

# 1. Project Overview

This project is a complete **E-commerce Backend API** built using **Node.js, Express, and MongoDB Atlas**, with **JWT authentication and Cart functionality**.

**With this API, we can:**

- Register & Login Users securely
- Fetch products from database
- Add products to cart
- Update quantity in cart
- Remove product from cart
- Get user's entire cart

All cart routes are **protected and require JWT authentication**.

**Smart features used:**

- Password encryption (bcrypt)
  - Token-based authentication (JWT)
- 

# 2. Middleware Overview

## a) Authentication Middleware

This middleware protects cart routes. It verifies the token sent in the request header:

- ✓ If the token is valid, it allows the request to proceed
- ✗ If invalid/missing, it returns **401 Unauthorized**

**Why useful?**

- Ensures only logged-in users can add/view cart items.
- Prevents unauthorized access to private data.

## b) Error Handling Middleware

There are two error handlers:

**notFoundHandler**

Used when a route does not exist.

"Route /unknown not found"

### 3. API Routes Explanation

#### User Authentication Routes

#### Post/api/register - Register New User

The screenshot shows the Postman interface with a successful user registration. The left sidebar lists collections like Blog APIs, Cart, Product, and User. In the main workspace, a POST request to `http://localhost:3000/api/register` is made with JSON body:

```
{
  "name": "Sangeeta34512",
  "email": "abc@gmail.com",
  "password": "67899"
}
```

The response is a 201 Created status with the message: "User registered successfully".

#### Validate Proper Email

The screenshot shows the Postman interface with an email validation error. The left sidebar lists collections like Blog APIs, Cart, Product, and User. In the main workspace, a POST request to `http://localhost:3000/api/register` is made with JSON body:

```
{
  "name": "Sangeeta34512",
  "email": "abc@gmail.com",
  "password": "67899"
}
```

The response is a 409 Conflict status with the message: "Email already in use".

## Post/api/login -user login

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:3000/api/login`. The response status is `200 OK` with a response time of `374 ms` and a size of `543 B`. The response body is:

```
1 {
2   "message": "Login successful",
3   "user": {
4     "name": "Sangeeta34512",
5     "email": "abc@gmail.com"
6   },
7   "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjY5Mzc1ODRiYjdjNmIwZWZjYTAwZTgxYSIsImhdCI6MTC2NTIzNTY5MiwiZXhwIjoxNzY1ODQwNC"
8 }
```

## If wrong password

The screenshot shows the Postman interface with an unauthorized response. The URL is `http://localhost:3000/api/login`. The response status is `401 Unauthorized` with a response time of `638 ms` and a size of `309 B`. The response body is:

```
1 {
2   "message": "Incorrect password"
3 }
```

## Product Routes

### Get /api/products

My Workspace

Product / Get Product

GET http://localhost:3000/api/products

Params: Authorization, Headers (6), Body, Scripts, Tests, Settings

Body (Pretty):

```

24:   [
25:     {
26:       "brand": "apple",
27:       "category": "Mobiles",
28:       "thumbnail": "https://example.com/img.jpg",
29:       "createdAt": "2025-12-08T23:00:15.400Z",
30:       "updatedAt": "2025-12-08T23:00:15.400Z",
31:       "_v": 0
32:     },
33:     {
34:       "_id": "69375922b7c6b0efca00e823",
35:       "title": "iPhone Pro",
36:       "description": "Latest flagship phone",
37:       "price": 129999,
38:       "stockQuantity": 10,
39:       "rating": 0,
40:       "brand": "apple",
41:       "category": "Mobiles",
42:       "thumbnail": "https://example.com/img.jpg",
43:       "createdAt": "2025-12-08T23:02:58.973Z",
44:       "updatedAt": "2025-12-08T23:02:58.973Z",
45:       "_v": 0
46:     }
  ]

```

200 OK | 91 ms | 1.17 KB | Save Response | ...

Postbot | Runner | Start Proxy | Cookies | Vault | Trash | 4:59 AM | 12/9/2025

## Fetch Single Product By id

Get /api/product/:id

My Workspace

Product / singleProduct

GET http://localhost:3000/api/product/69375922b7c6b0efca00e823

Params: Authorization, Headers (6), Body, Scripts, Tests, Settings

Body (Pretty):

```

1: {
2:   "product": {
3:     "_id": "69375922b7c6b0efca00e823",
4:     "title": "iPhone Pro",
5:     "description": "Latest flagship phone",
6:     "price": 129999,
7:     "stockQuantity": 10,
8:     "rating": 0,
9:     "brand": "apple",
10:    "category": "Mobiles",
11:    "thumbnail": "https://example.com/img.jpg",
12:    "createdAt": "2025-12-08T23:02:58.973Z",
13:    "updatedAt": "2025-12-08T23:02:58.973Z",
14:    "_v": 0
15:  }
16: }

```

200 OK | 95 ms | 583 B | Save Response | ...

Postbot | Runner | Start Proxy | Cookies | Vault | Trash | 5:00 AM | 12/9/2025

## Create Products

Post /api/products

## Cart Routes

### Post/api/cart -(product Add to cart)

# Update Product Quantity Cart

## Put/api/cart /productid

The screenshot shows the Postman interface with the following details:

- Collection:** My Workspace
- Request:** Cart / Update Cart
- Method:** PUT
- URL:** http://localhost:3000/api/cart/69375922b7c6b0efca00e823
- Body (JSON):**

```
1 {
2   "quantity": 10
3 }
```
- Response Status:** 200 OK
- Response Body (Pretty JSON):**

```
1 {
2   "_id": "69375942b7c6b0efca00e82b",
3   "user": "6937584bb7c6b0efca00e81a",
4   "items": [
5     {
6       "product": "69375922b7c6b0efca00e823",
7       "quantity": 10,
8       "_id": "69375942b7c6b0efca00e82d"
9     }
10   ],
11   "createdAt": "2025-12-08T23:03:30.591Z",
12   "updatedAt": "2025-12-08T23:07:16.794Z",
13   "__v": 1
14 }
```

# Get Product Quantity Cart

## get/api/cart

The screenshot shows the Postman interface with the following details:

- Collection:** My Workspace
- Request:** Cart / Get Cart
- Method:** GET
- URL:** http://localhost:3000/api/cart
- Headers:** Authorization (with token)
- Response Status:** 200 OK
- Response Body (Pretty JSON):**

```
1 {
2   "_id": "69375942b7c6b0efca00e82b",
3   "user": "6937584bb7c6b0efca00e81a",
4   "items": [
5     {
6       "product": {
7         "_id": "69375922b7c6b0efca00e823",
8         "title": "iPhone Pro",
9         "description": "Latest flagship phone",
10        "price": 12999,
11        "stockQuantity": 10,
12        "rating": 0,
13        "brand": "apple",
14        "category": "Mobiles",
15        "thumbnail": "https://example.com/img.jpg",
16        "createdAt": "2025-12-08T23:02:58.973Z",
17        "updatedAt": "2025-12-08T23:02:58.973Z",
18        "__v": 0
19      },
20      "quantity": 10,
21      "_id": "69375942b7c6b0efca00e82d"
22    }
23  ]
24 }
```

# Delete Product Cart

## Delete/api/cart

My Workspace

Cart / Delete Product Cart

DELETE http://localhost:3000/api/cart/69375922b7c6b0efca00e823

Headers (7)

Key	Value	Description
Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9eyJpZC... Key	
		Description

Body (Pretty)

```
{
  "_id": "69375942b7c6b0efca00e82b",
  "user": "6937584bb7c6b0efca00e81a",
  "items": [],
  "createdAt": "2025-12-08T23:03:30.591Z",
  "updatedAt": "2025-12-08T23:12:30.991Z",
  "__v": 2
}
```

200 OK 272 ms 433 B Save Response

## 4. Error Handling

### If path not found -404

My Workspace

Cart / Get Cart

GET http://localhost:3000/keeji

Headers (7)

Key	Value	Description
Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9eyJpZC... Key	
		Description

404 Not Found 11 ms 372 B Save Response

Body (Pretty)

```
{"success": false, "error": "Not Found", "message": "The requested route 'GET /keeji' does not exist."}
```

.proper status code

**400 (Bad Request)**

**401 (Unauthorized)**

**404 (Not Found)**

# 500 (Internal Server Error)

## //user data

The screenshot shows the MongoDB Compass interface for a cluster named 'Cluster0'. The left sidebar has 'Data Explorer' selected. In the main area, under the 'test' database, the 'users' collection is selected. The 'Find' tab is active. A query is run, resulting in two documents:

```
_id: ObjectId('6936d9f90913e983176fa92e')
name: "Sangeeta34512"
email: "abc@gmail.com"
password: "$2b$12$yVY3ZR92.LSSuZIsHk1dK0OKYOUl0sV0zq1QGXHhpCz/SJdqkfj."
createdAt: 2025-12-08T14:00:25.073+00:00
updatedAt: 2025-12-08T14:00:25.073+00:00
__v: 0

_id: ObjectId('6937584bb7c6b0efca00e81a')
name: "Sangeeta34512"
email: "abc@gmail.com"
password: "$2b$12$RgCgzmzq9JQgKavh06TReI0j2A0qr0jdVpOvCoTwe6k8oRdKP.a"
createdAt: 2025-12-08T14:00:25.073+00:00
updatedAt: 2025-12-08T14:00:25.073+00:00
__v: 0
```

The URL in the address bar is <https://cloud.mongodb.com/v2/6936c3dedfc6cae5f35ed99#/metrics/replicaSet/6936c469c0ab315153fab3fe/explorer/test/users/find>.

## //product

The screenshot shows the MongoDB Compass interface for a cluster named 'Cluster0'. The left sidebar has 'Data Explorer' selected. In the main area, under the 'test' database, the 'products' collection is selected. The 'Find' tab is active. A query is run, resulting in three documents:

```
_id: ObjectId('6936dd76959f06ec6625bc7b')
title: "iPhone 16 Pro"
description: "Latest flagship phone"
price: 12999
stockQuantity: 10
rating: 0
brand: "apple"
category: "Mobiles"
thumbnail: "https://example.com/img.jpg"
createdAt: 2025-12-08T14:15:18.699+00:00
updatedAt: 2025-12-08T14:15:18.699+00:00
__v: 0

_id: ObjectId('6936dd76959f06ec6625bc7b')
title: "iPhone 16 Pro"
description: "Latest flagship phone"
price: 12999
stockQuantity: 10
rating: 0
brand: "apple"
category: "Mobiles"
thumbnail: "https://example.com/img.jpg"
createdAt: 2025-12-08T14:15:18.699+00:00
updatedAt: 2025-12-08T14:15:18.699+00:00
__v: 0

_id: ObjectId('6936dd76959f06ec6625bc7b')
title: "iPhone 16 Pro"
description: "Latest flagship phone"
price: 12999
stockQuantity: 10
rating: 0
brand: "apple"
category: "Mobiles"
thumbnail: "https://example.com/img.jpg"
createdAt: 2025-12-08T14:15:18.699+00:00
updatedAt: 2025-12-08T14:15:18.699+00:00
__v: 0
```

The URL in the address bar is <https://cloud.mongodb.com/v2/6936c3dedfc6cae5f35ed99#/metrics/replicaSet/6936c469c0ab315153fab3fe/explorer/test/products/find>.