**A blue and white logo

AI-generated content may be incorrect. Medical Inventory Management System**

**BHARATHIAR UNIVERSITY, COIMBATORE**

**KATHIR COLLEGE OF ARTS AND SCIENCE**

**College Code: bruah**

**TEAM ID: NM2025TMID24381**

**TEAM MEMBERS:**

**Team Leader Name: MOHAMMED ARSATH A**

**Email: arsathm360@gmail.com**

**Team Member: LOGESH V**

**Email: logeshlogesh12120@gmail.com**

**Team Member: HARISH N**

**Email: karuppusaami49@gmail.com**

**Team Member: SATHISH KUMAR S**

**Email: kumarsathish29220@gmail.com**

**Academic Year: 2025 – 2026**

**Mentor: Ms. Infant Angel**

ABSTRACT

Effective inventory management is critical in the healthcare sector to ensure the availability of medical supplies, reduce waste, and maintain compliance with regulatory standards. This project introduces a **Medical Inventory Management System** built on the **Salesforce platform**, leveraging its cloud-based infrastructure, automation capabilities, and customizable data models. The system enables real-time tracking of medical inventory, including pharmaceuticals, equipment, and consumables, across multiple locations. Key features include automated stock level alerts, expiry tracking, purchase order management, vendor integration, and comprehensive reporting dashboards. By utilizing Salesforce's native tools such as **Salesforce Flow, Apex, Lightning Components, and Reports & Dashboards**, the solution enhances operational efficiency, minimizes human error, and provides data-driven insights for inventory optimization. This project demonstrates how Salesforce can be adapted beyond traditional CRM use cases to support critical healthcare logistics and supply chain processes.

1.INTRODUCTION

1.1 Project Overview

The **Medical Inventory Management System** is a cloud-based solution developed on the **Salesforce platform** to streamline the tracking, ordering, and management of medical supplies and equipment in healthcare organizations. Traditional inventory systems in the medical field often suffer from issues such as manual errors, lack of real-time visibility, stockouts, overstocking, and difficulties in regulatory compliance. This project addresses these challenges by utilizing the flexibility and automation capabilities of Salesforce to create a centralized, intelligent inventory system.

1.2 Purpose

This project aims to solve these problems by leveraging Salesforce’s robust CRM platform and customizing it for inventory management. The system is designed to:

* **Ensure optimal stock levels** and timely replenishment
* **Improve patient safety** by preventing the use of expired or unapproved items
* **Enhance operational efficiency** by automating routine inventory tasks
* **Provide real-time data and insights** for better decision-making
* **Ensure compliance** with healthcare regulations through traceability and audit logs

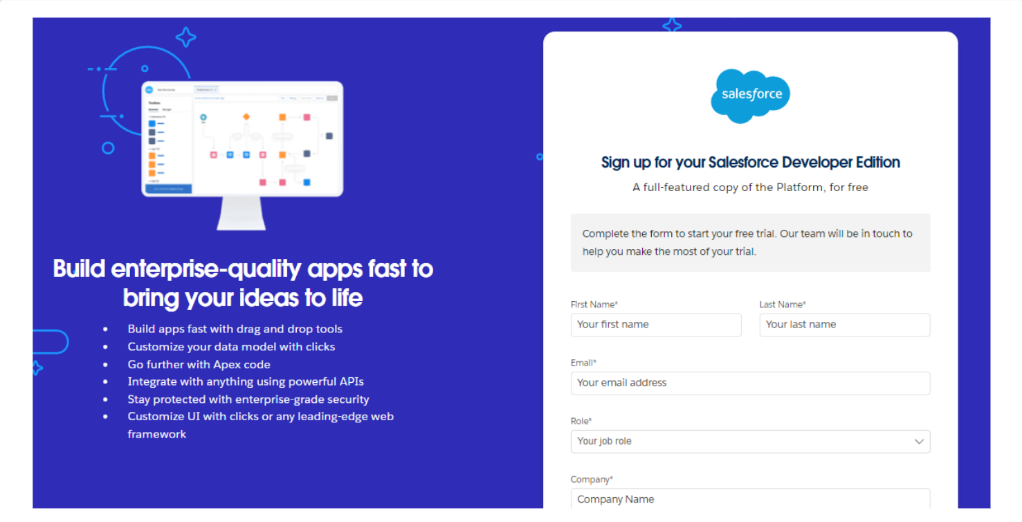
Ultimately, the project seeks to enhance the quality of healthcare service delivery by ensuring that the right medical supplies are available at the right time and place, while minimizing waste, cost, and administrative burden.

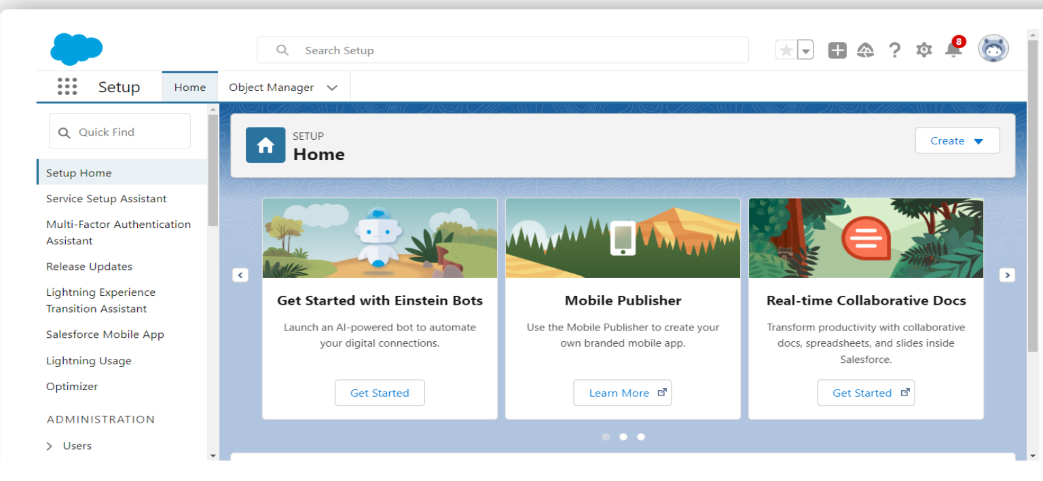
DEVELOPMENT PHASE:

Creating Developer Account:

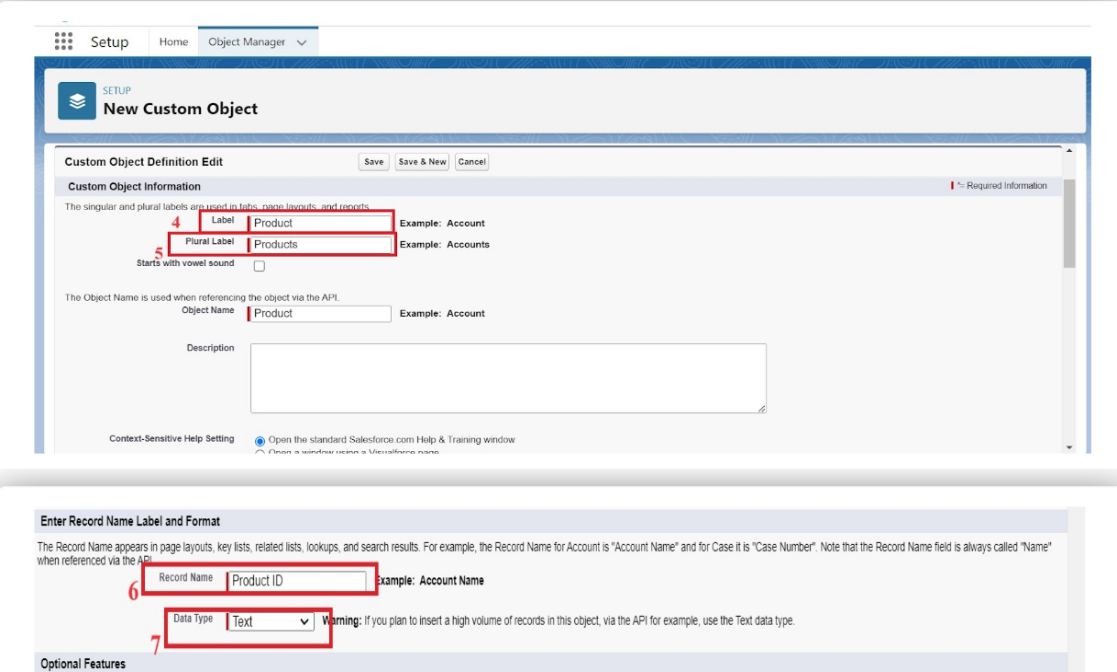
By using this URL: <https://www.salesforce.com/form/developer-signup/?d=pb>



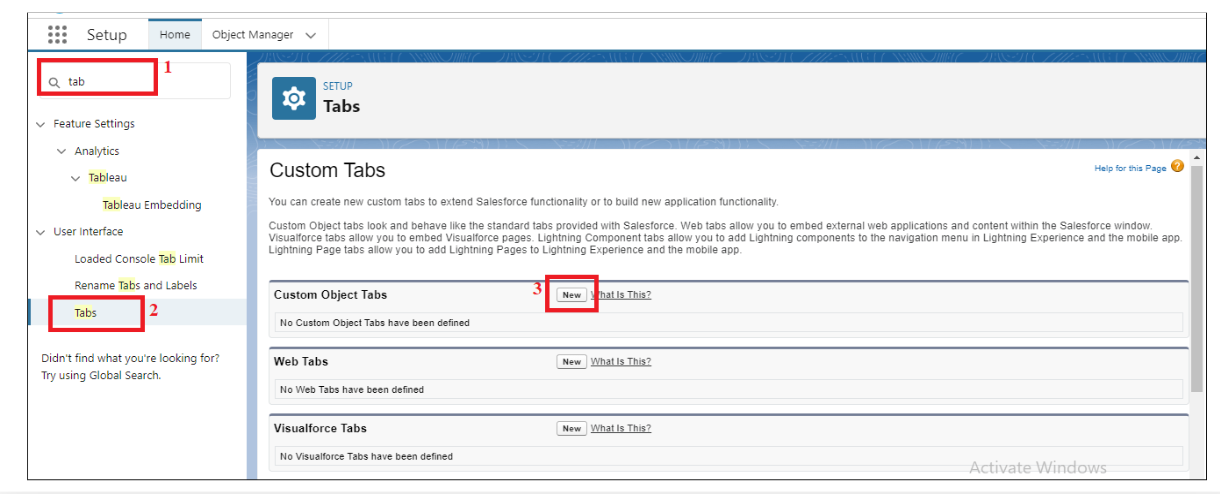


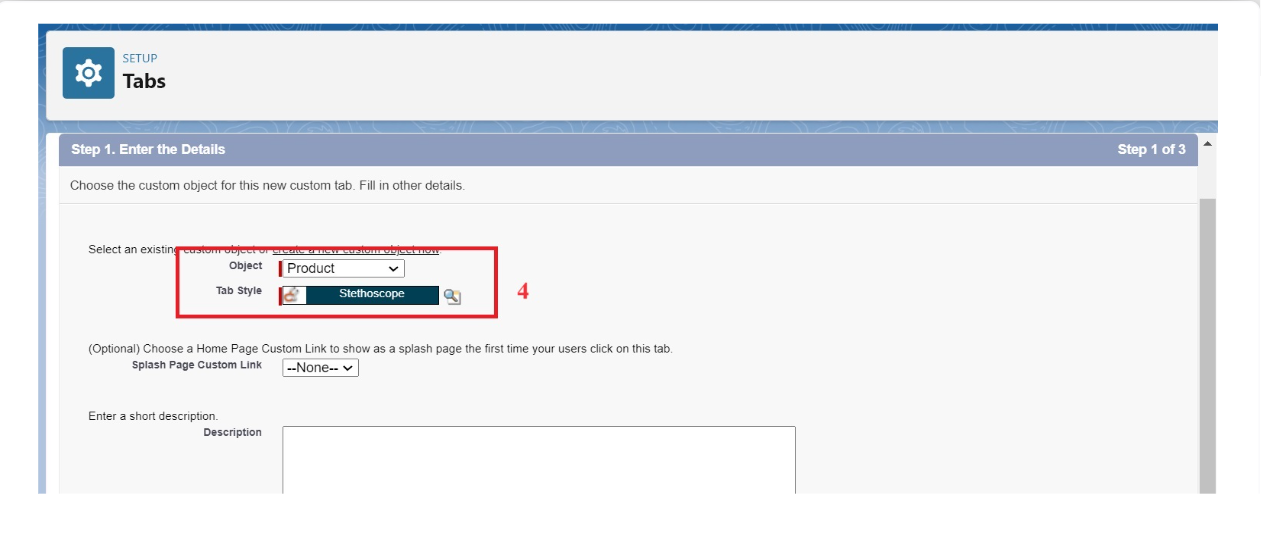


Create a Object Creation

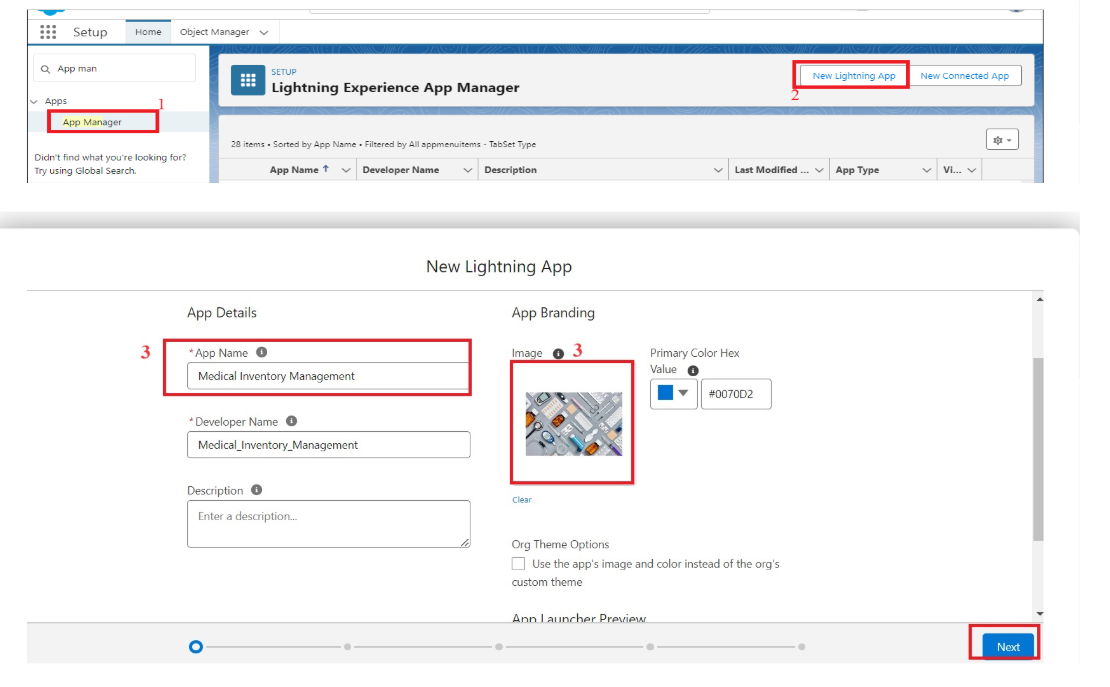


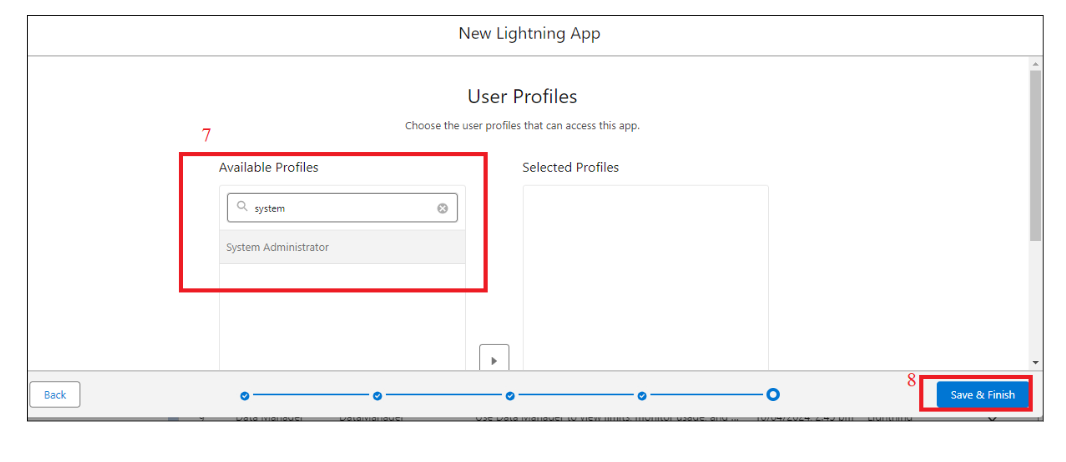
Tabs

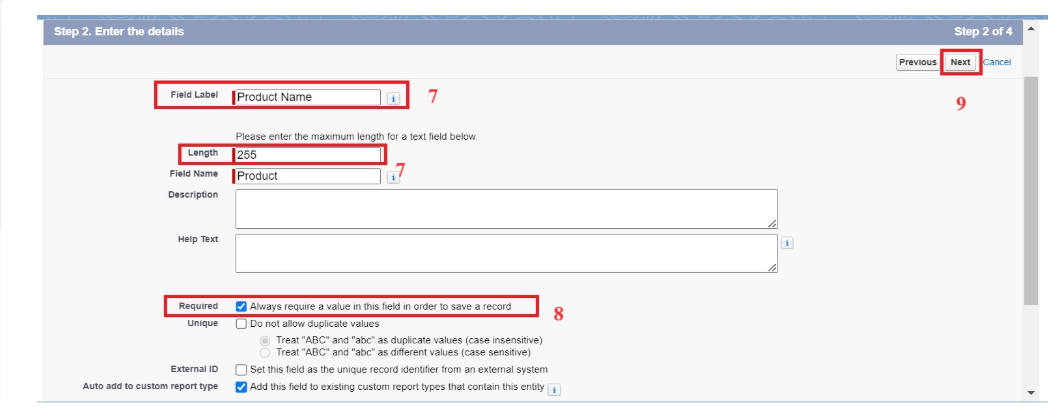




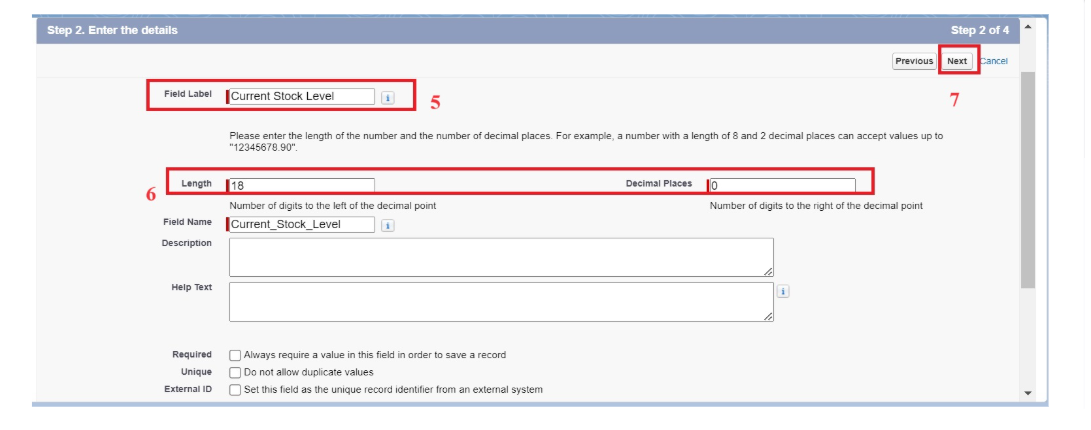
The Lightning App

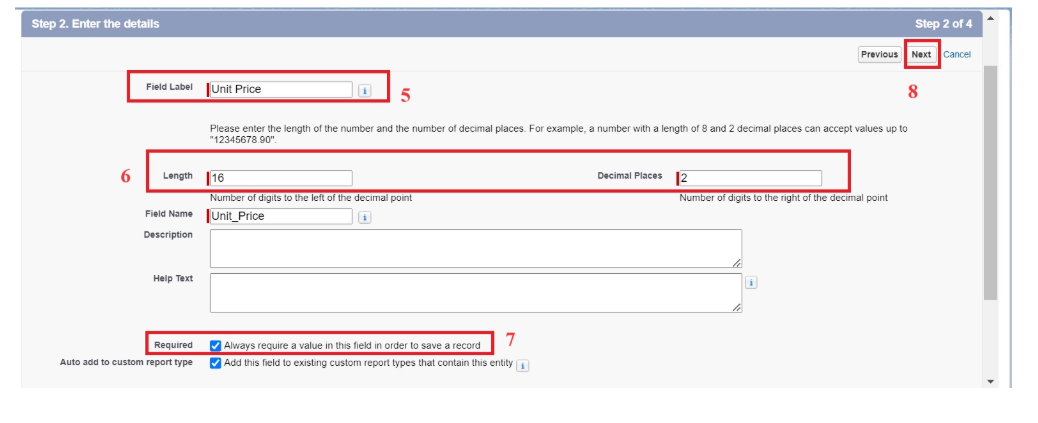


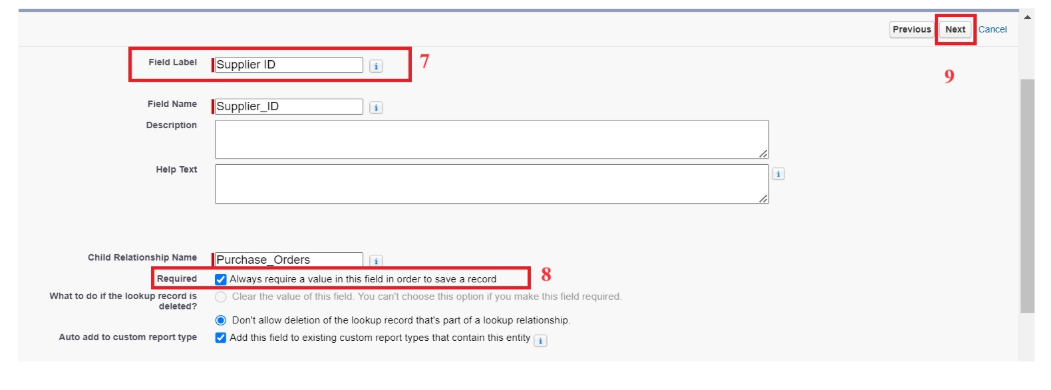
 Fields



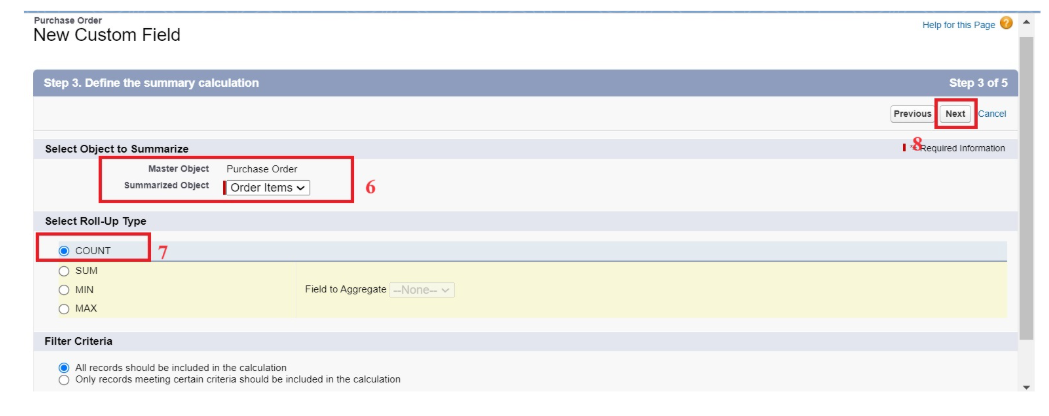


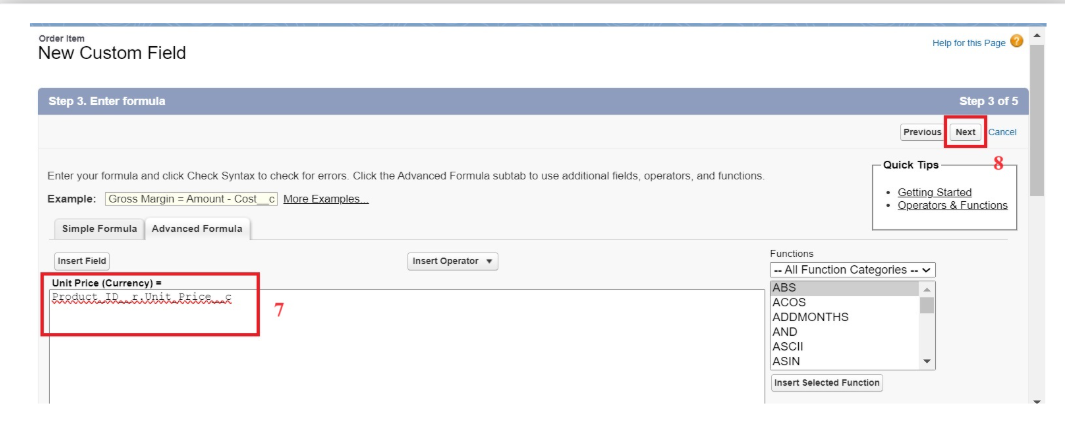


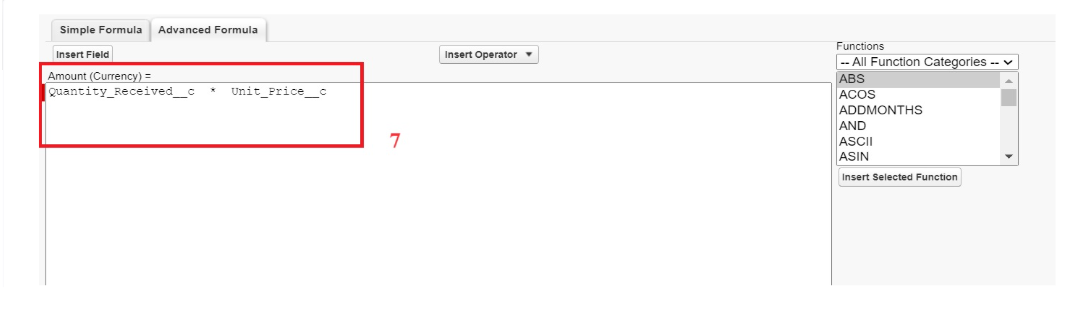


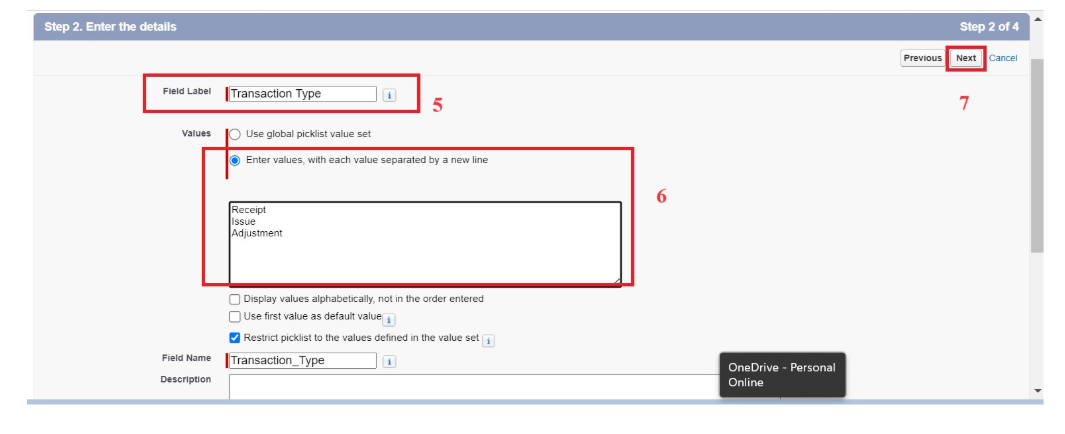


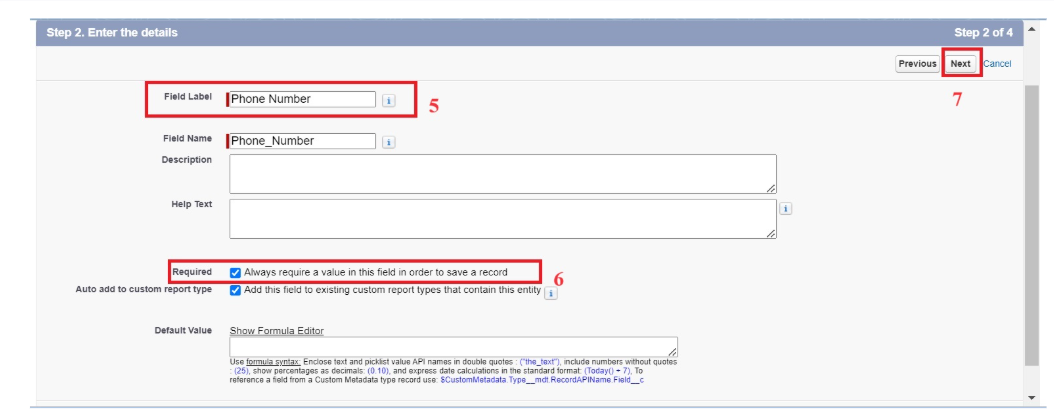




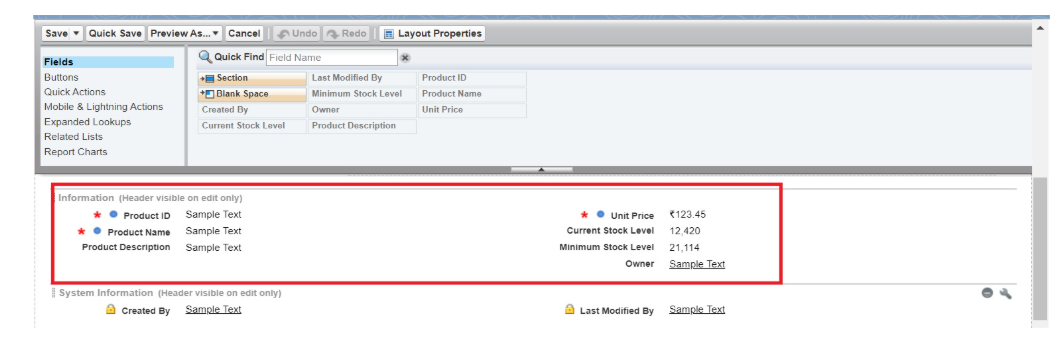


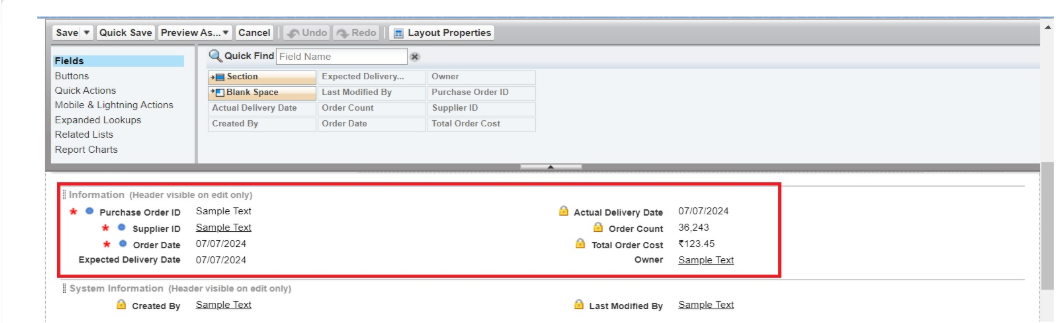


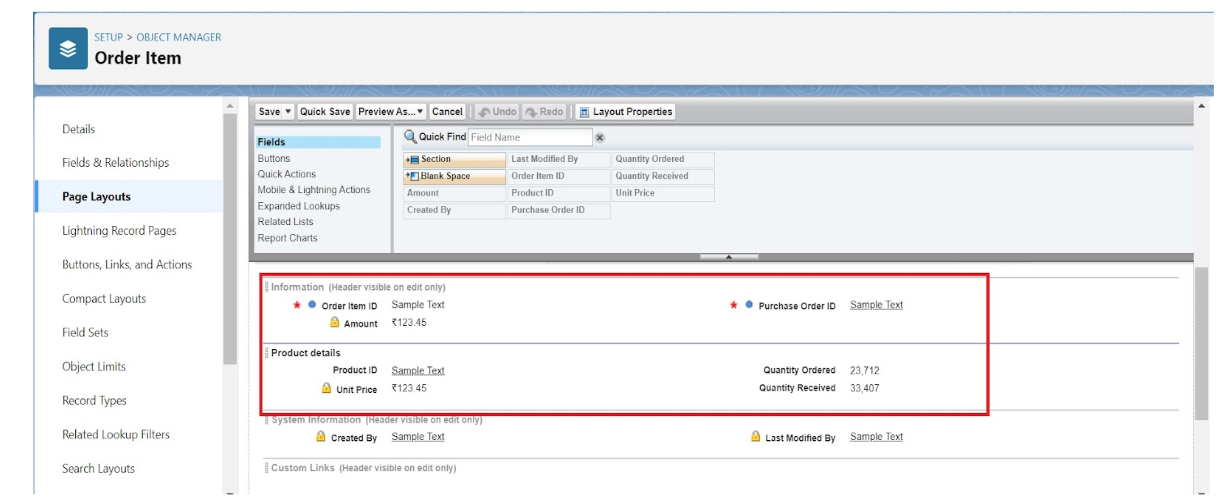


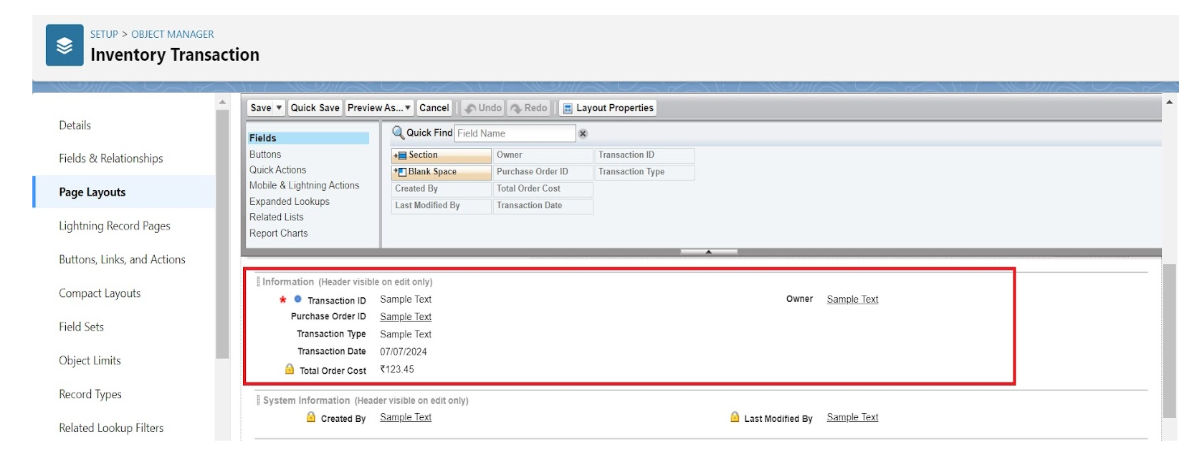


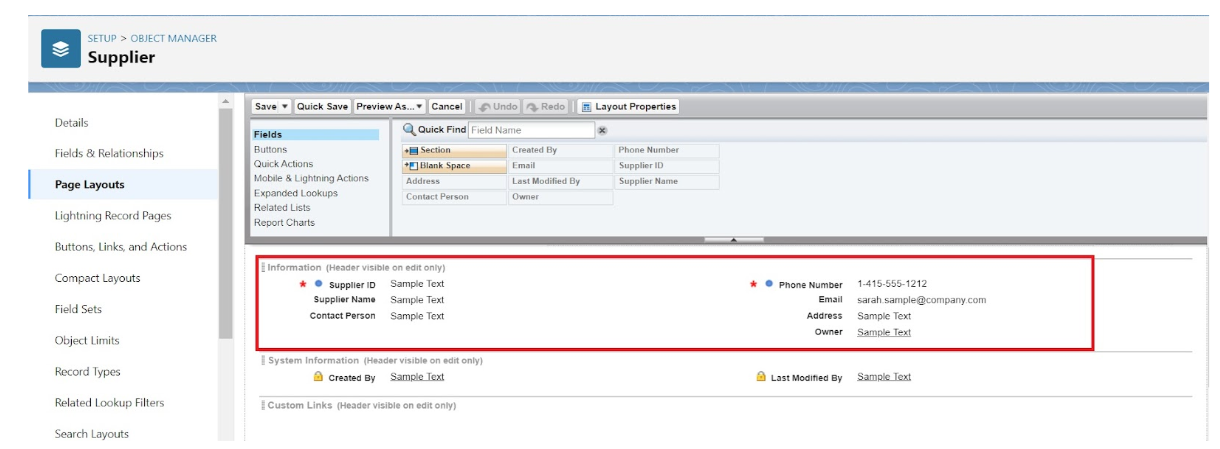
Updating of Page Layouts









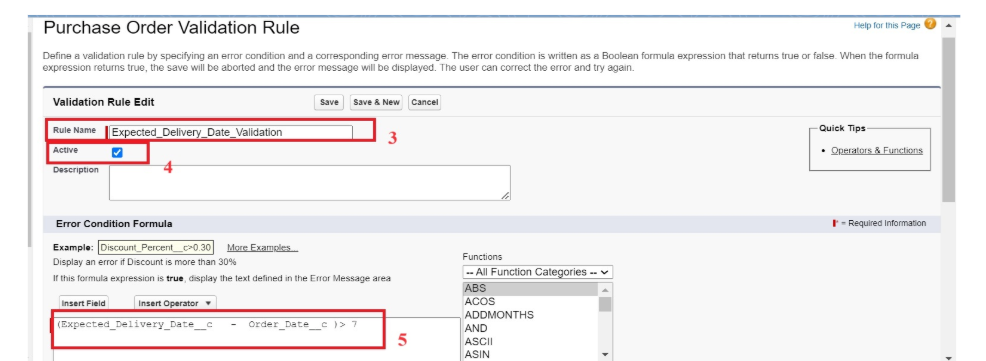


Compact Layouts

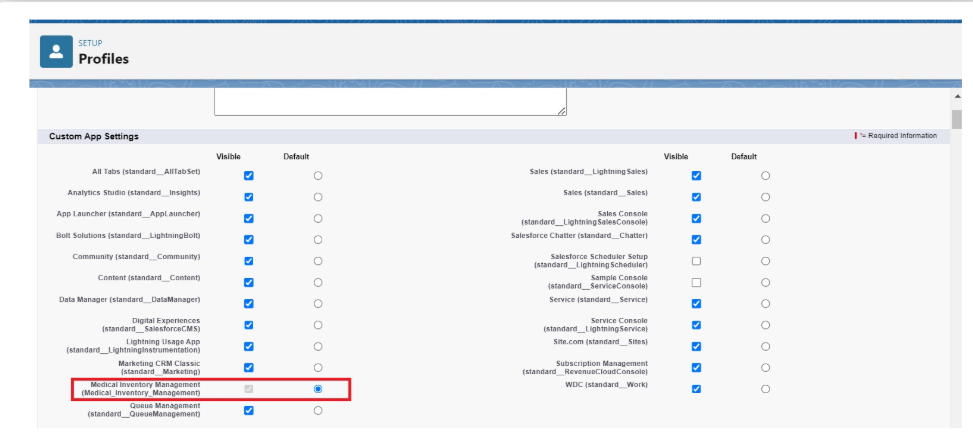


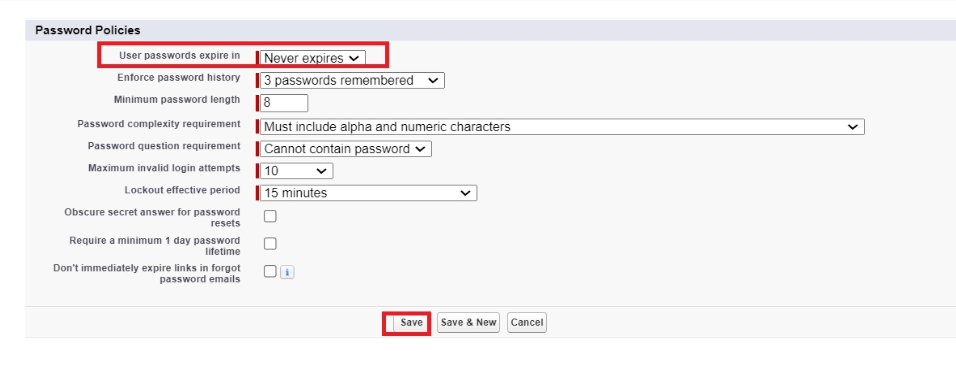


Validation rules

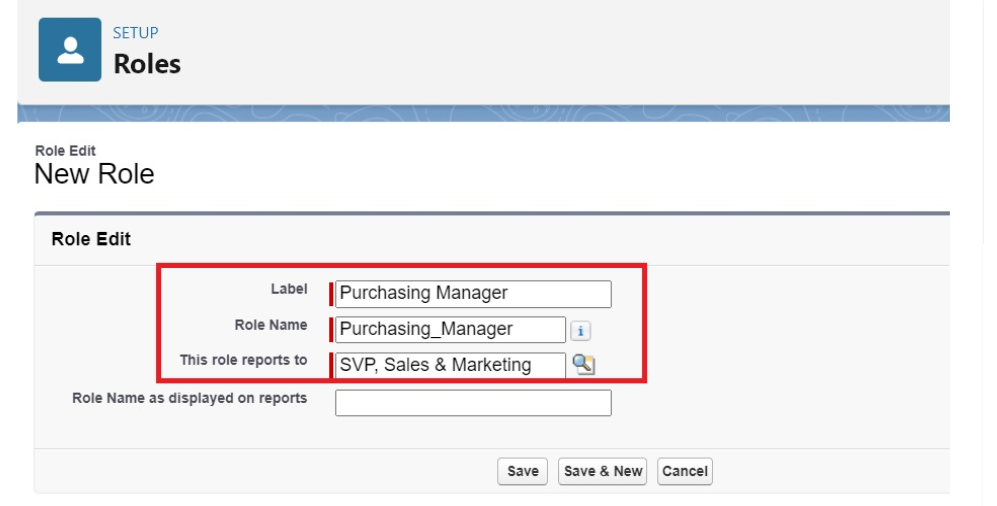


Profiles

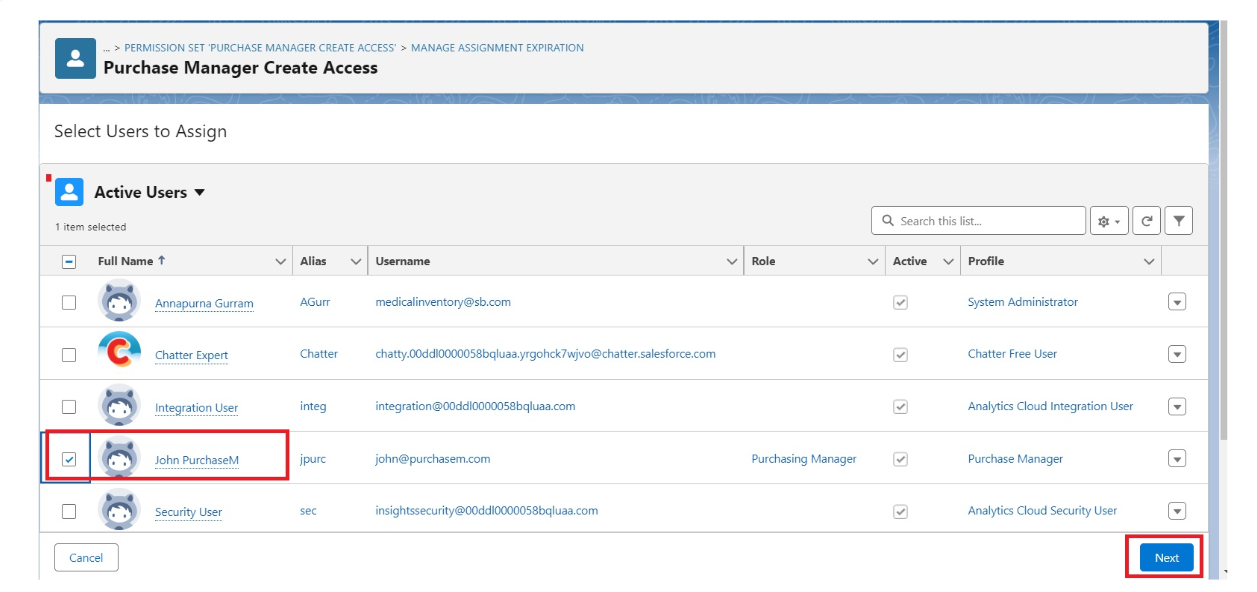




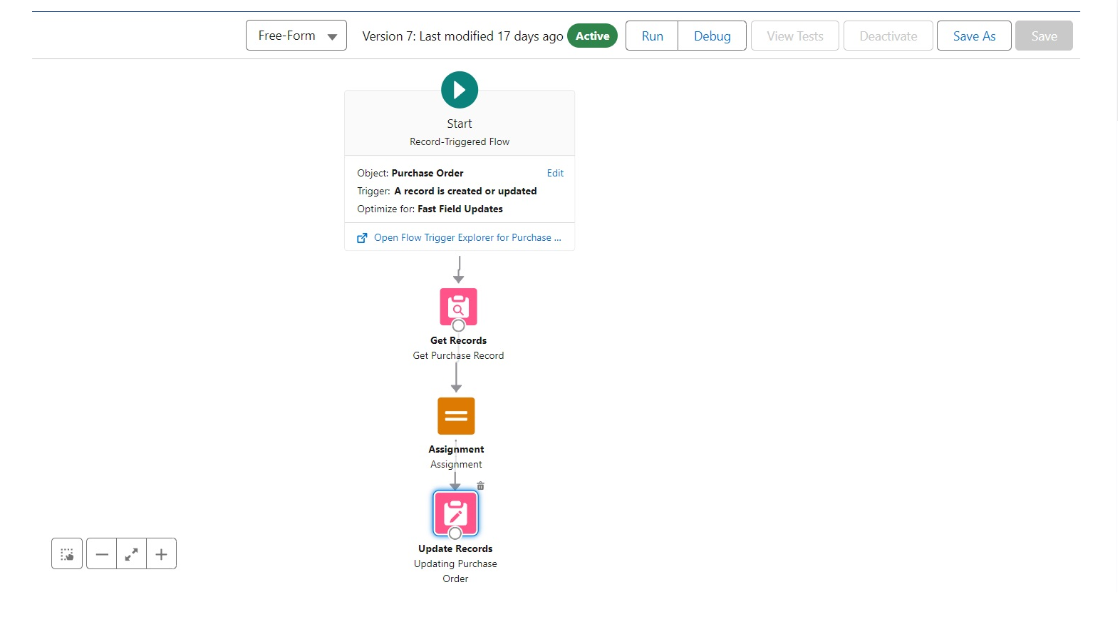
Roles



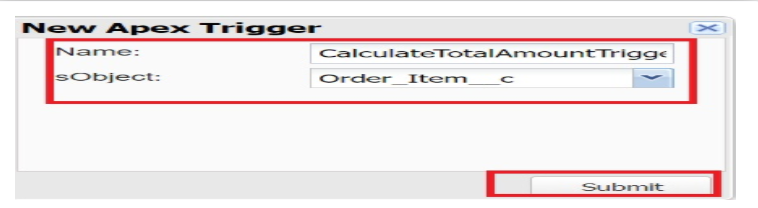
Permission Sets



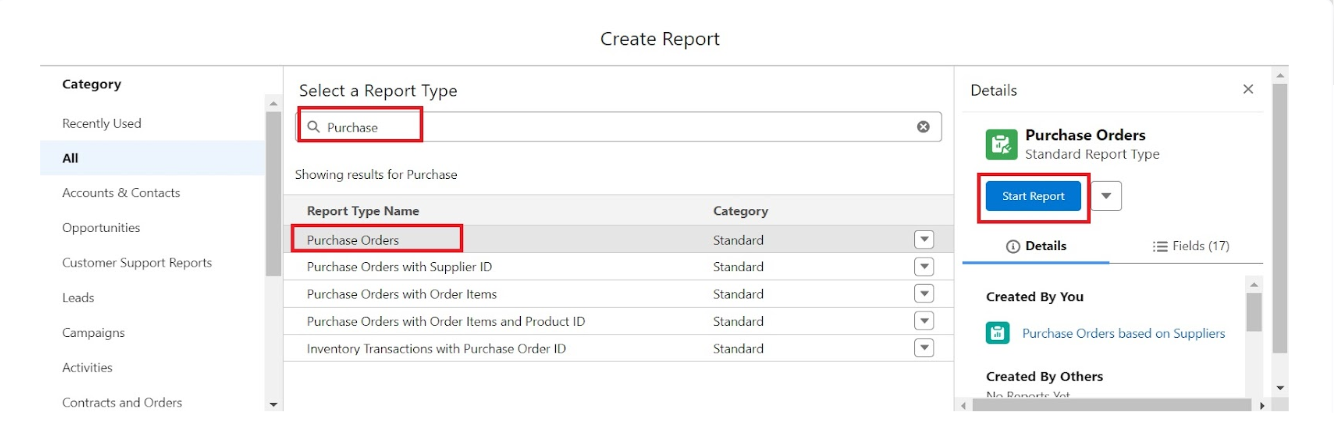
Flows



Triggers

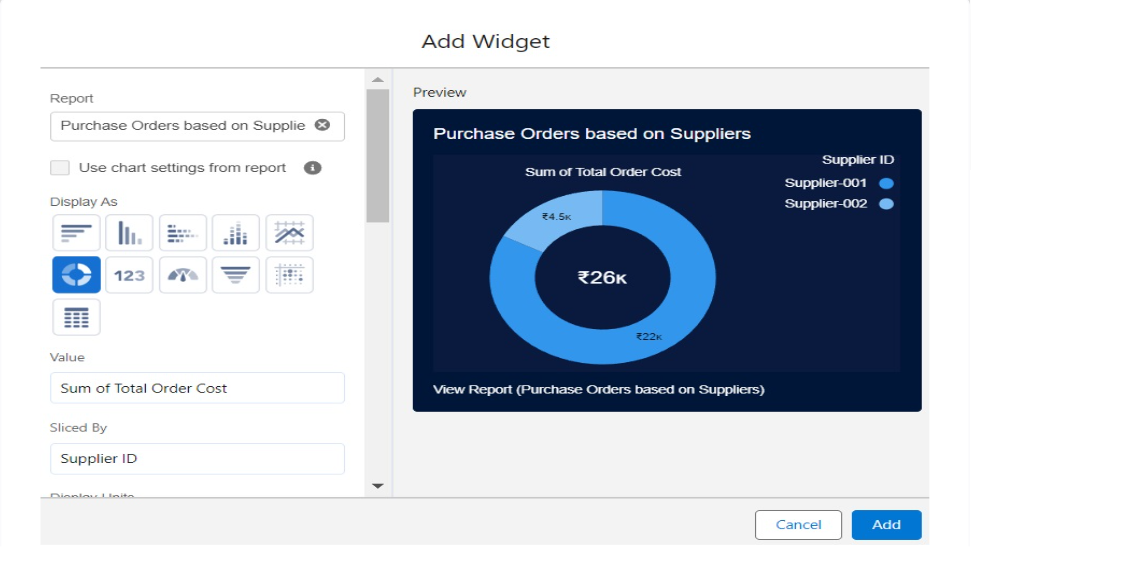


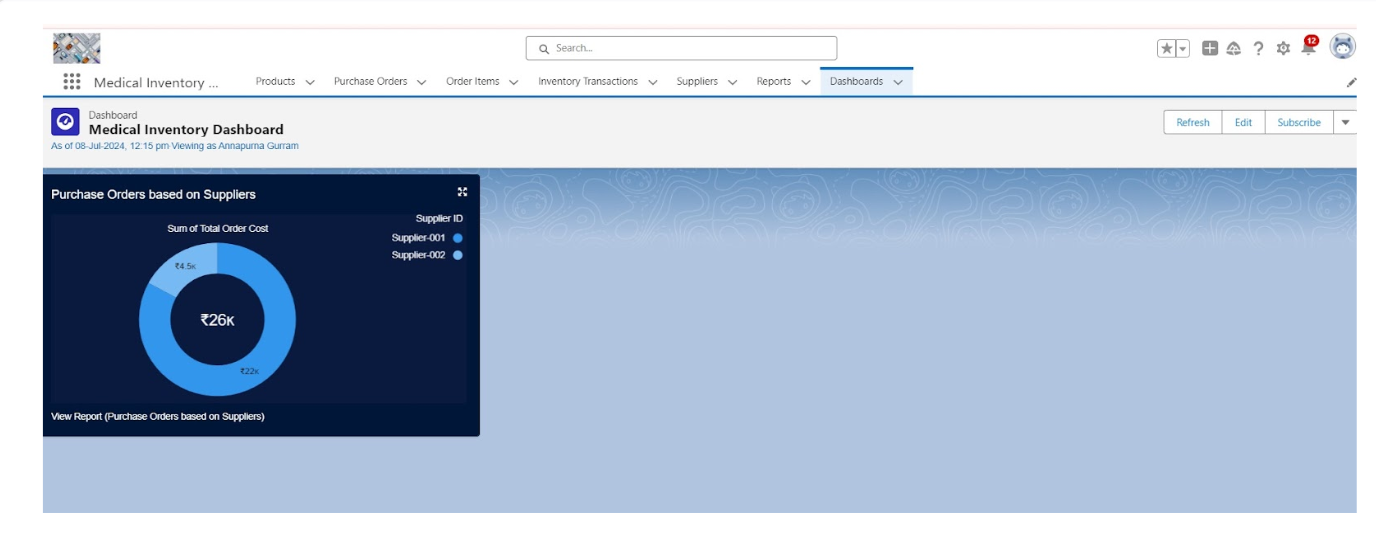
Reports





Dashboards





Conclusion:

The Medical Inventory Management System developed on the Salesforce platform successfully demonstrates how cloud-based CRM technology can be adapted to meet the complex and critical needs of the healthcare industry. By automating inventory tracking, managing stock levels, monitoring expiry dates, and generating real-time reports, the system significantly improves the efficiency, accuracy, and reliability of medical inventory operations.

APPENDIX

* Source Code: Provided in Apex Classes And Trigger

CODE:

public class CalculateTotalAmountHandler {

// Method to calculate the total amount for Purchase Orders based on related Order Items

public static void calculateTotal(List<Order\_Item\_\_c> newItems, List<Order\_Item\_\_c> oldItems, Boolean isInsert, Boolean isUpdate, Boolean isDelete, Boolean isUndelete) {

// Collect Purchase Order IDs affected by changes in Order\_Item\_\_c records

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

// For insert, update, and undelete scenarios

if (isInsert || isUpdate || isUndelete) {

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

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

}

}

// For update and delete scenarios

if (isUpdate || isDelete) {

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

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

}

}

// Calculate the total amounts for affected Purchase Orders

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

if (!parentIds.isEmpty()) {

// Perform an aggregate query to sum the Amount\_\_c for each Purchase Order

List<AggregateResult> aggrList = [

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\_\_c9\*

];

// Map the result to Purchase Order IDs

for (AggregateResult aggr : aggrList) {

Id purchaseOrderId = (Id)aggr.get('Purchase\_Order\_Id\_\_c');

Decimal totalAmount = (Decimal)aggr.get('totalAmount');

purchaseToUpdateMap.put(purchaseOrderId, totalAmount);

}

// Prepare Purchase Order records for update

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

for (Id purchaseOrderId : purchaseToUpdateMap.keySet()) {

Purchase\_Order\_\_c purchaseOrder = new Purchase\_Order\_\_c(Id = purchaseOrderId, Total\_Order\_cost\_\_c = purchaseToUpdateMap.get(purchaseOrderId));

purchaseToUpdate.add(purchaseOrder);

}

// Update Purchase Orders if there are any changes

if (!purchaseToUpdate.isEmpty()) {

update purchaseToUpdate;

}

}

}

}