Guide to Running the Inventory Management App on Your Computer

This guide provides all the necessary steps and code to set up and run the "FactoryFlow" inventory management application on your local machine. This setup uses Vite, a modern and fast build tool for web projects.

Part 1: Prerequisites

Before you begin, you need to have a few essential tools installed on your computer.

1. **Node.js and npm:** This is the runtime environment for JavaScript projects. Installing Node.js will also automatically install npm (Node Package Manager).
    - **Download and Install:** Go to the official Node.js website and download the "LTS" (Long Term Support) version for your operating system (Windows, macOS, or Linux).
2. **Code Editor:** You need a place to edit the code. Visual Studio Code is a popular, free choice that works great for this type of project.
3. **Firebase Project:** The application uses Google Firebase for its database and authentication. You need to create a free Firebase project.
    - Go to the Firebase Console.
    - Click "Add project" and follow the on-screen instructions to create a new project.
    - Once your project is created, you will need to get your project's configuration details. We'll cover this in Part 2.

Part 2: Step-by-Step Setup Instructions

Follow these steps carefully in your computer's terminal (like Command Prompt, PowerShell, or the integrated terminal in VS Code).

Step 1: Create Your Project Folder

First, create a new folder for your project and navigate into it.

```
mkdir factoryflow-inventory-app
cd factoryflow-inventory-app
```

Step 2: Initialize a Node.js Project

This command creates a package.json file, which will manage your project's details and dependencies.

```
npm init -y
```

Step 3: Install Required Packages

Now, install all the necessary libraries. This command installs a specific, stable version of Tailwind CSS (3.4.4) to prevent compatibility issues.

```
# Install React, Firebase, and icons
npm install react react-dom firebase lucide-react

# Install development tools (Vite, Tailwind CSS)
npm install --save-dev vite @vitejs/plugin-react tailwindcss@3.4.4 postcss autoprefixer
```

Step 4: Configure Tailwind CSS

This step creates the necessary configuration files for styling.

1. **Manually Create tailwind.config.js:**
   - In the root of your project folder (factoryflow-inventory-app), create a new file named tailwind.config.js.
   - Copy and paste the following code into it:

```
/** @type {import('tailwindcss').Config} */
export default {
  content: [
    "./index.html",
    "./src/**/*.{js,ts,jsx,tsx}",
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```

2. **Manually Create postcss.config.js:**
   - In the same root folder, create another new file named postcss.config.js.
   - Copy and paste the following code into it. This is the standard and correct configuration for this version.

```
export default {
  plugins: {
    tailwindcss: {},
```

```
      autoprefixer: {},
    },
  }
```

Step 5: Create the Main index.html File

In your project's root folder (factoryflow-inventory-app), create a file named index.html and add the following code. This is the main entry point for your application.

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
    <title>FactoryFlow - Inventory Management</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

Step 6: Create the CSS File

1. Create a new folder named src.

2. Inside the src folder, create a file named index.css and add the following lines. These are required for Tailwind CSS to work.

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Step 7: Create the React Entry File

Inside the src folder, create a file named main.jsx. This file tells React where to render the main App component in the HTML.

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.jsx'
import './index.css'

ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
)
```

Step 8: Add the Application Code

This is the main application code.

1. Inside the src folder, create a file named App.jsx.
2. Copy and paste the entire code block below into this file.

3. **Crucially, you will need to replace the placeholder Firebase configuration with your own.**

```
import React, { useState, useEffect, useRef } from 'react';
import { initializeApp } from 'firebase/app';
import { getAuth, signInAnonymously, onAuthStateChanged } from
'firebase/auth';
import { getFirestore, collection, doc, addDoc, onSnapshot, writeBatch,
query, orderBy, serverTimestamp, setDoc } from 'firebase/firestore';
import { ArrowDown, ArrowUp, BarChart, Box, DollarSign, Factory,
Package, PlusCircle, ShoppingCart, Trash2, Truck, Warehouse, X } from
'lucide-react';

// --- Firebase Configuration ---
// IMPORTANT: Replace this with your actual Firebase project configuration!
// You can find this in your Firebase project settings.
const firebaseConfig = {
    apiKey: "PASTE_YOUR_API_KEY_HERE",
    authDomain: "PASTE_YOUR_AUTH_DOMAIN_HERE",
    projectId: "PASTE_YOUR_PROJECT_ID_HERE",
    storageBucket: "PASTE_YOUR_STORAGE_BUCKET_HERE",
    messagingSenderId: "PASTE_YOUR_MESSAGING_SENDER_ID_HERE",
    appId: "PASTE_YOUR_APP_ID_HERE"
};

// --- Initialize Firebase ---
const app = initializeApp(firebaseConfig);
```

```
const auth = getAuth(app);
const db = getFirestore(app);

// --- Main App Component ---
export default function App() {
    const [activeTab, setActiveTab] = useState('dashboard');
    const [userId, setUserId] = useState(null);

    useEffect(() => {
        const unsubscribe = onAuthStateChanged(auth, async (user) => {
            if (user) {
                setUserId(user.uid);
            } else {
                try {
                    await signInAnonymously(auth);
                } catch (error) {
                    console.error("Anonymous sign-in failed:", error);
                }
            }
        });
        return () => unsubscribe();
    }, []);

    const renderContent = () => {
        if (!userId) return null;
        const collectionPrefix = `users/${userId}`; // Path for user-specific
```

```
data
        switch (activeTab) {
            case 'dashboard': return <Dashboard setActiveTab={setActiveTab}
collectionPrefix={collectionPrefix} />;
            case 'products': return <Products collectionPrefix={collectionPrefix}
/>;
            case 'materials': return <RawMaterials
collectionPrefix={collectionPrefix} />;
            case 'inventory': return <Inventory collectionPrefix={collectionPrefix}
/>;
            case 'sales': return <SalesOrders collectionPrefix={collectionPrefix}
/>;
            case 'manufacturing': return <ManufacturingOrders
collectionPrefix={collectionPrefix} />;
            default: return <Dashboard setActiveTab={setActiveTab}
collectionPrefix={collectionPrefix} />;
        }
    };

    if (!userId) {
        return (
            <div className="flex items-center justify-center h-screen bg-
gray-900 text-white">
                <div className="text-center">
                    <Package className="w-16 h-16 mx-auto animate-pulse"
/>
```

```
                <h1 className="text-2xl mt-4">Authenticating & Loading System...</h1>
            </div>
        </div>
    );
  }

  return (
    <div className="flex h-screen bg-gray-900 text-gray-200 font-sans">
        <Sidebar activeTab={activeTab} setActiveTab={setActiveTab} />
        <main className="flex-1 p-6 overflow-y-auto">
            {renderContent()}
        </main>
    </div>
  );
}

// --- Sidebar Navigation ---
const Sidebar = ({ activeTab, setActiveTab }) => {
  const navItems = [
    { id: 'dashboard', icon: BarChart, label: 'Dashboard' },
    { id: 'inventory', icon: Warehouse, label: 'Inventory' },
    { id: 'products', icon: Package, label: 'Products' },
    { id: 'materials', icon: Box, label: 'Raw Materials' },
    { id: 'sales', icon: ShoppingCart, label: 'Sales Orders' },
```

```
    { id: 'manufacturing', icon: Factory, label: 'Manufacturing' },
  ];
  return (
    <nav className="w-64 bg-gray-900 border-r border-gray-700 p-4 flex flex-col">
      <div className="flex items-center mb-8">
        <Package className="w-10 h-10 text-indigo-400" />
        <h1 className="text-xl font-bold ml-3">FactoryFlow</h1>
      </div>
      <ul>
        {navItems.map(item => (
          <li key={item.id} className="mb-2">
            <button onClick={() => setActiveTab(item.id)} 
className={`w-full flex items-center p-3 rounded-lg text-left transition-colors ${activeTab === item.id ? 'bg-indigo-600 text-white' : 'hover:bg-gray-700'}`}>
              <item.icon className="w-5 h-5 mr-4" />
              <span>{item.label}</span>
            </button>
          </li>
        ))}
      </ul>
      <div className="mt-auto text-center text-xs text-gray-500">
        <p>Inventory Management System</p>
        <p>© 2024</p>
      </div>
```

```
        </nav>
    );
};

// --- Dashboard Component ---
const Dashboard = ({ setActiveTab, collectionPrefix }) => {
    const [stats, setStats] = useState({ sales: 0, openOrders: 0,
lowStockProducts: 0, lowStockMaterials: 0 });
    const [recentSales, setRecentSales] = useState([]);
    useEffect(() => {
        if (!collectionPrefix) return;
        const salesUnsub = onSnapshot(collection(db,
`${collectionPrefix}/salesOrders`), snapshot => {
            let totalSales = 0, openOrders = 0;
            const salesData = [];
            snapshot.forEach(doc => {
                const data = doc.data();
                totalSales += data.totalAmount;
                if (data.status === 'Unfulfilled') openOrders++;
                if (salesData.length < 5) salesData.push({ id: doc.id, ...data });
            });
            setStats(s => ({ ...s, sales: totalSales, openOrders }));
            setRecentSales(salesData);
        });
        const productsUnsub = onSnapshot(collection(db,
`${collectionPrefix}/products`), snapshot => {
```

```
            let lowStock = 0;
            snapshot.forEach(doc => {
                if (doc.data().stockOnHand < (doc.data().reorderPoint || 10))
lowStock++;
            });
            setStats(s => ({ ...s, lowStockProducts: lowStock }));
        });
        const materialsUnsub = onSnapshot(collection(db,
`${collectionPrefix}/rawMaterials`), snapshot => {
            let lowStock = 0;
            snapshot.forEach(doc => {
                if (doc.data().stockLevel < (doc.data().reorderPoint || 20))
lowStock++;
            });
            setStats(s => ({ ...s, lowStockMaterials: lowStock }));
        });
        return () => { salesUnsub(); productsUnsub(); materialsUnsub(); };
    }, [collectionPrefix]);
    const StatCard = ({ icon, title, value, color, onClick }) => (
        <div className={`bg-gray-800 p-6 rounded-lg shadow-lg flex
items-center cursor-pointer hover:bg-gray-700 transition-colors ${color}`}
onClick={onClick}>
            <div className="mr-4">{icon}</div>
            <div>
                <p className="text-gray-400 text-sm">{title}</p>
                <p className="text-2xl font-bold">{value}</p>
```

```
        </div>
      </div>
    );
    return (
      <div>
        <h2 className="text-3xl font-bold mb-6">Dashboard</h2>
        <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6 mb-8">
          <StatCard icon={<DollarSign className="w-8 h-8 text-green-400"/>} title="Total Revenue" value={`₹${stats.sales.toLocaleString()}`} color="border-l-4 border-green-400" />
          <StatCard icon={<ShoppingCart className="w-8 h-8 text-yellow-400"/>} title="Open Sales Orders" value={stats.openOrders} color="border-l-4 border-yellow-400" onClick={() => setActiveTab('sales')} />
          <StatCard icon={<Package className="w-8 h-8 text-red-400"/>} title="Low Stock Products" value={stats.lowStockProducts} color="border-l-4 border-red-400" onClick={() => setActiveTab('inventory')} />
          <StatCard icon={<Box className="w-8 h-8 text-orange-400"/>} title="Low Stock Materials" value={stats.lowStockMaterials} color="border-l-4 border-orange-400" onClick={() => setActiveTab('inventory')} />
        </div>
        <div className="bg-gray-800 p-6 rounded-lg shadow-lg">
          <h3 className="text-xl font-bold mb-4">Recent Sales
```

```
Activity</h3>
            <div className="overflow-x-auto">
                <table className="w-full text-left">
                    <thead>
                        <tr className="border-b border-gray-700">
                            <th className="p-3">Order ID</th>
                            <th className="p-3">Platform</th>
                            <th className="p-3">Total Amount</th>
                            <th className="p-3">Status</th>
                        </tr>
                    </thead>
                    <tbody>
                        {recentSales.map(sale => (
                            <tr key={sale.id} className="border-b border-
gray-600 hover:bg-gray-700">
                                <td className="p-3 font-mono text-
sm">{sale.id.slice(0, 8)}...</td>
                                <td className="p-3">{sale.platform}</td>
                                <td className="p-
3">₹{sale.totalAmount.toLocaleString()}</td>
                                <td className="p-3"><span className={`px-2
py-1 text-xs rounded-full ${sale.status === 'Fulfilled' ? 'bg-green-500/20
text-green-300' : 'bg-yellow-500/20 text-yellow-
300'}`}>{sale.status}</span></td>
                            </tr>
                        ))}
```

```
                    </tbody>
                </table>
            </div>
        </div>
    );
};

// --- Modal Component ---
const Modal = ({ isOpen, onClose, title, children }) => {
    if (!isOpen) return null;
    return (
        <div className="fixed inset-0 bg-black bg-opacity-70 flex items-center justify-center z-50">
            <div className="bg-gray-800 rounded-lg shadow-xl w-full max-w-2xl p-6 relative">
                <h3 className="text-xl font-bold mb-4">{title}</h3>
                <button onClick={onClose} className="absolute top-4 right-4 text-gray-400 hover:text-white"><X className="w-6 h-6" /></button>
                {children}
            </div>
        </div>
    );
};

// --- Products Component ---
```

```
const Products = ({ collectionPrefix }) => {
    const [products, setProducts] = useState([]);
    const [materials, setMaterials] = useState([]);
    const [isModalOpen, setIsModalOpen] = useState(false);
    const [newProduct, setNewProduct] = useState({ sku: '', productName: '',
category: '', sellingPrice: 0, stockOnHand: 0, reorderPoint: 10, bom: [] });
    useEffect(() => {
        if (!collectionPrefix) return;
        const unsubProducts = onSnapshot(collection(db,
`${collectionPrefix}/products`), snapshot =>
setProducts(snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }))));
        const unsubMaterials = onSnapshot(collection(db,
`${collectionPrefix}/rawMaterials`), snapshot =>
setMaterials(snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }))));
        return () => { unsubProducts(); unsubMaterials(); };
    }, [collectionPrefix]);
    const handleAddBomItem = () => setNewProduct(prev => ({ ...prev, bom:
[...prev.bom, { materialId: '', quantity: 1 }] }));
    const handleBomItemChange = (index, field, value) => {
        const updatedBom = [...newProduct.bom];
        updatedBom[index][field] = value;
        setNewProduct(prev => ({ ...prev, bom: updatedBom }));
    };
    const handleRemoveBomItem = (index) => setNewProduct(prev =>
({ ...prev, bom: prev.bom.filter((_, i) => i !== index) }));
    const handleAddProduct = async (e) => {
```

```
        e.preventDefault();
        if (!newProduct.sku || !newProduct.productName
|| !newProduct.sellingPrice) return;
        await addDoc(collection(db, `${collectionPrefix}/products`), {
            ...newProduct,
            sellingPrice: Number(newProduct.sellingPrice),
            stockOnHand: Number(newProduct.stockOnHand),
            reorderPoint: Number(newProduct.reorderPoint),
            bom: newProduct.bom.map(item => ({ ...item, quantity:
Number(item.quantity) }))
        });
        setIsModalOpen(false);
        setNewProduct({ sku: '', productName: '', category: '', sellingPrice: 0,
stockOnHand: 0, reorderPoint: 10, bom: [] });
    };
    return (
        <div>
            <div className="flex justify-between items-center mb-6">
                <h2 className="text-3xl font-bold">Products Master</h2>
                <button onClick={() => setIsModalOpen(true)} className="flex
items-center bg-indigo-600 text-white px-4 py-2 rounded-lg hover:bg-
indigo-700"><PlusCircle className="w-5 h-5 mr-2" />Add New
Product</button>
            </div>
            <div className="bg-gray-800 rounded-lg shadow-lg overflow-
hidden">
```

```
            <div className="overflow-x-auto">
                <table className="w-full text-left">
                    <thead>
                        <tr className="border-b border-gray-700 bg-gray-700/50">
                            <th className="p-4">SKU</th><th className="p-4">Product Name</th><th className="p-4">Category</th><th className="p-4">Selling Price</th><th className="p-4">Stock on Hand</th>
                        </tr>
                    </thead>
                    <tbody>
                        {products.map(product => (
                            <tr key={product.id} className="border-b border-gray-700 hover:bg-gray-700/50">
                                <td className="p-4 font-mono">{product.sku}</td>
                                <td className="p-4">{product.productName}</td>
                                <td className="p-4">{product.category}</td>
                                <td className="p-4">₹{product.sellingPrice.toLocaleString()}</td>
                                <td className={`p-4 font-bold ${product.stockOnHand < product.reorderPoint ? 'text-red-400' : 'text-green-400'}`}>{product.stockOnHand}</td>
                            </tr>
```

```
                        ))}
                    </tbody>
                </table>
            </div>
        </div>
        <Modal isOpen={isModalOpen} onClose={() => setIsModalOpen(false)}
title="Add New Product">
            <form onSubmit={handleAddProduct} className="space-y-4
max-h-[70vh] overflow-y-auto pr-2">
                <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                    <input type="text" placeholder="SKU"
value={newProduct.sku} onChange={e => setNewProduct({ ...newProduct, sku:
e.target.value })} className="bg-gray-700 p-2 rounded w-full" required />
                    <input type="text" placeholder="Product Name"
value={newProduct.productName} onChange={e =>
setNewProduct({ ...newProduct, productName: e.target.value })}
className="bg-gray-700 p-2 rounded w-full" required />
                </div>
                <input type="text" placeholder="Category"
value={newProduct.category} onChange={e =>
setNewProduct({ ...newProduct, category: e.target.value })} className="bg-
gray-700 p-2 rounded w-full" />
                <div className="grid grid-cols-1 md:grid-cols-3 gap-4">
                    <input type="number" placeholder="Selling Price"
value={newProduct.sellingPrice} onChange={e =>
setNewProduct({ ...newProduct, sellingPrice: e.target.value })}
```

```
className="bg-gray-700 p-2 rounded w-full" required />
                    <input type="number" placeholder="Initial Stock"
value={newProduct.stockOnHand} onChange={e =>
setNewProduct({ ...newProduct, stockOnHand: e.target.value })}
className="bg-gray-700 p-2 rounded w-full" />
                    <input type="number" placeholder="Reorder Point"
value={newProduct.reorderPoint} onChange={e =>
setNewProduct({ ...newProduct, reorderPoint: e.target.value })}
className="bg-gray-700 p-2 rounded w-full" />
                </div>
                <h4 className="text-lg font-bold mt-4 border-b border-
gray-600 pb-2">Bill of Materials (BOM)</h4>
                {newProduct.bom.map((item, index) => (
                    <div key={index} className="flex items-center gap-2">
                        <select value={item.materialId} onChange={e =>
handleBomItemChange(index, 'materialId', e.target.value)} className="bg-
gray-700 p-2 rounded w-full" required>
                            <option value="">Select Material</option>
                            {materials.map(m => <option key={m.id}
value={m.id}>{m.materialName}</option>)}
                        </select>
                        <input type="number" placeholder="Qty"
value={item.quantity} onChange={e => handleBomItemChange(index,
'quantity', e.target.value)} className="bg-gray-700 p-2 rounded w-1/4"
required />
                        <button type="button" onClick={() =>
```

```
handleRemoveBomItem(index)} className="text-red-400 hover:text-red-300 p-2"><Trash2 /></button>
                        </div>
                    ))}
                    <button type="button" onClick={handleAddBomItem} className="text-indigo-400 hover:text-indigo-300 flex items-center gap-2 text-sm"><PlusCircle className="w-4 h-4" /> Add Material</button>
                    <div className="flex justify-end pt-4"><button type="submit" className="bg-indigo-600 text-white px-6 py-2 rounded-lg hover:bg-indigo-700">Save Product</button></div>
                </form>
            </Modal>
        </div>
    );
};

// --- Raw Materials Component ---
const RawMaterials = ({ collectionPrefix }) => {
    const [materials, setMaterials] = useState([]);
    const [isModalOpen, setIsModalOpen] = useState(false);
    const [newMaterial, setNewMaterial] = useState({ materialName: '', supplier: '', costPerUnit: 0, stockLevel: 0, unitOfMeasure: 'units', reorderPoint: 20 });
    useEffect(() => {
        if (!collectionPrefix) return;
        const unsubscribe = onSnapshot(collection(db,
```

```
`${collectionPrefix}/rawMaterials`), snapshot =>
setMaterials(snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() })));
        return () => unsubscribe();
    }, [collectionPrefix]);
    const handleAddMaterial = async (e) => {
        e.preventDefault();
        await addDoc(collection(db, `${collectionPrefix}/rawMaterials`), {
            ...newMaterial,
            costPerUnit: Number(newMaterial.costPerUnit),
            stockLevel: Number(newMaterial.stockLevel),
            reorderPoint: Number(newMaterial.reorderPoint)
        });
        setIsModalOpen(false);
        setNewMaterial({ materialName: '', supplier: '', costPerUnit: 0,
stockLevel: 0, unitOfMeasure: 'units', reorderPoint: 20 });
    };
    return (
        <div>
            <div className="flex justify-between items-center mb-6">
                <h2 className="text-3xl font-bold">Raw Materials</h2>
                <button onClick={() => setIsModalOpen(true)} className="flex
items-center bg-indigo-600 text-white px-4 py-2 rounded-lg hover:bg-
indigo-700"><PlusCircle className="w-5 h-5 mr-2" />Add New
Material</button>
            </div>
            <div className="bg-gray-800 rounded-lg shadow-lg overflow-
```

```
hidden">
            <div className="overflow-x-auto">
                <table className="w-full text-left">
                    <thead>
                        <tr className="border-b border-gray-700 bg-gray-
700/50">
                            <th className="p-4">Material Name</th><th
className="p-4">Supplier</th><th className="p-4">Cost per Unit</th><th
className="p-4">Stock Level</th><th className="p-4">Unit</th>
                        </tr>
                    </thead>
                    <tbody>
                        {materials.map(material => (
                            <tr key={material.id} className="border-b border-
gray-700 hover:bg-gray-700/50">
                                <td className="p-
4">{material.materialName}</td>
                                <td className="p-4">{material.supplier}</td>
                                <td className="p-
4">₹{material.costPerUnit.toLocaleString()}</td>
                                <td className={`p-4 font-bold
${material.stockLevel < material.reorderPoint ? 'text-red-400' : 'text-
green-400'}`}>{material.stockLevel}</td>
                                <td className="p-
4">{material.unitOfMeasure}</td>
                            </tr>
```

```
                        ))}
                    </tbody>
                </table>
            </div>
        </div>
        <Modal isOpen={isModalOpen} onClose={() => setIsModalOpen(false)}
title="Add New Raw Material">
            <form onSubmit={handleAddMaterial} className="space-y-4">
                <input type="text" placeholder="Material Name"
value={newMaterial.materialName} onChange={e =>
setNewMaterial({ ...newMaterial, materialName: e.target.value })}
className="bg-gray-700 p-2 rounded w-full" required />
                <input type="text" placeholder="Supplier"
value={newMaterial.supplier} onChange={e =>
setNewMaterial({ ...newMaterial, supplier: e.target.value })} className="bg-
gray-700 p-2 rounded w-full" />
                <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                    <input type="number" placeholder="Cost per Unit"
value={newMaterial.costPerUnit} onChange={e =>
setNewMaterial({ ...newMaterial, costPerUnit: e.target.value })}
className="bg-gray-700 p-2 rounded w-full" />
                    <input type="number" placeholder="Initial Stock"
value={newMaterial.stockLevel} onChange={e =>
setNewMaterial({ ...newMaterial, stockLevel: e.target.value })}
className="bg-gray-700 p-2 rounded w-full" />
                </div>
```

```
                    <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
                        <input type="text" placeholder="Unit (e.g., kg, m, units)"
value={newMaterial.unitOfMeasure} onChange={e =>
setNewMaterial({ ...newMaterial, unitOfMeasure: e.target.value })}
className="bg-gray-700 p-2 rounded w-full" />
                        <input type="number" placeholder="Reorder Point"
value={newMaterial.reorderPoint} onChange={e =>
setNewMaterial({ ...newMaterial, reorderPoint: e.target.value })}
className="bg-gray-700 p-2 rounded w-full" />
                    </div>
                    <div className="flex justify-end pt-4"><button
type="submit" className="bg-indigo-600 text-white px-6 py-2 rounded-
lg hover:bg-indigo-700">Save Material</button></div>
                </form>
            </Modal>
        </div>
    );
};

// --- Inventory Component ---
const Inventory = ({ collectionPrefix }) => {
    const [products, setProducts] = useState([]);
    const [materials, setMaterials] = useState([]);
    useEffect(() => {
        if (!collectionPrefix) return;
        const unsubProducts = onSnapshot(collection(db,
```

```
`${collectionPrefix}/products`), snapshot =>
setProducts(snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() })));
        const unsubMaterials = onSnapshot(collection(db,
`${collectionPrefix}/rawMaterials`), snapshot =>
setMaterials(snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() })));
        return () => { unsubProducts(); unsubMaterials(); };
    }, [collectionPrefix]);
    return (
        <div>
            <h2 className="text-3xl font-bold mb-6">Inventory Status</h2>
            <div className="grid grid-cols-1 lg:grid-cols-2 gap-8">
                <div className="bg-gray-800 rounded-lg shadow-lg p-6">
                    <h3 className="text-xl font-bold mb-4 flex items-
center"><Package className="mr-2" /> Finished Products</h3>
                    <div className="overflow-y-auto max-h-[60vh]">
                        <table className="w-full text-left">
                            <thead><tr className="border-b border-gray-
700"><th className="p-3">Product</th><th className="p-
3">Stock</th><th className="p-3">Status</th></tr></thead>
                            <tbody>
                                {products.map(p => (<tr key={p.id}
className="border-b border-gray-600"><td className="p-
3">{p.productName} <span className="text-xs text-gray-
500">({p.sku})</span></td><td className="p-3 font-
bold">{p.stockOnHand}</td><td className="p-3">{p.stockOnHand <
p.reorderPoint ? <span className="text-red-400 flex items-
```

```
center"><ArrowDown className="w-4 h-4 mr-1"/>Low Stock</span> :
<span className="text-green-400 flex items-center"><ArrowUp
className="w-4 h-4 mr-1"/>In Stock</span>}</td></tr>))}
                        </tbody>
                    </table>
                </div>
            </div>
            <div className="bg-gray-800 rounded-lg shadow-lg p-6">
                <h3 className="text-xl font-bold mb-4 flex items-
center"><Box className="mr-2" /> Raw Materials</h3>
                <div className="overflow-y-auto max-h-[60vh]">
                    <table className="w-full text-left">
                        <thead><tr className="border-b border-gray-
700"><th className="p-3">Material</th><th className="p-
3">Stock</th><th className="p-3">Status</th></tr></thead>
                        <tbody>
                            {materials.map(m => (<tr key={m.id}
className="border-b border-gray-600"><td className="p-
3">{m.materialName}</td><td className="p-3 font-bold">{m.stockLevel}
{m.unitOfMeasure}</td><td className="p-3">{m.stockLevel <
m.reorderPoint ? <span className="text-red-400 flex items-
center"><ArrowDown className="w-4 h-4 mr-1"/>Low Stock</span> :
<span className="text-green-400 flex items-center"><ArrowUp
className="w-4 h-4 mr-1"/>In Stock</span>}</td></tr>))}
                        </tbody>
                    </table>
```

```
                </div>
            </div>
        </div>
    </div>
    );
};

// --- Sales Orders Component ---
const SalesOrders = ({ collectionPrefix }) => {
    const [sales, setSales] = useState([]);
    const [products, setProducts] = useState([]);
    const [isModalOpen, setIsModalOpen] = useState(false);
    const [newSale, setNewSale] = useState({ platform: 'Flipkart', productId:
'', quantity: 1, status: 'Unfulfilled' });
    useEffect(() => {
        if (!collectionPrefix) return;
        const q = query(collection(db, `${collectionPrefix}/salesOrders`),
orderBy('orderDate', 'desc'));
        const unsubSales = onSnapshot(q, snapshot =>
setSales(snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }))));
        const unsubProducts = onSnapshot(collection(db,
`${collectionPrefix}/products`), snapshot =>
setProducts(snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }))));
        return () => { unsubSales(); unsubProducts(); };
    }, [collectionPrefix]);
    const handleAddSale = async (e) => {
```

```
        e.preventDefault();
        const product = products.find(p => p.id === newSale.productId);
        if (!product) return;
        const batch = writeBatch(db);
        const saleRef = doc(collection(db, `${collectionPrefix}/salesOrders`));
        const productRef = doc(db, `${collectionPrefix}/products`,
newSale.productId);
        const totalAmount = product.sellingPrice * Number(newSale.quantity);
        batch.set(saleRef, { ...newSale, quantity: Number(newSale.quantity),
totalAmount, productName: product.productName, orderDate:
serverTimestamp() });
        const newStock = product.stockOnHand - Number(newSale.quantity);
        if (newStock < 0) { alert(`Not enough stock for
${product.productName}. Only ${product.stockOnHand} available.`); return; }
        batch.update(productRef, { stockOnHand: newStock });
        await batch.commit();
        setIsModalOpen(false);
        setNewSale({ platform: 'Flipkart', productId: '', quantity: 1, status:
'Unfulfilled' });
    };
    const markAsFulfilled = async (saleId) => {
        const saleRef = doc(db, `${collectionPrefix}/salesOrders`, saleId);
        await setDoc(saleRef, { status: 'Fulfilled' }, { merge: true });
    };
    return (
        <div>
```

```
            <div className="flex justify-between items-center mb-6">
                <h2 className="text-3xl font-bold">Sales Orders</h2>
                <button onClick={() => setIsModalOpen(true)} className="flex items-center bg-indigo-600 text-white px-4 py-2 rounded-lg hover:bg-indigo-700"><PlusCircle className="w-5 h-5 mr-2" />Add New Sale</button>
            </div>
            <div className="bg-gray-800 rounded-lg shadow-lg overflow-hidden">
                <div className="overflow-x-auto">
                    <table className="w-full text-left">
                        <thead>
                            <tr className="border-b border-gray-700 bg-gray-700/50">
                                <th className="p-4">Date</th><th className="p-4">Product</th><th className="p-4">Platform</th><th className="p-4">Qty</th><th className="p-4">Total</th><th className="p-4">Status</th><th className="p-4">Actions</th>
                            </tr>
                        </thead>
                        <tbody>
                            {sales.map(sale => (
                                <tr key={sale.id} className="border-b border-gray-700 hover:bg-gray-700/50">
                                    <td className="p-4">{sale.orderDate ? new Date(sale.orderDate.seconds * 1000).toLocaleDateString() : '...'}</td>
```

```
                                <td className="p-4">{sale.productName}</td>
                                <td className="p-4">{sale.platform}</td>
                                <td className="p-4">{sale.quantity}</td>
                                <td className="p-
4">₹{sale.totalAmount.toLocaleString()}</td>
                                <td className="p-4"><span className={`px-2
py-1 text-xs rounded-full ${sale.status === 'Fulfilled' ? 'bg-green-500/20
text-green-300' : 'bg-yellow-500/20 text-yellow-
300'}`}>{sale.status}</span></td>
                                <td className="p-4">{sale.status ===
'Unfulfilled' && (<button onClick={() => markAsFulfilled(sale.id)}
className="flex items-center text-xs bg-green-600 px-2 py-1 rounded
hover:bg-green-700"><Truck className="w-4 h-4 mr-1"/>
Fulfill</button>)}</td>
                            </tr>
                        ))}
                    </tbody>
                </table>
            </div>
        </div>
        <Modal isOpen={isModalOpen} onClose={() => setIsModalOpen(false)}
title="Add New Sales Order">
            <form onSubmit={handleAddSale} className="space-y-4">
                <select value={newSale.platform} onChange={e =>
setNewSale({ ...newSale, platform: e.target.value })} className="bg-gray-
700 p-2 rounded w-full">
```

```
<option>Flipkart</option><option>Myntra</option><option>Amazon</option
><option>Brand Website</option><option>Other</option>
                    </select>
                    <select value={newSale.productId} onChange={e =>
setNewSale({ ...newSale, productId: e.target.value })} className="bg-gray-
700 p-2 rounded w-full" required>
                        <option value="">Select Product</option>
                        {products.map(p => <option key={p.id}
value={p.id}>{p.productName} ({p.sku}) - Stock: {p.stockOnHand}</option>)}
                    </select>
                    <input type="number" placeholder="Quantity"
value={newSale.quantity} onChange={e => setNewSale({ ...newSale, quantity:
e.target.value })} className="bg-gray-700 p-2 rounded w-full" required
min="1" />
                    <div className="flex justify-end pt-4"><button
type="submit" className="bg-indigo-600 text-white px-6 py-2 rounded-
lg hover:bg-indigo-700">Log Sale</button></div>
                </form>
            </Modal>
        </div>
    );
};

// --- Manufacturing Orders Component ---
const ManufacturingOrders = ({ collectionPrefix }) => {
```

```
    const [orders, setOrders] = useState([]);
    const [products, setProducts] = useState([]);
    const [materials, setMaterials] = useState([]);
    const [isModalOpen, setIsModalOpen] = useState(false);
    const [newOrder, setNewOrder] = useState({ productId: '',
quantityToProduce: 10 });
    const [materialCheck, setMaterialCheck] = useState(null);
    useEffect(() => {
        if (!collectionPrefix) return;
        const q = query(collection(db,
`${collectionPrefix}/manufacturingOrders`), orderBy('creationDate', 'desc'));
        const unsubOrders = onSnapshot(q, snapshot =>
setOrders(snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }))));
        const unsubProducts = onSnapshot(collection(db,
`${collectionPrefix}/products`), snapshot =>
setProducts(snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }))));
        const unsubMaterials = onSnapshot(collection(db,
`${collectionPrefix}/rawMaterials`), snapshot =>
setMaterials(snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }))));
        return () => { unsubOrders(); unsubProducts(); unsubMaterials(); };
    }, [collectionPrefix]);
    const checkMaterials = () => {
        const product = products.find(p => p.id === newOrder.productId);
        if (!product || !product.bom) { setMaterialCheck(null); return; }
        const needed = product.bom.map(item => {
            const material = materials.find(m => m.id === item.materialId);
```

```
            const required = item.quantity *
Number(newOrder.quantityToProduce);
            const available = material ? material.stockLevel : 0;
            return { name: material ? material.materialName : 'Unknown',
required, available, sufficient: available >= required };
        });
        setMaterialCheck(needed);
    };
    useEffect(checkMaterials, [newOrder, products, materials]);
    const handleCreateOrder = async (e) => {
        e.preventDefault();
        if (!materialCheck || materialCheck.some(m => !m.sufficient))
{ alert("Not enough raw materials to start production."); return; }
        const product = products.find(p => p.id === newOrder.productId);
        const batch = writeBatch(db);
        const moRef = doc(collection(db,
`${collectionPrefix}/manufacturingOrders`));
        batch.set(moRef, { ...newOrder, quantityToProduce:
Number(newOrder.quantityToProduce), productName:
product.productName, status: 'In Progress', creationDate:
serverTimestamp() });
        product.bom.forEach(item => {
            const materialRef = doc(db, `${collectionPrefix}/rawMaterials`,
item.materialId);
            const material = materials.find(m => m.id === item.materialId);
            const quantityNeeded = item.quantity *
```

```
Number(newOrder.quantityToProduce);
            batch.update(materialRef, { stockLevel: material.stockLevel -
quantityNeeded });
        });
        await batch.commit();
        setIsModalOpen(false);
        setNewOrder({ productId: '', quantityToProduce: 10 });
        setMaterialCheck(null);
    };
    const markAsCompleted = async (orderId) => {
        const order = orders.find(o => o.id === orderId);
        const product = products.find(p => p.id === order.productId);
        const batch = writeBatch(db);
        const moRef = doc(db, `${collectionPrefix}/manufacturingOrders`,
orderId);
        batch.update(moRef, { status: 'Completed', completionDate:
serverTimestamp() });
        const productRef = doc(db, `${collectionPrefix}/products`,
order.productId);
        batch.update(productRef, { stockOnHand: product.stockOnHand +
order.quantityToProduce });
        await batch.commit();
    };
    return (
        <div>
            <div className="flex justify-between items-center mb-6">
```

```
            <h2 className="text-3xl font-bold">Manufacturing
Orders</h2>
            <button onClick={() => setIsModalOpen(true)} className="flex
items-center bg-indigo-600 text-white px-4 py-2 rounded-lg hover:bg-
indigo-700"><PlusCircle className="w-5 h-5 mr-2" />Create New
Order</button>
        </div>
        <div className="bg-gray-800 rounded-lg shadow-lg overflow-
hidden">
            <div className="overflow-x-auto">
                <table className="w-full text-left">
                    <thead>
                        <tr className="border-b border-gray-700 bg-gray-
700/50">
                            <th className="p-4">Date Created</th><th
className="p-4">Product</th><th className="p-4">Qty to
Produce</th><th className="p-4">Status</th><th className="p-
4">Actions</th>
                        </tr>
                    </thead>
                    <tbody>
                        {orders.map(order => (
                            <tr key={order.id} className="border-b border-
gray-700 hover:bg-gray-700/50">
                                <td className="p-4">{order.creationDate ? new
Date(order.creationDate.seconds * 1000).toLocaleDateString() : '...'}</td>
```

```
                                <td className="p-
4">{order.productName}</td>
                                <td className="p-
4">{order.quantityToProduce}</td>
                                <td className="p-4"><span className={`px-2
py-1 text-xs rounded-full ${order.status === 'Completed' ? 'bg-green-
500/20 text-green-300' : 'bg-blue-500/20 text-blue-
300'}`}>{order.status}</span></td>
                                <td className="p-4">{order.status === 'In
Progress' && (<button onClick={() => markAsCompleted(order.id)}
className="flex items-center text-xs bg-green-600 px-2 py-1 rounded
hover:bg-green-700"><Factory className="w-4 h-4 mr-1"/> Mark as
Completed</button>)}</td>
                            </tr>
                        ))}
                    </tbody>
                </table>
            </div>
        </div>
        <Modal isOpen={isModalOpen} onClose={() => setIsModalOpen(false)}
title="Create Manufacturing Order">
            <form onSubmit={handleCreateOrder} className="space-y-4">
                <select value={newOrder.productId} onChange={e =>
setNewOrder({ ...newOrder, productId: e.target.value })} className="bg-
gray-700 p-2 rounded w-full" required>
                    <option value="">Select Product to Manufacture</option>
```

```
                    {products.map(p => <option key={p.id}
value={p.id}>{p.productName} ({p.sku})</option>)}
                </select>
                <input type="number" placeholder="Quantity to Produce"
value={newOrder.quantityToProduce} onChange={e =>
setNewOrder({ ...newOrder, quantityToProduce: e.target.value })}
className="bg-gray-700 p-2 rounded w-full" required min="1" />
                {materialCheck && (
                    <div className="bg-gray-700/50 p-4 rounded-lg">
                        <h4 className="font-bold mb-2">Required
Materials:</h4>
                        <ul className="space-y-1 text-sm">
                            {materialCheck.map((m, i) => (<li key={i}
className={`flex justify-between ${m.sufficient ? 'text-gray-300' : 'text-
red-400 font-
bold'}`}><span>{m.name}</span><span>{m.required.toLocaleString()} /
{m.available.toLocaleString()}</span></li>))}
                        </ul>
                    </div>
                )}
                <div className="flex justify-end pt-4"><button
type="submit" className="bg-indigo-600 text-white px-6 py-2 rounded-
lg hover:bg-indigo-700 disabled:bg-gray-500" disabled={!materialCheck ||
materialCheck.some(m => !m.sufficient)}>Start Production</button></div>
            </form>
        </Modal>
```

```
        </div>
    );
};
```

Step 9: Configure Vite

Vite needs a configuration file to know how to run the project. Create a file named vite.config.js in your root folder and replace its content with this.

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

// https://vitejs.dev/config/
export default defineConfig({
  plugins: [react()],
})
```

Step 10: Add the "dev" Script

Open your package.json file and replace its entire content with the following valid JSON. This ensures there are no syntax errors.

```
{
  "name": "factoryflow-inventory-app",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
```

```
    "dev": "vite",
    "build": "vite build",
    "lint": "eslint . --ext js,jsx --report-unused-disable-directives --max-warnings 0",
    "preview": "vite preview"
  },
  "dependencies": {
    "firebase": "^10.12.2",
    "lucide-react": "^0.395.0",
    "react": "^18.3.1",
    "react-dom": "^18.3.1"
  },
  "devDependencies": {
    "@tailwindcss/postcss": "^4.0.0-alpha.13",
    "@types/react": "^18.3.3",
    "@types/react-dom": "^18.3.0",
    "@vitejs/plugin-react": "^4.3.1",
    "autoprefixer": "^10.4.19",
    "eslint": "^8.57.0",
    "eslint-plugin-react": "^7.34.2",
    "eslint-plugin-react-hooks": "^4.6.2",
    "eslint-plugin-react-refresh": "^0.4.7",
    "postcss": "^8.4.38",
    "tailwindcss": "3.4.4",
    "vite": "^5.3.1"
  }
```

}

Step 11: Set Up Firestore and Security Rules

For the app to work locally, you must configure your Firebase database.

1. **Navigate to Firestore:** In your Firebase project console, go to the "Build" section and click on "Firestore Database".
2. **Create Database:** Click "Create database" and start in **test mode**. This allows open access during development.**Warning:** Test mode allows anyone to read and write to your database. It is only for initial development. Before launching a real application, you must set up secure rules.
3. **Use these rules for development:** Go to the "Rules" tab in your Firestore database and paste the following, then click "Publish":

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    // Allow read/write access to all users in their own "users/{userId}" path
    match /users/{userId}/{document=**} {
      allow read, write: if request.auth != null && request.auth.uid == userId;
    }
  }
}
```

Step 12: Run the Application!

You are all set! Open your terminal in the project folder and run the final command:

npm run dev

Your terminal will show you a local URL, usually http://localhost:5173. Open this URL in your web browser, and you should see the FactoryFlow inventory management application running.

Troubleshooting

If the app is stuck on the **"Authenticating & Loading System..."** screen, it's almost always a Firebase configuration issue. Follow these two steps to fix it.

1. Double-Check Your Firebase Config

The most common error is a typo when copying the configuration from Firebase.

- Go back to your **Firebase Project Settings** -> **General** -> **Your apps**.
- Select your web app and find the **Firebase SDK snippet** -> **Config**.
- Carefully compare **every single character** of the apiKey, authDomain, projectId, etc., with what you have in your App.jsx file. Even one character being wrong will cause it to fail.

2. Enable Anonymous Authentication

This application uses anonymous sign-in so you don't have to create a username and password. You must enable this in your Firebase project.

- In the Firebase Console, go to the **Build** section on the left and click on

Authentication.

- Go to the **Sign-in method** tab.
- Find **"Anonymous"** in the list of providers and click on it.
- **Enable** the toggle switch and click **Save**.

After verifying both of these steps, restart your development server by stopping the npm run dev command (press Ctrl + C in the terminal) and running it again.