### ****API Documentation****.

## **Base URL**

**http://localhost:3000**

## **Endpoints**

### ****1. Product Management****

#### **1.1 Create a Product**

**Endpoint**: **POST /products**

**Description**: Add a new product to the inventory.

**Request Body**:

{

"name": "Product A",

"sku": "SKU001",

"price": 10.50,

"quantity": 50

}

**Response**:

{

"message": "Product created successfully.",

"productId": 1

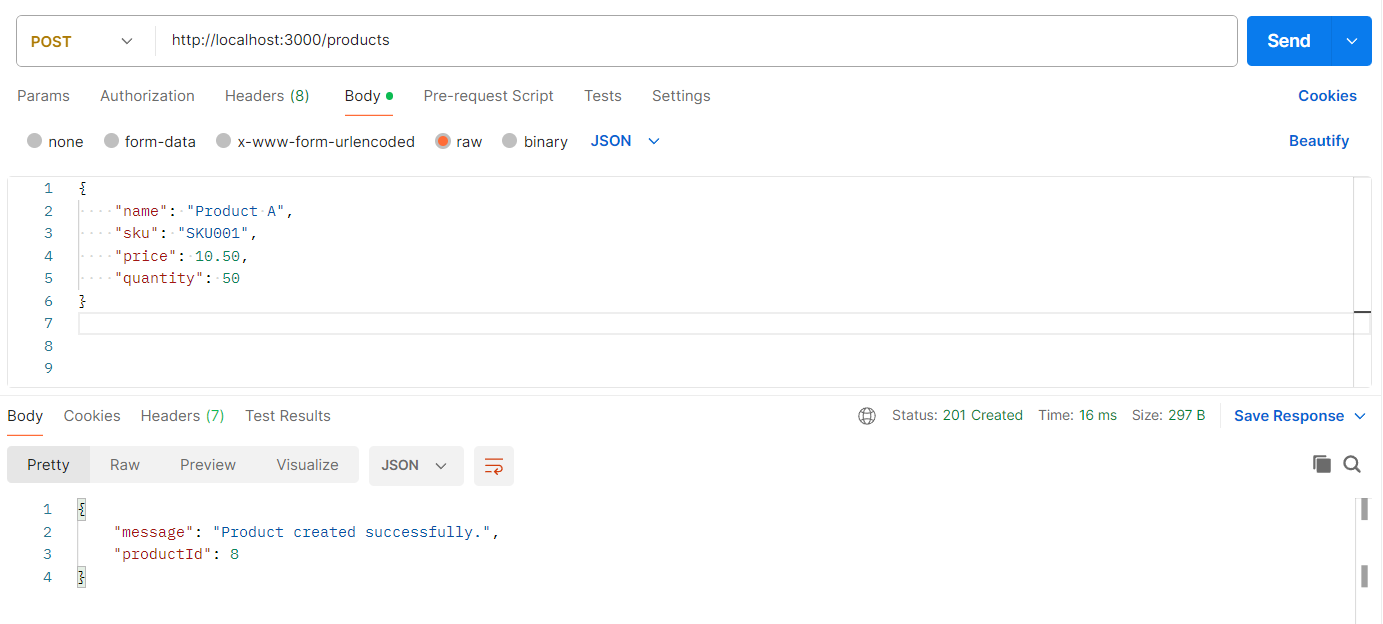
}

**Error Response**:

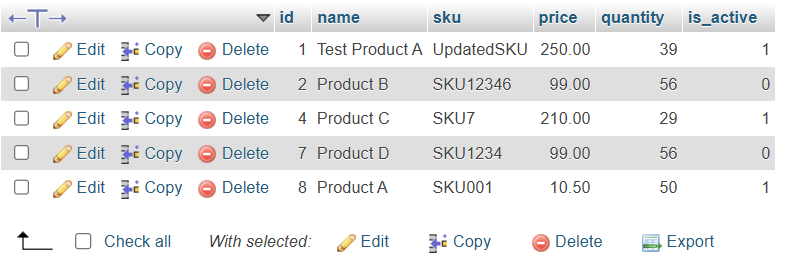
{

"error": "Failed to create product."

}



Product got added to the database:



#### **1.2 Update a Product**

**Endpoint**: **PUT /products/:id**

**Description**: Update details of an existing product.

**Path Parameter**:

id: Product ID to update.

**Request Body**:

{

"name": "Updated Product A",

"sku": "SKU001",

"price": 12.00,

"quantity": 40

}

**Response**:

{

"message": "Product updated successfully."

}

**Error Responses**:

If product not found:

{

"error": "Product not found."

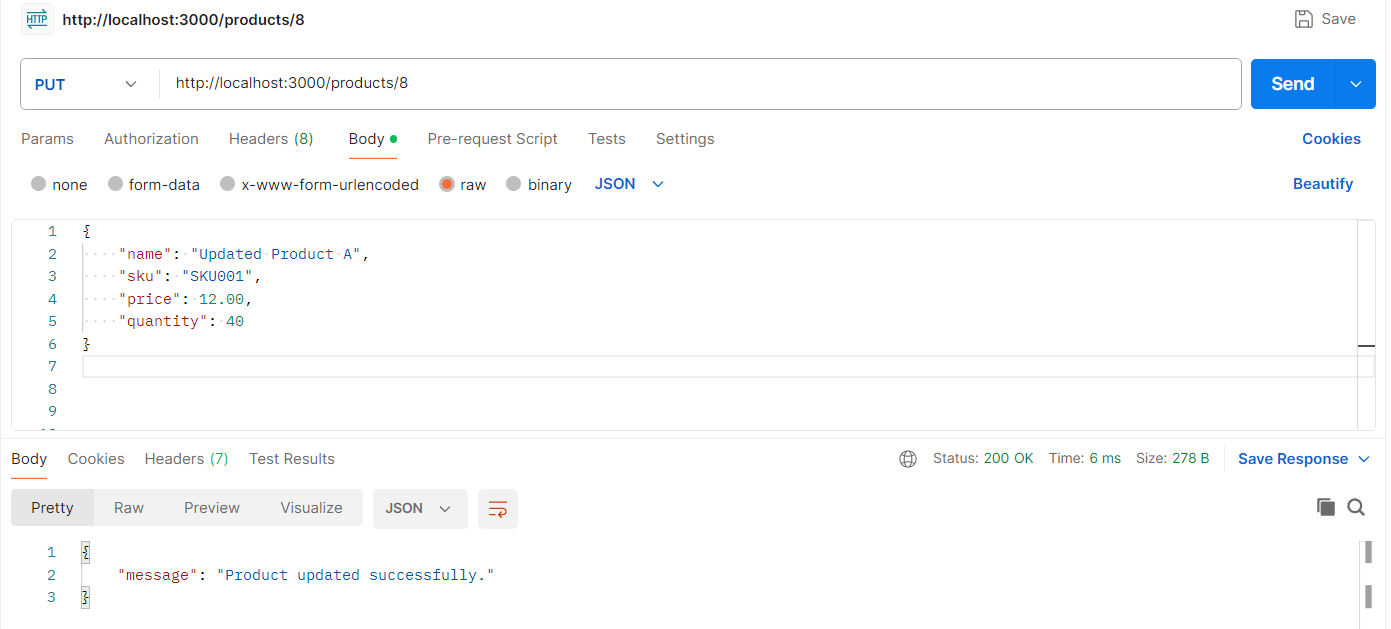
}

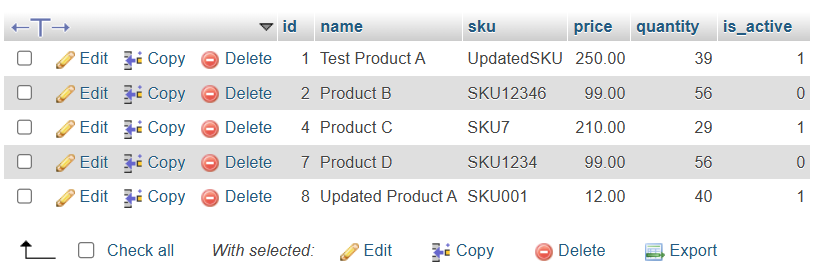
On failure:

{

"error": "Failed to update product."

}





#### **1.3 List Products**

**Endpoint**: **GET /products**

**Description**: Retrieve all products with pagination and search.

**Query Parameters**:

page (optional): Page number (default: 1).

limit (optional): Number of results per page (default: 10).

search (optional): Search term for name or sku.

**Example Request**:

GET /products?page=1&limit=5&search=Product

**Response**:

{

"products": [

{

"id": 1,

"name": "Product A",

"sku": "SKU001",

"price": 10.50,

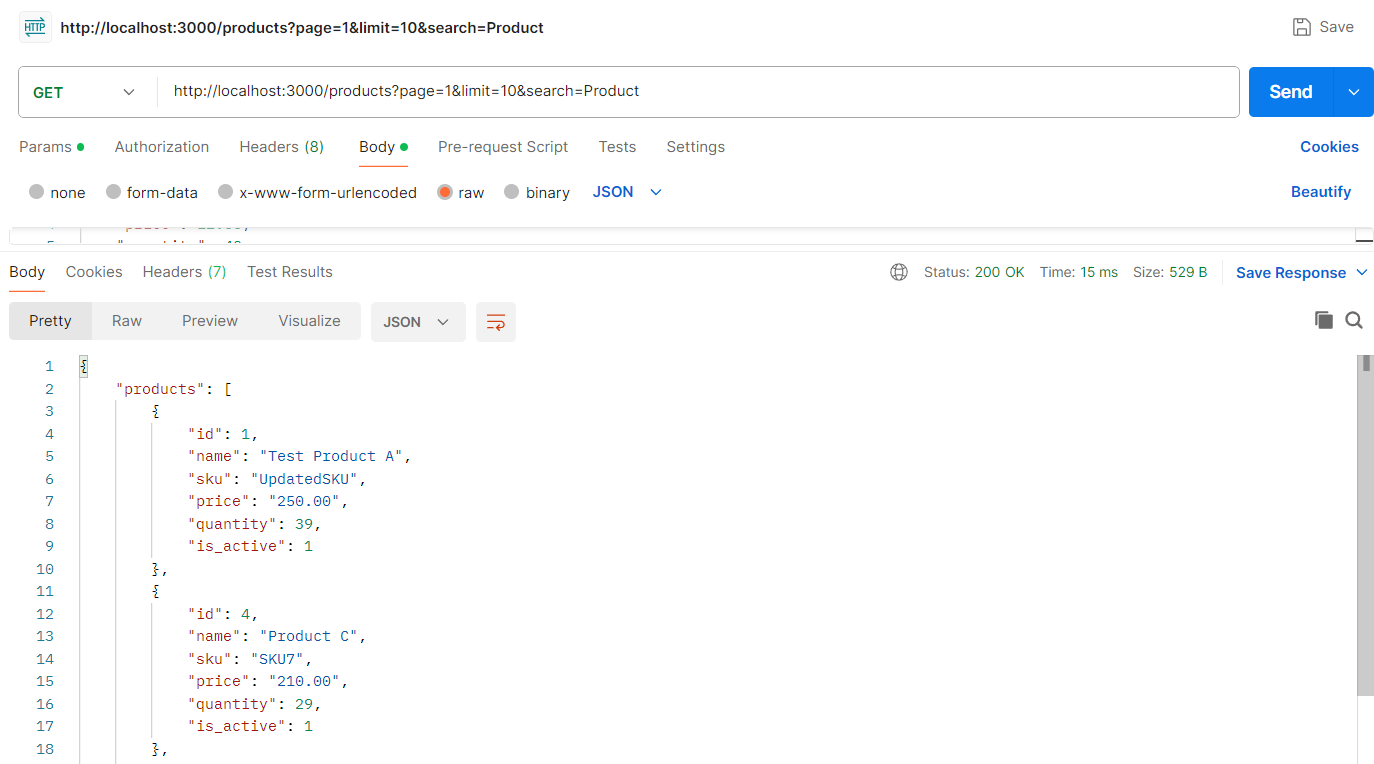
"quantity": 50,

"is\_active": true

}

]

}



#### **1.4 Soft Delete a Product**

**Endpoint**: **DELETE /products/:id**

**Description**: Mark a product as inactive instead of permanently deleting it.

**Path Parameter**:

id: Product ID to soft delete.

**Response**:

{

"message": "Product soft deleted successfully."

}

**Error Responses**:

If product not found:

{

"error": "Product not found."

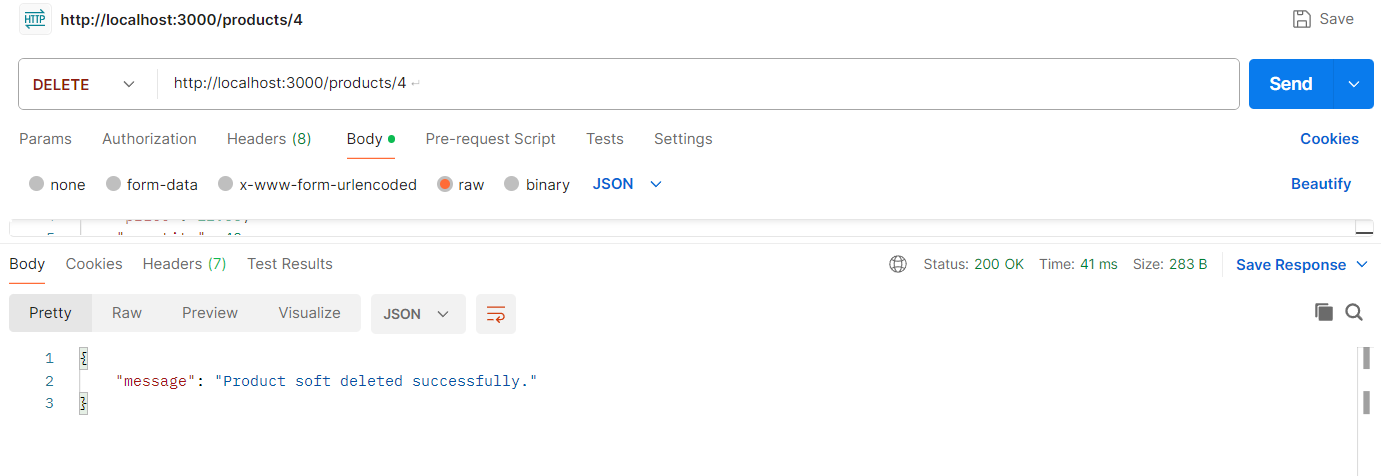
}

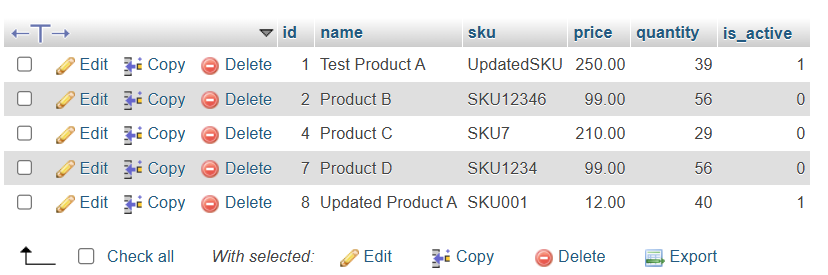
On failure:

{

"error": "Failed to soft delete product."

}





### ****2. Order Management****

#### **2.1 Create an Order**

**Endpoint**: **POST /orders**

**Description**: Create a new order with multiple products.

**Request Body**:

{

"products": [

{ "productId": 1, "quantity": 2 },

{ "productId": 3, "quantity": 5 }

]

}

**Response**:

{

"message": "Order created successfully.",

"orderId": 42

}

**Error Responses**:

If products are missing:

{

"error": "Products are required to create an order."

}

If stock is insufficient:

{

"error": "Insufficient stock for product ID 1"

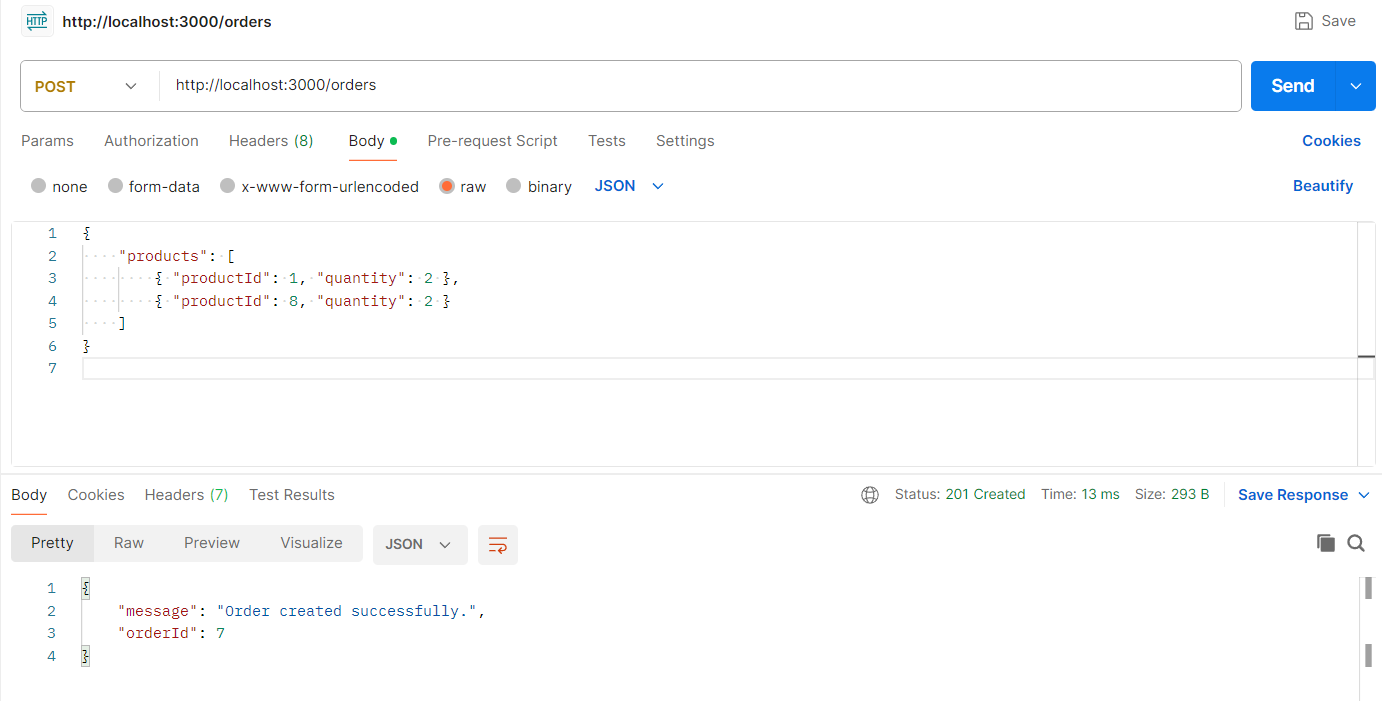
}

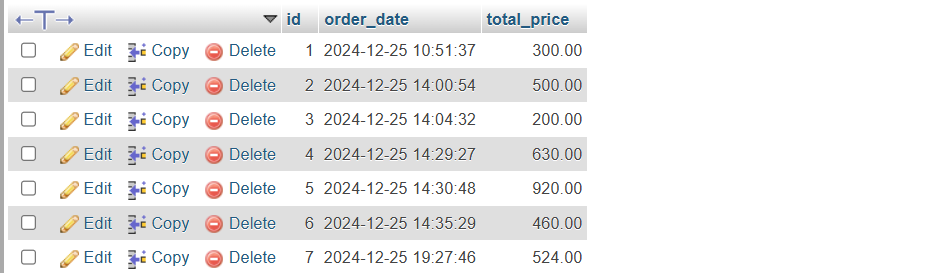
**On failure:**

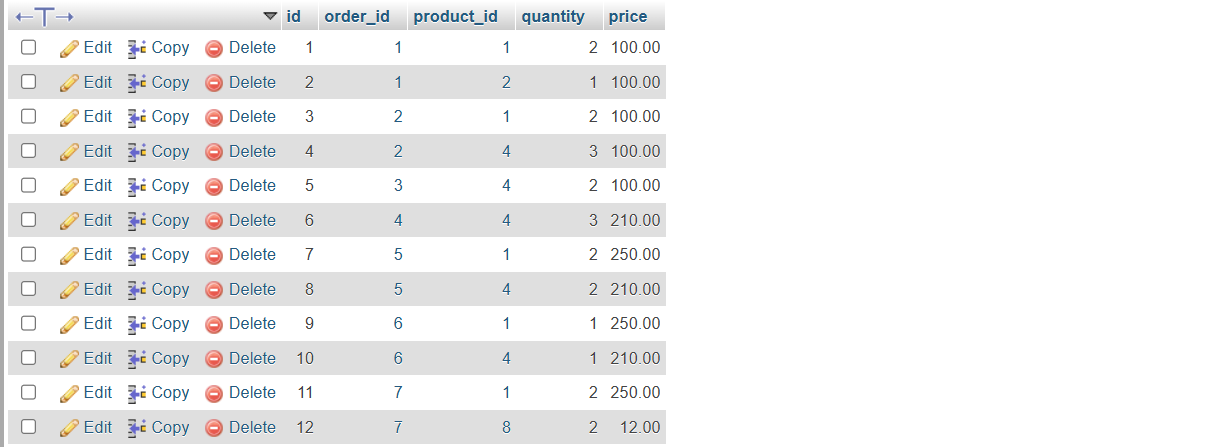
{

"error": "Failed to create order."

}







#### **2.2 Get Order Details**

**Endpoint**: **GET /orders/:id**

**Description**: Retrieve details of a specific order, including all ordered items.

**Path Parameter**:

id: Order ID to retrieve.

**Response**:

{

"order": [

{

"orderId": 42,

"totalPrice": 78.25,

"orderDate": "2024-12-25T10:00:00.000Z",

"productId": 1,

"quantity": 2,

"price": 10.50

},

{

"orderId": 42,

"totalPrice": 78.25,

"orderDate": "2024-12-25T10:00:00.000Z",

"productId": 3,

"quantity": 5,

"price": 7.25

}

]

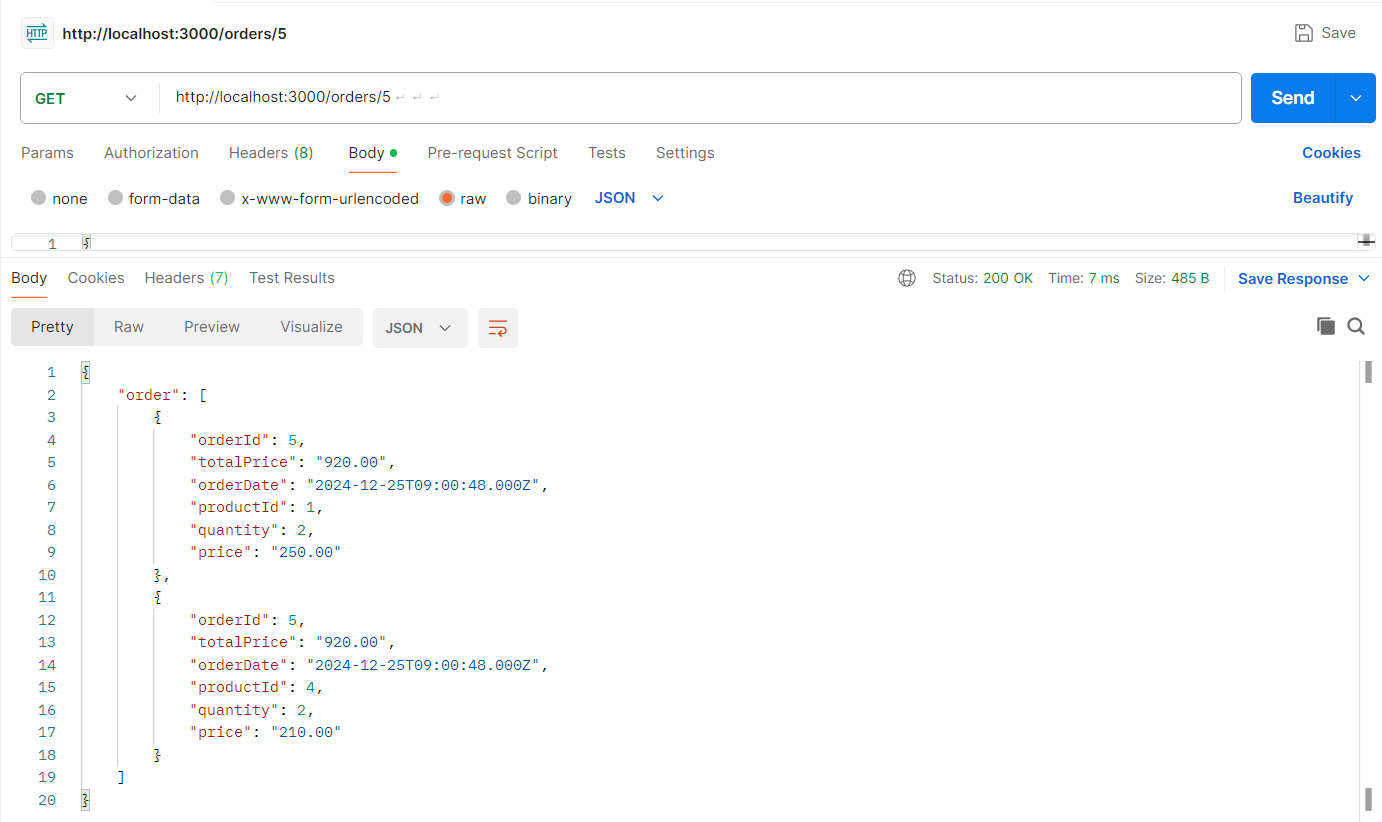
}

**Error Response**:

{

"error": "Failed to fetch order details."

}



#### **2.3 List Orders**

**Endpoint**: **GET /orders**

**Description**: Retrieve all orders with pagination.

**Query Parameters**:

page (optional): Page number (default: 1).

limit (optional): Number of results per page (default: 10).

**Example Request**:

GET /orders?page=1&limit=5

**Response**:

{

"orders": [

{

"id": 42,

"total\_price": 78.25,

"order\_date": "2024-12-25T10:00:00.000Z"

}

]

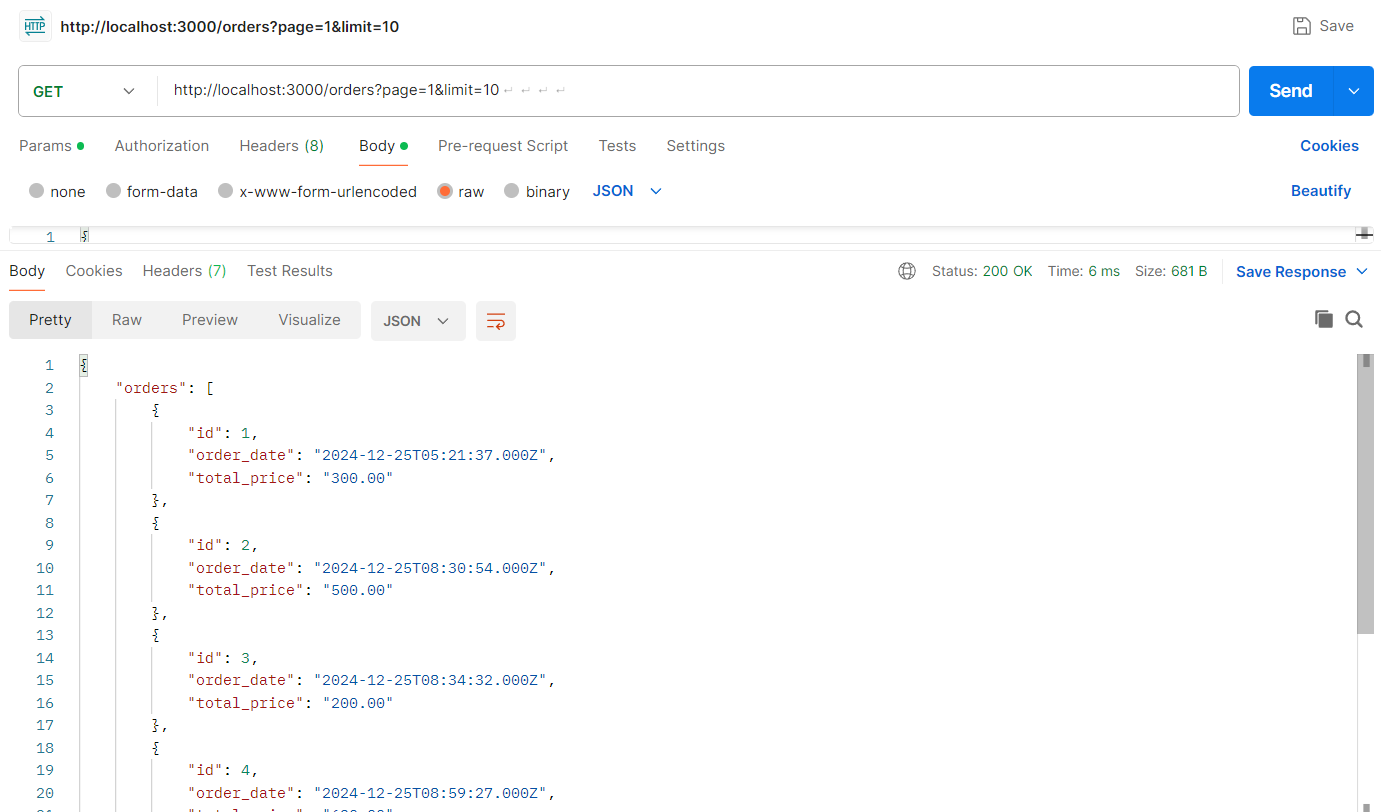
}

**Error Response**:

{

"error": "Failed to fetch orders."

}



### ****Error Codes****

| **Status Code** | **Description** |
| --- | --- |
| 201 | Resource successfully created. |
| 200 | Request successful. |
| 400 | Bad request (e.g., missing fields, invalid input). |
| 404 | Resource not found. |
| 500 | Internal server error (e.g., database or server issues). |

### ****How to Test Using Postman****

**Setup**:

Ensure the server is running (node app.js).

Use **http://localhost:3000** as the base URL.

**Example Requests**:

Test **Product Creation** with a POST request:

POST /products

Body:

{

"name": "Product A",

"sku": "SKU001",

"price": 10.50,

"quantity": 50

}

Test **Order Creation** with a POST request:

POST /orders

Body:

{

"products": [

{ "productId": 1, "quantity": 2 },

{ "productId": 3, "quantity": 5 }

]

}