

Screenshots of ThunderClient tests (Register, Login, Products, Cart)

The screenshot shows a POST request to `http://localhost:5000/api/auth/login`. The request body is JSON containing email and password. The response status is 200 OK, size 232 bytes, and time 365 ms. The response body contains a success message and a token.

```

POST http://localhost:5000/api/auth/login
Status: 200 OK Size: 232 Bytes Time: 365 ms

Response Headers: 7 Cookies: Results: Docs: {}

1 {
2   "success": true,
3   "message": "Login successful",
4   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
      .eyJ1c2VycWQiOiI2OTMzMzMy4NjlkOGQyZDIZTTjimZAxMGYiLCJpYXQiO
      jE3NjUwMTU2NjMsImVaccI6MTc2NTAxDOTI2M30
      .jbeYf3QDQ749_LgZ17QWppnGCyppOz6ow91THUwSTQ"
5 }
  
```

TERMINAL:

```
C:\Users\verma\OneDrive\Desktop\React-Project\shoppy-globe\backend>npm run dev
> backend@1.0.0 dev
> nodemon server.js

[nodemon] 3.1.11
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
[dotenv@17.2.3] injecting env (3) from .env -- tip: ⚡️ backup and recover secrets: https://dotenvx.com/ops
⚡️ Server running on http://localhost:5000
MongoDB connected
```

The screenshot shows a GET request to `http://localhost:5000/api/products`. The response status is 200 OK, size 68 bytes, and time 86 ms. The response body indicates success, message, and an empty data array.

```

GET http://localhost:5000/api/products
Status: 200 OK Size: 68 Bytes Time: 86 ms

Response Headers: 7 Cookies: Results: Docs: {}

1 {
2   "success": true,
3   "message": "Products fetched successfully",
4   "data": []
5 }
  
```

TERMINAL:

```
C:\Users\verma\OneDrive\Desktop\React-Project\shoppy-globe\backend>npm run dev
> backend@1.0.0 dev
> nodemon server.js

[nodemon] 3.1.11
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): ***!
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
[dotenv@17.2.3] injecting env (3) from .env -- tip: ⚡️ backup and recover secrets: https://dotenvx.com/ops
⚡️ Server running on http://localhost:5000
MongoDB connected
```

GET <http://localhost:5000/api/products> Send

Query Headers 2 Auth Body Tests Pre Run

Query Parameters

parameter	value
-----------	-------

Status: 200 OK Size: 329 Bytes Time: 190 ms

Response Headers 7 Cookies Results Docs { }

```
1 {
2   "success": true,
3   "message": "Products fetched successfully",
4   "data": [
5     {
6       "stockQuantity": 0,
7       "image": "https://via.placeholder.com/300",
8       "rating": 0,
9       "_id": "69340225b8d7bd5dcaa95178",
10      "name": "Sample T-Shirt",
11      "description": "Blue cotton t-shirt for testing",
12      "price": 499,
13      "category": "Clothing",
14      "stock": 20,
15      "createdAt": "2025-12-06T10:15:39.518Z"
}
```

Response Chart

GET <http://localhost:5000/api/products/69340225b8d7bd5dcaa95178> Send

Query Headers 2 Auth Body Tests Pre Run

Query Parameters

parameter	value
-----------	-------

Status: 200 OK Size: 326 Bytes Time: 98 ms

Response Headers 7 Cookies Results Docs { }

```
1 {
2   "success": true,
3   "message": "Product fetched successfully",
4   "data": {
5     "stockQuantity": 0,
6     "image": "https://via.placeholder.com/300",
7     "rating": 0,
8     "_id": "69340225b8d7bd5dcaa95178",
9     "name": "Sample T-Shirt",
10    "description": "Blue cotton t-shirt for testing",
11    "price": 499,
12    "category": "Clothing",
13    "stock": 20,
14    "createdAt": "2025-12-06T10:18:06.405Z"
}
```

Response Chart

GET <http://localhost:5000/api/products/69340225b8d7bd5dcaa951> Send

Query Headers 2 Auth Body Tests Pre Run

Query Parameters

parameter	value
-----------	-------

Status: 400 Bad Request Size: 55 Bytes Time: 6 ms

Response Headers 7 Cookies Results Docs { }

```
1 {
2   "success": false,
3   "message": "Invalid product ID format"
4 }
```

Response Chart

POST <http://localhost:5000/api/cart>

Body 1

```
{
  "productId": "69340225b8d7bd5dcaa95178",
  "quantity": 2
}
```

Status: 400 Bad Request Size: 62 Bytes Time: 207 ms

Response Headers Cookies Results Docs

```
1 {
2   "success": false,
3   "message": "Insufficient stock. Available: 0"
4 }
```

POST <http://localhost:5000/api/cart>

Body 1

```
{
  "productId": "69340225b8d7bd5dcaa95178",
  "quantity": 2
}
```

Status: 201 Created Size: 390 Bytes Time: 723 ms

Response Headers Cookies Results Docs

```
1 {
2   "success": true,
3   "message": "Product added to cart successfully",
4   "data": {
5     "userId": "6933ff869d8d2d23e2b3010f",
6     "items": [
7       {
8         "productId": "69340225b8d7bd5dcaa95178",
9         "quantity": 2,
10        "price": 499,
11        "id": "6934053dcbab06de69eeeeeaa7",
12        "addedAt": "2025-12-06T10:28:13.676Z"
13      },
14    ],
15   "totalPrice": 998,
16 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

C:\Users\verma\OneDrive\Desktop\React-Project\shopy-globe\backend>npm run dev

```
> backend@1.0.0 dev
> nodemon server.js

[nodemon] 3.1.11
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
[dotenv@17.2.3] injecting env (3) from .env -- tip: 📁 backup and recover secrets: https://dotenvx.com/ops
⚡ Server running on http://localhost:5000
MongoDB connected
```

PUT <http://localhost:5000/api/cart/69340225b8d7bd5dcaa95178>

Body 1

```
{
  "quantity": 3
}
```

Status: 200 OK Size: 382 Bytes Time: 1.67 s

Response Headers Cookies Results Docs

```
1 {
2   "success": true,
3   "message": "Cart updated successfully",
4   "data": {
5     "_id": "6934053dcbab06de69eeeeeaa6",
6     "userId": "6933ff869d8d2d23e2b3010f",
7     "items": [
8       {
9         "productId": "69340225b8d7bd5dcaa95178",
10        "quantity": 3,
11        "price": 499,
12        "id": "6934053dcbab06de69eeeeeaa7",
13        "addedAt": "2025-12-06T10:28:13.676Z"
14      },
15    ],
16   "totalPrice": 998,
17 }
```

The screenshot shows a Postman request to `http://localhost:5000/api/cart/69340225b8d7bd5dcaa95178`. The request method is `DELETE`. The response status is `200 OK`, size is `257 Bytes`, and time is `335 ms`. The response body is a JSON object:

```

1  {
2    "success": true,
3    "message": "Product removed from cart successfully",
4    "data": {
5      "_id": "6934053dcbab06de69eeeea6",
6      "userId": "6933ff869d8d2d23e2b3010f",
7      "items": [],
8      "totalPrice": 0,
9      "createdAt": "2025-12-06T10:28:13.680Z",
10     "updatedAt": "2025-12-06T10:32:52.718Z",
11     "__v": 1
12   }
13 }

```

Screenshot of MongoDB collections

The screenshot shows the MongoDB Data Explorer for the `Cluster0 > shoppyglobe > orders` collection. There are 3 documents listed:

- `_id: ObjectId('69355088ed677ffba93e519')`
`userId: ObjectId('6933ff869d8d2d23e2b3010f')`
`items: Array (1)`
`totalPrice: 1798`
`shippingDetails: Object`
`status: "Pending"`
`createdAt: 2025-12-07T10:00:26.441+00:00`
`updatedAt: 2025-12-07T10:00:26.441+00:00`
`__v: 0`
- `_id: ObjectId('69355088ed677ffba93e534')`
`userId: ObjectId('6933ff869d8d2d23e2b3010f')`
`items: Array (1)`
`totalPrice: 1299`
`shippingDetails: Object`
`status: "Pending"`
`createdAt: 2025-12-07T10:01:44.428+00:00`

The screenshot shows the MongoDB Data Explorer for the `Cluster0 > shoppyglobe > carts` collection. There is 1 document listed:

- `_id: ObjectId('6934053dcbab06de69eeeea6')`
`userId: ObjectId('6933ff869d8d2d23e2b3010f')`
`items: Array (1)`
`totalPrice: 449`
`createdAt: 2025-12-06T10:28:13.680+00:00`
`updatedAt: 2025-12-07T10:04:33.372+00:00`
`__v: 20`

Cluster0 > shoppyglobe > products

Documents 36 **Aggregations** **Schema** **Indexes** 1 **Validation**

Type a query: { field: 'value' } or [Generate query](#):

ADD DATA **UPDATE** **DELETE**

Document 1

```
_id: ObjectId('693425aa2ceca54002c071')
name: "Premium Lipstick"
description: "Long-lasting and vibrant lipstick with matte finish"
price: 499
stockQuantity: 50
category: "makeup"
image: "https://plus.unsplash.com/premium_photo-1661772819014-1fe8103e12b?w=4..."
rating: 4.5
createdAt: 2025-12-06T12:46:34.267+00:00
__v: 0
updatedAt: 2025-12-06T12:46:34.275+00:00
```

Document 2

```
_id: ObjectId('693425aa2ceca54002c072')
name: "Mascara Waterproof"
description: "Waterproof mascara for dramatic lashes"
price: 349
```

Cluster0 > shoppyglobe > users

Documents 1 **Aggregations** **Schema** **Indexes** 3 **Validation**

Type a query: { field: 'value' } or [Generate query](#):

ADD DATA **UPDATE** **DELETE**

Document 1

```
_id: ObjectId('6933ff069d8d2d23e2b3010f')
username: "khushboo"
email: "khushbo@example.com"
password: "$2b$10$J3AQLE51G@B1l0rx18lLutBkCcIpAwC4gvzP28ugvtPRAlnEVVAq"
createdAt: 2025-12-06T10:03:50.541+00:00
updatedAt: 2025-12-06T10:03:50.541+00:00
__v: 0
```

Checkout success page

ShoppyGlobe

Home Cart Logout Register

Order Placed Successfully!

Thank you for your purchase. Your order has been confirmed.

Redirecting to home page...

Website look :

The image displays three screenshots of the ShoppyGlobe e-commerce platform.

Home Page: The header features the "ShoppyGlobe" logo. The main banner says "Welcome to ShoppyGlobe" and "Discover amazing products from around the world". A search bar is present. Below, a section titled "Shop by Category" lists five categories: Groceries, Electronics, Home Decor, Jewellery, and Clothes & Fashion, each with a "View Products" button.

Login Page: The header shows the "ShoppyGlobe" logo. A central modal window is titled "Login" with the sub-instruction "Use your ShoppyGlobe account to continue". It contains fields for "Email" (with placeholder "khushboo@example.com") and "Password" (with placeholder "*****" and a "Show" link). A large orange "Login" button is at the bottom.

Cart Page: The header includes the "ShoppyGlobe" logo and navigation links: Home, Cart, Logout, and Register. The main content area displays the message "Your cart is empty" and "Start shopping to add items to your cart", with an orange "Continue Shopping" button.

Page Footer: At the bottom of the site, there is a dark footer bar containing copyright information ("© 2025 ShoppyGlobe. All rights reserved."), links to "About Us", "Contact", and "Privacy Policy", and a social media icon for GitHub.

Create Account

Join ShoppyGlobe to start shopping.

Username

Email

Password

[Show](#)[Sign Up](#)

Already have an account? [Log in](#)