**ShopSmart - Your Digital Grocery Store Experience**

Team Members:

TEAM ID : LTVIP2025TMID59448

| **MEMBER** | **RESPONSIBILITY** |
| --- | --- |
| 1. Bhukya Pavan Nayak | Testing & Documentation. |
| 2. Balaga Geya Varshini | Frontend & Backend. |
| 3. Arla Pooja | Resource Gathering & Database Connection. |
| 4. Achyutha Subramanyam | Frontend & Documentation. |

**1. Introduction**

Project Title:  
ShopSmart - Your Digital Grocery Store Experience

**2. Project Overview**

***Purpose:***  
The purpose of our grocery-web app (ShopSmart) is to offer a seamless, secure, and user-friendly online shopping platform that caters to diverse customer needs. It simplifies the buying process, supports efficient seller and admin management, and prioritizes data privacy and satisfaction. With intuitive design and robust backend support, it empowers both users and vendors. Our goal is to enhance the online grocery shopping experience for everyone.

***Features:***

* Intuitive and responsive user interface for smooth navigation
* Categorized product browsing with detailed item descriptions
* Easy cart management and secure checkout process
* Seller dashboard to manage products, inventory, and orders
* Admin panel for overseeing operations and customer support
* Seamless payment gateway integration
* Scalable architecture for future feature expansion
* Real-time order tracking and notifications

**3. Architecture**

***Frontend:***  
Built using React.js, the frontend follows a component-based architecture:

* Component Structure: Functional components using hooks like useState, useEffect, and useContext
* Routing: Handled with **React Router v6** for dynamic and secure navigation across user, seller, and admin interfaces
* State Management: Context API (or Redux for scalability) used for managing global state (user authentication, cart, order flow)
* UI Library: Material-UI (MUI) ensures a responsive, modern design across all devices
* HTTP Requests: Managed using Axios for clean and consistent API interactions with the backend

***Backend:***  
The backend is built using Node.js and Express.js :

* **API Structure:** RESTful endpoints divided by functionality (users, products, orders, authentication)
* **Middleware:** Includes cors, dotenv, morgan, express-validator for logging, environment config, security, and input validation
* **Authentication:** JWT -based midleware secures protected routes for users, sellers, and admins
* **Modular Organization:** Follows a clear separation of **Routes**, **Controllers**, **Services**, and **Models** for maintainability and scalability

***Database:***

MongoDB (NoSQL) is used with Mongoose ORM for flexible data handling.

**4. Setup Instructions**

***Prerequisites:***

Ensure you have the following installed:

Node.js (v16.x or higher)

npm or yarn

MongoDB Atlas account or local MongoDB instance

Git

***Installation Steps:***

* Clone the repository

git clone <https://github.com/varsshi-12/ShopSmart-Web-App>

cd ShopSmart

* Set up environment variables:

Create .env file in the server directory

PORT=5000

MONGO\_URI=mongodb://127.0.0.1:27017/grocery

JWT\_SECRET=mysecretpassword

* Install dependencies :

Backend

cd Backend

node index.js

#Frontend

cd ../Frontend

npm install

npm start

**Folder Structure**

ShopSmart/

│

├── Backend/

│ ├── db/

│ │ ├── connect.js # MongoDB connection logic

│ │ ├── products.js # Product-related logic (routes/controllers)

│ │ └── schema.js # Mongoose schema definitions

│ ├── node\_modules/ # Backend dependencies

│ ├── index.js # Entry point of backend (Express server)

│ ├── package.json # Backend package configuration

│ └── package-lock.json # Backend lock file

│

├── Frontend/

│ ├── node\_modules/ # Frontend dependencies

│ ├── public/ # Static files (e.g., index.html)

│ ├── src/

│ │ ├── admin\_components/ # Components for admin functionality

│ │ ├── components/ # Reusable components

│ │ ├── context/ # Context API (global state management)

│ │ ├── App.js # Main app component

│ │ ├── App.test.js # Tests for App component

│ │ ├── index.js # App entry point

│ │ ├── index.css # Global styles

│ │ ├── App.css # App-specific styles

│ │ ├── logo.svg # App logo

│ │ ├── reportWebVitals.js # Performance measuring

│ │ └── setupTests.js # Testing setup (Jest)

│ ├── package.json # Frontend package configuration

│ └── package-lock.json # Frontend lock file

**6. Running the Application**

To start both servers:

***Frontend:***

Navigate to the client directory:

cd Frontend

Start the development server:

npm start

The frontend application runs on: http://localhost:3000

***Backend:***

Navigate to the server directory:

cd Backend

Start the backend server:

node index.js

The backend application runs on: http://localhost:5000

**7. API Documentation**

**Base URL:**

http://localhost:5000/api

**Auth Endpoints:**

POST /auth/register

Registers a new user (customer or seller) with name, email, password, and role.

POST /auth/login

Logs in a registered user and returns a JWT token for authentication.

**Product Endpoints**:

GET /products (Protected)

Retrieves all available products with optional filtering by category, price, or search keyword.

POST /appointments (Protected: Seller only)

Adds a new product listing with name, price, category, stock, and description.

PUT /appointments/:id/status (Protected: Seller only)

Updates an existing product's details.

DELETE /products/:id (Protected: Seller only)

Deletes a product listing by ID.

**Order Endpoints:**

GET /orders (Protected)

Retrieves all past orders of the logged-in customer.

PUT /orders/:id/status (Protected: Admin only)

Updates the order status (e.g., processing, shipped, delivered, cancelled).

**User Endpoints:**

GET /users/me (Protected)

Returns the profile information of the currently logged-in user.

GET /users/customers (Protected)

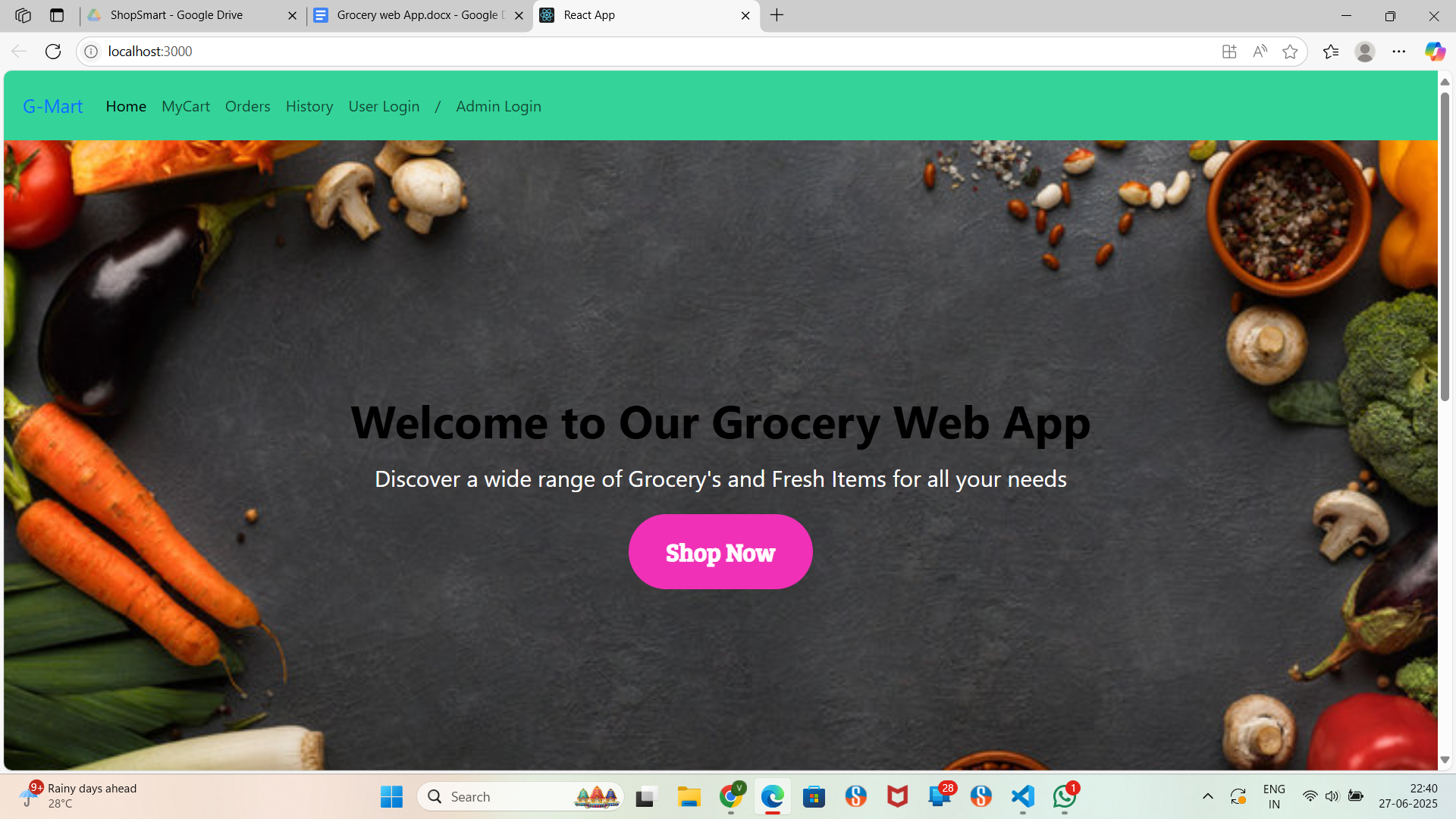
Lists all registered customers.

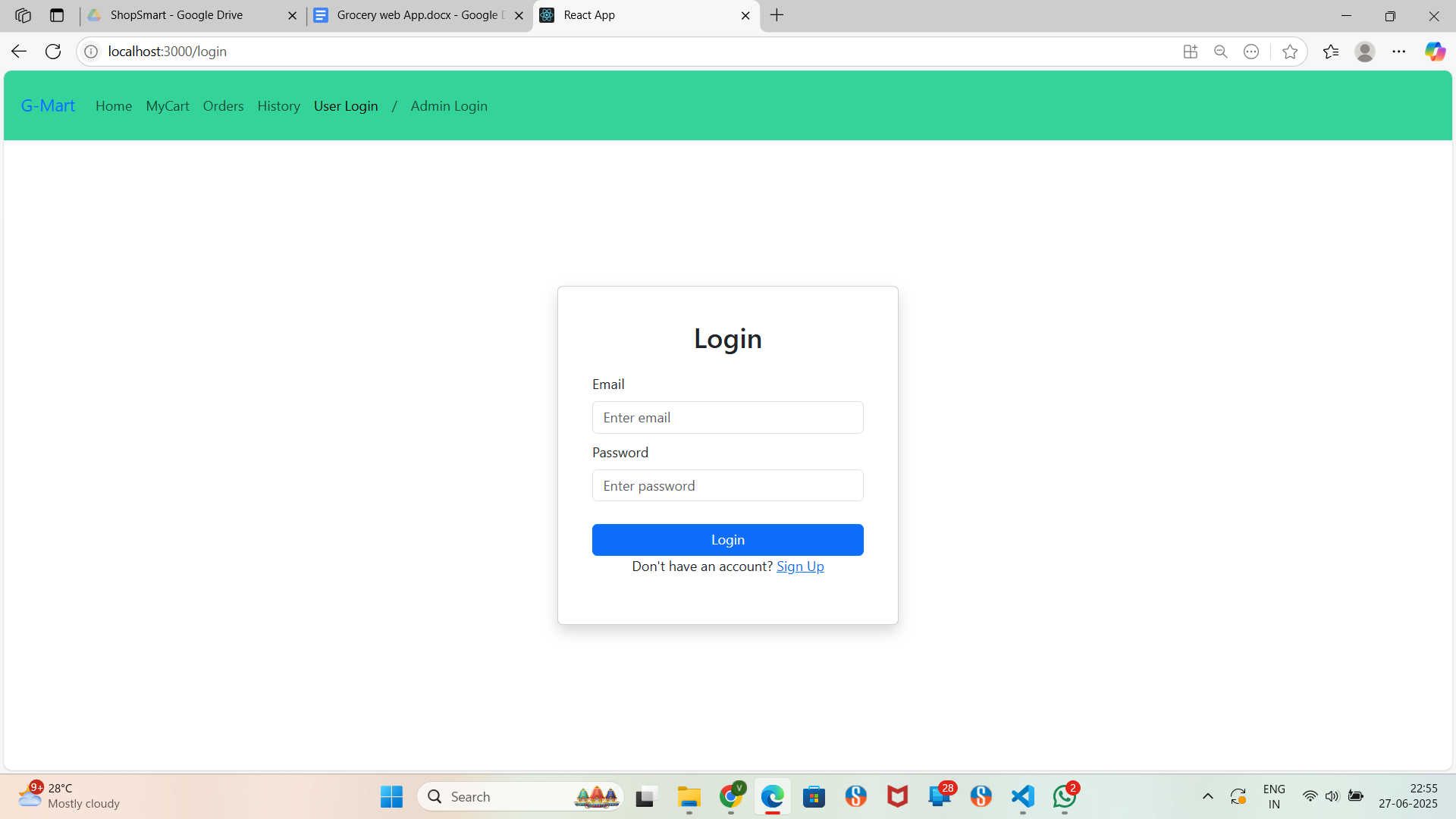
**8. Authentication**

Authentication is handled using **JWT (JSON Web Tokens)**:

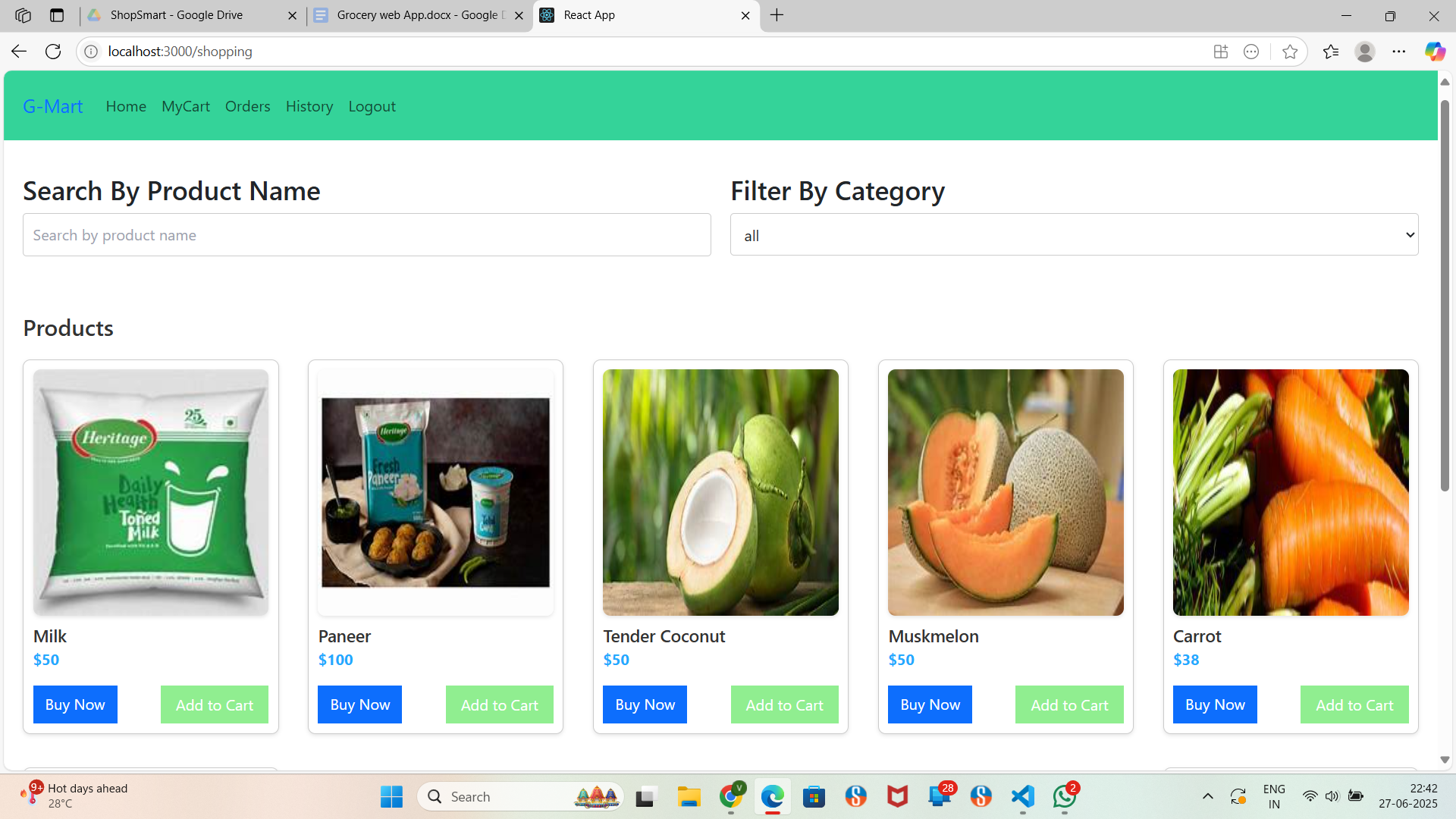
* Upon successful login or registration, the server generates a JWT token.
* This token is stored in the browser's **localStorage**
* All **protected routes** require a valid token in the Authorization header as Bearer <token>
* Middleware validates the token and authorizes access based on user roles: **customer** or **admin**.
* Unauthorized or expired tokens result in a 401 Unauthorized response.

**9. Screenshots or Demo**

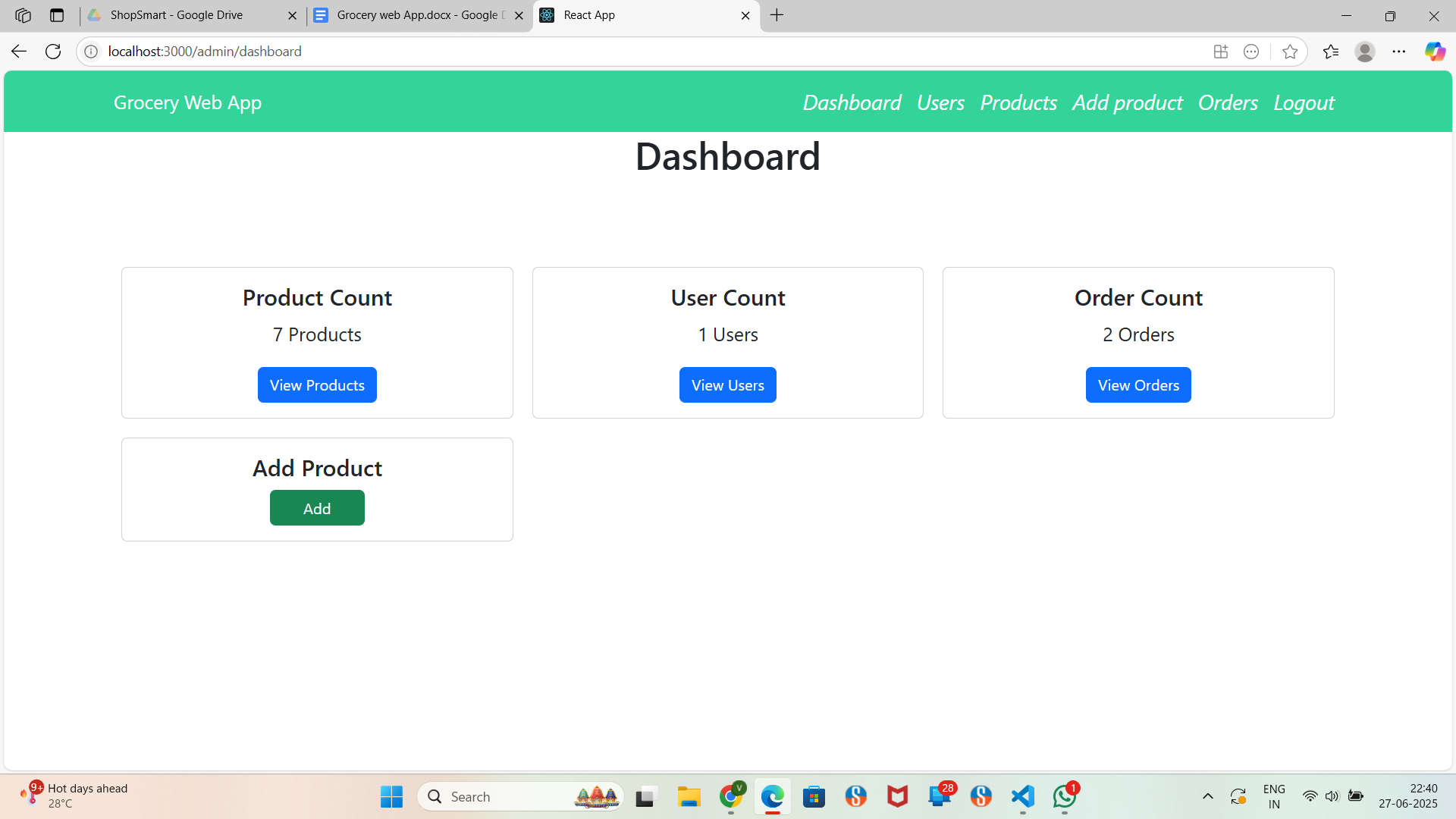
HOME PAGE :

LOGIN PAGE:

PRODUCTS PAGE :



DASHBOARD:



**10. Testing**

***Testing Strategy:***

**Frontend testing** is carried out using Jest and React Testing Library to validate component rendering, user interactions, and state management logic. UI flows are manually tested to ensure responsiveness and accessibility across devices.

**Backend testing** **Jest** and **Supertest** are used to perform unit tests on controllers and services, as well as integration tests on RESTful API endpoints to verify data flow and error handling. Combined, these ensure that both client-side and server-side functionalities work reliably and as expected.

**11. Known Issues**

Currently, the app has a few known issues, such as limited error handling in certain user flows, occasional latency in product search filtering, and lack of mobile responsiveness on older devices. These are under review and scheduled for improvement.

**12. Future Enhancements**

Planned enhancements include implementing a wishlist feature, integrating payment gateways for live transactions, adding multi-language support, enhancing analytics for sellers and admins, and developing a native mobile app version to improve accessibility and user engagement.