

# **AGROLINK : BRIDGE BETWEEN CUSTOMERS AND FARMERS**

Project submitted to the  
SRM University - AP, Andhra Pradesh  
for the partial fulfillment of the requirements to award the degree of

**Bachelor of Technology**  
In  
**Computer Science and Engineering**  
**School of Engineering and Sciences**

Submitted by  
**Md. Afthab Raza Hussain, AP23110010398 | J. Akhil Joy, AP23110010037**  
**N Hemanth Sai, AP23110011391 | P Surya, AP23110010788**



Under the Guidance of  
**Mr. Yatharth sir**  
**Assistant Professor, Department of CSE**

**SRM University-AP**  
**Neerukonda, Mangalagiri, Guntur**  
**Andhra Pradesh - 522 240**  
**[December, 2025]**

# Acknowledgement

The satisfaction that accompanies the successful completion of any task would be incomplete without introducing the people who made it possible and whose constant guidance and encouragement crowns all efforts with success.

I am extremely grateful and express my profound gratitude and indebtedness to my project guide, **Mr. Yatharth**, Department of Computer Science & Engineering, SRM University, Andhra Pradesh, for her kind help and for giving me the necessary guidance and valuable suggestions in completing this project work.

Md. Afthab Raza Hussain, AP23110010398,

J. Akhil Joy, AP23110010037,

N Hemanth Sai, AP23110011391,

P Surya, AP23110010788

# Table of Contents

Acknowledgements	2
Table of Contents	3
1. Introduction	4
2. Scenario-Based Intro	5
3. Target Audience	6
4. Project Goals and Objectives	7
5. Key Features	8
6. Pree-Requisites for Agrolink	9
7. Project Structure	11
8. Project Flow	12
9. Project Execution	20
10. Project Demo Links	24

# 1. Introduction

Agriculture is one of the most important sectors of the economy, yet many farmers continue to face challenges in receiving fair prices for their produce due to the presence of multiple intermediaries. Customers, on the other hand, often struggle to access fresh, high-quality farm products at affordable rates. To address this gap, **Agrolink** is developed as a digital platform that directly connects farmers with customers, eliminating middlemen and ensuring transparency in the buying and selling process.

Agrolink empowers farmers by allowing them to **showcase, manage, and sell their produce online** at fair and profitable prices. Customers can browse a wide range of farm-fresh products, view detailed information, and purchase directly from verified farmers. This direct communication strengthens trust, enhances market accessibility, and promotes fairness in the agricultural trade ecosystem.

The project uses **React.js** for a dynamic and responsive user interface, **Tailwind CSS** for styling, and **JSON-Server** for simulating a backend database. The platform aims to support local farmers, reduce exploitation, and create a sustainable digital marketplace where both buyers and sellers benefit.

## 2. Scenario-Based Intro

Ravi, a small-scale farmer from a rural village, wakes up every morning before sunrise to tend to his crops. After months of hard work, he finally harvests his fresh tomatoes and takes them to the local market. But despite the quality of his produce, Ravi earns very little. Middlemen and wholesalers dictate the prices, leaving him with profits barely enough to support his family.

Meanwhile, in the nearby town, Priya wants to buy fresh and chemical-free vegetables for her family. She visits supermarkets and local vendors, but the prices are high, and the quality is often uncertain.

Both Ravi and Priya face problems—but their needs are perfectly aligned.

### **What if Ravi could sell directly to Priya?**

No middlemen.

Fair prices.

Fresh produce.

Mutual trust.

This is where **Agrolink** comes into the picture.

Agrolink is a digital platform designed to **bridge the gap between farmers and customers**. It empowers farmers like Ravi to showcase their produce online, set fair prices, and reach a wider market. Customers like Priya can browse fresh farm products, view farmer profiles, and purchase directly with confidence.

Through this platform, Agrolink creates a **win-win ecosystem**: farmers earn better income, and customers receive high-quality, fresh produce at affordable rates. By eliminating intermediaries, Agrolink not only supports rural livelihoods but also brings transparency and efficiency to the agricultural supply chain.

### **3. Target Audience**

Agrolink is designed to serve multiple groups of users who directly or indirectly participate in the agricultural value chain. The main target audiences include:

#### **1. Farmers**

This platform primarily targets small, medium, and large-scale farmers who want to:

- Sell their produce at fair prices
- Eliminate middlemen and market exploitation
- Reach customers directly
- Promote their farm products online
- Increase visibility and income

#### **2. Customers / Consumers**

Urban and rural customers who:

- Want to buy fresh, organic, and chemical-free produce
- Prefer direct purchase from farmers
- Look for affordable prices compared to supermarkets
- Want transparency about the source of products.

#### **3. Local Retailers / Small Vendors**

Retail shop owners who:

- Need reliable sources of fresh fruits, vegetables, and grains
- Want to buy directly in bulk from farmers
- Prefer stable quality and reasonable rates

## 4. Project Goals and Objectives

The primary goal of **Agrolink** is to create a seamless digital platform that empowers farmers and enables customers to access fresh, affordable produce directly from the source. Our objectives include:

### **Fair Market Access:**

Provide farmers with a transparent platform where they can showcase and sell their products without middlemen, ensuring fair pricing and increased income.

### **User-Friendly Experience:**

Develop an intuitive and accessible interface that allows both farmers and customers to navigate, list, browse, and purchase products effortlessly, promoting trust and convenience.

### **Efficient Product Management:**

Enable farmers to easily add, update, and manage their produce listings with details such as price, quantity, category, and images—all in one place.

### **Direct Buyer–Seller Interaction:**

Facilitate smooth communication between customers and farmers through clear product details and farmer-specific information, ensuring transparency in the entire process.

### **Modern Tech Stack:**

Utilize advanced web development tools, including React.js, Tailwind CSS, and JSON Server, to deliver a fast, responsive, and scalable marketplace experience.

### **Strengthening the Agricultural Ecosystem:**

Support the growth of local agriculture by connecting rural farmers to urban markets, reducing dependency on intermediaries, and fostering sustainable development.

## 5. Key Features

### **Farmer Dashboard**

Farmers can easily add, update, and delete product listings, manage prices, and track their available stock.

### **Product Listing & Detailed View**

Customers can browse all products with images, descriptions, categories, and pricing details for better decision-making.

### **Search & Filter Options**

Improves user experience by helping customers quickly find specific products based on category or name.

### **Transparent Farmer Profiles**

Displays essential details such as farmer name, location, and contact information to build trust between buyers and sellers.

### **Clean and Responsive UI**

Built using React.js and Tailwind CSS to provide a modern, responsive, and user-friendly interface across all devices.

### **Lightweight Backend with JSON Server**

Uses JSON Server to simulate a backend system for storing and managing real-time product, farmer, and order data.

### **Error-Free Navigation with React Router**

Ensures smooth transitions between pages like Home, Product Details, Dashboard, and Farmers page.

### **Secure and Organized Data Handling**

All data dynamically loads from the backend, ensuring consistent product updates and reliable operation.

## 6. PRE-REQUISITES FOR AGROLINK

### **Node.js and npm:**

Node.js is a powerful JavaScript runtime environment that allows developers to execute JavaScript code outside the browser. It is essential for running development tools, package scripts, and local servers during frontend development.

For Agrolink, Node.js and npm are required to:

- Run the React development server
- Install project dependencies such as React Router and Tailwind CSS
- Operate JSON Server, which is used to simulate the backend for storing farmer and product data

Before building the Agrolink application, ensure that **Node.js and npm** are properly installed on your system.

### **React.js:**

React.js is a widely used JavaScript library for building fast, interactive, and component-based user interfaces. For the Agrolink application, React.js enables the creation of reusable UI components such as product cards, farmer profiles, dashboards, and navigation elements. It ensures a smooth and responsive experience for both farmers and customers.

To begin using React.js, follow these steps:

- Create a new React app:

```
npm create vite@latest
```

Enter project name (e.g., **agrolink**) & select the preferred framework options.

- Navigate to the project directory and install dependencies

```
cd project-name
```

```
npm install
```

- Running the React App:

With the React app created, we can now start the development server and see React application in action.

- Start the development server:

```
npm run dev
```

This command starts the development environment, and you can view the Agrolink application in your browser at:

<http://localhost:5174/>

② **HTML, CSS, and JavaScript:** Fundamental knowledge of HTML, CSS, and JavaScript is required to structure pages, create styles, handle component logic, and manage interactivity throughout the Agrolink application.

② **JSON Server:** Agrolink uses JSON Server to simulate backend. It stores data for:

- Farmer profiles
- Product listings
- Orders

JSON Server allows frontend developers to work with real-time data without needing a fully built backend during development.

Run JSON Server using: `npx json-server --watch db.json --port 5000`

② **Version Control:** Git enables tracking changes, maintaining different versions, and collaborating with team members during Agrolink development.  
<https://git-scm.com/downloads>

② **Development Environment:** Choose a code editor or Integrated Development Environment (IDE) that supports efficient React development. Common choices include:

- 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>

## 7. Project Structure

The screenshot shows a code editor interface with the following details:

- EXPLORER** pane: Shows the project structure under "AGRO FRONTEND". It includes folders like node\_modules, public, src, assets, components, context, pages, and various files such as db.json, App.css, App.jsx, index.css, main.jsx, Footer.jsx, Navbar.jsx, AgroContext.jsx, Cart.jsx, CustomerDashboard.jsx, FarmerDashboard.jsx, Registration.jsx, UserSelection.jsx, and db.json.
- LANDING PAGE**: The current file being edited is LandingPage.jsx. The code defines a LandingPage component that returns a div with a height of 100vh, flex direction of column, and a hero section with a background image and white text.
- TERMINAL**: The terminal shows the command "npm run dev" being run, and it outputs "VITE v7.2.6 ready in 448 ms" and provides local and network URLs.
- PROBLEMS**, **OUTPUT**, **DEBUG CONSOLE**, and **PORTS** tabs are visible at the bottom of the terminal area.
- SIDE BAR**: Shows recent files and a list of open editors.

In the Agrolink project, the folders and files are structured to ensure modularity, clarity, and smooth integration between the frontend interface and the simulated backend. Important Files in our Agrolink project:

- **index.html** → The root HTML file where the React application is mounted.
- **package.json** → Contains all major dependencies such as React, React Router, Tailwind CSS, and development tools.
- **db.json** → Stores farmer information, product details, and orders using JSON Server to simulate a backend.

agrolink/

- |- public/
- |- src/
  - | |- components/ → Reusable UI elements (Navbar, Footer, ProductCard, etc.)
  - | |- pages/ → Screens such as Home, Farmers, Dashboard, Product Details
  - | |- api/ → Handles API calls to JSON Server (fetch products, farmers)
  - | |- App.jsx → Root component holding routes and overall layout
  - | |- main.jsx → Entry point that renders the React application
- |- db.json → Fake backend containing farmers, products, and orders
- |- index.html → Base HTML file where <div id="root"> resides
- |- package.json → Contains dependencies and project metadata
- |- vite.config.js → Configuration file for Vite build tool

## 8. Project Flow

### Project demo:

Before starting to work on this project, let's see the demo.

<https://github.com/mohammedafthabrazahussain/AGROLINK-FRONTEND>

### Agrolink User Flow

1. User opens the Agrolink application.
2. The system fetches **products, farmers, and order data** from JSON Server.
3. The homepage displays all farm products using attractive product cards.
4. The user clicks a product to view **detailed information**, including price, quantity, description, and farmer details.
5. Farmers can log in to their dashboard to **add, edit, or delete** product listings.
6. Customers can browse farmer profiles and directly access produce details.
7. All data updates are reflected instantly from JSON Server.
8. The entire UI adapts responsively for mobile, tablet, and desktop screens using TailwindCSS.

### MILESTONE 1: Project Setup & Configuration

#### 1. Install Required Tools

Open the project folder and install the necessary libraries:

- **React.js** – UI framework
- **React Router** – Page navigation
- **Axios / Fetch API** – API data handling
- **React Toastify** – Notifications
- **JSON Server** – Local backend for storing farmers/products
- **Tailwind CSS** – Responsive UI styling

All installed libraries will appear inside **package.json**

## 2. Recommended Documentation

- React Installation → <https://react.dev/learn/installation>
- React Router → <https://reactrouter.com>
- TailwindCSS → <https://tailwindcss.com/docs/installation>
- JSON Server → <https://www.npmjs.com/package/json-server>

## MILESTONE 2: Web Development

### 1. Setup React Application

- Create a new React app using Vite
- Install required dependencies (React Router, Toastify, Tailwind, JSON Server)
- Configure routing for all pages
- Set up JSON Server backend (db.json)
- Establish API connection using Axios or Fetch
- App.jsx:

```
App.jsx > ...
import React from 'react';
import { BrowserRouter as Router, Routes, Route } from 'react-router-dom';
import { AgroProvider } from './context/AgroContext';
import LandingPage from './pages/LandingPage';
import UserSelection from './pages/UserSelection';
import Registration from './pages/Registration';
import FarmerDashboard from './pages/FarmerDashboard';
import CustomerDashboard from './pages/CustomerDashboard';
import Cart from './pages/Cart';

function App() {
  return (
    <AgroProvider>
      <Router>
        <Routes>
          <Route path="/" element={<LandingPage />} />
          <Route path="/select-user" element={<UserSelection />} />
          <Route path="/register" element={<Registration />} />
          <Route path="/farmer-dashboard" element={<FarmerDashboard />} />
          <Route path="/customer-dashboard" element={<CustomerDashboard />} />
          <Route path="/cart" element={<Cart />} />
          {/* Placeholder routes */}
          <Route path="/about" element={<LandingPage />} />
          <Route path="/login" element={<LandingPage />} />
        </Routes>
      </Router>
    </AgroProvider>
  )
}

export default App;
```

### 2. Design UI Components

Develop reusable components for the Agrolink UI:

- Navbar
- Footer
- ProductCard
- FarmerCard
- Dashboard components
- Forms for Add/Edit Product

**Build pages:**

- Home Page
- Farmers Page
- Product Detail Page
- Farmer Dashboard
- Add Product Page
- Edit Product Page

Use TailwindCSS for layout, spacing, responsiveness, and styling.

### **3. Implement Frontend Logic**

- Fetch product and farmer data from JSON Server
- Display product list dynamically
- Handle CRUD operations:
  - Add product
  - Edit product
  - Delete product
- Bind backend data into UI components
- Implement toast notifications for actions like “Product Added”, “Updated”, “Deleted”
- Add search and filtering options (optional enhancement)

## Data Provider:

```

111      await axios.put(`${API_URL}/cart/${existingItem.id}`, updatedItem);
112      setCart(cart.map(item => item.id === product.id ? updatedItem : item));
113    } catch (error) {
114      console.error("Error adding to cart:", error);
115    }
116  };
117
118  const removeFromCart = async (productId) => {
119    try {
120      await axios.delete(`${API_URL}/cart/${productId}`);
121      setCart(cart.filter(item => item.id !== productId));
122    } catch (error) {
123      console.error("Error removing from cart:", error);
124    }
125  };
126
127  const updateCartQuantity = async (productId, quantity) => {
128    if (quantity < 1) return;
129    try {
130      const item = cart.find(i => i.id === productId);
131      if (!item) return;
132      const newItem = { ...item, quantity };
133      await axios.put(`${API_URL}/cart/${productId}`, newItem);
134      setCart(cart.map(i => i.id === productId ? newItem : i));
135    } catch (error) {
136      console.error("Error updating cart quantity:", error);
137    }
138  };
139
140  const clearCart = async () => {
141    try {
142      await Promise.all(cart.map(item => axios.delete(`${API_URL}/cart/${item.id}`)));
143      setCart([]);
144    } catch (error) {
145      console.error("Error clearing cart:", error);
146    }
147  };
148
149  const getCartTotal = () => {
150    return cart.reduce((total, item) => total + (item.price * item.quantity), 0);
151  };
152
153  return (
154    <AgroContext.Provider value={{
155      user,
156      products,
157      cart,
158      login,
159      registerOrLogin,
160      logout,
161      updateUser,
162    }}>
163
164      <AgroContext.Consumer>
165        {({user, products, cart, login, registerOrLogin, logout, updateUser}) =>
166          <div>
167            <h1>Welcome to AgroMarket</h1>
168            <p>Your one-stop shop for fresh produce and more!</p>
169            <ul>
170              <li>User: <code>{user}</code></li>
171              <li>Products: <code>{products}</code></li>
172              <li>Cart: <code>{cart}</code></li>
173            </ul>
174            <button onClick={login}>Login</button>
175            <button onClick={registerOrLogin}>Register</button>
176            <button onClick={logout}>Logout</button>
177            <button onClick={updateUser}>Update User</button>
178            <button onClick={addProduct}>Add Product</button>
179            <button onClick={updateProduct}>Update Product</button>
180            <button onClick={deleteProduct}>Delete Product</button>
181            <button onClick={addToCart}>Add to Cart</button>
182            <button onClick={removeFromCart}>Remove from Cart</button>
183            <button onClick={clearCart}>Clear Cart</button>
184            <button onClick={getCartTotal}>Get Cart Total</button>
185          </div>
186        }
187      </AgroContext.Consumer>
188    </AgroContext.Provider>
189  );
190
191  export default AgroContext;
192
193  // Actions
194  const login = (userData) => {
195    setUser(userData);
196    localStorage.setItem('agro_user', JSON.stringify(userData));
197  };
198
199  const registerOrLogin = async (userData) => {
200    try {
201      // Check if user exists
202      const res = await axios.get(`${API_URL}/users?phone=${userData.phone}`);
203      const existingUser = res.data.find(p => p.type === userData.type);
204
205      if (existingUser) {
206        console.log(`Logging in existing user: ${existingUser}`);
207        login(existingUser);
208      } else {
209        // Register new user
210        const newUser = { ...userData, id: String(Date.now()) };
211        await axios.post(`${API_URL}/users`, newUser);
212        console.log(`Registering new user: ${newUser}`);
213        login(newUser);
214      }
215    } catch (error) {
216      console.error(`Error fetching initial data: ${error}`);
217    }
218  };
219
220  const logout = () => {
221    setUser(null);
222    localStorage.removeItem('agro_user');
223  };
224
225  const updateUser = async (updatedData) => {
226    if (!user) return;
227    try {
228      const newUser = { ...user, ...updatedData };
229      await axios.put(`${API_URL}/users/${user.id}`, newUser);
230      setUser(newUser);
231      localStorage.setItem('agro_user', JSON.stringify(newUser));
232    } catch (error) {
233      console.error(`Error updating user: ${error}`);
234    }
235  };
236
237  const addProduct = async (product) => {
238    try {
239      const新产品 = { ...product, id: String(Date.now()) };
240      const res = await axios.post(`${API_URL}/products`,新产品);
241      setProducts([...products, res.data]);
242    } catch (error) {
243      console.error(`Error adding product: ${error}`);
244    }
245  };
246
247  const updateProduct = async (id, updatedProduct) => {
248    try {
249      await axios.patch(`${API_URL}/products/${id}`, updatedProduct);
250      setProducts(products.map(p => p.id === id ? { ...p, ...updatedProduct } : p));
251    } catch (error) {
252      console.error(`Error updating product: ${error}`);
253    }
254  };
255
256  const deleteProduct = async (id) => {
257    try {
258      await axios.delete(`${API_URL}/products/${id}`);
259      setProducts(products.filter(p => p.id !== id));
260    } catch (error) {
261      console.error(`Error deleting product: ${error}`);
262    }
263  };
264
265  const addToCart = async (product) => {
266    try {
267      const existingItem = cart.find(item => item.id === product.id);
268      if (existingItem) {
269        const updatedItem = { ...existingItem, quantity: existingItem.quantity + 1 };
270        await axios.put(`${API_URL}/cart/${existingItem.id}`, updatedItem);
271        setCart(cart.map(item => item.id === product.id ? updatedItem : item));
272      } else {
273        const newItem = { ...product, quantity: 1 };
274        const res = await axios.post(`${API_URL}/cart`, newItem);
275        setCart([...cart, res.data]);
276      }
277    } catch (error) {
278      console.error(`Error adding to cart: ${error}`);
279    }
280  };
281
282  const removeFromCart = async (productId) => {
283    try {
284      await axios.delete(`${API_URL}/cart/${productId}`);
285      setCart(cart.filter(item => item.id !== productId));
286    } catch (error) {
287      console.error(`Error removing from cart: ${error}`);
288    }
289  };
290
291  const updateCartQuantity = async (productId, quantity) => {
292    if (quantity < 1) return;
293    try {
294      const item = cart.find(i => i.id === productId);
295      if (!item) return;
296      const newItem = { ...item, quantity };
297      await axios.put(`${API_URL}/cart/${productId}`, newItem);
298      setCart(cart.map(i => i.id === productId ? newItem : i));
299    } catch (error) {
300      console.error(`Error updating cart quantity: ${error}`);
301    }
302  };
303
304  const clearCart = async () => {
305    try {
306      await Promise.all(cart.map(item => axios.delete(`${API_URL}/cart/${item.id}`)));
307      setCart([]);
308    } catch (error) {
309      console.error(`Error clearing cart: ${error}`);
310    }
311  };
312
313  const getCartTotal = () => {
314    return cart.reduce((total, item) => total + (item.price * item.quantity), 0);
315  };
316
317  return (
318    <AgroContext.Provider value={{
319      user,
320      products,
321      cart,
322      login,
323      registerOrLogin,
324      logout,
325      updateUser,
326    }}>
327
328      <AgroContext.Consumer>
329        {({user, products, cart, login, registerOrLogin, logout, updateUser}) =>
330          <div>
331            <h1>Welcome to AgroMarket</h1>
332            <p>Your one-stop shop for fresh produce and more!</p>
333            <ul>
334              <li>User: <code>{user}</code></li>
335              <li>Products: <code>{products}</code></li>
336              <li>Cart: <code>{cart}</code></li>
337            </ul>
338            <button onClick={login}>Login</button>
339            <button onClick={registerOrLogin}>Register</button>
340            <button onClick={logout}>Logout</button>
341            <button onClick={updateUser}>Update User</button>
342            <button onClick={addProduct}>Add Product</button>
343            <button onClick={updateProduct}>Update Product</button>
344            <button onClick={deleteProduct}>Delete Product</button>
345            <button onClick={addToCart}>Add to Cart</button>
346            <button onClick={removeFromCart}>Remove from Cart</button>
347            <button onClick={clearCart}>Clear Cart</button>
348            <button onClick={getCartTotal}>Get Cart Total</button>
349          </div>
350        }
351      </AgroContext.Consumer>
352    </AgroContext.Provider>
353  );
354
355  export default AgroContext;
356
357  // Actions
358  const login = (userData) => {
359    setUser(userData);
360    localStorage.setItem('agro_user', JSON.stringify(userData));
361  };
362
363  const registerOrLogin = async (userData) => {
364    try {
365      // Check for persisted user in localStorage (optional, for session persistence)
366      const savedUser = localStorage.getItem('agro_user');
367      if (savedUser) {
368        setUser(JSON.parse(savedUser));
369      }
370    } catch (error) {
371      console.error(`Error fetching initial data: ${error}`);
372    }
373  };
374
375  const logout = () => {
376    setUser(null);
377    localStorage.removeItem('agro_user');
378  };
379
380  const updateUser = async (updatedData) => {
381    if (!user) return;
382    try {
383      const newUser = { ...user, ...updatedData };
384      await axios.put(`${API_URL}/users/${user.id}`, newUser);
385      setUser(newUser);
386      localStorage.setItem('agro_user', JSON.stringify(newUser));
387    } catch (error) {
388      console.error(`Error in registerOrLogin: ${error}`);
389      alert(`Failed to login/register. Please check if server is running.`);
390    }
391  };
392
393  const addProduct = async (product) => {
394    try {
395      const新产品 = { ...product, id: String(Date.now()) };
396      const res = await axios.post(`${API_URL}/products`,新产品);
397      setProducts([...products, res.data]);
398    } catch (error) {
399      console.error(`Error adding product: ${error}`);
400    }
401  };
402
403  const updateProduct = async (id, updatedProduct) => {
404    try {
405      await axios.patch(`${API_URL}/products/${id}`, updatedProduct);
406      setProducts(products.map(p => p.id === id ? { ...p, ...updatedProduct } : p));
407    } catch (error) {
408      console.error(`Error updating product: ${error}`);
409    }
410  };
411
412  const deleteProduct = async (id) => {
413    try {
414      await axios.delete(`${API_URL}/products/${id}`);
415      setProducts(products.filter(p => p.id !== id));
416    } catch (error) {
417      console.error(`Error deleting product: ${error}`);
418    }
419  };
420
421  const addToCart = async (product) => {
422    try {
423      const existingItem = cart.find(item => item.id === product.id);
424      if (existingItem) {
425        const updatedItem = { ...existingItem, quantity: existingItem.quantity + 1 };
426        await axios.put(`${API_URL}/cart/${existingItem.id}`, updatedItem);
427        setCart(cart.map(item => item.id === product.id ? updatedItem : item));
428      } else {
429        const newItem = { ...product, quantity: 1 };
430        const res = await axios.post(`${API_URL}/cart`, newItem);
431        setCart([...cart, res.data]);
432      }
433    } catch (error) {
434      console.error(`Error adding to cart: ${error}`);
435    }
436  };
437
438  const removeFromCart = async (productId) => {
439    try {
440      await axios.delete(`${API_URL}/cart/${productId}`);
441      setCart(cart.filter(item => item.id !== productId));
442    } catch (error) {
443      console.error(`Error removing from cart: ${error}`);
444    }
445  };
446
447  const updateCartQuantity = async (productId, quantity) => {
448    if (quantity < 1) return;
449    try {
450      const item = cart.find(i => i.id === productId);
451      if (!item) return;
452      const newItem = { ...item, quantity };
453      await axios.put(`${API_URL}/cart/${productId}`, newItem);
454      setCart(cart.map(i => i.id === productId ? newItem : i));
455    } catch (error) {
456      console.error(`Error updating cart quantity: ${error}`);
457    }
458  };
459
460  const clearCart = async () => {
461    try {
462      await Promise.all(cart.map(item => axios.delete(`${API_URL}/cart/${item.id}`)));
463      setCart([]);
464    } catch (error) {
465      console.error(`Error clearing cart: ${error}`);
466    }
467  };
468
469  const getCartTotal = () => {
470    return cart.reduce((total, item) => total + (item.price * item.quantity), 0);
471  };
472
473  return (
474    <AgroContext.Provider value={{
475      user,
476      products,
477      cart,
478      login,
479      registerOrLogin,
480      logout,
481      updateUser,
482    }}>
483
484      <AgroContext.Consumer>
485        {({user, products, cart, login, registerOrLogin, logout, updateUser}) =>
486          <div>
487            <h1>Welcome to AgroMarket</h1>
488            <p>Your one-stop shop for fresh produce and more!</p>
489            <ul>
490              <li>User: <code>{user}</code></li>
491              <li>Products: <code>{products}</code></li>
492              <li>Cart: <code>{cart}</code></li>
493            </ul>
494            <button onClick={login}>Login</button>
495            <button onClick={registerOrLogin}>Register</button>
496            <button onClick={logout}>Logout</button>
497            <button onClick={updateUser}>Update User</button>
498            <button onClick={addProduct}>Add Product</button>
499            <button onClick={updateProduct}>Update Product</button>
500            <button onClick={deleteProduct}>Delete Product</button>
501            <button onClick={addToCart}>Add to Cart</button>
502            <button onClick={removeFromCart}>Remove from Cart</button>
503            <button onClick={clearCart}>Clear Cart</button>
504            <button onClick={getCartTotal}>Get Cart Total</button>
505          </div>
506        }
507      </AgroContext.Consumer>
508    </AgroContext.Provider>
509  );
510
511  export default AgroContext;
512
513  // Actions
514  const login = (userData) => {
515    setUser(userData);
516    localStorage.setItem('agro_user', JSON.stringify(userData));
517  };
518
519  const registerOrLogin = async (userData) => {
520    try {
521      // Initial Load
522      React.useEffect(() => {
523        const fetchData = async () => {
524          try {
525            const [productsRes, cartRes] = await Promise.all([
526              axios.get(`${API_URL}/products`),
527              axios.get(`${API_URL}/cart`)
528            ]);
529            setProducts(productsRes.data);
530            setCart(cartRes.data);
531
532            // Check for persisted user in localStorage (optional, for session persistence)
533            const savedUser = localStorage.getItem('agro_user');
534            if (savedUser) {
535              setUser(JSON.parse(savedUser));
536            }
537
538            // Fetch initial data
539            const fetchInitialData = async () => {
540              const fetchUserData = async () => {
541                const res = await axios.get(`${API_URL}/users`);
542                setUser(res.data[0]);
543                localStorage.setItem('agro_user', JSON.stringify(res.data[0]));
544              };
545
546              const fetchCartData = async () => {
547                const res = await axios.get(`${API_URL}/cart`);
548                setCart(res.data);
549              };
550
551              await Promise.all([fetchUserData(), fetchCartData()]);
552            };
553
554            fetchInitialData();
555          } catch (error) {
556            console.error(`Error fetching initial data: ${error}`);
557          }
558        };
559
560        fetchData();
561      }, []);
562    } catch (error) {
563      console.error(`Error in registerOrLogin: ${error}`);
564      alert(`Failed to login/register. Please check if server is running.`);
565    }
566  };
567
568  const logout = () => {
569    setUser(null);
570    localStorage.removeItem('agro_user');
571  };
572
573  const updateUser = async (updatedData) => {
574    if (!user) return;
575    try {
576      const newUser = { ...user, ...updatedData };
577      await axios.put(`${API_URL}/users/${user.id}`, newUser);
578      setUser(newUser);
579      localStorage.setItem('agro_user', JSON.stringify(newUser));
580    } catch (error) {
581      console.error(`Error in registerOrLogin: ${error}`);
582    }
583  };
584
585  const addProduct = async (product) => {
586    try {
587      const新产品 = { ...product, id: String(Date.now()) };
588      const res = await axios.post(`${API_URL}/products`,新产品);
589      setProducts([...products, res.data]);
590    } catch (error) {
591      console.error(`Error adding product: ${error}`);
592    }
593  };
594
595  const updateProduct = async (id, updatedProduct) => {
596    try {
597      await axios.patch(`${API_URL}/products/${id}`, updatedProduct);
598      setProducts(products.map(p => p.id === id ? { ...p, ...updatedProduct } : p));
599    } catch (error) {
600      console.error(`Error updating product: ${error}`);
601    }
602  };
603
604  const deleteProduct = async (id) => {
605    try {
606      await axios.delete(`${API_URL}/products/${id}`);
607      setProducts(products.filter(p => p.id !== id));
608    } catch (error) {
609      console.error(`Error deleting product: ${error}`);
610    }
611  };
612
613  const addToCart = async (product) => {
614    try {
615      const existingItem = cart.find(item => item.id === product.id);
616      if (existingItem) {
617        const updatedItem = { ...existingItem, quantity: existingItem.quantity + 1 };
618        await axios.put(`${API_URL}/cart/${existingItem.id}`, updatedItem);
619        setCart(cart.map(item => item.id === product.id ? updatedItem : item));
620      } else {
621        const newItem = { ...product, quantity: 1 };
622        const res = await axios.post(`${API_URL}/cart`, newItem);
623        setCart([...cart, res.data]);
624      }
625    } catch (error) {
626      console.error(`Error adding to cart: ${error}`);
627    }
628  };
629
630  const removeFromCart = async (productId) => {
631    try {
632      await axios.delete(`${API_URL}/cart/${productId}`);
633      setCart(cart.filter(item => item.id !== productId));
634    } catch (error) {
635      console.error(`Error removing from cart: ${error}`);
636    }
637  };
638
639  const updateCartQuantity = async (productId, quantity) => {
640    if (quantity < 1) return;
641    try {
642      const item = cart.find(i => i.id === productId);
643      if (!item) return;
644      const newItem = { ...item, quantity };
645      await axios.put(`${API_URL}/cart/${productId}`, newItem);
646      setCart(cart.map(i => i.id === productId ? newItem : i));
647    } catch (error) {
648      console.error(`Error updating cart quantity: ${error}`);
649    }
650  };
651
652  const clearCart = async () => {
653    try {
654      await Promise.all(cart.map(item => axios.delete(`${API_URL}/cart/${item.id}`)));
655      setCart([]);
656    } catch (error) {
657      console.error(`Error clearing cart: ${error}`);
658    }
659  };
660
661  const getCartTotal = () => {
662    return cart.reduce((total, item) => total + (item.price * item.quantity), 0);
663  };
664
665  return (
666    <AgroContext.Provider value={{
667      user,
668      products,
669      cart,
670      login,
671      registerOrLogin,
672      logout,
673      updateUser,
674    }}>
675
676      <AgroContext.Consumer>
677        {({user, products, cart, login, registerOrLogin, logout, updateUser}) =>
678          <div>
679            <h1>Welcome to AgroMarket</h1>
680            <p>Your one-stop shop for fresh produce and more!</p>
681            <ul>
682              <li>User: <code>{user}</code></li>
683              <li>Products: <code>{products}</code></li>
684              <li>Cart: <code>{cart}</code></li>
685            </ul>
686            <button onClick={login}>Login</button>
687            <button onClick={registerOrLogin}>Register</button>
688            <button onClick={logout}>Logout</button>
689            <button onClick={updateUser}>Update User</button>
690            <button onClick={addProduct}>Add Product</button>
691            <button onClick={updateProduct}>Update Product</button>
692            <button onClick={deleteProduct}>Delete Product</button>
693            <button onClick={addToCart}>Add to Cart</button>
694            <button onClick={removeFromCart}>Remove from Cart</button>
695            <button onClick={clearCart}>Clear Cart</button>
696            <button onClick={getCartTotal}>Get Cart Total</button>
697          </div>
698        }
699      </AgroContext.Consumer>
700    </AgroContext.Provider>
701  );
702
703  export default AgroContext;
704
705  // Actions
706  const login = (userData) => {
707    setUser(userData);
708    localStorage.setItem('agro_user', JSON.stringify(userData));
709  };
710
711  const registerOrLogin = async (userData) => {
712    try {
713      // Initial Load
714      React.useEffect(() => {
715        const fetchData = async () => {
716          try {
717            const [productsRes, cartRes] = await Promise.all([
718              axios.get(`${API_URL}/products`),
719              axios.get(`${API_URL}/cart`)
720            ]);
721            setProducts(productsRes.data);
722            setCart(cartRes.data);
723
724            // Check for persisted user in localStorage (optional, for session persistence)
725            const savedUser = localStorage.getItem('agro_user');
726            if (savedUser) {
727              setUser(JSON.parse(savedUser));
728            }
729
730            // Fetch initial data
731            const fetchInitialData = async () => {
732              const fetchUserData = async () => {
733                const res = await axios.get(`${API_URL}/users`);
734                setUser(res.data[0]);
735                localStorage.setItem('agro_user', JSON.stringify(res.data[0]));
736              };
737
738              const fetchCartData = async () => {
739                const res = await axios.get(`${API_URL}/cart`);
740                setCart(res.data);
741              };
742
743              await Promise.all([fetchUserData(), fetchCartData()]);
744            };
745
746            fetchInitialData();
747          } catch (error) {
748            console.error(`Error fetching initial data: ${error}`);
749          }
750        };
751
752        fetchData();
753      }, []);
754    } catch (error) {
755      console.error(`Error in registerOrLogin: ${error}`);
756      alert(`Failed to login/register. Please check if server is running.`);
757    }
758  };
759
760  const logout = () => {
761    setUser(null);
762    localStorage.removeItem('agro_user');
763  };
764
765  const updateUser = async (updatedData) => {
766    if (!user) return;
767    try {
768      const newUser = { ...user, ...updatedData };
769      await axios.put(`${API_URL}/users/${user.id}`, newUser);
770      setUser(newUser);
771      localStorage.setItem('agro_user', JSON.stringify(newUser));
772    } catch (error) {
773      console.error(`Error in registerOrLogin: ${error}`);
774    }
775  };
776
777  const addProduct = async (product) => {
778    try {
779      const新产品 = { ...product, id: String(Date.now()) };
780      const res = await axios.post(`${API_URL}/products`,新产品);
781      setProducts([...products, res.data]);
782    } catch (error) {
783      console.error(`Error adding product: ${error}`);
784    }
785  };
786
787  const updateProduct = async (id, updatedProduct) => {
788    try {
789      await axios.patch(`${API_URL}/products/${id}`, updatedProduct);
790      setProducts(products.map(p => p.id === id ? { ...p, ...updatedProduct } : p));
791    } catch (error) {
792      console.error(`Error updating product: ${error}`);
793    }
794  };
795
796  const deleteProduct = async (id) => {
797    try {
798      await axios.delete(`${API_URL}/products/${id}`);
799      setProducts(products.filter(p => p.id !== id));
800    } catch (error) {
801      console.error(`Error deleting product: ${error}`);
802    }
803  };
804
805  const addToCart = async (product) => {
806    try {
807      const existingItem = cart.find(item => item.id === product.id);
808      if (existingItem) {
809        const updatedItem = { ...existingItem, quantity: existingItem.quantity + 1 };
810        await axios.put(`${API_URL}/cart/${existingItem.id}`, updatedItem);
811        setCart(cart.map(item => item.id === product.id ? updatedItem : item));
812      } else {
813        const newItem = { ...product, quantity: 1 };
814        const res = await axios.post(`${API_URL}/cart`, newItem);
815        setCart([...cart, res.data]);
816      }
817    } catch (error) {
818      console.error(`Error adding to cart: ${error}`);
819    }
820  };
821
822  const removeFromCart = async (productId) => {
823    try {
824      await axios.delete(`${API_URL}/cart/${productId}`);
825      setCart(cart.filter(item => item.id !== productId));
826    } catch (error) {
827      console.error(`Error removing from cart: ${error}`);
828    }
829  };
830
831  const updateCartQuantity = async (productId, quantity) => {
832    if (quantity < 1) return;
833    try {
834      const item = cart.find(i => i.id === productId);
835      if (!item) return;
836      const newItem = { ...item, quantity };
837      await axios.put(`${API_URL}/cart/${productId}`, newItem);
838      setCart(cart.map(i => i.id === productId ? newItem : i));
839    } catch (error) {
840      console.error(`Error updating cart quantity: ${error}`);
841    }
842  };
843
844  const clearCart = async () => {
845    try {
846      await Promise.all(cart.map(item => axios.delete(`${API_URL}/cart/${item.id}`)));
847      setCart([]);
848    } catch (error) {
849      console.error(`Error clearing cart: ${error}`);
850    }
851  };
852
853  const getCartTotal = () => {
854    return cart.reduce((total, item) => total + (item.price * item.quantity), 0);
855  };
856
857  return (
858    <AgroContext.Provider value={{
859      user,
860      products,
861      cart,
862      login,
863      registerOrLogin,
864      logout,
865      updateUser,
866    }}>
867
868      <AgroContext.Consumer>
869        {({user, products, cart, login, registerOrLogin, logout, updateUser}) =>
870          <div>
871            <h1>Welcome to AgroMarket</h1>
872            <p>Your one-stop shop for fresh produce and more!</p>
873            <ul>
874              <li>User: <code>{user}</code></li>
875              <li>Products: <code>{products}</code></li>
876              <li>Cart: <code>{cart}</code></li>
877            </ul>
878            <button onClick={login}>Login</button>
879            <button onClick={registerOrLogin}>Register</button>
880            <button onClick={logout}>Logout</button>
881            <button onClick={updateUser}>Update User</button>
882            <button onClick={addProduct}>Add Product</button>
883            <button onClick={updateProduct}>Update Product</button>
884            <button onClick={deleteProduct}>Delete Product</button>
885            <button onClick={addToCart}>Add to Cart</button>
886            <button onClick={removeFromCart}>Remove from Cart</button>
887            <button onClick={clearCart}>Clear Cart</button>
888            <button onClick={getCartTotal}>Get Cart Total</button>
889          </div>
890        }
891      </AgroContext.Consumer>
892    </AgroContext.Provider>
893  );
894
895  export default AgroContext;
896
897  // Actions
898  const login = (userData) => {
899    setUser(userData);
900    localStorage.setItem('agro_user', JSON.stringify(userData));
901  };
902
903  const registerOrLogin = async (userData) => {
904    try {
905      // Initial Load
906      React.useEffect(() => {
907        const fetchData = async () => {
908          try {
909            const [productsRes, cartRes] = await Promise.all([
910              axios.get(`${API_URL}/products`),
911              axios.get(`${API_URL}/cart`)
912            ]);
913            setProducts(productsRes.data);
914            setCart(cartRes.data);
915
916            // Check for persisted user in localStorage (optional, for session persistence)
917            const savedUser = localStorage.getItem('agro_user');
918            if (savedUser) {
919              setUser(JSON.parse(savedUser));
920            }
921
922            // Fetch initial data
923            const fetchInitialData = async () => {
924              const fetchUserData = async () => {
925                const res = await axios.get(`${API_URL}/users`);
926                setUser(res.data[0]);
927                localStorage.setItem('agro_user', JSON.stringify(res.data[0]));
928              };
929
930              const fetchCartData = async () => {
931                const res = await axios.get(`${API_URL}/cart`);
932                setCart(res.data);
933              };
934
935              await Promise.all([fetchUserData(), fetchCartData()]);
936            };
937
938            fetchInitialData();
939          } catch (error) {
940            console.error(`Error fetching initial data: ${error}`);
941          }
942        };
943
944        fetchData();
945      }, []);
946    } catch (error) {
947      console.error(`Error in registerOrLogin: ${error}`);
948      alert(`Failed to login/register. Please check if server is running.`);
949    }
950  };
951
952  const logout = () => {
953    setUser(null);
954    localStorage.removeItem('agro_user');
955  };
956
957  const updateUser = async (updatedData) => {
958    if (!user) return;
959    try {
960      const newUser = { ...user, ...updatedData };
961      await axios.put(`${API_URL}/users/${user.id}`, newUser);
962      setUser(newUser);
963      localStorage.setItem('agro_user', JSON.stringify(newUser));
964    } catch (error) {
965      console.error(`Error in registerOrLogin: ${error}`);
966    }
967  };
968
969  const addProduct = async (product) => {
970    try {
971      const新产品 = { ...product, id: String(Date.now()) };
972      const res = await axios.post(`${API_URL}/products`,新产品);
973      setProducts([...products, res.data]);
974    } catch (error) {
975      console.error(`Error adding product: ${error}`);
976    }
977  };
978
979  const updateProduct = async (id, updatedProduct) => {
980    try {
981      await axios.patch(`${API_URL}/products/${id}`, updatedProduct);
982      setProducts(products.map(p => p.id === id ? { ...p, ...updatedProduct } : p));
983    } catch (error) {
984      console.error(`Error updating product: ${error}`);
985    }
986  };
987
988  const deleteProduct = async (id) => {
989    try {
990      await axios.delete(`${API_URL}/products/${id}`);
991      setProducts(products.filter(p => p.id !== id));
992    } catch (error) {
993      console.error(`Error deleting product: ${error}`);
994    }
995  };
996
997  const addToCart = async (product) => {
998    try {
999      const existingItem = cart.find(item => item.id === product.id);
1000     if (existingItem) {
1001       const updatedItem = { ...existingItem, quantity: existingItem.quantity + 1 };
1002       await axios.put(`${API_URL}/cart/${existingItem.id}`, updatedItem);
1003       setCart(cart.map(item => item.id === product.id ? updatedItem : item));
1004     } else {
1005       const newItem = { ...product, quantity: 1 };
1006       const res = await axios.post(`${API_URL}/cart`, newItem);
1007       setCart([...cart, res.data]);
1008     }
1009   } catch (error) {
1010     console.error(`Error adding to cart: ${error}`);
1011   }
1012 };
1013
1014  const removeFromCart = async (productId) => {
1015   try {
1016     await axios.delete(`${API_URL}/cart/${productId}`);
1017     setCart(cart.filter(item => item.id !== productId));
1018   } catch (error) {
1019     console.error(`Error removing from cart: ${error}`);
1020   }
1021 };
1022
1023  const updateCartQuantity = async (productId, quantity) => {
1024    if (quantity < 1) return;
1025    try {
1026      const item = cart.find(i => i.id === productId);
1027      if (!item) return;
1028      const newItem = { ...item, quantity };
1029      await axios.put(`${API_URL}/cart/${productId}`, newItem);
1030      setCart(cart.map(i => i.id === productId ? newItem : i));
1031    } catch (error) {
1032      console.error(`Error updating cart quantity: ${error}`);
1033    }
1034  };
1035
1036  const clearCart = async () => {
1037    try {
1038      await Promise.all(cart.map(item => axios.delete(`${API_URL}/cart/${item.id}`)));
1039      setCart([]);
1040    } catch (error) {
1041      console.error(`Error clearing cart: ${error}`);
1042    }
1043  };
1044
1045  const getCartTotal = () => {
1046    return cart.reduce((total, item) => total + (item.price * item.quantity), 0);
1047  };
1048
1049  return (
1050    <AgroContext.Provider value={{
1051      user,
1052      products,
1053      cart,
1054      login,
1055      registerOrLogin,
1056      logout,
1057      updateUser,
1058    }}>
1059
1060      <AgroContext.Consumer>
1061        {({user, products, cart, login, registerOrLogin, logout, updateUser}) =>
1062          <div>
1063            <h1>Welcome to AgroMarket</h1>
1064            <p>Your one-stop shop for fresh produce and more!</p>
1065            <ul>
1
```

Creates an **AgroContext** using `createContext()` to hold global data such as user details, product listings, and cart information.

Defines an AgroProvider component that wraps the entire application, allowing all nested components to access shared data through the context.

Uses useState hooks to create and manage important global state variables:

- user → stores logged-in farmer/customer
- products → stores all product listings
- cart → stores items added by the user

Loads initial data using `useEffect()`, where it fetches products and cart items from JSON Server, and retrieves saved user information from `localStorage`.

Implements authentication functions, such as:

- `login()` → logs in existing user
- `registerOrLogin()` → registers a new user or logs in if already registered
- `logout()` → clears user session

Provides CRUD operations for products, including:

- `addProduct()`
- `updateProduct()`
- `deleteProduct()`

Manages customer cart functionality, offering:

- `addToCart()`
- `removeFromCart()`
- `updateCartQuantity()`
- `clearCart()`
- `getCartTotal()` → calculates total cost

Returns an `AgroContext.Provider` wrapping all child components, passing down state values and functions via the `value` prop.

Exports the `useAgro()` hook to easily access context data inside any component using `useContext`.

Wraps important components inside the AgroProvider, ensuring every page (Home, Dashboard, Product Details, Cart, Farmers) receives shared data without prop-drilling

## Code Component:-UI Components

### 1. footer Component:

```
src > components > Footer.jsx > ...
1  import React from 'react';
2
3  const Footer = () => {
4    return (
5      <footer style={{
6        backgroundColor: 'var(--color-primary)',
7        color: 'white',
8        padding: '3rem 0',
9        marginTop: 'auto'
10       }}>
11      <div className="container text-center">
12        <p>&copy; {new Date().getFullYear()} AgroLink. Connecting Farmers and Customers.</p>
13      </div>
14    </footer>
15  );
16}
17
18 export default Footer;
19
```

#### Code Description:

- **Imports React** to create the functional component.
- **Imports Link from React Router** to enable navigation between pages without reloading the browser.
- **Imports TailwindCSS classes** to style the navbar with spacing, colors, and layout.
- **Defines the Navbar functional component**, which serves as the top navigation bar for the Agrolink application.  
Creates a **navigation container** using a `<nav>` element styled with background color, padding, and flex layout for proper alignment.
- **Displays the Agrolink brand/title**, usually positioned on the left side of the navbar.
- **Renders navigation links** such as:
  - Home, Farmers
  - Dashboard, Cart
  - Login / Profile

Each link is wrapped inside a `Link` component so users can navigate smoothly.

- **Applies hover effects and text styles** using Tailwind classes to improve the user experience and make navigation interactive.
- **Ensures the Navbar is responsive**, adjusting layout automatically for different screen sizes using utility classes.
- **Exports the Navbar component as the default export**, allowing it to be used inside `App.js` and displayed across all pages.

## 2. Navbar Component:

```
src>components > @ Navbar.jsx <...
  1 import React, { useState, useEffect } from 'react';
  2 import { Link, useLocation } from 'react-router-dom';
  3 import { Sprout, ShoppingCart, LogOut, User } from 'lucide-react';
  4 import { useAgro } from '../context/AgroContext';
  5
  6 const Navbar = () => {
  7   const { user, cart, logout } = useAgro();
  8   const [scrolled, setScrolled] = useState(false);
  9   const location = useLocation();
 10
 11   useEffect(() => {
 12     const handleScroll = () => {
 13       setScrolled(window.scrollY > 50);
 14     };
 15     window.addEventListener('scroll', handleScroll);
 16     return () => window.removeEventListener('scroll', handleScroll);
 17   }, []);
 18
 19   const scrollToAbout = (e) => {
 20     e.preventDefault();
 21     if (location.pathname !== '/') {
 22       window.location.href = '#about';
 23     } else {
 24       const aboutSection = document.getElementById('about');
 25       if (aboutSection) {
 26         aboutSection.scrollIntoView({ behavior: 'smooth' });
 27       }
 28     }
 29   };
 30
 31   return (
 32     <nav style={{
 33       backgroundColor: scrolled ? 'rgba(255, 255, 255, 0.95)' : 'transparent',
 34       backdropFilter: scrolled ? 'blur(10px)' : 'none',
 35       boxShadow: scrolled ? 'var(--shadow-md)' : 'none',
 36       padding: '1.25rem 0',
 37       position: 'fixed',
 38       top: 0,
 39       left: 0,
 40       right: 0,
 41       zIndex: 1000,
 42       transition: 'all 0.3s ease'
 43     }}>
 44       <div className="container" style={{ display: 'flex', justifyContent: 'space-between', alignItems: 'center' }}>
 45         <Link to="/" style={{
 46           display: 'flex',
 47           alignItems: 'center',
 48           gap: '0.75rem',
 49           fontSize: '1.75rem',
 50           fontWeight: 700,
 51           color: scrolled ? 'var(--color-primary)' : 'var(--color-primary-dark)',
 52           fontFamily: 'var(--font-heading)',
 53           letterSpacing: '-0.02em'
 54         }}>
 55           <div style={{
 56             backgroundColor: 'var(--color-accent)',
 57             padding: '0.5rem',
 58             borderRadius: '50%',
 59             color: 'white',
 60             display: 'flex'
 61           }>
 62             <span style={{ size: 24 }} />
 63           </div>
 64           <span>AgroLink</span>
 65         </Link>
 66         <div style={{ display: 'flex', gap: '2rem', alignItems: 'center' }}>
 67           <a href="#about" onClick={scrollToAbout} style={{ fontWeight: 600, color: 'var(--color-text-main)', fontSize: '0.95rem', textTransform: 'uppercase', letterSpacing: '0.05em', cursor: 'pointer' }}>About</a>
 68         </div>
 69       </div>
 70       <div style={{ display: 'flex', alignItems: 'center' }}>
 71         <div style={{ display: 'flex', gap: '1.5rem' }}>
 72           <span>User && </span>
 73           <div style={{ position: 'relative', color: 'var(--color-primary)' }}>
 74             <User type="circle" style={{ position: 'absolute', top: -10px, left: -10px, width: 20px, height: 20px, border: '2px solid var(--color-accent)', border-radius: '50%', overflow: 'hidden' }}>
 75               <img alt="User profile picture" style={{ width: '100%', height: '100%', objectFit: 'cover' }} />
 76             </User>
 77             <span style={{ position: 'absolute', top: 0, left: 0, width: '100%', height: '100%', background: 'var(--color-accent)', color: 'white', borderRadius: '50%', border: '2px solid #fff', width: 20px, height: 20px, display: 'flex', alignItems: 'center', justifyContent: 'center', fontSize: '0.7rem', fontWeight: 'bold' }}>
 78               <span>Cart</span>
 79             </span>
 80           </div>
 81           <span style={{ color: 'var(--color-accent)', border: '1px solid #fff', padding: '0.25rem 0.5rem', border-radius: '0.5rem', display: 'flex', alignItems: 'center', gap: '0.5rem' }}>
 82             <span>Cart</span>
 83             <span>0</span>
 84           </span>
 85           <span style={{ color: 'var(--color-accent)', border: '1px solid #fff', padding: '0.25rem 0.5rem', border-radius: '0.5rem', display: 'flex', alignItems: 'center', gap: '0.5rem' }}>
 86             <span>Logout</span>
 87             <span>Logout</span>
 88           </span>
 89         </div>
 90         <div style={{ background: 'none', color: 'var(--color-text-suited)', display: 'flex', alignItems: 'center', gap: '0.5rem' }}>
 91           <button style={{ border: '1px solid #fff', padding: '0.25rem 0.5rem', border-radius: '0.5rem', color: 'var(--color-accent)', fontWeight: 'bold' }}>Log Out</button>
 92         </div>
 93       </div>
 94     </nav>
 95   );
 96   export default Navbar;
 97 
```

### Code Description:

Imports React hooks (useState, useEffect) for managing scroll state and controlling behavior when the navbar becomes sticky.

Imports Routing utilities (Link, useLocation) from React Router, allowing navigation without page reloads and detecting the current route.

Imports icons (Sprout, ShoppingCart, LogOut, User) from lucide-react to visually enhance the navbar.

Imports the global AgroContext using `useAgro()` to access shared data such as:

- `user` → logged-in user details
- `cart` → items in user's cart
- `logout` → function to log out users

Creates a scrolled state using `useState(false)` to change navbar style (blur, shadow, background) when the user scrolls past 50px.

Adds a scroll listener inside `useEffect` to dynamically update the navbar's appearance based on scroll behavior.

Defines `scrollToAbout` function, which:

- Smoothly scrolls to the "About" section
- Redirects to home page first if the user is on another route

Returns a modern, fixed Navbar layout using inline CSS and Tailwind variables:

- Background dynamically changes based on scroll
- Navbar stays fixed at the top with smooth transitions
- Uses flexbox for alignment

Displays brand logo "AgroLink" using the Sprout icon, styled with custom color, spacing, and font.

Renders navigation links, including:

- "About" section link with smooth scroll animation.

Conditionally displays logged-in user controls, only if user exists:

- User name
- Cart icon for customers with live badge count (total quantity)
- Logout button with LogOut icon

Computes cart quantity using:

```
cart.reduce((acc, item) => acc + item.quantity, 0)  
to show total number of items.
```

Applies clean UI styling, animations, spacing, hover effects, and shadow using Tailwind variables and inline CSS.

Exports the Navbar component for use across all pages via `<Navbar />`.

## 9. Project Execution

After successfully completing the development of the Agrolink platform, you can run the React project using the command:

```
npm run dev
```

This will start the Vite development server and launch the Agrolink frontend in your browser at: <http://localhost:5174>

Agrolink uses JSON Server as a lightweight backend for storing:

- Farmer details
- Customer details
- Products
- Cart items

To start the JSON Server, run:

```
npx json-server --watch db.json --port 5000
```

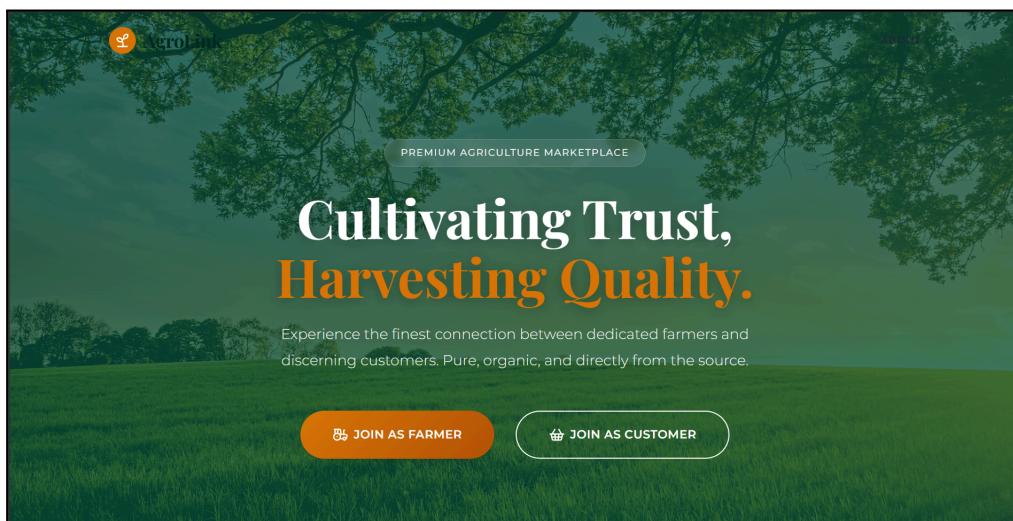
This will start the backend API at: <http://localhost:5000>

The API exposes useful endpoints such as:

- /users
- /products
- /cart

**Below are some screenshots representing the different components and pages of the application:**

**LandingPage Component:** The Landing Page welcomes users to Agrolink and provides quick access to customer and farmer sections with a clean, engaging interface



 AgroLink

ABOUT Surya  

## About AgroLink

We are on a mission to revolutionize the agricultural supply chain by eliminating middlemen and ensuring fair prices for farmers while delivering the freshest organic produce to customers.

 **Community First**  
Building a strong community of local farmers and conscious consumers who care about quality and sustainability.

 **Sustainable Future**  
Promoting eco-friendly farming practices that protect our soil and environment for future generations.

 **Quality Guaranteed**  
Every product is vetted for quality. We ensure that only the best, freshest produce reaches your table.



**Registration Component(Farmer,Customer):** The Registration Page allows new farmers and customers to create an account by entering their basic details, which are stored through JSON Server.

 AgroLink

ABOUT

### Join as Farmer

**FULL NAME**

**PHONE NUMBER**

**LOCATION (CITY/VILLAGE)**

**FARM NAME (OPTIONAL)**

**REGISTER & CONTINUE**

© 2025 AgroLink. Connecting Farmers and Customers.

 AgroLink

ABOUT

### Join as Customer

**FULL NAME**

**PHONE NUMBER**

**LOCATION (CITY/VILLAGE)**

**REGISTER & CONTINUE**

© 2025 AgroLink. Connecting Farmers and Customers.

**Farmer Dashboard Component:** The Farmer Dashboard enables farmers to manage their products—add, edit, delete—and view all their listings in one place.

**Welcome, afthab**

My Products

Rice (₹55/kg) | gauva (₹50/bunch)

**Logout**

**Welcome, afthab**

+ ADD NEW PRODUCT

Add New Product

PRODUCT NAME:

PRICE (₹):  UNIT: Per kg

PRODUCT IMAGE:  UPLOAD IMAGE

ADD PRODUCT

**Logout**

**Welcome, afthab**

My Profile

afthab FARMER

PHONE: 09182226362

LOCATION: Mangalagiri

FARM NAME: AFTHAB FARMS

+ ADD NEW PRODUCT

**Logout**

**Customer Dashboard Component:** The Customer Dashboard displays all available farm products, allowing customers to browse, search, and buy directly from farmers.

The screenshot shows the homepage of the AgroLink platform. At the top, a banner reads "Fresh from the Farm to Your Table" with a subtext "Discover the finest organic produce from local farmers." Below the banner is a search bar with placeholder text "Search for vegetables, fruits". A "MY PROFILE" button is located in the top right corner. The main section is titled "Featured Products" and displays three items: "Rice" (₹55/kg), "APPLE" (₹100/kg), and "MANGO" (₹150/kg). Each product card includes a small image, the name, the farmer's name, and a "ADD TO CART" button.

The screenshot shows the "Your Cart" page. At the top, it says "AgroLink" and has links for "ABOUT", "Surya", a shopping cart icon, and a profile icon. Below that is a "Back to Dashboard" link. The main area is titled "Your Cart" and shows a single item: "Rice" (₹55) with a quantity of 1. To the right is an "Order Summary" table with rows for Subtotal (₹55), Delivery Fee (₹40), and Total (₹95). A "PROCEED TO PAY" button is at the bottom.

The screenshot shows the "Order Placed!" page. It features a large green checkmark icon and the title "Order Placed!". Below it is a message: "Thank you for supporting local farmers. Your fresh produce will be delivered soon." A "CONTINUE SHOPPING" button is at the bottom.

The screenshot shows the "My Profile" page. At the top, it says "AgroLink" and has links for "ABOUT", "Surya", a shopping cart icon, and a profile icon. Below that is a "BACK" button and an "EDIT PROFILE" button. The main area shows a circular placeholder for a profile picture, the name "Surya", and the title "CUSTOMER". Below that are two sections: "PHONE" with the number "9669698745" and "LOCATION" with the city "Nellore".

## 10. Project Demo Links

**Project Demo Explanation:**

[https://drive.google.com/file/d/1Iafq9OM0LkmV6XTufuTdZhgHkwVW7T4u/view?usp=drive\\_link](https://drive.google.com/file/d/1Iafq9OM0LkmV6XTufuTdZhgHkwVW7T4u/view?usp=drive_link)

**Project Code Explanation:**

[https://drive.google.com/file/d/15q-5htwWOFzutZwB-7yJ0Hs2xxPKvEz\\_/view?usp=sharing](https://drive.google.com/file/d/15q-5htwWOFzutZwB-7yJ0Hs2xxPKvEz_/view?usp=sharing)

**Github Link:** <https://github.com/mohammedafthabrazahussain/AGROLINK-FRONTEND>