**SMART SUPPLY CHAIN & INVENTORY TRACKING SYSTEM**

**Capstone Project Report**

Submitted By: M. Sampath Kumar

Under the Guidance of: Nadeem Sir

Abstract

The Smart Supply Chain & Inventory Tracking System is a Salesforce-based solution designed to streamline inventory, order, and shipment management for businesses. Traditional supply chain operations face challenges such as stockouts, overstocking, delayed shipments, and lack of real-time visibility, which lead to inefficiencies and customer dissatisfaction.

This project leverages Salesforce CRM capabilities (Admin + Developer) to build a centralized system where:

Products, warehouses, and inventory are tracked in real time.

Orders automatically reduce inventory, with low-stock alerts triggering reorder requests.

Suppliers are notified instantly, and delivery agents can update shipment statuses.

Dashboards and reports provide warehouse managers with complete visibility into operations.

Automation, Apex triggers, Lightning Web Components (LWC), and integrations ensure smooth workflows and secure role-based access.

The solution is lightweight, customizable, and scalable—making it ideal for small to medium businesses seeking a smart alternative to costly ERP systems.

Phase 1: Problem Understanding & Industry Analysis

Companies struggle with stockouts, overstocking, shipment delays, and no single source of truth.

Our solution: A Salesforce app to manage Products, Warehouses, Inventory, Orders, Shipments, with automation and dashboards.

**Stakeholders (Roles):**

**Admin:**

**Purpose:** Full access and system ownership.

**Responsibilities:**

* Configure Salesforce org (setup, profiles, OWD, sharing).
* Create & manage users, roles, and permissions.
* Approve escalated Reorder Requests.
* Oversee all warehouses, orders, and shipments.
* Generate dashboards & reports for management.

**Warehouse Manager:**

**Purpose:** Manages inventory and orders in their assigned warehouse.

**Responsibilities:**

* Monitor **Inventory** stock levels.
* Approve **Reorder Requests** (up to certain quantity).
* Manage **Orders** (assign to Delivery Agents, track processing).
* Generate reports for warehouse performance.

**Delivery Agent:**

**Purpose:** Handles shipment and delivery of customer orders.

**Responsibilities:**

* View **Orders** assigned to them.
* Update **Shipment** records (status from “In Transit” → “Delivered”).
* Report delays (triggers escalation to Admin).
* Cannot modify inventory or create orders.

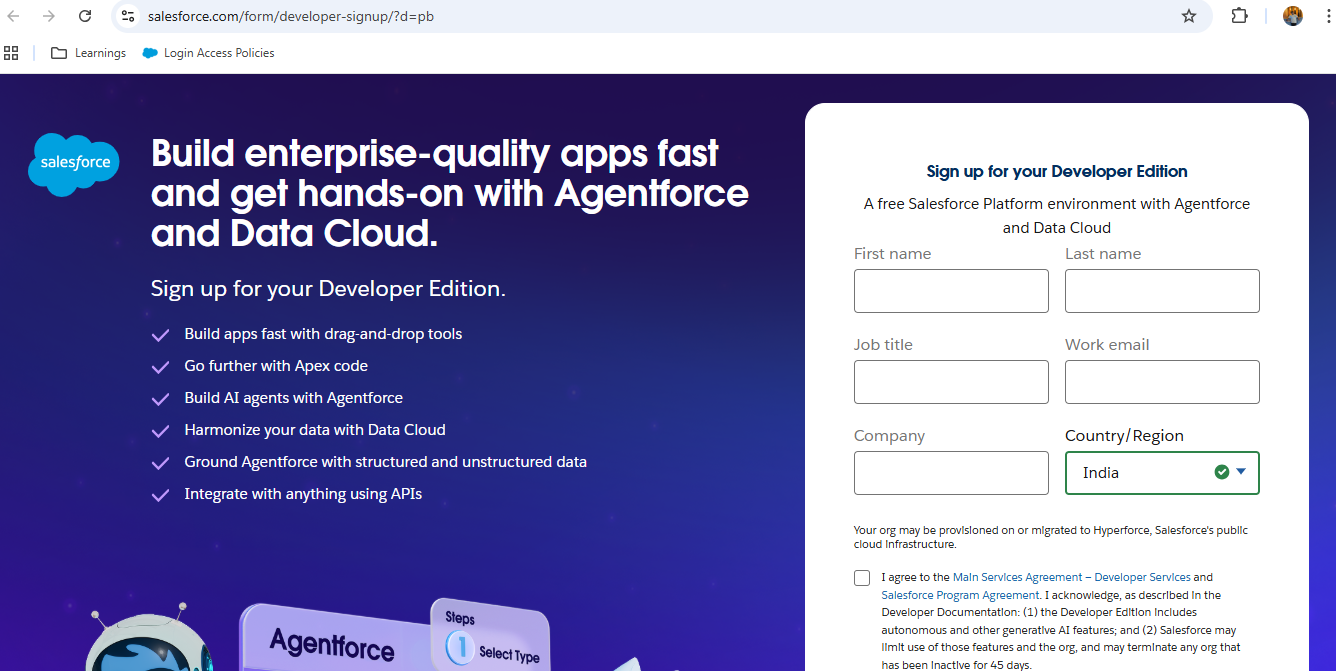
**Supplier** → receives reorder requests.

**Customer** → receives orders.

Phase 2: Org Setup & Configuration

**Step 1: Get Salesforce Environment**

* Created a **free Developer Org**:  
   developer.salesforce.com/signup



**Step 2: Company Settings**

* Setup → **Company Information**:
  + Company Name: *Smart Supply Chain Inc.*
  + Fiscal Year: Standard Gregorian.
  + Business Hours: 9 AM – 6 PM, Mon–Sat.

**Step 3: Users & Roles**

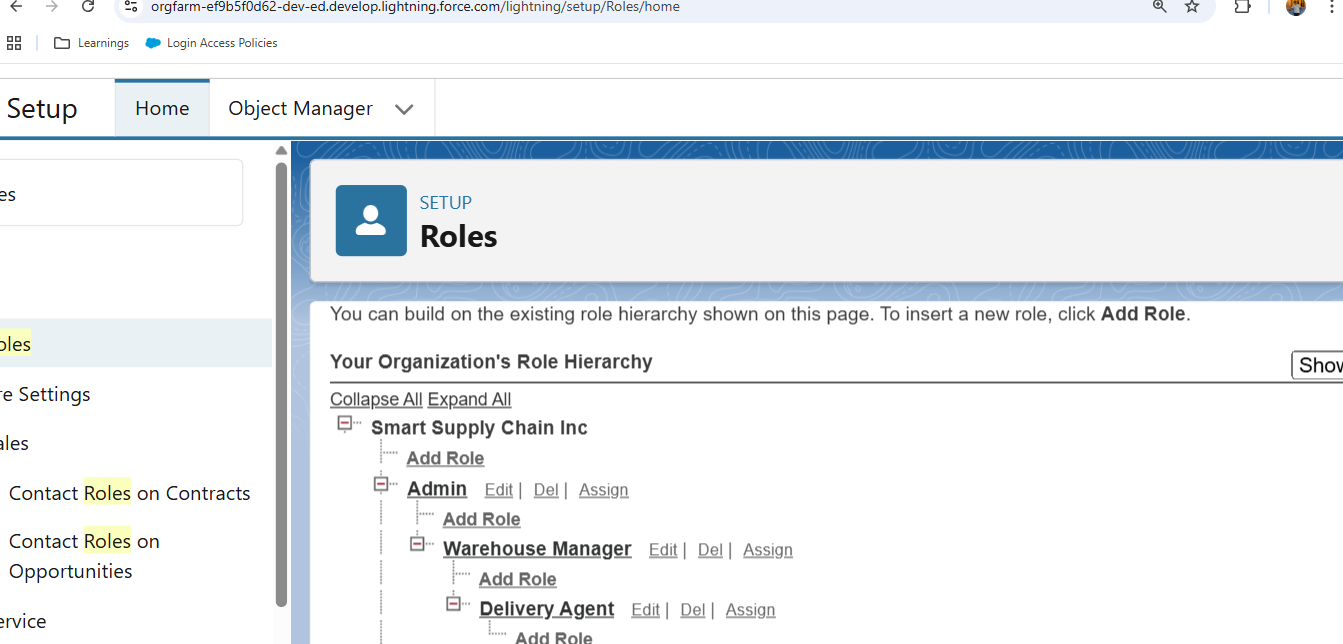
Created the role hierarchy:

Admin

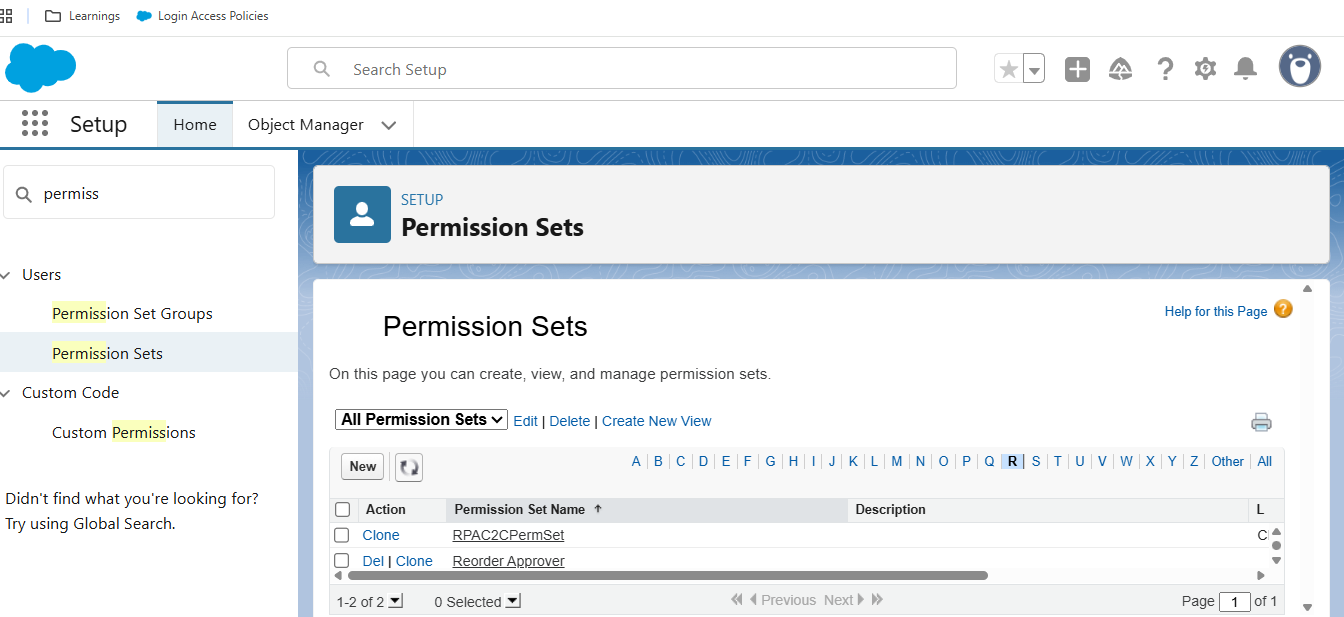
└── Warehouse Manager

└── Delivery Agent

* **Profiles:**
  + Admin → System Administrator
  + Warehouse Manager → Custom Profile (Modify All on Products/Warehouses, Read-Only Orders)
  + Delivery Agent → Custom Profile (Read Orders, Update Shipments only)
  + Supplier → (optional, read-only access via Experience Cloud)



* **Permission Sets:**
  + “Reorder Approver” → Approve large purchase orders.
  + “Shipment Tracker” → Can view Shipment dashboards.

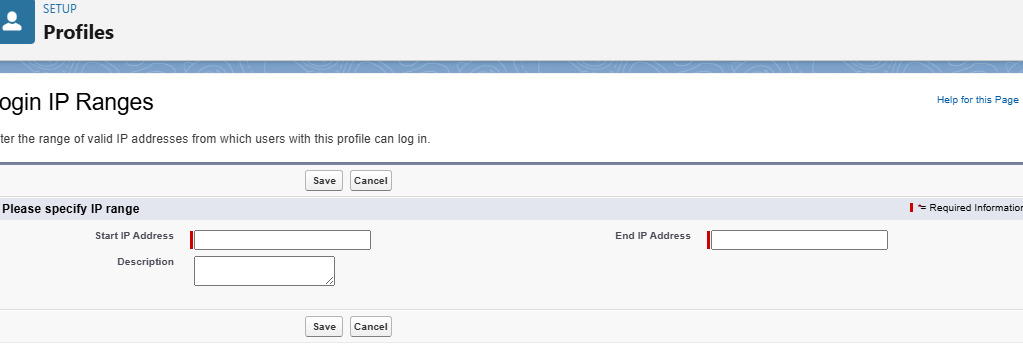


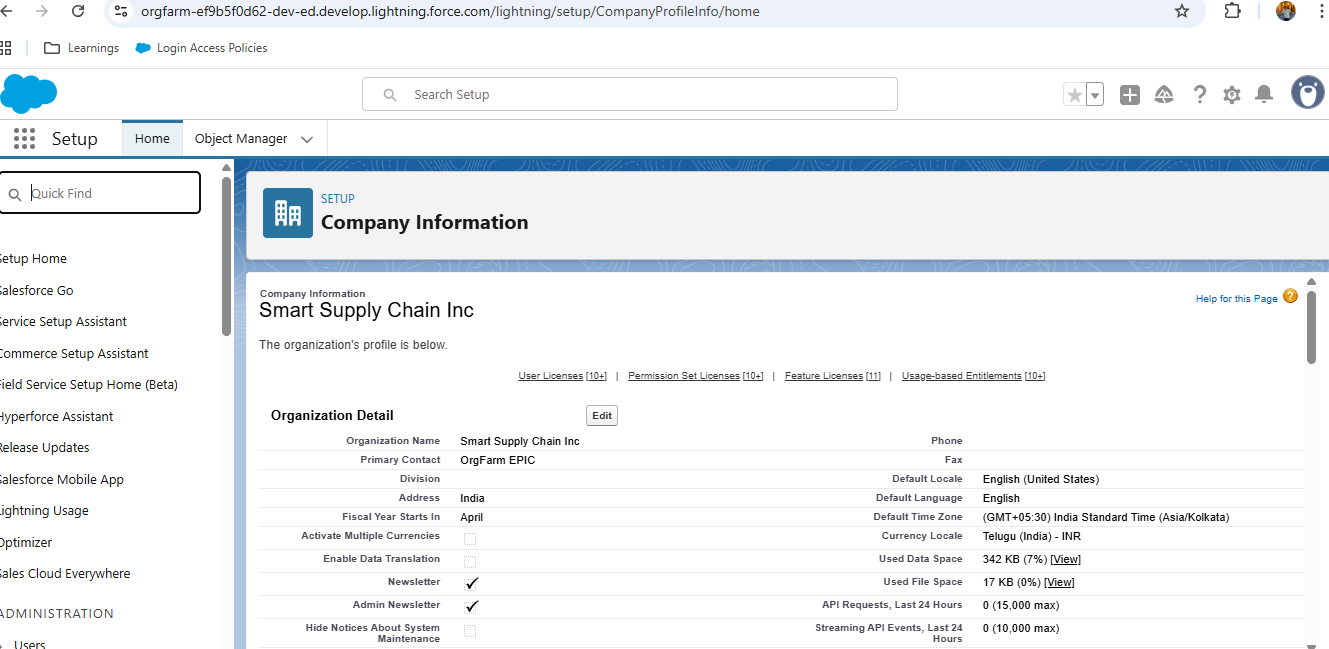
**Step 4: Security Settings**

* **OWD (Organization-Wide Defaults):**
  + Products: Public Read/Write.
  + Warehouses: Public Read/Write.
  + Orders: Private (only owner + shared users).
  + Shipments: Private.
* **Sharing Rules:**
  + Share Orders with Warehouse Manager role.
  + Share Shipments with Delivery Agent role.

**Step 5: Login & Access**

* Set **Password Policies**: Min length 8, expire 90 days.
* Set **Login Hours** for Delivery Agents: 7 AM – 10 PM only.





Phase 3: Data Modeling & Relationships

**Step 1: Identifying Key Objects**

We need both **Standard Objects** and **Custom Objects**.

**Standard Objects we’ll use**

* **Account** → to represent **Suppliers** & **Customers**
* **User** → Admin, Warehouse Manager, Delivery Agent

**Custom Objects created:**

1. **Product**→ to store product details
2. **Warehouse** → to store warehouse locations
3. **Inventory** → to track stock levels
4. **Order** → to represent customer orders
5. **Shipment** → to track deliveries
6. **Reorder Request** → (for suppliers when stock is low)

**Step 2: Create Custom Objects**

**Path:** **Setup → Object Manager → Create → Custom Object → New Custom Object**

**For each:**

**Product**

* Label: Product
* Plural: Products
* Record Name: Auto Number (e.g., PRD- {0000})
* Allow Reports, Track Field History

**Warehouse**

* Label: Warehouse
* Plural: Warehouses
* Record Name: Auto Number (WH- {000})

**Inventory**

* Label: Inventory
* Record Name: Auto Number (INV- {0000})

**Order**

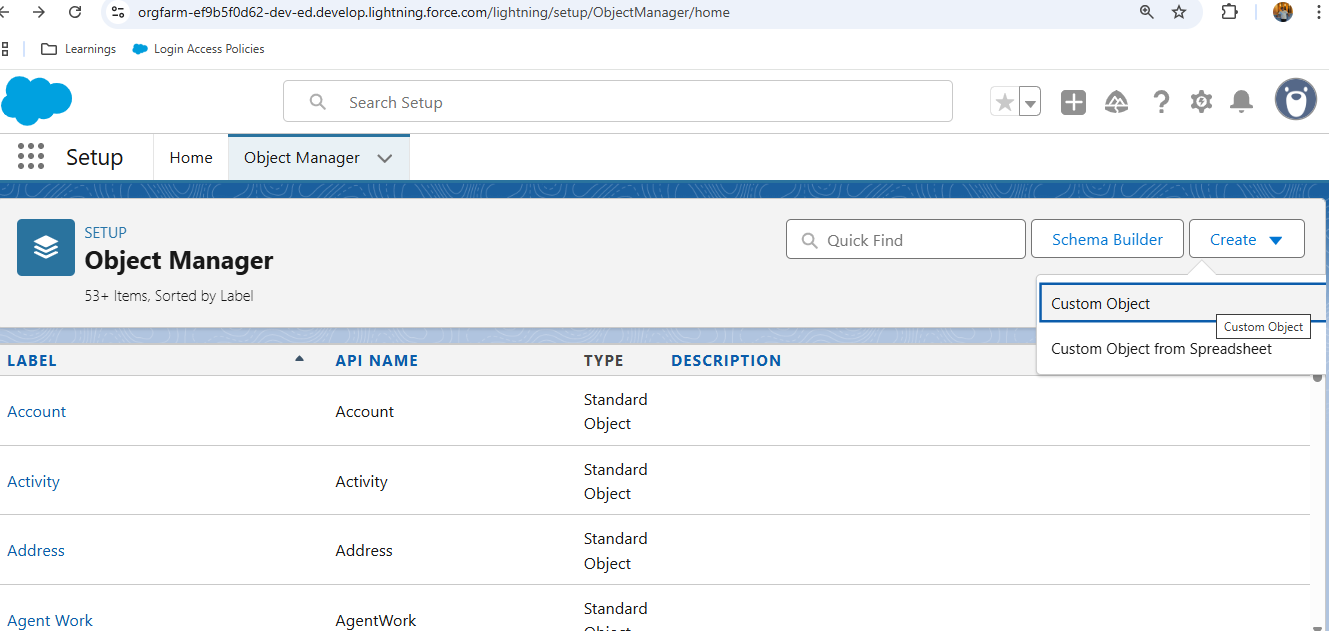
* Label: Order
* Record Name: Auto Number (ORD- {0000})

**Shipment**

* Label: Shipment
* Record Name: Auto Number (SHP- {0000})

**Reorder Request**

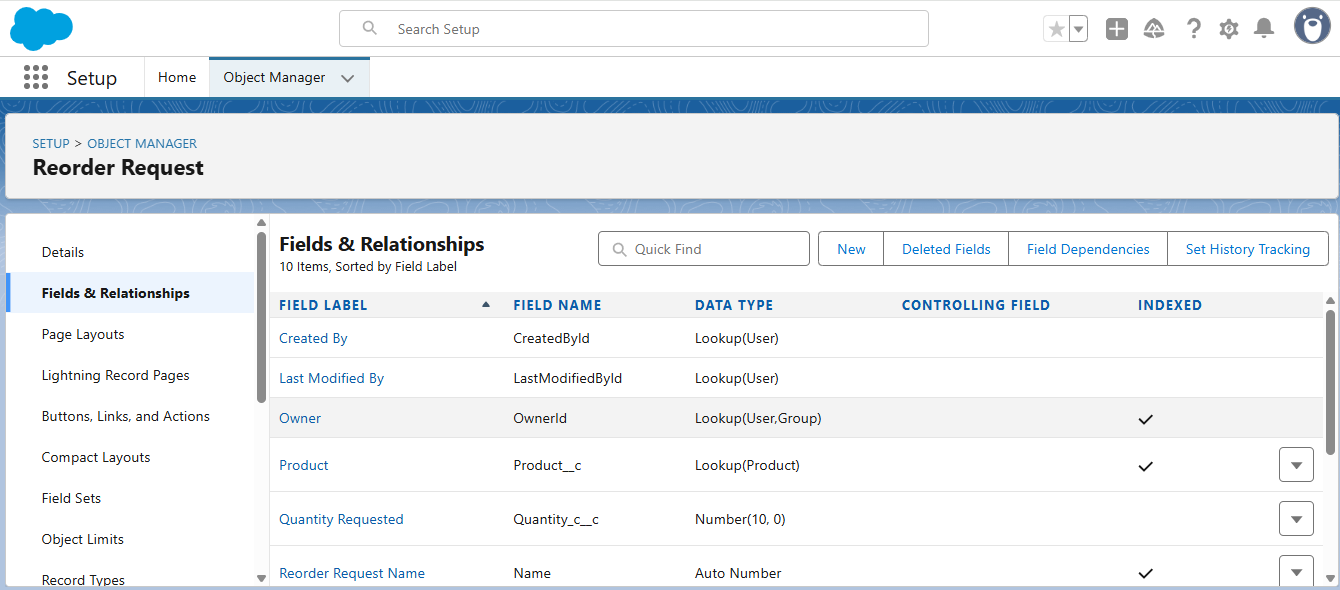
* Label: Reorder Request
* Record Name: Auto Number (REQ- {0000})



**Step 3: Create Relationships**

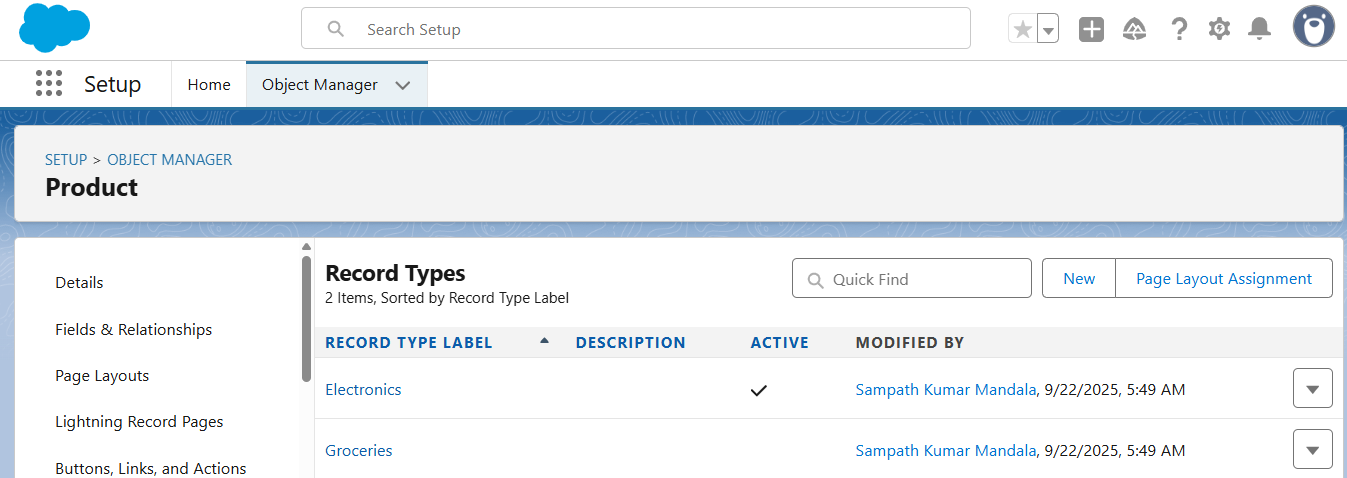
Go to **Object Manager → [each object] → Fields & Relationships → New**

1. **Product→ Inventory**
   * Relationship: **Master-Detail** (Product → Inventory)
   * Each inventory record is linked to a single product.
2. **Warehouse→ Inventory**
   * Relationship: **Lookup** (Warehouse → Inventory)
   * Inventory belongs to a warehouse, but warehouse can exist without inventory.
3. **Order→ Product**
   * Relationship: **Lookup** (Order → Product).
   * (Or use a **Junction Object** Order Item\_ c if you want many products per order).
4. **Order → Account**
   * Relationship: **Lookup** (Customer placing the order).
5. **Shipment → Order**
   * Relationship: **Master-Detail** (Shipment tied to one Order).
6. **Reorder Request → Supplier (Account)**
   * Relationship: **Lookup** (Reorder is raised against Supplier).



**Step 4: Record Types**

* **Order**
  + Record Type 1: Online Order
  + Record Type 2: Bulk Order
* **Product**
  + Record Type 1: Electronics
  + Record Type 2: Groceries

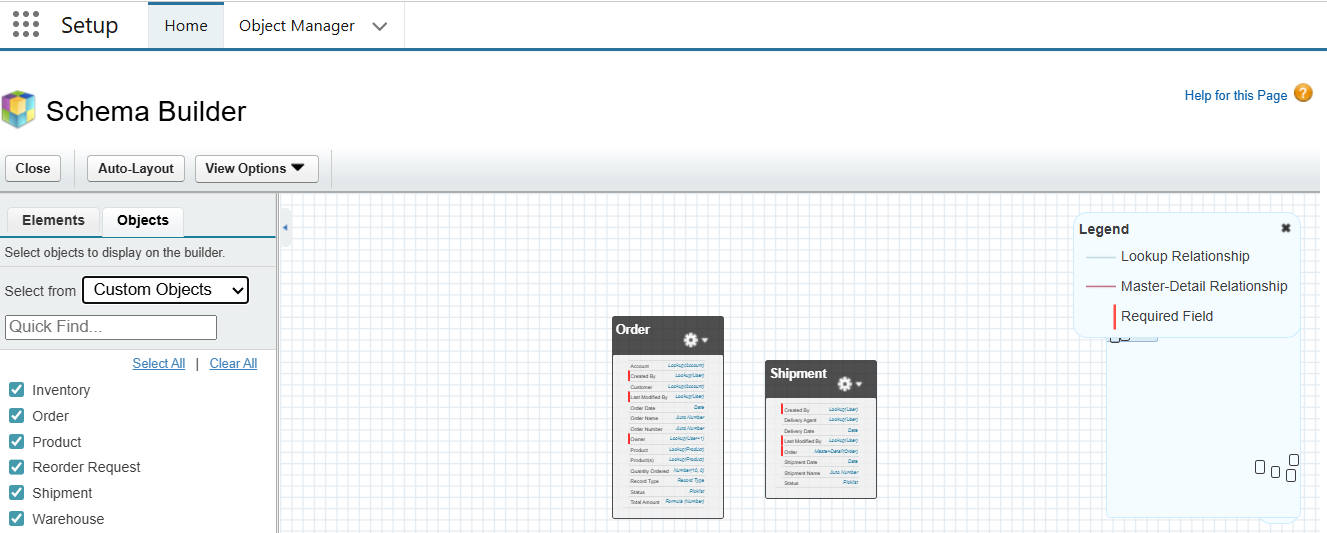


**Step 5: Page Layouts & Compact Layouts**

* Customize **Page Layouts** for each object to show the most useful fields .
* Create **Compact Layouts** for mobile view.

**Step 6: Schema Builder**

1. Setup → Schema Builder.
2. Drag **Product, Warehouse, Inventory, Order, Shipment, Reorder Request** into the canvas.



**Over all of phase:**

Warehouse (Custom) ──< Inventory (Custom) >── Product (Custom)

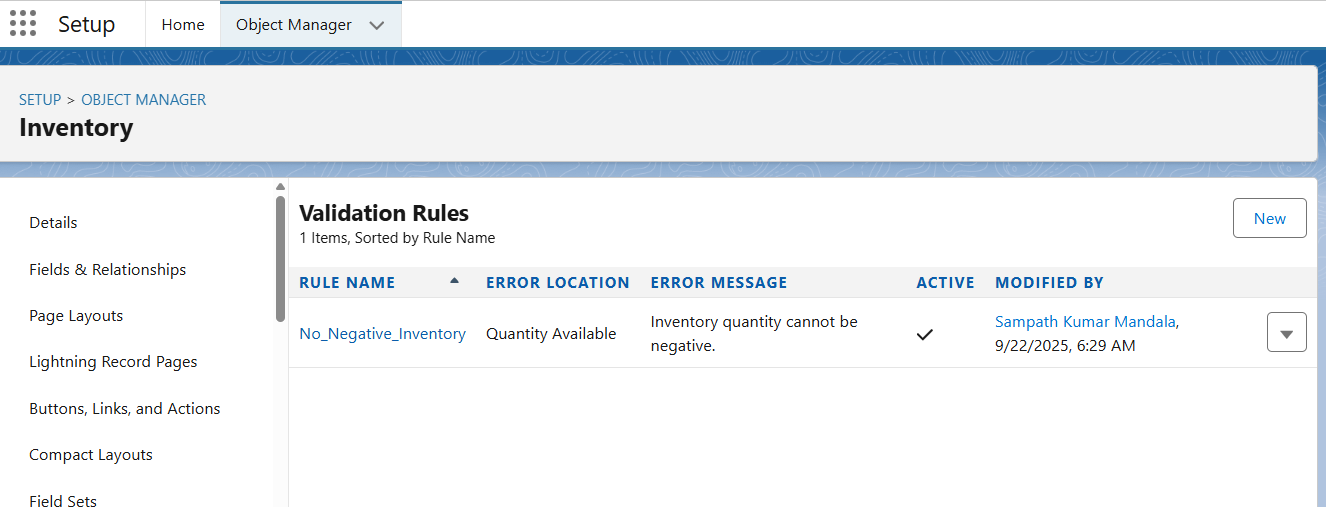
Account (Standard: Customer/Supplier) ──< Order (Custom) >── Shipment (Custom)

└──< Reorder Request (Custom)

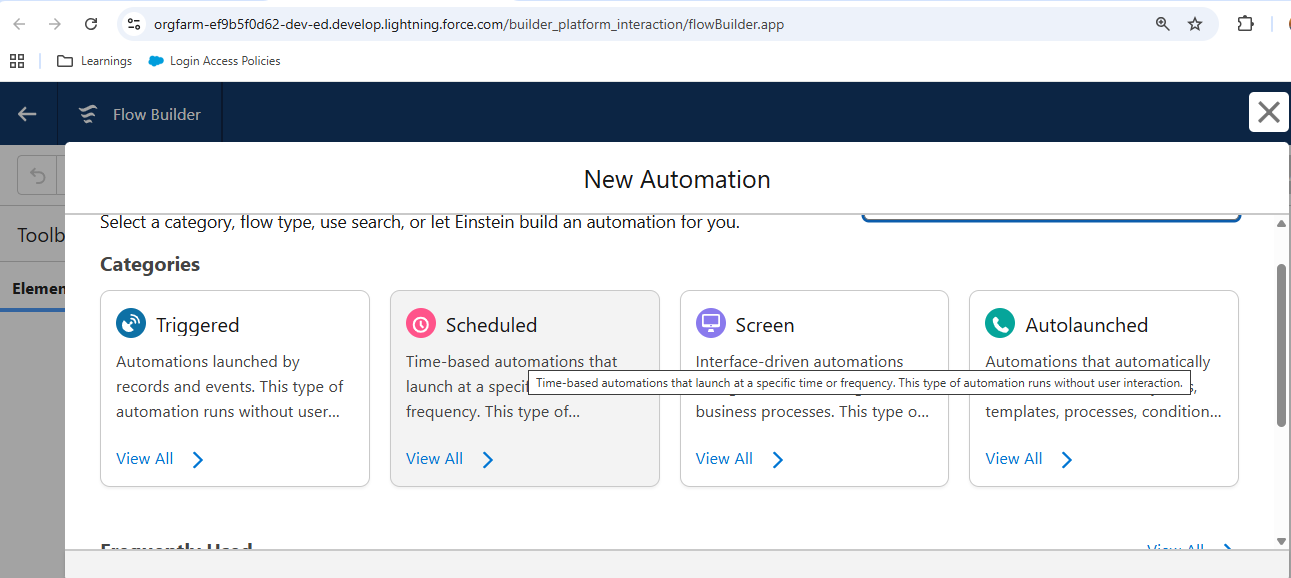
Phase 4: Process Automation

**Step 1: Validation Rules**

1. **Inventory cannot go negative**
   * Object: Inventory
   * Rule: Quantity < 0
   * Error Message: "Inventory quantity cannot be negative."
2. **Reorder quantity must be greater than zero**
   * Object: Reorder Request
   * Rule: Quantity <= 0
   * Error Message: "Reorder quantity must be positive."



**Step 2: Flows**

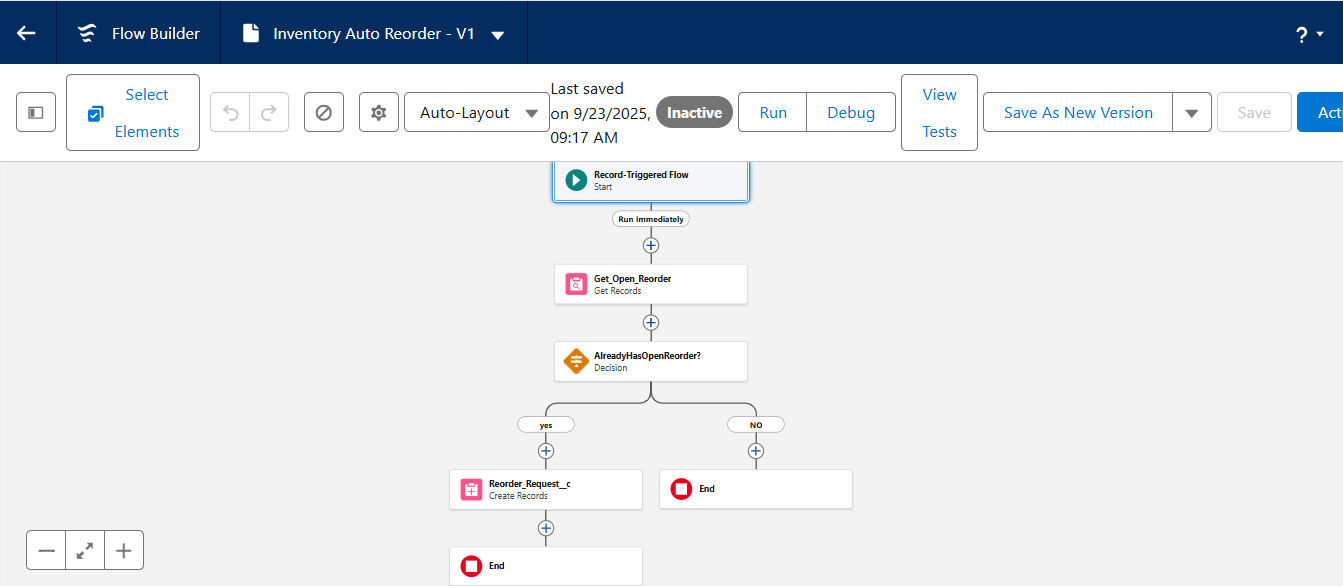
****

1. **Auto Reorder Trigger**
   * Object: Inventory
   * Flow Type: Record-Triggered Flow
   * When: After Save
   * Condition: Quantity <= Reorder Level
   * Action: Create a new Reorder Request record.
     + Fill in Product, Warehouse, Quantity Needed.
2. **Auto Update Order Status**
   * Object: Shipment
   * Flow Type: Record-Triggered Flow
   * When: After Save
   * Condition: Status = 'Delivered'
   * Action: Update related Order → Status = 'Delivered'.

**Step 3: Approval Process**

For **Reorder Requests**:

* Object: Reorder Request
* Entry Criteria: Quantity> 500
* Approver: Warehouse Manager → Admin
* Action: If Approved → Status = Approved. If Rejected → Status = Rejected.



**Step 4: Assignment Rules**

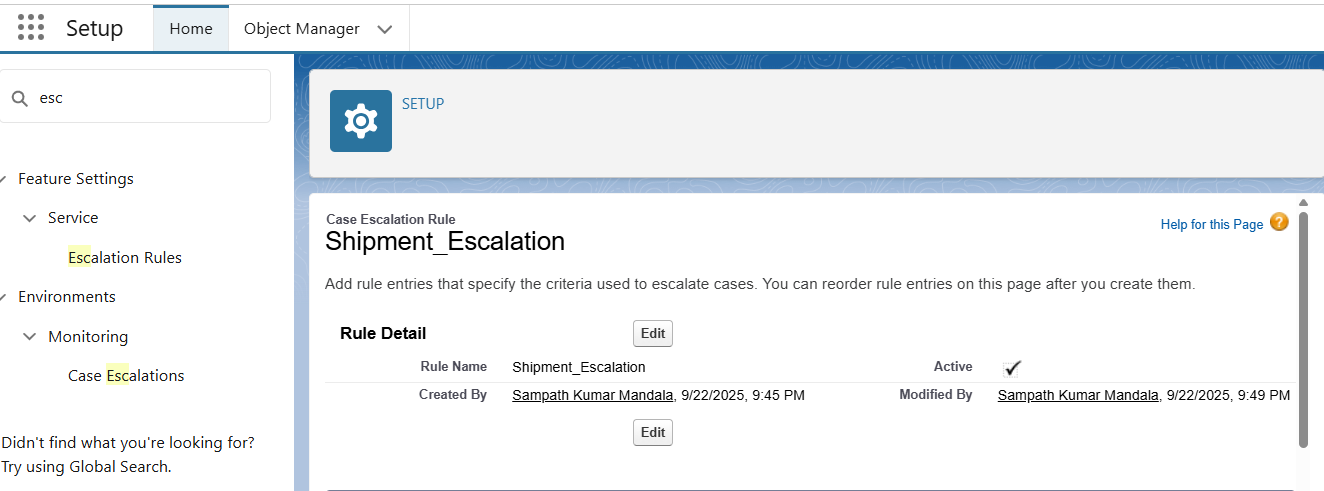
For **Orders**:

* Setup → Assignment Rules → Orders
* Rule: If Product ->Category = 'Groceries', assigned to **Warehouse Manager 1**.
* If Product->Category = 'Electronics', assigned to **Warehouse Manager 2**.

**Step 5: Escalation Rule**

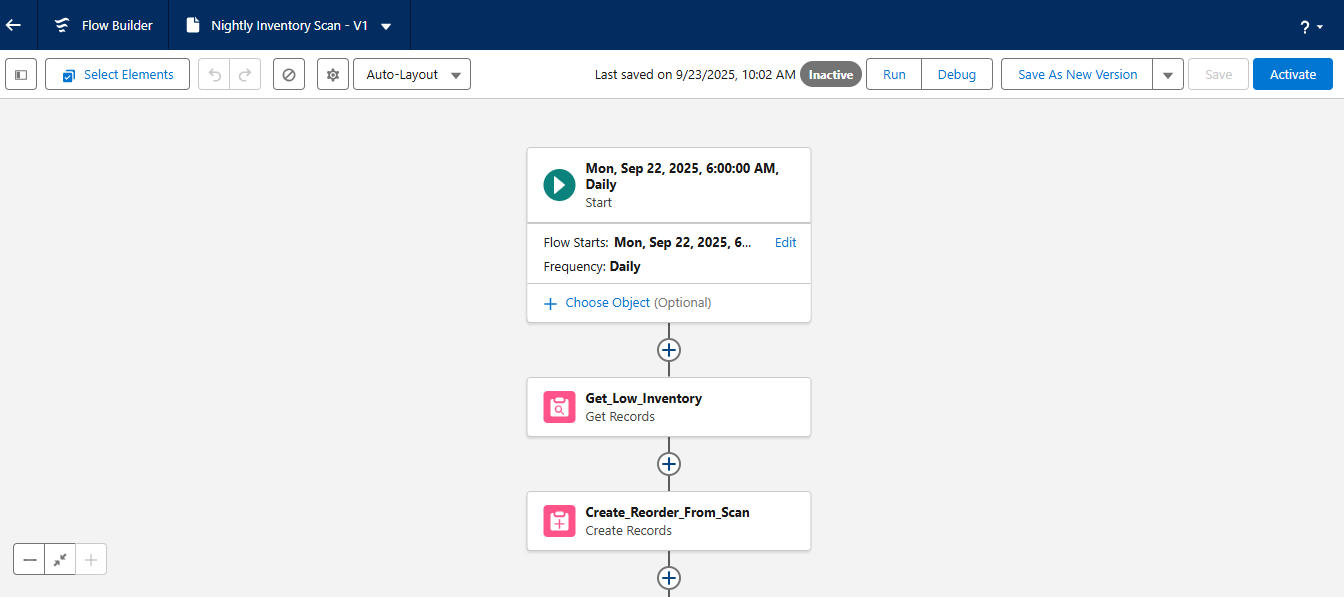
For **Shipments**:

* If Status = In Transit for more than **7 days**, escalate case to Admin.



**Step 6: Scheduled Flow**

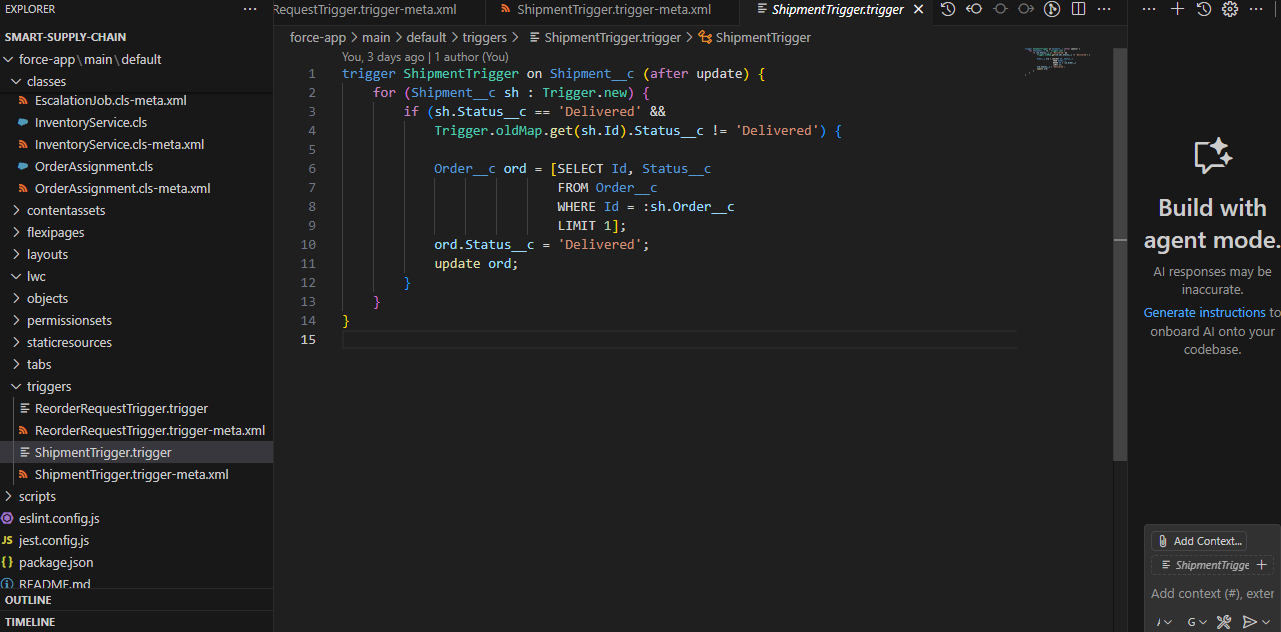
* Object: Inventory
* Run daily at 6 AM.
* Check all Inventory where Quantity < Reorder Level.
* If found, send email alert to Warehouse Manager.

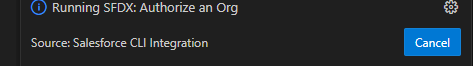


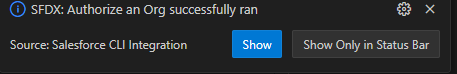
Phase 5: Apex Programming

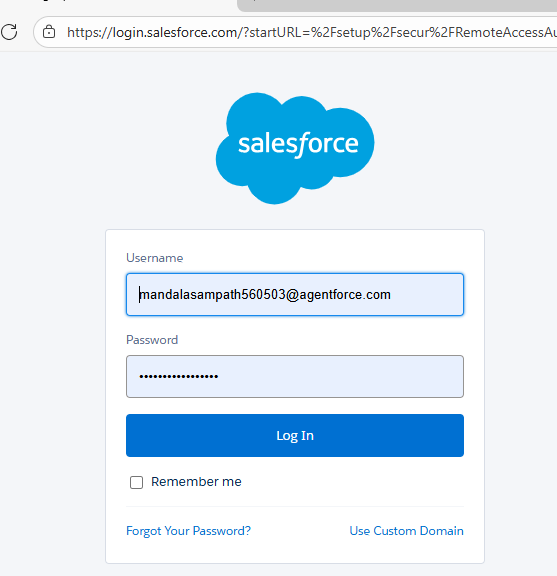
| **Apex File** | **Type** | **Purpose** |
| --- | --- | --- |
| **Inventory Service.cls** | Class | Handles inventory updates & auto reorders |
| **ReorderRequestTrigger.trigger** | Trigger | Creates reorder requests when stock is low |
| **ShipmentTrigger.trigger** | Trigger | Updates order status when shipment delivered |
| **OrderAssignment.cls** | Class | Assigns orders to correct warehouse manager |
| **EscalationJob.cls** | Class (Schedulable) | Escalates delayed shipments (> 7 days) |

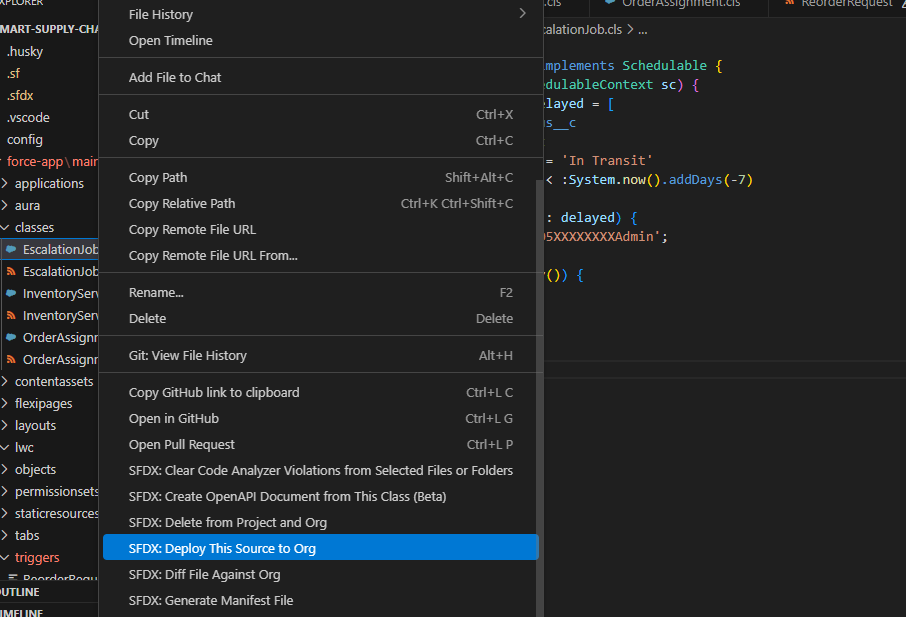
Apex Classes: Inventory Service, Order Assignment. Triggers for Reorders & Shipments. Batch Apex, Queueable Apex, Scheduled Apex, Test Classes.

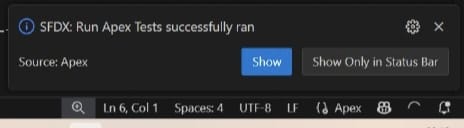








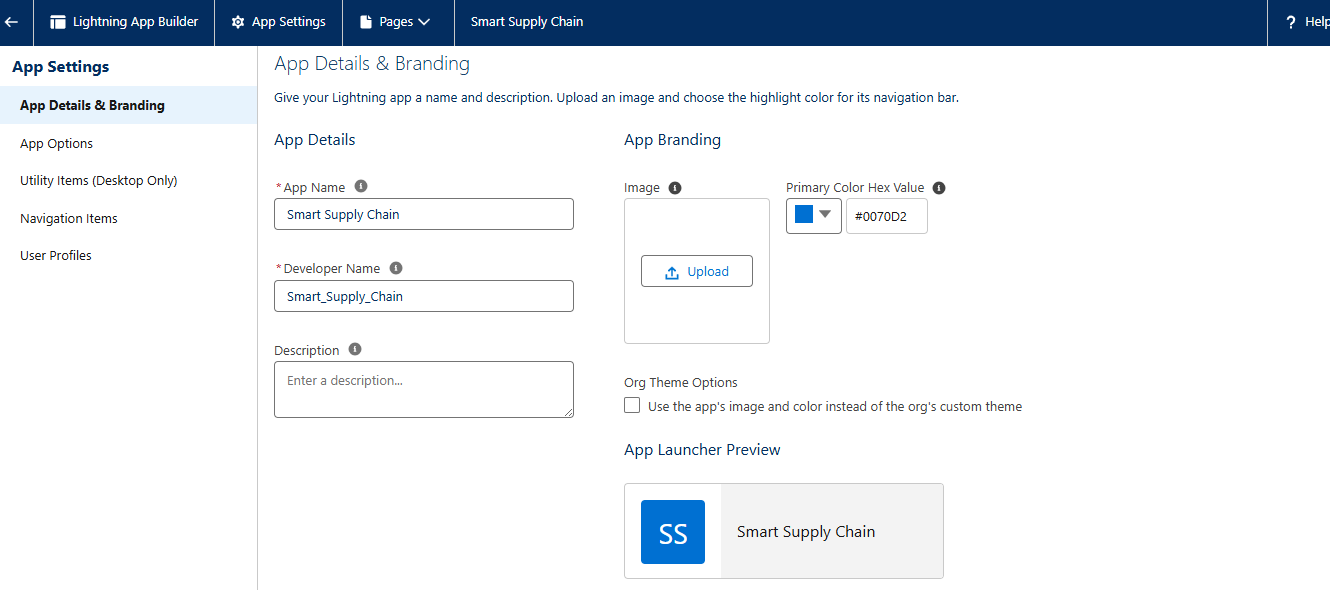




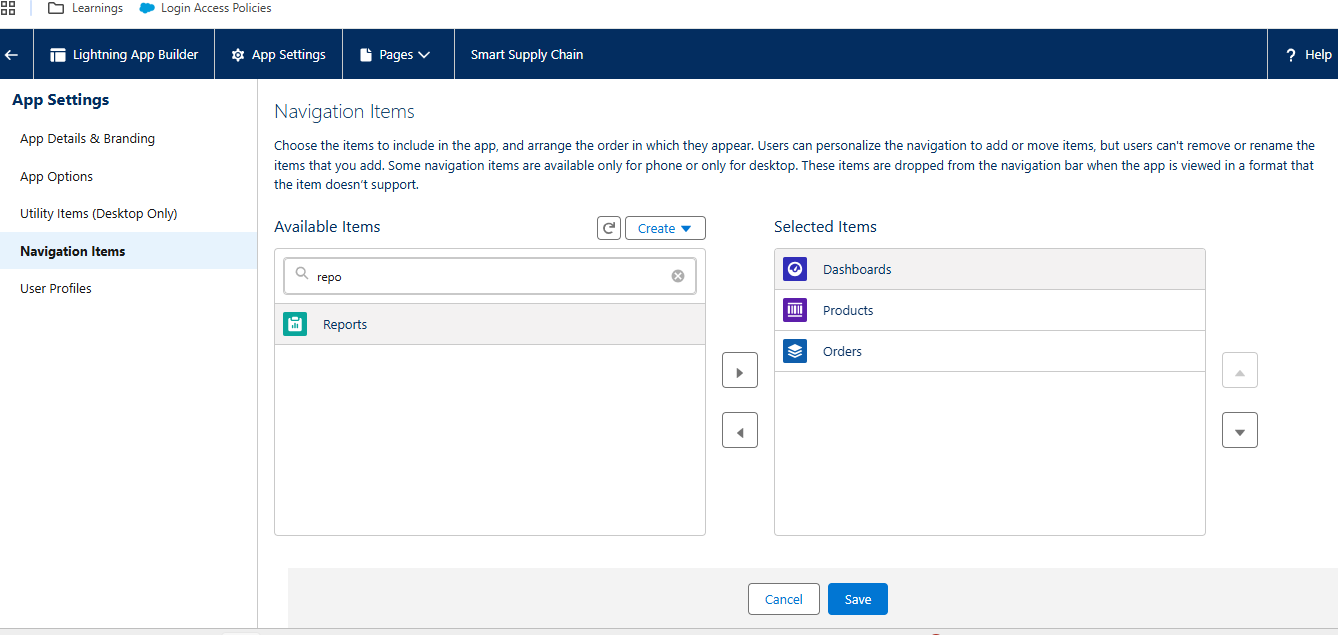
Phase 6: User Interface Development

**Step 1: Create a Custom Lightning App**

1. Go to **Setup → App Manager**.
2. Click **New Lightning App**.
3. Enter:
   * **App Name:** Smart Supply Chain
   * **Developer Name:** Smart\_ Supply\_ Chain
   * **Navigation Style:** Standard Navigation
4. Add **Utility Items** (bottom toolbar): Reports, Dashboards, Notifications.
5. Assign to **Profiles** → Admin, Warehouse Manager, Delivery Agent.
6. Save.

 **Step 2: Add Tabs to App**

1. In App Manager → Edit Smart Supply Chain.
2. Go to **Navigation Items**.
3. Add Tabs:
   * Products
   * Inventory
   * Orders
   * Shipments
   * Reorder Requests
   * Dashboards
4. Save.



**Step 3: Customize Record Pages**

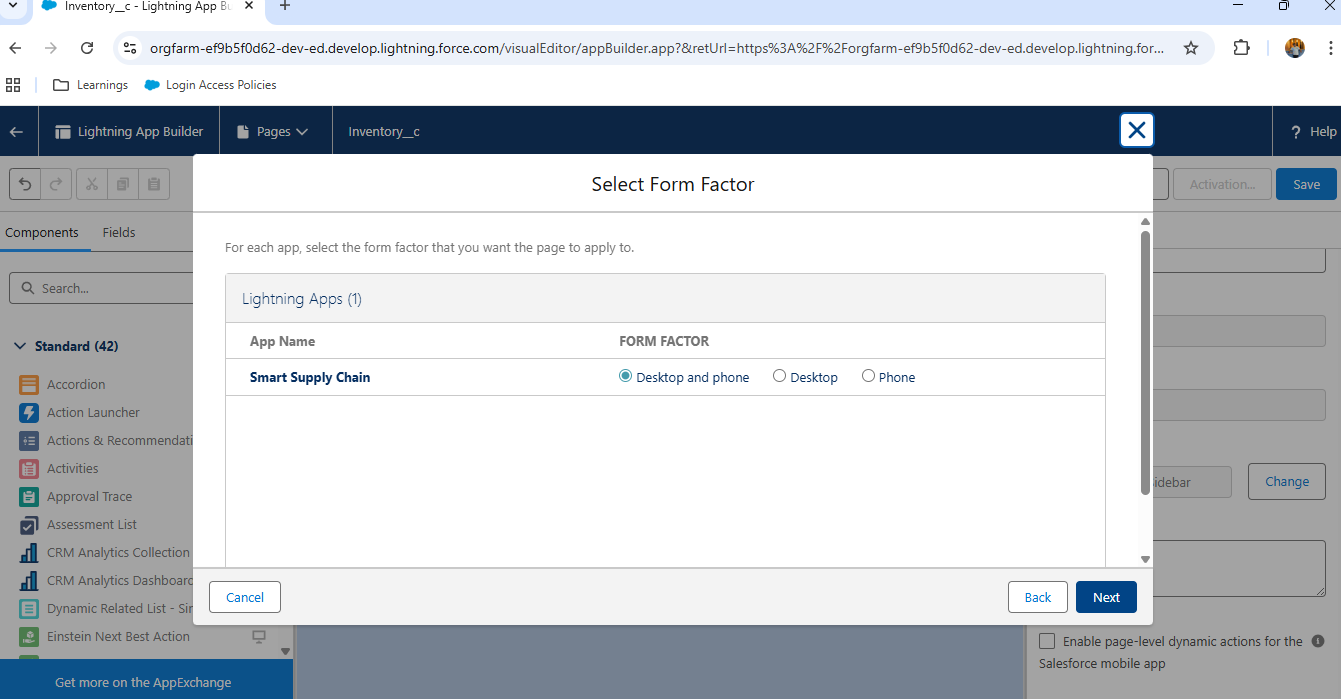
For each custom object (Inventory, Order, Shipment):

1. Open **Object Manager → [Object] → Lightning Record Pages**.
2. Click **New Page** → **Record Page**.

**Inventory**:

* + Layout: Header → Stock details.
  + Tabs: Details, Related, Activity.
  + Add Components:
    - **Highlights Panel** → Show Product, Quantity, Reorder Level.
    - **Related List** → Reorder Requests.

1. Save & **Activate for App, Record Type, Profile** → Choose Smart Supply Chain app.



**Step 4: Build LWC Components**

**a) Inventory Dashboard LWC**

* Shows stock levels in a bar chart.
* Fetch data with @wire from Inventory.

**b) Shipment Tracker LWC**

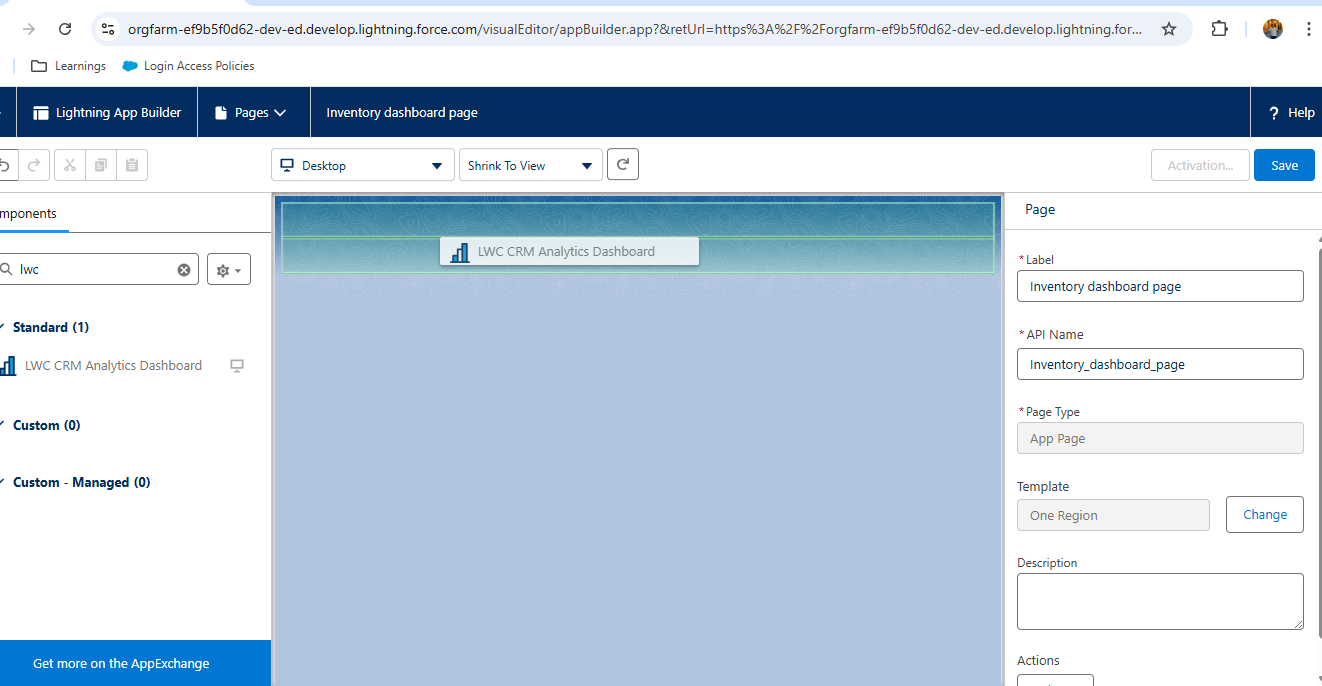
* Shows shipments with colour codes:
  + Green = Delivered
  + Orange = In Transit
  + Red = Delayed

**c) Reorder Approval LWC**

* Approver can approve/reject reorder requests directly from the component.

**Step 5: Home Page Customization**

1. Go to **App Builder → Home Page → Clone Standard Home Page**.
2. Add:
   * **Dashboard Component**
   * **List View** (Pending Reorders).
   * **News/Tips**.
3. Save & Activate for **Smart Supply Chain App**.



**Step 6: Utility Bar Setup**

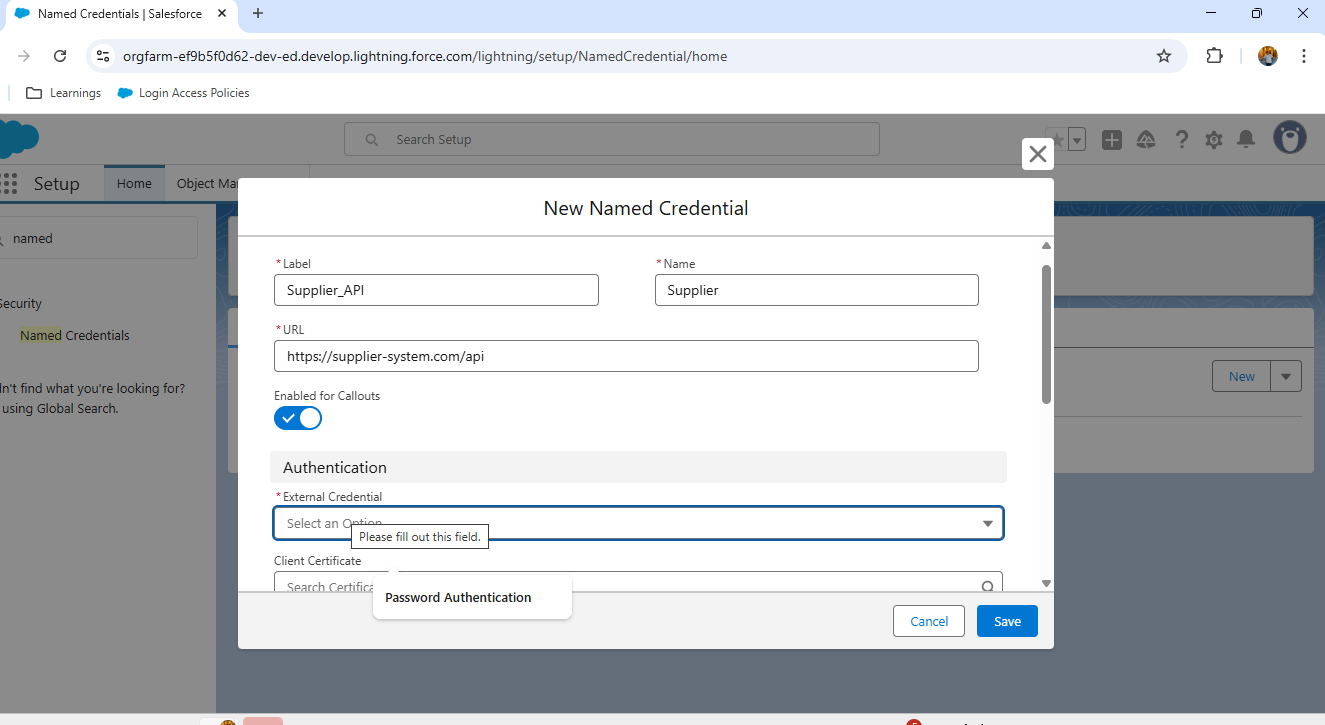
In **App Builder → Utility Items**:

* Add **Reports** → Quick access.
* Add **Dashboard** → Real-time KPI view.
* Add **Chatter** → Communication.

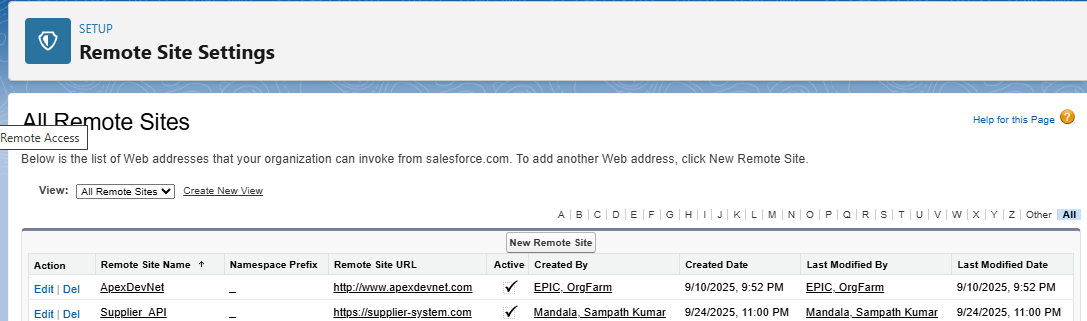
Phase 7: Integration & External Access

**Step 1: Configure Named Credentials**

1. Go to **Setup → Named Credentials**.
2. Click **New Named Credential**.
   * **Label:** Supplier\_ API
   * **URL:** https://supplier-system.com/api
   * **Identity Type:** Named Principal
   * **Authentication Protocol:** Password Authentication
3. Save.

**Step 2: Remote Site Settings**

1. **Setup → Remote Site Settings → New Remote Site**
   * Name: Supplier\_ API
   * URL: https://supplier-system.com
   * Active: ✅



**Step 3: REST Callout (Apex)**

When stock is low, send reorder details to supplier system.

**Example Apex class:**

public with sharing class Supplier Integration {

public static void sendReorder(Reorder\_Request\_\_c req) {

Http http = new Http();

HttpRequest request = new HttpRequest();

request.setEndpoint('callout:Supplier\_API/reorders');

request.setMethod('POST');

request.setHeader('Content-Type', 'application/json');

Map<String, Object> body = new Map<String, Object>{

'productId' => req.Product\_\_c,

'warehouseId' => req.Warehouse\_\_c,

'quantity' => req.Quantity\_\_c

};

request.setBody(JSON.serialize(body));

HttpResponse response = http.send(request);

System.debug('Supplier Response: ' + response.getBody());

}

}

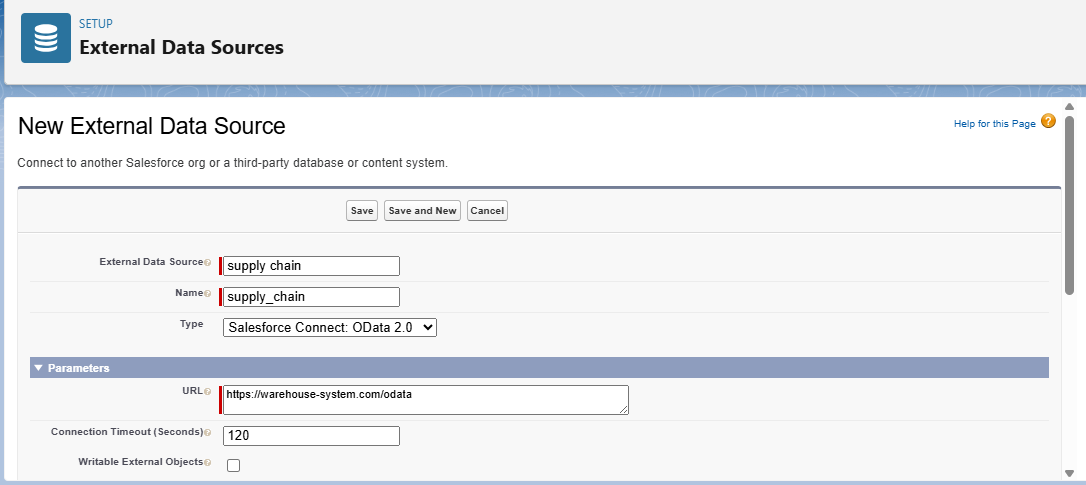
**Step 4: Platform Events**

1. Go to **Setup → Platform Events → New Platform Event**.
   * Label: Shipment\_ Update
   * Fields: Order\_\_ c (Lookup), Status\_\_ c
2. Publish event from Apex when shipment updates.
3. External system can subscribe to these events via API → get notified instantly when Salesforce updates a shipment.

**Step 5: Salesforce Connect**

If warehouse data is stored in an external database, you can view it inside Salesforce without copying data.

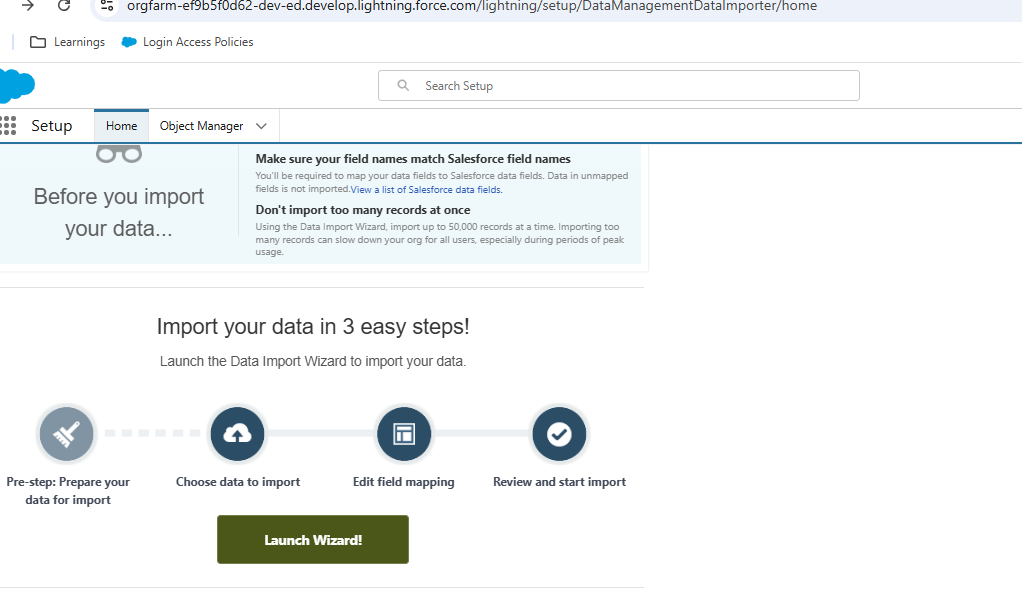
1. Setup → External Data Sources → New.
2. Type: OData 2.0/4.0.
3. Provide endpoint (https://warehouse-system.com/odata).
4. Validate & sync external objects.



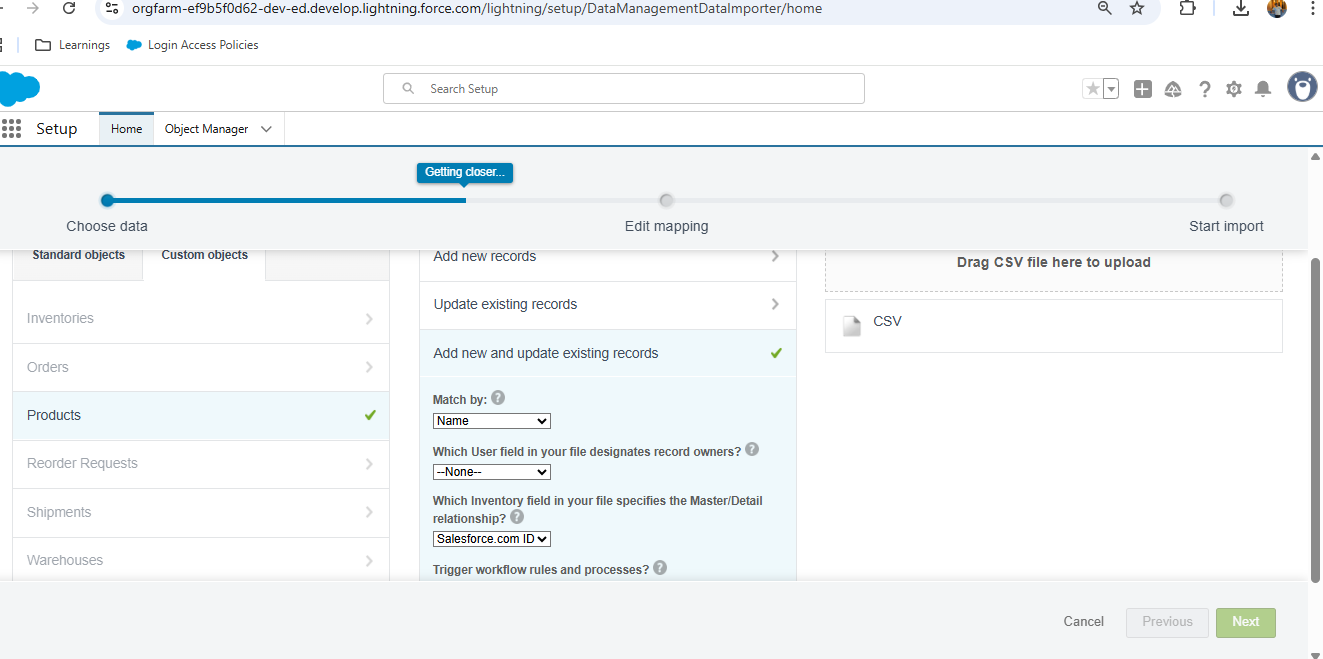
Phase 8: Data Management & Deployment

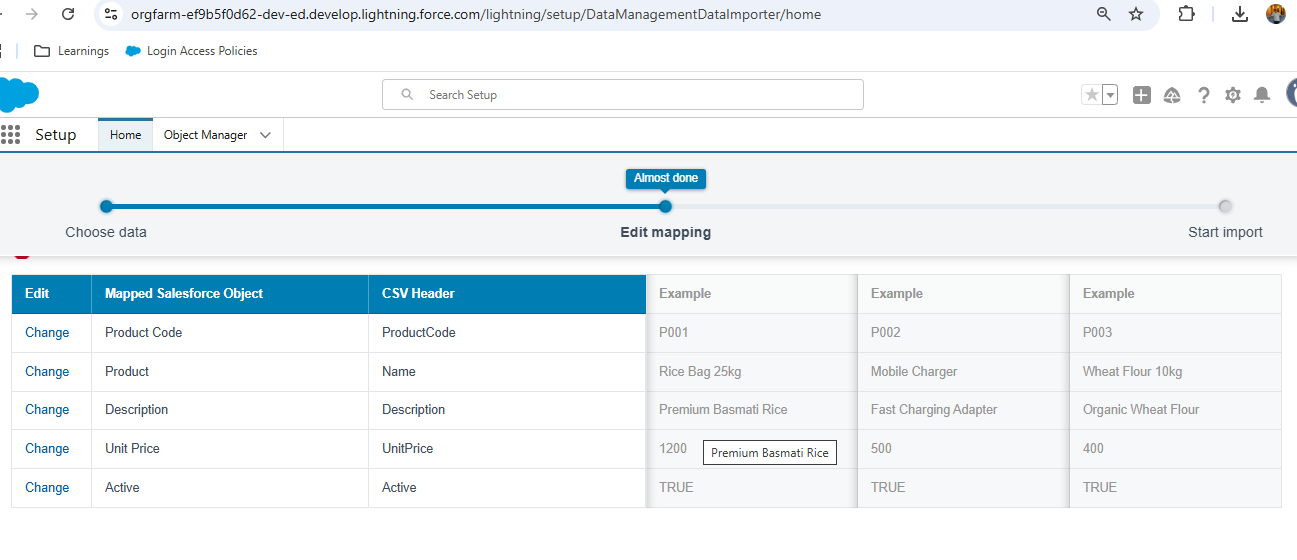
**Step 1: Data Import Wizard**

* Go to **Setup → Data Import Wizard**.
* Use it for **small data loads** (Products, Warehouses, Orders).



* Steps:
  1. Select Object → e.g., Product2.
  2. Upload CSV.
  3. Map fields (Name → Product Name, Unit Price → Unit\_ Price\_\_ c).
  4. Start Import → Monitor progress.



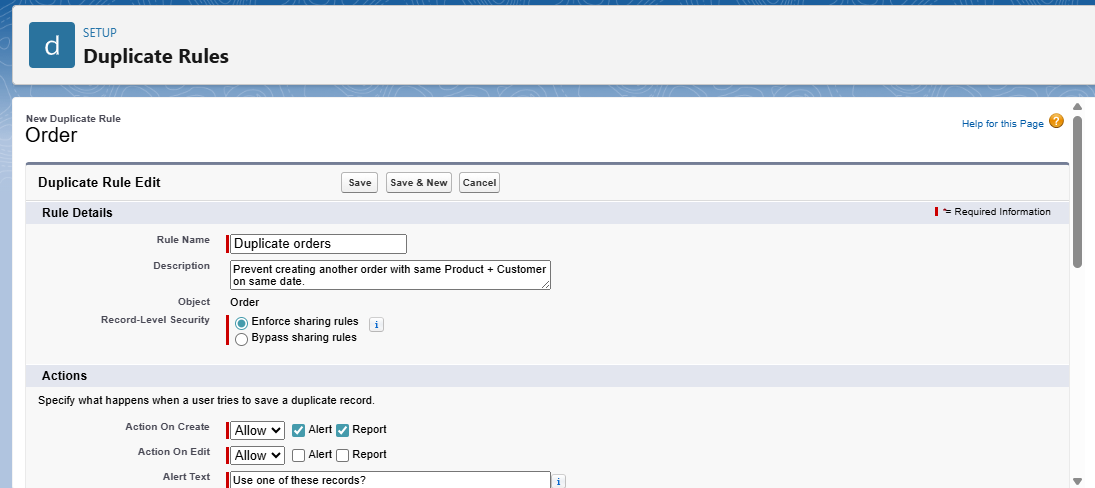


**Step 2: Data Loader**

* Install **Salesforce Data Loader**
* Use it for **bulk data updates**
* Operations supported: Insert, Update Delete, Export.
* Example:
  + Export all Inventory where Quantity\_\_ c < 10.
  + Update them with new stock levels.

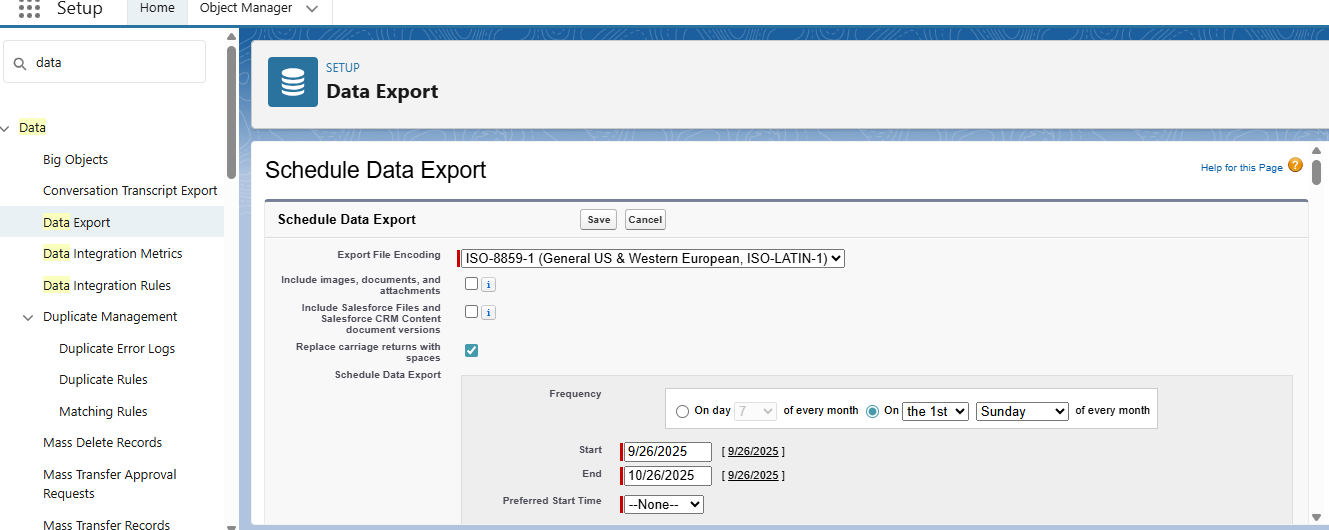
**Step 3: Duplicate Management**

* Go to **Setup → Duplicate Rules**.
* Create rule for Orders: Prevent creating another order with same Product + Customer on same date.
* Action: Block or Allow with Alert.



**Step 4: Data Backup**

* Use **Data Export Service**:
  + Setup → Data Export → Schedule weekly backup.
  + Choose objects (Inventory, Orders, Shipments).
  + Salesforce emails a ZIP file.



**Step 5: Deployment**

1. In **Source Org:** Setup → Outbound Change Sets.
2. Add components: Objects, Flows, Apex Classes, Triggers.
3. Upload to Target Org
4. In **Target Org** → Inbound Change Sets → Deploy.

**Step 6: Deployment with SFDX (VS Code)**

For GitHub-driven workflow:

* To **deploy from VS Code** →
* sfdx force:source:deploy -p force-app/main/default
* To **retrieve from Salesforce** →
* sfdx force:source:retrieve -p force-app/main/default
* To **push to scratch org** →
* sfdx force:source:push

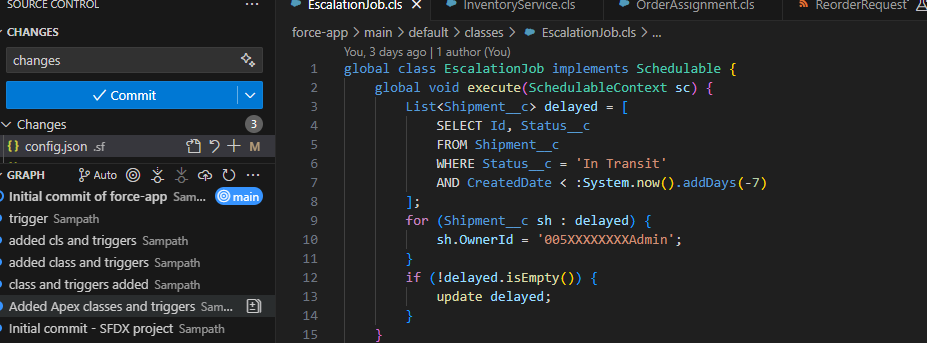
**✅ Step 7: GitHub Version Control**

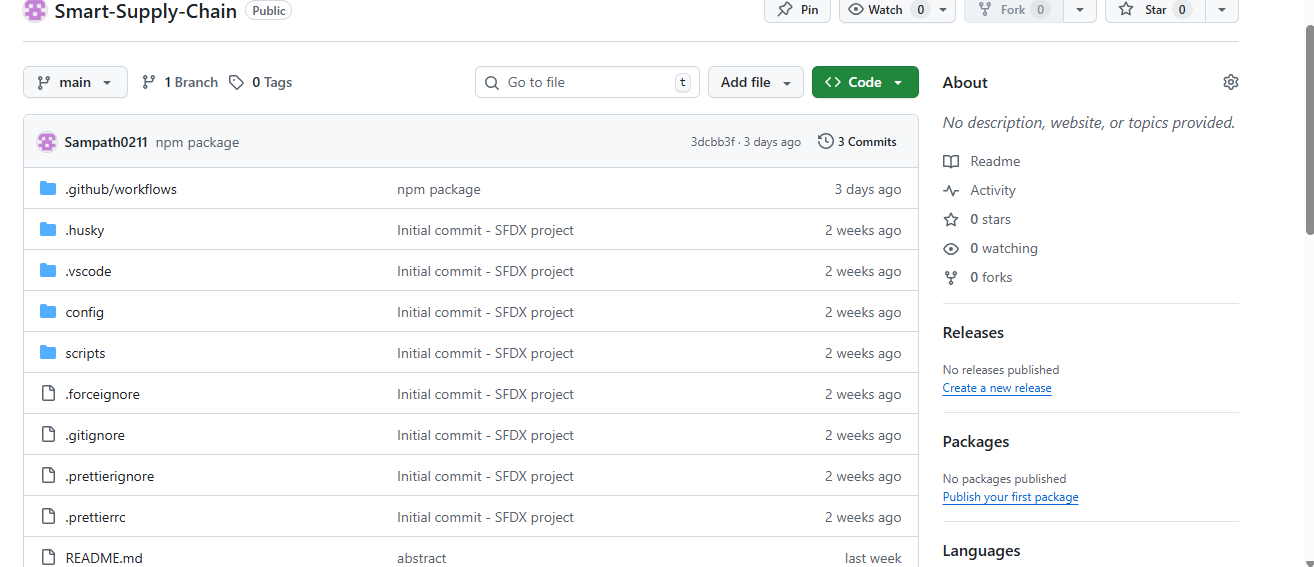
Keep your project versioned:

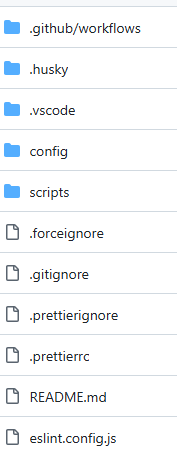
git add .

git commit -m "changes"

git push





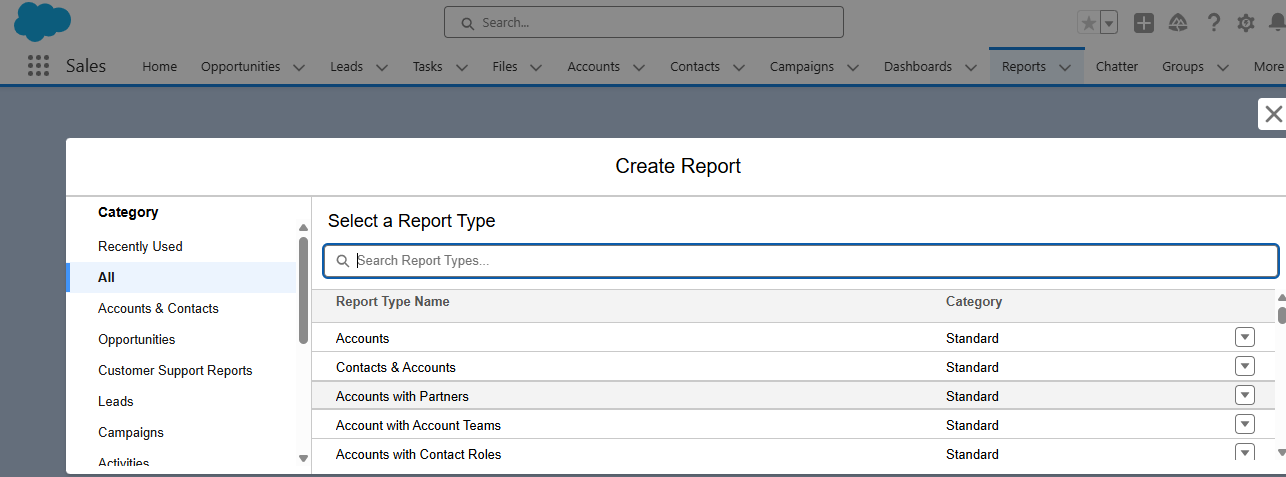


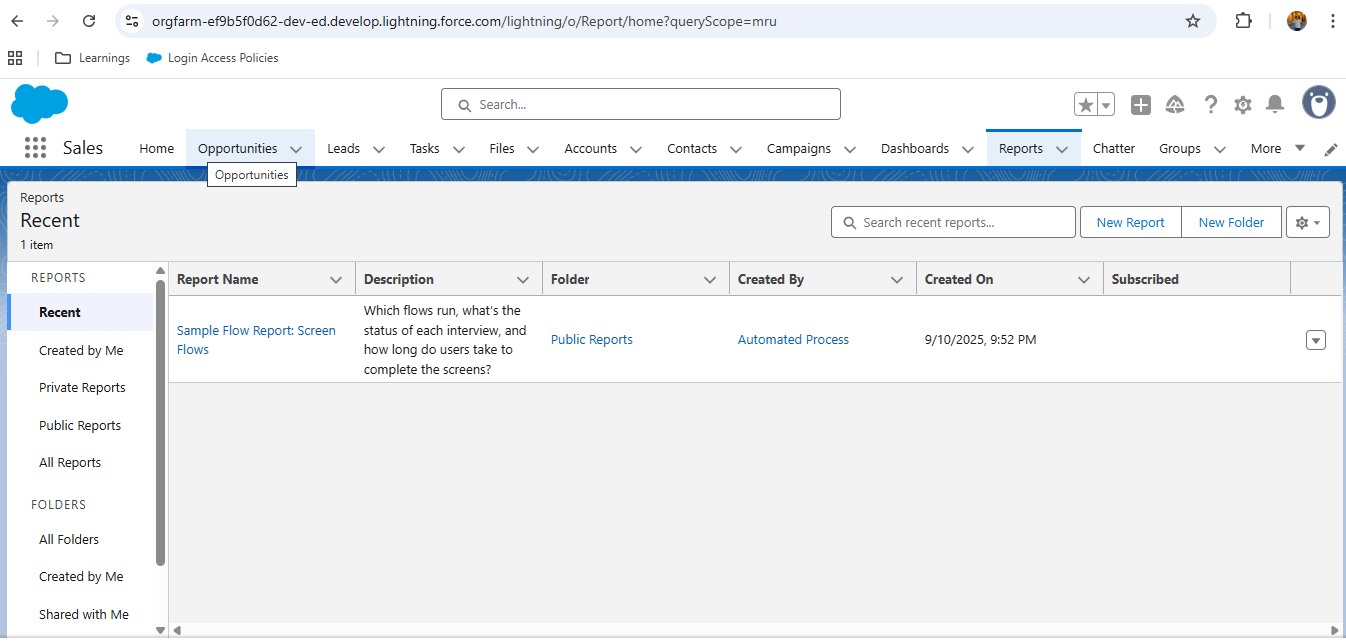
Phase 9: Reporting, Dashboards & Security Review

**Step 1: Reports**

**Reports Created:**

1. **Inventory by Warehouse Report**
   * Report Type: Inventory\_\_ c with Warehouse\_\_ c
   * Fields: Product, Quantity, Reorder\_ Level, Warehouse.
   * Filters: Quantity < Reorder\_ Level → “Low Stock” products.
2. **Orders by Status Report**
   * Report Type: Orders
   * Fields: Order ID, Product, Quantity, Customer, Status.
   * Group by: Status (Draft, Activated, Delivered, Cancelled).
3. **Shipment Performance Report**
   * Report Type: Shipments
   * Fields: Shipment ID, Order, Delivery Agent, Status, Delivery\_ Date.
   * Filter: Status = “In Transit” AND Age > 7 days → Escalations.





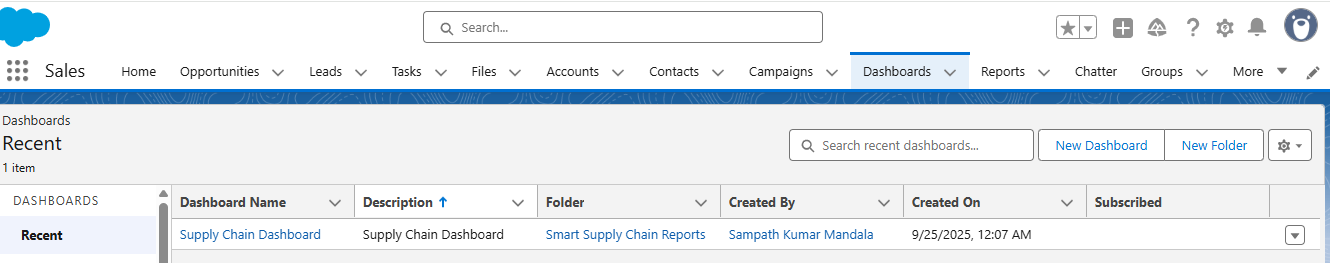
**Step 2: Dashboards**

Dashboards **visualize your reports** into charts & KPIs.

**Create Dashboard → Smart Supply Chain Dashboard**

**Components:**

* **Bar Chart → Inventory by Warehouse**
  + Shows stock levels across warehouses.
* **Pie Chart → Orders by Status**
  + Visual split of how many orders are Draft, Shipped, Delivered.
* **Line Chart → Shipments over Time**
  + Shows delivery performance trends.
* **Metric → Low Stock Products**
  + Quick KPI: number of products below reorder level.



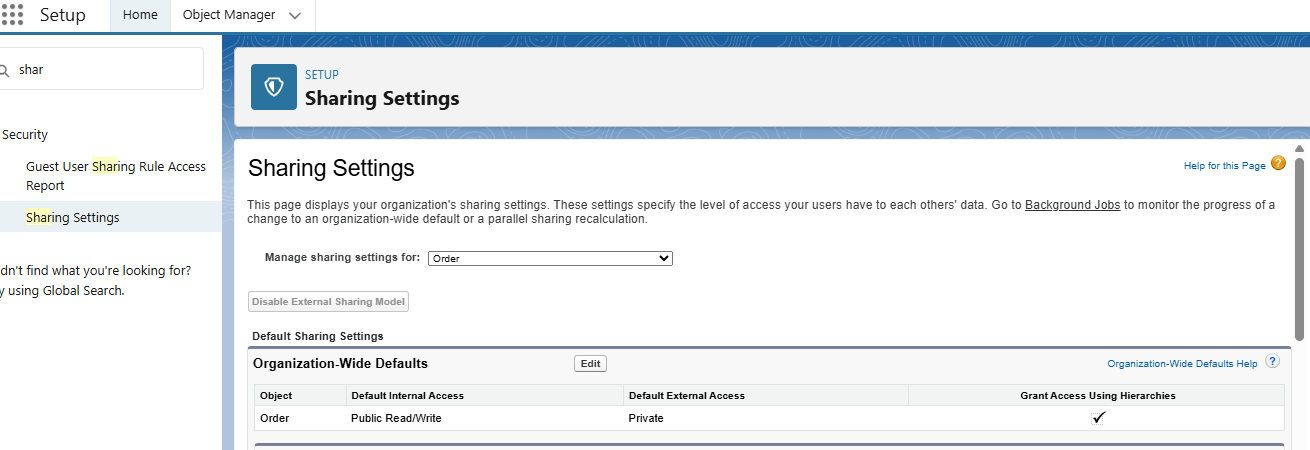
**Step 3: Dynamic Dashboards**

* Go to **Dashboard Settings → Run Dashboard As → Logged-in User**.
* This makes dashboards **role-based**:
  + Admin → Sees all warehouses, orders, shipments.
  + Warehouse Manager → Sees only their warehouse stock & orders.
  + Delivery Agent → Sees only shipments assigned to them.

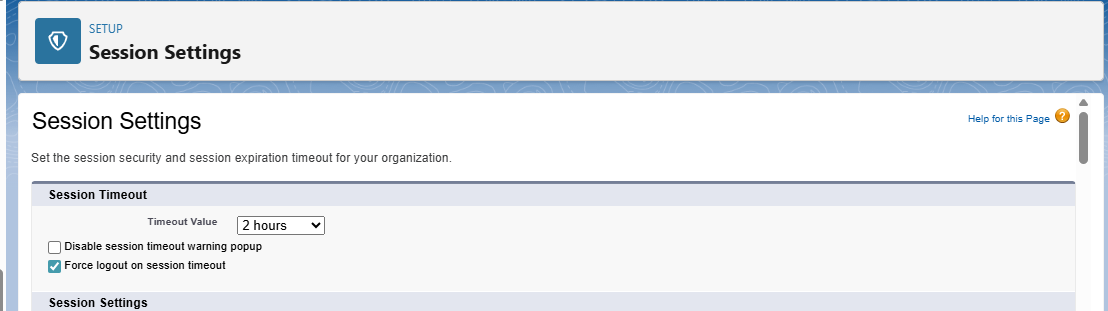
**Step 4: Security Review**

**security review**:

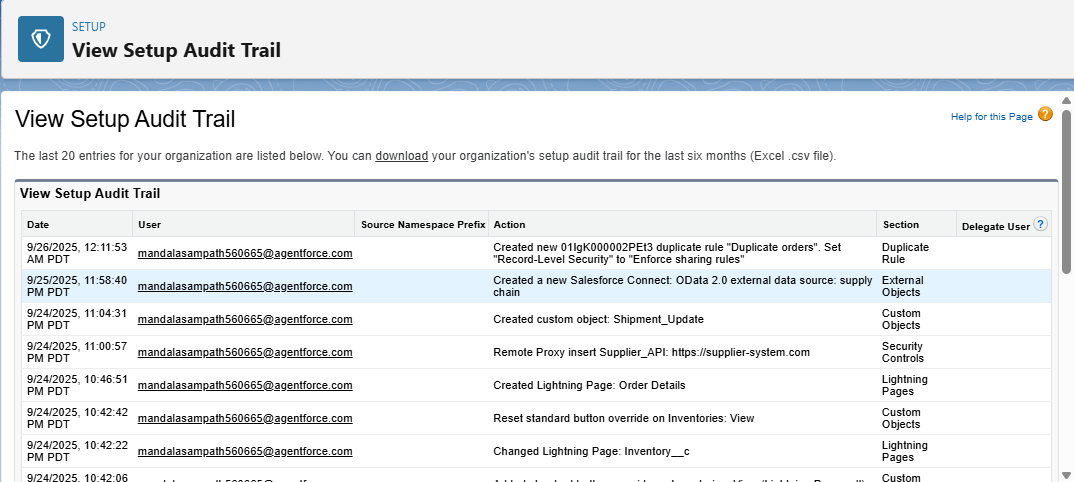
1. **OWD**
   * Inventory → Private
   * Orders → Private
   * Shipments → Private
   * Warehouse → Public Read/Write



1. **Sharing Rules**
   * Orders → Shared with Warehouse Manager.
   * Shipments → Shared with Delivery Agent.
2. **Field-Level Security (FLS)**
   * Hide sensitive fields (e.g., Supplier Cost) from Delivery Agents.
3. **Session Settings**
   * Timeout: 2 hrs.
   * Enforce IP restrictions for Warehouse Managers (only office network).



1. **Audit Trail**
   * Setup → Security → View Setup Audit Trail → track configuration changes.



Phase 10: Final Presentation

Presentation