### **Supply Chain Traceability & Inventory Management System**

**Problem:** A food or retail company needs to track products from their origin (farm or factory) to the final customer to ensure quality and address recalls quickly. Current systems lack a unified view of the entire supply chain.

**Salesforce Solution:** Create a system that provides end-to-end visibility of a product's journey.

### **Phase 1: Problem Understanding & Industry Analysis**

* **Requirement Gathering:** The core needs are to track products, manage inventory, and provide a quick way to trace a product's history. The system must be able to handle a high volume of data and provide a unified view for both inventory managers and quality assurance teams.
* **Business Process Mapping:** The process begins with a product being assigned to a Lot at the factory. The lot is then included in a Shipment to a Retail Location. At the store, the product is sold to a customer. The system needs to track each step and link the data, allowing for backward and forward traceability.
* **Industry-specific Use Case Analysis:** A key use case is a product recall. If a product is found to be defective, the company needs to quickly identify which specific lots and shipments were affected, and which retail locations received them. This project will demonstrate the ability to handle this critical scenario efficiently.

### **Phase 2: Org Setup & Configuration**

* **Dev Org Setup:** A developer org will be used for building this project.
* **User Setup & Profiles:** Create profiles for Inventory Manager and Quality Assurance Specialist. Inventory Managers will have access to create and manage Shipments and Retail Locations, while Quality Assurance Specialists will use the system primarily for traceability and recalls.
* **Security:** Configure roles and sharing rules to ensure that a manager for one region's retail locations cannot see the data for another region's locations.

### **Phase 3: Data Modeling & Relationships**

* **Custom Objects:** Create the following custom objects to represent the supply chain:
  + Product\_\_c: Represents the type of product (e.g., "Organic Apples"). This will have fields like Product Name and SKU.
  + Lot\_\_c: Represents a batch of products from a single manufacturing run. This object will have a master-detail relationship to Product\_\_c and a Quantity field.
  + Shipment\_\_c: Tracks a single delivery of products. This will have a master-detail relationship to Lot\_\_c and a Destination field.
  + Retail\_Location\_\_c: Represents a final point of sale, with fields like Store Name and Address.
* **Relationships:**
  + Create a lookup relationship from Shipment\_\_c to Retail\_Location\_\_c to link a shipment to its destination.
  + Use a junction object to link Shipment\_\_c to Retail\_Location\_\_c for shipments with multiple destinations, if required.
* **Fields:** Add a Barcode\_\_c text field to Lot\_\_c to store a unique barcode identifier. Add a Current\_Inventory\_\_c number field to Product\_\_c or Retail\_Location\_\_c to track stock.

### **Phase 4: Process Automation (Admin)**

* **Validation Rules:** Implement a validation rule on the Shipment\_\_c object to prevent the quantity of a shipment from exceeding the available quantity in the Lot\_\_c.
* **Flow Builder:** Create a **Record-Triggered Flow** that fires when a new Shipment\_\_c record is created. The flow will update the Current\_Inventory\_\_c field on the corresponding Retail\_Location\_\_c by adding the shipment's quantity.

### **Phase 5: Apex Programming (Developer)**

* **Apex Trigger:** Write a **before insert/update Apex Trigger** on the Shipment\_\_c object. This trigger will automatically **decrement the inventory count** on the related Lot\_\_c record to reflect the shipped quantity.
* **Scheduled Apex:** Create a **Scheduled Apex** class that runs every night at a specific time. This class will query all Product\_\_c or Retail\_Location\_\_c records where the Current\_Inventory\_\_c field is below a predefined threshold. It will then create an Email Alert or Custom Notification to the Inventory Manager to reorder the product.
* **Apex Class (LWC Controller):** Write an Apex class with a method that takes a barcode string as input. The method will perform a SOQL query to find the Lot\_\_c record and then traverse the relationships to retrieve all associated Shipment\_\_c and Retail\_Location\_\_c records.

### **Phase 6: User Interface Development**

* **Lightning App Builder:** Use the **Lightning App Builder** to create a custom app with tabs for each of the main objects. Design custom record pages to provide a clear view of related records (e.g., showing all Shipments for a specific Lot).
* **Lightning Web Component (LWC):**
  + **Traceability Scanner LWC:** Build a **Lightning Web Component** that allows a user to input or scan a barcode. The component will call the Apex controller method to retrieve the product's full history. It will then display this history in a clear, easy-to-read format on the page, showing the product's origin, all shipments, and its current location.
  + **Inventory Dashboard LWC:** Create a component on the app's home page that shows a summary of low-inventory products across all retail locations.

### **Phase 7: Integration & External Access**

* **API:** While not mandatory, this project can be extended to include API integration. For example, a third-party app for barcode scanning could be integrated with Salesforce's REST API to create a Shipment\_\_c record automatically.

### **Phase 8: Data Management & Deployment**

* **Data Loader:** Use the **Data Loader** to import the initial product, lot, and retail location data into the system. This allows for a realistic demonstration with pre-existing data.
* **Change Sets:** Package all the custom components, including the custom objects, fields, Apex, and LWC, into a change set for deployment.
* **VS Code & SFDX:** Use these tools for a streamlined development process, especially for writing the Apex code and the LWC.

### **Phase 9: Reporting, Dashboards & Security Review**

* **Reports:** Create a Summary Report to track product shipments by Retail Location. A Matrix Report can be used to analyze inventory levels by product and store.
* **Dashboards:** Build a **Dynamic Dashboard** for managers showing real-time inventory levels, low-stock alerts, and shipment status.
* **Field Level Security:** Review and configure field-level security to ensure sensitive data, such as a product's manufacturing origin, is only visible to authorized personnel.

### **Phase 10: Final Presentation & Demo Day**

* **Demo Walkthrough:** Prepare a demonstration that shows the entire process, from a Shipment being created to the Apex Trigger updating the Lot quantity. Then, use the LWC to scan a barcode and show the full traceability history.
* **Handoff Documentation:** Provide documentation that explains the data model and the purpose of the Apex code and LWC, as well as a user guide for how to perform a product traceability search.