# Pro-Level YAML Sample: openapi-shopsphere.yaml

```yaml
openapi: 3.1.0
info:
  title: ShopSphere Cloud API
  version: 1.0.0
  description: |
    ShopSphere is a scalable, cloud-native e-commerce platform API enabling
    multi-vendor, multi-region online sales with built-in support for authentication,
    analytics, order tracking, and secure payments.

    This OpenAPI spec provides complete documentation of the RESTful endpoints,
    schemas, security models, and reusable components.

  contact:
    name: Developer Relations
    email: devrel@shopsphere.io
    url: https://shopsphere.io/docs

  termsOfService: https://shopsphere.io/legal/terms
  license:
    name: Apache 2.0
    url: https://www.apache.org/licenses/LICENSE-2.0
```

```
servers:
  - url: https://api.shopsphere.io/v1
    description: Production Server (v1)
  - url: https://staging-api.shopsphere.io/v1
    description: Staging Server

tags:
  - name: Products
    description: Manage and browse product catalogs
  - name: Orders
    description: Handle customer orders and checkout
  - name: Auth
    description: Authentication and authorization
  - name: Vendors
    description: Vendor management and account details

paths:
  /products:
    get:
      tags: [Products]
      summary: List all available products
      description: |
        Retrieve a paginated list of products available for sale.
        Supports filtering by category, price range, and availability.
```

```
operationId: listProducts
parameters:
  - name: category
    in: query
    schema:
      type: string
    description: Filter by product category
  - name: price_min
    in: query
    schema:
      type: number
      format: float
    description: Minimum price filter
  - name: price_max
    in: query
    schema:
      type: number
      format: float
    description: Maximum price filter
  - $ref: '#/components/parameters/PageParam'
  - $ref: '#/components/parameters/PageSizeParam'
responses:
  '200':
    description: List of products
    content:
```

```
          application/json:
            schema:
              $ref: '#/components/schemas/ProductList'
      '400':
        $ref: '#/components/responses/BadRequest'
      '500':
        $ref: '#/components/responses/InternalServerError'

/orders:
  post:
    tags: [Orders]
    summary: Create a new order
    requestBody:
      description: Order payload
      required: true
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/NewOrder'
    responses:
      '201':
        description: Order successfully created
        content:
          application/json:
            schema:
```

```
            $ref: '#/components/schemas/OrderConfirmation'
        '400':
          $ref: '#/components/responses/BadRequest'
        '401':
          $ref: '#/components/responses/Unauthorized'
        '500':
          $ref: '#/components/responses/InternalServerError'
      security:
        - bearerAuth: []

components:
  securitySchemes:
    bearerAuth:
      type: http
      scheme: bearer
      bearerFormat: JWT
      description: |
        JWT token required for access to protected endpoints.
        Use `/auth/token` to obtain a valid token.

  parameters:
    PageParam:
      name: page
      in: query
      schema:
```

```
      type: integer
      minimum: 1
      default: 1
    description: Page number for pagination
  PageSizeParam:
    name: pageSize
    in: query
    schema:
      type: integer
      minimum: 1
      maximum: 100
      default: 20
    description: Number of items per page

responses:
  BadRequest:
    description: Invalid request
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/ErrorResponse'
  Unauthorized:
    description: Authentication required
    content:
      application/json:
```

```
        schema:
          $ref: '#/components/schemas/ErrorResponse'
    InternalServerError:
      description: Internal server error
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/ErrorResponse'

  schemas:
    Product:
      type: object
      required: [id, name, price, currency]
      properties:
        id:
          type: string
          format: uuid
        name:
          type: string
        description:
          type: string
        price:
          type: number
          format: float
        currency:
```

```
      type: string
      enum: [USD, EUR, GBP, INR, JPY]
    available:
      type: boolean
    tags:
      type: array
      items:
        type: string
    vendorId:
      type: string
      format: uuid

ProductList:
  type: object
  properties:
    total:
      type: integer
    page:
      type: integer
    pageSize:
      type: integer
    items:
      type: array
      items:
        $ref: '#/components/schemas/Product'
```

```
NewOrder:
  type: object
  required: [customerId, items, payment]
  properties:
    customerId:
      type: string
      format: uuid
    items:
      type: array
      items:
        type: object
        required: [productId, quantity]
        properties:
          productId:
            type: string
            format: uuid
          quantity:
            type: integer
            minimum: 1
    shipping:
      type: object
      properties:
        address:
          type: string
```

```
        postalCode:
          type: string
        country:
          type: string
    payment:
      oneOf:
        - $ref: '#/components/schemas/CreditCardPayment'
        - $ref: '#/components/schemas/PayPalPayment'
      discriminator:
        propertyName: method

CreditCardPayment:
  type: object
  required: [method, cardNumber, expiryDate, cvv]
  properties:
    method:
      type: string
      enum: [credit_card]
    cardNumber:
      type: string
    expiryDate:
      type: string
      format: date
    cvv:
      type: string
```

```
PayPalPayment:
  type: object
  required: [method, email]
  properties:
    method:
      type: string
      enum: [paypal]
    email:
      type: string
      format: email

OrderConfirmation:
  type: object
  properties:
    orderId:
      type: string
      format: uuid
    status:
      type: string
      enum: [pending, confirmed, shipped, delivered]
    estimatedDelivery:
      type: string
      format: date
```

```
ErrorResponse:
  type: object
  properties:
    code:
      type: string
    message:
      type: string
    traceId:
      type: string
```