ShopSmart: Grocery Store Web App – Documentation

1. Introduction

- **Project Title:** ShopSmart: Your Digital Grocery Store Experience.
- **Team ID:** LTVIP2025TMID56355

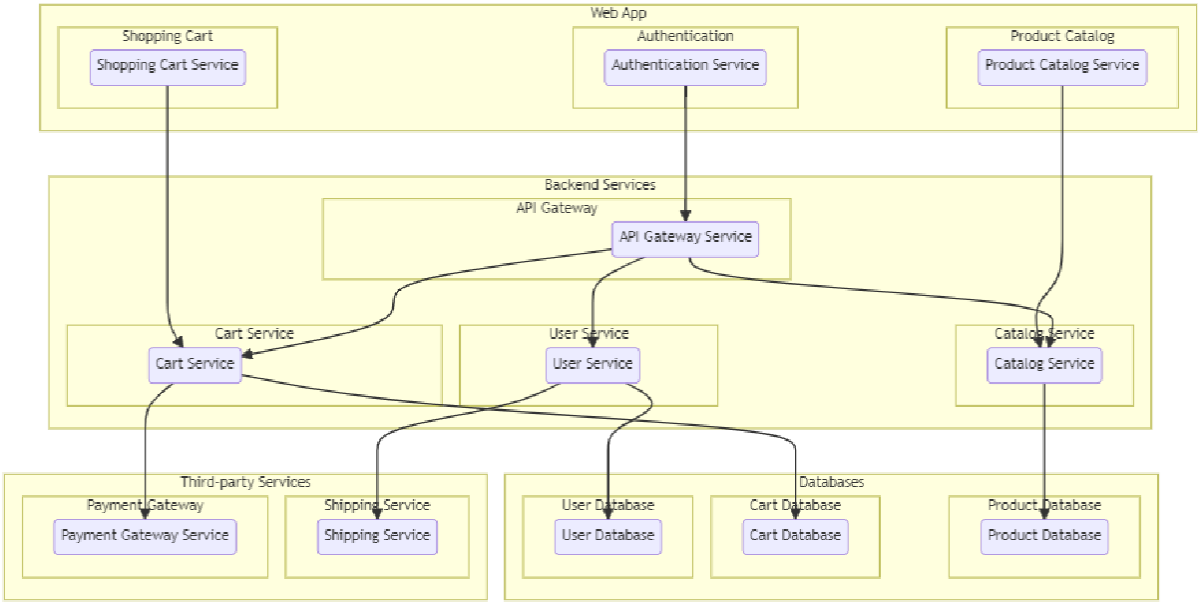
2. Project Overview

- **Purpose:**
  The project is designed to replicate a digital grocery shopping platform, simulating real-world shopping functionality. It enables users to browse products, add items to a cart, and place orders securely after signing in. It aims to improve the ease and accessibility of grocery shopping through a web-based platform.
- **Features:**
  - User authentication handled in a single auth.js component
  - Product listing with images, price, and description
  - "Add to Cart" and "Buy Now" functionalities
  - Cart page showing selected items and total amount
  - Orders page displaying all placed orders per user
  - Persistent backend storage using MongoDB

3. Architecture

- **Tech Stack Used:**
  - Frontend: ReactJS, HTML, CSS, Bootstrap
  - Backend: Node.js with Express.js
  - Database: MongoDB using Mongoose
  - Development Tools: Git, GitHub, Visual Studio Code
- **Frontend (React):**
  The frontend is built using ReactJS with a component-based structure. React Router is used for page navigation without full reloads. Bootstrap is used to maintain a responsive, consistent, and elegant UI. Functional components manage state via React hooks.
- **Backend (Node.js & Express.js):**
  REST API built using Express.js manages routing, middleware, and server logic. API endpoints handle authentication, product management, and order creation.
- **Database (MongoDB):**
  MongoDB stores structured data for users and orders. Mongoose is used to define

schemas and handle data operations. Relationships are established by linking orders to the username field.



4. Setup Instructions

- **Prerequisites:**
    - Node.js (v14 or higher)
    - MongoDB (Community Server or MongoDB Atlas)
    - npm (Node Package Manager)

5. Folder Structure

- **Client (React Frontend):**

```
client/
└── src/
    ├── components/
    |   ├── Home.js
    |   ├── Cart.js
    |   ├── Orders.js
    |   └── Auth.js
    ── App.js
    └── index.js
```

- **Server (NodeJS Backend):**

```
server/
├── models/
│   ├── User.js
│   └── Order.js
├── routes/
│   ├── authRoutes.js
│   └── orderRoutes.js
├── index.js
└── .env
```

6. Running the Application

- **Frontend (React):**

  cd client

  npm start

- **Backend (Express):**

  cd server

  node index.js

The frontend will run on http://localhost:3000/ and backend on http://localhost:5000/.

7. API Documentation

The backend provides a small set of RESTful APIs for handling user authentication and order management. Authentication is handled through a **single route** that determines whether the user is signing up or logging in, based on the request logic.

- **POST /auth**
  This unified endpoint handles both **signup and login** based on the user's input. The request body includes name, mobile, place, and password.
  - If the mobile number already exists, it is treated as a login attempt.
  - If the mobile number is new, a new user is created and stored in MongoDB. This logic is implemented in a single component (auth.js) on the frontend and a corresponding route on the backend.
- **POST /order**
  Accepts product details and the username from the frontend when a user clicks **Buy Now**. Saves the order to MongoDB.

- **GET /orders/:username**
  Fetches all orders associated with the provided username from the database and displays them on the **Orders** page.

These endpoints are designed to keep the project simple and functional, focusing only on essential user-side operations.

8. Authentication

Authentication in this project is handled through a single Auth.js component on the frontend, which allows users to either sign up or log in from the same interface. Users enter their name, mobile number, place, and password. These details are sent to the backend via POST requests for verification or registration.

The backend uses basic matching logic to check whether a user already exists (for login) or to create a new entry in the MongoDB database (for signup). There is no use of advanced authentication mechanisms such as JWT tokens or session cookies.

After successful login, the username is passed through React Router's state and is used across pages like Home, Cart, and Orders to fetch and manage user-specific data. This simple approach is suitable for basic functionality but can be extended in the future for more secure and persistent login sessions.

9. User Interface

The user interface is built using ReactJS and styled with Bootstrap for a clean and responsive layout.

- The **Home page** displays a fixed list of grocery products, each with an image, name, price, and buttons for **"Add to Cart"** and **"Buy Now"**.
- The **Auth page** handles both signup and login using a single form with fields for name, mobile, place, and password.
- The **Cart page** shows selected items and calculates the total cost.
- The **Orders page** lists all products ordered by the current user.
- Navigation between pages is managed using React Router for a smooth experience.

The UI is simple, functional, and focused on providing a basic e-commerce flow.

10. Testing

The application was tested manually to ensure core features work correctly:

- Signup and login were tested with valid and invalid inputs.

- Product cards, cart actions, and total amount calculation were verified.
- Order placement and order history display were checked for the logged-in user.
- Page navigation using React Router was tested without reloads.

No automated testing tools were used; all testing was done through browser and console.

11. Future Enhancements

Planned improvements for future versions include:

- Add secure login using JWT and password hashing.
- Build an admin panel to manage products and view all orders.
- Store and fetch products from the database instead of hardcoding.
- Save cart data to database for persistence.
- Improve mobile responsiveness and add a logout option.
- Integrate search, filters, and online payment support.

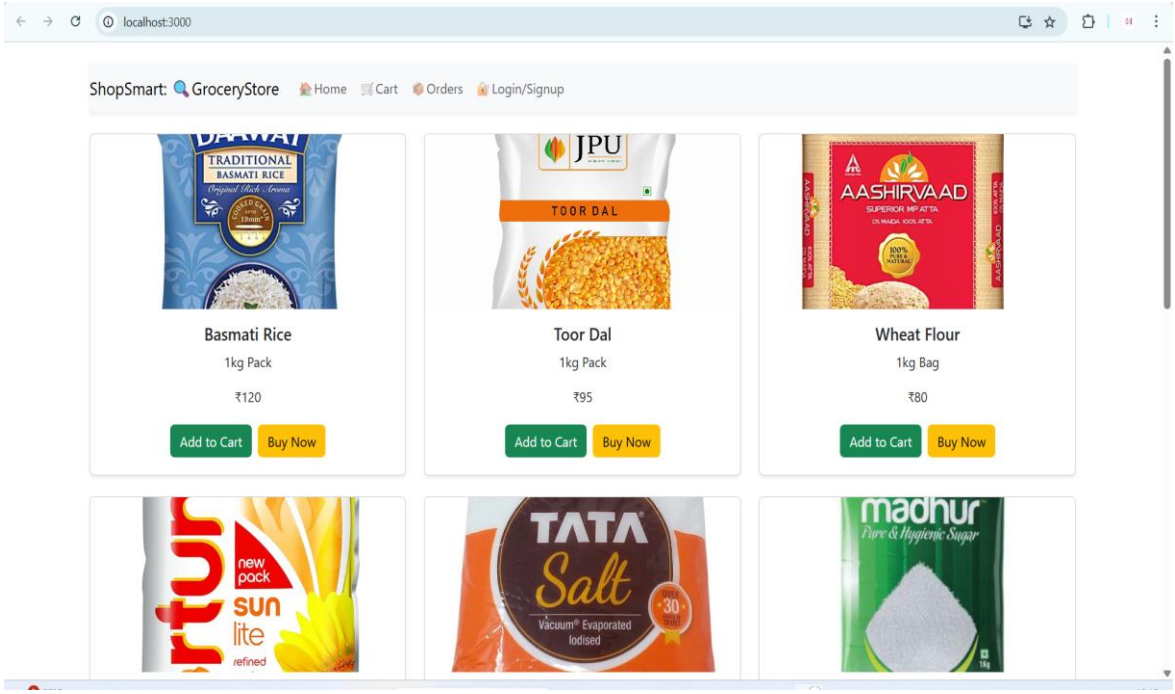
12. Screenshots or Demo

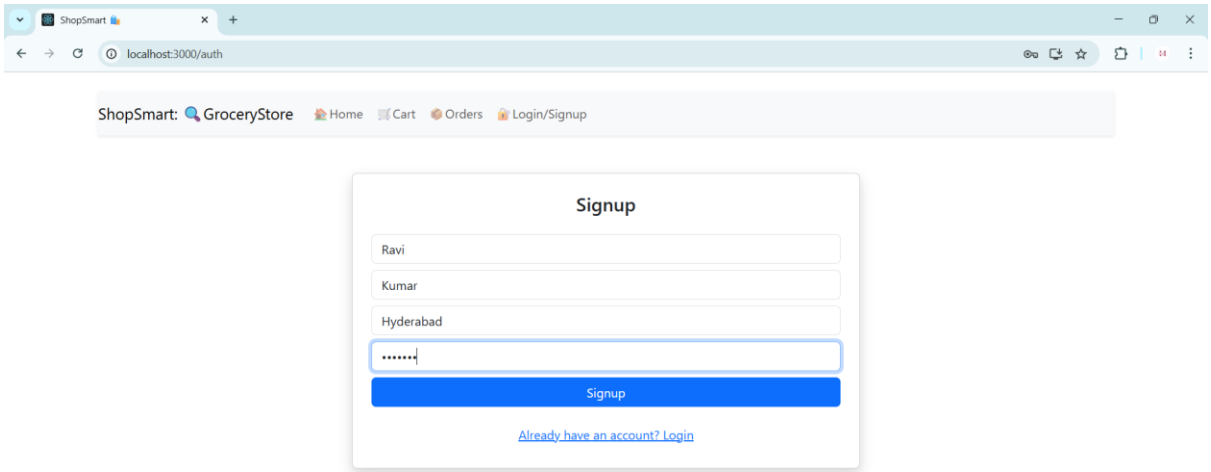
The complete source code and a demo video of this Grocery Store Web Application project are available on GitHub:

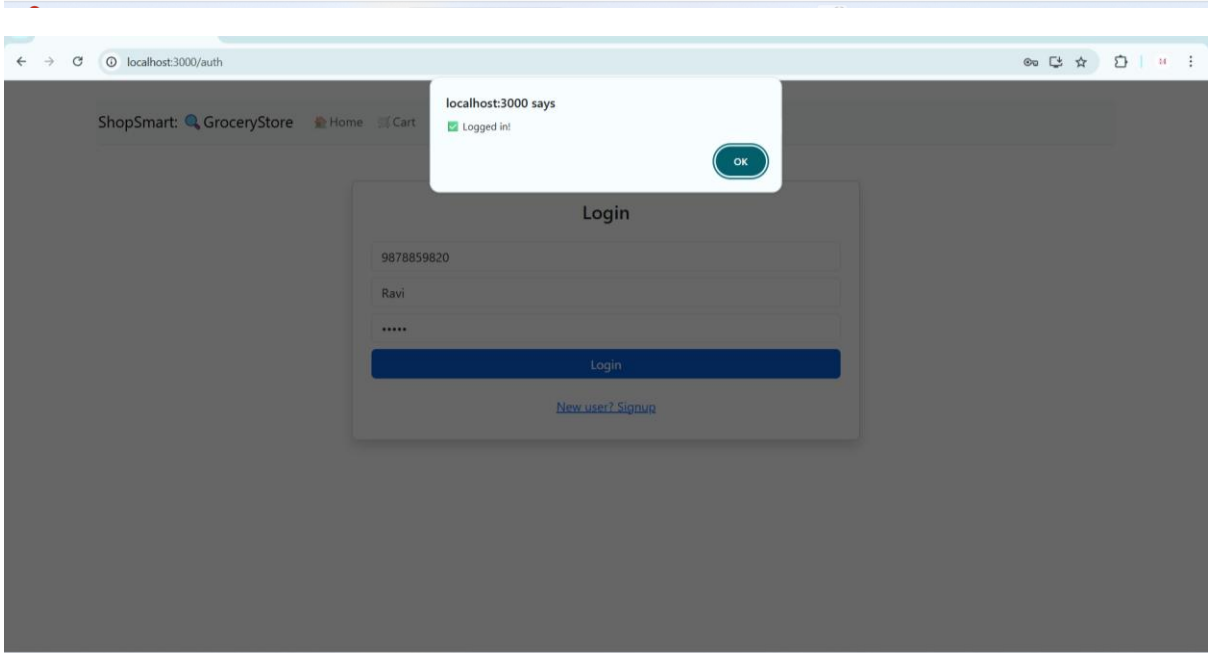
🔗 GitHub Repository Link:
https://github.com/ManjushaBezawada/Shopsmart-Your-Digital-Grocery-Store-Experience.git

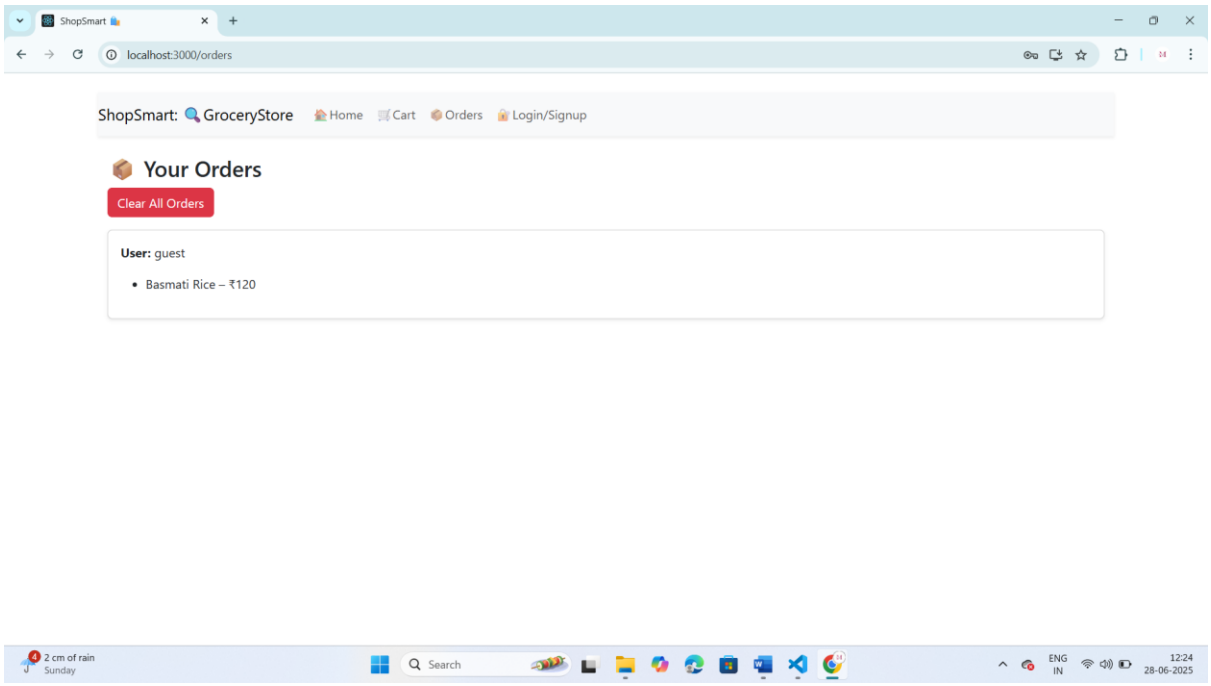
The repository contains:

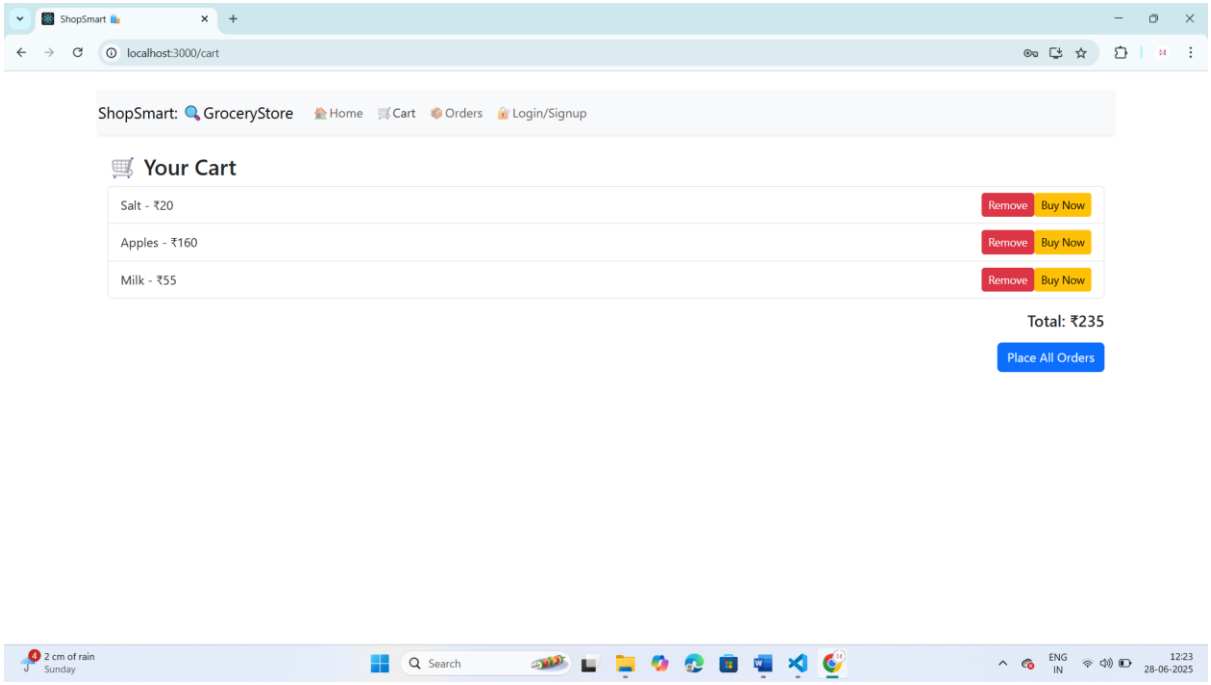
- All frontend and backend source code
- A demo video showcasing the application's features and user flow


localhost:3000
ShopSmart: GroceryStore
Home
Cart
Orders
Login/Signup
TRADITIONAL
BASMATI RICE
Basmati Rice
1kg Pack
₹120
Add to Cart
Buy Now
JPU
TOOR DAL
Toor Dal
1kg Pack
₹95
Add to Cart
Buy Now
AASHIRVAAD
SUPERIOR MP ATTA
Wheat Flour
1kg Bag
₹80
Add to Cart
Buy Now
new pack
sun
lite
refined
TATA
Salt
Vacuum® Evaporated
Iodised
madhur
Pure & Hygienic Sugar


ShopSmart
localhost:3000/auth
ShopSmart: GroceryStore
Home
Cart
Orders
Login/Signup
Signup
Ravi
Kumar
Hyderabad
Signup
Already have an account? Login


localhost:3000/auth
localhost:3000 says
Logged in!
OK
ShopSmart: GroceryStore
Home
Cart
Login
9878859820
Ravi
Login
New user? Signup





Conclusion

This project showcases a simple and functional grocery store web application built with ReactJS, Node.js, Express, and MongoDB. It includes key features like user authentication, product display, cart, and order management. The complete source code and demo video are available on GitHub for reference and future improvements.