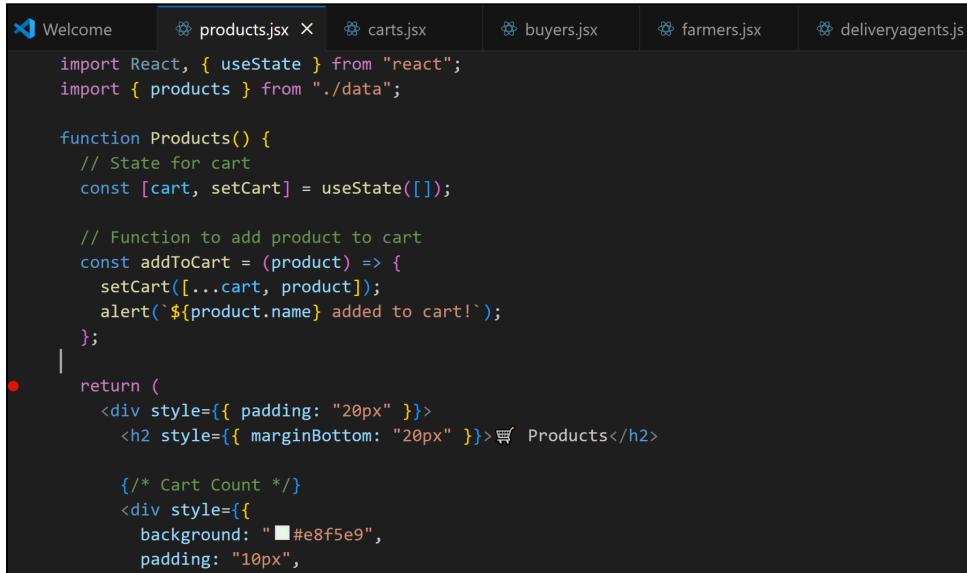


SOURCE CODE:



A screenshot of a code editor showing the `products.jsx` file. The tab bar at the top includes `Welcome`, `products.jsx` (which is the active tab), `carts.jsx`, `buyers.jsx`, `farmers.jsx`, and `deliveryagents.jsx`. The code itself is a functional component named `Products` that imports React and useState from react, and products from ./data. It uses useState to manage a cart state and a function `addCart` to add products to it. The component returns a heading and a styled div containing the product count and a grid of products.

```
import React, { useState } from "react";
import { products } from "./data";

function Products() {
    // State for cart
    const [cart, setCart] = useState([]);

    // Function to add product to cart
    const addCart = (product) => {
        setCart([...cart, product]);
        alert(`"${product.name}" added to cart!`);
    };

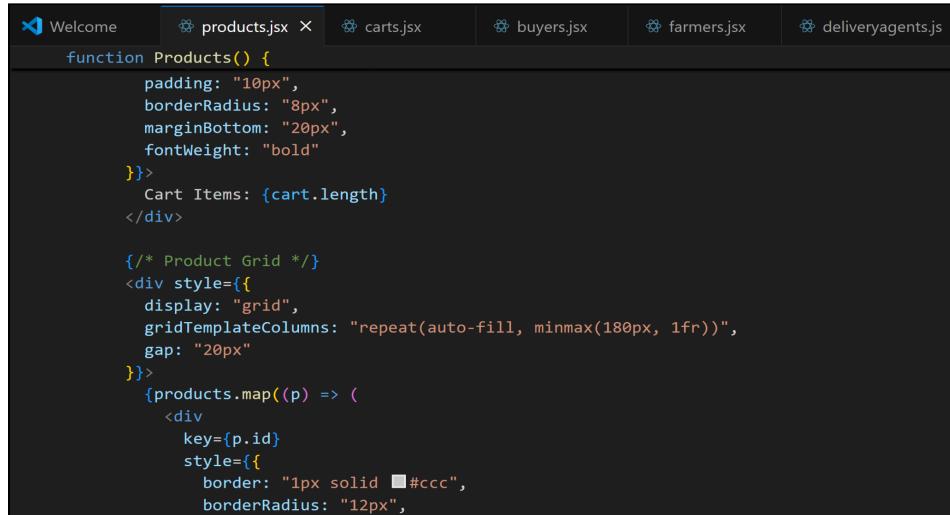
    return (
        <div style={{ padding: "20px" }}>
            <h2 style={{ marginBottom: "20px" }}>🛒 Products</h2>

            {/* Cart Count */}
            <div style={{
                background: "#e8f5e9",
                padding: "10px",
                border: "1px solid #ccc",
                border-radius: "8px",
                margin-bottom: "20px",
                fontWeight: "bold"
            }}>
                Cart Items: {cart.length}
            </div>

            {/* Product Grid */}
            <div style={{
                display: "grid",
                gridTemplateColumns: "repeat(auto-fill, minmax(180px, 1fr))",
                gap: "20px"
            }}>
                {products.map((p) => (
                    <div
                        key={p.id}
                        style={{
                            border: "1px solid #ccc",
                            border-radius: "12px",
                            padding: "10px",
                            width: "160px",
                            height: "140px"
                        }}>
                        {p.name} - {p.description} - {p.price}
                    </div>
                ))
            </div>
        </div>
    );
}

export default Products;
```

Figure 1.1



A screenshot of a code editor showing the `products.jsx` file. The tab bar at the top includes `Welcome`, `products.jsx` (which is the active tab), `carts.jsx`, `buyers.jsx`, `farmers.jsx`, and `deliveryagents.jsx`. The code is identical to Figure 1.1, showing the `Products` component definition.

```
function Products() {
    padding: "10px",
    borderRadius: "8px",
    marginBottom: "20px",
    fontWeight: "bold"
}>
    Cart Items: {cart.length}
</div>

/* Product Grid */
<div style={{
    display: "grid",
    gridTemplateColumns: "repeat(auto-fill, minmax(180px, 1fr))",
    gap: "20px"
}}>
    {products.map((p) => (
        <div
            key={p.id}
            style={{
                border: "1px solid #ccc",
                border-radius: "12px",
                padding: "10px",
                width: "160px",
                height: "140px"
            }}>
                {p.name} - {p.description} - {p.price}
            </div>
    ))
</div>

```

Figure 1.2

```
function Products() {
  {products.map((p) => (
    /* Product Image */
    <img
      src={p.image}
      alt={p.name}
      style={{ width: "100px", height: "100px", objectFit: "cover", borderRadius: "8px" }}
    />

    /* Product Info */
    <h3 style={{ margin: "10px 0 5px" }}>{p.name}</h3>
    <p style={{ margin: "5px 0" }}>₹ {p.price}</p>
    <p style={{ margin: "5px 0", color: "green" }}>{'Freshness'} {p.freshness}</p>
  ))}
}
```

Figure 1.3

```
function Products() {
  {products.map((p) => (
    /* Add to Cart Button */
    <button
      onClick={() => addToCart(p)}
      style={{
        marginTop: "10px",
        padding: "8px 12px",
        background: "#4CAF50",
        color: "white",
        border: "none",
        borderRadius: "8px",
        cursor: "pointer"
      }}
    >
      Add to Cart
    </button>
  </div>
  )));
}
</div>
</div>
```

Figure 1.4

```
import React, { useState } from "react";
import { products } from "./data";

function Cart() {
  // Preload cart with 2 sample products for demo
  const [cart, setCart] = useState([
    { ...products[0], quantity: 1 },
    { ...products[1], quantity: 2 },
  ]);

  // Increase quantity
  const increaseQty = (id) => {
    setCart(cart.map(item =>
      item.id === id ? { ...item, quantity: item.quantity + 1 } : item
    ));
  };

  // Decrease quantity
  const decreaseQty = (id) => {
    setCart(cart.map(item =>
      item.id === id && item.quantity > 1
        ? { ...item, quantity: item.quantity - 1 }
        : item
    ));
  };
}

export default Cart;
```

Figure 1.5

A screenshot of a code editor showing a file named `carts.jsx`. The code defines a `Cart` function that returns a JSX component. It includes methods for decreasing quantity, removing items, and calculating the total price.

```
function Cart() {
  const decreaseQty = (id) => {
    setCart(cart.map(item =>
      item.id === id && item.quantity > 1
        ? { ...item, quantity: item.quantity - 1 }
        : item
    )));
  };

  // Remove item from cart
  const removeItem = (id) => {
    setCart(cart.filter(item => item.id !== id));
  };

  // Calculate total
  const total = cart.reduce((sum, item) => sum + item.price * item.quantity, 0);

  return (
    <div style={{ padding: "20px" }}>
      <h2>🛒 Cart / Checkout</h2>
    </div>
  );
}
```

Figure 1.6

A screenshot of a code editor showing a file named `carts.jsx`. The code defines a `Cart` function that returns a JSX component. It checks if the cart is empty and displays a message or a list of products with their details.

```
function Cart() {
  if(cart.length === 0) {
    <p>Your cart is empty.</p>
  } else {
    <div>
      <div>
        {cart.map(item => (
          <div key={item.id} style={{
            display: "flex",
            alignItems: "center",
            justifyContent: "space-between",
            padding: "10px",
            borderBottom: "1px solid #ddd"
          }>
            /* Product Info */
            <div style={{ display: "flex", alignItems: "center", gap: "10px" }>
              <img src={item.image}>
            </div>
          </div>
        ))
      </div>
    </div>
  }
}
```

Figure 1.7

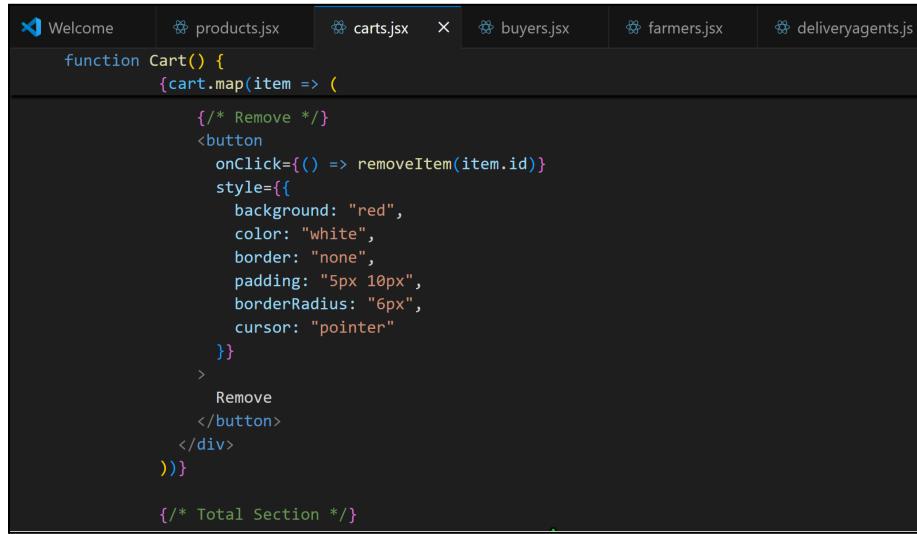
A screenshot of a code editor showing a file named `carts.jsx`. The code defines a `Cart` function that returns a JSX component. It displays product details and includes buttons for adjusting quantity and removing items.

```
function Cart() {
  cart.map(item => (
    <div>
      <img alt={item.name} style={{ width: "60px", height: "60px", borderRadius: "6px", objectFit: "cover" }} />
      <div>
        <h4>{item.name}</h4>
        <p>$ {item.price} x {item.quantity}</p>
      </div>
    </div>
  ));

  /* Quantity Controls */
  <div>
    <button onClick={() => decreaseQty(item.id)}>-</button>
    <span style={{ margin: "0 10px" }}>{item.quantity}</span>
    <button onClick={() => increaseQty(item.id)}>+</button>
  </div>

  /* Remove */
  <button>Remove</button>
}
```

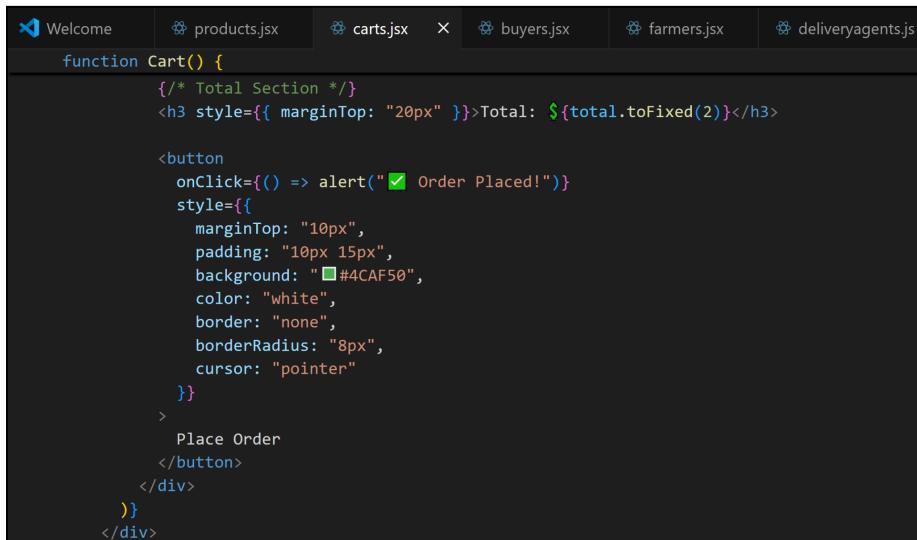
Figure 1.8



```
function Cart() {
  {cart.map(item => (
    /* Remove */
    <button
      onClick={() => removeItem(item.id)}
      style={{
        background: "red",
        color: "white",
        border: "none",
        padding: "5px 10px",
        borderRadius: "6px",
        cursor: "pointer"
      }}
    >
      Remove
    </button>
  </div>
))}

/* Total Section */
```

Figure 1.9



```
function Cart() {
  /* Total Section */
  <h3 style={{ marginTop: "20px" }}>Total: ${total.toFixed(2)}</h3>

  <button
    onClick={() => alert("✓ Order Placed!" )}
    style={{
      marginTop: "10px",
      padding: "10px 15px",
      background: "#4CAF50",
      color: "white",
      border: "none",
      borderRadius: "8px",
      cursor: "pointer"
    }}
  >
    Place Order
  </button>
</div>
)}
```

Figure 1.10

```
>Welcome products.jsx carts.jsx buyers.jsx X farmers.jsx deliveryagents.jsx

import React from "react";
import { buyers } from "./data";

function Buyers() {
  return (
    <div style={{ padding: "20px" }}>
      <h2>Buyers</h2>

      <div
        style={{
          display: "grid",
          gridTemplateColumns: "repeat(auto-fill, minmax(280px, 1fr))",
          gap: "20px"
        }}
      >
        {buyers.map((b) => (
          <div
            key={b.id}
            style={{
              border: "1px solid #ddd",
              borderRadius: "12px",
              padding: "15px",
              background: "#fefefe",
              boxShadow: "0 3px 8px rgba(0,0,0,0.1)"
            })
          >
            /* Buyer Info */
            <h3 style={{ margin: "5px 0", color: "#2e7d32" }}>{b.name}</h3>
            <p style={{ margin: "5px 0" }}>
              <b>Email:</b> {b.email}
            </p>
            <p style={{ margin: "5px 0" }}>
              <b>Address:</b> {b.address}
            </p>
            /* Orders */
            <div style={{ marginTop: "10px" }}>
```

Figure 1.11

```
>Welcome products.jsx carts.jsx buyers.jsx X farmers.jsx deliveryagents.jsx

function Buyers() {
  {buyers.map((b) => (
    <div
      style={{
        border: "1px solid #ddd",
        borderRadius: "12px",
        padding: "15px",
        background: "#fefefe",
        boxShadow: "0 3px 8px rgba(0,0,0,0.1)"
      })
    >
      /* Buyer Info */
      <h3 style={{ margin: "5px 0", color: "#2e7d32" }}>{b.name}</h3>
      <p style={{ margin: "5px 0" }}>
        <b>Email:</b> {b.email}
      </p>
      <p style={{ margin: "5px 0" }}>
        <b>Address:</b> {b.address}
      </p>
      /* Orders */
      <div style={{ marginTop: "10px" }}>
```

Figure 1.12

A screenshot of a code editor showing the `buyers.jsx` file. The code defines a `Buyers` component that maps over an array of buyers. For each buyer, it displays their name and a list of their orders. The orders are presented in a grid format with a gap of 20px between columns.

```
function Buyers() {
  {buyers.map((b) => (
    /* Orders */
    <div style={{ marginTop: "10px" }}>
      <b>Orders:</b>
      <ul style={{ paddingLeft: "18px", margin: "5px 0" }}>
        {b.orders.map((orderId, idx) => (
          <li key={idx}>{orderId}</li>
        )))
      </ul>
    </div>
  ));
}

export default Buyers;
```

Figure 1.13

A screenshot of a code editor showing the `farmers.jsx` file. The code defines a `Farmers` component that imports React and a `farmers` array from a local file. It displays a heading and a grid of farmer profiles, where each profile is a div with a border and rounded corners.

```
import React from "react";
import { farmers } from "./data";

function Farmers() {
  return (
    <div style={{ padding: "20px" }}>
      <h2>👨‍🌾 Farmers</h2>

      <div
        style={{
          display: "grid",
          gridTemplateColumns: "repeat(auto-fill, minmax(250px, 1fr))",
          gap: "20px"
        }}
      >
        {farmers.map((f) => (
          <div
            key={f.id}
            style={{
              border: "1px solid #ccc",
              borderRadius: "12px",
              padding: "10px"
            })
        ))}
      </div>
    </div>
  );
}

export default Farmers;
```

Figure 1.14

```
function Farmers() {
  {farmers.map((f) => (
    borderRadius: "12px",
    padding: "15px",
    background: "#fefefe",
    boxShadow: "0 3px 8px rgba(0,0,0,0.1)",
    transition: "0.3s"
  )})
  >
  {/* Farmer Profile */}
  <div style={{" display: "flex", alignItems: "center", gap: "10px" }}>
    <img
      src={f.profileImage}
      alt={f.name}
      style={{{
        width: "60px",
        height: "60px",
        borderRadius: "50%",
        objectFit: "cover",
        border: "2px solid #4CAF50"
      }}}
  
```

Figure 1.15

```
function Farmers() {
  {farmers.map((f) => (
    border: "2px solid #4CAF50"
  )})
  >
  <div>
    <h3 style={{ margin: "5px 0" }}>{f.name}</h3>
    <p style={{ margin: 0, color: "gray" }}>{f.location}</p>
  </div>
</div>

/* Farm Info */
<div style={{ marginTop: "10px" }}>
  <p><b>Farm:</b> {f.farm}</p>
  <img
    src={f.farmImage}
    alt={f.farm}
    style={{{
      width: "100%",
      height: "150px",
      borderRadius: "8px",
    }}}
  
```

Figure 1.16

```
function Farmers() {
    {farmers.map((f) => (
        <div>{f.name}</div>
        <div style={{
            width: "100%",
            height: "150px",
            borderRadius: "8px",
            objectFit: "cover",
            marginTop: "10px"
        }}>
        </div>
    ))}
    </div>
</div>
);
}

export default Farmers;
```

Figure 1.17

MONGODB USING MOONGOOSE:

The screenshot shows the MongoDB Compass interface connected to the 'greenmart' database. The left sidebar lists connections, and the main area displays five collections: buyers, deliveryAgents, farmers, orders, and products. Each collection card provides storage statistics.

Collection	Storage size	Documents	Avg. document size	Indexes	Total index size
buyers	4.10 kB	2	140.00 B	1	4.10 kB
deliveryAgents	4.10 kB	4	313.00 B	1	4.10 kB
farmers	4.10 kB	1	975.00 B	1	4.10 kB
orders	4.10 kB	1	286.00 B	1	4.10 kB
products	20.48 kB	5	264.00 B	1	20.48 kB

Figure 2.1

The screenshot shows the MongoDB interface for the 'deliveryAgents' collection. The top navigation bar includes 'Greenmart > greenmart > deliveryAgents' and an 'Open MongoDB shell' button. Below the navigation is a toolbar with 'Documents 4', 'Aggregations', 'Schema', 'Indexes 1', and 'Validation'. A search bar says 'Type a query: { field: 'value' } or Generate query'. Below the search are buttons for 'ADD DATA', 'EXPORT DATA', 'UPDATE', and 'DELETE'. A status bar at the bottom shows '25 1-4 of 4'.

```

_id: "d1"
name: "Sarah Johnson"
phone: "+1-555-111-2222"
profileImage: "https://images.unsplash.com/photo-1595152772835-219674b2a8a0"
rating: 4.9
totalDeliveries: 245
status: "available"
avgTime: 25
serviceAreas: Array (3)
assignedOrders: Array (1)

_id: "d2"
name: "Mike Chen"
phone: "+1-555-333-4444"
profileImage: "https://images.unsplash.com/photo-1607746882042-944635df1e0e"
rating: 4.8
totalDeliveries: 189
status: "busy"
avgTime: 30

```

Figure 2.2

The screenshot shows the MongoDB interface for the 'buyers' collection. The top navigation bar includes 'Greenmart > greenmart > buyers' and an 'Open MongoDB shell' button. Below the navigation is a toolbar with 'Documents 2', 'Aggregations', 'Schema', 'Indexes 1', and 'Validation'. A search bar says 'Type a query: { field: 'value' } or Generate query'. Below the search are buttons for 'ADD DATA', 'EXPORT DATA', 'UPDATE', and 'DELETE'. A status bar at the bottom shows '25 1-2 of 2'.

```

_id: "b1"
name: "Olivia Brown"
email: "michael@example.com"
address: "123 Main Street, California, USA"
orders: Array (2)

_id: "b2"
name: "Liam Wilson"
email: "sarah@example.com"
address: "456 Oak Avenue, Texas, USA"
orders: Array (1)

```

Figure 2.3

The screenshot shows the MongoDB interface for the 'farmers' collection. The top navigation bar includes 'Greenmart > greenmart > farmers' and an 'Open MongoDB shell' button. Below the navigation is a toolbar with 'Documents 1', 'Aggregations', 'Schema', 'Indexes 1', and 'Validation'. A search bar says 'Type a query: { field: 'value' } or Generate query'. Below the search are buttons for 'ADD DATA', 'EXPORT DATA', 'UPDATE', and 'DELETE'. A status bar at the bottom shows '25 1-1 of 1'.

```

_id: "f1"
name: "John Doe"
farmName: "Green Valley Farm"
location: "Mumbai, India"
categories: Array (5)
certifications: Array (5)
profileImage: "https://images.unsplash.com/photo-1592928302603-3b7f1aaef0c3b"
farmImage: "https://images.unsplash.com/photo-1501004318641-b39e6451bec6"
contact: Object
description: "Sunny Fields Farm is a family-owned farm committed to providing fresh,..."
```

Figure 2.4

```

_id: "o1"
buyerId: "b1"
farmerId: "f1"
products: Array (4)
totalAmount: 16
status: "Pending"
deliveryAgentId: "d1"

```

Figure 2.5

	_id	name	category	quantity
1	"p1"	"Baby Carrots"	"Vegetable"	50
2	"p2"	"Cherry Tomatoes"	"Vegetable"	40
3	"p3"	"Fresh Broccoli"	"Vegetable"	30
4	"p4"	"Honey Crisp Apples"	"Fruit"	20
5	"p5"	"Microgreens Mix"	"Microgreens"	25

Figure 2.6

```

_id: "p1"
name: "Baby Carrots"
category: "Vegetable"
quantity: 50
basePrice: 2
currentPrice: 2.5
harvestDate: "2025-09-20"
expiryDate: "2025-09-30"
imageUrl: "https://images.unsplash.com/photo-1567306226416-28f0efdc88ce"
farmerId: "f1"

_id: "p2"
name: "Cherry Tomatoes"
category: "Vegetable"
quantity: 40
basePrice: 3
currentPrice: 3.5
harvestDate: "2025-09-21"
expiryDate: "2025-09-29"

```

Figure 2.7

OUTPUT:

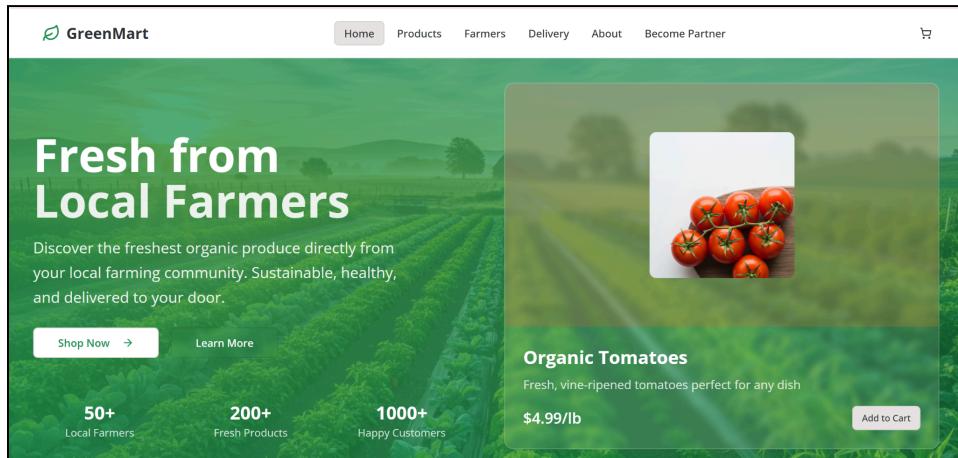


Figure 3.1

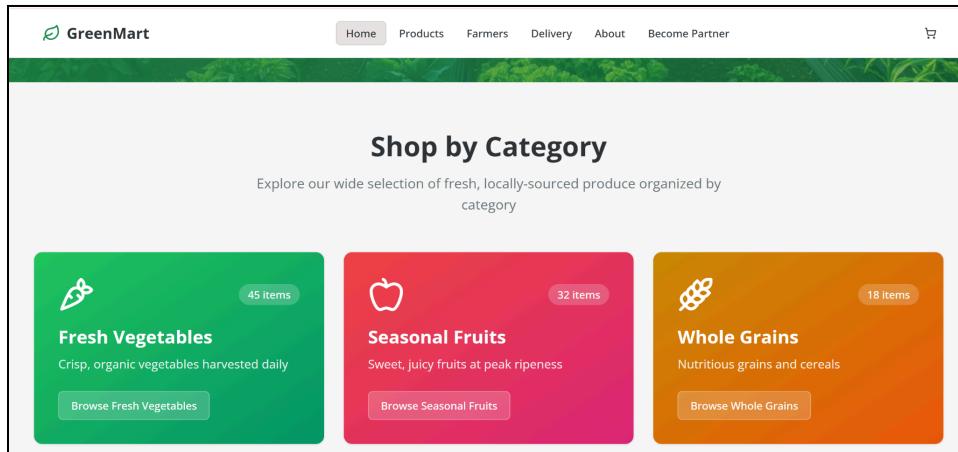


Figure 3.2

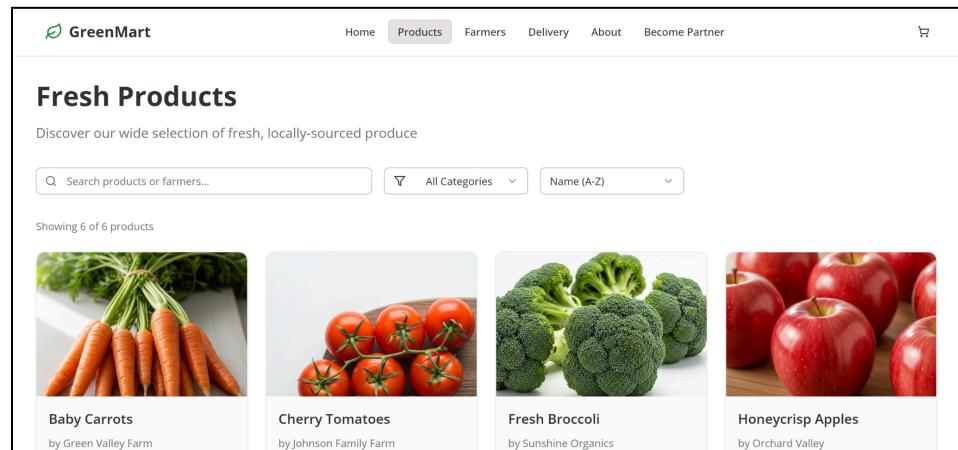


Figure 3.3

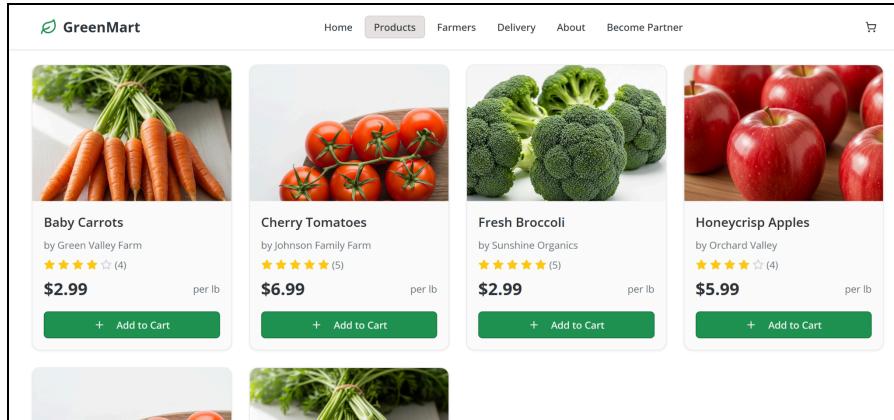


Figure 3.4

Meet Our Farmers

Get to know the dedicated farmers who grow your food with passion, expertise, and commitment to sustainability

Farmer	Location	Experience	Description
John Peterson Verified	Green Valley, CA	15 years	Committed to sustainable farming practices and providing the freshest produce to our local community.
Maria Rodriguez Verified	Sunshine Valley, CA	12 years	Third-generation farmer specializing in heirloom varieties and regenerative agriculture practices.
David Chen	Mountain View, CA	20 years	Family orchard focusing on seasonal fruit varieties and sustainable water management.

Figure 3.5

Our Delivery Team

Meet the dedicated professionals who ensure your fresh produce arrives quickly and safely at your doorstep

Service	Description
Fast Delivery	Average delivery time of 25 minutes from local farms to your door
Fresh Transport	Temperature-controlled vehicles to maintain freshness during delivery
Wide Coverage	Serving over 15 neighborhoods with expanding coverage area
Safe & Secure	Background-checked drivers with contactless delivery options

Our Delivery Agents

Figure 3.6

The screenshot shows the 'Our Delivery Agents' section of the GreenMart website. It displays four delivery agents in cards:

- Sarah Johnson**: Available, 4.9 (245 deliveries), Avg. Time: 25 min, Deliveries: 245+. Service areas: Downtown, Green Valley, Riverside.
- Mike Chen**: Busy, 4.8 (189 deliveries), Avg. Time: 30 min, Deliveries: 189+. Service areas: Westside, University, Oak Ridge.
- Emily Rodriguez**: Available, 4.9 (312 deliveries), Avg. Time: 22 min, Deliveries: 312+. Service areas: Northside, Coastal Valley, Heights.
- David Thompson**: Offline, 4.7 (156 deliveries), Avg. Time: 28 min, Deliveries: 156+. Service areas: Mountain View, Pine Hills, Cedar Park.

Figure 3.7

The screenshot shows the 'Service Areas' section of the GreenMart website. It displays two groups of service areas:

- Left group: Northside, Coastal Valley, Heights.
- Right group: Mountain View, Pine Hills, Cedar Park.

Below these groups is a central section titled 'Service Areas' containing the following neighborhoods:

Downtown	Green Valley	Riverside	Westside
University	Oak Ridge	Northside	Coastal Valley
Heights	Mountain View	Pine Hills	Cedar Park

Text at the bottom: "Don't see your area? Contact us to request delivery service expansion."

Figure 3.8

The screenshot shows the 'Our Mission & Values' section of the GreenMart website. It features three cards:

- Sustainable Farming**: Supporting eco-friendly farming practices that protect our environment for future generations. Icon: Leaf.
- Community First**: Connecting local farmers directly with consumers to strengthen community bonds and local economy. Icon: People.
- Fresh Delivery**: Farm-to-door delivery ensuring the freshest produce reaches your table within hours of harvest. Icon: Truck.

Text above the cards: "GreenMart is more than just a marketplace. We're building a sustainable future where fresh, local produce is accessible to everyone while supporting our farming communities."

Figure 3.9

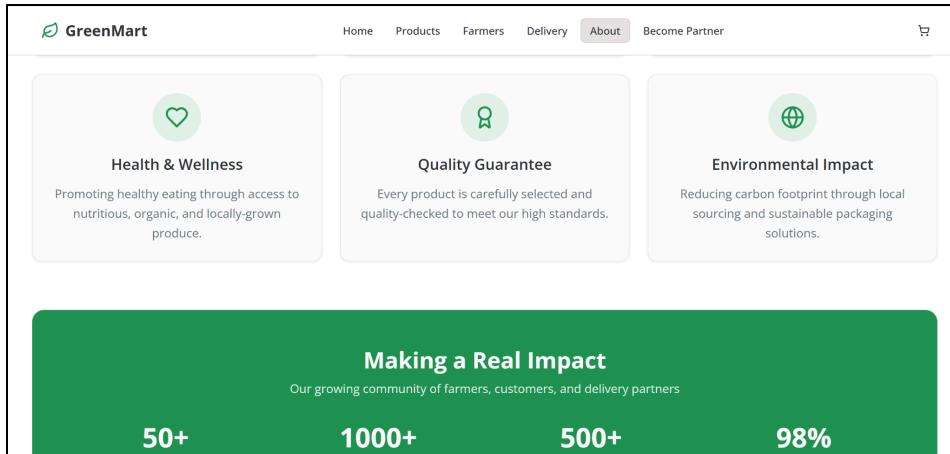


Figure 3.10

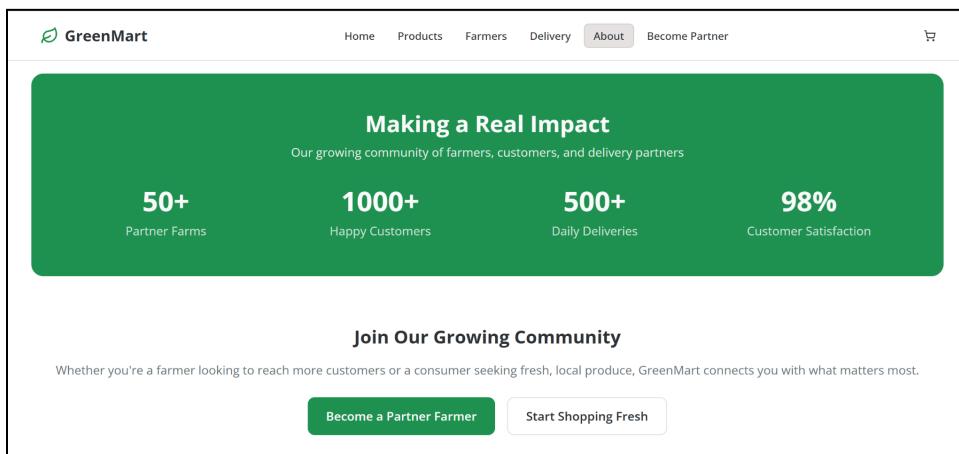


Figure 3.11

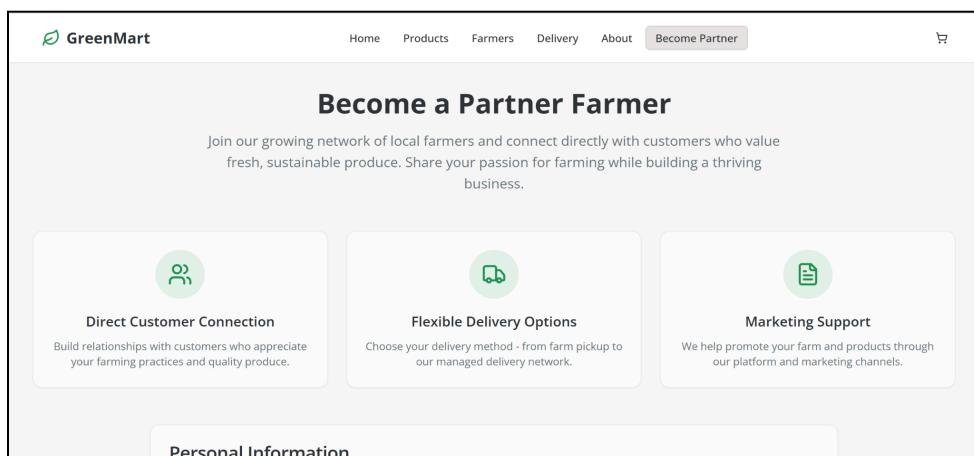


Figure 3.12

Personal Information

Full Name Farm Name

Email Address Phone Number

Profile Image

(Profile image uploaded)

Upload a professional photo of yourself for your farmer profile

Figure 3.13

Farm Information

Farm Location Farm Size

Complete Farm Address

Farming Experience

Farm Image

(Farm image uploaded)

Upload a photo showcasing your farm or main growing area

Figure 3.14

Farm Description

Sunny Fields Farm is a family-owned farm committed to providing fresh, organic, and pesticide-free produce. Nestled in the fertile valleys of California, our farm grows a wide variety of fruits, vegetables, and herbs using sustainable practices. We believe in connecting local communities with healthy, natural food while supporting eco-friendly farming. Every product from our farm is hand-picked to ensure quality, freshness, and taste.

Minimum 50 characters. This will be displayed on your farmer profile.

Products & Certifications

Product Categories

Select all categories that apply to your farm production

<input checked="" type="checkbox"/> Vegetables	<input checked="" type="checkbox"/> Fruits
<input checked="" type="checkbox"/> Herbs	<input checked="" type="checkbox"/> Grains
<input type="checkbox"/> Legumes	<input type="checkbox"/> Dairy
<input type="checkbox"/> Eggs	<input type="checkbox"/> Honey
<input type="checkbox"/> Flowers	<input checked="" type="checkbox"/> Microgreens

Certifications (Optional)

Figure 3.15

Certifications (Optional)
Select any certifications your farm currently holds

- USDA Organic
- Biodynamic
- Rainforest Alliance
- Non-GMO
- Certified Naturally Grown
- Fair Trade
- Local Organic

Operations

Delivery Capacity

Regional Delivery (within 15 miles)

Sustainable Practices (Optional)

Our farm follows sustainable practices to protect the environment and promote healthy growth. We use organic fertilizers, crop rotation, and natural pest control methods to reduce chemical usage. Water conservation techniques, such as drip irrigation and rainwater harvesting, ensure minimal waste. We focus on maintaining soil health, supporting biodiversity, and reducing carbon footprint, so our produce is not only fresh and healthy but also eco-friendly.

Tell customers about your environmental practices (composting, water conservation, etc.)

Figure 3.16

Tell customers about your environmental practices (composting, water conservation, etc.)

I agree to the terms and conditions and partner agreement
By checking this box, you agree to our partner terms, quality standards, and marketplace policies.

Submit Partner Application

Have Questions?

Our farmer partnership team is here to help you through the application process.

Email: partners@greenmart.com
Phone: (555) 123-FARM
Hours: Monday-Friday, 9AM-6PM PST

Figure 3.17

Application Under Review

Application Details

Applicant:	John Doe
Farm Name:	Green Valley Farm
Email:	johnFarm@gmail.com
Submitted:	9/27/2025

What's Next?

Document Review

Figure 3.18

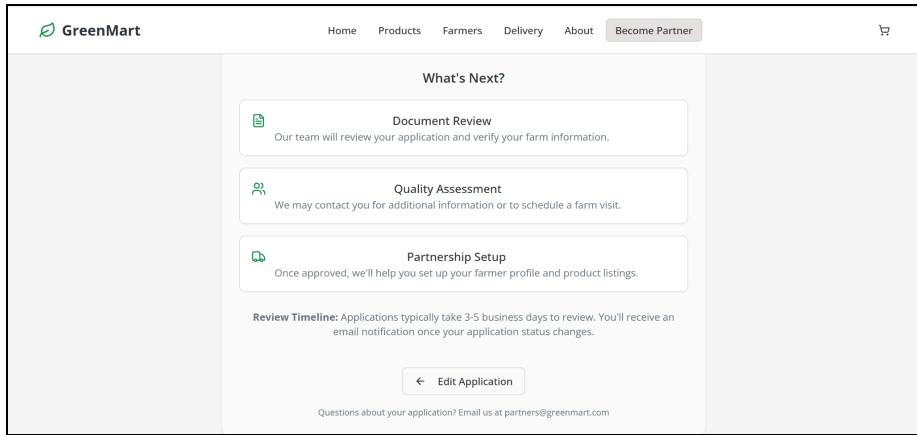


Figure 3.19

Shopping Cart (5 items)

	Baby Carrots \$2.99 per lb <small>Good</small>	-	2	+	\$5.98	Remove
	Cherry Tomatoes \$6.99 per lb <small>Premium</small>	-	1	+	\$6.99	Remove
	Fresh Broccoli \$2.99 per lb <small>Premium</small>	-	1	+	\$2.99	Remove

Order Summary

Subtotal	\$21.95
Delivery Fee	\$4.99
Tax	\$1.76
Total	\$28.70

Delivery Address
Enter your address

Special Instructions
Leave at door, etc.

Figure 3.20

Order Summary

Subtotal	\$21.95
Delivery Fee	\$4.99
Tax	\$1.76
Total	\$28.70

Delivery Address
Mumbai

Special Instructions
Dont ring the bell

Place Order →

This is a demo checkout. No real payment will be processed.

d4499cbe-bc0c-4076-bef7-d7570e621bab-00-25t96bemg9xs7.spock.replit.dev says

Order placed successfully! (This is a demo)

OK

Figure 3.21

30% EXTRA CONTRIBUTION:

- RBAC & Dynamic Rendering: Multi-tier access control with JWT-driven component injection.
- Reactive State Management: Atomic global state via React Context; supports optimistic UI updates.
- Data-Driven UI: Freshness and discount flags computed from product metadata; real-time rendering.
- Modular Architecture: Decoupled, reusable components adhering to SOLID principles.
- Async Media Handling: Remote image URLs with lazy-loading and caching for optimized performance.
- FSM Delivery Workflow: Deterministic delivery state transitions.
- Client Validation & Feedback: Defensive validation with reactive alerts for robust UX.
- Full-Stack Orchestration: React SPA integrated with Express.js REST APIs and Mongoose models; real-time data sync.