**POS & Northwind Business Simulation – React Requirements**

**1. Project Overview**

**Purpose:**  
A web-based simulation of a wholesale/retail trading company (like Northwind Traders) to manage products, inventory, customers, suppliers, orders, employees, shippers, and reports.

**Target Platform:**

* **Frontend:** React 18+ (Vite or CRA)
* **State Management:** Redux Toolkit / React Query
* **UI Libraries:** Bootstrap 5 / MUI (Material-UI)
* **Backend (optional):** Node.js + Express or ASP.NET Core API
* **Database (optional):** SQL Server / SQLite / In-memory JSON

**Core Modules:**

1. Product & Inventory Management
2. Customer Management
3. Supplier Management
4. Employee & Territory Management
5. Orders / Sales Management
6. Shippers / Logistics
7. Reporting & Analytics
8. Optional Enhancements: Authentication, Role-based access, Export reports

**2. Functional Requirements**

**2.1 Products & Inventory**

* CRUD Products: Add, update, delete, view details
* Attributes: ID, Name, Category, Supplier, Price, Stock
* Categories: CRUD management
* Suppliers: CRUD and link to products
* Stock Management: Increase/Reduce stock, low-stock alerts
* Filters & Search: By category, supplier, stock status

**Menu Workflow:**

* Products → List / Add / Edit / Delete / Stock Adjustments
* Categories → List / Add / Edit / Delete

**2.2 Customers**

* CRUD customer info: ID, Company, Contact, Address, City, Country, Phone
* Assign orders to customers
* Search & Filter by name, city, or country

**Menu Workflow:**

* Customers → List / Add / Edit / Delete
* Orders → Create order for selected customer

**2.3 Suppliers**

* CRUD suppliers
* Link products to suppliers
* Attributes: ID, Company, Contact, Address, City, Country, Phone

**Menu Workflow:**

* Suppliers → List / Add / Edit / Delete
* Products → Link/Unlink supplier

**2.4 Employees & Territories**

* CRUD employees: ID, Name, Title, HireDate
* CRUD territories: ID, Name
* Assign employees to territories
* Assign orders to employees by territory

**Menu Workflow:**

* Employees → List / Add / Edit / Delete
* Territories → List / Add / Edit / Delete
* Employee → Assign Territories

**2.5 Orders / Sales**

* Create orders workflow:
  1. Select customer
  2. Add products with quantity
  3. Validate stock availability
  4. Assign employee and shipper
  5. Confirm order → reduce stock
* Order Attributes: ID, Customer, Employee, Shipper, Products, Total, Date
* Track order history per customer
* Edit / Cancel orders

**Menu Workflow:**

* Orders → List / Add / Edit / Cancel / View Details

**2.6 Shippers / Logistics**

* CRUD shippers
* Assign shippers to orders
* Attributes: ID, Company, Contact, Phone

**Menu Workflow:**

* Shippers → List / Add / Edit / Delete
* Orders → Assign shipper

**2.7 Reporting & Analytics**

* Top-selling products
* Sales per customer / category / employee / territory
* Revenue and profit reports
* Inventory reports (stock levels, low-stock alerts)
* Export reports (CSV/Excel optional)

**Menu Workflow:**

* Reports → Products / Customers / Categories / Employees / Inventory
* Export → CSV / Excel

**3. Non-Functional Requirements**

* **Performance:** Handle 1000+ products, customers, orders efficiently
* **Usability:** Clean, responsive UI, intuitive navigation
* **Maintainability:** Modular components & services per business area
* **Persistence:** Optional database or localStorage for offline use
* **Security:** Optional role-based authentication (Admin, Cashier)

**4. Technical Requirements**

* **Language & Framework:** React 18+ (TypeScript optional)
* **State Management:** Redux Toolkit / React Query / Context API
* **UI Library:** Bootstrap 5 / MUI (Material-UI)
* **Data Handling:** Axios / Fetch API
* **Backend:** Optional Node.js / ASP.NET Core API
* **Database:** SQL Server / SQLite / In-memory

**Architecture:**

* Layered & modular:
* src/
* ├── components/ # UI components
* ├── pages/ # Page-level components
* ├── features/ # Redux slices / React Query hooks per module
* ├── services/ # API calls
* ├── utils/ # Helper functions
* ├── App.jsx
* └── main.jsx

**5. UI / Menu Structure**

**Main Menu / Sidebar:**

* Dashboard
* Products
  + List Products
  + Add Product
  + Categories
  + Stock Management
* Customers
  + List Customers
  + Add Customer
  + Order History
* Suppliers
  + List Suppliers
  + Add Supplier
  + Link Products
* Employees
  + List Employees
  + Add Employee
  + Territories
* Orders
  + List Orders
  + Add Order
  + Track Order Status
* Shippers
  + List Shippers
  + Add Shipper
* Reports
  + Sales Reports
  + Inventory Reports
  + Export Reports
* Settings (optional)
  + User Management
  + Roles & Permissions

**Page / Workflow Example: Add Order**

1. Select Customer → Dropdown / Search
2. Add Products → Table with search + add quantity
3. Assign Employee → Dropdown based on territory
4. Assign Shipper → Dropdown
5. Confirm Order → Reduce stock, generate order summary

**6. Optional Enhancements**

* Discounts / Promotions per order
* Multi-currency support
* Authentication & Role-based access (Admin / Cashier)
* Export reports to CSV/Excel
* Responsive UI (Desktop + Mobile)
* Notification system (stock alerts, order updates)

**✅ Suggested Development Steps**

1. **Products & Inventory Module** → CRUD + stock management
2. **Customer Module** → CRUD + order linking
3. **Supplier Module** → CRUD + product linking
4. **Employee & Territory Module** → CRUD + assignment
5. **Shippers Module** → CRUD
6. **Orders Module** → Full workflow
7. **Reports Module** → Sales, inventory, analytics
8. **Optional Enhancements** → Authentication, export, promotions

If you want, I can **draw a visual React sidebar/menu layout with all modules and sub-menus** showing workflows for your POS project—this makes it much easier for dev planning.

Do you want me to do that?

Northwind Business Simulation with purchase,return,stock adjustment full elaborate requirement document

Okay, here's a comprehensive requirements document for a Northwind Business Simulation that includes purchase orders, returns, and stock adjustments. This document aims to be detailed enough to guide the development or selection of such a simulation.

**Northwind Business Simulation: Purchase Orders, Returns, and Stock Adjustments**

**1. Introduction**

This document outlines the requirements for a business simulation based on the Northwind Traders database. The simulation will focus on the procurement, inventory management, and sales aspects of the business, with a particular emphasis on purchase orders, returns (both customer and vendor), and stock adjustments. The goal is to provide users with a realistic and interactive experience of managing a supply chain and inventory within the context of a fictional company.

**2. Goals and Objectives**

* **Educational:** To teach users about the processes involved in purchasing, inventory control, and handling returns.
* **Strategic Thinking:** To encourage users to make informed decisions about inventory levels, supplier selection, and pricing strategies.
* **Problem-Solving:** To challenge users to address issues such as stockouts, overstocking, and quality problems.
* **Data Analysis:** To provide users with the tools to analyze key performance indicators (KPIs) related to procurement and inventory.
* **Decision-Making:** To allow users to make decisions that directly impact the simulated company's profitability and customer satisfaction.

**3. Target Audience**

* Business students (undergraduate and graduate)
* Supply chain management professionals
* Inventory managers
* Individuals interested in learning about business operations

**4. Functional Requirements**

**4.1. Core Functionality:**

* **Product Catalog:**
  + Display a list of products with details (name, description, unit price, supplier, category, reorder level, units in stock, units on order).
  + Allow users to search and filter products.
  + Display product images (optional).
* **Supplier Management:**
  + Maintain a list of suppliers with contact information, lead times, and payment terms.
  + Allow users to view supplier performance metrics (e.g., on-time delivery rate, defect rate).
  + Enable adding new suppliers (with appropriate validation).
* **Customer Management:**
  + Maintain a list of customers with contact information and order history.
  + Allow users to view customer order patterns and preferences.
* **Sales Orders:**
  + Allow users to create and process sales orders.
  + Automatically update inventory levels upon order fulfillment.
  + Calculate order totals, including discounts and shipping costs.
  + Generate invoices.
* **Reporting:**
  + Generate reports on sales, inventory levels, purchase orders, returns, and financial performance.
  + Allow users to customize reports based on various criteria (e.g., date range, product category, supplier).
  + Provide graphical representations of data (e.g., charts, graphs).

**4.2. Purchase Order Management:**

* **Purchase Order Creation:**
  + Automatically generate purchase orders based on reorder levels and demand forecasts.
  + Allow users to manually create purchase orders.
  + Select suppliers for each purchase order.
  + Specify quantities, prices, and delivery dates.
  + Calculate total purchase order cost.
  + Track purchase order status (e.g., pending, approved, sent, received, partially received, completed, cancelled).
* **Purchase Order Approval:**
  + Implement an approval workflow for purchase orders exceeding a certain value.
  + Allow designated users to approve or reject purchase orders.
  + Provide a reason for rejection.
* **Goods Receipt:**
  + Record the receipt of goods against a purchase order.
  + Update inventory levels upon receipt.
  + Handle partial deliveries.
  + Record any discrepancies between the purchase order and the received goods (e.g., quantity differences, damaged goods).
* **Purchase Order Modification:**
  + Allow users to modify purchase orders before they are fully received (subject to approval if necessary).
  + Track changes made to purchase orders.
* **Purchase Order Cancellation:**
  + Allow users to cancel purchase orders (with appropriate justification).
  + Handle any associated costs or penalties.

**4.3. Returns Management:**

* **Customer Returns:**
  + Allow customers to initiate returns for various reasons (e.g., damaged goods, wrong item, dissatisfaction).
  + Record the reason for the return.
  + Generate return authorization numbers (RANs).
  + Track the status of the return (e.g., requested, approved, received, inspected, processed).
  + Inspect returned goods to determine their condition.
  + Issue refunds or replacements based on the inspection results.
  + Update inventory levels based on the disposition of the returned goods (e.g., restock, scrap).
* **Vendor Returns:**
  + Allow users to initiate returns to suppliers for defective or incorrect goods.
  + Record the reason for the return.
  + Generate return merchandise authorizations (RMAs).
  + Track the status of the return (e.g., requested, approved, shipped, received, processed).
  + Request replacements, credits, or refunds from the supplier.
  + Update inventory levels based on the outcome of the return.
* **Return Analysis:**
  + Generate reports on return rates, reasons for returns, and associated costs.
  + Identify products or suppliers with high return rates.

**4.4. Stock Adjustments:**

* **Manual Stock Adjustments:**
  + Allow users to manually adjust inventory levels to account for discrepancies (e.g., spoilage, theft, damage, cycle counts).
  + Record the reason for the adjustment.
  + Require authorization for adjustments exceeding a certain threshold.
  + Track all stock adjustments with date, time, user, product, quantity, and reason.
* **Automated Stock Adjustments:**
  + Implement automated stock adjustments based on predefined rules (e.g., write-offs for expired products).
* **Stock Adjustment Reporting:**
  + Generate reports on stock adjustments, including the reasons for adjustments and the impact on inventory value.
  + Identify patterns of stock discrepancies.

**4.5. Forecasting and Planning:**

* **Demand Forecasting:**
  + Implement basic demand forecasting techniques (e.g., moving average, exponential smoothing).
  + Allow users to adjust forecast parameters.
  + Display forecast accuracy metrics.
* **Inventory Planning:**
  + Calculate optimal order quantities based on demand forecasts, lead times, and holding costs.
  + Suggest reorder points and safety stock levels.

**4.6. Financial Aspects:**

* **Cost of Goods Sold (COGS):** Calculate COGS based on purchase prices and inventory levels.
* **Inventory Valuation:** Use a standard inventory valuation method (e.g., FIFO, weighted average).
* **Profit and Loss (P&L) Statement:** Generate a simplified P&L statement based on sales, COGS, and operating expenses.
* **Balance Sheet:** Generate a simplified balance sheet showing assets, liabilities, and equity.

**5. Non-Functional Requirements**

* **Usability:**
  + The simulation should be easy to use and intuitive.
  + The user interface should be clear and well-organized.
  + The simulation should provide helpful feedback to the user.
* **Performance:**
  + The simulation should respond quickly to user actions.
  + The simulation should be able to handle a large number of products, suppliers, and customers.
* **Scalability:**
  + The simulation should be scalable to accommodate future growth.
* **Security:**
  + The simulation should protect sensitive data from unauthorized access.
* **Accessibility:**
  + The simulation should be accessible to users with disabilities (following WCAG guidelines).
* **Platform:**
  + Specify the target platform (e.g., web-based, desktop application).
* **Data Persistence:**
  + The simulation should save user progress and data.

**6. Data Model (Simplified)**

This is a simplified representation. A full database schema would be more detailed.

* **Products:** ProductID, ProductName, SupplierID, CategoryID, UnitPrice, UnitsInStock, UnitsOnOrder, ReorderLevel, Discontinued
* **Suppliers:** SupplierID, CompanyName, ContactName, ContactTitle, Address, City, Region, PostalCode, Country, Phone, Fax
* **Customers:** CustomerID, CompanyName, ContactName, ContactTitle, Address, City, Region, PostalCode, Country, Phone, Fax
* **Categories:** CategoryID, CategoryName, Description, Picture
* **Orders:** OrderID, CustomerID, EmployeeID, OrderDate, RequiredDate, ShippedDate, ShipVia, Freight, ShipName, ShipAddress, ShipCity, ShipRegion, ShipPostalCode, ShipCountry
* **Order Details:** OrderID, ProductID, UnitPrice, Quantity, Discount
* **PurchaseOrders:** PurchaseOrderID, SupplierID, OrderDate, DeliveryDate, CreatedBy, ApprovedBy, Status
* **PurchaseOrderDetails:** PurchaseOrderID, ProductID, Quantity, UnitPrice, ReceivedQuantity
* **CustomerReturns:** ReturnID, CustomerID, OrderID, ProductID, ReturnDate, Reason, Quantity, Disposition (e.g., Restock, Scrap, Refund)
* **VendorReturns:** ReturnID, PurchaseOrderID, ProductID, ReturnDate, Reason, Quantity, Disposition (e.g., Replacement, Credit, Refund)
* **StockAdjustments:** AdjustmentID, ProductID, AdjustmentDate, QuantityChange, Reason, UserID

**7. User Interface (UI) Considerations**

* **Dashboard:** A central dashboard providing an overview of key metrics (e.g., sales, inventory levels, profit margin).
* **Navigation:** A clear and intuitive navigation system.
* **Data Entry Forms:** Well-designed forms for entering data (e.g., creating purchase orders, processing returns).
* **Data Grids:** Display data in tabular format with sorting and filtering capabilities.
* **Visualizations:** Use charts and graphs to present data in a visually appealing and informative way.
* **Alerts and Notifications:** Provide alerts for critical events (e.g., low stock levels, overdue purchase orders).

**8. Simulation Parameters**

* **Simulation Speed:** Allow users to adjust the speed of the simulation (e.g., real-time, accelerated).
* **Initial Conditions:** Define the initial state of the business (e.g., starting inventory levels, cash balance).
* **Random Events:** Introduce random events (e.g., unexpected demand surges, supplier disruptions, quality problems) to make the simulation more realistic.
* **Difficulty Level:** Offer different difficulty levels by adjusting parameters such as demand variability, supplier reliability, and cost pressures.

**9. Reporting and Analytics**

* **Key Performance Indicators (KPIs):**
  + Inventory Turnover
  + Fill Rate
  + On-Time Delivery Rate
  + Return Rate
  + Gross Profit Margin
  + Net Profit Margin
  + Days Sales of Inventory (DSI)
* **Report Types:**
  + Sales Reports
  + Inventory Reports
  + Purchase Order Reports
  + Return Reports
  + Stock Adjustment Reports
  + Financial Reports
* **Data Export:** Allow users to export data in various formats (e.g., CSV, Excel).

**10. Technology Stack (Example)**

* **Frontend:** HTML, CSS, JavaScript, React, Angular, or Vue.js
* **Backend:** Node.js, Python (with Django or Flask), Java (with Spring), or .NET
* **Database:** MySQL, PostgreSQL, MongoDB, or SQL Server

**11. Testing**

* **Unit Testing:** Test individual components of the simulation.
* **Integration Testing:** Test the interaction between different components.
* **System Testing:** Test the entire simulation as a whole.
* **User Acceptance Testing (UAT):** Allow target users to test the simulation and provide feedback.

**12. Future Enhancements (Optional)**

* **Multi-User Mode:** Allow multiple users to participate in the simulation simultaneously.
* **Advanced Forecasting Techniques:** Implement more sophisticated forecasting methods (e.g., ARIMA, machine learning).
* **Supply Chain Optimization:** Provide tools for optimizing the supply chain (e.g., network design, transportation planning).
* **Integration with External Systems:** Integrate with real-world data sources (e.g., market data, weather forecasts).
* **Gamification:** Incorporate game mechanics (e.g., points, badges, leaderboards) to increase user engagement.

**13. Success Criteria**

* The simulation accurately models the core business processes of Northwind Traders.
* Users find the simulation engaging and educational.
* Users are able to make informed decisions that improve the simulated company's performance.
* The simulation meets the non-functional requirements for usability, performance, and security.

This document provides a comprehensive framework for developing a Northwind Business Simulation. Remember to tailor these requirements to your specific needs and resources. Good luck!

new tables needed beside northwind database tables

Okay, based on the requirements document I provided, here's a breakdown of the new tables needed *in addition* to the standard Northwind database tables, along with their purpose and key fields. These tables are primarily to support the simulation's specific features like purchase order management, returns, stock adjustments, and user management (if applicable).

**1. PurchaseOrders Table:**

* **Purpose:** Stores information about purchase orders placed with suppliers.
* **Fields:**
  + PurchaseOrderID (INT, Primary Key, Identity/Auto-increment) - Unique identifier for the purchase order.
  + SupplierID (INT, Foreign Key referencing Suppliers.SupplierID) - The supplier the purchase order is placed with.
  + OrderDate (DATETIME) - Date the purchase order was created.
  + DeliveryDate (DATETIME) - Expected delivery date.
  + CreatedBy (INT, Foreign Key referencing Users.UserID - if user management is implemented) - User who created the purchase order.
  + ApprovedBy (INT, Foreign Key referencing Users.UserID - if user management is implemented) - User who approved the purchase order (if approval workflow is in place).
  + Status (VARCHAR(50)) - Current status of the purchase order (e.g., "Pending," "Approved," "Sent," "Received," "Cancelled"). Consider using an ENUM or lookup table for this.
  + Notes (TEXT, Optional) - Any notes or comments related to the purchase order.

**2. PurchaseOrderDetails Table:**

* **Purpose:** Stores the individual line items within a purchase order.
* **Fields:**
  + PurchaseOrderID (INT, Foreign Key referencing PurchaseOrders.PurchaseOrderID, Part of Composite Key) - The purchase order this line item belongs to.
  + ProductID (INT, Foreign Key referencing Products.ProductID, Part of Composite Key) - The product being ordered.
  + Quantity (INT) - Quantity of the product ordered.
  + UnitPrice (DECIMAL) - Agreed-upon unit price for the product.
  + ReceivedQuantity (INT) - Quantity of the product actually received. This allows for tracking partial deliveries.
  + Notes (TEXT, Optional) - Any notes specific to this line item.

**3. CustomerReturns Table:**

* **Purpose:** Stores information about customer returns.
* **Fields:**
  + ReturnID (INT, Primary Key, Identity/Auto-increment) - Unique identifier for the return.
  + CustomerID (INT, Foreign Key referencing Customers.CustomerID) - The customer making the return.
  + OrderID (INT, Foreign Key referencing Orders.OrderID, Optional - if the return isn't directly tied to a specific order) - The order the returned product was originally part of.
  + ProductID (INT, Foreign Key referencing Products.ProductID) - The product being returned.
  + ReturnDate (DATETIME) - Date the return was initiated.
  + Reason (VARCHAR(255)) - Reason for the return (e.g., "Damaged," "Wrong Item," "Defective"). Consider using an ENUM or lookup table.
  + Quantity (INT) - Quantity of the product being returned.
  + Disposition (VARCHAR(50)) - How the return was handled (e.g., "Restock," "Scrap," "Refund," "Replacement"). Consider using an ENUM or lookup table.
  + Notes (TEXT, Optional) - Any notes related to the return.

**4. VendorReturns Table:**

* **Purpose:** Stores information about returns made *to* suppliers.
* **Fields:**
  + ReturnID (INT, Primary Key, Identity/Auto-increment) - Unique identifier for the return.
  + PurchaseOrderID (INT, Foreign Key referencing PurchaseOrders.PurchaseOrderID) - The purchase order the returned product was originally part of.
  + ProductID (INT, Foreign Key referencing Products.ProductID) - The product being returned.
  + ReturnDate (DATETIME) - Date the return was initiated.
  + Reason (VARCHAR(255)) - Reason for the return (e.g., "Defective," "Incorrect Shipment"). Consider using an ENUM or lookup table.
  + Quantity (INT) - Quantity of the product being returned.
  + Disposition (VARCHAR(50)) - How the return was handled (e.g., "Replacement," "Credit," "Refund"). Consider using an ENUM or lookup table.
  + Notes (TEXT, Optional) - Any notes related to the return.

**5. StockAdjustments Table:**

* **Purpose:** Stores information about manual or automated adjustments to inventory levels.
* **Fields:**
  + AdjustmentID (INT, Primary Key, Identity/Auto-increment) - Unique identifier for the adjustment.
  + ProductID (INT, Foreign Key referencing Products.ProductID) - The product being adjusted.
  + AdjustmentDate (DATETIME) - Date of the adjustment.
  + QuantityChange (INT) - The amount the inventory level was changed (positive for additions, negative for reductions).
  + Reason (VARCHAR(255)) - Reason for the adjustment (e.g., "Spoilage," "Theft," "Cycle Count," "Write-off"). Consider using an ENUM or lookup table.
  + UserID (INT, Foreign Key referencing Users.UserID - if user management is implemented) - User who made the adjustment (if manual).
  + Notes (TEXT, Optional) - Any notes related to the adjustment.

**6. Users Table (Optional, but Recommended for Security and Auditability):**

* **Purpose:** Stores information about users who can access and interact with the simulation. This is crucial for tracking who created purchase orders, approved them, or made stock adjustments.
* **Fields:**
  + UserID (INT, Primary Key, Identity/Auto-increment) - Unique identifier for the user.
  + Username (VARCHAR(50), Unique) - User's login username.
  + Password (VARCHAR(255)) - Hashed password. **Important: Never store passwords in plain text!**
  + FirstName (VARCHAR(50)) - User's first name.
  + LastName (VARCHAR(50)) - User's last name.
  + Email (VARCHAR(100), Unique) - User's email address.
  + Role (VARCHAR(50)) - User's role (e.g., "Administrator," "Purchaser," "Inventory Manager"). Consider using an ENUM or lookup table.
  + IsActive (BOOLEAN) - Indicates whether the user account is active.

**7. Lookup Tables (Recommended for Data Integrity and Consistency):**

* **Purpose:** To store predefined values for fields like Status in PurchaseOrders, Reason in CustomerReturns, VendorReturns, and StockAdjustments, and Disposition in CustomerReturns and VendorReturns. This helps ensure data consistency and makes reporting easier.
* **Examples:**
  + PurchaseOrderStatus (StatusID, StatusName)
  + ReturnReasons (ReasonID, ReasonName)
  + ReturnDispositions (DispositionID, DispositionName)
  + StockAdjustmentReasons (ReasonID, ReasonName)
  + UserRoles (RoleID, RoleName)

**Important Considerations:**

* **Data Types:** Choose appropriate data types for each field based on the type of data being stored (e.g., INT for integers, DECIMAL for monetary values, VARCHAR for strings, DATETIME for dates and times).
* **Relationships:** Define foreign key relationships between tables to enforce data integrity and ensure that related data is consistent.
* **Indexing:** Create indexes on frequently queried fields to improve performance.
* **Normalization:** Ensure that the database is properly normalized to reduce data redundancy and improve data integrity.
* **Security:** Implement appropriate security measures to protect sensitive data.
* **ENUMs vs. Lookup Tables:** For fields with a limited and relatively static set of values (like Status or Disposition), consider using ENUMs (if your database system supports them) or small lookup tables. Lookup tables are generally more flexible if the values might change over time.

By implementing these additional tables, you'll have a robust database structure to support the full functionality of your Northwind Business Simulation, including purchase orders, returns, stock adjustments, and user management. Remember to adapt this structure to your specific requirements and the capabilities of your chosen database system.

Try this Agent on your favorite IDE

Northwind Business Simulation & inventory management system,also give new tables for return or adjustment feature – Requirements Document

Here's a requirements document for the Northwind Business Simulation & Inventory Management System, including new tables for a return or adjustment feature.

## 1. Introduction

This document outlines the functional and non-functional requirements for a business simulation and inventory management system based on the classic Northwind database. The system will simulate business operations, including sales, purchasing, and inventory tracking, while also introducing new features for handling returns and adjustments.

## 2. Scope

The system will manage the following core business processes:

* **Sales Order Management:** Creating, processing, and fulfilling customer orders.
* **Inventory Management:** Tracking stock levels, reordering points, and product information.
* **Purchasing:** Generating purchase orders and receiving goods from suppliers.
* **Reporting:** Generating reports on sales performance, inventory levels, and other key metrics.
* **Returns & Adjustments:** Handling returned goods from customers and making manual inventory adjustments.

## 3. Functional Requirements

### 3.1. User Management

* **Login/Logout:** Users must be able to securely log in and out of the system.
* **Role-Based Access Control:** The system must support different user roles (e.g., Administrator, Sales Representative, Warehouse Manager) with specific permissions.

### 3.2. Sales Order Management

* **Order Creation:** Users must be able to create new sales orders by selecting products and specifying quantities.
* **Order Status Tracking:** Orders must have a clear status (e.g., Pending, Shipped, Invoiced, Canceled).
* **Customer & Employee Data:** Orders must be linked to a specific customer and the employee who placed the order.

### 3.3. Inventory Management

* **Product Information:** The system must store detailed product information, including name, supplier, category, unit price, units in stock, and units on order.
* **Stock Level Updates:** The system must automatically decrement units in stock when an order is shipped and increment units in stock when goods are received from a supplier or returned.
* **Low-Stock Alerts:** The system should provide a mechanism to alert users when a product's stock level falls below a predefined reorder point.

### 3.4. Purchasing & Suppliers

* **Supplier Management:** The system must store information about suppliers.
* **Purchase Order Generation:** Users must be able to create and manage purchase orders to replenish stock.
* **Goods Receipt:** The system must allow users to record the receipt of goods from a supplier, updating inventory levels.

### 3.5. Returns & Adjustments

* **Return Authorization:** Users must be able to initiate a return request for a specific sales order.
* **Return Reasons:** The system should allow users to specify a reason for the return (e.g., "damaged item," "wrong size," "customer changed mind").
* **Inventory Adjustment:** Users must be able to manually adjust inventory levels for reasons other than a customer return (e.g., "stock count," "damaged in warehouse").
* **Historical Tracking:** The system must log all returns and adjustments for auditing purposes.

### 3.6. Reporting

* **Sales Reports:** The system must generate reports on sales by product, customer, and employee.
* **Inventory Reports:** The system must provide reports on current stock levels, low-stock items, and inventory valuation.
* **Return & Adjustment Reports:** The system must generate reports detailing all returns and inventory adjustments.

## 4. Database Schema

The system will use a database schema based on the classic Northwind model, with the addition of new tables to support the **Returns & Adjustments** feature.

### 4.1. Existing Tables (Northwind)

* **Categories:** Stores product categories.
* **Customers:** Stores customer information.
* **Employees:** Stores employee information.
* **Shippers:** Stores shipping company information.
* **Suppliers:** Stores supplier information.
* **Products:** Stores product details, including UnitsInStock and UnitsOnOrder.
* **Orders:** Stores main sales order information, linked to customers and employees.
* **Order Details:** Stores line items for each order, linking to products and specifying quantities.

### 4.2. New Tables for Returns & Adjustments

#### Returns Table

This table will store information about returned items from customers.

* **ReturnID** (Primary Key, Auto-increment): Unique identifier for each return.
* **OrderID** (Foreign Key): Links the return to the original sales order.
* **ReturnDate** (Date): The date the return was processed.
* **Reason** (String): A short description of the return reason.
* **ProcessedByEmployeeID** (Foreign Key): Links to the employee who processed the return.

#### ReturnDetails Table

This table will store the specific products and quantities for each return.

* **ReturnDetailID** (Primary Key, Auto-increment): Unique identifier for each return line item.
* **ReturnID** (Foreign Key): Links to the Returns table.
* **ProductID** (Foreign Key): Links to the Products table.
* **Quantity** (Integer): The number of units of the product that were returned.

#### InventoryAdjustments Table

This table will track all manual inventory changes.

* **AdjustmentID** (Primary Key, Auto-increment): Unique identifier for each adjustment.
* **ProductID** (Foreign Key): The product being adjusted.
* **AdjustmentDate** (Date): The date the adjustment was made.
* **QuantityChanged** (Integer): The amount by which the inventory was changed. A **positive** value indicates an increase (e.g., found new stock), and a **negative** value indicates a decrease (e.g., spoiled goods).
* **Reason** (String): A description of why the adjustment was made (e.g., "damaged during transfer," "physical count discrepancy").
* **AdjustedByEmployeeID** (Foreign Key): Links to the employee who made the adjustment.

## 5. Non-Functional Requirements

* **Performance:** The system must be responsive, with sales order and inventory queries completing in less than 2 seconds.
* **Security:** All user data and sensitive business information must be encrypted and protected. Access must be restricted based on user roles.
* **Usability:** The user interface should be intuitive and easy to navigate for employees with minimal training.
* **Scalability:** The system should be able to handle an increasing number of users and data records without significant performance degradation.

also add purchase feature

Here is the updated and more comprehensive requirements document for the Northwind Business Simulation & Inventory Management System, now including a detailed **Purchasing Feature**.

## 1. Introduction

This document outlines the functional and non-functional requirements for a business simulation and inventory management system based on the classic Northwind database. The system will simulate core business operations, including sales, purchasing, inventory tracking, and new features for handling returns and adjustments.

## 2. Scope

The system will manage the following key business processes:

* **Sales Order Management:** Creating, processing, and fulfilling customer orders.
* **Inventory Management:** Tracking stock levels, managing reordering points, and maintaining product information.
* **Purchasing & Supplier Management:** Generating purchase orders, receiving goods, and managing supplier relationships.
* **Returns & Adjustments:** Handling customer returns and making manual inventory adjustments.
* **Reporting:** Generating reports on sales performance, inventory levels, purchasing activities, and other key metrics.

## 3. Functional Requirements

### 3.1. User Management

* **Login/Logout:** Users must be able to securely log in and out of the system.
* **Role-Based Access Control:** The system must support different user roles (e.g., Administrator, Sales Representative, Warehouse Manager, Purchasing Agent) with specific permissions.

### 3.2. Sales Order Management

* **Order Creation:** Users must be able to create new sales orders by selecting products and specifying quantities.
* **Order Status Tracking:** Orders must have a clear status (e.g., Pending, Shipped, Invoiced, Canceled).
* **Customer & Employee Data:** Orders must be linked to a specific customer and the employee who placed the order.

### 3.3. Inventory Management

* **Product Information:** The system must store detailed product information, including name, supplier, category, unit price, units in stock, and units on order.
* **Stock Level Updates:** The system must automatically decrement units in stock when an order is shipped and increment units in stock when goods are received from a supplier or returned.
* **Low-Stock Alerts:** The system should provide a mechanism to alert users when a product's stock level falls below a predefined reorder point.

### 3.4. Purchasing & Supplier Management

* **Supplier Management:** The system must store and manage information about all suppliers, including contact details and terms.
* **Purchase Order Generation:** Users with the appropriate role (e.g., Purchasing Agent) must be able to create new purchase orders.
  + **PO Line Items:** The PO must include details for each item being ordered, such as product, quantity, and agreed-upon unit price.
  + **Supplier Selection:** The PO must be linked to a specific supplier.
* **Purchase Order Status:** Purchase orders must have a status (e.g., Draft, Sent to Supplier, Received, Canceled).
* **Goods Receipt:** The system must allow users to record the receipt of goods against a purchase order.
  + **Partial Receipt:** The system should support receiving only a portion of the ordered quantity.
  + **Automatic Inventory Update:** Upon receiving goods, the system must automatically update the UnitsInStock for the corresponding products.

### 3.5. Returns & Adjustments

* **Return Authorization:** Users must be able to initiate a return request for a specific sales order.
* **Return Reasons:** The system should allow users to specify a reason for the return (e.g., "damaged item," "wrong size," "customer changed mind").
* **Inventory Adjustment:** Users must be able to manually adjust inventory levels for reasons other than a customer return (e.g., "stock count," "damaged in warehouse").
* **Historical Tracking:** The system must log all returns and adjustments for auditing purposes.

### 3.6. Reporting

* **Sales Reports:** The system must generate reports on sales by product, customer, and employee.
* **Inventory Reports:** The system must provide reports on current stock levels, low-stock items, and inventory valuation.
* **Purchasing Reports:** The system must provide reports on purchase order history, supplier performance, and total spending by supplier.
* **Return & Adjustment Reports:** The system must generate reports detailing all returns and inventory adjustments.

## 4. Database Schema

The system will use a database schema based on the classic Northwind model, with the addition of new tables to support the **Purchasing** and **Returns & Adjustments** features.

### 4.1. Existing Tables (Northwind)

* **Categories:** Stores product categories.
* **Customers:** Stores customer information.
* **Employees:** Stores employee information.
* **Shippers:** Stores shipping company information.
* **Suppliers:** Stores supplier information.
* **Products:** Stores product details, including UnitsInStock and UnitsOnOrder.
* **Orders:** Stores main sales order information.
* **Order Details:** Stores line items for each sales order.

### 4.2. New Tables for Purchasing

#### PurchaseOrders Table

This table will store the main purchase order information.

* **PurchaseOrderID** (Primary Key, Auto-increment): Unique identifier for each purchase order.
* **SupplierID** (Foreign Key): Links the PO to a specific supplier.
* **OrderDate** (Date): The date the purchase order was created.
* **RequiredDate** (Date): The requested delivery date.
* **ShippedDate** (Date): The actual date the goods were received.
* **Status** (String): The current status of the purchase order (e.g., 'Draft', 'Submitted', 'Received').
* **PlacedByEmployeeID** (Foreign Key): Links to the employee who created the purchase order.

#### PurchaseOrderDetails Table

This table will store the line items for each purchase order.

* **PurchaseOrderDetailID** (Primary Key, Auto-increment): Unique identifier for each line item.
* **PurchaseOrderID** (Foreign Key): Links to the PurchaseOrders table.
* **ProductID** (Foreign Key): Links to the Products table.
* **QuantityOrdered** (Integer): The number of units ordered.
* **QuantityReceived** (Integer): The number of units actually received.
* **UnitPrice** (Decimal): The agreed-upon price per unit.

### 4.3. New Tables for Returns & Adjustments

#### Returns Table

This table will store information about returned items from customers.

* **ReturnID** (Primary Key, Auto-increment): Unique identifier for each return.
* **OrderID** (Foreign Key): Links the return to the original sales order.
* **ReturnDate** (Date): The date the return was processed.
* **Reason** (String): A short description of the return reason.
* **ProcessedByEmployeeID** (Foreign Key): Links to the employee who processed the return.

#### ReturnDetails Table

This table will store the specific products and quantities for each return.

* **ReturnDetailID** (Primary Key, Auto-increment): Unique identifier for each return line item.
* **ReturnID** (Foreign Key): Links to the Returns table.
* **ProductID** (Foreign Key): Links to the Products table.
* **Quantity** (Integer): The number of units of the product that were returned.

#### InventoryAdjustments Table

This table will track all manual inventory changes.

* **AdjustmentID** (Primary Key, Auto-increment): Unique identifier for each adjustment.
* **ProductID** (Foreign Key): The product being adjusted.
* **AdjustmentDate** (Date): The date the adjustment was made.
* **QuantityChanged** (Integer): The amount by which the inventory was changed. A **positive** value indicates an increase, and a **negative** value indicates a decrease.
* **Reason** (String): A description of why the adjustment was made.
* **AdjustedByEmployeeID** (Foreign Key): Links to the employee who made the adjustment.

## 5. Non-Functional Requirements

* **Performance:** The system must be responsive, with sales order, purchasing, and inventory queries completing in less than 2 seconds.
* **Security:** All user data and sensitive business information must be encrypted and protected. Access must be restricted based on user roles.
* **Usability:** The user interface should be intuitive and easy to navigate for employees with minimal training.
* **Scalability:** The system should be able to handle an increasing number of users and data records without significant performance degradation.

can you give me a complete UI or web pages list

Here is a complete list of UI pages for the Northwind Business Simulation & Inventory Management System, categorized by function.

### 1. User Authentication & Home Pages

* **Login Page:** Form for users to enter credentials to access the system.
* **Dashboard/Home Page:** An overview page with key metrics like recent sales, low-stock items, and recent purchase orders. It provides a quick snapshot of the business.

### 2. Inventory & Product Management

* **Product List:** A table displaying all products with details like name, category, supplier, units in stock, and unit price. It should have search and filter functionality.
* **Product Detail Page:** A detailed view for a single product, showing all information and providing options to edit it. This page also shows recent stock movements (e.g., sales, returns, purchases) for that product.
* **New Product Form:** A form to add a new product to the inventory.

### 3. Sales Order Management

* **Sales Orders List:** A table showing all customer orders with columns for order ID, customer name, order date, and status. It should be sortable and searchable.
* **New Sales Order Form:** A multi-step form to create a new order by selecting a customer, adding products, and specifying quantities.
* **Sales Order Detail Page:** A page showing the complete details of a single order, including all line items, customer information, shipping details, and a history of status changes. It should have options to "Ship Order" or "Cancel Order."

### 4. Purchasing & Supplier Management

* **Suppliers List:** A list of all registered suppliers with their contact information.
* **Supplier Detail Page:** A page with detailed supplier information and a history of all purchase orders made to that supplier.
* **Purchase Orders List:** A table showing all purchase orders with columns for PO number, supplier, order date, and status.
* **New Purchase Order Form:** A form to create a new purchase order by selecting a supplier and adding products and quantities.
* **Purchase Order Detail Page:** A detailed view of a single purchase order. It should show line items and a button to **"Receive Goods"** which would trigger the inventory update.

### 5. Returns & Inventory Adjustments

* **Returns List:** A list of all processed customer returns with details like return ID, original order, and return date.
* **New Return Form:** A form to process a new return against an existing sales order. Users would select the items and quantities being returned and specify a reason.
* **Inventory Adjustment Form:** A form to manually adjust inventory levels for a specific product. Users would select the product, enter the quantity to add or remove, and provide a reason (e.g., damaged, lost).
* **Adjustment History:** A log page showing all past inventory adjustments.

### 6. Reporting

* **Sales Reports:** A page with interactive charts and tables for sales data. Options should include "Sales by Product," "Sales by Employee," and "Sales by Customer."
* **Inventory Reports:** A page with reports on current inventory, including a list of **low-stock items** and an **inventory valuation report**.
* **Purchasing Reports:** A page showing total spending by supplier and a list of all purchase orders.

