FULL STACK DEVELOPMENT PROJECT WITH “MERN” - DOCUMENTATION

**1. Introduction**

PROJECT TITLE:ShopSmart-Your Digital Grocery Store Experience

TEAM ID:LTVIP2025TMID56023

Team Members:

**1.Team Leader:**

VALLURI VINAY–GitHub management and integration of frontend and backend.

**2.Team Member**:

V TEJASRI– Frontend Developer, Backend Developer

3.**Team Member**:

V S N V PUSHPANJALI– Full Stack Developer & Project Coordinator.

2. Project Overview

**Purpose:**

ShopSmart is designed to digitize the grocery shopping experience, allowing users to browse products, add items to a cart, place orders, and track them in real-time—all from the comfort of their device.

Features:

User registration and login

* Product browsing and search
* Add to cart and checkout functionality
* Order tracking
* Admin panel to manage inventory and orders
* Responsive user interface

3. Architecture

Frontend:

Built using React.js, the frontend features a responsive design with reusable components for product listings, cart management, and order status. It uses React Router for page navigation and Axios for API calls.

Backend:

Developed using Node.js and Express.js, the backend handles user authentication, product and order management, and serves RESTful APIs for frontend integration.

Database:

MongoDB is used to store user information, product data, orders, and cart details. Mongoose is used for schema modeling and interaction with the database.

4. Setup Instructions

Prerequisites:

* Node.js
* MongoDB
* Express.js
* Angular
* React js

Installation:

1. Clone the repository:

git clone https://github.com/your-username/shopsmart.git

2. Install dependencies:

For frontend:

cd client

npm install

For backend:

cd server

npm install

3. Set up environment variables:

Create a .env file in the server folder with MongoDB URI and JWT secret

5. Folder Structure

Client:

├── /src

│ ├── /components

│ ├── /pages

│ ├── /api

│ ├── App.js

│ └── index.js

Server:

/server

├── /controllers

├── /routes

├── /models

├── server.js

└── .env

6. Running the Application

Frontend:

* cd client
* npm start

Backend:

* cd server
* npm start

7. API Documentation

Users

* POST /api/register – Register a new user
* POST /api/login – Login and receive JWT token

Products

* GET /api/products – Get all products
* GET /api/products/:id – Get product by ID
* POST /api/products – Add a new product (admin only)

Orders

* POST /api/orders – Place an order
* GET /api/orders/:userId – Get orders by user

8. Authentication

* JWT-based authentication is implemented.
* Users receive a token on successful login, which is stored in localStorage.
* Protected routes use middleware to verify the token and authorize access.
* Admin users have elevated privileges to manage products and orders.

9. User Interface

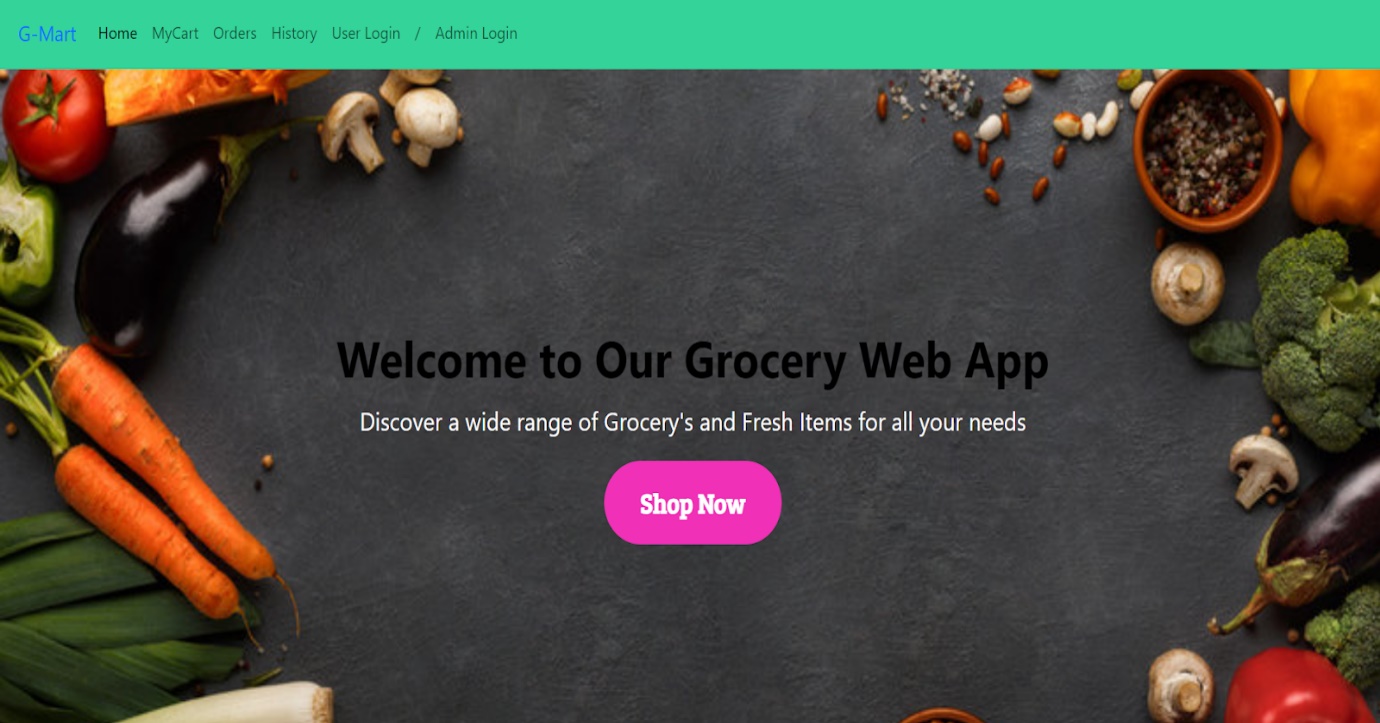
* Key UI elements include:
* Product grid view
* Search and filter bar
* Shopping cart preview
* Checkout and order summary

10. Testing

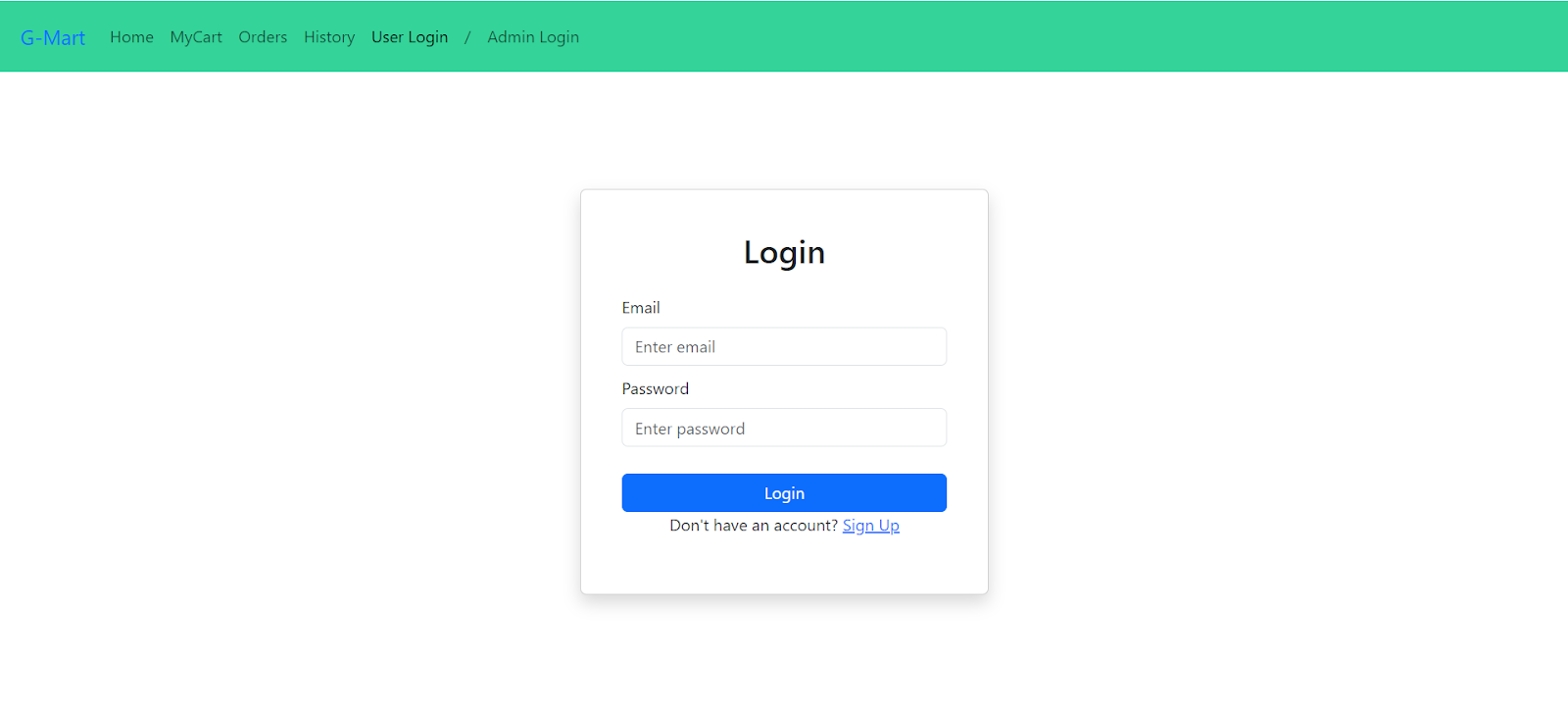
* Manual testing for all functionalities including login, cart, and order management.
* Postman used for backend API testing.
* Unit tests written using Jest and React Testing Library (optional depending on scope).

11. Screenshots or Demo

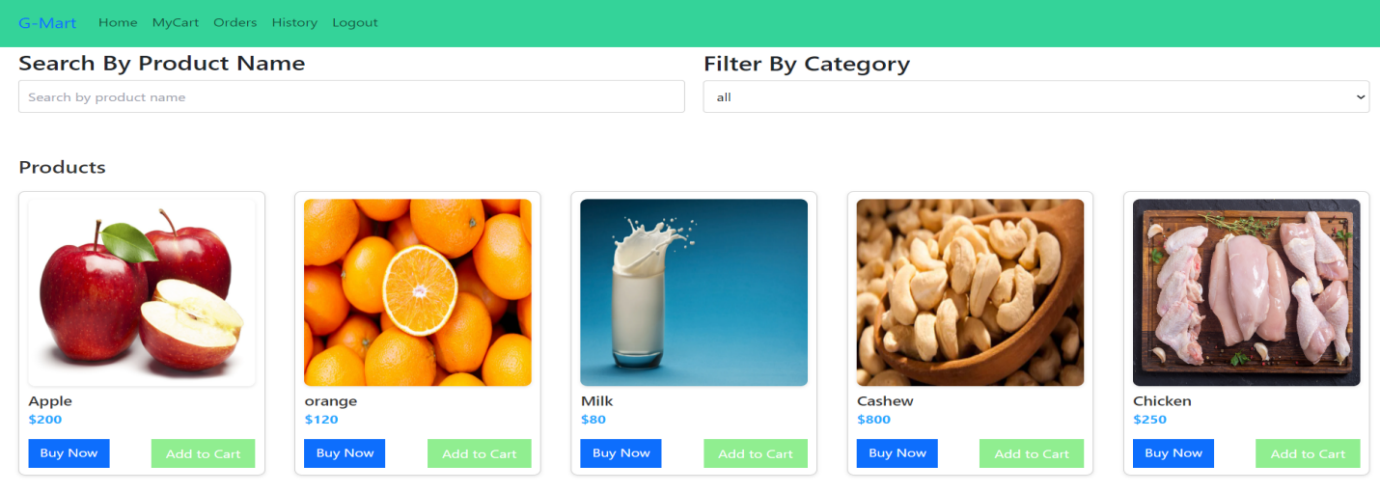
Landing page:-



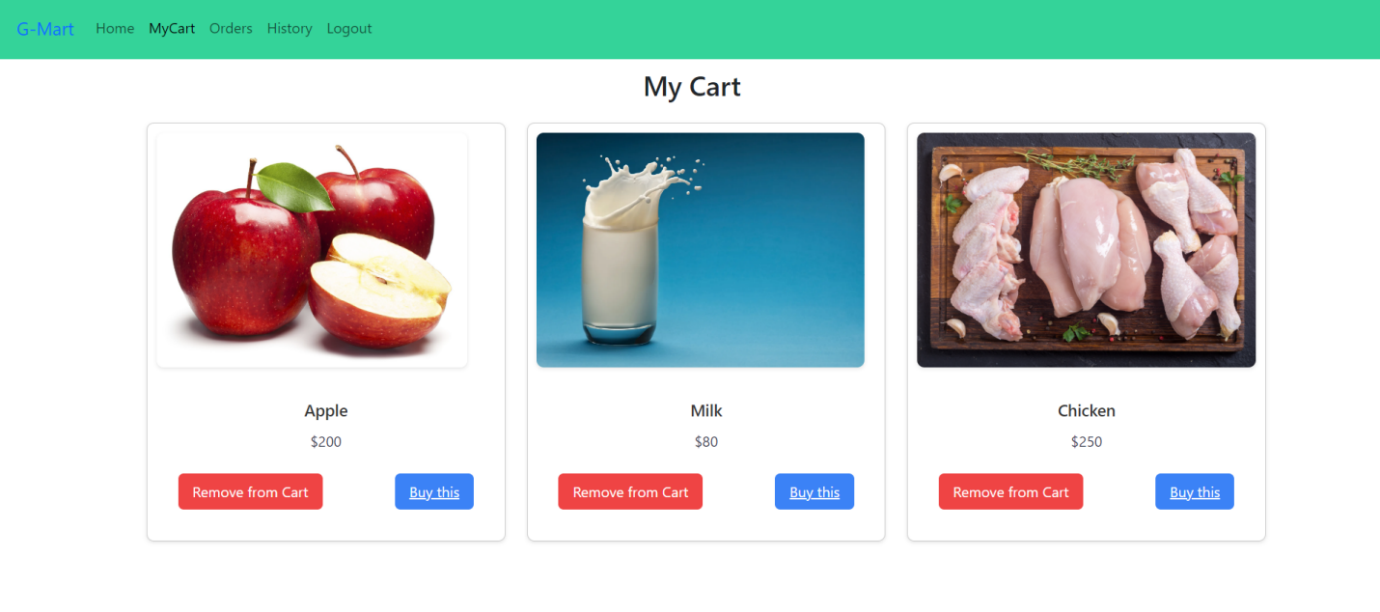
Login Page:-



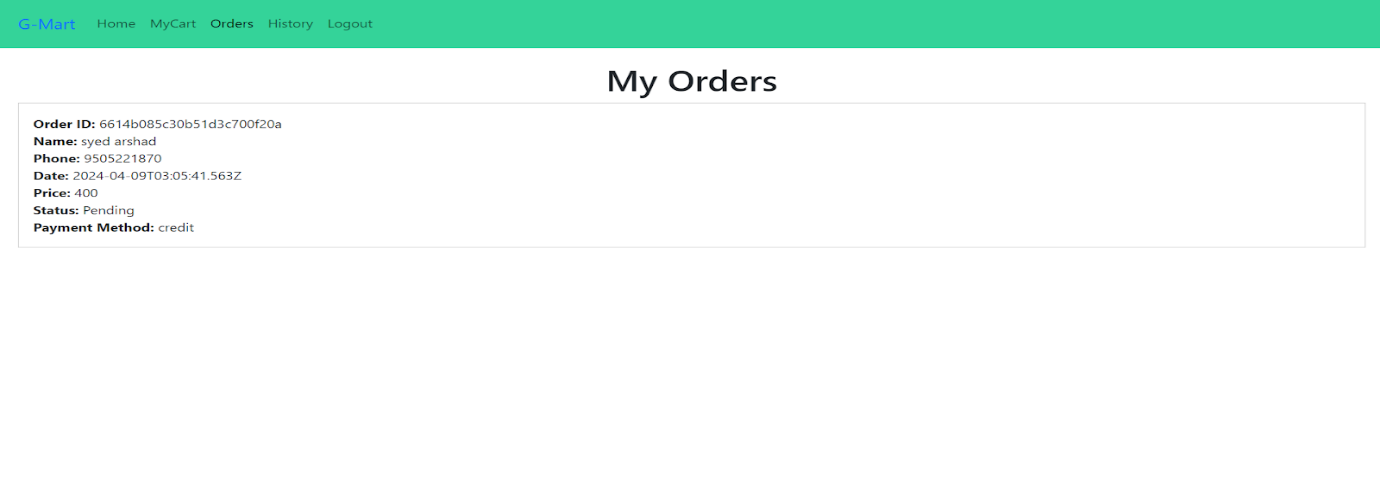
Items Page:-



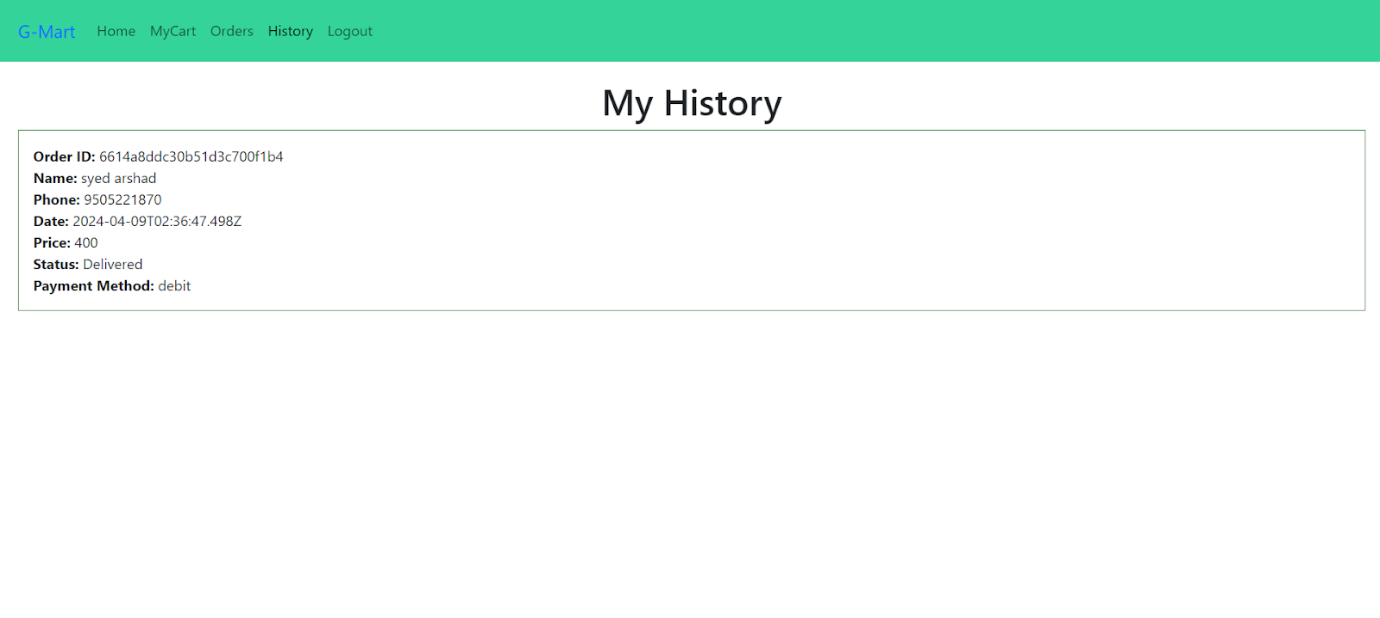
My Cart:-

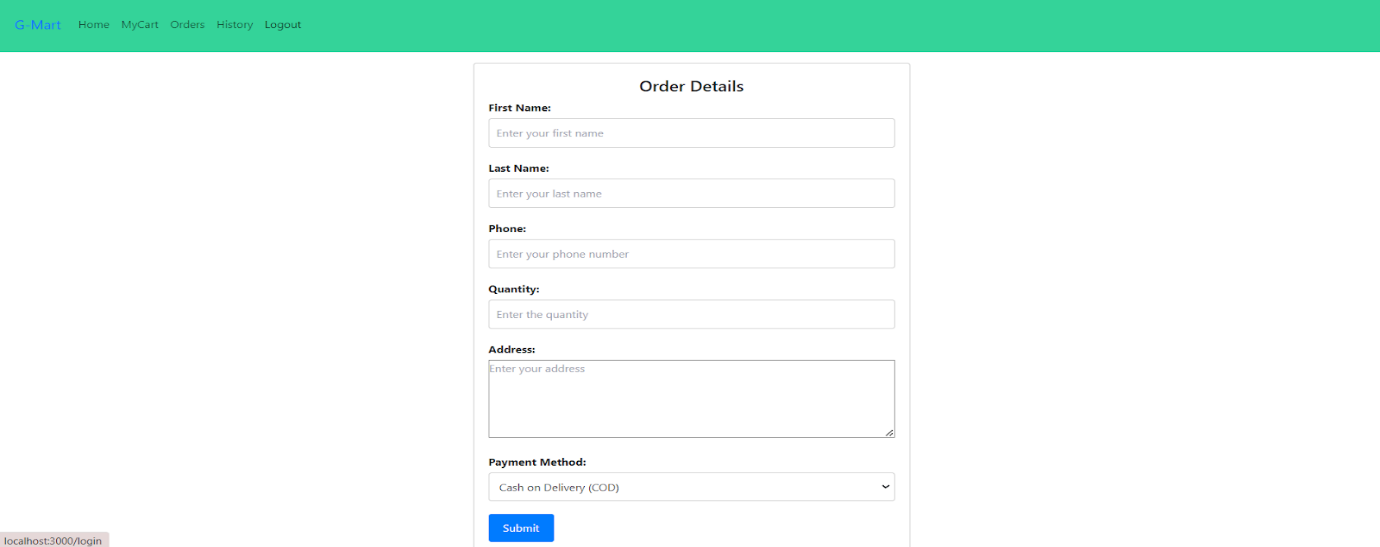


My Orders Page:-

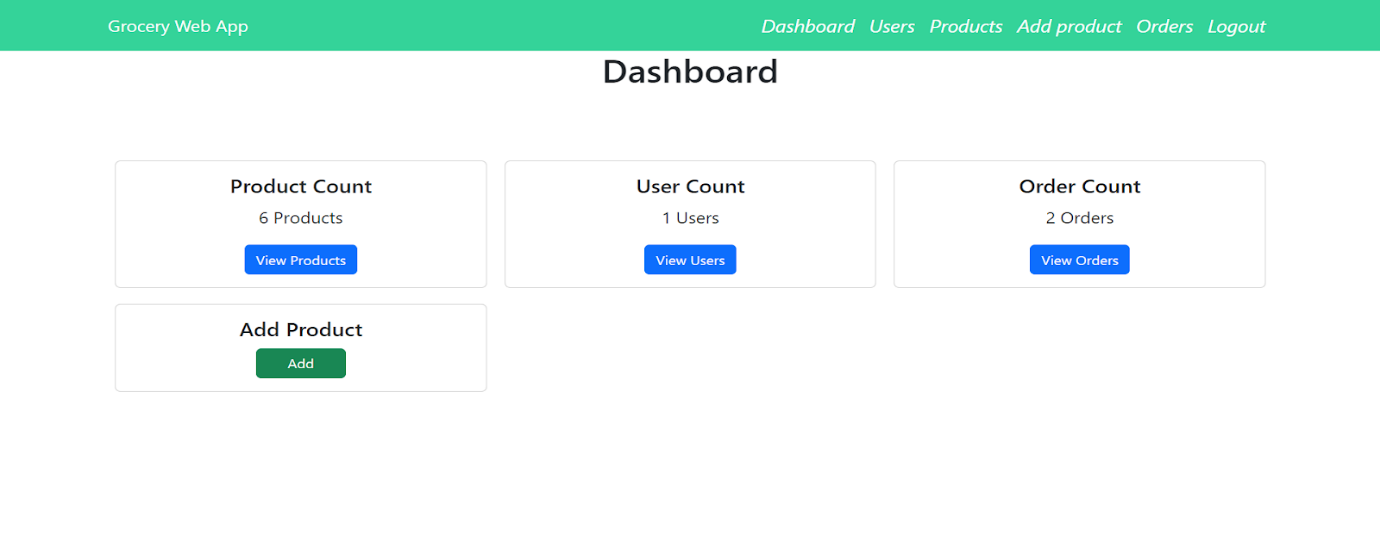


My History Page:-

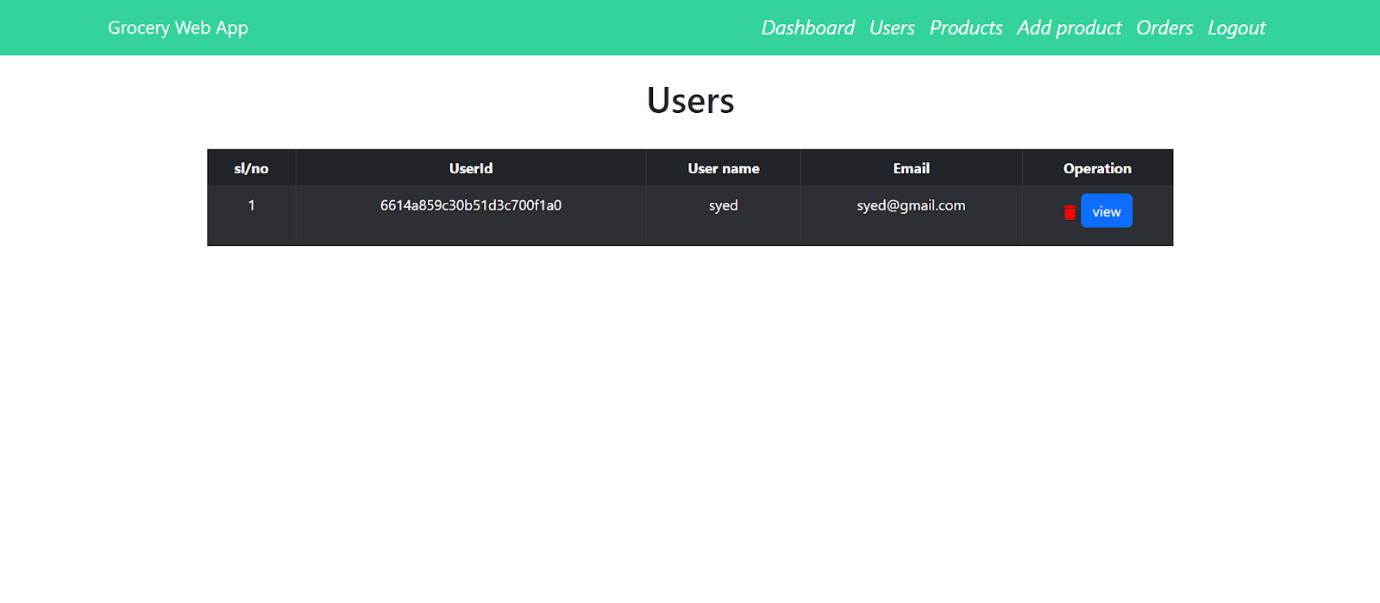
Place Order Page:-



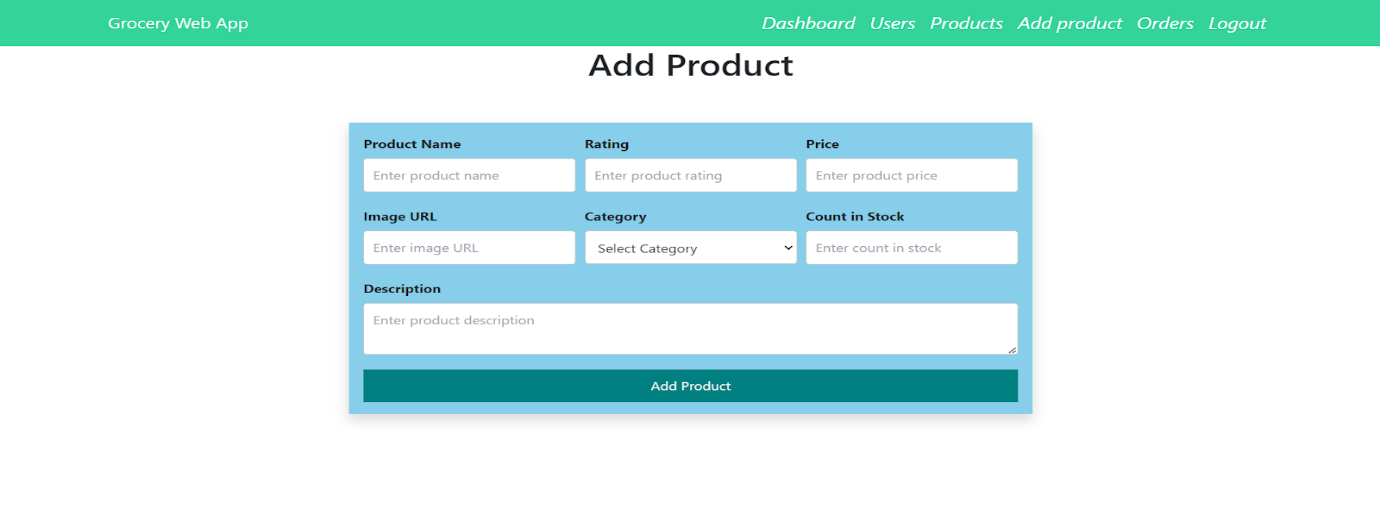
Admin Dashboard Page:



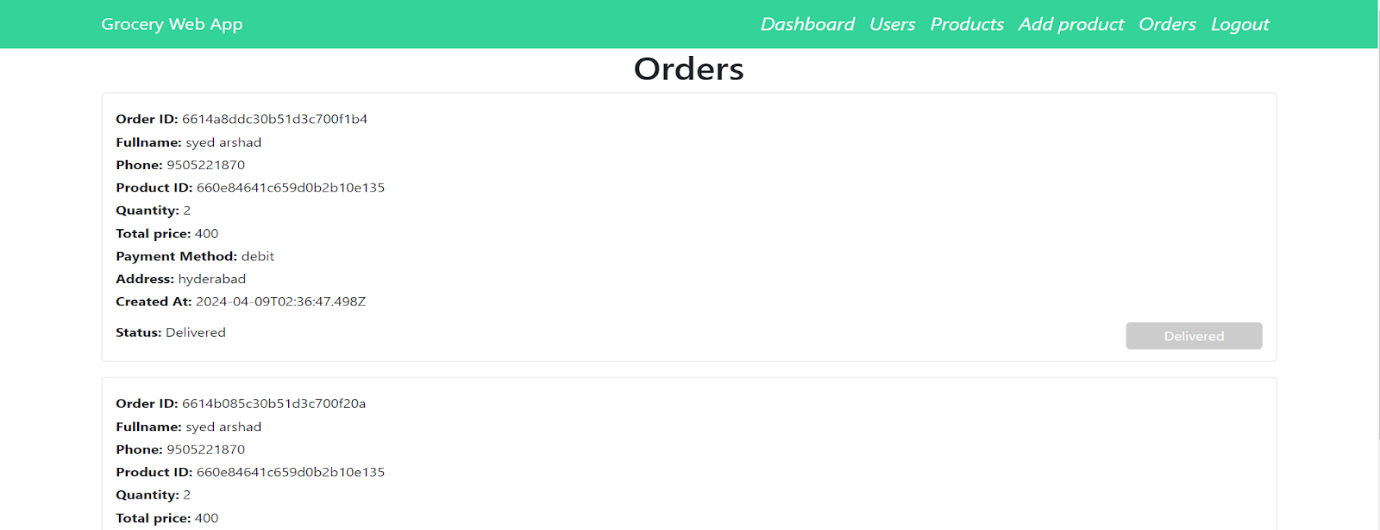
Users Page:-



Add Product page:-



Admin Orders Page:-



LINK:

<https://drive.google.com/file/d/1HLowcIqs2d8lxTprS2jqPmR4AOnUW8xD/view?usp=drive_link>

12. Known Issues

* Image upload functionality not yet implemented
* Search filtering may not be optimized for large datasets
* No email notifications for order updates

13. Future Enhancements

* Implement product image uploads using Cloudinary or Firebase
* XAdd product reviews and ratings