**Full Stack Development with MERN**

1. **Introduction :**

* **Project Title :** ShopSmart Your Digital Grocery Store Experience.
* **Team Members :** 
  + - 1. A Sai
      2. Adari Hema Manoha
      3. Chitturi Jaya Shalini
      4. Ganireddy Divya Venkata Darni

1. **Project Overview:**

* **Purpose:**

The purpose of Our Grocery Web App is more than just a convenient way to shop for groceries—it's a comprehensive solution designed to enhance every aspect of your shopping experience. Imagine a virtual supermarket at your fingertips, offering an extensive selection of high-quality products that cater to your every need.

From fresh produce sourced directly from local farms to pantry staples and household essentials, our app provides everything you need to keep your kitchen well-stocked. Our user-friendly interface ensures that you can navigate our virtual aisles effortlessly, easily finding what you're looking for and adding items to your cart with a simple click. But the convenience doesn't stop there.

With our app, you can schedule deliveries at your preferred time, ensuring that your groceries arrive when it's most convenient for you. Say goodbye to long lines and tedious grocery trips—our app brings the supermarket to you, saving you time and hassle. We understand that everyone's dietary preferences and requirements are different, which is why our app offers a wide range of options, including organic, gluten-free, and vegan products. You can also discover new and exciting products through our app, making grocery shopping an adventure rather than a chore.

Our commitment to quality extends beyond our products to our customer service. Our dedicated team is always available to assist you with any questions or concerns you may have, ensuring that your shopping experience is nothing short of exceptional. Experience the future of grocery shopping with our Grocery Web App. Download it today and discover the convenience of fresh, high-quality groceries delivered right to your doorstep

* **Features:**

1. **Product CatLog:** Our grocery-webapp app provides an extensive product catalog with various categories and subcategories. Users can easily search, browse, and filter products based on their preferences, making it effortless to find the desired items.
2. **Shopping Cart and Checkout:** The app includes a shopping cart feature that enables users to add products, review their cart, and proceed to checkout. The checkout process offers multiple payment options, ensuring a smooth and secure transaction experience.
3. **Product Reviews and Ratings:** Customers can provide feedback and rate products, helping other users make informed purchasing decisions. This feature fosters a sense of community and trust among users.
4. **Order Tracking:** Once an order is placed, users can track its status in real-time. They receive updates on order processing, shipping, and delivery, providing transparency and peace of mind.
5. **Admin Dashboard:** For administrators, our grocery-webapp app offers a comprehensive dashboard to manage products, inventory, orders, and customer information. It provides insights into sales performance, stock levels, and customer analytics, enabling efficient business operations.
6. **Order Management:** The app manages the order lifecycle, including order placement, tracking, and status updates. Users can view their order history, track shipments, and request returns or cancellations.
7. **Search and Filtering:** Users can search for products using keywords and apply filters to narrow down the search results based on criteria such as price range, brand, or customer ratings.
8. **Shopping Cart and Checkout:** The app includes a shopping cart feature that enables users to add products, review their cart, and proceed to checkout. The checkout process offers multiple payment options, ensuring a smooth and secure transaction experience.
9. **Product Reviews and Ratings:** Customers can provide feedback and rate products, helping other users make informed purchasing decisions. This feature fosters a sense of community and trust among users.
10. **Order Tracking:** Once an order is placed, users can track its status in real-time. They receive updates on order processing, shipping, and delivery, providing transparency and peace of mind.
11. **Admin Dashboard:** For administrators, our grocery-webapp app offers a comprehensive dashboard to manage products, inventory, orders, and customer information. It provides insights into sales performance, stock levels, and customer analytics, enabling efficient business operations.
12. **Order Management:** The app manages the order lifecycle, including order placement, tracking, and status updates. Users can view their order history, track shipments, and request returns or cancellations.
13. **Search and Filtering:** Users can search for products using keywords and apply filters to narrow down the search results based on criteria such as price range, brand, or customer ratings.

**Functionalities:**

* End-to-end order lifecycle management.
* Role-based access control for users and admins.
* Inventory and user data management.
* Secure payment integration.
* Insightful analytics for administrators.

**4.Architecture**

* **Frontend:**

The frontend of ShopSmart is developed using two modern frameworks:

* **Angular**: Used for online demonstration and testing purposes via StackBlitz. It includes modular design, reusable UI components, and routing. Angular’s powerful CLI allowed for quick prototyping of product cards, navigation, and search components.
* **React**: Implemented for the local full-scale development. React’s component-based architecture simplifies UI updates and allows dynamic rendering of product listings, cart contents, and user data using hooks and props. The project follows a container-component structure, and React Router DOM is used for client-side routing.

Styling is handled using CSS, with some reusable design utilities and flex-based layout for responsiveness.

* **Backend:**

The backend is powered by **Node.js** using the **Express.js** framework. Key backend responsibilities include:

* API routing for product, user, and order modules
* User registration, login, and role-based access management
* Order creation, cart syncing, and history retrieval
* Handling review submission and fetching
* Error handling middleware for server-side validations

Authentication is handled using **JSON Web Tokens (JWT)** for stateless sessions.

* **Database:**

**MongoDB** serves as the primary NoSQL database for ShopSmart. It stores:

* Product information (name, category, price, image, stock count)
* User profiles with roles (admin/customer)
* Order details and payment statuses
* Ratings and reviews per product

**Mongoose** is used to define schemas, enforce data consistency, and simplify CRUD operations. The database is either locally hosted during development or deployed using MongoDB Atlas in production scenarios.

**4.Setup Instructions:**

* **Prerequisites:**
  + Node.js
  + npm (Node Package Manager)
  + MongoDB (local or cloud-based like Atlas)
* **Installation:**

# Clone the repository

git clone [your-repo-link]

# Navigate to client folder (React)

cd client

npm install

# Navigate to server folder

cd ../server

npm install

# Start servers

cd server && npm start

cd ../client && npm start

Set environment variables in .env file for MongoDB URI and secret keys.

**5. Folder Structure**

**• Client:**

* src/components/ – UI Components like ProductCard, Cart, Header
* src/pages/ – Page-level components (Home, Checkout, Dashboard)
* src/App.js – Main routing file

**• Server:**

* routes/ – API route handlers (products, users, orders)
* controllers/ – Business logic for routes
* models/ – Mongoose schemas (Product, User, Order)
* server.js – Main Express app

**6. Running the Application:**

* **Frontend:**

cd client

npm start

* **Backend:**

cd server

npm start

**7. API Documentation**

User Authentication

* POST /auth/register  
  Registers a new user in the system.  
  Request Body:  
  { "name": "Kavya", "email": "kavya@example.com", "password": "securePassword" }  
  Response:  
  { "message": "User registered successfully", "userId": "64a84a12b6c91b" }
* POST /auth/login  
  Authenticates a user and returns a JWT token.  
  Request Body:  
  { "email": "kavya@example.com", "password": "securePassword" }  
  Response:  
  { "token": "jwt\_token\_string", "role": "user" }

🔹 Product APIs

* GET /products  
  Returns a list of all available products.  
  Response:

[

{

"id": "1",

"name": "Apple",

"category": "Fruits",

"price": 100,

"stock": 50,

"image": "apple.jpg"

}

]

* GET /products/:id  
  Retrieves details of a specific product using its ID.

🔹 Cart Management

* POST /cart/add  
  Adds a selected product to the user’s cart.  
  Request Body:  
  { "userId": "64a8...", "productId": "1", "quantity": 2}  
  Response:  
  { "message": "Product added to cart"}

🔹 Order Management

* POST /orders  
  Places a new order from the user's cart.  
  Request Body:  
  { "userId": "64a8...", "paymentMethod": "COD"}  
  Response:  
  { "message": "Order placed successfully", "orderId": "ORD6789"}
* GET /orders/: userId  
  Retrieves the list of orders placed by a specific user.

🔹 Admin APIs

* POST /admin/product  
  Allows the admin to add a new product.  
  Request Body:  
  { "name": "Milk", "category": "Dairy", "price": 50, "stock": 100, "image": "milk.jpg" }  
  Response:  
  { "message": "Product added successfully"}
* DELETE /admin/product/:id  
  Allows the admin to delete a product by its ID.

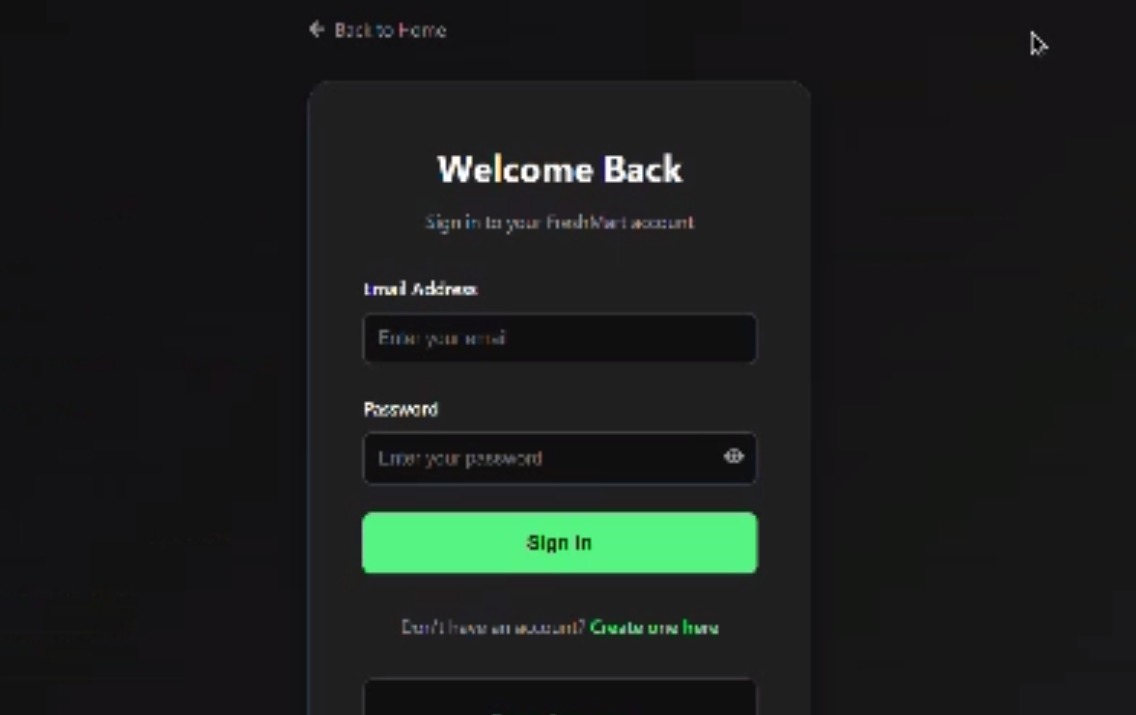
**8. Authentication:**

Authentication and authorization in the ShopSmart Grocery Web App are implemented using a secure, token-based mechanism to ensure that users can only access features appropriate to their role (customer or admin).

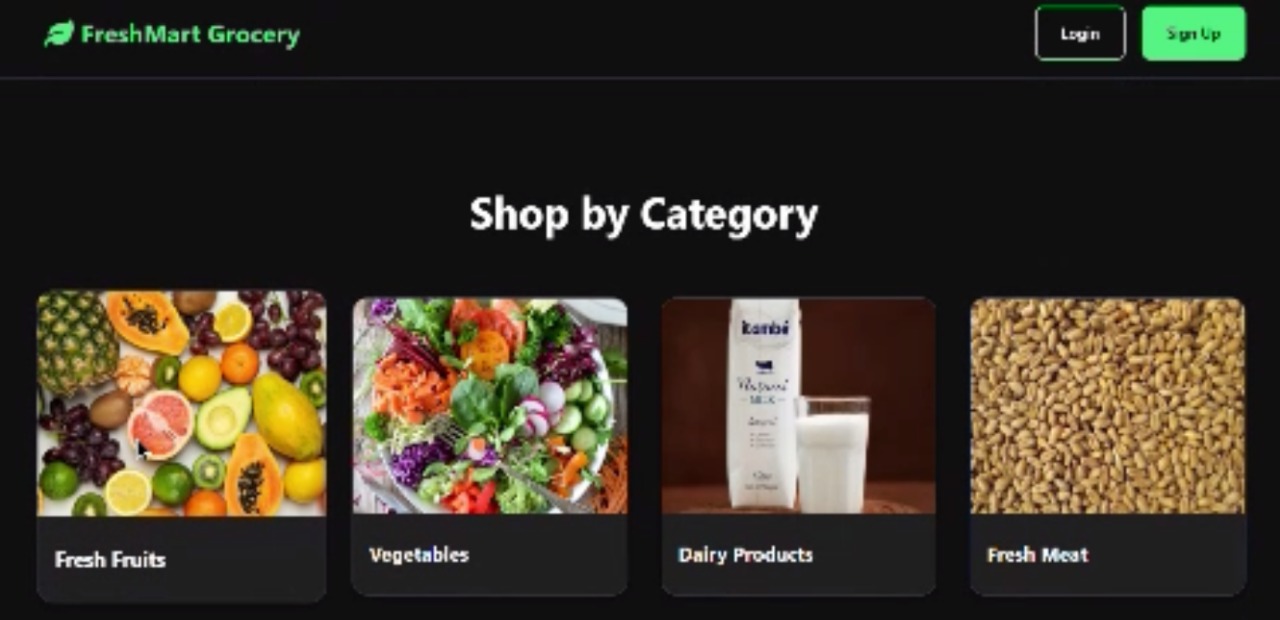
* **Authentication Process**
  + The app uses **JWT (JSON Web Tokens)** for secure, stateless authentication.
  + When a user logs in via the /auth/login endpoint, a **JWT token** is generated and sent back to the client.
  + This token contains encoded user information, including their unique ID and role (e.g., user, admin).
  + The token is stored on the client side (typically in localStorage or sessionStorage) and is attached to subsequent requests to protected routes.
* **Authorization Process**
* Role-based access control (RBAC) is enforced throughout the backend.
* Middleware functions validate the token using the secret key and check user roles.
  + Example: Admin-only routes such as /admin/product are accessible only if the token confirms an admin role.
* Unauthorized users attempting to access protected routes will receive a 403 Forbidden or 401 Unauthorized responses.

**9. User Interface:**

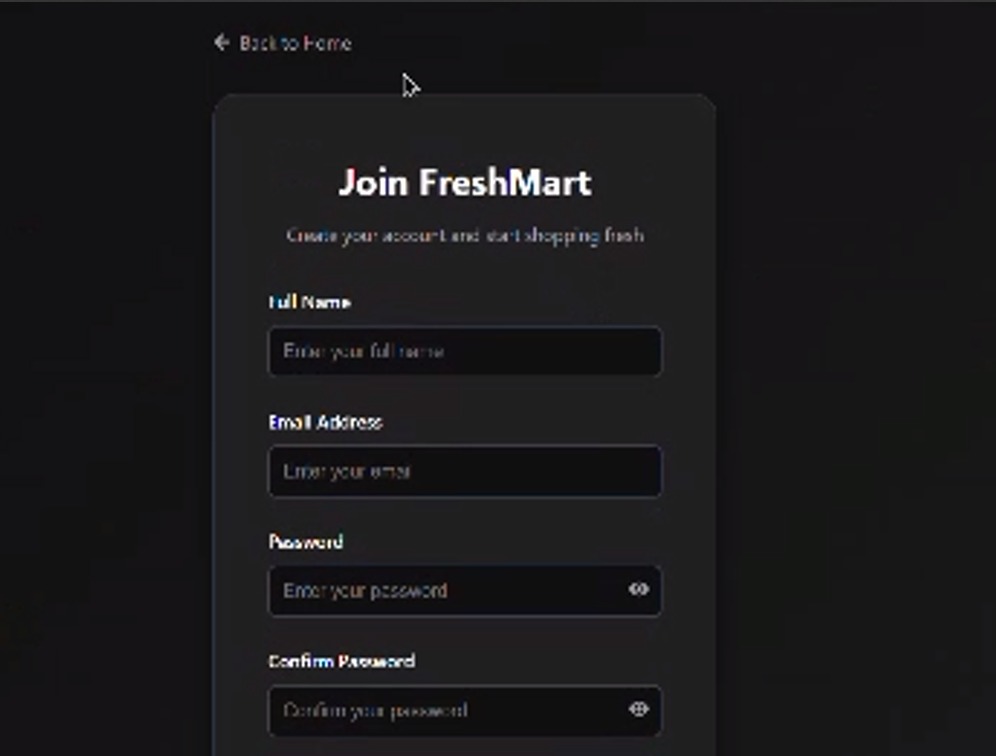
1.Home Page



2.User Home Page







**10. Testing:**

The ShopSmart Grocery Web App was tested using both manual and automated methods to ensure the reliability and stability of core features across the frontend and backend.

**Frontend Testing**

* Manual Testing
  + Angular components were manually tested using StackBlitz during early prototyping.
  + React components were tested by interacting with product listing, cart functionality, and navigation in the local browser.
  + Cross-browser compatibility and responsiveness were verified manually.
* Automated Testing (React)
  + Jest was used as the testing framework.
  + React Testing Library was used to test the behavior of individual components like ProductCard, CartItem, and CheckoutForm.
  + Focused on unit testing logic, rendering output, button clicks, and state changes.

**Backend Testing**

* Postman was used to test all RESTful APIs:
  + User registration and login
    - Product CRUD operations
  + Cart and order endpoints
* API responses, error messages, and authorization logic were validated using Postman collections.
* Validation Middleware was tested to ensure:
  + Required fields are provided
  + Only valid users/admins can access protected routes

**11. Screenshots or Demo:**

The Demo for the ShopSmart project is

https://drive.google.com/file/d/1jwsiSyYI3uYxVtfbbDpRaEBpvHh-r0Ww/view?usp=sharing

**12. Known Issues:**

**1. Angular Preview Issues (StackBlitz)**

* Certain Angular dependencies like @angular/platform-browser-dynamic may not function fully in StackBlitz.
* Real-time previews sometimes fail to load external images or components correctly due to sandbox limitations.

**2. Image Load Delays**

* Some product images (from external sources like CDN) may take time to load or fail intermittently, especially on slow connections.
* A fallback/default image is not yet implemented for broken image links.

**3. No Payment Gateway Integration**

* Checkout functionality is simulated; real payment gateway (e.g., Razorpay/Stripe) integration is not yet added.
* Orders are marked as confirmed without actual transaction validation.

**4. Limited Admin Dashboard Validation**

* While product and order management work, error handling for invalid data or admin-side form validations is minimal.
* Form submission with empty or duplicate values might not be handled gracefully in all cases.

**5. Session Expiry Handling**

* Token expiration is not visually indicated to users.
* If a JWT expires, the app may silently fail to fetch protected data unless the user manually logs in again.

**13. Future Enhancements:**

The ShopSmart Grocery Web App has a strong foundational architecture, but several additional features and improvements can be implemented in the future to enhance functionality, user experience, and scalability.

**1. Payment Gateway Integration**

* Integrate real-time payment gateways such as **Razorpay**, **Stripe**, or **PayPal** to handle secure online transactions.
* Enable support for UPI, credit/debit cards, and net banking.

**2. Delivery Agent Panel**

* Introduce a separate module or login for delivery agents.
* Features could include delivery status updates, assigned orders, and route tracking.

**3. Wishlist and Saved Items**

* Allow users to add products to a personal **wishlist** for future purchases.
* Enable a “Save for later” feature in the shopping cart.

**4. Notification System**

* Implement push/email/SMS notifications for order confirmation, delivery updates, and promotional offers.

**5. Mobile App Version**

* Build a companion **mobile app** using **React Native** or **Flutter** for both Android and iOS platforms.
* Ensure cross-device synchronization of cart and orders.

**6. Product Recommendation Engine**

* Use AI/ML to suggest products based on user behavior and past orders.
* Improve personalization to boost user engagement and sales.

**7. Multi-language and Currency Support**

* Localize the app to support multiple languages (e.g., Telugu, Hindi, Tamil).
* Allow currency switching for international scalability.

**8. Admin Reports and Analytics**

* Add visual dashboards with charts and graphs for sales, orders, user growth, and inventory trends.
* Downloadable PDF/CSV reports for business use.