ShoppyGlobe E-commerce API

Overview

ShoppyGlobe is a RESTful API designed for managing e-commerce operations such as products, cart, and user authentication. The backend is built using **Node.js**, **Express**, and **MongoDB**.

Features

1. **Product Management**: Add, update, view, and delete products.
2. **Cart Management**: Add items to the cart, update quantities, fetch cart items, and remove items.
3. **Authentication**: JWT-based user registration and login system.
4. **Error Handling**: Comprehensive error responses for invalid inputs or unauthorized actions.

Technologies Used

- **Node.js**: JavaScript runtime for backend development.
- **Express**: Framework for building RESTful APIs.
- **MongoDB**: Database for storing products, cart items, and user details.
- **Mongoose**: ODM for MongoDB.
- **JWT**: For authentication and authorization.

Setup Instructions

1. Clone the repository from GitHub:
2. `git clone https://github.com/PreciousUmang/Shoppy-Backend.git`
3. `cd shoppy backend`
4. Install dependencies:
5. `npm install`
6. Start MongoDB locally or configure a remote MongoDB instance.
7. Create a `.env` file (optional) with the following variables:
8. `JWT_SECRET=Precious!23`
9. `PORT=3000`
10. Start the server:
11. `npm start`
12. Access the API at `http://localhost:3000`.

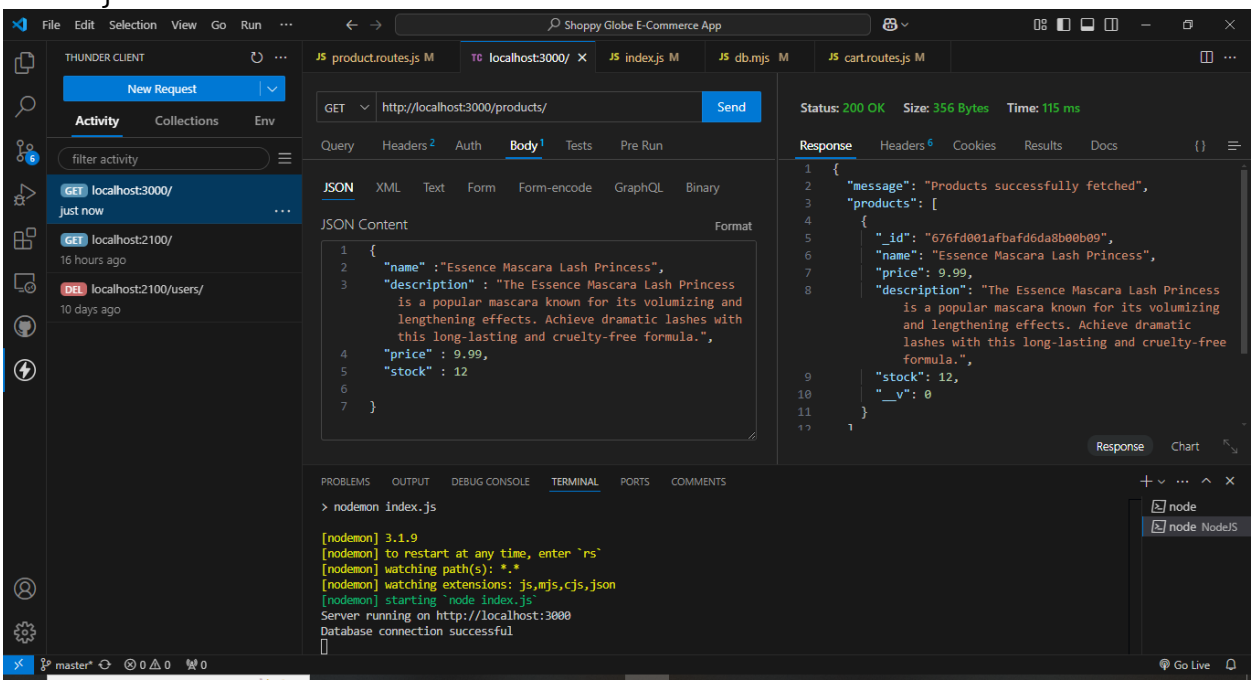
API Endpoints

1. Product Management

GET /products

Fetches all products.

- **Response**:
- {
-   "message": "Products successfully fetched",
-   "products": [
-     {
-       "_id": "<product-id>",
-       "name": "Product Name",
-       "price": 100,
-       "description": "Product Description",
-       "stock": 10
-     }
-   ]
- }

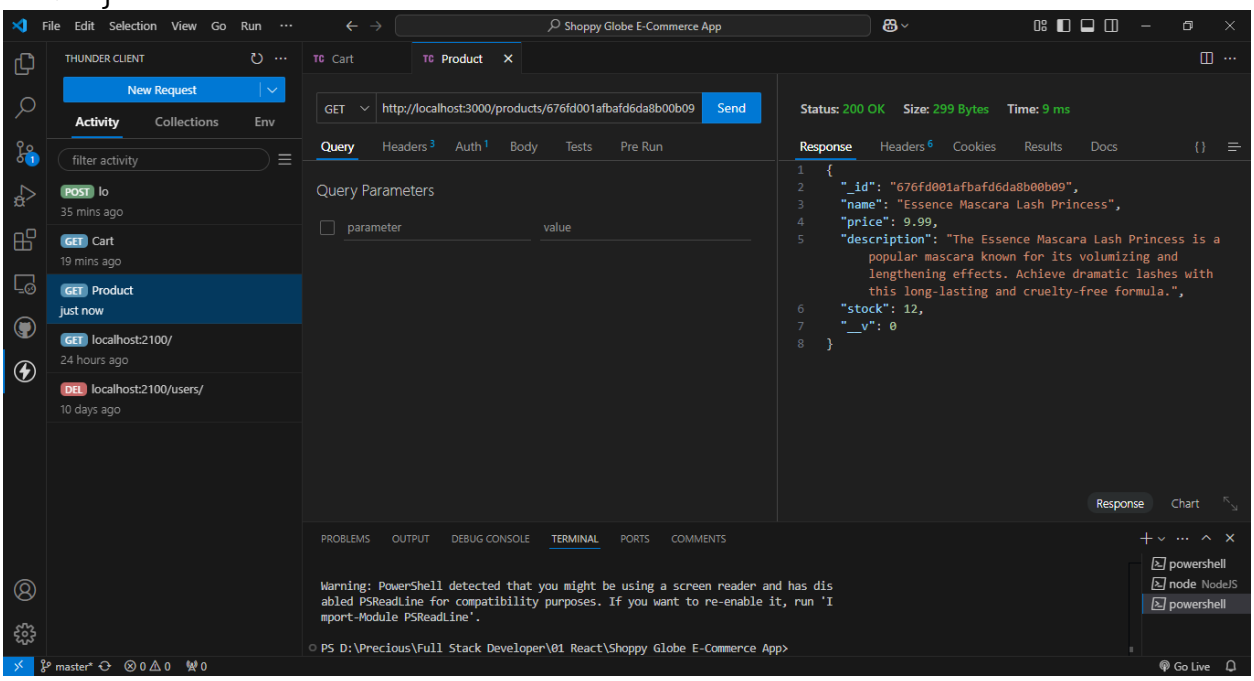


GET /products/:id

Fetches a product by its ID.

- **Parameters**:
  - `id`: MongoDB ObjectId.
- **Response**:
- {

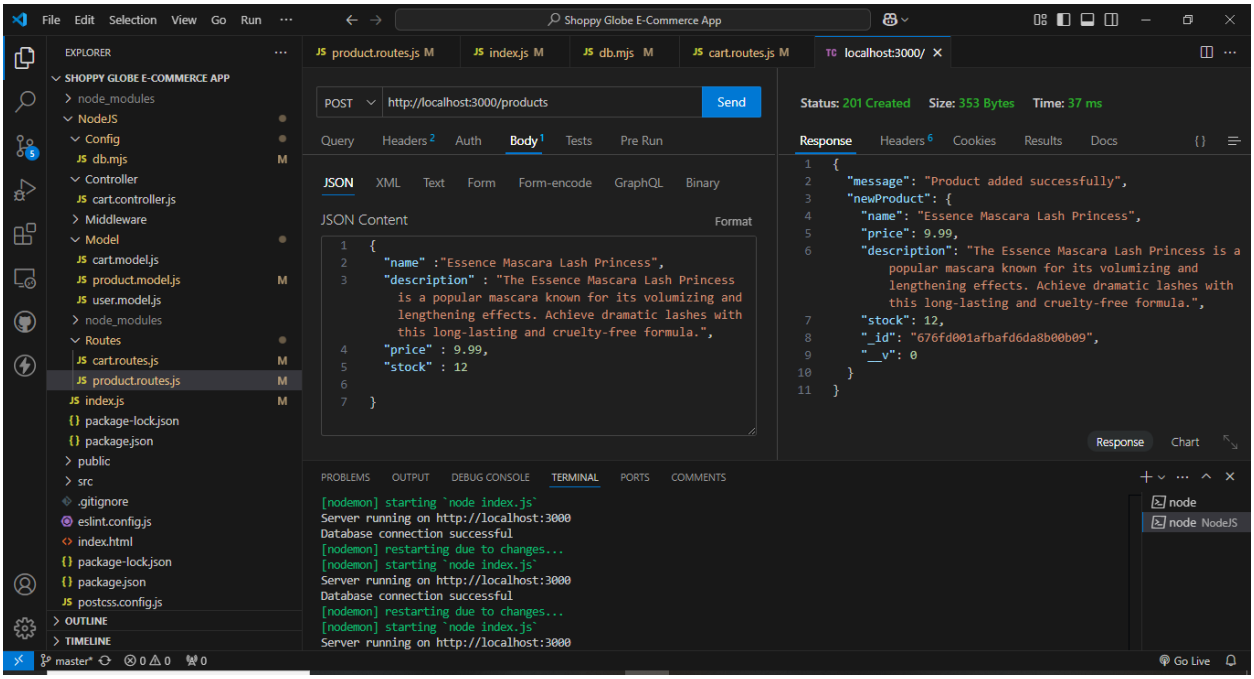
- "_id": "<product-id>",
- "name": "Product Name",
- "price": 100,
- "description": "Product Description",
- "stock": 10
- }



POST /products

Adds a new product.

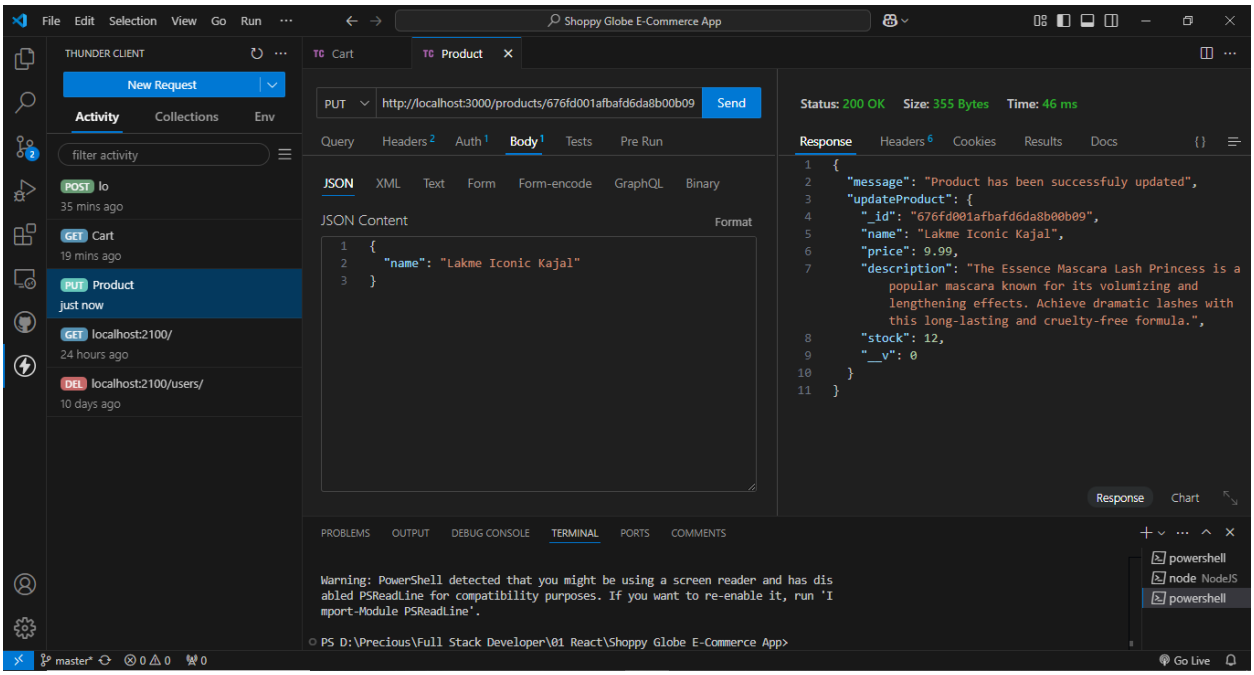
- **Request Body**:
- {
- "name": "Product Name",
- "price": 100,
- "description": "Product Description",
- "stock": 10
- }
- **Response**:
- {
- "message": "Product added successfully",
- "newProduct": {
- "_id": "<product-id>",
- "name": "Product Name",
- "price": 100,
- "description": "Product Description",
- "stock": 10
- }
- }



PUT /products/:id

Updates an existing product.

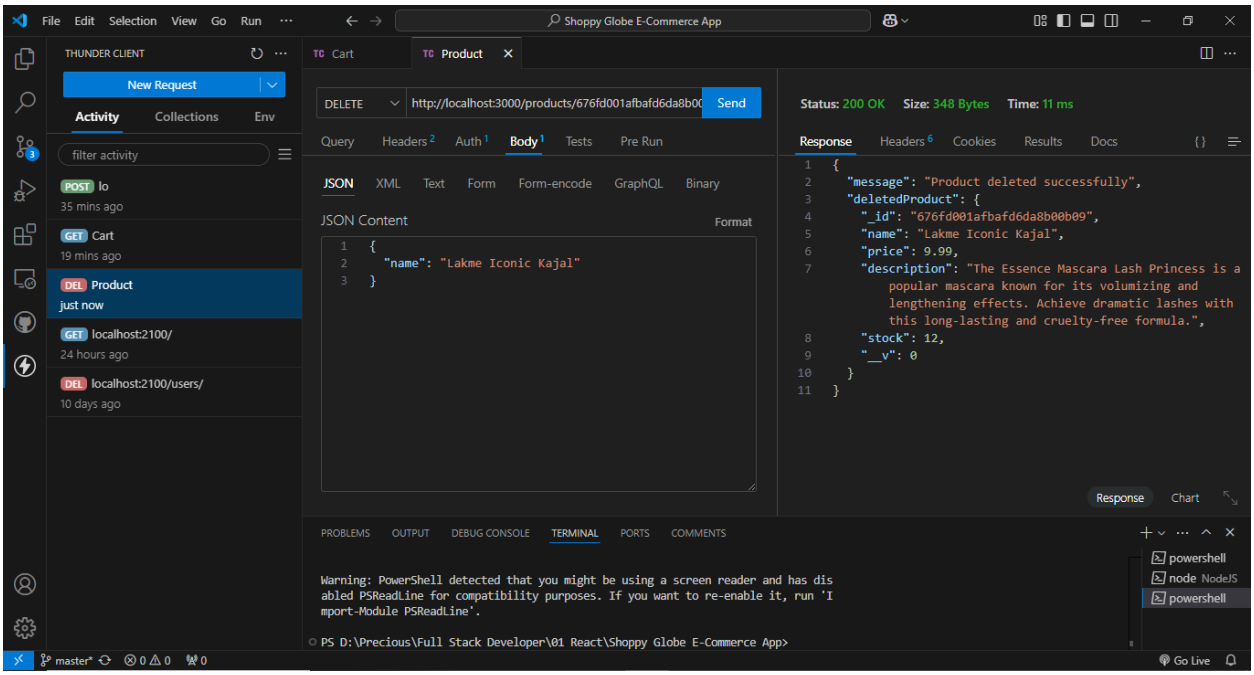
- **Request Body**:
- {
- "name": "Updated Product Name",
- "price": 150,
- "description": "Updated Description",
- "stock": 20
- }
- **Response**:
- {
- "message": "Product has been successfully updated",
- "updateProduct": {
- "_id": "<product-id>",
- "name": "Updated Product Name",
- "price": 150,
- "description": "Updated Description",
- "stock": 20
- }
- }



DELETE /products/:id

Deletes a product by its ID.

- **Response**:
- {
- "message": "Product deleted successfully",
- "deletedProduct": {
- "_id": "<product-id>",
- "name": "Product Name",
- "price": 100,
- "description": "Product Description",
- "stock": 10
- }
- }

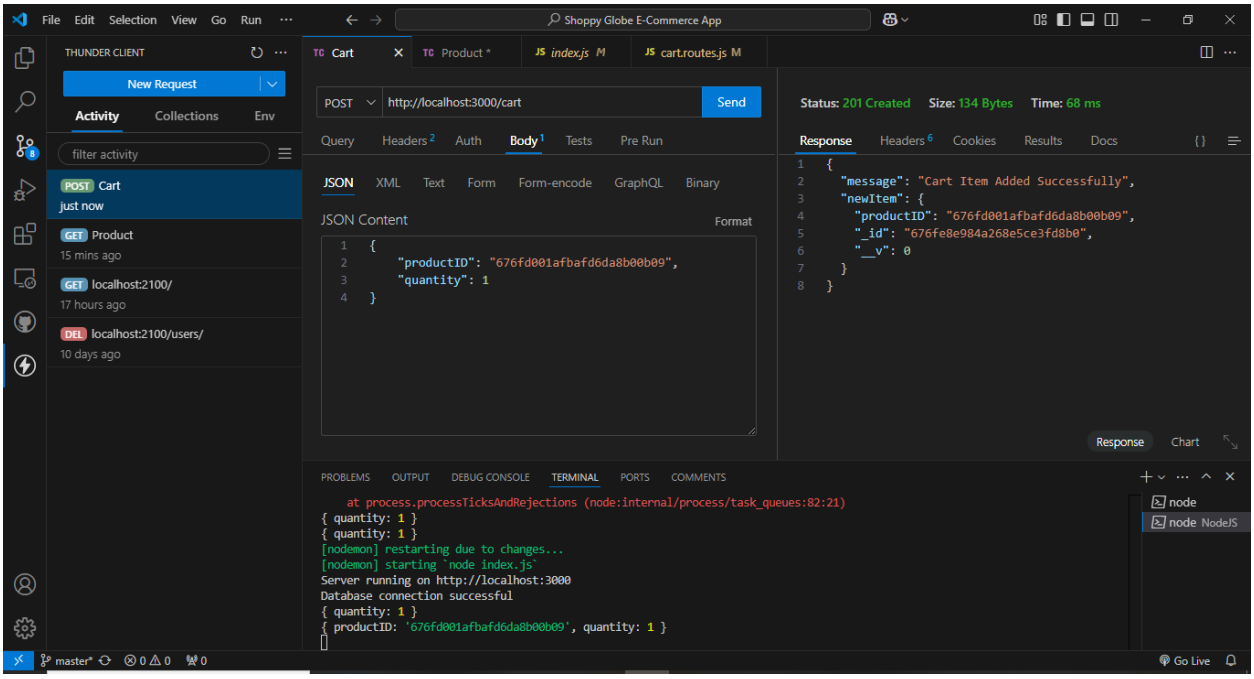


2. Cart Management

POST /cart

Adds an item to the cart.

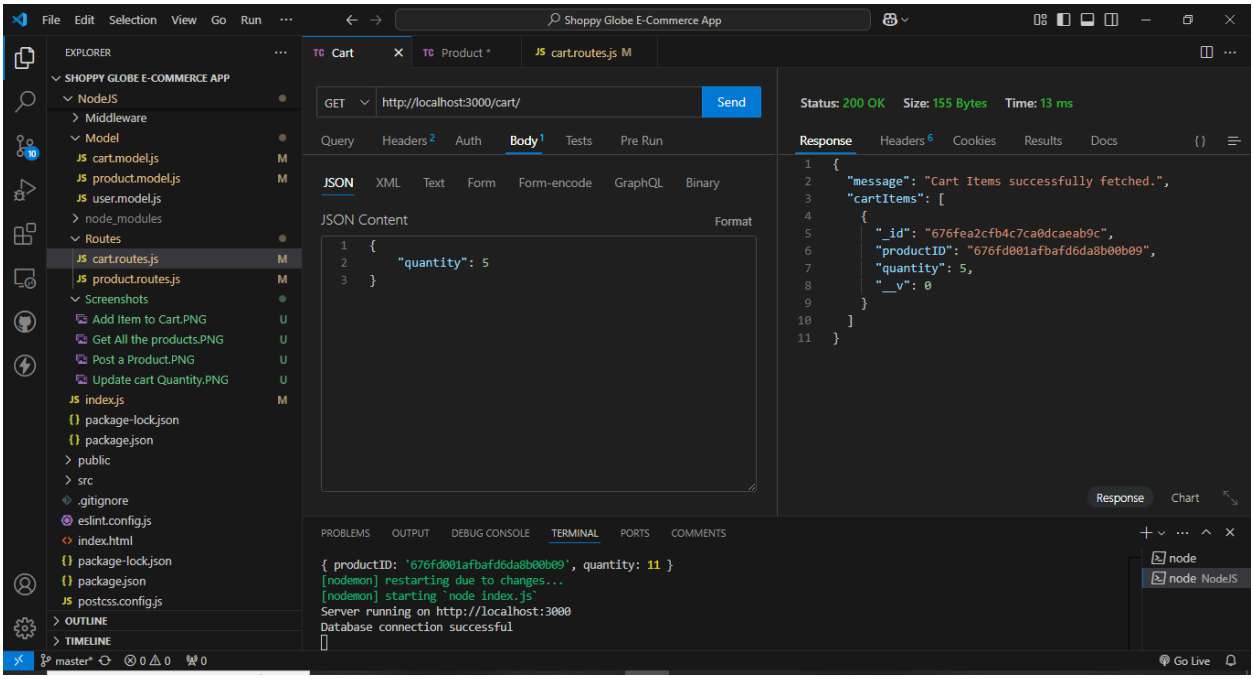
- **Request Body**:
- {
- "productID": "<product-id>",
- "quantity": 2
- }
- **Response**:
- {
- "message": "Cart Item Added Successfully",
- "newItem": {
- "_id": "<cart-item-id>",
- "productID": "<product-id>",
- "quantity": 2
- }
- }



GET /cart

Fetches all cart items.

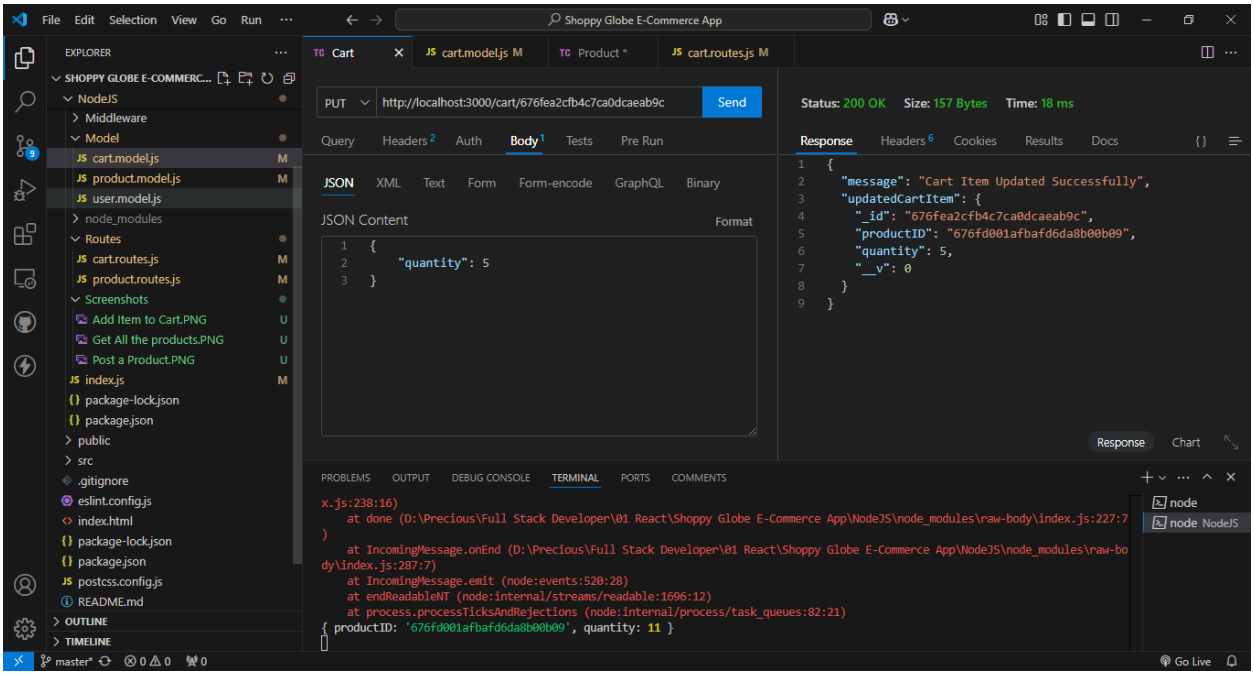
- **Response**:
- {
- "message": "Cart Items successfully fetched",
- "cartItems": [
- {
- "_id": "<cart-item-id>",
- "productID": {
- "_id": "<product-id>",
- "name": "Product Name",
- "price": 100
- },
- "quantity": 2
- }
-]
- }



PUT /cart/:id

Updates the quantity of a cart item.

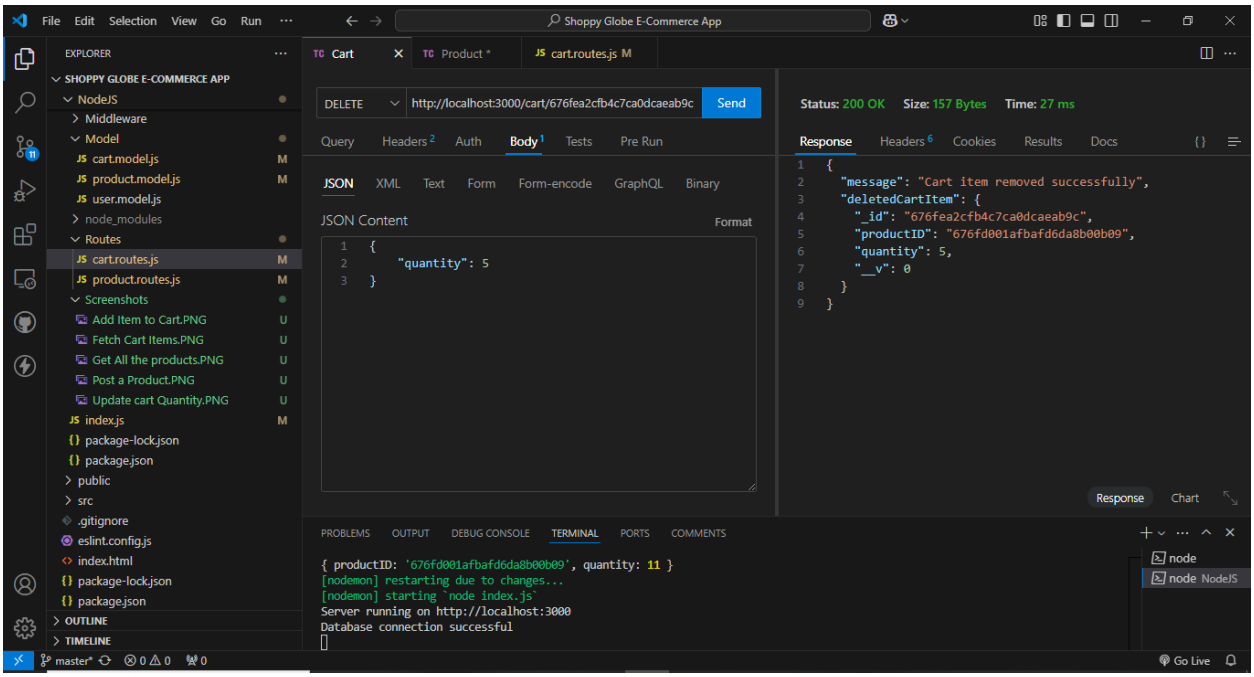
- **Request Body**:
- {
- "quantity": 3
- }
- **Response**:
- {
- "message": "Cart item quantity updated successfully",
- "cartItem": {
- "_id": "<cart-item-id>",
- "productID": "<product-id>",
- "quantity": 3
- }
- }



DELETE /cart/:id

Deletes a cart item.

- **Response**:
- {
- "message": "Cart item removed successfully",
- "deletedCartItem": {
- "_id": "<cart-item-id>",
- "productID": "<product-id>",
- "quantity": 2
- }
- }

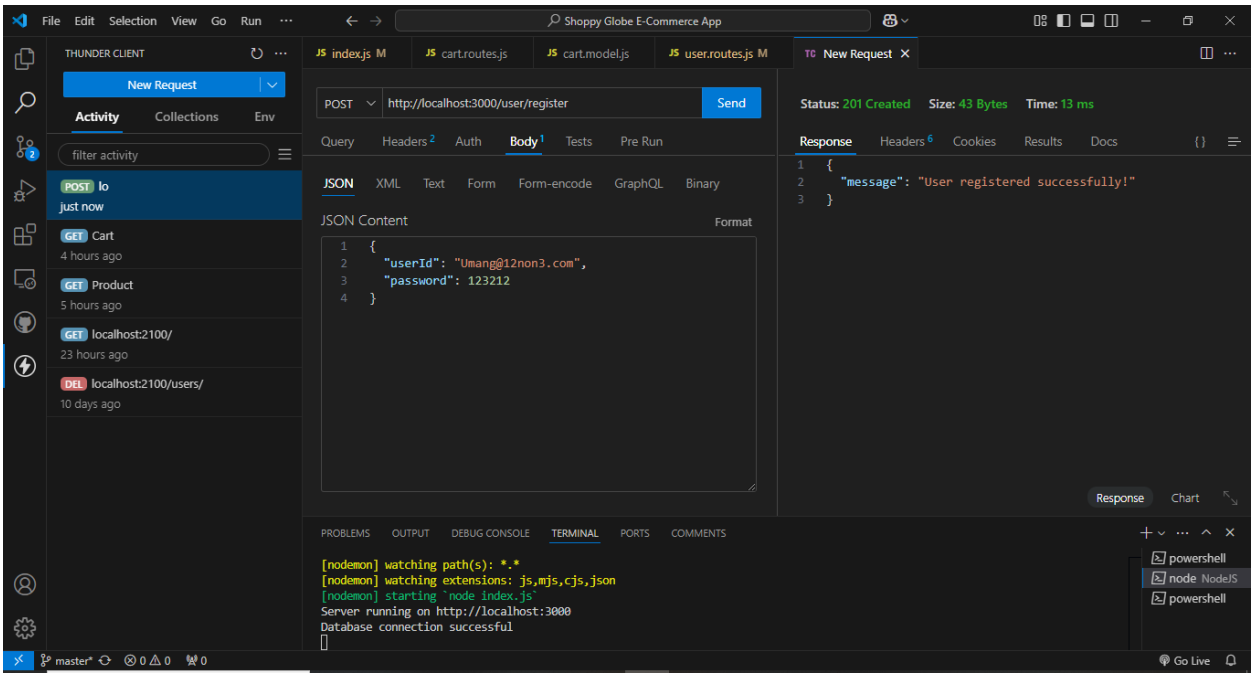


3. Authentication

POST /user/register

Registers a new user.

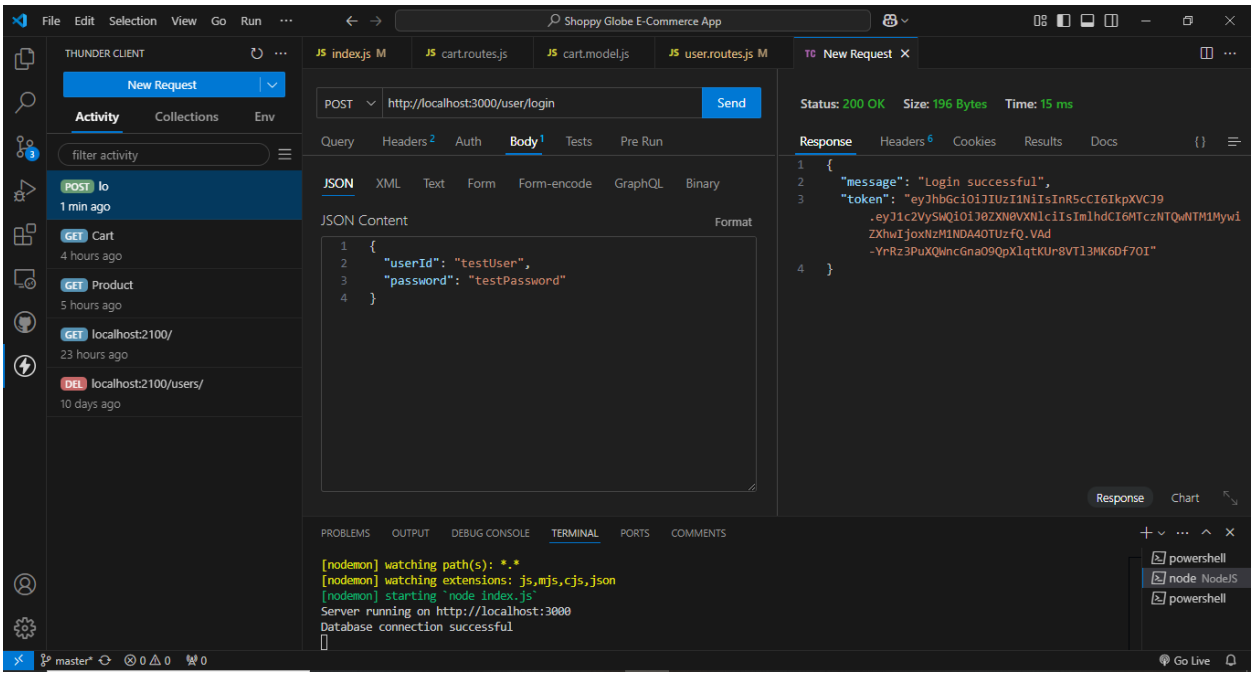
- **Request Body**:
- {
- "userId": "exampleUser",
- "password": "examplePassword"
- }
- **Response**:
- {
- "message": "User registered successfully!"
- }



POST /user/login

Logs in a user and generates a JWT token.

- **Request Body**:
- {
- "userId": "exampleUser",
- "password": "examplePassword"
- }
- **Response**:
- {
- "message": "Login successful",
- "token": "<jwt-token>"
- }



Error Handling

- Returns appropriate HTTP status codes and error messages for invalid inputs or server issues.
- Protects sensitive routes with JWT authentication.