

# ShoppYGlobe Backend API

**Technologies:** Node.js, Express.js, MongoDB, JWT Authentication

---

## 1. Introduction

This project implements the backend for **ShoppYGlobe**, an e-commerce application. It provides APIs for **products management, shopping cart operations, and user authentication**.

---

## 2. Tech Stack

- **Backend Framework:** Node.js + Express.js
  - **Database:** MongoDB (Atlas / Local)
  - **Authentication:** JWT (JSON Web Tokens)
  - **Testing Tool:** ThunderClient
- 

## 3. Features

### Products API

- **GET /products** – Fetch all products
- **GET /products/:id** – Fetch product details by ID

### Cart API (*Protected Routes*)

- **POST /cart** – Add product to cart
- **PUT /cart/:id** – Update product quantity in cart
- **DELETE /cart/:id** – Remove product from cart

### Authentication API

- **POST /register** – Register new user

#### Register a New User

- **Endpoint:** POST /register
- **Description:** Creates a new user account.

#### Request Body:

```
{
```

```
"userName": "Sourabh",  
"email":  
"Sourabh4812@gmail.com",  
"password": "123456"  
}
```

### Response:

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

### Notes:

Password must be at least 8 characters (validation implemented in schema).

Email must be unique and valid format.

### Screenshots showing request & response for each API:

POST <http://localhost:5000/register->

The screenshot displays a REST client interface with a POST request to `http://localhost:5000/register`. The request body is a JSON object: `{ "userName": "Vaishnavi", "email": "Vaishnavi@gmail.com", "password": "Vaishnavi1234" }`. The response is a 200 OK status with a JSON body: `{ "message": "User Registered Successfully" }`. The bottom terminal shows the server running on port 5000 with a database connection.

```
POST http://localhost:5000/register  
Status: 200 OK Size: 42 Bytes Time: 77 ms  
Response  
1 {  
2   "message": "User Registered Successfully"  
3 }  
JSON Content  
1 {  
2   "userName": "Vaishnavi",  
3   "email": "Vaishnavi@gmail.com",  
4   "password": "Vaishnavi1234"  
5 }  
[nodemon] to restart at any time, enter `rs`  
[nodemon] watching path(s): *.*  
[nodemon] watching extensions: js,mjs,cjs,json  
[nodemon] starting `node server.js`  
[dotenv@17.2.1] injecting env (3) from .env -- tip: suppress all logs with { quiet: true }  
Server is running on port 5000  
DataBase Connected Successfully
```

- **POST /login** – Login user & return JWT **Endpoint:** POST /login

**Description:** Logs in a user and generates a JWT token.

**Request Body:**

```
{  
  "email": "soniamalik@gmail.com",  
  "password": "sonia@1234"  
}
```

**Screenshots showing request & response for each API:**

POST <http://localhost:5000/login>

The screenshot displays an API client interface with the following details:

- Request:**
  - Method: POST
  - URL: <http://localhost:5000/login>
  - Body (JSON):

```
{  
  "email": "soniamalik@gmail.com",  
  "password": "sonia@1234"  
}
```
- Response:**
  - Status: 200 OK
  - Size: 271 Bytes
  - Time: 68 ms
  - Response body:

```
password matched , User Logged In and Token generated  
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
eyJ1c2VySWQiOiI2OGIwMWZiMzVlNTkNGF1ZGjMDViMzEiLCJlbWFPbCI6InVbm1hbWVsaWtAZ21haWwY29tIiwiaWF0IjoxNzU2MzcyOTkxLCJleHAiOiJlMzNTYzNzQ3OTF9.  
WnODT8N1rkG3twag10EinCfYcP_4FaGRbmt1EkJEpyo
```

The bottom of the image shows a terminal window with the following output:

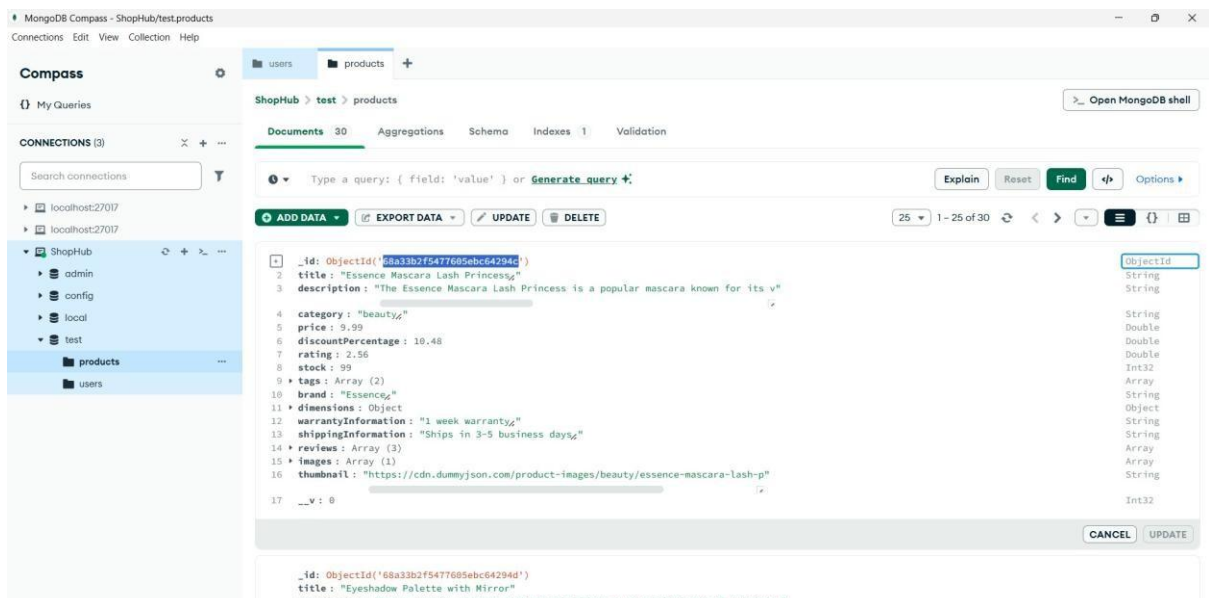
```
[nodemon] to restart at any time, enter `rs`  
[nodemon] watching path(s): *.*  
[nodemon] watching extensions: js,mjs,cjs,json  
[nodemon] starting `node server.js`  
[dotenv@17.2.1] injecting env (3) from .env -- tip: ✨ suppress all logs with { quiet: true }  
Server is running on port 5000  
DataBase Connected Successfully
```

## 4. Database Structure

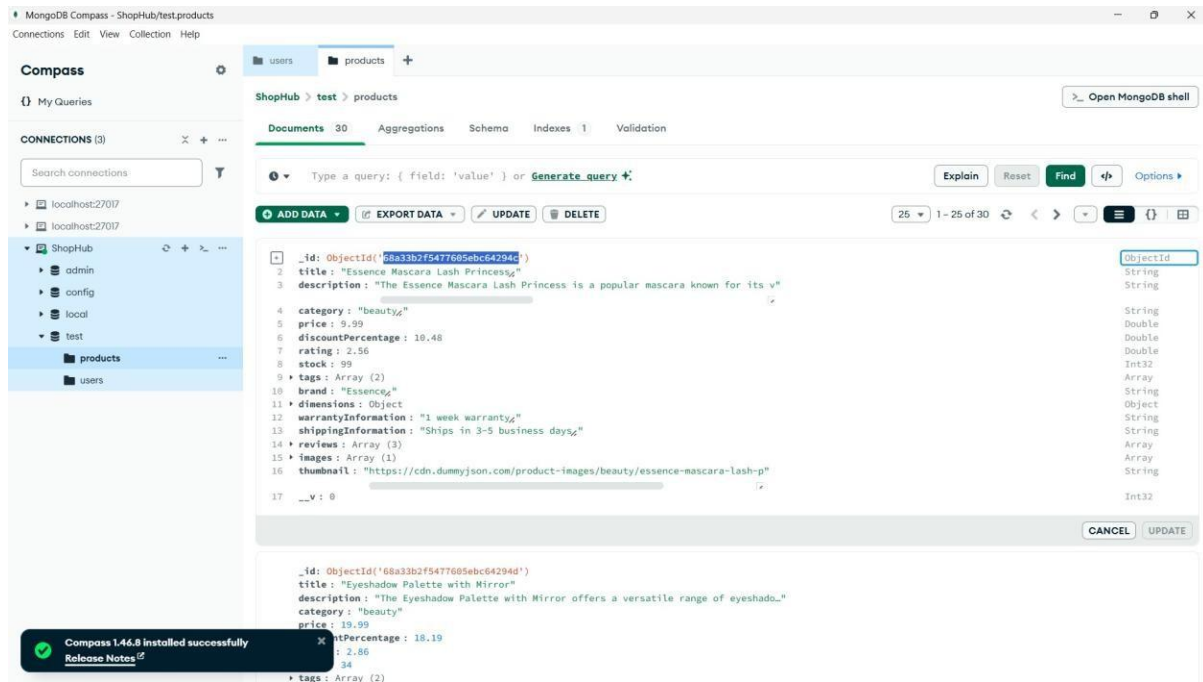
### Schema Structure-

```
{  
  title: String,  
  description: String,  
  category: String,  
  price: Number,  
  discountPercentage: Number,  
  rating: Number,  
  stock: Number,  
  tags: Array,  
  brand: String,  
  dimensions: Object,  
  warrantyInformation: String,  
  shippingInformation: String,  
  reviews: Array,  
  images: Array,  
  thumbnail: String,  
}
```

## Products Collection Screenshot



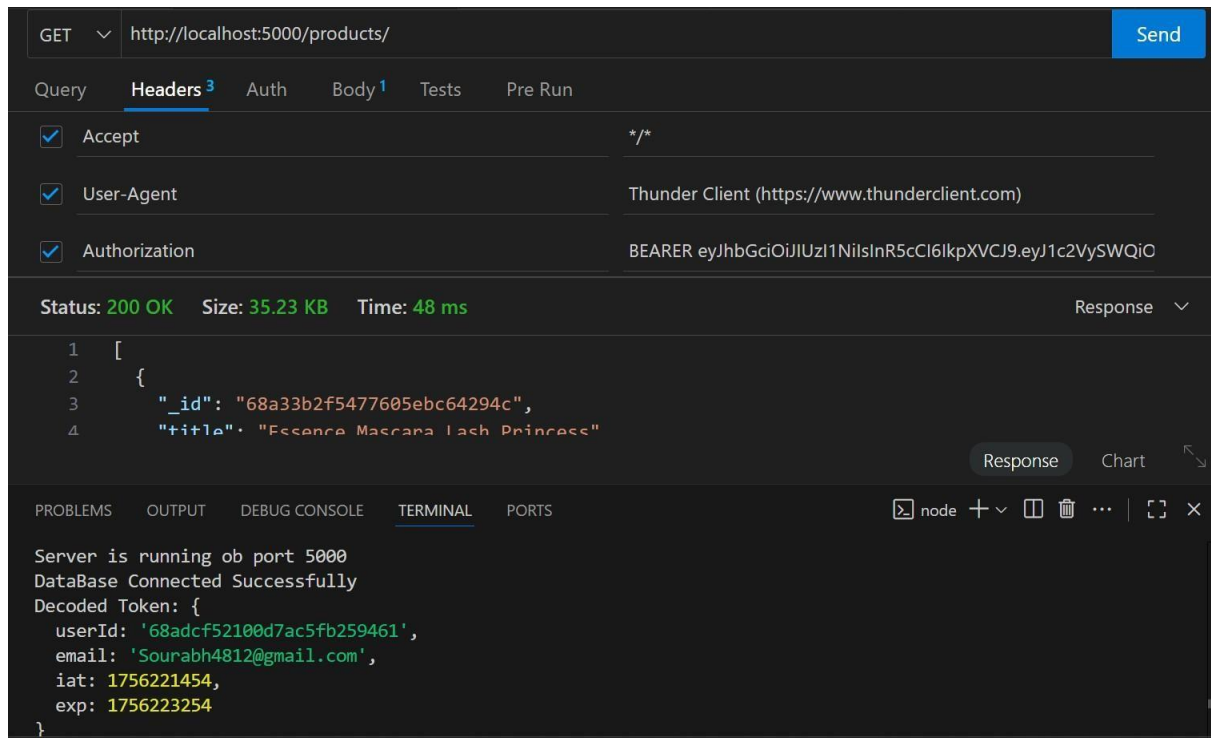
## Users Collection Screenshot



## 5. API Testing with ThunderClient

Screenshots showing request & response for each API:

- Screenshot: GET /products response



This screenshot shows the ThunderClient interface for a GET request to `http://localhost:5000/products/`. The request is configured with the following headers:

- Accept:** \*/\*
- User-Agent:** Thunder Client (https://www.thunderclient.com)
- Authorization:** BEARER eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQ6OiO

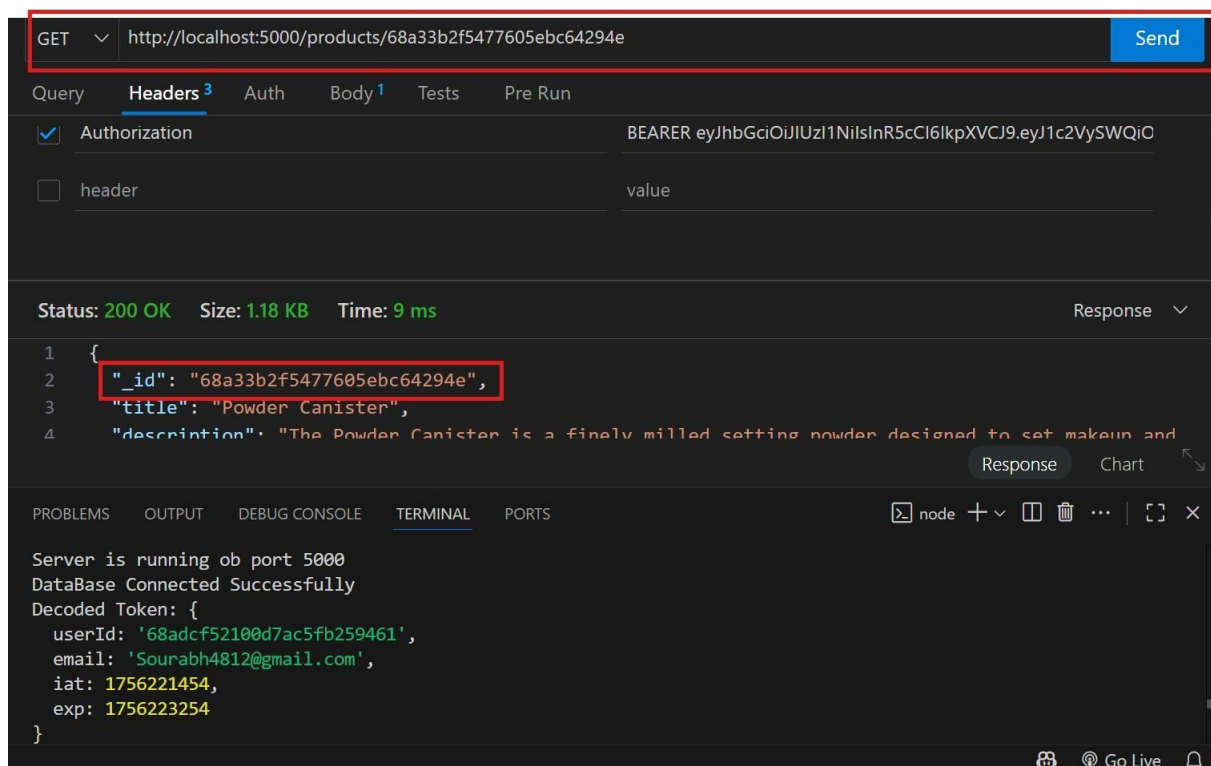
The response status is **200 OK**, with a size of **35.23 KB** and a time of **48 ms**. The response body is a JSON array containing one object:

```
[
  {
    "_id": "68a33b2f5477605ebc64294c",
    "title": "Essence Mascara Lash Princess"
  }
]
```

The terminal at the bottom shows the server is running on port 5000, the database is connected successfully, and the decoded token is:

```
{
  userId: '68adcf52100d7ac5fb259461',
  email: 'Sourabh4812@gmail.com',
  iat: 1756221454,
  exp: 1756223254
}
```

- Screenshot: GET /products/:id response



This screenshot shows the ThunderClient interface for a GET request to `http://localhost:5000/products/68a33b2f5477605ebc64294e`. The request is configured with the following headers:

- Authorization:** BEARER eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQ6OiO

The response status is **200 OK**, with a size of **1.18 KB** and a time of **9 ms**. The response body is a JSON object:

```
{
  "_id": "68a33b2f5477605ebc64294e",
  "title": "Powder Canister",
  "description": "The Powder Canister is a finely milled setting powder designed to set makeup and"
}
```

The terminal at the bottom shows the server is running on port 5000, the database is connected successfully, and the decoded token is:

```
{
  userId: '68adcf52100d7ac5fb259461',
  email: 'Sourabh4812@gmail.com',
  iat: 1756221454,
  exp: 1756223254
}
```

- Screenshot: GET /cart with JWT token

The screenshot shows the Thunder Client interface. The request is a GET to `http://localhost:5000/cart/68ae67c48f62aed93efbe90b`. The Headers tab is active, showing `Accept: */*`, `User-Agent: Thunder Client (https://www.thunderclient.com)`, and `Authorization: BEARER eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQ`. The status is `200 OK`, size is `34 Bytes`, and time is `16 ms`. The response body is `{ "message": "No Items In the cart" }`. The terminal at the bottom shows the JWT token's `iat` and `exp` values.

```
GET http://localhost:5000/cart/68ae67c48f62aed93efbe90b
```

Query Headers Auth Body Tests Pre Run

☒ Accept \*/\*

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

☒ Authorization BEARER eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQ

Status: 200 OK Size: 34 Bytes Time: 16 ms

```
1 {
2   "message": "No Items In the cart"
3 }
```

Response Chart

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
iat: 1756260395,
exp: 1756262195
}
```

- Screenshot: Unauthorized request invalid token

The screenshot shows the Thunder Client interface. The request is a GET to `http://localhost:5000/products/68a33b2f5477605ebc64294c`. The Headers tab is active, showing `Accept: */*`, `User-Agent: Thunder Client (https://www.thunderclient.com)`, and `Authorization: BEARER eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQ`. The status is `200 OK`, size is `13 Bytes`, and time is `4 ms`. The response body is `invalid token`. The terminal at the bottom shows the `nodemon` process running.

```
GET http://localhost:5000/products/68a33b2f5477605ebc64294c
```

Query Headers Auth Body Tests Pre Run

☒ Accept \*/\*

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

☒ Authorization BEARER eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQ

Status: 200 OK Size: 13 Bytes Time: 4 ms

```
1 invalid token
```

Response Preview Chart

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
```

- Screenshot: POST /cart with JWT token

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:5000/cart
- Body (JSON):**

```
{  "userId": "68adf6018121fc193f86865b",  "productId": "68a33b2f5477605ebc64294c",  "quantity": 2}
```
- Status:** 201 Created
- Size:** 253 Bytes
- Time:** 30 ms
- Response (JSON):**

```
{  "userId": "68adf6018121fc193f86865b",  "items": [    {      "productId": "68a33b2f5477605ebc64294c",      "quantity": 2,      "_id": "68adf6678121fc193f868661"    }  ]}
```
- Terminal:** Shows the user ID and email: `userId: '68adf6018121fc193f86865b', email: 'om4213@gmail.com',`

- Screenshot: PUT /cart with JWT token

The screenshot shows a REST client interface with the following details:

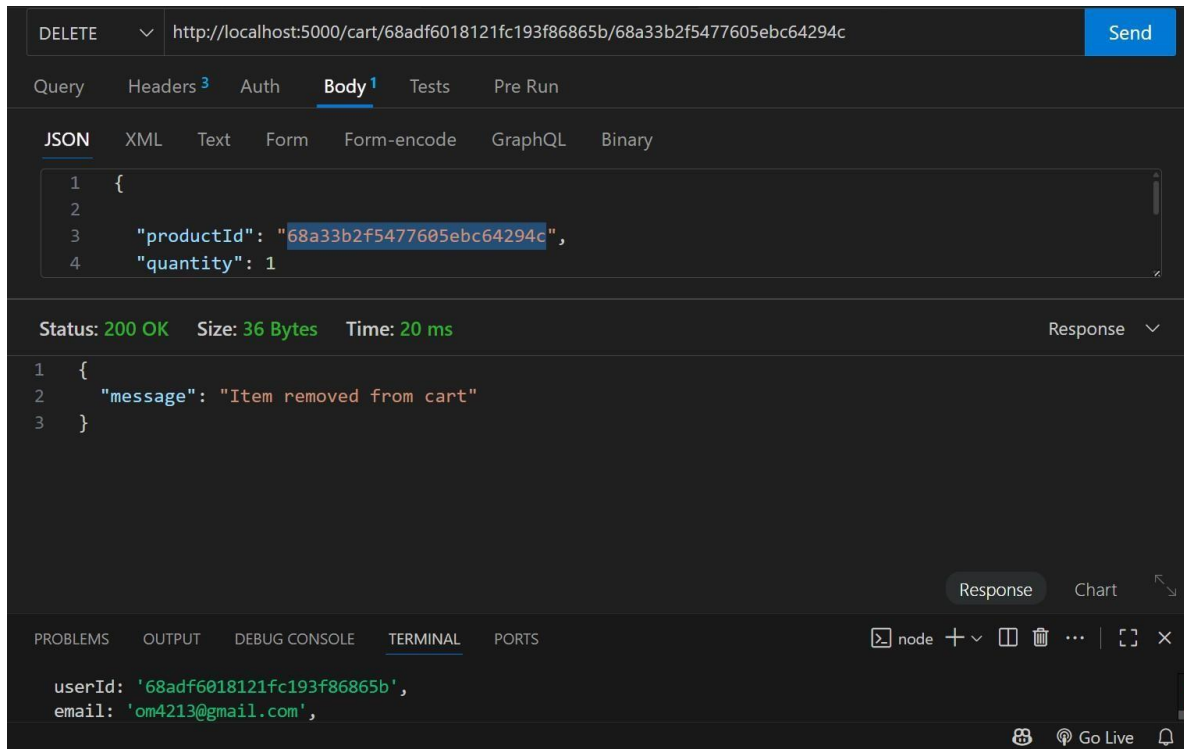
- Method:** PUT
- URL:** http://localhost:5000/cart/68adf6018121fc193f86865b
- Body (JSON):**

```
{  "productId": "68a33b2f5477605ebc64294c",  "quantity": 1}
```
- Status:** 200 OK
- Size:** 253 Bytes
- Time:** 25 ms
- Response (JSON):**

```
{  "_id": "68adf6678121fc193f868660",  "userId": "68adf6018121fc193f86865b",  "items": [    {      "productId": "68a33b2f5477605ebc64294c",      "quantity": 1,      "id": "68adf6678121fc193f868661"    }  ]}
```
- Terminal:** Shows the user ID and email: `userId: '68adf6018121fc193f86865b', email: 'om4213@gmail.com',`

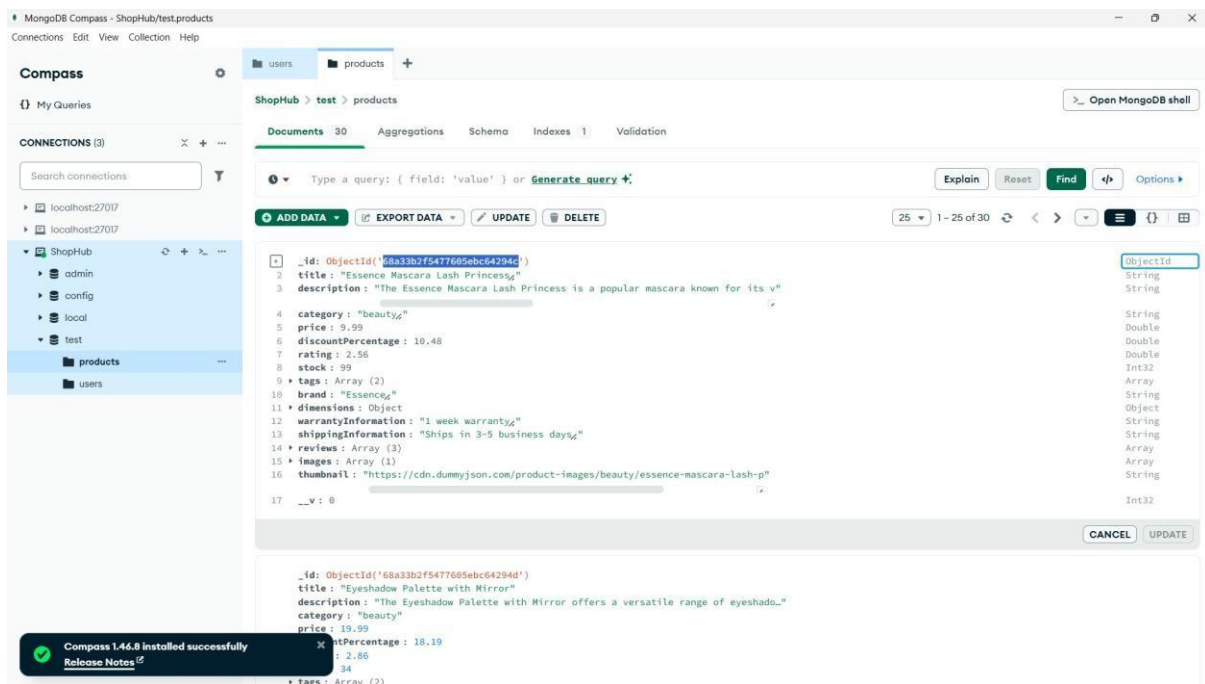


- Screenshot: DELETE/cart with JWT token

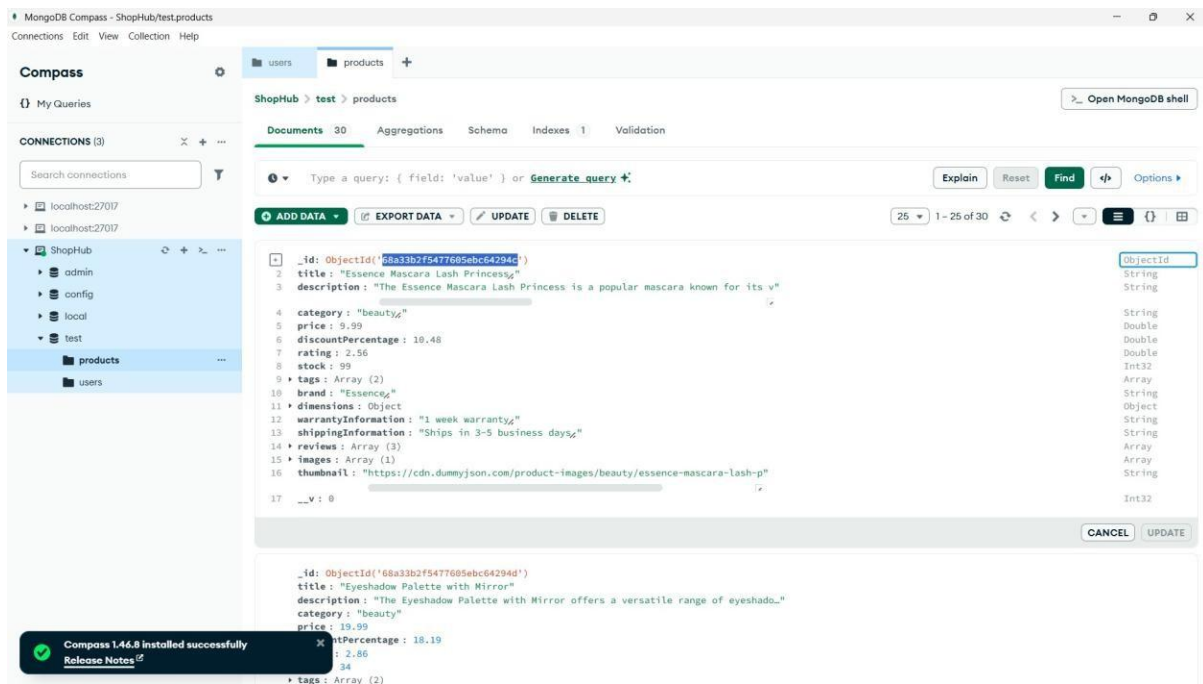


## 6. MongoDB Screenshots

- Screenshot: Product Collection



- **Screenshot: Users collection after registration**



## 7. Error Handling

- Invalid product ID → Returns 404 with error message.

The screenshot shows a Thunder Client interface with a GET request to `http://localhost:5000/products/68a33b2f5477605ebc64294b`. The request headers are: `Accept: */*`, `User-Agent: Thunder Client (https://www.thunderclient.com)`, and `Authorization: BEARER eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOi...`. The response status is **404 Not Found**, with a size of 57 Bytes and a time of 8 ms. The response body is a JSON object: `{ "message": "No Product With Id 68a33b2f5477605ebc64294b" }`. The bottom terminal shows `[nodemon] watching extensions: js,mjs,cjs,json` and `[nodemon] starting 'node server.js'`.

- Unauthorized cart access →

The screenshot shows a Thunder Client interface with a GET request to `http://localhost:5000/products/68a33b2f5477605ebc64294c`. The request headers are: `Accept: */*`, `User-Agent: Thunder Client (https://www.thunderclient.com)`, and `Authorization: BEARER eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOi...`. The response status is **200 OK**, with a size of 13 Bytes and a time of 4 ms. The response body is the text `invalid token`. The bottom terminal shows `[nodemon] watching extensions: js,mjs,cjs,json` and `[nodemon] starting 'node server.js'`.

- **Invalid Password → Returns 404 with error message.**

The screenshot shows a REST client interface with a POST request to `http://localhost:5000/login`. The request body is a JSON object: `{ "email": "vivek@gmail.com", "password": "Vivek@123" }`. The status is `200 OK`, size is `18 Bytes`, and time is `71 ms`. The response body is `1 Incorrect Password`. The terminal at the bottom shows the server is running on port 5000 and the database is connected successfully.

```
POST http://localhost:5000/login
```

Query Headers 2 Auth **Body 1** Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

```
1 {
2   "email" : "vivek@gmail.com",
3   "password": "Vivek@123"
4 }
5
```

Status: 200 OK Size: 18 Bytes Time: 71 ms Response ▾

1 Incorrect Password

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

node + ▾ □ 🗑️ ⋮ | [ ] ×

Server is running ob port 5000  
DataBase Connected Successfully  
□

- **Minimum Password Length Should be 8→**

The screenshot shows a REST client interface with a POST request to `http://localhost:5000/register`. The request body is a JSON object: `{ "username": "Vivek Pawar", "email": "vivek@gmail.com", "password": "Viv" }`. The status is `200 OK`, size is `53 Bytes`, and time is `1.77 s`. The response body is a JSON object: `{ "message": "Password Must be min 8 characters long!" }`. The terminal at the bottom shows the server is running on port 5000 and the database is connected successfully.

```
POST http://localhost:5000/register
```

Query Headers 2 Auth **Body 1** Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

```
1 {
2   "username": "Vivek Pawar",
3   "email" : "vivek@gmail.com",
4   "password": "Viv"
5 }
```

Status: 200 OK Size: 53 Bytes Time: 1.77 s Response ▾

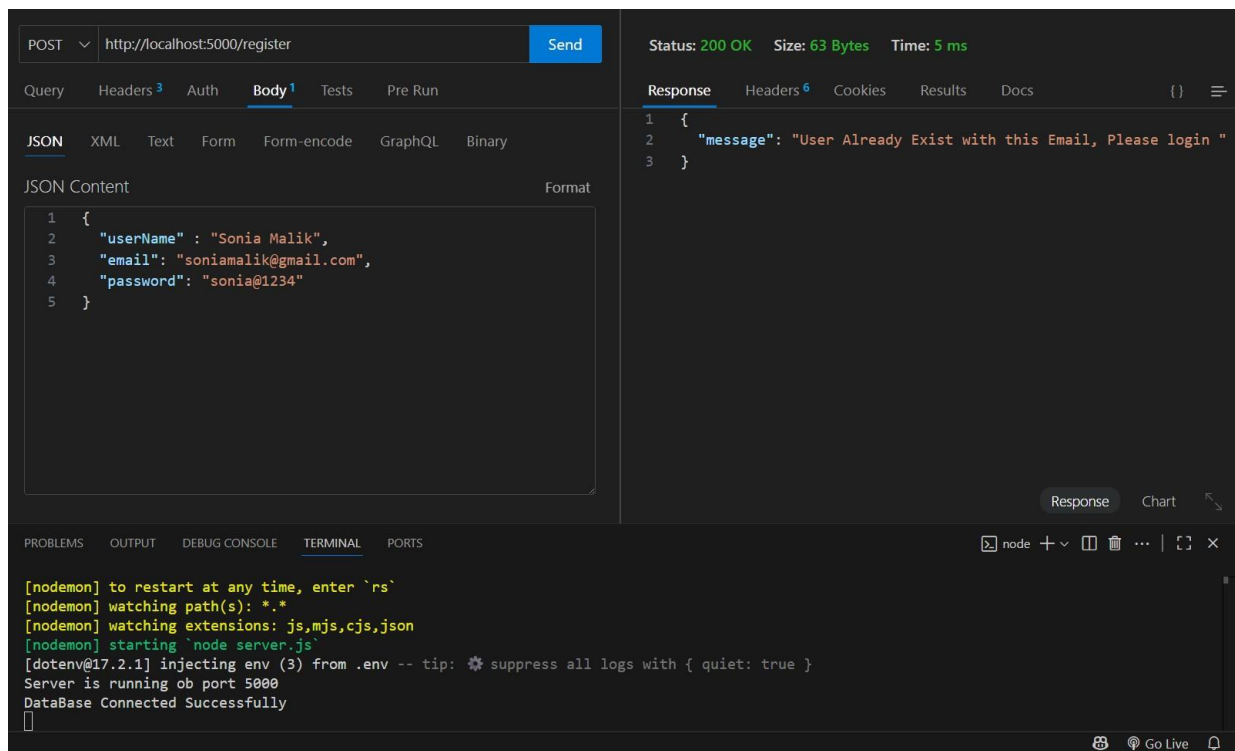
```
1 {
2   "message": "Password Must be min 8 characters long!"
3 }
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

node + ▾ □ 🗑️ ⋮ | [ ] ×

Server is running ob port 5000  
DataBase Connected Successfully  
□

- Existing User→.



## 8. How to Run

### 1. Clone repository

```
git clone https://github.com/Prajwal1412/backend
```

```
cd backend
```

### 2. Install dependencies

```
npm install
```

### 3. Set environment variables in .env

```
PORT=<port number>
```

```
MONGO_URI=<your_mongodb_connection_string>
```

```
SECRET_KEY=<your_secret_key>
```

### 4. Start Server

```
npm start
```