

Project: Build APIs with Node.js and Express.js for Shoppyglobe E-commerce

GET <http://localhost:5100/products>

The screenshot shows the Thunder Client interface. A GET request to `http://localhost:5100/products` has been sent, resulting in a 200 OK status. The response is a JSON array of product data. The terminal at the bottom shows the server is running on `http://localhost:5100`.

```
GET http://localhost:5100/products

Status: 200 OK Size: 7.34 KB Time: 88 ms

Response
1 [
2   {
3     "_id": "6862989df4ef09a6fd5facad",
4     "name": "Essence Mascara Lash Princess",
5     "price": "9.99",
6     "brand": "beauty",
7     "description": "The Essence Mascara Lash Princess is a popular mascara known for its volumizing and lengthening effects. Achieve dramatic lashes with this long-lasting and cruelty-free formula.",
8     "stock": 99,
9     "__v": 0
10  },
11  {
12    "_id": "6862989df4ef09a6fd5facae",
13    "name": "Eyeshadow Palette with Mirror",
14    "price": "19.99",
15    "brand": "beauty",
16  },
17 ]
```

Terminal output:

```
MongoDB connected
● Fetched 30 products
● Cleared old products
● Products inserted successfully
● Products seeded from dummyjson.com
PS C:\Users\arti\OneDrive\Desktop\shoppyglobe_backend> npm start
> shoppyglobe_backend@1.0.0 start
> node server.js
MongoDB connected
Server running on http://localhost:5100
```

The screenshot shows the MongoDB Compass interface. The `products` collection in the `shoppyglobe` database is selected. The documents tab shows a list of product documents.

```
Users > shoppyglobe > products

Documents 0 Aggregations Schema Indexes 1 Validation

Type a query: { field: 'value' } or Generate query +

ADD DATA EXPORT DATA UPDATE DELETE 25 1 - 25 of 30

_id: ObjectId('6862989df4ef09a6fd5facad')
name: "Essence Mascara Lash Princess"
price: "9.99"
brand: "beauty"
description: "The Essence Mascara Lash Princess is a popular mascara known for its v..."
stock: 99
__v: 0

_id: ObjectId('6862989df4ef09a6fd5facae')
name: "Eyeshadow Palette with Mirror"
price: "19.99"
brand: "beauty"
description: "The Eyeshadow Palette with Mirror offers a versatile range of eyeshado..."
stock: 34
__v: 0

_id: ObjectId('6862989df4ef09a6fd5facaf')
name: "Powder Canister"
price: "14.99"
brand: "beauty"
description: "The Powder Canister is a finely milled setting powder designed to set ..."
stock: 89
__v: 0

_id: ObjectId('6862989df4ef09a6fd5facb0')
name: "Lipstick Set"
price: "24.99"
brand: "beauty"
description: "The Lipstick Set includes a variety of shades of lipstick, perfect for any o..."
stock: 15
__v: 0
```

GET <http://localhost:5100/products/68627db0d5f3f8dabc7e0f>

The screenshot displays the Thunder Client interface with a REST client tab titled 'shoppyglobe_backend'. The 'New Request' tab is active, showing a GET request to the URL `http://localhost:5100/products/68627db0d5f3f8dabc7e0f`. The request has been sent, resulting in a 200 OK status, 312 Bytes of data, and a response time of 16 ms.

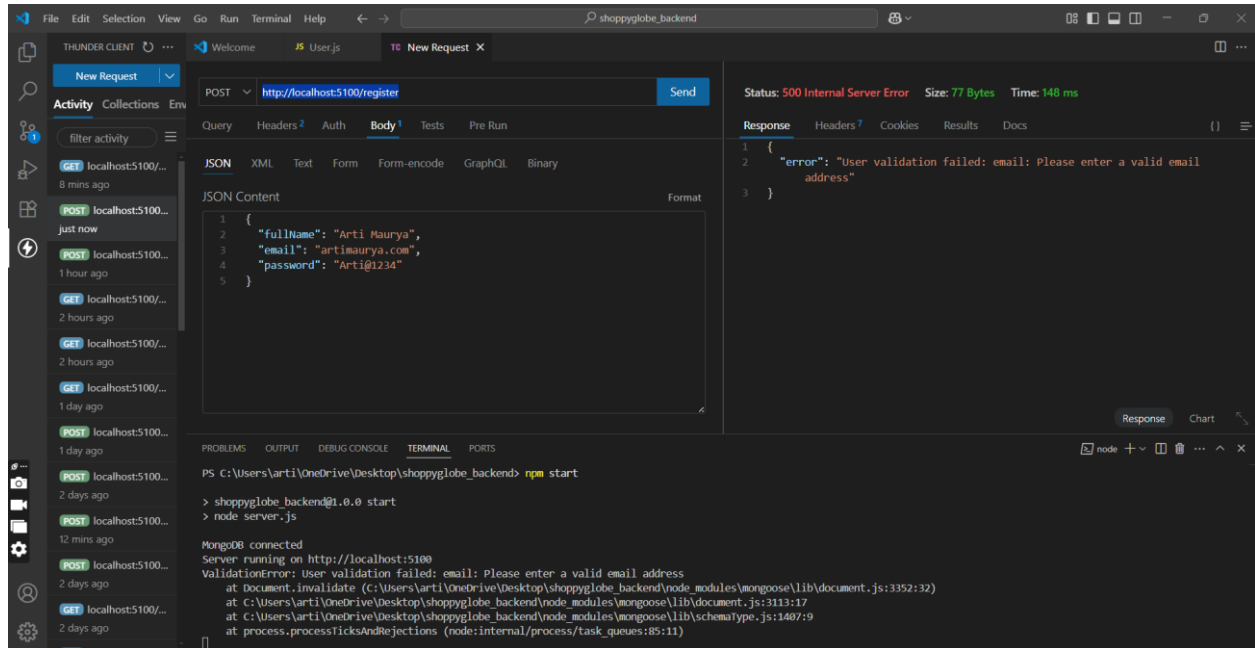
The response body is a JSON object representing a product:

```
1 {
2   "_id": "68627db0d5f3f8dabc7e0f",
3   "name": "Powder Canister",
4   "price": "$14.99",
5   "brand": "Velvet Touch",
6   "description": "The Powder Canister is a finely milled setting powder
7   designed to set makeup and control shine. With a lightweight and
8   translucent formula, it provides a smooth and matte finish.",
9   "stock": 4,
10  "_v": 0
11 }
```

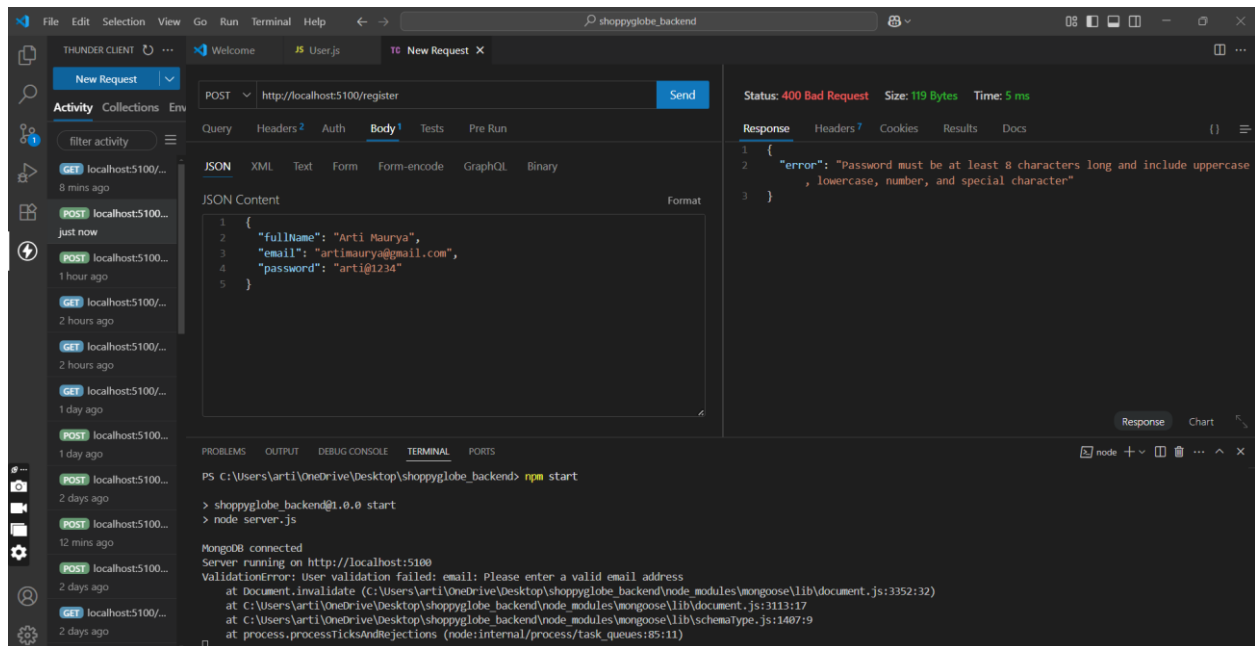
The bottom panel shows the terminal output, indicating that the server is running on `http://localhost:5100` and that MongoDB is connected.

POST <http://localhost:5100/register>

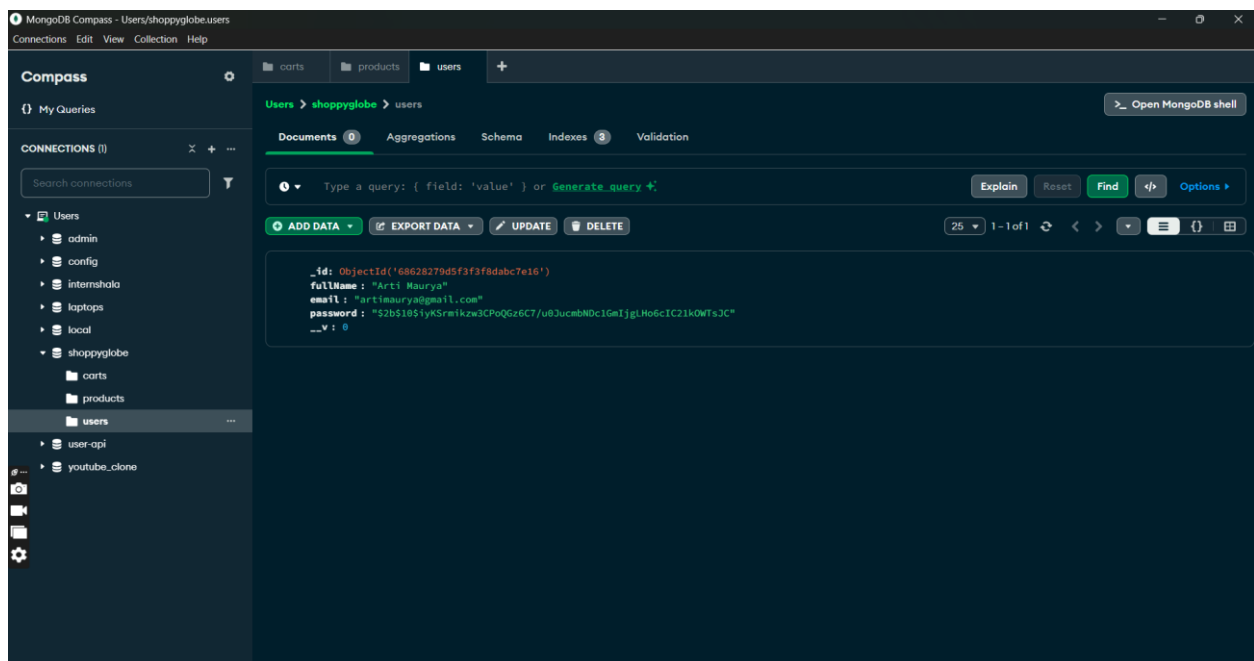
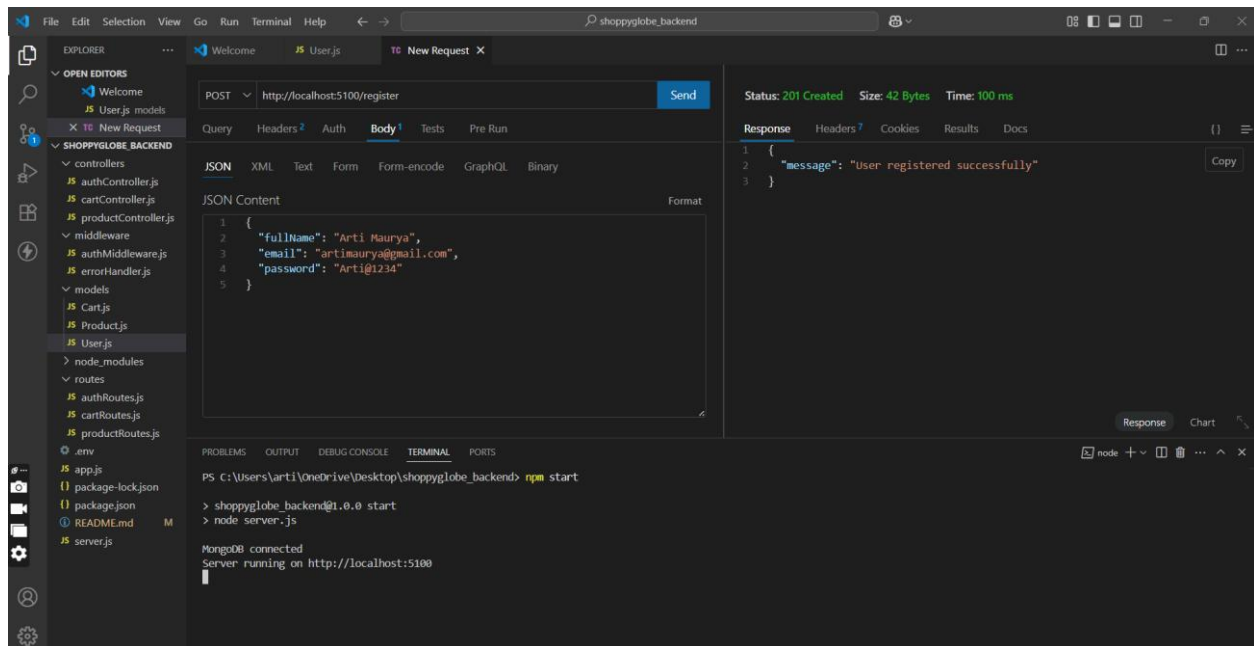
Error status (500 Internal Server Error): email validation: email is not in valid form



Error status (400 Bad Request): password validation: password is invalid not in required format



Status (201 Created): User registered successfully



Error status (400 Bad Request): Email already in use

The screenshot shows the Visual Studio Code interface with a REST client tab titled "New Request". The request is a POST to `http://localhost:5100/register` with a JSON body:

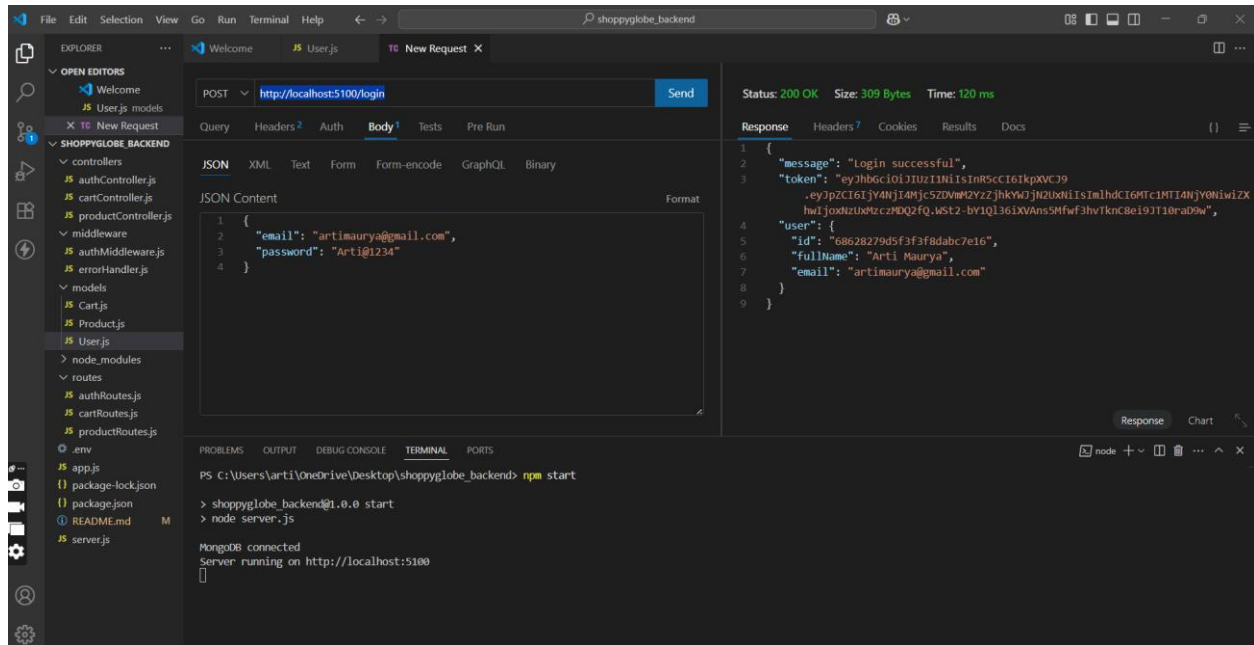
```
{  "fullName": "Arti Maurya",  "email": "artimaurya@gmail.com",  "password": "Arti@1234"}
```

The response is a 400 Bad Request with a status of 400, size of 32 Bytes, and time of 56 ms. The response body is:

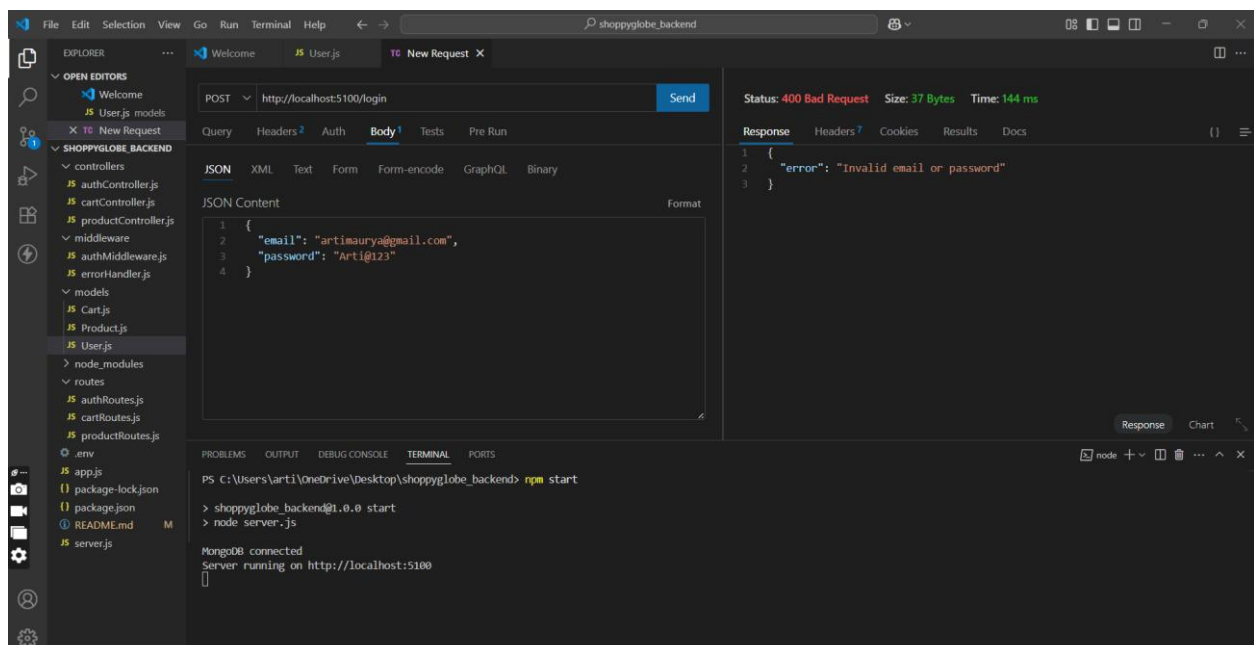
```
{  "error": "Email already in use"}
```

The terminal at the bottom shows the command `npm start` being executed, and the output indicates that the server is running on `http://localhost:5100`.

POST <http://localhost:5100/login>



Error status (400 Bad Request): Invalid email or password



POST <http://localhost:5100/cart>

Error status (401 Unauthorized): "Access denied"

The screenshot shows the Thunder Client interface. On the left, a list of requests is displayed, including several POST requests to localhost:5100. The main panel shows the details of a selected POST request to http://localhost:5100/cart. The Headers tab is active, showing the following headers:

Header	Value
Accept	*/*
User-Agent	Thunder Client (https://www.thunderclient.com)
header	value

The bottom panel shows the terminal output, which includes the command `npm start` and the response `Server running on http://localhost:5100`.

The screenshot shows the Thunder Client interface with the Body tab selected for the POST request to http://localhost:5100/cart. The JSON Content is displayed as follows:

```
1 {
2   "productId": "68627d74d5f3f8dabc7e0d",
3   "quantity": 2
4 }
```

The right panel shows the response details, indicating a **Status: 401 Unauthorized** with a size of 25 Bytes and a time of 5 ms. The response body is:

```
1 {
2   "error": "Access denied"
3 }
```

The bottom panel shows the terminal output, which includes the command `npm start` and the response `Server running on http://localhost:5100`.

Error status (400 Bad Request): "Only 2 item(s) in stock. You cannot add 4 to your cart."

THUNDER CLIENT

New Request

Activity Collections Env

filter activity

POST localhost:5100... just now

GET localhost:5100/... 35 mins ago

POST localhost:5100... 18 mins ago

POST localhost:5100... 8 mins ago

GET localhost:5100/... 2 hours ago

GET localhost:5100/... 2 hours ago

GET localhost:5100/... 1 day ago

POST localhost:5100... 1 day ago

POST localhost:5100... 5 mins ago

POST localhost:5100... 38 mins ago

POST localhost:5100... 2 days ago

POST http://localhost:5100/cart

Send

Query Headers 3 Auth Body 1 Tests Pre Run

HTTP Headers

Raw

Accept */*

User-Agent Thunder Client (https://www.thunderclient.com)

Authorization JWT eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJp...

header value

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\arti\OneDrive\Desktop\shoppyglobe_backend> npm start

> shoppyglobe_backend@1.0.0 start

> node server.js

o MongoDB connected

Server running on http://localhost:5100

THUNDER CLIENT

New Request

Activity Collections Env

filter activity

POST localhost:5100... just now

GET localhost:5100/... 35 mins ago

POST localhost:5100... 18 mins ago

POST localhost:5100... 8 mins ago

GET localhost:5100/... 2 hours ago

GET localhost:5100/... 2 hours ago

GET localhost:5100/... 1 day ago

POST localhost:5100... 1 day ago

POST localhost:5100... 5 mins ago

POST localhost:5100... 38 mins ago

POST localhost:5100... 2 days ago

POST http://localhost:5100/cart

Send

Query Headers 3 Auth Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content

Format

1 {

2 "productId": "68627d74d5f3f3f8dabc7e0d",

3 "quantity": 4

4 }

Status: 400 Bad Request Size: 67 Bytes Time: 9 ms

Response Headers 7 Cookies Results Docs

1 {

2 "error": "Only 2 item(s) in stock. You cannot add 4 to your cart."

3 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\arti\OneDrive\Desktop\shoppyglobe_backend> npm start

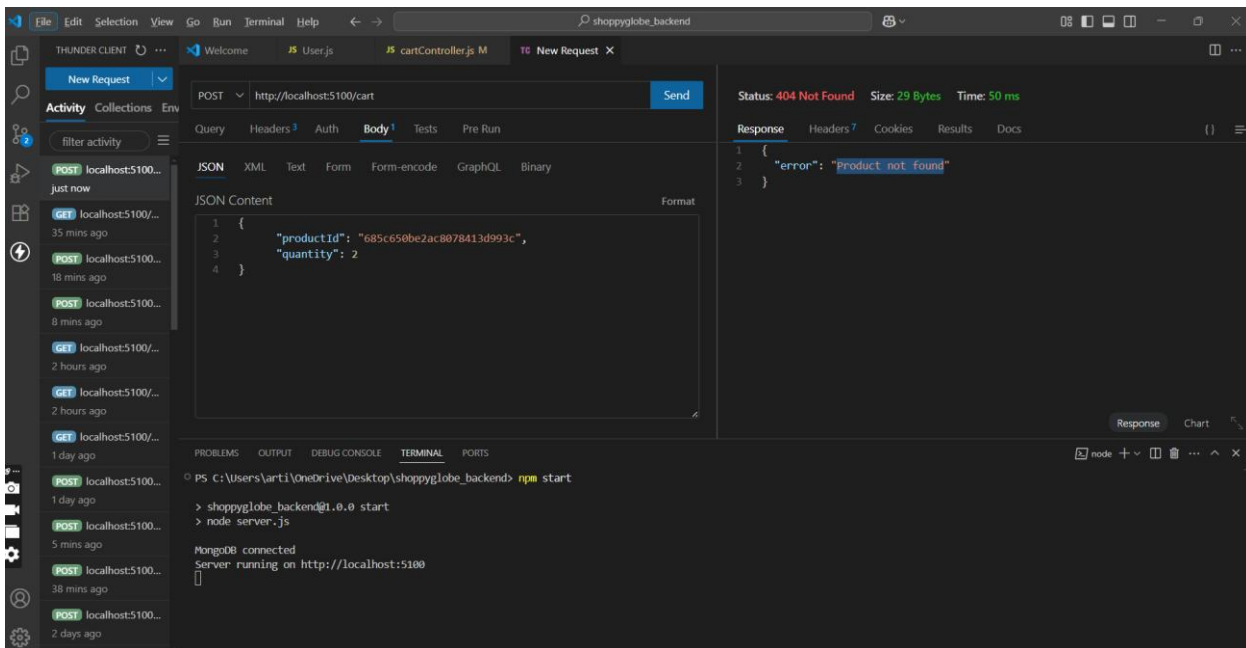
> shoppyglobe_backend@1.0.0 start

> node server.js

o MongoDB connected

Server running on http://localhost:5100

Error status (404 Not Found): " Product not found."



Status(200 OK) :

The screenshot shows the Thunder Client interface. A POST request to `http://localhost:5100/cart` has been sent successfully, returning a 200 OK status. The response is a JSON object with the following structure:

```
1 {
2   "userId": "68628279d5f3f8dabc7e16",
3   "items": [
4     {
5       "productId": "68627d74d5f3f8dabc7e0d",
6       "quantity": 2,
7       "_id": "686287f382e5783e9e1d9d75"
8     }
9   ],
10  "_id": "686287f382e5783e9e1d9d74",
11  "_v": 0
12 }
```

The terminal at the bottom shows the command `npm start` being executed, and a message indicating that MongoDB is connected and the server is running on `http://localhost:5100`.

The screenshot shows the MongoDB Compass interface. The 'carts' collection in the 'shoppyglobe' database is selected. The document view shows a single document with the following structure:

```
{
  "_id": ObjectId("6862896982e5783e9e1d9d7b"),
  "userId": ObjectId("68628279d5f3f8dabc7e16"),
  "items": Array (1)
  - 0: Object
    productId: ObjectId("68627d74d5f3f8dabc7e0d")
    quantity: 2
    _id: ObjectId("6862896982e5783e9e1d9d7c")
  "_v": 0
}
```

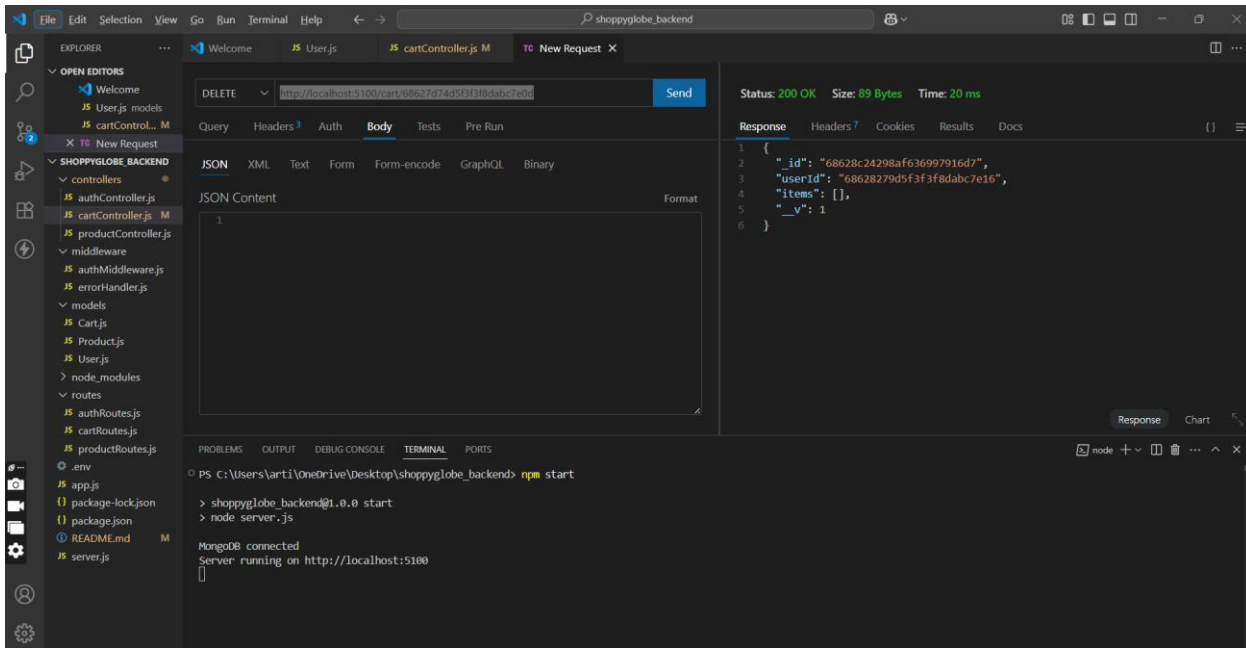
The interface also shows the 'My Queries' section on the left and the 'Open MongoDB shell' button on the right.

PUT <http://localhost:5100/cart/68627d74d5f3f3f8dabc7e0d>

The screenshot shows the Visual Studio Code interface with a REST client request and response. The request is a PUT method to the URL `http://localhost:5100/cart/68627d74d5f3f3f8dabc7e0d. The request body is a JSON object: { "quantity": 1 }. The response is a 200 OK status with a size of 175 Bytes and a time of 19 ms. The response body is a JSON object: { "_id": "68628c24298af636997916d7", "userId": "68628279d5f3f3f8dabc7e16", "items": [{ "productId": "68627d74d5f3f3f8dabc7e0d", "quantity": 1, "_id": "68628c24298af636997916d8" }], "_v": 0 }. The terminal shows the command npm start and the output shoppyglobe_backend@1.0.0 start and node server.js. The MongoDB connection is established and the server is running on http://localhost:5100.`

The screenshot shows the MongoDB Compass interface. The database is `shoppyglobe` and the collection is `cart`. The documents tab is selected, showing a single document with the following structure: `{ "_id": ObjectId("68628c24298af636997916d7"), "userId": ObjectId("68628279d5f3f3f8dabc7e16"), "items": Array (1), "_v": 0 }`. The document is expanded to show the `items` array, which contains one object: `{ "productId": ObjectId("68627d74d5f3f3f8dabc7e0d"), "quantity": 1, "_id": ObjectId("68628c24298af636997916d8") }`. The interface also shows the `collections` list on the left, including `admin`, `config`, `intemshala`, `laptops`, `local`, `shoppyglobe`, `users`, `user-api`, and `youtube_clone`.

DELETE <http://localhost:5100/cart/68627d74d5f3f8dabc7e0d>



Github Link: - <https://github.com/Arti2510/shopyglobeProject.git>