

PROJECT TITLE: Medical Inventory Management System

College Name: United College of Arts and Science

College Code: brubl

TEAM ID: NM2025TMID27831

TEAM MEMBERS:

- **Team Leader Name:** Mithuna A mithunaanand4@gmail.com
- **Team Member 1:** Priyanka R priyankaramesh2804@gmail.com
- **Team Member 2:** Kaleeshwaran N kaleeshjack05@gamil.com
- **Team Member 3:** Deepan kumar S deebankumar332006@gmail.com

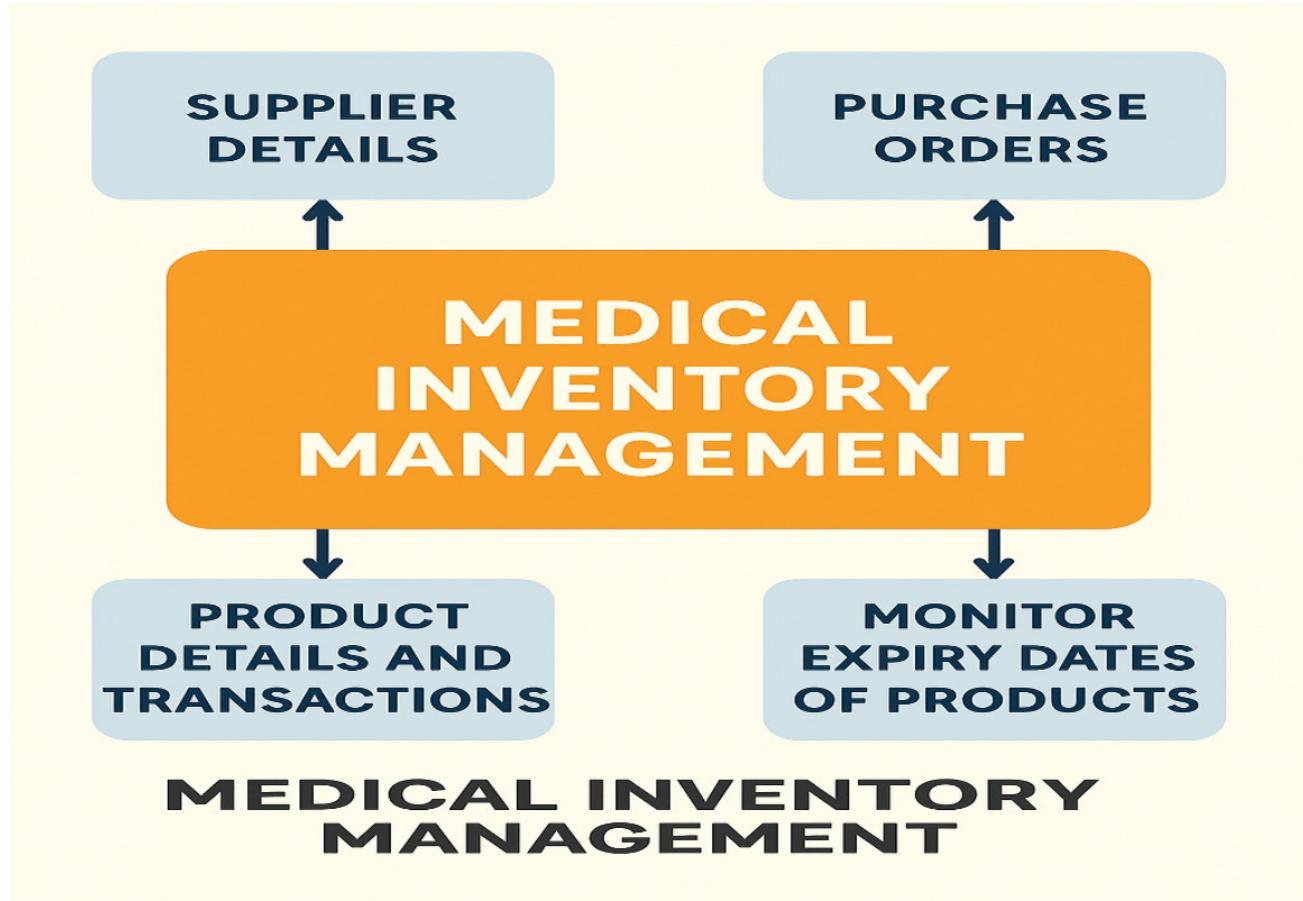
DEMO VIDEO LINK:

 NM_Demo video(Medical inventory management).mp4

INTRODUCTION

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.



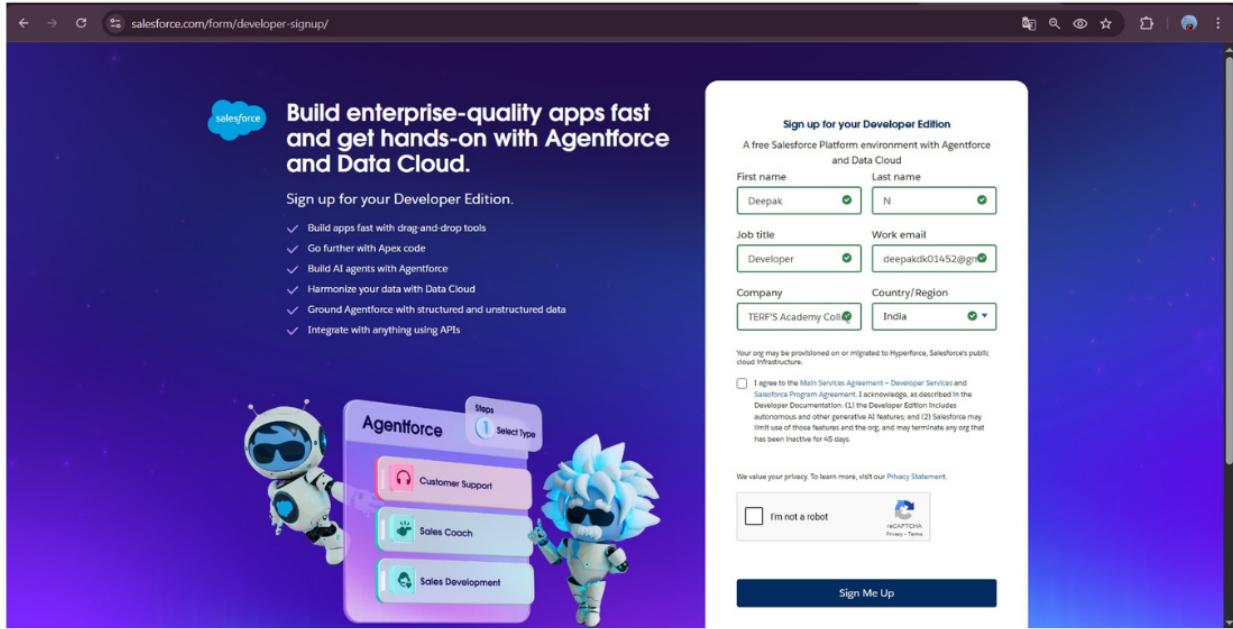
Purpose

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

Creation of Developer Account

- A Salesforce Developer account was created using the signup link:
<https://www.salesforce.com/form/developer-signup>



Creating a Product Object

Creating a tab for Product Object and Remaining Tabs

Custom Tabs

You can create new custom tabs to extend Salesforce functionality or to build new application functionality.

Custom Object Tabs

Action	Label	Tab Style	Description
Edit Del	Inventory Transactions	Hands	
Edit Del	Order Items	Box	
Edit Del	Products	Stethoscope	
Edit Del	Purchase Orders	Shopping Cart	
Edit Del	Suppliers	Truck	

Web Tabs

No Web Tabs have been defined

Visualforce Tabs

No Visualforce Tabs have been defined

Create a Lightning App for Medical Inventory Management

Lightning Experience App Manager

27 items • Sorted by App Name • Filtered by All appmenuitems - TabSet Type, App Type

App Name ↑	Developer Name	Description	Last Modified...	Ap...	Vi...
11 Digital Experiences	SalesforceCMS	Manage content and media for all of your sites.	8/31/2025, 4:27 PM	Lightning	✓
12 Lightning Usage App	LightningInstrumentation	View Adoption and Usage Metrics for Lightning Experi...	8/31/2025, 4:27 PM	Lightning	✓
13 Marketing CRM Classic	Marketing	Track sales and marketing efforts with CRM objects.	8/31/2025, 4:27 PM	Classic	✓
14 Medical Inventory Management	Medical_Inventory_Manageme...		9/5/2025, 5:13 AM	Lightning	✓
15 My Service Journey	MSJApp	Discover new customer service capabilities.	8/31/2025, 4:27 PM	Lightning	✓
16 Platform	Platform	The fundamental Lightning Platform	8/31/2025, 4:27 PM	Classic	
17 Queue Management	QueueManagement	Create and manage queues for your business.	8/31/2025, 4:27 PM	Lightning	✓
18 Sales	Sales	The world's most popular sales force automation (SFA...)	8/31/2025, 4:27 PM	Classic	
19 Sales	LightningSales	Manage your sales process with accounts, leads, oppo...	8/31/2025, 4:27 PM	Lightning	✓
20 Sales Cloud Mobile	SalesCloudMobile	New seller focused mobile first experience	8/31/2025, 4:27 PM	Lightning	✓
21 Sales Console	LightningSalesConsole	(Lightning Experience) Lets sales reps work with multi...	8/31/2025, 4:27 PM	Lightning	✓
22 Salesforce Chatter	Chatter	The Salesforce Chatter social network, including profil...	8/31/2025, 4:27 PM	Classic	✓

Creating a Text Field in Product Object

Fields & Relationships					
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Page Layouts	Created By	CreatedById	Lookup(User)		
Lightning Record Pages	Current Stock Level	Current_Stock_Level__c	Number(18, 0)		
Buttons, Links, and Actions	Expiry Date	Expiry_Date__c	Date		
Compact Layouts	Last Modified By	LastModifiedById	Lookup(User)		
Field Sets	Minimum Stock Level	Minimum_Stock_Level__c	Number(18, 0)		
Object Limits	Owner	OwnerId	Lookup(User,Group)	✓	
Record Types	Product Description	Product_Description__c	Text Area(255)		
Related Lookup Filters	Product ID	Name	Text(80)	✓	
Search Layouts	Product Name	Product_Name__c	Text(255)		
List View Button Layout	Unit Price	Unit_Price__c	Currency(16, 2)		
Restriction Rules					
Scoping Rules					

Creating Lookup Relationship in Purchase Order Object

Fields & Relationships					
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Page Layouts	Actual Delivery Date	Actual_Delivery_Date__c	Date		
Lightning Record Pages	Created By	CreatedById	Lookup(User)		
Buttons, Links, and Actions	Expected Delivery Date	Expected_Delivery_Date__c	Date		
Compact Layouts	Last Modified By	LastModifiedById	Lookup(User)		
Field Sets	Order Count	Order_Count__c	Roll-Up Summary (COUNT Order Item)		
Object Limits	Order Date	Order_Date__c	Date		
Record Types	Owner	OwnerId	Lookup(User,Group)	✓	
Related Lookup Filters	Purchase Order ID	Name	Text(80)	✓	
Search Layouts	Supplier ID	Supplier_ID__c	Lookup(Supplier)	✓	
List View Button Layout	Total Order Cost	Total_Order_Cost__c	Currency(16, 2)		
Restriction Rules					
Scoping Rules					

Creating a Unit Price Formula Field in Order Item object

Fields & Relationships
10 Items, Sorted by Field Label

	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Amount	Amount_c	Formula (Currency)		✓
Created By	CreatedBy	Lookup(User)		✓
Last Modified By	LastModifiedBy	Lookup(User)		✓
Order Item ID	Name	Text(80)		✓
Product ID	Product_ID_c	Lookup(Product)		✓
Purchase Order ID	Purchase_Order_ID_c	Master-Detail(Purchase Order)		✓
Quantity Ordered	Quantity_Ordered_c	Number(18, 0)		✓
Quantity Received	Quantity_Received_c	Number(18, 0)		✓
Total Order Cost	Total_Order_Cost_c	Formula (Currency)		✓
Unit Price	Unit_Price_c	Formula (Currency)		✓

Creating a Picklist Field in Inventory Transaction Object

Fields & Relationships
5 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedBy	Lookup(User)		✓
Inventory Transaction ID	Name	Text(80)		✓
Last Modified By	LastModifiedBy	Lookup(User)		✓
Owner	OwnerId	Lookup(User,Group)		✓
Transaction Type	Transaction_Type_c	Picklist		✓

To edit a Page Layout in Product Object

The screenshot shows the Salesforce Setup interface for the Product object. The left sidebar lists various tabs: Details, Fields & Relationships, Page Layouts (which is selected), Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, Search Layouts, List View Button Layout, Restriction Rules, and Scoping Rules. The main area is titled 'Product' and contains a 'Layout Properties' toolbar with Save, Quick Save, Preview As..., Cancel, Undo, Redo, and Layout Properties buttons. Below this is a 'Fields' section with a 'Quick Find' field for 'Field Name'. The layout includes sections for 'Actions in the Salesforce Classic Publisher section...' and 'Product Detail'. The 'Product Detail' section has standard buttons for Edit, Delete, Clone, Change Owner, Change Record Type, Printable View, Sharing, Sharing Hierarchy, and Edit Labels. It also includes 'Information', 'System Information', and 'Custom Links' sections with sample data.

To edit a Page Layout in Purchase Order Object

The screenshot shows the Salesforce Setup interface for the Purchase Order object. The left sidebar lists various tabs: Details, Fields & Relationships, Page Layouts (selected), Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, Search Layouts, List View Button Layout, Restriction Rules, and Scoping Rules. The main area is titled 'Purchase Order' and contains a 'Layout Properties' toolbar with Save, Quick Save, Preview As..., Cancel, Undo, Redo, and Layout Properties buttons. Below this is a 'Fields' section with a 'Quick Find' field for 'Field Name'. The layout includes sections for 'Actions in the Salesforce Classic Publisher section...' and 'Purchase Order Detail'. The 'Purchase Order Detail' section has standard buttons for Edit, Delete, Clone, Change Owner, Change Record Type, Printable View, Sharing, Sharing Hierarchy, and Edit Labels. It also includes 'Information', 'System Information', and 'Custom Links' sections with sample data.

To edit a Page Layout in Order Item Object

SETUP > OBJECT MANAGER
Order Item

Save Quick Save Preview As... Cancel Undo Redo Layout Properties

Fields

Field Name	Type
Order Item ID	Text
Amount	Text
Product ID	Text
Unit Price	Text
Purchase Order ID	Text
Total Order Cost	Text

Information (Header visible on edit only)

* Order Item ID Sample Text
Amount \$123.45
Product ID Sample Text
Unit Price \$123.45

* Purchase Order ID Sample Text
Quantity Ordered 24.404
Quantity Received 68.497

System Information (Header visible on edit only)

Created By Sample Text
Last Modified By Sample Text

Custom Links (Header visible on edit only)

Mobile Cards (Salesforce mobile only) Drag expanded lookups and mobile-enabled Visualforce pages here to display them as mobile cards.

Related Lists

To edit a Page Layout in Inventory Transaction Object

SETUP > OBJECT MANAGER
Inventory Transaction

Save Quick Save Preview As... Cancel Undo Redo Layout Properties

Fields

Field Name	Type
Inventory Transaction ID	Text
Transaction Type	Text
Owner	Text
Created By	Text

Information (Header visible on edit only)

* Inventory Transaction ID Sample Text
Transaction Type Sample Text
Owner Sample Text

System Information (Header visible on edit only)

Created By Sample Text
Last Modified By Sample Text

Custom Links (Header visible on edit only)

Mobile Cards (Salesforce mobile only) Drag expanded lookups and mobile-enabled Visualforce pages here to display them as mobile cards.

Related Lists

To edit a Page Layout in Supplier Object

To create a Compact Layout to a Product Object

To create a Compact Layout to a Purchase Order Object

The screenshot shows the Salesforce Setup interface under the Object Manager section for the Purchase Order object. A modal window titled "Purchase Order Compact Layout" is open, allowing the user to edit the compact layout settings. The "Label" field is set to "Purchase Order Compact L" and the "Name" field is set to "Purchase_Order_Compact". The "Select Compact Layout Fields" section lists several fields available for selection: Actual Delivery Date, Created By, Expected Delivery Date, Last Modified By, Order Count, and Owner. The "Selected Fields" section contains "Purchase Order ID", "Order Date", "Total Order Cost", and "Supplier ID". The "Selected Fields" section also includes a vertical list of sorting options: Top, Up, Down, and Bottom, with "Top" currently selected.

To create an Expected Delivery Date Validation rule to a Employee Object

The screenshot shows the Salesforce Setup interface under the Object Manager section for the Purchase Order object. A modal window titled "Purchase Order Validation Rule" is open, displaying the validation rule details. The "Validation Rule Detail" section shows the following configuration:

- Rule Name:** Expected_Delivery_Date_Validation
- Error Condition Formula:** (Expected_Delivery_Date__c - Order_Date__c) > 7
- Error Message:** The Expected Delivery Date should not exceed 7 days.
- Description:** (empty)
- Created By:** Deepak.N, 9/5/2025, 11:25 PM
- Active:** checked
- Error Location:** Top of Page
- Modified By:** Deepak.N, 9/5/2025, 11:25 PM

To create an Inventory Manager Profile

The screenshot shows the Salesforce Setup interface under the Profiles section. The profile name is 'Inventory Manager'. The 'Profile Detail' section includes fields for Name (Inventory Manager), User License (Salesforce), and a checked 'Custom Profile' checkbox. The 'Page Layouts' section lists various object layouts with their global assignments:

Object	Global Assignment	Location Group Assignment
Global	Global Layout [View Assignment]	Location Group Assignment Layout [View Assignment]
Email Application	Not Assigned [View Assignment]	Macro Layout [View Assignment]
Home Page Layout	Home Page Default [View Assignment]	Object Milestone Layout [View Assignment]
Account	Account Layout [View Assignment]	Operating Hours Layout [View Assignment]
Alternative Payment Method	Alternative Payment Method Layout [View Assignment]	Opportunity Layout [View Assignment]
Appointment Invitation	Appointment Invitation Layout	Opportunity Product Layout

To create an Purchase Manager Profile

The screenshot shows the Salesforce Setup interface under the Profiles section. The profile name is 'Purchase Manager'. The 'Profile Detail' section includes fields for Name (Purchase Manager), User License (Salesforce), and a checked 'Custom Profile' checkbox. The 'Page Layouts' section lists various object layouts with their global assignments:

Object	Global Assignment	Location Group Assignment
Global	Global Layout [View Assignment]	Location Group Assignment Layout [View Assignment]
Email Application	Not Assigned [View Assignment]	Macro Layout [View Assignment]
Home Page Layout	Home Page Default [View Assignment]	Object Milestone Layout [View Assignment]
Account	Account Layout [View Assignment]	Operating Hours Layout [View Assignment]
Alternative Payment Method	Alternative Payment Method Layout [View Assignment]	Opportunity Layout [View Assignment]
Appointment Invitation	Appointment Invitation Layout	Opportunity Product Layout

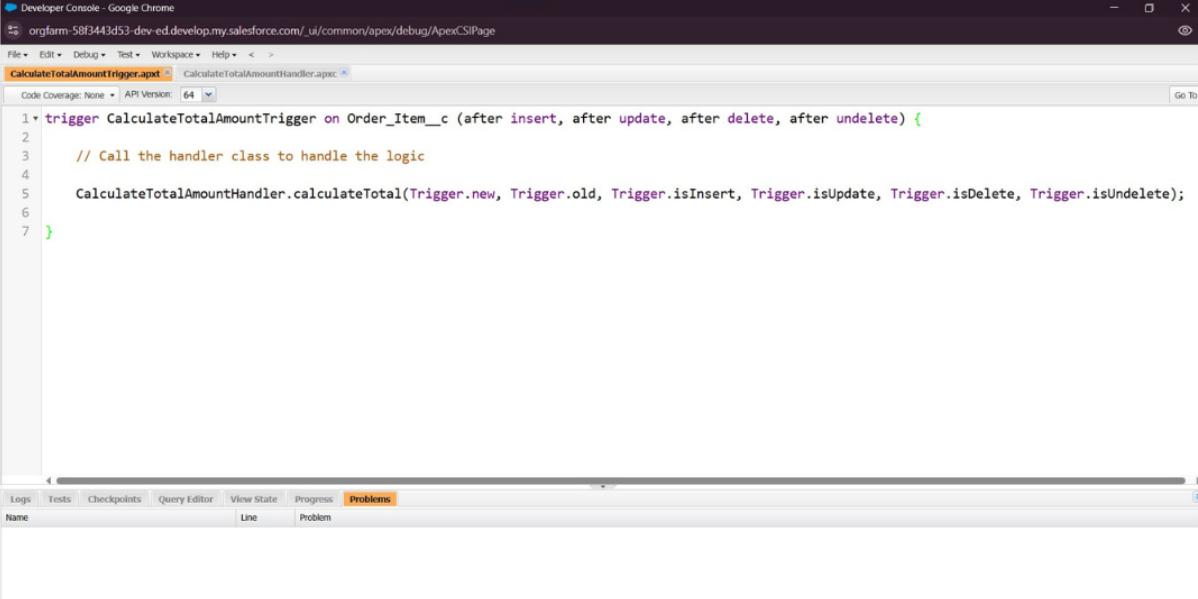
Create a Purchasing Manager Role.

The screenshot shows the Salesforce Setup Roles page. The 'Role Detail' section for 'Purchasing Manager' is displayed, showing details like Label (Purchasing Manager), Modified By (Deepak.N), Opportunity Access (Users can edit all opportunities associated with accounts that they own, regardless of who owns the opportunities), and Case Access (Users in this role can edit all cases associated with accounts that they own, regardless of who owns the cases). The 'Users in Purchasing Manager Role' section lists one user: John.P (johnpurchasem@test.com) with the status 'Active'. The URL in the browser is <https://orgfarm-58f3443d53-dev-ed.develop.lightning.force.com/lightning/setup/Roles/page?address=%2FO0EgK000003xcbD%3Fsetupid%3DRoles>.

Create Flow to update the Actual Delivery Date.

The screenshot shows the Salesforce Flow Builder interface with a flow titled 'Actual Delivery Date Updating - V1'. The flow starts with a 'Record-Triggered Flow' trigger for 'Purchase Order' objects, triggered by 'A record is created or updated'. It then branches into three parallel steps: 'Get Purchase Record', 'Assignment', and 'Updating Purchasing Order'. Finally, it ends. The URL in the browser is https://orgfarm-58f3443d53-dev-ed.develop.lightning.force.com/builder_platform.interaction/flowBuilder.app?flowId=301gK00000JDbMzQAL.

Create a Trigger to Calculate total amount on Order Item.

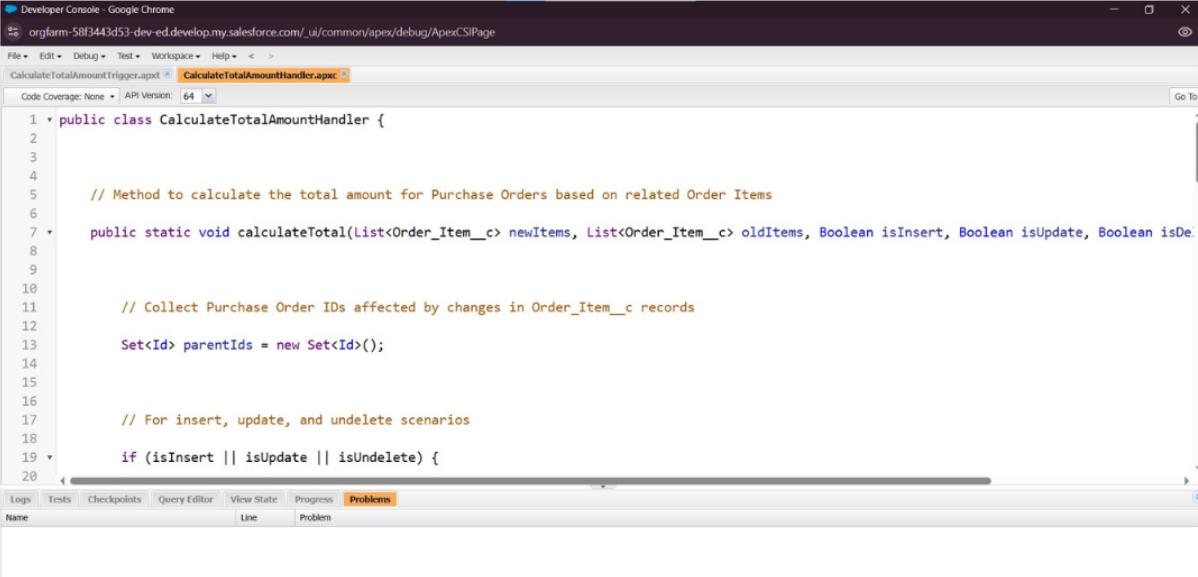


The screenshot shows the Salesforce Developer Console in Google Chrome. The URL is org://58f3443d53-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCIPage. The tab title is CalculateTotalAmountTrigger.apxc. The code editor contains the following Apex trigger:

```
1 trigger CalculateTotalAmountTrigger on Order_Item__c (after insert, after update, after delete, after undelete) {
2     // Call the handler class to handle the logic
3     CalculateTotalAmountHandler.calculateTotal(Trigger.new, Trigger.old, Trigger.isInsert, Trigger.isUpdate, Trigger.isDelete, Trigger.isUndelete);
4 }
5
6
7 }
```

The console interface includes tabs for Logs, Tests, Checkpoints, Query Editor, View State, Progress, and Problems. The Problems tab is selected, showing no errors or warnings.

Choose Apex Class: Name it as CalculateTotalAmountHandler



The screenshot shows the Salesforce Developer Console in Google Chrome. The URL is org://58f3443d53-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCIPage. The tab title is CalculateTotalAmountHandler.apxc. The code editor contains the following Apex class:

```
1 public class CalculateTotalAmountHandler {
2
3
4
5     // Method to calculate the total amount for Purchase Orders based on related Order Items
6     public static void calculateTotal(List<Order_Item__c> newItems, List<Order_Item__c> oldItems, Boolean isInsert, Boolean isUpdate, Boolean isDelete) {
7
8         // Collect Purchase Order IDs affected by changes in Order_Item__c records
9         Set<Id> parentIds = new Set<Id>();
10
11
12         // For insert, update, and undelete scenarios
13         if (isInsert || isUpdate || isDelete) {
14
15             if (isInsert || isUpdate) {
16
17                 for (Order_Item__c item : newItems) {
18                     if (item.ParentOrder__c != null) {
19                         parentIds.add(item.ParentOrder__c);
20
21
22                     }
23
24                 }
25
26             }
27
28             else {
29
30                 for (Order_Item__c item : oldItems) {
31                     if (item.ParentOrder__c != null) {
32                         parentIds.add(item.ParentOrder__c);
33
34                     }
35
36                 }
37
38             }
39
40             // Calculate the total amount for each parent order
41             for (Id parentId : parentIds) {
42                 Order__c parentOrder = [SELECT Id, Name, Total_Amount__c FROM Order__c WHERE Id = :parentId];
43
44                 if (parentOrder != null) {
45                     parentOrder.Total_Amount__c = calculateTotal(newList, newList, true, false, false);
46
47                     update parentOrder;
48
49                 }
50
51             }
52
53         }
54
55     }
56
57     private static Double calculateTotal(List<Order_Item__c> items, Boolean isInsert, Boolean isUpdate, Boolean isDelete) {
58
59         Double total = 0.0;
60
61         for (Order_Item__c item : items) {
62             if (item.Total_Amount__c == null) {
63                 item.Total_Amount__c = 0.0;
64
65             }
66
67             if (isInsert || isUpdate) {
68
69                 item.Total_Amount__c += item.Quantity * item.Unit_Price__c;
70
71             }
72
73             else if (isDelete) {
74
75                 item.Total_Amount__c -= item.Quantity * item.Unit_Price__c;
76
77             }
78
79         }
80
81         return total;
82
83     }
84
85 }
```

The console interface includes tabs for Logs, Tests, Checkpoints, Query Editor, View State, Progress, and Problems. The Problems tab is selected, showing no errors or warnings.

Create a Purchase Orders based on Suppliers(Summary) Report

The screenshot shows a Salesforce Lightning report titled "Purchase Orders based on Suppliers". The report summary at the top indicates 5 total records, 14 total order count, and a total total order cost of \$26,325.00. The main table lists purchase orders grouped by supplier. Supplier-001 has 4 entries with a total cost of \$2,075.00. Supplier-002 has 1 entry with a total cost of \$4,500.00. The table includes columns for Supplier ID, Purchase Order ID, Order Count, and Total Order Cost.

Supplier ID	Purchase Order ID	Order Count	Total Order Cost
Supplier-001 (4)	Purchase-0001 (1)	3	\$2,075.00
	Purchase-0002 (1)	2	\$3,250.00
	Purchase-0003 (1)	3	\$7,000.00
	Purchase-0004 (1)	4	\$9,500.00
Supplier-002 (1)	Purchase-0005 (1)	2	\$4,500.00
Total (5)		14	\$26,325.00

At the bottom of the report, there are checkboxes for Row Counts, Detail Rows, Subtotals, and Grand Total.

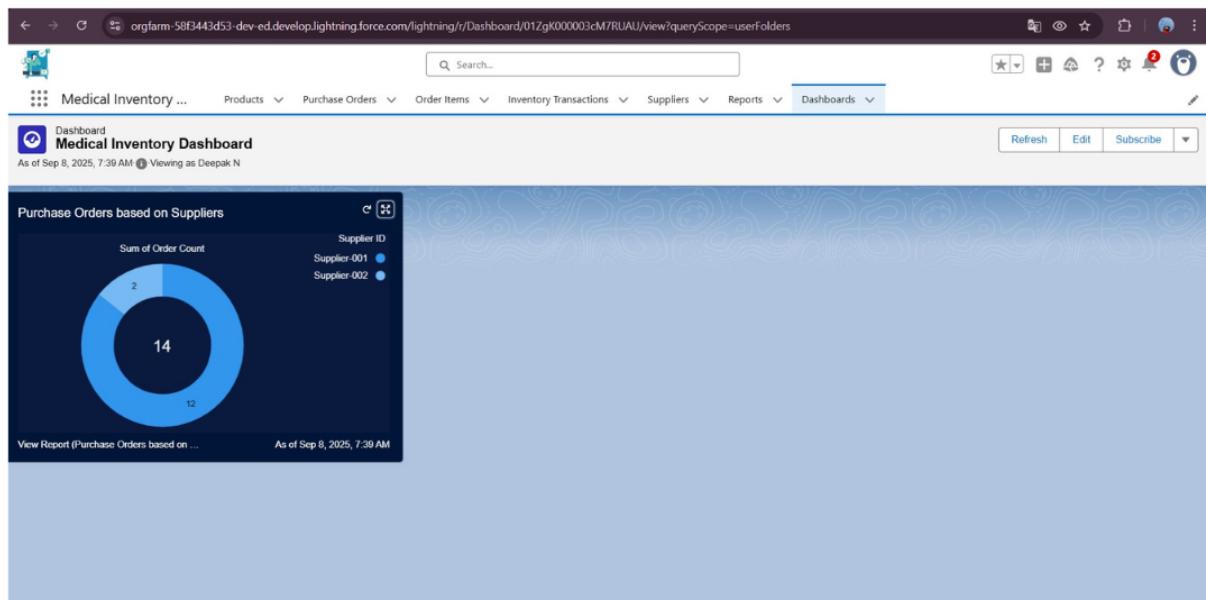
Create a Complete Purchase Details Report

The screenshot shows a Salesforce Lightning report titled "Complete Purchase Details Report". The report summary at the top indicates 14 total records, 14 total order count, 276 total quantity received, and a total amount of \$26,325.00. The main table lists purchase items grouped by supplier and delivery date. Supplier-001 has 12 entries. Supplier-002 has 8 entries. Supplier-003 has 3 entries. The table includes columns for Supplier ID, Actual Delivery Date, Purchase Order ID, Product ID, Order Count, Product Name, Quantity Received, and Amount.

Supplier ID	Actual Delivery Date	Purchase Order ID	Product ID	Order Count	Product Name	Quantity Received	Amount
Supplier-001 (12)	9/10/2025 (4)	Purchase-0004 (4)	Antibiotic Tablets	4	Antibiotic Tablets	40	\$6,000.00
			Paracetamol	4	Paracetamol 500mg	20	\$500.00
			Vitamin C Tablets	4	Vitamin C Tablets	10	\$500.00
			Antacid	4	Antacid	25	\$2,500.00
				4		95	\$9,500.00
				4		95	\$9,500.00
	9/11/2025 (8)	Purchase-0001 (3)	Antibiotic Tablets	3	Antibiotic Tablets	10	\$1,500.00
			Paracetamol	3	Paracetamol 500mg	7	\$175.00
			Antacid	3	Antacid	4	\$400.00
				3		21	\$2,075.00
		Purchase-0002 (2)	Antibiotic Tablets	2	Antibiotic Tablets	15	\$2,250.00
			Antacid	2	Antacid	10	\$1,000.00
				2		25	\$3,250.00
		Purchase-0003 (3)	Antibiotic Tablets	3	Antibiotic Tablets	24	\$3,600.00
			Paracetamol	3	Paracetamol 500mg	56	\$1,400.00

At the bottom of the report, there are checkboxes for Row Counts, Detail Rows, Subtotals, and Grand Total.

View Dashboard



RESULTS

- Tabs for Product, Supplier, Purchase Order, Inventory.
- Reports for Expired Products and Supplier Performance.
- Dashboard showing Stock Levels and Purchase Order Summary.
- Trigger execution results (auto-calculated total order amount).
- Validation Rule error messages (when wrong data is entered).

ADVANTAGES & DISADVANTAGES

Advantages

- Accurate tracking of products and expiry dates.
- Easy management of supplier and purchase orders.
- Reduced manual work with automation (flows and triggers).
- Visual dashboards for quick decision-making.

Disadvantages

- Requires Salesforce knowledge for customization.
- Limited offline functionality.
- Integration with external systems (e.g., hospital management software) not implemented yet.

APPENDIX

Create an Apex Trigger:

```
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 Apex Class:

```
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;
}
}
}
}

```

Future Enhancements

- Add **barcode scanning** for products to make stock entry faster.
 - Implement **email or SMS alerts** for products nearing expiry.
 - Create **mobile-friendly pages** for quick access by staff.
 - Add **AI predictions** for stock demand and reordering.
 - Integrate with **external hospital systems** for real-time updates.

CONCLUSION

The Medical Inventory Management System successfully streamlines the operations of managing medical supplies using Salesforce. It ensures better accuracy, reduces errors, and improves efficiency in handling suppliers, purchase orders, and products. With features like validation rules, flows, triggers, reports, and dashboards, the project demonstrates the practical use of Salesforce in real-time business scenarios.