Full Stack Development with MERN

1. Introduction

- **Project Title:** Grocery Web App
- **Team Members:**

  1. Surepalli pujitha
  2. Patchigolla Moulika
  3. Lohith Pattubala
  4. Himasruthi S

  Everyone is involved in all the phases of development.

2. Project Overview

- **Purpose:**

  The primary purpose of our grocery web app is to provide a seamless and convenient online shopping experience for customers, allowing them to explore and purchase a wide range of products. By offering user-friendly navigation and intuitive design, we aim to make online shopping accessible and enjoyable for everyone, from tech enthusiasts and fashionistas to homemakers seeking everyday essentials.

- **Features:**

  1. **User Authentication:** Secure login and registration for users, ensuring that only authorized users can access certain features.
  2. **Detailed Product Information**: Users can view comprehensive details about each product.
  3. **Shopping Cart**: Customers can add items to their cart and manage their selections.
  4. **Secure Checkout:** The app provides a secure and straightforward checkout process.
  5. **Search Functionality**: Users can search Products through Filtering results by various criteria.
  6. **Category Navigation:** Easy navigation through product categories and subcategories.
  7. **Security and Privacy:** The app ensures the protection of customer data, secure transactions, and confidentiality of personal information.
  8. **Payment Options:** The app provides different payment options

3. Architecture

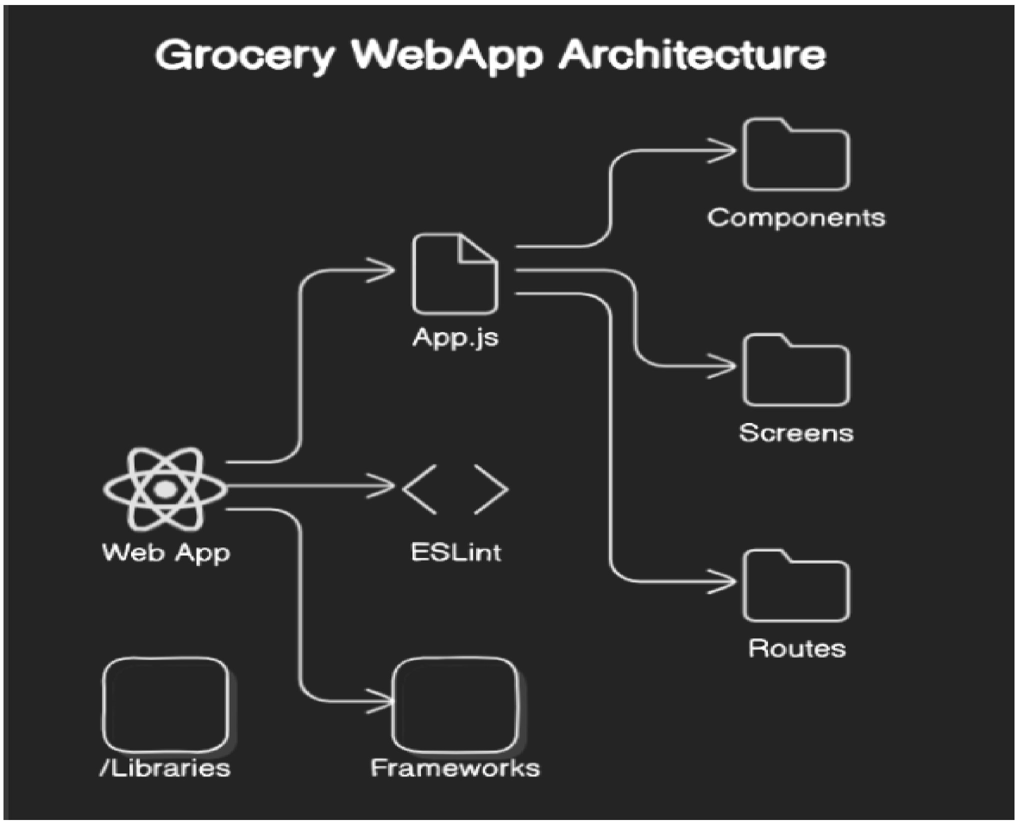
Frontend: Our frontend architecture for the Grocery Web App is designed to be modular, scalable, and maintainable. It uses React, React Bootstrap, Router Bootstrap, React Toastify and React Router to build a responsive and dynamic user interface.

Overview:

1. React Router: Provides routing capabilities for React applications, allowing navigation between different screens or pages.
2. React Bootstrap: Provides Bootstrap components for React applications, allowing the use of pre-styled UI components.
3. React Router Bootstrap: Integrates React Router with React Bootstrap, allowing the use of Bootstrap-styled links with React Router.
4. React Toastify: Provides toast notifications for React applications.
5. React: For building user interfaces
6. Axios: For HTTP requests.
7. Context API : Provides a way to pass data through the component tree without having to pass props down manually at every level.

Key Components :

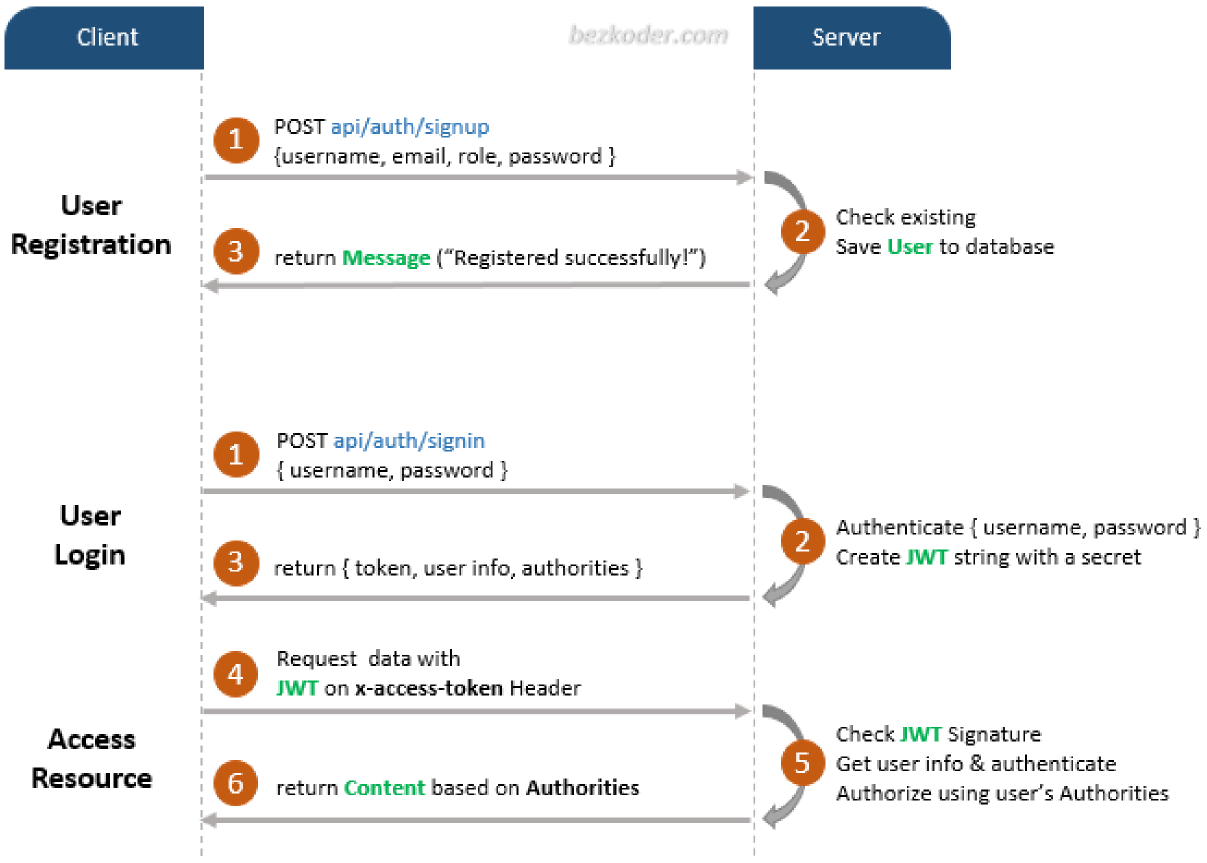
1. App.js: Main application layout and routing.
2. components: Reusable UI components.
3. pages: Screens like Home, Search, and Product List,etc
4. routes: Routing configurations.
5. lib: Utility functions, loaders, and API request



Backend: Our backend architecture for the Grocery Web App is designed to be modular, scalable, and maintainable. It uses Node.js, Express.js, to interact with a MongoDB database.

Overview :

1. Node.js: JavaScript runtime for server-side operations.
2. Express.js: Web framework for creating the RESTful API.
3. MongoDB: NoSQL database for storing application data.
4. JWT: For secure token-based authentication.



Client
bezkoder.com
Server
User Registration
1 POST api/auth/signup {username, email, role, password }
2 Check existing Save User to database
3 return Message ("Registered successfully!")
User Login
1 POST api/auth/signin { username, password }
2 Authenticate { username, password } Create JWT string with a secret
3 return { token, user info, authorities }
Access Resource
4 Request data with JWT on x-access-token Header
5 Check JWT Signature Get user info & authenticate Authorize using user's Authorities
6 return Content based on Authorities

Database: The Grocery webapp's database architecture utilizes MongoDB for efficient and type-safe data management. The architecture includes the following key collections:

1. Users Collection: Stores user information and manages authentication and relationships.
2. Product Collection: Contains details about available products on the webpage.
3. Order details Collection: stores order information such as order ID , delivery details and payment status.

4. Setup Instructions

- **Prerequisites:**

    1. Node.js
    2. MongoDB
    3. Git

- **Installation:**

    1. Clone the repository:

https://github.com/HimaSruthi25/Grocery-webapp.git

2. Setup MongoDB

- Local MongoDB
    - In .env file update MONGODB_URI=mongodb://localhost/grocery

3. **Run Backend**

```
$ cd backend
$ npm install
$ npm start
```

4. **Run Frontend**

```
# open new terminal
$ cd frontend
$ npm install
$ npm start
```

5. Folder Structure

- **Client:**

```
client/
│
├── node_modules/
│   └── …
│
├── public/
│   └── … (Assets)
│
```

```
│
├── src/
│   ├── components/
│   │   └── AdminRoute.js
│   │   └── CheckoutSteps.js
│   │   └──LoadingBox.js
│   │   └── MessageBox.js
│   │   └── Product.js
│   │   └── ProtectedRoute.js
│   │   └── Rating.js
      │   │   └── SearchBox.js
      │   ├── screens/
│   │   └── CartScreen.js
│   │   └──DashboardScreen.js
│   │   └──ForgetPasswordScreen.js
│   │   └── HomeScreen.js
│   │   └── MapScreen.js
│   │   └──OrderHistoryScreen.js
│   │   └── OrderListScreen.js
│   │   └── orderScreen.js
│   │   └── PaymentMethodScreen.js
│   │   └── PlaceOrderScreen.js
│   │   └── ProductEditScreen.js
│   │   └── ProductListScreen.js
│   │   └── ProductScreen.js
│   │   └── ProfileScreen.js
│   │   └── ResetPasswordScreen.js
│   │   └── SearchScreen.js
│   │   └── ShippingAddressScreen.js
│   │   └── SigninScreen.js
│   │   └── SignupScreen.js
│   │   └── UserEditScreen.js
│   │   └── UserListScreen.js
│   │
│   ├── App.css
│   ├── App.js
│   ├── App.test.js
│   ├── data.js
│   ├── index.css
│   ├── index.js/
│   ├── logo.svg
│   ├──reportWebVitals.js
│   ├── setupTests.js
│   ├── Store.js
│   ├── utils.js │
│
│ ── package-lock.json
└── package.json
```

- **Server:**

```
client/
│
├── models/
│   ├── orderModel.js
│   ├── productModel.js
│   ├── userModel.js
│
├── routes/
│   ├── orderRoutes.js │
│   ├── productRoutes.js
│   ├── seedRoutes.js
│   ├──uploadRoutes.js
│   └── user.Routes.js
│
├── .env
├── .gitignore
├── data.js
├── server.js
└── utils.js
├── package-lock.json
└── package.json
```

6. Running the Application

- Commands to start the frontend and backend servers locally.
    - **Frontend:** in terminal cd .\frontend\ then npm start
    - **Backend:** in terminal cd .\backend\ then npm start

7. API Documentation

➢ **User Authentication**

- **POST /api/users/register**: Registers a new user.
- **POST /api/users/login**: Authenticates a user and returns a token.

➢ **User Management**

- **GET /api/users/**: Retrieves user information by ID.
- **PUT /api/users/**: Updates user information by ID.

➢ **Product Management**

- **GET /api/products**: Retrieves all products.
- **POST /api/products**: Creates a new product.
- **GET /api/products/**: Retrieves product information by ID.
- **PUT /api/products/**: Updates product information by ID.

➢ **Order Management**

- **GET /api/orders**: Retrieves all orders.
- **POST /api/orders**: Creates a new order.
- **GET /api/orders/**: Retrieves order information by ID.
- **PUT /api/orders/**: Updates order information by ID.
- **GET /api/orders/mine**: Retrieves orders of the authenticated user.
- **PUT /api/orders//pay**: Updates order payment status.

➢ **Seed Data**

- **POST /api/seed**: Seeds initial data.

➢ **File Uploads**

- **POST /api/upload**: Uploads a file.

8. Authentication

In our Grocery Web App Project, authentication and authorization are handled through a secure token-based system using JSON Web Tokens (JWT). Below are the key aspects of how authentication and authorization are managed:

1. Authentication Flow

1. **User Registration:**

    - Endpoint: POST /api/auth/register
    - Users can register by providing their username, email, and password.
    - The password is hashed before being stored in the database for security.

2. **User Login:**

    - Endpoint: POST /api/auth/login
    - Users authenticate by providing their email and password.
    - If the credentials are correct, a JWT token is generated and sent back to the client.

3. **User Logout:**

    - Endpoint: POST /api/auth/logout
    - Users can log out, which invalidates the token on the client side

2. Token Handling

- **JWT Generation:**

  Upon Successful Login:
    - A JWT (JSON Web Token) is generated using a secret key upon successful user login.
    - The token contains the user's ID and expiration information.

- **Token Storage:**

  Client-Side Storage:
    - The token is stored on the client side, typically in localStorage or cookies.
    - For API requests requiring authentication, the token is included in the request headers.

- **Token Verification:**

  Middleware for Protected Routes:
    - Each protected route on the backend uses middleware to verify the JWT.
    - **Middleware:** isAuth and isAdmin check the validity of the token. If the token is valid, the request is allowed to proceed; otherwise, it is rejected with an error.

3. Security Measures

- **Password Hashing:**

* User passwords are hashed using bcrypt before being stored in the database, ensuring that plain text passwords are never stored.

- **Token Expiration:**

* JWT tokens have an expiration time set to limit their validity, reducing the risk of token misuse.

➢ **Secure Token Storage:**

- Storing tokens in secure, http-only cookies to protect against cross-site scripting (XSS) attacks.

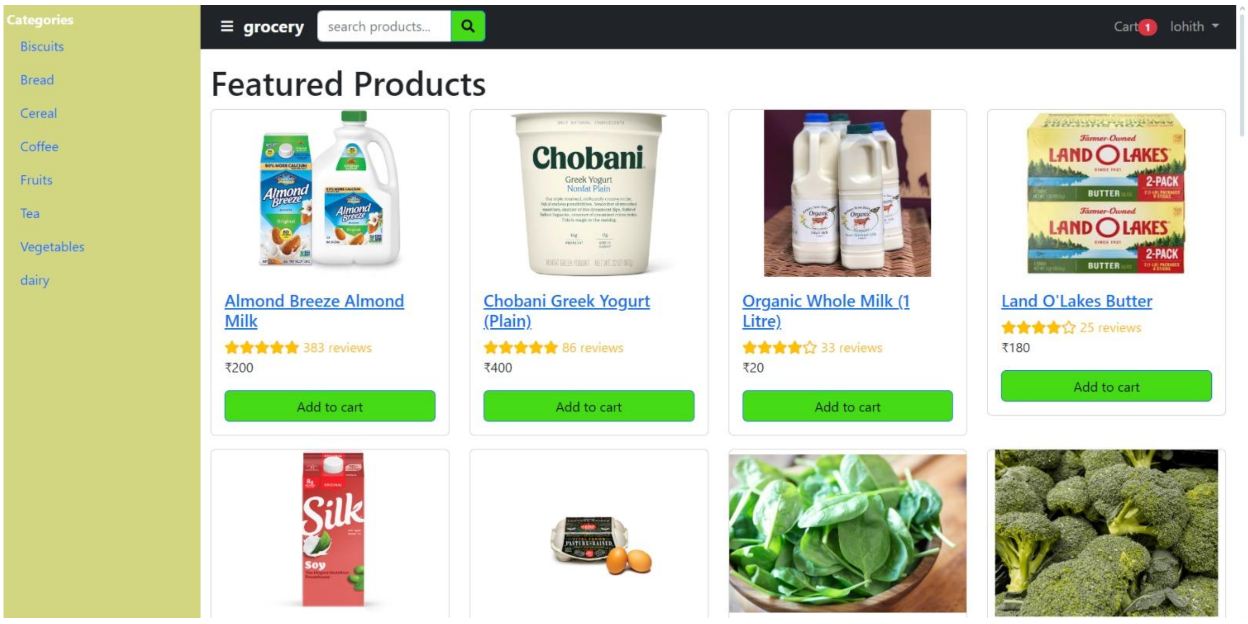
➢ **Token Verification:**

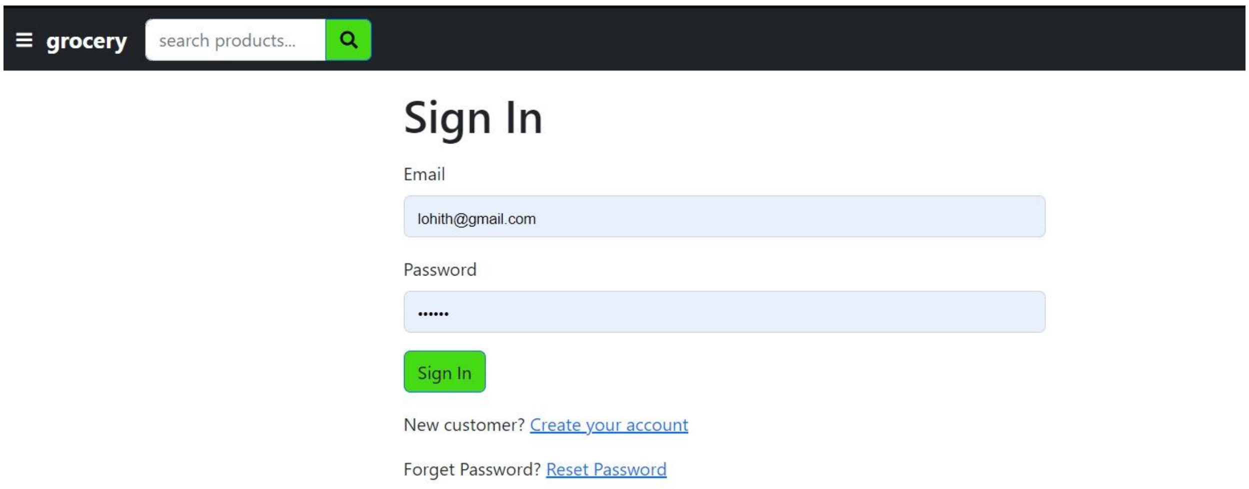
- Each protected route on the backend uses middleware to verify the JWT. Middleware checks the validity of the token. If the token is valid, the request is allowed to proceed; otherwise, it is rejected with an error.

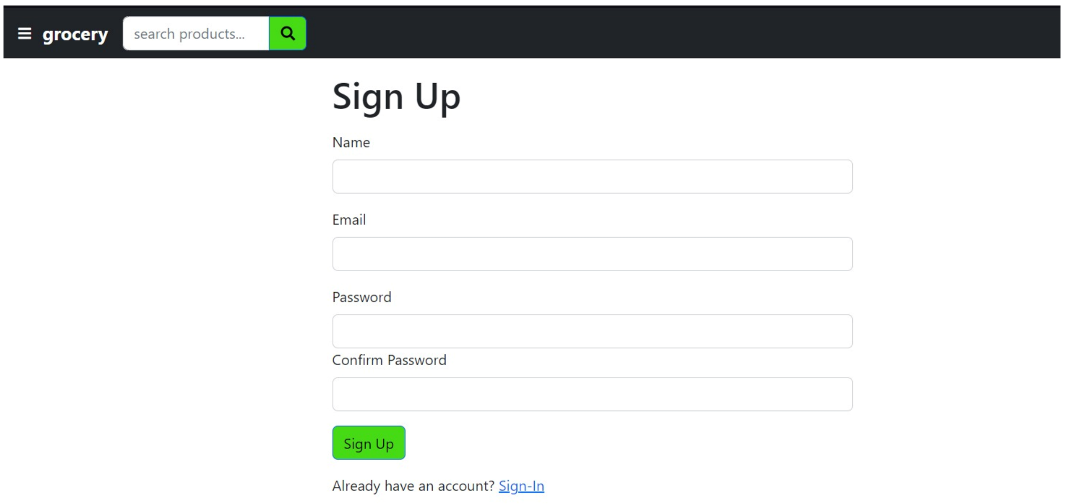
➢ **HTTPS:**

- Ensure that all communications between the client and server are encrypted using HTTPS to protect sensitive data in transit.

9. User Interface:






≡ grocery search products...

Sign Up

Name

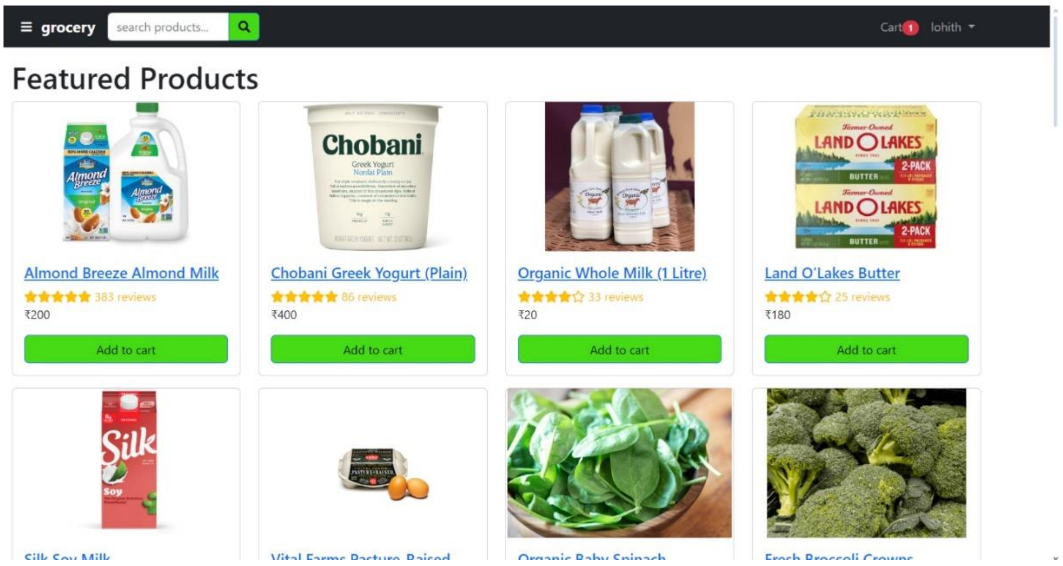
Email

Password

Confirm Password

Sign Up

Already have an account? Sign-In


≡ grocery search products... Cart 1 lohith

Featured Products

Almond Breeze Almond Milk ★★★★★ 383 reviews ₹200 Add to cart

Chobani Greek Yogurt (Plain) ★★★★★ 86 reviews ₹400 Add to cart

Organic Whole Milk (1 Litre) ★★★★☆ 33 reviews ₹20 Add to cart

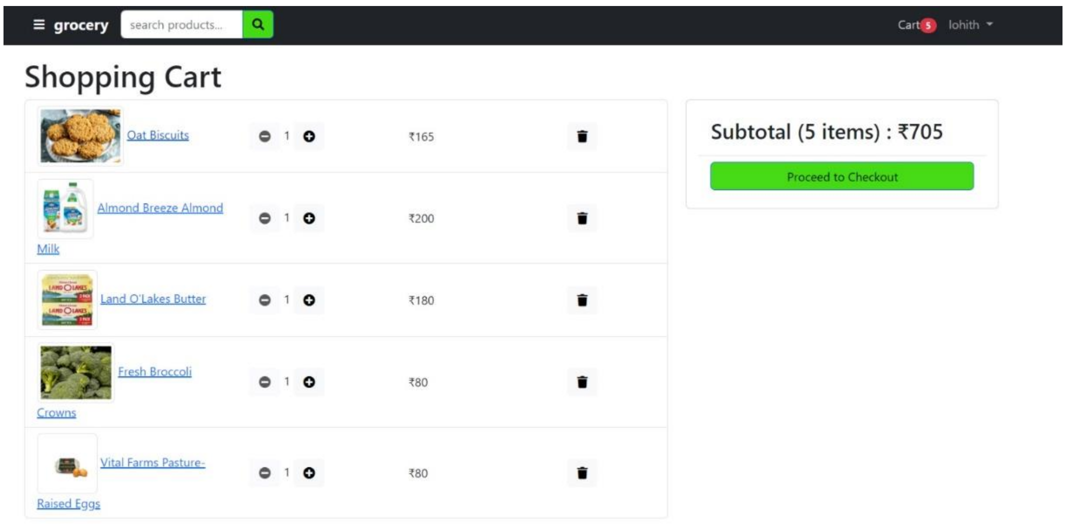
Land O'Lakes Butter ★★★★☆ 25 reviews ₹180 Add to cart

Silk Soy Milk

Vital Farms Pasture-Raised

Organic Baby Spinach

Fresh Broccoli Crowns

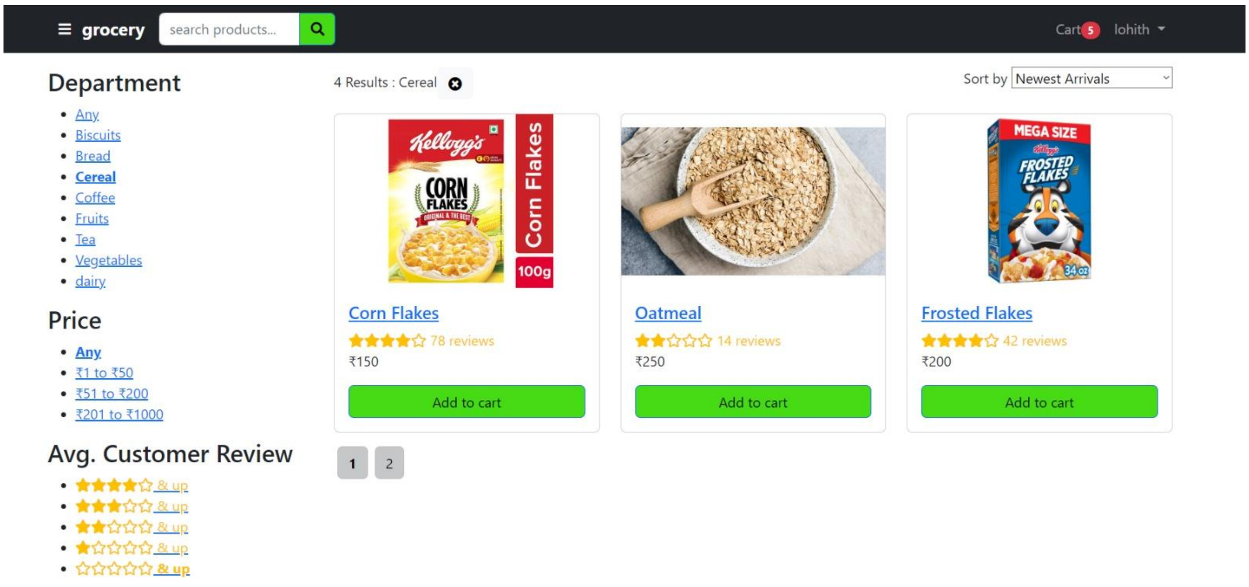

≡ grocery search products... Cart 5 lohith

Shopping Cart

Oat Biscuits	1	₹165	
Almond Breeze Almond Milk	1	₹200	
Land O'Lakes Butter	1	₹180	
Fresh Broccoli Crowns	1	₹80	
Vital Farms Pasture-Raised Eggs	1	₹80	

Subtotal (5 items) : ₹705

Proceed to Checkout



≡ grocery search products... Cart 5 lohith

4 Results : Cereal ⊗

Sort by Newest Arrivals

Department

- Any
- Biscuits
- Bread
- **Cereal**
- Coffee
- Fruits
- Tea
- Vegetables
- dairy

Price

- **Any**
- ₹1 to ₹50
- ₹51 to ₹200
- ₹201 to ₹1000

Avg. Customer Review

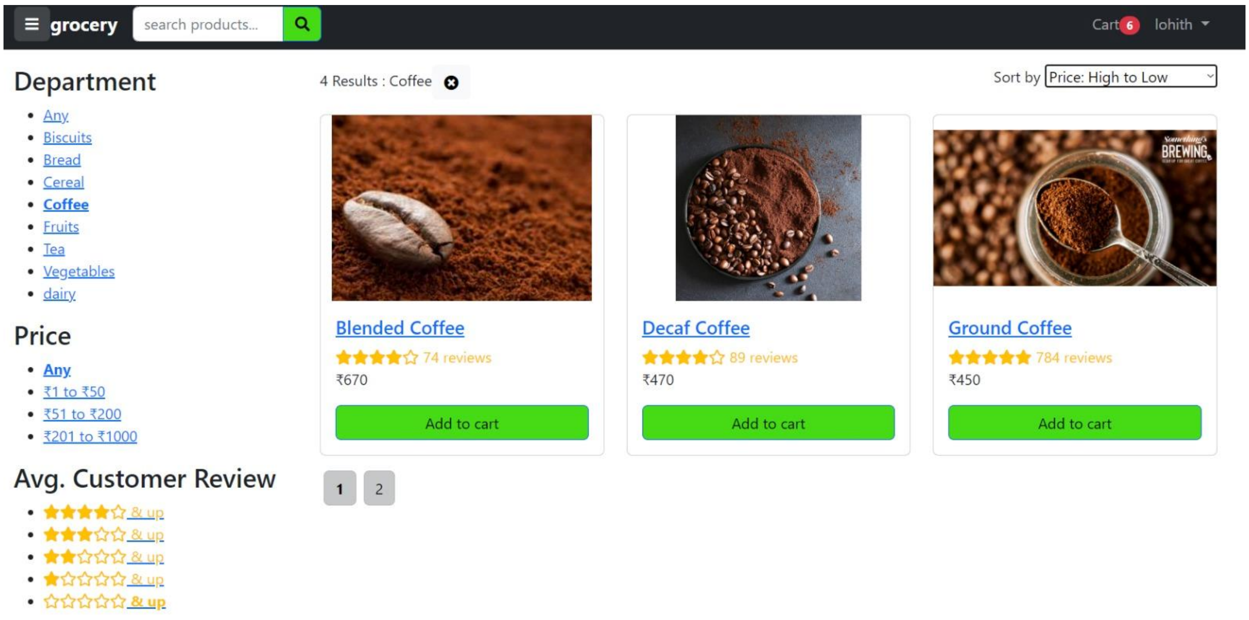
- ★★★★☆ & up
- ★★★☆☆ & up
- ★★☆☆☆ & up
- ★☆☆☆☆ & up
- ☆☆☆☆☆ **& up**

Corn Flakes
★★★★☆ 78 reviews
₹150
Add to cart

Oatmeal
★★☆☆☆ 14 reviews
₹250
Add to cart

Frosted Flakes
★★★★☆ 42 reviews
₹200
Add to cart

1 2



≡ grocery search products... Cart 6 lohith

4 Results : Coffee ⊗

Sort by Price: High to Low

Department

- Any
- Biscuits
- Bread
- Cereal
- **Coffee**
- Fruits
- Tea
- Vegetables
- dairy

Price

- **Any**
- ₹1 to ₹50
- ₹51 to ₹200
- ₹201 to ₹1000

Avg. Customer Review

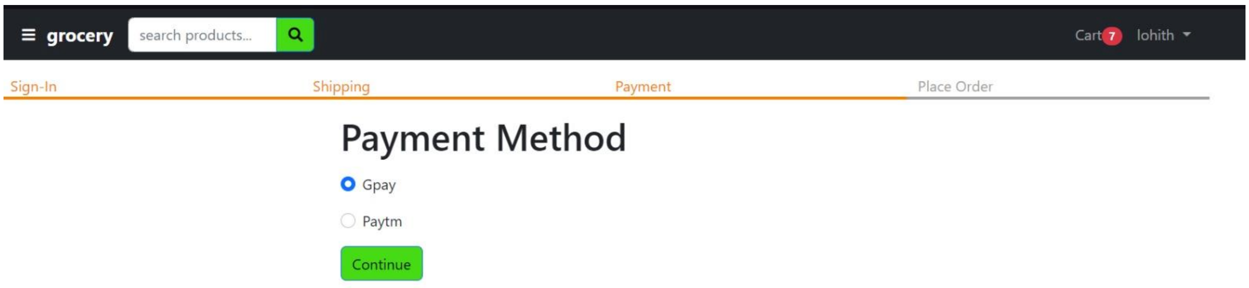
- ★★★★☆ & up
- ★★★☆☆ & up
- ★★☆☆☆ & up
- ★☆☆☆☆ & up
- ☆☆☆☆☆ **& up**

Blended Coffee
★★★★☆ 74 reviews
₹670
Add to cart

Decaf Coffee
★★★★☆ 89 reviews
₹470
Add to cart

Ground Coffee
★★★★★ 784 reviews
₹450
Add to cart

1 2



≡ grocery search products... Cart 7 lohith

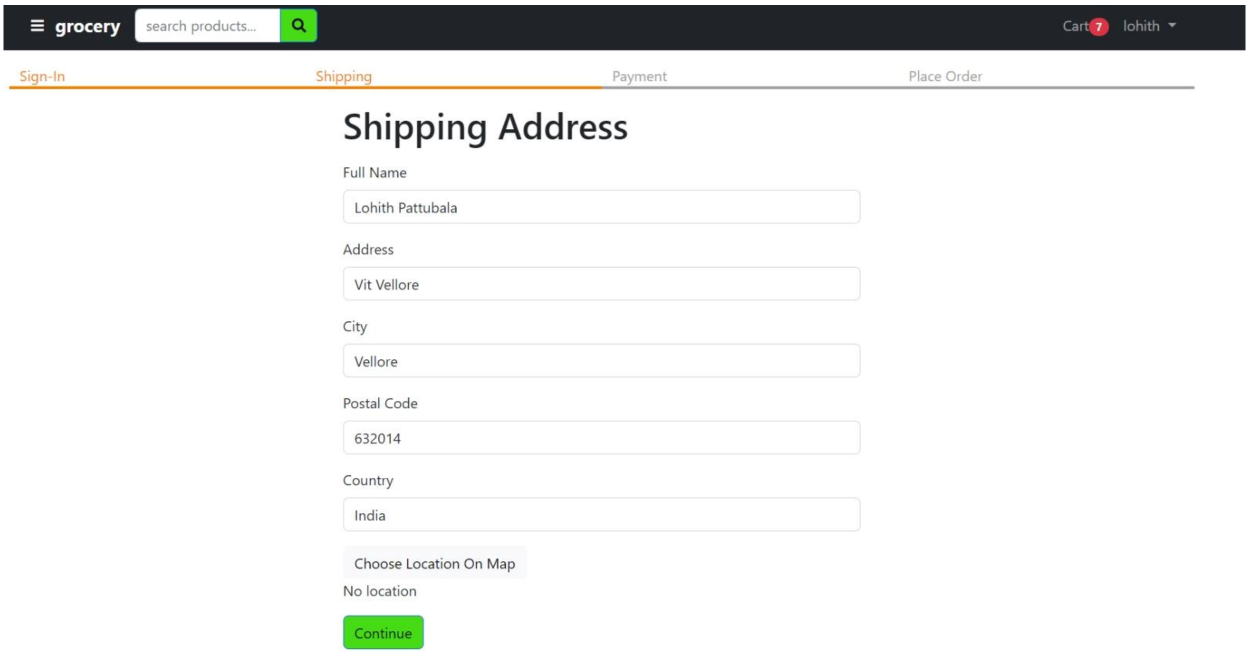
Sign-In Shipping Payment Place Order

Payment Method

● Gpay

○ Paytm

Continue



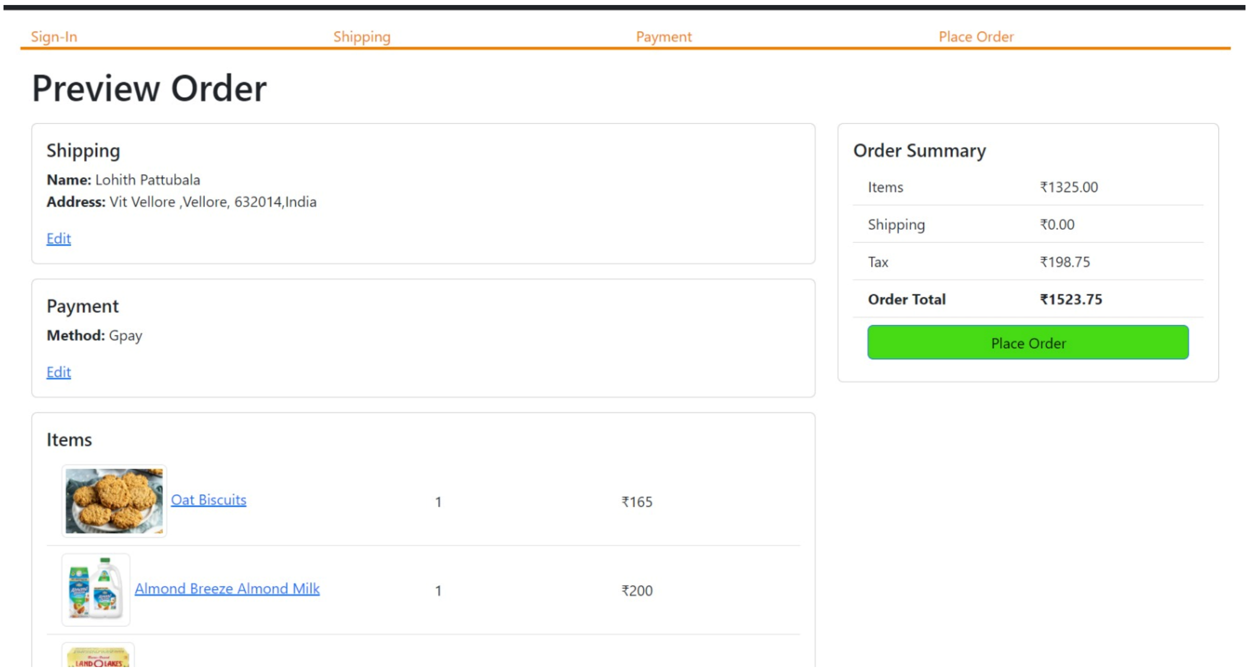
Shipping Address

Full Name	Lohith Pattubala
Address	Vit Vellore
City	Vellore
Postal Code	632014
Country	India

Choose Location On Map

No location

Continue



Preview Order

Shipping

Name: Lohith Pattubala

Address: Vit Vellore ,Vellore, 632014,India

Edit

Payment

Method: Gpay

Edit

Items

Oat Biscuits	1	₹165
Almond Breeze Almond Milk	1	₹200

Order Summary

Items	₹1325.00
Shipping	₹0.00
Tax	₹198.75
Order Total	**₹1523.75**

Place Order

10. Testing

Testing Strategy and Tools Used:

Our testing strategy for the house hunt website primarily focused on ensuring functionality, performance, and usability across various components. We employed a combination of manual testing by team members and automated testing using Postman. The goal was to validate both frontend and backend functionalities, ensuring seamless integration and user experience.

- **Manual Testing:** Team members conducted thorough manual testing across different browsers and device sizes to ensure responsiveness and visual consistency. Key aspects tested include user registration, login/logout, house search, listing details, and user interactions.
- **Automated Testing with Postman:** We utilized Postman for API testing to validate backend functionalities such as CRUD operations for house listings, user authentication, and data validation. This approach helped us ensure API endpoints returned expected responses and handled edge cases effectively.

11. Screenshots or Demo

Demo Link : https://drive.google.com/file/d/1H7a4Esn6UiGVKVMS45-b078hlSDf4Ivo/view?usp=sharing

12. Known Issues

Currently, there are one issue with the developed features of the Grocery Web App.

Issue: filtering and sorting products by prices, reviews.

The team has thoroughly tested the existing functionalities, and everything is working as expected except the issue mentioned above.

However, as with any software project, new issues may arise as more features are added and more users interact with the application.

13.Future Enhancements

Outline potential future features or improvements that could be made to the project :

Personalized Recommendations:

- Implement AI algorithms to provide personalized product recommendations based on user behavior and purchase history.

Subscription Services:

- Offer subscription-based services for regular delivery of frequently purchased items, such as household essentials.

Admin Dashboard

- Manage listings, users, and platform activity.
- Analytics and reporting tools.

Multi-language Support

- Implement multi-language options for a broader audience.