Inventory Management Preparation for Stock Manager with [React.js](http://react.js)

**TOPIC:STORE MANAGER KEEP TRACK OF INVENTORY**

# 1. Introduction

This document provides a step-by-step guide to preparing an inventory management system for a stock manager using React.js. The goal is to create an efficient interface for managing products, tracking stock levels, adding or removing items, and providing reports.

Team Members and their role:

Merlin J:Documentaion

Yogeshwari S: Coding

Devadharshini B: Coding

Roslin Mary J : Demo video

**2. Project Setup**

2.1 Purpose and Features

* Used to check the product stock
* This application will use in online shopping
* Keep real time records of products in stock,including quantities availabl,sold or returned
* To store the product information
* To manage the products stock digitally

2.2 Prerequisites  
Before starting, ensure you have the following installed:  
- Node.js (v14 or above)  
- npm or yarn  
- Code editor (VS Code recommended)  
- Git (optional for version control)  
  
2.2 Create React App  
```  
npx create-react-app stock-inventory-manager  
cd stock-inventory-manager  
npm start  
```  
This will initialize your project and launch the development server at http://localhost:3000.

# 3. Project Structure

```  
stock-inventory-manager/  
├── public/  
├── src/  
│ ├── components/  
│ │ ├── InventoryList.js  
│ │ ├── InventoryForm.js  
│ │ └── InventoryItem.js  
│ ├── context/  
│ │ └── InventoryContext.js  
│ ├── App.js  
│ └── index.js  
├── package.json  
└── README.md  
```

# 4. Key Features

1. Display Inventory List  
2. Add New Items  
3. Edit Existing Items  
4. Delete Items  
5. Track Stock Levels  
6. Search and Filter Items

# 5. Inventory Context Setup

We will use React Context API to manage state globally.  
  
\*\*src/context/InventoryContext.js\*\*  
```javascript  
import React, { createContext, useState } from 'react';  
  
export const InventoryContext = createContext();  
  
export const InventoryProvider = ({ children }) => {  
 const [items, setItems] = useState([]);  
  
 const addItem = (item) => {  
 setItems([...items, item]);  
 };  
  
 const removeItem = (id) => {  
 setItems(items.filter(item => item.id !== id));  
 };  
  
 const updateItem = (updatedItem) => {  
 setItems(items.map(item => item.id === updatedItem.id ? updatedItem : item));  
 };  
  
 return (  
 <InventoryContext.Provider value={{ items, addItem, removeItem, updateItem }}>  
 {children}  
 </InventoryContext.Provider>  
 );  
};  
```

# 6. Inventory List Component

\*\*src/components/InventoryList.js\*\*  
```javascript  
import React, { useContext } from 'react';  
import { InventoryContext } from '../context/InventoryContext';  
import InventoryItem from './InventoryItem';  
  
const InventoryList = () => {  
 const { items } = useContext(InventoryContext);  
  
 return (  
 <div>  
 <h2>Inventory</h2>  
 {items.length === 0 ? (  
 <p>No items in stock.</p>  
 ) : (  
 items.map(item => <InventoryItem key={item.id} item={item} />)  
 )}  
 </div>  
 );  
};  
  
export default InventoryList;  
```

# 7. Inventory Form Component

\*\*src/components/InventoryForm.js\*\*  
```javascript  
import React, { useState, useContext } from 'react';  
import { InventoryContext } from '../context/InventoryContext';  
  
const InventoryForm = () => {  
 const { addItem } = useContext(InventoryContext);  
 const [name, setName] = useState('');  
 const [quantity, setQuantity] = useState('');  
  
 const handleSubmit = (e) => {  
 e.preventDefault();  
 addItem({ id: Date.now(), name, quantity: parseInt(quantity) });  
 setName('');  
 setQuantity('');  
 };  
  
 return (  
 <form onSubmit={handleSubmit}>  
 <input  
 type="text"  
 placeholder="Item Name"  
 value={name}  
 onChange={(e) => setName(e.target.value)}  
 required  
 />  
 <input  
 type="number"  
 placeholder="Quantity"  
 value={quantity}  
 onChange={(e) => setQuantity(e.target.value)}  
 required  
 />  
 <button type="submit">Add Item</button>  
 </form>  
 );  
};  
  
export default InventoryForm;  
```

# 8. Inventory Item Component

\*\*src/components/InventoryItem.js\*\*  
```javascript  
import React, { useContext } from 'react';  
import { InventoryContext } from '../context/InventoryContext';  
  
const InventoryItem = ({ item }) => {  
 const { removeItem } = useContext(InventoryContext);  
  
 return (  
 <div>  
 <span>{item.name} - {item.quantity}</span>  
 <button onClick={() => removeItem(item.id)}>Delete</button>  
 </div>  
 );  
};  
  
export default InventoryItem;  
```

# 9. Integrating Context in App.js

\*\*src/App.js\*\*  
```javascript  
import React from 'react';  
import { InventoryProvider } from './context/InventoryContext';  
import InventoryList from './components/InventoryList';  
import InventoryForm from './components/InventoryForm';  
  
const App = () => {  
 return (  
 <InventoryProvider>  
 <div style={{ padding: '20px' }}>  
 <h1>Stock Manager Inventory</h1>  
 <InventoryForm />  
 <InventoryList />  
 </div>  
 </InventoryProvider>  
 );  
};  
  
export default App;  
```

**10.Screenshot or Demo**

**https://drive.google.com/file/d/1XUZv2h4D3dZzpcv6GKGFLZEnfNgNDhWi/view?usp=drivesdk**

# 11. Testing & Deployment

1. Test functionalities by adding, editing, and deleting items.  
2. Validate that stock updates correctly.  
3. Use tools like Netlify or Vercel for deployment.

# 12. Future Improvements

1. Integrate with backend using REST API or GraphQL.  
2. Add user authentication.  
3. Include charts and analytics for better stock management.  
4. Improve styling with Material-UI or Tailwind CSS.