

GROCERY APP

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

P.Sai vineela	(AP22110010044)
K.Jayasree lakshmi	(AP22110010043)
N.Nanditha	(AP22110010018)
K.Saranya	(AP22110010015)
R.Lakshmi narasimham	(AP22110010028)



Under the Guidance of

Mr. Yatharth shahrawat

SRM University-AP

Neerukonda, Mangalgiri, Guntur

Andhra Pradesh-522 240

[November, 2024]

Introduction:

Introducing Grocery App – a seamless shopping experience built with React.js. Designed with user-centric functionality, our app combines simplicity with efficiency, offering a complete journey from login to managing your products.

The Grocery App opens with Login and Signup pages. New users can create accounts by entering their name, email, and password, while existing users can log in effortlessly. Upon successful login, users land on the Homepage, featuring eight highlighted products like apples and grapes, alongside a View Products button.

Clicking View Products unveils the Products Page, the heart of the app. This page empowers users with full CRUD operations—Create, Read, Update, and Delete products—seamlessly managed using a JSON server. It's a perfect space to customize your grocery inventory.

Additionally, the app features a Contact Page with a direct email for queries and a Logout button that redirects users to the login page, ensuring a secure and smooth user flow.

With its focus on streamlined CRUD functionality, the Grocery App sets a new standard for online product management, merging React's power with an intuitive interface for all users.

Scenario-based intro:

Imagine a typical day in the life of a busy homemaker or store manager. You've just unlocked your phone, ready to organize your grocery inventory and streamline your day. You open your favorite app, Grocery App.

As you log in, the app welcomes you with a clean, intuitive interface. The simple layout ensures that navigating through your tasks is effortless. On the homepage, you see your curated list of products—apples, grapes, and more—beautifully displayed, with the View Products button inviting you to explore further.

Clicking through, you enter the Products Page, the core of the app. Here, the power of CRUD operations unfolds: adding new items, editing existing ones, or removing outdated products becomes second nature. The integration with a JSON server ensures smooth performance, allowing you to manage your inventory without a hitch.

Later, you need assistance. The Contact Page provides direct support, and with a single tap on Logout, your data remains secure, taking you back to the login page.

With its intuitive design and efficient functionality, the Grocery App transforms the mundane task of grocery management into a seamless and stress-free experience.

Target Audience:

The Grocery App is designed to cater to a diverse audience, including:

1. **Homemakers:** Simplifying daily grocery management with an easy-to-use interface for adding, organizing, and tracking items.
2. **Small Business Owners:** Streamlining product inventory for local grocery shops with efficient CRUD operations.
3. **Students and Beginners:** Offering an intuitive way to practice and learn CRUD functionalities and web app navigation.
4. **Busy Professionals:** Helping them save time by quickly managing grocery lists and inventory on the go.

Accessible and versatile, the app meets the needs of anyone looking for a smooth grocery management solution.

Project goals and Objectives:

The Grocery App aims to provide an efficient and user-friendly platform for grocery management with the following objectives:

1. **Streamlined CRUD Operations:** Enable users to easily Create, Read, Update, and Delete product data with the help of JSON Server, ensuring smooth inventory management.
2. **Intuitive User Interface:** Design a clean, responsive layout for effortless navigation across Login, Signup, Home, Products, and Contact pages.
3. **Secure User Authentication:** Implement robust login and signup features for personalized and secure access to the application.
4. **Enhanced Accessibility:** Ensure compatibility across devices to allow users to manage groceries conveniently anytime, anywhere.
5. **Simplified Query Resolution:** Provide a dedicated Contact page to streamline communication and support for user inquiries.

The app prioritizes usability and innovation, delivering a seamless experience for diverse user needs.

Key features(for grocery app):

- **User Authentication:** Secure Login and Signup functionalities with validation for user credentials.
- **Product Showcase:** A visually appealing homepage displaying 8 featured products.
- **CRUD Operations:** Robust product management system to Create, Read, Update, and Delete items seamlessly.
- **Contact Page:** Provides contact details for user queries.
- **Logout Functionality:** Ensures easy user session termination, redirecting to the login page.
- **JSON Server Integration:** Simulates a real database for data handling.

Prerequisites for Building the Grocery App:

1. Node.js and npm

- **Purpose:** Node.js enables server-side execution of JavaScript, crucial for running the React development server and managing dependencies.
- **Installation:**
 - **Download:** [Node.js Official Site](#)
 - **Instructions:** [Node.js Installation Guide](#)

2. React.js

- **Purpose:** A popular library for building interactive, reusable UI components, making your app dynamic and responsive.
- **Setup:**
 1. Create a new React app:

```
npx create-react-app project-name
```
 2. Navigate to the project folder:

```
cd project-name
```
 3. Run the development server to launch your app:

```
npm start
```

3. JSON Server

- **Purpose:** Used to simulate a backend for storing product data and enabling CRUD operations.
- **Setup:** Install JSON Server using:

```
npm install -g json-server
```

Run the server with:

```
json-server --watch db.json --port 4000
```

4. React Router DOM

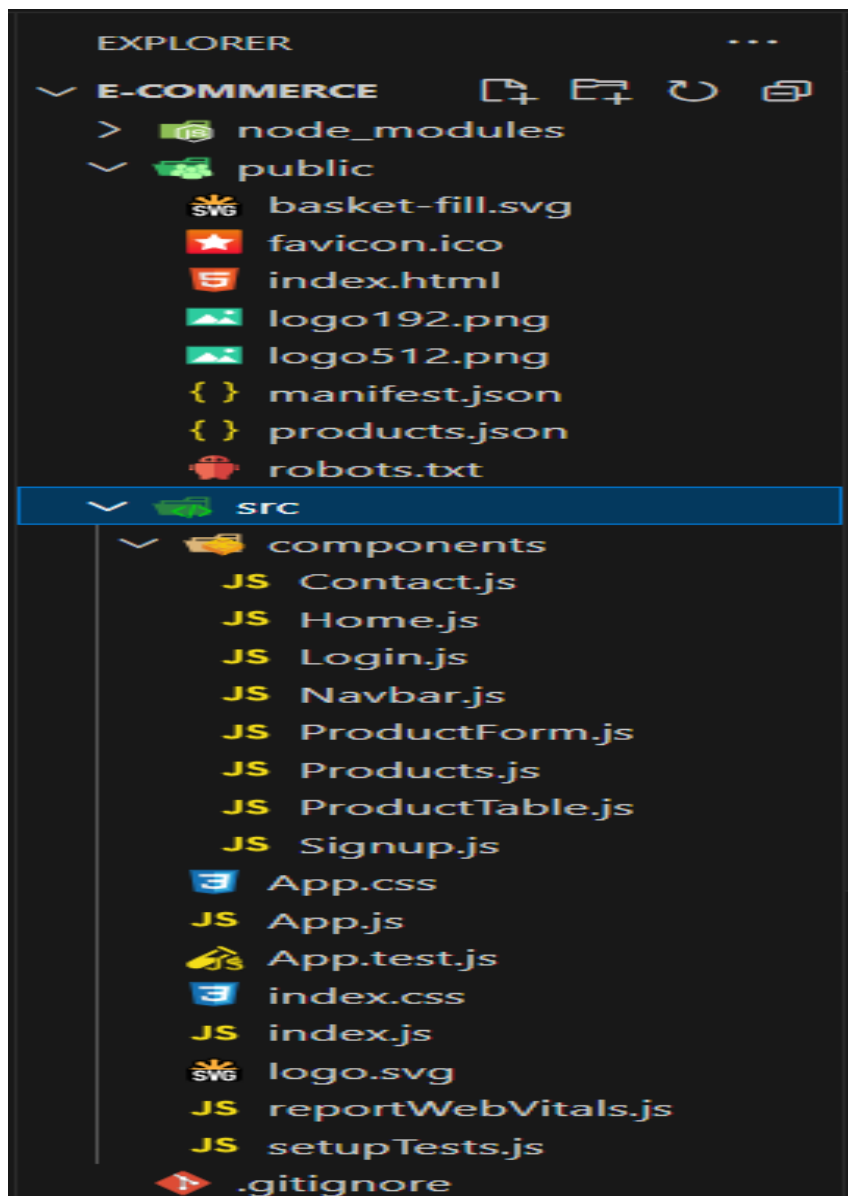
- **Purpose:** Enables seamless navigation across different pages like Login, Home, Products, and Contact.
- **Installation:**

```
npm install react-router-dom
```

5. Development Environment

- **Preferred IDE:** Visual Studio Code (VS Code).
 - **Download:** [VS Code Official Site](#)

Project structure:



The **Grocery App** project is organized into well-defined directories and files to ensure maintainability and scalability. Below is the breakdown of its structure:

Root Directory

- **node_modules/**: Contains all npm dependencies.
- **public/**: Holds static assets like index.html, icons, and JSON files (e.g., products.json).
- **src/**: Core application files, including React components and styles.

Key Directories and Files

1. Public Folder

- index.html: Main HTML template for the React app.
- products.json: Data source for product information (used with JSON Server).
- Static assets (icons like favicon.ico, and logos).

2. Source Folder (src/)

- **components/**: Contains reusable React components:
 - Login.js: Login page with email and password inputs.
 - Signup.js: Signup page with fields for name, email, password, and confirm password.
 - Home.js: Displays featured products and navigation to the Products page.
 - Navbar.js: Handles app navigation across pages.
 - Products.js: Main page for managing products (CRUD operations).
 - ProductForm.js: Form for adding and editing products.
 - ProductTable.js: Displays the list of products.
 - Contact.js: Provides the contact email for queries.
- App.js: Root component managing the layout and routing for the app.
- App.css: Stylesheet for global and component-specific designs.
- index.js: Entry point rendering the App component into the DOM.

3. Other Files

- .gitignore: Specifies files to ignore in version control.
- reportWebVitals.js and setupTests.js: Used for performance monitoring and testing.

Project Flow:

Before going to the project let us see the demo video

Demo video link:

<https://1drv.ms/v/c/ebe33d8d80c7a337/EdHcDuyXk2JPpkwLNH2TKAoBa-VH-jLVOKTYy6IlwZMvDA?e=QwkW25>

Code link:

<https://github.com/vineela1304/React-CRUD>

Milestone1:Project setup and configuration

1. Install Required Tools and Software:

- **React.js:** Install React and set up your project folder.

```
npx create-react-app grocery-app
```

```
cd grocery-app
```

```
npm start
```
- **React Router DOM:** Add for routing between pages.

```
npm install react-router-dom
```
- **JSON Server:** Simulate a backend for CRUD operations.

```
npm install json-server
```
- **React Icons (Optional):** Add icons for styling components.

```
npm install react-icons
```

2. Resources:

- [React Installation](#)
- [JSON Server Setup](#)

Milestone2:Web development

Setup React Application:

- **App.js Component:**
 - Import BrowserRouter for routing.
 - Wrap routes for Login, Signup, Home, Products, and Contact.

```

src > JS Appjs > App
1  import React from "react";
2  import Home from "../components/Home";
3  import Contact from "../components/Contact";
4  import Login from "../components/Login";
5  import Signup from "../components/Signup";
6  import { BrowserRouter as Router, Route, Routes } from "react-router-dom";
7  import Products from "../components/Products";
8
9  const App = () => (
10   <Router>
11
12     <Routes>
13       <Route path="/" element={<Login />} />
14       <Route path="/Home" element={<Home />} />
15       <Route path="/Products" element={<Products />} />
16
17       <Route path="/contact" element={<Contact />} />
18       <Route path="/login" element={<Login />} />
19       <Route path="/contact" element={<Contact />} />
20
21       <Route path="/signup" element={<Signup />} />
22     </Routes>
23   </Router>
24 );
25
26 export default App;
27

```

Description:

1. Purpose:

- App.js serves as the root component of the application, managing routing and navigation.

2. Code Details:

○ React Router Integration:

- Utilizes BrowserRouter, Routes, and Route from react-router-dom to define paths and render components dynamically based on the URL.

○ Defined Routes:

- /: Displays the Login component.
- /Home: Displays the Home component.
- /Products: Displays the Products component.
- /contact: Displays the Contact component.
- /signup: Displays the Signup component.

- Ensures a clean, modular structure by importing and rendering components for different pages.

□ Design UI Components:

- Create individual pages: Login, Signup, Home, Products, Contact.
- Style components using CSS.

□ Implement CRUD Logic in Products.js:

- Use JSON Server for CRUD operations.
- Add axios or fetch for API calls.

Home.js

```
import React from 'react';
import { Button } from 'react-bootstrap'; // To use the Bootstrap Button component
import { useNavigate } from 'react-router-dom'; // To navigate between pages
import NavigationBar from './Navbar';
const latestProducts = [
  {
    id: 1,
    name: 'Lays',
    image: 'https://tse1.mm.bing.net/th?id=OIP_v8e23rhy1ksswk0d3cgw6wqIdg5-qIP-48w-128',
  },
  {
    id: 2,
    name: 'Good day',
    image: 'https://tse1.mm.bing.net/th?id=OIP_89F5vD67p8b3u4qem9qIdg5-qIP-48w-128',
  },
  {
    id: 3,
    name: 'Peach',
    image: 'https://tse1.mm.bing.net/th?id=OIP_27imgGwC7fawse9FVaw6pIdg5-qIP-48w-128',
  },
  {
    id: 4,
    name: 'Orion',
    image: 'https://tse1.mm.bing.net/th?id=OIP_Fyc3k0mXk0f03p8e2Vaw6pIdg5-qIP-48w-128',
  },
  {
    id: 5,
    name: 'Beetroot',
    image: 'https://tse1.mm.bing.net/th?id=OIP_0rmaK9r-F9q9J1P0G0aw6pIdg5-qIP-48w-128',
  },
  {
    id: 6,
    name: 'Greece',
    image: 'https://tse1.mm.bing.net/th?id=OIP_w8pQr25w6U9w82T9w6pIdg5-qIP-48w-128',
  },
  {
    id: 7,
    name: 'Gulab Jamun',
    image: 'https://tse1.mm.bing.net/th?id=OIP_v8e23rhy1ksswk0d3cgw6wqIdg5-qIP-48w-128',
  },
  {
    id: 8,
    name: 'Kurture',
    image: 'https://tse1.mm.bing.net/th?id=OIP_c7r3k4Q4P5s-X7X8v8eaw6pIdg5-qIP-48w-128',
  },
];

export default function Home() {
  const navigate = useNavigate();

  const handleExploreClick = () => {
    navigate('/products');
  };

  return (
    <div>
      <NavigationBar />
      <div style={
        {
          display: 'grid',
          gridTemplateColumns: 'repeat(auto-fit, minmax(300px, 1fr))',
          gap: '20px',
        }
      }
    </div>
  );
}
```

```
gap: '20px',
justifyItems: 'center',
width: '90%',
margin: '20px auto',
));
<latestProducts.map((product) => (
  <div
    key={product.id}
    style={{
      width: '200px',
      background: '#fff',
      padding: '10px',
      borderRadius: '8px',
      boxShadow: '0 4px 8px rgba(0, 0, 0, 0.2)',
      textAlign: 'center',
    }}
  >
    <img
      src={product.image}
      alt={product.name}
      style={{
        width: '100%',
        height: '150px',
        objectFit: 'cover',
        borderRadius: '8px',
      }}
    />
    <p style={{ marginTop: '10px', color: '#333' }}>{product.name}</h4>
  </div>
));
</div>

<Button
  variant="success"
  onClick={handleExploreClick}
  style={{
    padding: '10px 20px',
    fontSize: '1.2rem',
    fontWeight: 'bold',
    borderRadius: '8px',
    marginTop: '20px',
    backgroundColor: '#fdfde0',
    borderColor: '#b16db3',
    boxShadow: '0 4px 8px rgba(0, 0, 0, 0.2)',
    transition: '0.3s ease-in-out',
  }}
  onMouseEnter={(e) => (e.target.style.backgroundColor = '#fdfde0')}
  onMouseLeave={(e) => (e.target.style.backgroundColor = '#b16db3')}
  >
    View All Products
  </Button>
</div>
);
}
```

Description:

Purpose:

- Provides a user interface for new users to register.

Key Features:

1. **Form Inputs:** Collects username, password, and balance using controlled inputs. State is updated via setUsername, setPassword, and setBalance.
2. **Error Message:** Displays error messages if errorMessage is provided.
3. **Form Submission:** Submits data using the handleSignUp function on form submission.
4. **Login Toggle:** Includes a button to switch to the login view via toggleLogin.
5. **Styling:** Uses SignUp.css for a clean and organized layout.

Focus:

- Captures user details for account creation with a seamless experience.

ProductForm.js

```

import React, { useState } from "react";
import { Modal, Button, Form } from "react-bootstrap";

const ProductForm = ({ onSubmit, onClose, product }) => {
  const [formData, setFormData] = useState(
    product || { name: "", brand: "", category: "", price: "", image: "", createdAt: "" }
  );

  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData({ ...formData, [name]: value });
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    onSubmit(formData);
  };

  onClose(); // Close the modal after showing the alert
};

return (
  <Modal show onHide={onClose}>
    <Modal.Header closeButton>
      <Modal.Title>{product ? "Edit Product" : "Create Product"}</Modal.Title>
    </Modal.Header>
    <Modal.Body>
      <Form onSubmit={handleSubmit}>
        <Form.Group className="mb-3">
          <Form.Label>Name / </Form.Label>
          <Form.Control
            name="name"
            value={formData.name}
            onChange={handleChange}
            required
          />
        </Form.Group>
        <Form.Group className="mb-3">
          <Form.Label>Brand / </Form.Label>
          <Form.Control
            name="brand"
            value={formData.brand}
            onChange={handleChange}
            required
          />
        </Form.Group>
        <Form.Group className="mb-3">
          <Form.Label>Category / </Form.Label>
          <Form.Control
            name="category"
            value={formData.category}
            onChange={handleChange}
            required
          />
        </Form.Group>
        <Form.Group className="mb-3">
          <Form.Label>Price / </Form.Label>
          <Form.Control
            name="price"
            type="number"
            value={formData.price}
            onChange={handleChange}
          />
        </Form.Group>
      </Form>
    </Modal.Body>
  </Modal>
);

```

```

    </Form.Group>
    <Form.Group className="mb-3">
      <Form.Label>Image URL</Form.Label>
      <Form.Control
        name="image"
        value={formData.image}
        onChange={handleChange}
        required
      />
    </Form.Group>
    <Form.Group className="mb-3">
      <Form.Label>Creation Date</Form.Label>
      <Form.Control
        name="creationDate"
        type="date"
        value={formData.creationDate}
        onChange={handleChange}
        required
      />
    </Form.Group>
    <Button type="submit" className="w-100">
      Submit
    </Button>
  </Form>
</Modal.Body>
</Modal>
);
};
}

export default ProductForm;

```

Description:

Purpose:

- ProductForm is a reusable modal for creating or editing product details.

Key Features:

1. **Dynamic Form Behavior:**
 - Prefills fields for editing with product data or remains empty for creating a new product.
2. **State Management:**
 - Tracks form input using `useState` and updates `formData` dynamically with `handleChange`.
3. **Form Submission:**
 - Submits data via the `onSubmit` callback and closes the modal using `onClose`.
4. **Modal Design:**
 - Uses react-bootstrap components for a clean, responsive UI.
 - Dynamic title adjusts to "Create Product" or "Edit Product."
5. **Validation and Layout:**
 - Ensures required fields are filled and organizes inputs neatly for user-friendly interaction.

ProductTable.js

```
import React, { memo, useState } from 'react';
import { Table, Button } from 'react-bootstrap';
import { FaSort, FaSortUp, FaSortDown } from 'react-icons/fa'; // Font Awesome Icons

const ProductTable = ({ products, onEdit, onDelete }) => {
  const [searchTerm] = useState("");
  const [sortConfig, setSortConfig] = useState({ key: null, direction: null }); // For sorting

  // Handle sorting
  const handleSort = (key) => {
    const direction =
      sortConfig.key === key && sortConfig.direction === "asc" ? "desc" : "asc";
    setSortConfig({ key, direction });
  };

  // Get filtered and sorted products
  const filteredAndSortedProducts = () => {
    let filteredProducts = products.filter((product) =>
      product.name.toLowerCase().includes(searchTerm.toLowerCase())
    );

    if (sortConfig.key) {
      filteredProducts.sort((a, b) => {
        if (a[sortConfig.key] < b[sortConfig.key]) return sortConfig.direction === "asc" ? -1 : 1;
        if (a[sortConfig.key] > b[sortConfig.key]) return sortConfig.direction === "asc" ? 1 : -1;
        return 0;
      });
    }

    return filteredProducts;
  };

  // Render sorting icons
  const renderSortIcon = (key) => {
    if (sortConfig.key === key) {
      return sortConfig.direction === "asc" ? <FaSortUp /> : <FaSortDown />;
    }
    return <FaSort />;
  };

  return (
    <div>
      {/* Product Table */}
      <table striped bordered hover className="text-center align-middle">
        <thead>
          <tr>
            <th onClick={() => handleSort("row")}>
              ID {renderSortIcon("row")}
            </th>
            <th onClick={() => handleSort("name")}>
              Name {renderSortIcon("name")}
            </th>
            <th onClick={() => handleSort("brand")}>
              Brand {renderSortIcon("brand")}
            </th>
            <th onClick={() => handleSort("category")}>
              Category {renderSortIcon("category")}
            </th>
            <th onClick={() => handleSort("price")}>
              Price {renderSortIcon("price")}
            </th>
            <th>Image</th>
            <th>Created At {renderSortIcon("createdAt")}>
              Created At {renderSortIcon("createdAt")}
            </th>
            <th>Action</th>
          </tr>
        </thead>
        <tbody>
          {filteredAndSortedProducts().map((product) => (
            <tr>
              <td>{product.id}</td>
              <td>{product.name}</td>
              <td>{product.brand}</td>
              <td>{product.category}</td>
              <td>{product.price}</td>
              <td>
                <img
                  src={product.image}
                  alt={product.name}
                  width="100"
                  height="100"
                  style={{ objectFit: "contain" }}
                />
              </td>
              <td>{product.createdAt}</td>
              <td>
                <Button variant="success" onClick={() => onEdit(product)}>
                  Edit
                </Button>
                <Button
                  variant="danger"
                  className="ms-2"
                  onClick={() => onDelete(product.id)}>
                  Delete
                </Button>
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  );
};

export default memo(ProductTable);
```

```

    <tr>
      <td>{product.id}</td>
      <td>{product.name}</td>
      <td>{product.brand}</td>
      <td>{product.category}</td>
      <td>{product.price}</td>
      <td>
        <img
          src={product.image}
          alt={product.name}
          width="100"
          height="100"
          style={{ objectFit: "contain" }}
        />
      </td>
      <td>{product.createdAt}</td>
      <td>
        <Button variant="success" onClick={() => onEdit(product)}>
          Edit
        </Button>
        <Button
          variant="danger"
          className="ms-2"
          onClick={() => onDelete(product.id)}>
          Delete
        </Button>
      </td>
    </tr>
  )}
</tbody>
</table>
</div>
);
};

export default memo(ProductTable);
```

Description:

Purpose:

- ProductTable displays a dynamic, interactive table for managing product data with features like sorting, searching, and editing.

Key Features:

1. Sorting:

- Allows sorting columns (ID, Name, Brand, etc.) by clicking the header.
- Uses sortConfig state to track sorting key and direction.

2. Filtering:

- Filters products based on searchTerm (name-based search).

3. Dynamic Table:

- Populates rows from products array with details like ID, name, brand, category, price, and image.

4. Actions:

- Includes **Edit** and **Delete** buttons for each product.
- Triggers onEdit and onDelete callbacks for respective actions.

5. Styling:

- Utilizes react-bootstrap for a clean, responsive table design.

Products.js

```

import React, { useEffect, useState } from "react";
import NavigationBar from "../Navbar";
import ProductTable from "../ProductTable";
import ProductForm from "../ProductForm";
import { Button, Row, Col, Form } from "react-bootstrap";
import "../App.css";

const Products = () => {
  const [products, setProducts] = useState([]);
  const [showForm, setShowForm] = useState(false);
  const [currentProduct, setCurrentProduct] = useState(null);
  const [searchTerm, setSearchTerm] = useState("");

  // Fetch products from json-server (only once on component mount)
  useEffect(() => {
    fetch("http://localhost:4000/products")
      .then((response) => {
        if (!response.ok) {
          throw new Error("Failed to fetch products");
        }
        return response.json();
      })
      .then((data) => setProducts(data))
      .catch((error) => console.error("Error fetching products:", error));
  }, []); // Empty dependency array ensures this runs only once

  // Handle create product
  const handleCreate = () => {
    setCurrentProduct(null);
    setShowForm(true);
  };

  // Handle edit product
  const handleEdit = (product) => {
    setCurrentProduct(product);
    setShowForm(true);
  };

  // Handle delete product
  const handleDelete = (id) => {
    fetch(`http://localhost:4000/products/${id}`, {
      method: "DELETE",
    })
      .then((response) => {
        if (!response.ok) {
          throw new Error("Failed to delete product");
        }
        // Update the products state after deletion
        setProducts(products.filter((product) => product.id !== id));
        window.alert("Product deleted successfully!"); // Success alert
      })
      .catch((error) => console.error("Error deleting product:", error));
  };

  // Handle form submit (Create or Update)
  const handleFormSubmit = (product) => {
    if (currentProduct) {
      // Update product
      fetch(`http://localhost:4000/products/${currentProduct.id}`, {
        method: "PUT",
        headers: {
          "Content-Type": "application/json",
        },
        body: JSON.stringify(product),
      })
        .then((response) => {
          if (!response.ok) {
            throw new Error("Failed to update product");
          }
          return response.json();
        })
        .then((updatedProduct) => {
          // Update the state with the updated product
          setProducts(
            products.map((p) => (p.id === updatedProduct.id ? updatedProduct : p))
          );
        });
    }
  };
};

```

```
const handleCreateProduct = (product) => {
  .then((response) => {
    if (!response.ok) {
      throw new Error("Failed to create product");
    }
    return response.json();
  })
  .then((createdProduct) => {
    // Add the new product directly to the state
    setProducts([...products, createdProduct]);
    setShowForm(false);
    alert("Product added successfully!");
  })
  .catch((error) => console.error("Error creating product:", error));
}

// }

return (
  <div>
    <NavigationBar />
    <div className="d-flex justify-content-center align-items-center mb-4">
      <h1>Products </h1>
    </div>

    { /* Row to contain search and create product button */ }
    <Row className="mb-4">
      { /* Search input aligned to the left */ }
      <Col xs={12} md={6} className="d-flex justify-content-start">
        <Form.Control
          type="text"
          placeholder="Search by product name..."
          value={searchTerm}
          onChange={(e) => setSearchTerm(e.target.value)}
          className="w-75 search-input"
          style={{
            border: '2px solid green', // Add a green border
            borderRadius: '4px', // Optional: rounded corners
            padding: '8px', // Optional: adjust padding
          }}
        />
      </Col>

      { /* Create product button aligned to the right */ }
      <Col xs={12} md={5} className="d-flex justify-content-end">
        <Button variant="success" onClick={handleCreate} className="create-btn">
          <div>Create Product</div>
        </Button>
      </Col>
    </Row>

    { /* Product Table */ }
    <ProductTable
      products={products.filter((product) =>
        product.name.toLowerCase().includes(searchTerm.toLowerCase())
      )}
      onEdit={handleEdit}
      onDelete={handleDelete}
    />

    <showForm && {
      <ProductForm
        onSubmit={handleFormSubmit}
        onClose={() => setShowForm(false)}
        product={currentProduct}
      />
    }
  </div>
);
}

export default Products;
```

Description:

Purpose:

The Products component is a complete CRUD interface for managing a product catalog. It integrates search, create, update, and delete functionalities with a dynamic UI and backend API.

Key Features:

1. **State Management:**
 - Manages product list (products), modal visibility (showForm), search term (searchTerm), and the current product for editing (currentProduct).
2. **Data Fetching:**
 - Fetches products from json-server (<http://localhost:4000/products>) on initial render using `useEffect`.
3. **Create Product:**
 - Opens a form modal to collect product details.
 - Sends a POST request to the API to add new products.
 - Dynamically assigns a unique ID (row) for new entries.

4. **Edit Product:**

- Opens the form pre-filled with product details for editing.
- Updates the product via a PUT request to the API.

5. **Delete Product:**

- Deletes a product with a DELETE request.
- Updates the product list state without refreshing the page.

6. **Search Functionality:**

- Filters products by name in real-time using the searchTerm.

7. **Product Table Integration:**

- Uses ProductTable to display products dynamically.
- Includes "Edit" and "Delete" buttons to trigger respective actions.

8. **Reusable Form:**

- Uses ProductForm for both creating and editing products.
- Handles form submission for both actions.

9. **Styling:**

- Responsive layout with Bootstrap for tables, forms, and buttons.
- Custom styles added for search input (border, padding) and "Create Product" button.

Flow:

1. On page load, fetches and displays products.
2. Users can:
 - **Search:** Filter products by name.
 - **Create:** Add new products via a modal form.
 - **Edit:** Modify existing products in a modal form.
 - **Delete:** Remove a product with confirmation.
3. Updates the state dynamically to reflect changes.

This component ensures seamless user interaction and clean backend integration for managing products.

Milestone 3: Testing and Refinement

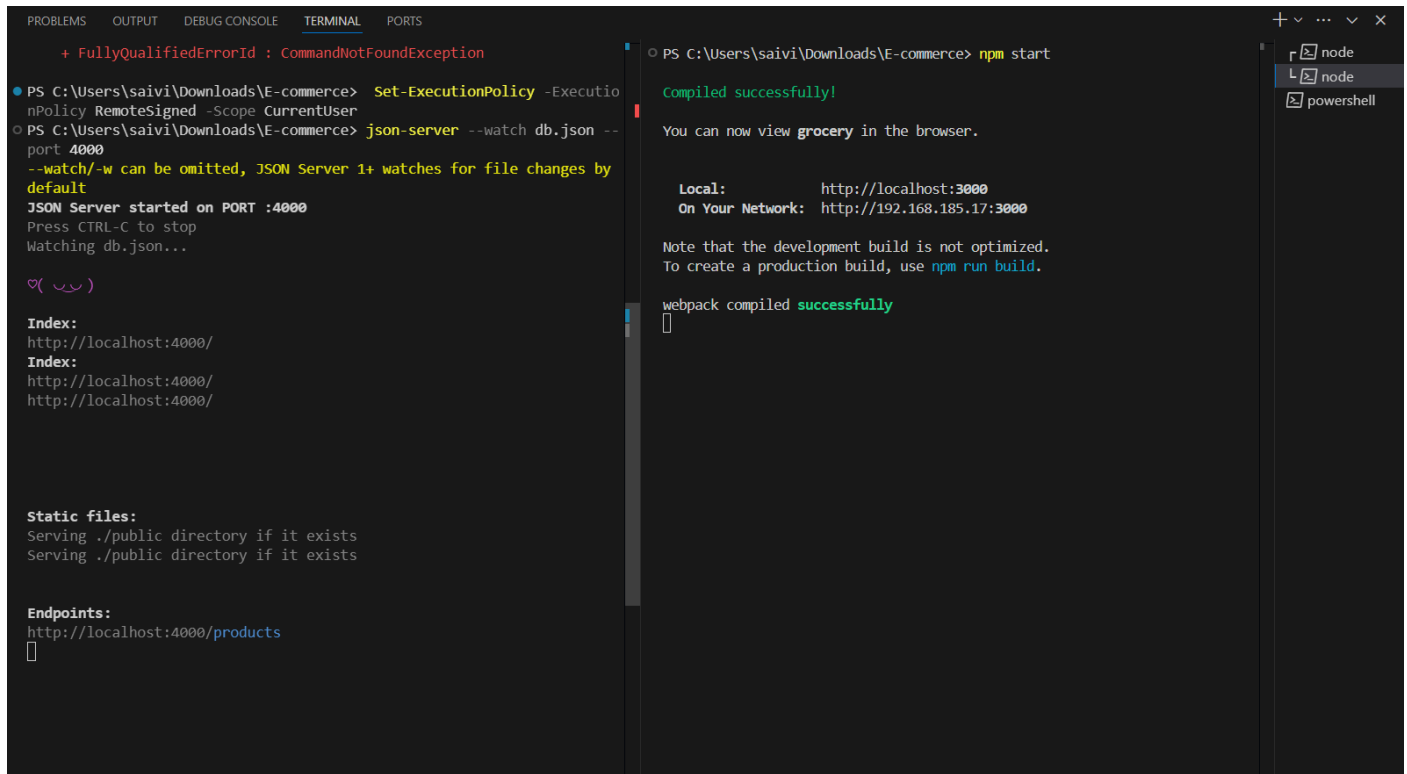
• **Test Features:**

- Ensure all CRUD operations work with JSON Server.
- Check page navigation and functionality of Login and Signup.

• **Deploy (Optional):**

- Use **GitHub Page** to deploy the project.

Project execution terminal:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
+ FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\saivi\Downloads\E-commerce> Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
PS C:\Users\saivi\Downloads\E-commerce> json-server --watch db.json --port 4000
--watch/-w can be omitted, JSON Server 1+ watches for file changes by default
JSON Server started on PORT :4000
Press CTRL-C to stop
Watching db.json...

Index:
http://localhost:4000/
Index:
http://localhost:4000/
http://localhost:4000/

Static files:
Serving ./public directory if it exists
Serving ./public directory if it exists

Endpoints:
http://localhost:4000/products

PS C:\Users\saivi\Downloads\E-commerce> npm start

Compiled successfully!

You can now view grocery in the browser.

Local:      http://localhost:3000
On Your Network:  http://192.168.185.17:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Outputs of the project:

Login page:



Login


Email address

Password


[Login](#) [Signup](#)

Home page:


 [Best Store](#) [Home](#) [Products](#) [Contact](#) [User](#) 




lays




Good day




Peach




Onion




Beetroot



Grapes








Gulab jamun



Kurkure

[View All Products](#)

Products page:

5	Gulab Jamun	Haldiram's	Sweets	150		2024-11-16	<div>EditDelete</div>
6	Kurkure	PepsiCo	Snacks	40		2024-11-16	<div>EditDelete</div>
7	Lays Potato Chips	Lay's	Snacks	30		2024-11-16	<div>EditDelete</div>
8	Good day	Britannia	Biscuits	25		2024-11-16	<div>EditDelete</div>
10	Pineapple	Dole	Fruits	60		2024-11-16	<div>EditDelete</div>

Create product option:

Name

Chocobar

Brand

arun

Category

ice-cream

Price

30


Image URL

=OIP.SprWx8hl64eh2mLAXTKhxAHaGq&pid=Api&P=0&h=180

Creation Date

24/11/2024

Submit

18	Chocobar	arun	ice-cream	30		2024-11-24	<div>EditDelete</div>
----	----------	------	-----------	----	--	------------	-----------------------

Edit product:

Name

Kurkure

Brand

PepsiCo

Category

Snacks

Price

40

Image URL

https://tse4.mm.bing.net/th?id=OIP.eTT3w24Pb9S-JKIlw85Aew






Creation Date

16 / 11 / 2024

Upadted 40rs to 30rs

6	Kurkure	PepsiCo	Snacks	30		2024-11-16	<div>EditDelete</div>
---	---------	---------	--------	----	--	------------	-----------------------

Delete product:

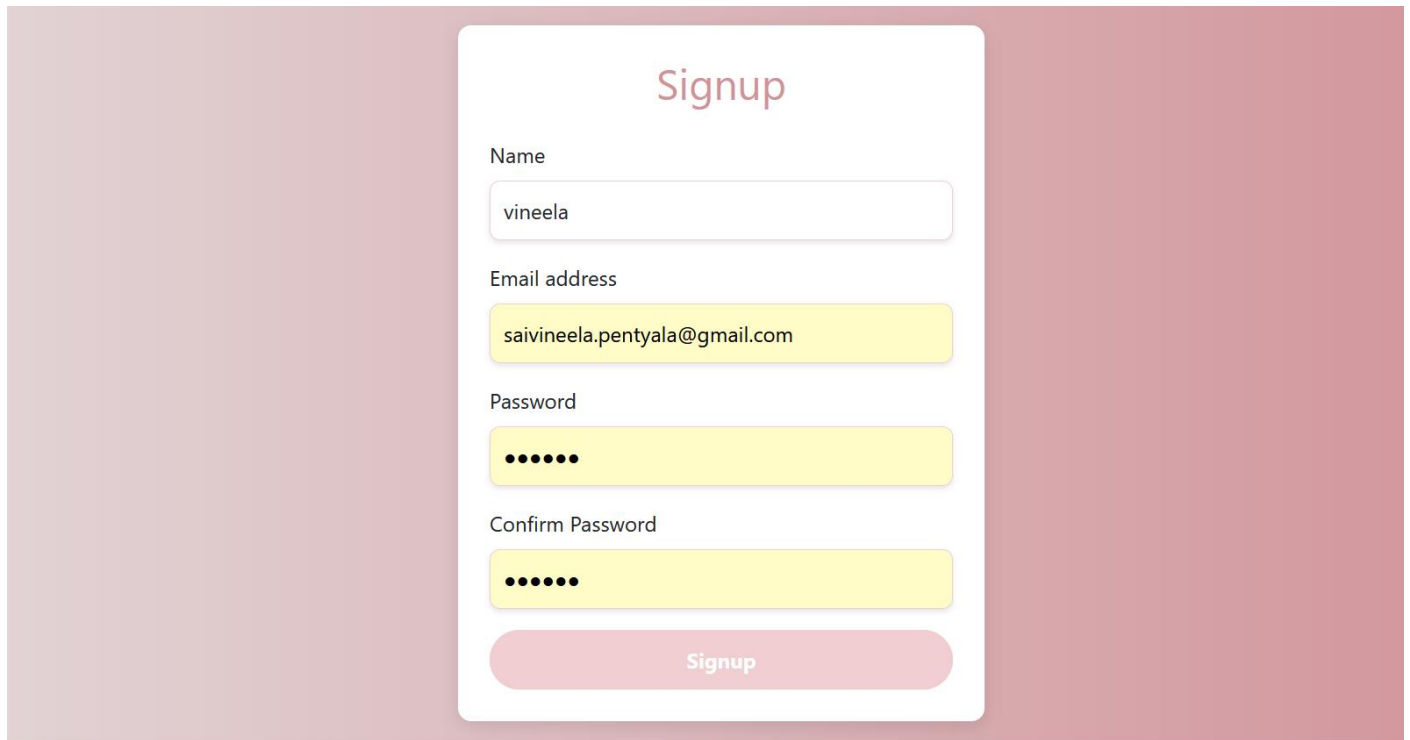
14	Beetroot	Pure Veggies	Vegetables	35		2024-11-16	<div>EditDelete</div>
15	Tomato	Farm Fresh	Vegetables	25		2024-11-16	<div>EditDelete</div>
16	Cucumber	Green Garden				2024-11-16	<div>EditDelete</div>
17	Potato	Golden Harvest	Vegetables	25		2024-11-16	<div>EditDelete</div>
18	Chocobar	arun	ice-cream	30		2024-11-24	<div>EditDelete</div>

localhost:3000

Product deleted successfully!

OK

Singup page:



Signup

Name

vineela

Email address

saivineela.pentyala@gmail.com

Password

•••••

Confirm Password

•••••

Signup

And the logout page will redirect to login page.

Demo video link:

<https://1drv.ms/v/c/ebe33d8d80c7a337/EdHcDuyXk2JPpkwLNH2TKAoBa-VH-jLVOKTYy6llwZMvDA?e=QwkW25>

Team members:

- 1.Sai Vineela(AP22110010044)
- 2.Jayasree Lakshmi(AP22110010043)
- 3.Nanditha(AP22110010015)
- 4.Saranya(AP22110010018)
- 5.Lakshmi Narasimham(AP22110010028)