

ShopSmart—your digital grocery store experience

Full Stack Development with MERN—Project Documentation

1. Introduction

- **Project Title:** Shop Smart – Your Digital Grocery Store Experience
 - **Team ID:** LTVIP2026TMIDS47270
 - **Team Size:** 5
 - **Team Members:**
 - **Team Leader:** Kollam lohitha
 - **Team member:** Konde poojitha
 - **Team member:** Yuvaraju Konedhala
 - **Team member:** Middepogu yuva raju
 - **Team member:** M Mallavika
-

2. Project Overview

- **Purpose:**

To build a user-friendly online grocery shopping platform where users can browse, search, and purchase groceries, while administrators can manage inventory and orders through a powerful backend.
 - **Key Features:**
 - User Registration/Login
 - Product Search and Filter
 - Shopping Cart and Wishlist
 - Order Placement and History
 - Admin Dashboard for Product and Order Management
 - Secure authentication and role-based access
-

3. Architecture

- **Frontend(React):**
 - Components: Home, ProductList, Cart, Wishlist, Login, Register, Orders, Admin Dashboard
 - Libraries: React Router, Axios, Bootstrap
- **Backend(Node.js+Express.js):**
 - REST APIs for all user and admin operations
 - JWT-based authentication and middleware for role protection
 - Routes: /auth, /products, /orders, /admin
- **Database(MongoDBAtlas):**
 - Collections: Users, Products, Orders
 - CRUD operations via Mongoose ORM
 - Indexed for performance

4. Setup Instructions

- **Prerequisites:**
 - Node.js(v18+)
 - MongoDB Atlas account
 - Git
- **Installation:**

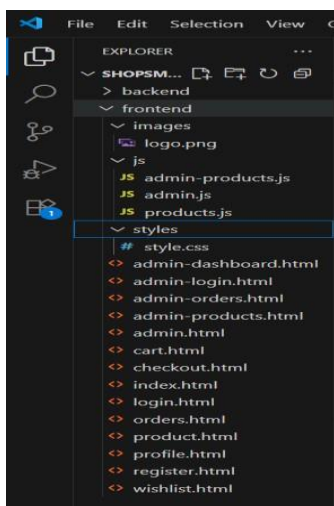
```
git clone https://github.com/Lohithareddy071/shopsmart
cd shopsmart
cd server
npm install
cd ../client
npm install
```

- **Environment Variables(.env):**
In server/folder:

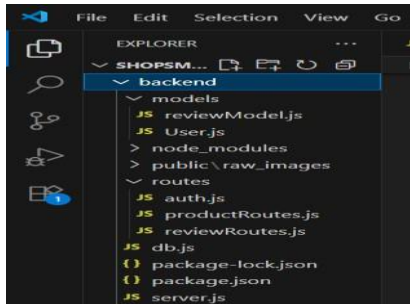
```
ini
MONGO_URI=your_mongodb_atlas_url
JWT_SECRET=your_jwt_secret
```

5. Folder Structure

- **Client (Frontend):**



- **Server(Node.jsBackend):**



6. Running the Application

- **Frontend:**

```
cd client
npm start
```

- **Backend:**

```
cd server
npm start
nodelserver.js
```

7. API Documentation

Endpoint	Method	Description
/api/auth/register	POST	User registration
/api/auth/login	POST	User login and token generation
/api/products	GET	List all products
/api/products/:id	PUT/DELETE	Admin:Update/Delete product
/api/orders	POST	Place new order
/api/orders/:userId	GET	Get all orders by user
/api/admin/orders	GET/PUT	Admin:View or update order status

8. Authentication

- **JWT-Based Authentication:**
 - Users receive a JWT token on login
 - Token stored in local Storage and passed in headers

- Role-based middleware(isAdmin,authUser) used to protect sensitive routes

9. User Interface

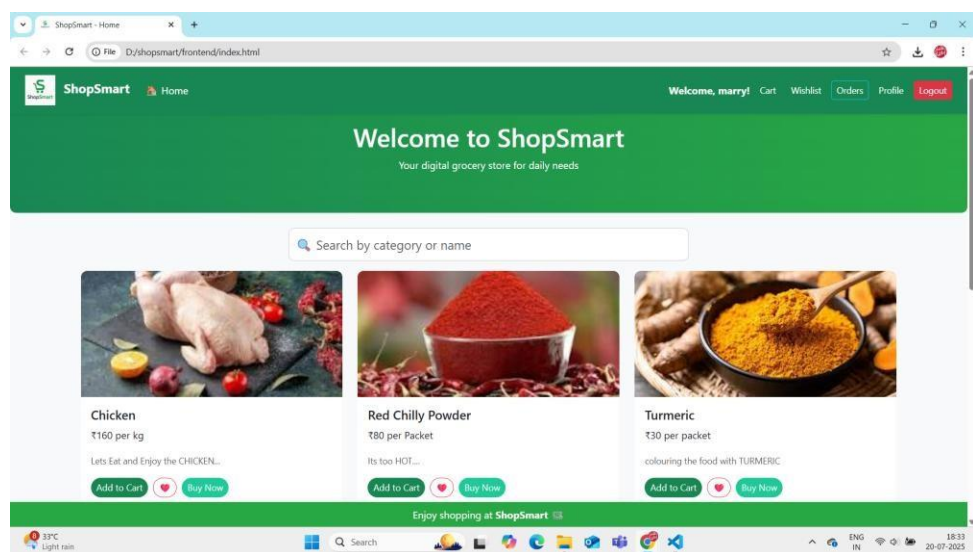
- Pages:
 - Home Page with productcards
 - Search bar,productfilter
 - Add to Cart/ Wishlist buttons
 - Order Historypage
 - Admin Dashboard:Add/Edit/Delete Products,Manage Orders
- **UI Technologies:**
React+Bootstrap5
Responsive layout for mobile & desktop

10. Testing

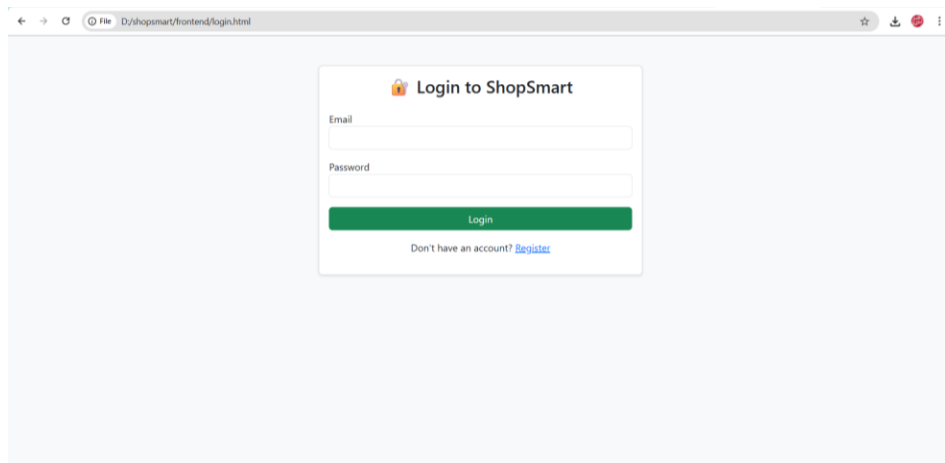
- **Tools Used:**
 - Postman (APItesting)
 - Manual UI testing on Chrome, Firefox, Mobile view
 - Console and network inspection using DevTools
- **Strategy:**
 - Unit tests forAPIs(optional)
 - Manual tests for all user/admin flows

11. Screenshots & Demovideo

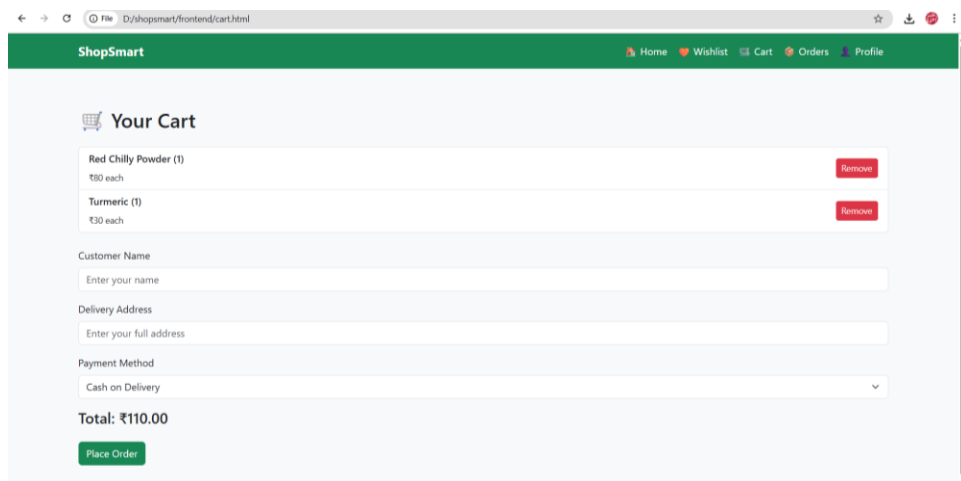
- Homepage



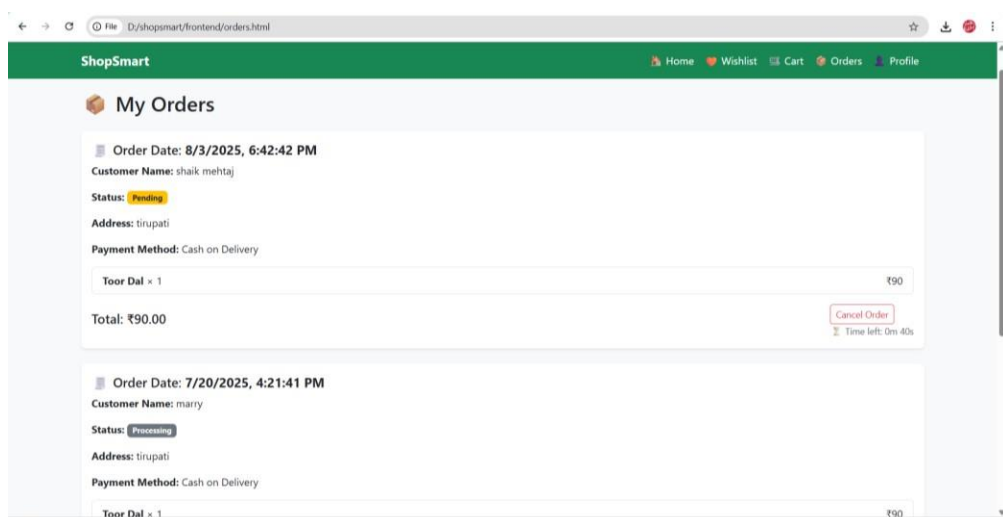
- Login/Register



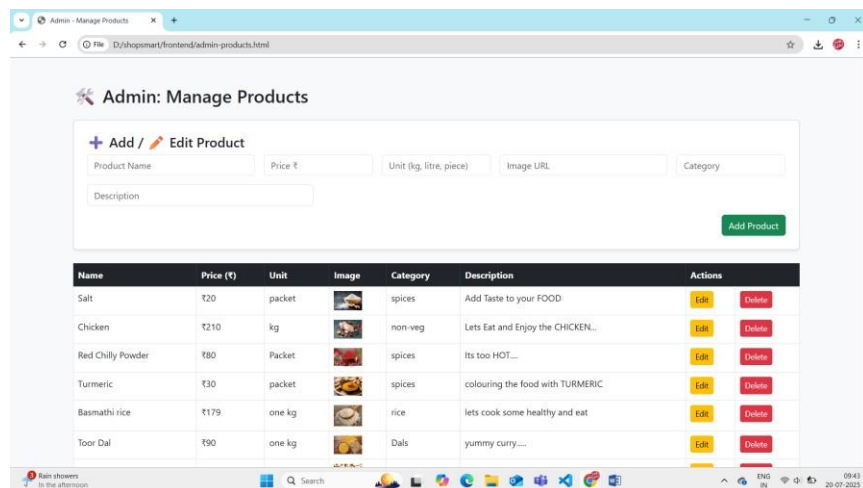
- Cart



- Orders



- Admin Dashboard



- **Demo Link:**
https://drive.google.com/file/d/1npDzANr0_yQ9ahc1rsONG8BpFcjdmzxk/view?usp=drive_link

12. Known Issues

- Payment gateway not integrated
- No real-time delivery tracking
- Cart items don't persist on logout

13. Future Enhancements

- Integrate Razorpay/Stripe for secure payments
- Build mobile app with ReactNative
- Add delivery tracking and notification system
- Implement product ratings and feedback