E-commerce Project Documentation

Table of Contents

Project Overview

This is a full-stack e-commerce application focused on men's fashion, particularly footwear. The project consists of a frontend built with Angular and a backend built with Node.js, Express, and MongoDB. The application allows users to browse products, view details, add items to cart, place orders, and manage their profile.

Technology Stack

Frontend

- **Framework**: Angular (version 17.3.2)
- **Language**: TypeScript
- **Styling**: CSS
- **HTTP Client**: Axios
- **Icons**: Remix Icon (ri-*)

Backend

- **Runtime**: Node.js
- **Framework**: Express.js (version 4.21.2)
- **Database**: MongoDB with Mongoose (version 8.13.2)
- **File Upload**: Multer
- **Cross-Origin Resource Sharing**: CORS

Project Structure

The project follows a standard structure with separate directories for frontend and backend.

Pages

1. Home Page

- Main landing page with product categories and featured items
- Includes a slider/carousel for promotional content
- Allows filtering products by categories (boots, shoes, sandals, slipper, jogging)

2. Product Details Page

- Displays detailed information about a specific product
- Shows product images, name, price, description, and rating
- Includes size and color selection options
- Features "Add to Cart" and "Buy Now" functionality
- Shows similar or recommended products

3. About Us Page

- Information about the e-commerce platform
- Company introduction and values
- Team member profiles with social media links

4. Profile Page

- User account information and settings
- Personal details (name, email, phone number)
- Contact information (address, city, state, country, pincode)
- Password management
- Logout functionality
- For admin users: "Add Product" button

5. Cart Page

- Displays items added to the shopping cart
- Allows quantity adjustment and item removal
- Shows price calculations

6. Order Placed/Confirmation Page

- Confirmation screen after successful order placement

7. Add Product Page (Admin only)

- Form to add new products to the inventory

How to Run the Project

Prerequisites

- Node.js and npm installed
- Angular CLI installed
- MongoDB installed and running

Backend Setup

1. Navigate to the Backend directory:

   ```
   cd backend
   ```

2. Install dependencies:

   ```
   npm install
   ```

3. Start the development server:

   ```
   npm start
   ```

   The backend server will start on http://localhost:3000

Frontend Setup

1. Navigate to the Frontend directory:

   ```
   cd frontend
   ```

2. Install dependencies:

   ```
   npm install
   ```

3. Start the development server:

   ```
   ng serve
   ```

   The frontend application will be available at http://localhost:4200

API Endpoints

User Management

Authentication & User Profile

- `GET /api/v1/users/getuserbyid` - Get user information by ID
- `GET /api/v1/users/getuserbyidwithpass` - Get user information with password by ID
- `POST /api/v1/users/register` - Register a new user
- `POST /api/v1/users/login` - User login

Product Management

Product Retrieval

- `GET /api/v1/products/getallproducts` - Get all products
- `GET /api/v1/products/getproductbyid` - Get product by ID
- `GET /api/v1/products/getproductbyname` - Search products by name
- `GET /api/v1/products/getproductbycategory` - Get products by category

Product Administration

- `POST /api/v1/products/addproduct` - Add a new product
- `PATCH /api/v1/products/updateproduct` - Update an existing product
- `PATCH /api/v1/products/deleteproduct` - Delete a product (soft delete)

Order Management

Order Processing

- `POST /api/v1/orders/addorderhistory` - Add a new order to history

Cart Management

- Cart functionality is handled client-side using localStorage
- No specific API endpoints for cart operations

API Response Structure

The API uses standardized response formats:

- Success responses use ApiResponse class
- Error responses use ApiError class

Each response includes:

- `statusCode` - HTTP status code
- `success` - Boolean indicating success/failure
- `data` - Response data (for successful requests)

- `message` - Response message
- `errors` - Array of errors (for error responses)

Authentication

Most user-specific operations require authentication via a token stored in localStorage as `userToken`.

Features

Product Management

- Product browsing with category filtering
- Detailed product view with images and specifications
- Product search functionality

User Management

- User profile management
- Address and contact information storage
- Password management

Shopping Features

- Add to cart functionality
- Local storage for cart persistence
- Order placement
- Duplicate item detection in cart

Admin Features

- Product addition interface
- User management

Team Members

According to the About Us page, the project was developed by:

1. Aniket Chakkarwar

- Role: Backend Developer

2. Soloman Varghese

- Role: Frontend Developer