**MEDICAL INVENTORY MANAGEMENT**

**College Name: shri nehru maha vidyalaya college of arts and science**

**College Code: bru26**

**TEAM ID: NM2025TMID20616**

**TEAM MEMBERS:**

**Team Leader :THANIKACHALAM P**

**EMAIL :** [**thanikachalam382@gmail.com**](mailto:lykasfrd@gmail.com)

**Team member 1 : MUTHULAKSHMI K**

**EMAIL :** [**muthulakshmi90974@gmail.com**](mailto:mr.adhi.k7@gmail.com)

**Team member 2 : JAYARAM N**

**EMAIL :** [**jayaramrx100boy@gmail.com**](mailto:lifethrottlers@gmail.com)

**Team member 3: ESWARAPANDIYAN P**

**EMAIL :** [**ajeezthachu558@gmail.com**](mailto:mirumiruthini@gmail.com)

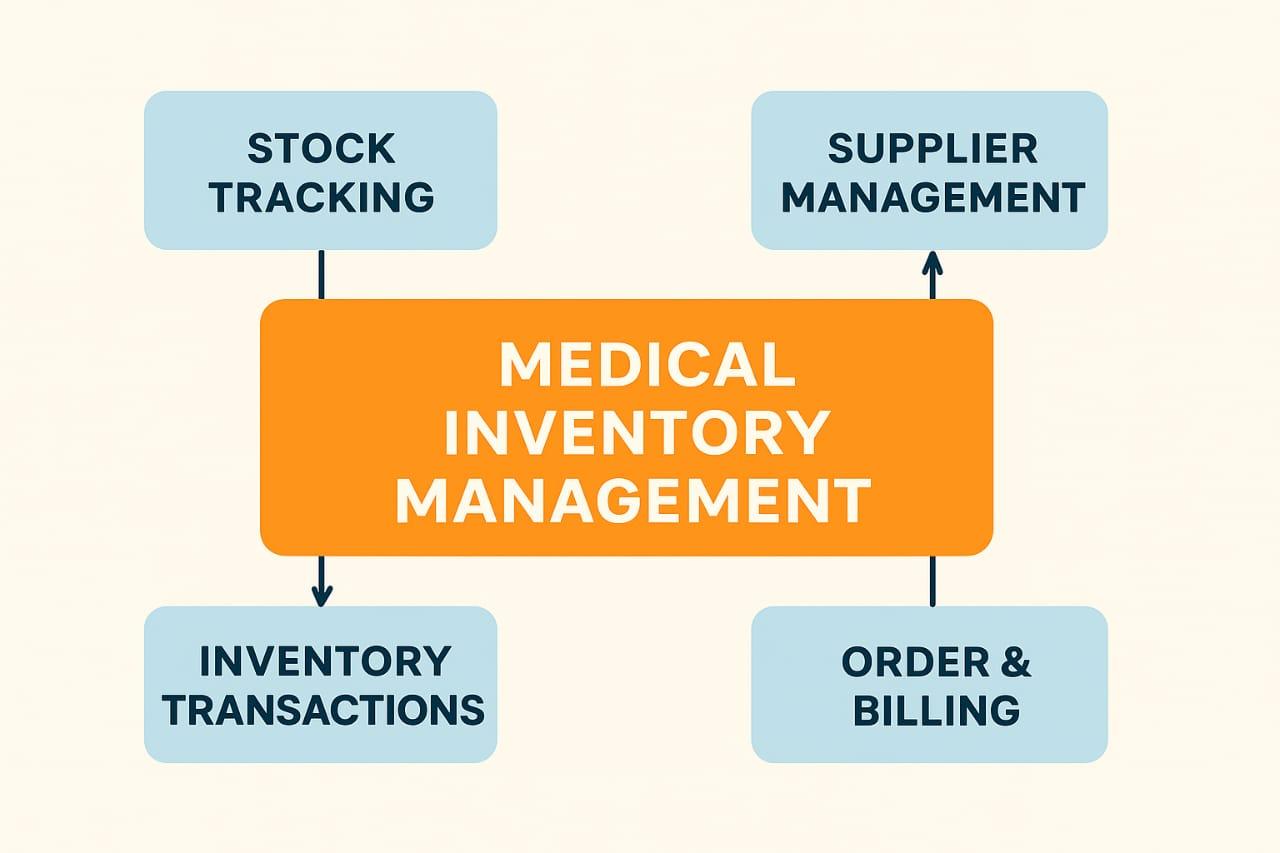
**Team member 4: KUMAR V**

**EMAIL :** [**Vijaykumar959721@gmail.com**](mailto:mirumiruthini@gmail.com)

# **INTRODUCTION**

## **Project Overview**

The Medical Inventory Management System is a Salesforce-based website designed to help hospitals, pharmacies, and clinics manage their medical supplies efficiently. It enables users to track stock levels, monitor expiry dates, manage suppliers, and automate reordering, reducing manual errors and preventing shortages or wastage.



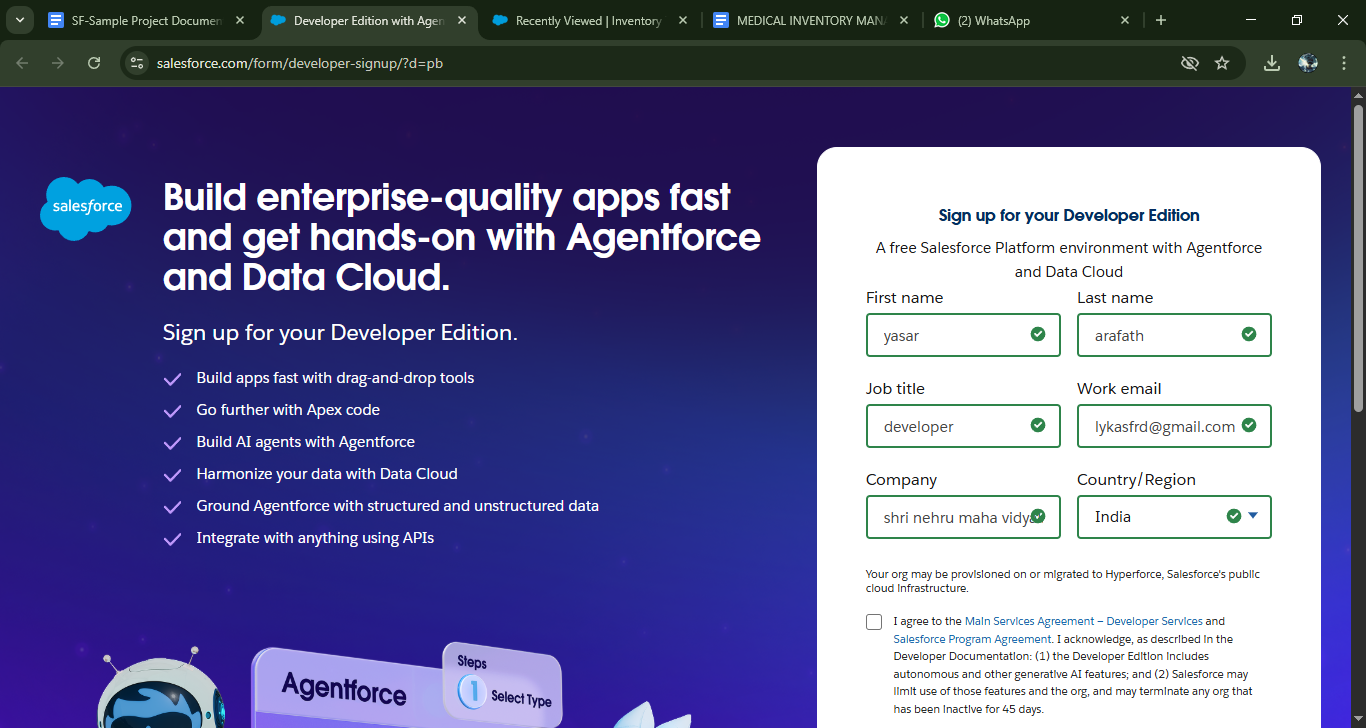
**1.2 Purpose**

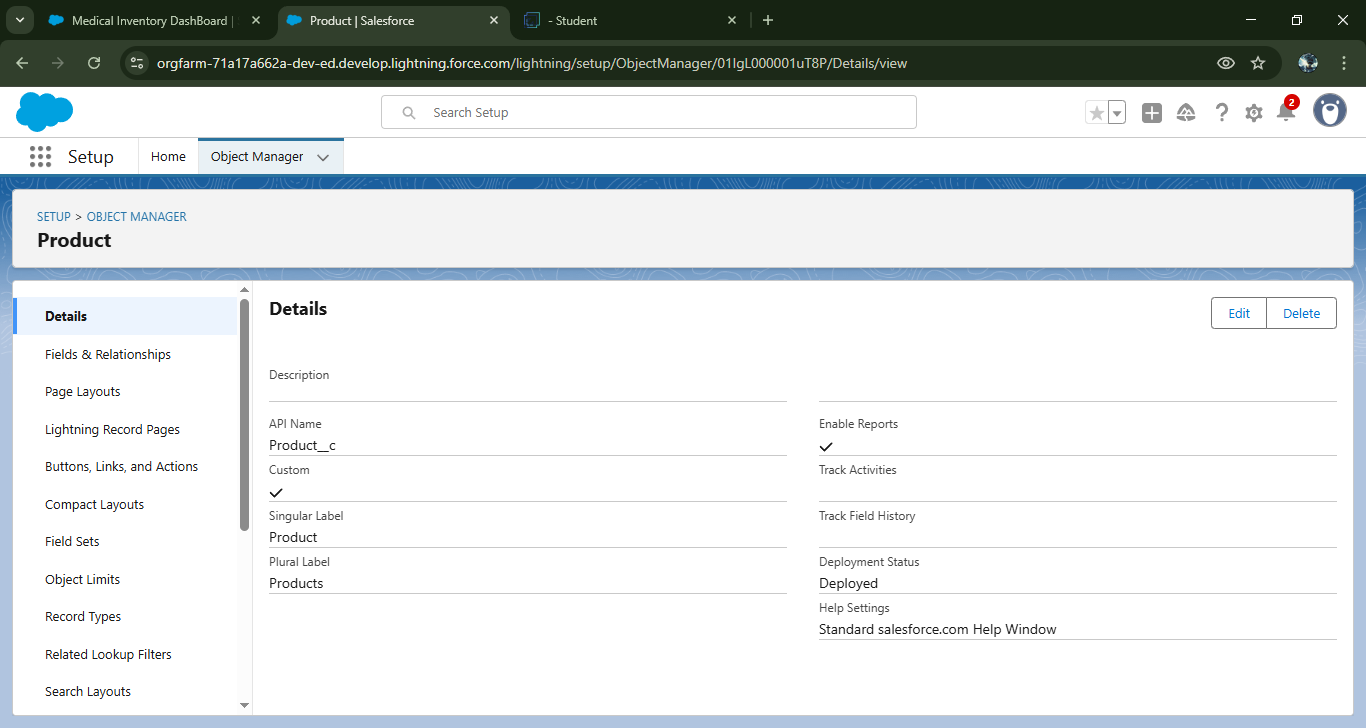
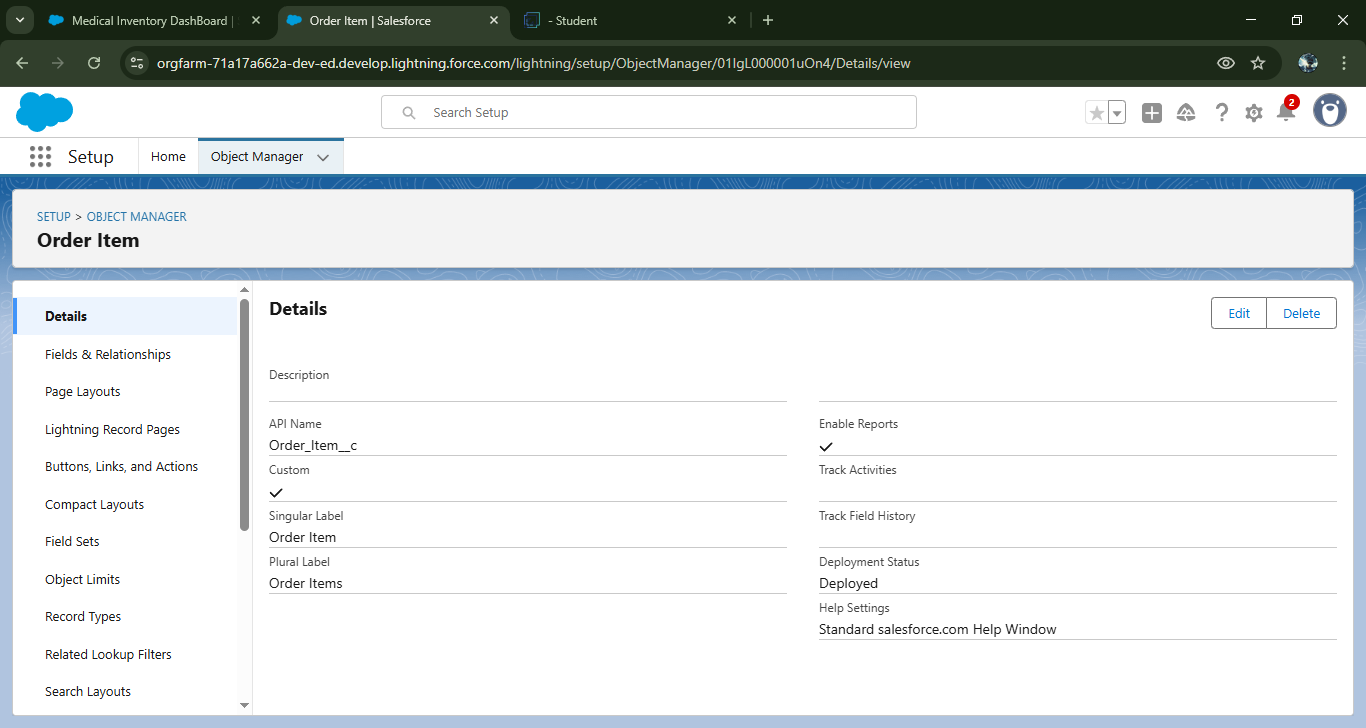
The main objective of the project is to help healthcare organizations manage medical supplies effectively. It reduces manual errors, prevents stock shortages or wastage, and ensures timely availability of essential medicines and equipment. The system also improves accuracy, saves time, and provides better control over suppliers and inventory data.

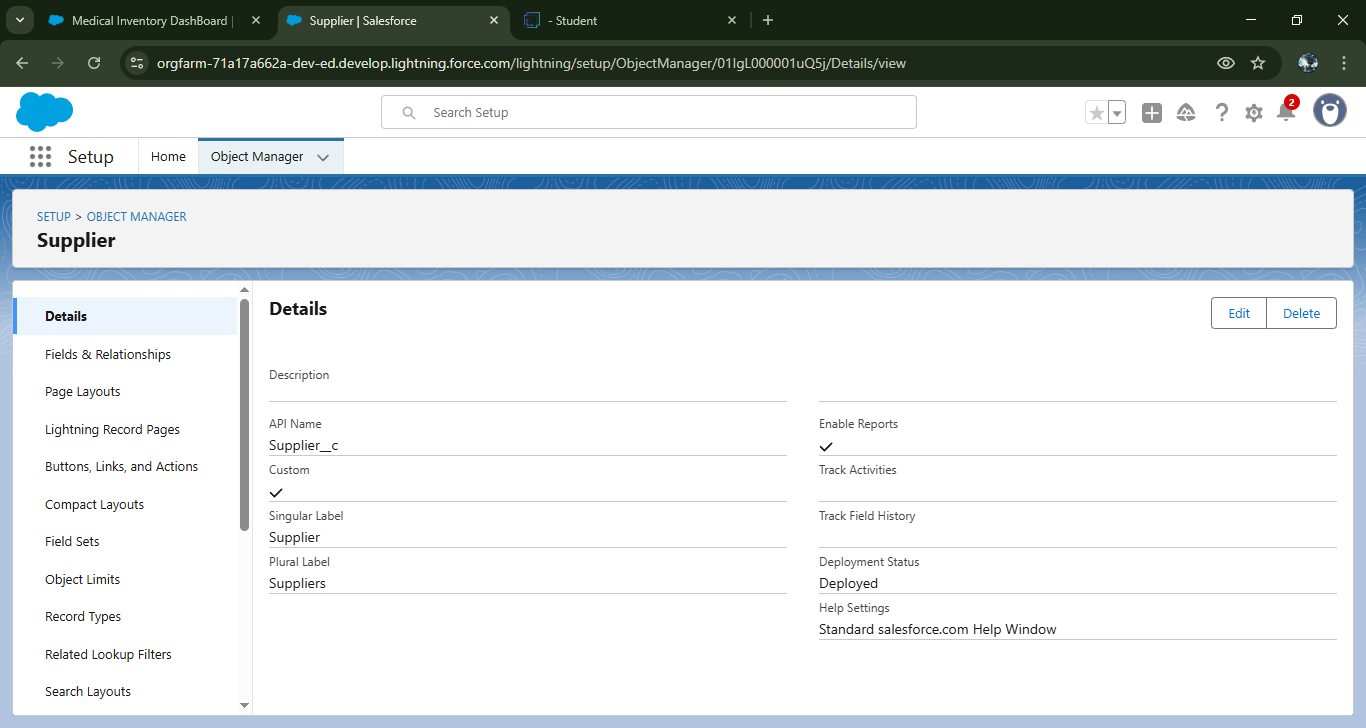
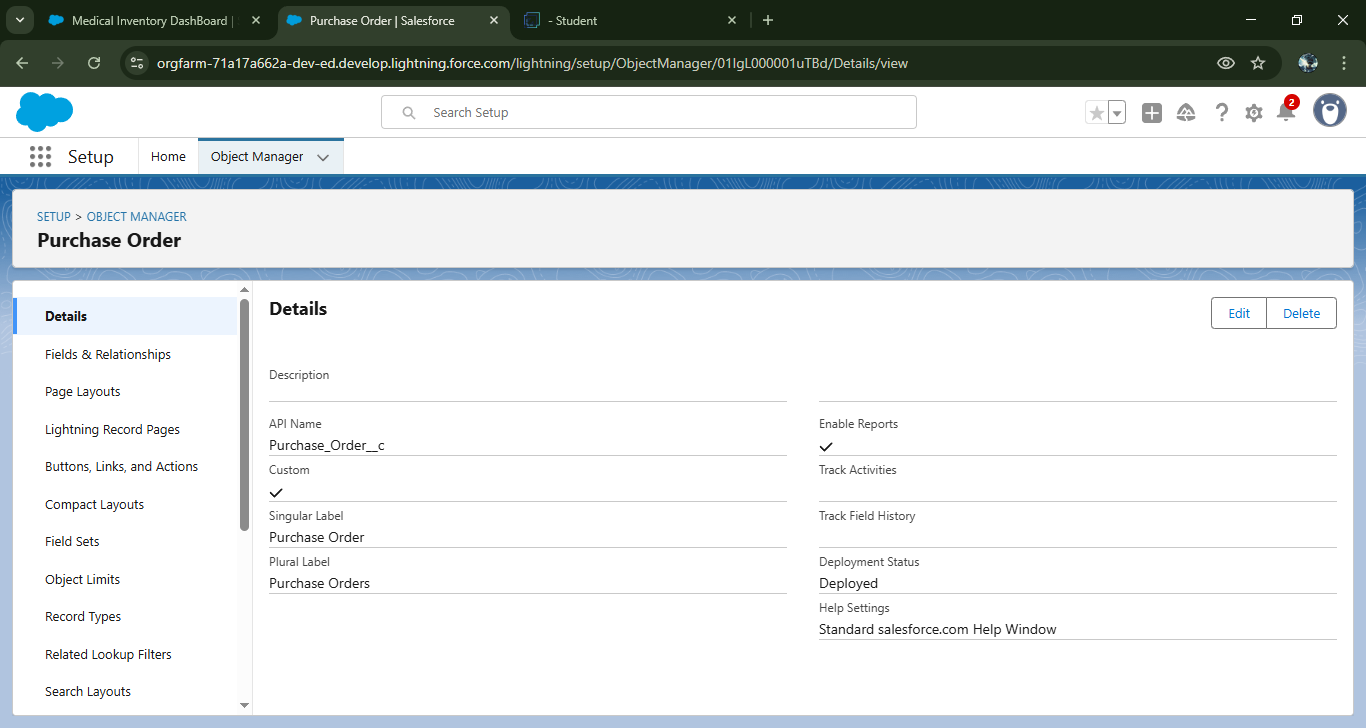
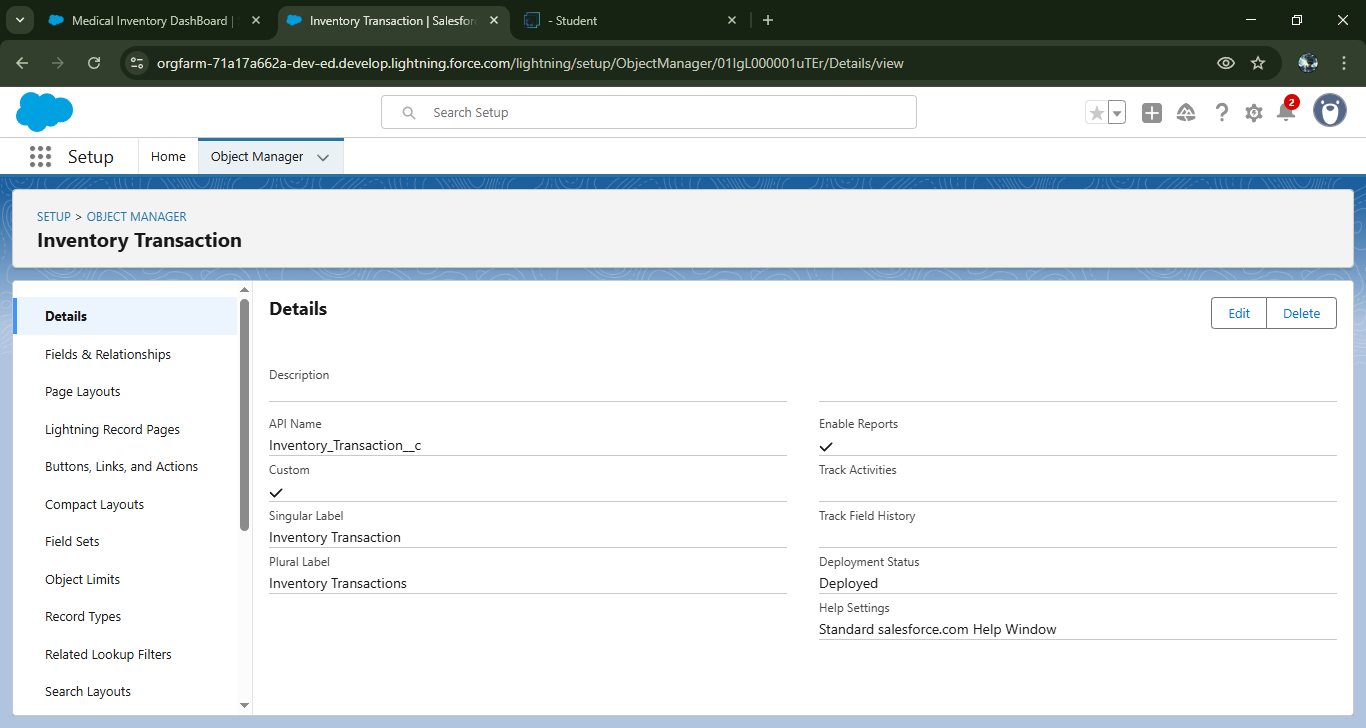
# **DEVELOPMENT PHASE**

### **Creating Developer Account:**

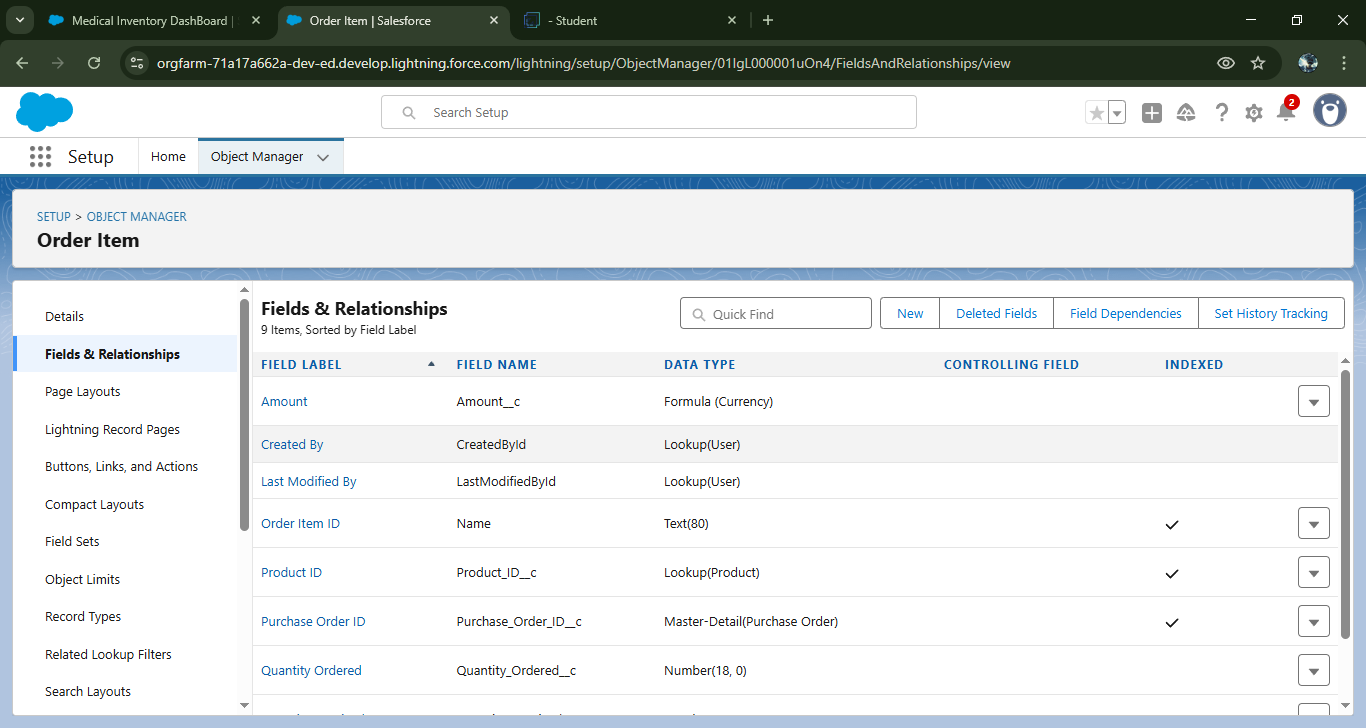
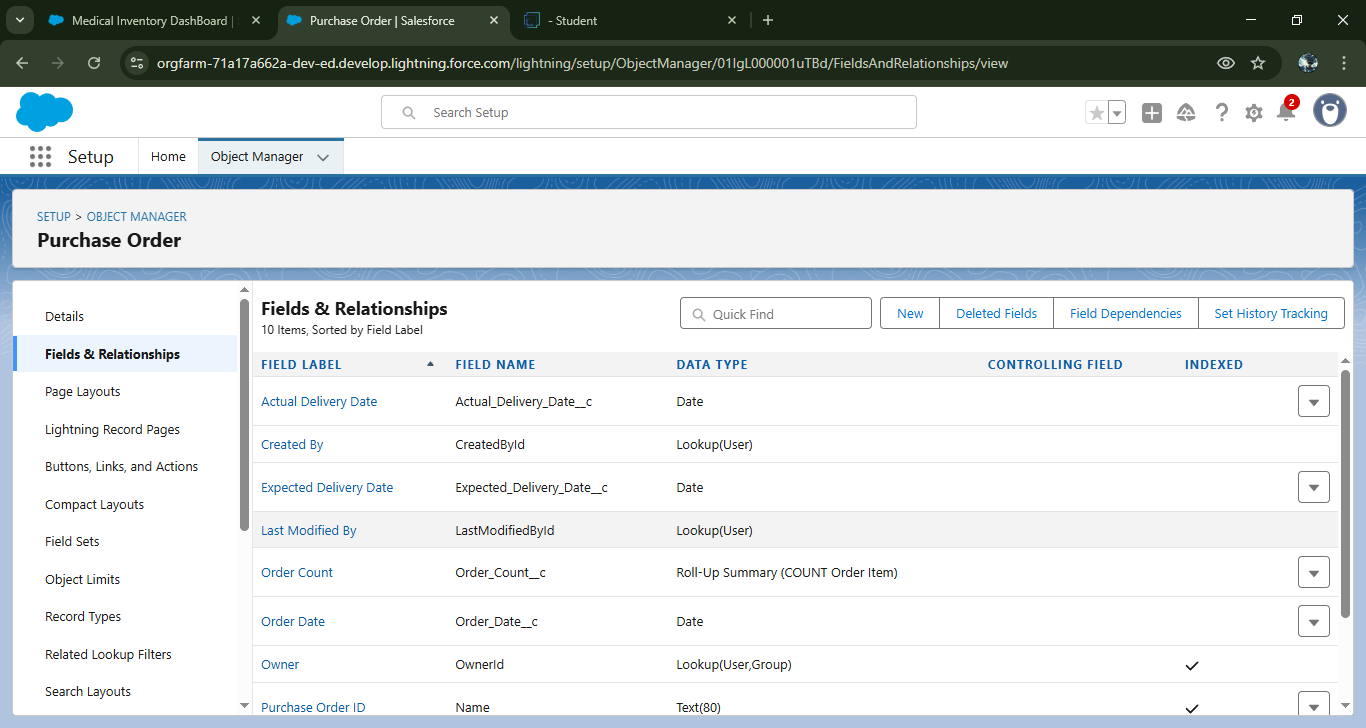
By using this URL **-** [**https://www.salesforce.com/form/developer-signup/?d=pb**](https://www.salesforce.com/form/developer-signup/?d=pb)

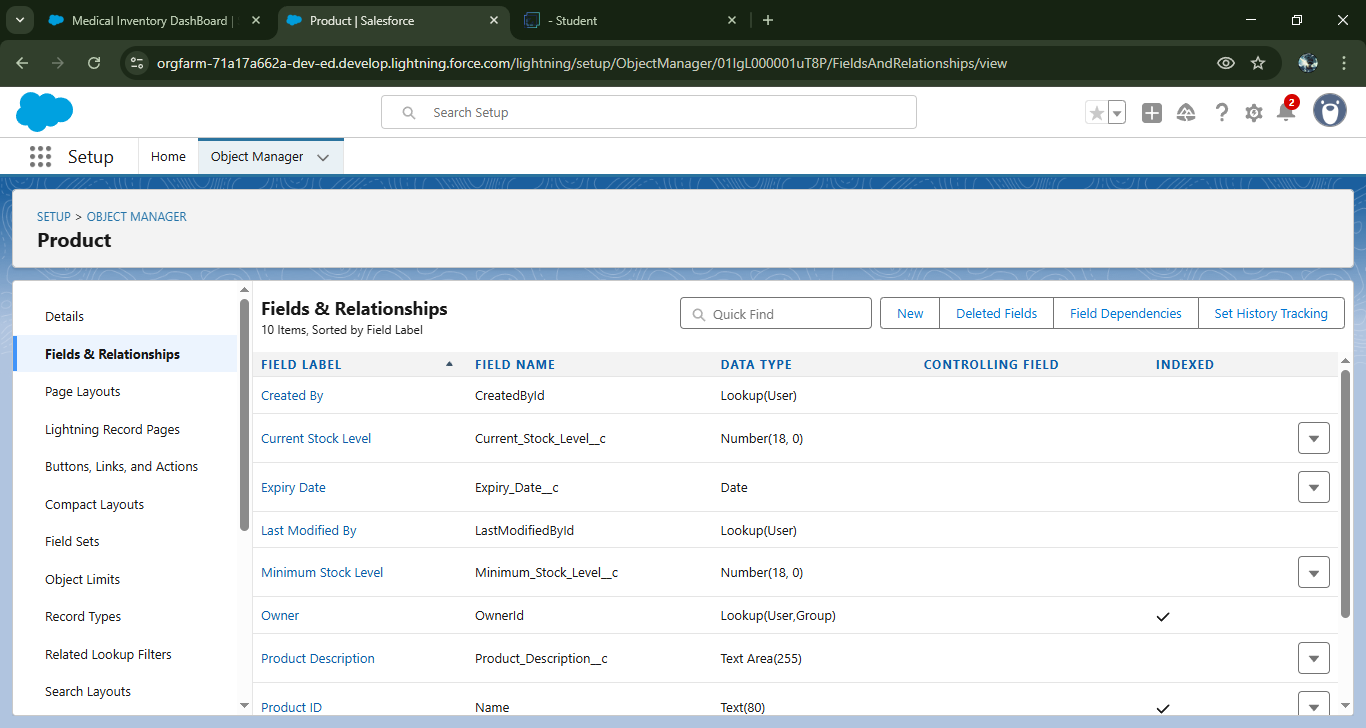
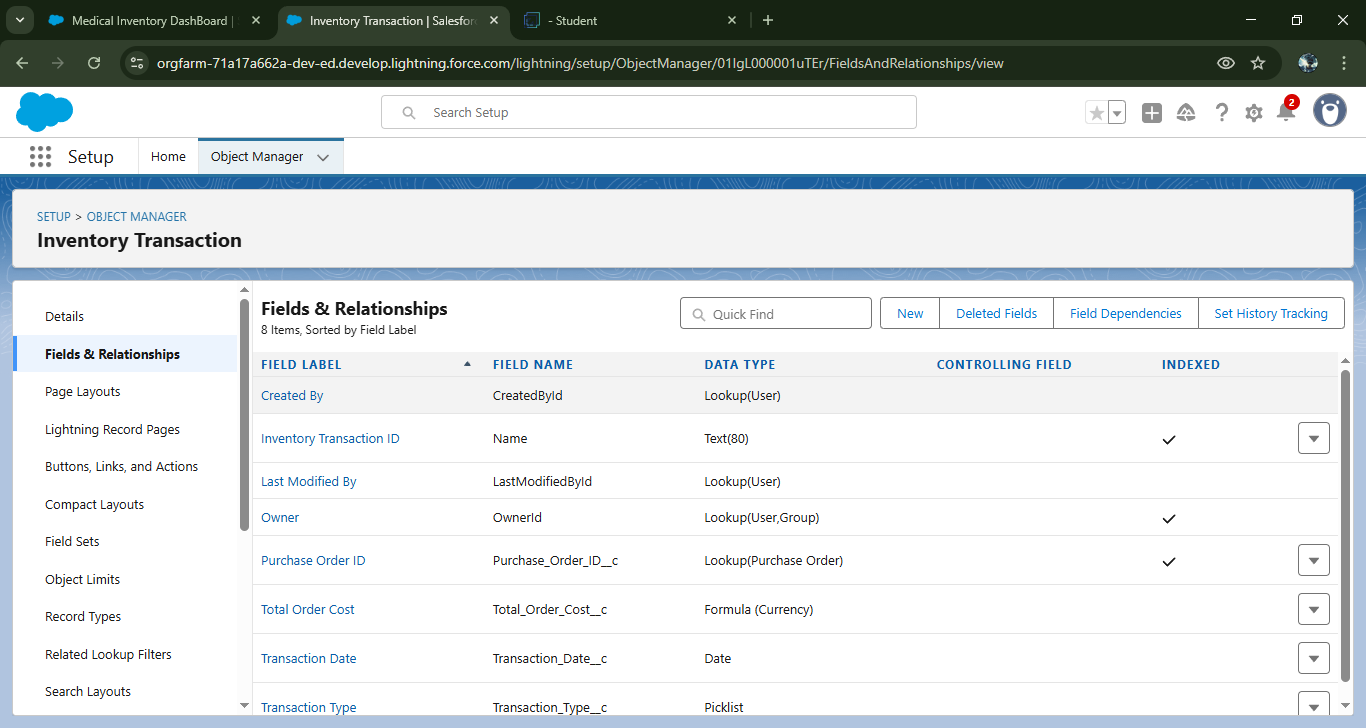
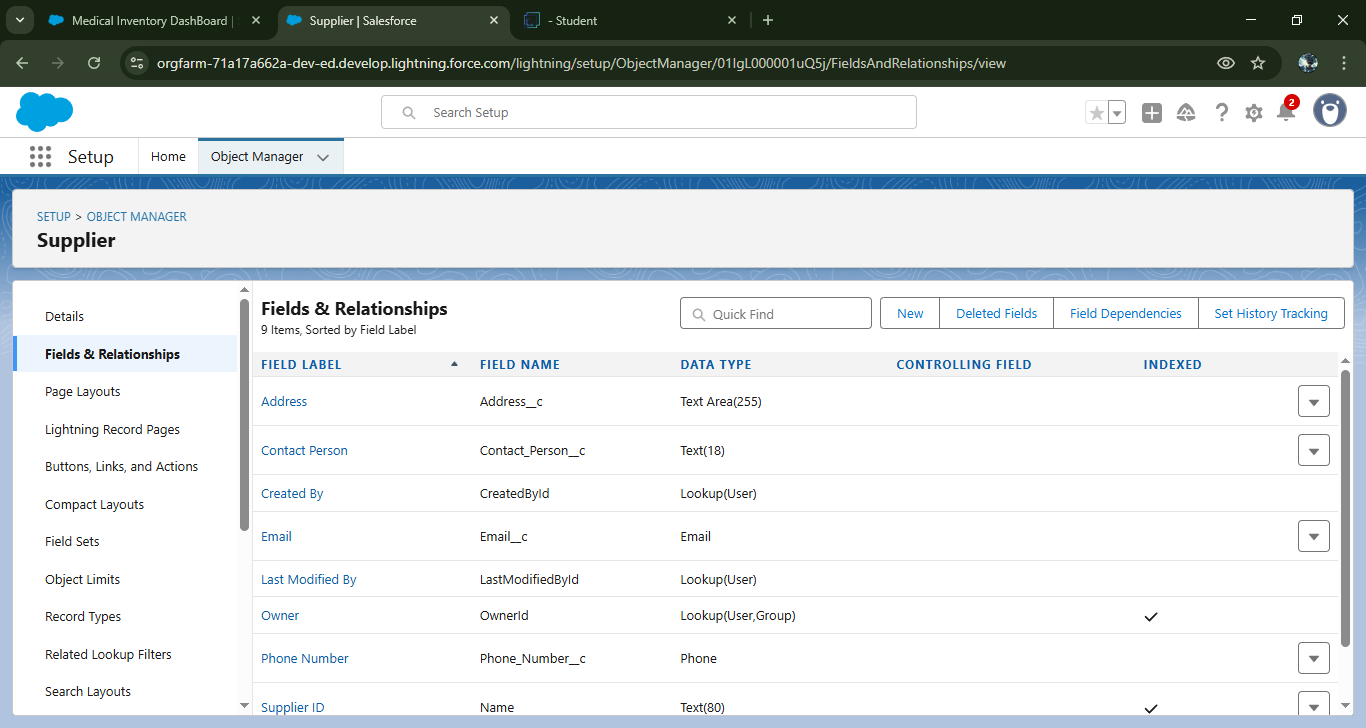


* Created objects: Product, Purchase order , Order Item , Inventory Transaction , Supplier.

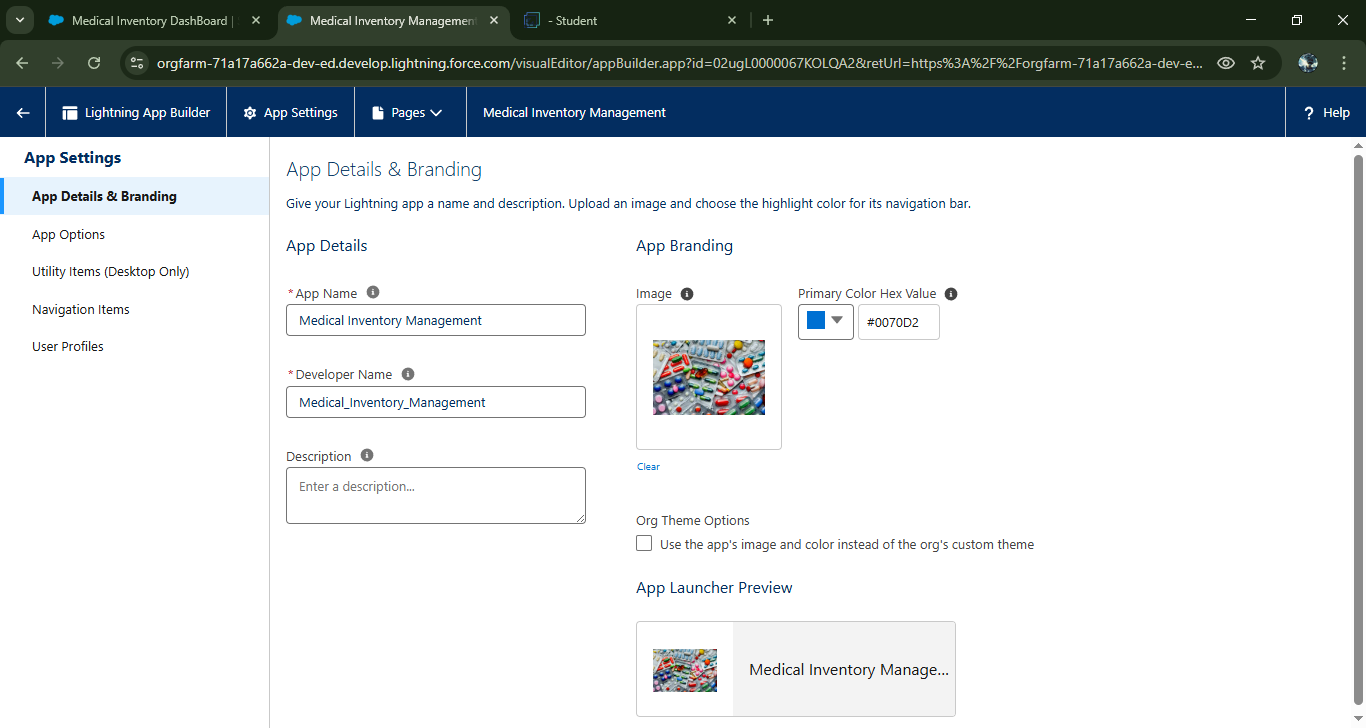


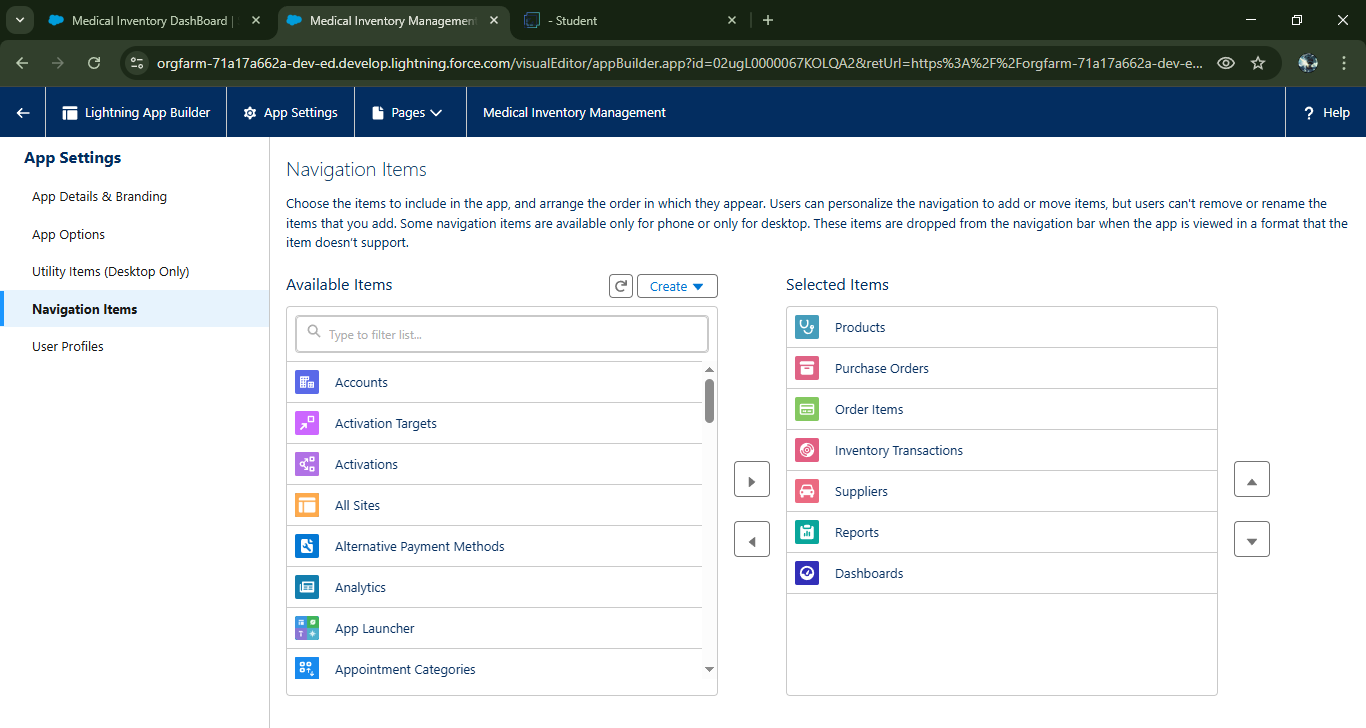
* Configured fields and relationships

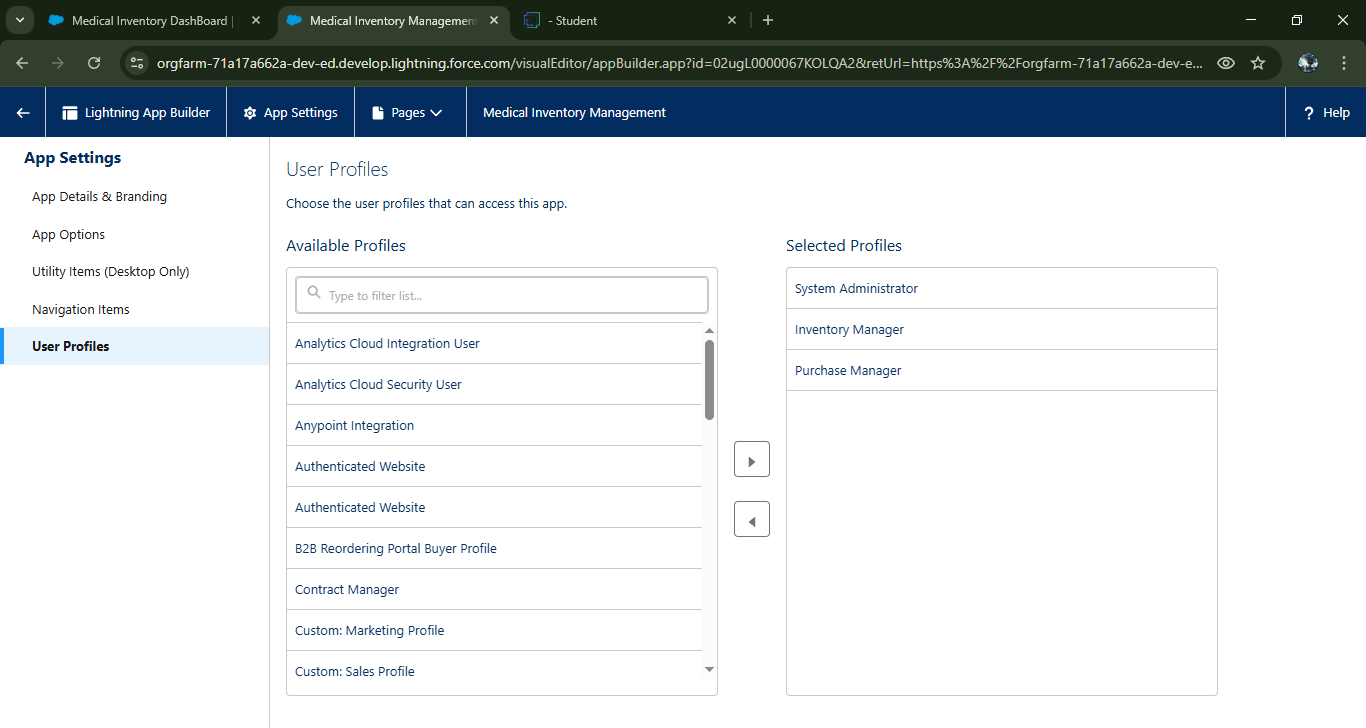


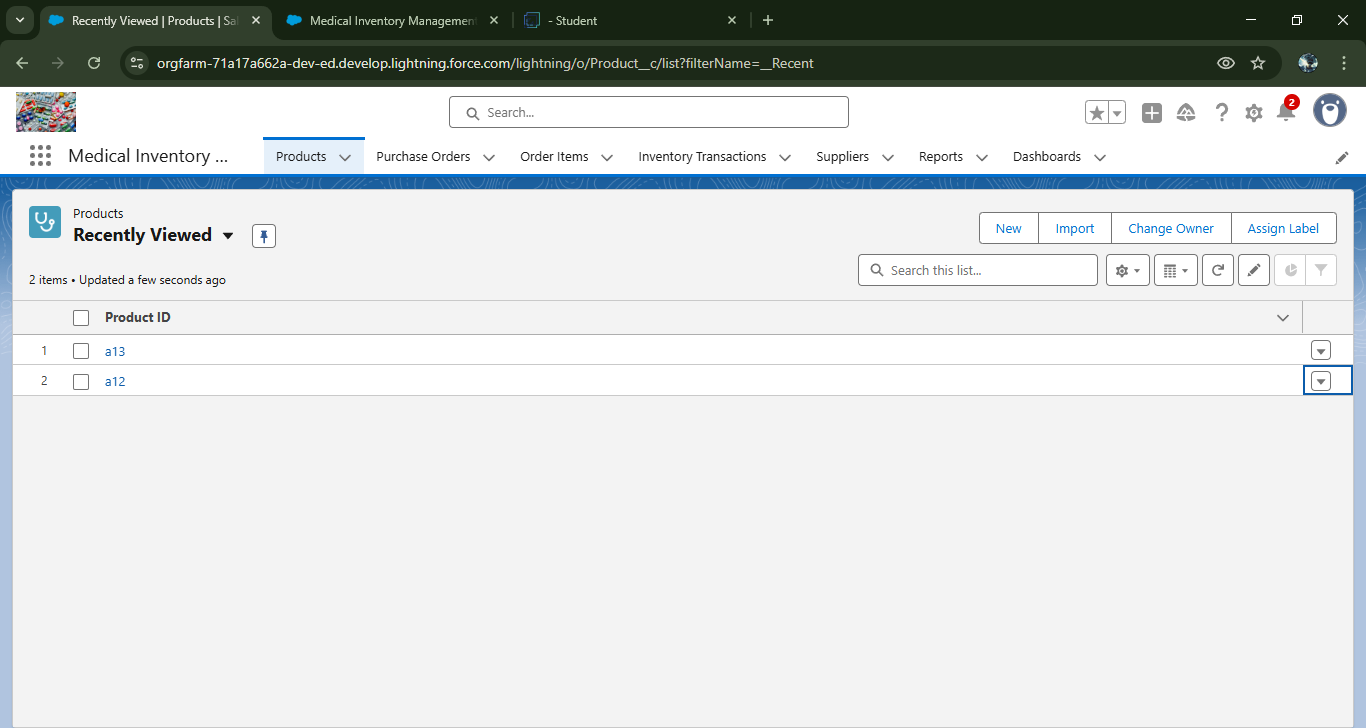


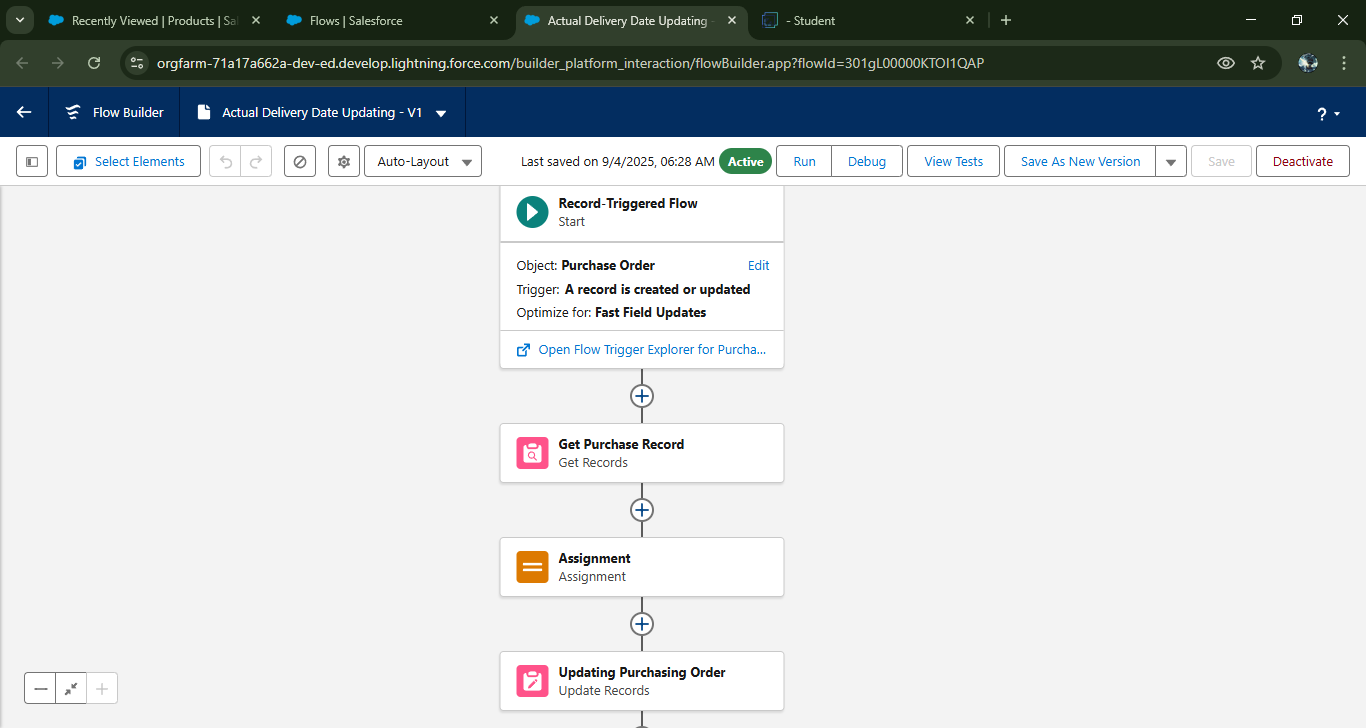
* + - Developed Lightning App with relevant tabs

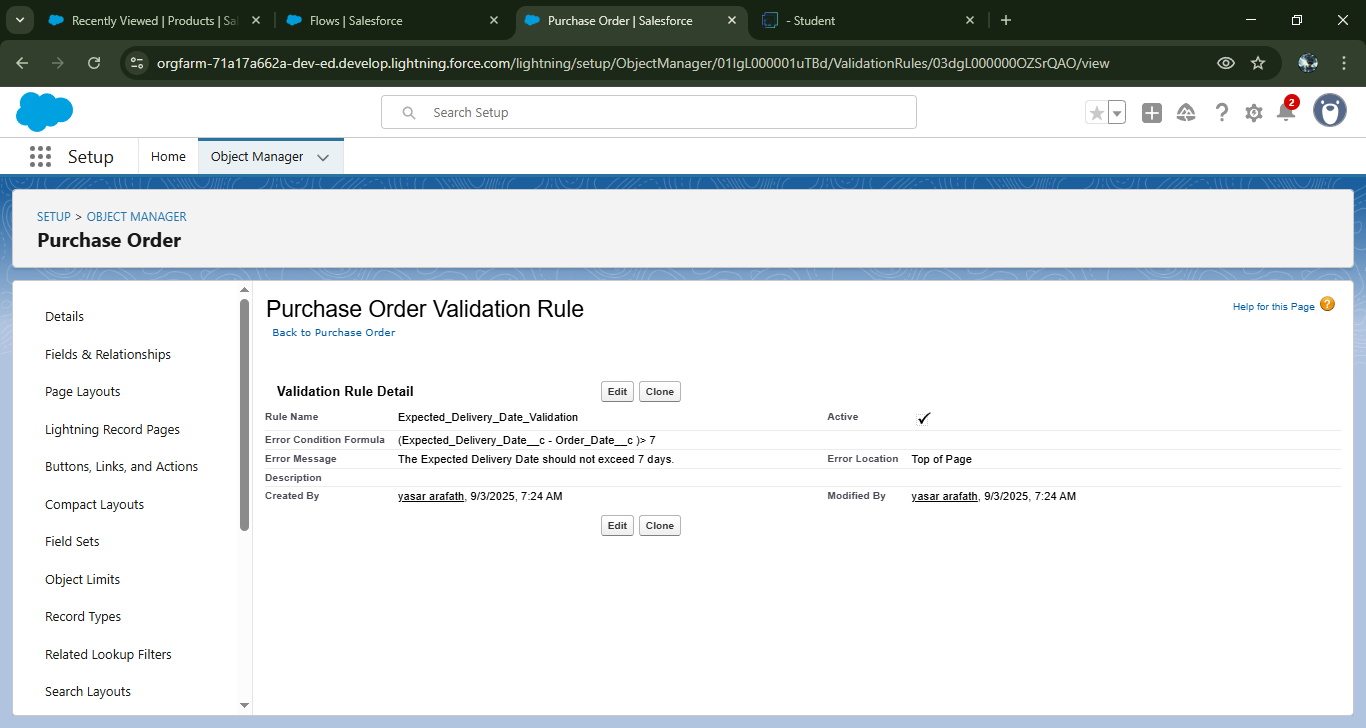


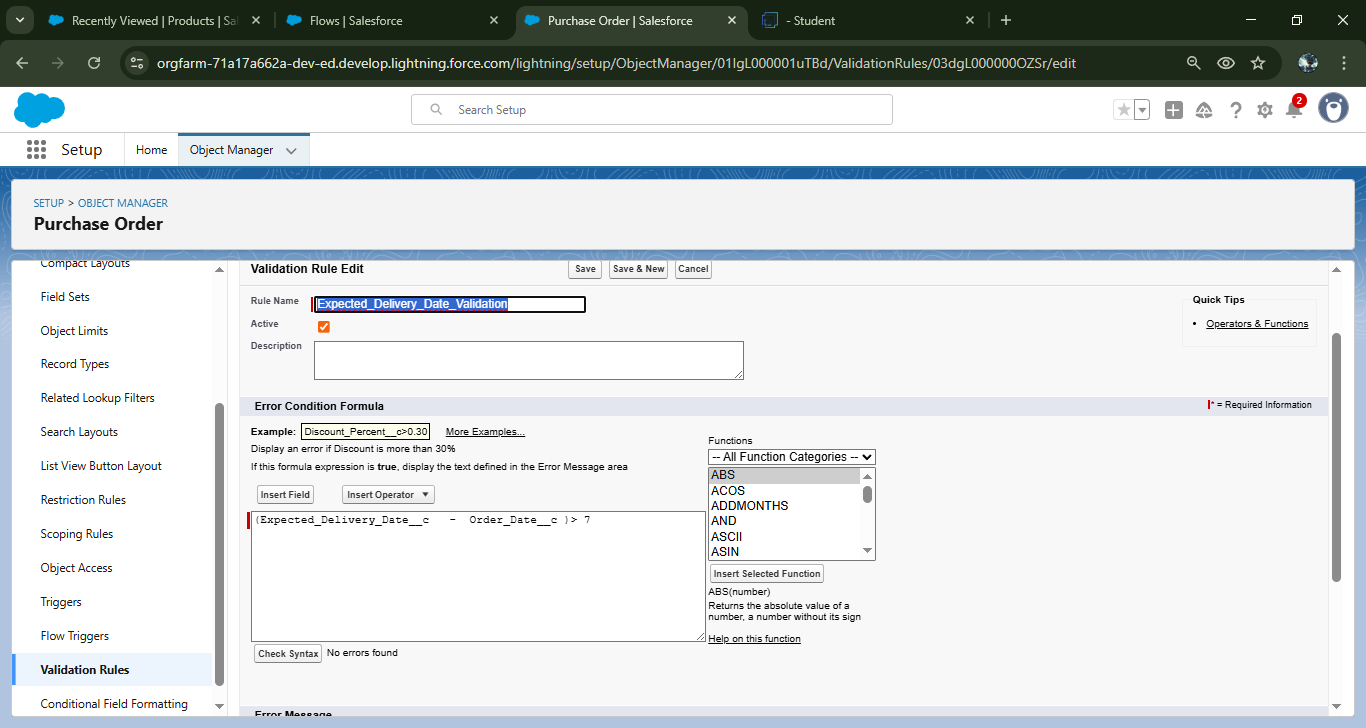




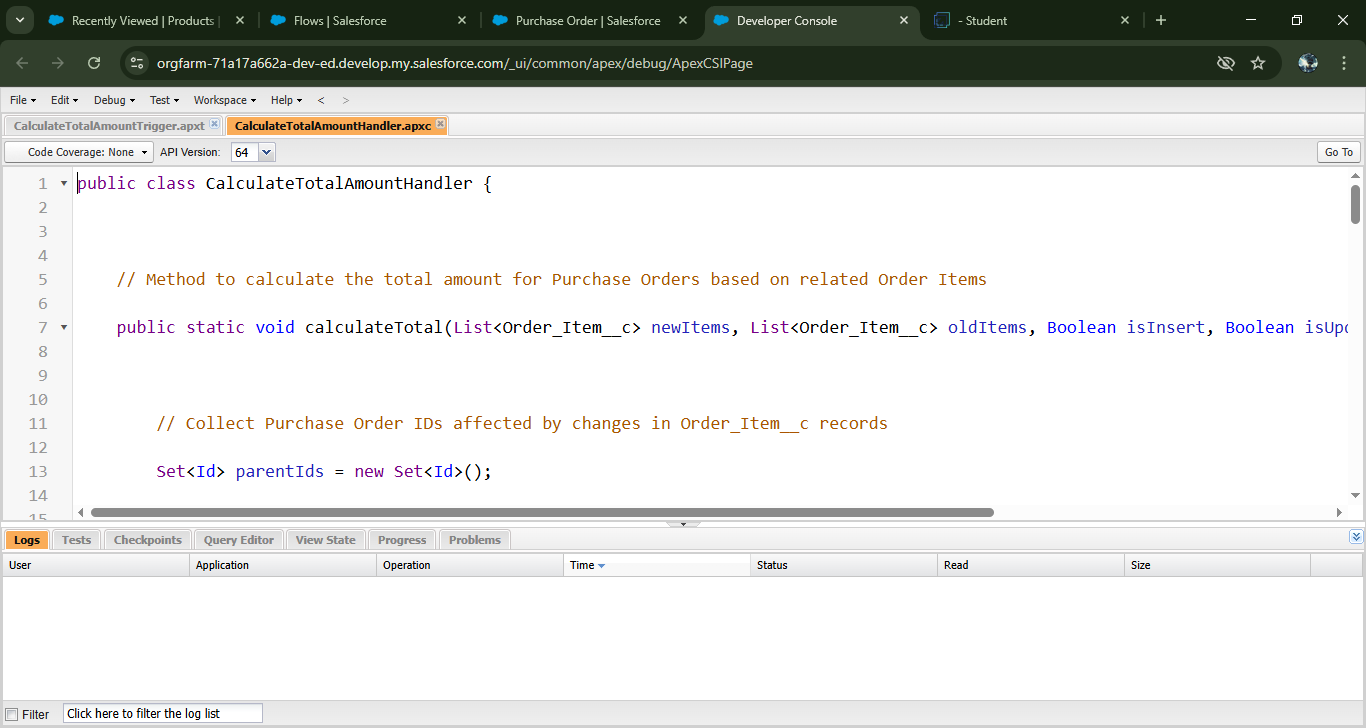


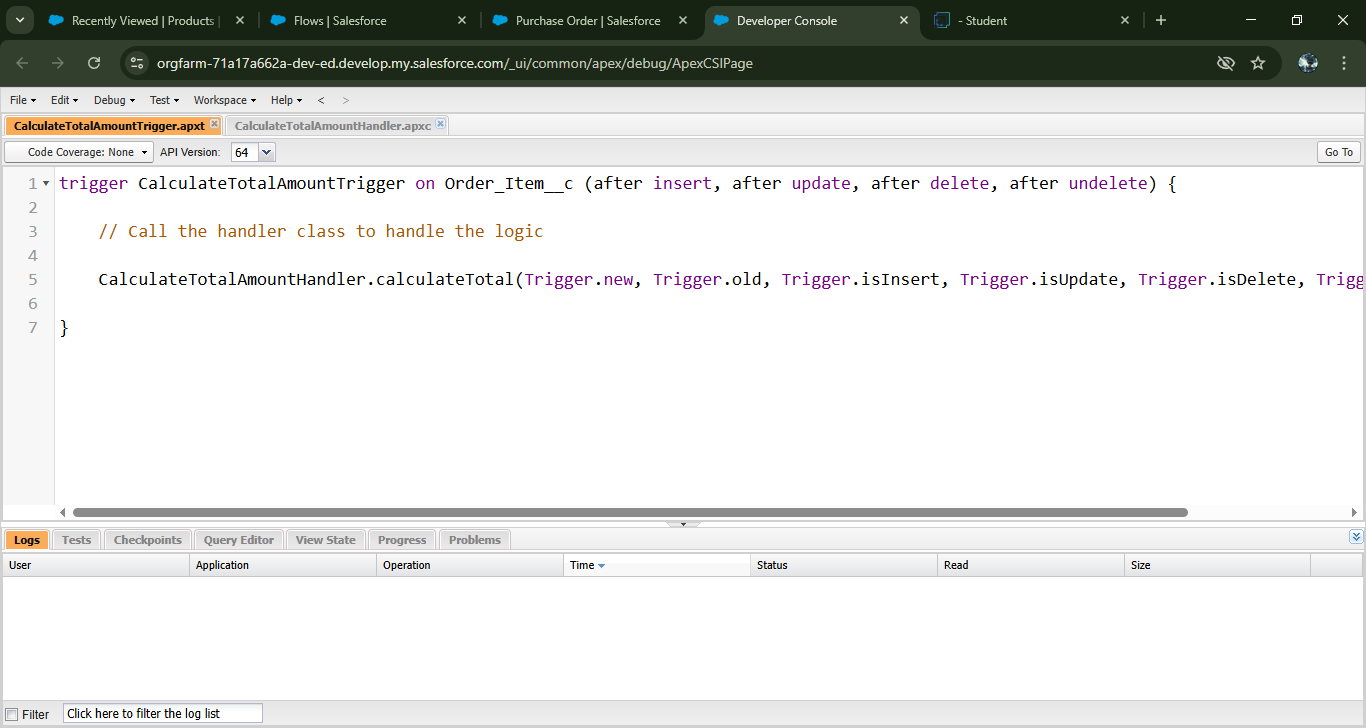
* + - Implemented Flow to update the Actual Delivery Date
    - To create a validation rule to a Purchase Order Object





### Added a apex class and Trigger to Calculate total amount on Order Item

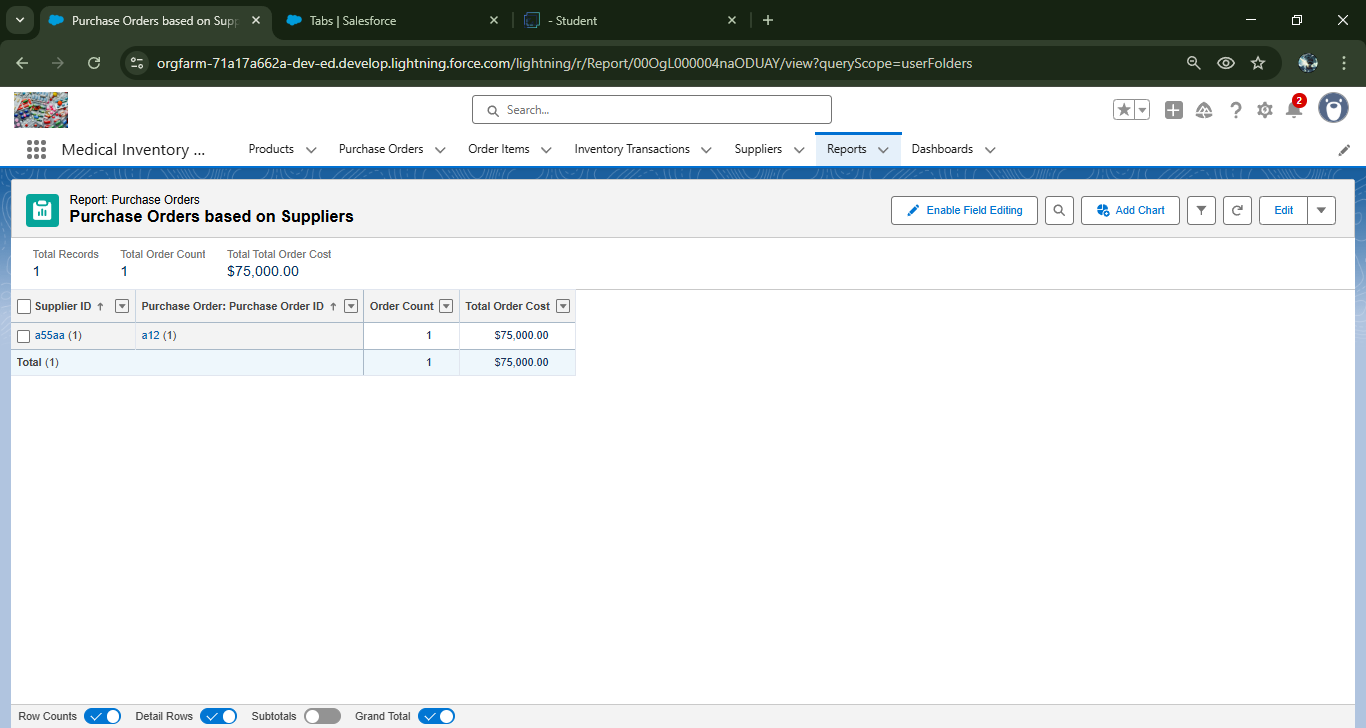


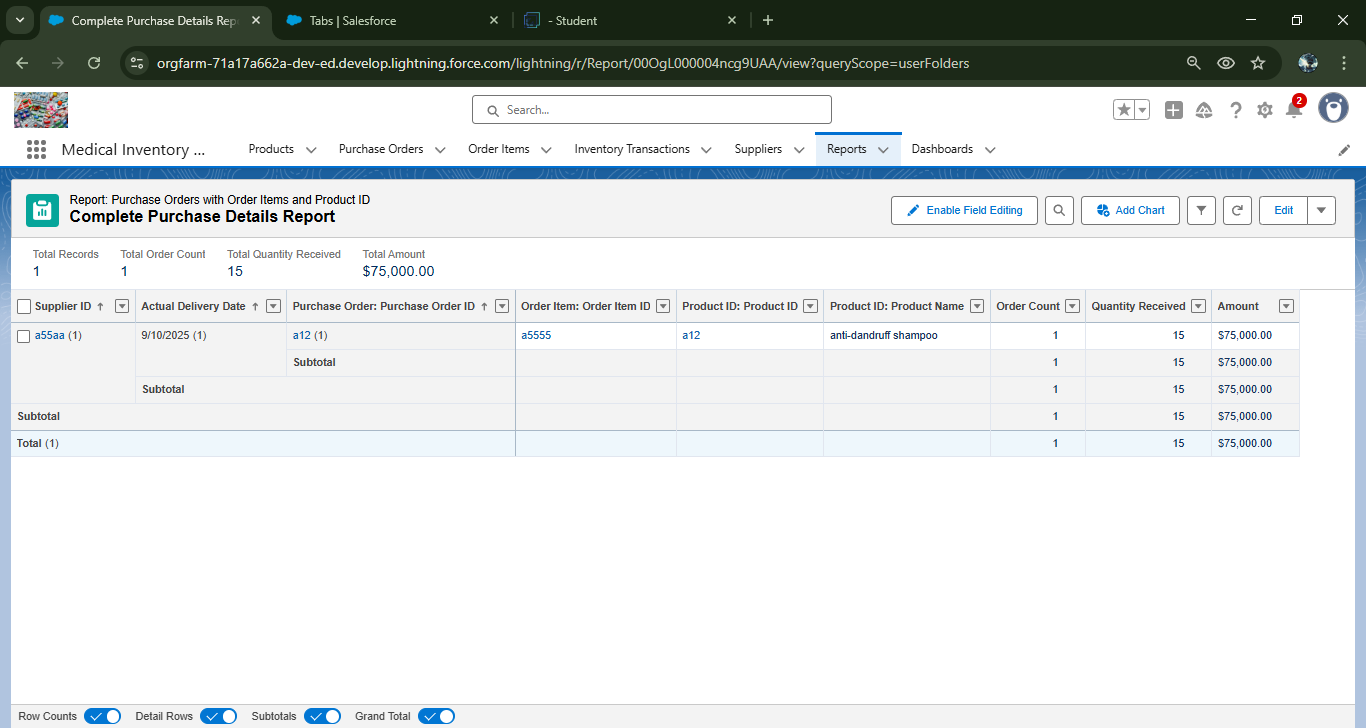


# **RESULTS**

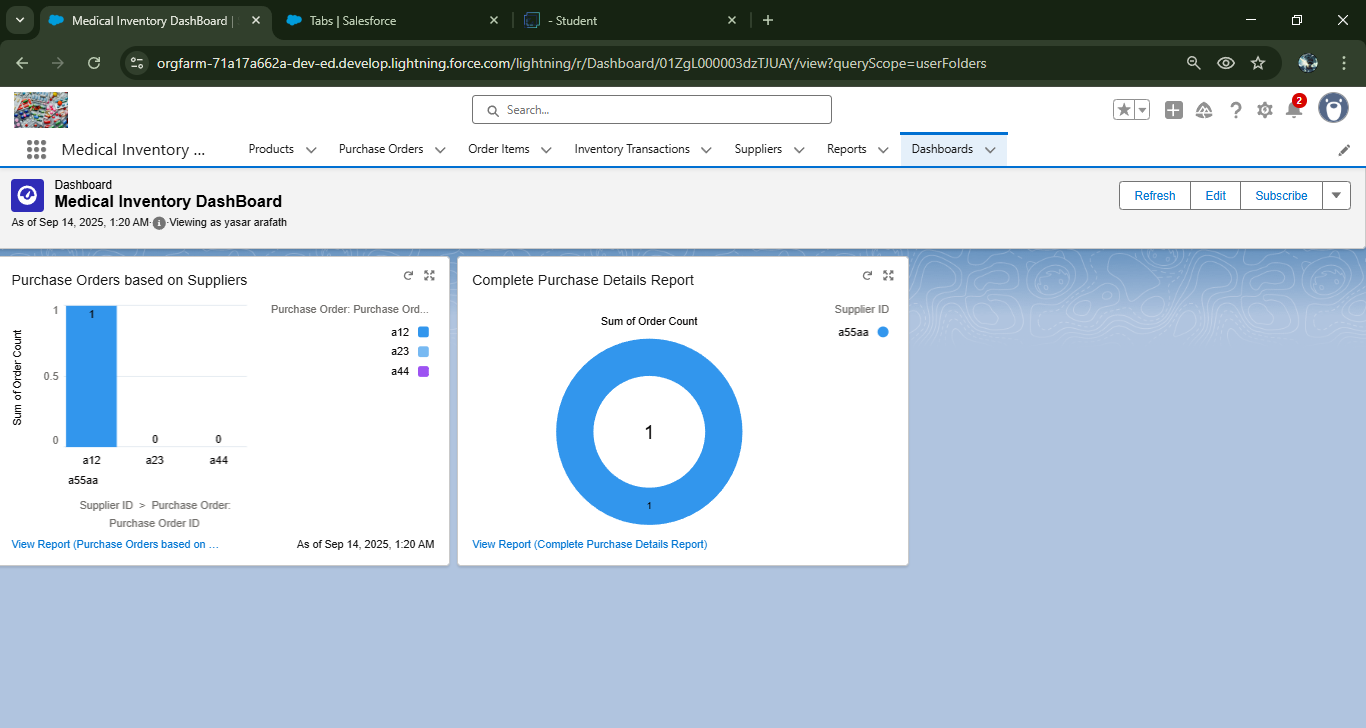
## **Output Screenshots**

* **REPORTS**

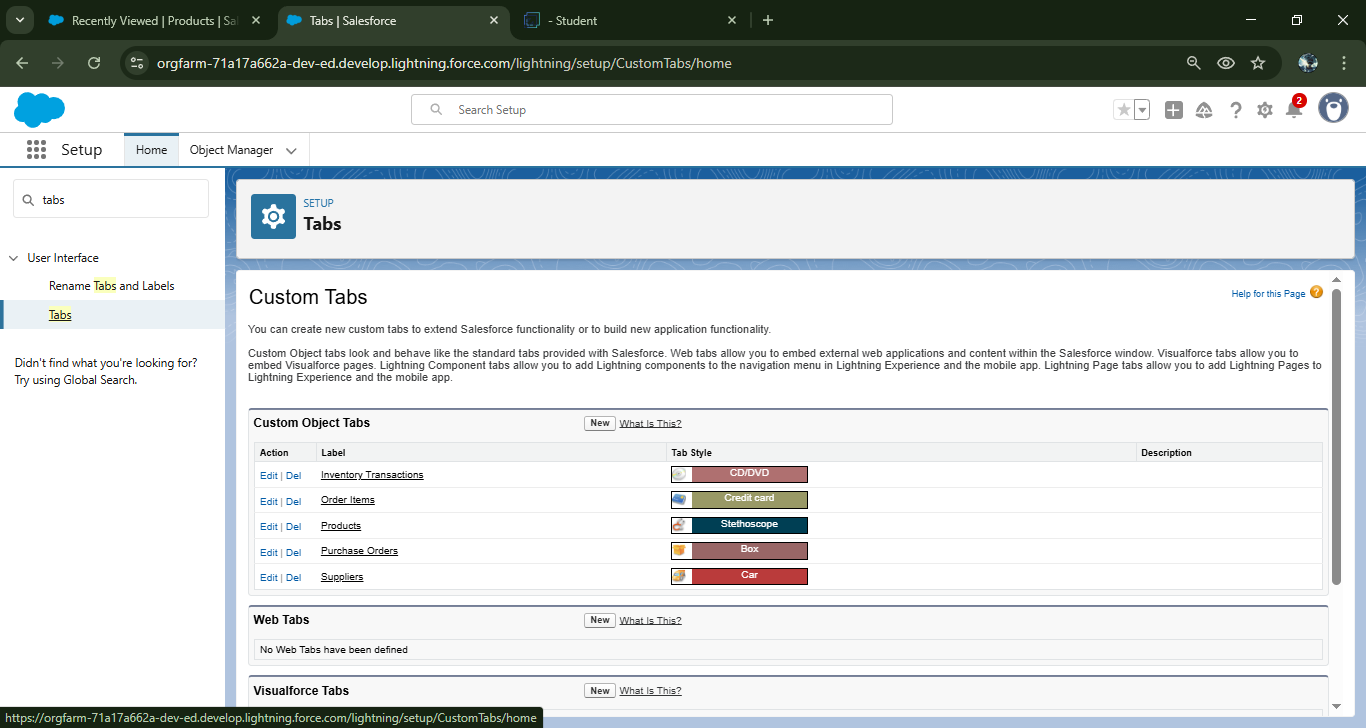
****

****

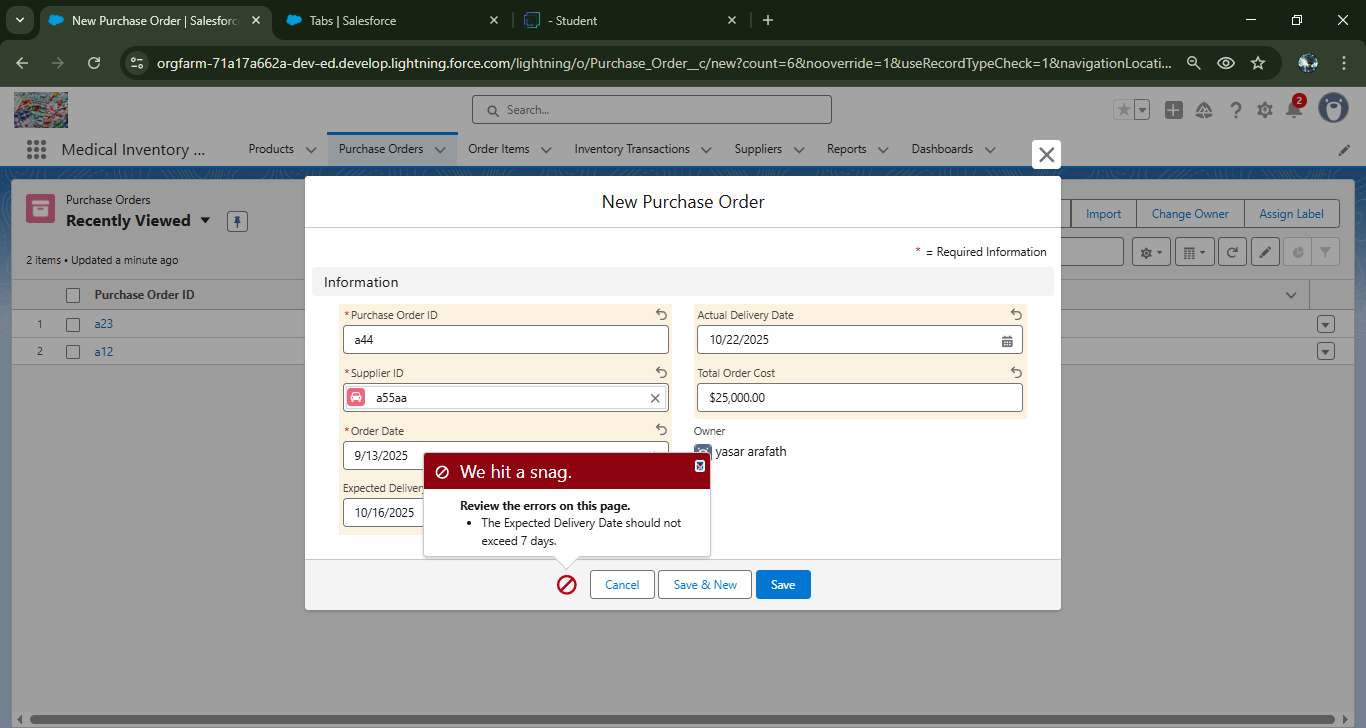
* **DASHBOARD**



* Tabs for product, purchase order, order item, inventory transaction, supplier objects



* Trigger error message



# **ADVANTAGES & DISADVANTAGES**

ADVANTAGE

1. Reduces Errors – Avoids mistakes in manual stock tracking.

2. Saves Time – Automates ordering and reporting.

3. Prevents Shortages – Sends alerts when stock is low.

4. Tracks Expiry – Keeps medicines safe and up to date.

5. Better Control – Easy monitoring of suppliers and transactions.

DISADVANTAGE

1. High Cost – Initial setup and maintenance can be expensive.

2. Training Needed – Staff must learn how to use the system.

3. Technical Issues – System errors or downtime may cause delays.

4. Data Security Risks – Sensitive information must be protected.

5. Dependence on Internet/Software – Cannot work properly if system or network fails.

# **CONCLUSION**

The Medical Inventory Management System makes it easier for healthcare organizations to track and manage supplies. It improves accuracy, saves time, and ensures that essential medicines and equipment are always available when needed.

## **APPENDIX**

* + **Source Code:** Provided in Apex Class and Trigger

### **CalculateTotalAmountHandler.apxc:**

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\_\_c

];

// 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;

}

}

}

}

### **CalculateTotalAmountTrigger.apxt:**

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);

}