API Documentation

Base URL

```
Deployed - https://ecom-backend-zun3.onrender.com
Local - http://locahost:<PORT>
```

Authentication Routes (`/auth`)

1. POST /auth/signup

Description: Registers a new user.

Request Body:

```
{
  "name": "John Doe",
  "email": "john@example.com",
  "password": "securePassword123",
  "role": "user"
}
```

- `name` (string, required): Full name of the user.
- `email` (string, required): Unique email address.
- `password` (string, required): User's password.
- `role` (string, optional): Can be `user`, `admin`, or `seller`. Defaults to `user`.

Response:

Success (201):

```
{
  "message": "User Signup Successful"
}
```

Error (400):

```
{
  "message": "User Already exists with this email"
}
```

2. POST /auth/login

Description: Authenticates a user and returns a JWT.

Request Body:

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

- `email` (string, required): Registered email address.
- `password` (string, required): User's password.

Response:

Success (200):

```
{
  "token": "JWT_TOKEN"
}
```

Error (401):

```
{
  "error": "Invalid Credentials"
}
```

Seller Routes (`/seller`)

1. GET /products

Description: Fetch all products owned by the authenticated seller.

- **Headers:**
  - `Authorization`: Bearer token (JWT).
- **Response:**

Success **(200):**

```
[
  {
    "_id": "12345",
    "seller": "67890",
    "title": "Product A",
    "description": "Description of Product A",
    "price": 100,
    "category": "Electronics",
    "stock": 50,
    "createdAt": "2024-11-01T12:00:00Z"
  }
]
```

Error (404):

```
{
  "message": "No products found"
}
```

Error (500):

```
{
  "error": "Error fetching products"
}
```

2. POST /product

Description: Add a new product under the authenticated seller.

- **Headers:**
  - `Authorization`: Bearer token (JWT).

Request Body:

```
{
  "title": "Product A",
  "description": "Description of Product A",
  "price": 100,
  "category": "Electronics",
  "stock": 50
}
```

- `title` (string, required): Name of the product.
- `description` (string, required): Detailed description.
- `price` (number, required): Price of the product.
- `category` (string, required): Product category.
- `stock` (number, required): Quantity in stock.

Response:

Success (201):

```
{
  "message": "Product added successfully",
  "product": {
    "_id": "12345",
    "seller": "67890",
    "title": "Product A",
    "description": "Description of Product A",
    "price": 100,
    "category": "Electronics",
    "stock": 50,
    "createdAt": "2024-11-01T12:00:00Z"
  }
}
```

Error (400):

```
{
  "message": "All fields are required"
}
```

Error (500):

```
{
  "error": "Error adding product"
}
```

3. DELETE /product/:productId

Description: Delete a product owned by the authenticated seller.

- **Headers:**
  - `Authorization`: Bearer token (JWT).
- **Request Params:**
  - `productId` (string, required): The ID of the product to delete.
- **Response:**

Success (200):

```
{
  "message": "Product deleted successfully"
}
```

Error (404):

```
{
  "message": "Product not found or not owned by you"
}
```

Error (500):

```
{
  "error": "Error deleting product"
}
```

4. PATCH /product/:productId

Description: Update an existing product owned by the authenticated seller.

- **Headers:**
  - `Authorization`: Bearer token (JWT).
- **Request Params:**
  - `productId` (string, required): The ID of the product to update.

Request Body:

```
{
  "title": "Updated Product A",
  "price": 150,
  "stock": 100
}
```

- Any combination of product fields (`title`, `description`, `price`, `category`, `stock`) can be updated.

Response:

Success (200):

```
{
  "message": "Product updated successfully",
  "updatedProduct": {
    "_id": "12345",
    "seller": "67890",
    "title": "Updated Product A",
    "description": "Description of Product A",
    "price": 150,
    "category": "Electronics",
    "stock": 100,
    "updatedAt": "2024-11-01T12:00:00Z"
  }
}
```

Error (404):

```
{ "message": "Product not found or not owned by you" }
```

Error (500):

```
{ "error": "Error updating product" }
```

User Routes (`/user`)

1. GET /order

Description: Fetch all orders placed by the authenticated user.

- **Headers:**
  - `Authorization`: Bearer token (JWT).
- **Response:**

Success **(200):**

```
[
  {
    "_id": "645abc1234",
    "user": "12345",
    "seller": "67890",
    "product": "998877",
    "quantity": 2,
    "shippingAddress": "123 Main Street, Cityville",
    "paymentMethod": "Credit Card",
    "totalPrice": 200,
    "isPaid": true,
    "isDelivered": false,
    "createdAt": "2024-11-01T12:00:00Z"
  }
]
```

Error **(500):**

```
{
  "error": "Error fetching orders"
}
```

2. POST /order

Description: Place a new order for a product.

- **Headers:**
  - `Authorization`: Bearer token (JWT).

Request Body:

```
{
  "productId": "998877",
  "quantity": 2,
  "shippingAddress": "123 Main Street, Cityville",
  "paymentMethod": "Credit Card",
  "isPaid": true,
  "isDelivered": false
}
```

- `productId` (string, required): The ID of the product to order.
- `quantity` (number, required): The quantity of the product.
- `shippingAddress` (string, required): Address for delivery.
- `paymentMethod` (string, required): Payment method (e.g., Credit Card, PayPal).
- `isPaid` (boolean, optional): Whether the order is paid for (default: `false`).
- `isDelivered` (boolean, optional): Whether the order is delivered (default: `false`).

Response:

Success (201):

```
{
  "message": "Order created successfully",
  "order": {
    "_id": "645abc1234",
    "user": "12345",
    "seller": "67890",
    "product": "998877",
    "quantity": 2,
    "shippingAddress": "123 Main Street, Cityville",
    "paymentMethod": "Credit Card",
    "totalPrice": 200,
    "isPaid": true,
    "isDelivered": false,
    "createdAt": "2024-11-01T12:00:00Z"
  }
}
```

Error (400):

```
{
  "message": "All fields are required"
}
```

Error (404):

```
{
  "message": "Product not found"
}
```

Error (500):

```
{
  "error": "Error creating order"
}
```

3. DELETE /order/:orderId

Description: Cancel an existing order.

- **Headers:**
  - `Authorization`: Bearer token (JWT).
- **Request Params:**
  - `orderId` (string, required): The ID of the order to cancel.
- **Response:**

Success (200):

```
{
  "msg": "Cancel order"
}
```

Error (500):

```
{
  "error": "Error canceling order"
}
```