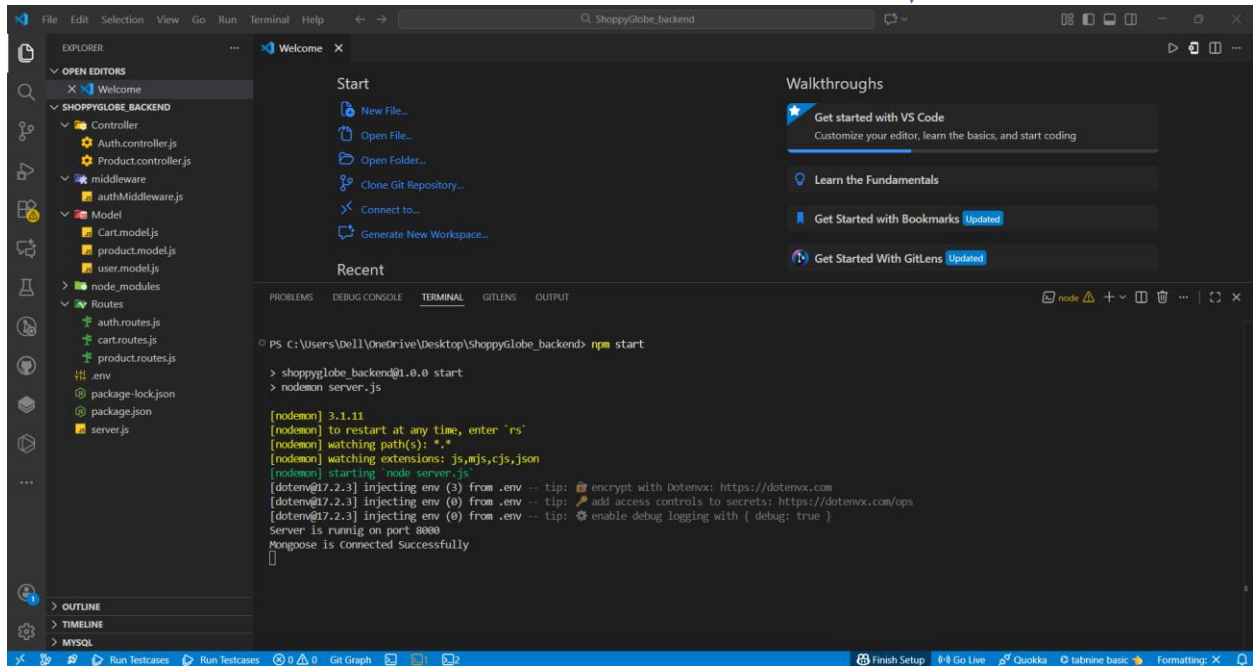
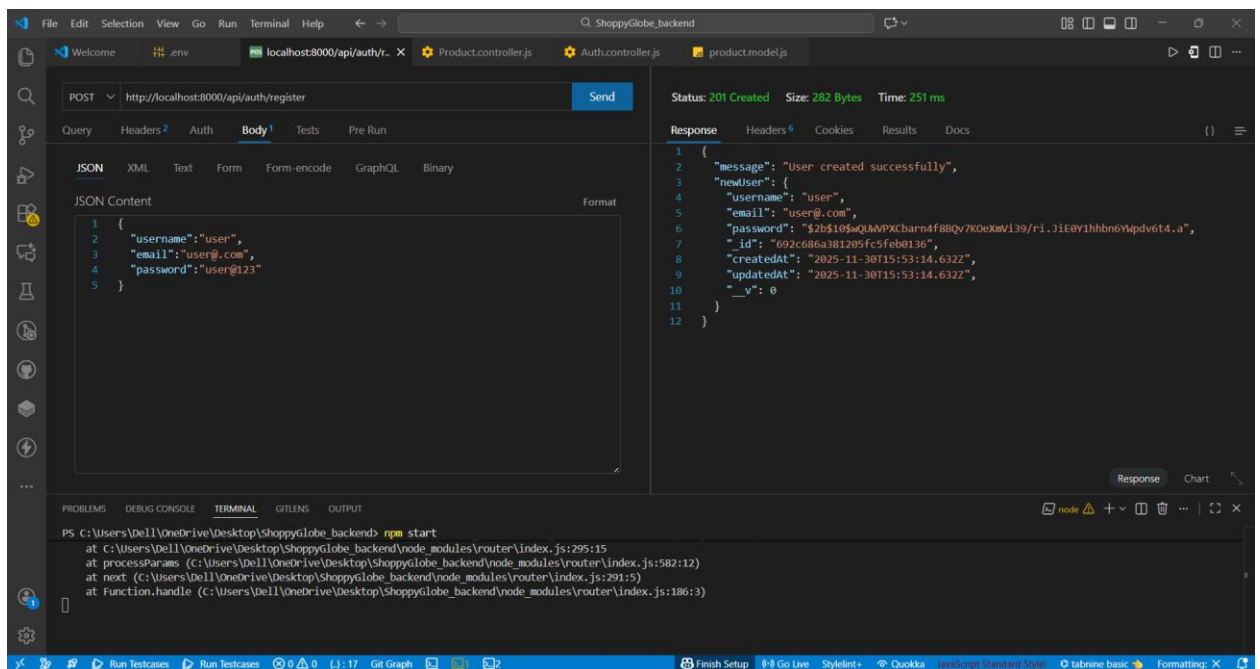


Screen-Shots of ShoppyGlobe_backend

1. Server and MongoDB connection



2. Register



3. User with same name or email

The screenshot shows a REST client in VS Code with the following details:

- Method:** POST
- URL:** http://localhost:8000/api/auth/register
- Body (JSON):**

```
{ 1: { 2: "username": "user123", 3: "email": "user@.com", 4: "password": "user@123" 5: }
```
- Status:** 500 Internal Server Error
- Size:** 159 Bytes
- Time:** 325 ms
- Response (JSON):**

```
1 { 2: "error": "Internal Server Error", 3: "message": "E11000 duplicate key error collection: Shoppyglobe_backend.users index: email_1 dup key: { email: \"user@.com\" }" 4: }
```
- Terminal:** Shows the command `npm start` and the application running on port 8000.

The screenshot shows a REST client in VS Code with the following details:

- Method:** POST
- URL:** http://localhost:8000/api/auth/register
- Body (JSON):**

```
1 { 2: "username": "user", 3: "email": "user123@.com", 4: "password": "user@123" 5: }
```
- Status:** 400 Bad Request
- Size:** 35 Bytes
- Time:** 111 ms
- Response (JSON):**

```
1 { 2: "error": "Username already exists" 3: }
```
- Terminal:** Shows the command `npm start` and the application running on port 8000.

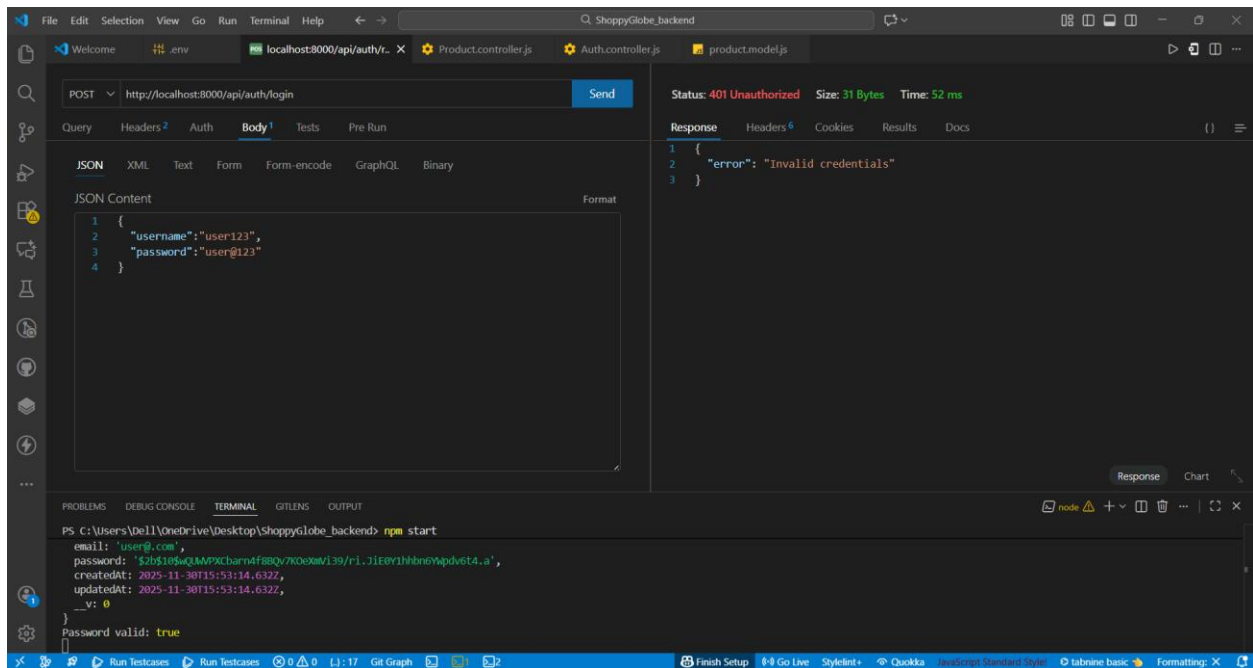
4. Register MongoDB

The screenshot shows the MongoDB Cloud console interface. The left sidebar contains navigation options like Overview, Data Explorer, Real Time, Cluster Metrics, Query Insights, Performance Advisor, Online Archive, Command Line Tools, and Infrastructure as Code. The main area displays the 'Data Explorer' for the 'ShoppYGlobe_backend.users' collection. It shows the database 'ShoppYGlobe_backend' and the collection 'users'. The data is displayed in a table format with columns for '_id', 'username', 'email', 'password', 'createdAt', 'updatedAt', and '__v'. The first document shows a user with username 'user' and email 'user@.com'. The second document shows a user with username 'user' and email 'user@123.com'. The interface also includes a search bar, a filter button, and a 'Generate queries from natural language in Compass' link.

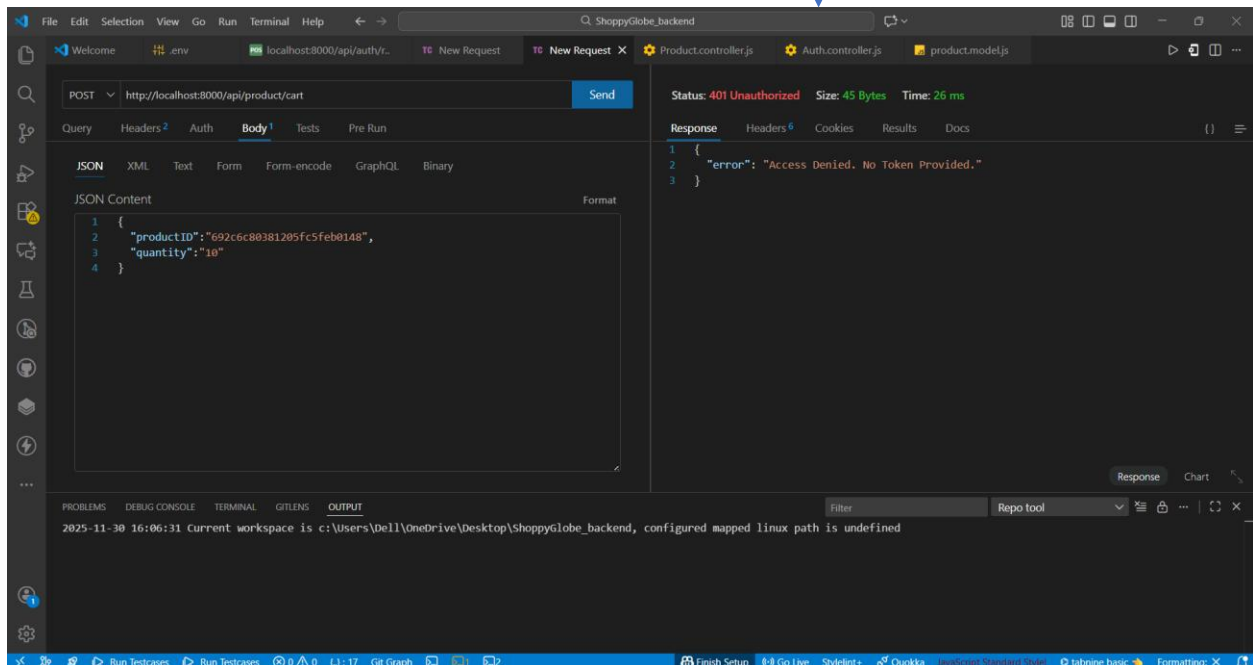
5. Login

The screenshot shows a web browser window displaying a REST client interface. The request is a POST to 'http://localhost:8000/api/auth/login'. The response is a 200 OK status with a size of 217 Bytes and a time of 328 ms. The response body is a JSON object containing a 'message' field with the value 'Login successful', a 'token' field with a long alphanumeric string, and a 'password' field with the value 'user@123'. The terminal at the bottom shows the command 'PS C:\Users\De11\OneDrive\Desktop\shoppYGlobe_backend> npm start' and the output of the application, which includes the user details and the password validation result 'Password valid: true'.

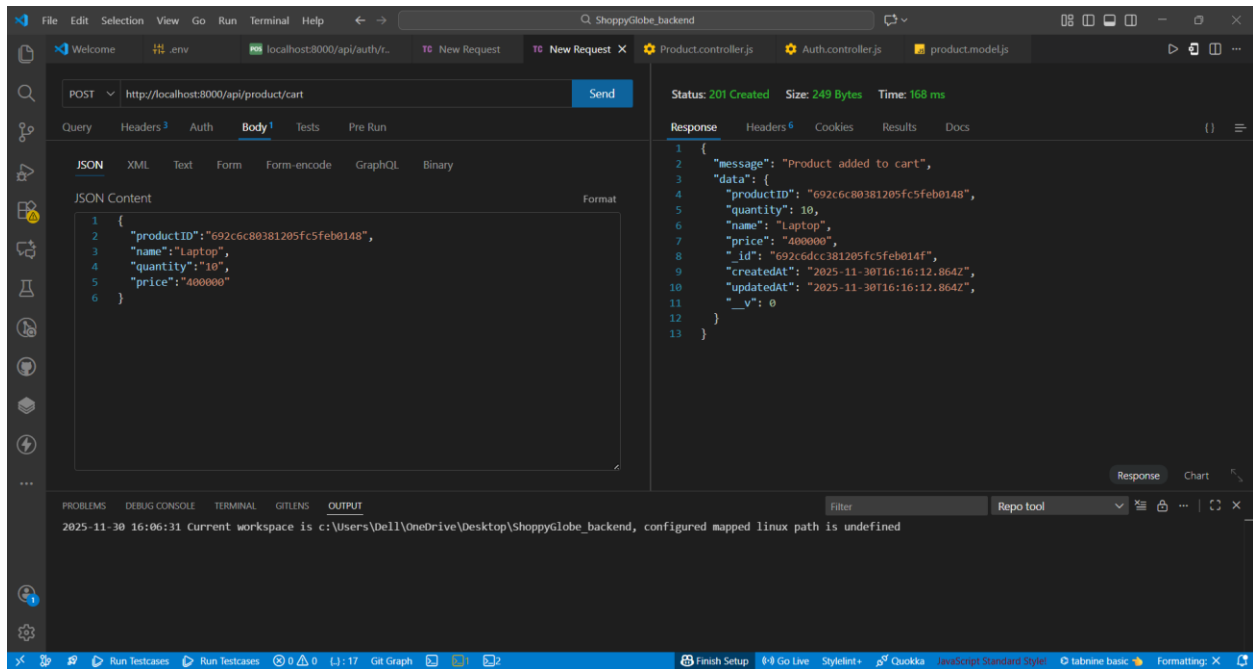
6. Wrong name or password



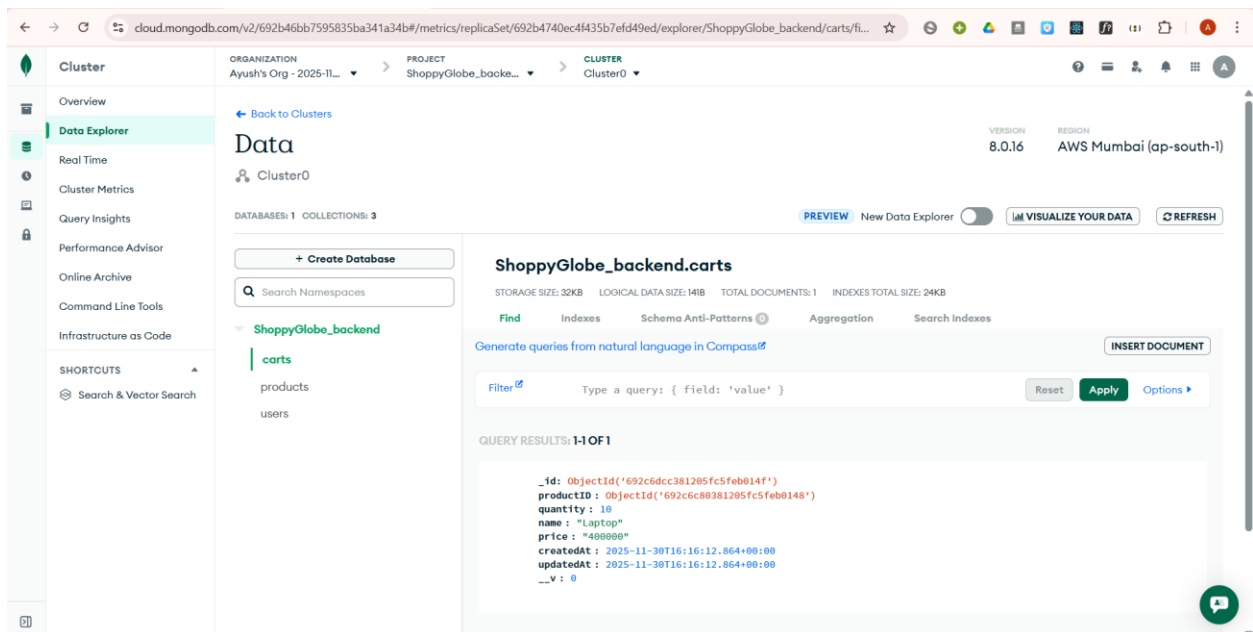
7. Cart before logging



8. After Valid Token/ logging



9. Cart in MongoDB



10. Cart Update

The screenshot shows the VS Code interface with a REST client request configured to update a cart item. The request is a PUT to `http://localhost:8000/api/product/cart/692c6dcc381205fc5feb014f` with a JSON body `{ "quantity": "5" }`. The response is a 200 OK status with a JSON body containing a success message and updated cart details.

```
PUT http://localhost:8000/api/product/cart/692c6dcc381205fc5feb014f
```

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

Status: 200 OK Size: 252 Bytes Time: 82 ms

```
{
  "message": "Cart updated successfully",
  "data": {
    "_id": "692c6dcc381205fc5feb014f",
    "productID": "692c6c80381205fc5feb0148",
    "quantity": 5,
    "name": "Laptop",
    "price": "400000",
    "createdAt": "2025-11-30T16:16:12.864Z",
    "updatedAt": "2025-11-30T17:17:23.897Z",
    "__v": 0
  }
}
```

11. Cart Update in MongoDB

The screenshot shows the MongoDB Cloud Portal interface. The 'Data Explorer' tab is active, displaying the 'carts' collection in the 'ShoppyGlobe_backend' database. The collection contains one document with the following fields: `_id`, `productID`, `quantity`, `name`, `price`, `createdAt`, `updatedAt`, and `__v`.

Cluster: Ayush's Org - 2025-11-30
Project: ShoppyGlobe_backend
Cluster: Cluster0

Version: 8.0.16
Region: AWS Mumbai (ap-south-1)

DATABASES: 1 COLLECTIONS: 3

ShoppyGlobe_backend

- products
- users

ShopyGlobe_backend.carts

STORAGE SIZE: 36KB LOGICAL DATA SIZE: 14KB TOTAL DOCUMENTS: 1 INDEXES TOTAL SIZE: 36KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

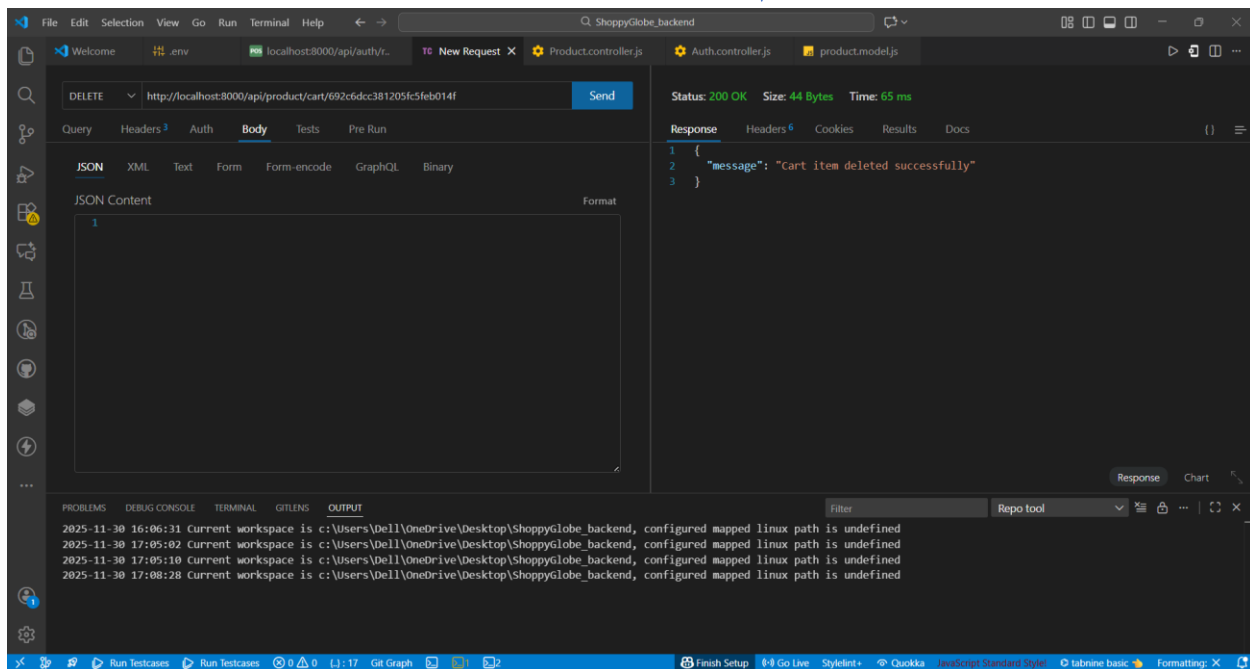
Generate queries from natural language in Compass

Filter Type a query: { field: 'value' } Reset Apply Options

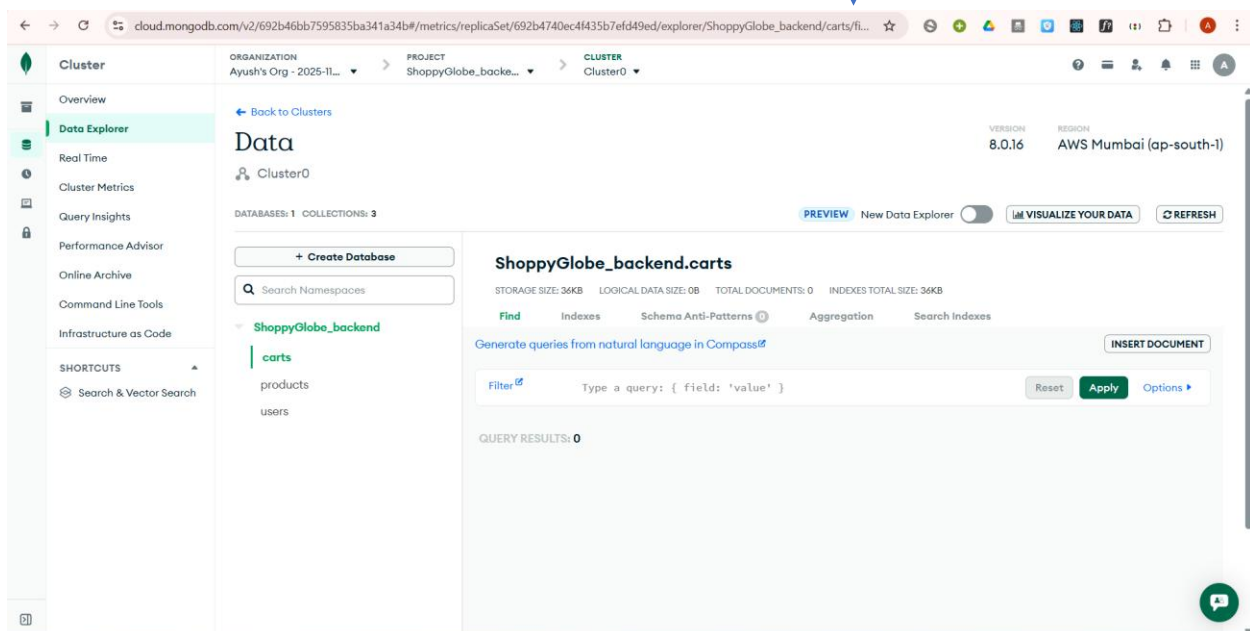
QUERY RESULTS: 1-1 OF 1

```
{
  "_id": ObjectId('692c6dcc381205fc5feb014f'),
  "productID": ObjectId('692c6c80381205fc5feb0148'),
  "quantity": 5,
  "name": "Laptop",
  "price": "400000",
  "createdAt": "2025-11-30T16:16:12.864+00:00",
  "updatedAt": "2025-11-30T17:17:23.897+00:00",
  "__v": 0
}
```

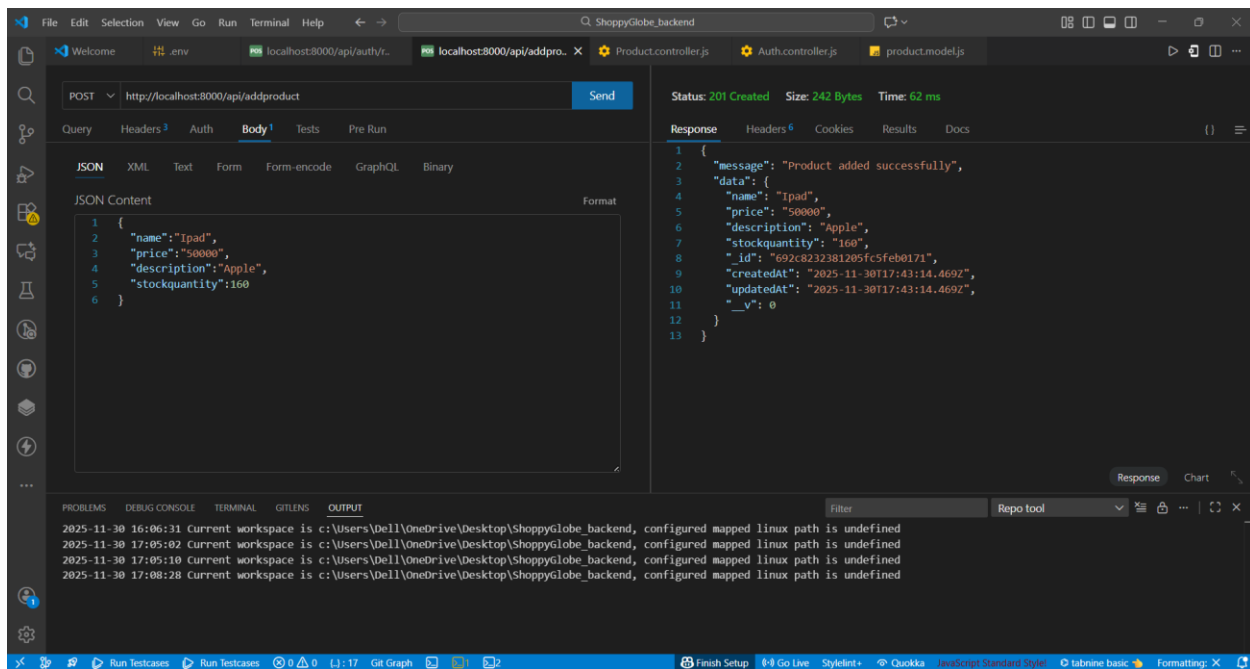
12. Cart Delete



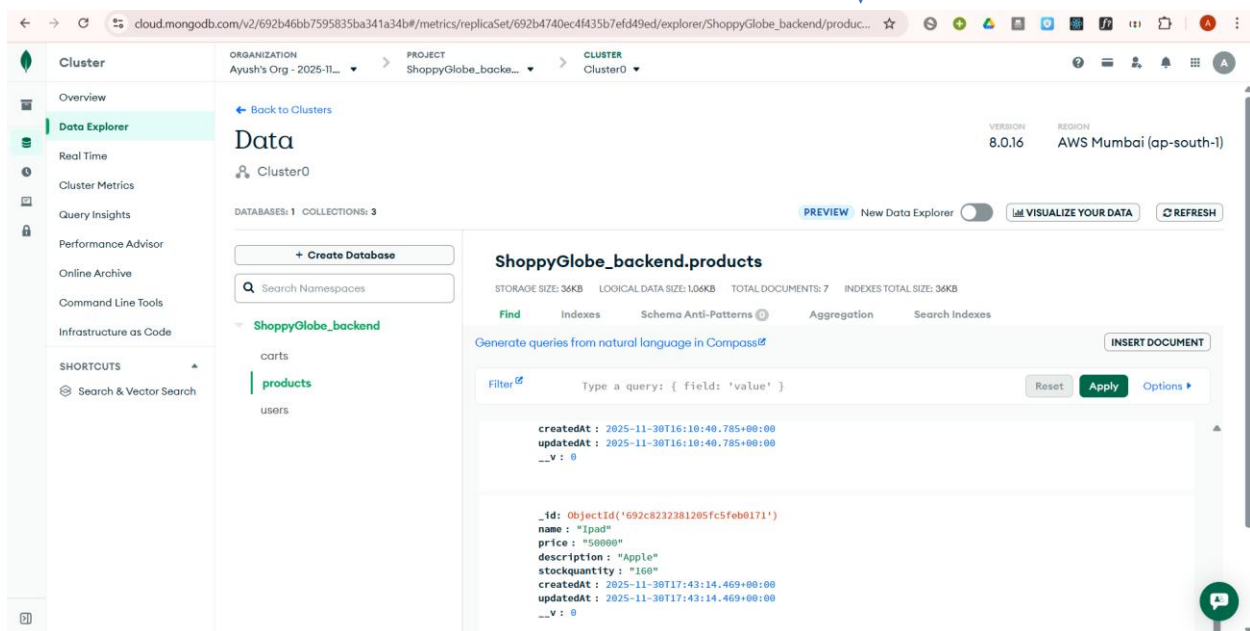
13. Cart Delete in MongoDB



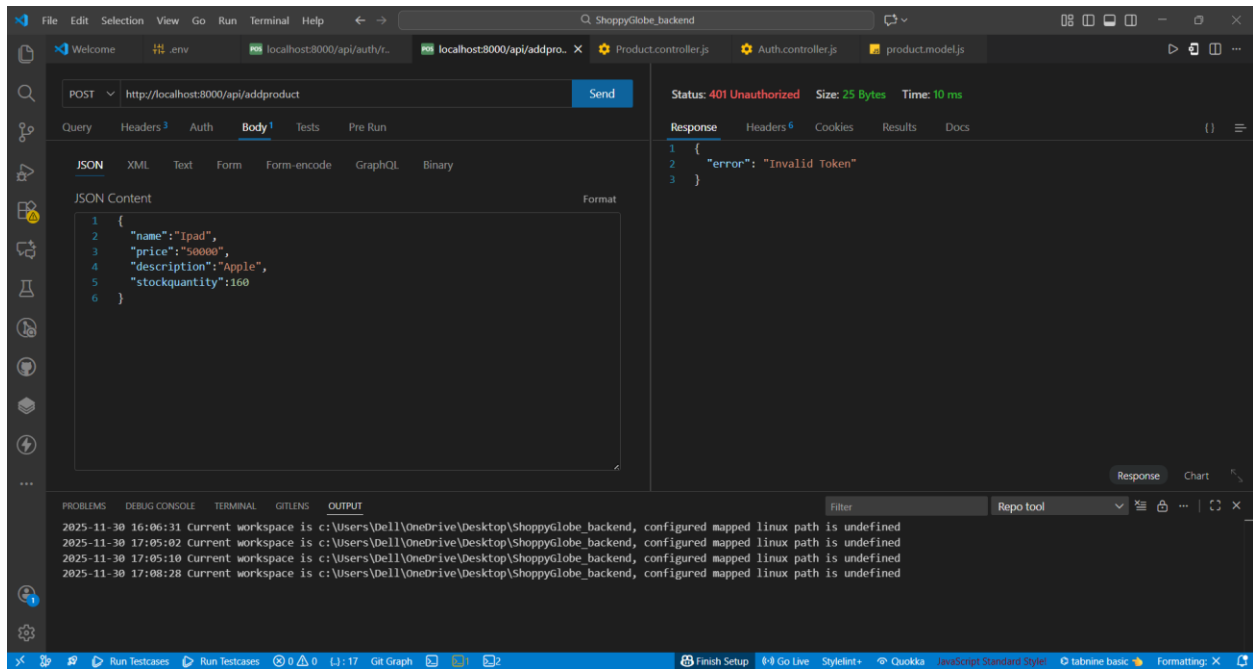
14. MongoDB CRUD Operation (Add Product) ↓



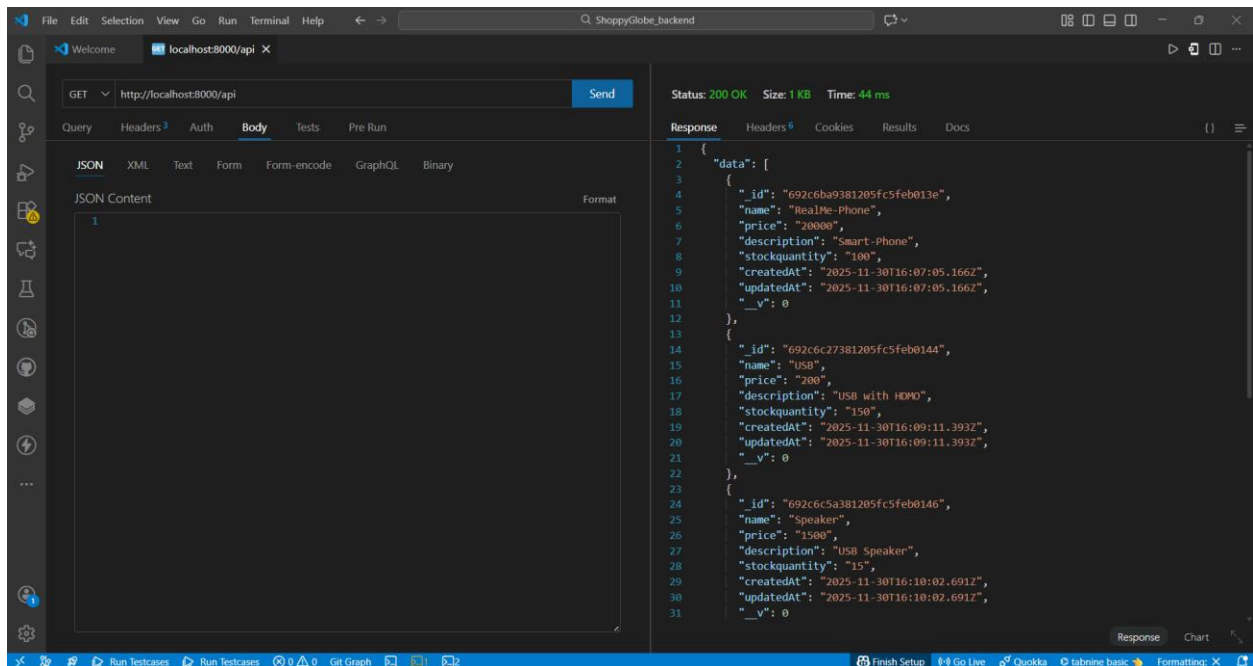
15. Add Product in MongoDB ↓



16. Add With Invalid Token



17. Fetch Product From database



18. Fetch product by id

The screenshot shows the VS Code interface with a REST client request configured to fetch a product by ID. The request is a GET method to the URL `http://localhost:8000/api/product/692c6c80381205fc5feb0148`. The response is a JSON object representing a product.

Request:

```
GET http://localhost:8000/api/product/692c6c80381205fc5feb0148
```

Response:

```
{
  "data": {
    "_id": "692c6c80381205fc5feb0148",
    "name": "Laptop",
    "price": "40000",
    "description": "Intel core i5",
    "stockquantity": "200",
    "createdAt": "2025-11-30T16:10:40.785Z",
    "updatedAt": "2025-11-30T16:10:40.785Z",
    "__v": 0
  }
}
```

19. Update Product

The screenshot shows the VS Code interface with a REST client request configured to update a product. The request is a PUT method to the URL `http://localhost:8000/api/updateproduct/692c6c80381205fc5feb0148`. The request body is a JSON object with the updated product details. The response is a JSON object with a success message and the updated product details.

Request:

```
PUT http://localhost:8000/api/updateproduct/692c6c80381205fc5feb0148
```

Request Body (JSON):

```
{
  "name": "Laptop-DELL",
  "price": "40000",
  "description": "Intel core i7",
  "stockquantity": "350"
}
```

Response:

```
{
  "message": "Product updated successfully",
  "data": {
    "_id": "692c6c80381205fc5feb0148",
    "name": "Laptop-DELL",
    "price": "40000",
    "description": "Intel core i7",
    "stockquantity": "350",
    "createdAt": "2025-11-30T16:10:40.785Z",
    "updatedAt": "2025-11-30T17:51:53.357Z",
    "__v": 0
  }
}
```

20. Before Update in MongoDB

The screenshot shows the MongoDB Cloud console interface. The left sidebar contains navigation options: Cluster, Overview, Data Explorer (selected), Real Time, Cluster Metrics, Query Insights, Performance Advisor, Online Archive, Command Line Tools, Infrastructure as Code, and SHORTCUTS (Search & Vector Search). The main panel displays the 'Data' view for the 'ShoppYGlobe_backend.products' collection. The collection has 7 documents. The first document is expanded, showing the following JSON structure:

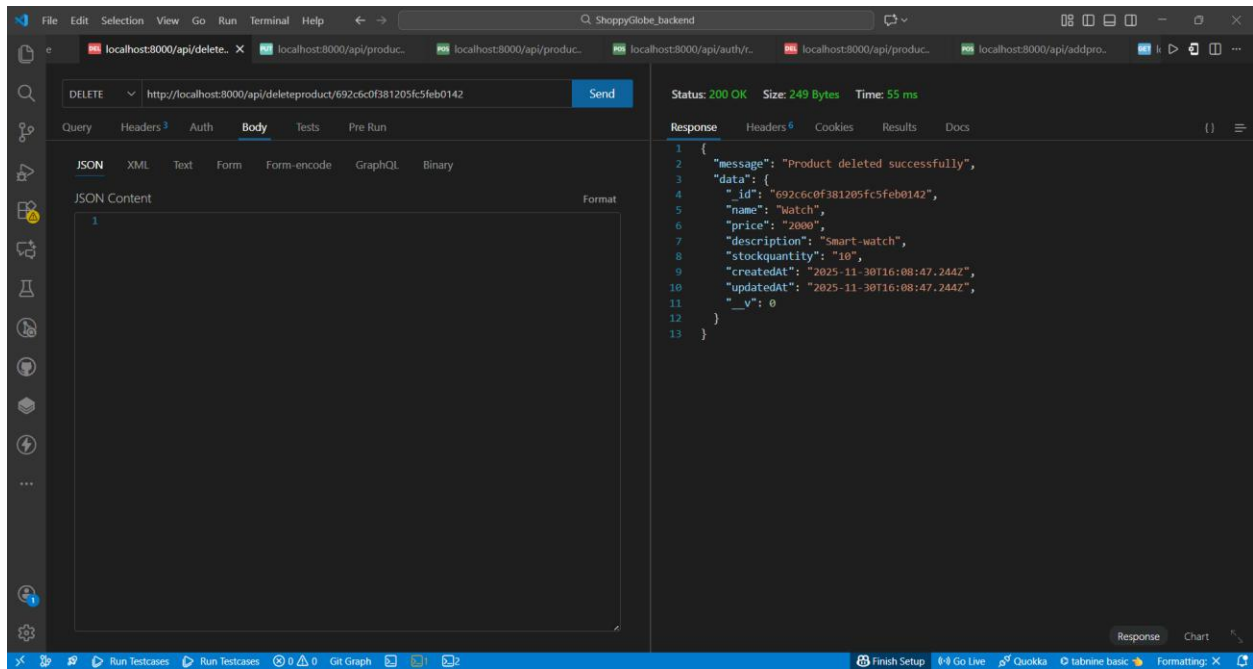
```
{
  "_id": ObjectId("692c6c80381205fc5feb0148"),
  "name": "Laptop",
  "price": "40000",
  "description": "Intel core i5",
  "stockquantity": "200",
  "createdAt": "2025-11-30T16:10:40.785+00:00",
  "updatedAt": "2025-11-30T16:10:40.785+00:00"
}
```

21. After Update in MongoDB

The screenshot shows the MongoDB Cloud console interface after an update. The left sidebar is the same as in the previous screenshot. The main panel displays the 'Data' view for the 'ShoppYGlobe_backend.products' collection. The collection has 7 documents. The first document is expanded, showing the following JSON structure:

```
{
  "_id": ObjectId("692c6c80381205fc5feb0148"),
  "name": "Laptop-DELL",
  "price": "40000",
  "description": "Intel core i7",
  "stockquantity": "350",
  "createdAt": "2025-11-30T16:10:40.785+00:00",
  "updatedAt": "2025-11-30T17:51:53.357+00:00"
}
```

22. Delete Product form Database



23. Delete from MongoDB

