

# Medical Inventory Management

---

**College Name:** Bishop Appasamy College of Arts and Science

**College Code:** BRU3G

**Team ID:** NM2025TMID20473

## **Team Members:**

- Team Leader: Pranesh S – pranesh8506s@gmail.com
- Member 1: Sakthi Dasan V – sakthidasan.b07@gmail.com
- Member 2: Indira Sidhu L – lssithu2005@gmail.com
- Member 3: Jenslin Samuel J – jenslinsamuel66@gmail.com

## **1. Introduction**

The Medical Inventory Management System is a comprehensive Salesforce application designed to streamline and manage various operational aspects of medical inventory. It efficiently maintains supplier details, manages purchase orders, tracks product details and transactions, and monitors product expiry dates. By implementing this system, operational efficiency, data accuracy, and reporting capabilities are significantly improved.

## 1.1 Project Overview

This project aims to develop a Salesforce-based Medical Inventory Management application for managing suppliers, purchase orders, products, inventory transactions, and reporting. The solution is designed for colleges and medical institutions to automate the traditional inventory process, ensuring transparency and real-time monitoring.

## 1.2 Purpose

The purpose of this project is to:

- Provide a centralized system for managing medical inventory.
- Improve efficiency by automating purchase orders and tracking supplier details.
- Reduce errors by validating critical fields such as expected delivery dates.
- Generate reports and dashboards for decision-making and performance monitoring.

## 2. Development Phase

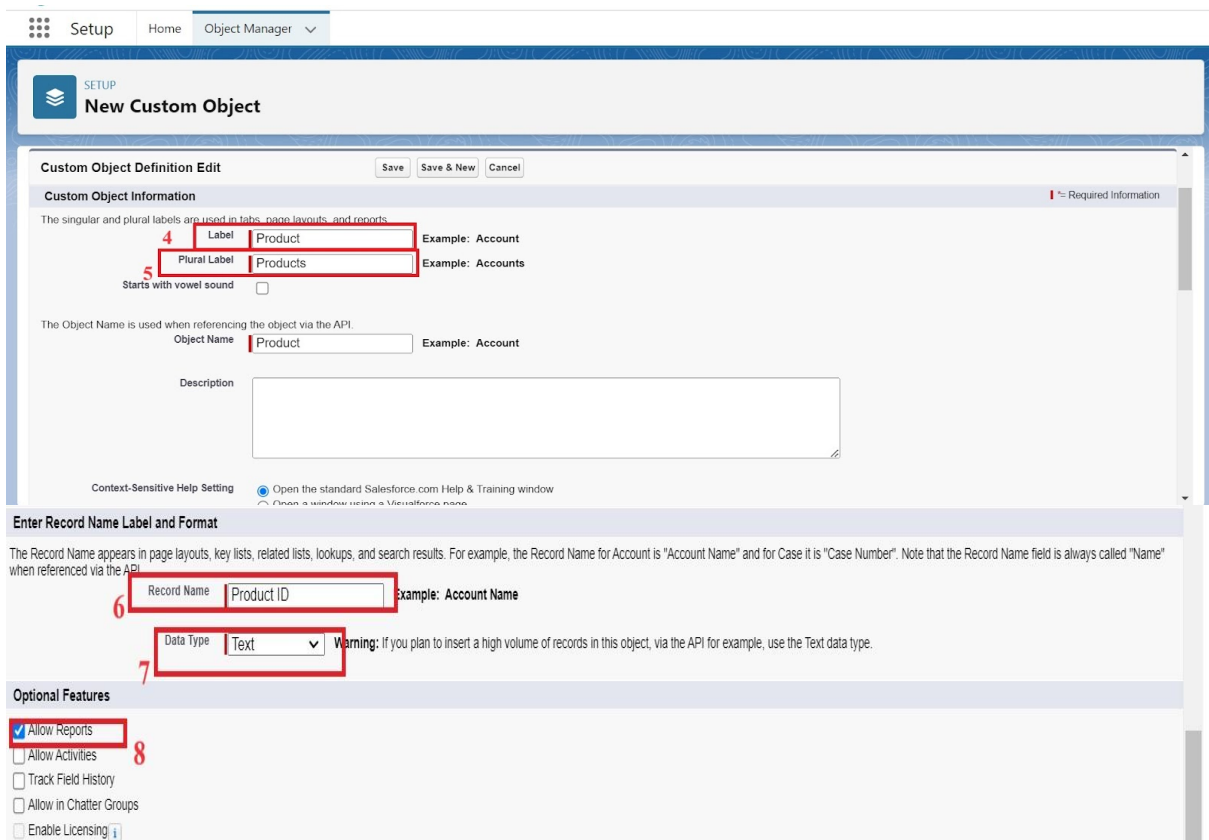
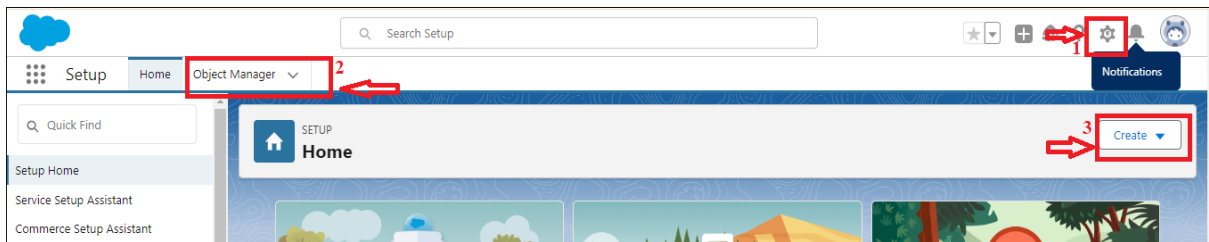
### 2.1 Creating Developer Account

A Salesforce Developer Account was created using the official website: <https://developer.salesforce.com>

## 2.2 Objects Creation

Custom objects created:

- Product
- Purchase Order
- Order Item
- Inventory Transaction
- Supplier



**Deployment Status** [What is this?](#)

☐ In Development  
☒ Deployed

**Search Status**

When this setting is enabled, your users can find records of this object type when they search. [Learn more.](#)

☒ Allow Search <sup>9</sup>

**Object Creation Options (Available only when custom object is first created)**

☐ Add Notes and Attachments related list to default page layout  
☐ Launch New Custom Tab Wizard after saving this custom object

<sup>10</sup>

Activate Windows  
Go to Settings to activate Windows.

## 2.3 Tabs and Lightning App

- Tabs were created for each object.
- A Lightning App named “Medical Inventory Management” was built.

Setup Home Object Manager

Q. tab <sup>1</sup>

Feature Settings

Analytics

Tableau

Tableau Embedding

User Interface

Loaded Console Tab Limit

Rename Tabs and Labels

**Tabs** <sup>2</sup>

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

**SETUP Tabs**

**Custom Tabs** [Help for this Page](#)

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

Custom Object tabs look and behave like the standard tabs provided with Salesforce. Web tabs allow you to embed external web applications and content within the Salesforce window. Visualforce tabs allow you to embed Visualforce pages. Lightning Component tabs allow you to add Lightning components to the navigation menu in Lightning Experience and the mobile app. Lightning Page tabs allow you to add Lightning Pages to Lightning Experience and the mobile app.

**Custom Object Tabs** <sup>3</sup>  [What Is This?](#)

No Custom Object Tabs have been defined

**Web Tabs**  [What Is This?](#)

No Web Tabs have been defined

**Visualforce Tabs**  [What Is This?](#)

No Visualforce Tabs have been defined

Activate Windows

SETUP

Tabs

Step 1. Enter the Details

Step 1 of 3

Choose the custom object for this new custom tab. Fill in other details.

Select an existing custom object or create a new one.

Object

Product

Tab Style

Stethoscope

(Optional) Choose a Home Page Custom Link to show as a splash page the first time your users click on this tab.

Splash Page Custom Link

--None--

Enter a short description.

Description

Next

Cancel

SetupHomeObject Manager

App man

Apps

App Manager

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

SETUP

Lightning Experience App Manager

New Lightning App

New Connected App

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

App Name

Developer Name

Description

Last Modified ...

App Type

Vi...

### New Lightning App

App Details

App Branding

\*App Name

Medical Inventory Management

\*Developer Name

Medical\_Inventory\_Management

Description

Enter a description...

Image

Primary Color Hex Value

#0070D2

Org Theme Options

☐ Use the app's image and color instead of the org's custom theme

App Launcher Preview

Next

## Navigation Items

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.

The screenshot shows the 'Available Items' and 'Selected Items' interface. The 'Available Items' section on the left has a search bar with 'Dash' and a list item 'Dashboards'. The 'Selected Items' section on the right lists 'Products', 'Purchase Orders', 'Order Items', 'Inventory Transactions', 'Suppliers', and 'Reports'. A red box highlights the 'Available Items' and 'Selected Items' sections, with a red number '6' to its right.

The screenshot shows the 'New Lightning App' 'User Profiles' configuration screen. The title is 'User Profiles' with the instruction 'Choose the user profiles that can access this app.' Below this, there are two sections: 'Available Profiles' and 'Selected Profiles'. The 'Available Profiles' section has a search bar with 'system' and a list item 'System Administrator'. A red box highlights the 'Available Profiles' section, with a red number '7' to its left. At the bottom of the screen, there is a 'Back' button on the left and a 'Save & Finish' button on the right, which is highlighted with a red box and a red number '8'.

## 2.4 Fields and Page Layouts

- Fields such as Product Name, Unit Price, Current Stock Level were created.
- Layouts were arranged for better usability.

Setup

Home

Object Manager

Search Setup

1

Object Manager

9 Items, Sorted by Label

Product

Schema Builder

Create

Label	API Name	Type	Description	Last Modified	Deployed
Fulfillment Order Product	FulfillmentOrderLineItem	Standard Object			
Opportunity Product	OpportunityLineItem	Standard Object			
Order Product	OrderItem	Standard Object			
Product	Product_c	Custom Object		18/06/2024	✓
Product	Product2	Standard Object			
Product Attribute	ProductAttribute	Standard Object			

Setup

Home

Object Manager

Product

Details

Fields & Relationships

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

Object Limits

Record Types

Related Lookup Filters

Search Layouts

Fields & Relationships

4 Items, Sorted by Field Label

Quick Find

New

Deleted Fields

Field Dependencies

Set History Tracking

Field Label	Field Name	Data Type	Controlling Field	Indexed
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Product ID	Name	Text(80)		✓

Geolocation

Number

Percent

Phone

Picklist

Picklist (Multi-Select)

Text

Text Area

Text Area (Long)

Text Area (Rich)

Text (Encrypted)

Time

URL

clicking Send an Email. Note that custom email addresses cannot be used for mass emails.

Allows users to define locations. Includes latitude and longitude components, and can be used to calculate distance.

Allows users to enter any number. Leading zeros are removed.

Allows users to enter a percentage number, for example, '10' and automatically adds the percent sign to the number.

Allows users to enter any phone number. Automatically formats it as a phone number.

Allows users to select a value from a list you define.

Allows users to select multiple values from a list you define.

Allows users to enter any combination of letters and numbers.

Allows users to enter up to 255 characters on separate lines.

Allows users to enter up to 131,072 characters on separate lines.

Allows users to enter formatted text, add images and links. Up to 131,072 characters on separate lines.

Allows users to enter any combination of letters and numbers and store them in encrypted form.

Allows users to enter a local time. For example, "2:40 PM", "14:40", "14:40:00", and "14:40:50.600" are all valid times for this field.

Allows users to enter any valid website address. When users click on the field, the URL will open in a separate browser window.

Next

Cancel

Step 2. Enter the details Step 2 of 4

Previous **Next** Cancel

Field Label  7

Please enter the maximum length for a text field below.

Length  9

Field Name  7

Description

Help Text

Required ☒ Always require a value in this field in order to save a record 8

Unique ☐ Do not allow duplicate values

☐ Treat "ABC" and "abc" as duplicate values (case insensitive)

☐ Treat "ABC" and "abc" as different values (case sensitive)

External ID ☐ Set this field as the unique record identifier from an external system

Auto add to custom report type ☒ Add this field to existing custom report types that contain this entity 1

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

Quick Find

Fields

- Buttons
- Quick Actions
- Mobile & Lightning Actions
- Expanded Lookups
- Related Lists
- Report Charts

Section	Last Modified By	Product ID
Blank Space	Minimum Stock Level	Product Name
Created By	Owner	Unit Price
Current Stock Level	Product Description	

Information (Header visible on edit only)

Product ID	Sample Text	Unit Price	₹123.45
Product Name	Sample Text	Current Stock Level	12,420
Product Description	Sample Text	Minimum Stock Level	21,114
		Owner	Sample Text

System Information (Header visible on edit only)

Created By Sample Text Last Modified By Sample Text

## 2.5 Validation Rules

Validation was implemented for Purchase Orders:

$(\text{Expected\_Delivery\_Date\_c} - \text{Order\_Date\_c}) > 7$

Error: "The Expected Delivery Date should not exceed 7 days."



**Purchase Order Validation Rule** Help for this Page

Define a validation rule by specifying an error condition and a corresponding error message. The error condition is written as a Boolean formula expression that returns true or false. When the formula expression returns true, the save will be aborted and the error message will be displayed. The user can correct the error and try again.

---

**Validation Rule Edit** Save Save & New Cancel

Rule Name  3

Active ☒ 4

Description

**Error Condition Formula** ⓘ = Required Information

Example:  [More Examples...](#)  
 Display an error if Discount is more than 30%  
 If this formula expression is **true**, display the text defined in the Error Message area

Insert Field Insert Operator (Expected\_Delivery\_Date\_\_c - Order\_Date\_\_c) > 7 5

Functions  
 -- All Function Categories --  
 ABS  
 ACOS  
 ADDMONTHS  
 AND  
 ASCII  
 ASIN

**Error Message**

Example:   
 This message will appear when Error Condition formula is **true**

Error Message  6

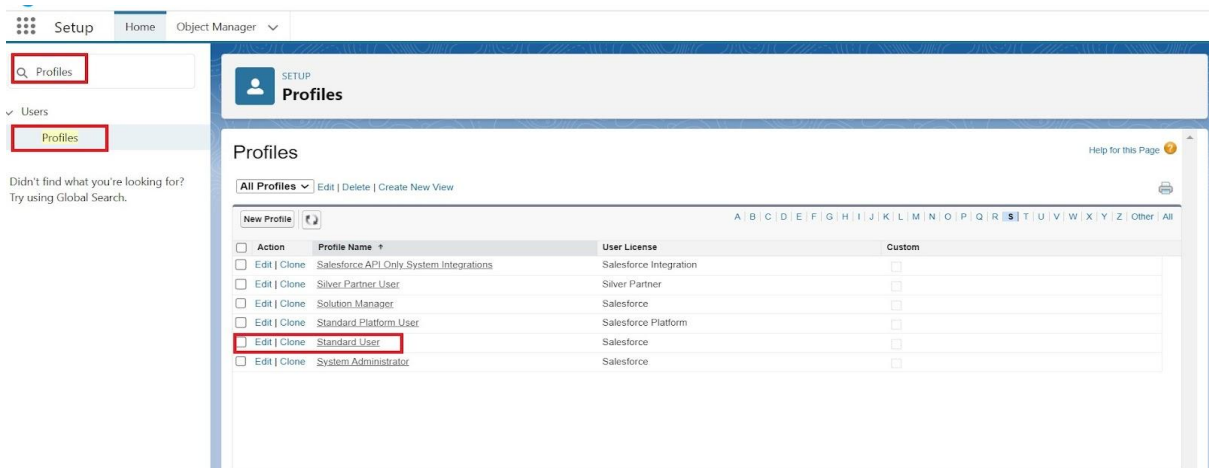
This error message can either appear at the top of the page or below a specific field on the page

Error Location ☒ Top of Page ☐ Field 7

8 Save Save & New Cancel

## 2.6 Profiles, Roles, and Permission Sets

- Created a profile: Inventory Manager.
- Created a role: Purchasing Manager.
- Created permission sets for Purchase Manager Create Access.



## Clone Profile

Enter the name of the new profile.

**You must select an existing profile to clone from.**

Existing Profile	Standard User
User License	Salesforce
Profile Name	<input type="text" value="Inventory Manager"/>

## 2.7 Flows

A Record-Triggered Flow was created to automatically update the Actual Delivery Date based on the order date + 3 days.

## New Flow

Select how you'd like to start building your automation.



### Start From Scratch

Select your automation type and start building on an empty canvas.



### Use a Template

Select a pre-built flow and customize it to fit your needs.

Back

Next

## New Flow

Core All + Templates



### Screen Flow

Guides users through a business process that's launched from Lightning pages, Experience Cloud sites, quick actions, and ...



### Record-Triggered Flow

Launches when a record is created, updated, or deleted. This autolaunched flow runs in the background.

2



### Schedule-Triggered Flow

Launches at a specified time and frequency for each record in a batch. This autolaunched flow runs in the background.



### Platform Event—Triggered Flow

Launches when a platform event message is received. This autolaunched flow runs in the background.



### Autolaunched Flow (No Trigger)

Launches when invoked by Apex, processes, REST API, and more. This autolaunched flow runs in the background.



### Record-Triggered Orchestration

Launches when a record is created or updated. An orchestration lets you create a multi-step, multi-user process.

Create



## Configure Start



### Select Object

Select the object whose records trigger the flow when they're created, updated, or deleted.

\* Object

Purchase Order

3

### Configure Trigger

\* Trigger the Flow When:

- ☐ A record is created
- ☐ A record is updated
- ☒ A record is created or updated
- ☐ A record is deleted

4

## Set Entry Conditions

Specify entry conditions to reduce the number of records that trigger the flow and the number of times the flow is executed. Minimizing unnecessary flow executions helps to conserve your org's resources.

If you create a flow that's triggered when a record is updated, we recommend first defining entry conditions. Then select the **Only when a record is updated to meet the condition requirements** option for When to Run the Flow for Updated Records.

Condition Requirements

None ▼

5

\*Optimize the Flow for:

**Fast Field Updates**

Update fields on the record that triggers the flow to run. This high-performance flow runs *before* the record is saved to the database.

6

**Actions and Related Records**

Update any record and perform actions, like send an email. This more flexible flow runs *after* the record is saved to the database.

☐ Include a Run Asynchronously path to access an external system after the original transaction for the triggering record is successfully committed

## 2.8 Triggers

Apex Trigger and Handler were written to calculate the Total Order Cost dynamically from Order Items.

**New Apex Trigger**

Name: CalculateTotalAmountTrigge

sObject: Order\_Item\_\_c ▼

Submit

## 3. Functional and Performance Testing

- Validated creation of records for all objects.
- Tested validation rules for expected delivery dates.
- Verified reports and dashboards for accuracy.
- Ensured Apex triggers update Total Order Cost correctly.

- Checked performance in terms of record updates and flow execution.

## 4. Results

- Successfully created Salesforce Medical Inventory Management App.
- Flows and triggers automated key processes.
- Reports summarized purchase orders by suppliers.
- Dashboards provided real-time visualization of data.

## 5. Outputs and Screenshots

Outputs included:

- Custom Objects & Fields
- Validation Rules
- Flows & Triggers
- Reports & Dashboards

## 6. Advantages and Disadvantages

### Advantages:

- Real-time tracking of medical inventory.
- Automated cost calculation and delivery date updates.
- Easy report generation for management.
- Enhanced data accuracy and security via Salesforce Cloud.

## Disadvantages:

- Dependent on internet connectivity.
- Requires Salesforce knowledge for customization.
- Limited offline access.

## 7. Conclusion

The Medical Inventory Management Salesforce Application successfully achieved its objectives by automating inventory processes, improving accuracy, and providing visual dashboards. This solution can be extended further with advanced analytics, mobile integration, and AI-driven inventory predictions.

## 8. Appendix

### Coding:

1.

```
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);  
}
```

2.

```
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;
}
}
}
}
```