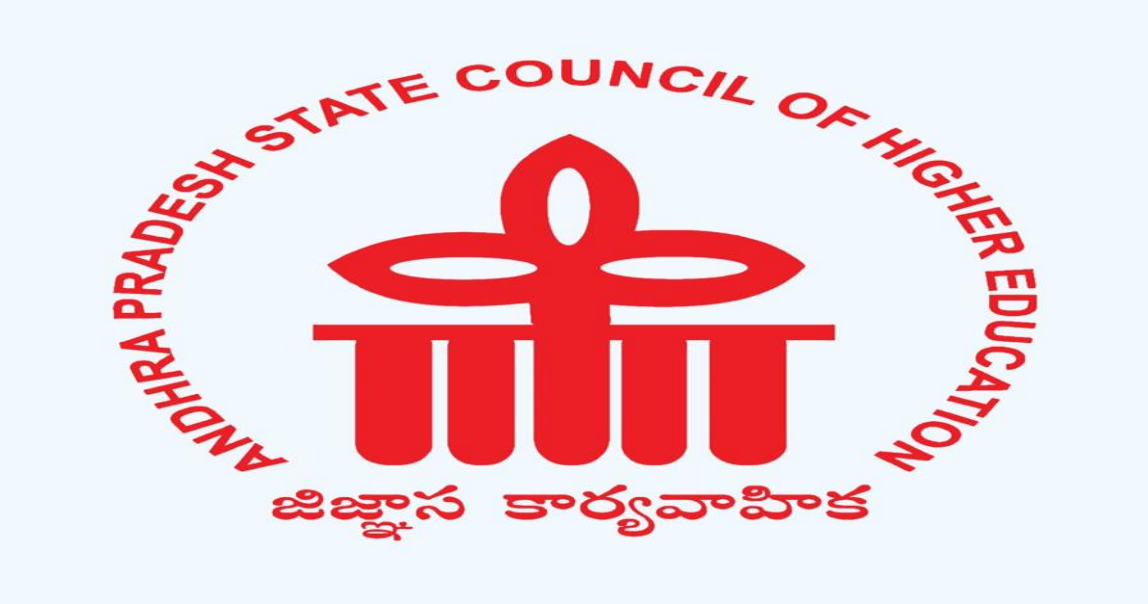
FULL STACK DEVELOPMENT WITH(MERN)



1.Introduction:

- **Project Title:** Shopsmart : Your Digital Grocery Store Experience
- **Team Members:**

Team Id :SBAP0042129

Team Leader :Dhulipalla Radhika

Team Member :Bangaru Hema

Team Member :Thiragati Hruday Raj

Team Member : Praneeth Raj

2.Project Overview:

- **Purpose**

  1.**Provide Online Shopping Convenience**

  Customers can buy groceries from home anytime without visiting a physical store.

  2.Save Time and Effort

  Users can quickly search, filter, and order products, avoiding long queues and travel.

  3.Expand Business Reach

  The website helps the store reach more customers beyond its local area.

  4.24/7 Availability

  Unlike physical stores, the website is always accessible for browsing and ordering.

  5.**Organize and Display Products Clearly**

  Categories and product pages make it easy to explore the entire product range.

- **Goals:**

1. Customer Convenience: Allow users to easily view products, compare prices, and place orders at any time from any device.
2. Inventory Management: Help store owners manage stock levels, track product availability, and receive alerts for low inventory.
3. Order Tracking: Enable customers to track their orders in real-time and receive timely updates on delivery status.
4. Secure Payments: Integrate safe and reliable payment gateways to facilitate smooth transactions.
5. User Accounts: Provide features like account creation, order history, saved items, and personalized recommendations.
6. Admin Dashboard: Allow administrators to manage products, categories, customer data, and generate sales reports.

- **Features:**

1.User Registration & Login
Allows customers to create accounts and securely log in to access personalized services.

2.Product Search & Categories
Users can easily search for items or browse by product categories like fruits, vegetables, dairy, etc.

3.Detailed Product Pages
Each product shows name, image, price, description, quantity options, and customer reviews.

4.Shopping Cart System
Users can add, remove, or update quantities of products before proceeding to checkout.

5.Secure Payment Gateway
Supports online payments through credit/debit cards, UPI, wallets, or Cash on Delivery (COD).

6.Order Tracking & History
Customers can view the status of current orders and check their past purchase history.

7.Admin Dashboard
Admins can manage inventory, products, categories, and view order statistics.

8.Inventory Management
Automatically updates stock levels and alerts the admin when products are low or out of stock.

9.Discounts & Coupon Codes
Users can apply promo codes during checkout to receive discounts.

10.Mobile-Friendly Design
The website works smoothly on smartphones, tablets, and desktops for user convenience.

3.Technical Architecture:

1. Frontend (User Interface)

- Built with React.js or Next.js for SEO and performance.
- Responsive UI with pages for product listings, search, cart, and checkout.
- Admin panel for managing products and orders.

2. Backend (API & Business Logic)

- RESTful or GraphQL APIs using Node.js (Express/NestJS) or Django/Spring Boot.
- Handles authentication, cart logic, payments, and order management.
- Modular architecture (monolith or microservices).

3. Database & Storage

- PostgreSQL for structured data (users, products, orders).
- Redis for session caching and fast cart operations.
- Cloud storage (AWS S3) for product images and receipts.

4. Payments & Notifications

- Integrate Stripe or PayPal for secure transactions.
- Send order updates via email/SMS using SendGrid, Twilio, or Firebase Cloud Messaging.

5. Deployment & Infrastructure

- Containerized with Docker, managed via Kubernetes (for scalability).
- Deployed on AWS/GCP/Azure with CI/CD pipelines.

4.Setup Instructions:

❖ PRE REQUISITES:

To develop a full-stack Ecommerce App for Furniture Tool using React js, Node.js,Express js and MongoDB, there are several prerequisites you should consider. Here are the key prerequisites for developing such an application:

Node.js and npm: Install Node.js, which includes npm (Node Package Manager), on your development machine. Node.js is required to run JavaScript on the server side.

- Download: https://nodejs.org/en/download/
- Installation instructions:https://nodejs.org/en/download/package-manager/

MongoDB: Set up a MongoDB database to store hotel and booking information. Install MongoDB locally or use a cloud-based MongoDB service.

- Download: https://www.mongodb.com/try/download/community
- Installation instructions:https://docs.mongodb.com/manual/installation/

Express.js: Express.js is a web application framework for Node.js. Install Express.js to handle server-side routing, middleware, and API development.

- Installation: Open your command prompt or terminal and run the following command: **npm install express**

React js: React is a JavaScript library for building client-side applications.
And Creating Single Page Web-Appliaction
You have successfully set up React on your machine and created a new React project. You can now start building your app by modifying the generated project files in the src directory.

Please note that these instructions provide a basic setup for React. You can explore more ad- vanced configurations and features by referring to the official React documentation: https://react.dev/

HTML, CSS, and JavaScript: Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

Database Connectivity: Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Node.js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations.

Front-end Library: Utilize React to build the user-facing part of the application, including products listings, booking forms, and user interfaces for the admin dashboard.

Version Control: Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.

• Git: Download and installation instructions can be found at: https://git-scm.com/downloads

Development Environment: Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

• Visual Studio Code: Download from https://code.visualstudio.com/download
• Sublime Text: Download from https://www.sublimetext.com/download
• WebStorm: Download from https://www.jetbrains.com/webstorm/download

5.Project Structure :

❖ Product Catalog

1.The store displays grocery items with images, prices, and categories.
2.Users can search, filter, and view product details.
3.This is the core of the customer-facing storefront.

❖ Shopping Cart & Checkout

1.Customers can add items to their cart and proceed to secure checkout.
2.Includes address input, payment gateway integration, and order confirm.
3.Cart data is stored per user or session.

❖ Product Management (Admin)

1.Admins can add, edit, or delete products from an admin dashboard.
2.They manage pricing, stock, categories, and product availability.
3.Only authorized users have access to these tools.

❖ Inventory Control

1.The backend tracks stock levels and updates automatically after sales.

2.Admins get alerts for low inventory and can restock items.

3.Helps prevent overselling and ensures availability.

❖ Secure Access Control

1.based authentication separates customer and admin actions.

2.Customers access the storefront; admins manage the system.

3.JWT or OAuth is used for secure and scalable auth.

6. Running Application:

❖ Project Setup and Configuration:

1. Install required tools and software:

- Node.js.
- MongoDB.
- Create-react-app.

2. Create project folders and files:

- Client folders.
- Server folders.

3. Install Packages:

Frontend npm Packages

- Axios.
- React-Router –dom.
- Bootstrap.
- React-Bootstrap.
- React-icons.

Backend npm Packages

- Express.
- Mongoose.
- Cors.

❖Frontend Development:

1. Setup React Application:

- Create React application.
- Configure Routing.
- Install required libraries.

2. Design UI components:

- Create Components.
- Implement layout and styling.
- Add navigation.

3. Implement frontend logic:

- Integration with API endpoints.
- Implement data binding.

```
<!-- Mobile Menu -->
<div id="mobile-menu" class="mobile-menu fixed inset-y-0 left-full w-64 bg-white shadow-
    <div class="flex justify-between items-center mb-8">
        <span class="text-xl font-bold text-green-600">FreshMart</span>
        <button id="close-menu-button" class="text-gray-700">
            <i class="fas fa-times"></i>
        </button>
    </div>
    <div class="mb-4">
        <input type="text" placeholder="Search products..." class="px-4 py-2 rounded-ful
    </div>
    <div class="space-y-4">
        <a href="#" class="block text-gray-700 hover:text-green-600">Home</a>
        <a href="#products" class="block text-gray-700 hover:text-green-600">Shop</a>
        <a href="#categories" class="block text-gray-700 hover:text-green-600">Categorie
        <a href="#about" class="block text-gray-700 hover:text-green-600">About</a>
        <a href="#contact" class="block text-gray-700 hover:text-green-600">Contact</a>
        <!-- <div class="pt-4 border-t border-gray-200">
            <a href="#" class="block text-gray-700 hover:text-green-600">Login</a>
            <a href="#" class="block text-gray-700 hover:text-green-600">Register</a>
        </div> -->
    </div>
</div>
```

❖ Backend Development:

- **Setup express server**
    1. Create index.js file in the server (backend folder).
    2. Create a .env file and define port number to access it globally.
    3. Configure the server by adding cors, body-parser.

- **User Authentication:**
    - Create routes and middleware for user registration, login, and logout.
    - Set up authentication middleware to protect routes that require user authentication.

- **Define API Routes:**
    - Create separate route files for different API functionalities such as users orders, and authentication.
    - Define the necessary routes for listing products, handling user registration and login,managing orders, etc.
    - Implement route handlers using Express.js to handle requests and interact with the database.

- **Implement Data Models:**
    - Define Mongoose schemas for the different data entities like products, users, and orders.
    - Create corresponding Mongoose models to interact with the MongoDB database.
    - Implement CRUD operations (Create, Read, Update, Delete) for each model to perform database operations.

- **User Authentication:**
    - Create routes and middleware for user registration, login, and logout.
    - Set up authentication middleware to protect routes that require user authentication.

- **Error Handling:**
    - Implement error handling middleware to catch and handle any errors that occur during the API requests.

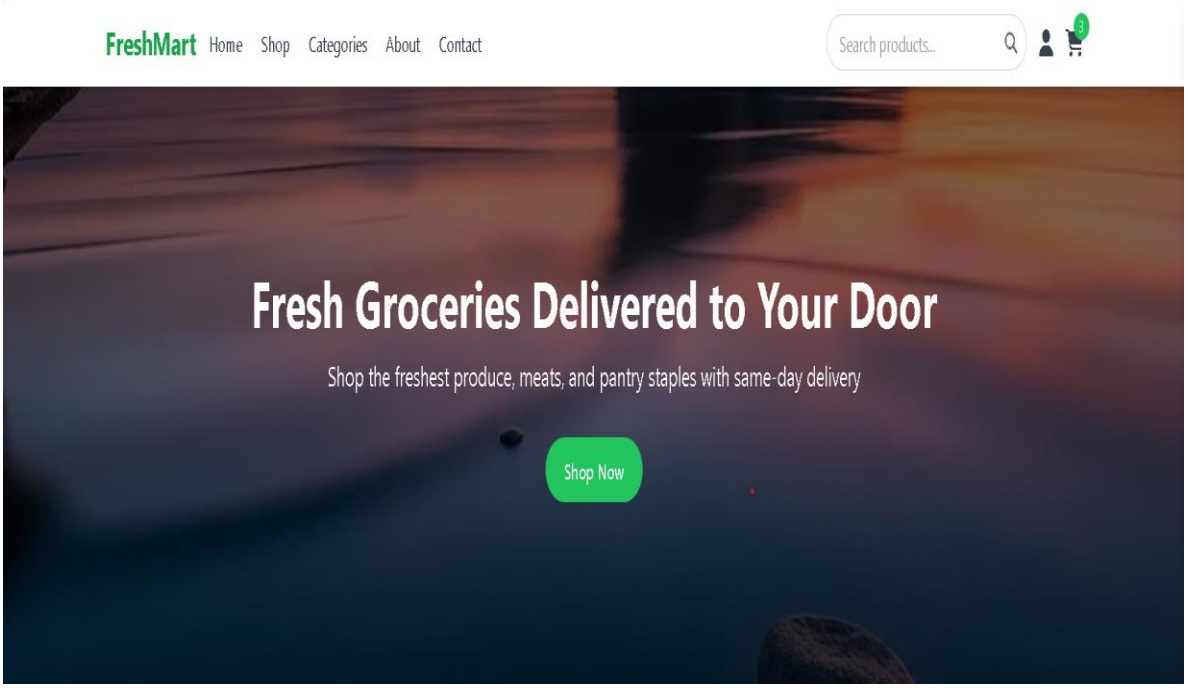
- Return appropriate error responses with relevant error messages and HTTP status codes.

```
<!-- Contact Section -->
<section id="contact" class="py-12 bg-white">
    <div class="container mx-auto px-4">
        <div class="flex flex-col md:flex-row gap-8">
            <div class="md:w-1/2">
                <h2 class="text-3xl font-bold mb-6">Contact Us</h2>
                <p class="text-gray-600 mb-6">Have questions or feedback? We'd love to h

                <div class="space-y-4 mb-8">
                    <div class="flex items-start">
                        <div class="bg-green-100 p-3 rounded-full mr-4">
                            <i class="fas fa-map-marker-alt text-green-600"></i>
                        </div>
                        <div>
                            <h3 class="font-bold mb-1">Address</h3>
                            <p class="text-gray-600">123 Fresh Street, Greenville, CA 12
                        </div>
                    </div>

                    <div class="flex items-start">
                        <div class="bg-green-100 p-3 rounded-full mr-4">
                            <i class="fas fa-phone-alt text-green-600"></i>
                        </div>
                        <div>
                            <h3 class="font-bold mb-1">Phone</h3>
```

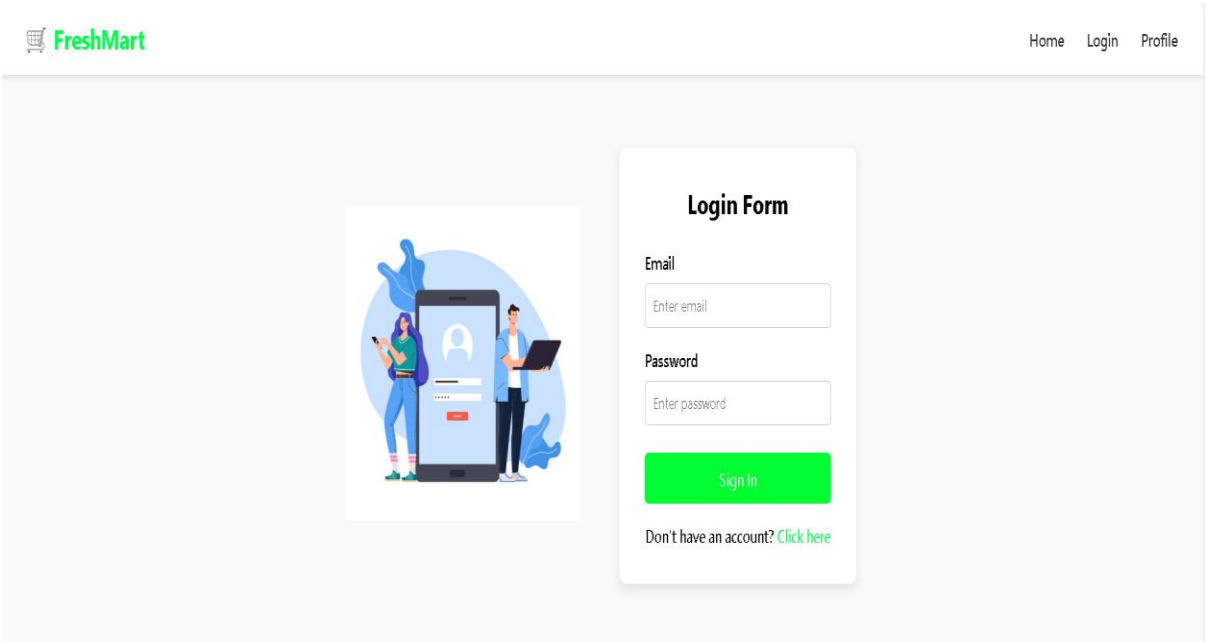
❖ Project Implementation:

Finally, after finishing coding the projects we run the whole project to test it's working process and look for bugs. Now, let's have a final look at the working of our Darshan Ease.

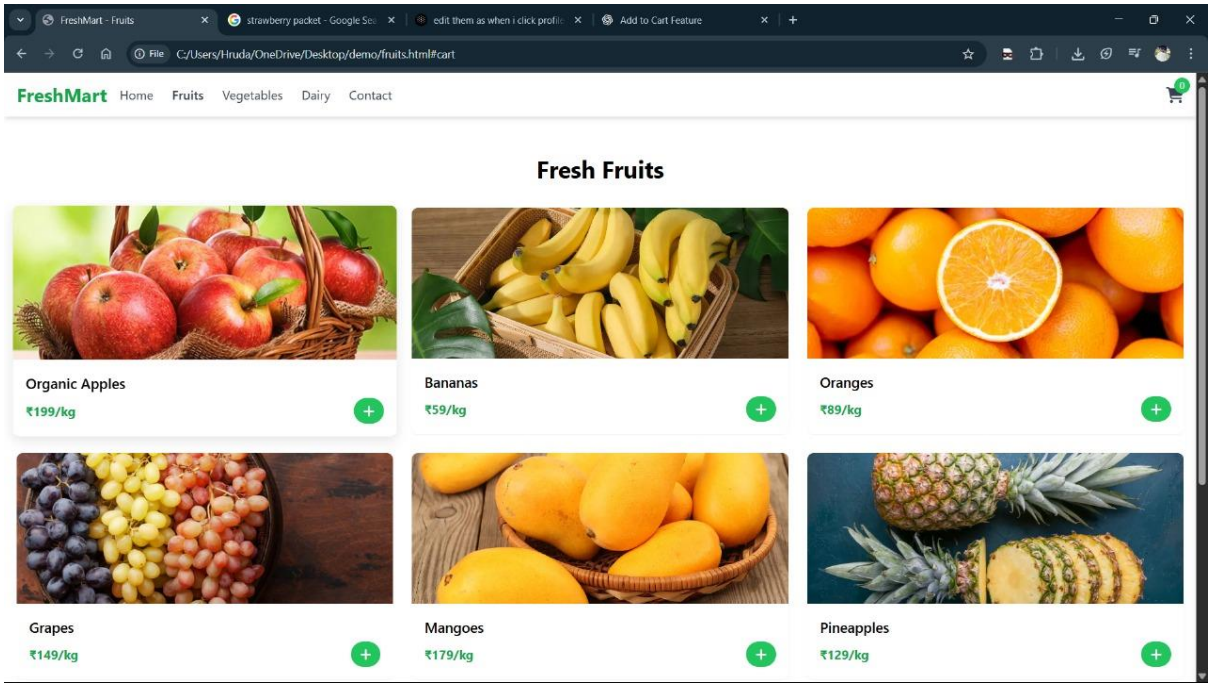
Landing page:-



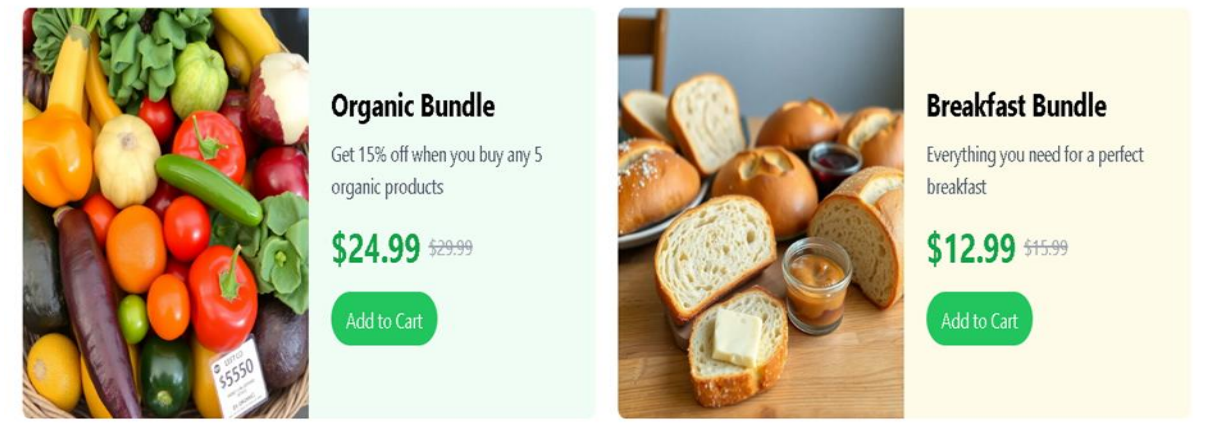
Login Page:-



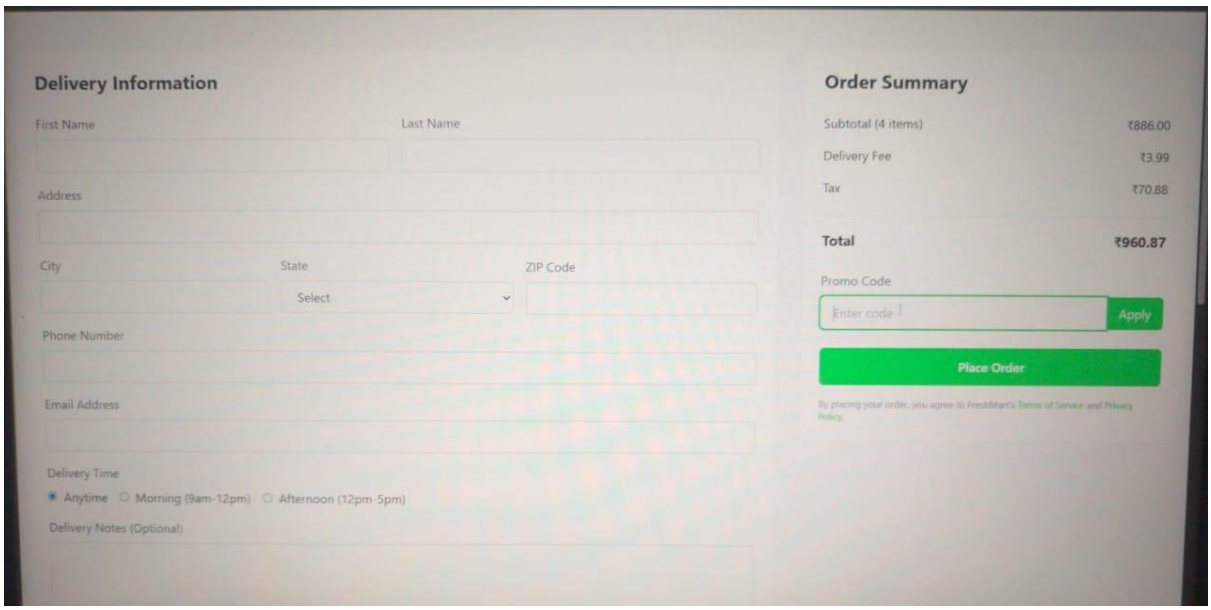
ItemsPage:-



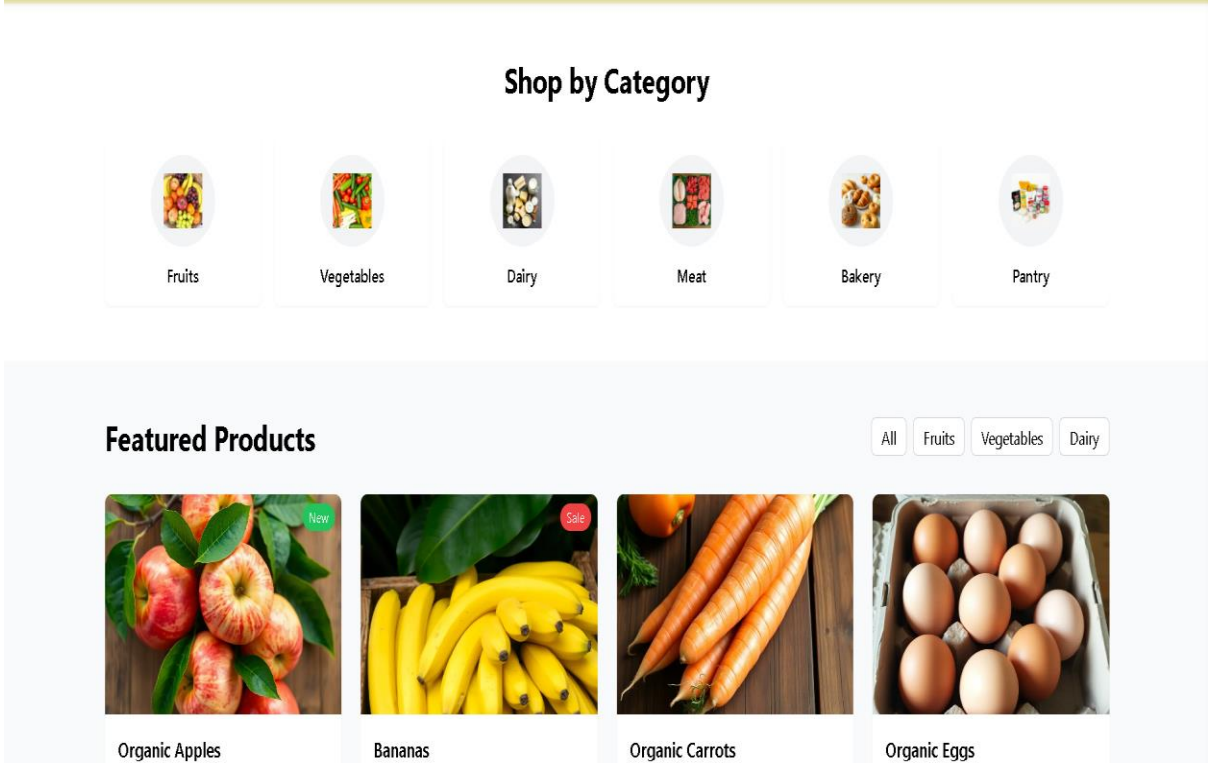
My Cart:-



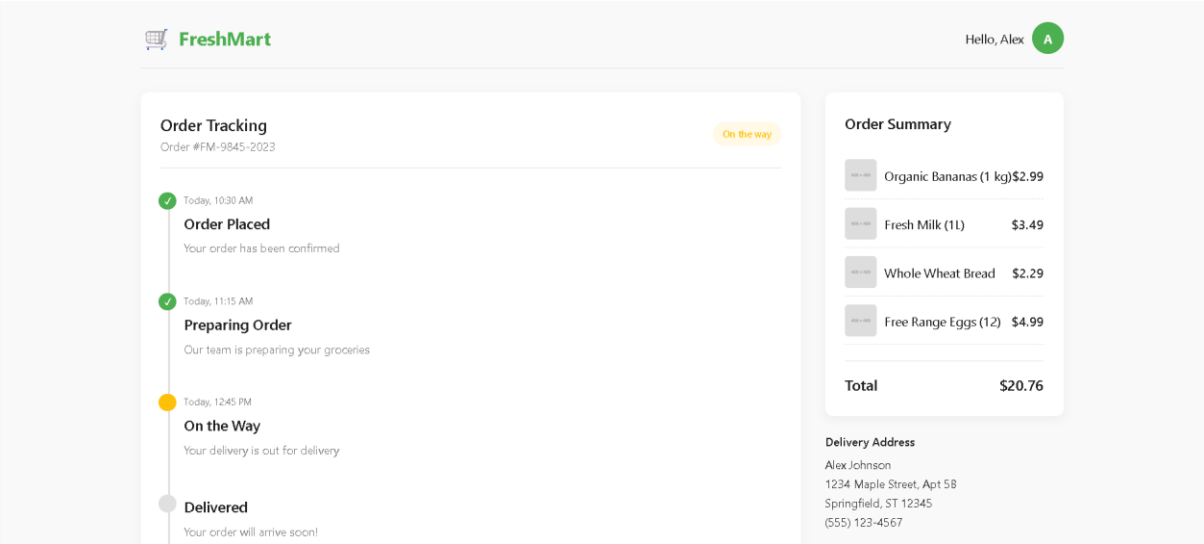
Place Order Page:-



Users Page:-



Delivery tracking:



8.Conclusion:

The grocery store website serves as a powerful digital solution to modernize the traditional shopping experience. It offers customers a fast, convenient, and reliable way to purchase essential groceries from the comfort of their homes. With features like product browsing, secure payments, order tracking, and admin management, the platform not only enhances customer satisfaction but also boosts business efficiency and reach.