstName": "string",
  "gender": "MALE",
  "lastName": "string",
  "password": "string",
  "phone": 0,
  "state": "string",
  "zoneId": "string"
}
```

Parameter content type: application/json

Responses

Response content type: application/json



Responses

Response content type: application/json

Code	Description
201	*Created*
400	*Bad request*
401	*Not Authorised*
403	*Access Forbidden*
405	*Method not supported*
409	*Conflict*

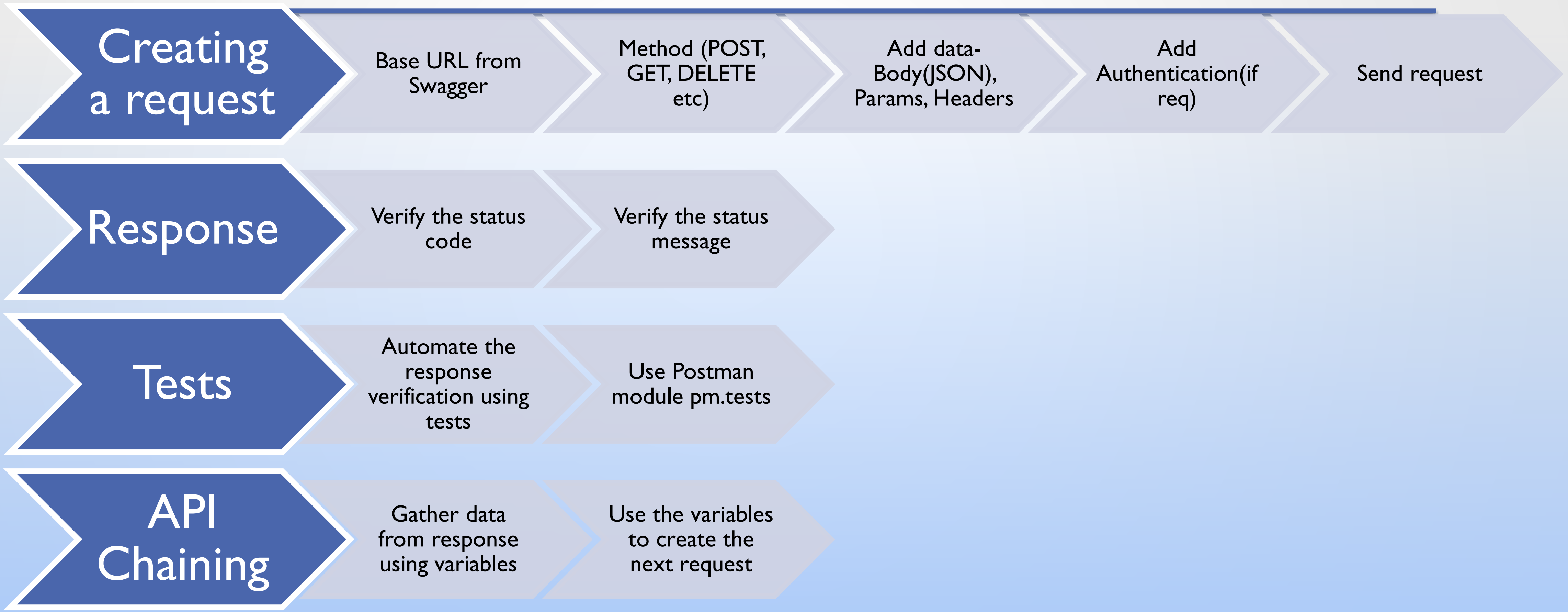
Example Value | Model

```
{
  "data": {
    "city": "string",
    "country": "string",
    "createdDateTime": "2024-04-14T16:11:36.434Z",
    "email": "string",
    "firstName": "string",
    "imageId": "string",
    "jwtToken": "string",
    "lastName": "string",
    "role": "ADMIN",
    "state": "string",
    "status": "ACTIVE",
    "userId": 0,
    "zoneId": "string"
  },
  "message": "string",
  "statusCode": 0
}
```

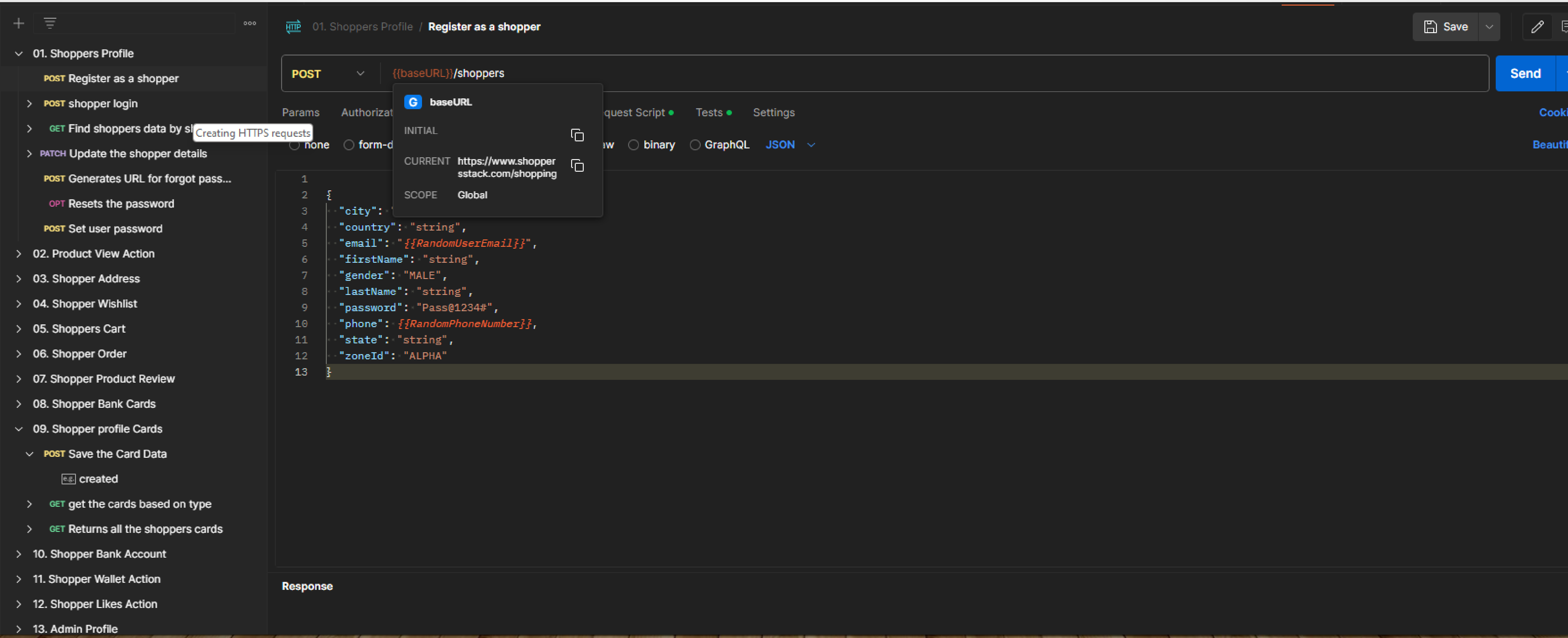
TOOLS AND ENVIRONMENTS

- Postman used as the primary tool used for testing the ShopperStack API.
  - Postman is a popular API development and testing platform that provides a user-friendly interface for designing, testing, and debugging APIs.
- Key features of Postman that were utilized in your testing process, such as:
  - Creating and sending HTTP requests (GET, POST, PUT, DELETE, etc.).
  - Writing and executing test scripts to automate validation and verification.
- There are others features provided by postman as well, but I used only these two.

FLOW CHART


Creating a request
Base URL from Swagger
Method (POST, GET, DELETE etc)
Add data-Body(JSON), Params, Headers
Add Authentication(if req)
Send request
Response
Verify the status code
Verify the status message
Tests
Automate the response verification using tests
Use Postman module pm.tests
API Chaining
Gather data from response using variables
Use the variables to create the next request

CREATING HTTPS REQUESTS



01. Shoppers Profile / Register as a shopper

- 01. Shoppers Profile
  - POST Register as a shopper
  - POST shopper login
  - GET Find shoppers data by sl
  - PATCH Update the shopper details
  - POST Generates URL for forgot pass...
  - OPT Resets the password
  - POST Set user password
- 02. Product View Action
- 03. Shopper Address
- 04. Shopper Wishlist
- 05. Shoppers Cart
- 06. Shopper Order
- 07. Shopper Product Review
- 08. Shopper Bank Cards
- 09. Shopper profile Cards
  - POST Save the Card Data
    - created
  - GET get the cards based on type
  - GET Returns all the shoppers cards
- 10. Shopper Bank Account
- 11. Shopper Wallet Action
- 12. Shopper Likes Action
- 13. Admin Profile

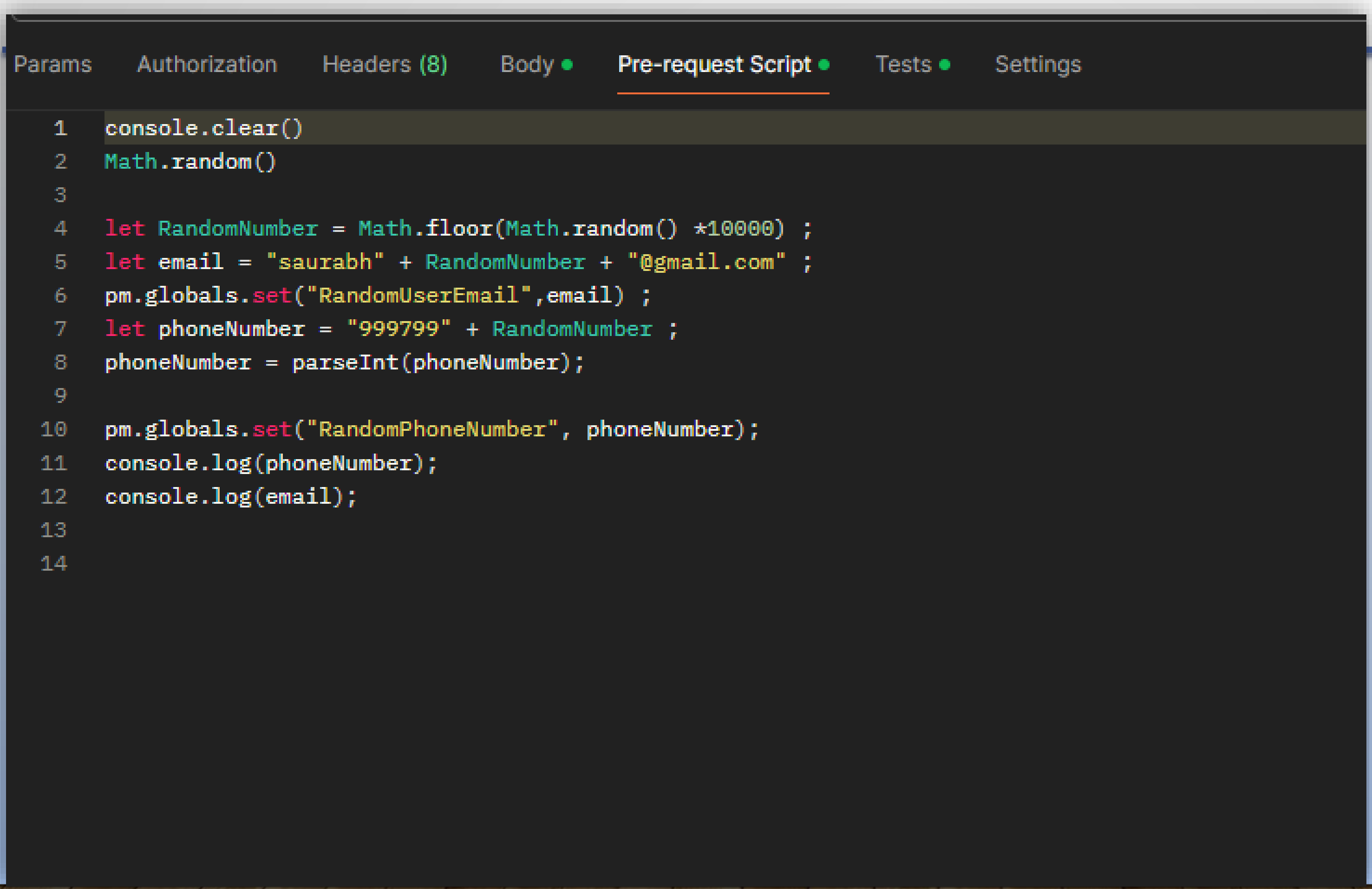
Creating HTTPS requests

POST {{baseURL}}/shoppers

baseURL
INITIAL
CURRENT https://www.shoppersstack.com/shopping
SCOPE Global

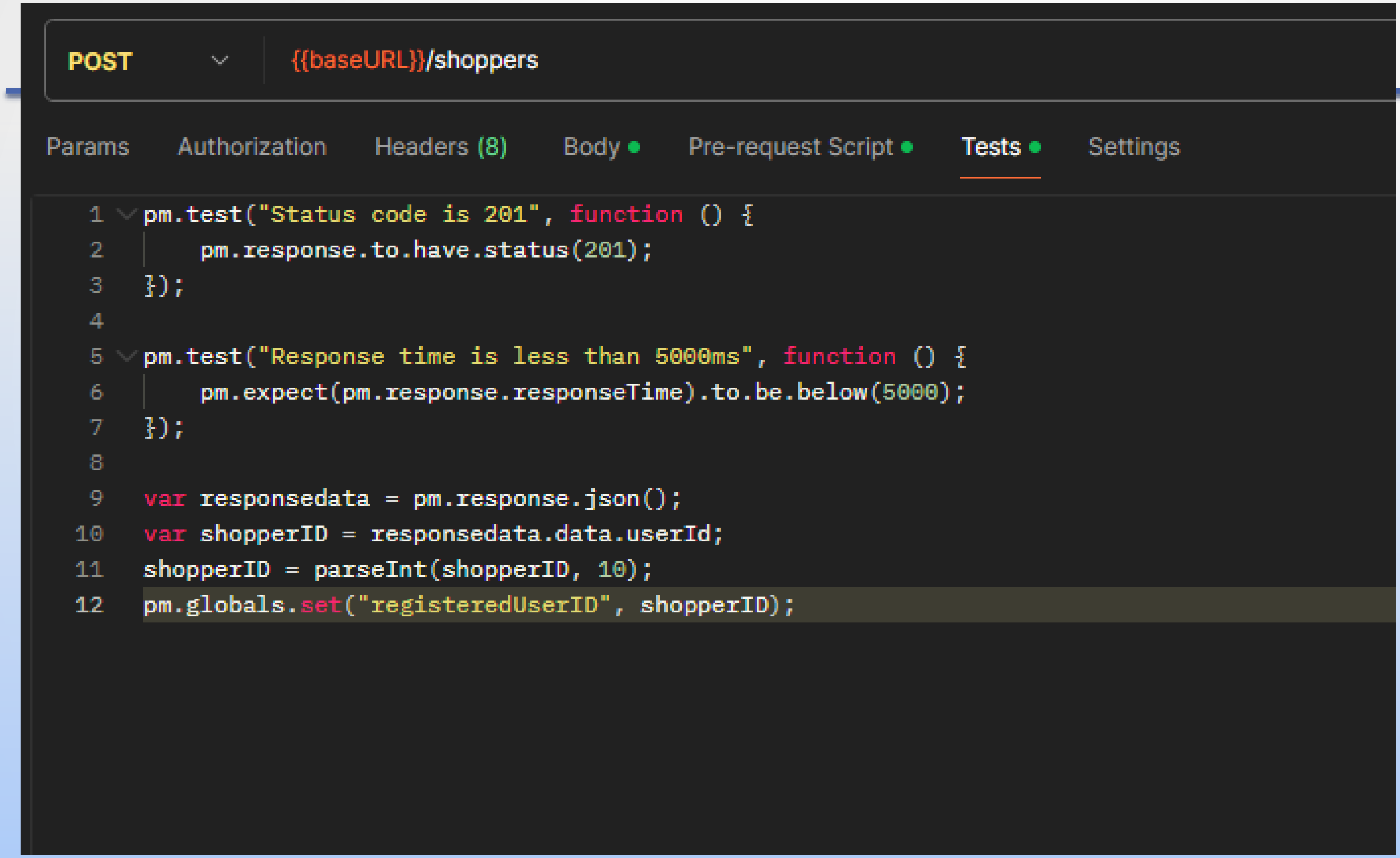
```
{
  "city": "
  "country": "string",
  "email": "{{RandomUserEmail}}",
  "firstName": "string",
  "gender": "MALE",
  "lastName": "string",
  "password": "Pass@1234#",
  "phone": {{RandomPhoneNumber}},
  "state": "string",
  "zoneId": "ALPHA"
}
```

AUTOMATION USING PRE-REQUEST SCRIPT



```
console.clear()
Math.random()

let RandomNumber = Math.floor(Math.random() *10000) ;
let email = "saurabh" + RandomNumber + "@gmail.com" ;
pm.globals.set("RandomUserEmail",email) ;
let phoneNumber = "999799" + RandomNumber ;
phoneNumber = parseInt(phoneNumber);

pm.globals.set("RandomPhoneNumber", phoneNumber);
console.log(phoneNumber);
console.log(email);


```

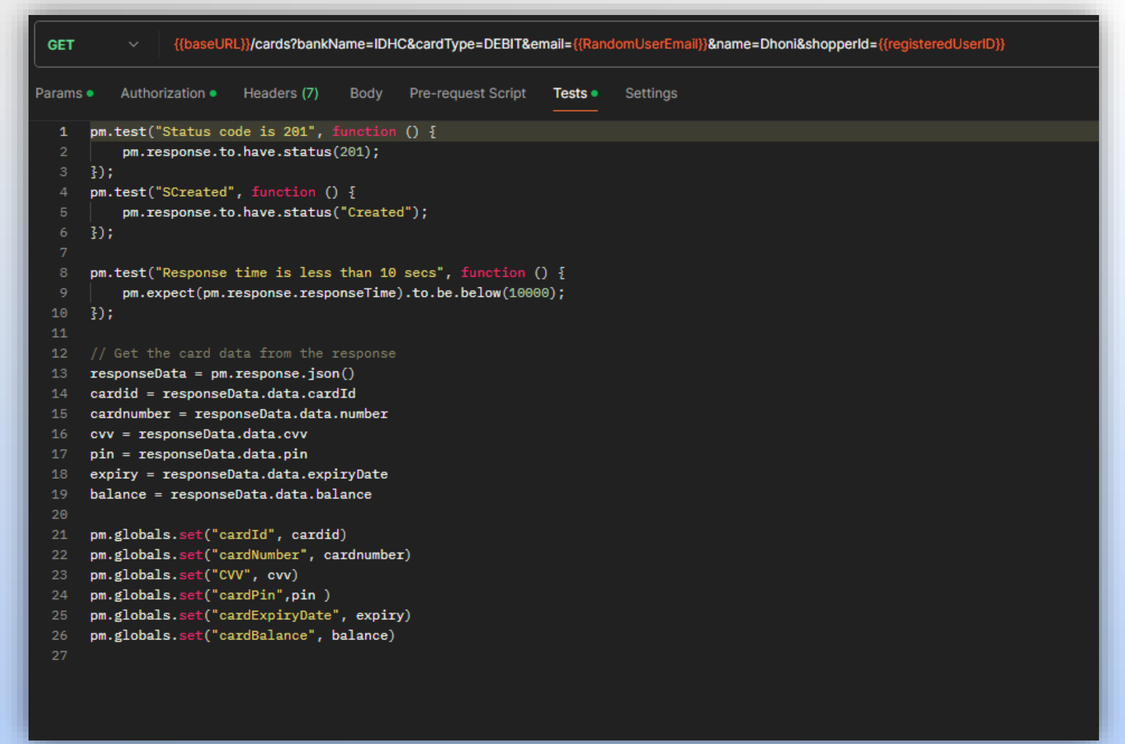
TESTING USING POST RESPONSE TEST



{{baseURL}}/shoppers

```
pm.test("Status code is 201", function () {
    pm.response.to.have.status(201);
});

pm.test("Response time is less than 5000ms", function () {
    pm.expect(pm.response.responseTime).to.be.below(5000);
});

var responsedata = pm.response.json();
var shopperID = responsedata.data.userId;
shopperID = parseInt(shopperID, 10);
pm.globals.set("registeredUserID", shopperID);
```

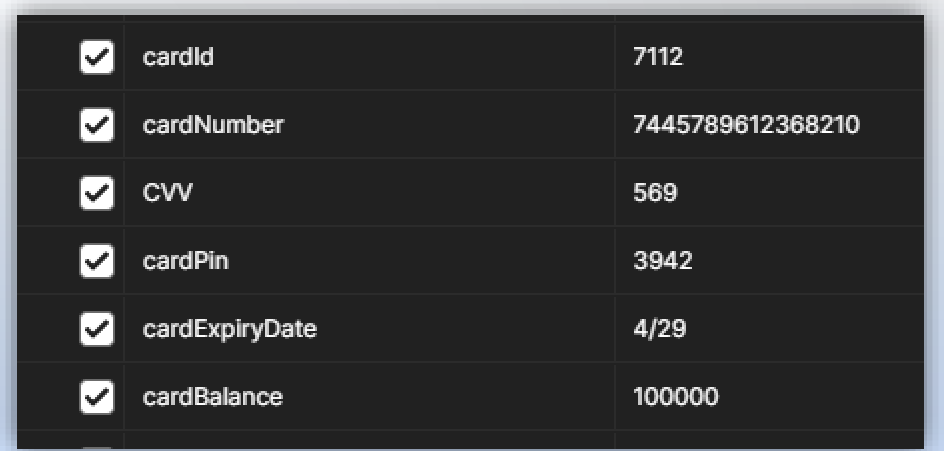
GATHERING DATA TO BE USED IN NEXT REQUEST



```
GET  {{baseURL}}/cards?bankName=IDHC&cardType=DEBIT&email={{RandomUserEmail}}&name=Dhoni&shopperId={{registeredUserID}}
```

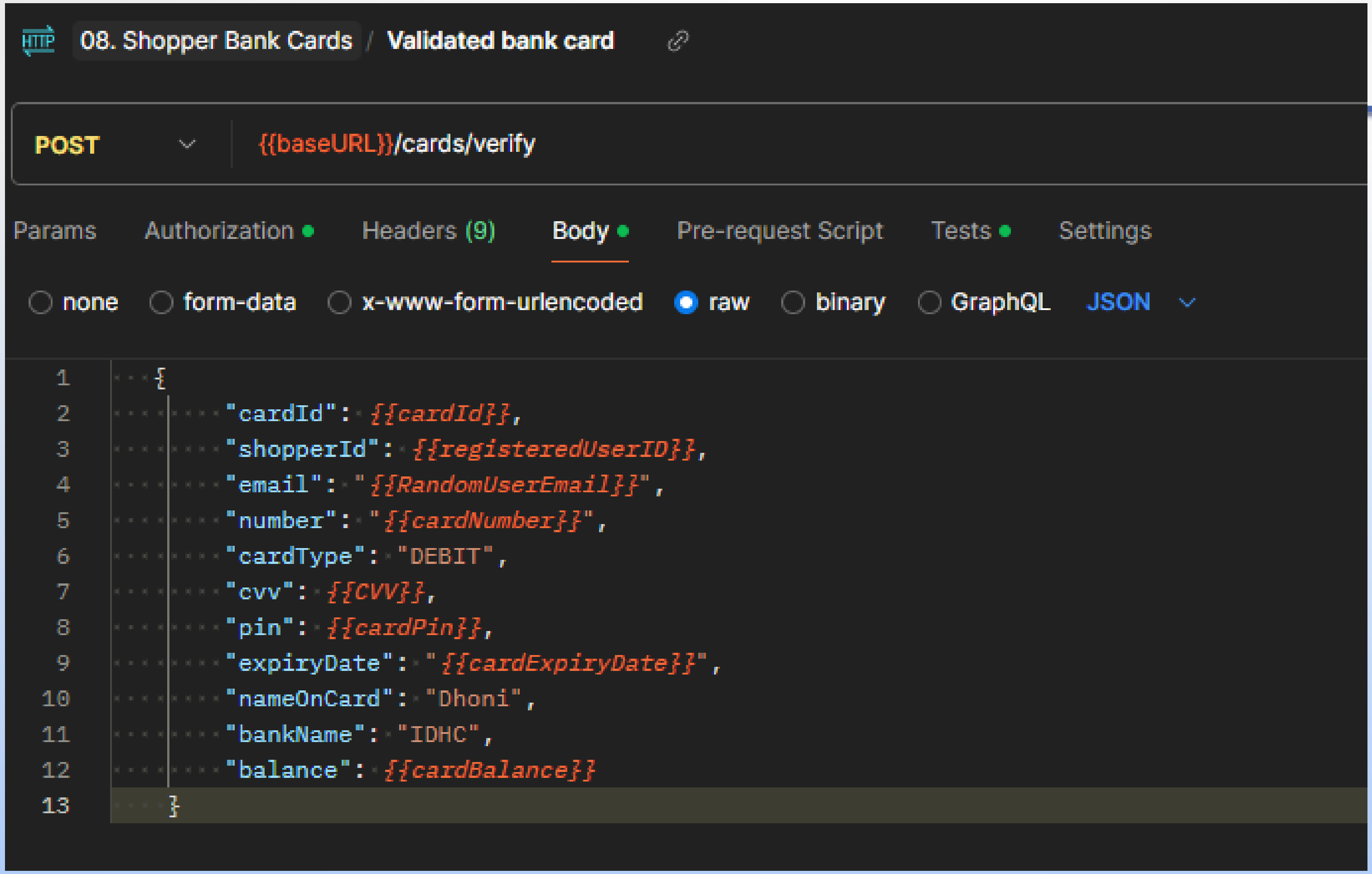
Params ● Authorization ● Headers (7) Body Pre-request Script Tests ● Settings

```
pm.test("Status code is 201", function () {
    pm.response.to.have.status(201);
});
pm.test("SCreated", function () {
    pm.response.to.have.status("Created");
});

pm.test("Response time is less than 10 secs", function () {
    pm.expect(pm.response.responseTime).to.be.below(10000);
});

// Get the card data from the response
responseData = pm.response.json()
cardid = responseData.data.cardId
cardnumber = responseData.data.number
cvv = responseData.data.cvv
pin = responseData.data.pin
expiry = responseData.data.expiryDate
balance = responseData.data.balance

pm.globals.set("cardId", cardid)
pm.globals.set("cardNumber", cardnumber)
pm.globals.set("CVV", cvv)
pm.globals.set("cardPin",pin )
pm.globals.set("cardExpiryDate", expiry)
pm.globals.set("cardBalance", balance)
```



✓	cardId	7112
✓	cardNumber	7445789612368210
✓	CVV	569
✓	cardPin	3942
✓	cardExpiryDate	4/29
✓	cardBalance	100000

CHAINING

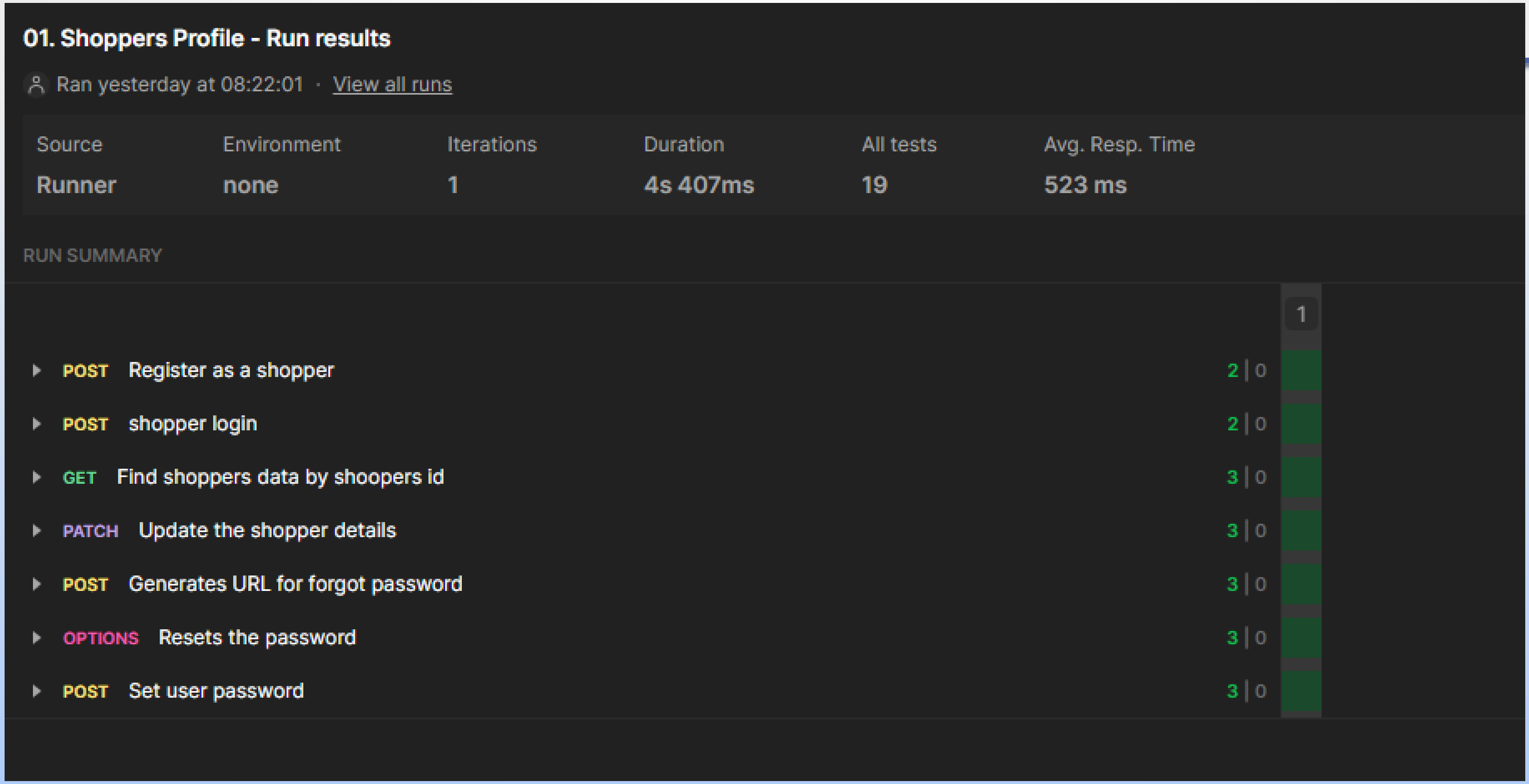


POST {{baseURL}}/cards/verify

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

```
{
    "cardId": {{cardId}},
    "shopperId": {{registeredUserID}},
    "email": "{{RandomUserEmail}}",
    "number": "{{cardNumber}}",
    "cardType": "DEBIT",
    "cvv": {{CVV}},
    "pin": {{cardPin}},
    "expiryDate": "{{cardExpiryDate}}",
    "nameOnCard": "Dhoni",
    "bankName": "IDHC",
    "balance": {{cardBalance}}
}
```

AUTOMATION AND RESULTS



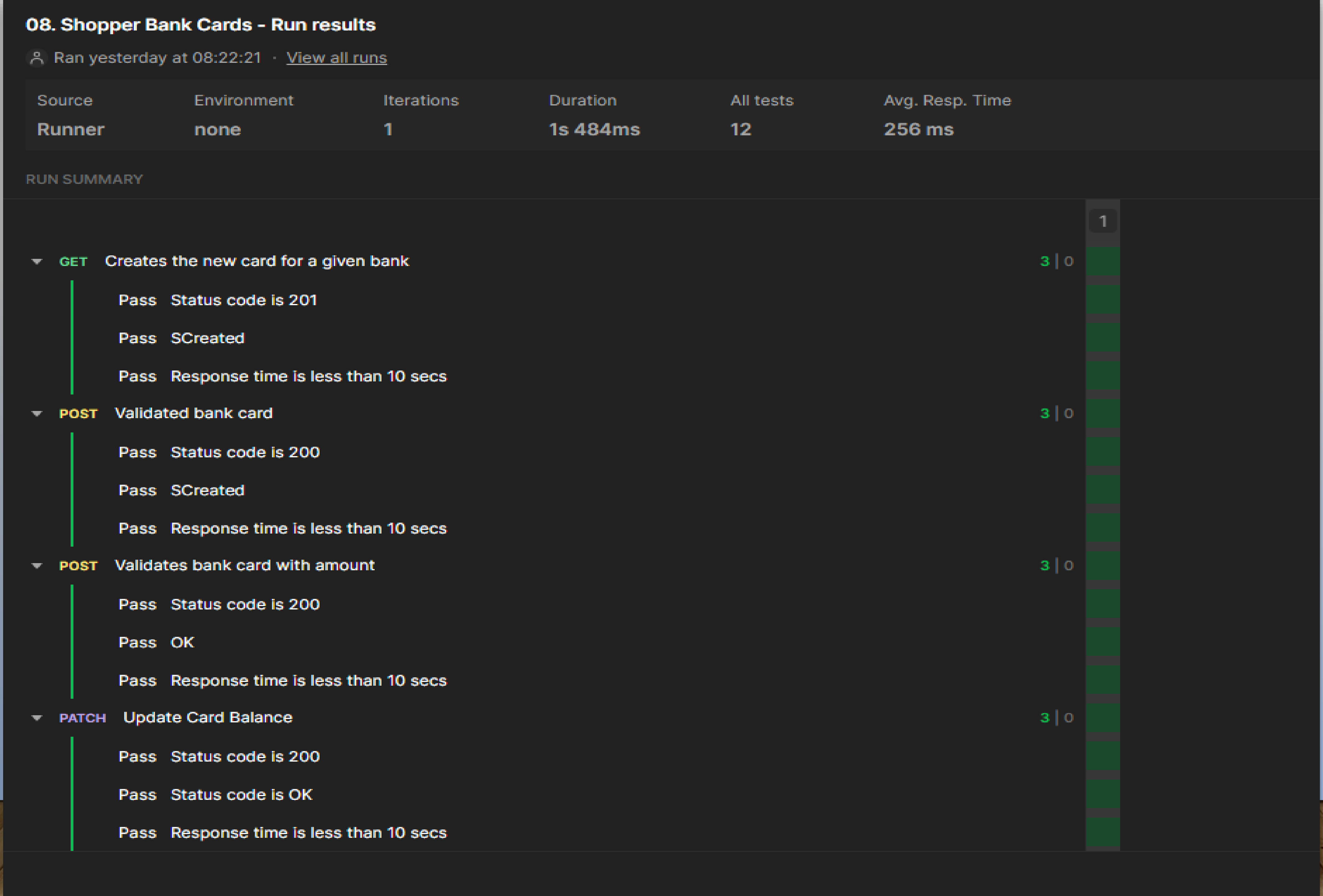
01. Shoppers Profile - Run results

Ran yesterday at 08:22:01 · View all runs

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	1	4s 407ms	19	523 ms

RUN SUMMARY

			1
POST	Register as a shopper	2 | 0	🟩
POST	shopper login	2 | 0	🟩
GET	Find shoppers data by shoopers id	3 | 0	🟩
PATCH	Update the shopper details	3 | 0	🟩
POST	Generates URL for forgot password	3 | 0	🟩
OPTIONS	Resets the password	3 | 0	🟩
POST	Set user password	3 | 0	🟩



08. Shopper Bank Cards - Run results

Ran yesterday at 08:22:21 · View all runs

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	1	1s 484ms	12	256 ms

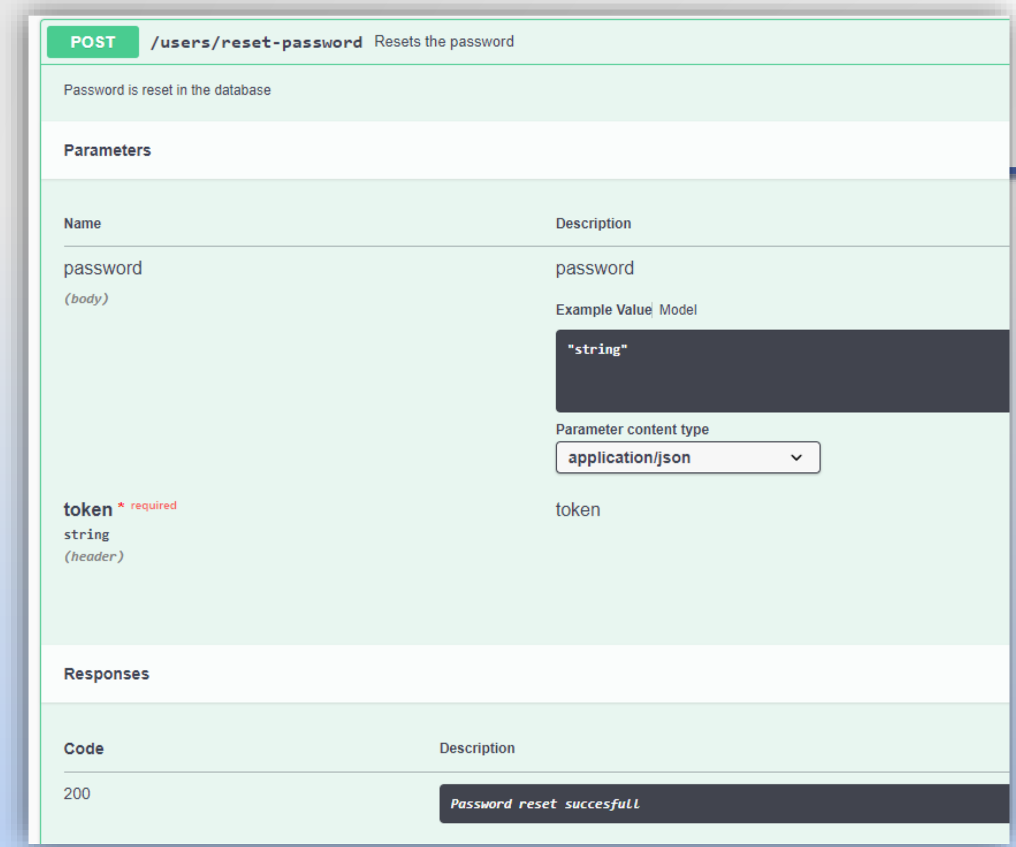
RUN SUMMARY

1

```
GET    Creates the new card for a given bank          3 | 0
         Pass  Status code is 201
         Pass  SCreated
         Pass  Response time is less than 10 secs
POST   Validated bank card                            3 | 0
         Pass  Status code is 200
         Pass  SCreated
         Pass  Response time is less than 10 secs
POST   Validates bank card with amount                3 | 0
         Pass  Status code is 200
         Pass  OK
         Pass  Response time is less than 10 secs
PATCH  Update Card Balance                            3 | 0
         Pass  Status code is 200
         Pass  Status code is OK
         Pass  Response time is less than 10 secs
```

ANALYSIS

2 Defects found



POST /users/reset-password Resets the password

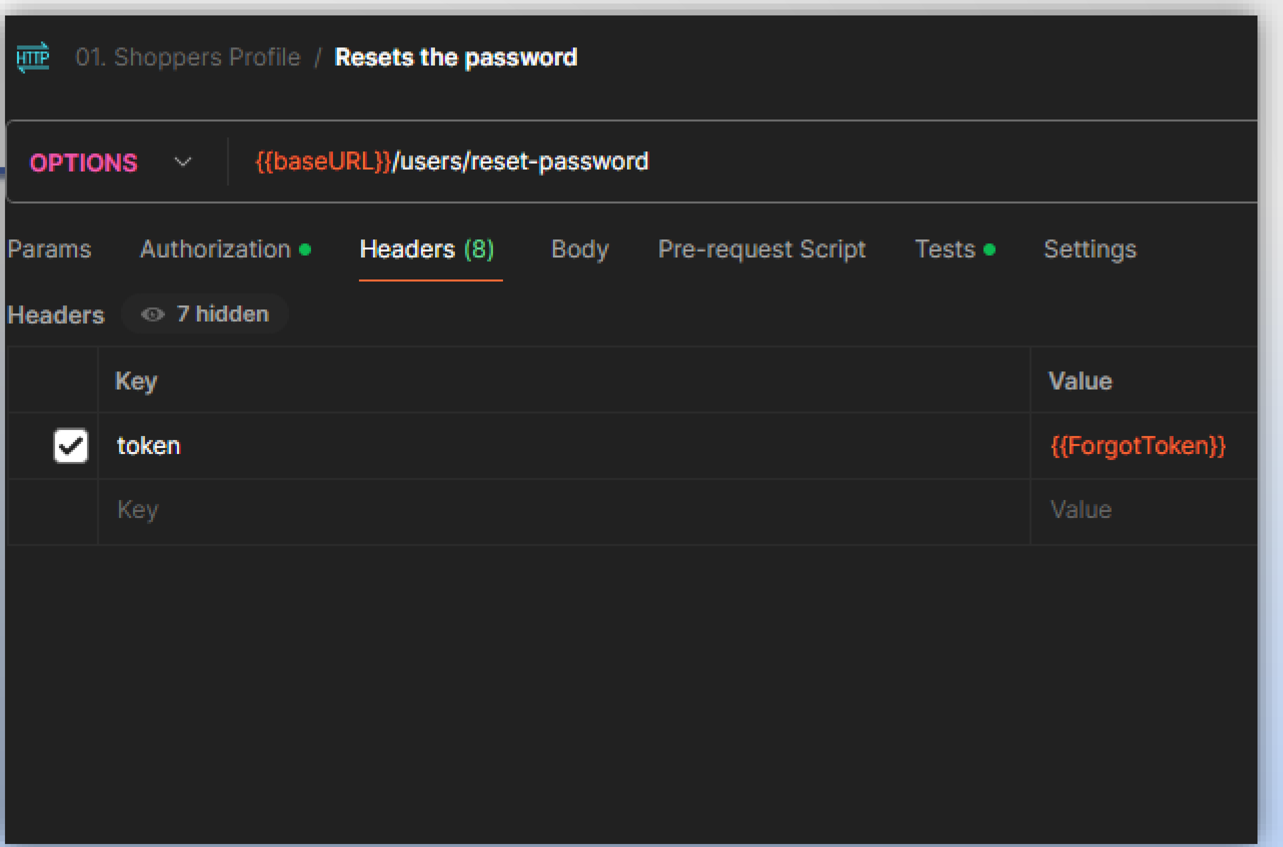
Password is reset in the database

Parameters

Name	Description
password *(body)*	password Example Value Model "string" Parameter content type application/json
token * required string *(header)*	token

Responses

Code	Description
200	Password reset succesfull

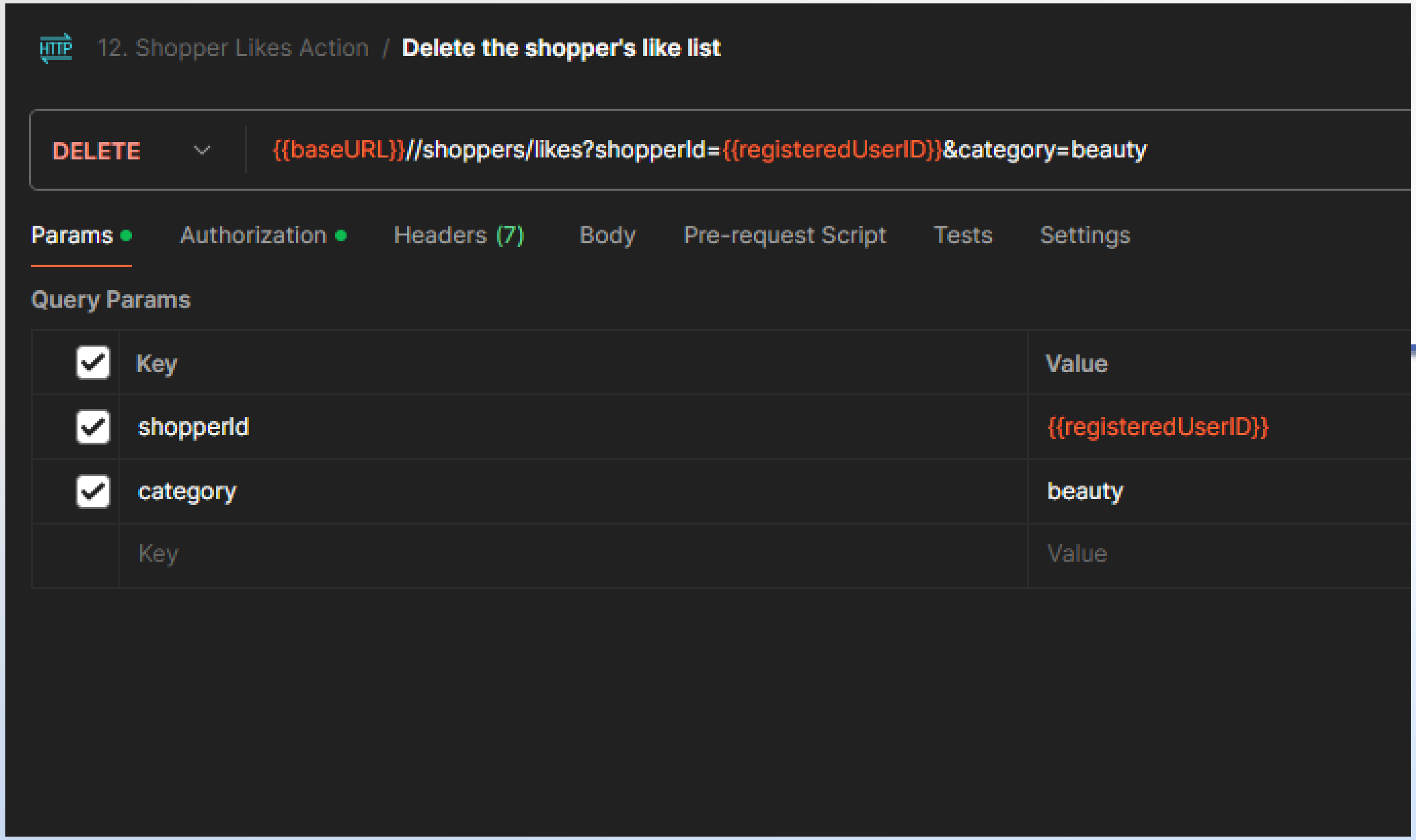


01. Shoppers Profile / Resets the password

OPTIONS {{baseURL}}/users/reset-password

	Key	Value
✓	token	{{ForgotToken}}
	Key	Value

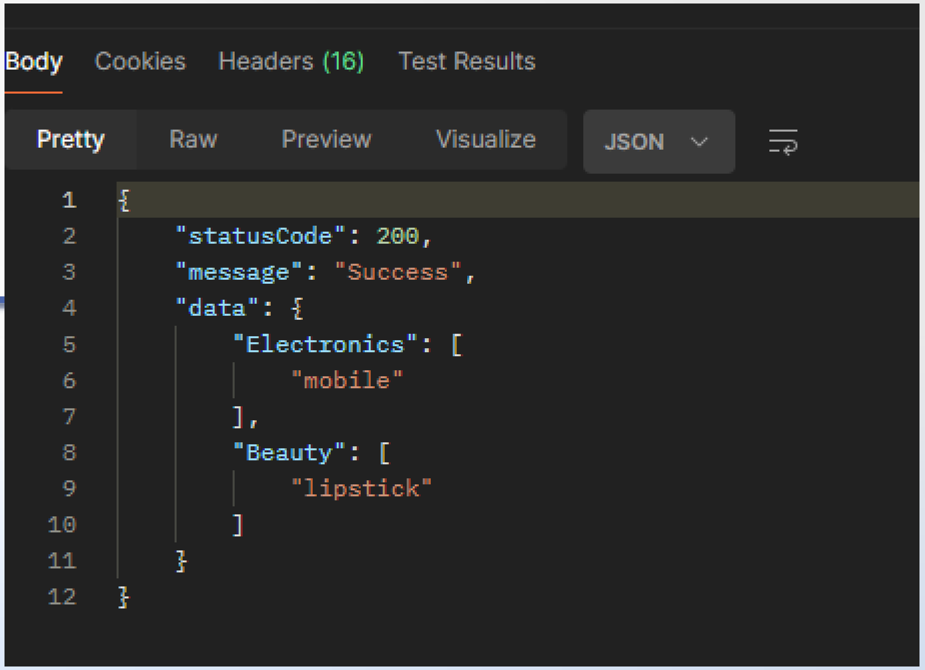
D1 :Wrong method used at backed which is not mentioned in the Swagger document


12. Shopper Likes Action / **Delete the shopper's like list**

DELETE {{baseURL}}//shoppers/likes?shopperId={{registeredUserID}}&category=beauty

Query Params

✓	Key	Value
✓	shopperId	{{registeredUserID}}
✓	category	beauty
	Key	Value



```
{
    "statusCode": 200,
    "message": "Success",
    "data": {
        "Electronics": [
            "mobile"
        ],
        "Beauty": [
            "lipstick"
        ]
    }
}
```

D2(?) :The category is case sensitive, so deleting beauty doesn’t work, but deleting Beauty does.

THANKS !!!