FULL STACK DEVELOPMENT

WITH MERN PROJECT DOCUMENTATION

INTRODUCTION

Project Title: ShopSmart: Your Digital Grocery Store Experience

Team ID: LTVIP2025TMID57814

Name: J.Hemanth [234E1A4727]

PROJECT OVERVIEW

Shop Mart is a user-centric, modern grocery store web application developed using the MERN stack. It provides features like browsing a wide range of grocery items, filtering by category, maintaining a dynamic shopping cart, managing user authentication, placing and tracking orders, and an admin dashboard for managing products and users.

With an aim to digitize grocery shopping for local stores and end-users alike, the application offers real-time interaction, smooth UI experience, and secure data handling.

Key features include:

- User authentication with role-based access
- Interactive product catalog with images and search filters
- Cart management and real-time stock validation
- Admin panel to add/edit/delete products and manage customer orders
- Order tracking and history logs for users
- Real-time inventory updates
- Responsive web design for mobile and desktop

Shop Mart bridges the gap between technology and daily needs, making grocery shopping accessible to all with just a few clicks.

SCENARIO-BASED CASE STUDY

Ravi, a busy software engineer, has no time to visit physical stores. He logs into Shop Mart and creates his grocery list. The platform allows him to filter products by category, check availability in real-time, and add them to his cart. Before placing the order, he uses the chat to confirm delivery timelines with the admin.

The admin assures him of same-day delivery, and Ravi makes the payment securely through the integrated payment system. The groceries are delivered on time, and Ravi rates the service 5 stars. Shop Mart becomes his go-to for weekly grocery shopping.

Case Study 2: Lakshmi - A local vendor's success

Lakshmi owns a small organic vegetable store. With Shop Mart, she lists her products online. Within weeks, her local customer base expands beyond her area. The admin dashboard allows her to update stock, prices, and offers. She gets real-time order notifications, reducing manual inventory management. Her revenue increases by 30% in two months, and Shop Mart becomes the digital face of her business.

TECHNICAL ARCHITECTURE

- **Frontend:** React.js (Hooks, Axios, React Router), Bootstrap, Material UI
- **Backend:** Node.js, Express.js
- **Database:** MongoDB using Mongoose ODM
- **Architecture Pattern:** MVC (Model View Controller)

The architecture follows the client-server model:

- The frontend (React) sends HTTP requests using Axios
- The backend (Express) handles the request logic and responds
- MongoDB stores the user data, product info, and order details

Security Measures Implemented: - Passwords are hashed using Bcrypt - JSON Web Tokens (JWT) are used for user session management - Role-based access for user and admin - CORS enabled for secure API communication

Performance Optimizations: - Lazy loading components in React - MongoDB indexing for faster search - Code splitting and minification

---

ER DIAGRAM OVERVIEW

- **User Table:** ID, Name, Email, Password, Role (user/admin), Address
- **Product Table:** ID, Name, Category, Price, Stock, ImageURL, Description
- **Cart Table:** CartID, UserID, ProductID, Quantity
- **Order Table:** OrderID, UserID, Products[], Amount, Date, Status, DeliveryAddress
- **Admin Table:** Inherits from User with elevated permissions

Relationships: - One user can have many orders - One product can be in many carts and orders

---

PROJECT STRUCTURE

```
/shop-mart
├── client
│   ├── public
│   └── src
│       ├── components
│       ├── pages
│       ├── context
│       ├── services
│       └── App.js
├── server
│   ├── models
│   ├── routes
│   ├── controllers
│   ├── middleware
│   └── server.js
```

The modular project structure promotes scalability, with services separated for API calls and global context for managing state.

PRE-REQUISITES

- Node.js & npm (for backend and frontend package management)
- MongoDB (local or Atlas cloud setup)
- React.js with Create React App
- Git & GitHub for source control
- VS Code or preferred IDE

Important Libraries & Tools: - Axios, React-Bootstrap, Material-UI - Bcrypt.js for hashing - JSON Web Token (JWT) for authentication - Mongoose for schema and validation - CORS for cross-origin requests - React Context API for state management - Postman for API testing

---

APPLICATION FLOW

Users: - Register/Login securely - Browse products by category - Add items to cart - Checkout and place orders - View past orders in history - Update profile and delivery address

Admins: - Login with secure credentials - Add/Edit/Delete products - View and manage all customer orders - Update product stock and categories - Manage users and respond to customer queries

Middleware Functions: - Auth check for protected routes - Role verification for admin-only actions

PROJECT IMPLEMENTATION

Milestone 1: Initial Setup - Create folders for client/server - Initialize npm and install dependencies - Setup GitHub repository

Milestone 2: Backend API Development - Define Mongoose models (User, Product, Order, Cart) - Create API routes (GET, POST, PUT, DELETE) - Use JWT for login authentication - Encrypt passwords with Bcrypt - Write middlewares for protected routes

Milestone 3: Frontend Development - Create components: Header, Footer, ProductCard, CartItem, OrderList - Setup routing with React Router DOM - Create pages: Home, Login, Register, Cart, Orders, Admin Dashboard - Use Context API for managing user and cart state

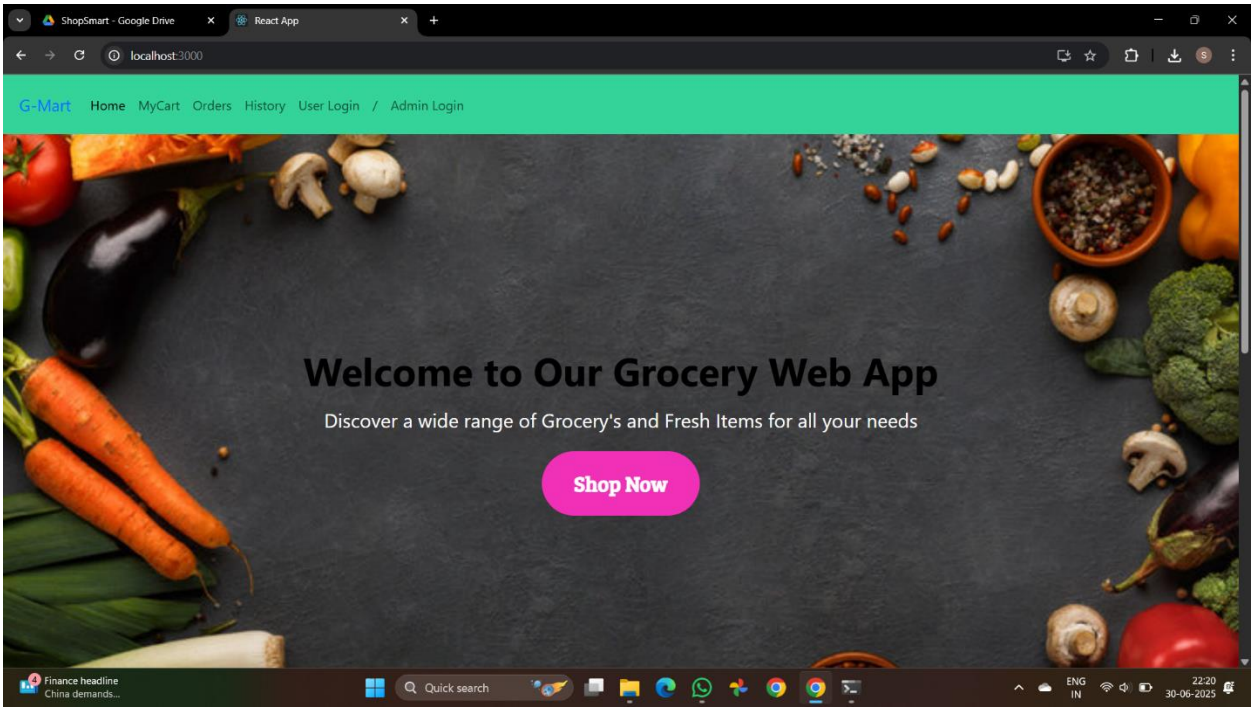
Milestone 4: Integration and Testing - Connect frontend to backend via Axios - Implement protected and dynamic routes - Perform unit testing with Jest and API testing using Postman - Debug and fix UI/UX issues

Milestone 5: Deployment - Deploy client to Netlify/Vercel - Deploy server to Render/Heroku - Configure environment variables securely - Use MongoDB Atlas for remote database access

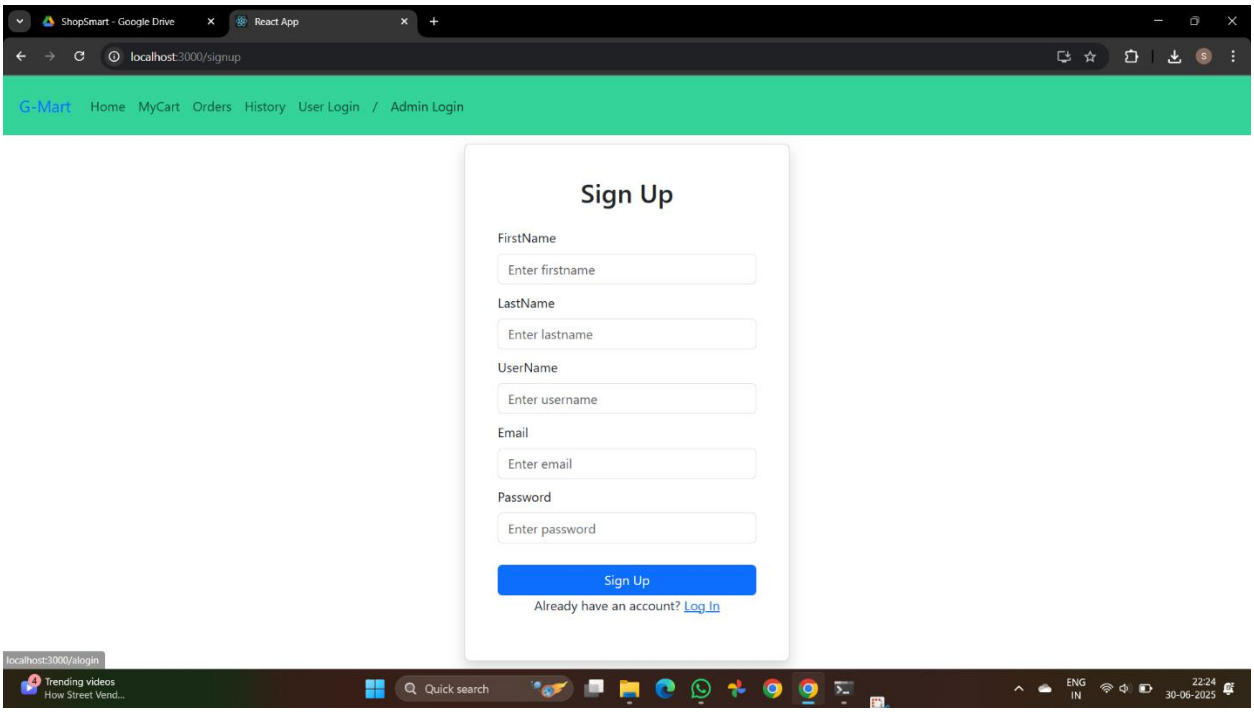
---

OUTPUT SCREENS (attach screenshots): - Home/Landing Page with featured products - Login/Register page with validation - Product listing page with filters and search - Cart page showing total price, items, quantities - Checkout flow and order confirmation - Order history with status updates - Admin dashboard with CRUD operations - Responsive mobile view of all components

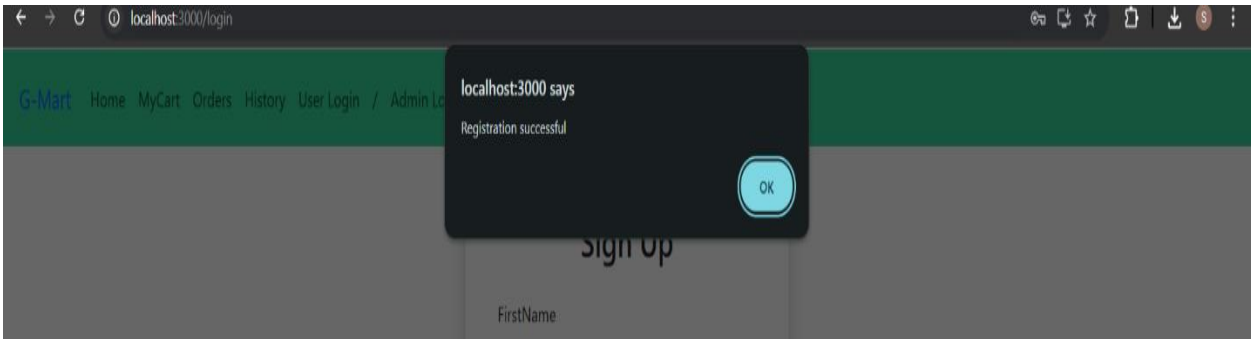
-Landing page



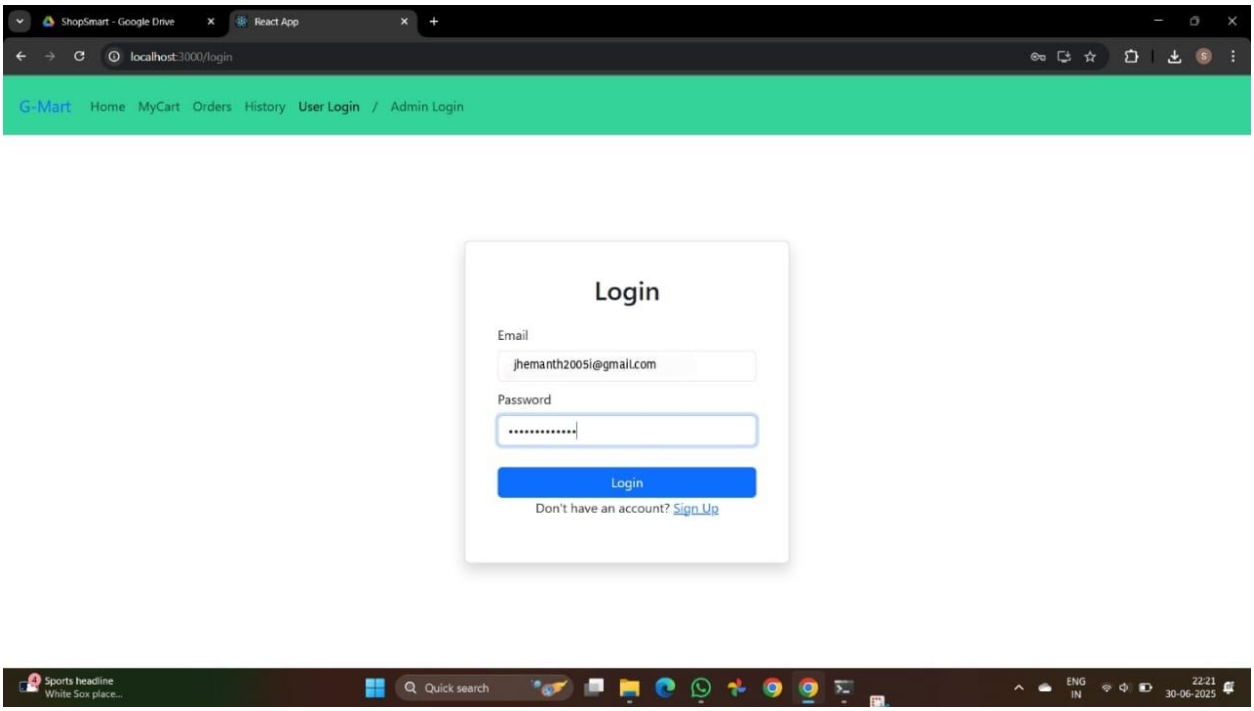
-Signup page



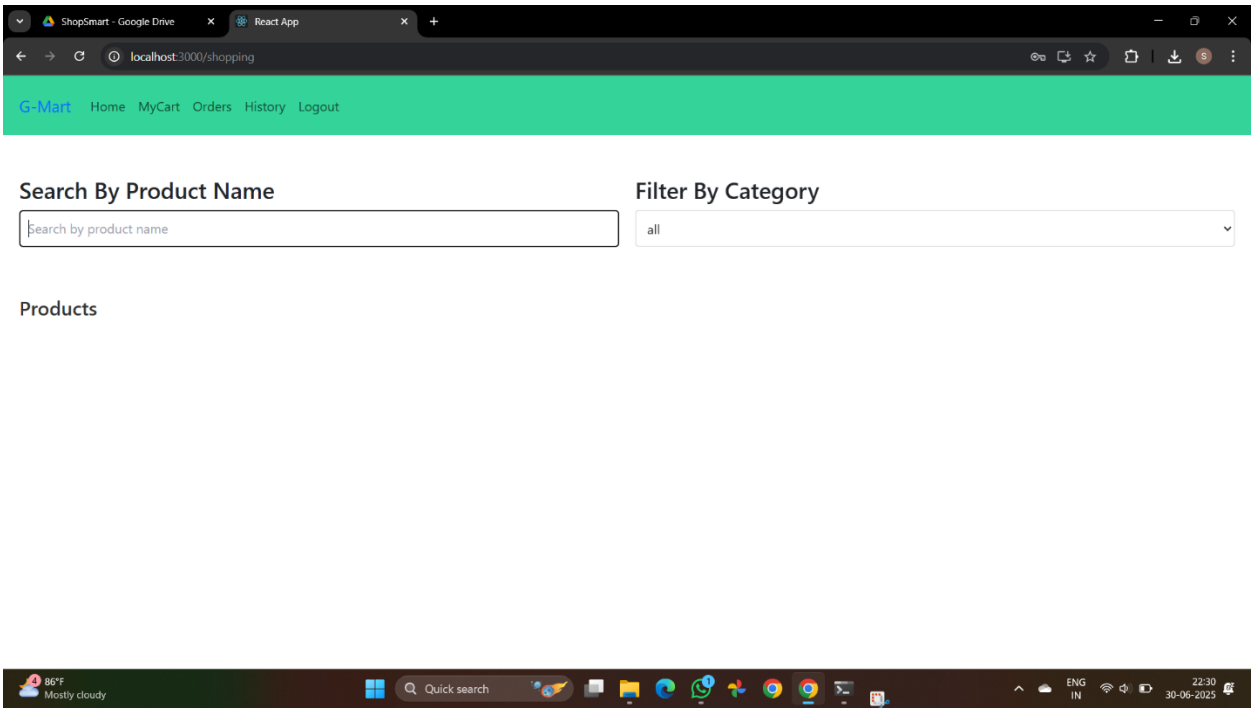
-Alert after Login



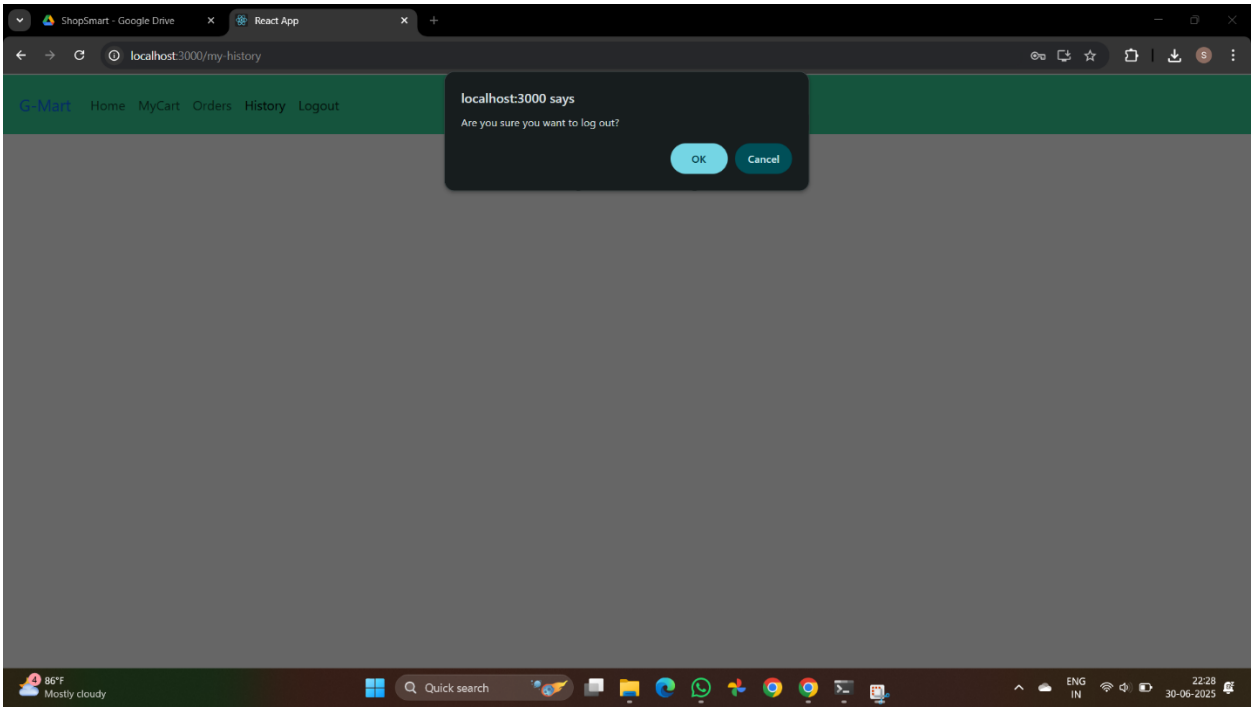
-Login/Register page



-Shop Mart Search page:



-Logout page



ADVANTAGES - Clean, mobile-responsive UI - Secure login and order system - Efficient stock and order management - Easy admin panel interface - Scalable design for future updates - Modern UI with Material-UI components

DISADVANTAGES - Requires internet connection - No built-in customer support chat (planned) - No PWA offline support (can be added later)

CONCLUSION

The Shop Mart MERN application showcases the power of full-stack JavaScript in transforming traditional businesses into modern e-commerce platforms. With real-time updates, secure APIs, and responsive design, Shop Mart stands out as a robust solution for local grocery digitization.

This project not only enhances user experience but also equips shopkeepers with a simple, efficient, and scalable system to manage inventory, engage with customers, and track orders seamlessly.

FUTURE SCOPE - Integrate payment gateways (Razorpay, Stripe) - OTP-based login for better security - Native mobile app using React Native - Chatbot and customer support integration - AI-based recommendations for products - Delivery partner integration with live tracking - Analytics dashboard for admin with charts

---

DEMO LINK:

“C:\Users\hemanth\video-demo\video-5q7189-3e34.mp4”

SOURCE CODE:

https://github.com/JHemanth2005/ShopSmart-Project

Developed using MERN Stack
Empowering Local Stores Through Technology
~ The Shop Mart Team