🛒 E-commerce Backend API Documentation

# Base URL

http://localhost:8000

# 1. Health Check

GET /

Checks if the server is running.

Response:

{  
 "message": "Ecommerce backend is up and running 🚀"  
}

# 2. Test MongoDB Connection

GET /test-mongo

Inserts a dummy product to verify MongoDB connection.

Response:

{  
 "inserted\_id": "64dfd...af7"  
}

# 3. Create Product

POST /products

Adds a new product.

Request Body Example:

{  
 "name": "T-shirt",  
 "price": 299.99,  
 "sizes": [  
 {  
 "size": "M",  
 "stock": 10  
 },  
 {  
 "size": "L",  
 "stock": 5  
 }  
 ]  
}

Response:

{  
 "id": "64df...bce"  
}

# 4. List Products

GET /products

Returns products list with optional filtering & pagination.

Query Parameters: name, size, limit (default=10), offset (default=0)

Example: GET /products?name=shirt&size=M&limit=5&offset=0

Response:

{  
 "data": [{  
 "id": "64df...bce",  
 "name": "T-shirt",  
 "price": 299.99  
 }],  
 "page": {  
 "next": 5,  
 "limit": 5,  
 "previous": 0  
 }  
}

# 5. Get User Orders

GET /orders/{user\_id}

Returns all orders for a user.

Response:

{  
 "orders": [{  
 "order\_id": "64ef...abc",  
 "products": [{  
 "product\_id": "64de...b12",  
 "name": "T-shirt",  
 "quantity": 2,  
 "price": 299.99  
 }]  
 }]  
}

# 6. Create Order

POST /orders

Creates a new order and reduces product stock.

Request Body Example:

{  
 "user\_id": "123456",  
 "items": [  
 {  
 "product\_id": "64de...b12",  
 "quantity": 2  
 },  
 {  
 "product\_id": "64de...c34",  
 "quantity": 1  
 }  
 ]  
}

Response:

{  
 "order\_id": "64ef...abc"  
}