ShoppyGlobe Backend API Testing Documentation

This document provides step-by-step instructions and expected results for testing all required API routes using ThunderClient . Screenshots of the requests, responses, and MongoDB structure must be added below the relevant sections for submission compliance.
GitHub Repository Link: https://github.com/Chakriroy11/shoppyglobe
Base API URL: http://localhost:5000/api

I. MongoDB Database Structure

The project uses MongoDB with the Mongoose ODM to store all persistent application data.

1. Products Collection Setup

Objective: Show the collection schema structure and initial product data insertion.
Proof: Screenshot showing the fields (**name**, **price**, **description**, **stock**, **thumbnail**, **category**) and at least one document inserted into the **products** collection.

---

2. Cart Collection Structure

Objective: Show how the cart structure links a user to an array of items. The cart document must be user-specific (**user ObjectId reference**) and contain item details (**product ObjectId reference, quantity**).
Proof: Screenshot showing the Cart collection schema with the user reference field and the structure of the items array.

II. Authentication & Authorization Routes

These routes secure the application by implementing JWT-based authentication.

---

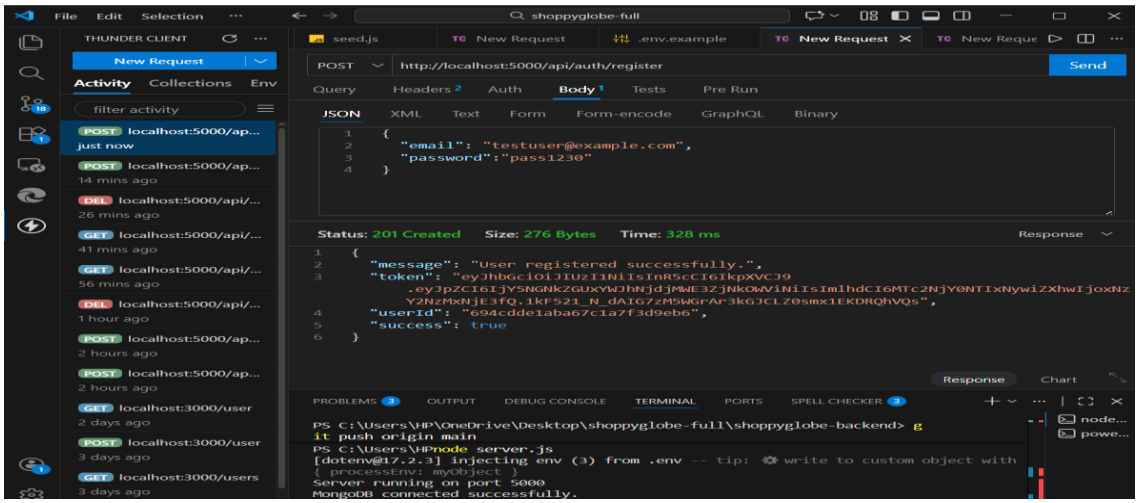
1. User Registration (POST /api/register)

Route: POST /api/register
Objective: Register a new user and receive a JWT token.
Expected Status: 201 Created
Body:

```
{
  "email": "testuser@example.com",
  "password": "password123"
}
```

Expected Response: JSON object containing success: true and the generated **token**

Screenshot:



---

2. User Login (POST /api/login)
Route: POST /api/login
Objective: Authenticate the user and receive a new JWT token.
Expected Status: 200 OK
Body: (Use the same email/password used for registration)

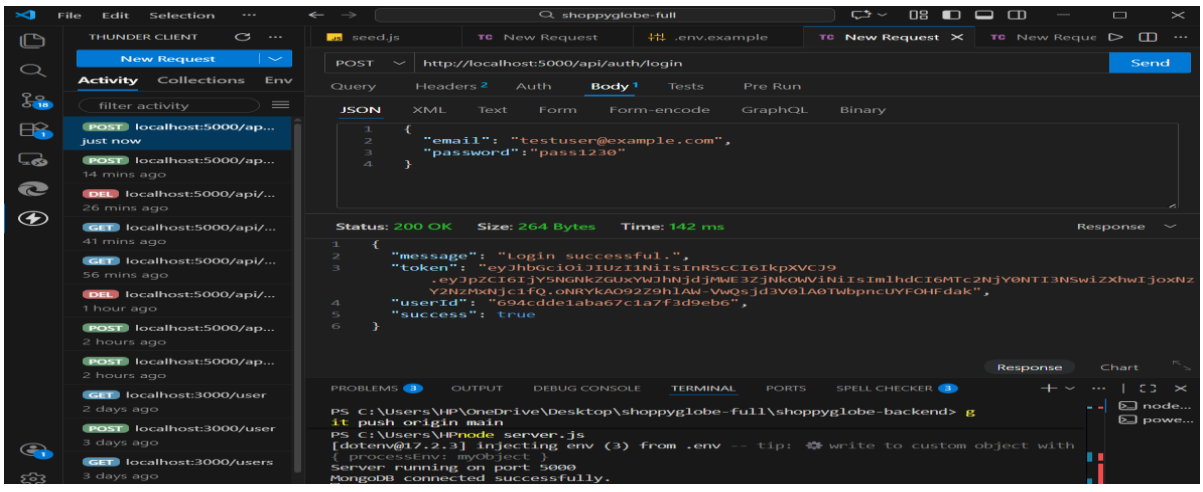
{

"email": "testuser@example.com",

"password": "password123"

}

Expected Response: JSON object containing **success: true** and the **token**. (The token must be saved for all subsequent Cart requests)
Screenshot:



Product Routes (Public Access)

These routes are public and allow any client (including the unauthenticated frontend) to retrieve product information.

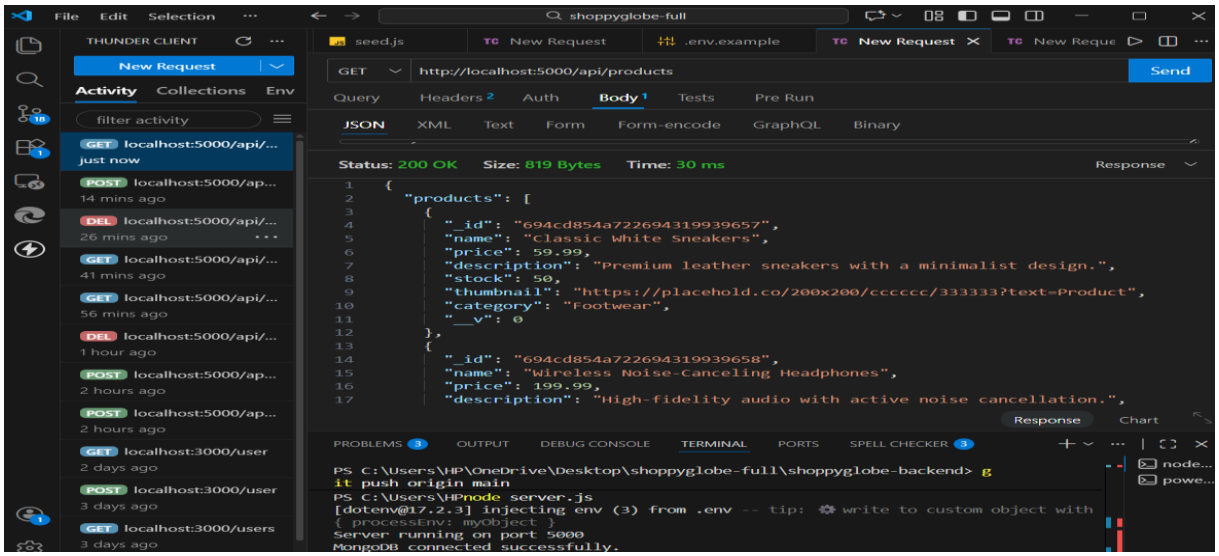
---

1. Fetch All Products (GET /api/products)

Route: GET /api/products
Objective: Retrieve the full list of products stored in MongoDB.
Expected Status: 200 OK
Headers: None required.

Expected Response: JSON object containing the **products array**.

Screenshot:

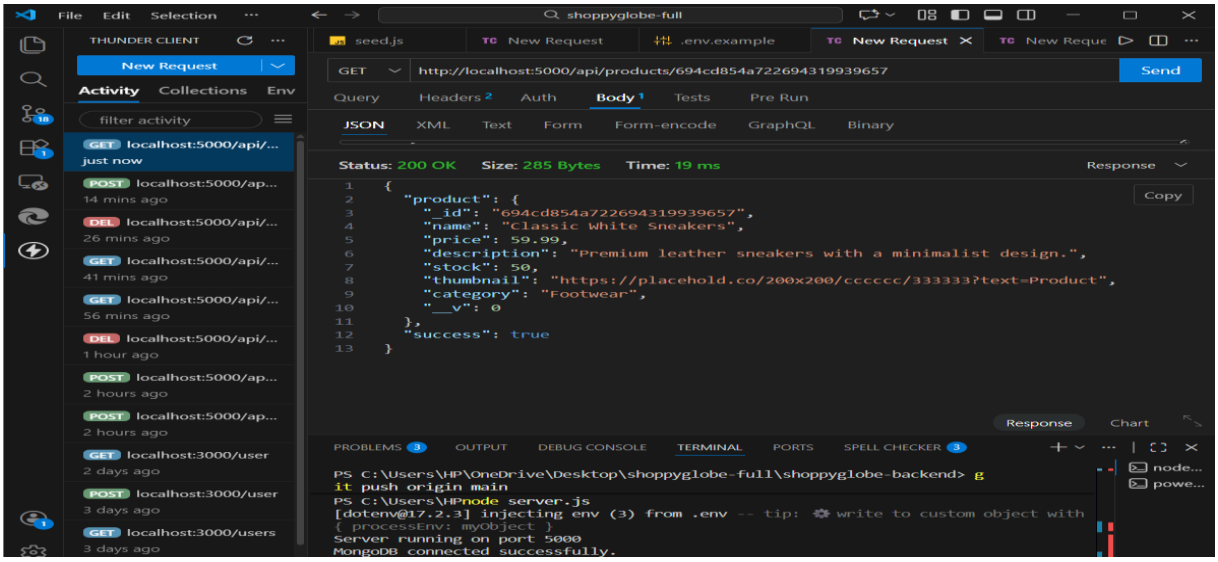


2. Fetch Single Product (GET /api/products/:id)

Route: GET /api/products/692b340f60c51de5ac134e30

Objective: Retrieve details for a single product.
Expected Status: 200 OK
Headers: None required.
Expected Response: JSON object containing the **single product object**.

Screenshot:



Protected Cart Routes (CRUD Operations)

Crucial Requirement: All routes in this section require the JWT token to be included in the request header.

Header Required for ALL Cart Requests:

Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY5MmIzMjZlYjBkYjg0ZTVlZWEyYzE1YSIsImlhdCI6MTc2NDQ0MjM5NSwiZXhwIjoxNzY0NTI4Nzk1fQ.zSLlgVaUlo1pyhSWqPFxdi8x9ROMKoDBuISHaDOqp54

1. Get Cart Items (GET /api/cart)

Route: GET /api/cart

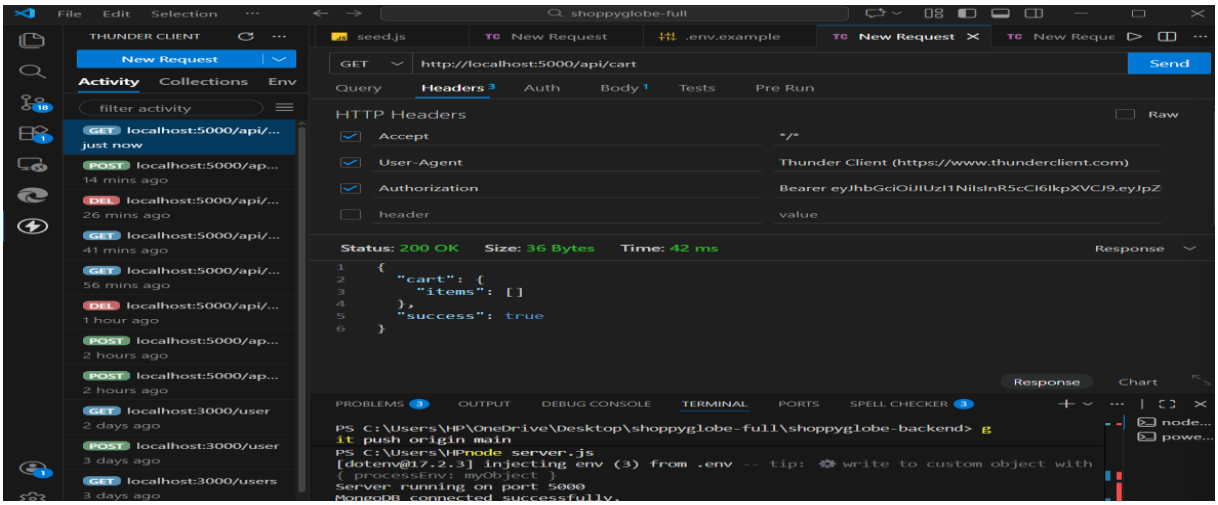
Objective: Retrieve the current cart contents for the authenticated user.

Expected Status: 200 OK

Initial Expected Response:

{"cart": {"items": []}, "success": true} (If the user's cart is empty).

Screenshot:



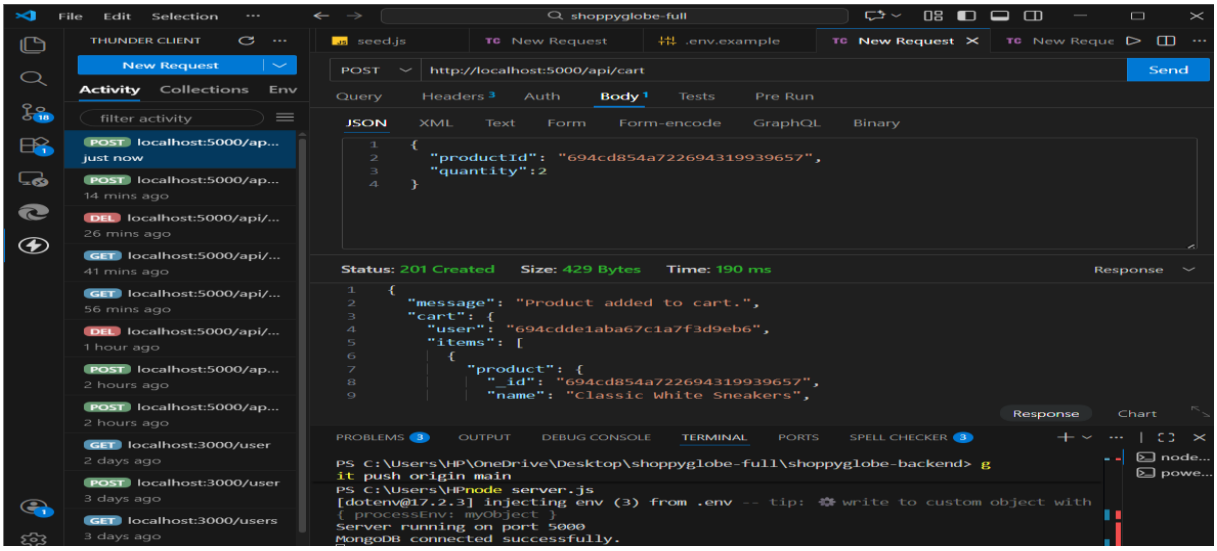
---

2. Add Product to Cart (POST /api/cart)

Route: POST /api/cart
Objective: Add a product (e.g., quantity 2) to the user's cart.
Expected Status: 201 Created
Body:

```
{
  "productId": "692b340f60c51de5ac134e30",
  "quantity": 2
}
```

Expected Response: JSON object containing the updated cart with the new item.
Screenshot:



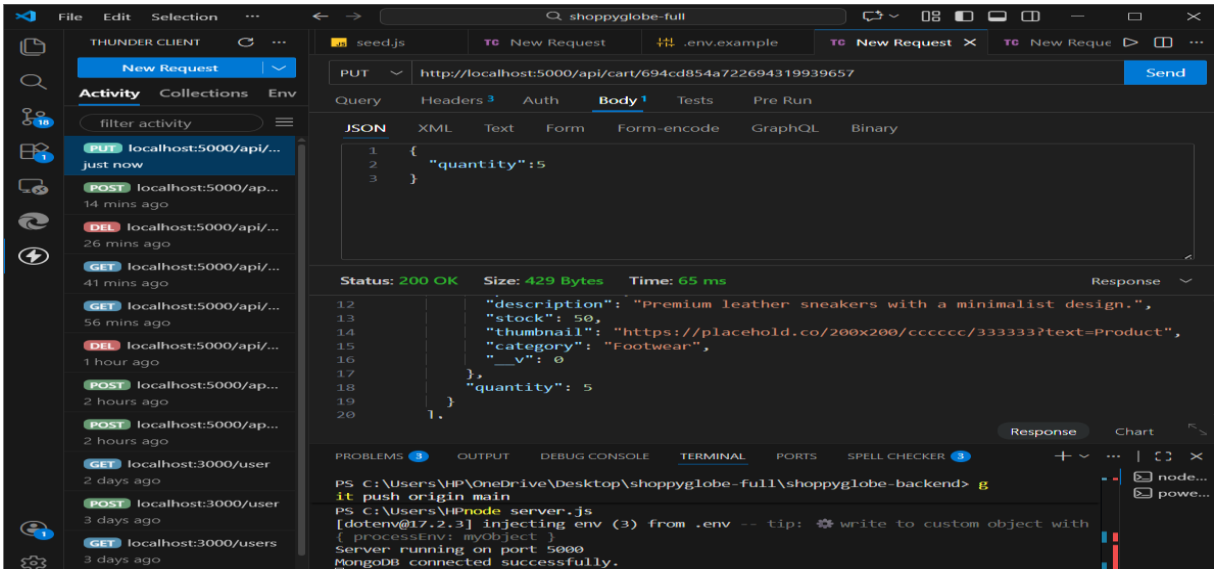
3. Update Cart Quantity (PUT /api/cart/:productId)

Route: PUT /api/cart/692b340f60c51de5ac134e30
Objective: Change the quantity of Product 1 (already in the cart) to 5.
Expected Status: 200 OK
Body:

```
{
  "quantity": 5
}
```

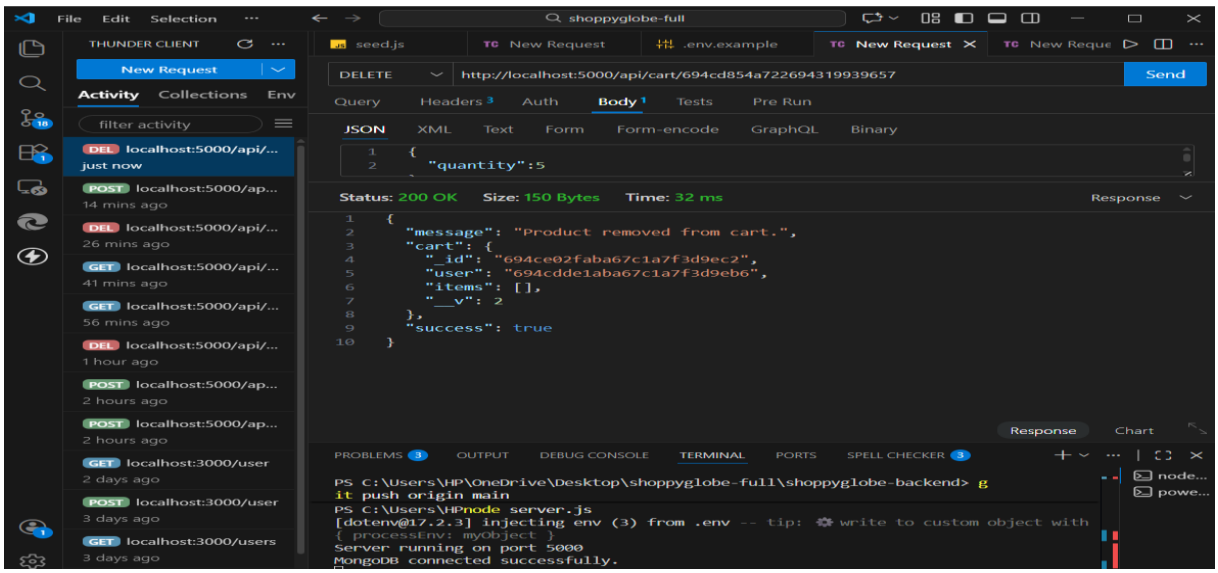
Expected Response: JSON object containing the updated cart showing the new quantity.

Screenshot:



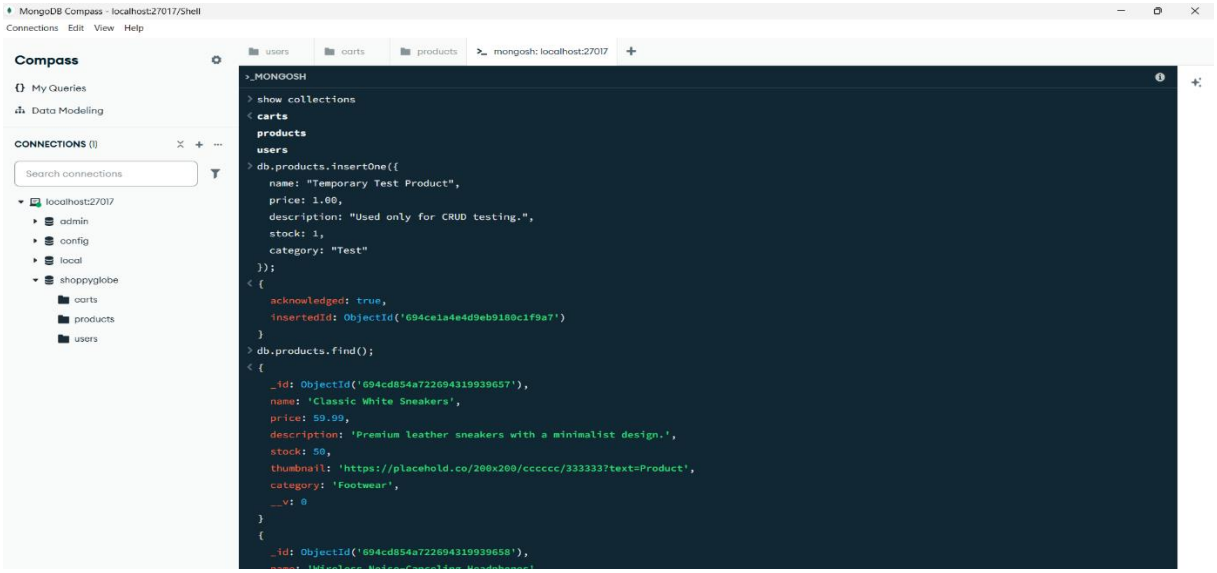
4.Remove Product from Cart (DELETE/api/cart/:productId)

Route: DELETE /api/cart/692b340f60c51de5ac134e30
Objective: Remove Product 1 entirely from the cart.
Expected Status: 200 OK
Body: None.
Expected Response: JSON object containing the updated cart with the item removed.
Screenshot:

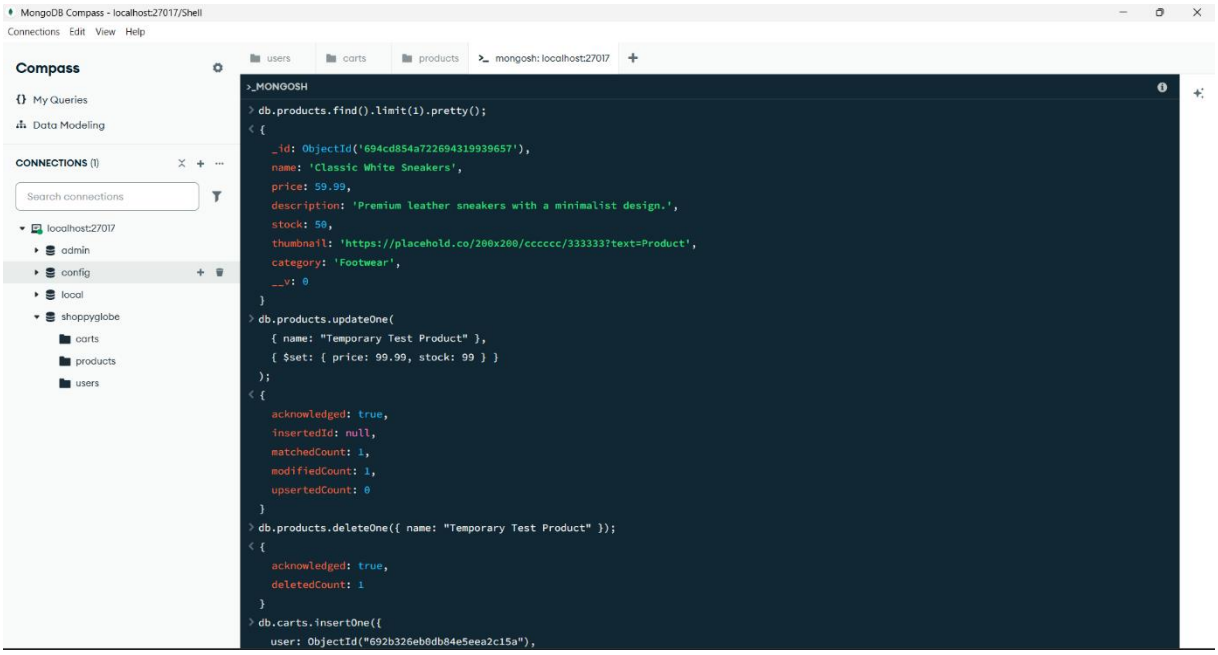


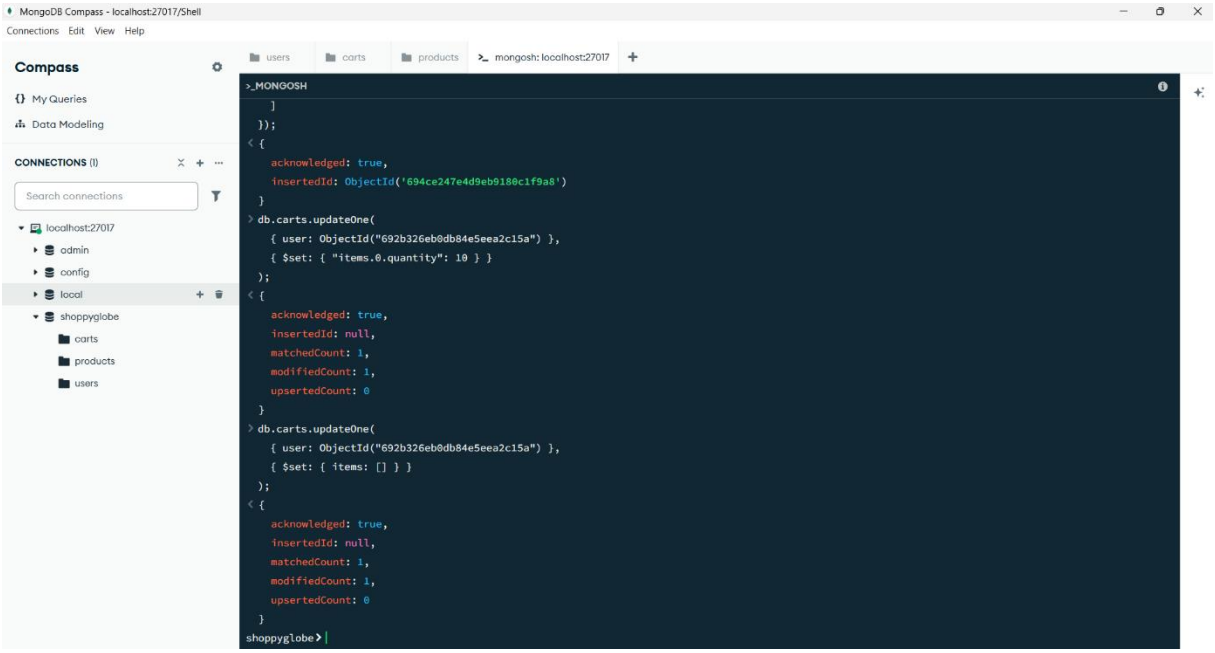
Implement CRUD operations on MongoDB collections

ALL CRUD OPERATION ON THE CART



CRUD OPERATIONS ON THE PRODUCTLIST




MongoDB Compass - localhost:27017/Shell
Connections Edit View Help
Compass
My Queries
Data Modeling
CONNECTIONS (1)
Search connections
localhost:27017
admin
config
local
shoppyglobe
carts
products
users
users
carts
products
mongosh: localhost:27017
>_MONGOSH
]
});
{
acknowledged: true,
insertedId: ObjectId('694ce247e4d9eb9180c1f9a8')
}
db.carts.updateOne(
{ user: ObjectId("692b326eb0db84e5eea2c15a") },
{ $set: { "items.0.quantity": 10 } }
);
{
acknowledged: true,
insertedId: null,
matchedCount: 1,
modifiedCount: 1,
upsertedCount: 0
}
db.carts.updateOne(
{ user: ObjectId("692b326eb0db84e5eea2c15a") },
{ $set: { items: [] } }
);
{
acknowledged: true,
insertedId: null,
matchedCount: 1,
modifiedCount: 1,
upsertedCount: 0
}
shoppyglobe>