Technical Design Document: E-Commerce Backend Architecture

Project Name: E-Commerce Platform MVP

Version: 1.0

Date: November 24, 2025

Tech Stack: Next.js (Frontend), FastAPI (Backend), MongoDB (Database)

---

1. Database Schema Design (MongoDB)

This architecture utilizes a **Document-Oriented** approach. We prioritize embedded documents for read-performance (Products/Users) and referencing for scalability (Orders/Carts).

1.1 Collection: users

- **Purpose:** Stores authentication credentials and user profiles.
- **Key Design:** Addresses are embedded to reduce query complexity during checkout.

JSON

```
{
  "_id": "ObjectId",
  "name": "String (Required)",
  "email": "String (Unique, Required)",
  "password_hash": "String (Bcrypt)",
  "role": "String ('customer' | 'admin')",
  "created_at": "ISODate",
  "addresses": [
    {
      "type": "String ('shipping' | 'billing')",
      "street": "String",
      "city": "String",
      "state": "String",
      "zip": "String",
```

```
    "phone": "String"
  }
 ]
}
```

1.2 Collection: products

- **Purpose:** Central catalog for store items.
- **Key Design:** Uses an embedded variants array to manage SKUs (e.g., Sizes/Colors) within a single parent document.

JSON

```
{
 "_id": "ObjectId",
 "name": "String (Index)",
 "description": "String",
 "category": "String (Index)",
 "base_price": "Number",
 "images": ["String (URL array)"],
 "is_active": "Boolean",
 "variants": [
  {
   "sku": "String (Unique)",
   "size": "String",
   "color": "String",
   "stock_quantity": "Number",
   "price_modifier": "Number (Default 0)"
  }
 ]
}
```

1.3 Collection: carts

- **Purpose:** Temporary storage for user sessions.
- **Key Design:** Separated from users to allow for session cleanup and lighter user objects.

JSON

```
{
 "_id": "ObjectId",
 "user_id": "ObjectId (Ref: users)",
 "items": [
  {
   "product_id": "ObjectId (Ref: products)",
   "variant_sku": "String",
   "quantity": "Number",
   "thumbnail": "String (URL)"
  }
 ],
 "updated_at": "ISODate (TTL Index: Expires in 30 days)"
}
```

1.4 Collection: orders

- **Purpose:** Permanent financial records.
- **Key Design:** Uses the **Snapshot Pattern**. Product details (price, name) are copied into the order at the time of purchase to preserve historical accuracy.

JSON

```
{
 "_id": "ObjectId",
 "user_id": "ObjectId (Ref: users)",
 "transaction_id": "String (Payment Gateway ID)",
 "status": "String ('pending', 'processing', 'shipped', 'delivered')",
 "total_amount": "Number",
 "created_at": "ISODate",
```

```
 "shipping_address": { "Object (Snapshot of user address)" },
 "items": [
  {
   "product_id": "ObjectId",
   "name": "String",
   "sku": "String",
   "quantity": "Number",
   "price_per_unit": "Number (Frozen Price)"
  }
 ]
}
```

---

2. API Endpoint Specification (FastAPI)

All API responses will return JSON. Protected routes require a header: Authorization: Bearer <token>.

2.1 Authentication Module

Method	Endpoint	Description	Access Level
POST	/api/v1/auth/register	Create a new user account	Public
POST	/api/v1/auth/login	Authenticate and retrieve JWT Token	Public
GET	/api/v1/auth/me	Retrieve current user profile	**User**

2.2 Product Module

Method	Endpoint	Description	Access Level
GET	/api/v1/products	List all products (supports filtering)	Public
GET	/api/v1/products/{id}	Get detailed info of one product	Public
POST	/api/v1/products	Create a new product	**Admin**
PUT	/api/v1/products/{id}	Update product details	**Admin**

Method	Endpoint	Description	Access Level
DELETE	/api/v1/products/{id}	Soft delete a product	**Admin**

2.3 Cart Module

Method	Endpoint	Description	Access Level
GET	/api/v1/cart	Get current user's cart	**User**
POST	/api/v1/cart/items	Add item to cart	**User**
PUT	/api/v1/cart/items/{id}	Update item quantity	**User**
DELETE	/api/v1/cart/items/{id}	Remove item from cart	**User**

2.4 Order Module

Method	Endpoint	Description	Access Level
POST	/api/v1/orders	Checkout (Create Order & Clear Cart)	**User**
GET	/api/v1/orders	Get purchase history for user	**User**
GET	/api/v1/orders/{id}	Get specific order receipt	**User**

2.5 Admin Management Module

Method	Endpoint	Description	Access Level
GET	/api/v1/admin/orders	View all orders across the platform	**Admin**
PUT	/api/v1/admin/orders/{id}	Update order status (e.g., to 'Shipped')	**Admin**
GET	/api/v1/admin/stats	Retrieve sales analytics and KPIs	**Admin**

---

3. Technical Implementation Strategy

3.1 Security & Authentication

- **JWT (JSON Web Tokens):** Used for stateless authentication.
- **Hashing:** Passwords will be hashed using bcrypt before storage.

- **Role-Based Access Control (RBAC):** Middleware will intercept requests to /admin routes and verify if user.role == 'admin'.

3.2 Performance Optimization

- **Indexing:** MongoDB indexes will be applied to email (Users), category (Products), and user_id (Orders) to ensure fast query speeds.
- **Projection:** API endpoints will use MongoDB projection to return only necessary fields (e.g., listing products excludes the full description and reviews arrays to save bandwidth).

3.3 Third-Party Integrations

- **Payment:** (Planned) Stripe/Razorpay integration. The backend will generate payment intents and verify webhooks.
- **Image Hosting:** (Planned) AWS S3 or Cloudinary for storing product images; Database will only store the URL strings.