E-commere website backend and frontend both

E-commere website backend and frontend both

Project over view

Tech stack

- **Frontend:** Next.js
- **Backend:** Express.js, Node.js
- **Database:** MongoDB
- **Payment Gateway:** Stripe (in testing mode)

Backend routes

E Commerce API Documentation

Base URL: `http://localhost:8000/api/v1`
Content-Type: `application/json`-----Table of Contents

Authentication RoutesBase Path: `/auth`

```
# ECommerce API Documentation

**Version:** 1.0.0
**Base URL:** `http://localhost:8000/api/v1`
**Content-Type:** `application/json`

---

## Table of Contents

1. [Authentication Routes](#authentication-routes)
2. [Product Routes](#product-routes)
3. [Cart Routes](#cart-routes)
4. [Payment Routes](#payment-routes)
5. [Admin Routes](#admin-routes)
6. [Error Handling](#error-handling)
```

~~~markdown
7. [Status Codes](#status-codes)

---

## Authentication Routes

### Base Path: `/auth`

#### 1. User Registration

**Endpoint:** `POST /auth/register`
**Authentication:** Not Required
**Description:** Register a new user account

**Request Body:**
```json
{
  "username": "john_doe",
  "email": "john@example.com",
  "fullName": "John Doe",
  "password": "securePassword123"
}
```

**Request Validation:**
- `username` (string): 3-20 characters, alphanumeric and underscore
only
- `email` (string): Valid email format
- `fullName` (string): Required, non-empty
- `password` (string): Minimum 6 characters

**Success Response (201 Created):**
```json
{
  "statusCode": 201,
  "data": {
    "user": {
      "_id": "507f1f77bcf86cd799439011",
      "username": "john_doe",
      "email": "john@example.com",
      "fullName": "John Doe",
      "createdAt": "2024-11-17T10:30:00Z"
    },
~~~

~~~markdown
    "accessToken": "eyJhbGciOiJIUzI1NiIs...",
    "refreshToken": "eyJhbGciOiJIUzI1NiIs..."
  },
  "message": "User registered successfully"
}
```

**Error Response (400):**
```json
{
  "statusCode": 400,
  "message": "Username must be 3-20 characters, alphanumeric and underscore only",
  "errors": []
}
```

**Error Response (409 Conflict):**
```json
{
  "statusCode": 409,
  "message": "User with this username or email already exists",
  "errors": []
}
```

---

#### 2. User Login

**Endpoint:** `POST /auth/login`
**Authentication:** Not Required
**Description:** Authenticate user and get access tokens

**Request Body:**
```json
{
  "username": "john_doe",
  "email": "john@example.com",
  "password": "securePassword123"
}
```
~~~

~~~markdown
**Request Validation:**
- Either `username` OR `email` is required (at least one)
- `password` (string): Required

**Success Response (200 OK):**
```json
{
  "statusCode": 200,
  "data": {
    "user": {
      "_id": "507f1f77bcf86cd799439011",
      "username": "john_doe",
      "email": "john@example.com",
      "fullName": "John Doe"
    },
    "accessToken": "eyJhbGciOiJIUzI1NiIs...",
    "refreshToken": "eyJhbGciOiJIUzI1NiIs..."
  },
  "message": "User logged in successfully"
}
```

**Error Response (401):**
```json
{
  "statusCode": 401,
  "message": "Invalid username/email or password",
  "errors": []
}
```


---

#### 3. Refresh Access Token

**Endpoint:** `POST /auth/refresh-token`
**Authentication:** Not Required
**Description:** Get a new access token using refresh token

**Request Body:**
```json
{
  "refreshToken": "eyJhbGciOiJIUzI1NiIs..."
~~~

~~~
}
```

**Success Response (200 OK):**
```json
{
  "statusCode": 200,
  "data": {
    "accessToken": "eyJhbGciOiJIUzI1NiIs..."
  },
  "message": "Access token refreshed successfully"
}
```

**Error Response (401):**
```json
{
  "statusCode": 401,
  "message": "Invalid or expired refresh token",
  "errors": []
}
```

---

#### 4. Admin Registration

**Endpoint:** `POST /auth/register-admin`
**Authentication:** Not Required
**Description:** Register a new admin account (requires admin secret
key)

**Request Body:**
```json
{
  "username": "admin_user",
  "email": "admin@example.com",
  "fullName": "Admin User",
  "password": "adminPassword123",
  "adminSecret": "MySecretAdminKey2024!"
}
```
~~~

~~~markdown
**Request Validation:**
- `adminSecret` must match `ADMIN_SECRET_KEY` in server `.env`
- All other validations same as user registration

**Success Response (201 Created):**
```json
{
  "statusCode": 201,
  "data": {
    "user": {
      "_id": "507f1f77bcf86cd799439012",
      "username": "admin_user",
      "email": "admin@example.com",
      "fullName": "Admin User",
      "role": "admin",
      "createdAt": "2024-11-17T10:30:00Z"
    },
    "accessToken": "eyJhbGciOiJIUzI1NiIs...",
    "refreshToken": "eyJhbGciOiJIUzI1NiIs..."
  },
  "message": "Admin registered successfully"
}
```

**Error Response (401):**
```json
{
  "statusCode": 401,
  "message": "Invalid admin secret key",
  "errors": []
}
```

---

#### 5. User Logout

**Endpoint:** `POST /auth/logout`
**Authentication:** Required (Bearer Token)
**Description:** Logout user and invalidate refresh token

**Headers:**
```
~~~

~~~~markdown
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
```

**Request Body:** Empty

**Success Response (200 OK):**
```json
{
  "statusCode": 200,
  "data": {},
  "message": "User logged out successfully"
}
```

**Error Response (401):**
```json
{
  "statusCode": 401,
  "message": "User not authenticated",
  "errors": []
}
```

---

## Product Routes

### Base Path: `/products`

#### 1. Get All Products

**Endpoint:** `GET /products/getall`
**Authentication:** Not Required
**Description:** Retrieve all products with optional filtering and
pagination

**Query Parameters:**
```
?page=1&limit=10&category=Electronics&minPrice=100&maxPrice=1000
```

**Query Details:**
- `page` (number): Page number (default: 1)
~~~~

~~~~markdown
- `limit` (number): Items per page (default: 10)
- `category` (string): Filter by category (optional)
- `minPrice` (number): Minimum price filter (optional)
- `maxPrice` (number): Maximum price filter (optional)

**Success Response (200 OK):**
```json
{
  "statusCode": 200,
  "data": {
    "products": [
      {
        "_id": "507f1f77bcf86cd799439013",
        "productName": "iPhone 15",
        "price": 999.99,
        "description": "Latest Apple smartphone",
        "category": "Electronics",
        "rating": 4.5,
        "createdAt": "2024-11-17T10:30:00Z"
      }
    ],
    "pagination": {
      "currentPage": 1,
      "totalPages": 5,
      "totalProducts": 45,
      "limit": 10
    }
  },
  "message": "Products retrieved successfully"
}
```

---

#### 2. Search Products

**Endpoint:** `GET /products/search`
**Authentication:** Not Required
**Description:** Search products by name, description, or category

**Query Parameters:**
```
?query=iPhone&page=1&limit=10
~~~~

~~~
```

**Query Details:**
- `query` (string): Search term (required)
- `page` (number): Page number (default: 1)
- `limit` (number): Items per page (default: 10)

**Success Response (200 OK):**
```json
{
  "statusCode": 200,
  "data": {
    "products": [
      {
        "_id": "507f1f77bcf86cd799439013",
        "productName": "iPhone 15",
        "price": 999.99,
        "description": "Latest Apple smartphone",
        "category": "Electronics",
        "rating": 4.5
      }
    ],
    "pagination": {
      "currentPage": 1,
      "totalPages": 1,
      "totalProducts": 1,
      "limit": 10
    }
  },
  "message": "Search results retrieved successfully"
}
```

---

#### 3. Get Product by ID

**Endpoint:** `GET /products/:id`
**Authentication:** Not Required
**Description:** Retrieve a single product by ID

**URL Parameters:**
- `id` (string): Product MongoDB ID
~~~

~~~markdown
**Example Request:**
```
GET /products/507f1f77bcf86cd799439013
```

**Success Response (200 OK):**
```json
{
  "statusCode": 200,
  "data": {
    "_id": "507f1f77bcf86cd799439013",
    "productName": "iPhone 15",
    "price": 999.99,
    "description": "Latest Apple smartphone with 5G",
    "category": "Electronics",
    "rating": 4.5,
    "createdAt": "2024-11-17T10:30:00Z"
  },
  "message": "Product retrieved successfully"
}
```

**Error Response (404):**
```json
{
  "statusCode": 404,
  "message": "Product not found",
  "errors": []
}
```

---

#### 4. Add Product (Admin Only)

**Endpoint:** `POST /products/add`
**Authentication:** Required (Bearer Token)
**Authorization:** Admin role required
**Description:** Create a new product

**Headers:**
```
~~~

~~~
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
```

**Request Body:**
```json
{
  "productName": "iPhone 15",
  "price": 999.99,
  "description": "Latest Apple smartphone with 5G connectivity",
  "category": "Electronics",
  "rating": 0
}
```

**Request Validation:**
- `productName` (string): Required, non-empty
- `price` (number): Required, must be positive
- `description` (string): Required, non-empty
- `category` (string): Required, non-empty
- `rating` (number): Optional, must be between 0-5

**Success Response (201 Created):**
```json
{
  "statusCode": 201,
  "data": {
    "_id": "507f1f77bcf86cd799439013",
    "productName": "iPhone 15",
    "price": 999.99,
    "description": "Latest Apple smartphone with 5G connectivity",
    "category": "Electronics",
    "rating": 0,
    "createdAt": "2024-11-17T10:30:00Z"
  },
  "message": "Product added successfully"
}
```

**Error Response (400):**
```json
{
  "statusCode": 400,
  "message": "Please provide all required fields (productName, price,
~~~

~~~markdown
description, category)",
  "errors": []
}
```

---

#### 5. Update Product (Admin Only)

**Endpoint:** `PUT /products/:id`
**Authentication:** Required (Bearer Token)
**Authorization:** Admin role required
**Description:** Update product details

**Headers:**
```
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
```

**URL Parameters:**
- `id` (string): Product MongoDB ID

**Request Body:** (All fields optional)
```json
{
  "productName": "iPhone 15 Pro",
  "price": 1099.99,
  "description": "Updated description",
  "category": "Premium Electronics",
  "rating": 4.5
}
```

**Success Response (200 OK):**
```json
{
  "statusCode": 200,
  "data": {
    "_id": "507f1f77bcf86cd799439013",
    "productName": "iPhone 15 Pro",
    "price": 1099.99,
    "description": "Updated description",
    "category": "Premium Electronics",
~~~

~~~markdown
    "rating": 4.5
  },
  "message": "Product updated successfully"
}
```

---

#### 6. Delete Product (Admin Only)

**Endpoint:** `DELETE /products/:id`
**Authentication:** Required (Bearer Token)
**Authorization:** Admin role required
**Description:** Delete a product

**Headers:**
```
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
```

**URL Parameters:**
- `id` (string): Product MongoDB ID

**Success Response (200 OK):**
```json
{
  "statusCode": 200,
  "data": {
    "_id": "507f1f77bcf86cd799439013",
    "productName": "iPhone 15",
    "price": 999.99
  },
  "message": "Product deleted successfully"
}
```

---

## Cart Routes

### Base Path: `/cart`

**Note:** All cart routes require authentication
~~~

~~~markdown
#### 1. Get User's Cart

**Endpoint:** `GET /cart`
**Authentication:** Required (Bearer Token)
**Description:** Retrieve the current user's cart

**Headers:**
```
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
```

**Success Response (200 OK):**
```json
{
  "statusCode": 200,
  "data": {
    "_id": "507f1f77bcf86cd799439020",
    "userId": "507f1f77bcf86cd799439011",
    "items": [
      {
        "productId": "507f1f77bcf86cd799439013",
        "productName": "iPhone 15",
        "price": 999.99,
        "quantity": 1,
        "totalPrice": 999.99
      }
    ],
    "totalAmount": 999.99,
    "totalItems": 1
  },
  "message": "Cart retrieved successfully"
}
```

---

#### 2. Get Cart Count

**Endpoint:** `GET /cart/count`
**Authentication:** Required (Bearer Token)
**Description:** Get the number of items in cart
~~~

~~~markdown
**Headers:**
```
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
```

**Success Response (200 OK):**
```json
{
  "statusCode": 200,
  "data": {
    "cartCount": 3
  },
  "message": "Cart count retrieved successfully"
}
```

---

#### 3. Get Cart Summary

**Endpoint:** `GET /cart/summary`
**Authentication:** Required (Bearer Token)
**Description:** Get cart total amount and item count

**Headers:**
```
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
```

**Success Response (200 OK):**
```json
{
  "statusCode": 200,
  "data": {
    "totalItems": 3,
    "totalAmount": 2999.97
  },
  "message": "Cart summary retrieved successfully"
}
```

---
~~~

~~~markdown
#### 4. Add to Cart

**Endpoint:** `POST /cart/add`
**Authentication:** Required (Bearer Token)
**Description:** Add product to cart or increase quantity

**Headers:**
```
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
```

**Request Body:**
```json
{
  "productId": "507f1f77bcf86cd799439013",
  "quantity": 2
}
```

**Request Validation:**
- `productId` (string): Valid MongoDB ID required
- `quantity` (number): Must be positive integer

**Success Response (200 OK):**
```json
{
  "statusCode": 200,
  "data": {
    "_id": "507f1f77bcf86cd799439020",
    "userId": "507f1f77bcf86cd799439011",
    "items": [
      {
        "productId": "507f1f77bcf86cd799439013",
        "productName": "iPhone 15",
        "price": 999.99,
        "quantity": 2,
        "totalPrice": 1999.98
      }
    ],
    "totalAmount": 1999.98,
    "totalItems": 2
  },
  "message": "Product added to cart successfully"
~~~

~~~
}
```

**Error Response (404):**
```json
{
  "statusCode": 404,
  "message": "Product not found",
  "errors": []
}
```

---

#### 5. Update Cart Item Quantity

**Endpoint:** `PUT /cart/update`
**Authentication:** Required (Bearer Token)
**Description:** Update quantity of item in cart

**Headers:**
```
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
```

**Request Body:**
```json
{
  "productId": "507f1f77bcf86cd799439013",
  "quantity": 5
}
```

**Request Validation:**
- `quantity` = 0: Will remove the item
- `quantity` > 0: Will update the quantity

**Success Response (200 OK):**
```json
{
  "statusCode": 200,
  "data": {
    "_id": "507f1f77bcf86cd799439020",
~~~

~~~
    "items": [
      {
        "productId": "507f1f77bcf86cd799439013",
        "productName": "iPhone 15",
        "price": 999.99,
        "quantity": 5,
        "totalPrice": 4999.95
      }
    ],
    "totalAmount": 4999.95,
    "totalItems": 5
  },
  "message": "Cart item quantity updated successfully"
}
```

---

#### 6. Remove from Cart

**Endpoint:** `POST /cart/remove`
**Authentication:** Required (Bearer Token)
**Description:** Remove product from cart

**Headers:**
```
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
```

**Request Body:**
```json
{
  "productId": "507f1f77bcf86cd799439013"
}
```

**Success Response (200 OK):**
```json
{
  "statusCode": 200,
  "data": {
    "_id": "507f1f77bcf86cd799439020",
    "items": [],
~~~

~~~markdown
    "totalAmount": 0,
    "totalItems": 0
  },
  "message": "Product removed from cart successfully"
}
```

---

#### 7. Clear Cart

**Endpoint:** `POST /cart/clear`
**Authentication:** Required (Bearer Token)
**Description:** Remove all items from cart

**Headers:**
```
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
```

**Request Body:** Empty

**Success Response (200 OK):**
```json
{
  "statusCode": 200,
  "data": {
    "_id": "507f1f77bcf86cd799439020",
    "items": [],
    "totalAmount": 0,
    "totalItems": 0
  },
  "message": "Cart cleared successfully"
}
```

---

## Payment Routes

### Base Path: `/payment`

**Note:** All payment routes require authentication
~~~

~~~markdown
#### 1. Create Payment Intent

**Endpoint:** `POST /payment/create-intent`
**Authentication:** Required (Bearer Token)
**Description:** Create a Stripe payment intent

**Headers:**
```
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
```

**Request Body:**
```json
{
  "amount": 1999.98,
  "currency": "usd",
  "orderId": "order_12345",
  "userId": "507f1f77bcf86cd799439011",
  "email": "user@example.com"
}
```

**Request Validation:**
- `amount` (number): Must be greater than 0
- `currency` (string): Required, typically "usd"
- `orderId` (string): Unique order identifier
- `email` (string): Required for receipt

**Success Response (200 OK):**
```json
{
  "statusCode": 200,
  "data": {
    "clientSecret": "pi_1234567890_secret_abcdefg",
    "paymentIntentId": "pi_1234567890"
  },
  "message": "Payment intent created successfully"
}
```

**Error Response (400):**
```json
~~~

~~~~
{
  "statusCode": 400,
  "message": "Amount and currency are required",
  "errors": []
}
```

---

#### 2. Confirm Payment

**Endpoint:** `POST /payment/confirm`
**Authentication:** Required (Bearer Token)
**Description:** Confirm payment after successful Stripe processing

**Headers:**
```
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
```

**Request Body:**
```json
{
  "paymentIntentId": "pi_1234567890",
  "orderId": "order_12345"
}
```

**Request Validation:**
- `paymentIntentId` (string): Required
- `orderId` (string): Optional, for order tracking

**Success Response (200 OK):**
```json
{
  "statusCode": 200,
  "data": {
    "status": "succeeded"
  },
  "message": "Payment confirmed successfully"
}
```
~~~~

Processing Response (202):
```json
{
  "statusCode": 202,
  "data": {
    "status": "processing"
  },
  "message": "Payment is processing"
}
```

Error Response (402):
```json
{
  "statusCode": 402,
  "message": "Payment failed",
  "errors": []
}
```

#### 3. Get Payment Status

Endpoint: `GET /payment/status/:paymentIntentId`
Authentication: Required (Bearer Token)
Description: Check payment status by payment intent ID

Headers:
```
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
```

URL Parameters:
- `paymentIntentId` (string): Stripe payment intent ID

Example Request:
```
GET /payment/status/pi_1234567890
```

Success Response (200 OK):
```json
```

~~~
{
  "statusCode": 200,
  "data": {
    "status": "succeeded",
    "amount": 199998,
    "currency": "usd"
  },
  "message": "Payment status retrieved successfully"
}
```

---

## Admin Routes

### Base Path: `/admin`

**Note:** All admin routes require authentication AND admin role

#### 1. Get Dashboard Statistics

**Endpoint:** `GET /admin/dashboard/stats`
**Authentication:** Required (Bearer Token)
**Authorization:** Admin role required
**Description:** Get overall store statistics

**Headers:**
```
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
```

**Success Response (200 OK):**
```json
{
  "statusCode": 200,
  "data": {
    "totalUsers": 45,
    "totalProducts": 120,
    "totalOrders": 340,
    "totalRevenue": 45678.90,
    "totalProductCategories": 8,
    "categories": [
      "Electronics",
~~~

~~~
      "Fashion",
      "Home",
      "Books",
      "Sports",
      "Toys",
      "Beauty",
      "Food"
    ]
  },
  "message": "Dashboard statistics retrieved successfully"
}
```

---

#### 2. Get All Users

**Endpoint:** `GET /admin/users`
**Authentication:** Required (Bearer Token)
**Authorization:** Admin role required
**Description:** List all users with pagination

**Headers:**
```
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
```

**Query Parameters:**
```
?page=1&limit=10&role=user
```

**Success Response (200 OK):**
```json
{
  "statusCode": 200,
  "data": {
    "users": [
      {
        "_id": "507f1f77bcf86cd799439011",
        "username": "john_doe",
        "email": "john@example.com",
        "fullName": "John Doe",
~~~

~~~~
        "createdAt": "2024-11-17T10:30:00Z"
      }
    ],
    "pagination": {
      "currentPage": 1,
      "totalPages": 5,
      "totalUsers": 45,
      "limit": 10
    }
  },
  "message": "Users retrieved successfully"
}
```

---

#### 3. Get Inventory Summary

**Endpoint:** `GET /admin/inventory/summary`
**Authentication:** Required (Bearer Token)
**Authorization:** Admin role required
**Description:** Get inventory value and category breakdown

**Headers:**
```
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
```

**Success Response (200 OK):**
```json
{
  "statusCode": 200,
  "data": {
    "totalProducts": 120,
    "totalInventoryValue": 456789.50,
    "categorySummary": [
      {
        "_id": "Electronics",
        "count": 35,
        "avgPrice": 899.99
      },
      {
        "_id": "Fashion",
~~~~

~~~
        "count": 45,
        "avgPrice": 49.99
      }
    ],
    "allProducts": [...]
  },
  "message": "Inventory summary retrieved successfully"
}
```

---

#### 4. Get Category Inventory

**Endpoint:** `GET /admin/inventory/category`
**Authentication:** Required (Bearer Token)
**Authorization:** Admin role required
**Description:** Get detailed inventory breakdown by category

**Headers:**
```
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
```

**Success Response (200 OK):**
```json
{
  "statusCode": 200,
  "data": [
    {
      "_id": "Electronics",
      "products": [...],
      "totalProducts": 35,
      "totalValue": 31499.65,
      "avgPrice": 899.99,
      "minPrice": 199.99,
      "maxPrice": 1999.99
    }
  ],
  "message": "Category inventory retrieved successfully"
}
```
~~~

~~~markdown
---

#### 5. Get Low Rating Products

**Endpoint:** `GET /admin/inventory/low-rating`
**Authentication:** Required (Bearer Token)
**Authorization:** Admin role required
**Description:** Get products with low ratings (quality control)

**Headers:**
```
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
```

**Query Parameters:**
```
?minRating=0&maxRating=3
```

**Success Response (200 OK):**
```json
{
  "statusCode": 200,
  "data": [
    {
      "_id": "507f1f77bcf86cd799439014",
      "productName": "Low Rating Product",
      "rating": 2.5,
      "price": 99.99
    }
  ],
  "message": "Low rating products retrieved successfully"
}
```

---

#### 6. Get Products by Price Range

**Endpoint:** `GET /admin/inventory/price-range`
**Authentication:** Required (Bearer Token)
**Authorization:** Admin role required
**Description:** Get products within price range
~~~

~~~~markdown
**Headers:**
```
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
```

**Query Parameters:**
```
?minPrice=100&maxPrice=1000
```

**Success Response (200 OK):**
```json
{
  "statusCode": 200,
  "data": {
    "minPrice": 100,
    "maxPrice": 1000,
    "productsInRange": 45,
    "products": [
      {
        "_id": "507f1f77bcf86cd799439013",
        "productName": "iPhone 15",
        "price": 999.99,
        "category": "Electronics"
      }
    ]
  },
  "message": "Products by price range retrieved successfully"
}
```

---

#### 7. Get All Products (Admin)

**Endpoint:** `GET /admin/products`
**Authentication:** Required (Bearer Token)
**Authorization:** Admin role required
**Description:** List all products for admin

**Headers:**
```
~~~~

~~~markdown
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
```

**Success Response (200 OK):**
```json
{
  "statusCode": 200,
  "data": {
    "products": [...]
  },
  "message": "Products retrieved successfully"
}
```

---

#### 8. Admin Product Management

**Add Product (Admin):**
**Endpoint:** `POST /admin/products`

**Update Product (Admin):**
**Endpoint:** `PUT /admin/products/:id`

**Delete Product (Admin):**
**Endpoint:** `DELETE /admin/products/:id`

*Same request/response format as Product routes above*

---

## Error Handling

### Error Response Format

All error responses follow this standard format:

```json
{
  "statusCode": <HTTP_STATUS_CODE>,
  "message": "<Error Description>",
  "errors": []
}
~~~

~~~
```

### Common Error Scenarios

#### Unauthorized (401)
```json
{
  "statusCode": 401,
  "message": "User not authenticated",
  "errors": []
}
```

#### Forbidden (403)
```json
{
  "statusCode": 403,
  "message": "Access denied - admin role required",
  "errors": []
}
```

#### Bad Request (400)
```json
{
  "statusCode": 400,
  "message": "Invalid input format",
  "errors": []
}
```

#### Not Found (404)
```json
{
  "statusCode": 404,
  "message": "Resource not found",
  "errors": []
}
```

#### Conflict (409)
```json
{
~~~

~~~markdown
  "statusCode": 409,
  "message": "Resource already exists",
  "errors": []
}
```

#### Server Error (500)
```json
{
  "statusCode": 500,
  "message": "Internal server error",
  "errors": []
}
```

---

## Status Codes

| Code | Status | Usage |
|------|--------|-------|
| 200 | OK | Successful GET, PUT, POST |
| 201 | Created | Successful resource creation |
| 202 | Accepted | Request accepted for processing |
| 400 | Bad Request | Invalid input or missing required fields |
| 401 | Unauthorized | Missing or invalid authentication token |
| 402 | Payment Required | Payment failed |
| 403 | Forbidden | Insufficient permissions (e.g., admin role) |
| 404 | Not Found | Resource does not exist |
| 409 | Conflict | Resource already exists |
| 500 | Server Error | Internal server error |

---

## Authentication

### Bearer Token Format

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

### Token Storage
~~~

```markdown
- **Access Token**: Short-lived (7 days), for API requests
- **Refresh Token**: Long-lived (30 days), for getting new access tokens

### Getting New Access Token

If access token expires:
1. Use refresh token to call `/auth/refresh-token`
2. Get new access token
3. Retry original request

---

## Rate Limiting

Currently, no rate limiting is implemented. Production systems should implement:
- 100 requests per minute per IP
- 1000 requests per hour per user

---

## CORS Configuration

Allowed Origins:
- `http://localhost:3000` (Development)
- `https://e-commerce-blond-three-76.vercel.app` (Production)

---

## Best Practices

1. **Always include Authorization header** for protected routes
2. **Use HTTPS in production** instead of HTTP
3. **Store tokens securely** in httpOnly cookies or secure storage
4. **Implement retry logic** for token refresh
5. **Validate input** on client-side before sending to API
6. **Handle errors gracefully** in frontend applications
7. **Use correct HTTP methods** (GET for retrieval, POST for creation, PUT for updates)
8. **Check response status codes** before processing data

---
```

~~~
## Testing with cURL

### Example: Get All Products
```bash
curl -X GET
http://localhost:8000/api/v1/products/getall?page=1&limit=10
```

### Example: Login
```bash
curl -X POST http://localhost:8000/api/v1/auth/login \
  -H "Content-Type: application/json" \
  -d '{
    "email": "user@example.com",
    "password": "password123"
  }'
```

### Example: Add to Cart (with token)
```bash
curl -X POST http://localhost:8000/api/v1/cart/add \
  -H "Authorization: Bearer YOUR_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "productId": "507f1f77bcf86cd799439013",
    "quantity": 2
  }'
```

---

## Version History

| Version | Date | Changes |
|---------|------|---------|
| 1.0.0 | 2024-11-17 | Initial documentation with all routes |

---

**Document Created:** 2024-11-17
**Last Updated:** 2024-11-17
**Status:** Complete and Verified ✓
~~~

Endpoint: `POST /auth/register`
Authentication: Not Required
Description: Register a new user account

Request Body:

```
{
  "username": "john_doe",
  "email": "john@example.com",
  "fullName": "John Doe",
  "password": "securePassword123"
}
```

Request Validation:

- `username` (string): 3-20 characters, alphanumeric and underscore only
- `email` (string): Valid email format
- `fullName` (string): Required, non-empty
- `password` (string): Minimum 6 characters

Success Response (201 Created):

```
{
  "statusCode": 201,
  "data": {
    "user": {
      "_id": "507f1f77bcf86cd799439011",
      "username": "john_doe",
      "email": "john@example.com",
      "fullName": "John Doe",
      "createdAt": "2024-11-17T10:30:00Z"
    },
    "accessToken": "eyJhbGciOiJIUzI1NiIs...",
    "refreshToken": "eyJhbGciOiJIUzI1NiIs..."
  },
  "message": "User registered successfully"
}
```

Error Response (400):

```
{
  "statusCode": 400,
  "message": "Username must be 3-20 characters, alphanumeric and underscore only",
  "errors": []
}
```

Error Response (409 Conflict):

```
{
  "statusCode": 409,
  "message": "User with this username or email already exists",
  "errors": []
}
```
-----2. User Login

Endpoint: `POST /auth/login`
Authentication: Not Required
Description: Authenticate user and get access tokens

Request Body:
```
{
  "username": "john_doe",
  "email": "john@example.com",
  "password": "securePassword123"
}
```
Request Validation:

- Either `username` OR `email` is required (at least one)
- `password` (string): Required

Success Response (200 OK):
```
{
  "statusCode": 200,
  "data": {
    "user": {
      "_id": "507f1f77bcf86cd799439011",
      "username": "john_doe",
      "email": "john@example.com",
      "fullName": "John Doe"
    },
    "accessToken": "eyJhbGciOiJIUzI1NiIs...",
    "refreshToken": "eyJhbGciOiJIUzI1NiIs..."
  },
  "message": "User logged in successfully"
}
```
Error Response (401):
```
{
  "statusCode": 401,
  "message": "Invalid username/email or password",
  "errors": []
}
```
-----3. Refresh Access Token

Endpoint: `POST /auth/refresh-token`
Authentication: Not Required
Description: Get a new access token using refresh token

Request Body:

```
{
  "refreshToken": "eyJhbGciOiJIUzI1NiIs..."
}
```

Success Response (200 OK):

```
{
  "statusCode": 200,
  "data": {
    "accessToken": "eyJhbGciOiJIUzI1NiIs..."
  },
  "message": "Access token refreshed successfully"
}
```

Error Response (401):

```
{
  "statusCode": 401,
  "message": "Invalid or expired refresh token",
  "errors": []
}
```

-----4. Admin Registration

Endpoint: `POST /auth/register-admin`
Authentication: Not Required
Description: Register a new admin account (requires admin secret key)

Request Body:

```
{
  "username": "admin_user",
  "email": "admin@example.com",
  "fullName": "Admin User",
  "password": "adminPassword123",
  "adminSecret": "MySecretAdminKey2024!"
}
```

Request Validation:

- `adminSecret` must match `ADMIN_SECRET_KEY` in server `.env`
- All other validations same as user registration

Success Response (201 Created):

```
{
  "statusCode": 201,
  "data": {
    "user": {
      "_id": "507f1f77bcf86cd799439012",
      "username": "admin_user",
      "email": "admin@example.com",
      "fullName": "Admin User",
      "role": "admin",
      "createdAt": "2024-11-17T10:30:00Z"
    },
```

    "accessToken": "eyJhbGciOiJIUzI1NiIs...",
    "refreshToken": "eyJhbGciOiJIUzI1NiIs..."
  },
  "message": "Admin registered successfully"
}
Error Response (401):
```
{
  "statusCode": 401,
  "message": "Invalid admin secret key",
  "errors": []
}
```
-----5. User Logout

Endpoint: `POST /auth/logout`
Authentication: Required (Bearer Token)
Description: Logout user and invalidate refresh token

Headers:
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
Request Body: Empty

Success Response (200 OK):
```
{
  "statusCode": 200,
  "data": {},
  "message": "User logged out successfully"
}
```
Error Response (401):
```
{
  "statusCode": 401,
  "message": "User not authenticated",
  "errors": []
}
```
-----Product RoutesBase Path: `/products`1. Get All Products

Endpoint: `GET /products/getall`
Authentication: Not Required
Description: Retrieve all products with optional filtering and pagination

Query Parameters:
?page=1&limit=10&category=Electronics&minPrice=100&maxPrice=1000
Query Details:

- `page` (number): Page number (default: 1)
- `limit` (number): Items per page (default: 10)
- `category` (string): Filter by category (optional)
- `minPrice` (number): Minimum price filter (optional)

- `maxPrice` (number): Maximum price filter (optional)

Success Response (200 OK):
```
{
 "statusCode": 200,
 "data": {
  "products": [
   {
    "_id": "507f1f77bcf86cd799439013",
    "productName": "iPhone 15",
    "price": 999.99,
    "description": "Latest Apple smartphone",
    "category": "Electronics",
    "rating": 4.5,
    "createdAt": " 2024-11-17 T10:30:00Z"
   }
  ],
  "pagination": {
   "currentPage": 1,
   "totalPages": 5,
   "totalProducts": 45,
   "limit": 10
  }
 },
 "message": "Products retrieved successfully"
}
```
-----2. Search Products

Endpoint: `GET /products/search`
Authentication: Not Required
Description: Search products by name, description, or category

Query Parameters:
?query=iPhone&page=1&limit=10
Query Details:

- `query` (string): Search term (required)
- `page` (number): Page number (default: 1)
- `limit` (number): Items per page (default: 10)

Success Response (200 OK):
```
{
 "statusCode": 200,
 "data": {
  "products": [
   {
    "_id": "507f1f77bcf86cd799439013",
        "productName": "iPhone 15",
        "price": 999.99,
        "description": "Latest Apple smartphone",
        "category": "Electronics",
        "rating": 4.5
      }
    ],
    "pagination": {
      "currentPage": 1,
      "totalPages": 1,
      "totalProducts": 1,
      "limit": 10
    }
  },
  "message": "Search results retrieved successfully"
}
```

-----3. Get Product by ID

Endpoint: `GET /products/:id`
Authentication: Not Required
Description: Retrieve a single product by ID

URL Parameters:

- `id` (string): Product MongoDB ID

Example Request:
GET /products/507f1f77bcf86cd799439013
Success Response (200 OK):

```
{
  "statusCode": 200,
  "data": {
    "_id": "507f1f77bcf86cd799439013",
    "productName": "iPhone 15",
    "price": 999.99,
    "description": "Latest Apple smartphone with 5G",
    "category": "Electronics",
    "rating": 4.5,
    "createdAt": "2024-11-17T10:30:00Z"
  },
  "message": "Product retrieved successfully"
}
```

Error Response (404):

```
{
  "statusCode": 404,
  "message": "Product not found",
  "errors": []
}
```

-----4. Add Product (Admin Only)

Endpoint: `POST /products/add`
Authentication: Required (Bearer Token)
Authorization: Admin role required
Description: Create a new product

Headers:
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
Request Body:
```
{
  "productName": "iPhone 15",
  "price": 999.99,
  "description": "Latest Apple smartphone with 5G connectivity",
  "category": "Electronics",
  "rating": 0
}
```
Request Validation:

- `productName` (string): Required, non-empty
- `price` (number): Required, must be positive
- `description` (string): Required, non-empty
- `category` (string): Required, non-empty
- `rating` (number): Optional, must be between 0-5

Success Response (201 Created):
```
{
  "statusCode": 201,
  "data": {
    "_id": "507f1f77bcf86cd799439013",
    "productName": "iPhone 15",
    "price": 999.99,
    "description": "Latest Apple smartphone with 5G connectivity",
    "category": "Electronics",
    "rating": 0,
    "createdAt": " 2024-11-17 T10:30:00Z"
  },
  "message": "Product added successfully"
}
```
Error Response (400):
```
{
  "statusCode": 400,
  "message": "Please provide all required fields (productName, price, description, category)",
  "errors": []
}
```
-----5. Update Product (Admin Only)

Endpoint: `PUT /products/:id`
Authentication: Required (Bearer Token)
Authorization: Admin role required
Description: Update product details

Headers:
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
URL Parameters:

- `id` (string): Product MongoDB ID

Request Body: (All fields optional)
```
{
  "productName": "iPhone 15 Pro",
  "price": 1099.99,
  "description": "Updated description",
  "category": "Premium Electronics",
  "rating": 4.5
}
```
Success Response (200 OK):
```
{
  "statusCode": 200,
  "data": {
    "_id": "507f1f77bcf86cd799439013",
    "productName": "iPhone 15 Pro",
    "price": 1099.99,
    "description": "Updated description",
    "category": "Premium Electronics",
    "rating": 4.5
  },
  "message": "Product updated successfully"
}
```
-----6. Delete Product (Admin Only)

Endpoint: `DELETE /products/:id`
Authentication: Required (Bearer Token)
Authorization: Admin role required
Description: Delete a product

Headers:
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
URL Parameters:

- `id` (string): Product MongoDB ID

Success Response (200 OK):
```
{
  "statusCode": 200,
  "data": {
    "_id": "507f1f77bcf86cd799439013",
    "productName": "iPhone 15",
    "price": 999.99
  },
  "message": "Product deleted successfully"
}
```

-----Cart RoutesBase Path: `/cart`

Note: All cart routes require authentication1. Get User's Cart

Endpoint: `GET /cart`
Authentication: Required (Bearer Token)
Description: Retrieve the current user's cart

Headers:
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
Success Response (200 OK):

```
{
  "statusCode": 200,
  "data": {
    "_id": "507f1f77bcf86cd799439020",
    "userId": "507f1f77bcf86cd799439011",
    "items": [
      {
        "productId": "507f1f77bcf86cd799439013",
        "productName": "iPhone 15",
        "price": 999.99,
        "quantity": 1,
        "totalPrice": 999.99
      }
    ],
    "totalAmount": 999.99,
    "totalItems": 1
  },
  "message": "Cart retrieved successfully"
}
```

-----2. Get Cart Count

Endpoint: `GET /cart/count`
Authentication: Required (Bearer Token)
Description: Get the number of items in cart

Headers:
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
Success Response (200 OK):

```
{
  "statusCode": 200,
  "data": {
    "cartCount": 3
  },
  "message": "Cart count retrieved successfully"
}
```
-----3. Get Cart Summary

Endpoint: `GET /cart/summary`
Authentication: Required (Bearer Token)
Description: Get cart total amount and item count

Headers:
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
Success Response (200 OK):
```
{
  "statusCode": 200,
  "data": {
    "totalItems": 3,
    "totalAmount": 2999.97
  },
  "message": "Cart summary retrieved successfully"
}
```
-----4. Add to Cart

Endpoint: `POST /cart/add`
Authentication: Required (Bearer Token)
Description: Add product to cart or increase quantity

Headers:
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
Request Body:
```
{
  "productId": "507f1f77bcf86cd799439013",
  "quantity": 2
}
```
Request Validation:

- `productId` (string): Valid MongoDB ID required
- `quantity` (number): Must be positive integer

Success Response (200 OK):
```
{
  "statusCode": 200,
  "data": {
    "_id": "507f1f77bcf86cd799439020",
    "userId": "507f1f77bcf86cd799439011",
    "items": [
      {
        "productId": "507f1f77bcf86cd799439013",
        "productName": "iPhone 15",
        "price": 999.99,
        "quantity": 2,
        "totalPrice": 1999.98
      }
    ],
    "totalAmount": 1999.98,
    "totalItems": 2
  },
  "message": "Product added to cart successfully"
}
```

Error Response (404):

```
{
  "statusCode": 404,
  "message": "Product not found",
  "errors": []
}
```

-----5. Update Cart Item Quantity

Endpoint: `PUT /cart/update`
Authentication: Required (Bearer Token)
Description: Update quantity of item in cart

Headers:
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
Request Body:

```
{
  "productId": "507f1f77bcf86cd799439013",
  "quantity": 5
}
```

Request Validation:

- `quantity` = 0: Will remove the item
- `quantity` > 0: Will update the quantity

Success Response (200 OK):

```
{
  "statusCode": 200,
  "data": {
    "_id": "507f1f77bcf86cd799439020",
    "items": [
      {
        "productId": "507f1f77bcf86cd799439013",
        "productName": "iPhone 15",
        "price": 999.99,
        "quantity": 5,
        "totalPrice": 4999.95
      }
    ],
    "totalAmount": 4999.95,
    "totalItems": 5
  },
  "message": "Cart item quantity updated successfully"
}
```
-----6. Remove from Cart

Endpoint: `POST /cart/remove`
Authentication: Required (Bearer Token)
Description: Remove product from cart

Headers:
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
Request Body:
```
{
  "productId": "507f1f77bcf86cd799439013"
}
```
Success Response (200 OK):
```
{
  "statusCode": 200,
  "data": {
    "_id": "507f1f77bcf86cd799439020",
    "items": [],
    "totalAmount": 0,
    "totalItems": 0
  },
  "message": "Product removed from cart successfully"
}
```
-----7. Clear Cart

Endpoint: `POST /cart/clear`
Authentication: Required (Bearer Token)
Description: Remove all items from cart

Headers:
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
Request Body: Empty

Success Response (200 OK):
```
{
  "statusCode": 200,
  "data": {
    "_id": "507f1f77bcf86cd799439020",
    "items": [],
    "totalAmount": 0,
    "totalItems": 0
  },
  "message": "Cart cleared successfully"
}
```
-----Payment RoutesBase Path: `/payment`

Note: All payment routes require authentication1. Create Payment Intent

Endpoint: `POST /payment/create-intent`
Authentication: Required (Bearer Token)
Description: Create a Stripe payment intent

Headers:
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
Request Body:
```
{
  "amount": 1999.98,
  "currency": "usd",
  "orderId": "order_12345",
  "userId": "507f1f77bcf86cd799439011",
  "email": "user@example.com"
}
```
Request Validation:

- `amount` (number): Must be greater than 0
- `currency` (string): Required, typically "usd"
- `orderId` (string): Unique order identifier
- `email` (string): Required for receipt

Success Response (200 OK):
```
{
  "statusCode": 200,
  "data": {
    "clientSecret": "pi_1234567890_secret_abcdefg",
    "paymentIntentId": "pi_1234567890"
  },
  "message": "Payment intent created successfully"
}
```
Error Response (400):
```
{
  "statusCode": 400,
  "message": "Amount and currency are required",
  "errors": []
}
```
-----2. Confirm Payment

Endpoint: `POST /payment/confirm`
Authentication: Required (Bearer Token)
Description: Confirm payment after successful Stripe processing

Headers:
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
Request Body:
```
{
  "paymentIntentId": "pi_1234567890",
  "orderId": "order_12345"
}
```
Request Validation:

- `paymentIntentId` (string): Required
- `orderId` (string): Optional, for order tracking

Success Response (200 OK):
```
{
  "statusCode": 200,
  "data": {
    "status": "succeeded"
  },
  "message": "Payment confirmed successfully"
}
```
Processing Response (202):
```
{
  "statusCode": 202,
  "data": {
    "status": "processing"
  },
  "message": "Payment is processing"
}
```
Error Response (402):
```
{
  "statusCode": 402,
  "message": "Payment failed",
  "errors": []
}
```
-----3. Get Payment Status

Endpoint: `GET /payment/status/:paymentIntentId`
Authentication: Required (Bearer Token)
Description: Check payment status by payment intent ID

Headers:
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...

URL Parameters:

- `paymentIntentId` (string): Stripe payment intent ID

Example Request:
GET /payment/status/pi_1234567890
Success Response (200 OK):
```
{
  "statusCode": 200,
  "data": {
    "status": "succeeded",
    "amount": 199998,
    "currency": "usd"
  },
  "message": "Payment status retrieved successfully"
}
```
-----Admin RoutesBase Path: `/admin`

Note: All admin routes require authentication AND admin role1. Get Dashboard Statistics

Endpoint: `GET /admin/dashboard/stats`
Authentication: Required (Bearer Token)
Authorization: Admin role required
Description: Get overall store statistics

Headers:
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
Success Response (200 OK):
```
{
  "statusCode": 200,
  "data": {
    "totalUsers": 45,
    "totalProducts": 120,
    "totalOrders": 340,
    "totalRevenue": 45678.90,
    "totalProductCategories": 8,
    "categories": [
      "Electronics",
      "Fashion",
      "Home",
      "Books",
      "Sports",
      "Toys",
      "Beauty",
      "Food"
    ]
  },
  "message": "Dashboard statistics retrieved successfully"
```

}
-----2. Get All Users

Endpoint: `GET /admin/users`
Authentication: Required (Bearer Token)
Authorization: Admin role required
Description: List all users with pagination

Headers:
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
Query Parameters:
?page=1&limit=10&role=user
Success Response (200 OK):
```
{
  "statusCode": 200,
  "data": {
    "users": [
      {
        "_id": "507f1f77bcf86cd799439011",
        "username": "john_doe",
        "email": "john@example.com",
        "fullName": "John Doe",
        "createdAt": " 2024-11-17 T10:30:00Z"
      }
    ],
    "pagination": {
      "currentPage": 1,
      "totalPages": 5,
      "totalUsers": 45,
      "limit": 10
    }
  },
  "message": "Users retrieved successfully"
}
```
-----3. Get Inventory Summary

Endpoint: `GET /admin/inventory/summary`
Authentication: Required (Bearer Token)
Authorization: Admin role required
Description: Get inventory value and category breakdown

Headers:
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
Success Response (200 OK):
```
{
  "statusCode": 200,
  "data": {
    "totalProducts": 120,
    "totalInventoryValue": 456789.50,
    "categorySummary": [
      {
        "_id": "Electronics",
        "count": 35,
        "avgPrice": 899.99
      },
      {
        "_id": "Fashion",
        "count": 45,
        "avgPrice": 49.99
      }
    ],
    "allProducts": [...]
  },
  "message": "Inventory summary retrieved successfully"
}
```
-----4. Get Category Inventory

Endpoint: `GET /admin/inventory/category`
Authentication: Required (Bearer Token)
Authorization: Admin role required
Description: Get detailed inventory breakdown by category

Headers:
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
Success Response (200 OK):
```
{
  "statusCode": 200,
  "data": [
    {
      "_id": "Electronics",
      "products": [...],
      "totalProducts": 35,
      "totalValue": 31499.65,
      "avgPrice": 899.99,
      "minPrice": 199.99,
      "maxPrice": 1999.99
    }
  ],
  "message": "Category inventory retrieved successfully"
}
```
-----5. Get Low Rating Products

Endpoint: `GET /admin/inventory/low-rating`
Authentication: Required (Bearer Token)

Authorization: Admin role required
Description: Get products with low ratings (quality control)

Headers:
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
Query Parameters:
?minRating=0&maxRating=3
Success Response (200 OK):
```
{
  "statusCode": 200,
  "data": [
    {
      "_id": "507f1f77bcf86cd799439014",
      "productName": "Low Rating Product",
      "rating": 2.5,
      "price": 99.99
    }
  ],
  "message": "Low rating products retrieved successfully"
}
```
-----6. Get Products by Price Range

Endpoint: `GET /admin/inventory/price-range`
Authentication: Required (Bearer Token)
Authorization: Admin role required
Description: Get products within price range

Headers:
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
Query Parameters:
?minPrice=100&maxPrice=1000
Success Response (200 OK):
```
{
  "statusCode": 200,
  "data": {
    "minPrice": 100,
    "maxPrice": 1000,
    "productsInRange": 45,
    "products": [
      {
        "_id": "507f1f77bcf86cd799439013",
        "productName": "iPhone 15",
        "price": 999.99,
        "category": "Electronics"
      }
    ]
  },
```

  "message": "Products by price range retrieved successfully"
}
-----7. Get All Products (Admin)

Endpoint: `GET /admin/products`
Authentication: Required (Bearer Token)
Authorization: Admin role required
Description: List all products for admin

Headers:
Authorization: Bearer eyJhbGciOiJIUzI1NiIs...
Success Response (200 OK):

```
{
  "statusCode": 200,
  "data": {
    "products": [...]
  },
  "message": "Products retrieved successfully"
}
```

-----8. Admin Product Management

Add Product (Admin):
Endpoint: `POST /admin/products`

Update Product (Admin):
Endpoint: `PUT /admin/products/:id`

Delete Product (Admin):
Endpoint: `DELETE /admin/products/:id`

Same request/response format as Product routes above-----Error HandlingError Response Format

All error responses follow this standard format:

```
{
  "statusCode": <HTTP_STATUS_CODE>,
  "message": "<Error Description>",
  "errors": []
}
```

Common Error ScenariosUnauthorized (401)

```
{
  "statusCode": 401,
  "message": "User not authenticated",
  "errors": []
}
```

Forbidden (403)

```
{
  "statusCode": 403,
  "message": "Access denied - admin role required",
  "errors": []
}
```
Bad Request (400)
```
{
  "statusCode": 400,
  "message": "Invalid input format",
  "errors": []
}
```
Not Found (404)
```
{
  "statusCode": 404,
  "message": "Resource not found",
  "errors": []
}
```
Conflict (409)
```
{
  "statusCode": 409,
  "message": "Resource already exists",
  "errors": []
}
```
Server Error (500)
```
{
  "statusCode": 500,
  "message": "Internal server error",
  "errors": []
}
```
-----Status Codes

Code	Status	Usage
200	OK	Successful GET, PUT, POST
201	Created	Successful resource creation
202	Accepted	Request accepted for processing
400	Bad Request	Invalid input or missing required fields
401	Unauthorized	Missing or invalid authentication token
402	Payment Required	Payment failed

403	Forbidden	Insufficient permissions (e.g., admin role)
404	Not Found	Resource does not exist
409	Conflict	Resource already exists
500	Server Error	Internal server error

-----AuthenticationBearer Token Format
Authorization: Bearer <JWT_TOKEN>
Token Storage

- **Access Token**: Short-lived (7 days), for API requests
- **Refresh Token**: Long-lived (30 days), for getting new access tokens

Getting New Access Token

If access token expires:

1. Use refresh token to call `/auth/refresh-token`
2. Get new access token
3. Retry original request

-----Rate Limiting

Currently, no rate limiting is implemented. Production systems should implement:

- 100 requests per minute per IP
- 1000 requests per hour per user

-----CORS Configuration

Allowed Origins:

- `http://localhost:3000` (Development)
- `https://e-commerce-blond-three-76.vercel.app` (Production)

-----Best Practices

1. **Always include Authorization header** for protected routes
2. **Use HTTPS in production** instead of HTTP
3. **Store tokens securely** in httpOnly cookies or secure storage
4. **Implement retry logic** for token refresh
5. **Validate input** on client-side before sending to API
6. **Handle errors gracefully** in frontend applications
7. **Use correct HTTP methods** (GET for retrieval, POST for creation, PUT for updates)
8. **Check response status codes** before processing data

-----Testing with cURLExample: Get All Products

```
curl -X GET http://localhost:8000/api/v1/products/getall?page=1&limit=10
```
Example: Login
```
curl -X POST http://localhost:8000/api/v1/auth/login \
  -H "Content-Type: application/json" \
  -d '{    "email": "user@example.com",    "password": "password123"  }'
```
Example: Add to Cart (with token)
```
curl -X POST http://localhost:8000/api/v1/cart/add \
  -H "Authorization: Bearer YOUR_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{    "productId": "507f1f77bcf86cd799439013",    "quantity": 2  }'
```
-----Version History

Version	Date	Changes
1.0.0	2024-11-17	Initial documentation with all routes

-----**Document Created:** 2024-11-17
Last Updated: 2024-11-17
Status: Complete and Verified ✓

Project overview

Tech stack

- **Frontend:** Next.js
- **Backend:** Express.js, Node.js
- **Database:** MongoDB
- **Payment Gateway:** Stripe (in testing mode)

Backend routes