***MEDICAL INVENTORY***

**COLLEGE NAME: SHRI NEHRU MAHA VIDYALAYA COLLEGE OF ARTS AND SCIENCE**

**COLLEGE CODE: BRU26**

**TEAM ID:** *NM2025TMID20600*

**TEAM MEMBERS:**

Team Leader Name: **JEMSHIDHA A**

**Email:** jemshidhabasheer@gmail.com

Team Member: **ASHOK KUMAR T**

**Email:** ashok55826@gmail.com

Team Member: **ASHWIN R**

**Email:** ashwinashwin5532@gmail.com

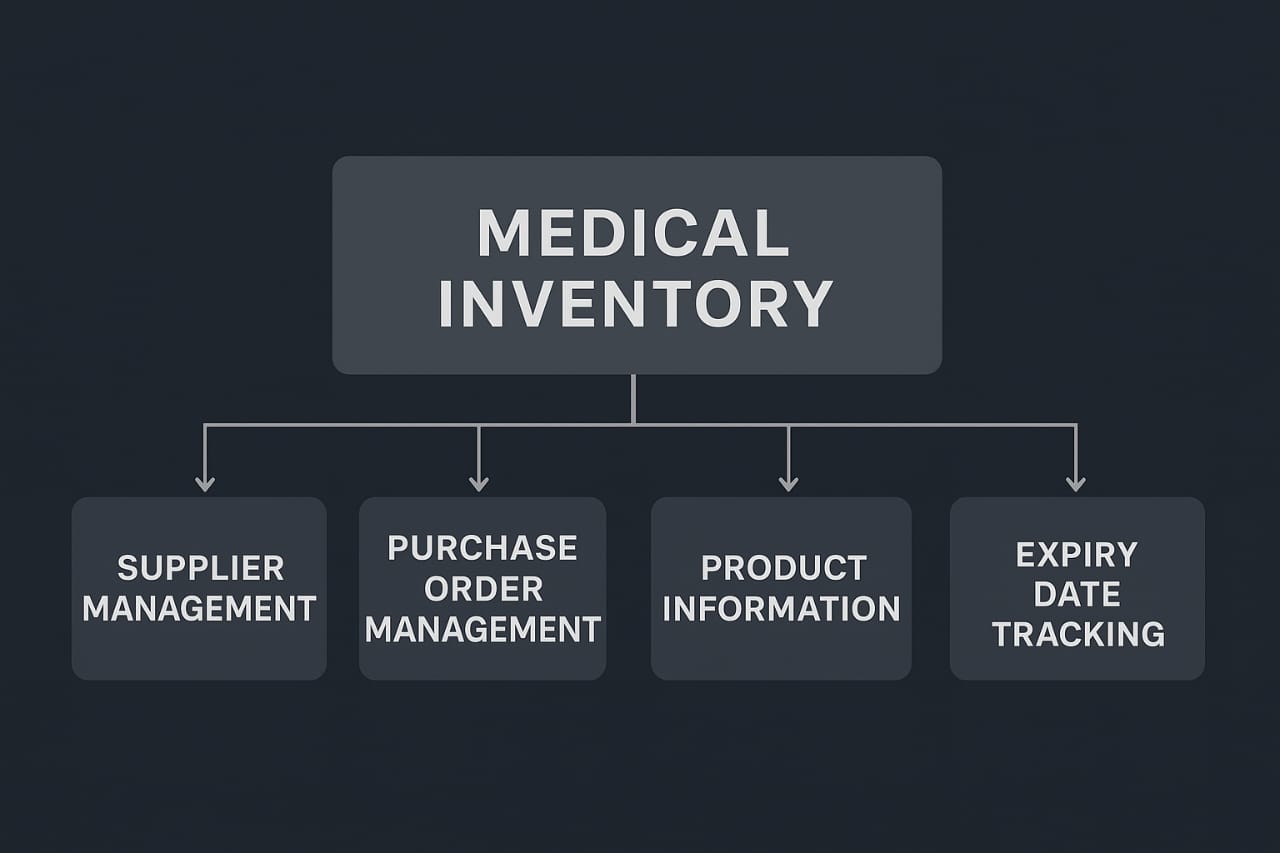
Team Member: **DEEPIKA S S**

**Email:** edeepika0180@gmail.com

# 1.INTRODUCTION

## 1.1 Project Overview

The Medical Inventory Management System is a comprehensive Salesforce application designed to streamline and manage various operational aspects of the medical inventory. It can efficiently maintain supplier details, manage purchase orders, track product details and transactions, and monitor expiry dates of products, thereby improving operational efficiency, data accuracy, and reporting capabilities.



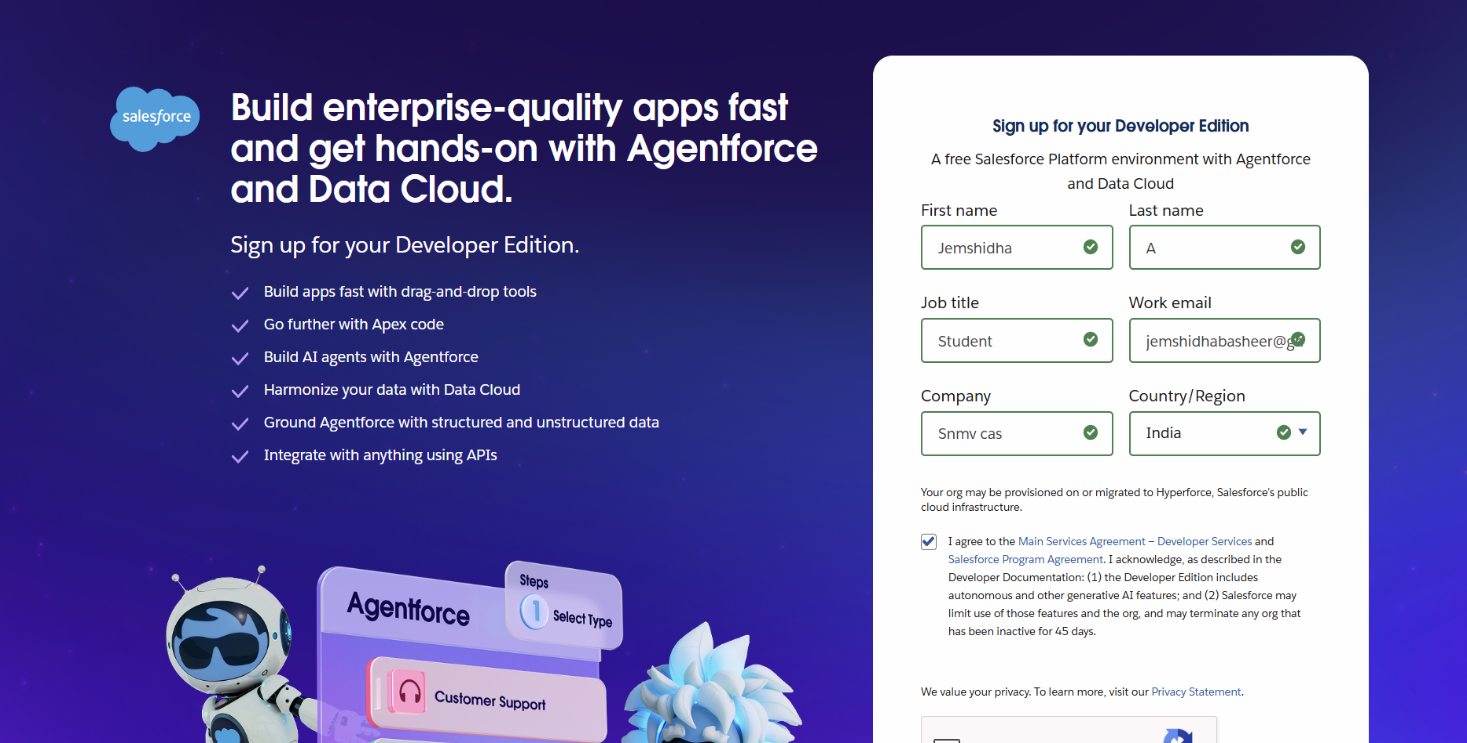
## 1.2 Purpose

This project is a comprehensive Salesforce application to streamline and manage various operational aspects of medical inventory. The system aims to efficiently maintain supplier details, manage purchase orders, track product details and transactions, and monitor the expiry dates of products. Maintain detailed records of suppliers, including contact information. Catalog product information, including descriptions, stock levels. Monitor and track product expiry dates to avoid using expired items. Comprehensive reports to track supplier performance, and purchase orders.

# DEVELOPMENT PHASE

**Creating Developer Account:**

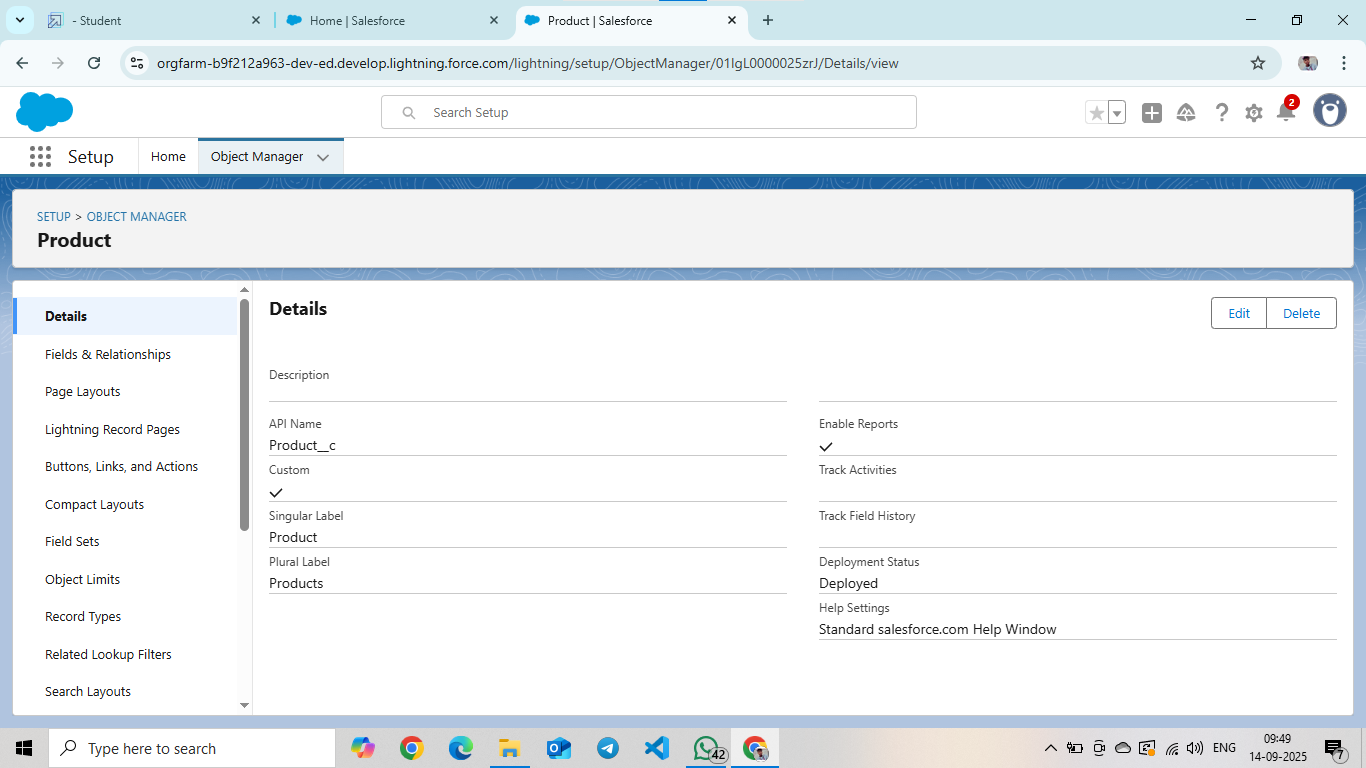
By using this URL[**-**](https://www.salesforce.com/form/developer-signup/?d=pb)  <https://developer.salesforce.com/signup>

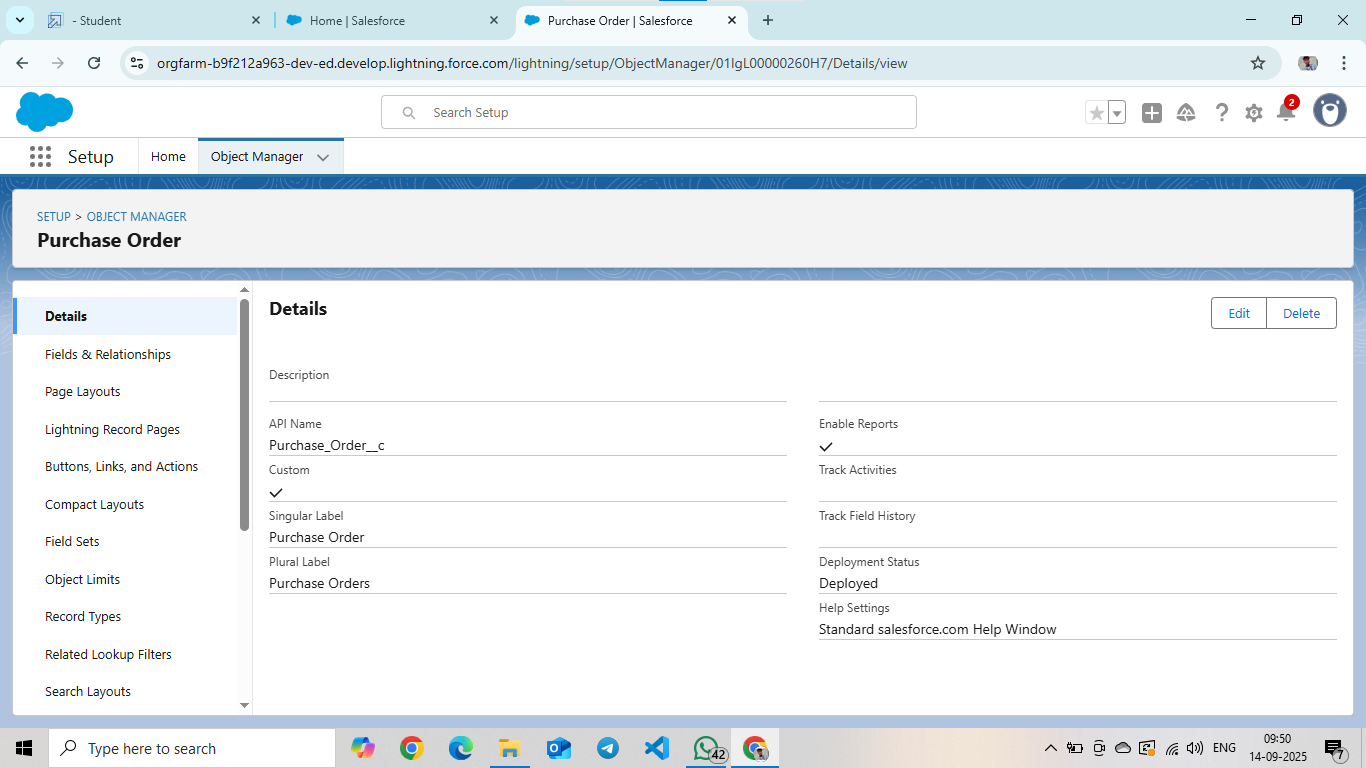


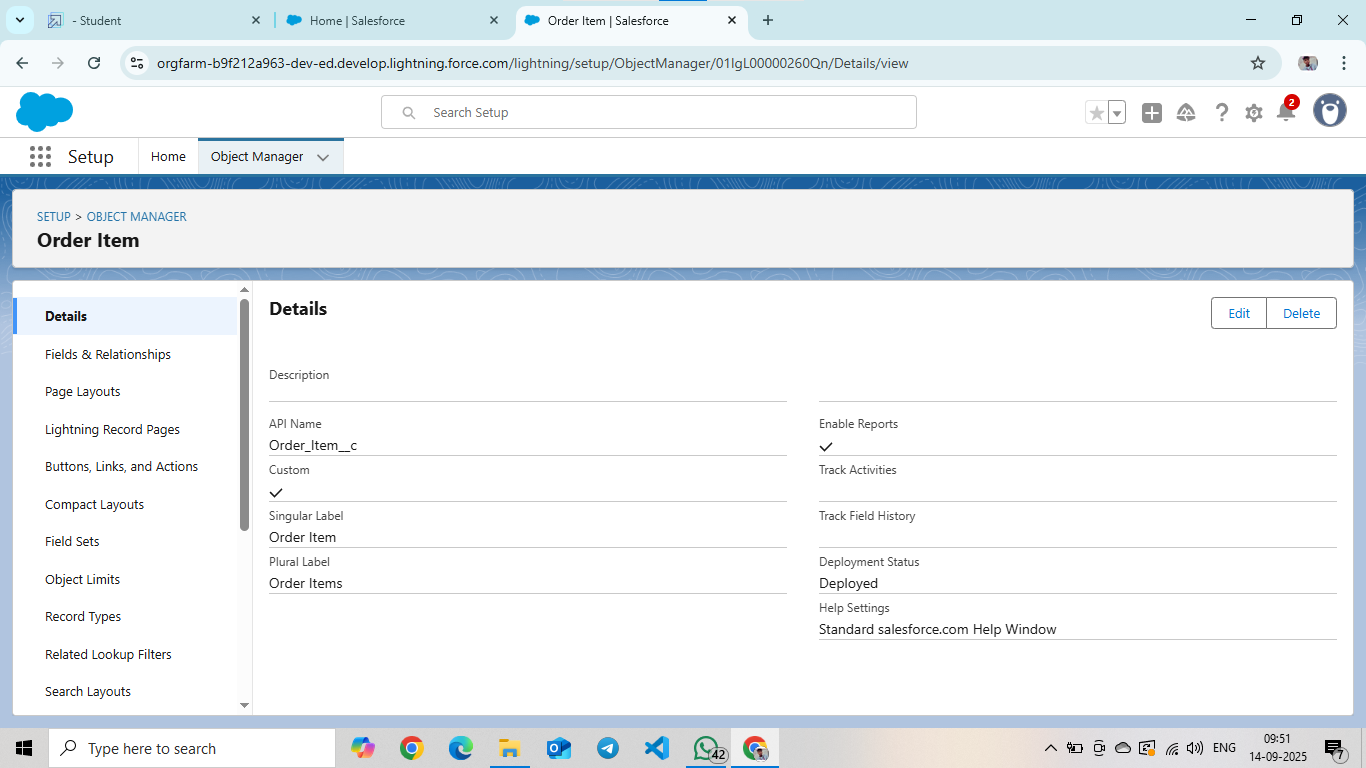
**OBJECT:**

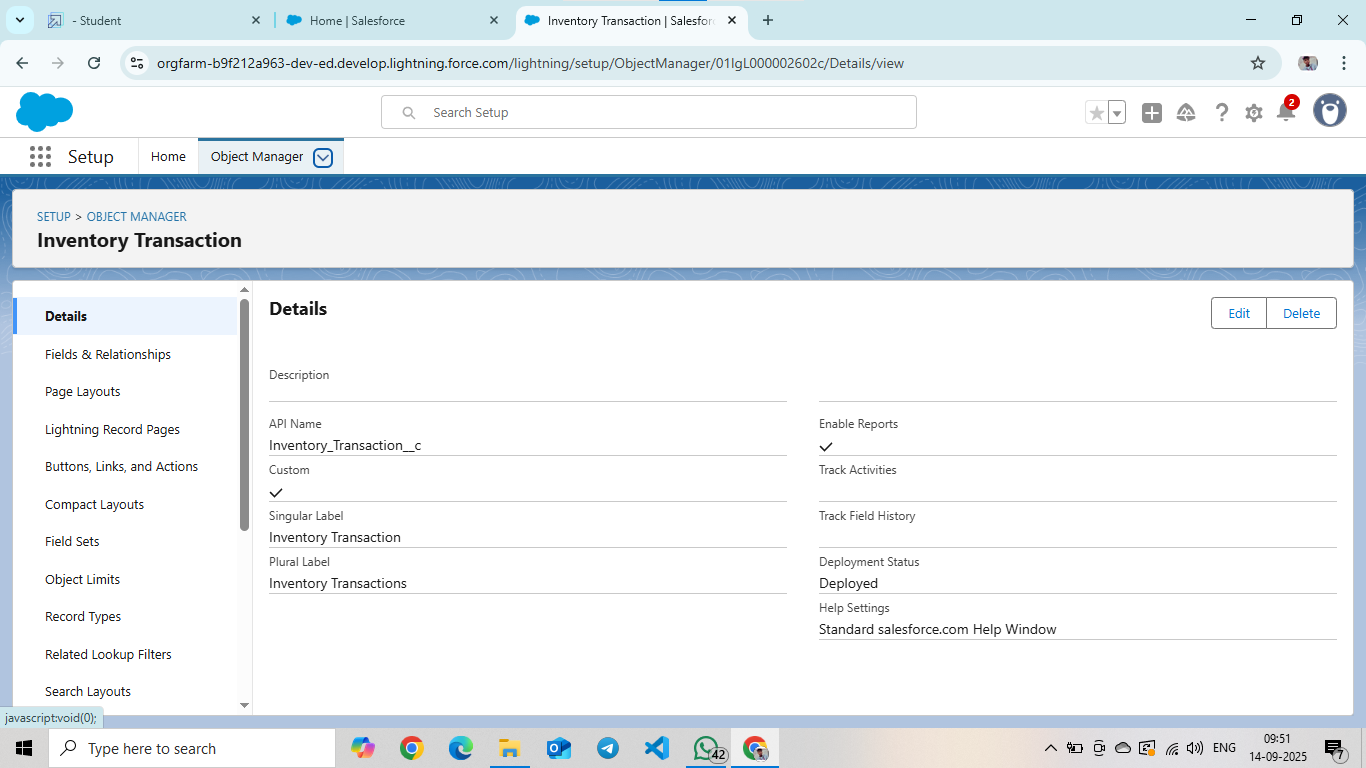
Created Objects:

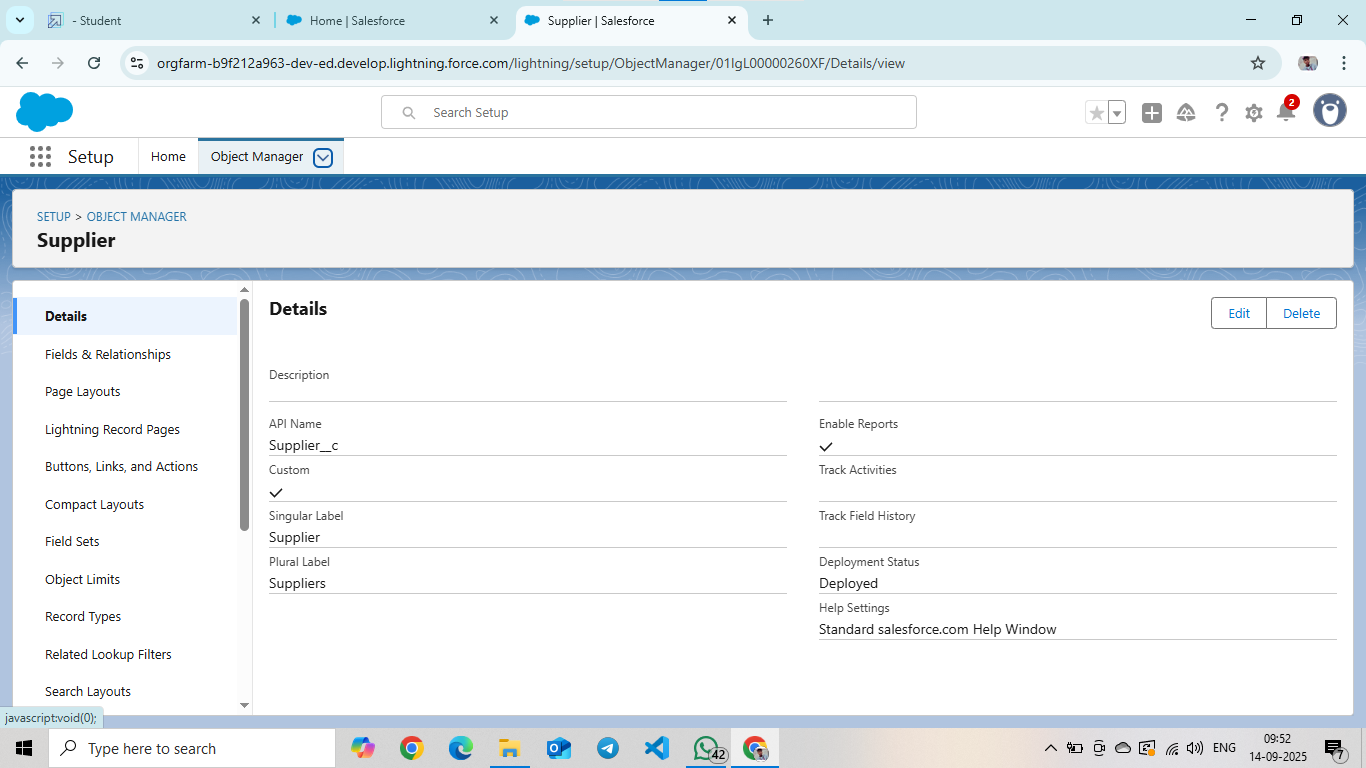
* **Product**
* **Purchase Order**
* **Order Item**
* **Inventory Transaction**
* **Supplier**



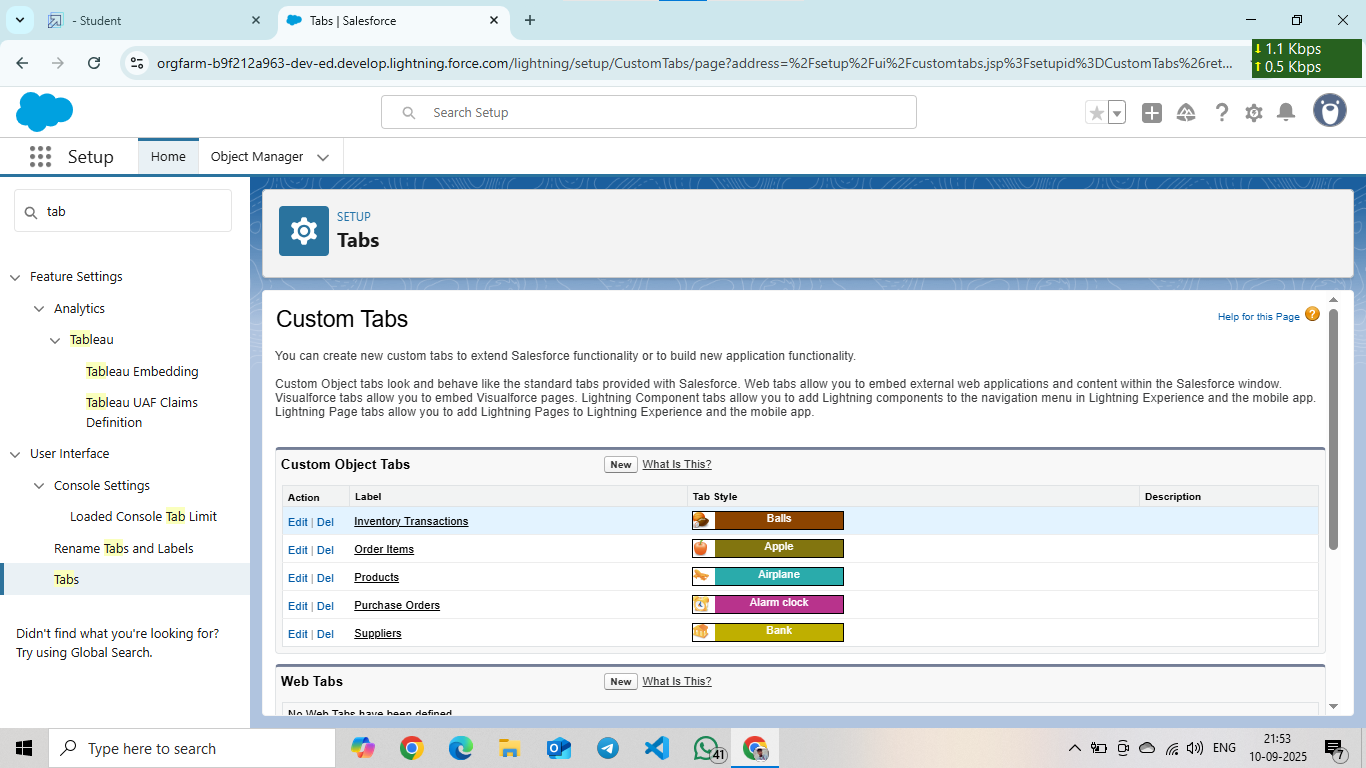




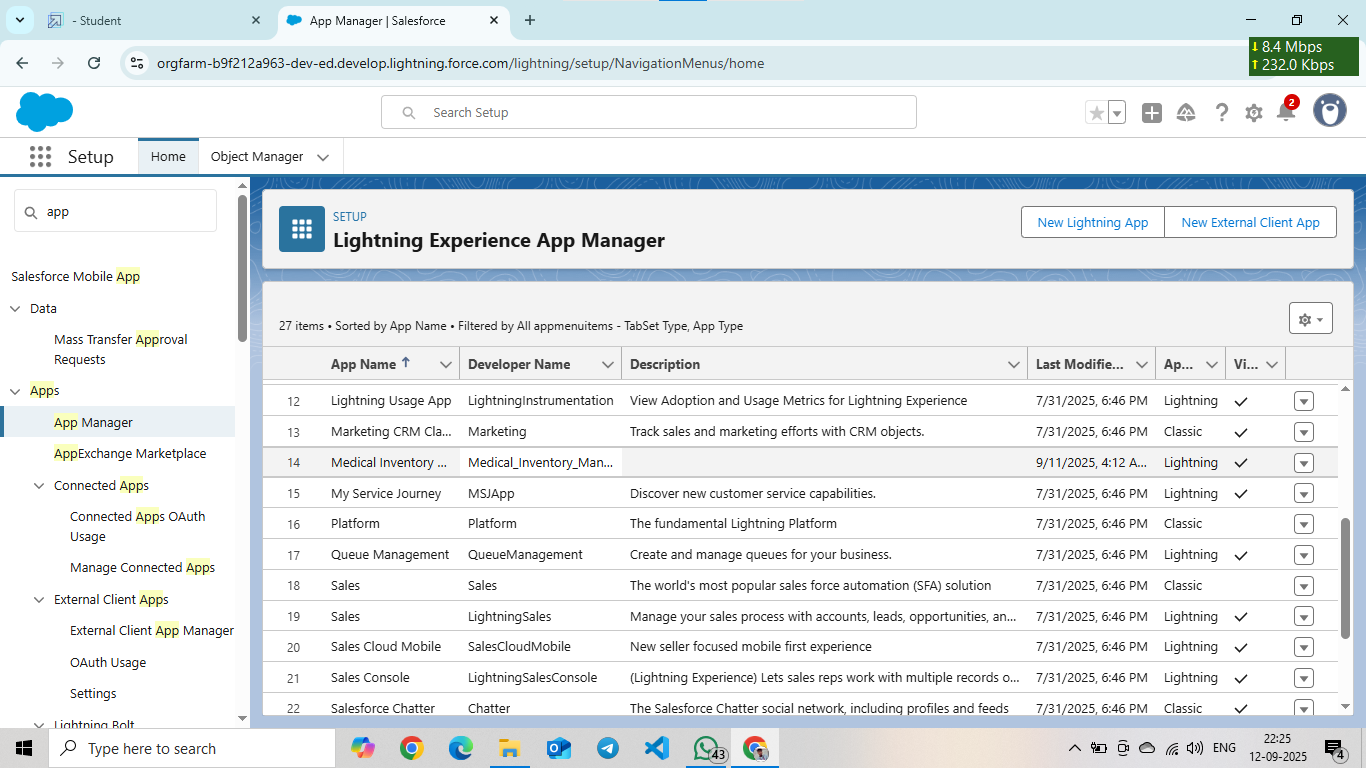




**Created Tabs:**

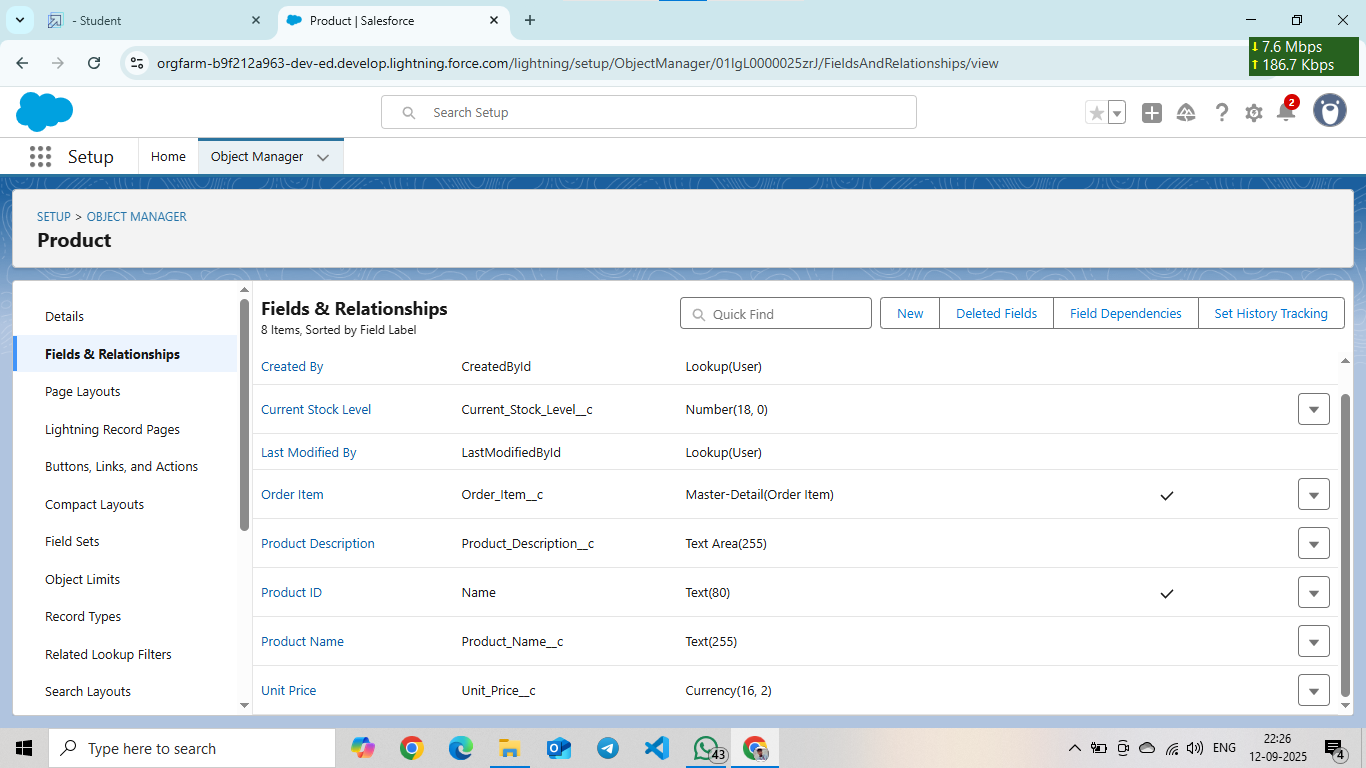
****

### **Created a Lightning App**

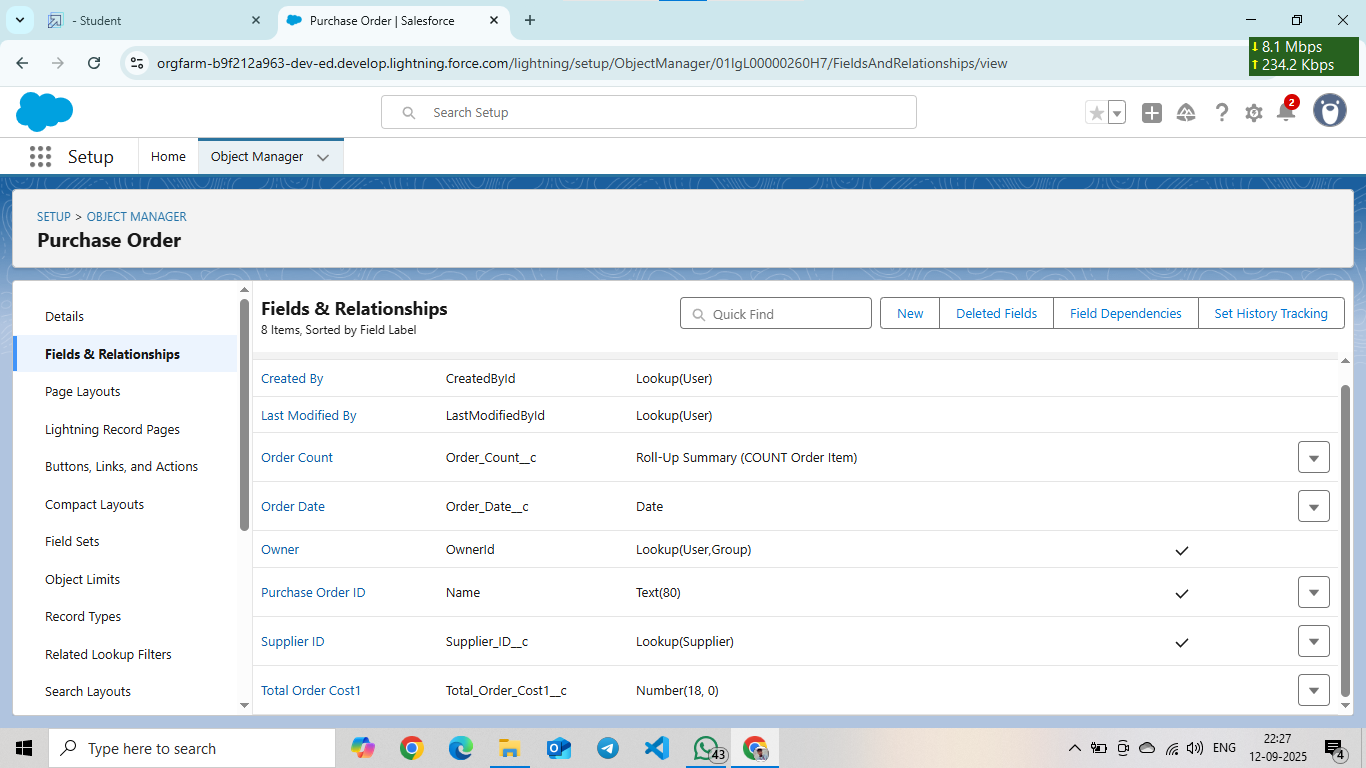


**FIELDS:**

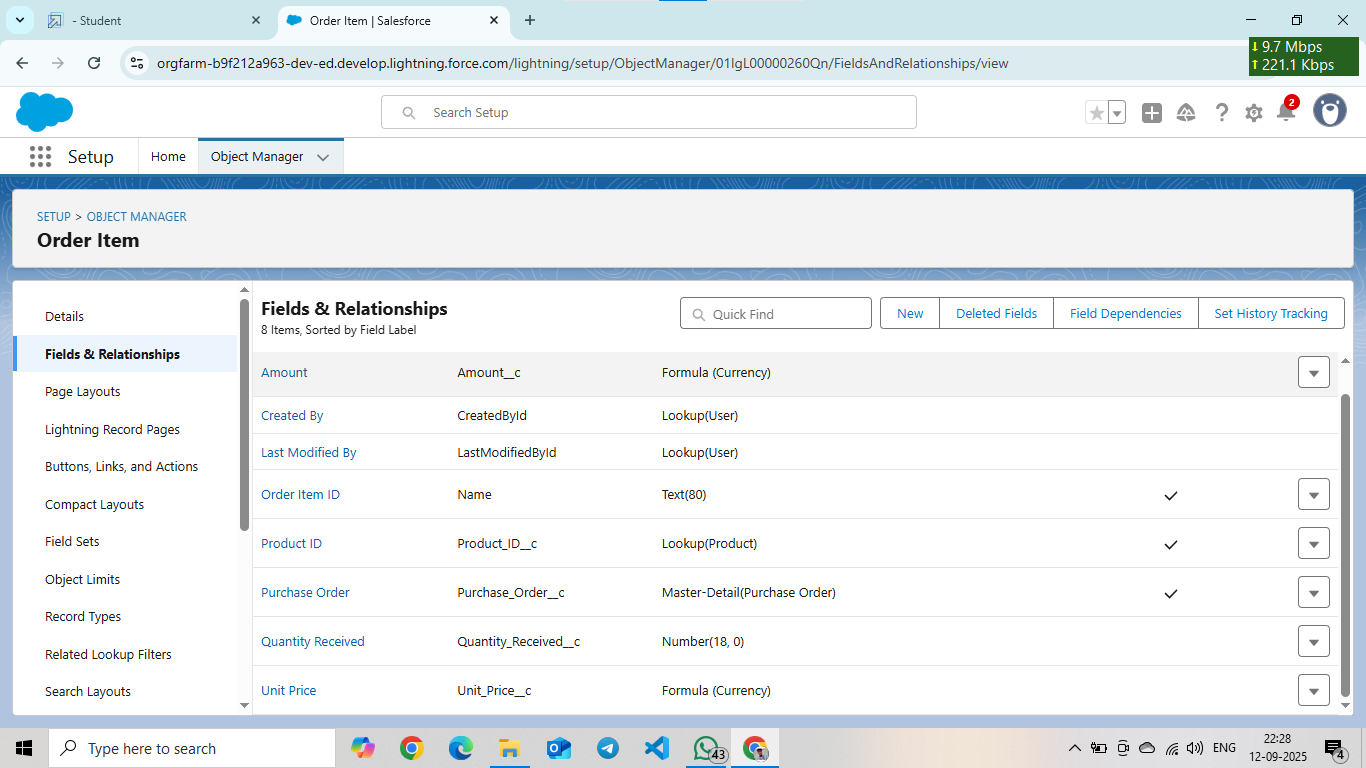
Creation of fields for the Product object:



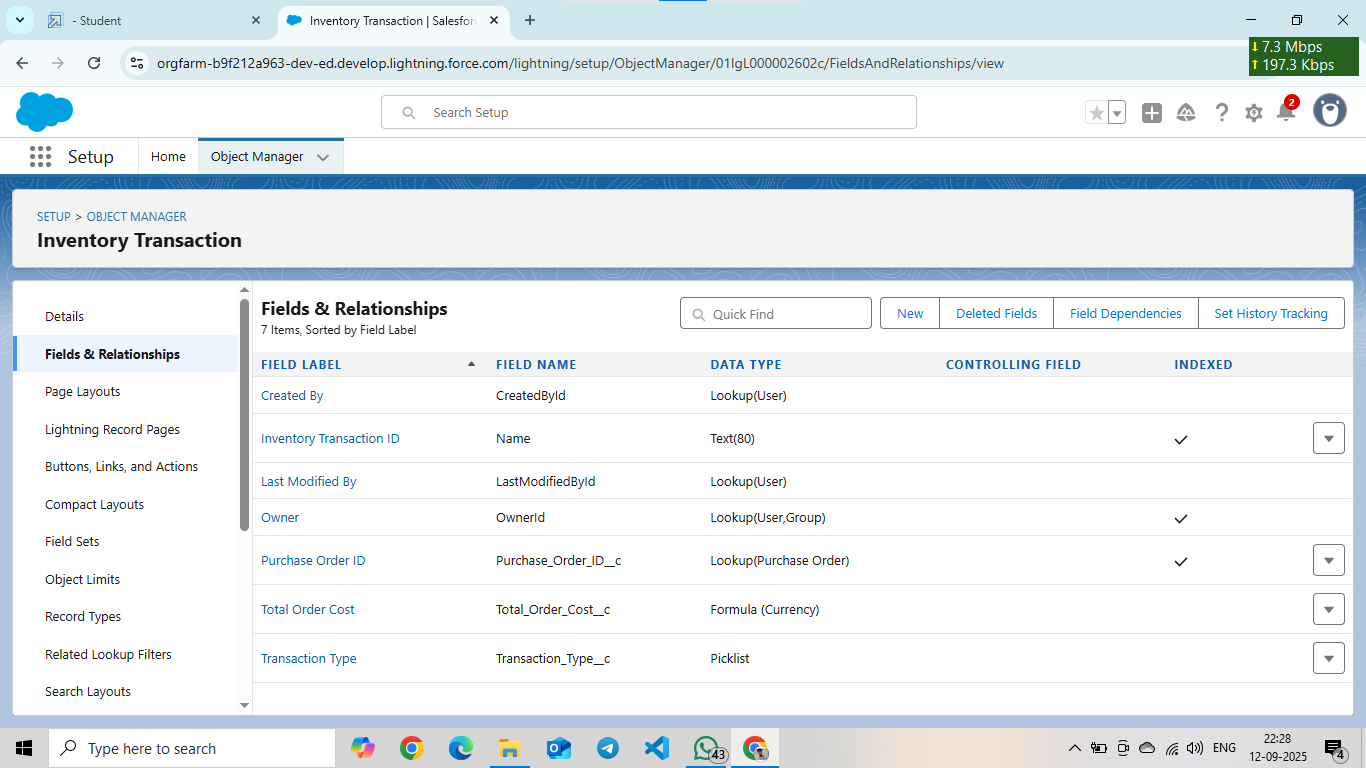
**Creation of fields for the Purchase Order object**



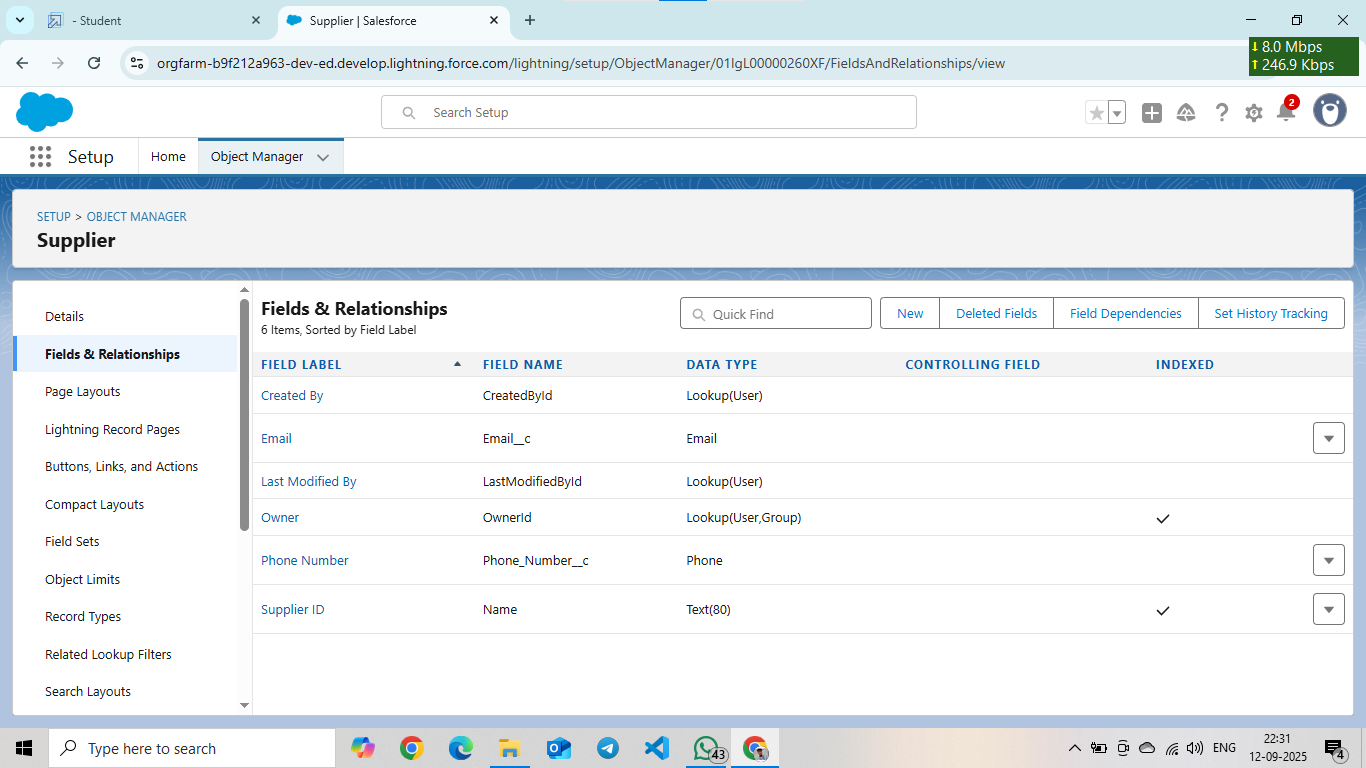
**Creation of fields for the Order Item object**

****

### **Creation of fields for the Inventory Transaction Object**



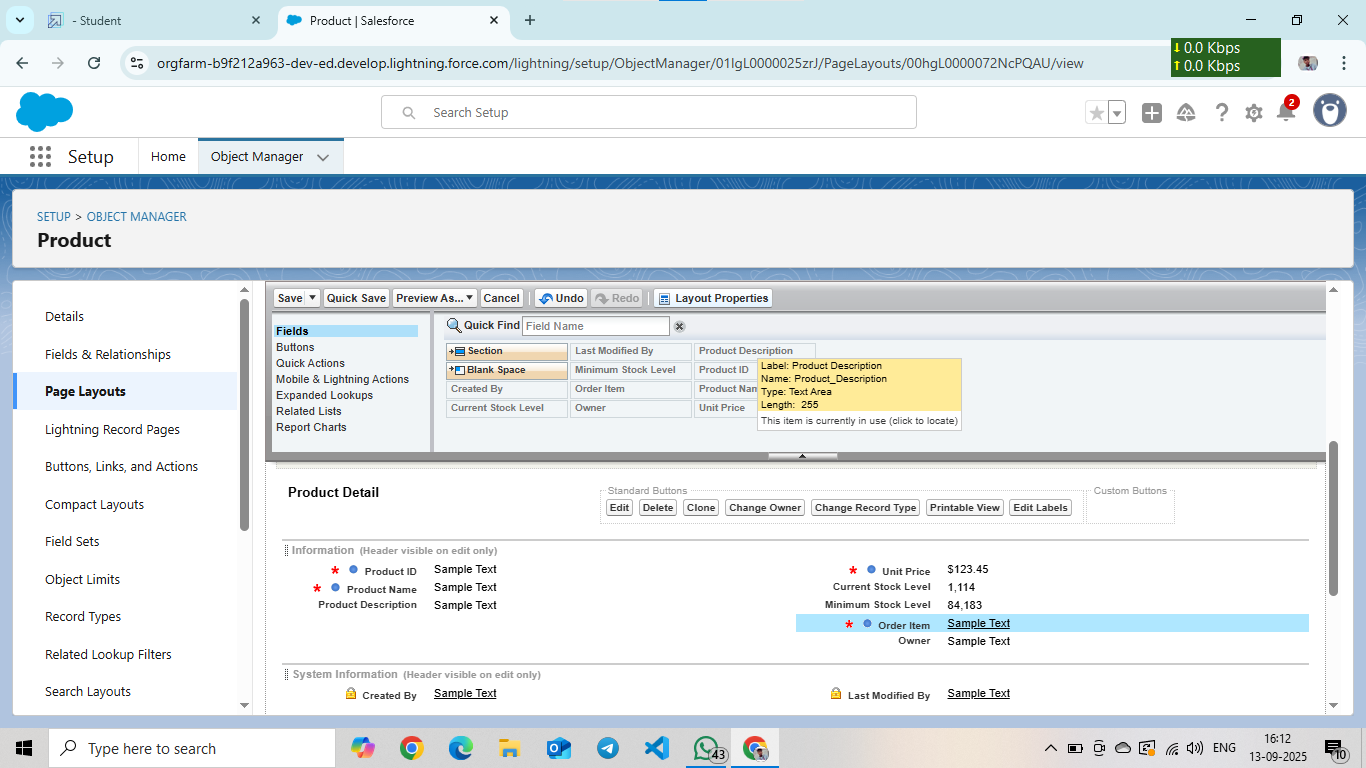
**Creation of fields for the Supplier object**



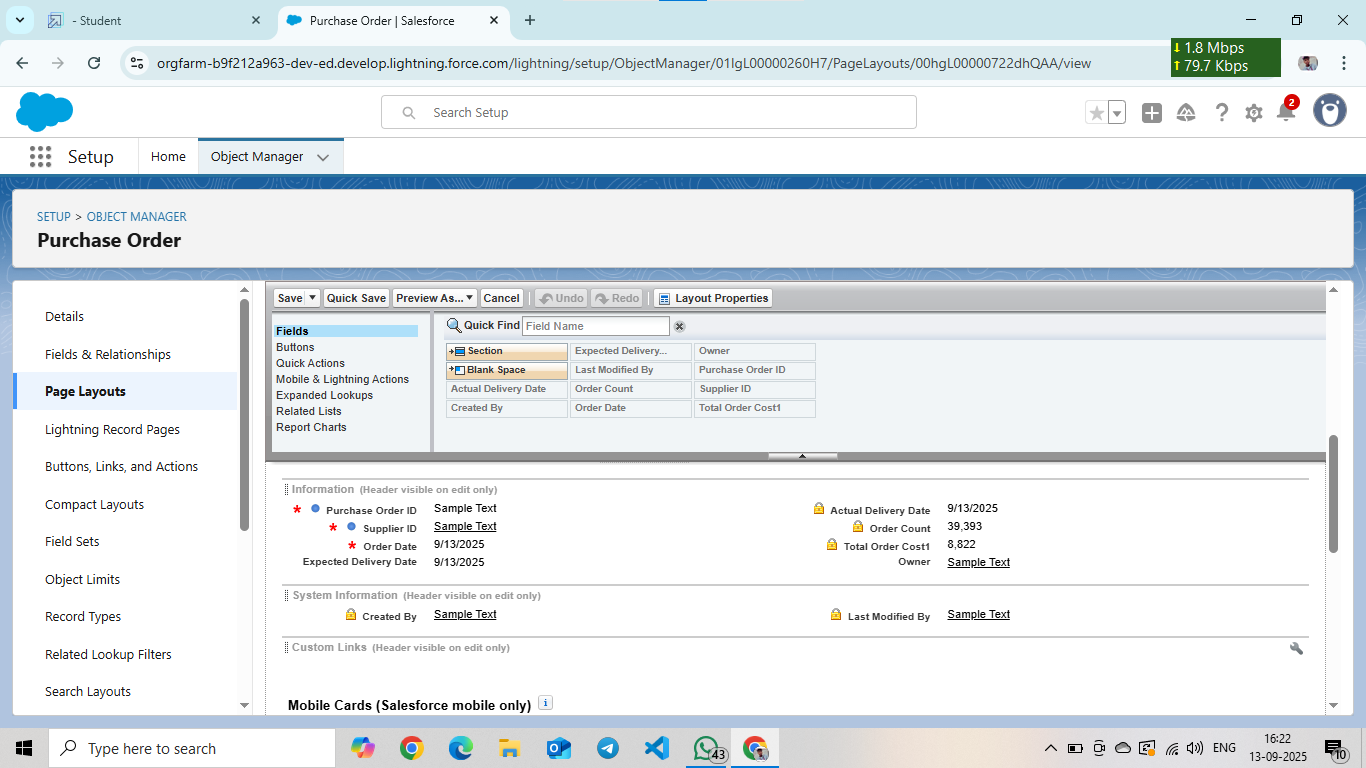
### **Page Layouts;**

**Editing of Page Layouts:**

Editing of Page Layout for the Product object



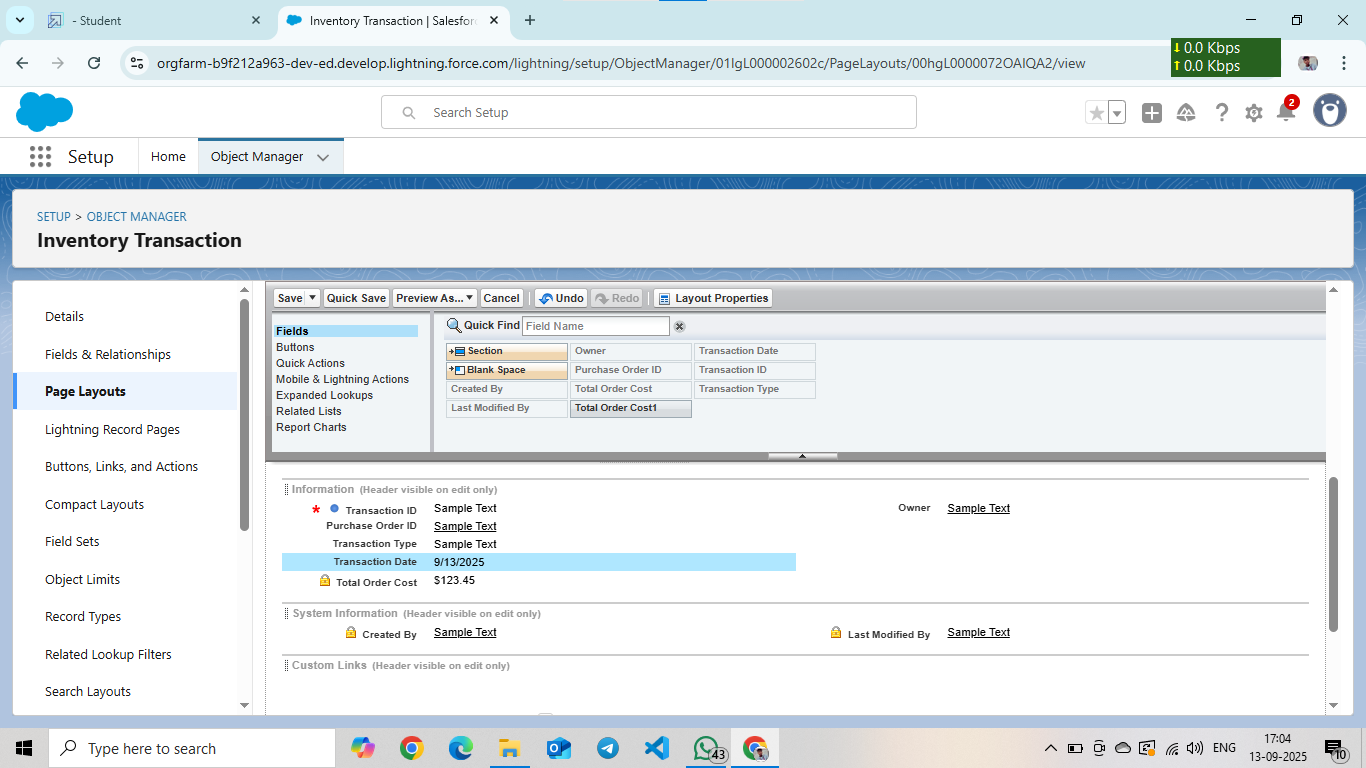
Editing of Page Layout for the Purchase Order object



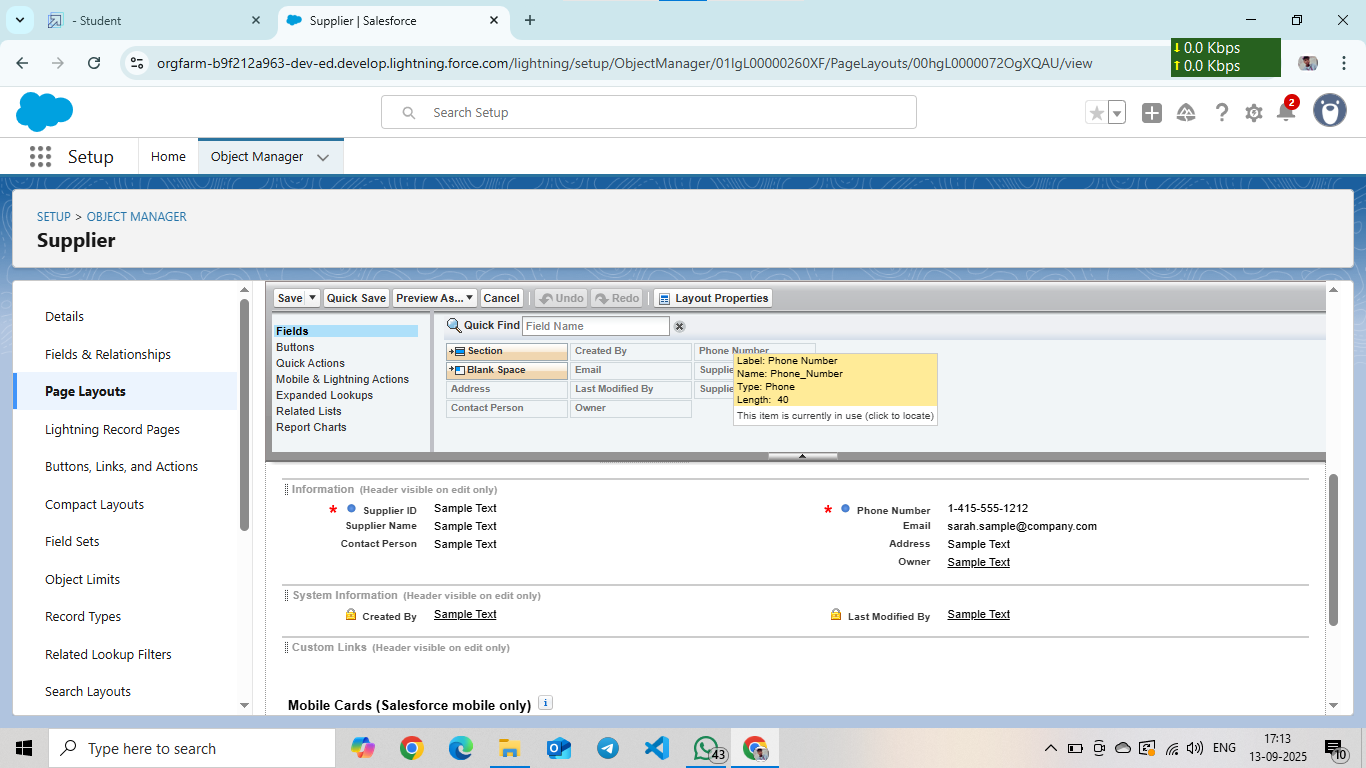
### Editing of Page Layout for the Order Item object



### Editing of Page Layout for the Inventory Transaction object

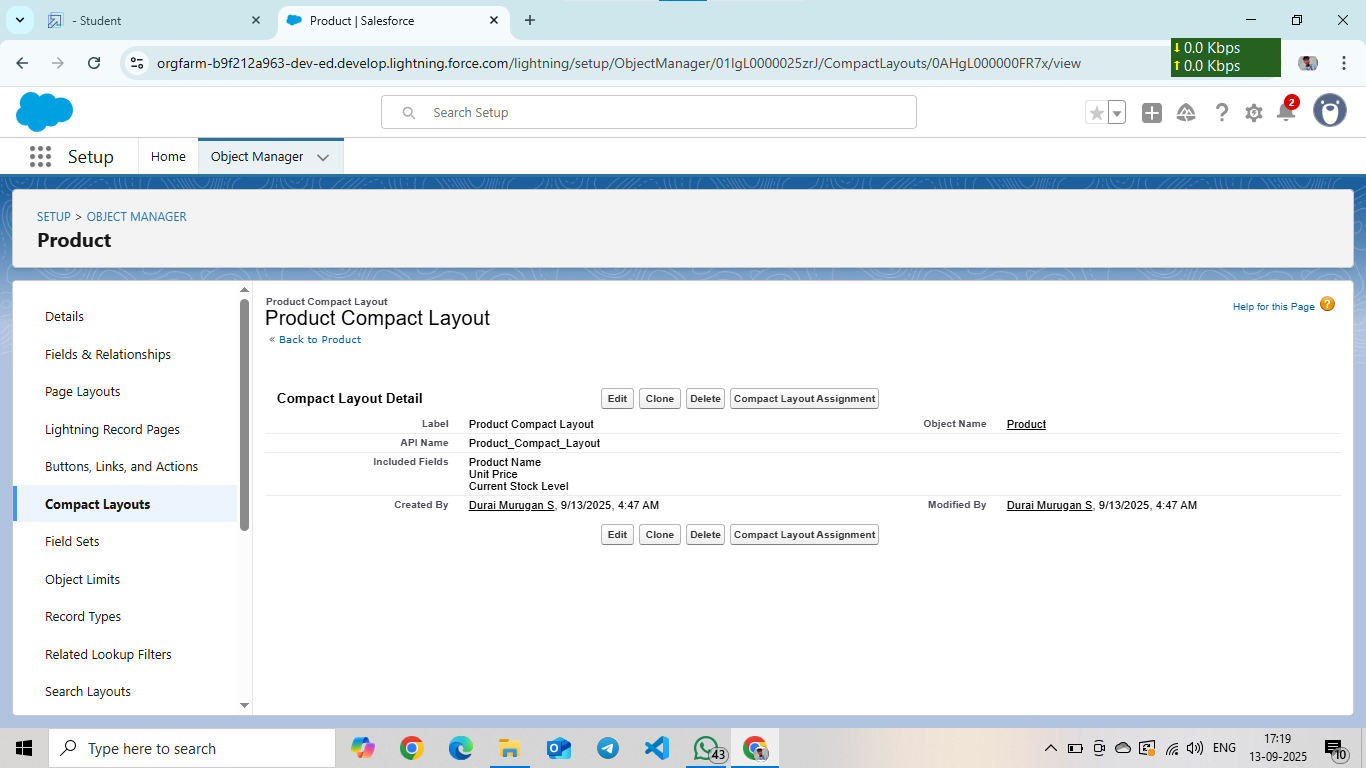


### Editing of Page Layout for the Supplier object

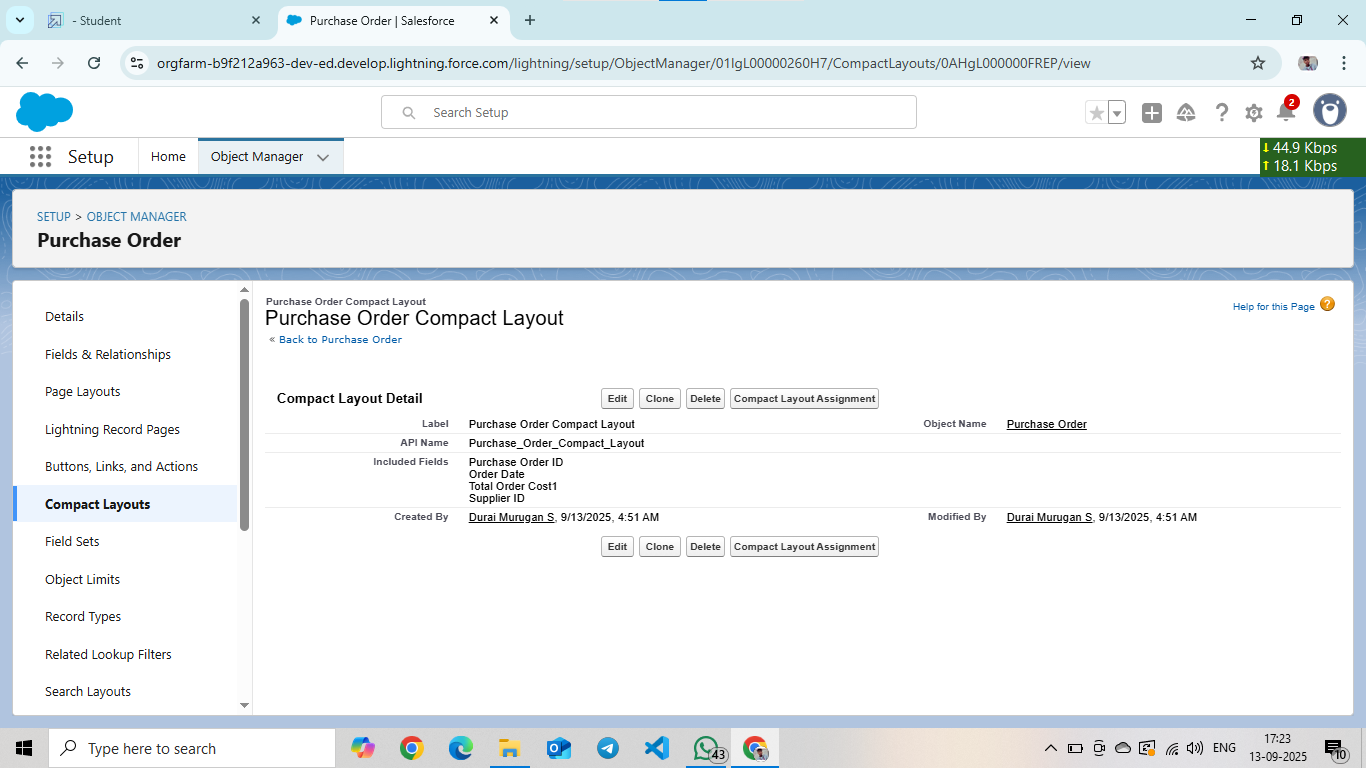


### **Compact Layouts**

Created a Compact Layout for the Product Object

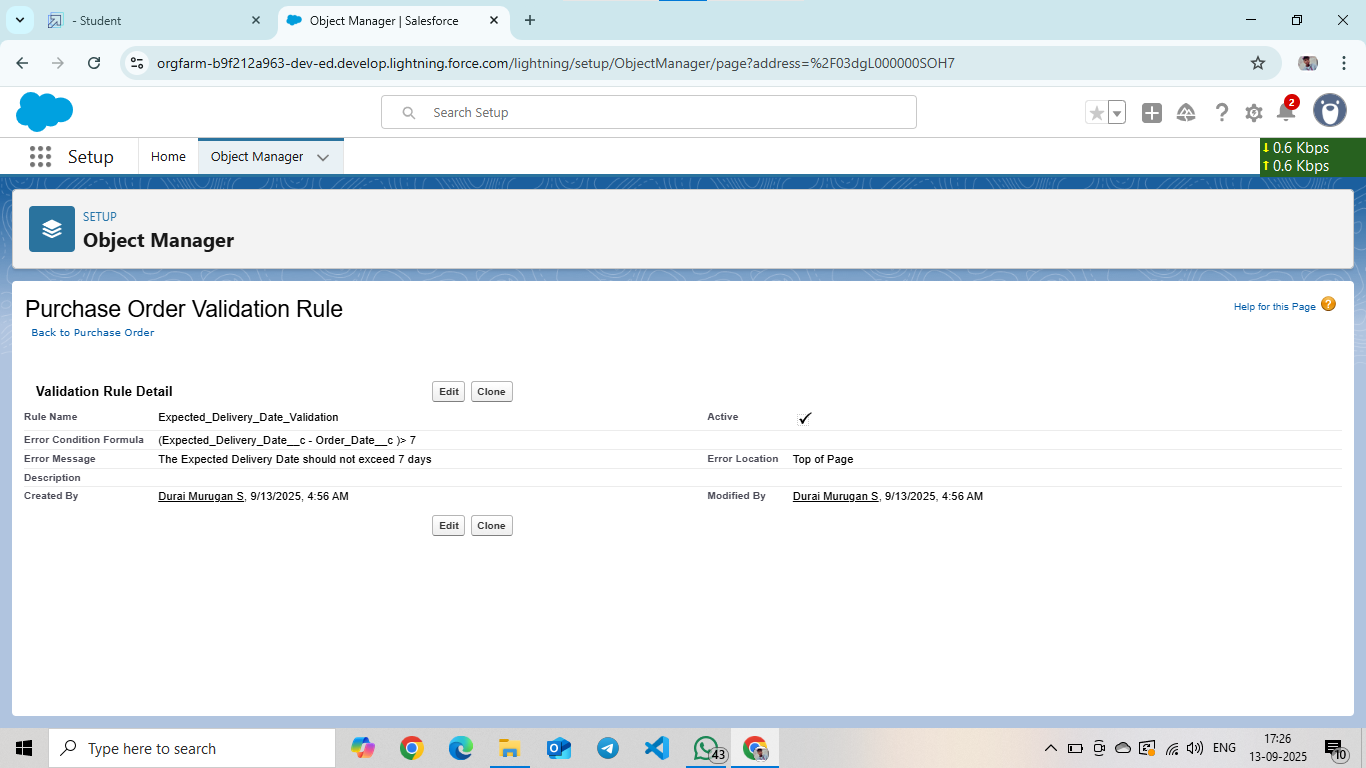


### Created a Compact Layout for the Purchase Order Object



### **Validation Rules**

Created an Expected Delivery Date Validation rule to an Employee Object

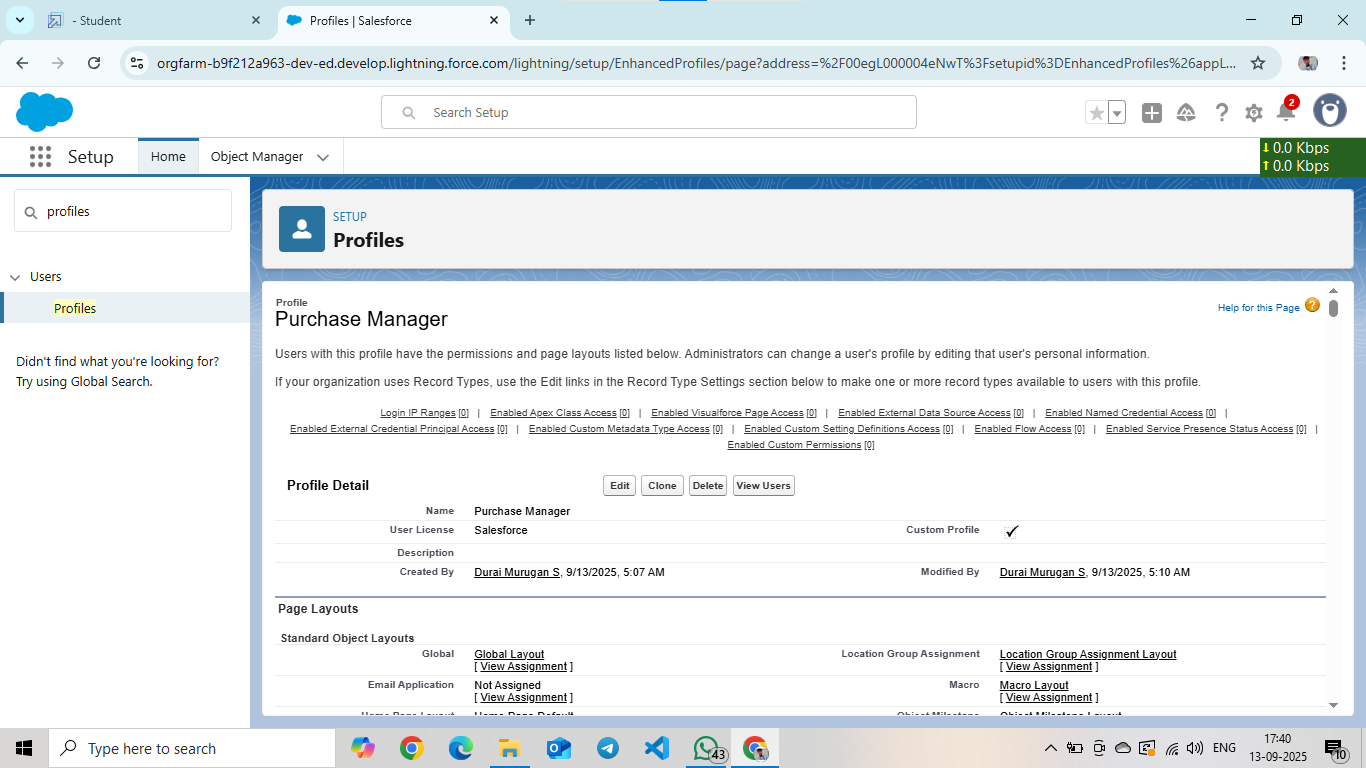


### **Profiles**

Created an Inventory Manager Profile

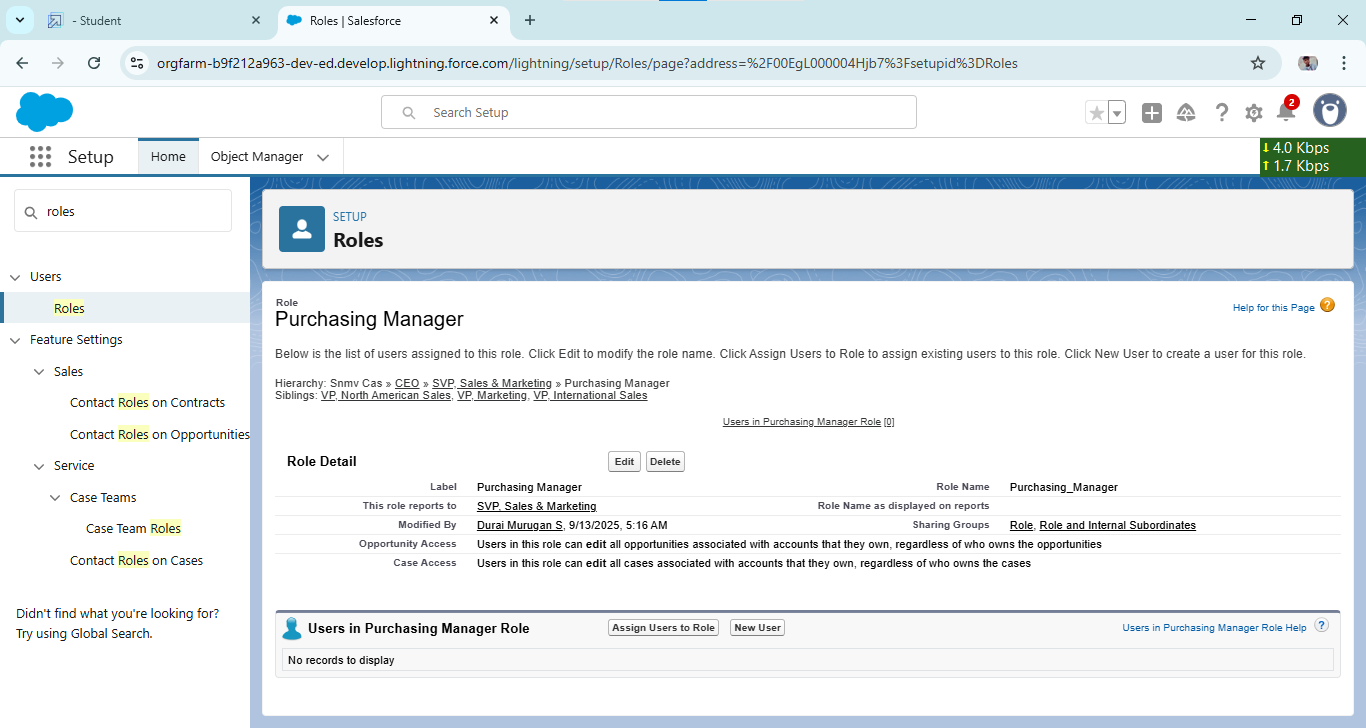


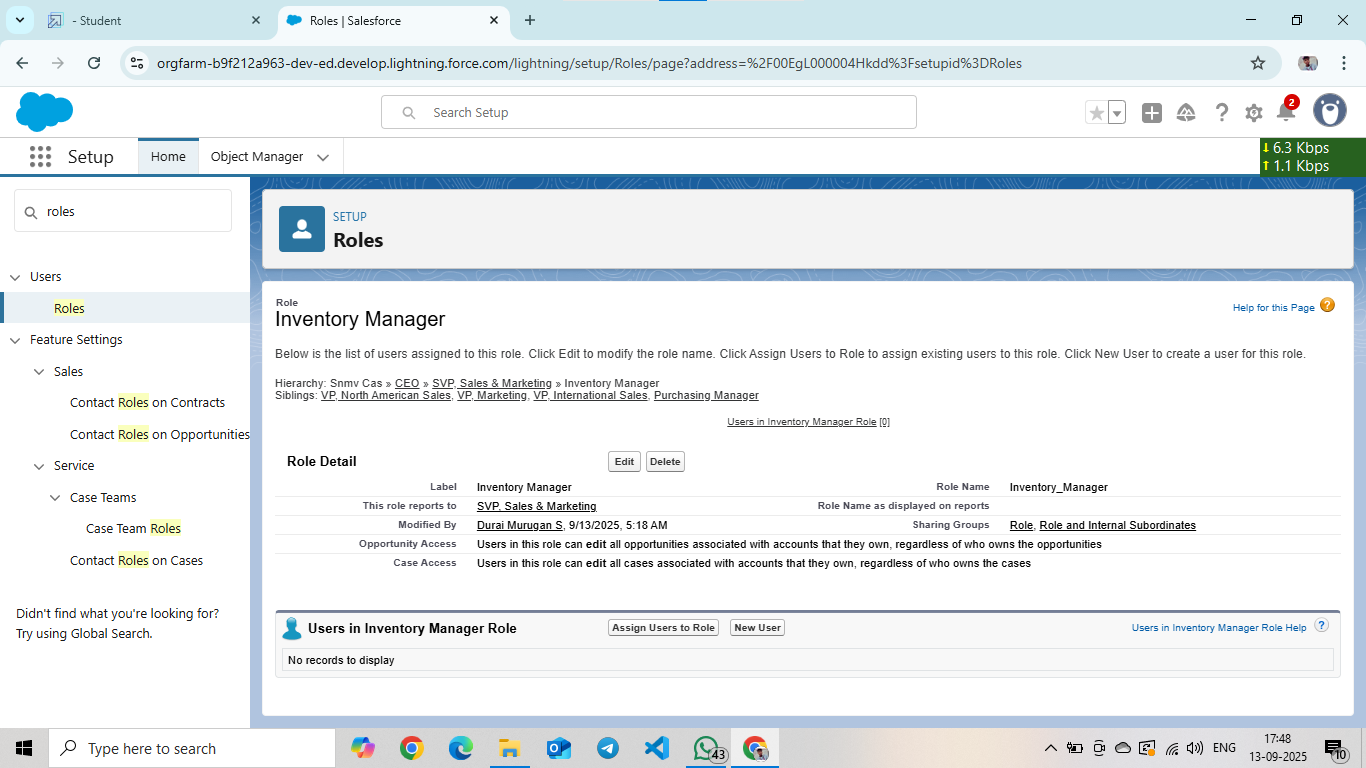
### Created a Purchase Manager Profile



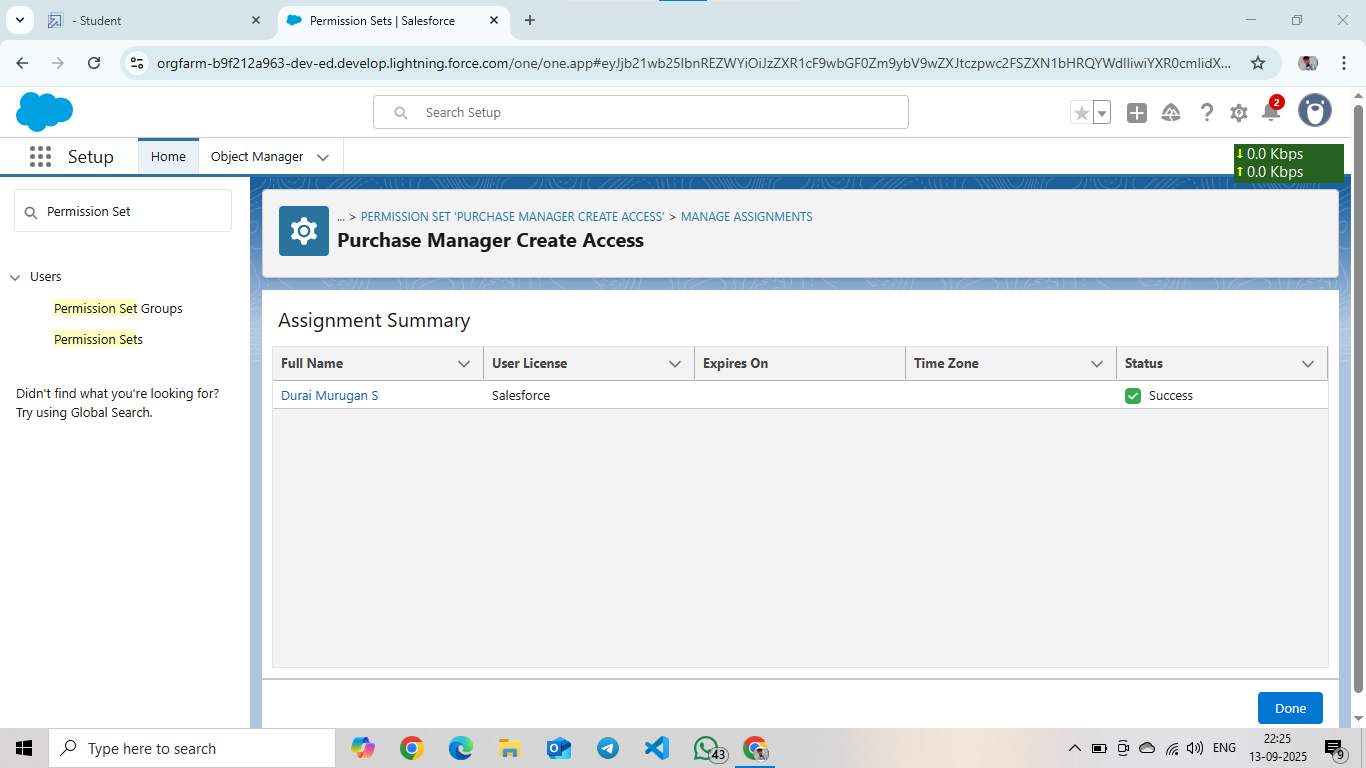
### **Roles**

Created a Purchasing Manager Role (Purchasing Manager)



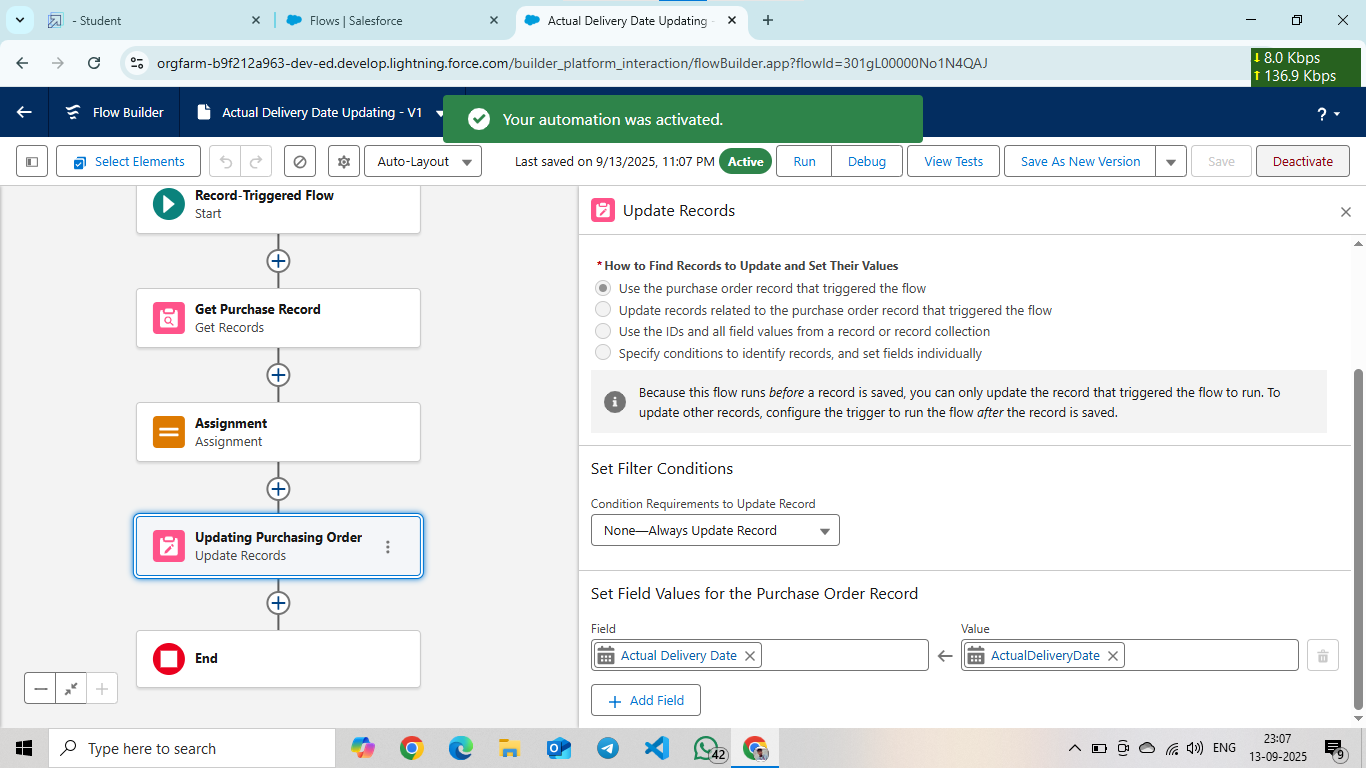
Created a Purchasing Manager Role (Inventory Manager)

### **Permission Sets**

Create a Permission Set

**Flows**

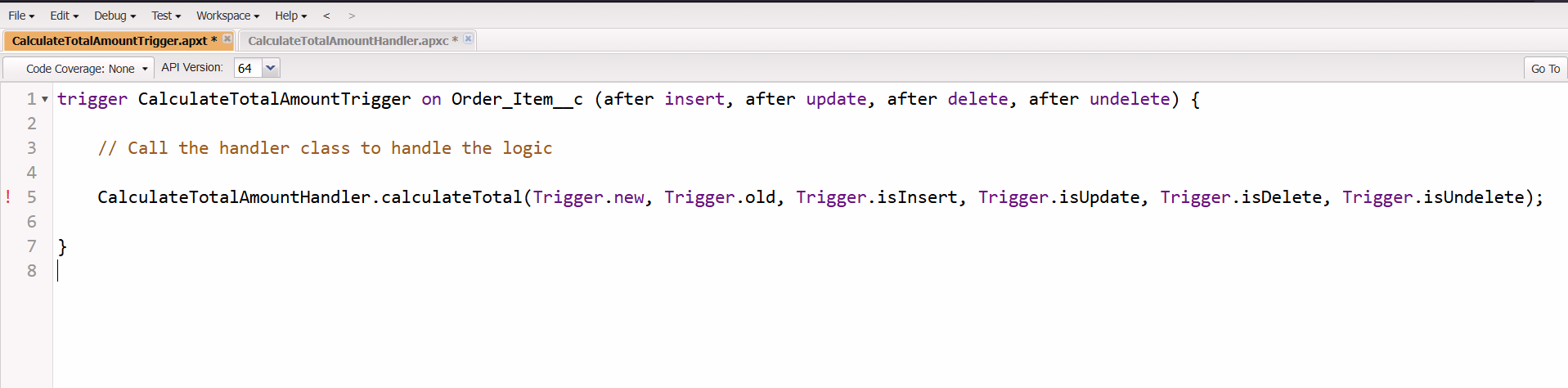
Created Flow to update the Actual Delivery Date



### 

### **Triggers**

Create an Apex Trigger



### Coding

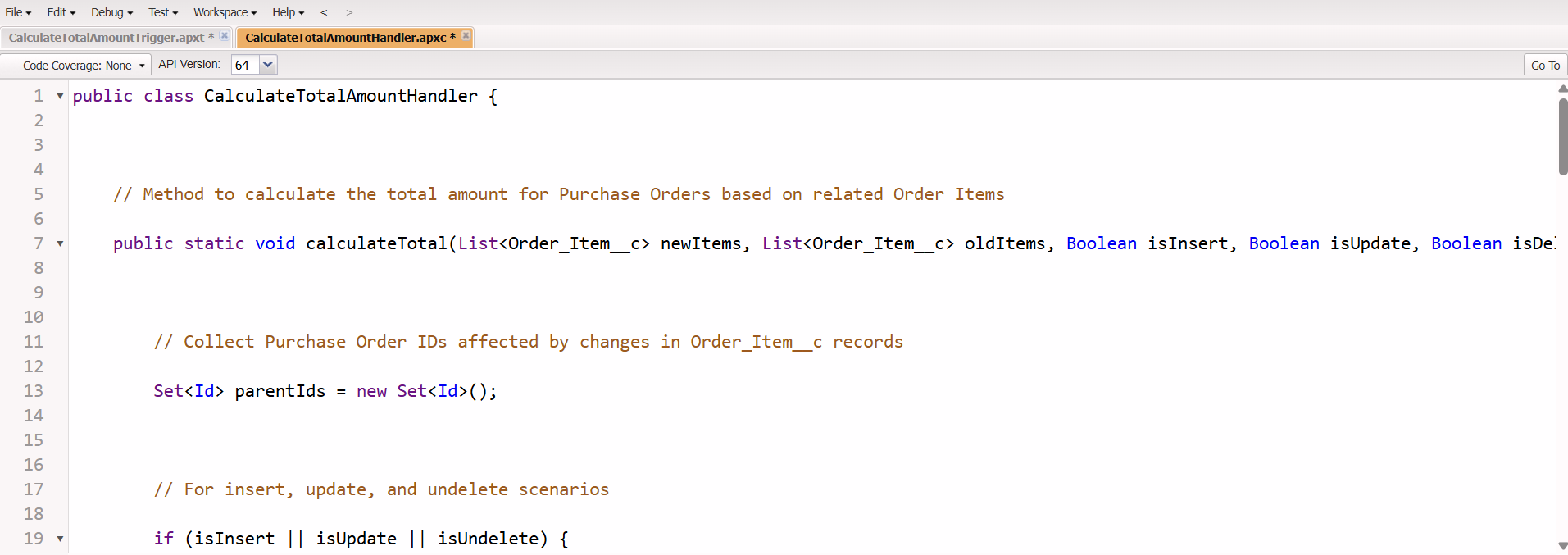
trigger CalculateTotalAmountTrigger on Order\_Item\_\_c (after insert, after update, after delete, after undelete) {

    // Call the handler class to handle the logic

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

}

### Create an Apex Class



### Coding

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 ordItem : newItems) {

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

            }

 }

   if (isUpdate || isDelete) {

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

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

            }

}

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

 if (!parentIds.isEmpty()) {

  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\_\_c

            ]:

            for (AggregateResult aggr : aggrList) {

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

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

                purchaseToUpdateMap.put(purchaseOrderId, totalAmount);

            }

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);

            }

            if (!purchaseToUpdate.isEmpty()) {

                update purchaseToUpdate;

            }

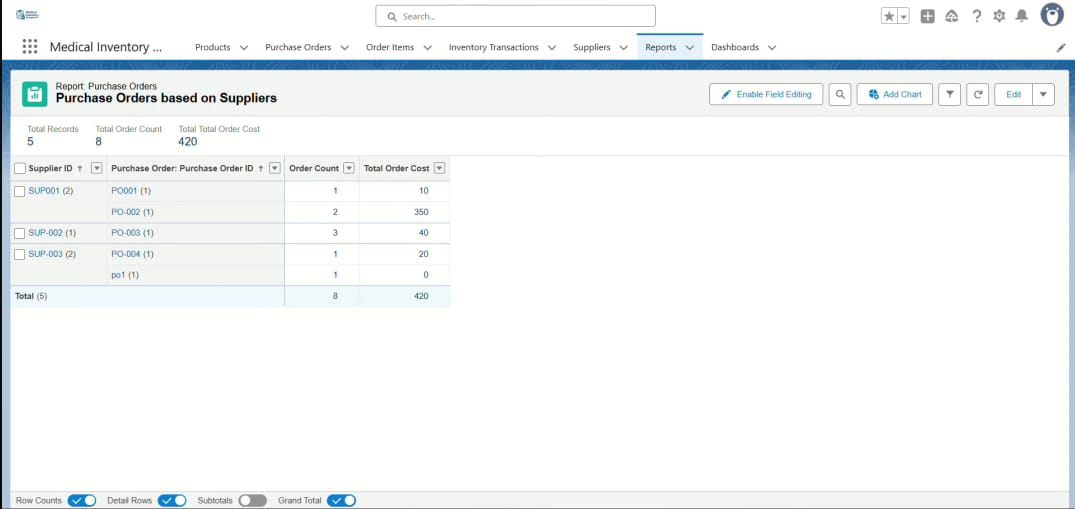
        }

    }

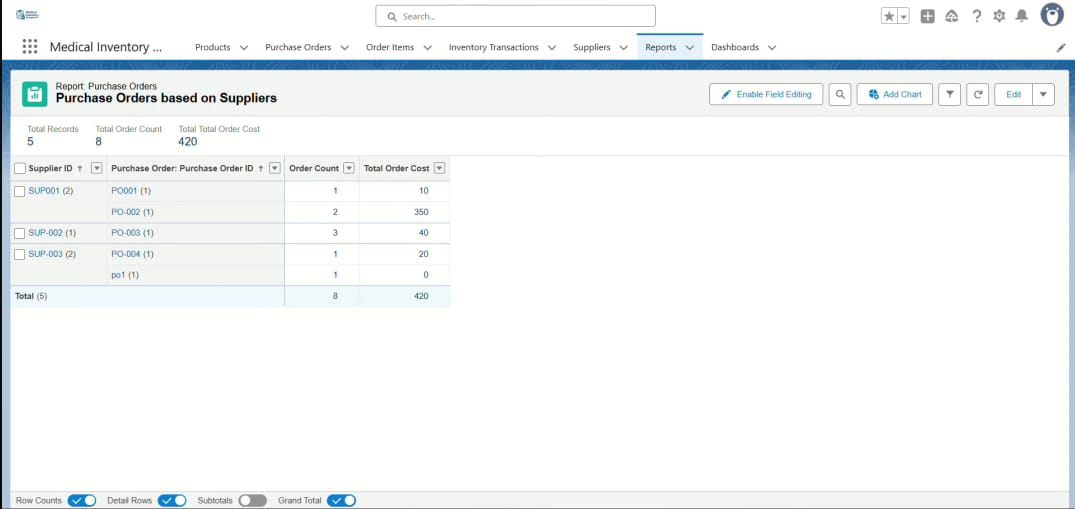
}

### **Reports**

Created a Purchase Orders based on Suppliers (Summary) Report

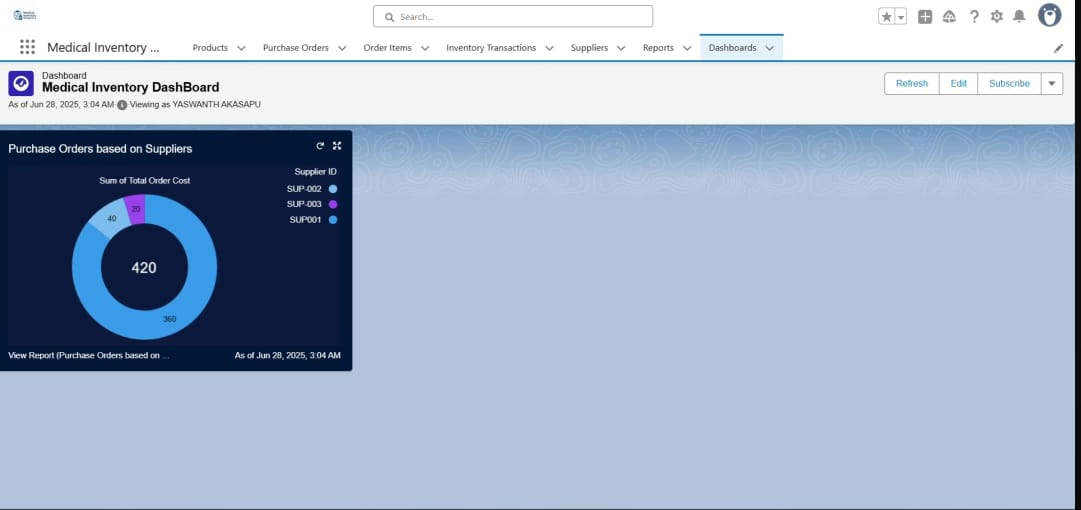


### Created a Complete Purchase Details Report



**Dashboards**

Creation and view of Dashboard



### **Conclusion**

The Salesforce-based Medical Inventory Management System successfully streamlines and automates the key operational aspects of medical inventory. By integrating supplier management, purchase order tracking, product cataloging, and expiry date monitoring, the system ensures accuracy, efficiency, and reliability in handling medical supplies. It reduces the risk of errors, prevents the use of expired products, and enhances decision-making through comprehensive reports and analytics. Overall, this project demonstrates how Salesforce can be effectively utilized to improve healthcare inventory operations, contributing to better resource management and patient safety.