

API testing for ShoppyGlobe application using Thunder Client

1. GET /products

Fetch a list of products from MongoDB.

The screenshot shows the Thunder Client interface. In the top navigation bar, there is a dropdown menu labeled "THUNDER CLIENT" with a "New Request" option. Below the menu, a search bar contains the URL "localhost:4500/products". The main area displays a "GET" request with the URL "http://www.localhost:4500/products". The response status is "200 OK", the size is "896 Bytes", and the time taken is "34 ms". The "Body" tab is selected, showing a JSON response with two products:

```
[{"id": 1, "name": "Kiwi", "price": 99, "description": "Nutrient-rich kiwi, perfect for snacking or adding a tropical twist to your dishes.", "stock": 100}, {"id": 2, "name": "Ice cream", "price": 65, "description": "Creamy and delicious ice cream, available in various flavors for a delightful treat."}]
```

The screenshot shows the MongoDB Compass interface. On the left, the "Connections" sidebar lists several databases, with "ShoppyGlobe" selected. Under "ShoppyGlobe", the "products" collection is highlighted. The main panel shows the "products" collection with 5 documents. A query builder is present at the top right, and a table below displays the document details:

_id	id	name	price	description	stock
ObjectId('687e6f836d0c320a6c718dd8')	1	Kiwi	99	Nutrient-rich kiwi, perfect for snacking or adding a tropical twist to your dishes.	100
ObjectId('687e6f836d0c320a6c718dd9')	2	Ice cream	65	Creamy and delicious ice cream, available in various flavors for a delightful treat.	100
ObjectId('687e6f836d0c320a6c718dd4')	3	Juice	100	Refreshing fruit juice, packed with vitamins and great for staying hydrated.	25
ObjectId('687e6f836d0c320a6c718ddb')	4	Honey	156		156

2. GET /products/:id

Fetch details of a single product by its ID.

The screenshot shows the Thunder Client interface. In the top navigation bar, 'THUNDER CLIENT' is selected. The main area displays a request to 'localhost:4500/products/2'. The response status is '200 OK', size is '183 Bytes', and time is '7 ms'. The response body contains JSON data for an ice cream product:

```
1 {
2   "id": "687e6f836d0c320a6c718dd9",
3   "id": 2,
4   "name": "Ice cream",
5   "price": 65,
6   "description": "Creamy and delicious ice cream, available in various flavors for a delightful treat.",
7   "stock": 30
8 }
```

Below the response, the terminal window shows the command to start the node server:

```
> shoppinglobe-backend@1.0.0 start
> nodeon server.js
```

Node.js logs indicate the server is running on port 4500.

3. POST /cart

Add a product to the shopping cart.

Before authentication:

The screenshot shows the Thunder Client interface. A POST request is made to 'localhost:4500/cart'. The response status is '401 Unauthorized', size is '27 Bytes', and time is '6 ms'. The response body is a JSON object with a single key 'message':

```
1 {
2   "message": "Invalid token"
3 }
```

Below the response, the terminal window shows the command to start the node server:

```
> shoppinglobe-backend@1.0.0 start
> nodeon server.js
```

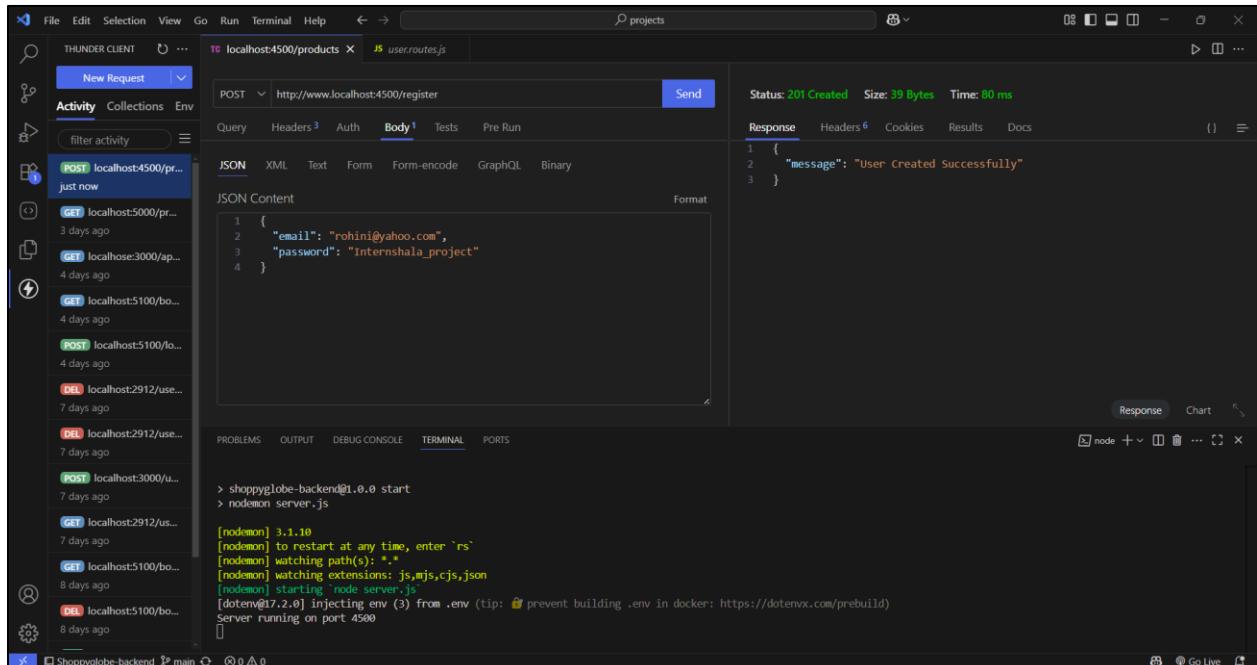
Node.js logs indicate the server is running on port 4500.

Users have to create their account and authenticate before accessing the cart.

● User registration

POST /register

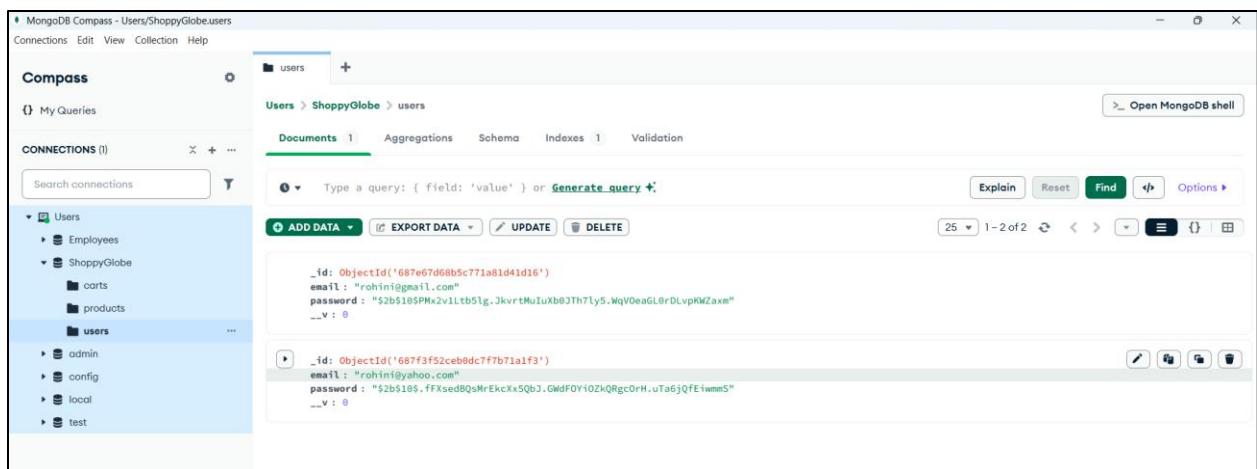
Creates the user in the Mongo DB



```
POST http://www.localhost:4500/register
{
  "email": "rohini@yahoo.com",
  "password": "internshala_project"
}
```

Response:

```
{
  "message": "User Created Successfully"
}
```

```
_id: ObjectId('687e67d68b5c771a81d41d16')
email: "rohini@gmail.com"
password: "$2b$10$PHx2v1tb5lg.JkvrMuIuKb0JTh7ly5.WqVoeGL0rDLvpKwZaxm"
__v: 0

_id: ObjectId('687f3f52ceb0dc7f7b71a1f3')
email: "rohini@yahoo.com"
password: "$2b$10$ffXsedBQsMrEkcXx5QbJ.GMdFOYiOZkQRgcOrH.uTa6jQfEiwmS"
__v: 0
```

● User login

POST /login

Generates the JWT token which can be used for login and access the cart

THUNDER CLIENT

localhost:4500/products user.routes.js

Activity Collections Env

POST localhost:4500/pr... just now

GET localhost:5000/pr... 3 days ago

GET localhost:3000/ap... 4 days ago

GET localhost:5100/b... 4 days ago

POST localhost:5100/lo... 4 days ago

DEL localhost:2912/us... 7 days ago

DEL localhost:2912/us... 7 days ago

POST localhost:3000/u... 7 days ago

GET localhost:2912/us... 7 days ago

GET localhost:5100/b... 8 days ago

DEL localhost:5100/b... 8 days ago

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Response Headers Cookies Results Docs

Status: 200 OK Size: 191 Bytes Time: 63 ms

```

1 {
2   "JWTToken": "eyJhbGciOiJTUzIiNiIsInR5cCI6IkpXVC99
3     .eyJpczvSMQj0iI20dW2YMMmlyjkVzdNUZIMMExZjMlCjpxQlojE3NTMxkj5M
4     jMsImV4cCI6Ic1M2E3MTcyM30.SHSlyWZ9U_tSECH-RvJihZ0Mark0HDLoesje
5     -wCc"
6   }

```

node + ... × Go Live

After Authentication:

Added the JWT token in Headers. Now, the users will be able to add the products to cart.

THUNDER CLIENT

localhost:4500/products user.routes.js

Activity Collections Env

POST localhost:4500/pr... just now

GET localhost:5000/pr... 3 days ago

GET localhost:3000/ap... 4 days ago

GET localhost:5100/b... 4 days ago

POST localhost:5100/lo... 4 days ago

DEL localhost:2912/us... 7 days ago

DEL localhost:2912/us... 7 days ago

POST localhost:3000/u... 7 days ago

GET localhost:2912/us... 7 days ago

GET localhost:5100/b... 8 days ago

DEL localhost:5100/b... 8 days ago

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Response Headers Cookies Results Docs

Status: 200 OK Size: 148 Bytes Time: 14 ms

```

1 {
2   "user": "687f3ff52ceb0dc7f7b71a1f3",
3   "items": [
4     {
5       "productID": 3,
6       "quantity": 4,
7       "_id": "687f40aeceb0dc7f7b71a1f9"
8     }
9   ],
10  "_id": "687f40aeceb0dc7f7b71a1f8",
11  "__v": 0
12 }

```

node + ... × Go Live

POST http://www.localhost:4500/cart

Headers

HTTP Headers Raw

<input checked="" type="checkbox"/> Accept	*/*
<input checked="" type="checkbox"/> User-Agent	Thunder Client (https://www.thunderclient.com)
<input checked="" type="checkbox"/> authorization	JWT eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1...
<input type="checkbox"/> header	value

MongoDB Compass - Users/ShoppGlobe.carts

Connections Edit View Collection Help

Compass

My Queries

CONNECTIONS (1)

- Search connections
- Users
 - Employees
 - ShopGlobe
 - carts
 - products
 - users
 - admin
 - config
 - local
 - test

Users > ShopGlobe > carts

Documents 2 Aggregations Schema Indexes 1 Validation

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

ADD DATA

25 1 – 2 of 2

```
_id: ObjectId('687e733d0801e0f58563c00f')
user: "687e67d6b5b5c771a81d41d16"
items: Array (1)
  0: Object
    productID: 3
    quantity: 4
    _id: ObjectId('687f40aeceb0dc7f7b71a1f9')
__v: 8
```

```
_id: ObjectId('687f40aeceb0dc7f7b71a1f8')
user: "687f3f52ceb0dc7f7b71a1f3"
items: Array (1)
  0: Object
    productID: 3
    quantity: 4
    _id: ObjectId('687f40aeceb0dc7f7b71a1f9')
__v: 8
```

4. PUT /cart/:id

Update the quantity of a product in the cart.

PUT http://www.localhost:4500/cart/3

Headers

HTTP Headers Raw

<input checked="" type="checkbox"/> Accept	*/*
<input checked="" type="checkbox"/> User-Agent	Thunder Client (https://www.thunderclient.com)
<input checked="" type="checkbox"/> authorization	JWT eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1...

Thunder Client Screenshot:

PUT http://www.localhost:4500/cart/3

Response:

```

1  {
2    "_id": "687f40aeceb0dc7fb71a1f8",
3    "user": "687f3f5ceb0dc7fb71a1f3",
4    "items": [
5      {
6        "productID": 3,
7        "quantity": 7,
8        "_id": "687f40aeceb0dc7fb71a1f9"
9      }
10    ],
11    "__v": 0
12  }

```

MongoDB Compass Screenshot:

Users > ShoppyGlobe > carts

Documents (2)

```

_id: ObjectId('687e73d3d0881ef58563c00f')
user: '687f3f5ceb0dc7fb71a1f6'
items: Array (1)
  0: Object
    productID: 3
    quantity: 7
    _id: ObjectId('687f40aeceb0dc7fb71a1f9')
__v: 8

_id: ObjectId('687f40aeceb0dc7fb71a1f8')
user: '687f3f5ceb0dc7fb71a1f3'
items: Array (1)
  0: Object
    productID: 3
    quantity: 7
    _id: ObjectId('687f40aeceb0dc7fb71a1f9')
__v: 8

```

5. DELETE /cart/:id

Remove a product from the cart.

DELETE http://www.localhost:4500/cart/3

Headers:

- Accept: */*
- User-Agent: Thunder Client (https://www.thunderclient.com)
- authorization: JWT eyJhbGciOiJIUzI1NiIsInR5cI6IkpxVCJ9.eyJ1...

The screenshot shows the Thunder Client interface. On the left, there's a sidebar with activity logs and environment variables. The main area has a request builder for a DELETE operation on the URL `http://www.localhost:4500/cart/3`. The response tab shows a status of 200 OK, size of 87 bytes, and time of 8 ms. The response body is an empty JSON object:

```

1  {
2    "_id": "687f40aeceb0dc7f7b71a1f8",
3    "user": "687f3f52ceb0dc7f7b71a1f3",
4    "items": [],
5    "__v": 1
6  }

```

Below the response, there's a terminal window showing the command `> shoppyglobe-backend@1.0.0 start` and the output of the node server starting on port 4500.

Deleted the item. So, now the cart is empty.

The screenshot shows the MongoDB Compass interface. The left sidebar shows connections and the current database is `ShoppyGlobe.carts`. The main area shows the `carts` collection with two documents listed under the `Documents` tab. Both documents have been deleted, as indicated by the empty JSON objects shown in the preview pane:

```

_id: ObjectId('687e733d0801e0fs8563c00f')
user : "687f676db5b5c771a81d41d16"
  items: Array (1)
    __v : 8

_id: ObjectId('687f40aeceb0dc7f7b71a1f8')
user : "687f3f52ceb0dc7f7b71a1f3"
  items: Array (empty)
    __v : 1

```

Screenshots from MongoDB:

Database Name: ShoppyGlobe

It has 3 collections:

- Carts
- Products
- Users

The screenshot shows the MongoDB Compass interface for the ShoppGlobe database. On the left, the connection tree shows 'Users' and 'ShoppGlobe' expanded, with 'carts', 'products', and 'users' under 'ShoppGlobe'. The main pane displays three collections: 'carts', 'products', and 'users'. Each collection summary includes storage size, document count, average document size, index count, and total index size.

Collection	Storage size	Documents	Avg. document size	Indexes	Total index size
carts	20.48 kB	2	105.00 B	1	36.86 kB
products	20.48 kB	5	168.00 B	2	73.73 kB
users	20.48 kB	2	134.00 B	1	36.86 kB

Carts Collection:

The screenshot shows the MongoDB Compass interface for the 'carts' collection. The left sidebar shows 'ShoppGlobe' selected. The main pane shows the 'Documents' tab with 2 results. Each document is displayed with its _id, user, and items fields.

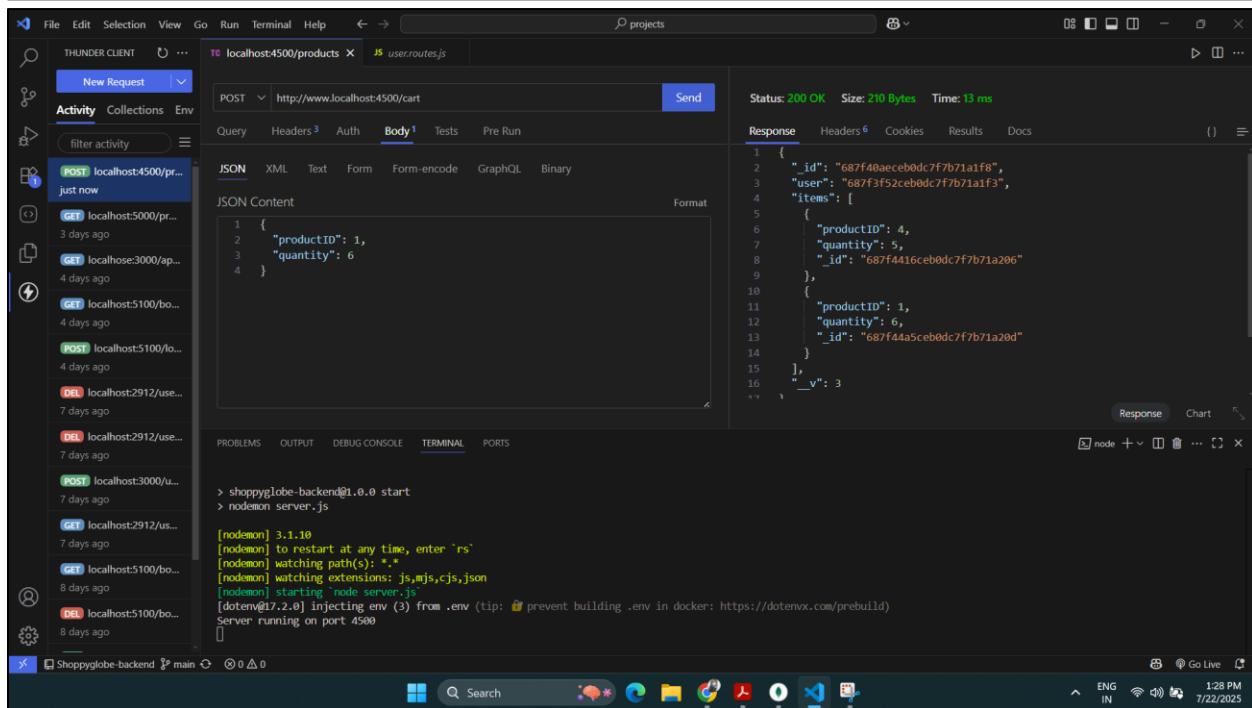
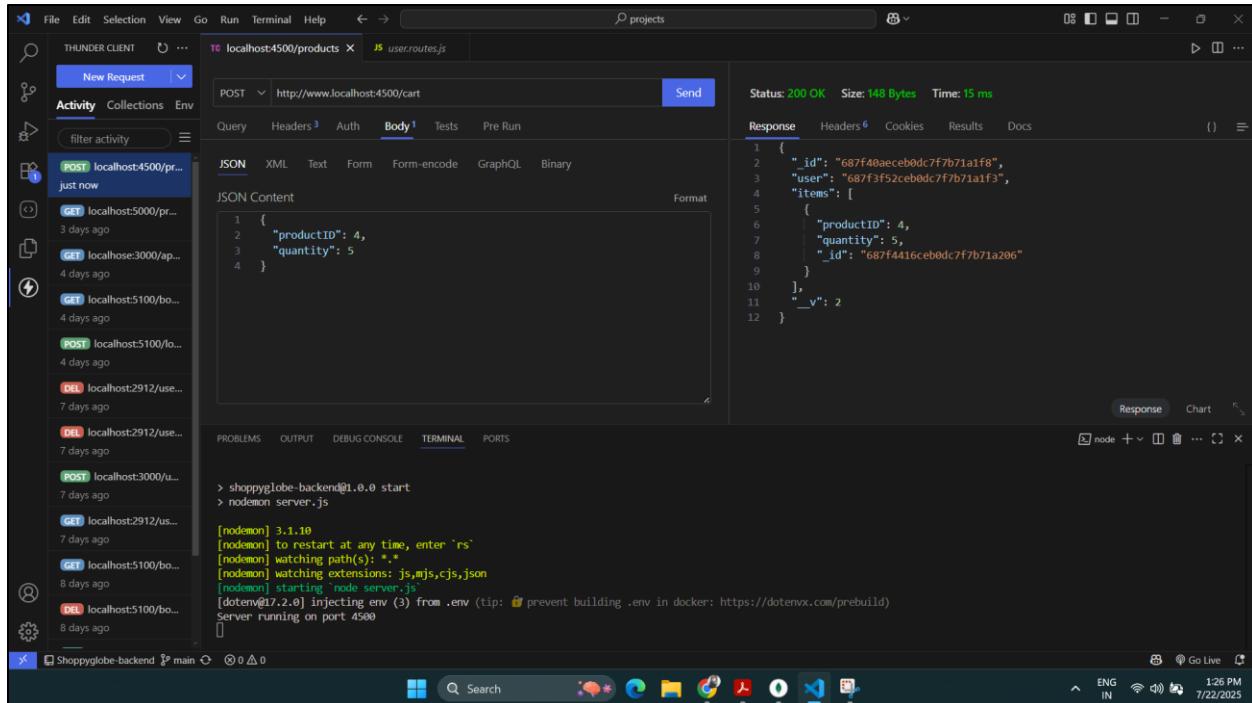
_id	user	items
<code>_id: ObjectId('687e733d0801e0f58563c00f')</code>	<code>"687e67d6db5c771a81d41d16"</code>	<code>items: Array (1)</code> <code>__v: 8</code>
<code>_id: ObjectId('687f40aeceb0dc7f7b71a1f8')</code>	<code>"687f3f52ceb0dc7f7b71a1f3"</code>	<code>items: Array (empty)</code> <code>__v: 1</code>

CRUD operation on Cart:

1. Create document

POST /cart

Add the items to the cart



2. Read document

GET /cart

Fetches all the items from the cart

```

1 {
2   "_id": "687f40aeceb0dc7f7b71a1f8",
3   "user": "687f3f52ceb0dc7f7b71a1f3",
4   "items": [
5     {
6       "productID": 4,
7       "quantity": 5,
8       "_id": "687f4416ceb0dc7f7b71a206"
9     },
10    {
11      "productID": 1,
12      "quantity": 6,
13      "_id": "687f44a5ceb0dc7f7b71a20d"
14    }
15  ],
16  "__v": 3

```

3. Update document

PUT /cart/:id

Edit the quantity of the specific cart item

The screenshot shows a developer environment with two main windows. The top window is a terminal and API client interface. In the terminal, the command `node app.js` is run, followed by `nodemon server.js`, and the message "Server running on port 4500". The API client shows a successful `PUT` request to `http://www.localhost:4500/cart/4` with a status of 200 OK, size of 210 Bytes, and time of 6 ms. The JSON response body is:

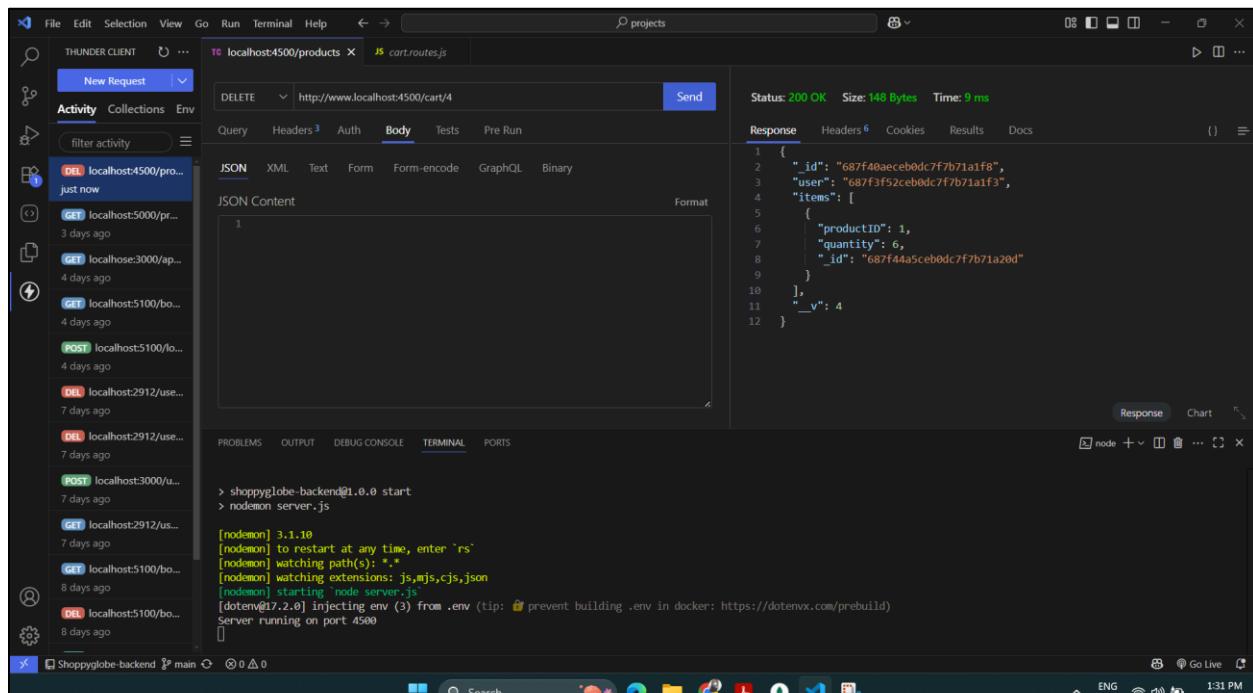
```
1 {
2   "_id": "687f40aeceb0dc7f7b71a1f8",
3   "user": "687f3f52ceb0dc7f7b71a1f3",
4   "items": [
5     {
6       "productID": 4,
7       "quantity": 3,
8       "_id": "687f4416ceb0dc7f7b71a206"
9     },
10    {
11      "productID": 1,
12      "quantity": 6,
13      "_id": "687f44a5ceb0dc7f7b71a20d"
14    }
15  ],
16  "__v": 3
17 }
```

The bottom window is a MongoDB Compass interface showing the `carts` collection. It displays two documents. The first document has an _id of `687e73d0881e0ff58563c00f` and an items array containing one item with productID 4 and quantity 3. The second document has an _id of `687f40aeceb0dc7f7b71a1f8` and an items array containing two items: one with productID 1 and quantity 6, and another with productID 4 and quantity 3.

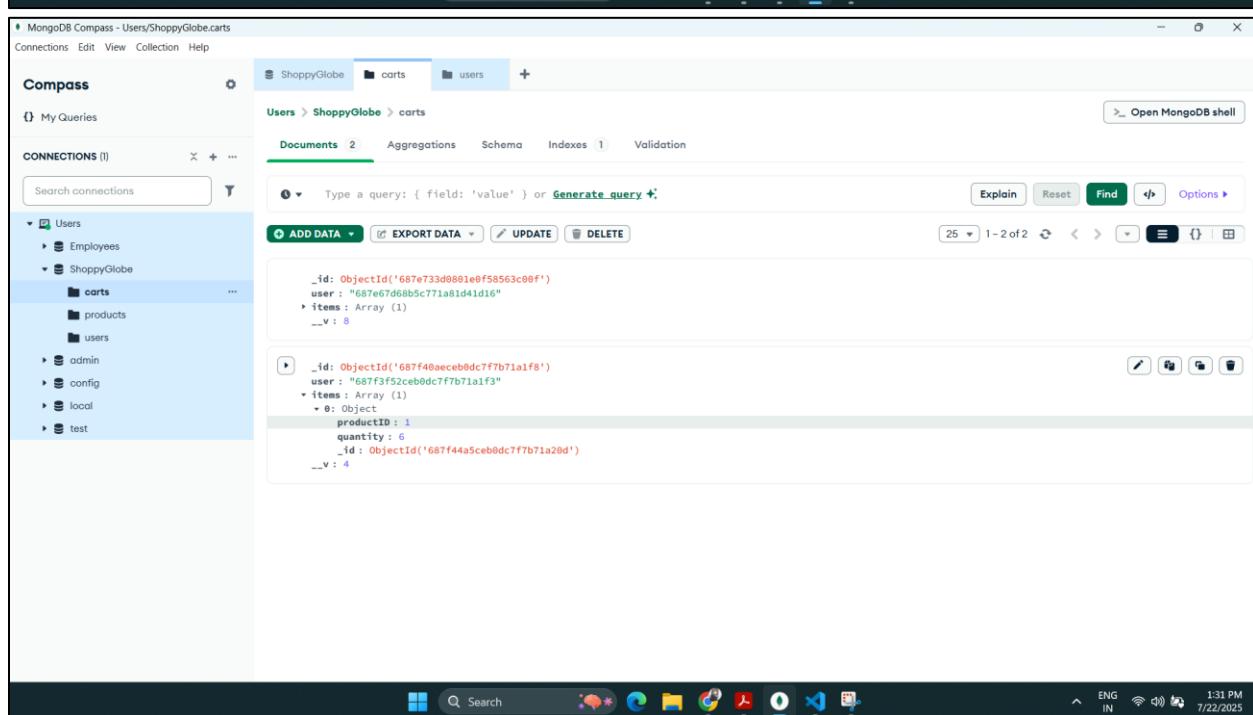
4. Delete document

DELETE /cart/:id

Delete the particular item from the cart.



```
1 {
2   "_id": "687f40aeceb0dc7f7b71a1f8",
3   "user": "687f3f52ceb0dc7f7b71a1f3",
4   "items": [
5     {
6       "productID": 1,
7       "quantity": 6,
8       "_id": "687f44a5ceb0dc7f7b71a20d"
9     }
10   ],
11   "__v": 4
12 }
```



```
_id: ObjectId('687e73d0881e0ff58563c00f')
user : "687e67d6db5c771a81d41d16"
> items: Array (1)
...v : 8

...
_id: ObjectId('687f40aeceb0dc7f7b71a1f8')
user : "687f3f52ceb0dc7f7b71a1f3"
> items: Array (1)
+ 0: Object
  + productID : 1
    quantity : 6
    _id: ObjectId('687f44a5ceb0dc7f7b71a20d')
...v : 4
```

Products Collection:

The screenshot shows the MongoDB Compass interface. On the left, the 'Connections' sidebar lists 'Users', 'Employees', 'ShoppyGlobe' (which contains 'carts', 'products', and 'users'), 'admin', 'config', 'local', and 'test'. The main area displays the 'products' collection under the 'ShoppyGlobe' database. It shows 5 documents with the following data:

- `_id: ObjectId('687e6f836d0c320a6c718dd8')`
id: 1
name: "Kiwi"
price: 99
description: "Nutrient-rich kiwi, perfect for snacking or adding a tropical twist to..."
stock: 100
- `_id: ObjectId('687e6f836d0c320a6c718dd9')`
id: 2
name: "Ice cream"
price: 65
description: "Creamy and delicious ice cream, available in various flavors for a del..."
stock: 30
- `_id: ObjectId('687e6f836d0c320a6c718dda')`
id: 3
name: "Juice"
price: 100
description: "Refreshing fruit juice, packed with vitamins and great for staying hyd..."
stock: 25
- `_id: ObjectId('687e6f836d0c320a6c718ddb')`
id: 4
name: "Honey"
price: 156
description: "Sweet and natural honey, great for cooking or adding a touch of flavor to..."
stock: 20

CRUD operation on Products:

1. Create document

POST /products

Create the new product

The screenshot shows the Thunder Client interface. On the left, the 'Activity' panel lists several recent requests. In the center, a new request is being created for 'localhost:4500/products' using a POST method. The 'Body' tab is selected, showing the JSON content of the request:

```
1 {
2   "id": 6,
3   "name": "Momos",
4   "price": 200,
5   "description": "A tasty and healthy momos",
6   "stock": 70
7 }
```

The response on the right shows a successful creation with status 201 Created, size 129 Bytes, and time 9 ms. The response body is identical to the request body.

MongoDB Compass - Users/ShappyGlobe.products

Connections Edit View Collection Help

Compass

My Queries

CONNECTIONS (1)

Search connections

Users Employees ShappyGlobe carts products users test config local admin

ShappyGlobe > ShappyGlobe > products

Documents 5 Aggregations Schema Indexes 2 Validation

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

[ADD DATA](#) [EXPORT DATA](#) [UPDATE](#) [DELETE](#)

Explain Reset Find Options

25 1–6 of 6

```

price : 100
description : "Refreshing fruit juice, packed with vitamins and great for staying hydrated."
stock : 25

_id: ObjectId('687e6f836d0c320a6c718ddb')
id: 4
name: "Honey"
price: 156
description: "Pure and natural honey in a convenient jar, perfect for sweetening beverages or drizzling over food."
stock: 35

_id: ObjectId('687e6f836d0c320a6c718ddc')
id: 5
name: "Fish"
price: 230
description: "Quality fish steak, suitable for grilling, baking, or pan-searing."
stock: 22

_id: ObjectId('687f4732bbf991f47c24d7f9')
id: 6
name: "Momos"
price: 200
description: "A tasty and healthy momos"
stock: 70
_v: 0

```

ENG IN 1:39 PM 7/22/2025

2. Read document

GET /products

Fetches all the products from the cart

THUNDER CLIENT

New Request

Activity Collections Env

filter activity

4 days ago

GET localhost:5100/b...

4 days ago

POST localhost:5100/lo...

4 days ago

DEL localhost:2912/us...

7 days ago

DEL localhost:2912/us...

7 days ago

POST localhost:3000/u...

7 days ago

GET localhost:2912/us...

7 days ago

GET localhost:5100/b...

8 days ago

DEL localhost:5100/b...

8 days ago

PUT localhost:5100/b...

8 days ago

POST thunderclient.co...

8 days ago

localhost:4500/projects

localhost:4500/products

product.routes.js

GET http://www.localhost:4500/products

Send

Query Headers Auth Body Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content

Format

Status: 200 OK Size: 1 KB Time: 10 ms

Response Headers Cookies Results Docs

```

23   "description": "Refreshing fruit juice, packed with vitamins and great
for staying hydrated.",
24   "stock": 25
25 },
26 {
27   "_id": "687e6f836d0c320a6c718ddb",
28   "id": 4,
29   "name": "Honey",
30   "price": 156,
31   "description": "Pure and natural honey in a convenient jar, perfect
for sweetening beverages or drizzling over food.",
32   "stock": 35
33 },
34 {
35   "_id": "687e6f836d0c320a6c718ddc",
36   "id": 5,
37   "name": "Fish",
38   "price": 230,
39   "description": "Quality fish steak, suitable for grilling, baking, or
pan-searing.",
40   "stock": 22
41 },
42 {
43   "_id": "687f4732bbf991f47c24d7f9",
44   "id": 6,
45   "name": "Momos",
46   "price": 200,
47   "description": "A tasty and healthy momos",
48   "stock": 70,
49   "_v": 0
50 }

```

Response Chart

ENG IN 1:40 PM 7/22/2025

3. Update document

PUT /products/:id

Edit the details of the product

The screenshot shows the Thunder Client interface. On the left, there's a sidebar with activity logs and a main panel for making requests. The current request is a PUT to `http://www.localhost:4500/products/1`. The body contains the following JSON:

```
1 {
2   "name": "Appple",
3   "price": 150,
4   "stock": 60
5 }
```

The response shows a status of 200 OK, size of 180 Bytes, and time of 9 ms. The response body is identical to the request body.

The screenshot shows the MongoDB Compass interface. The left sidebar shows connections and collections. The current collection is `products`. The documents list shows four documents, each representing a product. The first document is highlighted and its details are shown in the main pane:

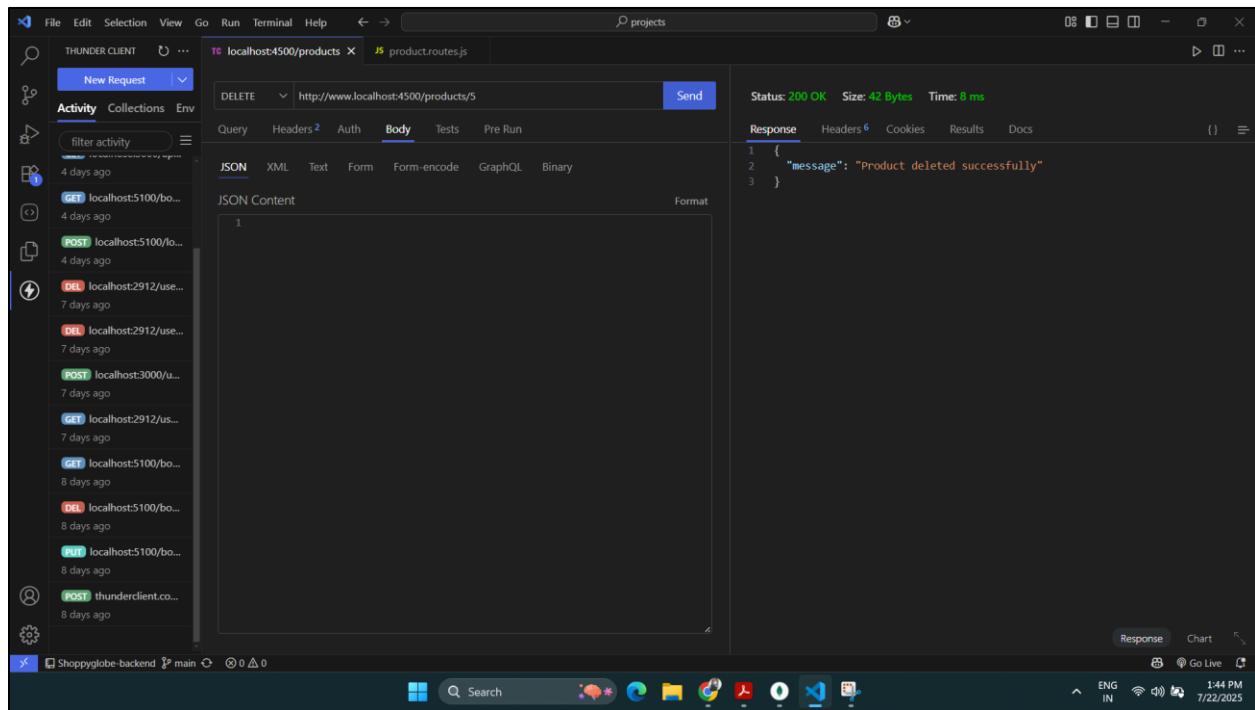
```
_id: ObjectId('687e6f836d0c320a6c718dd8')
id: 1
name: "Appple"
price: 150
description: "Nutrient-rich kiwi, perfect for snacking or adding a tropical twist to your dishes."
stock: 60
```

The other three documents are partially visible below it.

4. Delete document

DELETE /products/:id

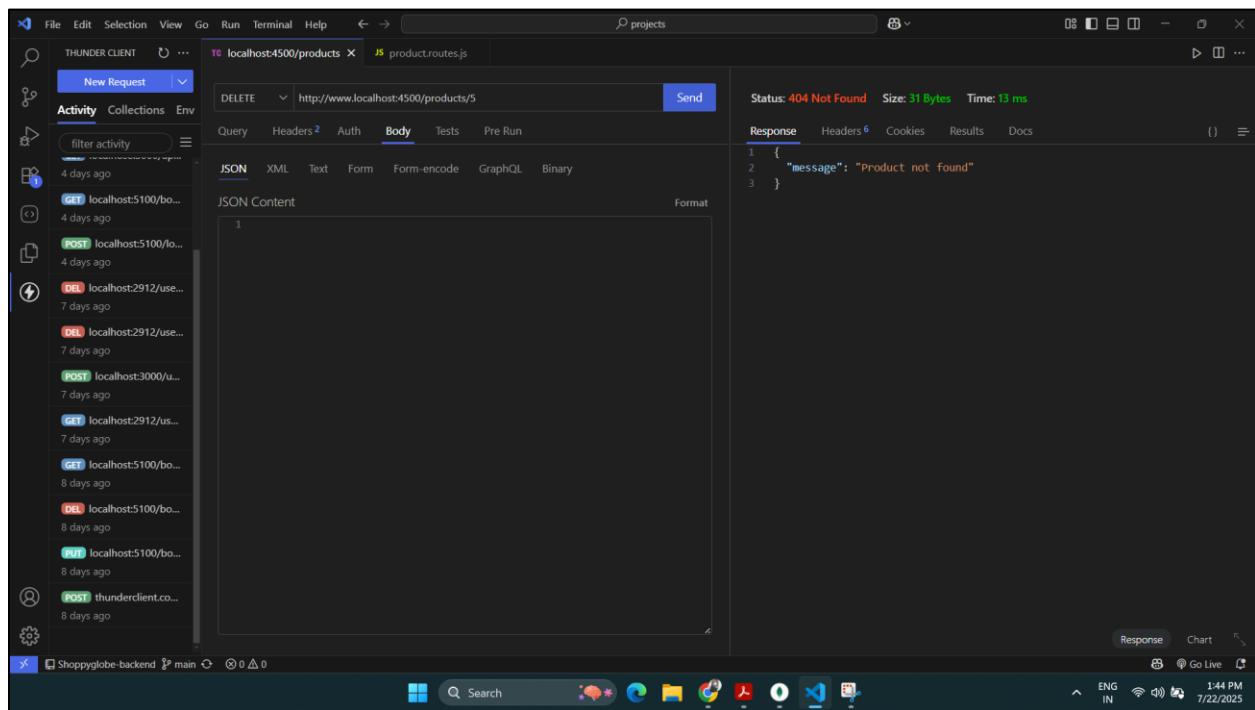
Deletes the product from DB



The screenshot shows the Thunder Client interface. On the left, there's a sidebar with activity logs. In the center, a request is being made to `http://www.localhost:4500/products/5`. The method is set to `DELETE`. The response status is `200 OK`, size is `42 Bytes`, and time is `8 ms`. The response body contains the JSON object:

```
1 {  
2   "message": "Product deleted successfully"  
3 }
```

If trying to delete the product that is not available in DB, it returns error.



The screenshot shows the Thunder Client interface. On the left, there's a sidebar with activity logs. In the center, a request is being made to `http://www.localhost:4500/products/5`. The method is set to `DELETE`. The response status is `404 Not Found`, size is `31 Bytes`, and time is `13 ms`. The response body contains the JSON object:

```
1 {  
2   "message": "Product not found"  
3 }
```

MongoDB Compass - Users/ShoppGlobe.products

Connections Edit View Collection Help

Compass

My Queries

CONNECTIONS (1)

Search connections

- Users
 - Employees
 - ShopGlobe
 - carts
 - products
 - users
 - ...
- admin
- config
- local
- test

ShopGlobe carts products users +

Users > ShopGlobe > products

Documents 5 Aggregations Schema Indexes 2 Validation

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

[ADD DATA](#) [EXPORT DATA](#) [UPDATE](#) [DELETE](#)

Explain Reset Find Options

25 1-4 of 4

```

price: 150
description: "Nutrient-rich kiwi, perfect for snacking or adding a tropical twist to..."
stock: 60

_id: ObjectId('687e6f836d0c320a6c718dda')
id: 3
name: "Juice"
price: 100
description: "Refreshing fruit juice, packed with vitamins and great for staying hyd..."
stock: 25

_id: ObjectId('687e6f836d0c320a6c718ddb')
id: 4
name: "Honey"
price: 156
description: "Pure and natural honey in a convenient jar, perfect for sweetening bev..."
stock: 35

_id: ObjectId('687f4732bbf991f47c24d7f9')
id: 6
name: "Momos"
price: 200
description: "A tasty and healthy momos"
stock: 70
__v: 0

```

ENG IN 1:45 PM 7/22/2025

Users Collection:

MongoDB Compass - Users/ShoppGlobe.users

Connections Edit View Collection Help

Compass

My Queries

CONNECTIONS (1)

Search connections

- Users
 - Employees
 - ShopGlobe
 - carts
 - products
 - users
 - ...
- admin
- config
- local
- test

ShopGlobe carts users +

Users > ShopGlobe > users

Documents 2 Aggregations Schema Indexes 1 Validation

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

[ADD DATA](#) [EXPORT DATA](#) [UPDATE](#) [DELETE](#)

Explain Reset Find Options

25 1-2 of 2

```

_id: ObjectId('687e67d68b5c771a81d41d16')
email: "rohini@gmail.com"
password: "$2b$10$PMx2v1ltb5lg.JkvrtMuIuXb0JTh7ly5.WqVOeaGL8rDLvpKwZaxm"
__v: 0

_id: ObjectId('687f3f52ceb0dc7ff7b71a1f3')
email: "rohini@yahoo.com"
password: "$2b$10$.ffXsedBQsMrEkcx5QbJ.GWdFOYiOZkQRgcOrH.uTa6jQfEiwmnS"
__v: 0

```

Cart Validation:

The API call checks if the product is available before adding it to cart. If not present, then it shows an error message.

The screenshot shows the Postman application interface. In the left sidebar, there's a project named "Shoppyglobe-backend" with various files listed under "PROJECTS". The main area shows a POST request to "http://www.localhost:4500/cart". The "Body" tab is selected, displaying the following JSON content:

```
1 {  
2   "productID": 5,  
3   "quantity": 2  
4 }
```

The response panel indicates a "Status: 404 Not Found" with a "Size: 31 Bytes" and a "Time: 7 ms". The response body is shown as:

```
1 {  
2   "message": "Product not found"  
3 }
```

The product Id - 5 is not present in the DB:

The screenshot shows the MongoDB Compass interface. On the left, the "Connections" sidebar lists databases like "Users", "Employees", "ShoppyGlobe", "carts", and "products". The "products" database is selected. The main pane displays the "products" collection with 5 documents. The documents are:

- Document 1:

```
price : 150
description : "Nutrient-rich kiwi, perfect for snacking or adding a tropical twist to..."
stock : 60
```
- Document 2:

```
_id: ObjectId('687e6f836d0c320a6c718dda')
id : 3
name : "Juice"
price : 100
description : "Refreshing fruit juice, packed with vitamins and great for staying hyd..."
stock : 25
```
- Document 3:

```
_id: ObjectId('687e6f836d0c320a6c718ddb')
id : 4
name : "Honey"
price : 150
description : "Pure and natural honey in a convenient jar, perfect for sweetening bev..."
stock : 35
```
- Document 4:

```
_id: ObjectId('687f4732bbff991f47c24d7f9')
id : 5
name : "Momos"
price : 200
description : "A tasty and healthy momos"
stock : 70
```