Full Stack Development with MERN
Grocery webapp

1. INTRODUCTION:

- **Project Title:** Grocery Webapp
- **Team Members:**
  - ➢ Santhoshram C - Frontend Developer
  - ➢ Madhan kumar M - Backend Developer
  - ➢ Naveen S - UI/UX Designer
  - ➢ Lokesh D – Frontend Developer

2. PROJECT OVERVIEW:

- **Purpose:**

  The Grocery Web App is a convenient online shopping platform, allowing users to browse, select, and purchase groceries and household essentials from the comfort of their home. Designed with user-friendly navigation and secure checkout, the app provides customers with a seamless experience, sellers with efficient product management, and administrators with robust control over inquiries and transactions.

- **Features:**
  - ➢ **User Account Management:**
    - ❖ Registration, login, and profile management for customers and sellers.

- **Product Browsing:**
  - Organized product categories with detailed views, search, and filtering capabilities.

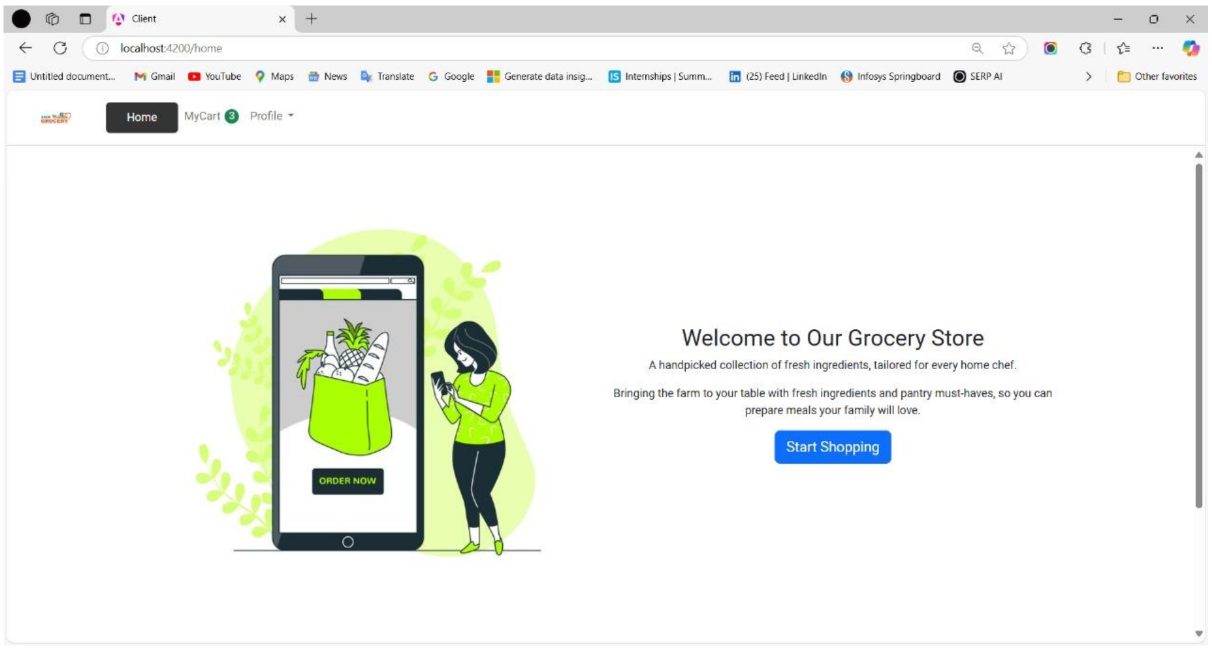
- **Shopping Cart and Checkout:**
  - Add items to the cart, view totals, and complete purchases securely.

- **Order Tracking:**
  - Customers can track order statuses.

- **Inventory Management:**
  - Sellers can manage product listings, track stock, and update availability.

- **Admin Dashboard:**
  - Administrators have access to manage users, oversee transactions, and address customer inquiries.

- **Security:**
  - Encrypted data, secure payments, and privacy protections.



Welcome to Our Grocery Store

A handpicked collection of fresh ingredients, tailored for every home chef.

Bringing the farm to your table with fresh ingredients and pantry must-haves, so you can prepare meals your family will love.

Home Page

3. ARCHITECTURE:

- **Frontend:**

  - ❖ The frontend is built with React.js, utilizing component-based architecture for scalability and reusability. Key pages include Home, Product Listing, Product Details, Cart, Checkout, and User Dashboard. State management is handled with Redux to centralize application state and ensure data consistency.

- **Backend:**

  - ❖ Node.js and Express.js power the backend, handling RESTful API requests and serving data to the frontend. Middleware is used for request logging, authentication, and error handling. Secure API routes separate functionalities for customers, sellers, and administrators.

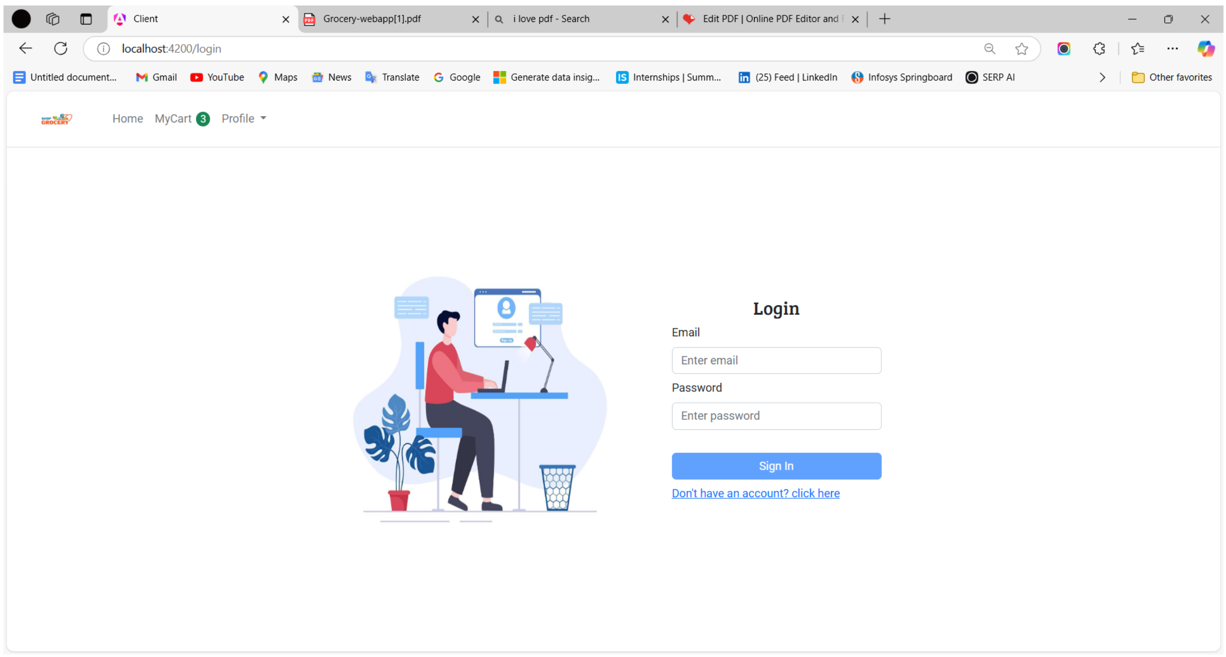
- **Database:**
  - ❖ MongoDB serves as the database, structured into collections such as Users, Products, Orders, and Messages. Each collection is designed for optimal data retrieval, with necessary indexing for common queries. Relationships are established by referencing IDs across collections (e.g., orders reference product and user IDs).

4. SETUP INSTRUCTIONS

- **Prerequisites:**

  - Node.js (v14+)

  - MongoDB (local instance or Atlas account)



Sign up page of Webapp

- **Installation:**

1. Clone the repository: git clone [repository_url]
2. Navigate to the project folder: cd grocery-webapp
3. Install dependencies for both frontend and backend:

```
bash
cd client
npm install
cd ../server
npm install
```

4. Configure environment variables: Create a .env file in the server directory, setting values for:
   - MONGO_URI: MongoDB connection string
   - JWT_SECRET: Secret key for JWT authentication
   - PORT: Server port number (optional)
   - STRIPE_SECRET_KEY: Key for payment processing (if using Stripe)

5. FOLDER STRUCTURE

- **Client:**

  - src/components: Reusable UI components (e.g., Navbar, Footer, ProductCard)

- src/pages: Main pages like Home, Products, Cart, and User Dashboard

- src/redux: State management files for Redux (actions, reducers, and store setup)

- src/styles: CSS files for custom styling

- **Server:**

  - routes: Defines API endpoints for different modules (e.g., auth…Project Title: Grocery Web App

6. RUNNING THE APPLICATION

- **Frontend:**
  - ❖ Run npm start in the client directory to start the React development server on localhost:3000.
- **Backend:**
  - ❖ Run npm start in the server directory to start the Node.js server on localhost:5000 (or the configured port).

- **The below will connect the client to the server:**

```
Const mongoose = require("mongoose");
// Middleware
// const db =
'mongodb+srv://madhanmurali0710:Oct07@3662292@mern.rtuy9.mongodb.net/Gr
oceryApp?retryWrites=true&w=majority&appName=MERN'

 const
db="mongodb+srv://test:test@mern.rtuy9.mongodb.net/GroceryApp?retryWrites=tr
ue&w=majority&appName=MERN"

// Connect to MongoDB using the connection string

mongoose.connect(db, {

  useNewUrlParser: true,

  useUnifiedTopology: true,

}).then(() => {

  console.log(`Connection successful`);

}).catch((e) => {

  console.log(`No connection: ${e}`);

});


// mongodb://localhost:27017
```

- npm start
- npm run dev

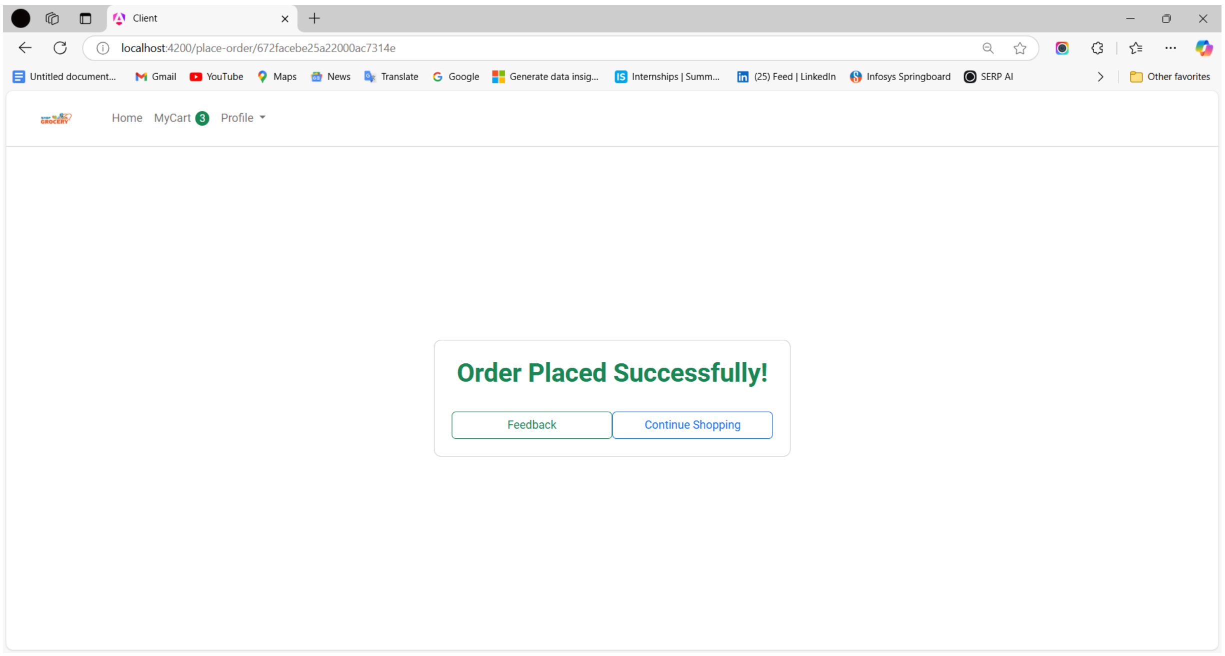
  The above code will connect the server to the client.

7. API DOCUMENTATION

-Endpoints:

- /api/auth/register - POST: Registers a new user. Parameters: { username, email, password }
- /api/auth/login - POST: Logs in a user. Parameters: { email, password }
- /api/products - GET: Retrieves a list of products.

- /api/cart - POST: Adds item to the cart. Parameters: { userId, productId, quantity }
- /api/orders - POST: Places an order. Parameters: { userId, cartItems, paymentInfo }
- /api/admin/users- GET: Admin endpoint to retrieve all users.



Order Placed Successfully!

When customer place order

8. AUTHENTICATION

- Method: JWT (JSON Web Tokens) is used to authenticate users. Upon login, users receive a JWT token stored in local storage for secure session management. Routes requiring authentication validate the token to ensure authorized access.

- Roles: Role-based access control is implemented, ensuring that users, sellers, and administrators have appropriate permissions for their roles.

9. User Interface

- Screenshots/GIFs:

- Home Page: Displaying categories and featured products.

- Product Page: Showing product details, pricing, and reviews.

-Cart and Checkout: Overview of selected items and checkout process.

- Admin Dashboard: Management interface for product and order overview.

10. Testing

- Testing Strategy:

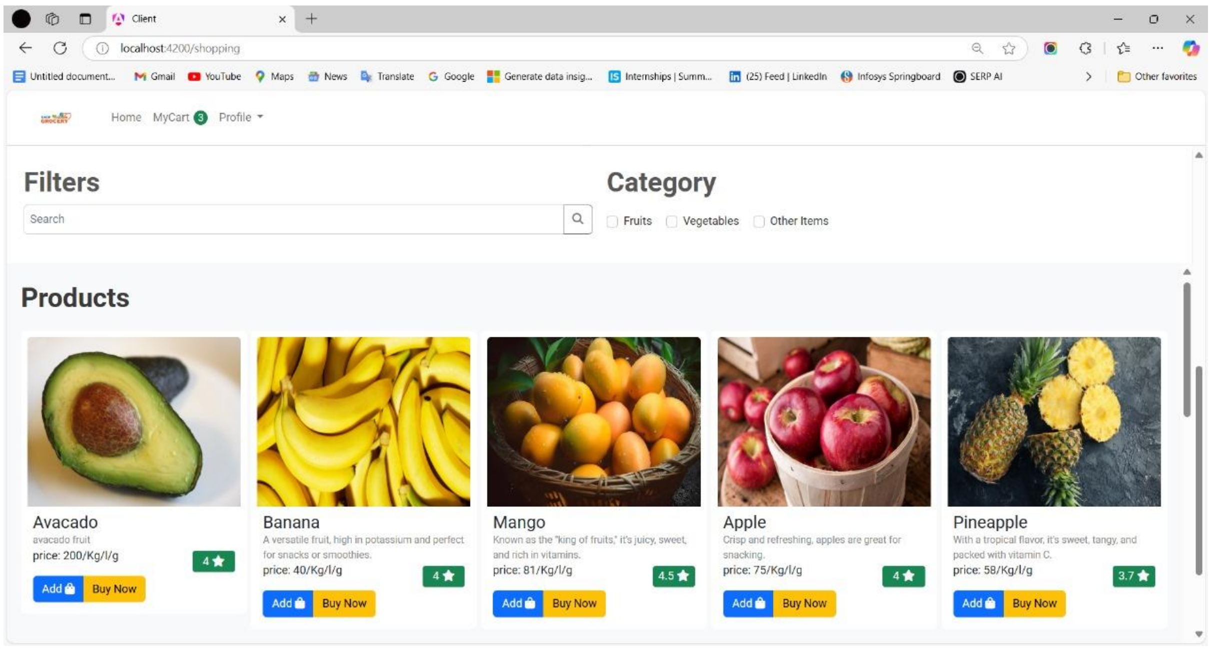
- Unit Testing: Using Jest for frontend components and backend endpoints.

- Integration Testing: API tests to verify functionality across components.

- End-to-End Testing: Simulating user interactions using Cypress.

11. Screenshots or Demo

https://drive.google.com/file/d/1BGFFRzYDsAtIUcytpZOFbtCBTMSx4Bfg/view?usp=sharing



Avacado
avacado fruit
price: 200/Kg/l/g
4
Add
Buy Now

Banana
A versatile fruit, high in potassium and perfect for snacks or smoothies.
price: 40/Kg/l/g
4
Add
Buy Now

Mango
Known as the "king of fruits," it's juicy, sweet, and rich in vitamins.
price: 81/Kg/l/g
4.5
Add
Buy Now

Apple
Crisp and refreshing, apples are great for snacking.
price: 75/Kg/l/g
4
Add
Buy Now

Pineapple
With a tropical flavor, it's sweet, tangy, and packed with vitamin C.
price: 58/Kg/l/g
3.7
Add
Buy Now

This is the catalogue for our Grocery webapp

12. Known Issues

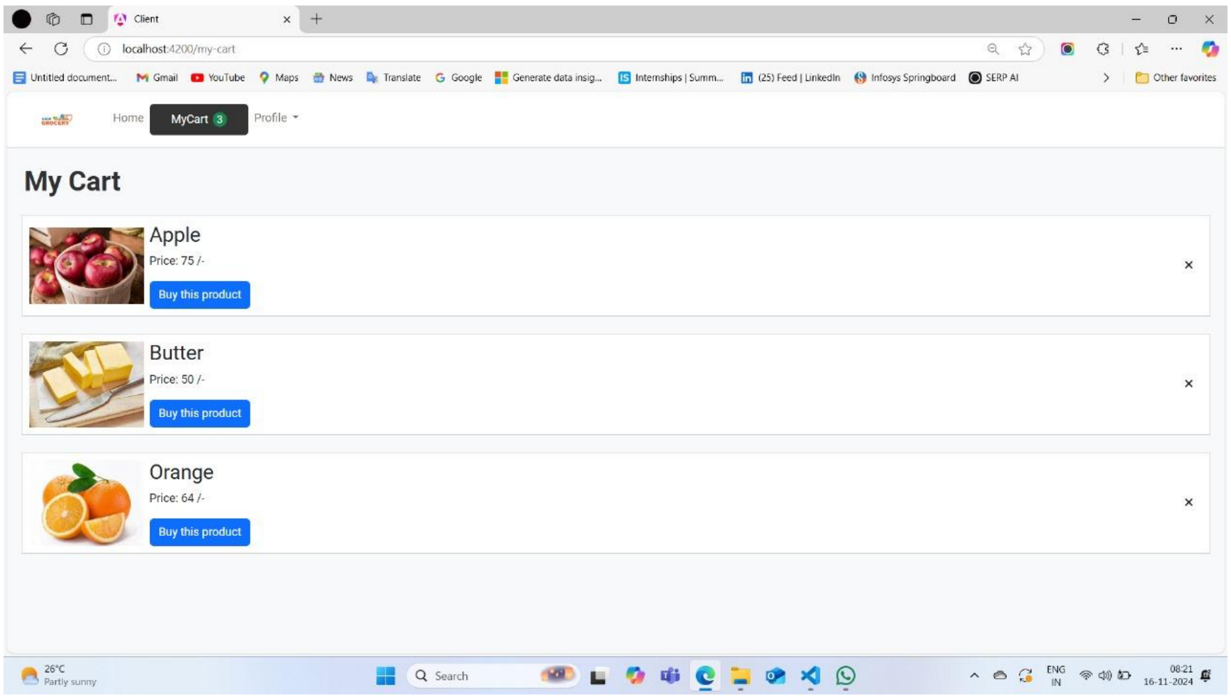
- Occasional delay in updating product inventory upon high-traffic periods.

- Some mobile UI components may not scale perfectly on smaller screens.

13. Future Enhancements

- Push Notifications: Notify users of discounts or new arrivals.
- Analytics Dashboard for Admins:Track sales and user activity.
- Multiple Payment Methods: Expand options to include various payment gateways.
- Wishlist Functionality: Allow users to save favorite products for later.

14. Placing orders



My Cart

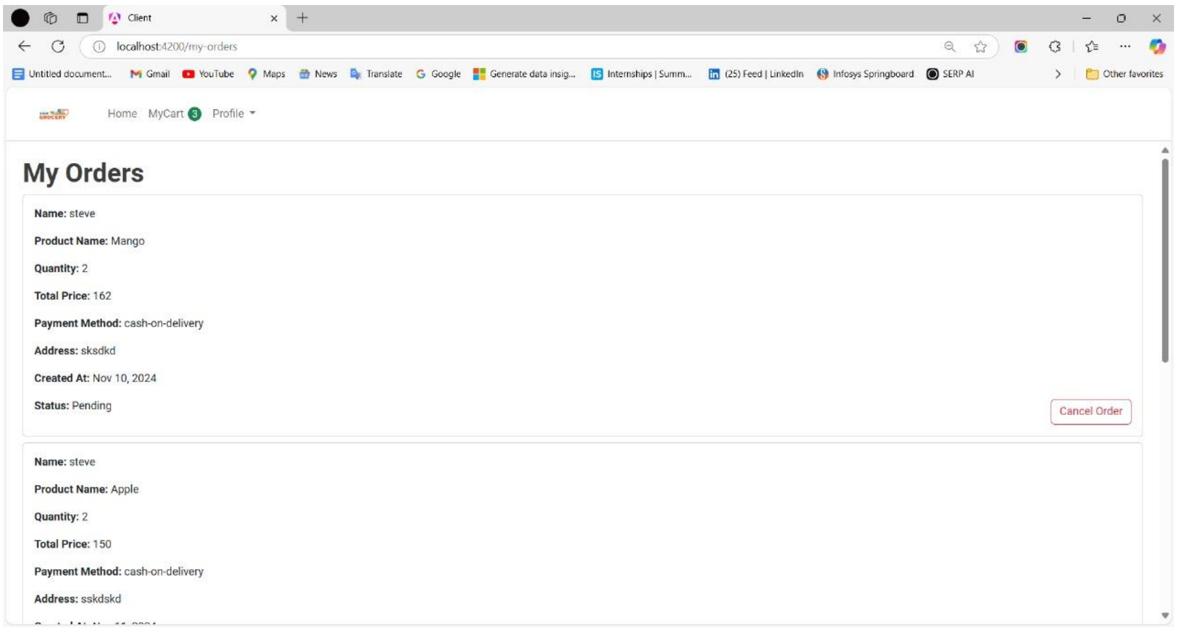
Apple
Price: 75 /-

Butter
Price: 50 /-

Orange
Price: 64 /-

It shows the items in the cart that are added by us

15. Listing of ordered items



My Orders

Name: steve

Product Name: Mango

Quantity: 2

Total Price: 162

Payment Method: cash-on-delivery

Address: sksdkd

Created At: Nov 10, 2024

Status: Pending

Name: steve

Product Name: Apple

Quantity: 2

Total Price: 150

Payment Method: cash-on-delivery

Address: sskdskd

It will shows the items that are ordered by us