Comforty E-commerce Workflow Documentation

1. Introduction

This document provides a comprehensive overview of the e-commerce website's workflow. The workflow is designed to ensure a seamless and user-friendly shopping experience by integrating frontend functionality, backend data management, and third-party API services.

01) Define Technical Requirements

Frontend Requirements

The frontend focuses on creating a seamless user experience with a responsive UI. Essential features include fast loading times, high-quality images, and user-friendly navigation. Key pages such as Home, Shop, Product, Cart, Checkout, and dynamic pages (e.g., Order Success/Failure) are critical for guiding users through the buying process. Technologies like Next.js, Tailwind CSS, ShadCN, and animation libraries ensure modern and interactive design.

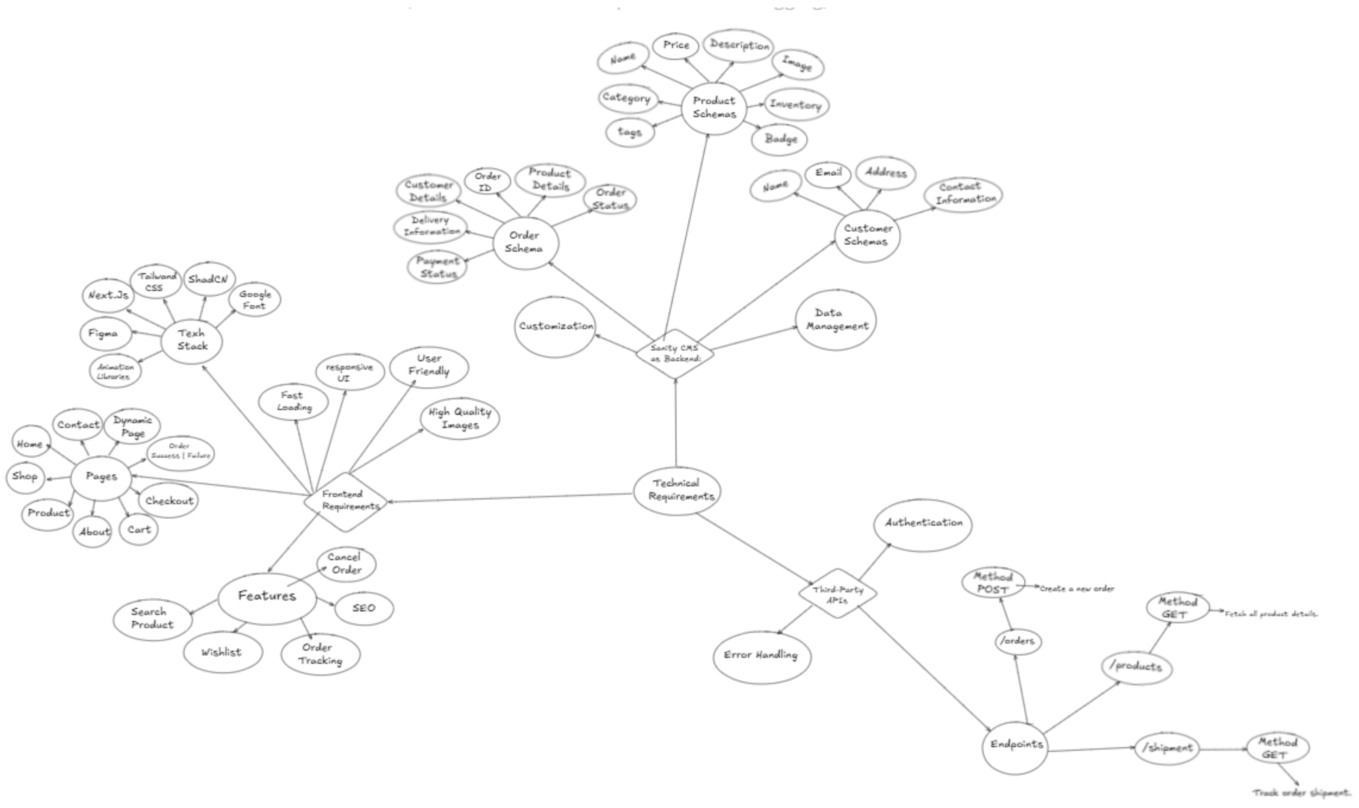
Sanity CMS as Backend

The backend is built on Sanity CMS to manage key data entities effectively. Custom schemas are defined for Products (name, price, description, images, inventory, categories), Customers (name, email, address, contact information), and Orders (order ID, product details, customer details, payment and delivery status). This setup ensures smooth data management and supports customization for specific marketplace needs.

Third-Party API Requirements

Integration with third-party APIs ensures extended functionality and efficient operations. Key endpoints include:

- **/Products**: A GET endpoint to fetch product details.
- **/Orders**: A POST endpoint to create new orders with customer and product data.
- **/Shipment**: A GET endpoint for tracking shipment details.
- Authentication and error handling mechanisms secure and maintain the reliability of these API calls, ensuring seamless communication between the backend and external services.



---

02) Design System Architecture

Workflow Steps

1 User Interaction

1. User Opens Website
   - The user visits the e-commerce website landing page.
2. Create Account
   - Users can create an account and securely store their data in Sanity CMS.
3. Login
   - The system verifies the user's email and password through Sanity CMS.

2 Product Management

4. Fetch Product Data
   - Product information is retrieved dynamically from Sanity CMS and displayed on the website.
5. Product Details
   - Users click on a product to view detailed prices, descriptions, and stock information.
6. Add to Cart
   - Users add the desired product to their cart. This data is stored temporarily in local storage.

3 Checkout and Order Placement

7. Checkout
   - Users proceed to checkout where they can review their cart.
8. Place Order
   - Order details, including user and product information, are sent to Sanity CMS for processing.

4 Shipping and Payment

9. Shipping Details
   - Users fill in their shipping information, such as address and postal code.
10. Shipping Cost Calculation
    - Based on the provided address, the system calculates the shipping costs and methods.
11. Total Amount Display
    - The total amount, including the product price and shipping cost, is displayed.

12. Payment

- Users proceed to make a payment through an integrated payment gateway. (stripe, Shop Pay)

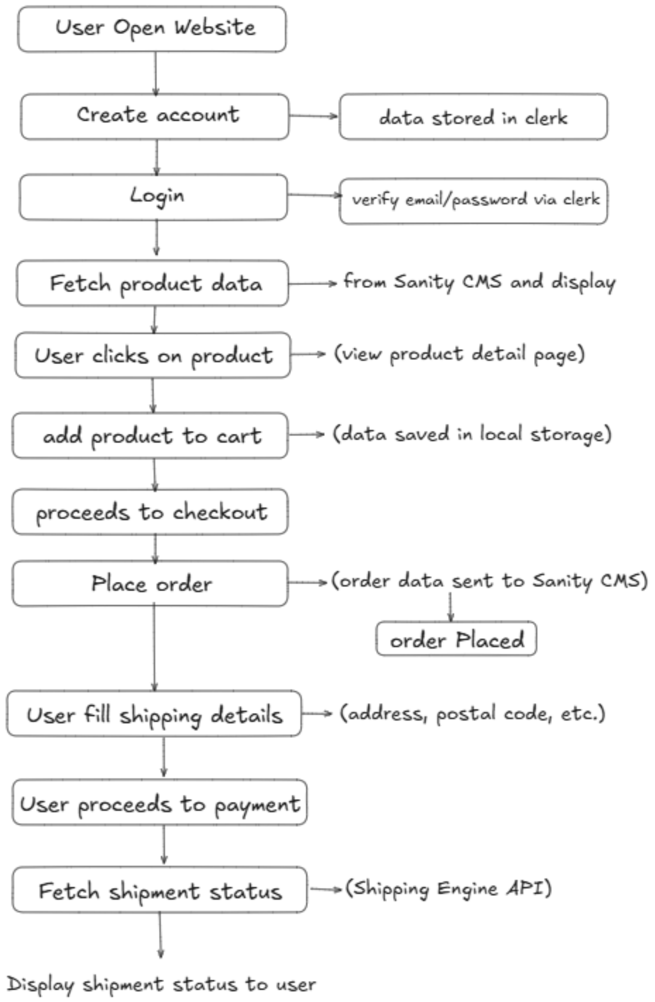
5 Post-Payment Services

13. Fetch Shipment Status

- Shipment status is retrieved via the Ship Engine API.

14. Display Shipment Status

- Shipment details, including tracking and estimated delivery, are displayed to the user.


User Open Website
Create account
data stored in clerk
Login
verify email/password via clerk
Fetch product data
from Sanity CMS and display
User clicks on product
(view product detail page)
add product to cart
(data saved in local storage)
proceeds to checkout
Place order
(order data sent to Sanity CMS)
order Placed
User fill shipping details
(address, postal code, etc.)
User proceeds to payment
Fetch shipment status
(Shipping Engine API)
Display shipment status to user

03) Plan API Requirements

API Overview

The APIs are categorized into three key functional areas:

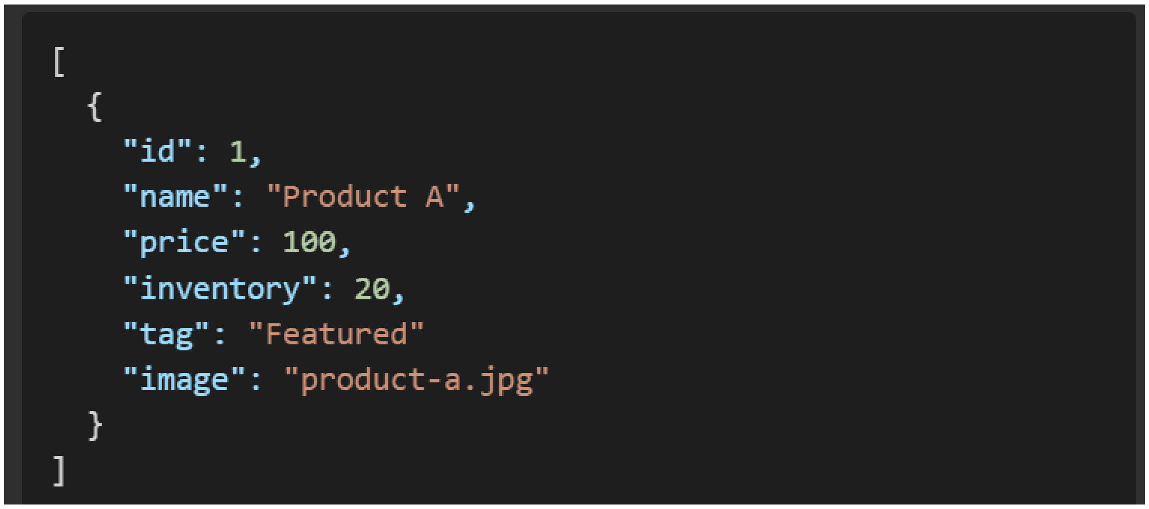
1. Product Management
2. Order Management
3. Shipping and Payment Services

API Endpoints

01) Product Management

1. Fetch All Products

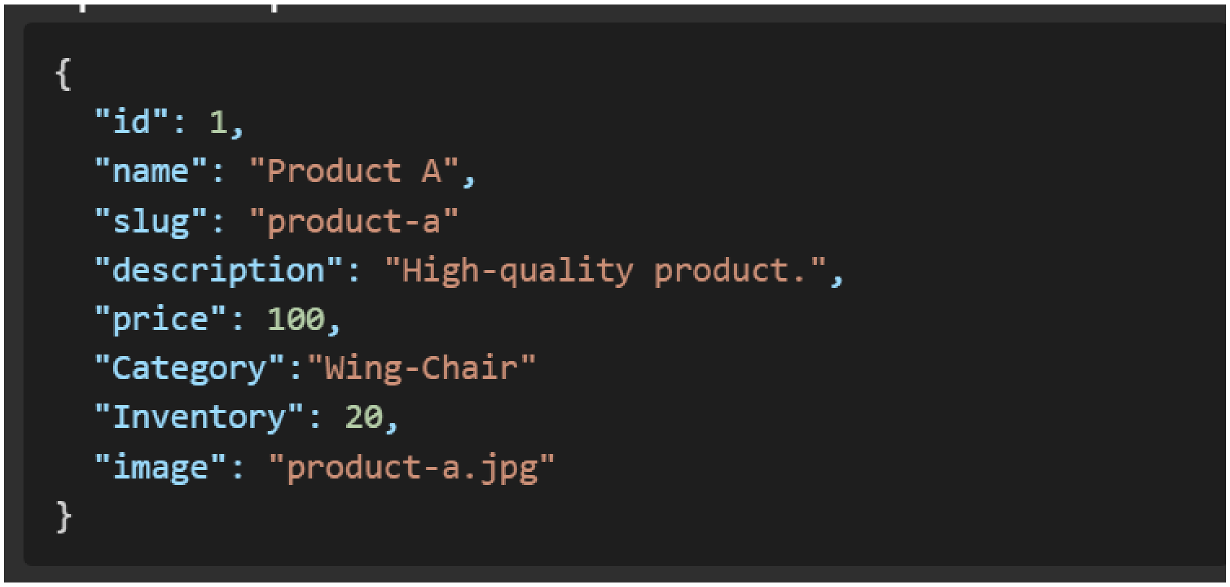
- **Endpoint**: /products
- **Method**: GET
- **Description**: Retrieves a list of all available products.
- **Request Parameters**: None
- **Response Example**:



```json
[
  {
    "id": 1,
    "name": "Product A",
    "price": 100,
    "inventory": 20,
    "tag": "Featured"
    "image": "product-a.jpg"
  }
]
```

2. Fetch Product Details

- **Endpoint**: /products/{id}
- **Method**: GET
- **Description**: Retrieves details of a specific product.
- **Request Parameters**:
  - id: Product ID
- **Response Example**:

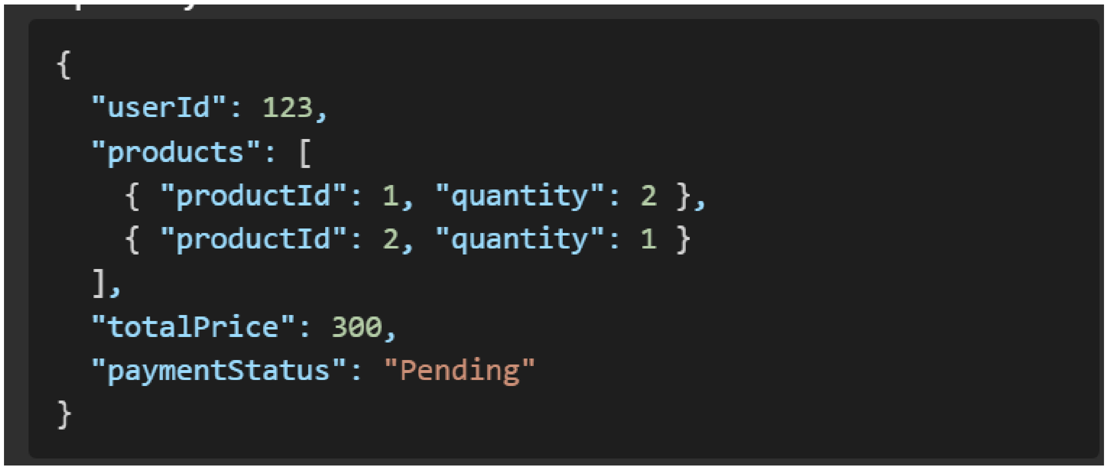


```
{
  "id": 1,
  "name": "Product A",
  "slug": "product-a"
  "description": "High-quality product.",
  "price": 100,
  "Category":"Wing-Chair"
  "Inventory": 20,
  "image": "product-a.jpg"
}
```

02). Order Management

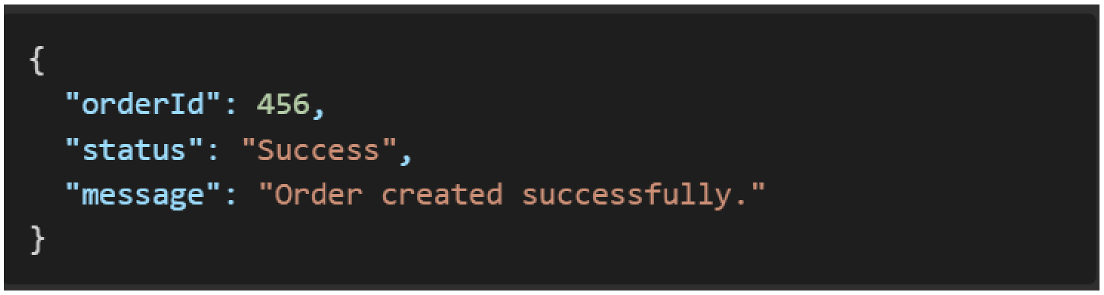
1. Create New Order

- **Endpoint**: /orders
- **Method**: POST
- **Description**: Creates a new order in the system.
- **Request Payload**:



```
{
  "userId": 123,
  "products": [
    { "productId": 1, "quantity": 2 },
    { "productId": 2, "quantity": 1 }
  ],
  "totalPrice": 300,
  "paymentStatus": "Pending"
}
```

- **Response Example**

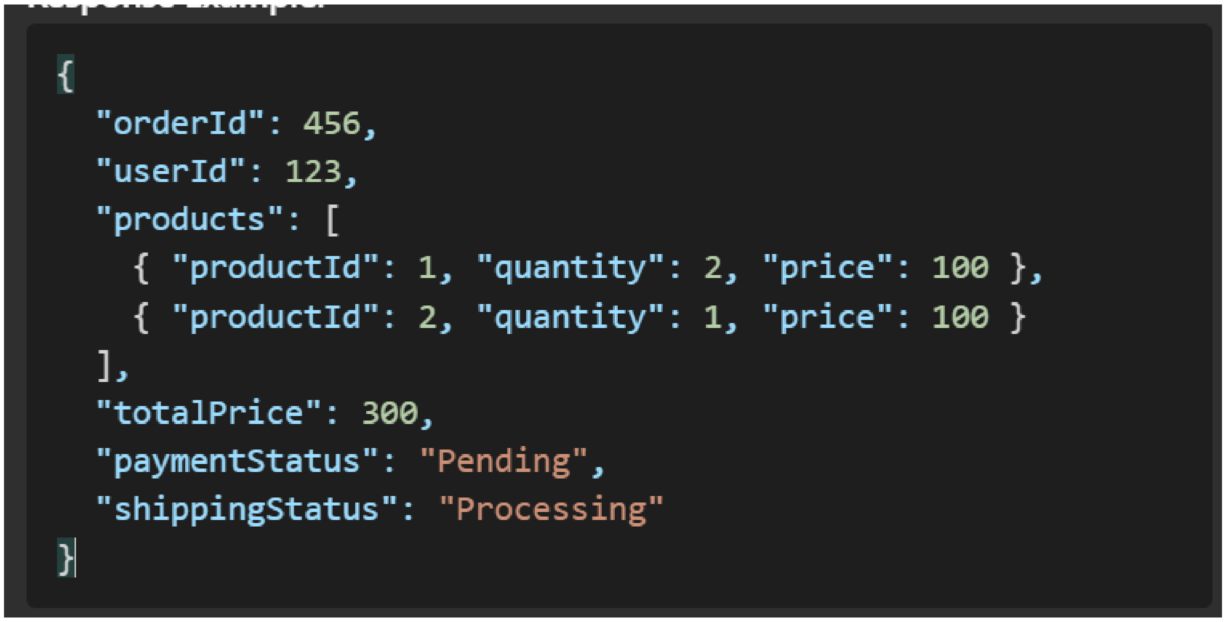


```
{
  "orderId": 456,
  "status": "Success",
  "message": "Order created successfully."
}
```

2. Fetch Order Details

- **Endpoint**: /orders/{id}
- **Method**: GET
- **Description**: Retrieves details of a specific order.
- **Request Parameters**:
    - id: Order ID

- **Response Example**:

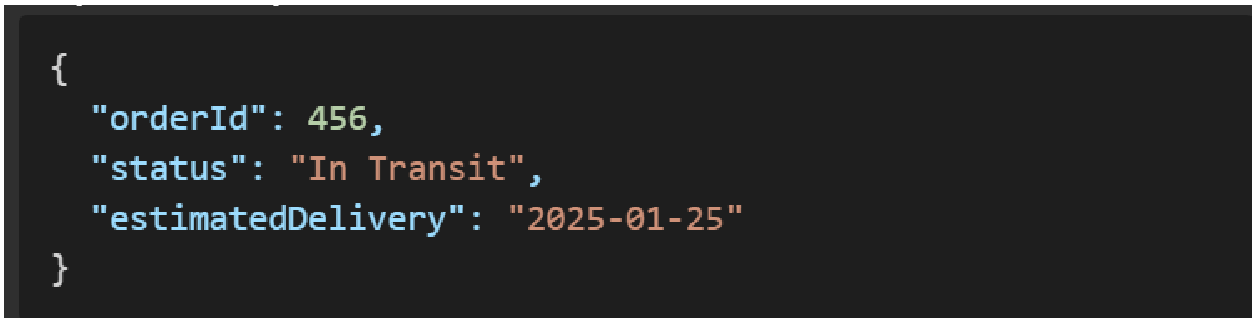


```
{
  "orderId": 456,
  "userId": 123,
  "products": [
    { "productId": 1, "quantity": 2, "price": 100 },
    { "productId": 2, "quantity": 1, "price": 100 }
  ],
  "totalPrice": 300,
  "paymentStatus": "Pending",
  "shippingStatus": "Processing"
}
```

3 Shipping and Payment Services

1. Fetch Shipping Status

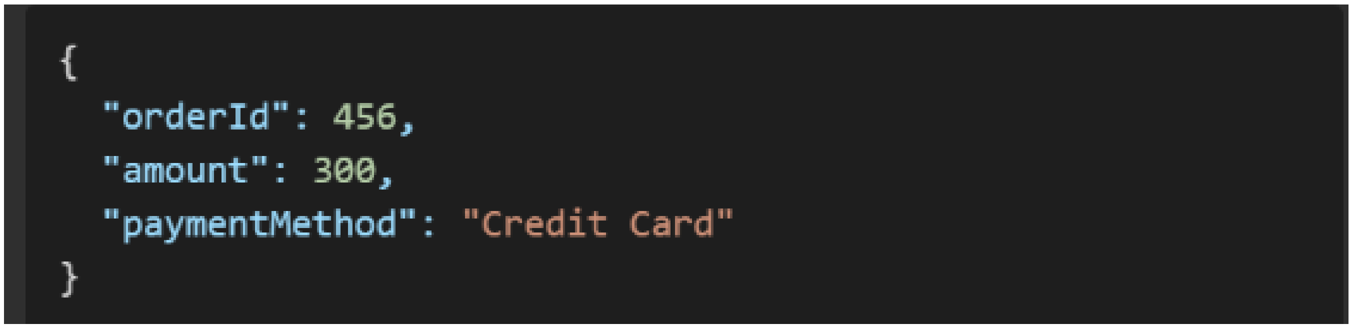
- **Endpoint**: /shipment/{orderId}
- **Method**: GET
- **Description**: Fetches the shipping status of an order.
- **Request Parameters**:
    - orderId: ID of the order
- **Response Example**:



```
{
  "orderId": 456,
  "status": "In Transit",
  "estimatedDelivery": "2025-01-25"
}
```

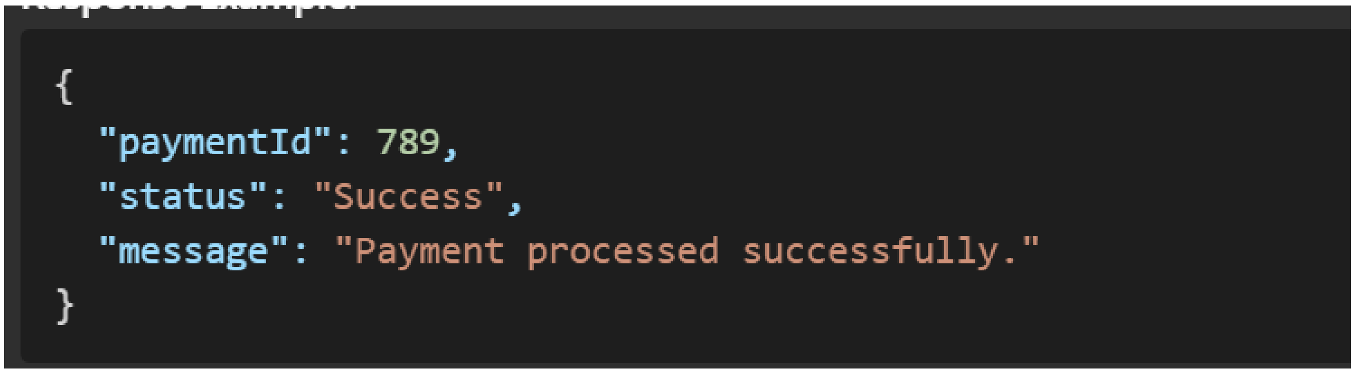
2. Process Payment

- **Endpoint**: /payment
- **Method**: POST
- **Description**: Processes payment for an order.
- **Request Payload**:



```
{
  "orderId": 456,
  "amount": 300,
  "paymentMethod": "Credit Card"
}
```

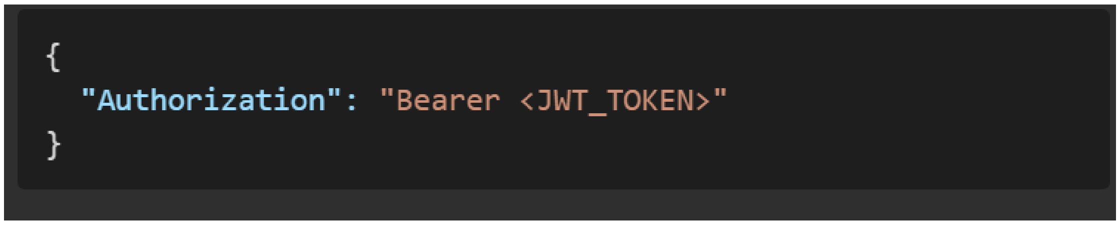
- **Response Example**:



```
{
  "paymentId": 789,
  "status": "Success",
  "message": "Payment processed successfully."
}
```

4. Authentication and Security

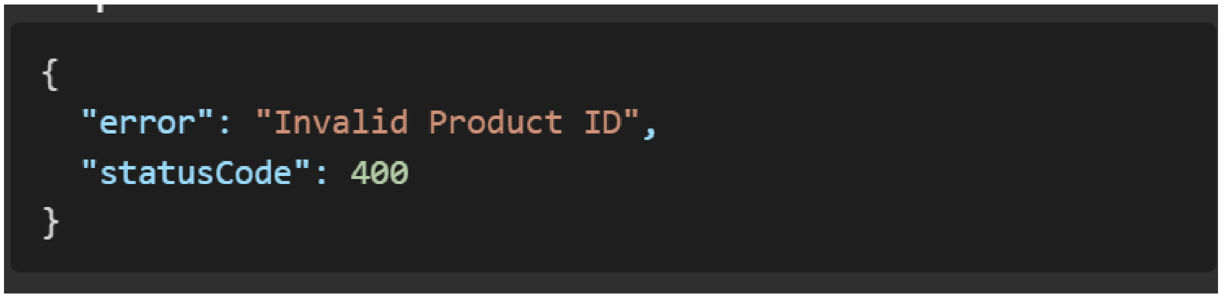
- All API endpoints are secured using **JWT Authentication**.
- Users must include an Authorization header with a valid token for accessing protected endpoints.



```
{
  "Authorization": "Bearer <JWT_TOKEN>"
}
```

5. Error Handling

- Standardized error codes and messages for consistency.
- **Example**:



```
{
  "error": "Invalid Product ID",
  "statusCode": 400
}
```