Authentication Module

Handles user registration, login, logout, email verification, password change, and token management.

Base Route

`/api/v1/auth`

Endpoints

1. Register User

URL: `POST /register`
Description: Registers a new user and sends an email verification link.

Request Body:

```json
{
  "fullName": "John Doe",
  "email": "john@example.com",
  "password": "password123",
  "mobile": "1234567890"
}
```

Validations:

- Full name, email, password, and mobile are required.
- Email must be valid.
- Password must be at least 6 characters.
- Mobile must be a valid phone number.

Response:
`201 Created`

```json
{
  "success": true,
  "message": "User registered successfully. Check your email for verification."
}
```

---

2. Verify Email

URL: `POST /verify-email`
Description: Verifies a user's email using the token from the verification link.

Request Body:

```
{
  "token": "jwt_verification_token"
}
```

Response:
`200 OK`

```
{
  "success": true,
  "message": "Email verified successfully! You can now log in."
}
```

---

3. Login

URL: `POST /login`
Description: Logs in a verified user and sets access and refresh tokens.

Request Body:

```
{
  "email": "john@example.com",
  "password": "password123"
}
```

Response:
`200 OK`

```
{
  "success": true,
  "message": "Login successful",
  "accessToken": "jwt_access_token"
}
```

Notes:

- Cookies are set: `accessToken`, `refreshToken`
- Inactive users are sent a new verification email.

---

4. Refresh Access Token

URL: `GET /refresh-token`
Description: Generates a new access token using the refresh token.

Headers or Cookies:

- `refreshToken` must be present in cookies or header.

Response:
`200 OK`

```json
{
  "success": true,
  "accessToken": "new_jwt_access_token"
}
```

---

5. Change Password

URL: `POST /change-password`
Description: Authenticated users can change their password.

Authentication: Required (`isAuthenticated` middleware)

Request Body:

```json
{
  "oldPassword": "oldpassword123",
  "newPassword": "newpassword456"
}
```

Validations:

- Both fields are required.
- New password must be at least 6 characters.

Response:
`200 OK`

```json
{
  "success": true,
  "message": "Password changed successfully"
}
```

---

6. Logout

URL: `GET /logout`
Description: Logs the user out by clearing the cookies.

Authentication: Required

Response:
`200 OK`

```json
{
  "success": true,
  "message": "Logged out successfully"
}
```

Middleware Used

- `validateRegister`, `validateLogin`, `validateChangePassword` - For request validations
- `isAuthenticated` - For routes that require login
- `validateResults` - Aggregates and returns validation errors

Address Module

Handles creation, retrieval, updating, and deletion of user addresses.

Base Route

`/api/v1/address`

Endpoints

1. Create Address

URL: `POST /`
Description: Creates a new address for the authenticated user.

Authentication: Required (`isAuthenticated` middleware)
Validations: `validateAddress`

Request Body:

```json
{
  "addressLine1": "123 Main St",
  "city": "New York",
  "state": "NY",
  "pinCode": "10001",
  "country": "USA",
  "mobile": "1234567890"
}
```

Validations:

- All fields are required.
- `pinCode` must be numeric and 5–10 characters long.
- `mobile` must be a valid phone number.

Response:
`201 Created`

```json
{
  "success": true,
  "message": "Address added successfully"
}
```

---

2. Get All Addresses

URL: `GET /`
Description: Retrieves all addresses of the logged-in user.

Authentication: Required

Response:
`200 OK`

```
{
  "success": true,
  "addresses": [
    {
      "_id": "address_id",
      "addressLine1": "123 Main St",
      "city": "New York",
      "state": "NY",
      "pinCode": "10001",
      "country": "USA",
      "mobile": "1234567890"
    }
  ]
}
```

---

3. Get Single Address

URL: `GET /:id`
Description: Retrieves a single address by its ID.

Authentication: Required
Params:

- `id` – Address ID (MongoDB ObjectId)

Response:
`200 OK`

```
{
  "success": true,
  "address": {
    "_id": "address_id",
    "addressLine1": "123 Main St",
    "city": "New York",
    "state": "NY",
    "pinCode": "10001",
    "country": "USA",
    "mobile": "1234567890"
  }
}
```

Error (if not found):
`404 Not Found`

```
{
  "success": false,
  "message": "Address not found"
}
```

---

4. Update Address

URL: `PUT /:id`
Description: Updates an existing address by ID.

Authentication: Required
Validations: `validateAddress`
Params:

- `id` – Address ID (MongoDB ObjectId)

Request Body: Same as Create Address

Response:
`200 OK`

```
{
  "success": true,
  "message": "Address updated successfully"
}
```

Error (if not found):
`404 Not Found`

```
{
  "success": false,
  "message": "Address not found"
}
```

---

5. Delete Address

URL: `DELETE /:id`
Description: Deletes an address by its ID.

Authentication: Required
Params:

- `id` – Address ID (MongoDB ObjectId)

Response:
`200 OK`

```json
{
  "success": true,
  "message": "Address deleted successfully"
}
```

Error (if not found):
`404 Not Found`

```json
{
  "success": false,
  "message": "Address not found"
}
```

Middleware Used

- `isAuthenticated` - Ensures user is logged in before accessing routes
- `validateAddress` - Validates the address fields for POST/PUT
- `validateResults` - Handles and formats validation errors

Here you go — your **Category Module documentation** in the same format:

Category Module

Handles category creation, retrieval, updating, and deletion. Only admins are allowed to create, update, or delete categories.

Base Route

`/api/v1/category`

Endpoints

1. Get All Categories

URL: `GET /`
Description: Retrieves all product categories.

Authentication: Required

Response:
`200 OK`

```
{
  "success": true,
  "categories": [
    {
      "_id": "category_id",
      "category": "Electronics",
      "slug": "electronics"
    }
  ]
}
```

---

2. Create Category

URL: `POST /`
Description: Creates a new category. Slug is generated from the category name.

Authentication: Required
Authorization: `ADMIN` role required

Request Body:

```
{
  "category": "Electronics"
}
```

Validations:

- Category name is required.

Response:
`201 Created`

```
{
  "success": true,
  "message": "Category created successfully!",
  "category": {
    "_id": "new_category_id",
    "category": "Electronics",
    "slug": "electronics"
  }
}
```

Error (if already exists):
`400 Bad Request`

```
{
  "success": false,
  "message": "Category already exists."
}
```

---

3. Update Category

URL: `PUT /:slug`
Description: Updates the category name and regenerates the slug.

Authentication: Required
Authorization: `ADMIN` role required
Params:

- `slug` – Slug of the category to update

Request Body:

```
{
  "category": "Home Appliances"
}
```

Validations:

- Category name is required.

Response:
`200 OK`

```
{
  "success": true,
  "message": "Category updated successfully",
  "category": {
    "_id": "category_id",
    "category": "Home Appliances",
    "slug": "home-appliances"
  }
}
```

Error (if not found):
`404 Not Found`

```json
{
  "success": false,
  "message": "Category not found"
}
```

---

4. Delete Category

URL: `DELETE /:slug`
Description: Deletes a category by slug and removes it from associated products.

Authentication: Required
Authorization: `ADMIN` role required
Params:

- `slug` – Slug of the category to delete

Response:
`200 OK`

```json
{
  "success": true,
  "message": "Category deleted successfully"
}
```

Error (if not found):
`404 Not Found`

```json
{
  "success": false,
  "message": "Category not found"
}
```

Middleware Used

- `isAuthenticated` - Ensures user is logged in
- `authorizeRole("ADMIN")` - Restricts creation, update, and deletion to admin users
- `validateResults` - Aggregates and formats validation errors if validations are added later

Product Module

Handles product management including creation, update, deletion, image upload, stock management, and category linking.

Base Route

`/api/v1/product`

Endpoints

1. Get All Products

URL: `GET /`

Query Parameters:

- `page` (optional): Page number (default: 1)
- `limit` (optional): Items per page (default: 10)

Response:
`200 OK`

```json
{
  "success": true,
  "products": [...],
  "currentPage": 1,
  "totalPages": 5,
  "totalProducts": 50
}
```

---

2. Get Products by Category

URL: `GET /category/:id`

Description: Retrieves products linked to a specific category.

Response:
`200 OK`

```json
{
  "success": true,
  "products": [...],
  "currentPage": 1,
  "totalPages": 3,
  "totalProducts": 30
}
```

---

3. Get Product by ID

URL: `GET /:id`

Description: Fetches a single product by ID.

Response:
`200 OK`

```
{
  "success": true,
  "product": { ... }
}
```

---

4. Create Product

URL: `POST /`

Authentication: Required (Admin)

Middleware:

- `isAuthenticated`
- `authorizeRole("ADMIN")`
- `upload.array("images", 10)`
- `validateProduct`

Form Data:

- `name`: string
- `description`: string
- `brand`: string (optional)
- `price`: number
- `countInStock`: number
- `category`: single ID or array of IDs
- `images`: array of files

Response:
`201 Created`

```
{
  "success": true,
  "message": "Product created successfully",
  "product": { ... }
}
```

---

5. Update Product

URL: `PUT /:id`

Authentication: Required (Admin)

Middleware:

- `isAuthenticated`
- `authorizeRole("ADMIN")`
- `upload.array("images", 10)`
- `validateProduct`

Description: Updates product fields and supports adding new images.

Response:
`200 OK`

```
{
  "success": true,
  "message": "Product updated successfully",
  "product": { ... }
}
```

---

6. Update Product Stock

URL: `PUT /stock/:id`

Authentication: Required (Admin)

Request Body:

```
{
  "countInStock": 15
}
```

Response:
`200 OK`

```
{
  "success": true,
  "message": "Product stock updated successfully",
  "product": { ... }
}
```

---

7. Delete Product

URL: `DELETE /:id`

Authentication: Required (Admin)

Description: Deletes a product and its associated reviews.

Response:
`200 OK`

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

You're right! Here's the updated and complete **Cart module documentation** with actual sample **responses** for each endpoint:

Cart Module

Cart Routes

Base Route: `/api/v1/cart`

Add to Cart

- **Endpoint**: `POST /`
- **Access**: Private (requires authentication)
- **Description**: Adds a product to the user's cart. If it already exists, updates the quantity.
- **Request Body**:

  ```json
  {
    "productId": "6622c24452f5a374799fb403",
    "quantity": 2
  }
  ```

- **Responses**:

  201 Created

  ```json
  {
    "success": true,
    "message": "Product added to cart",
    "cartItem": {
      "_id": "6622c4561aabb340a14f4e91",
      "userId": "6622c123abc456def789abcd",
      "productId": "6622c24452f5a374799fb403",
      "quantity": 2,
      "createdAt": "2025-04-07T10:12:38.123Z",
      "updatedAt": "2025-04-07T10:12:38.123Z",
      "__v": 0
  ```

```
  }
}
```

200 OK (if product already exists in cart)

```
{
  "success": true,
  "message": "Cart updated successfully",
  "cartItem": {
    "_id": "6622c4561aabb340a14f4e91",
    "userId": "6622c123abc456def789abcd",
    "productId": "6622c24452f5a374799fb403",
    "quantity": 4,
    "createdAt": "2025-04-07T10:12:38.123Z",
    "updatedAt": "2025-04-07T10:14:20.456Z",
    "__v": 0
  }
}
```

400 Bad Request (e.g., insufficient stock)

```
{
  "success": false,
  "message": "Insufficient stock"
}
```

404 Not Found (e.g., product does not exist)

```
{
  "success": false,
  "message": "Product not found."
}
```

Get Cart

- **Endpoint**: `GET /`
- **Access**: Private (requires authentication)
- **Description**: Retrieves all items in the user's cart.
- **Response**:

```
{
  "success": true,
  "cartItems": [
    {
      "_id": "6622c4561aabb340a14f4e91",
      "userId": "6622c123abc456def789abcd",
      "productId": {
        "_id": "6622c24452f5a374799fb403",
        "name": "Bluetooth Headphones",
        "price": 59.99,
        "images": [
          {
            "url": "https://res.cloudinary.com/sample/image.jpg",
            "public_id": "product/headphones1"
          }
        ],
        "countInStock": 25
      },
      "quantity": 2
    }
  ],
  "totalAmount": 119.98
}
```

Remove Item from Cart

- **Endpoint**: `PUT /:id`
- **Access**: Private (requires authentication)
- **Description**: Removes a specific item from the user's cart.
- **Route Params**:
  - `id`: Cart item ID
- **Responses**:

  200 OK

  ```
  {
    "success": true,
    "message": "Item removed from cart"
  }
  ```

  404 Not Found

  ```
  {
    "success": false,
    "message": "Cart item not found."
  }
  ```

Clear Cart

- **Endpoint**: `DELETE /`
- **Access**: Private (requires authentication)
- **Description**: Removes all items from the user's cart.
- **Response**:

  ```
  {
    "success": true,
    "message": "Cart cleared successfully"
  }
  ```

  Here's the complete **Order module documentation** in the same format as the **Cart module**, including base route, endpoints, request bodies, and responses:

Order Module

Order Routes

Base Route: `/api/v1/order`

Create Payment Intent

- **Endpoint**: `POST /payment-intent`
- **Access**: Private (requires authentication)
- **Description**: Creates a Stripe payment intent for the user based on their cart.
- **Request Body**:

  ```
  {
    "addressId": "6623c812e8045105dc218929",
    "paymentMethodId": "pm_card_visa"
  }
  ```

- **Responses**:

  200 OK

  ```
  {
    "paymentIntentId": "pi_3NyAeUSGd5p1xKjn1FVRPquF",
    "clientSecret": "pi_3NyAeUSGd5p1xKjn1FVRPquF_secret_QbBOH...",
  ```

```
  "totalAmount": 159.99
}
```

400 Bad Request

```
{
  "success": false,
  "message": "Cart is empty."
}
```

404 Not Found

```
{
  "success": false,
  "message": "Address not found."
}
```

Create Order

- **Endpoint**: `POST /`
- **Access**: Private (requires authentication)
- **Description**: Creates an order for the user after successful payment.
- **Request Body**:

  ```
  {
    "paymentIntentId": "pi_3NyAeUSGd5p1xKjn1FVRPquF",
    "addressId": "6623c812e8045105dc218929"
  }
  ```

- **Responses**:

  201 Created

  ```
  {
    "message": "Order placed successfully",
    "order": {
      "_id": "6623dd97d4df9a7a2d9ec50b",
      "orderId": "ORD-8f1c7a24",
      "userId": "6622c123abc456def789abcd",
      "paymentId": "pi_3NyAeUSGd5p1xKjn1FVRPquF",
      "paymentStatus": "Completed",
      "products": [
        {
          "productId": "6622c24452f5a374799fb403",
          "quantity": 2,
          "priceAtPurchase": 59.99
        }
  ```

```
    ],
    "totalAmount": 119.98,
    "deliveryAddress": "6623c812e8045105dc218929",
    "orderStatus": "Processing",
    "createdAt": "2025-04-07T14:12:39.456Z",
    "updatedAt": "2025-04-07T14:12:39.456Z",
    "__v": 0
  }
}
```

400 Bad Request

```
{
  "success": false,
  "message": "Payment not successful."
}
```

400 Duplicate Payment

```
{
  "success": false,
  "message": "This payment has already been processed."
}
```

404 Not Found

```
{
  "success": false,
  "message": "Address not found."
}
```

Get User Orders

- **Endpoint**: `GET /`
- **Access**: Private (requires authentication)
- **Description**: Retrieves the logged-in user's order history.
- **Response**:

```
{
  "orders": [
    {
      "_id": "6623dd97d4df9a7a2d9ec50b",
      "orderId": "ORD-8f1c7a24",
      "paymentStatus": "Completed",
      "orderStatus": "Processing",
      "totalAmount": 119.98,
      "products": [
        {
          "productId": {
            "_id": "6622c24452f5a374799fb403",
            "name": "Bluetooth Headphones",
            "price": 59.99
          },
          "quantity": 2,
          "priceAtPurchase": 59.99
        }
      ]
    }
  ]
}
```

Get Order By ID

- **Endpoint**: `GET /:id`
- **Access**: Private (requires authentication)
- **Description**: Retrieves a specific order by its `orderId`.
- **Route Params**:
  - `id`: Order ID (e.g., `ORD-8f1c7a24`)
- **Responses**:

  200 OK

  ```
  {
    "_id": "6623dd97d4df9a7a2d9ec50b",
    "orderId": "ORD-8f1c7a24",
    "paymentStatus": "Completed",
    "orderStatus": "Processing",
    "totalAmount": 119.98,
    "products": [
      {
        "productId": {
          "_id": "6622c24452f5a374799fb403",
          "name": "Bluetooth Headphones",
          "price": 59.99
        },
        "quantity": 2,
        "priceAtPurchase": 59.99
      }
  ```

```
  ]
}
```

404 Not Found

```
{
  "success": false,
  "message": "Order not found"
}
```

Get All Orders (Admin)

- **Endpoint**: `GET /admin/all`
- **Access**: Admin only
- **Description**: Retrieves all orders in the system.
- **Response**:

```
{
  "orders": [
    {
      "_id": "6623dd97d4df9a7a2d9ec50b",
      "orderId": "ORD-8f1c7a24",
      "userId": {
        "_id": "6622c123abc456def789abcd",
        "fullName": "John Doe",
        "email": "john@example.com"
      },
      "paymentStatus": "Completed",
      "orderStatus": "Processing",
      "totalAmount": 119.98
    }
  ]
}
```

Update Order Status (Admin)

- **Endpoint**: `PUT /admin/:id/status`
- **Access**: Admin only
- **Description**: Advances the order status to the next stage (e.g., from “Processing” to “Shipped”).
- **Route Params**:

  - `id`: Order ID (e.g., `ORD-8f1c7a24`)

- **Responses**:

  200 OK

  ```json
  {
    "message": "Order status updated successfully",
    "order": {
      "orderId": "ORD-8f1c7a24",
      "orderStatus": "Shipped"
    }
  }
  ```

  403 Forbidden

  ```json
  {
    "success": false,
    "message": "Not authorized"
  }
  ```

  404 Not Found

  ```json
  {
    "success": false,
    "message": "Order not found"
  }
  ```

  400 Bad Request

  ```json
  {
    "success": false,
    "message": "Invalid or final order status"
  }
  ```

Here is the **`review.md`** documentation following the exact format used in your `order.md`:

Review Module

Base Route: `/api/v1/review`
Authentication: Required for creating, updating, and deleting reviews.

GET `/:productId`

Get all reviews for a specific product.

Authentication: Not required
Params:

- `productId` (Mongo ID) — required

Response:

```json
{
  "success": true,
  "reviews": [
    {
      "_id": "reviewId",
      "product": "productId",
      "user": {
        "_id": "userId",
        "name": "John Doe"
      },
      "rating": 4,
      "comment": "Great product",
      "createdAt": "2024-04-01T10:00:00Z",
      "updatedAt": "2024-04-01T10:00:00Z",
      "__v": 0
    }
  ]
}
```

POST /

Create a new review for a product.

Authentication: Required
Body:

```json
{
  "product": "productId",
  "rating": 5,
  "comment": "Excellent!"
}
```

Response:

```json
{
  "success": true,
  "review": {
    "_id": "reviewId",
    "product": "productId",
    "user": "userId",
    "rating": 5,
    "comment": "Excellent!",
    "createdAt": "2024-04-01T10:00:00Z",
    "updatedAt": "2024-04-01T10:00:00Z",
    "__v": 0
  }
}
```

PUT `/:id`

Update an existing review. Only the user who created the review can update it.

Authentication: Required
Params:

- `id` (Mongo ID of review) — required

Body:

```
{
  "rating": 4,
  "comment": "Changed my mind, it's good!"
}
```

Response:

```
{
  "success": true,
  "review": {
    "_id": "reviewId",
    "product": "productId",
    "user": "userId",
    "rating": 4,
    "comment": "Changed my mind, it's good!",
    "createdAt": "2024-04-01T10:00:00Z",
    "updatedAt": "2024-04-01T10:10:00Z",
    "__v": 0
  }
}
```

DELETE `/:id`

Delete a review. Only the user who created the review can delete it.

Authentication: Required
Params:

- `id` (Mongo ID of review) — required

Response:

```json
{
  "success": true,
  "message": "Review deleted successfully"
}
```
