ShopKit API Documentation

Version 1.0.0

March 1, 2026

Base URL: `http://localhost:3000`

Swagger UI: `http://localhost:3000/api-docs`

RESTful e-commerce backend API for the ShopKit iOS application.
Supports two user roles: **Customer** and **Seller**.

Contents

Overview

ShopKit is a RESTful backend API designed for an iOS e-commerce application. It provides:

- **Authentication** — JWT-based registration and login with role-based access control.
- **Product Management** — Full CRUD for sellers with ownership protection.
- **Order Processing** — ACID-transactional order creation with stock verification, price snapshotting, and payment integration.
- **Order Fulfillment** — Seller-scoped order viewing and shipment tracking.

Tech Stack

Component	Technology
Runtime	Node.js
Framework	Express.js
Database	PostgreSQL 16
ORM	Prisma
Authentication	JWT (JSON Web Tokens)
Containerization	Docker, Docker Compose
Documentation	Swagger UI (OpenAPI 3.0.3)

Authentication

All protected endpoints require a `Bearer` token in the `Authorization` header:

```
Authorization: Bearer <JWT_TOKEN>
```

Tokens are issued on registration and login, and contain the user's `id` and `role`.

Error Response Format

All errors follow a consistent JSON format:

```
{
  "success": false,
  "message": "Description of the error"
}
```

Health Check

GET /api/health

Returns the API health status. No authentication required.

Response `200 OK`

```
{
  "success": true,
  "message": "OK",
  "timestamp": "2026-02-26T08:26:12.000Z"
}
```

Authentication

POST /api/auth/register

Register a new user account.

Request Body

Field	Type	Required	Description
`email`	string	*	Valid email address
`password`	string	*	Password (min 6 characters)
`role`	string	*	`CUSTOMER` or `SELLER`

```
{
  "email": "user@example.com",
  "password": "securePassword123",
  "role": "CUSTOMER"
}
```

Response `201 Created`

```
{
  "success": true,
  "data": {
    "id": "550e8400-e29b-41d4-a716-446655440000",
    "email": "user@example.com",
    "role": "CUSTOMER",
    "token": "eyJhbGciOiJIUzI1NiIs..."
  }
}
```

Error Responses

Status	Description
400	Missing required fields or invalid role
409	Email already registered

POST /api/auth/login

Authenticate and receive a JWT token.

Request Body

Field	Type	Required	Description
`email`	string	*	Registered email
`password`	string	*	Account password

```
{
  "email": "user@example.com",
  "password": "securePassword123"
}
```

Response 200 OK

```
{
  "success": true,
  "data": {
    "id": "550e8400-e29b-41d4-a716-446655440000",
    "email": "user@example.com",
    "role": "CUSTOMER",
    "token": "eyJhbGciOiJIUzI1NiIs..."
  }
}
```

Error Responses

Status	Description
400	Missing email or password
401	Invalid email or password

Products

GET /api/products

List all products. **Public** — no authentication required.

Response 200 OK

```
{
  "success": true,
  "data": [
    {
      "id": "uuid",
      "seller_id": "uuid",
      "name": "Widget Pro",
      "description": "Premium widget",
      "price": "49.99",
      "stock_quantity": 50,
      "created_at": "2026-02-26T08:26:12.000Z",
      "updated_at": "2026-02-26T08:26:12.000Z",
      "seller": { "id": "uuid", "email": "seller@ex.com" }
    }
  ]
}
```

GET /api/products/:id

Get a single product by ID. **Public**.

Path Parameters

Parameter	Type	Description
id	UUID	Product ID

Response 200 OK

```
{
  "success": true,
  "data": {
    "id": "550e8400-e29b-41d4-a716-446655440000",
    "seller_id": "uuid",
    "name": "Widget Pro",
    "description": "Premium widget",
    "price": "49.99",
    "stock_quantity": 50,
    "created_at": "2026-02-26T08:26:12.000Z",
    "updated_at": "2026-02-26T08:26:12.000Z",
    "seller": { "id": "uuid", "email": "seller@ex.com" }
  }
}
```

Error Responses

Status	Description
404	Product not found

POST /api/products

Create a new product. **Auth:** Bearer Token. **Role:** SELLER only.

Request Body

Field	Type	Required	Description
name	string	*	Product name
description	string		Product description
price	decimal	*	Unit price (e.g. 49.99)
stock_quantity	integer	*	Available stock count

```
{
  "name": "Widget Pro",
  "description": "Premium widget",
  "price": 49.99,
  "stock_quantity": 50
}
```

Response 201 Created

Returns the created product object (without seller relation).

Error Responses

Status	Description
400	Missing required fields or negative values
401	Not authenticated
403	Not a seller

PUT /api/products/:id

Update a product. **Auth:** Bearer Token. **Role:** `SELLER` (owner only).

Only the seller who created the product can update it. All fields are optional.

Request Body

```
{
  "name": "Widget Pro v2",
  "price": 59.99
}
```

Response 200 OK

```
{
  "success": true,
  "data": {
    "id": "550e8400-e29b-41d4-a716-446655440000",
    "seller_id": "uuid",
    "name": "Widget Pro v2",
    "description": "Premium widget",
    "price": "59.99",
    "stock_quantity": 50,
    "created_at": "2026-02-26T08:26:12.000Z",
    "updated_at": "2026-02-27T09:00:00.000Z"
  }
}
```

Error Responses

Status	Description
401	Not authenticated
403	Not the product owner, or not a seller
404	Product not found

DELETE /api/products/:id

Delete a product. **Auth:** Bearer Token. **Role:** `SELLER` (owner only).

Response 200 OK

```
{
  "success": true,
  "message": "Product deleted"
}
```

Error Responses

Status	Description
401	Not authenticated
403	Not the product owner, or not a seller
404	Product not found

Orders (Customer)

POST /api/orders

Create a new order. **Auth:** Bearer Token. **Role:** `CUSTOMER` only.

This endpoint runs inside an **ACID database transaction**:

1. Verify stock availability for each item.
2. **Snapshot** the product's current `unit_price` into the order item.
3. Deduct stock quantities.
4. Calculate `total_amount`.
5. Process payment via the Dummy payment service.
6. If payment **succeeds**: order status → `PAID`, transaction record created, DB committed.
7. If payment **fails**: entire transaction **rolled back** (stock restored, no order persisted).

Request Body

```
{
  "items": [
    { "product_id": "uuid-1", "quantity": 2 },
    { "product_id": "uuid-2", "quantity": 1 }
  ]
}
```

Response `201 Created`

```
{
  "success": true,
  "data": {
    "id": "order-uuid",
    "customer_id": "uuid",
    "total_amount": "75.00",
    "status": "PAID",
    "tax_rate": "0.0000",
    "invoice_url": null,
    "created_at": "2026-02-27T10:00:00.000Z",
    "updated_at": "2026-02-27T10:00:00.000Z",
    "items": [
      {
        "id": "item-uuid",
        "order_id": "order-uuid",
        "product_id": "uuid-1",
        "quantity": 2,
        "unit_price": "25.00"
      }
    ]
  }
}
```

Error Responses

Status	Description
400	Empty or invalid items array
402	Payment failed (transaction rolled back, stock unchanged)
403	Not a customer
404	Product not found
409	Insufficient stock

GET /api/orders

List the authenticated customer's orders. **Auth:** Bearer Token. **Role:** `CUSTOMER`.

Includes order items (with nested product name) and transaction records.

Response `200 OK`


```
{
  "success": true,
  "data": [
    {
      "id": "order-uuid",
      "customer_id": "uuid",
      "total_amount": "75.00",
      "status": "PAID",
      "tax_rate": "0.0000",
      "invoice_url": null,
      "created_at": "2026-02-27T10:00:00.000Z",
      "updated_at": "2026-02-27T10:00:00.000Z",
      "items": [
        {
          "id": "item-uuid",
          "order_id": "order-uuid",
          "product_id": "uuid-1",
          "quantity": 2,
          "unit_price": "25.00",
          "product": { "id": "uuid-1", "name": "Widget Pro" }
        }
      ],
      "transactions": [
        {
          "id": "txn-uuid",
          "order_id": "order-uuid",
          "amount": "75.00",
          "status": "SUCCESS",
          "provider": "DummyPay",
          "created_at": "2026-02-27T10:00:00.000Z"
        }
      ]
    }
  ]
}
```

Seller Fulfillment

GET /api/seller/orders

List all orders that contain the authenticated seller's products. **Auth:** Bearer Token. **Role:** `SELLER`.

Response `200 OK`

Returns an array of orders where at least one order item references a product owned by this seller. Includes customer info, all items, and transaction records.

Error Responses

Status	Description
`401`	Not authenticated
`403`	Not a seller

PATCH /api/seller/orders/:id/ship

Mark an order as `SHIPPED`. **Auth:** Bearer Token. **Role:** `SELLER`.

Only orders in `PAID` status that contain the seller's products can be shipped.

Path Parameters

Parameter	Type	Description
`id`	UUID	Order ID

Response `200 OK`

```
{
  "success": true,
  "data": {
    "id": "order-uuid",
    "status": "SHIPPED",
    "items": [...]
  }
}
```

Error Responses

Status	Description
`400`	Order is not in `PAID` status
`403`	Order does not contain this seller's products
`404`	Order not found

Database Schema

Entity Relationship

`User`	⟶	`Product`	(1 seller → many products)
`User`	⟶	`Order`	(1 customer → many orders)
`Order`	⟶	`OrderItem`	(1 order → many items)
`Order`	⟶	`Transaction`	(1 order → many transactions)
`Product`	⟶	`OrderItem`	(referenced in items)

Users

Column	Type	Description
`id`	UUID	Primary key
`email`	String	Unique email
`password_hash`	String	bcrypt hash (12 rounds)
`role`	Enum	`CUSTOMER` | `SELLER`
`created_at`	DateTime	Auto-set on creation
`updated_at`	DateTime	Auto-updated

Products

Column	Type	Description
`id`	UUID	Primary key
`seller_id`	UUID (FK)	References `Users.id`
`name`	String	Product name
`description`	String?	Optional description
`price`	Decimal(10,2)	Unit price
`stock_quantity`	Integer	Available stock (default: 0)
`created_at`	DateTime	Auto-set on creation
`updated_at`	DateTime	Auto-updated

Orders

Column	Type	Description
`id`	UUID	Primary key
`customer_id`	UUID (FK)	References `Users.id`
`total_amount`	Decimal(10,2)	Calculated total
`status`	Enum	`PENDING` | `PAID` | `SHIPPED` | `CANCELLED`
`tax_rate`	Decimal(5,4)	Tax rate for invoicing (default: 0)
`invoice_url`	String?	Future invoice worker URL
`created_at`	DateTime	Auto-set on creation
`updated_at`	DateTime	Auto-updated

OrderItems

Column	Type	Description
`id`	UUID	Primary key
`order_id`	UUID (FK)	References `Orders.id`
`product_id`	UUID (FK)	References `Products.id`
`quantity`	Integer	Quantity ordered
`unit_price`	Decimal(10,2)	**Snapshot** of price at purchase time

Transactions

Column	Type	Description
`id`	UUID	Primary key
`order_id`	UUID (FK)	References `Orders.id`
`amount`	Decimal(10,2)	Payment amount
`status`	Enum	`SUCCESS` | `FAILED`
`provider`	String	Default: `DummyPay`
`created_at`	DateTime	Auto-set on creation

Endpoint Summary

Method	Endpoint	Description	Auth	Role
GET	`/api/health`	Health check	–	–
POST	`/api/auth/register`	Register user	–	–
POST	`/api/auth/login`	Login	–	–
GET	`/api/products`	List products	–	–
GET	`/api/products/:id`	Get product	–	–
POST	`/api/products`	Create product	Yes	Seller
PUT	`/api/products/:id`	Update product	Yes	Seller†
DELETE	`/api/products/:id`	Delete product	Yes	Seller†
POST	`/api/orders`	Create order (ACID)	Yes	Customer
GET	`/api/orders`	List my orders	Yes	Customer
GET	`/api/seller/orders`	Seller's orders	Yes	Seller
PATCH	`/api/seller/orders/:id/ship`	Ship order	Yes	Seller
GET	`/api-docs`	Swagger UI	–	–

† Owner only — the seller must own the product.