**FINAL REPORT**

**PROJECT TITLE**

**“CRM APPLICATION FOR MEDICAL INVENTORY MANAGEMENT – (DEVELOPER)”**

**College Name**

**Anil Neerukonda Institute of Technology and Sciences (ANITS)**

**Team ID** : **LTVIP2025TMID28927**

**Team Size** : 4

**Team Leader** :M. Tirumala

**Email ID :** [malipedditirumala.le.23.it@anits.edu.in](mailto:malipedditirumala.le.23.it@anits.edu.in)

**Team member :** Gayathri Vaddi

**Email ID :** [vaddigayathri.22.it@anits.edu.in](mailto:vaddigayathri.22.it@anits.edu.in)

**Team member** : Ch Venkata Balaji

**Email ID :**[balajich0307@gmail.com](mailto:balajich0307@gmail.com)

**Team member** : Gorji Jagadeesh  
**Email ID :** [jagadeeshgorji1@gmail.com](mailto:jagadeeshgorji1@gmail.com)

**1. INTRODUCTION**

**1.1 Project Overview**

The **CRM Application for Medical Inventory Management** is a purpose-built solution designed to help hospitals, pharmacies, and medical supply chains move away from manual tracking methods and adopt a more efficient, centralized, and automated system using the Salesforce platform.

1. **Customer Management** – Maintaining comprehensive records of each customer, including their contact details, preferences, gender, and order history.
2. **Inventory/Item Management** – Keeping track of the jewelry items available (categorized as Gold and Silver), their attributes (weight, ornament type, pricing components), and current stock.
3. **Order Management** – Allowing the staff to place and monitor customer orders with details such as quantity, item type, and linked customer.
4. **Billing and Payments** – Automating the invoice generation process by calculating total amounts using formula fields and sending confirmation emails to customers using flows.

The project leverages Salesforce’s **custom objects**, **page layouts**, **record types**, **profiles**, **roles**, **Apex triggers**, and **record-triggered flows** to ensure seamless operation and maintain high data accuracy and integrity. The use of dashboards and custom reports enhances visibility for business owners and stakeholders, enabling them to make data-driven decisions.

The Lightning App interface allows all users (admin, goldsmith, workers) to navigate between different functionalities with ease, ensuring user-friendliness and efficiency in day-to-day operations.

**1.2 Purpose**

The primary purpose of this project is to provide a **scalable, automated, and user-friendly CRM solution** for jewelry businesses using the Salesforce platform. By developing this system, the team aims to solve real-world business problems related to manual tracking and customer mismanagement that many local and regional jewelers face. The solution focuses on the following key goals

**Customer Data Handling**

* Create a structured repository for customer records.
* Provide lookup-based linking between customer records and orders/billings.
* Enable role-based access so that only authorized users can view/edit sensitive customer data.

**Order Tracking**

* Keep a detailed log of customer orders, including item selection, quantity, and order date.
* Allow users to view all orders linked to a specific customer or item.
* Make it easier to retrieve historical orders for references, refunds, or exchanges.

**Automated Billing**

* Automatically compute the total billing amount based on item details like price, KDM charges, stone charges, and making charges.
* Minimize manual calculation errors using **formula fields** in the Billing object.
* Ensure billing records are always synchronized with payment records using Apex Triggers.
* Allow users to view paid amount, outstanding amount, and update payments with ease.

**Pricing History Management**

* Maintain historical pricing of gold and silver using the **Price\_\_c** object.
* Provide a time-based reference to pricing, allowing for price trend analysis.
* Support dynamic referencing of valid prices during billing and inventory updates.

**Role-Based Access Control**

* Define different **profiles** and **roles** such as Admin, Goldsmith, and Worker.
* Ensure that users can access only the data and functionalities relevant to their job role.
* Improve security and accountability within the organization.

**Email Notifications & Communication**

* Send automated billing confirmation emails to customers using Record-Triggered Flows.
* Provide customers with a digital proof of their transactions.
* Enhance professionalism and customer experience.

**Reporting and Business Intelligence**

* Build custom **reports** and **dashboards** to visualize KPIs such as:
  + Customer count by gender
  + Total revenue by item
  + Item type distribution
  + Orders over time
* Empower business owners to analyze trends, performance, and make informed decisions.

**Data Accuracy and Validation**

* Implement **Validation Rules** to prevent data entry errors, such as entering a paid amount greater than the bill amount.
* Enforce business logic directly at the database level, reducing the chances of fraud or mistakes.

**2. IDEATION PHASE**

**2.1 Problem Statement**

**Purpose**:  
Problem statements help ensure that the solution being built addresses real-world issues that medical staff, inventory managers, and procurement officers face. These statements define what the user is struggling with, why it’s happening, and what the consequences are — setting a focused foundation for CRM development.

**Problem Statement 1: Operational Breakdown and Stock Mismanagement in Medical Facilities**

We believe that medical inventory managers and procurement staff are struggling with real-time stock monitoring, expiry tracking, and purchase order management because of the absence of an integrated digital system and reliance on manual tools such as spreadsheets and notebooks. This causes untracked stockouts and overstocking, increased risk of expired product usage, and significant delays in the procurement cycle.

**Elaboration:**

* Customer Type: Inventory Managers, Pharmacists, Procurement Officers in hospitals, clinics, or healthcare supply chains.
* Core Problem: “Struggling with real-time stock monitoring and procurement” reflects the inability to control what’s available, what’s needed, and what’s outdated.
* Root Causes:
  + “Absence of an integrated system” means that inventory, order, supplier, and expiry information is not connected.
  + “Reliance on manual tools” introduces errors, slow access to data, and no automated checks or workflows.
* Negative Impacts:
  + Untracked stockouts and overstocking: Medical items may be ordered in excess or not reordered in time, leading to waste or lack of supply during critical needs.
  + Increased risk of expired product usage: Manually tracking expiry dates can lead to use of expired medications or equipment, compromising patient safety.
  + Procurement delays: Without automated alerts or workflows, purchase orders are delayed or lost, resulting in operational inefficiencies.

**Problem Statement 2: Lack of Proactive Communication and Uncertainty in Supply Chain Coordination**

We believe that procurement managers and inventory teams are struggling with timely communication and visibility into order fulfillment and supplier delivery because of manual follow-ups and lack of automated tracking systems. This causes uncertainty in delivery timelines, frequent miscommunication with suppliers, and increased time spent on status checks and error correction.

**Elaboration:**

* Customer Type: Procurement Officers, Store Supervisors, Inventory Staff.
* Core Problem: “Struggling with timely communication and delivery visibility” reflects a broken feedback loop between ordering and receiving goods.
* Root Causes:
  + “Manual follow-ups” means staff are emailing or calling suppliers individually, without standardized communication.
  + “Lack of automated tracking systems” means there's no way to get notified when an order is delayed or fulfilled.
* Negative Impacts:
  + Uncertainty in delivery timelines: Orders may arrive late or be forgotten altogether, leaving critical departments understocked.
  + Frequent miscommunication: Mismatched expectations between internal teams and suppliers leads to finger-pointing and blame.
  + Time wasted on manual tracking: Employees spend hours chasing updates instead of focusing on core tasks.

**Problem Statement 3: Compliance Risks and Data Fragmentation in Medical Inventory Handling**

We believe that healthcare inventory teams are struggling with maintaining accurate and auditable records for inventory movements and financial reconciliation because of poor system integration and lack of role-based visibility. This causes difficulty in meeting compliance standards, fragmented record-keeping, and loss of accountability across users.

**Elaboration:**

* Customer Type: Compliance Officers, Inventory Managers, Admins.
* Core Problem: “Struggling with accurate and auditable record-keeping” is critical in highly regulated environments like healthcare.
* Root Causes:
  + “Poor system integration” means there’s no consolidated log of inventory transactions, billing, and supplier histories.
  + “Lack of role-based visibility” means too many users have full access or no access, increasing the risk of unauthorized changes and user confusion.
* Negative Impacts:
  + Compliance difficulties: Auditors cannot easily track transaction history, approval workflows, or expired items.
  + Fragmented data: Different teams use different tools, causing gaps in information.
  + Loss of accountability: Without audit trails or user-specific access controls, errors can’t be traced to specific users.

**2.2 Empathy Map Canvas**

The Empathy Map Canvas serves as a powerful tool to step into the shoes of your target users, understanding their world beyond just explicit requirements. By exploring what users say, think, do, and feel, we gain deeper insights into their needs, pain points, and motivations, which are crucial for designing a truly effective solution.

**Who Are We Empathizing With?**

1. Medical Inventory Managers:  
   Responsible for maintaining accurate records of incoming/outgoing medical stock, monitoring expiry dates, managing reorders, and ensuring compliance. They are under pressure to ensure that no item goes expired, stockouts are avoided, and regulatory documentation is up to date.
2. Procurement Officers:  
   Responsible for creating purchase orders, tracking supplier deliveries, managing invoices, and maintaining supplier relationships. Their key concerns include cost control, timely deliveries, vendor performance, and financial reconciliation.

**User Says (What the user verbalizes)**

* “I want to know what’s about to expire without checking everything manually.”  
  Implication: They need expiry monitoring that is proactive, not reactive—automated alerts, filters for soon-to-expire items, and prioritized task lists.
* “I don’t want to calculate total costs every time.”  
  Implication: Purchase order cost tracking should be automated through related Order Items, minimizing the manual arithmetic they currently perform.
* “I can’t track which supplier is delivering late.”  
  Implication: They lack visibility into supplier performance and delivery timelines. This requires detailed purchase order tracking and supplier analytics.

**User Thinks (What the user is pondering but may not say)**

* “There has to be an easier way to reorder stock.”  
  Implication: They are mentally burdened by complex or manual restocking processes. The system must automate reorder suggestions based on minimum stock levels.
* “I hope I don’t miss another expired item.”  
  Implication: There's constant stress due to limited visibility. The CRM must make expiry status visible at a glance—dashboards, color-coded indicators, and reports.

**User Does (What the user does in their routine)**

* Maintains Excel sheets or notebooks for stock & expiry tracking  
  Implication: This is inefficient, error-prone, and non-collaborative. The CRM should serve as a centralized, digital record-keeping and alert system.
* Calls suppliers or sends emails manually for order follow-up  
  Implication: Time-consuming communication that should be replaced with automated reminders, status updates, and delivery logs in the system.

**User Feels (Emotional triggers)**

* Stressed when an item expires or stock runs out  
  Implication: Errors can impact patient safety and compliance. CRM must prevent this with proactive alerts and reporting.
* Relieved when inventory levels and expiry dates are clearly visible  
  Implication: Simple, well-visualized data reduces anxiety. Dashboards and real-time data panels are not luxury—they’re essential for peace of mind.

**Insights Gained**

1. Critical Need for Centralized Inventory Visibility  
   Insight: Manual stock tracking via notebooks and spreadsheets creates silos and increases error risk. Inventory Managers require a centralized, cloud-based system to track:
   * Current stock
   * Expiry dates
   * Minimum stock alerts
   * Supplier and purchase order linkages  
     Impact on Design:
   * Custom objects: Product, Supplier, Purchase Order, Order Item, Inventory Transaction
   * Relationships between objects: Lookup and Master-Detail fields to connect records
   * Dashboards to visualize stock and expiry risks
2. Automation of Procurement & Expiry Alerts  
   Insight: Procurement staff currently rely on mental math or manual calculations for reordering and cost estimation. Expiry monitoring is entirely manual.  
   Impact on Design:
   * Validation Rules: e.g., Expected Delivery Date should not exceed 7 days
   * Flows: Record-Triggered Flows to auto-update delivery dates, notify on low stock or expiry
   * Apex Trigger: Automatically update Purchase Order total based on Order Items
3. Need for Role-Based Access Control  
   Insight: Inventory Managers and Procurement Officers have overlapping but distinct responsibilities. Not everyone should see or edit all data.  
   Impact on Design:
   * Profiles: Inventory Manager, Purchase Manager
   * Roles: Procurement Manager, Inventory Officer
   * Permission Sets: Allow specific tasks like editing Order Items or generating reports
   * Field-Level Security and Page Layouts: Tailored per role for focused user experience
4. Reporting & Dashboarding is Core, Not Optional  
   Insight: Users value insights like supplier performance, expiring stock, and purchase order volume—but don't currently have tools to access them easily.  
   Impact on Design:
   * Custom Reports: Purchase Orders by Supplier, Items Nearing Expiry
   * Summary & Matrix Reports: Complete Purchase Details
   * Dashboards: Medical Inventory Overview, Reorder Insights, Procurement Health
5. Communication Automation is a Must  
   Insight: Manual calls and follow-ups with suppliers and internal teams consume valuable time and lead to inconsistencies.  
   Impact on Design:
   * Record-Triggered Flows to send email notifications (e.g., order received, expiry warning)
   * Email Templates for communication standardization
   * Email Logs to track delivery and status of automated message

**2.3 Brainstorming**

During the ideation stage, our team conducted multiple whiteboard sessions and brainstorming activities using tools like **Miro** and **Google Jamboard**. We collected all user pain points and mapped them to potential technical features that Salesforce could support.

**Step 1: Team Gathering, Collaboration, and Problem Identification**  
The team interviewed healthcare professionals, pharmacists, and inventory personnel from clinics and local medical supply stores to understand operational pain points in managing medical stock. Problems identified include:

* Lack of real-time visibility of stock levels and product expirations
* Manual handling of supplier and purchase order records
* No automation for tracking expiry dates or low stock alerts
* Inability to monitor supplier performance and order history effectively
* Poor communication between inventory and procurement departments

**Selected Problem Statement:  
“Healthcare facilities lack a centralized system to manage medical inventory, monitor expiry dates, automate replenishment, and streamline supplier communication.”**

**Step 2: Brainstorming, Idea Listing, and Grouping**

**Raw Ideas Collected:**

* Automate low-stock alerts and expiry date warnings
* Track supplier delivery timelines and performance
* Auto-calculate purchase order cost from order items
* Enable real-time inventory deduction after transaction
* Send automated notifications to procurement
* Create dashboards for inventory and procurement managers
* Generate reports: Expiry Summary, Order History, Supplier Efficiency
* Use record-triggered flows to auto-update delivery dates
* Validation rules for stock thresholds and delivery timelines
* Role-based data access for pharmacists, procurement staff, and admin

**Grouped Ideas:**

1. Automation & Workflow
   * Low-stock/expiry alerts
   * Auto-calculate order cost
   * Auto-update delivery dates via Flow
2. Inventory Control
   * Monitor stock by category (medicine, equipment, disposables)
   * Track expiry and minimum stock levels
   * Real-time update after transactions
3. Role-Based Access
   * Profiles: Pharmacist, Procurement Manager, Inventory Assistant
   * Permission sets for controlled access
4. Reporting & Dashboards
   * Inventory Summary Dashboard
   * Supplier-based Purchase Reports
   * Expiry Date Reports
5. Communication Automation
   * Email notifications for low stock or received orders
   * Notifications for delayed or missed deliveries

**Step 3: Idea Prioritization Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Idea | Impact | Feasibility | Priority |
| Low-stock alert automation | High | High | High |
| Expiry date validation rule | High | Medium | High |
| Role-based access configuration | Medium | High | Medium |
| Purchase Order cost calculation | High | High | High |
| Dashboards for inventory | Medium | Medium | Medium |
| Supplier efficiency reports | Medium | Medium | Medium |

**3. REQUIREMENT ANALYSIS**

**3.1 Customer Journey Map**  
This journey map outlines the end-to-end experience of internal users (Inventory Managers, Procurement Officers) interacting with the Medical Inventory Management CRM. It highlights how system interactions automate critical processes, reduce error rates, and enhance operational transparency.

|  |  |  |
| --- | --- | --- |
| Step | User Action (Detailed Scenario & Intent) | System Interaction (CRM Role & Data Capture) |
| 1 | Receives low-stock alert or performs inventory check The inventory manager begins the day by reviewing inventory levels. They need to identify products nearing expiry or below the minimum stock level. | Dashboard component displays low-stock/expiry alerts The system uses Roll-Up Summary fields and validation rules to flag inventory items that meet reorder or expiry conditions. Dashboards present this data visually using bar charts, lists, and gauges to guide the user’s action. |
| 2 | Initiates new purchase order The procurement officer decides to replenish stock and begins drafting a purchase order. | New Purchase\_Order\_\_c record created The CRM captures fields like Supplier, Order\_Date\_\_c, Expected\_Delivery\_Date\_\_c, and auto-generates a Purchase Order ID. Lookup fields connect this order to the Supplier\_\_c object. |
| 3 | Adds line items to order The user enters each item to be ordered, including quantity and unit cost. | Related Order\_Item\_\_c records created Each item is logged as a child of the Purchase Order via Master-Detail. Fields include Product (via lookup), Quantity\_Ordered, and Unit\_Price. An Apex Trigger calculates Amount\_\_c = Quantity \* Unit\_Price. |
| 4 | Submits purchase order and awaits delivery After saving the purchase order, the user expects confirmation and proceeds to monitor delivery. | Flow or Email Alert sends confirmation A Flow automatically triggers upon order submission to notify suppliers or internal teams. Meanwhile, the system tracks Order\_Status\_\_c and Expected\_Delivery\_Date\_\_c. |
| 5 | Updates delivery status and logs receipt The goods arrive and the delivery status is updated. The intent is to acknowledge delivery and ensure quantity received matches ordered. | Actual\_Delivery\_Date\_\_c auto-calculated A Record-Triggered Flow auto-fills Actual\_Delivery\_Date\_\_c as Order\_Date + 3. User updates Quantity\_Received on Order\_Item\_\_c. Inventory\_Transaction\_\_c is created with Transaction\_Type = ‘Receipt’. |
| 6 | Inventory auto-updated Inventory levels must reflect the newly received stock to maintain real-time accuracy. | Current\_Stock\_Level\_\_c on Product updated via Flow Upon delivery confirmation, an after-save Flow or Trigger increments Product’s Current\_Stock\_Level\_\_c. |
| 7 | Views reports and dashboard summaries Admins and managers view summarized data for performance tracking, budgeting, and reordering insights. | Reports and Dashboards display KPIs Reports like “Purchase Orders by Supplier” and “Complete Purchase Details” feed into dashboards showing metrics like Total Order Cost, Supplier Performance, and Inventory Value. |

This journey ensures seamless automation and accurate data flow between users and the system, eliminating manual errors, ensuring compliance, and improving procurement efficiency.

**3.2 Solution Requirements**

**Functional Requirements:**

* Ability to create Product records (Product\_\_c): The system must allow for detailed creation and classification of all medical items, including medicines, equipment, consumables, etc.
* Ability to register and manage Supplier records (Supplier\_\_c): The system must support adding new suppliers with necessary contact, license, and business details.
* Create and track Purchase Orders: The CRM must enable the procurement team to generate Purchase Orders, track their status, and link them to specific suppliers and items.
* Auto-update Current Stock Level based on Quantity Received (Flow/Trigger): Automation must exist to update Product\_\_c’s Current\_Stock\_Level\_\_c field whenever items are received via Order\_Item\_\_c.
* Send delivery notifications to inventory managers (Flow): A record-triggered Flow should be implemented to notify relevant users upon delivery status change or item receipt.
* Generate reports and dashboards: The system must support reporting features for procurement costs, supplier performance, order timelines, stock levels, and product expiry.
* Validate Expected Delivery Date ≤ 7 days from Order Date (Validation Rule): A validation rule must ensure delivery expectations are realistic and within compliance.
* The following are the functional requirements of the proposed solution.

**Functional Requirements Table:**

|  |  |  |
| --- | --- | --- |
| FR No. | Functional Requirement (Epic) | Sub Requirement (Story / Sub-Task) |
| FR-1 | User Registration | - Create internal user accounts (Admin, Manager) - Assign profiles (Inventory Manager, Purchase Officer) |
| FR-2 | Supplier Management | - Create Supplier\_\_c records - Store licensing, contact, and address info |
| FR-3 | Medical Inventory Tracking | - Create Product\_\_c records - Include fields: Min Stock Level, Expiry Date, Unit Price |
| FR-4 | Purchase Order Management | - Create Purchase\_Order\_\_c records - Track delivery and order statuses |
| FR-5 | Order Line Item Tracking | - Add Order\_Item\_\_c under Purchase Orders - Auto-calculate Amount\_\_c = Quantity \* Unit Price |
| FR-6 | Inventory Update Automation | - Trigger Flow on delivery confirmation - Update Current\_Stock\_Level\_\_c on Product\_\_c |
| FR-7 | Notifications & Email Communication | - Notify stakeholders when delivery is confirmed - Use Flow + Email Template |
| FR-8 | Validation Rules | - Expected\_Delivery\_Date\_\_c must be ≤ 7 days from Order\_Date\_\_c - Quantity Received ≤ Quantity Ordered |
| FR-9 | Role-Based Access | - Define access for Inventory Manager, Procurement Officer, Admin - Set permissions using Permission Sets |
| FR-10 | Reporting & Dashboards | - Generate reports: Low Stock, Order Cost, Supplier Performance - Create dashboards to visualize KPIs |

**Non-Functional Requirements:**

* System responsiveness and low latency: Ensure system operations like loading inventory, submitting orders, and generating reports are completed within seconds.
* Enforce user access control (Profiles & Permission Sets): Security protocols must restrict access to only relevant data based on user roles.
* Maintain real-time inventory accuracy: The system must reflect up-to-date stock information for safe procurement decisions.
* Enable secure and traceable communication: All automated emails and internal communications must log status and ensure delivery.
* Field history tracking for accountability: Key objects like Product, Purchase Order, and Order Item should have field history tracking enabled.

**Non-Functional Requirements Table:**

|  |  |  |
| --- | --- | --- |
| NFR No. | Non-Functional Requirement | Description |
| NFR-1 | Usability | The interface must be intuitive and easy to use for non-technical medical staff and procurement officers. |
| NFR-2 | Security | Use profiles and permission sets to protect sensitive stock and supplier data. |
| NFR-3 | Reliability | The system must operate without functional errors during high-volume periods. |
| NFR-4 | Performance | Reports, dashboards, and Flows must respond with minimal delays. |
| NFR-5 | Availability | System should remain available during working hours across departments with minimal downtime. |
| NFR-6 | Scalability | System should support growing supplier networks, product catalogs, and departmental usage. |
| NFR-7 | Maintainability | Admins must be able to modify flows, validation rules, and triggers as policies evolve. |
| NFR-8 | Auditability | Enable Field History Tracking on Purchase\_Order\_\_c and Product\_\_c to monitor changes for compliance. |

**3.3 Data Flow Diagram (DFD)**

A Data Flow Diagram visually represents how data moves between system entities, processes, and storage components.

**Level 0 DFD – Context Diagram**

Description:  
This Level 0 DFD (also known as the Context Diagram) represents the entire Medical Inventory Management CRM as a single unified process. It highlights the interactions between external entities (users) and the CRM system and the major data inputs and outputs.

**External Entities:**

* Inventory Manager
* Procurement Officer
* Supplier

**Process:**

* CRM for Medical Inventory Management (Single System)

**Data Flows:**

* Inventory Manager reviews stock, updates inventory records
* Procurement Officer creates purchase orders and logs deliveries
* Supplier provides goods and delivery confirmations
* CRM system processes, stores, and reports inventory and procurement data
* Automated notifications and reports are generated and emailed

**Visual Representation (Conceptual):**

[Inventory Manager]  
│  
▼  
[Procurement Officer] ────▶ (CRM for Medical Inventory Management) ────▶ [Supplier]  
│ │  
▼ ▼  
[Email Notifications / Reports] [Updated Inventory & Orders]

**Level 1 DFD – Detailed System Flow**

**Description**:  
The Level 1 DFD breaks the CRM system into its major internal processes, objects (data stores), and the data flow between them.

**Step-by-Step Breakdown:**

1. Product Inventory Monitoring
   * Inventory Manager checks current stock and expiry status
   * CRM checks Product object: Current\_Stock\_Level\_\_c, Expiry\_Date\_\_c
   * If threshold crossed, flags item via dashboard or alert
2. Purchase Order Creation
   * Procurement Officer creates new Purchase\_Order\_\_c record
   * Lookup: Linked to Supplier\_\_c
   * Details captured: Order\_Date\_\_c, Expected\_Delivery\_Date\_\_c, etc.
3. Order Item Entry
   * Order\_Item\_\_c records are created as children of Purchase Order
   * Includes fields: Product\_ID\_\_c (Lookup), Quantity\_Ordered, Unit\_Price
   * Apex Trigger auto-calculates Amount\_\_c = Quantity \* Unit Price
4. Delivery & Receipt Update
   * When products are received, Actual\_Delivery\_Date\_\_c is updated via Flow
   * Quantity\_Received field updated in each Order Item
   * Inventory\_Transaction\_\_c created to log stock movement (Transaction\_Type = 'Receipt')
5. Stock Update
   * A Trigger or Flow updates Product object
   * Current\_Stock\_Level\_\_c incremented by Quantity\_Received
6. Reporting & Notifications
   * Reports: Purchase Orders by Supplier, Inventory Expiry Summary
   * Flows send email alerts to notify procurement teams or suppliers of received orders or overdue deliveries

**Key CRM Objects in Level 1 DFD:**

* Product (Product\_\_c)
* Supplier (Supplier\_\_c)
* Purchase Order (Purchase\_Order\_\_c)
* Order Item (Order\_Item\_\_c)
* Inventory Transaction (Inventory\_Transaction\_\_c)

**System Automations:**

* Apex Trigger for Total Order Cost
* Record-Triggered Flows for Delivery Date Update & Email Alerts
* Validation Rules for Expected Delivery Limits and Stock Reorder Flags
* Certainly! Here's the Level 1 – Detailed DFD table for your Medical Inventory Management project, structured to match the same format used in the Jewel Management CRM:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Step | Process | Input | Output | Data Store |
| 1 | Create Product Record | Product details (name, unit, expiry, etc.) | Product\_\_c record | Product\_\_c |
| 2 | Create Supplier Profile | Supplier info (name, contact, license) | Supplier\_\_c record | Supplier\_\_c |
| 3 | Create Purchase Order | Selected Supplier + Order details | Purchase\_Order\_\_c record | Purchase\_Order\_\_c |
| 4 | Add Order Items | Product + Quantity + Unit Price | Order\_Item\_\_c records (linked to PO) | Order\_Item\_\_c |
| 5 | Auto-calculate Total Amount | Quantity & Price from each item | Total\_Amount\_\_c on Purchase Order updated | Purchase\_Order\_\_c |
| 6 | Update Delivery Status | Delivery confirmation | Actual\_Delivery\_Date\_\_c + Quantity\_Received | Purchase\_Order\_\_c, Order\_Item\_\_c |
| 7 | Auto-update Stock Level | Quantity Received | Updated Current\_Stock\_Level\_\_c in Product | Product\_\_c |
| 8 | Send Email Notification | Purchase Order + Delivery Status | Email sent to Procurement/Inventory Manager | Email Log / Notification |

**3.4 Technology Stack**

The Medical Inventory Management CRM is developed using Salesforce’s low-code and pro-code tools, enabling quick deployment while supporting complex automation and data modeling. It focuses on tracking stock levels, streamlining supplier interactions, and automating purchase workflows.

|  |  |  |
| --- | --- | --- |
| **Category** | **Tools/Technologies Used** | **Explanation** |
| Platform | Salesforce Lightning | The application is developed on Salesforce Lightning Experience to offer a clean, responsive UI optimized for inventory workflows and medical procurement users. |
| Automation | Record-Triggered Flows, Workflow Rules | Flows automate processes such as updating stock after delivery and sending delivery confirmation emails. Workflow Rules can be used for field updates on status change. |
| Scripting | Apex Triggers | Triggers automate backend logic, e.g., updating Product\_\_c’s Current\_Stock\_Level\_\_c when Quantity\_Received is entered in Order\_Item\_\_c. |
| Data Modeling | Custom Objects: Product\_\_c, Supplier\_\_c, Purchase\_Order\_\_c, Order\_Item\_\_c, Inventory\_Transaction\_\_c | Custom objects represent core inventory entities. Relationships (Master-Detail or Lookup) enforce data linkage between Products, Orders, and Suppliers. |
| Validation & Rules | Validation Rules, Formula Fields | Validation Rules ensure data accuracy, e.g., Expected\_Delivery\_Date\_\_c cannot exceed 7 days from Order\_Date\_\_c. Formula fields calculate total cost, tax, etc. |
| Communication | Email Alerts, Email Templates, Flows | Flows are configured to send automatic email notifications to procurement teams on order creation or delivery confirmation using custom templates. |
| Reporting & Insights | Reports, Dashboards | Reports such as “Purchase Orders by Supplier” or “Low Stock Products” are combined in Dashboards to track KPIs like order cost, supplier performance, and inventory levels. |
| Access Control | Profiles, Permission Sets | Different user roles such as Inventory Manager, Procurement Officer, and Admin are configured with Profiles and Permission Sets for role-based access to modules and records. |

**Why This Stack Was Chosen:**

* Salesforce Lightning accelerates development while maintaining scalability and a user-friendly experience.
* Apex enables automation of complex business logic, such as auto-updating stock levels or handling order rollups.
* Flows and Email Alerts reduce manual effort and ensure timely communication between departments.
* Reports and Dashboards help teams make data-driven decisions by offering real-time operational visibility.
* Validation Rules and Formula Fields improve data integrity and automate calculations critical to procurement processes.

**Sample Tools Used in Development:**

* Object Manager – for defining custom fields, relationships, and object structures like Product\_\_c and Supplier\_\_c.
* Flow Builder – for creating automations like delivery notifications and stock updates.
* Developer Console – for developing and testing Apex Triggers (e.g., auto-calculation of inventory levels).
* Email Template Builder – for designing standardized communication templates for delivery or order confirmations.
* Report Builder – for generating procurement and inventory performance reports.
* Setup Menu (Profiles/Permission Sets) – for applying access restrictions based on roles (e.g., Inventory Manager vs. Admin).

**4. PROJECT DESIGN**

**4.1 Problem-Solution Fit**

**Objective of Problem–Solution Fit**  
The Problem–Solution Fit phase ensures the Medical Inventory Management system addresses real, high-impact issues faced by healthcare procurement teams and inventory managers. It ensures the solution is not only technically robust but directly aligned with practical day-to-day challenges in medical supply environments.

This phase helps:  
• Align system features with actual hospital/clinic inventory operations  
• Understand the behavioral patterns of procurement officers and staff  
• Increase solution adoption and operational efficiency  
• Minimize inventory-related risks before scaling

1. **Target Customer Segments**

|  |  |
| --- | --- |
| Customer Type | Description |
| Inventory Managers | Monitor stock levels, identify low-stock or expiring items, and ensure compliance with medical inventory policies |
| Procurement Officers | Handle purchase orders, vendor communications, and ensure timely replenishment |
| Store Clerks / Staff | Receive stock, update item statuses, and maintain inventory logs |
| Hospital Admins | Oversee all inventory operations and require insights for audits and budgeting |

1. **Problem Statement (As-Is Situation)**  
   Healthcare and pharmacy inventory systems often rely on manual tracking or loosely connected spreadsheets, leading to:  
   • Manual purchase order generation  
   • No real-time alert for low or expired stock  
   • Lack of visibility into supplier performance  
   • Communication delays during order approval  
   • Frequent mismatches between ordered vs. received quantities

Key Problems Identified:  
• No centralized tracking of stock movement or expiry  
• Delayed stock replenishment leading to medical risks  
• Errors in pricing or quantity without proper validation  
• No automation in delivery tracking or procurement follow-up  
• Lack of dashboards to monitor supplier reliability or order volume

1. **Current Workaround (Before CRM Solution)**

|  |  |
| --- | --- |
| Existing Practice | Limitation |
| Excel sheets to track stock | No real-time updates, error-prone |
| Phone/email to follow up on orders | No delivery tracking, manual effort |
| Manual entry of received items | Inconsistencies in received vs. ordered quantities |
| No expiry alert for medicines/supplies | Risk of using expired products |
| No insights into stock trends | Poor forecasting and budgeting |

1. Proposed Solution (To-Be State)  
   The “Medical Inventory Management” CRM is a Salesforce-based solution designed to automate inventory workflows and ensure secure, trackable stock management in clinical or hospital environments.

Core Solution Features:  
• Product/Medicine Object: Stores stock levels, expiry dates, and batch info  
• Purchase Order Automation: Create, manage, and track Purchase\_Order\_\_c records  
• Delivery Logging: Quantity\_Received and Actual\_Delivery\_Date tracked automatically  
• Trigger-Based Stock Update: Updates Current\_Stock\_Level\_\_c on Product\_\_c post delivery  
• Expiry/Low Stock Alerts: Dashboards flag critical items using Roll-Up and Formula Fields  
• Email Alerts: Automated emails sent on delivery status or low stock  
• Role-Based Access: Restrict views and edits for different roles (e.g., Clerk vs. Admin)  
• Dashboards/Reports: Monitor supplier trends, order frequency, stock movement

1. How the Solution Solves the Problem

|  |  |
| --- | --- |
| Problem | Feature/Function that Solves It |
| Stockout due to missed reorder | Dashboard alerts and expiry triggers |
| Manual stock logging | Record-Triggered Flows and Order\_Item\_\_c automation |
| No delivery tracking | Purchase\_Order\_\_c with status, delivery dates |
| No purchase analytics | Reports on purchase trends, supplier performance |
| Human error in calculations | Formula fields and validation rules |
| Poor access control | Profiles and Permission Sets based on roles |

1. Solution Adoption Channels  
   • Web-based Salesforce Lightning App  
   • Custom Tabs for Products, Orders, Suppliers  
   • Mobile-friendly dashboards for real-time stock checks  
   • Email alerts triggered upon delivery or purchase order creation  
   • Reports and dashboards visible only to authorized profiles
2. Solution Validation  
   The system was tested using test records and validated for:  
   • Trigger-based update of stock level post-delivery  
   • Alerts working for near-expiry and low-stock items  
   • Purchase Order and Delivery flow execution via test flows  
   • Dashboards accurately reflecting order and stock data  
   • Profiles correctly restricting Create/Edit access per role.

**4.2 Proposed Solution**

|  |  |  |
| --- | --- | --- |
| S. No. | Parameter | Description |
| 1 | Problem Statement (Problem to be solved) | Hospitals and clinics often lack a centralized, automated system to manage procurement, inventory tracking, supplier coordination, and expiry alerts. Manual tracking using spreadsheets or paper logs results in delayed replenishment, expired stock usage, inventory mismatches, and compliance risks. |
| 2 | Idea / Solution Description | A cloud-based inventory management CRM built on the Salesforce platform tailored for medical supply and pharmacy management. The solution uses custom objects such as Product (medicine/equipment), Purchase Order, Order Item, Inventory Transaction, and Supplier. Key features include:- Low stock and expiry alerts via Dashboards and Flows- Trigger-based stock updates upon delivery- Email notifications for order status- Validation rules to prevent incorrect data entries- Reports and dashboards for procurement and compliance tracking- Role-based access using Profiles and Permission Sets |
| 3 | Novelty / Uniqueness | - Integrates procurement, stock management, delivery tracking, and reporting into a single intelligent system- Specifically designed for medical inventory workflows including batch tracking, expiry monitoring, and supplier performance- Primarily built using Salesforce declarative tools (Flows, Validation Rules) to ensure ease of use and quick deployment |
| 4 | Social Impact / Customer Satisfaction | - Improves operational efficiency, ensuring hospitals never run out of critical medicines or use expired supplies- Enhances staff productivity by eliminating manual logs and redundant tracking- Encourages digital transformation in healthcare supply chain- Supports better patient care through improved inventory availability and reliability |
| 5 | Business Model (Revenue Model) | - Can be offered as a SaaS (Software as a Service) product to hospitals, pharmacies, and clinics- Tiered subscription model based on number of users, storage needs, or feature access- Optional onboarding/training/support services can be monetized for additional revenue |
| 6 | Scalability of the Solution | - Salesforce infrastructure ensures scalability across multi-location hospitals or health networks- Extendable to mobile-based apps for on-the-go inventory checks- Easily integrates with barcode scanners, pharmacy POS systems, and supplier portals- Can be adapted for other inventory-heavy domains like dental clinics or diagnostic labs |

**4.3 Solution Architecture**

**What is Solution Architecture?**  
Solution Architecture provides the technical blueprint for implementing your business requirements into a working system. It bridges the gap between healthcare inventory challenges (like stock mismanagement or delayed orders) and platform-specific tools (e.g., Salesforce Flows, Apex triggers, validation rules) used to automate and streamline operations.

It includes:

* Structure of system components (custom objects, data relationships)
* Process automation logic and flow behavior
* Technology stack used for development
* User roles and deployment strategy

**Goals of the Solution Architecture for This Project:**

* Leverage Salesforce Lightning to digitize inventory, procurement, and supplier management workflows
* Clearly define custom objects: Product, Purchase Order, Order Item, Inventory Transaction, Supplier
* Automate business processes using Flows, Triggers, and Validation Rules
* Provide secure, role-based access via Profiles and Permission Sets
* Build a scalable, automated, and audit-compliant inventory solution for hospitals or clinics

**Core Components of the Architecture**

|  |  |  |
| --- | --- | --- |
| Layer | Component | Description |
| Presentation Layer | Salesforce Lightning UI | Users (inventory staff, procurement officers, admins) interact with the system via custom Lightning App pages and layouts |
| Business Logic Layer | Record-Triggered Flows, Apex Triggers, Validation Rules | Business automation such as updating inventory after delivery, validating expected vs. actual delivery dates, sending order confirmations |
| Data Layer | Custom Objects: Product, Purchase\_Order\_\_c, Order\_Item\_\_c, Inventory\_Transaction\_\_c, Supplier\_\_c | Stores structured healthcare inventory and procurement data. Objects are linked via Lookup and Master-Detail relationships |
| Security Layer | Profiles, Permission Sets | Role-based access ensures that only authorized staff (e.g., pharmacists, storekeepers) can access specific objects or fields |
| Reporting Layer | Reports and Dashboards | Visual representations of procurement spend, order delivery status, low-stock items, and supplier performance |

📊 Data Flow within the System

**User Journey Flow:**

1. Inventory Manager identifies low stock or upcoming expiries → Dashboard notifies → Product record reviewed
2. New purchase order is initiated → Purchase\_Order\_\_c created
3. Items added → Order\_Item\_\_c records created under the Purchase Order
4. Delivery received → Actual\_Delivery\_Date\_\_c filled, Inventory\_Transaction\_\_c logged
5. Inventory automatically updated → Trigger/Flow updates Product’s Current\_Stock\_Level\_\_c
6. Email sent to stakeholders → Flow notifies supplier or admin about status updates

**Sample Architecture Diagram (Conceptual)**

[Inventory Manager]  
↓   
[Product]  
↓   
[Purchase\_Order\_\_c] ← Lookup to Supplier  
↓ (Master-Detail)   
[Order\_Item\_\_c] ← Lookup to Product  
↓   
[Inventory\_Transaction\_\_c] ← Created when delivery is logged  
↓ (Flow/Trigger)   
[Product stock auto-updated]  
↓ (Flow)   
[Email sent to Supplier/Admin]

**Summary**

The Medical Inventory Management architecture enables:

* Streamlined procurement and stock tracking
* Automation of delivery updates and inventory adjustments
* Improved supplier communication via email automation
* Secure, role-based data handling
* Scalable system fit for multi-department or multi-location healthcare institutions

**5. PROJECT PLANNING & SCHEDULING**

**5.1 Project Planning**

* **Agile Overview: Key Concepts**

|  |  |
| --- | --- |
| Term | Description |
| Sprint | A 5-day time-boxed iteration to deliver prioritized features. |
| Epic | A large function (e.g., Procurement Flow) that spans multiple sprints. |
| User Story | A user-centric feature that adds value; breakdown of an Epic. |
| Story Points | Complexity estimation using Fibonacci (1, 2, 3, 5...)—reflects effort, not time. |

* Sprint Planning Table – 5 Days Per Sprint
* Sprint 1 – Inventory Setup & Data Load

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Day | Task | Story Points | Type | Notes |
| 1 | Collect medical product & supplier data | 2 | Data Sourcing | From spreadsheets and vendor files |
| 2 | Load data into Product and Supplier objects | 1 | Configuration | Data Import Wizard |
| 3 | Set up custom fields and relationships | 3 | Object Modeling | Lookup fields between objects |
| 4 | Configure picklists (Category, Unit Type) | 2 | Setup | Required for inventory classification |
| 5 | Sprint Review + Bug Fixes | - | QA | Review configuration and fix errors |
|  | Total Story Points (Sprint 1) | 8 Points |  |  |

* Sprint 2 – Procurement Automation & Flow Design

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Day** | **Task** | **Story Points** | **Type** | **Notes** |
| 1 | Create custom objects (Purchase Order, Order Item) | 5 | Object Creation | With relationships to Product and Supplier |
| 2 | Design Lightning Record Pages | 3 | UI/UX | Tabs, components, and layouts |
| 3 | Build Flow to auto-update stock on delivery | 3 | Automation | Stock level increase on Quantity\_Received |
| 4 | Create Flow to notify supplier upon PO submission | 5 | Flow | Email confirmation using Email Templates |
| 5 | Sprint Review & Test | - | QA | Validate triggers and flows |
|  | Total Story Points (Sprint 2) | 16 Points |  |  |

* Velocity Calculation

|  |  |
| --- | --- |
| **Metric** | **Value** |
| Story Points in Sprint 1 | 8 Points |
| Story Points in Sprint 2 | 16 Points |
| Total Points | 24 Points |
| Number of Sprints | 2 |
| Velocity | 24 ÷ 2 = 12 Points/Sprint |

* Sprint Status Summary

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sprint** | **Duration (Days)** | **Points Planned** | **Points Completed** | **Completion %** | **Remarks** |
| Sprint 1 | 5 | 8 | 8 | 100% | Data modeled and loaded |
| Sprint 2 | 5 | 16 | 16 | 100% | Automation for procurement completed |

* Product Backlog, Sprint Schedule & Estimation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sprint** | **Functional Requirement (Epic)** | **User Story No.** | **User Story / Task** | **Story Points** | **Priority** | **Team Member** |
| Sprint-1 | Product Setup | USM-1 | As an admin, I can create product and supplier records with proper fields. | 3 | High | Member 1 |
| Sprint-1 | Inventory Baseline | USM-2 | As a manager, I can upload medical product inventory from Excel. | 2 | High | Member 2 |
| Sprint-1 | Categorization | USM-3 | As a user, I can see item categories (e.g., surgical, diagnostic). | 1 | Medium | Member 3 |
| Sprint-1 | Supplier Relationships | USM-4 | As a user, I can link suppliers to products. | 2 | High | Member 4 |
| Sprint-2 | Purchase Order Management | USM-5 | As a user, I can create POs and view order details. | 4 | High | Member 1 |
| Sprint-2 | Delivery Logging | USM-6 | As a user, I can enter delivery date and received quantity. | 3 | High | Member 2 |
| Sprint-2 | Inventory Automation | USM-7 | As a system, I update stock automatically when delivery is logged. | 5 | High | Member 3 |
| Sprint-2 | Communication Flow | USM-8 | As a user, I receive confirmation when a PO is submitted. | 3 | Medium | Member 4 |

* Project Tracker & Velocity

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint | Total Story Points | Duration | Start Date | End Date | Points Completed | Release Date |
| Sprint 1 | 8 | 5 Days | 20 Feb 2025 | 24 Feb 2025 | 8 | 24 Feb 2025 |
| Sprint 2 | 16 | 5 Days | 25 Feb 2025 | 29 Feb 2025 | 16 | 29 Feb 2025 |

* Velocity = (8 + 16) / 2 = 12 Story Points per Sprint
* Burndown Chart (Sprint 2 Example)

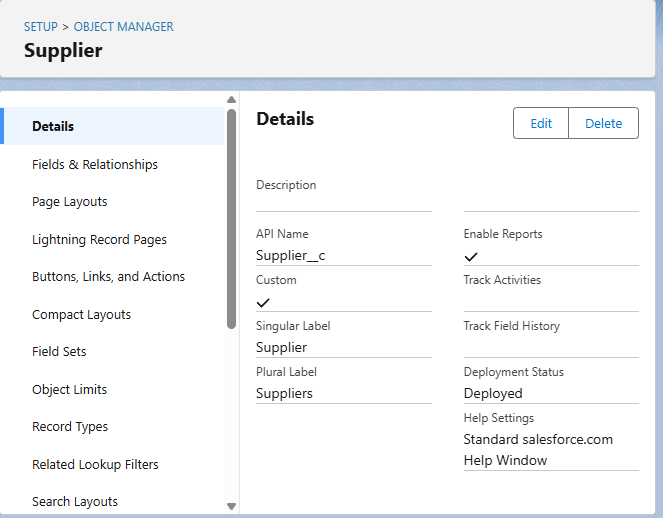
|  |  |  |
| --- | --- | --- |
| **Day** | **Remaining Story Points (Ideal)** | **Actual** |
| Day 0 | 16 | 16 |
| Day 1 | 13 | 14 |
| Day 2 | 10 | 10 |
| Day 3 | 7 | 7 |
| Day 4 | 3 | 3 |
| Day 5 | 0 | 0 |

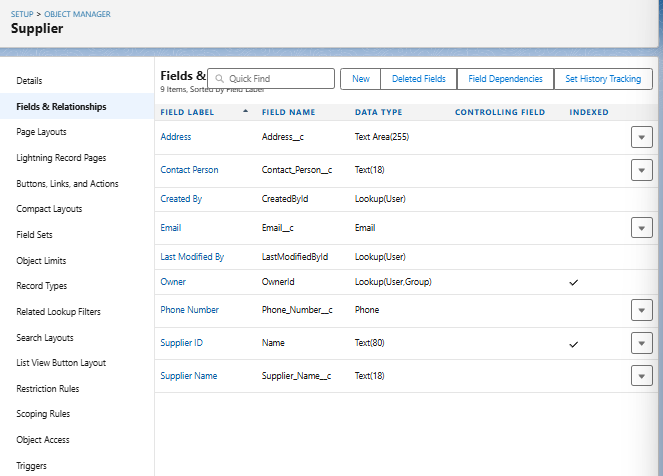
**6. Project Development Phase**

### A. Custom Objects and Their Roles

#### 1. Supplier\_\_c

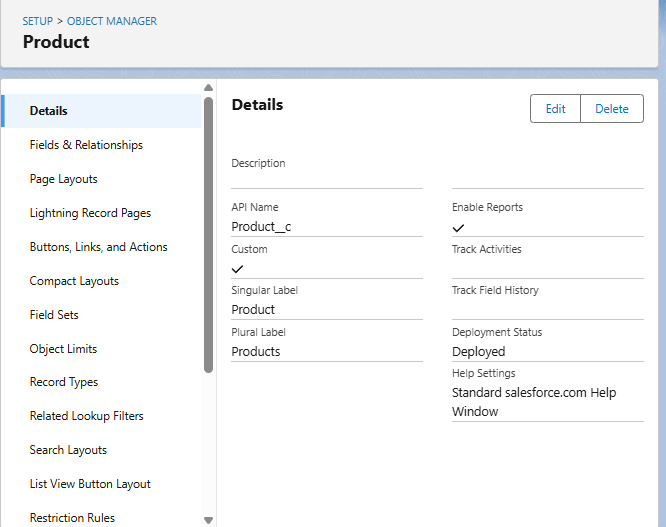
* **Purpose**: Stores comprehensive information about medical product suppliers, including their name, contact person, phone number, email address, physical address, and any specific terms or notes related to their partnership.
* **Usage**: This object serves as the central point for all supplier-related data, enabling efficient management of procurement and communication. It is critical for maintaining accurate records of where medical products are sourced.
* **Key Fields**:
  + Supplier\_Name\_\_c (Text)
  + Contact\_Person\_\_c (Text)
  + Phone\_Number\_\_c (Phone)
  + Email\_\_c (Email)
  + Address\_\_c (Text Area)

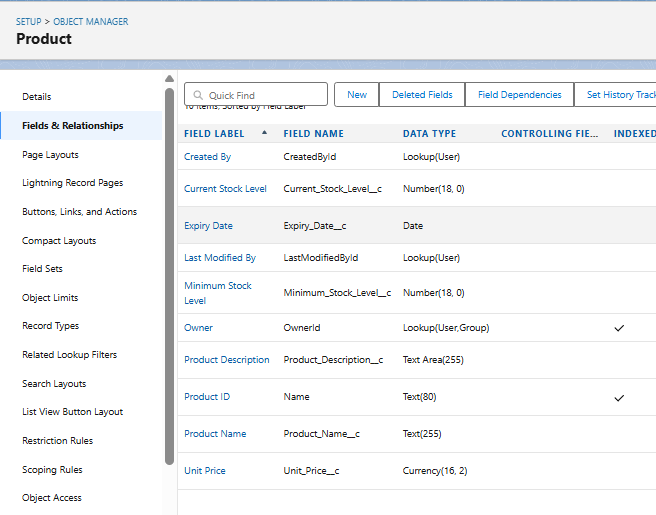
****

****

#### 2. Product\_\_c

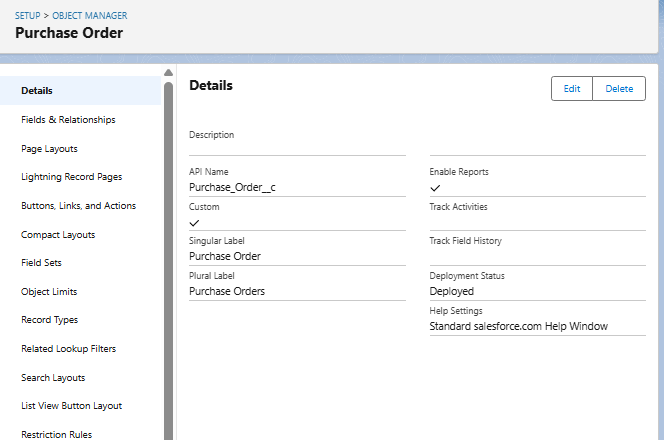
* **Purpose**: Stores detailed information about each medical product in the inventory. This includes crucial attributes like product name, description, unique identifiers, current stock levels, minimum reorder levels, unit price, and vital expiry dates.
* **Usage**: This object is fundamental for tracking all individual items within the medical inventory. It allows for efficient stock management, helps prevent stockouts, and ensures that expired products are identified and managed appropriately.
* **Key Fields**:
  + Product\_Name\_\_c (Text)
  + Product\_Description\_\_c (Text Area)
  + Minimum\_Stock\_Level\_\_c (Number)
  + Current\_Stock\_Level\_\_c (Number)
  + Unit\_Price\_\_c (Currency)
  + Expiry\_Date\_\_c (Date)

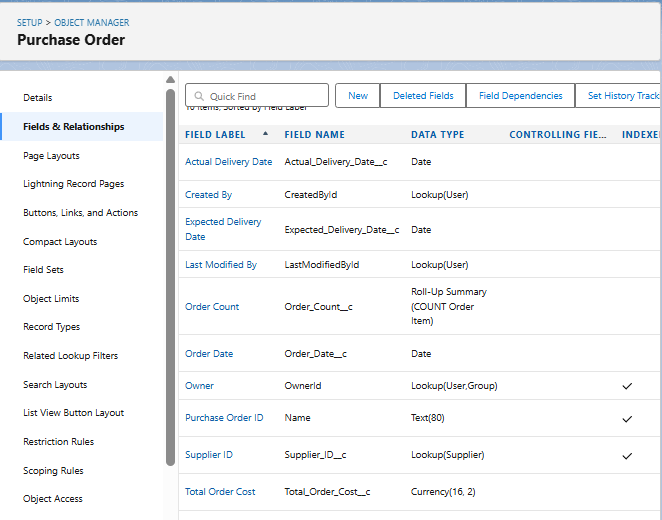




#### 3. Purchase\_Order\_\_c

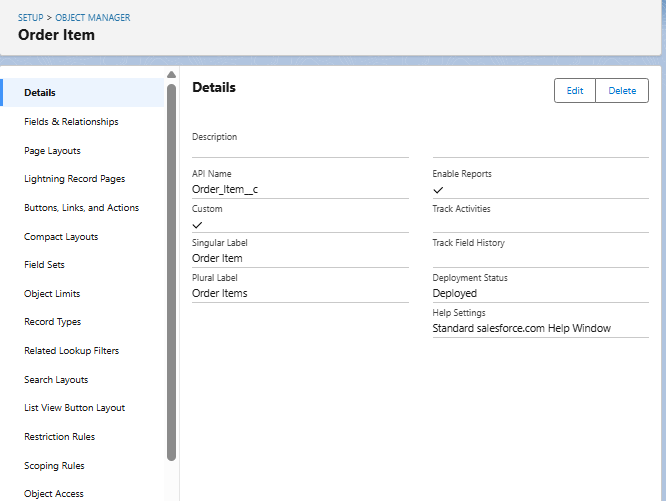
* **Purpose**: Represents a formal request made to a Supplier\_\_c for a specific quantity of Product\_\_cs. It captures details such as the order date, expected and actual delivery dates, the associated supplier, the total cost of the order, and the number of distinct items ordered.
* **Usage**: This object is essential for managing the procurement process. It allows for tracking the status of incoming inventory, linking orders to specific suppliers, and providing an overview of pending and completed purchases.
* **Key Fields**:
  + Supplier\_ID\_\_c (Lookup to Supplier\_\_c)
  + Order\_Date\_\_c (Date)
  + Expected\_Delivery\_Date\_\_c (Date)
  + Actual\_Delivery\_Date\_\_c (Date)
  + Order\_Count\_\_c (Roll-Up Summary, COUNT of Order\_Item\_\_c)
  + Total\_Order\_Cost\_\_c (Currency)

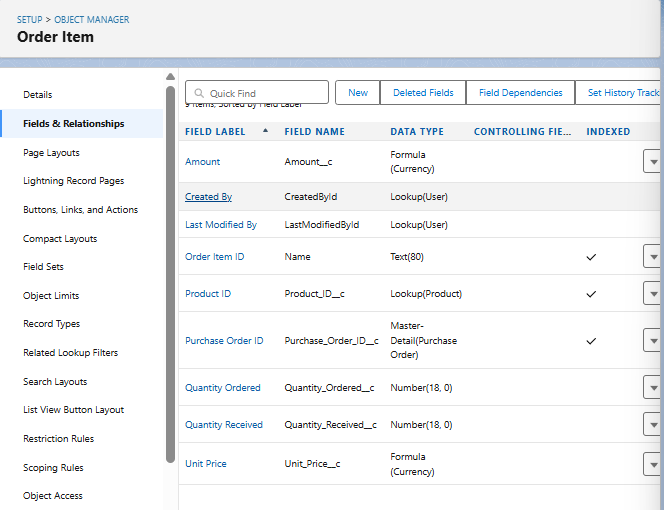




#### 4. Order\_Item\_\_c

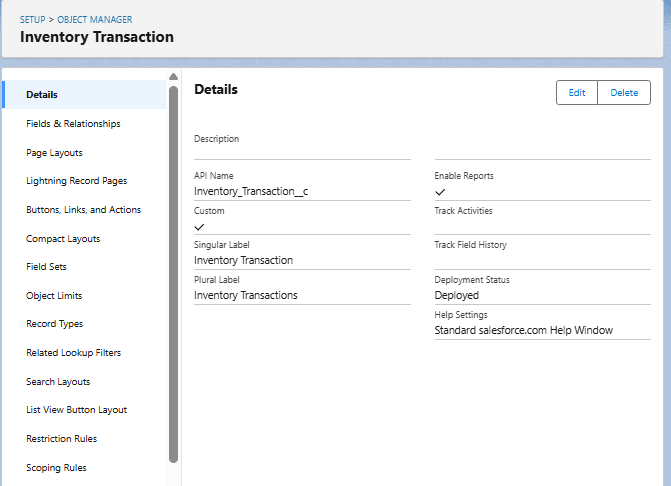
* **Purpose**: Represents a specific line item within a Purchase\_Order\_\_c. Each record details a particular Product\_\_c being ordered, the quantity requested, the quantity received, and the calculated amount for that item.
* **Usage**: This object forms the bridge between Purchase\_Order\_\_c and Product\_\_c, breaking down a single purchase order into its constituent products. It's crucial for granular tracking of what products are part of each order and their individual quantities and costs.
* **Key Fields**:
  + Product\_ID\_\_c (Lookup to Product\_\_c)
  + Purchase\_Order\_ID\_\_c (Master-Detail to Purchase\_Order\_\_c)
  + Quantity\_Ordered\_\_c (Number)
  + Quantity\_Received\_\_c (Number)
  + Unit\_Price\_\_c (Formula, Currency)
  + Amount\_\_c (Formula, Currency)

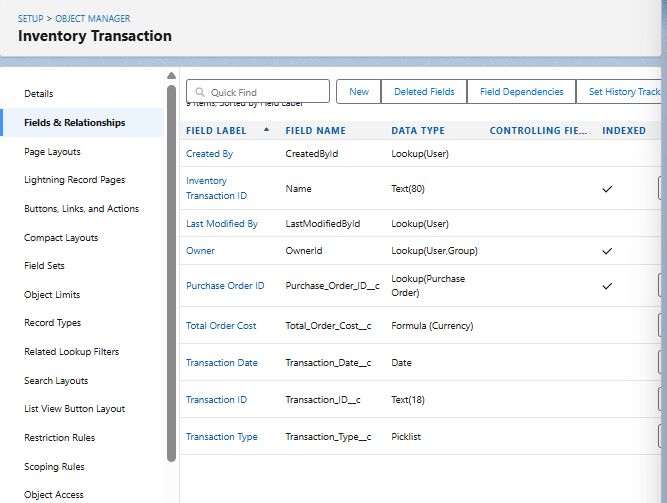




#### 5. Inventory\_Transaction\_\_c

* **Purpose**: Records all movements and adjustments of products within the inventory. This includes various transaction types such as receiving stock (from purchase orders), dispensing items, returns, or adjustments due to spoilage/damage. It links back to the relevant Purchase\_Order\_\_c if applicable and captures the date and type of transaction.
* **Usage**: This object provides an audit trail for all changes to product stock levels, allowing for accurate inventory reconciliation, identifying discrepancies, and understanding the flow of products in and out of storage.
* **Key Fields**:
  + Purchase\_Order\_ID\_\_c (Lookup to Purchase\_Order\_\_c)
  + Transaction\_Date\_\_c (Date)
  + Transaction\_Type\_\_c (Picklist - e.g., 'Received', 'Dispensed', 'Adjustment')
  + Total\_Order\_Cost\_\_c (Formula, Currency - Note: This field might be more relevant to a Purchase Order. For an inventory transaction, a 'Quantity\_Changed' or 'Value\_Changed' might be more appropriate depending on transaction type. Reconfirm if Total\_Order\_cost\_\_c makes sense here, perhaps it refers to the cost of the items in this specific transaction.)





**B. AUTOMATION ELEMENTS**

The Medical Inventory Management System leverages Salesforce's automation capabilities to streamline business processes, reduce manual interventions, and enhance operational accuracy. The automation comprises Apex Triggers, Record-Triggered Flows, and Validation Rules, all tailored to meet the critical needs of a medical inventory and procurement operations.

**1. Apex Trigger –**

* **Name:**CalculateTotalAmountTrigger
* **Trigger Type:**after insert, after update, after delete, after undelete
* **Description:**  
  This use case works for Amount Distribution for each Service the customer selected for the Vehicle
* **Core Logic:**

trigger CalculateTotalAmountTrigger on Order\_Item\_\_c (after insert, after update, after delete, after undelete) {

CalculateTotalAmountHandler.calculateTotal(Trigger.new, Trigger.old, Trigger.isInsert, Trigger.isUpdate, Trigger.isDelete, Trigger.isUndelete);

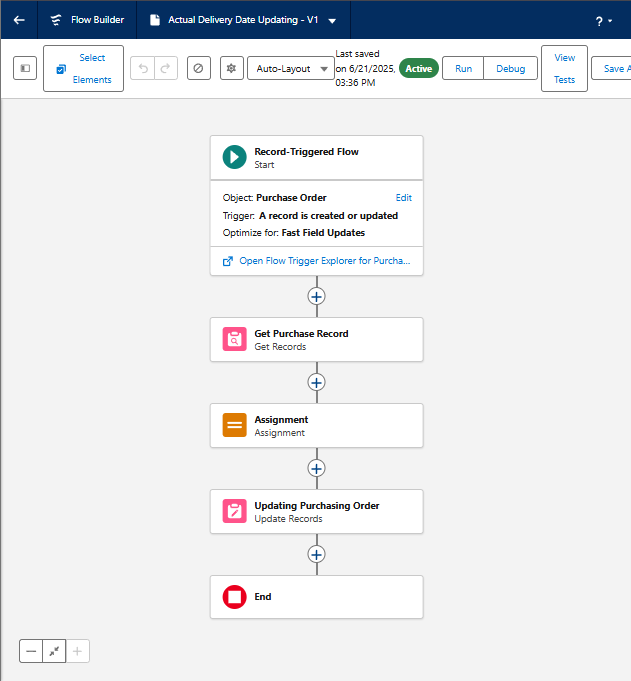
}

### 2. Flow Name: Actual\_Delivery\_Date\_Updating

* **Purpose**: This Record-Triggered Flow automates the initial estimation of the Actual\_Delivery\_Date\_\_c on a Purchase\_Order\_\_c record. Upon the creation or update of a Purchase Order, the flow automatically calculates and populates this field to be 3 days after the Order\_Date\_\_c. This provides an immediate, intelligent default for the actual delivery date, streamlining data entry and offering a preliminary timeline for incoming inventory.
* **Associated Object**: Purchase\_Order\_\_c
* **Trigger Event**: A record is **created or updated**.
* **Optimization for Related Records**: Fast Field Updates (The flow runs before the record is saved to the database).

#### Step-by-Step Flow Logic:

1. **Start Element (Trigger)**:
   * **Object**: Purchase\_Order\_\_c
   * **Trigger the Flow When**: A record is created or updated.
   * **Entry Conditions**: None (The flow should run for all new or updated Purchase Orders).
   * **Optimize the Flow for**: Fast Field Updates (This ensures the Actual\_Delivery\_Date\_\_c is updated before the record is officially committed, making it ideal for updating fields on the triggering record itself).
2. **Assignment Element**:
   * **Label**: Assignment (or more descriptive: Set Actual Delivery Date)
   * **Purpose**: This element calculates the new Actual\_Delivery\_Date\_\_c value by adding 3 days to the Order\_Date\_\_c.
   * **Variables/Fields Set**:
     + **Variable**: A Date type variable (e.g., vActualDeliveryDate)
     + **Operator**: Equals
     + **Value**: {!$Record.Order\_Date\_\_c} + 3
     + (Note: While your milestone used a variable ActualDeliveryDate and then an update element, for a "Fast Field Update" flow updating the same record, you can directly update the record's field without an explicit "Get Records" if the field is on the triggering record itself. However, sticking to your provided logic for clarity based on your Milestone 13 helps.)
3. **Update Records Element**:
   * **Label**: Updating Purchase Order (or more descriptive: Update Actual Delivery Date)
   * **Purpose**: This element takes the calculated Actual\_Delivery\_Date\_\_c and applies it to the Purchase\_Order\_\_c record that initiated the flow.
   * **How to Find Records to Update and Set Their Values**: Use the Purchase\_Order\_\_c record that triggered the flow.
   * **Set Field Values for the Purchase\_Order\_\_c Record**:
     + **Field**: Actual\_Delivery\_Date\_\_c
     + **Value**: {!vActualDeliveryDate} (Referring to the variable from the Assignment step).
4. **Save and Activate**:
   * Save the flow with the name Actual Delivery Date Updating.
   * Activate the flow to ensure it runs automatically whenever a Purchase\_Order\_\_c record is created or updated.

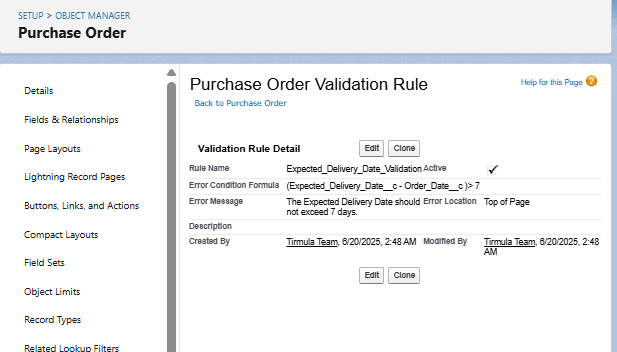


## 3. Validation Rules

Validation Rules are essential for maintaining data quality and consistency within the Medical Inventory Management System. They enforce business logic by preventing users from saving records that do not meet specified criteria.

### Validation Rule Name: Expected\_Delivery\_Date\_Validation

* **Associated Object**: Purchase\_Order\_\_c
* **Purpose**: This Validation Rule ensures that the Expected\_Delivery\_Date\_\_c on a Purchase\_Order\_\_c record is set within a reasonable timeframe relative to the Order\_Date\_\_c. Specifically, it prevents the expected delivery date from being more than 7 days after the order date. This helps in managing supplier expectations and ensuring timely procurement of medical supplies.
* **Formula Logic**:
* (Expected\_Delivery\_Date\_\_c - Order\_Date\_\_c) > 7
  + **Explanation**: This formula checks if the number of days between the Expected\_Delivery\_Date\_\_c and the Order\_Date\_\_c is greater than 7. If this condition evaluates to TRUE, the validation rule fires, preventing the record from being saved.
* **Error Message**: "The Expected Delivery Date should not exceed 7 days."
* **Error Location**: Top of Page
* **Usage Scenario**:
  + When a user creates a new Purchase\_Order\_\_c record or modifies an existing one, they are required to input an Order\_Date\_\_c and an Expected\_Delivery\_Date\_\_c.
  + If the user enters an Expected\_Delivery\_Date\_\_c that is, for example, 10 days after the Order\_Date\_\_c, this validation rule will trigger.
  + The system will display the error message "The Expected Delivery Date should not exceed 7 days." at the top of the record page, preventing the user from saving the invalid data.
  + The user must then adjust the Expected\_Delivery\_Date\_\_c to be within 7 days of the Order\_Date\_\_c to successfully save the record.



**C. UI COMPONENTS**

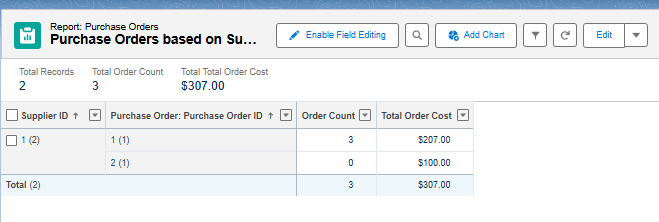
**Reports Configuration**

**Use Case:**

Reports in the Medical Inventory Management System provide valuable insights into supplier performance, purchase orders, and overall inventory status. They are essential tools for tracking key metrics and enabling data-driven decision-making.

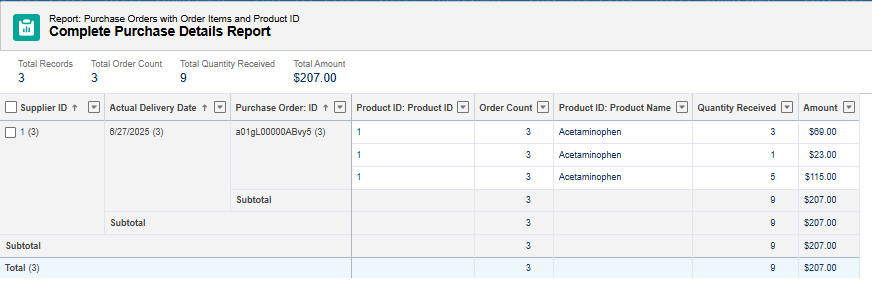
**Activity 1: Create a "Purchase Orders based on Suppliers" Report**

1. **Navigate to Reports**:
   * From App Launcher, select Medical Inventory Management App.
   * Click on the Reports tab.
   * Click New Report.
2. **Select Report Type**: Choose Purchase Orders. Click Start Report.
3. **Configure Filters**:
   * Show Me: All Purchase Orders.
   * Date Field: Order Date.
   * Range: All Time. Click Apply.
4. **Add Groupings & Columns**:
   * **Group Rows**: Supplier ID, Purchase Order: Purchase Order ID.
   * **Columns**: Order Count, Total Order Cost.
5. **Save Report**:
   * Click Save & Run.
   * **Report Name**: Purchase Orders based on Suppliers.
   * Click Save.



**Activity 2: Create a "Complete Purchase Details Report"**

1. **Navigate to Reports**:
   * From App Launcher, select Medical Inventory Management App.
   * Click on the Reports tab.
   * Click New Report.
2. **Select Report Type**: Choose Purchase Orders with Order Items and Product ID. Click Start Report.
3. **Configure Filters**:
   * Show Me: All Purchase Orders.
   * Order Item Date: All Time. Click Apply.
4. **Add Groupings & Columns**:
   * **Group Rows**: Supplier ID, Actual Delivery Date, Purchase Order: Purchase Order ID.
   * **Columns**: Product ID: Product ID, Product ID: Product Name, Order Item: Quantity Ordered, Order Item: Quantity Received, Order Item: Amount.
5. **Save Report**:
   * Click Save & Run.
   * **Report Name**: Complete Purchase Details Report.
   * Click Save.

****

**Dashboards**

**Use Case:**

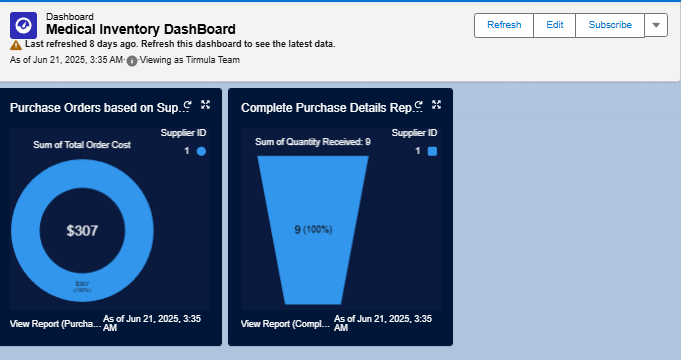
Dashboards provide visual representations of key data from your reports, offering a quick overview of critical metrics within the Medical Inventory Management System.

**Activity 1: Create "Medical Inventory DashBoard"**

1. **Navigate to Dashboards**:
   * From the App Launcher, select the Medical Inventory Management application.
   * Click on the Dashboards tab.
2. **Create New Dashboard**:
   * Click New Dashboard.
   * **Dashboard Name**: Medical Inventory DashBoard.
   * Click Create.
3. **Add Widget (Component)**:
   * Click +Widget.
   * Select the report: Purchase Orders based on Suppliers.
   * Choose a desired chart type (e.g., Bar Chart, Pie Chart) or table for data visualization.
   * Click Add.
4. **Save Dashboard**:
   * Click Save.

**Activity 2: View Dashboard**

1. **Navigate to Dashboards**:
   * From the App Launcher, search for and select the Medical Inventory Management App.
   * Click on the Dashboards Tab.
2. **Open Dashboard**:
   * Click on Medical Inventory DashBoard from the list to view its graphical representation of records.



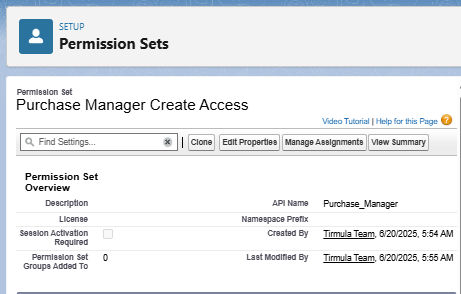
**Users and Permission Set Management**

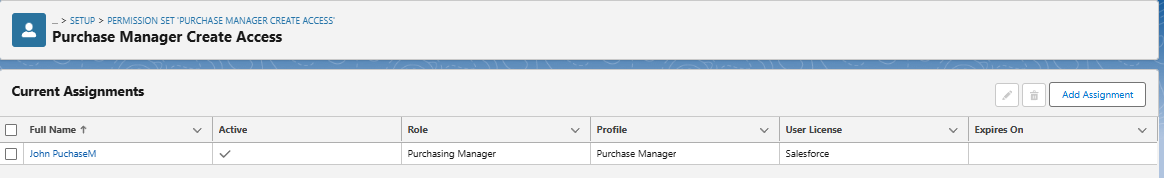
**Permission Sets:**

Permission Sets are used to grant additional permissions and access settings to users, extending their capabilities beyond what is defined in their assigned profile. This provides a flexible way to manage user access without cloning profiles unnecessarily.

**Activity 1: Create "Purchase Manager Create Access" Permission Set**

1. **Navigate to Permission Sets**:
   * From Setup, enter Permission Sets in the Quick Find box.
   * Select Permission Sets.
   * Click New.
2. **Define Permission Set**:
   * **Label**: Purchase Manager Create Access
   * Click Save.
3. **Configure Object Permissions (Order Item)**:
   * From the Permission Set detail page, click Object Settings.
   * Select Order Item.
   * Click Edit.
   * Enable Tab Available and Visible for Order Item.
   * Enable Read and Create under Object Permissions for Order Item.
   * Click Save.
4. **Assign to User**:
   * From the Permission Set detail page, click Manage Assignments.
   * Click Add Assignments.
   * Select the user John PurchaseM.
   * Click Next.
   * Select No Expiration Date.
   * Click Assign.

****

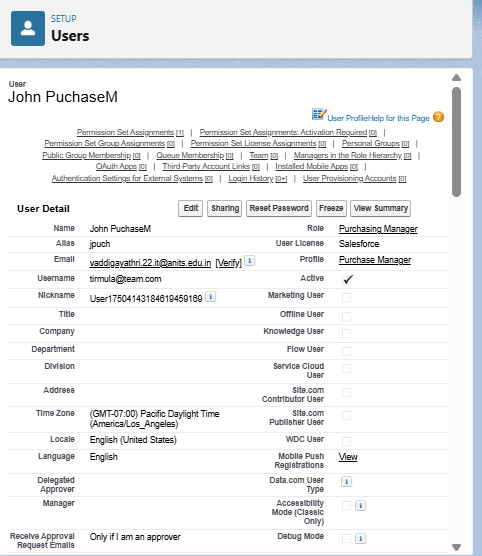
****

**Users:**

Users are the individuals who log into your Salesforce organization to work with the Medical Inventory Management System. Each user requires a unique username, email address, user license, profile, and optionally a role.

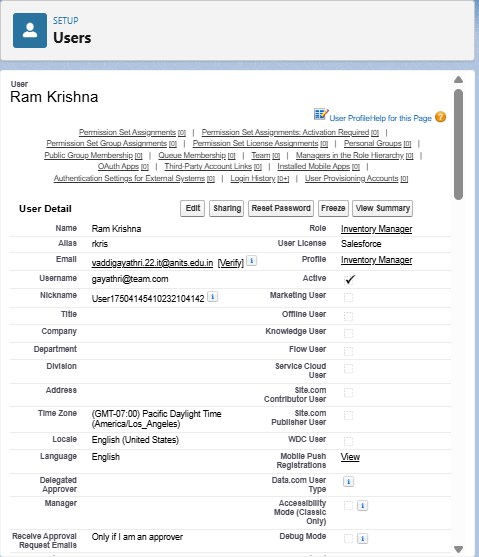
**Activity 1: Create "Purchasing Manager" User**

1. **Navigate to Users**:
   * From Setup, enter Users in the Quick Find box.
   * Select Users.
   * Click New User.
2. **Enter User Details**:
   * **First Name**: John
   * **Last Name**: PurchaseM
   * **Email**: (Your email address)
   * **Username**: (Unique username, e.g., john.purchasem@yourcompany.com)
   * **Role**: Purchasing Manager
   * **User License**: Salesforce
   * **Profile**: Purchase Manager
   * Check Generate new password and notify the user immediately.
3. **Save User**:
   * Click Save.



**Activity 2: Create "Inventory Manager" User**

1. **Navigate to Users**:
   * From Setup, enter Users in the Quick Find box.
   * Select Users.
   * Click New User.
2. **Enter User Details**:
   * **First Name**: Ram
   * **Last Name**: Krishna
   * **Email**: (Your email address)
   * **Username**: (Unique username, e.g., ram.krishna@yourcompany.com)
   * **Role**: Inventory Manager
   * **User License**: Salesforce
   * **Profile**: Inventory Manager
   * Check Generate new password and notify the user immediately.
3. **Save User**:
   * Click Save.



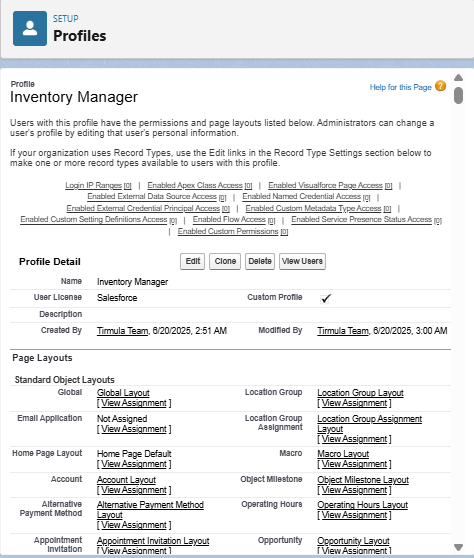
**D. Implementation Activities**

## Profiles

Profiles define a user's permissions for performing different operations in Salesforce, such as creating records, editing records, viewing tabs, and accessing specific apps. They are a fundamental part of controlling data access and user capabilities.

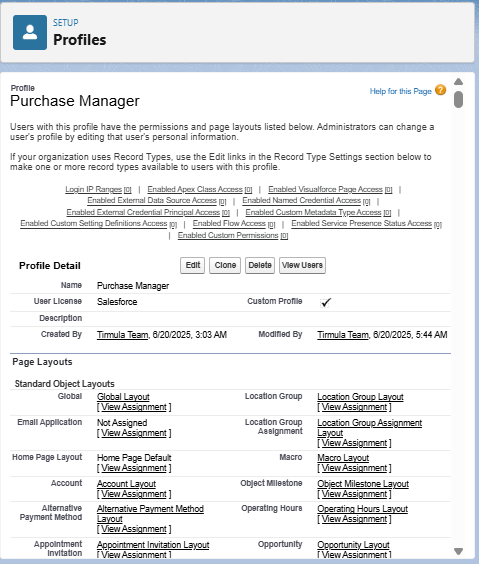
### Activity 1: Create "Inventory Manager" Profile

1. **Navigate to Profiles**:
   * From Setup, enter Profiles in the Quick Find box.
   * Select Profiles.
2. **Clone Standard User Profile**:
   * Find and click Standard User profile.
   * Click Clone.
   * **Profile Name**: Inventory Manager
   * Click Save.
3. **Edit Profile Settings**:
   * While on the Inventory Manager profile page, click Edit.
   * **Custom App Settings**: Set Medical Inventory Management App to Default.
   * **Custom Object Permissions**: Configure permissions for custom objects relevant to an Inventory Manager.
     + For Product\_\_c: Read, Create, Edit, Delete (as needed for managing products)
     + For Inventory\_Transaction\_\_c: Read, Create, Edit, Delete (for managing stock movements)
     + For Purchase\_Order\_\_c: Read (may not need create/edit if Purchase Manager handles it)
     + For Order\_Item\_\_c: Read (similarly)
     + For Supplier\_\_c: Read (similarly)
   * **Password Policies**:
     + User passwords expire in: Never expires
     + Minimum password length: 8
   * Click Save.



### Activity 2: Create "Purchase Manager" Profile

1. **Navigate to Profiles**:
   * From Setup, enter Profiles in the Quick Find box.
   * Select Profiles.
2. **Clone Standard User Profile**:
   * Find and click Standard User profile.
   * Click Clone.
   * **Profile Name**: Purchase Manager
   * Click Save.
3. **Edit Profile Settings**:
   * While on the Purchase Manager profile page, click Edit.
   * **Custom App Settings**: Set Medical Inventory Management App to Default.
   * **Custom Object Permissions**: Configure permissions for custom objects relevant to a Purchase Manager.
     + For Purchase\_Order\_\_c: Read, Create, Edit, Delete (for managing purchase orders)
     + For Order\_Item\_\_c: Read, Create, Edit, Delete (for managing items within orders)
     + For Supplier\_\_c: Read, Create, Edit, Delete (for managing supplier details)
     + For Product\_\_c: Read (for viewing product details when creating orders)
     + For Inventory\_Transaction\_\_c: Read (for viewing transaction history)
   * **Password Policies**:
     + User passwords expire in: Never expires
     + Minimum password length: 8
   * Click Save.



**Role Hierarchy and Record-Level Access Control**

## Roles

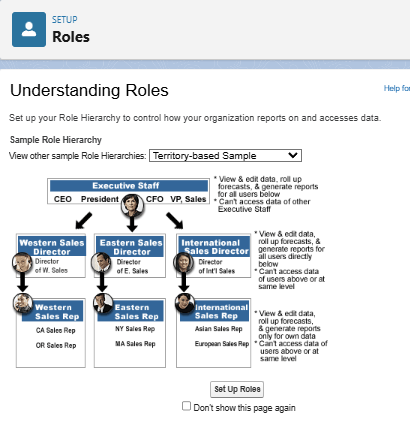
Roles in Salesforce are used to control the level of visibility users have to an organization's data based on their position in the role hierarchy. Users at higher levels in the hierarchy can view, edit, and report on all data owned by or shared with users below them in the hierarchy.

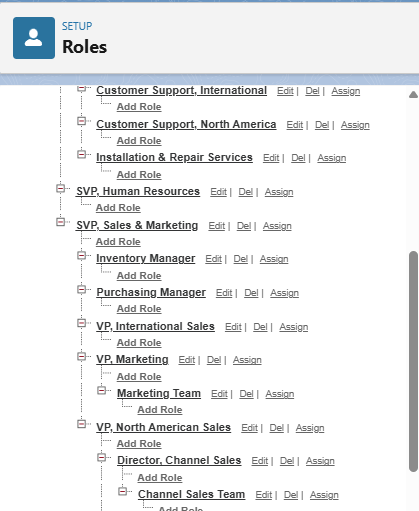
### Activity 1: Create "Purchasing Manager" Role

1. **Navigate to Roles**:
   * From Setup, enter Roles in the Quick Find box.
   * Click Set Up Roles.
2. **Add Role to Hierarchy**:
   * Click Expand All.
   * Locate SVP, Sales & Marketing (or a suitable parent role in your org structure).
   * Click Add Role beneath it.
3. **Define Role Details**:
   * **Label**: Purchasing Manager
   * (Role Name will auto-populate)
   * Click Save.

### Activity 2: Create "Inventory Manager" Role

1. **Navigate to Roles**:
   * From Setup, enter Roles in the Quick Find box.
   * Click Set Up Roles.
2. **Add Role to Hierarchy**:
   * Click Expand All.
   * Locate SVP, Sales & Marketing (or the same parent role as Purchasing Manager for peer relationship, or a different one if a different hierarchy is desired).
   * Click Add Role beneath it.
3. **Define Role Details**:
   * **Label**: Inventory Manager
   * (Role Name will auto-populate)
   * Click Save.



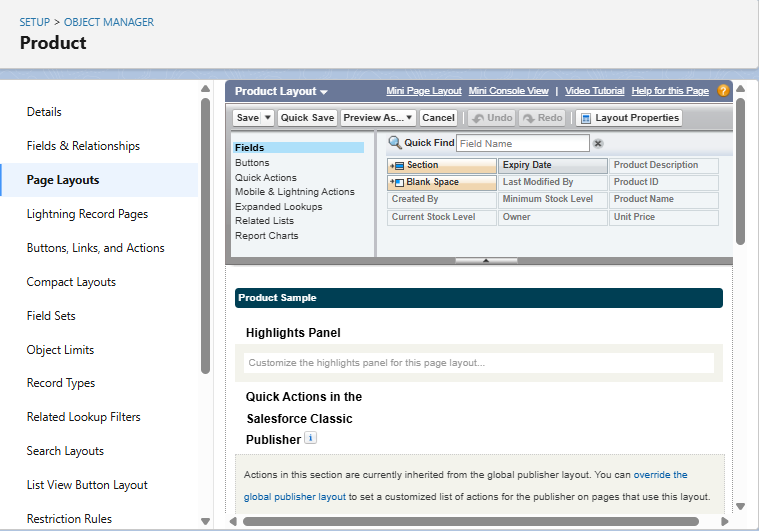


## E. Page Layouts

Page Layouts control the arrangement of fields, custom links, and related lists on record detail and edit pages in Salesforce. They help present information clearly and efficiently to users.

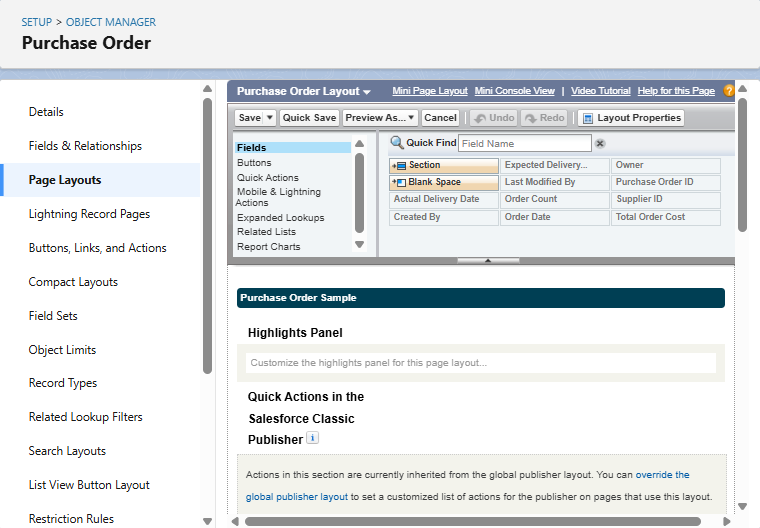
### Activity 1: Edit "Product" Object Page Layout

1. **Navigate to Object Manager**:
   * From Setup, click Object Manager.
   * In the Quick Find box, type Product and click on the Product object.
2. **Access Page Layouts**:
   * Click Page Layouts in the sidebar.
   * Click on Product Layout.
3. **Arrange Fields**:
   * Drag and arrange fields (e.g., Product Name, Product Description, Unit Price, Minimum Stock Level, Current Stock Level, Expiry Date) into a logical order on the layout.
   * Click Save.



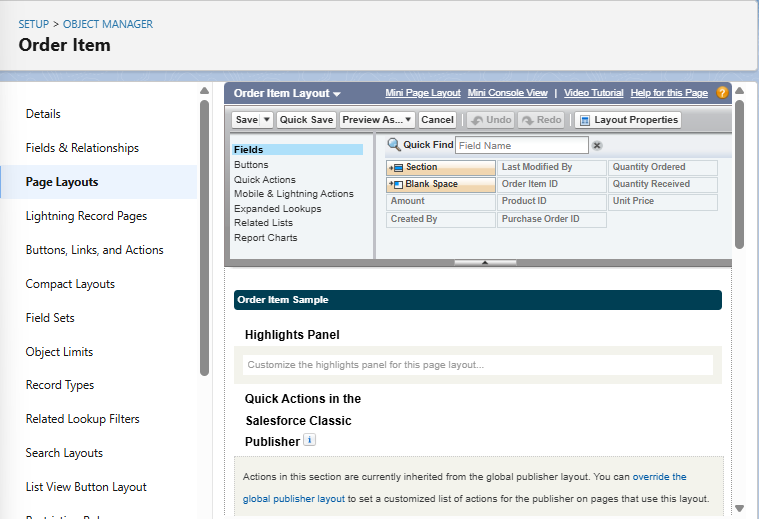
### Activity 2: Edit "Purchase Order" Object Page Layout

1. **Navigate to Object Manager**:
   * From Setup, click Object Manager.
   * In the Quick Find box, type Purchase Order and click on the Purchase Order object.
2. **Access Page Layouts**:
   * Click Page Layouts in the sidebar.
   * Click on Purchase Order Layout.
3. **Arrange Fields and Configure Properties**:
   * Drag and arrange fields (e.g., Supplier ID, Order Date, Expected Delivery Date, Actual Delivery Date, Order Count, Total Order Cost) into a logical order.
   * **Configure Field Properties**:
     + Click the wrench icon (Settings) next to Order Date. Select Required. Click OK.
     + Click the wrench icon (Settings) next to Total Order Cost. Select Read Only. Click OK.
   * Click Save.



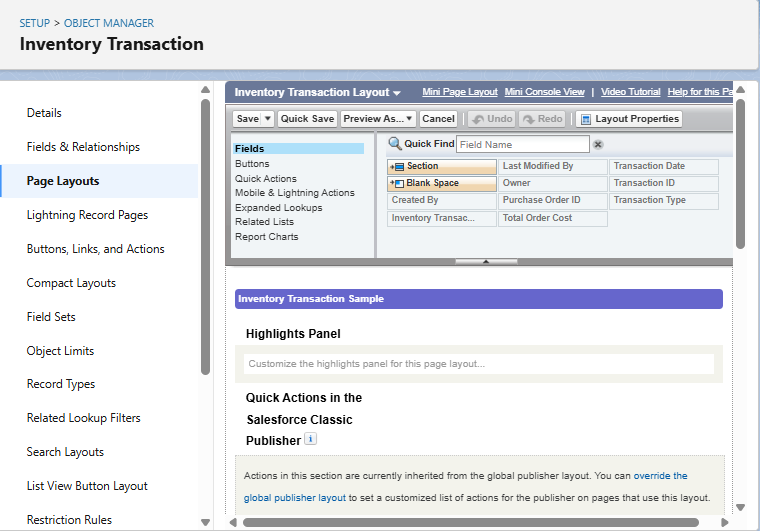
### Activity 3: Edit "Order Item" Object Page Layout

1. **Navigate to Object Manager**:
   * From Setup, click Object Manager.
   * In the Quick Find box, type Order Item and click on the Order Item object.
2. **Access Page Layouts**:
   * Click Page Layouts in the sidebar.
   * Click on Order Item Layout.
3. **Arrange Fields**:
   * Drag and arrange fields (e.g., Product ID, Purchase Order ID, Quantity Ordered, Quantity Received, Unit Price, Amount) into a logical order.
   * Click Save.



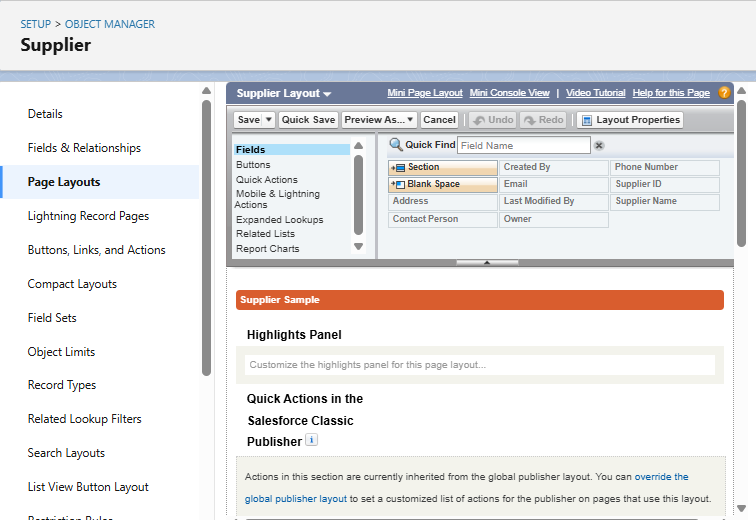
### Activity 4: Edit "Inventory Transaction" Object Page Layout

1. **Navigate to Object Manager**:
   * From Setup, click Object Manager.
   * In the Quick Find box, type Inventory Transaction and click on the Inventory Transaction object.
2. **Access Page Layouts**:
   * Click Page Layouts in the sidebar.
   * Click on Inventory Transaction Layout.
3. **Arrange Fields**:
   * Drag and arrange fields (e.g., Purchase Order ID, Transaction Date, Transaction Type, Total Order Cost - *note: ensure this field's relevance for this object's purpose as discussed before*) into a logical order.
   * Click Save.



### Activity 5: Edit "Supplier" Object Page Layout

1. **Navigate to Object Manager**:
   * From Setup, click Object Manager.
   * In the Quick Find box, type Supplier and click on the Supplier object.
2. **Access Page Layouts**:
   * Click Page Layouts in the sidebar.
   * Click on Supplier Layout.
3. **Arrange Fields**:
   * Drag and arrange fields (e.g., Supplier Name, Contact Person, Phone Number, Email, Address) into a logical order.
   * Click Save.

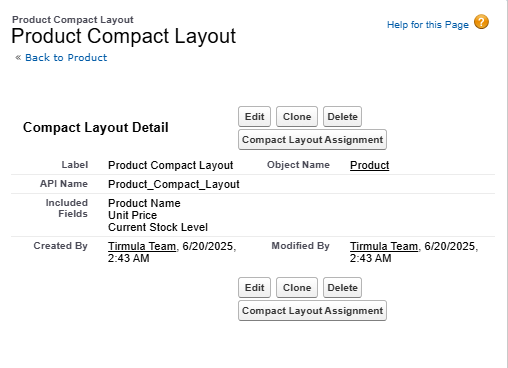


**F.Compact Layouts**

Compact Layouts control the fields that appear in the highlight panel at the top of a record page in the Salesforce mobile app and Lightning Experience, as well as in record previews. They are essential for providing a quick glance at key information for each record.

### Activity 1: Create Compact Layout for "Product" Object

1. **Navigate to Product Object**:
   * From Setup, click Object Manager.
   * In the Quick Find box, type Product and click on the Product object.
2. **Create New Compact Layout**:
   * Click Compact Layouts in the sidebar.
   * Click New.
   * **Label**: Product Compact Layout
   * **Selected Fields**: Select Product Name, Unit Price, Current Stock Level.
   * Click Save.
3. **Assign as Primary Compact Layout**:
   * Click Compact Layout Assignment.
   * Click Edit Assignment.
   * From the dropdown, choose Product Compact Layout.
   * Click Save.



### Activity 2: Create Compact Layout for "Purchase Order" Object

1. **Navigate to Purchase Order Object**:
   * From Setup, click Object Manager.
   * In the Quick Find box, type Purchase Order and click on the Purchase Order object.
2. **Create New Compact Layout**:
   * Click Compact Layouts in the sidebar.
   * Click New.
   * **Label**: Purchase Order Compact Layout
   * **Selected Fields**: Select Purchase Order ID, Order Date, Total Order Cost, Supplier ID.
   * Click Save.
3. **Assign as Primary Compact Layout**:
   * Click Compact Layout Assignment.
   * Click Edit Assignment.
   * From the dropdown, choose Purchase Order Compact Layout.
   * Click Save.

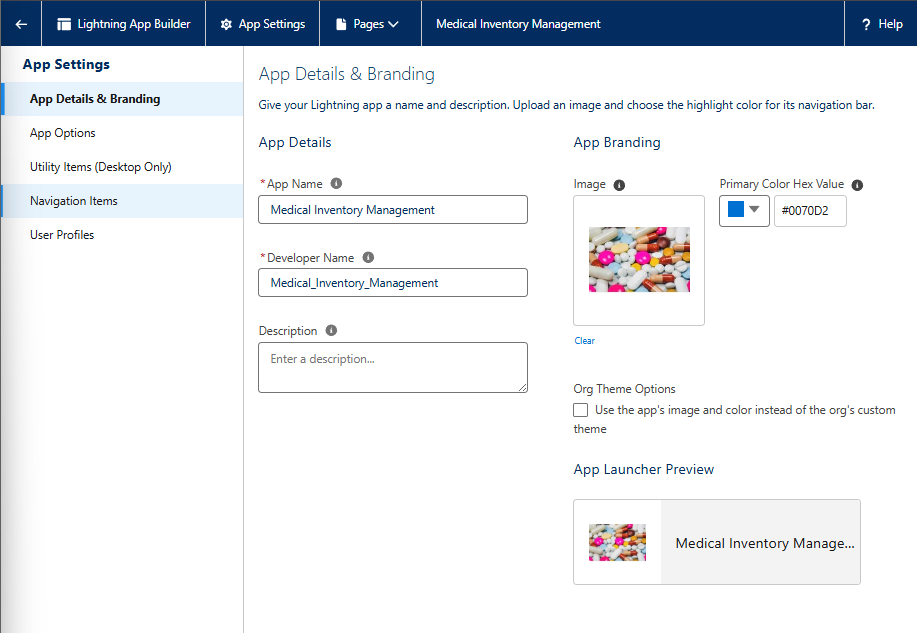
### 

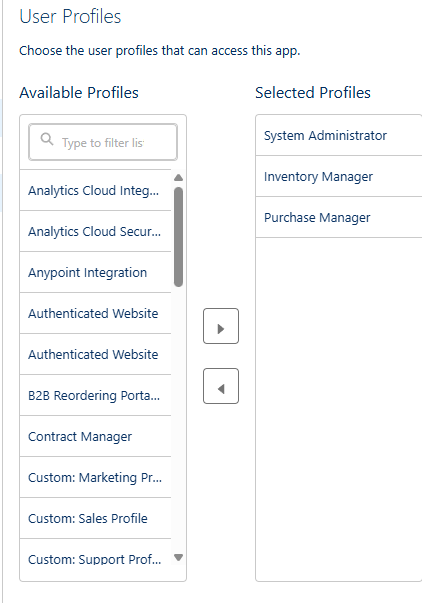
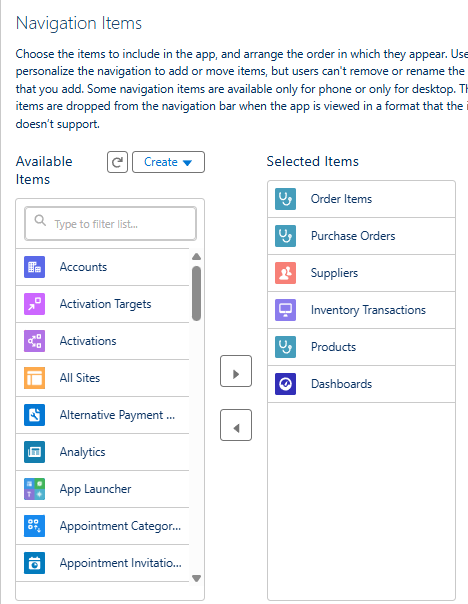
### G. The Lightning App Configuration

**Overview:** A Lightning App in Salesforce provides a branded, organized workspace for users to access the objects, tabs, and features relevant to their tasks. It serves as the central hub for the Medical Inventory Management System.

**Activity 1: Create "Medical Inventory Management" Lightning App**

1. **Navigate to App Manager**:
   * From Setup, enter App Manager in the Quick Find box.
   * Select App Manager.
2. **Start New Lightning App**:
   * Click New Lightning App.
3. **Define App Details**:
   * **App Name**: Medical Inventory Management
   * (Optional: Upload an image related to Medical Inventory if desired).
   * Click Next.
4. **Configure App Options**:
   * Leave default selections for App Options and Utility Items. Click Next for each.
5. **Add Navigation Items**:
   * From Available Items, select and move the following to Selected Items:
     + Products
     + Purchase Orders
     + Order Items
     + Inventory Transactions
     + Suppliers
     + Reports
     + Dashboards
   * Click Next.
6. **Assign to Profiles**:
   * From Available Profiles, select System Administrator and move it to Selected Profiles. (Add other relevant profiles like Inventory Manager and Purchase Manager as well for broader access).
   * Click Save & Finish.





**8. ADVANTAGES & DISADVANTAGES**

The implementation of the **CRM Application for Medical Inventory Management** on the Salesforce platform brought significant improvements in operational control, data visibility, and automation within medical supply chains. However, like any system, there are limitations that can be improved in future phases of the project.

**Advantages**

**1. Centralized Supplier, Inventory, and Purchase Order Management**

* All supplier records, medical product details, purchase orders, and stock transactions are maintained in one unified system.
* Eliminates the use of Excel sheets or manual registers for tracking inventory.
* Enhances collaboration across teams with cloud-based, real-time access.

**2. Process Automation via Flows and Triggers**

* Apex Trigger automatically recalculates the **Total Order Cost** whenever order items are added or modified.
* Record-triggered Flow sends **email confirmations** to suppliers when a new purchase order is created.
* Ensures consistent stock level updates, accurate data entry, and reduced manual workload.

**3. Role-Based Access and Security Controls**

* Custom profiles like **Inventory Manager** and **Purchase Manager** ensure users access only relevant objects and data.
* Supports granular field-level and object-level security to prevent unauthorized edits.
* Strengthens accountability and helps maintain compliance with hospital procurement standards.

**4. Dynamic Dashboards and Reports**

* Dashboards provide real-time visibility into:
  + Stock availability
  + Purchase trends
  + Supplier performance
  + Order costs and volume
* Summary and joint reports combine multiple objects (e.g., Purchase Orders with Order Items and Products) for actionable insights.
* Empowers medical administrators to make informed decisions without third-party analytics tools.

**5. Cloud-Based Scalability and Flexibility**

* Being hosted on Salesforce, the system can scale to multiple departments, users, or locations as needed.
* Supports future growth, such as integrating additional objects like **Returns**, **Prescriptions**, or **Maintenance Requests**.

**6. Admin-Friendly Tools and Low-Code Configuration**

* Admins can use point-and-click tools (e.g., Flow Builder, Schema Builder, Report Builder) for changes or enhancements.
* Minimizes dependency on developers for routine customizations.
* Supports agile modifications to match evolving inventory needs.

**Disadvantages**

**1. Limited Integration with External Healthcare Systems**

* Current version does not include integration with:
  + Hospital Management Systems (HMS)
  + Barcode scanners or IoT-based stock sensors
  + External SMS/email platforms for delivery alerts
* This limits real-time updates from external systems and automated replenishment cycles.

**2. Usability Challenges for Non-Technical Medical Staff**

* Staff with limited exposure to CRM tools (e.g., storekeepers or procurement clerks) may face difficulty using the Salesforce UI.
* Requires initial training to navigate objects, layouts, and dashboards.
* Standard error messages (e.g., from validation rules) may confuse non-technical users.

**3. Initial Setup and Customization Time**

* The system setup required significant configuration effort:
  + Creating five custom objects with multiple relationships
  + Setting up complex field types and automation
  + Configuring validation rules and error handling flows
* This time and resource investment may be a concern for small clinics or rural hospitals.

**4. Licensing and Cost Considerations**

* Salesforce licensing fees may be high for smaller medical institutions or non-profit facilities.
* Limited access to certain platform features (e.g., APIs or custom domains) in Developer Edition or low-tier licenses.
* Additional users require more paid licenses, which may impact scalability for larger hospitals.

**9. CONCLUSION**

The **CRM Application for Medical Inventory Management** was successfully designed, developed, and implemented on the Salesforce platform, delivering an efficient and reliable solution tailored to the needs of hospitals, clinics, and medical supply chains. The project replaced traditional, error-prone manual processes with a cloud-based system capable of streamlining supplier handling, product tracking, purchase orders, stock transactions, and reporting—achieving the core objective with high effectiveness.

Through the use of **custom objects**, **record-triggered flows**, **Apex triggers**, and **validation rules**, the application automated core processes such as purchase order creation, stock updates, and supplier email notifications. Custom **field configurations**, **page layouts**, and **relationships** enabled smooth and error-free data entry aligned with real-world use cases.

**Reports** and **dashboards** empowered stakeholders with real-time insights into stock status, purchase history, and supplier performance. At the same time, **profiles, roles**, and **permission sets** ensured secure, role-specific access to critical data across departments.

**Key Outcomes**

* **Automated Stock Management**: Triggers automatically updated current stock levels based on quantities received.
* **Operational Efficiency**: Validation rules and workflows reduced manual errors and ensured timely updates.
* **Supplier Communication**: Email flows sent instant confirmation upon PO creation, improving vendor coordination.
* **Visibility and Control**: Dashboards showed purchase trends, low stock alerts, and order summaries for decision-making.
* **Security and Role Segmentation**: Only authorized users could access or modify sensitive inventory and financial data.
* **Scalable Design**: Future modules such as Returns, Expiry Alerts, or Maintenance Logs can be added without rework.

**Business Impact**

* **For Procurement Teams**: Streamlined PO creation and tracking, with built-in alerts and delivery timelines.
* **For Inventory Staff**: Simplified data entry and automated stock updates ensured real-time accuracy.
* **For Admins and Management**: Reports and dashboards gave complete control over inventory, suppliers, and financial data.

By replacing Excel-based tracking and manual coordination with a robust Salesforce CRM, the system improved reliability, reduced overhead, and optimized communication across departments.

**Learnings from the Project**

1. **Salesforce Mastery**: Deepened our skills in custom object creation, automation (flows/triggers), validation, and reporting.
2. **Real-World Application**: Understood how to translate inventory and procurement workflows into scalable digital processes.
3. **Team-Based Development**: Collaborated effectively using Agile principles to assign roles, test features, and meet deadlines.
4. **User-Centric Design**: Designed UI elements (layouts, tabs, apps) around ease-of-use for non-technical hospital staff.
5. **Technical and Functional Integration**: Balanced technical solutions with functional outcomes to improve usability and maintainability.

**Final Note**

This project not only delivered a functional CRM tailored to the medical inventory use case but also strengthened our understanding of CRM principles, process automation, and collaborative development using Salesforce. It demonstrates how cloud-based systems can bring measurable improvements to operational workflows in critical industries like healthcare.

**11. APPENDIX**

The appendix provides essential supporting materials that showcase the implementation of technical components in the **Medical Inventory Management CRM**. It includes the Apex Trigger source code, flow configuration details, data creation approach, and links to the GitHub repository and project demo video—allowing stakeholders and evaluators to validate and trace the execution flow and logic effectively.

**Source Code**

**Apex Trigger: CalculateTotalAmountTrigger**

trigger CalculateTotalAmountTrigger on Order\_Item\_\_c (after insert, after update, after delete, after undelete) {

CalculateTotalAmountHandler.calculateTotal(

Trigger.new,

Trigger.old,

Trigger.isInsert,

Trigger.isUpdate,

Trigger.isDelete,

Trigger.isUndelete

);

}

**Handler Class: CalculateTotalAmountHandler**

public class CalculateTotalAmountHandler {

public static void calculateTotal(List<Order\_Item\_\_c> newItems, List<Order\_Item\_\_c> oldItems,

Boolean isInsert, Boolean isUpdate, Boolean isDelete, Boolean isUndelete) {

Set<Id> parentIds = new Set<Id>();

if (isInsert || isUpdate || isUndelete) {

for (Order\_Item\_\_c item : newItems) {

parentIds.add(item.Purchase\_Order\_Id\_\_c);

}

}

if (isUpdate || isDelete) {

for (Order\_Item\_\_c item : oldItems) {

parentIds.add(item.Purchase\_Order\_Id\_\_c);

}

}

Map<Id, Decimal> purchaseToAmount = new Map<Id, Decimal>();

List<AggregateResult> results = [

SELECT Purchase\_Order\_Id\_\_c, SUM(Amount\_\_c) totalAmount

FROM Order\_Item\_\_c

WHERE Purchase\_Order\_Id\_\_c IN :parentIds

GROUP BY Purchase\_Order\_Id\_\_c

];

for (AggregateResult ar : results) {

purchaseToAmount.put((Id)ar.get('Purchase\_Order\_Id\_\_c'), (Decimal)ar.get('totalAmount'));

}

List<Purchase\_Order\_\_c> updateList = new List<Purchase\_Order\_\_c>();

for (Id pid : purchaseToAmount.keySet()) {

updateList.add(new Purchase\_Order\_\_c(Id = pid, Total\_Order\_Cost\_\_c = purchaseToAmount.get(pid)));

}

if (!updateList.isEmpty()) {

update updateList;

}

}

}

**Purpose:**

* Automatically calculates and updates Total\_Order\_Cost\_\_c on Purchase\_Order\_\_c.
* Ensures accuracy in cost tracking when related Order\_Item\_\_c records are added, updated, or deleted.

**Flow: Actual\_Delivery\_Date\_Updating**

* **Flow Type**: Record-Triggered Flow
* **Object**: Purchase\_Order\_\_c
* **Trigger**: On Create or Update
* **Logic**:
  + Retrieves Order\_Date\_\_c of the triggering Purchase Order.
  + Adds 3 days to simulate a realistic delivery timeline.
  + Updates the Actual\_Delivery\_Date\_\_c field automatically.

**Dataset Handling**

No external data files or CSV imports were used.

* **All records were manually entered** into the custom Lightning App to simulate real-world data.
* Sample data covered:
  + Product\_\_c: Multiple items with unit prices and expiry dates.
  + Supplier\_\_c: 10+ sample suppliers with contact info.
  + Purchase\_Order\_\_c: Linked to Suppliers and included delivery dates.
  + Order\_Item\_\_c: Mapped with quantities and product lookups.
  + Inventory\_Transaction\_\_c: Used for stock-level validation.

**GitHub & Demo Links**

* **GitHub Repository**  
  https://github.com/gayathriv515/Medical\_Inventory\_Management.git
* **Video Demonstration Link**  
  https://drive.google.com/file/d/10fyCHrUDKxeejeXW6isnRlI6zA2F3YN6/view?usp=sharing  
  The video walkthrough includes:
  + Lightning App overview
  + Creation of Product, Supplier, and Purchase Order
  + Flow execution for delivery date
  + Trigger-based stock updates
  + Dashboard and report visualization

**Final Note**

All components described in this appendix validate the full-stack implementation of the **Medical Inventory Management CRM**. The source code, flows, data modeling, and automation mechanisms demonstrate a complete, functional, and test-ready solution aligned with real-world operational workflows in medical supply chain environments.