SHOP.CO API Specification Document

Hackathon Day 2: API Specification Detailed Document

This document provides a comprehensive specification for the APIs powering SHOP.CO, an e-commerce platform specializing in garments for men, women, and children. The API facilitates core functionalities such as product management, user authentication, order processing, payment integration, and shipment tracking.

General Information

Base URL: `https://api.shopco.com/v1 (will decide later )`

Authentication:

- Uses JWT (JSON Web Token) for authentication.
- All endpoints requiring authentication must include an `Authorization` header: Authorization: Bearer <token>

Content Type:

- Requests and responses use `application/json` unless specified otherwise.

API Endpoints

1. User Management

1.1 User Registration

- **Endpoint**: `POST /auth/signup`
- **Description**: Registers a new user.

Request Body:

```
{
 "name": "John Doe",
 "email": "john@example.com",
 "password": "securepassword"
```

- }

Response:

```
{
 "message": "User registered successfully",
 "user": {
  "id": "64f7b94c1a",
  "name": "John Doe",
  "email": "john@example.com"
 }
```

- }

1.2 User Login

- **Endpoint**: `POST /auth/login`
- **Description**: Logs in a user and returns a JWT token.

Request Body:

```
{
 "email": "john@example.com",
 "password": "securepassword"
```

- }

Response:

```
{
 "token": "eyJhbGciOiJIUzI1NiIsInR5...",
 "user": {
  "id": "64f7b94c1a",
  "name": "John Doe",
  "email": "john@example.com"
 }
```

- }

1.3 Get User Orders

- **Endpoint**: `GET /users/:id/orders`
- **Description**: Retrieves all orders for a specific user.
- **Authentication**: Required.

Response:


```
[
 {
  "orderId": "abc123",
  "total": 99.99,
  "status": "Shipped",
  "createdAt": "2025-01-10T14:52:23Z"
 },
 {
  "orderId": "xyz456",
  "total": 49.99,
  "status": "Processing",
  "createdAt": "2025-01-12T08:32:45Z"
 }
```


-]

---

2. Product Management

2.1 Get All Products

- **Endpoint**: `GET /products`
- **Description**: Fetches all available products.

Response:


```
[
 {
  "id": "p12345",
  "name": "Men's T-Shirt",
  "price": 19.99,
  "category": "Men's Fashion",
  "stock": 100,
```

```
  "image": "https://shopco.com/images/p12345.jpg"
 },
 {
  "id": "p67890",
  "name": "Women's Hoodie",
  "price": 34.99,
  "category": "Women's Fashion",
  "stock": 50,
  "image": "https://shopco.com/images/p67890.jpg"
 }
```


-]

2.2 Get Product by ID

- **Endpoint**: `GET /products/:id`
- **Description**: Fetches details of a specific product.

Response:


```
{
 "id": "p12345",
 "name": "Men's T-Shirt",
 "description": "High-quality cotton t-shirt available in multiple sizes and colors.",
 "price": 19.99,
 "category": "Men's Fashion",
 "stock": 100,
 "images": [
  "https://shopco.com/images/p12345-front.jpg",
  "https://shopco.com/images/p12345-back.jpg"
 ]
```


- }

2.3 Add New Product

- **Endpoint**: `POST /products`
- **Description**: Adds a new product (Admin only).
- **Authentication**: Required.

Request Body:

```
{
 "name": "Kids' Jacket",
 "description": "Warm winter jacket for kids.",
 "price": 49.99,
 "category": "Kids' Fashion",
 "stock": 25,
 "images": [
  "https://shopco.com/images/kids-jacket.jpg"
 ]
```

- }

Response:

```
{
 "message": "Product added successfully",
 "product": {
  "id": "p67891",
  "name": "Kids' Jacket",
  "price": 49.99
 }
```

- }

---

3. Order Management

3.1 Create Order

- **Endpoint**: `POST /orders`
- **Description**: Creates a new order for the user.
- **Authentication**: Required.

Request Body:

```
{
 "userId": "64f7b94c1a",
 "products": [
  {
   "productId": "p12345",
```

```
    "quantity": 2
   },
   {
    "productId": "p67890",
    "quantity": 1
   }
 ],
 "total": 74.97,
 "shippingAddress": "123 Main St, City, Country"
```


- }

Response:


```
{
 "message": "Order placed successfully",
 "orderId": "abc123",
 "status": "Processing"
```


- }

3.2 Get Order by ID

- **Endpoint**: `GET /orders/:id`
- **Description**: Retrieves details of a specific order.
- **Authentication**: Required.

Response:


```
{
 "orderId": "abc123",
 "userId": "64f7b94c1a",
 "products": [
   {
    "productId": "p12345",
    "name": "Men's T-Shirt",
    "quantity": 2,
    "price": 19.99
   },
   {
    "productId": "p67890",
    "name": "Women's Hoodie",
```

```
    "quantity": 1,
    "price": 34.99
  }
 ],
 "total": 74.97,
 "shippingAddress": "123 Main St, City, Country",
 "status": "Shipped",
 "createdAt": "2025-01-10T14:52:23Z"
```


- }

---

4. Payment Management

4.1 Initiate Payment

- **Endpoint**: `POST /payments`
- **Description**: Initiates a payment using Stripe.
- **Authentication**: Required.

Request Body:


```
{
 "orderId": "abc123",
 "amount": 74.97,
 "paymentMethod": "card"
```


- }

Response:


```
{
 "message": "Payment initiated successfully",
 "paymentId": "pi_1Gq3Xo2eZvKYlo2C9kKFe9tH"
```


- }

4.2 Handle Stripe Webhook

- **Endpoint**: `POST /webhooks/stripe`
- **Description**: Handles Stripe webhook events to update order statuses.
- **Request Body**: Stripe event payload.

Response:

```
{
  "message": "Webhook processed successfully"
```

- }

---

5. Shipment Tracking

5.1 Track Shipment

- **Endpoint**: `GET /shipments/:trackingId`
- **Description**: Retrieves real-time shipment tracking details.

Response:

```
{
  "trackingId": "1Z999AA10123456784",
  "status": "In Transit",
  "estimatedDelivery": "2025-01-15T18:00:00Z",
  "lastUpdated": "2025-01-13T12:30:00Z"
```

- }

Error Handling

Error Response Format:

```
{
  "error": "Error description",
  "code": 400
```

- }
- **Common Error Codes**:
  - 400: Bad Request
  - 401: Unauthorized
  - 403: Forbidden
  - 404: Not Found
  - 500: Internal Server Error

Note : Will update more.