



# **GROCERY APP USING MERN STACK**

**A PROJECT REPORT**

*Submitted by*

SURYAPRAKASH R - 412021205004

THAMODHARAN N - 412021104701

BARATH S - 412021104001

**BACHELOR OF ENGINEERING**

**IN**

**SEVENTH SEMESTER**

**COMPUTER SCIENCE AND ENGINEERING  
AND  
INFORMATION TECHNOLOGY**

**SRI KRISHNA ENGINEERING COLLEGE**

<b>S.NO</b>	<b>TABLE OF CONTENT</b>	<b>PAGE.NO</b>
<b>1.</b>	<b>INTRODUCTION</b>	<b>1</b>
<b>2.</b>	<b>PROJECT OVERVIEW</b>	<b>2</b>
<b>3.</b>	<b>ARCHITECTURE</b>	<b>2</b>
<b>4.</b>	<b>SETUP INSTRUCTIONS</b>	<b>3</b>
<b>5.</b>	<b>FOLDER STRUCTURE</b>	<b>3</b>
<b>6.</b>	<b>RUNNING THE APPLICATION</b>	<b>4</b>
<b>7.</b>	<b>API DOCUMENTATION</b>	<b>5</b>
<b>8.</b>	<b>AUTHENTICATION</b>	<b>5</b>
<b>9.</b>	<b>USER INTERFACE</b>	<b>5</b>
<b>10.</b>	<b>TESTING</b>	<b>6</b>
<b>11.</b>	<b>DEMO</b>	<b>6</b>
<b>12.</b>	<b>ISSUES AND ENHANCEMENTS</b>	<b>9</b>
<b>13.</b>	<b>CONCLUSION</b>	<b>10</b>

# **GROCERY APP USING MERN STACK**

## **ABSTRACT**

The MERN Full-Stack Grocery Website project aims to simplify and enhance the experience of online grocery shopping. By leveraging the power of the MERN stack (MongoDB, Express.js, React, and Node.js), the platform provides users with an intuitive and efficient interface to browse a diverse selection of grocery products, add desired items to their shopping cart, and place orders for either home delivery or in-store pickup. The system ensures a seamless and responsive user experience, catering to both desktop and mobile users.

Additionally, the project features an admin panel designed to streamline the management of essential backend operations, including inventory control, order processing, and user account management. This admin interface allows administrators to efficiently update product details, monitor order statuses, and oversee user activities, ensuring smooth operational flow. The combination of user-friendly design and robust backend functionality aims to provide an optimal grocery shopping and management solution.

## **INTRODUCTION**

The MERN Full-Stack Grocery Website project is designed to facilitate online grocery shopping. It provides customers with a convenient platform to browse a wide range of grocery products, add items to their cart, and place orders for home delivery or pickup. The project also includes an admin panel for efficient management of inventory, orders, and user accounts.

## PROJECT OVERVIEW

The grocery website is divided into two main sections: the user interface for

customers and the admin dashboard for site administrators. Key features include.

**User Registration and Login:** Secure authentication using JWT.

**Product Catalog:** Browsing and searching for grocery items by category or keyword.

**Shopping Cart:** Users can add, remove, and update quantities of items.

**Order Placement and Management:** Users can view order history, and admins can manage orders and inventory.

**Admin Features:** Adding, updating, and deleting products, and viewing user and order information.

## ARCHITECTURE

The project architecture is based on the MEAN stack

**Frontend:** Angular for building a dynamic, responsive user interface.

**Backend:** Node.js and Express.js to handle server-side logic and APIs.

**Database:** MongoDB for efficient data management and storage.

**Data Flow:** Communication between the frontend and backend happens via RESTful API calls, with data transmitted in JSON format.

### Architectural Flow

1. **Frontend (Client-Side):** Angular handles user interactions and sends requests to the backend.
2. **Backend (Server-Side):** Node.js and Express.js process requests, interact with MongoDB, and return responses.
3. **Database:** MongoDB stores information about products, users, and orders.

## SETUP INSTRUCTION

Follow these steps to set up the project on your local machine:

Prerequisites

Node.js and npm installed.

MongoDB installed and running locally or hosted on a cloud service like MongoDB Atlas.

Setup Steps

**Clone the repository:**

```
git clone <repository-url>
```

```
cd <repository-name>
```

**Install dependencies:**

**Backend:**

```
cd backend
```

```
npm install
```

**Frontend:**

```
cd frontend
```

```
npm install
```

**Configure environment variables:**

Create a .env file in the backend directory with values for the database URL, JWT secret, and other configurations.

**Start MongoDB:**

```
mongodb
```

**Run the application:**

Backend: In the backend folder, npm run dev.

Frontend: In the frontend folder, run ng serve.

## FOLDER STRUCTURE

The project is organized as follows:

## Backend (Node.js and Express.js)

backend/

```
|
|
|— controllers/      # Handles business logic for routes
|— models/           # Mongoose schemas for products, users, and orders
|— routes/           # Defines API routes
|— middleware/       # Custom middleware for authentication
|— config/           # Database configuration and other settings
|— .env              # Environment variables
|— app.js            # Main entry point for the server
```

## Frontend (Angular)

frontend/

```
|
|
|— src/
|   |— app/
|   |   |— components/  # UI components (navbar, footer, etc.)
|   |   |— services/    # API services for data fetching
|   |   |— pages/       # Page components (home, product, cart)
|   |   └— app.module.ts # Angular module configuration
|   └— assets/          # Static files (images, stylesheets)
|       └— index.html   # Main HTML template
└— angular.json         # Angular configuration
```

## RUNNING THE APPLICATION

**1. Start the backend:** Navigate to the backend folder and run:

```
npm run dev
```

**2. Start the frontend:** Navigate to the frontend folder and run:

```
npm start
```

**3. Access the application:** Open your web browser and go to <http://localhost:4200>.

## API DOCUMENTATION

The backend provides the following RESTful API endpoints:

### User APIs

POST /api/users/register: Register a new user.

POST /api/users/login: Authenticate a user and return a JWT.

GET /api/users/profile: Get user profile (authentication required).

### Product APIs

GET /api/products: Fetch all grocery items.

GET /api/products/:id: Retrieve details of a specific item.

POST /api/products: Add a new product (admin only).

PUT /api/products/:id: Update product information (admin only).

DELETE /api/products/:id: Remove a product from the catalog (admin only).

### Order APIs

POST /api/orders: Place a new order.

GET /api/orders/user/:userId: Retrieve orders for a specific user.

PUT /api/orders/:id: Update order status (admin only).

## AUTHENTICATION

JWT: Used for secure user authentication.

Role-Based Access Control: Ensures that only admins can perform certain actions like managing products.

## USER INTERFACE

The UI is built using Angular, with Bootstrap for styling to ensure a responsive design.

**Key features:**

Homepage: Displays featured products and categories.

Product List and Search: Users can view and search for grocery items.

Product Details Page: Detailed information about selected items.

Shopping Cart: Users can add, update, or remove items and proceed to checkout.

Admin Dashboard: Manage products, view users, and process orders.

## TESTING

**Unit Testing:** Jasmine and Karma are used for testing Angular components.

**API Testing:** Postman is used to test backend API endpoints, ensuring that requests and responses are handled correctly and align with expected outcomes.

**End-to-End Testing:** Manual testing is conducted to ensure the application functions correctly across different devices and browsers.

## DEMO

A live demo of the grocery website is available at:

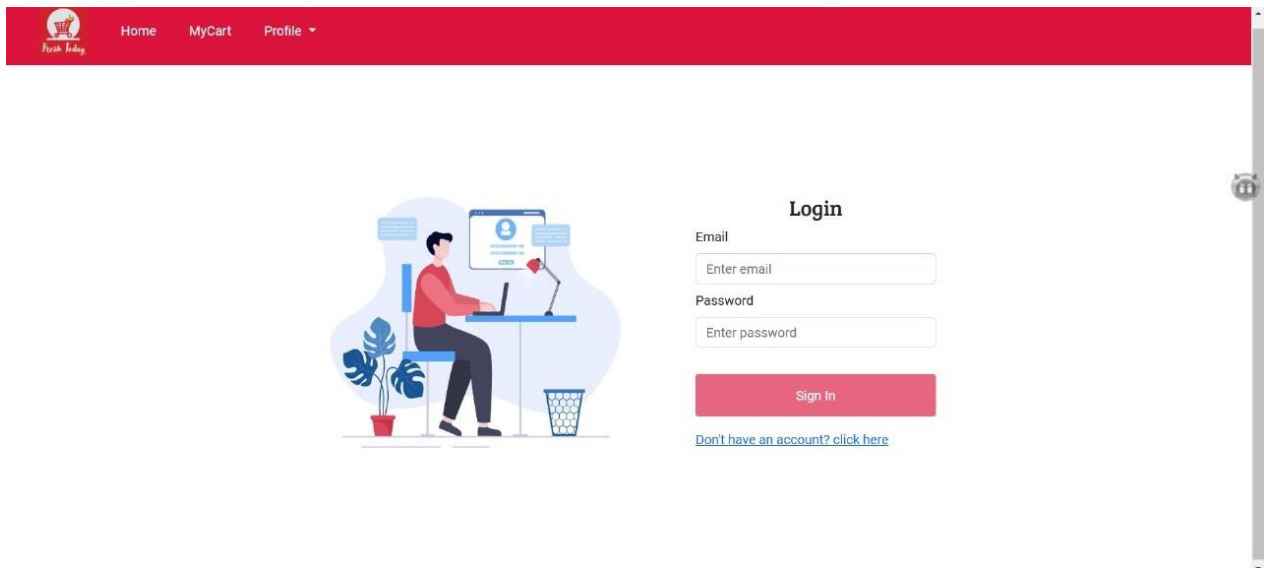
<https://drive.google.com/file/d/1cEgJCxVSN21kh5WrcDyo09qmebWcwnNp/view?usp=sharing>

### Steps to Test the Demo

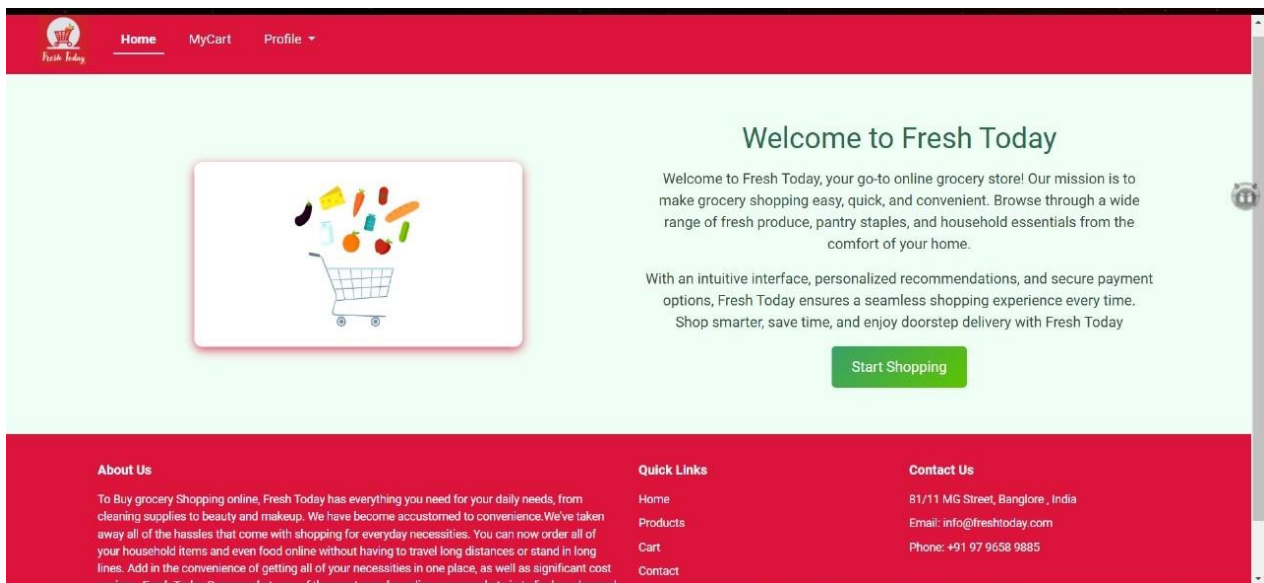
1. Register or log in as a user.
2. Browse the grocery catalog and add items to the cart.
3. Proceed to checkout and place an order.
4. Admin Access: Use admin credentials to manage products and view orders.

### Login Page:






## Landing page:




## Home page:


[Home](#)
[MyCart](#)
[Profile](#)

**Filters**

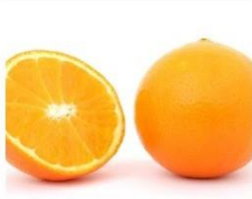
**Category**
☐ Fruits
 ☐ Vegetables
 ☐ Dairy
 ☐ Meat
 ☐ Bakery
 ☐ Beverages
 ☐ Others

**Products**




**Apple**  
Apples are sweet and crisp fruits that come in a variety of colors, including red, green, and yellow. They are rich in fiber and vitamin C and make a healthy snack  
price: 60/kg/l/g

4 ★




**orange**  
Oranges are juicy and tangy citrus fruits known for their high vitamin C content. They are typically peeled and eaten fresh or used to make orange juice  
price: 45/kg/l/g

4.2 ★



**strawberries**  
Strawberries are sweet and fragrant berries with a bright red color. They are rich in antioxidants, vitamin C, and fiber  
price: 80/kg/l/g

3.5 ★



**Banana**  
Bananas are a soft, sweet fruit with a creamy texture, covered by a yellow peel when ripe.  
price: 25/kg/l/g

5 ★

[Add](#) [Buy Now](#)

## Cart page:


[Home](#)
[MyCart](#)
[Profile](#)

**My Cart**



**Apple**  
Price: 60 /-

[Buy](#)


×



**Carrot**  
Price: 45 /-

[Buy](#)

×

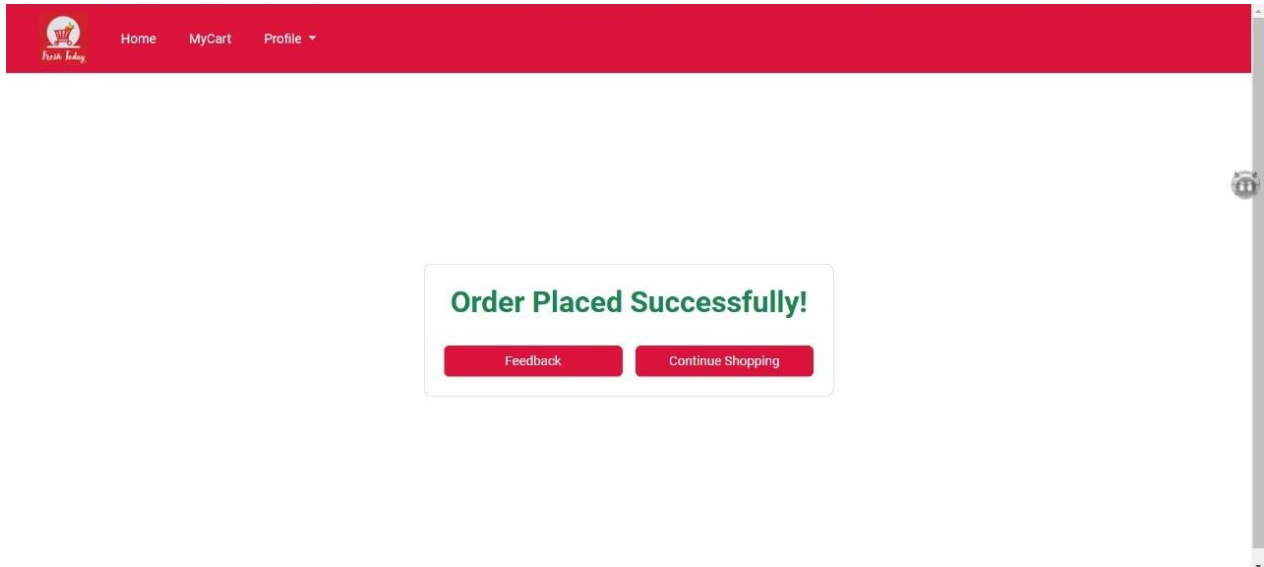


**Broccoli**  
Price: 55 /-

[Buy](#)

×

## Order successful page:



## ISSUES AND FUTURE ENHANCEMENTS

### Current Issues

**Performance:** The site may experience slow loading times with a large number of products.

**Limited Payment Options:** Only basic payment functionality is implemented.

### Future Enhancements

**Payment Integration:** Implement secure payment gateways like Stripe or PayPal.

**Product Recommendations:** Use machine learning to suggest products to users.

**Wishlist Feature:** Allow users to save items for later.

**Order Tracking:** Enable users to track the status of their orders in real-time.

**Enhanced Filtering and Sorting:** Provide more options for users to filter and sort grocery items.

## **CONCLUSION**

The MERN Full-Stack Grocery Website project successfully demonstrates the implementation of a robust and user-friendly platform for online grocery shopping. By leveraging Angular for a dynamic and responsive frontend, Node.js and Express.js for efficient backend services, and MongoDB for scalable data management, the project achieves a seamless integration between all components of the MEAN stack. While there are opportunities for optimization and expansion, such as adding advanced payment solutions and personalized recommendations, the current application provides a solid foundation for meeting user needs and simplifying grocery shopping experiences. Future enhancements and continuous improvements will ensure the platform remains competitive and aligned with evolving user expectations and technological advancements.

<b>S.NO</b>	<b>TABLE OF CONTENT</b>	<b>PAGE.NO</b>
<b>1.</b>	<b>INTRODUCTION</b>	<b>1</b>
<b>2.</b>	<b>PROJECT OVERVIEW</b>	<b>2</b>
<b>3.</b>	<b>ARCHITECTURE</b>	<b>2</b>
<b>4.</b>	<b>SETUP INSTRUCTIONS</b>	<b>3</b>
<b>5.</b>	<b>FOLDER STRUCTURE</b>	<b>3</b>
<b>6.</b>	<b>RUNNING THE APPLICATION</b>	<b>4</b>
<b>7.</b>	<b>API DOCUMENTATION</b>	<b>5</b>
<b>8.</b>	<b>AUTHENTICATION</b>	<b>5</b>
<b>9.</b>	<b>USER INTERFACE</b>	<b>5</b>
<b>10.</b>	<b>TESTING</b>	<b>6</b>
<b>11.</b>	<b>DEMO</b>	<b>6</b>
<b>12.</b>	<b>ISSUES AND ENHANCEMENTS</b>	<b>9</b>
<b>13.</b>	<b>CONCLUSION</b>	<b>10</b>

# **GROCERY APP USING MERN STACK**

## **ABSTRACT**

The MERN Full-Stack Grocery Website project aims to simplify and enhance the experience of online grocery shopping. By leveraging the power of the MERN stack (MongoDB, Express.js, React, and Node.js), the platform provides users with an intuitive and efficient interface to browse a diverse selection of grocery products, add desired items to their shopping cart, and place orders for either home delivery or in-store pickup. The system ensures a seamless and responsive user experience, catering to both desktop and mobile users.

Additionally, the project features an admin panel designed to streamline the management of essential backend operations, including inventory control, order processing, and user account management. This admin interface allows administrators to efficiently update product details, monitor order statuses, and oversee user activities, ensuring smooth operational flow. The combination of user-friendly design and robust backend functionality aims to provide an optimal grocery shopping and management solution.

## **INTRODUCTION**

The MERN Full-Stack Grocery Website project is designed to facilitate online grocery shopping. It provides customers with a convenient platform to browse a wide range of grocery products, add items to their cart, and place orders for home delivery or pickup. The project also includes an admin panel for efficient management of inventory, orders, and user accounts.

## PROJECT OVERVIEW

The grocery website is divided into two main sections: the user interface for

customers and the admin dashboard for site administrators. Key features include.

**User Registration and Login:** Secure authentication using JWT.

**Product Catalog:** Browsing and searching for grocery items by category or keyword.

**Shopping Cart:** Users can add, remove, and update quantities of items.

**Order Placement and Management:** Users can view order history, and admins can manage orders and inventory.

**Admin Features:** Adding, updating, and deleting products, and viewing user and order information.

## ARCHITECTURE

The project architecture is based on the MEAN stack

**Frontend:** Angular for building a dynamic, responsive user interface.

**Backend:** Node.js and Express.js to handle server-side logic and APIs.

**Database:** MongoDB for efficient data management and storage.

**Data Flow:** Communication between the frontend and backend happens via RESTful API calls, with data transmitted in JSON format.

### Architectural Flow

1. **Frontend (Client-Side):** Angular handles user interactions and sends requests to the backend.
2. **Backend (Server-Side):** Node.js and Express.js process requests, interact with MongoDB, and return responses.
3. **Database:** MongoDB stores information about products, users, and orders.

## SETUP INSTRUCTION

Follow these steps to set up the project on your local machine:

Prerequisites

Node.js and npm installed.

MongoDB installed and running locally or hosted on a cloud service like MongoDB Atlas.

Setup Steps

**Clone the repository:**

```
git clone <repository-url>
```

```
cd <repository-name>
```

**Install dependencies:**

**Backend:**

```
cd backend
```

```
npm install
```

**Frontend:**

```
cd frontend
```

```
npm install
```

**Configure environment variables:**

Create a .env file in the backend directory with values for the database URL, JWT secret, and other configurations.

**Start MongoDB:**

```
mongodb
```

**Run the application:**

Backend: In the backend folder, npm run dev.

Frontend: In the frontend folder, run ng serve.

## FOLDER STRUCTURE

The project is organized as follows:



## Backend (Node.js and Express.js)

backend/

```
|
|
|— controllers/      # Handles business logic for routes
|— models/          # Mongoose schemas for products, users, and orders
|— routes/          # Defines API routes
|— middleware/      # Custom middleware for authentication
|— config/          # Database configuration and other settings
|— .env             # Environment variables
|— app.js           # Main entry point for the server
```

## Frontend (Angular)

frontend/

```
|
|
|— src/
|   |— app/
|   |   |— components/  # UI components (navbar, footer, etc.)
|   |   |— services/    # API services for data fetching
|   |   |— pages/       # Page components (home, product, cart)
|   |   └— app.module.ts # Angular module configuration
|   └— assets/          # Static files (images, stylesheets)
|       └— index.html    # Main HTML template
└— angular.json          # Angular configuration
```

## RUNNING THE APPLICATION

**1. Start the backend:** Navigate to the backend folder and run:

```
npm run dev
```

**2. Start the frontend:** Navigate to the frontend folder and run:

```
npm start
```

**3. Access the application:** Open your web browser and go to <http://localhost:4200>.

## **API DOCUMENTATION**

The backend provides the following RESTful API endpoints:

### **User APIs**

POST /api/users/register: Register a new user.

POST /api/users/login: Authenticate a user and return a JWT.

GET /api/users/profile: Get user profile (authentication required).

### **Product APIs**

GET /api/products: Fetch all grocery items.

GET /api/products/:id: Retrieve details of a specific item.

POST /api/products: Add a new product (admin only).

PUT /api/products/:id: Update product information (admin only).

DELETE /api/products/:id: Remove a product from the catalog (admin only).

### **Order APIs**

POST /api/orders: Place a new order.

GET /api/orders/user/:userId: Retrieve orders for a specific user.

PUT /api/orders/:id: Update order status (admin only).

## **AUTHENTICATION**

JWT: Used for secure user authentication.

Role-Based Access Control: Ensures that only admins can perform certain actions like managing products.

## **USER INTERFACE**

The UI is built using Angular, with Bootstrap for styling to ensure a responsive design.

**Key features:**

Homepage: Displays featured products and categories.

Product List and Search: Users can view and search for grocery items.

Product Details Page: Detailed information about selected items.

Shopping Cart: Users can add, update, or remove items and proceed to checkout.

Admin Dashboard: Manage products, view users, and process orders.

## TESTING

**Unit Testing:** Jasmine and Karma are used for testing Angular components.

**API Testing:** Postman is used to test backend API endpoints, ensuring that requests and responses are handled correctly and align with expected outcomes.

**End-to-End Testing:** Manual testing is conducted to ensure the application functions correctly across different devices and browsers.

## DEMO

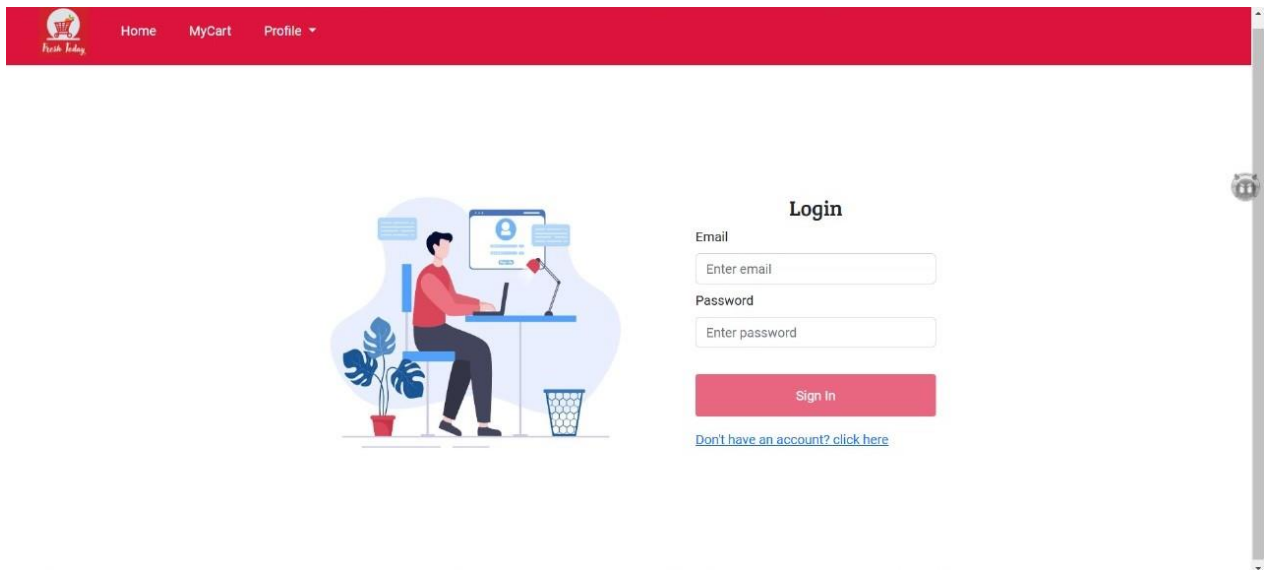
A live demo of the grocery website is available at:

<https://drive.google.com/file/d/1cEgJCxVSN21kh5WrcDyo09qmebWcwnNp/view?usp=sharing>

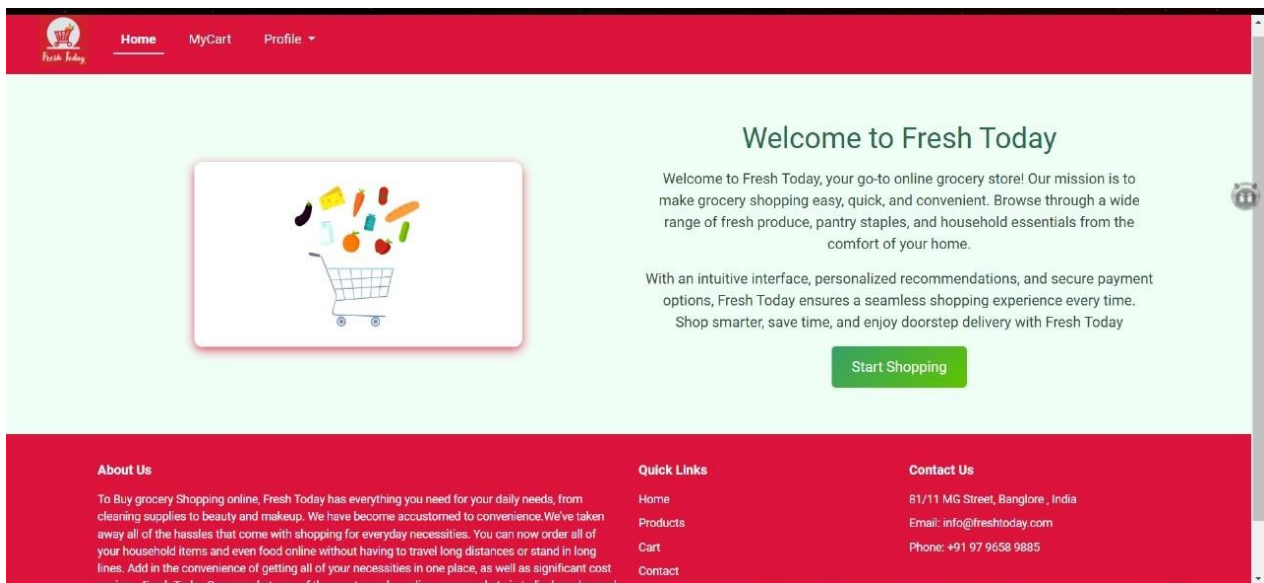
### Steps to Test the Demo

1. Register or log in as a user.
2. Browse the grocery catalog and add items to the cart.
3. Proceed to checkout and place an order.
4. Admin Access: Use admin credentials to manage products and view orders.


### Login Page:



## Landing page:




## Home page:


[Home](#)
[MyCart](#)
[Profile](#)

**Filters**

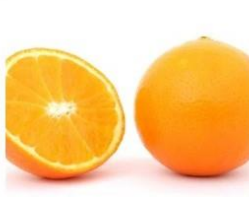
**Category**
☐ Fruits
 ☐ Vegetables
 ☐ Dairy
 ☐ Meat
 ☐ Bakery
 ☐ Beverages
 ☐ Others

**Products**




**Apple**  
Apples are sweet and crisp fruits that come in a variety of colors, including red, green, and yellow. They are rich in fiber and vitamin C and make a healthy snack  
price: 60/kg/l/g

4 ★




**orange**  
Oranges are juicy and tangy citrus fruits known for their high vitamin C content. They are typically peeled and eaten fresh or used to make orange juice  
price: 45/kg/l/g

4.2 ★



**strawberries**  
Strawberries are sweet and fragrant berries with a bright red color. They are rich in antioxidants, vitamin C, and fiber  
price: 80/kg/l/g

3.5 ★



**Banana**  
Bananas are a soft, sweet fruit with a creamy texture, covered by a yellow peel when ripe.  
price: 25/kg/l/g


5 ★

[Add](#) [Buy Now](#)

## Cart page:


[Home](#)
[MyCart](#)
[Profile](#)


**My Cart**



**Apple**  
Price: 60 /-

[Buy](#)


×



**Carrot**  
Price: 45 /-

[Buy](#)

×

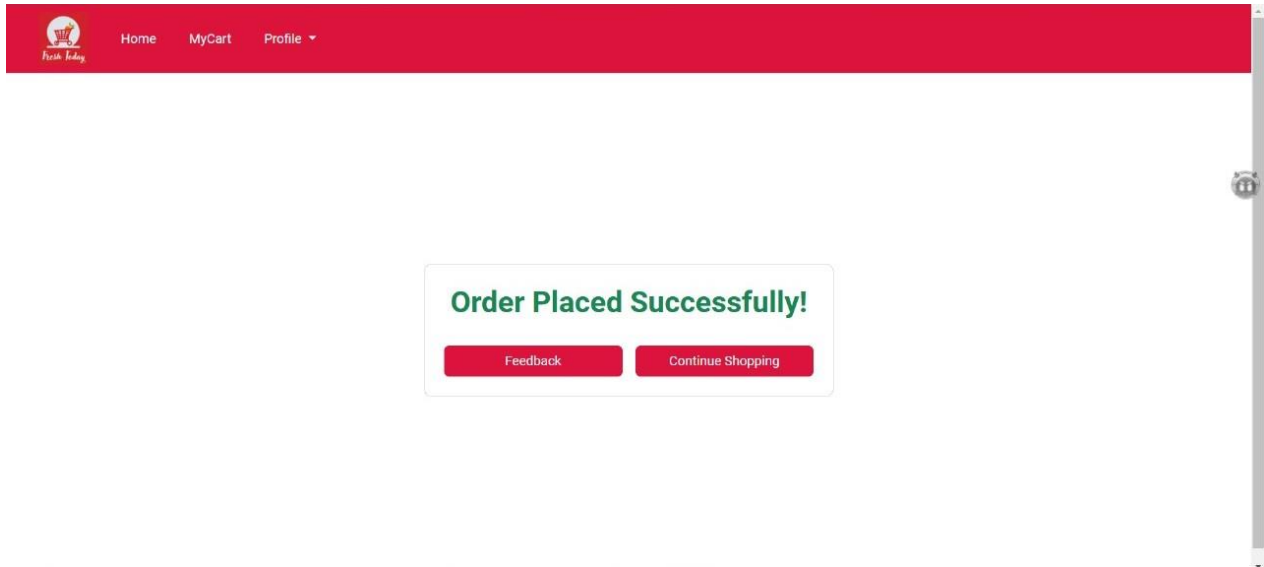


**Broccoli**  
Price: 55 /-

[Buy](#)

×

## Order successful page:



## ISSUES AND FUTURE ENHANCEMENTS

### Current Issues

**Performance:** The site may experience slow loading times with a large number of products.

**Limited Payment Options:** Only basic payment functionality is implemented.

### Future Enhancements

**Payment Integration:** Implement secure payment gateways like Stripe or PayPal.

**Product Recommendations:** Use machine learning to suggest products to users.

**Wishlist Feature:** Allow users to save items for later.

**Order Tracking:** Enable users to track the status of their orders in real-time.

**Enhanced Filtering and Sorting:** Provide more options for users to filter and sort grocery items.

## **CONCLUSION**

The MERN Full-Stack Grocery Website project successfully demonstrates the implementation of a robust and user-friendly platform for online grocery shopping. By leveraging Angular for a dynamic and responsive frontend, Node.js and Express.js for efficient backend services, and MongoDB for scalable data management, the project achieves a seamless integration between all components of the MEAN stack. While there are opportunities for optimization and expansion, such as adding advanced payment solutions and personalized recommendations, the current application provides a solid foundation for meeting user needs and simplifying grocery shopping experiences. Future enhancements and continuous improvements will ensure the platform remains competitive and aligned with evolving user expectations and technological advancements.