

ShoppyGlobe RESTful API - Test Case Documentation

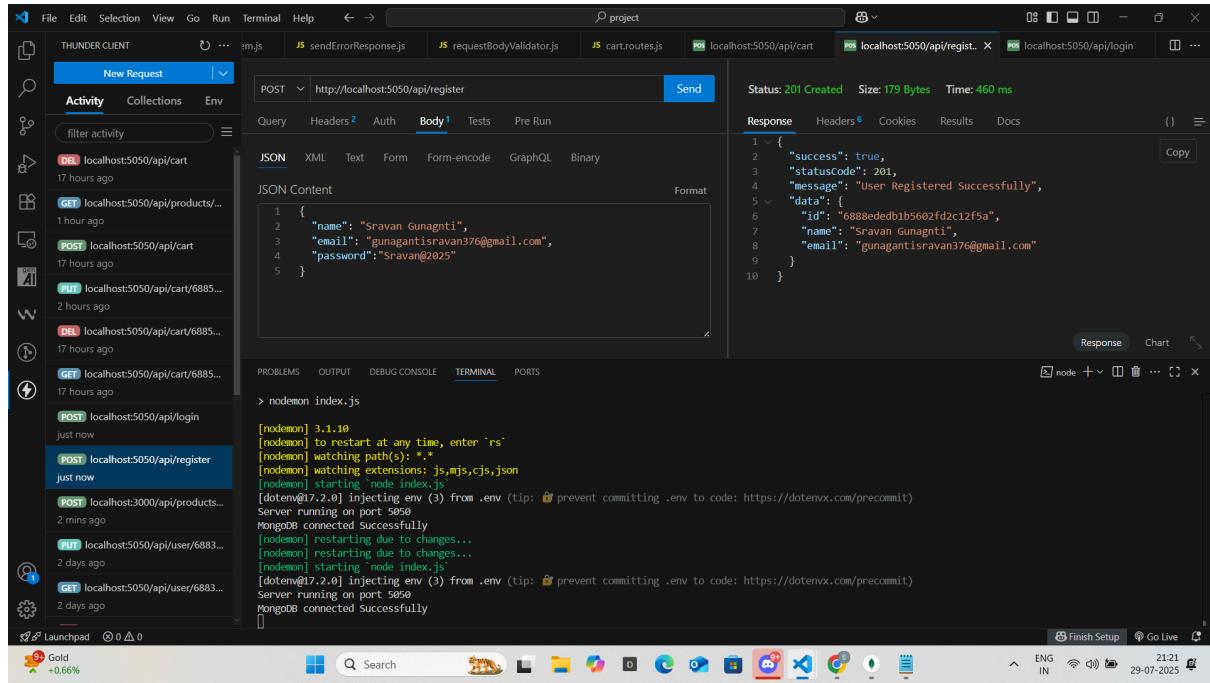
Overview

This document includes all possible test cases for the REST API routes developed for the ShoppyGlobe backend project. The test cases are organized based on the route and functionality. Each section includes the HTTP method, endpoint, description, request body (if applicable), expected status, and expected response message.

User Authentication

1. Register - Successful

- **POST /api/register**
- **Description:** Registers a new user
- **Expected:** **201 Created**, JWT token and user data returned



The screenshot shows the Thunder Client application interface. In the top navigation bar, the 'Body' tab is selected. Below it, the 'JSON' tab is active. The JSON content pane contains the following data:

```
1  {
2   "name": "Sravan Gunagni",
3   "email": "gunagantisravan376@gmail.com",
4   "password": "Sravan@2025"
5 }
```

In the top right corner of the main window, there is a status message: "Status: 201 Created Size: 179 Bytes Time: 460 ms". The bottom right corner of the window shows the date and time: "29-07-2025 21:21".

2. Register - Missing Fields

- **Expected:** 400 Bad Request, "Missing Required Fields"

The screenshot shows the Thunder Client interface. On the left, the activity log lists various API calls. In the center, a new request is being prepared for `http://localhost:5050/api/register`. The `Body` tab is selected, showing the following JSON content:

```
1 {
2   "email": "gunagantisravan376@gmail.com",
3   "password": "Sravan@2025"
4 }
```

The response pane on the right shows a 400 Bad Request status with the following JSON error message:

```
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 }
```

3. Register - Invalid Email

- **Expected:** 400 Bad Request, "Invalid Email"

The screenshot shows the Thunder Client interface. The activity log on the left includes a recent `POST localhost:5050/api/register` call. In the center, a new request is being prepared for `http://localhost:5050/api/register`. The `Body` tab is selected, showing the following JSON content:

```
1 {
2   "name": "Sravan Gunaganti",
3   "email": "gunagantisravanmail.com",
4   "password": "Sravan"
5 }
```

The response pane on the right shows a 400 Bad Request status with the following JSON error message:

```
1 {
2   "success": false,
3   "statusCode": 400,
4   "error": {
5     "title": "Invalid Email",
6     "message": "Please enter a valid email address. Please ensure it follows the standard format like: example@domain.com"
7   }
8 }
```

A warning message at the bottom right states: "Warning: The free version will no longer support collections starting August 3rd, 2025." and "Source: Thunder Client".

4. Register - Weak Password

- **Expected: 400 Bad Request, "Invalid Password"**

The screenshot shows the Thunder Client interface. In the top navigation bar, the 'Body' tab is selected. Below it, the 'JSON Content' section contains the following JSON payload:

```

1 {
2   "name": "Sravan Gunaganti",
3   "email": "gunagantisravan@gmail.com",
4   "password": "Sravan"
5 }

```

The response pane on the right displays the following details:

Status: 400 Bad Request | Size: 220 Bytes | Time: 112 ms

Response:

```

1 {
2   "success": false,
3   "statusCode": 400,
4   "error": [
5     {
6       "title": "Invalid Password",
7       "message": "Password must start with an uppercase letter, include lowercase letters, numbers, special characters, and be at least 6 characters long."
8     }
9   ]
10 }

```

5. Register - Email Already Exists

- **Expected: 409 Conflict, "Email already registered"**

The screenshot shows the Thunder Client interface. In the top navigation bar, the 'Body' tab is selected. Below it, the 'JSON Content' section contains the following JSON payload:

```

1 {
2   "name": "Sravan Gunagni",
3   "email": "gunagantisravan376@gmail.com",
4   "password": "Sravan@2025"
5 }

```

The response pane on the right displays the following details:

Status: 409 Conflict | Size: 152 Bytes | Time: 513 ms

Response:

```

1 {
2   "success": false,
3   "statusCode": 409,
4   "error": [
5     {
6       "title": "Email already registered",
7       "message": "A user with this email already exists. Please login instead."
8     }
9   ]
10 }

```

6. Login - Successful

- **POST /api/login**
- **Expected:** 200 OK, data and JWT returned

The screenshot shows the Thunder Client interface. A new request is being made to `http://localhost:5050/api/login`. The body contains the following JSON:

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

The response status is 201 Created, with a size of 367 Bytes and a time of 194 ms. The response body is a JSON object containing a success message, status code, access token, and user data.

```
{"success": true, "statusCode": 201, "message": "User Registered Successfully", "accesstoken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9 .eyJpZC16IjY4ODhlZGVkyfjNTYwMzkhmtxMny1YSisiMhdC1MzgwNDM 2HSwizXhIjoxNzUzOkWnWYxfQ .1mf3QhyIPWQnOaAjyzhy1692NUixAOXCSD8LiOwc", "data": {"id": "688edeb1b5602fd2c12f5a", "name": "Sravan Gunagni", "email": "gunagantisravan376@gmail.com"}}
```

The terminal pane shows the output of the `nodemon index.js` command, indicating successful node.js execution and MongoDB connection.

7. Login - Invalid Credentials

- **Expected:** 401 Unauthorized, "Invalid credentials"

The screenshot shows the Thunder Client interface. A new request is being made to `http://localhost:5050/api/login`. The body contains the following JSON:

```
[{"email": "gunagantisravan376@gmail.com", "password": "Sravan@20"}]
```

The response status is 401 Unauthorized, with a size of 125 Bytes and a time of 333 ms. The response body is a JSON object indicating failure, an error code, and a message.

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

The terminal pane shows the output of the `nodemon index.js` command, indicating successful node.js execution and MongoDB connection.

8. Login - Missing Fields

- **Expected:** 400 Bad Request, "Missing Required Fields"

The screenshot shows the Thunder Client interface. On the left, the activity log lists various API requests. In the center, a new request is being sent to `http://localhost:5050/api/login`. The body contains the following JSON:

```
1 {
2   "email": "gunagantisravan173@gmail.com"
3 }
```

The response on the right shows a 400 Bad Request status with the message: "Email, Password are required. Please enter all the fields to login".

Product Management

9. Add Product - Successful

- **POST /api/product**
- **Headers:** JWT required
- **Expected:** 201 Created, Product data returned

The screenshot shows the Thunder Client interface. On the left, the activity log lists various API requests. In the center, a new request is being sent to `http://localhost:5050/api/product`. The body contains the following JSON:

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

The response on the right shows a 201 Created status with the message: "Product Created Successfully". The response body includes the created product data:

```
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 }
```

At the bottom, the terminal shows node.js logs indicating the application is running and MongoDB is connected.

10. Add Product - Missing Fields

- **Expected:** 400 Bad Request, "Missing Required Fields"

The screenshot shows the Thunder Client interface. On the left, a sidebar lists various API interactions. In the center, a main panel shows a POST request to `http://localhost:5050/api/product`. The request body contains the following JSON:

```
1 {
2   "name": "Essence Mascara Lash Princess",
3   "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.",
4   "stock": 99
5 }
```

The response on the right indicates a **400 Bad Request** with a size of 151 bytes and a time of 455 ms. The error message is:

```
1 {
2   "success": false,
3   "statusCode": 400,
4   "error": {
5     "title": "Missing Required Fields",
6     "message": "Name, description, price , stock are required to add product"
7   }
8 }
```

11. Add Product - Invalid Stock Type

- **Expected:** 400 Bad Request, "Invalid Stock Input"

The screenshot shows the Thunder Client interface. On the left, a sidebar lists various API interactions. In the center, a main panel shows a POST request to `http://localhost:5050/api/product`. The request body contains the following JSON:

```
1 {
2   "name": "Essence Mascara Lash Princess",
3   "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.",
4   "price": 9.99,
5   "stock": 9.7
6 }
```

The response on the right indicates a **400 Bad Request** with a size of 135 bytes and a time of 928 ms. The error message is:

```
1 {
2   "success": false,
3   "statusCode": 400,
4   "error": {
5     "title": "Invalid Stock Input",
6     "message": "stock is required and must be a positive number."
7   }
8 }
```

12. Bulk Product Insert - Successful

- **POST /api/products/bulk**
- **Headers:** JWT required
- **Request Body:** Valid array of products
- **Expected:** 201 Created, products inserted

The screenshot shows the Thunder Client interface. In the top navigation bar, 'THUNDER CLIENT' is selected. Below it, there are tabs for 'New Request', 'localhost:3000/api/produ...', 'localhost:5050/api/produ...', 'New Request', 'products.json', 'dummy.json', and 'localhost:5050/api/produ...'. The main area has a 'New Request' button and a dropdown menu. Under 'Activity', there is a list of recent API interactions. The 'Headers' tab is active, showing the following configuration for the POST request to http://localhost:5050/api/products/bulk:
Accept: */*
User-Agent: Thunder Client (https://www.thunderclient.co)
Authorization: JWT eyJhbGciOiJIUzI1NiIsInR5cCI6IkpxVC19.e
header: value

The 'Response' tab displays the JSON response:

```
1 {
2   "success": true,
3   "statusCode": 201,
4   "message": "100 products inserted successfully.",
5   "data": [
6     {
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,
```

The 'Terminal' tab shows the Node.js logs:

```
[nodemon] 3.1.10
[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
```

13. Get All Products

- **GET /api/products**
- **Expected:** 200 OK, product array returned

The screenshot shows the Thunder Client interface. In the top navigation bar, 'THUNDER CLIENT' is selected. Below it, there are tabs for 'controller copy.js', 'localhost:5050/api/produ...', 'localhost:5050/api/user/6...', 'New Request', 'New Request', 'credentials', and 'localhost:5050/api/produ...'. The main area has a 'New Request' button and a dropdown menu. Under 'Activity', there is a list of recent API interactions. The 'Query' tab is active, showing the following configuration for the GET request to http://localhost:5050/api/products:
parameter: value

The 'Response' tab displays the JSON response:

```
1 {
2   "success": true,
3   "statusCode": 200,
4   "data": [
5     {
6       "_id": "688be09ccbf5086297e158c",
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,
```

The 'Terminal' tab shows the Node.js 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
```

14. Get Product by ID - Valid ID

- **GET /api/products/:id**
- **Expected:** 200 OK, product object returned

```
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 }
```

15. Get Product by ID - Non existing product

- **Expected:** 404 Not Found, "Product Not Found"

```
1 {
2   "success": false,
3   "statusCode": 404,
4   "error": {
5     "title": "Product Not Found",
6     "message": "No product found with ID: 6888f04fb1b5602fd2c1301f"
7   }
8 }
```

16. Update Product - Valid Update

- **PUT /api/products/:id**
- **Headers:** JWT required
- **Expected:** 200 OK, updated product

The screenshot shows the Thunder client interface. On the left, the activity log lists various API calls, including a recent successful PUT request to update a product. In the center, a new request is being prepared with the URL `http://localhost:5050/api/products/6888f04fb1b5602fd2c1301f`. The body contains the following JSON:

```
1 {
2   "name": "Mechanical Keyboard",
3   "description": "Mechanical keyboard With Backlight Keyword",
4   "price": 3010,
5   "stock": 15
6 }
```

The response panel shows a successful 200 OK status with a response body indicating the product was updated successfully:

```
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 }
```

17. Update Product - Not Found

- **Expected:** 404 Not Found, "Product Not Found"

The screenshot shows the Thunder client interface. On the left, the activity log lists various API calls, including a recent failed PUT request due to a product not found. In the center, a new request is being prepared with the URL `http://localhost:5050/api/products/688872331823e1f958b7321b`. The body contains the following JSON:

```
1 {
2   "name": "Calvin Klein CK One",
3   "description": "CK One by Calvin Klein is a classic unisex fragrance, known for its fresh and clean scent. It's a versatile fragrance suitable for everyday wear.",
4   "price": 49.99,
5   "stock": 25
6 }
```

The response panel shows a 404 Not Found status with an error message indicating no product was found:

```
1 {
2   "success": false,
3   "statusCode": 404,
4   "error": {
5     "title": "Product Not Found",
6     "message": "No product found with ID: 688872331823e1f958b7321b to update"
7   }
8 }
```

18. Delete Product - Valid

- **DELETE /api/products/:id**
- **Expected:** 200 OK, deleted message

The screenshot shows the Thunder Client interface. On the left, there's a sidebar with activity logs and a gear icon for settings. The main area has tabs for 'New Request', 'Activity', 'Collections', and 'Env'. A 'DELETE' request is selected with the URL `http://localhost:5050/api/products/6888f04fb1b5602fd2c1301f`. The 'Headers' tab is active, showing 'Accept: */*', 'User-Agent: Thunder Client (https://www.thunderclient.co)', and 'Authorization: JWT eyJhbGciOiJIUzI1NiJnR5cCl6lkpXVCj9.e'. The 'Response' tab shows a JSON object with fields: success: true, statusCode: 200, message: "Product Deleted Successfully", data: { id: "6888f04fb1b5602fd2c1301f", name: "Mechanical Keyboard", description: "Mechanical keyboard With Backlight Keyword", price: 3010, stock: 15, __v: 0 }. Below the response, the terminal shows node.js logs indicating the server is running on port 5050 and MongoDB is connected successfully.

19. Delete Product - Not Found

- **Expected:** 404 Not Found, "Product Not Found"

This screenshot shows a failed DELETE request. The URL is `http://localhost:5050/api/products/6889bf0cbaaf2c183421f729`. The response status is 404 Not Found, with the message: "No product found with ID: 6889bf0cbaaf2c183421f729 to delete". The rest of the interface is similar to the previous screenshot, with activity logs on the left and a terminal window at the bottom.

Cart Management

20. Add to Cart - New Item

- **POST /api/cart**
- **Headers:** JWT required
- **Expected:** 201 OK, product added to cart

The screenshot shows the Thunder Client interface. In the top navigation bar, the project name "ShoppyGlobe_Backend" is visible. The main area displays a POST request to "http://localhost:5050/api/cart". The "Body" tab is selected, showing JSON content: {"productId": "6888f61cbc92c7e91895f47f"}. The "Response" tab shows a successful response with status 201 Created, size 141 Bytes, and time 1.18 s. The response body is: { "success": true, "statusCode": 201, "message": "Product added to cart successfully", "data": { "productId": "6888f61cbc92c7e91895f47f", "quantity": 1 } }. Below the request, the "Activity" panel lists several API interactions, including POST requests to /register, /cart, /login, /product, and /products, along with a POST to /api/cart. The bottom status bar shows the date 31-07-2025 and time 00:08.

21. Add to Cart - Product Not Found

- **Expected:** 404 Not Found, "Product Not Found"

The screenshot shows the Thunder Client interface. In the top navigation bar, the project name "ShoppyGlobe_Backend" is visible. The main area displays a POST request to "http://localhost:5050/api/cart". The "Body" tab is selected, showing JSON content: {"productId": "6888f61cbc92c7e91895f470"}. The "Response" tab shows a failed response with status 404 Not Found, size 150 Bytes, and time 591 ms. The response body is: { "success": false, "statusCode": 404, "error": { "title": "Product Not Found", "message": "No product found with ID: 6888f61cbc92c7e91895f470 to add to cart" } }. Below the request, the "Activity" panel lists several API interactions, including a failed POST to /cart with status 404. The bottom status bar shows the date 30-07-2025 and time 14:28.

22. Add to Cart - Invalid ProductId

- **Expected:** 400 Bad Request, "Invalid Product ID"

The screenshot shows the Thunder Client interface. On the left, there's a sidebar with activity logs and a main request builder. In the request builder, a POST method is selected for the endpoint `http://localhost:5050/api/cart`. The body contains the JSON object `{"productId": "6888f61cbc92c"}`. The response panel shows a status of 400 Bad Request with a message: "The provided product ID '6888f61cbc92c' is not a valid MongoDB ObjectId." A warning message at the bottom right states: "Warning: The free version will no longer support collections starting August 3rd, 2025." The system tray at the bottom shows battery level and date/time.

23. Get User Cart

- **GET /api/cart**
- **Expected:** 200 OK, populated cart

The screenshot shows the Thunder Client interface. A GET method is selected for the endpoint `http://localhost:5050/api/cart`. The response panel shows a status of 200 OK with a message: "The provided product ID '6888f61cbc92c' is not a valid MongoDB ObjectId." A warning message at the bottom right states: "Warning: The free version will no longer support collections starting August 3rd, 2025." The system tray at the bottom shows battery level and date/time.

24. Increase product quantity of cart item

- **PUT /api/cart/increment/:productId**
- **Expected:** 200 OK, quantity incremented successfully

The screenshot shows the Thunder Client interface. On the left, the 'Activity' sidebar lists several previous API calls. In the center, a new request is being sent to `http://localhost:5050/api/cart/increment/688a66fae453afbb1a4f070e`. The 'Body' tab is selected, showing a JSON payload with a single digit '1'. The 'Response' tab on the right displays a successful 200 OK response with a size of 181 bytes and a time of 34 ms. The response body contains a JSON object with a 'success' key set to true, a 'statusCode' of 200, a 'message' of 'Incremented product quantity successfully', and a 'data' array containing one element: a product with an ID of 688a66fae453afbb1a4f070e, a quantity of 5, and an _id of 688a670be453afbb1a4f0714.

```
1 {
2   "success": true,
3   "statusCode": 200,
4   "message": "Incremented product quantity successfully",
5   "data": [
6     {
7       "productId": "688a66fae453afbb1a4f070e",
8       "quantity": 5,
9       "_id": "688a670be453afbb1a4f0714"
10    }
]
```

25. Decrease product quantity of cart item

- **PUT /api/cart/increment/:productId**
- **Expected:** 200 OK, Quantity decremented successfully

The screenshot shows the Thunder Client interface. On the left, the 'Activity' sidebar lists previous API calls. In the center, a new request is being sent to `http://localhost:5050/api/cart/decrement/688a66fae453afbb1a4f070e`. The 'Body' tab is selected, showing a JSON payload with a single digit '1'. The 'Response' tab on the right displays a successful 200 OK response with a size of 181 bytes and a time of 14 ms. The response body contains a JSON object with a 'success' key set to true, a 'statusCode' of 200, a 'message' of 'Decremented product quantity successfully', and a 'data' array containing one element: a product with an ID of 688a66fae453afbb1a4f070e, a quantity of 3, and an _id of 688a670be453afbb1a4f0714.

```
1 {
2   "success": true,
3   "statusCode": 200,
4   "message": "Decremented product quantity successfully",
5   "data": [
6     {
7       "productId": "688a66fae453afbb1a4f070e",
8       "quantity": 3,
9       "_id": "688a670be453afbb1a4f0714"
10    }
]
```

26. Update Cart Item - Invalid id

- **Expected:** 400 Bad Request, "Invalid product id"

The screenshot shows the Thunder Client interface with a request to `http://localhost:5050/api/cart/decrement/688a66fae45`. The response status is 400 Bad Request, with a message indicating the provided Product ID is not valid.

```
1 {
2     "success": false,
3     "statusCode": 400,
4     "error": {
5         "title": "Invalid Product ID",
6         "message": "The provided Product ID '688a66fae45' is not a valid MongoDB ObjectId."
7     }
8 }
```

27. Update Cart Item - Stock exceeded

- **Expected:** 400 Bad Request, "Stock Limit Exceeded"

The screenshot shows the Thunder Client interface with a request to `http://localhost:5050/api/cart/increment/688a66fae45a1fb1a4f070e`. The response status is 400 Bad Request, with a message indicating the stock limit has been reached.

```
1 {
2     "success": false,
3     "statusCode": 400,
4     "error": {
5         "title": "Stock Limit Reached",
6         "message": "Current quantity in cart: 91, stock available: 91."
7     }
8 }
```

28. Decrease Cart Item quantity - Minimum Quantity

- **Expected:** 400 Bad Request, " Minimum Quantity Reached"

PUT http://localhost:5050/api/cart/decrement/688a66fae453afbb1a4f070e

```

Status: 400 Bad Request Size: 140 Bytes Time: 13 ms
Response Headers Cookies Results Docs
1 {
2   "success": false,
3   "statusCode": 400,
4   "error": {
5     "title": "Minimum Quantity Reached",
6     "message": "Cannot decrease quantity, minimum quantity is 1."
7   }
8 }

```

Activity Collections Env

PUT localhost:5050/api/product/ 50 mins ago

PUT localhost:5050/api/cart just now

PUT localhost:3000/api/products 54 mins ago

PUT localhost:5050/api/cart 1 hour ago

PUT localhost:5050/api/cart/6885e9... 1 day ago

PUT localhost:5050/api/user/6883ac... 3 days ago

PUT localhost:5000/user/2 5 days ago

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

[dotenv@17.2.0] injecting env (3) from .env (tip: ⚡ enable debug logging with { debug: true })
Server running on port 5050
MongoDB connected Successfully
[nodemon] restarting due to changes...
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
[dotenv@17.2.0] injecting env (3) from .env (tip: ⚡ prevent building .env in docker: https://dotenvx.com/prebuild)
Server running on port 5050
MongoDB connected Successfully

main* 28°C Mostly cloudy Search Go Live ENG IN 00:19 31-07-2025

29. Update Cart Item - Non existing product

- **Expected:** 404 Not Found, "Product Not Found"

PUT http://localhost:5050/api/cart/decrement/688a66fae453afbb1a4f070b

```

Status: 404 Not Found Size: 185 Bytes Time: 26 ms
Response Headers Cookies Results Docs
1 {
2   "success": false,
3   "statusCode": 404,
4   "error": {
5     "title": "Product Not Found",
6     "message": "The product with ID '688a66fae453afbb1a4f070b' you are trying to update does not exist in your cart."
7   }
8 }

```

Activity Collections Env

PUT localhost:5050/api/product/ 50 mins ago

PUT localhost:5050/api/cart just now

PUT localhost:3000/api/products 54 mins ago

PUT localhost:5050/api/cart 1 hour ago

PUT localhost:5050/api/cart/6885e9... 1 day ago

PUT localhost:5050/api/user/6883ac... 3 days ago

PUT localhost:5000/user/2 5 days ago

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

[dotenv@17.2.0] injecting env (3) from .env (tip: ⚡ enable debug logging with { debug: true })
Server running on port 5050
MongoDB connected Successfully
[nodemon] restarting due to changes...
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
[nodemon] restarting due to changes...
[nodemon] starting `node index.js`
[dotenv@17.2.0] injecting env (3) from .env (tip: ⚡ prevent building .env in docker: https://dotenvx.com/prebuild)
Server running on port 5050
MongoDB connected Successfully

main* 28°C Mostly cloudy Search Go Live ENG IN 00:23 31-07-2025

30. Remove Item from Cart

- **DELETE** /api/cart/:productId

- **Expected:** 200 OK, Deleted cart item successfully

The screenshot shows the Thunder Client interface. On the left, a sidebar lists various API requests. In the center, a main panel shows a DELETE request to `http://localhost:5050/api/cart/6888f61cbc92c7e9189547f`. The response status is 200 OK, size 76 bytes, and time 81 ms. The response body is a JSON object:

```

1 {
2   "success": true,
3   "statusCode": 200,
4   "message": "Deleted Cart Item Successfully"
5 }

```

Below the main panel, a terminal window shows server logs indicating the application is running on port 5050 and MongoDB is connected successfully.

31. Remove Item from Cart - Invalid ProductID

- **Expected:** 400 Bad Request, "Invalid ProductId"

The screenshot shows the Thunder Client interface. On the left, a sidebar lists various API requests. In the center, a main panel shows a DELETE request to `http://localhost:5050/api/cart/6888f61cbc`. The response status is 400 Bad Request, size 154 bytes, and time 10 ms. The response body is a JSON object:

```

1 {
2   "success": false,
3   "statusCode": 400,
4   "error": {
5     "title": "Invalid productid",
6     "message": "The provided product ID '6888f61cbc' is not a valid MongoDB ObjectId."
7   }
8 }

```

Below the main panel, a terminal window shows server logs indicating the application is running on port 5050 and MongoDB is connected successfully.

32. Clear Entire Cart

- **DELETE** /api/cart

- **Expected:** 200 OK, "Cart cleared"

Status: 200 OK Size: 86 Bytes Time: 44 ms

```

1 {
2   "success": true,
3   "statusCode": 200,
4   "message": "Your cart has been cleared successfully."
5 }

```

Error & Misc Testing

33. Access Protected routes without token

- **Expected:** 401 Unauthorized, "Access token missing"

Status: 401 Unauthorized Size: 113 Bytes Time: 375 ms

```

1 {
2   "success": false,
3   "statusCode": 401,
4   "error": {
5     "title": "Unauthorized",
6     "message": "Access token missing or malformed"
7   }
8 }

```

34. Invalid Route Access

- **GET /invalidroute**
- **Expected:** 404 Not Found, "Invalid Route"

The screenshot shows the Thunder Client interface. In the left sidebar, there is a list of activity logs. In the main panel, a new request is being sent to `http://localhost:5050/invalidroute`. The response status is 404 Not Found, with a size of 132 bytes and a time of 110 ms. The response body contains JSON with the following content:

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

35. Expired Token

- **Expected:** 403 Forbidden, "Token Expired"

The screenshot shows the Thunder Client interface. In the left sidebar, there is a list of activity logs. In the main panel, a DELETE request is being sent to `http://localhost:5050/api/cart`. The response status is 403 Forbidden, with a size of 125 bytes and a time of 90 ms. The response body contains JSON with the following content:

```
1 {
2     "success": false,
3     "statusCode": 403,
4     "error": {
5         "title": "Token Expired",
6         "message": "Your token has expired. Please log in again."
7     }
8 }
```

36. Invalid Token Format

- **Expected: 403 Forbidden, "Invalid Token"**

```

Status: 403 Forbidden Size: 126 Bytes Time: 547 ms
Response Headers Cookies Results Docs
1 {
2   "success": false,
3   "statusCode": 403,
4   "error": {
5     "title": "Invalid Token",
6     "message": "Access token is invalid. Please log in again."
7   }
8 }

```

37. Invalid Json

- **Expected: 400 Bad Request, "Invalid Json"**

```

Status: 400 Bad Request Size: 149 Bytes Time: 13 ms
Response Headers Cookies Results Docs
1 {
2   "success": false,
3   "statusCode": 400,
4   "error": {
5     "title": "Invalid JSON",
6     "message": "The JSON payload is malformed. Please check your request
body syntax."
7   }
8 }

```