Shopsmart – Your Digital Grocery Store Experience

Team Members:

TEAM ID : LTVIP2025TMID59331

MEMBER	RESPONSIBILITY
1. Katakam Chandrika	Frontend,Database
2.Jammisetty Pragna	Backend,testing,Implementation
3.Govathoti Pooja	Documentation,Demo
4.Hanumanthu Pavankumar	Resources gathering,Design

1. Introduction

Project Title:
Shopsmart – Your Digital Grocery Store Experience

2. Project Overview

Purpose:

"Welcome to our Grocery Web App, your one-stop shop for all your grocery needs! With our user-friendly interface and wide selection of high-quality products, we aim to make your grocery shopping experience convenient and enjoyable. Whether you're looking for fresh produce, pantry staples, or household essentials, our app has you covered. Explore our virtual aisles, add items to your cart with ease, and have your groceries delivered right to your doorstep. Experience the future of grocery shopping with our Grocery Web App today!"

Features:

- **User Authentication & Roles**
  Secure login/signup system with role-based access for Customers, Sellers, and Admins.

- **Product Browsing & Search**
  Easy navigation with category filters and keyword search to find products quickly.

- **Shopping Cart & Checkout**
  Add products to cart, update quantities, and securely complete orders with a streamlined checkout.

- **Order Tracking & History**
  Customers can view their past orders and track the delivery status in real-time.

- **Seller Dashboard**
  Sellers can manage products, update inventory, and process orders from a single interface.

- **Admin Panel**
  Admins can manage users, monitor app performance, and oversee orders and product listings.

- **Product Management**
  Sellers can easily add, edit, or delete product listings with real-time stock updates.

- **Responsive Design**
  Mobile-friendly interface for a smooth experience across smartphones, tablets, and desktops.

- **Secure Payments**
  Safe and encrypted payment processing ensures user data and transactions are protected.

- **Analytics & Reports**
  Sellers and Admins get insights on sales, user behavior, and app performance.

3. Architecture

Frontend:

The frontend handles the user interface and experience — what customers, sellers, and admins interact with.

- **HTML5 / CSS3 / JavaScript**
- **React.js** *(Recommended)* – for building responsive, component-based UI
- **Redux / Context API** – for state management (e.g., cart, user data)
- **Axios / Fetch API** – for making API calls to the backend
- **Tailwind CSS / Bootstrap** – for modern and responsive styling
- **React Router** – for navigation and routing between pages

Backend:
The backend handles business logic, authentication, APIs, and database operations.

- **Node.js** – JavaScript runtime
- **Express.js** – Web framework for building RESTful APIs

- **JWT (JSON Web Tokens)** – for user authentication and secure routes
- **Multer / Cloudinary** – for file/image uploads (optional for product images)

Database:

MongoDB *(Recommended)* – NoSQL, flexible schema, scalable

4. Setup Instructions

Prerequisites:

1. **Node.js & npm –** Install from nodejs.org to run server-side code and manage packages.
2. **MongoDB –** Use local MongoDB or MongoDB Atlas to store application data.
3. **Express.js –** Install with npm install express to handle backend APIs and routing.
4. **React.js –** A frontend library to build the user interface and create single-page applications.
5. **Vite Setup –** Run: npm create vite@latest, cd my-app, npm install, npm run dev to start the frontend.
6. Use VS Code for development, API testing, and version control.

Installation Steps:

- Clone the repository

git clone https://github.com/Jammisetty-Pragna/Shopsmart/tree/main

- Set up environment variables:

Create .env file in the server directory

PORT=5000

MONGO_URI='mongodb://localhost:127.0.0.1:27017/grocery'

JWT_SECRET=mysecretpassword

- Install dependencies :
  cd Backend
  npm run dev
  cd Frontend
  npm start

Folder Structure

Shopsmart/

```
├── Backend/              # Express + MongoDB backend
|   ├── db/               # Database connection logic
|   ├── index.js          # Server entry point
|   ├── package.json
|   └── ...
├── Frontend/             # React frontend
|   ├── public/
|   ├── src/
|   |   ├── admin_components/ # Admin dashboard components
|   |   ├── components/      # Reusable UI components
|   |   ├── context/         # Context API for global state
|   |   ├── App.js
|   |   ├── index.js
|   |   ├── App.css, index.css
|   └── package.json
├── .gitignore
```

6. Running the Application

To start both servers:

cd Frontend:

Navigate to the client directory:

cd frontend

Start the development server:

npm start

The frontend application runs on: http://localhost:5100

cd Backend:

Navigate to the server directory:

cd backend

Start the backend server:

npm run dev

The backend application runs on: http://localhost:5100

7. API Documentation

Base URL:

http://localhost:5000/api

Auth Endpoints:

1. **POST /auth/register** – Register new user (name, email, password)
2. **POST /auth/login** – Login and receive JWT token
3. **GET /user/profile** – Get current user profile *(auth required)*
4. **GET /products** – Fetch all products
5. **GET /products/:id** – Fetch product by ID
6. **POST /products** – Add new product *(admin/seller only)*
7. **PUT /products/:id** – Update product *(admin only)*
8. **DELETE /products/:id** – Delete product *(admin only)*
9. **POST /cart** – Add product to cart *(auth required)*
10. **GET /cart** – View current user's cart
11. **DELETE /cart/:productId** – Remove item from cart
12. **POST /orders** – Place a new order *(auth required)*
13. **GET /orders** – View user's order history
14. **GET /admin/users** – Admin fetch all users
15. **GET /categories** / **POST /categories** – View or add product categories *(admin)*

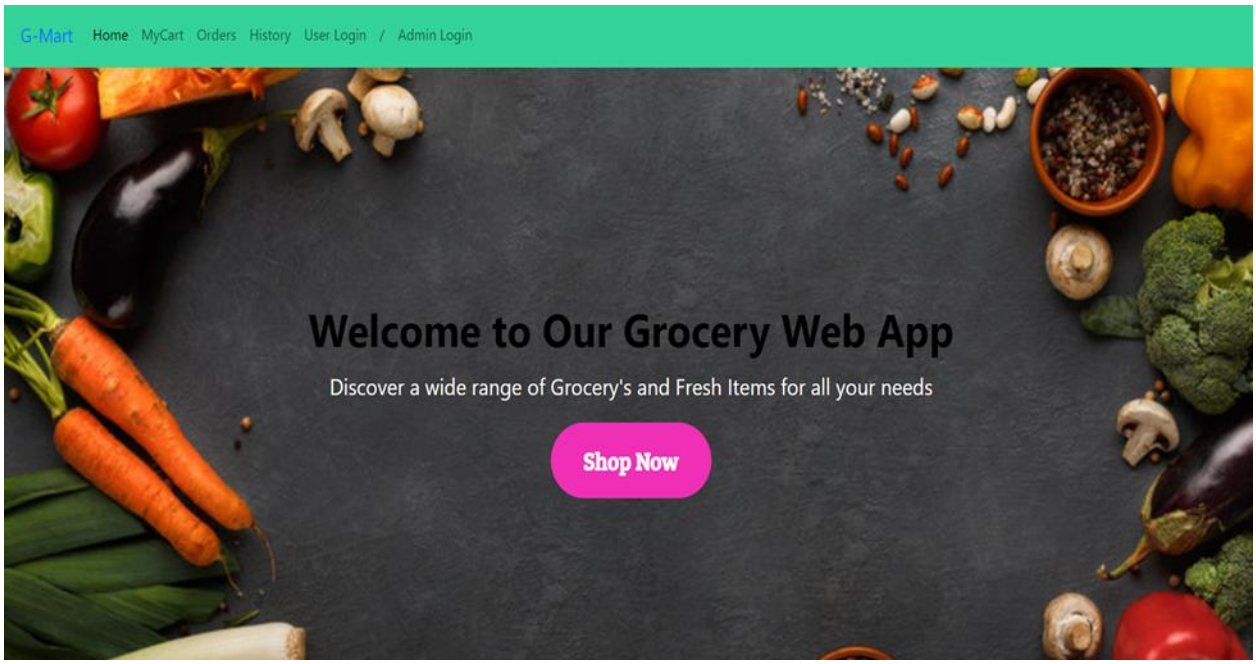
8. Authentication

- POST /auth/register – Register a new user with name, email, and password.
- POST /auth/login – Log in and receive a JWT token for session management.
- GET /user/profile – Fetch logged-in user's profile using the token.
- Passwords are securely hashed using bcrypt before storing in the database.
- JWT tokens are generated on login and must be sent in headers:

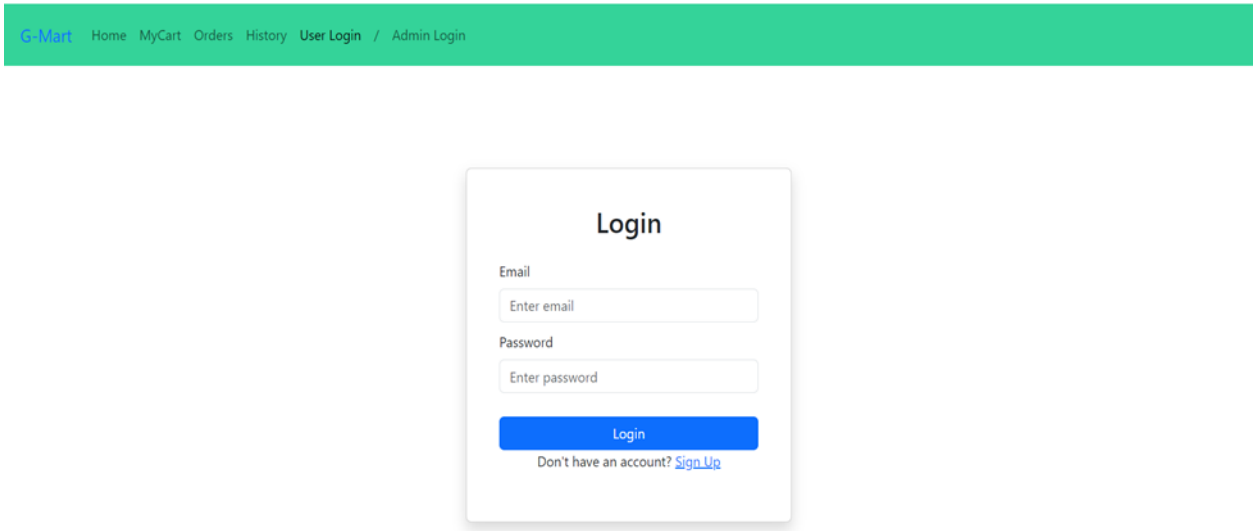
- Middleware is used to protect private routes by verifying the token.
- Only authenticated users can access cart, orders, and profile routes.

9. Screenshots or Demo

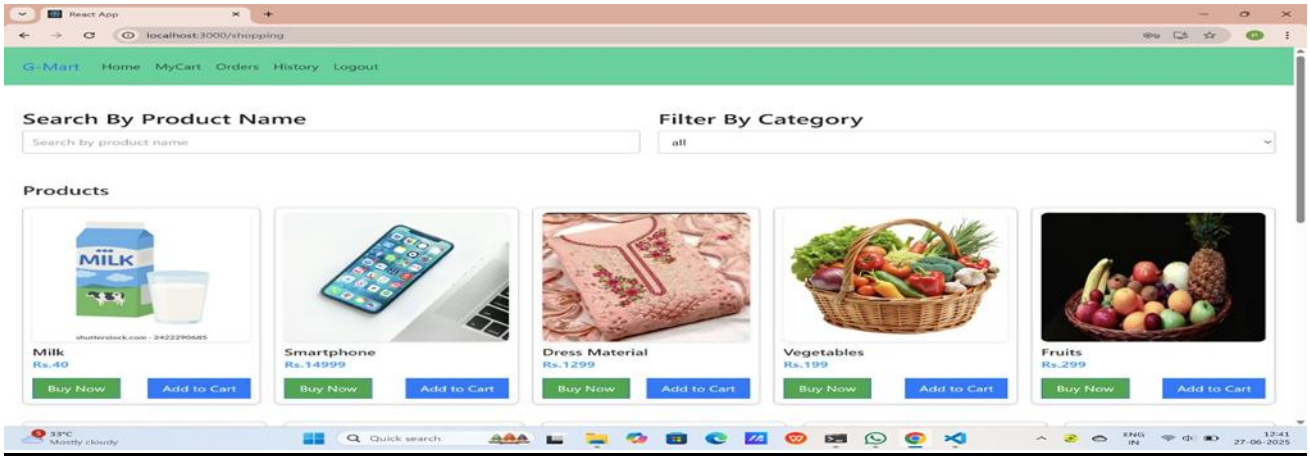
HOME PAGE :



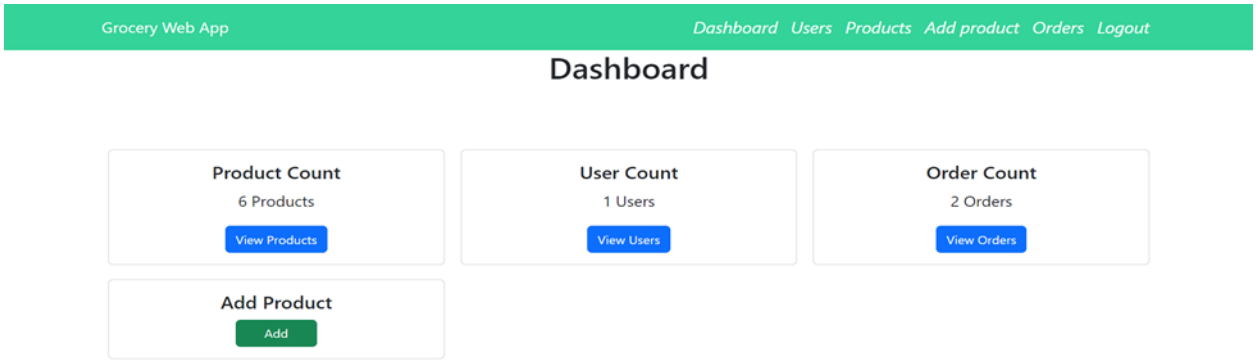
LOGIN PAGE:



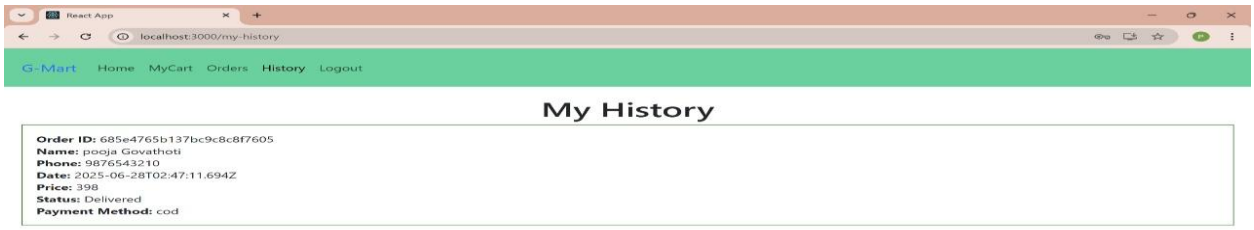
ITEMS PAGE :



ADMIN DASHBOARD PAGE:



HISTORY:



10. Testing

Testing Strategy:

- **Frontend Testing** is done using **Jest** and **React Testing Library** to validate component rendering, UI behavior, and state management.

- **Backend API Testing** uses **Mocha** and **Chai** to verify route responses, status codes, and error handling logic.
- **End-to-End (E2E) Testing** is conducted with **Cypress**, simulating complete user workflows like login, adding to cart, and checkout.
- Tests are written to cover both **positive and negative scenarios**, ensuring reliability.
- Continuous testing ensures new features don't break existing functionality.

11. Known Issues

- No Email Verification – New user accounts are created without email validation.
- Limited Payment Integration – Only basic payment methods (e.g., COD) are supported; no real-time gateway.
- Admin Panel Access Control – Role-based UI restrictions are basic and can be bypassed if tokens are misused.
- Mobile Responsiveness – Some UI components may not display correctly on smaller screens.
- No Image Uploads – Product images are static or missing due to lack of media upload support.

12. Future Enhancements

- Real-Time Updates – Implement WebSocket support for live cart, order, and appointment status updates.
- Multi-Format File Uploads – Extend document upload to support images (JPG, PNG) and DOCX files.
- Advanced Payment Integration – Integrate secure payment gateways like Razorpay, Stripe, or PayPal.
- Push Notifications – Notify users about order updates, offers, and appointment status via browser/mobile alerts.
- AI-Based Recommendations – Use ML to suggest products based on browsing and purchase history.
- Role-Based Dashboards – Provide customized dashboards for customers, sellers, and admins.