A REPORT

ON

Grocerease: A Smart Solution for Seamless Grocery Shopping

By

R. Abhiram Chowdary	AP23110010260
T. Bala Manindra	AP23110010293
V. Mohith Chowdary	AP23110010226
Y. Teja Swaroop	AP23110010282

Prepared in the partial fulfillment of the

FULLSTACK

AT

SMART BRIDGE

SRM UNIVERSITY, AP

(December, 2025)

Introduction

GrocerEase is a modern, responsive online grocery shopping platform built using:

- **React.js (with Vite)**
- **React Router**
- **Tailwind CSS**
- **React Context API**
- **LocalStorage / SessionStorage** for data persistence

The platform gives users a smooth shopping experience with:

- Real-time product browsing with search and category filter
- Cart management
- Basic checkout summary
- Role-based login (Admin & Customer)
- Admin product management with full **CRUD** (Create, Read, Update, Delete)

All product and cart data is stored in the browser using **LocalStorage**, simulating a lightweight backend without needing a real server.

The project is designed to be:

- **Simple & clean** – easy to understand for academic evaluation
- **Fast & responsive** – Tailwind + Vite
- **Extendable** – easy to attach a real backend later

Scenario-Based Intro

Customer View

Imagine a customer opening the GrocerEase website.

They first see a **login page**. After logging in as a customer, they are taken to the **Home / Shop** page.

On the home page they see:

- A clean **navbar** with links to Home, Cart, and Login/Logout
- A **product grid** with cards showing:
    - Product image
    - Name and category
    - Price and unit (e.g., “per kg”, “per dozen”)
    - Short description
    - An **“Add to Cart”** button

Users can:

- Search products by name
- Filter by category (Fruits, Vegetables, Dairy, Snacks, etc.)
- Add items to their cart
- Increase/decrease quantity in the cart
- See a **live subtotal**
- Go to **Checkout** and view an order summary (mock checkout – no payment gateway)

Each logged-in customer has **their own cart**, stored separately using cartsMap in LocalStorage, so carts are user-specific.

Admin View

Now imagine an admin logging in with:

username: admin
password: admin123

The admin can access the **Admin Panel**.

From the **Admin Panel**, they can:

- View all products in a grid
- Add new products using an **Add Product Form**:
    - Name, price, category, unit, stock, image URL, description
- Edit existing products (prefilled form)
- Delete products permanently

Every change the admin makes is:

- **Saved to LocalStorage**
- Instantly reflected on the customer Home page

So the project demonstrates:

- Role-based access (admin vs customer)
- Shared state using Context

- Persistent data using LocalStorage
- Full CRUD on the frontend

Target Audience

GrocerEase is designed for:

1. **Everyday Users (Customers)**
    - To explore how an online grocery UI works from a user perspective.
2. **Store Owners / Admin Users**
    - To understand a basic product management dashboard.
3. **Frontend / Full-stack Students & Developers**
    - To practice React, React Router, Context API, Tailwind, and CRUD concepts.
4. **Academic Evaluators**
   To evaluate skills in:
    - UI/UX design
    - Component-based architecture
    - State management with Context
    - LocalStorage persistence
    - Basic authentication/authorization logic

Project Goals & Objectives

Goals

- Build a **fully functional grocery shopping interface**.
- Implement an **Admin CRUD panel** for product management.
- Use a **modern frontend stack** (React + Vite + Tailwind).
- Demonstrate **role-based access** and basic authentication.
- Keep the UI **clean, minimal, and responsive.**

Objectives

✓ User-Friendly Frontend

Provide simple pages for:

- **Login Page** – for both admin and customers
- **Home / Shop Page** – product grid with search and filters
- **Cart Page** – cart items, quantity update, remove item
- **Checkout Page** – order summary and total

✓ Admin CRUD Module

Admin can:

- **Create** – Add new products
- **Read** – View all products in a table/grid
- **Update** – Edit existing product details
- **Delete** – Remove products permanently

All operations update global state via **Context** and persist data to **LocalStorage**.

✓ Modern Tech Stack

- **React** (with Vite as bundler)
- **React Router** for client-side routing
- **Tailwind CSS** for styling
- **React Context API + Hooks** for state management
- **LocalStorage / SessionStorage** for data persistence

✓ Component Reusability

Reusable components like:

- Navbar
- ProductCard
- ProductGrid
- CartPanel
- AddProductForm
- LoginForm

✓ **Performance & UX**

- Fast dev/build with Vite
- Smooth, responsive layout with Tailwind
- Minimal re-renders using Context + hooks
- Persistent login and cart between refreshes

.

Key Features

1. Grocery Shopping Interface (Customer Side)

- **Product Grid** with cards showing image, name, price, unit, and description
- **Category-based browsing** using dropdown/filters
- **Search bar** to filter products by name
- Responsive design that works across laptop/tablet/mobile

2. Authentication & Role-Based Access

- **LoginForm**:
  - admin / admin123 → Admin role
  - Any other username + any password → Customer role
- **Role-based routing**:
  - Only admin can access /admin route
  - Unauthorized access redirects to home/login
- currentUser is stored in **SessionStorage** to keep user data while the tab is open.

3. Cart & Checkout System

- Add products to cart from ProductCard
- **CartPanel** page:
  - Increase / decrease quantity
  - Remove items from cart
  - Auto calculation of **subtotal**
- **Per-user carts** using cartsMap in LocalStorage
- **CheckoutPage**:
  - Shows order summary (items and total)
  - Text note: mock checkout (no real payment)

4. Admin Product Management (CRUD)

Admin Page includes:

- Product list with basic details
- **AddProductForm**:
  - Name, category, price, unit, stock, image URL, description

- Optional image preview from URL or upload
- **Edit product** by opening same form with initial data
- **Delete** product from list
- All operations update products state in Context and persist to LocalStorage.

5. Tailwind CSS UI Design

- Modern, minimal layout
- Rounded cards, shadows, and spacing
- Responsive layouts for product grid & cart
- Simple color palette (green/white for grocery theme)

6. Client-Side Routing & State Management

- **Routes** handled via React Router (examples):
  - / – Login / Landing
  - /home – Main shopping page (ProductGrid)
  - /cart – Cart page
  - /checkout – Checkout summary
  - /admin – Admin panel (protected)
- **State Management**:
  - Single StoreContext provider wrapping the entire app
  - Holds products, cartsMap, and currentUser
  - Provides helper methods: addNewProduct, updateProduct, deleteProduct, addToCart, updateQuantity, removeFromCart, loginUser, logoutUser, getCartFor

Pre-Requisites

✓ Node.js & npm

To run the React + Vite project.

✓ Vite + React Setup

Basic commands:

```
# install dependencies
npm install

# run in development mode
npm run dev

# build for production
npm run build

# preview production build
npm run preview
```

✓ Tailwind CSS

Already integrated via @tailwindcss/vite in vite.config.js and imported in index.css.

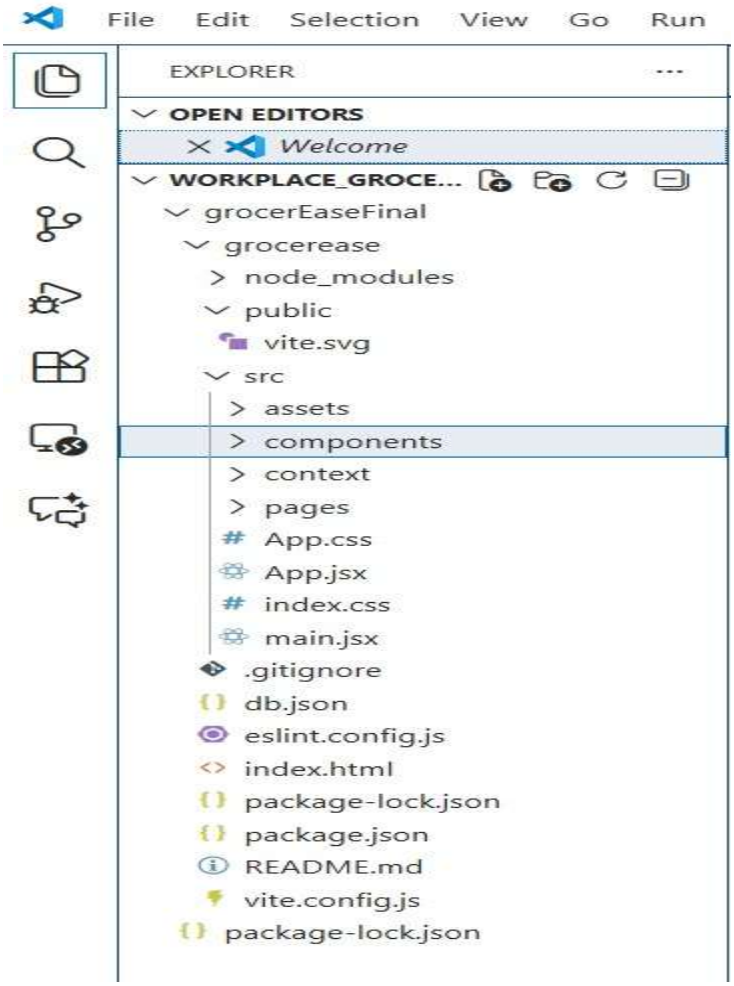
✓ **Browser Storage**

No additional backend setup is needed.

Products and carts are persisted using localStorage and current user using sessionStorage.

✅ No JSON-Server, no Axios – everything is handled on the frontend.

Project structure:



The **project structure shown above** is organized for a modern React.js application. It includes key directories such as src, components, pages, and assets, along with configuration files like vite.config.js, package.json, and eslint.config.js. The src folder contains the main application logic, while reusable UI parts are grouped under components, and separate screen-level views are maintained in pages. Public assets and routing entry files are also clearly defined to ensure a clean architecture.

The structure may vary based on the tools, frameworks, or architectural choices used in development. Maintaining a logical and consistent folder hierarchy helps improve readability, scalability, and team collaboration throughout the software lifecycle.

Project demo:
https://github.com/tejaswaroop2005/Grocerease1

Project Flow

Milestone 1 – Setup

- Initialize project with **Vite + React**
- Add **Tailwind CSS** and basic global styling
- Create StoreContext and wrap app in StoreProvider (main.jsx)
- Set up basic routes and Navbar

Milestone 2 – Customer UI

- Build **ProductCard** and **ProductGrid** components
- Add search and category filter for products
- Implement **CartPage** with CartPanel
- Implement **CheckoutPage** showing order summary

Milestone 3 – State & Persistence

- Create StoreContext with:
  - products, cartsMap, currentUser
  - CRUD helpers and cart operations
- Persist products and carts to **LocalStorage**
- Persist logged-in user to **SessionStorage**

Milestone 4 – Admin CRUD & Auth

- Implement **LoginForm** and loginUser / logoutUser
- Add role-based logic (admin vs customer)
- Implement **AdminPage**:
  - List products
  - Add new product
  - Edit existing product
  - Delete product
- Protect the /admin route so only admin role can access it.

PACKAGE.JSON:

package.json

grocerEaseFinal > grocerease > package.json > ...

```
{
  "name": "grocerease",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  ▷Debug
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "lint": "eslint .",
    "preview": "vite preview"
  },
  "dependencies": {
    "@tailwindcss/vite": "^4.1.17",
    "react": "^19.2.0",
    "react-dom": "^19.2.0",
    "react-router-dom": "^7.9.6",
    "tailwindcss": "^4.1.17"
  },
  "devDependencies": {
    "@eslint/js": "^9.39.1",
    "@types/react": "^19.2.5",
    "@types/react-dom": "^19.2.3",
    "@vitejs/plugin-react": "^5.1.1",
    "eslint": "^9.39.1",
    "eslint-plugin-react-hooks": "^7.0.1",
    "eslint-plugin-react-refresh": "^0.4.24",
    "globals": "^16.5.0",
    "vite": "^7.2.4"
  }
}

```

Main.jsx:

main.jsx

grocerEaseFinal > grocerease > src > main.jsx

```jsx
import { StrictMode } from "react";
import { createRoot } from "react-dom/client";
import App from "./App.jsx";
import "./index.css";
import StoreProvider from "./context/StoreContext";

createRoot(document.getElementById("root")).render(
  <StrictMode>
    <StoreProvider>
      <App />
    </StoreProvider>
  </StrictMode>
);

```

ProductCard.jsx:

ProductCard.jsx

grocerEaseFinal > grocerease > src > components > ProductCard.jsx > ...

```jsx
import { useState } from "react";
import { useStore } from "../context/StoreContext";

export default function ProductCard({ product }) {
  const { addToCart } = useStore();
  const [adding, setAdding] = useState(false);

  const handleAdd = () => {
    setAdding(true);
    addToCart(product);
    setTimeout(()=>setAdding(false), 700);
  };

  return (
    <div className="bg-white rounded-xl shadow p-4 flex flex-col">
      <div className="h-40 w-full mb-3 overflow-hidden rounded-lg">
        <img src={product.image || "https://via.placeholder.com/400x300"} alt={product.name} className="w-
      </div>

      <div className="flex-1">
        <h3 className="font-semibold text-lg">{product.name}</h3>
        <p className="text-sm text-gray-500">{product.unit || ""}</p>
        <p className="text-sm text-gray-600 mt-1">{product.description || ""}</p>
      </div>

      <div className="mt-4 flex items-center justify-between">
        <div className="text-green-700 font-bold text-lg">₹{product.price}</div>
        <button className="bg-green-600 text-white px-4 py-2 rounded flex items-center gap-2" onClick={har
          {adding ? "Added" : "Add to Cart"}
        </button>
      </div>
    </div>
  );
}

```

ProductGrid.jsx:

ProductGrid.jsx

grocerEaseFinal > grocerease > src > components > ProductGrid.jsx > ProductGrid

```jsx
import { useMemo, useState } from "react";
import { useStore } from "../context/StoreContext";
import ProductCard from "./ProductCard";
export default function ProductGrid() {
  const { products } = useStore();
  const [query, setQuery] = useState("");
  const [category, setCategory] = useState("All");
  const categories = useMemo(() => ["All", ...Array.from(new Set(products.map(p=>p.category)))], [products
  const filtered = products.filter(p => {
    const byName = p.name.toLowerCase().includes(query.toLowerCase());
    const byCat = category === "All" || p.category === category;
    return byName && byCat;
  });
  return (
    <div className="max-w-7xl mx-auto p-6 flex gap-6">
      <aside className="w-72 bg-white rounded-xl p-4 shadow h-fit">
        <h3 className="font-semibold mb-3">Categories</h3>
        <ul className="flex flex-col gap-2">
          {categories.map(c => (
            <li key={c}>
              <button className={`w-full text-left p-2 rounded ${category===c ? "bg-green-100" : "hover:bg
            </li>
          ))}
        </ul>
      </aside>
      <main className="flex-1">
        <div className="flex gap-4 items-center mb-6">
          <input value={query} onChange={(e)=>setQuery(e.target.value)} placeholder="Search products..." c
          <div className="bg-white p-3 rounded shadow">
            <span className="text-sm text-gray-600">Showing {filtered.length} items</span>
          </div>
        </div>
        <div className="grid grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-6">
          {filtered.map(p => <ProductCard key={p.id} product={p} />)}
        </div>
      </main>
    </div>
```

db.json:

App.jsx:

App.jsx

grocerEaseFinal > grocerease > src > App.jsx > App

```jsx
import { BrowserRouter, Routes, Route, Navigate } from "react-router-dom";
import { useStore } from "./context/StoreContext";
import Navbar from "./components/Navbar";
import HomePage from "./pages/HomePage";
import AdminPage from "./pages/AdminPage";
import CartPage from "./pages/CartPage";
import CheckoutPage from "./pages/CheckoutPage";
import LoginPage from "./pages/LoginPage";
export default function App() {
  const { currentUser } = useStore();
  return (
    <BrowserRouter>
      {/* show navbar only when logged in */}
      {currentUser && <Navbar />}
      <Routes>
        <Route path="/" element={!currentUser ? <LoginPage /> : <Navigate to="/home" />} />
        <Route path="/home" element={currentUser ? <HomePage /> : <Navigate to="/" />} />
        <Route path="/cart" element={currentUser ? <CartPage /> : <Navigate to="/" />} />
        <Route path="/checkout" element={currentUser ? <CheckoutPage /> : <Navigate to="/" />} />
        {/* admin protected route */}
        <Route
          path="/admin"
          element={
            currentUser && currentUser.role === "admin" ? (
              <AdminPage />
            ) : (
              <Navigate to="/" />
            )
          }
        />
        <Route path="*" element={<Navigate to="/" />} />
      </Routes>
    </BrowserRouter>
  );
}
```

CRUD Components:

Admin Products.jsx

AdminPage.jsx

grocerEaseFinal > grocerease > src > pages > AdminPage.jsx > AdminPage > products.map() callback

```jsx
import { useState } from "react";
import { useStore } from "../context/StoreContext";
import AddProductForm from "../components/AddProductForm";
export default function AdminPage() {
  const { products, deleteProduct } = useStore();
  const [showAdder, setShowAdder] = useState(false);
  return (
    <div className="max-w-6xl mx-auto p-6">
      <div className="flex justify-between items-center mb-6">
        <h1 className="text-3xl font-bold text-green-800">Admin Panel</h1>
        <button className="bg-green-600 text-white px-4 py-2 rounded" onClick={()=>setShowAdder(s=>!s)}>
          {showAdder ? "Close" : "+ Add Product"}
        </button>
      </div>
      {showAdder && <div className="mb-6"><AddProductForm onDone={()=>setShowAdder(false)} /></div>}
      <div className="space-y-4">
        {products.map(p => (
          <div key={p.id} className="bg-white rounded-xl shadow p-4 flex items-center gap-6">
            <img src={p.image || "https://via.placeholder.com/120"} alt="" className="w-28 h-28 object-co
            <div className="flex-1">
              <h3 className="text-xl font-semibold">{p.name}</h3>
              <p className="text-sm text-gray-500">{p.category} • {p.unit}</p>
              <p className="text-gray-600 mt-1">{p.description}</p>
            </div>
            <div className="text-right">
              <div className="text-green-700 font-bold">₹{p.price}</div>
              <div className="text-sm text-gray-600">Stock: {p.stock}</div>
              <div className="mt-3 flex gap-2">
                <button className="px-3 py-2 border rounded" onClick={()=>{/* edit feature optional */}}>E
                <button className="px-3 py-2 border rounded text-red-600" onClick={()=>deleteProduct(p.id
              </div>
            </div>
          </div>
        ))}
      </div>
    </div>
```

CartPage.jsx

CartPage.jsx

grocerEaseFinal > grocerease > src > pages > CartPage.jsx > ...

```jsx
import CartPanel from "../components/CartPanel";
export default function CartPage(){ return <CartPanel />; }

```

Project Execution

Project Execution

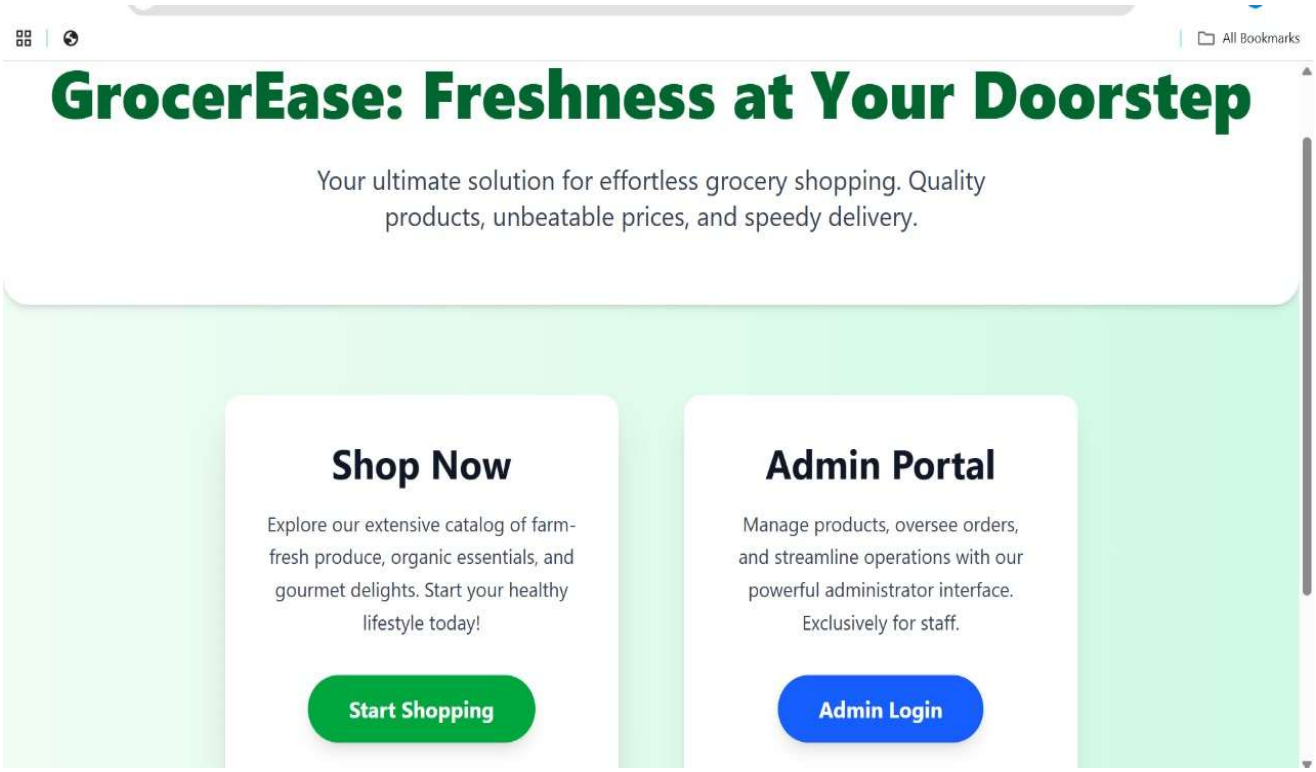
1. Start Backend

npm run server
Runs on:
http://localhost:5001/products

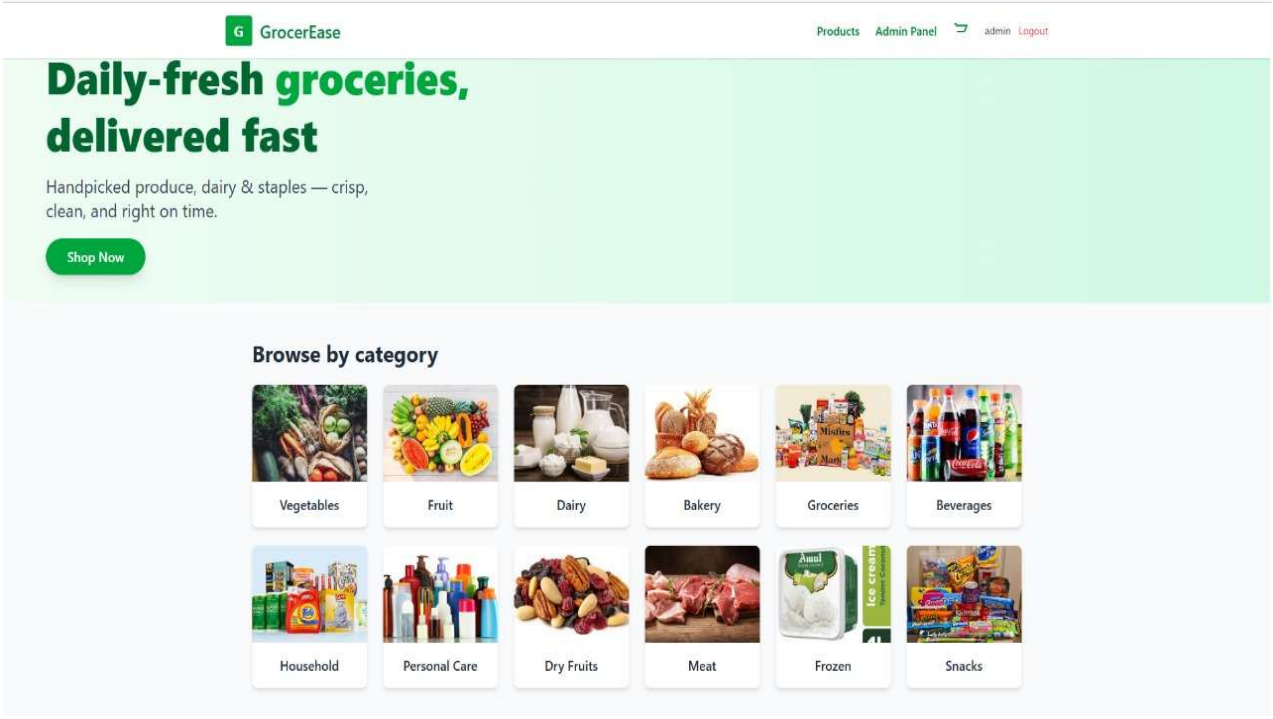
2. Start Frontend

npm run dev
Runs on:
http://localhost:5173

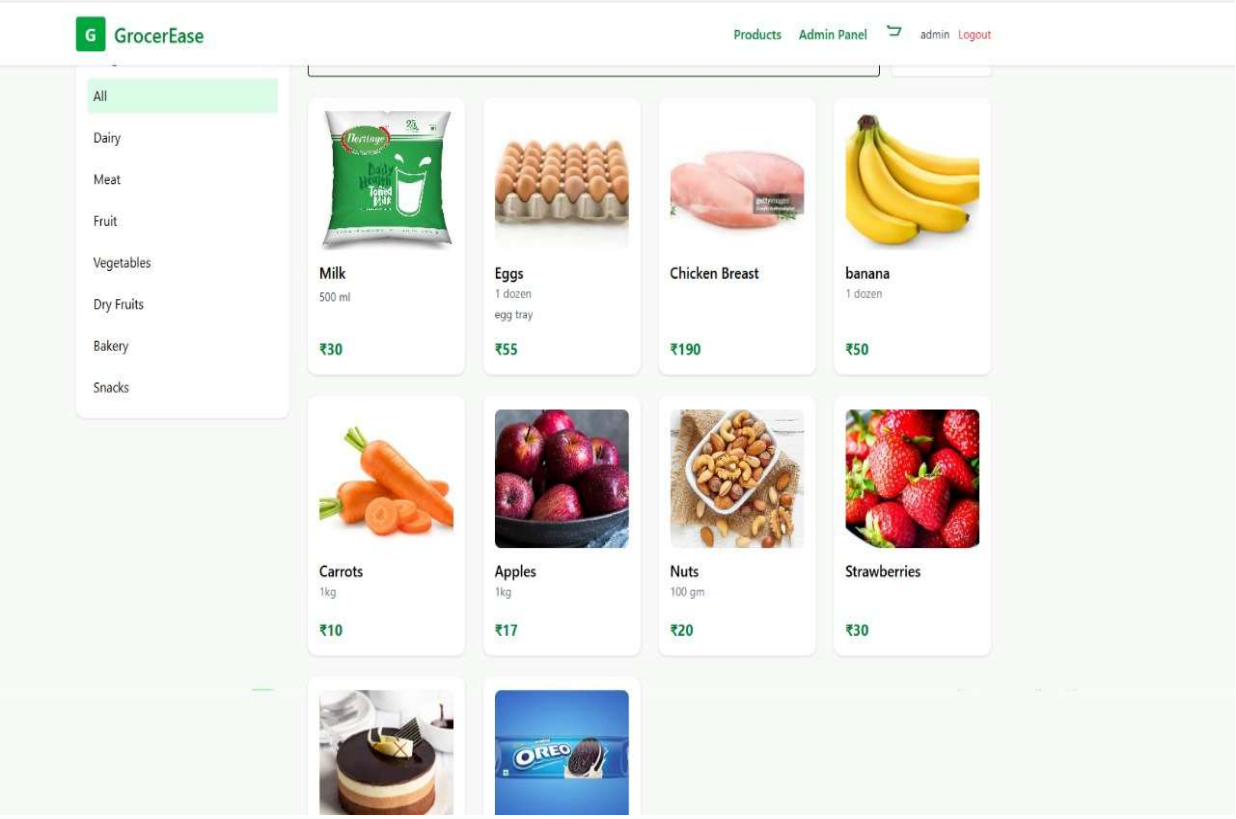
Landing Page:



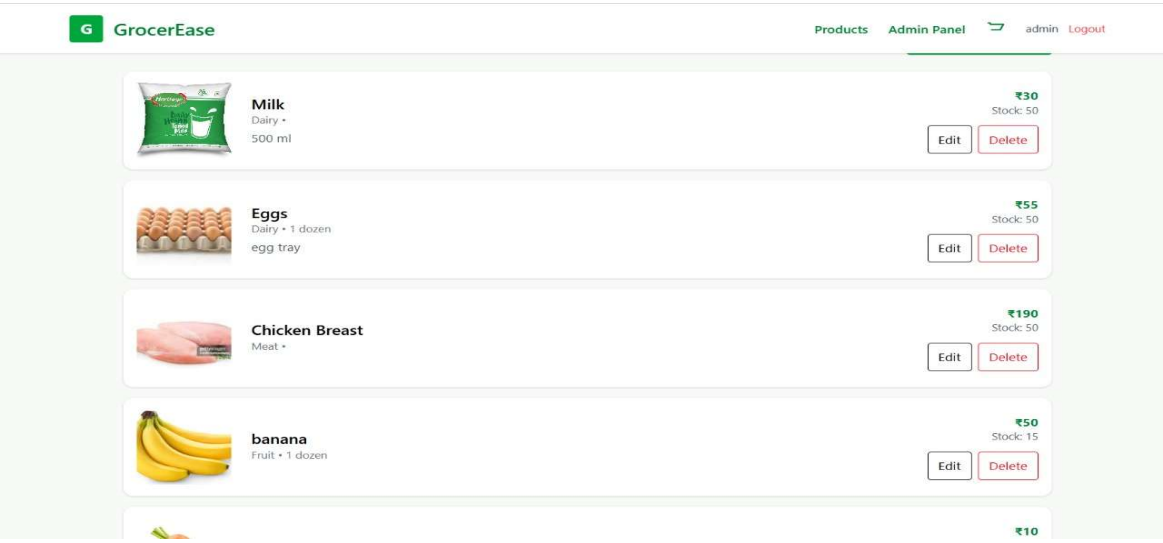
Home page:


GrocerEase
Products
Admin Panel
admin
Logout
Daily-fresh groceries, delivered fast
Handpicked produce, dairy & staples — crisp, clean, and right on time.
Shop Now
Browse by category
Vegetables
Fruit
Dairy
Bakery
Groceries
Beverages
Household
Personal Care
Dry Fruits
Meat
Frozen
Snacks

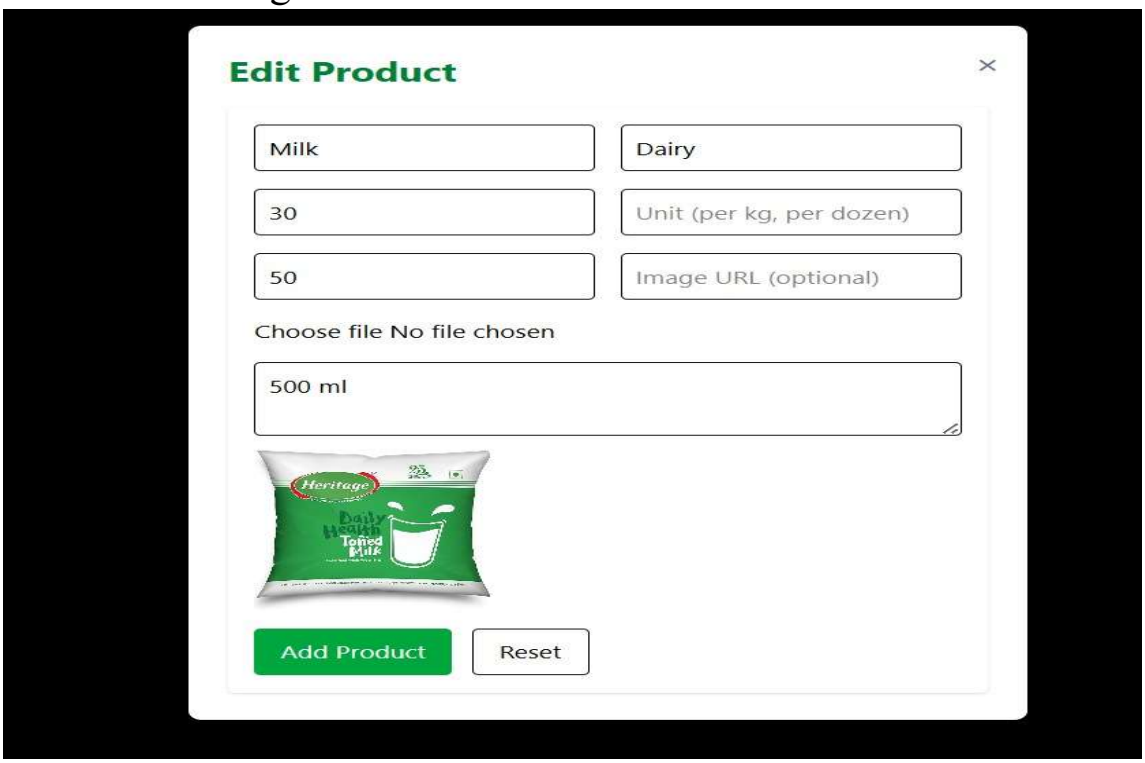
Product page:



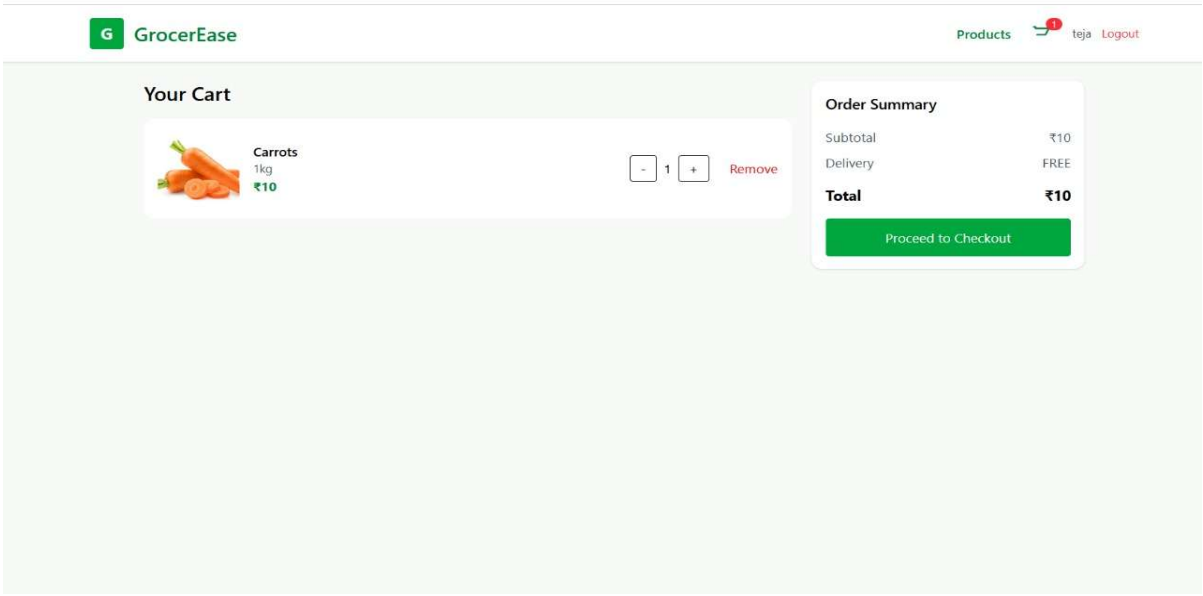
Admin Panel



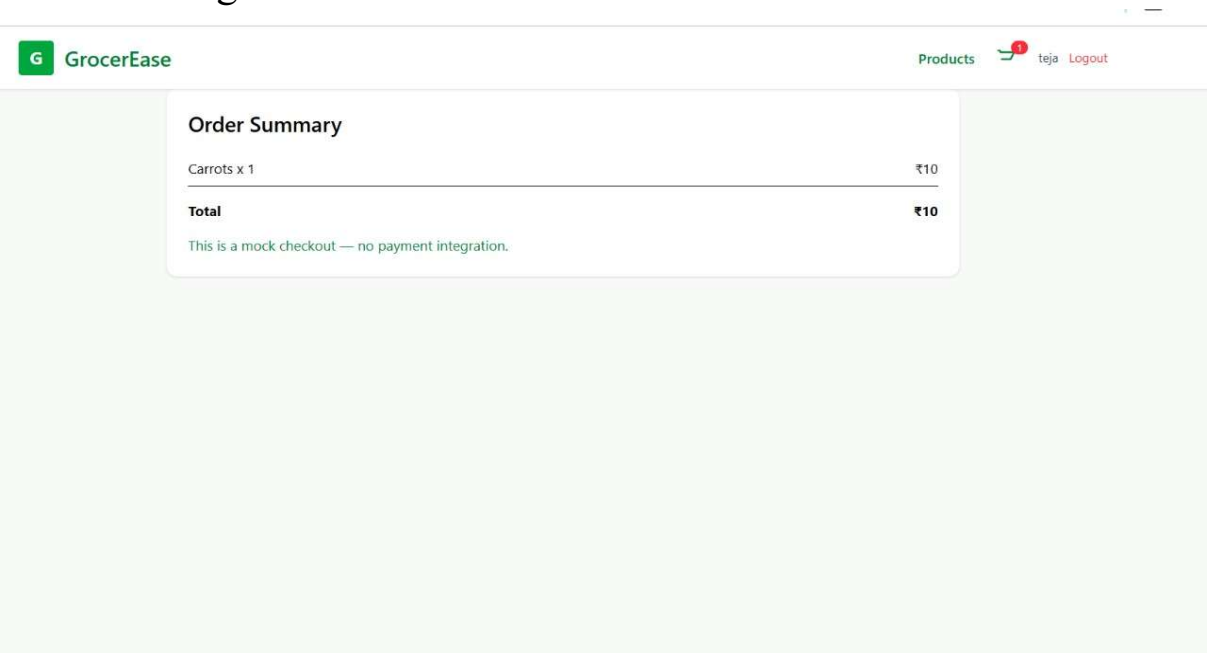
Edit Product Page



Cart Page



Checkout Page



Project Demo Link:
https://github.com/tejaswaroop2005/Grocerease1