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

Response:

Success **(200):**

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

Error **(401):**

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

User Routes (`/user`)

1. GET /profile

Description: Fetch the profile details of the authenticated user

Authorization

- Required roles: `user`, `admin`

Headers:

- `Authorization`: Bearer token (JWT).

Response:

Success **(200):**

```
[
    {
        "_id": "67468b254b96c0aeb3262fe0",
        "name": "user",
        "email": "user@gmail.com",
        "role": "user",
        "createdAt": "2024-11-27T02:59:49.873Z",
        "updatedAt": "2024-11-27T02:59:49.873Z",
        "__v": 0
    }
]
```

Error **(404):**

```
{
  "error": "User not found""
}
```

Error **(500):**

```
{
  "error": "Failed to fetch profile""
}
```

2. PATCH /profile

Description: Update the authenticated user profile. Admins can update profiles for any user.

Authorization

- Required roles: `user`, `admin`

Headers:

- `Authorization`: Bearer token (JWT).

Request Body

- For sellers: Any updatable user fields like `name`, `email`, etc.
- For admins: Provide `userId` in the body to specify which user's profile to update.

Response:

Success **(200):**

```
{
  "message": "Profile updated successfully",
  "user": {
    "_id": "645f6...",
    "name": "Updated user Name",
    "email": "updated.email@example.com",
    "role": "user"
  }
}
```

Error **(404):**

```
{
  "error": "User not found""
}
```

Error **(500):**

```
{
  "error": "Failed to update profile""
}
```

3. DELETE /profile

Description: Delete the authenticated user profile. Admins can delete profiles for any user.

Authorization

- Required roles: `user`, `admin`

Headers:

- `Authorization`: Bearer token (JWT).

Request Body

- For user: No additional fields required.
- For admins: Provide `userId` in the body to specify which user's profile to delete.

Response:

Success **(200):**

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

Error **(404):**

```
{
  "error": "User not found""
}
```

Error **(500):**

```
{
  "error": "Failed to fetch profile""
}
```

Seller Routes (`/seller`)

1. GET /profile

Description: Fetch the profile details of the authenticated seller

Authorization

- Required roles: `seller`, `admin`

Headers:

- `Authorization`: Bearer token (JWT).

Response:

Success **(200):**


```
[
    {
        "_id": "67468b254b96c0aeb3262fe0",
        "name": "seller",
        "email": "seller@gmail.com",
        "role": "seller",
        "createdAt": "2024-11-27T02:59:49.873Z",
        "updatedAt": "2024-11-27T02:59:49.873Z",
        "__v": 0
    }
]
```


Error **(404):**


```
{
  "error": "User not found""
}
```


Error **(500):**


```
{
  "error": "Failed to fetch profile""
}
```

2. PATCH /profile

Description: Update the authenticated seller's profile. Admins can update profiles for any user.

Authorization

- Required roles: `seller`, `admin`

Headers:

- `Authorization`: Bearer token (JWT).

Request Body

- For sellers: Any updatable user fields like `name`, `email`, etc.
- For admins: Provide `userId` in the body to specify which user's profile to update.

Response:

Success **(200):**

```
{
  "message": "Profile updated successfully",
  "user": {
    "_id": "645f6...",
    "name": "Updated Seller Name",
    "email": "updated.email@example.com",
    "role": "seller"
  }
}
```

Error **(404):**

```
{
  "error": "User not found""
}
```

Error **(500):**

```
{
  "error": "Failed to update profile""
}
```

3. DELETE /profile

Description: Delete the authenticated seller's profile. Admins can delete profiles for any user.

Authorization

- Required roles: `seller`, `admin`

Headers:

- `Authorization`: Bearer token (JWT).

Request Body

- For sellers: No additional fields required.
- For admins: Provide `userId` in the body to specify which user's profile to delete.

Response:

Success **(200):**

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

Error **(404):**

```
{
  "error": "User not found""
}
```

Error **(500):**

```
{
  "error": "Failed to fetch profile""
}
```

Product Routes (`/product`)

1. GET /all

Description: Fetch a list of all products

- **Response:**

Success **(200):**

```
[
  {
    "_id": "64cf...",
    "name": "Product 1",
    "price": 100,
    "description": "Description of Product 1",
    "category": "Category 1",
    "createdAt": "2024-11-01T00:00:00.000Z"
  },
  {
    "_id": "64df...",
    "name": "Product 2",
    "price": 200,
    "description": "Description of Product 2",
    "category": "Category 2",
    "createdAt": "2024-11-02T00:00:00.000Z"
  }
]
```

Error **(500):**

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

2. GET /:productId

Description: Fetch details of a specific product by its ID.

Request Body

- `productId` (string): The ID of the product to retrieve.

Response:

Success **(200):**

```
{
  "_id": "64cf...",
  "name": "Product 1",
  "price": 100,
  "description": "Description of Product 1",
  "category": "Category 1",
  "createdAt": "2024-11-01T00:00:00.000Z"
}
```

Error **(404):**

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

Error **(500):**

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

3. POST /

Description: Add a new product under the authenticated seller.

Headers

- `Authorization`: Bearer token (JWT).

Authorization

- Required roles: `seller`, `admin`

Request Body:

- `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 **(200):**

```
{
  "message": "Product updated successfully",
  "product": {
    "_id": "64cf...",
    "name": "Updated Product Name",
    "price": 200,
    "description": "Description of the product",
    "category": "Category 1",
    "updatedAt": "2024-11-30T00:00:00.000Z"
  }
}
```

Error (400):

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

Error (500):

```
{  "error": "Failed to update product" }
```

4. PATCH /

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

Authorization:

- Required roles: `seller`, `admin`

Headers:

- `Authorization`: Bearer token (JWT).

Request Body:

```
{
  "productId": "64cf...",
  "name": "Updated Product Name",
  "price": 200
}
```

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

Response:

Success **(200):**

```
{
  "message": "Product updated successfully",
  "product": {
    "_id": "64cf...",
    "name": "Updated Product Name",
    "price": 200,
    "description": "Description of the product",
    "category": "Category 1",
    "updatedAt": "2024-11-30T00:00:00.000Z"
  }
}
```

Error **(404):**

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

Error **(500):**

```
{  "error": "Failed to update product" }
```

5. DELETE /

Description: Delete an existing product.

Authorization:

- Required roles: `seller`, `admin`

Headers:

- `Authorization`: Bearer token (JWT).

Request Body

```
{
  "productId": "64cf..."
}
```

Response:

Success **(200):**

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

Error **(404):**

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

Error **(500):**

```
{
  “error": "Failed to delete product"
}
```

Order Routes (`/order`)

1. POST /

Description: Allows a user to place an order for a product.

Authorization

- Required role: `user`

Headers

- `Authorization: Bearer <token>`

Request Body

```
{
  "productId": "64cf...",
  "quantity": 2,
  "shippingAddress": "123 Street, City, Country",
  "paymentMethod": "Credit Card",
  "isPaid": true,
  "isDelivered": false
}
```

Response

Success (201):

```
{
  "message": "Order created successfully",
  "order": {
    "_id": "64ef...",
    "user": "64df...",
    "seller": "64ff...",
    "product": "64cf...",
    "quantity": 2,
    "shippingAddress": "123 Street, City, Country",
    "paymentMethod": "Credit Card",
    "totalPrice": 200,
    "isPaid": true,
    "isDelivered": false
  }
```

```
}
```

Error (400):

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

Error (404):

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

Error (500):

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

2. GET /all

Description: Fetch all orders in the system.

Authorization

- Required role: `admin`

Headers

- `Authorization: Bearer <token>`

Response

Success (200):

```
{
  "orders": [
    {
      "_id": "64ef...",
      "user": "64df...",
      "seller": "64ff...",
      "product": "64cf...",
      "quantity": 2,
      "shippingAddress": "123 Street, City, Country",
      "paymentMethod": "Credit Card",
      "totalPrice": 200,
      "isPaid": true,
      "isDelivered": false
    }
  ]
}
```

Error (500):

```
{
  "error": "Error Fetching Orders"
}
```

3. GET /

Description: Fetch all orders placed by the authenticated user.

Authorization

- Required roles: `user`, `admin`

Headers

- `Authorization: Bearer <token>`

Admin-Specific Behavior

- Admin can pass a `userId` in the request body to fetch orders for a specific user.

Request Body (Admin Only)

```
{
  "userId": "64df..."
}
```

Response

Success **(200):**

```
[
  {
    "_id": "64ef...",
    "user": "64df...",
    "seller": "64ff...",
    "product": "64cf...",
    "quantity": 2,
    "shippingAddress": "123 Street, City, Country",
    "paymentMethod": "Credit Card",
    "totalPrice": 200,
    "isPaid": true,
    "isDelivered": false
  }
]
```

Error **(500):** `{  "error": "Error Fetching Orders" }`

4. DELETE /

Description: Allows a user or admin to cancel and delete an order.

Authorization

- Required roles: `user`, `admin`

Headers

- `Authorization: Bearer <token>`

Admin-Specific Behavior

- Admin can pass a `userId` in the request body to fetch orders for a specific user.

Request Body (Admin Only)

```
{
  "userId": "64df..."
}
```

Response

Success **(200):**

```
{
  "message": "Order has been successfully cancelled and
deleted"
}
```

Error **(404):**

```
{
  "error": "Order not found"
}
```

Error **(400):**

```
{
  "error": "Order has already been delivered and cannot be
cancelled"
}
```

Error **(500):**

```
{
  "error": "Error cancelling and deleting the order"
}
```

Admin Routes (`/admin`)

1. GET /users

Description: Fetch all users with the role `user`.

Authorization

- Required role: `admin`

Headers

- `Authorization: Bearer <token>`

Response

Success **(200):**

```
[
  {
    "_id": "63b91d72fa45d64fd2c3ed0b",
    "name": "John Doe",
    "email": "john@example.com",
    "role": "user"
  },
  {
    "_id": "63b91d72fa45d64fd2c3ed0c",
    "name": "Jane Smith",
    "email": "jane@example.com",
    "role": "user"
  }
]
```

Error **(500):**

```
{
  "error": "Failed to fetch users"
}
```

2. GET /sellers

Description: Fetch all users with the role `seller`.

Authorization

- Required role: `admin`

Headers

- `Authorization: Bearer <token>`

Response

Success **(200):**

```
[
  {
    "_id": "63c12d72fa45d64fd2c3ed0d",
    "name": "Vendor One",
    "email": "vendor1@example.com",
    "role": "seller"
  },
  {
    "_id": "63c12d72fa45d64fd2c3ed0e",
    "name": "Vendor Two",
    "email": "vendor2@example.com",
    "role": "seller"
  }
]
```

Error **(500):**

```
{
  "error": "Failed to fetch sellers"
}
```