

RESTful API Assignment Submission

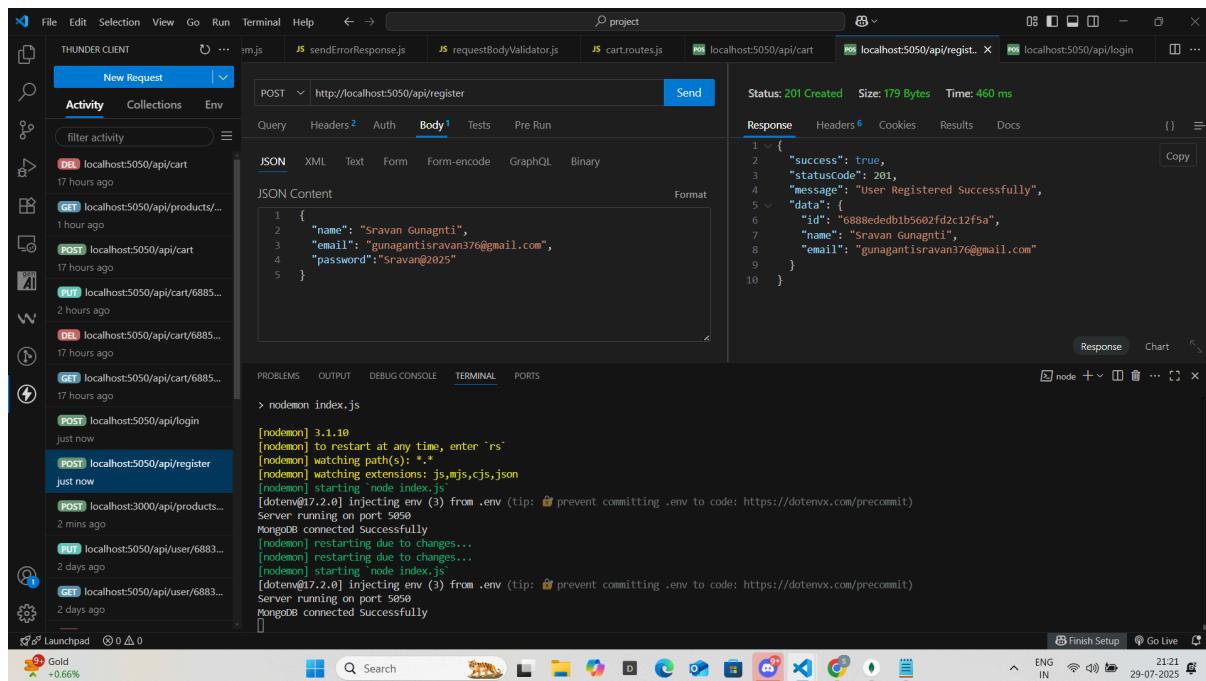
Project: ShoppyGlobe - RESTful API using Node.js, Express, MongoDB

1. POST /api/register

- **Endpoint:** POST <http://localhost:5050/api/register>
- **Description:** Registers a new user.
- **Request Body:**

```
{  
  "name": "Sravan Gunaganti",  
  "email": "gunagantisravan376@gmail.com",  
  "password": "Sravan@2025"  
}
```

- **ThunderClient Screenshot:**

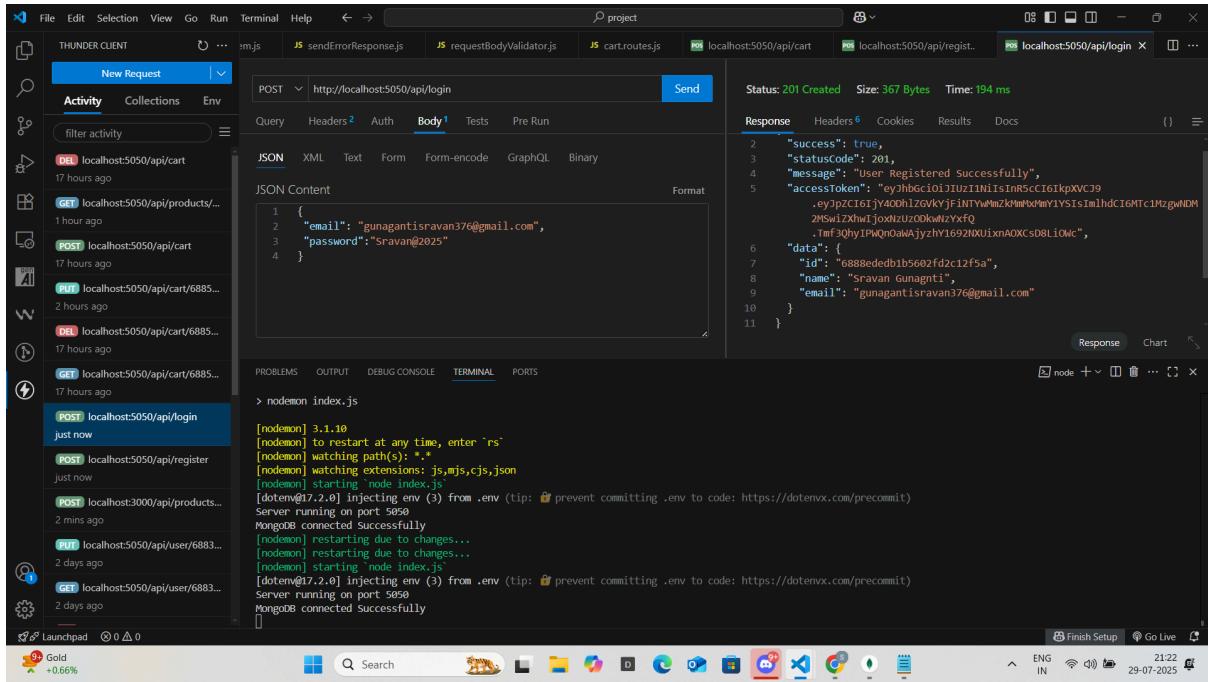


2. POST /api/login

- **Endpoint:** POST <http://localhost:5050/api/login>
- **Description:** Authenticates user and returns token.
- **Request Body:**

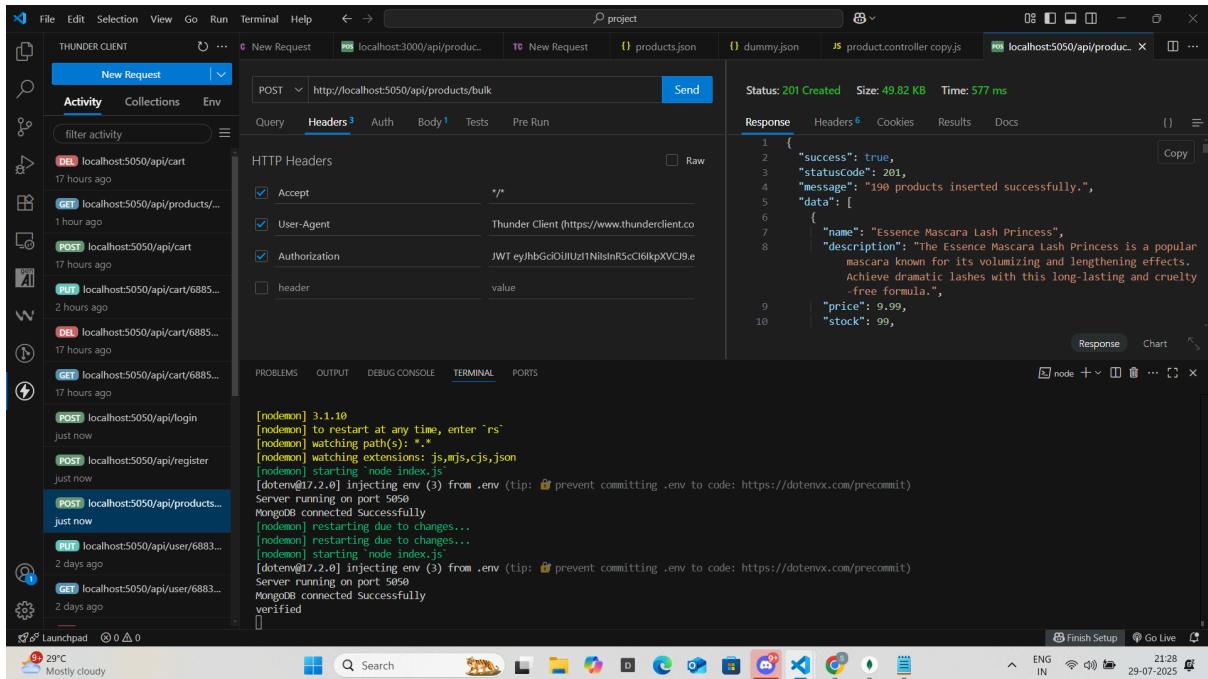
```
{  
  "email": "gunagantisravan376@gmail.com",  
  "password": "Sravan@2025"  
}
```

- **ThunderClient Screenshot:**



3. POST /api/products/bulk

- **Endpoint:** POST `http://localhost:5050/api/products/bulk`
 - **Description:** Inserts multiple products.
 - **Headers:** JWT token required.
 - **Request Body:** Array of products provided in project folder also
 - **ThunderClient Screenshot:**



4. POST /api/product

- **Endpoint:** POST <http://localhost:5050/api/product>
- **Description:** Adds a single product.
- **Headers:** JWT token required.
- **ThunderClient Screenshot:**

The screenshot shows the ThunderClient interface. In the left sidebar, there's a list of recent requests. The main area shows a POST request to `http://localhost:5050/api/product`. The Body tab contains the following JSON payload:

```
1 {
2   "name": "Keyboard",
3   "description": "Mechanical keyboard With Backlight Keyword",
4   "price": 3000,
5   "stock": 20
6 }
```

The Response tab shows the server's response:

```
1 {
2   "success": true,
3   "statusCode": 201,
4   "message": "Product Created Successfully",
5   "data": {
6     "name": "Keyboard",
7     "description": "Mechanical keyboard With Backlight Keyword",
8     "price": 3000,
9     "stock": 20,
10    "_id": "6888f04fb1b5602fd2c1301f",
11    "__v": 0
12  }
13 }
```

The Terminal tab shows the Node.js application logs:

```
[nodemon] to restart at any time, enter `rs`  
[nodemon] watching path(s): *.*  
[nodemon] watching extensions: js,mjs,cjs,json  
[nodemon] starting 'node index.js'  
[dotenv@17.2.0] injecting env (3) from .env (tip: ⚡ prevent committing .env to code: https://dotenvx.com/precommit)  
Server running on port 5050  
MongoDB connected Successfully  
[nodemon] restarting due to changes...  
[nodemon] restarting due to changes...  
[nodemon] starting 'node index.js'  
[dotenv@17.2.0] injecting env (3) from .env (tip: ⚡ prevent committing .env to code: https://dotenvx.com/precommit)  
Server running on port 5050  
MongoDB connected Successfully  
verified  
verified  
verified  
[nodemon] to restart at any time, enter `rs`  
[nodemon] watching path(s): *.*  
[nodemon] watching extensions: js,mjs,cjs,json  
[nodemon] starting 'node index.js'  
[dotenv@17.2.0] injecting env (3) from .env (tip: ⚡ prevent committing .env to code: https://dotenvx.com/precommit)  
Server running on port 5050  
MongoDB connected Successfully  
verified  
verified  
verified
```

5. GET /api/products

- **Endpoint 1:** GET <http://localhost:5050/api/products>
- **Endpoint 1:** GET <http://localhost:5050/api/products?limit=number>
- **Description:** Fetches all products with limit query parameter .by default limit is 30
- **ThunderClient Screenshot:**

The screenshot shows the ThunderClient interface. In the left sidebar, there's a list of recent requests. The main area shows a GET request to `http://localhost:5050/api/products`. The Query Parameters tab shows an empty table with one row: `parameter` and `value`. The Response tab shows the server's response:

```
1 {
2   "success": true,
3   "statusCode": 200,
4   "data": [
5     {
6       "_id": "6888e0ccbf5f086297e158c",
7       "name": "Essence Mascara Lash Princess",
8       "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.",
9       "price": 9.99,
10      "stock": 99
11    }
12  ]
13 }
```

The Terminal tab shows the Node.js application logs:

```
[nodemon] to restart at any time, enter `rs`  
[nodemon] watching path(s): *.*  
[nodemon] watching extensions: js,mjs,cjs,json  
[nodemon] starting 'node index.js'  
[dotenv@17.2.0] injecting env (3) from .env (tip: ⚡ prevent committing .env to code: https://dotenvx.com/precommit)  
Server running on port 5050  
MongoDB connected Successfully  
[nodemon] restarting due to changes...  
[nodemon] restarting due to changes...  
[nodemon] starting 'node index.js'  
[dotenv@17.2.0] injecting env (3) from .env (tip: ⚡ prevent committing .env to code: https://dotenvx.com/precommit)  
Server running on port 5050  
MongoDB connected Successfully  
verified  
verified  
verified  
[nodemon] to restart at any time, enter `rs`  
[nodemon] watching path(s): *.*  
[nodemon] watching extensions: js,mjs,cjs,json  
[nodemon] starting 'node index.js'  
[dotenv@17.2.0] injecting env (3) from .env (tip: ⚡ prevent committing .env to code: https://dotenvx.com/precommit)  
Server running on port 5050  
MongoDB connected Successfully  
verified  
verified  
verified
```

Status: 200 OK Size: 37.48 KB Time: 167 ms

```

1  {
2    "success": true,
3    "statusCode": 200,
4    "data": [
5      {
6        "_id": "6888e09ccbf5086297e158c",
7        "name": "Essence Mascara Lash Princess",
8        "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.",
9        "price": 9.99,
10       "stock": 99

```

6. GET /api/products/:id

- Endpoint:** GET `http://localhost:5050/api/products/<productId>`
- Description:** Fetch product by ID.
- ThunderClient Screenshot:**

Status: 200 OK Size: 176 Bytes Time: 506 ms

```

1  {
2    "success": true,
3    "statusCode": 200,
4    "data": {
5      "_id": "6888f04fb1b5602fd2c1301f",
6      "name": "Keyboard",
7      "description": "Mechanical keyboard With Backlight Keyword",
8      "price": 3000,
9      "stock": 20
10     }
11   }

```

7. PUT /api/products/:id

- **Endpoint:** `PUT http://localhost:5050/api/products/6888f04fb1b5602fd2c1301f`
- **Headers:** JWT token required.
- **Request Body:**
- **ThunderClient Screenshot:**

The screenshot shows the ThunderClient interface. In the top navigation bar, the URL is set to `PUT http://localhost:5050/api/products/6888f04fb1b5602fd2c1301f`. The "Body" tab is selected, displaying a JSON payload:

```
1 {
  2   "name": "Mechanical Keyboard",
  3   "description": "Mechanical keyboard With Backlight Keyword",
  4   "price": 3010,
  5   "stock": 15
}
```

The "Response" tab shows the server's response:

```
1 {
  2   "success": true,
  3   "statusCode": 200,
  4   "message": "Product Updated Successfully",
  5   "data": {
  6     "_id": "6888f04fb1b5602fd2c1301f",
  7     "name": "Mechanical Keyboard",
  8     "description": "Mechanical keyboard With Backlight Keyword",
  9     "price": 3010,
 10     "stock": 15,
 11     "__v": 0
 12   }
 13 }
```

The "Terminal" tab shows the Node.js logs indicating the product was updated successfully.

8. DELETE /api/products/:id

- **Endpoint:** `DELETE http://localhost:5050/api/products/6888f04fb1b5602fd2c1301f`
- **Headers:** JWT token required.
- **ThunderClient Screenshot:**

The screenshot shows the ThunderClient interface. In the top navigation bar, the URL is set to `DELETE http://localhost:5050/api/products/6888f04fb1b5602fd2c1301f`. The "Headers" tab is selected, showing the following headers:

Header	Value
Accept	/*
User-Agent	Thunder Client (https://www.thunderclient.co)
Authorization	JWT eyJhbGciOiUzI1niInR5Cl6lkpXVC19.e
header	value

The "Response" tab shows the server's response:

```
1 {
  2   "success": true,
  3   "statusCode": 200,
  4   "message": "Product Deleted Successfully",
  5   "data": {
  6     "_id": "6888f04fb1b5602fd2c1301f",
  7     "name": "Mechanical Keyboard",
  8     "description": "Mechanical keyboard With Backlight Keyword",
  9     "price": 3010,
 10     "stock": 15,
 11     "__v": 0
 12   }
 13 }
```

The "Terminal" tab shows the Node.js logs indicating the product was deleted successfully.

9. POST /api/cart

- Endpoint:** POST <http://localhost:5050/api/cart>
- Headers:** JWT token required
- Description:** Adds product to cart or increments quantity for existing and returns populated cart
- ThunderClient Screenshot:**

The screenshot shows the ThunderClient interface with a POST request to `http://localhost:5050/api/cart`. The response is successful (200 OK) with a size of 881 bytes and a duration of 98 ms. The response body is a JSON object representing a product in a cart:

```
1 {
  "productId": "6888f61cbc92c7e91895f50d"
}
```

10. GET /api/cart

- Endpoint:** GET <http://localhost:5050/api/cart>
- Headers:** JWT token required.
- Description:** Fetch user's cart by userId which get by verifying token
- ThunderClient Screenshot:**

The screenshot shows the ThunderClient interface with a GET request to `http://localhost:5050/api/cart`. The response is successful (200 OK) with a size of 881 bytes and a duration of 419 ms. The response body is a JSON object indicating success and returning the user's cart data:

```
1 {
  "success": true,
  "statusCode": 200,
  "data": {
    "userId": "6888edeb1b5602fd2c12f9a",
    "products": [
      {
        "productId": {
          "id": "6888f61cbc92c7e91895f4d8",
          "name": "Rolex Cellini Moonphase",
          "description": "The Rolex Cellini Moonphase is a masterpiece of horology, featuring a moon phase complication and exquisite design. It reflects Rolex's commitment to precision and elegance.",
          "price": 12999.99,
          "stock": 36
        },
        "quantity": 2,
        "id": "6888f61cbc92c7e91895f50d",
        "name": "Tennis Ball",
        "description": "The Tennis Ball is a standard ball used in the sport of tennis. It is designed for bouncing and hitting with tennis rackets during matches or practice sessions."
      }
    ]
  }
}
```

11. PUT /api/cart

- Endpoint:** PUT `http://localhost:5050/api/cart`
- Headers:** JWT token required.
- Description:** Updates cart item quantity.
- ThunderClient Screenshot:**

The screenshot shows the ThunderClient interface. In the left sidebar, there is a list of activity logs. The main area shows a 'PUT' request to `http://localhost:5050/api/cart`. The 'Body' tab contains the following JSON payload:

```
1 {
2   "productId": "6888f61cbc92c7e91895f4d8",
3   "quantity": 4
4 }
```

The response pane shows a successful 200 OK status with a response body containing updated cart data. The response JSON includes product details like name, description, price, and stock level, along with the updated quantity of 4.

12. DELETE /api/cart/:productId

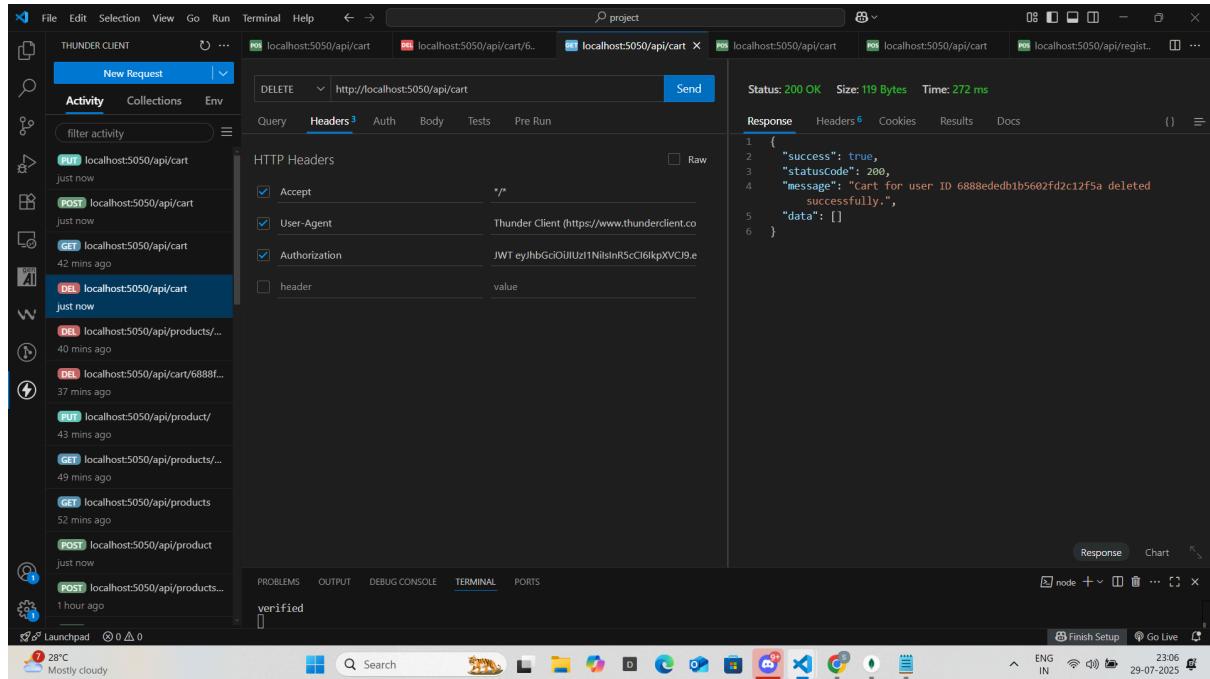
- Endpoint:** DELETE `http://localhost:5050/api/cart/6888f61cbc92c7e91895f4d8`
- Headers:** JWT token required.
- Description:** Removes product from cart.
- ThunderClient Screenshot:**

The screenshot shows the ThunderClient interface. In the left sidebar, there is a list of activity logs. The main area shows a 'DELETE' request to `http://localhost:5050/api/cart/6888f61cbc92c7e91895f4d8`. The 'Body' tab contains the number 1.

The response pane shows a successful 200 OK status with a response body indicating the item was deleted successfully. The response JSON includes the deleted product's details and a message confirming the deletion.

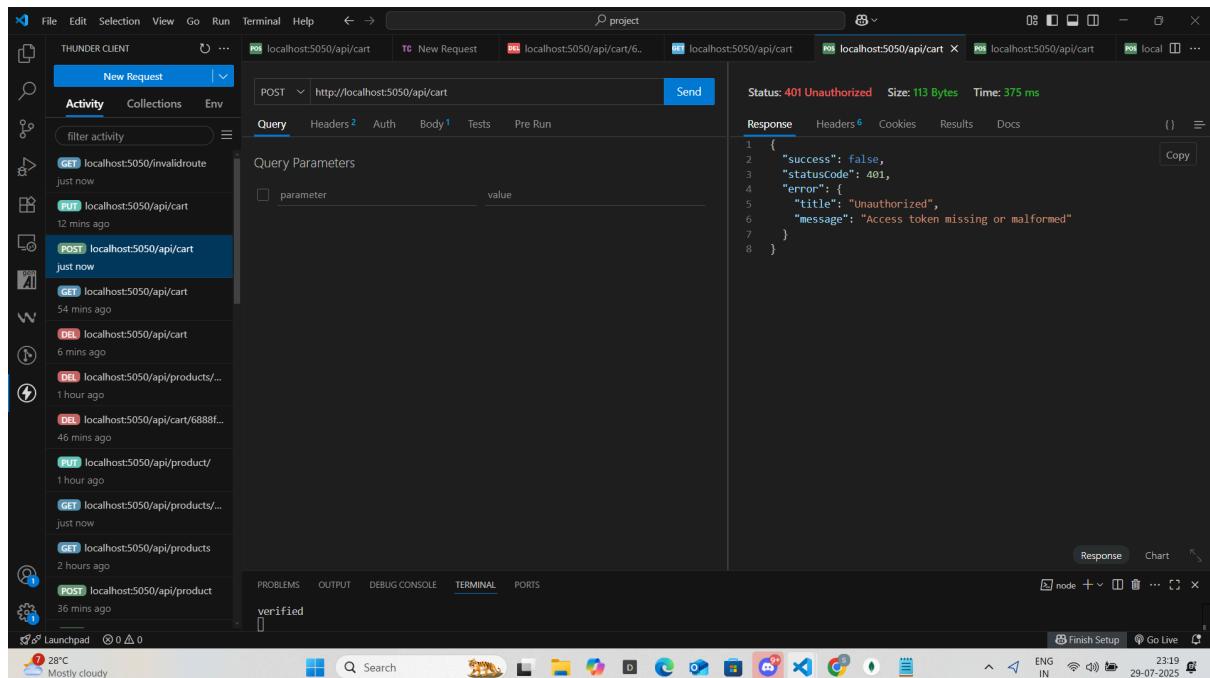
13. DELETE /api/cart

- Endpoint:** `DELETE http://localhost:5050/api/cart`
- Headers:** Bearer token required.
- Description:** Clears entire cart.
- ThunderClient Screenshot:**



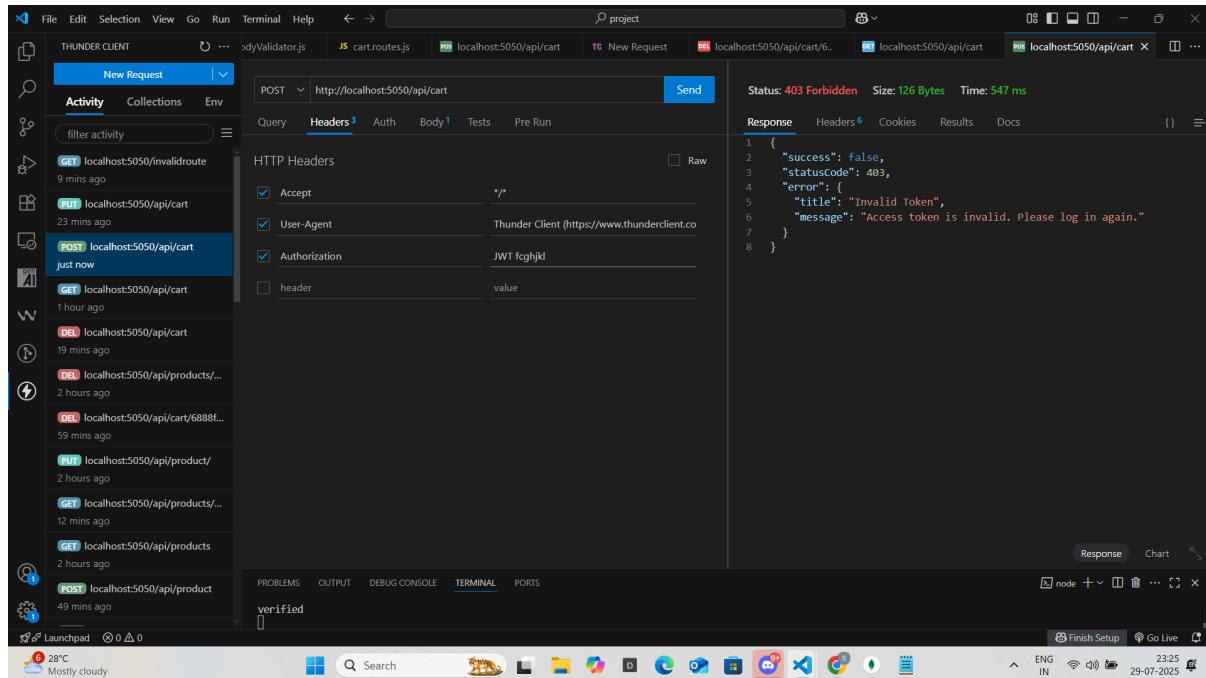
14. Access Protected Route without headers

- Endpoint:** `POST http://localhost:5050/api/products/bulk`
- Headers:** None
- Expected:** 401 Unauthorized
- Thunderclient screenshot:**



15. Access Protected Route with invalidToken

- **Endpoint:** POST `http://localhost:5050/api/products/bulk`
- **Headers:** Authorization: JWT `fcghjkl`
- **Expected:** 401 Unauthorized
- **Thunderclient screenshot:**



16. POST Register with existing email

- **Endpoint:** POST `http://localhost:5050/api/register`
- **Expected:** 409 conflict -Email already registered

Activity

- POST localhost:5050/api/register
- POST localhost:5050/api/cart/6885...
- POST localhost:5050/api/login
- POST localhost:5050/api/register
- PUT localhost:5050/api/cart/6885...
- PUT localhost:3000/api/products...
- POST localhost:3000/api/products...
- DELETE localhost:3000/api/products/...
- DELETE localhost:5050/api/cart
- GET localhost:5050/api/cart/6885...
- Launchpad

Body

```
{
  "name": "Sravan Gunaganti",
  "email": "gunagantisravan376@gmail.com",
  "password": "Sravan@2025"
}
```

Response

```
{
  "success": false,
  "statusCode": 409,
  "error": {
    "title": "Email already registered",
    "message": "A user with this email already exists. Please login instead."
  }
}
```

17. POST /api/login

- Endpoint:** POST `http://localhost:5050/api/login`
- Expected:** 401-Unauthorized with Invalid credentials
- ThunderClient Screenshot:**

Activity

- POST localhost:5050/api/login
- POST localhost:5050/api/cart/6885...
- POST localhost:5050/api/login just now
- POST localhost:5050/api/register
- PUT localhost:5050/api/cart/6885...
- PUT localhost:3000/api/products...
- POST localhost:3000/api/products...
- DELETE localhost:3000/api/products/...
- DELETE localhost:5050/api/cart
- GET localhost:5050/api/cart/6885...
- Launchpad

Body

```
{
  "email": "gunagantisravan376@gmail.com",
  "password": "Sravan@2025"
}
```

Response

```
{
  "success": false,
  "statusCode": 401,
  "error": {
    "title": "Invalid Credentials",
    "message": "Incorrect password. Please try again."
  }
}
```

14. Remaining Error Testing

- POST /api/register with missing field**
- Expected:** 400 Bad Request with Missing Required Fields
- ThunderClient Screenshot:**

The screenshot shows the ThunderClient interface. On the left, the activity log lists various API calls. In the center, a new request is being made to `http://localhost:5050/api/register` using a POST method. The request body contains:

```

1 {
2   "email": "gunagantisravam376@gmail.com",
3   "password": "Sravan@2025"
4 }

```

The response status is **400 Bad Request**, size **166 Bytes**, and time **174 ms**. The response body is:

```

1 {
2   "success": false,
3   "statusCode": 400,
4   "error": {
5     "title": "Missing Required Fields",
6     "message": "Name, Email, Password are required. Please enter all the fields to register"
7   }
8 }

```

The bottom of the screen shows a terminal window with the message `verified`.

- **GET /api/products/:id with invalid ID**
- **Expected: 400 Bad Request - Invalid Id**
- **ThunderClient Screenshot:**

The screenshot shows the ThunderClient interface. On the left, the activity log lists various API calls. In the center, a new request is being made to `http://localhost:5050/api/products/6888f04fb` using a GET method. The request parameters are empty.

The response status is **400 Bad Request**, size **146 Bytes**, and time **207 ms**. The response body is:

```

1 {
2   "success": false,
3   "statusCode": 400,
4   "error": {
5     "title": "Invalid Id",
6     "message": "The provided product ID '6888f04fb' is not a valid MongoDB ObjectId."
7   }
8 }

```

A warning message in the bottom right corner states: `Warning: The free version will no longer support collections starting August 3rd, 2025.`

- **POST /api/cart with non-existent product ID**
- **Expected: 404 Not Found - Product Not Found**
- **ThunderClient Screenshot:**

The screenshot shows the Thunder Client interface. On the left, a sidebar lists recent API interactions. In the center, a request is being built: Method: PUT, URL: http://localhost:5050/api/cart/6. The body contains JSON data:

```

1 {
2   "productId": "6888f61cbc92c7e91895f4d8",
3   "quantity":4
4 }

```

The response panel shows:

Status: 404 Not Found | Size: 144 Bytes | Time: 21 ms

Response:

```

1 {
2   "success": false,
3   "statusCode": 404,
4   "error": {
5     "title": "Product Not Found",
6     "message": "No product found with ID '6888f61cbc92c7e91895f4d8' in cart"
7   }
8 }

```

Below the main interface, a message box displays: "Warning: The free version will no longer support collections starting August 3rd, 2025. Source: Thunder Client".

- **GET /invalidroute**
- **Expected:** 404 Not Found - Route not found
- **ThunderClient Screenshot:**

The screenshot shows the Thunder Client interface. On the left, a sidebar lists recent API interactions. In the center, a request is being built: Method: GET, URL: http://localhost:5050/invalidroute. The response panel shows:

Status: 404 Not Found | Size: 132 Bytes | Time: 110 ms

Response:

```

1 {
2   "success": false,
3   "statusCode": 404,
4   "error": {
5     "title": "Invalid Route",
6     "message": "The requested route '/invalidroute' does not exist."
7   }
8 }

```

- **PUT /api/cart without cart**
- **Expected:** 404 Not Found - Cartnot found
- **ThunderClient Screenshot:**

The screenshot shows the Thunder Client interface. On the left, there's a sidebar titled "THUNDER CLIENT" with various icons and a list of recent API requests. The main area has tabs for "New Request", "Activity", "Collections", and "Env". A "PUT" request is selected with the URL "http://localhost:5050/api/cart". The "Body" tab contains JSON content:

```
1 {
2   "productId": "6888f61cbc92c7e91895f4d8",
3   "quantity":4
4 }
```

The "Response" tab shows the following JSON output:

```
1 {
2   "success": false,
3   "statusCode": 404,
4   "error": {
5     "title": "Cart Not Found",
6     "message": "No cart found for user ID 6888f61cbc92c7e91895f4d8"
7   }
8 }
```

Below the main interface, there's a "TERMINAL" tab and a system tray at the bottom with weather information (28°C, Mostly cloudy), system icons, and a date/time stamp (29-07-2025).