E-Commerce Platform Documentation

1. Project Overview

This project is a **Full-Stack E-Commerce Application** built using the MERN stack (MongoDB, Express.js, React.js, Node.js).
It provides user authentication, product browsing, cart management, checkout with Stripe, order management, and an admin dashboard for analytics.

Main Features

- User registration and login (JWT authentication)
- Product catalog with search, filter, and pagination
- Add to cart & wishlist functionality
- Order history and order tracking
- Admin dashboard to manage products, view analytics, and track sales
- Stock-based charting for admin

2. Technology Stack

Frontend

- React.js (Vite)
- Redux Toolkit (State Management)
- Tailwind CSS (Styling)
- Axios (API Calls)
- React Router DOM (Routing)
- Chart.js / Recharts (Admin Analytics)

Backend

- Node.js
- Express.js
- MongoDB + Mongoose

- JSON Web Token (JWT) for authentication
- bcrypt.js for password hashing
- Multer / Cloudinary (for image upload)
- Stripe API (for simulated payment)

3. User Roles & Responsibilities

3.1 Customer

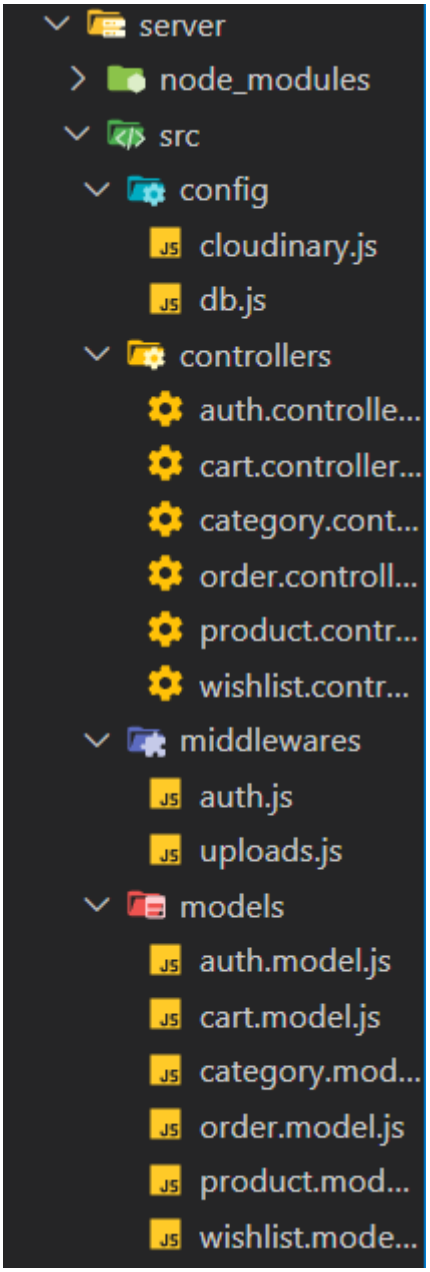
- Browse products with search & filters
- Add to cart or wishlist
- Checkout & place orders
- View order history
- Update profile details

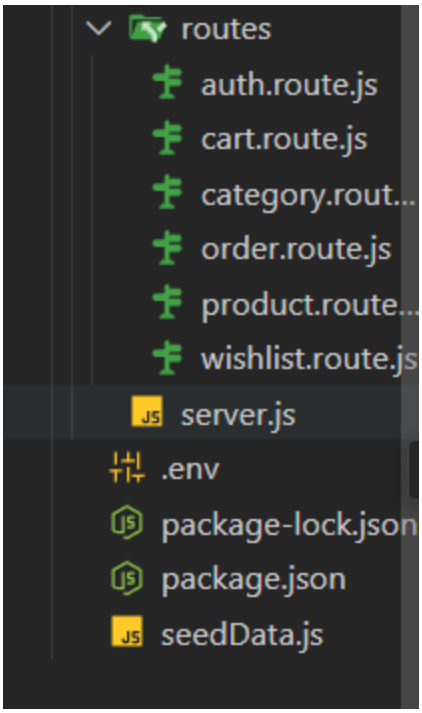
3.2 Admin

- Manage products (add, update, delete)
- Track total sales & stock analytics
- Manage orders (update status, cancel, refund)
- Access admin dashboard charts

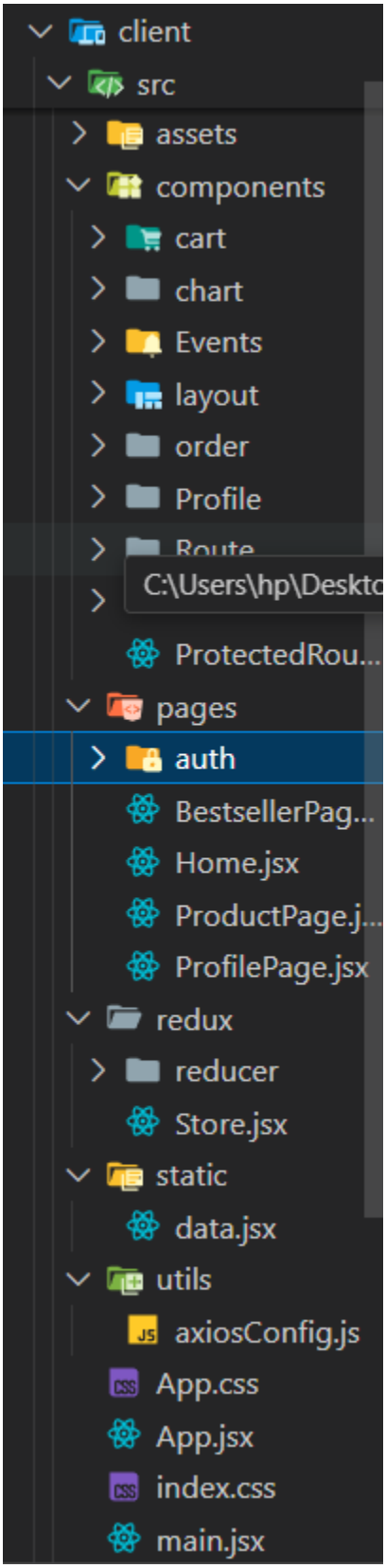
4. Folder Structure (MVC)

Backend (Express + MVC)




routes
auth.route.js
cart.route.js
category.rout...
order.route.js
product.route...
wishlist.route.js
server.js
.env
package-lock.json
package.json
seedData.js

Frontend (React + Redux Toolkit)



5. Feature-wise Component & API Flow

Feature	Frontend Component	Backend API Endpoint(s)
User Auth	Login.jsx, Register.jsx, Profile.jsx	POST /api/auth/register, POST /api/auth/login, GET /api/auth/user, PUT /api/auth/user/address, PATCH /api/auth/user/profile-image
Product Listing	Home.jsx, ProductCard.jsx, ProductDetails.jsx	GET /api/products, GET /api/products/:id
Product Management (Admin)	AdminProduct.jsx, ProductForm.jsx	POST /api/products, PUT /api/products/:id, DELETE /api/products/:id
Cart Management	Cart.jsx	POST /api/cart, GET /api/cart, PUT /api/cart/:productId, DELETE /api/cart/:productId
Wishlist Management	Wishlist.jsx	POST /api/wishlist, GET /api/wishlist, DELETE /api/wishlist/:productId
Category Management (Admin)	AdminCategory.jsx, CategoryForm.jsx	GET /api/category, POST /api/category, PUT /api/category/:id, DELETE /api/category/:id
Orders	Checkout.jsx, Orders.jsx, OrderDetails.jsx	POST /api/orders, GET /api/orders/my-orders, GET /api/orders/:orderId, PATCH /api/orders/:orderId/status
Analytics (Admin)	AdminDashboard.jsx, AdminChart.jsx	GET /api/products (stock data), GET /api/orders (sales data)

6. Component Descriptions

- Navbar.jsx – Navigation bar with search, cart, wishlist, and authentication buttons.
- ProductCard.jsx – Displays product info, stock status, and an add-to-cart button.
- ProductDetails.jsx – Detailed product page with buy/add-to-wishlist functionality.
- AdminProduct.jsx – Admin view to manage product list.
- ProductForm.jsx – Admin form to add or edit products.

- Cart.jsx – Shows cart items with quantity updates and removal options.
- Wishlist.jsx – Displays saved wishlist items with remove option.
- AdminCategory.jsx – Admin view to manage product categories.
- Checkout.jsx – Manages order placement and payment integration (Stripe).
- Orders.jsx – Displays order history for the logged-in user.
- OrderDetails.jsx – Shows detailed info for a single order.
- AdminDashboard.jsx – Summary of key stats for the admin.
- AdminChart.jsx – Graphical representation of sales and stock data.

---

7. API Documentation

Authentication

- POST /api/auth/register → Register new user *(with optional profile image upload)*
- POST /api/auth/login → Login user
- GET /api/auth/user → Get logged-in user details *(Protected)*
- PUT /api/auth/user/address → Update user address *(Protected)*
- PATCH /api/auth/user/profile-image → Update profile image *(Protected)*

Products

- GET /api/products → Get all products
- GET /api/products/:id → Get single product by ID
- POST /api/products → Add product *(Admin only, with image upload)*
- PUT /api/products/:id → Update product *(Admin only, with image upload)*
- DELETE /api/products/:id → Delete product *(Admin only)*

Cart

- POST /api/cart → Add item to cart
- GET /api/cart → Get user cart
- PUT /api/cart/:productId → Update cart item quantity
- DELETE /api/cart/:productId → Remove cart item

Wishlist

- POST /api/wishlist → Add item to wishlist
- GET /api/wishlist → Get user wishlist
- DELETE /api/wishlist/:productId → Remove wishlist item

Category

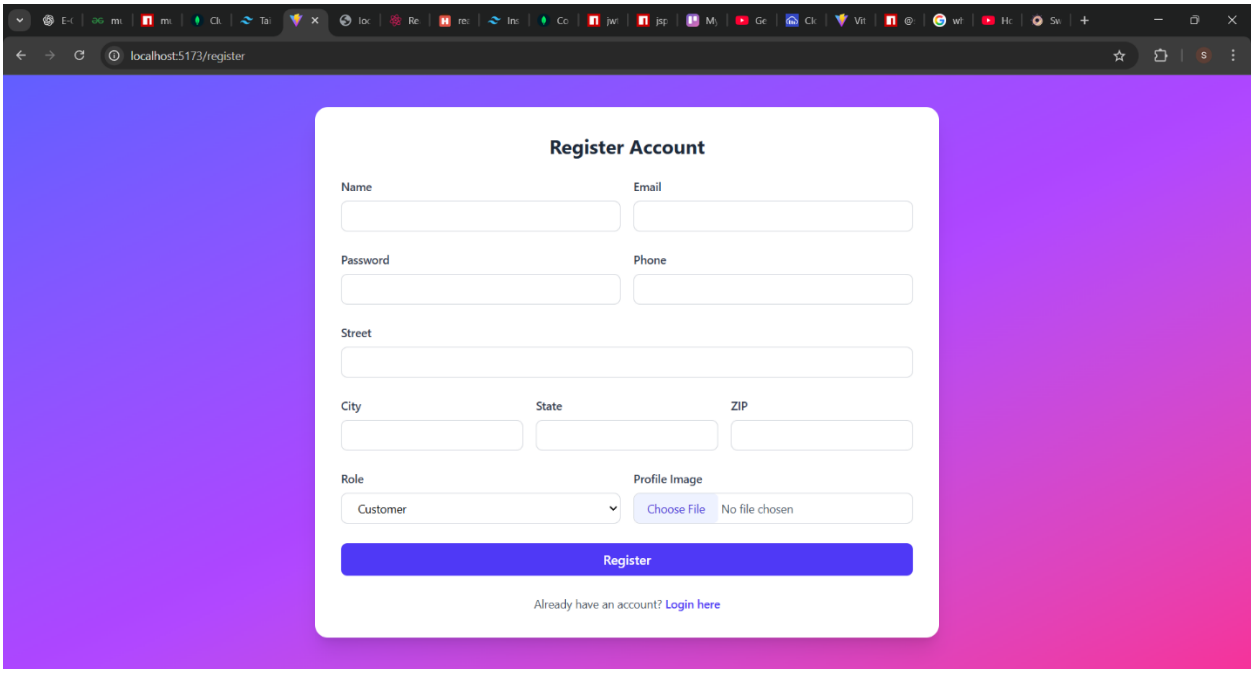
- GET /api/category → Get all categories
- POST /api/category → Create category *(Admin only)*
- PUT /api/category/:id → Update category *(Admin only)*
- DELETE /api/category/:id → Delete category *(Admin only)*

Orders

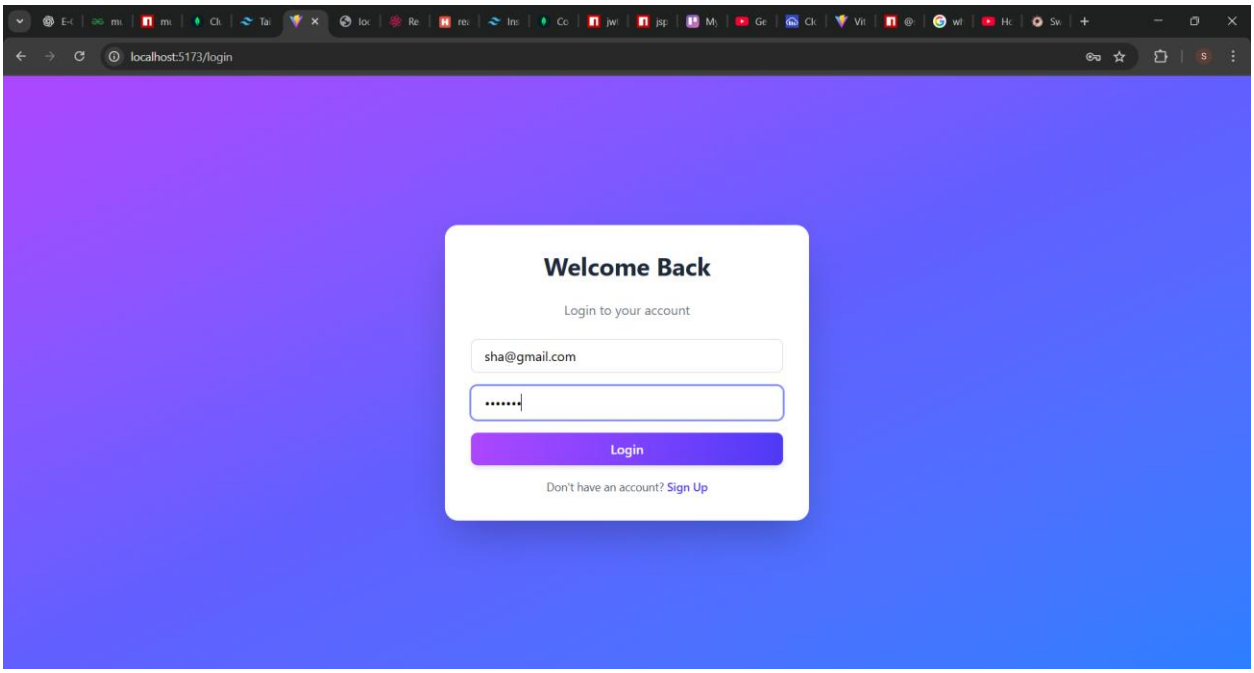
- POST /api/orders → Place new order
- GET /api/orders/my-orders → Get orders of logged-in user
- GET /api/orders/:orderId → Get single order details
- PATCH /api/orders/:orderId/status → Update order status *(Admin only)*

8. Screenshots

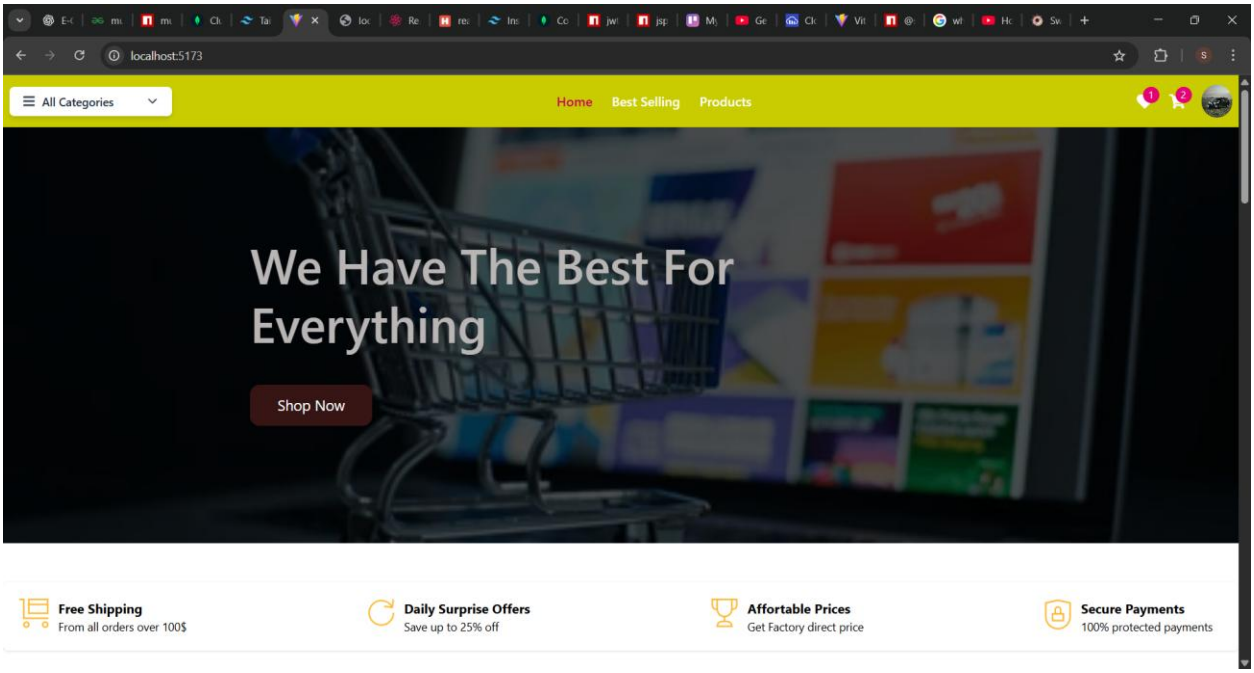
8.1 Registration



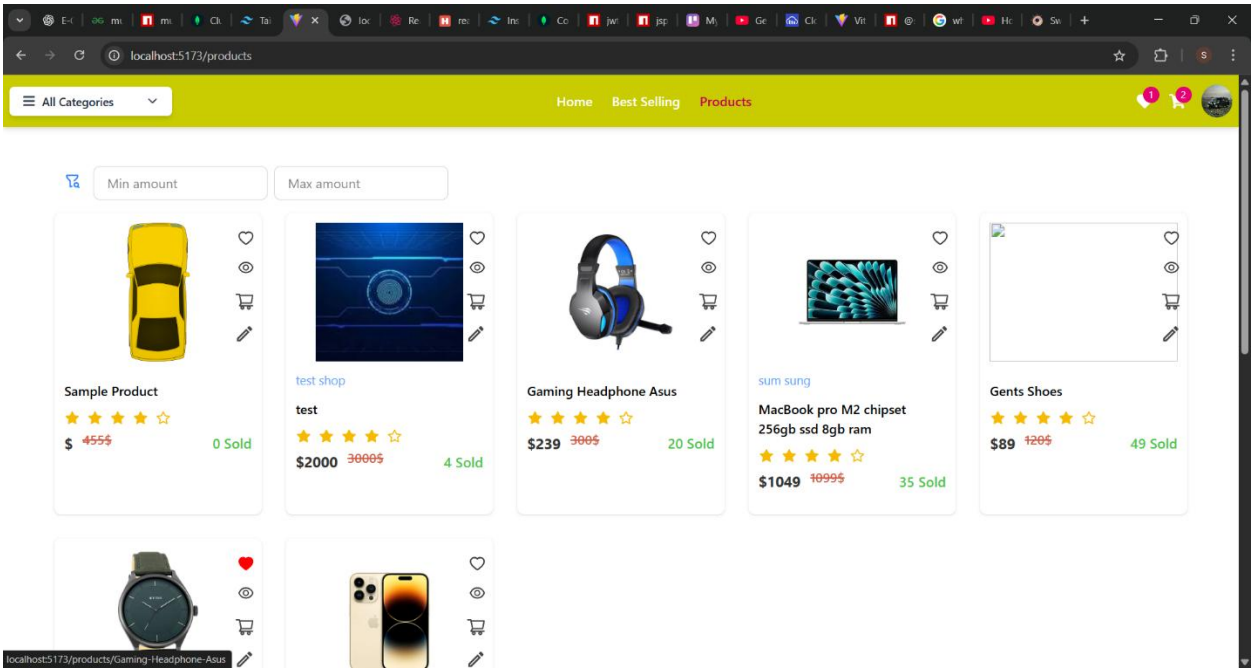
8.2 Login



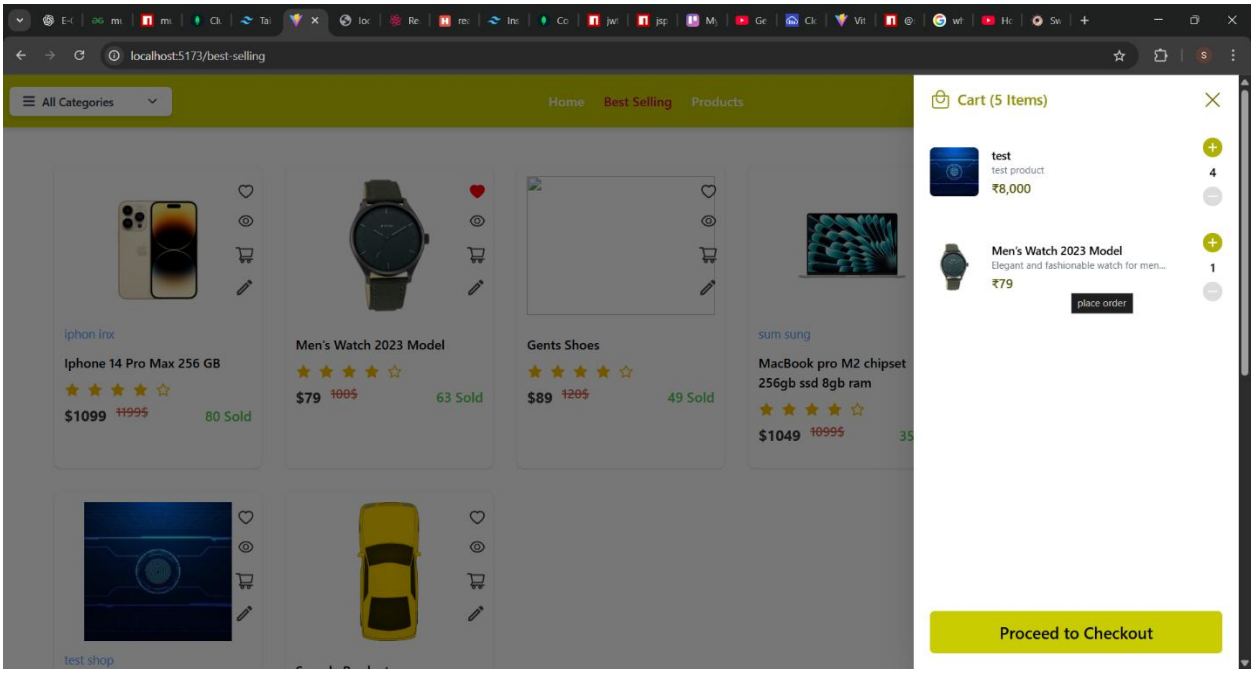
8.3 Home



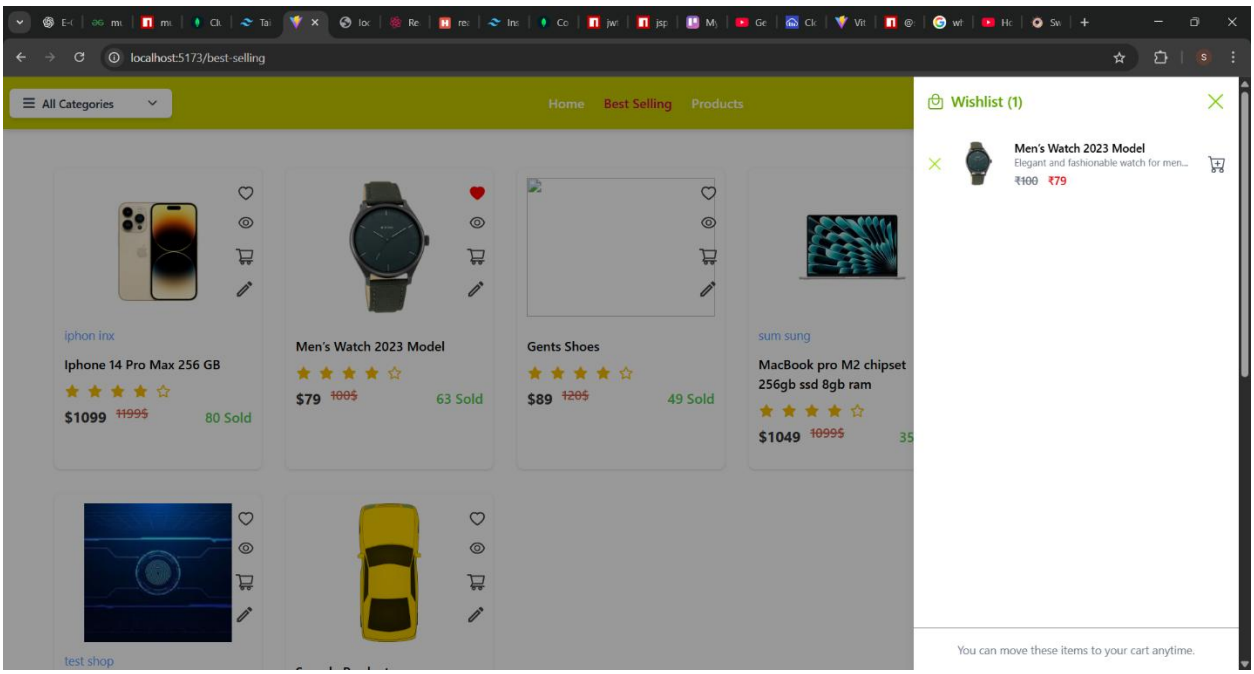
8.4 Products



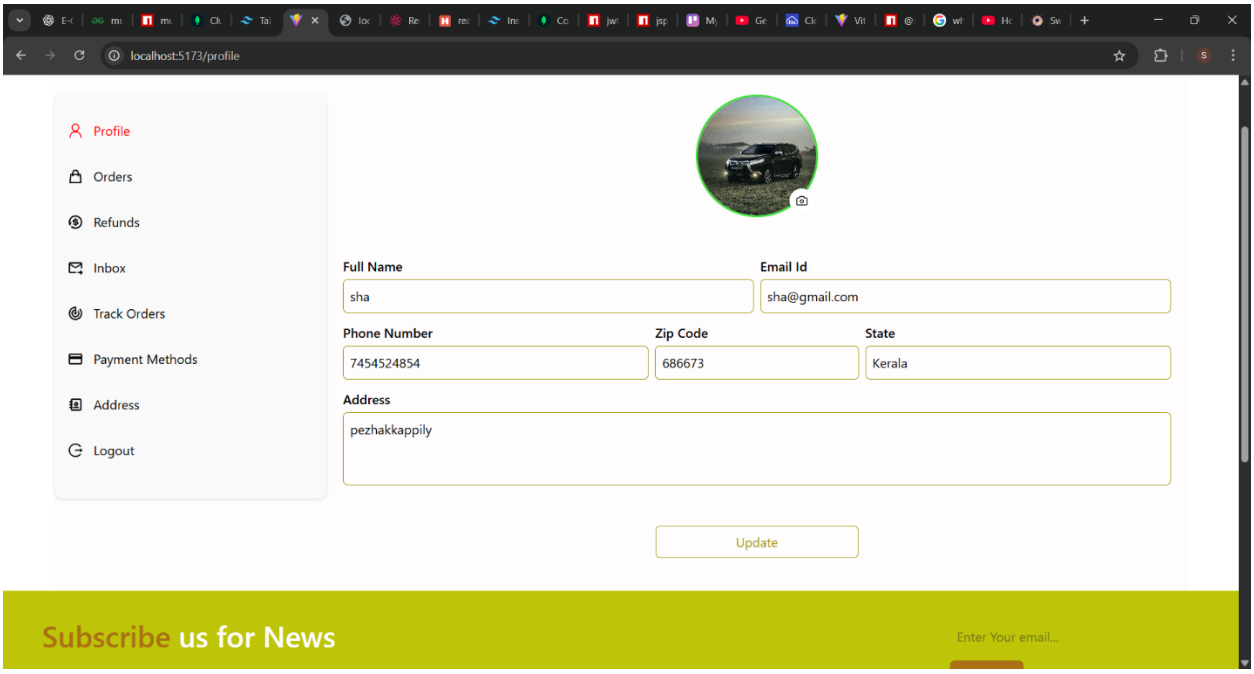
8.5 Cart



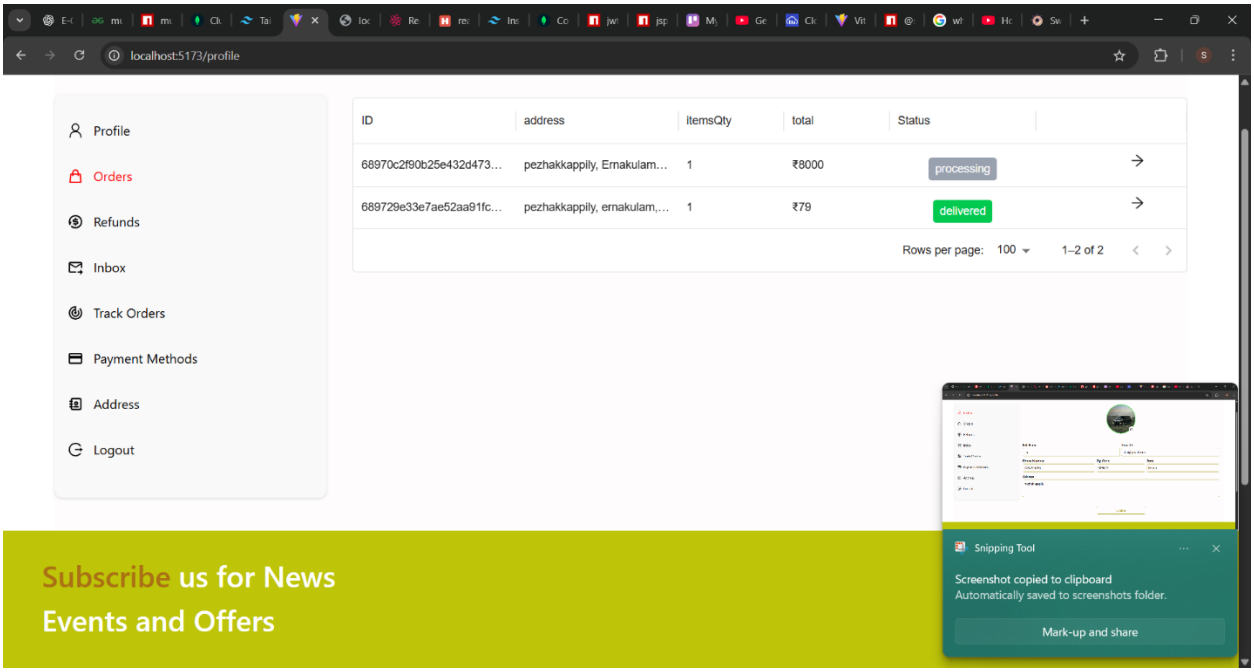
8.6 Wishlist



8.7 Profile



8.8 Order



9. How to Run the Project

9.1 Backend

cd server

npm install

npm run dev

9.2 Frontend

cd frontend

npm install

npm run dev

9.2 MongoDB Connection

- Use MongoDB Atlas or local MongoDB
- Update config/db.js with your connection string

10. Conclusion

Outcomes:

- Successfully built a role-based e-commerce application
- Integrated Stripe API for simulated payments
- Implemented analytics using Chart.js/Recharts

Challenges Faced:

- Managing role-based routing & permissions
- Handling file uploads and storage (Cloudinary)