Medical Inventory Management

College Name: Bishop Appasamy College of Arts and Science
College Code: BRU3G
Team ID: NM2025TMID20459

Team Members:
- Team Leader: Adithya R – adithyavar6@gmail.com
- Member 1: Prabhu S – prabhu7339201211@gmail.com
- Member 2: Jeeva P – jeevapandi2006@gmail.com
- Member 3: Sridhar M – sridharmanikandan51@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

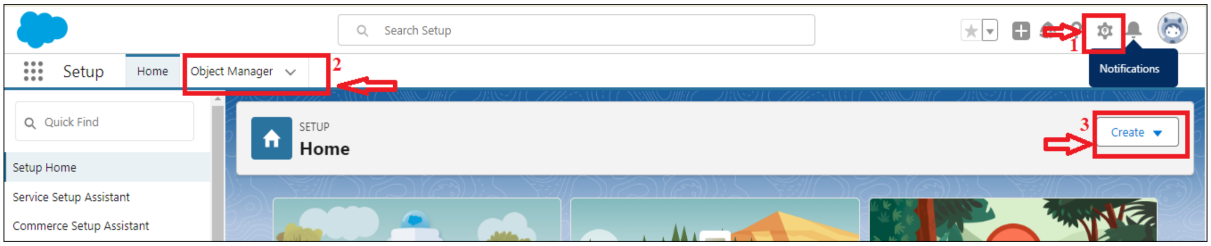
2.1 Creating Developer Account

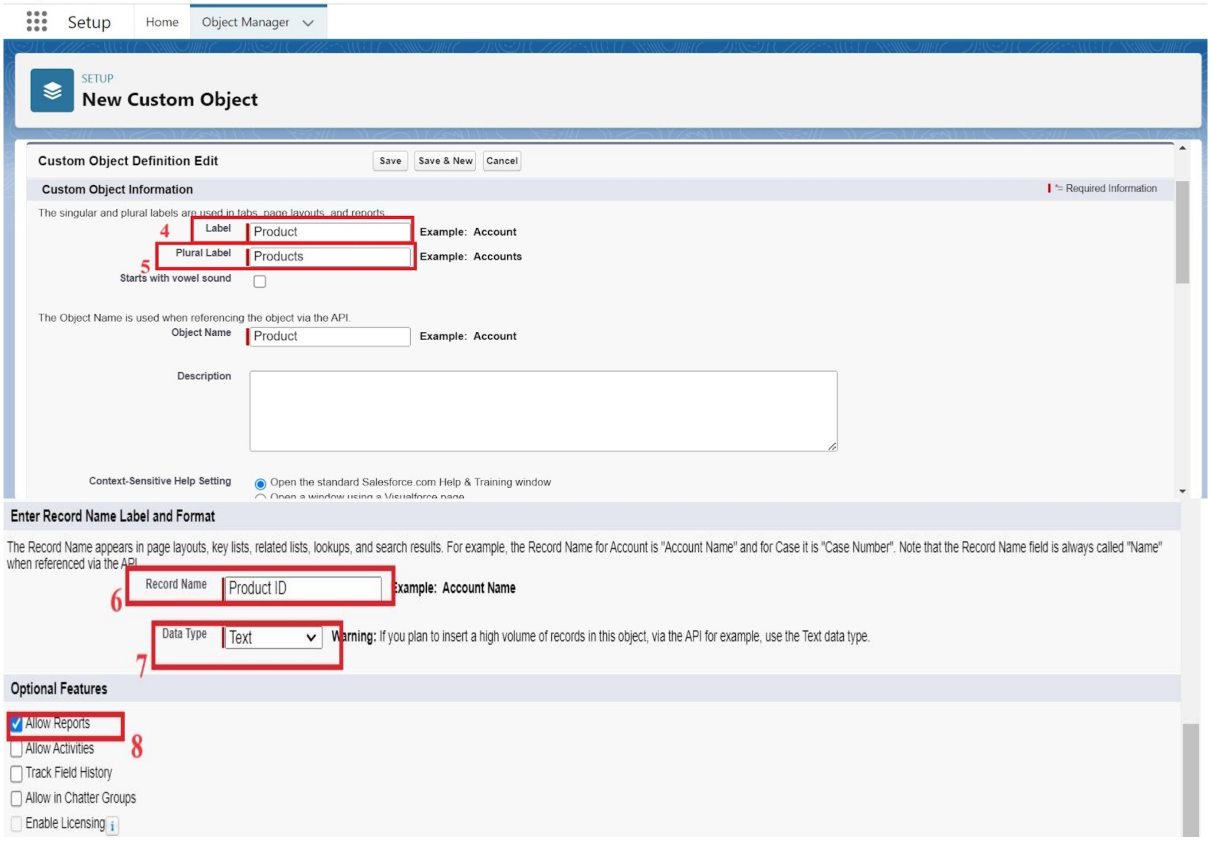
A Salesforce Developer Account was created using the official website: https://developer.salesforce.com

2.2 Objects Creation

Custom objects created:

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





Custom Object Definition Edit

Custom Object Information

| = Required Information

The singular and plural labels are used in tabs, page layouts, and reports.

Label: Product — Example: Account

Plural Label: Products — Example: Accounts

Starts with vowel sound: □

The Object Name is used when referencing the object via the API.

Object Name: Product — Example: Account

Description:

Context-Sensitive Help Setting: ● Open the standard Salesforce.com Help & Training window ○ Open a window using a Visualforce page

Enter Record Name Label and Format

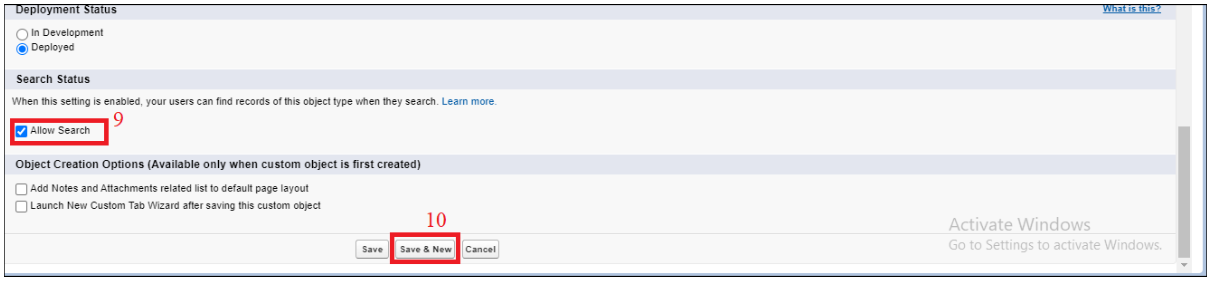
The Record Name appears in page layouts, key lists, related lists, lookups, and search results. For example, the Record Name for Account is "Account Name" and for Case it is "Case Number". Note that the Record Name field is always called "Name" when referenced via the API.

Record Name: Product ID — Example: Account Name

Data Type: Text — Warning: If you plan to insert a high volume of records in this object, via the API for example, use the Text data type.

Optional Features

☑ Allow Reports
□ Allow Activities
□ Track Field History
□ Allow in Chatter Groups
□ Enable Licensing



Deployment Status

○ 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

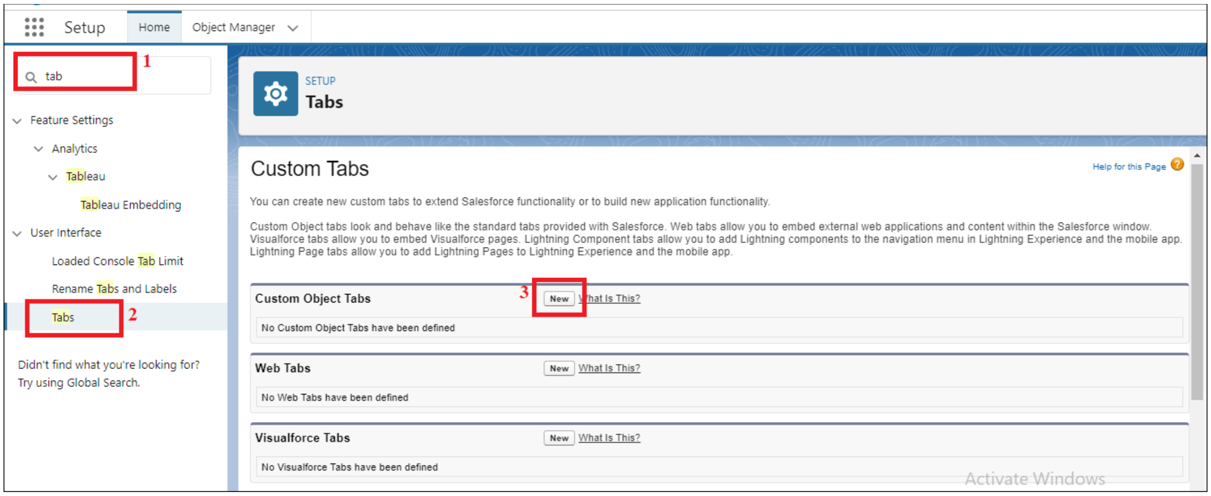
Object Creation Options (Available only when custom object is first created)

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

Save | Save & New | Cancel

2.3 Tabs and Lightning App

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



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

Custom Tabs

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

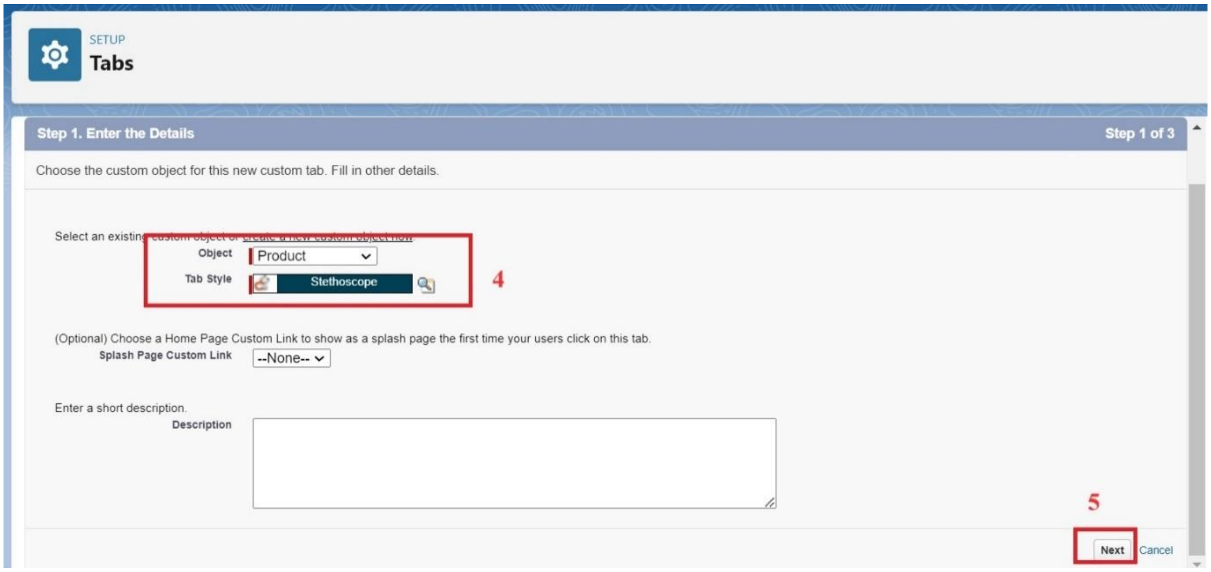
No Custom Object Tabs have been defined

Web Tabs

No Web Tabs have been defined

Visualforce Tabs

No Visualforce Tabs have been defined



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 custom object now

Object: Product

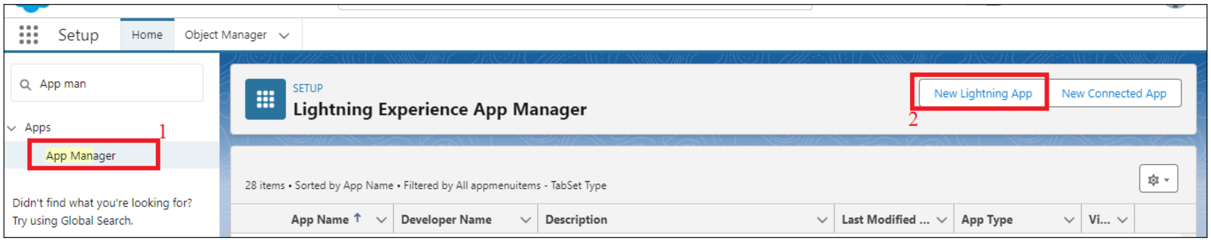
Tab Style: Stethoscope

(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:



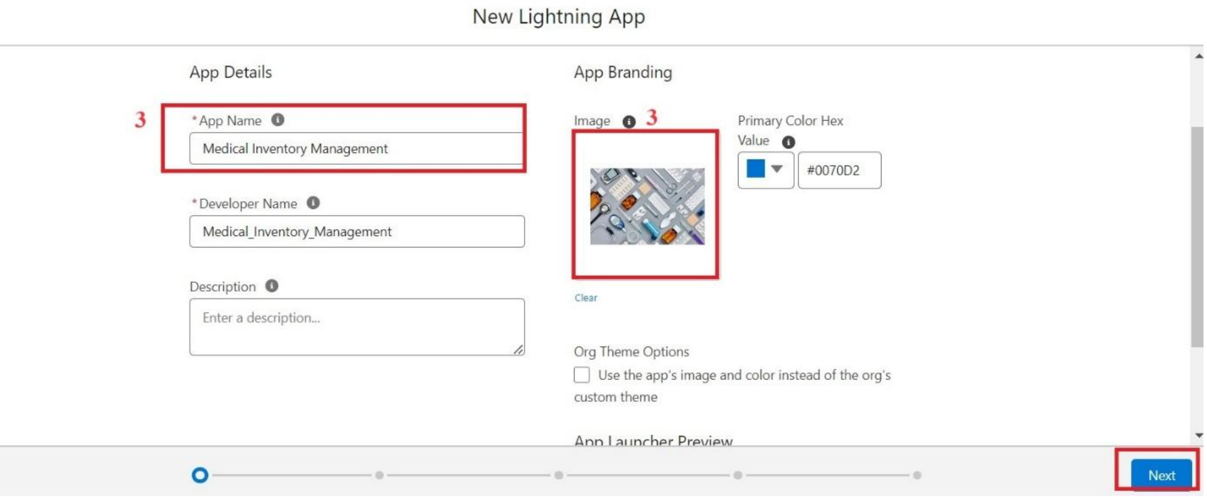
SETUP

Lightning Experience App Manager

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

App Name ↑	Developer Name	Description	Last Modified ...	App Type	Vi...

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



New Lightning App

App Details

*App Name: Medical Inventory Management

*Developer Name: Medical_Inventory_Management

Description: Enter a description...

App Branding

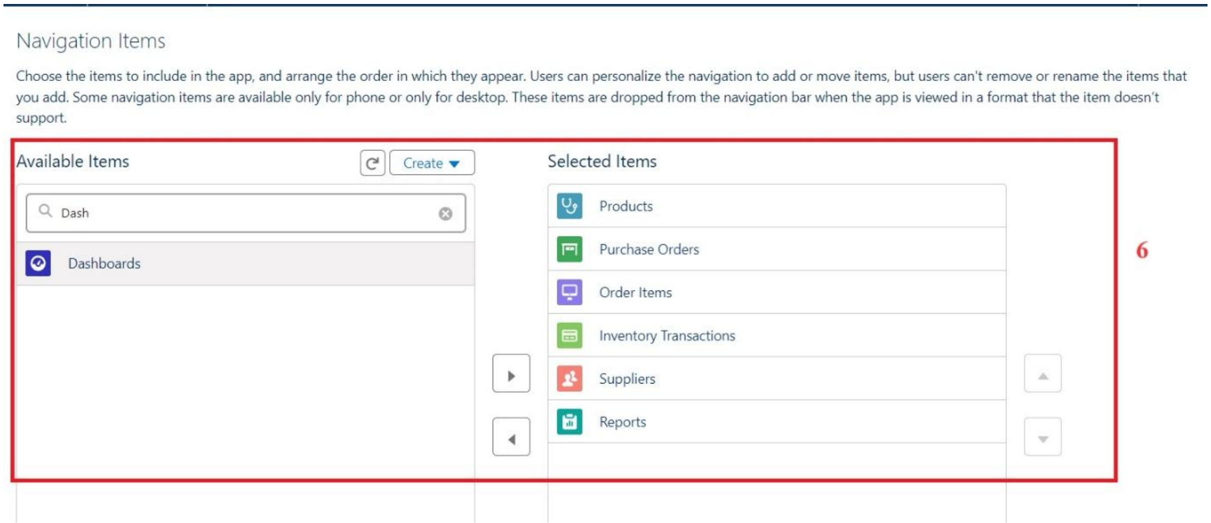
Image

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



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.

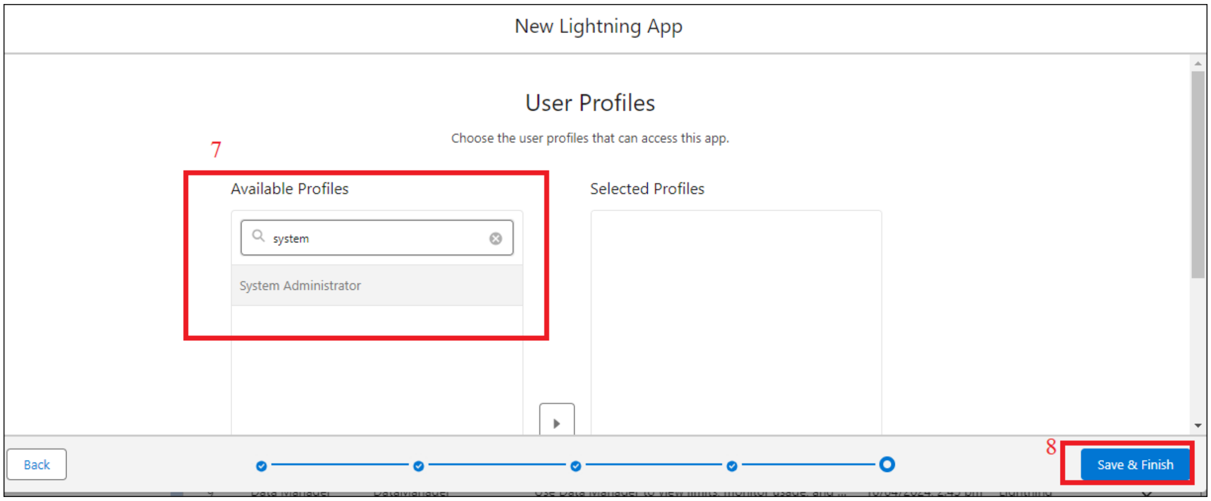
Available Items

Dash

- Dashboards

Selected Items

- Products
- Purchase Orders
- Order Items
- Inventory Transactions
- Suppliers
- Reports



New Lightning App

User Profiles

Choose the user profiles that can access this app.

Available Profiles

system

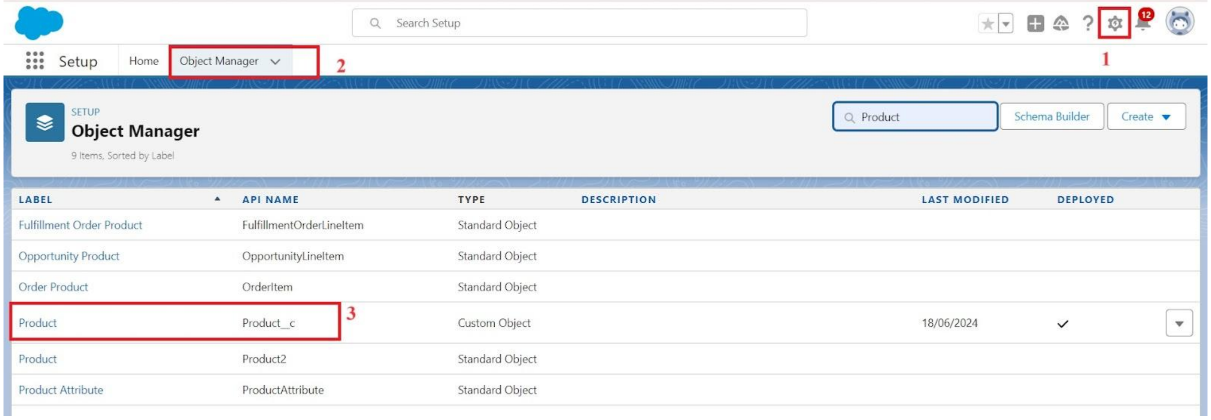
- System Administrator

Selected Profiles

2.4 Fields and Page Layouts

- Fields such as Product Name, Unit Price, Current Stock Level were created.

- Layouts were arranged for better usability.



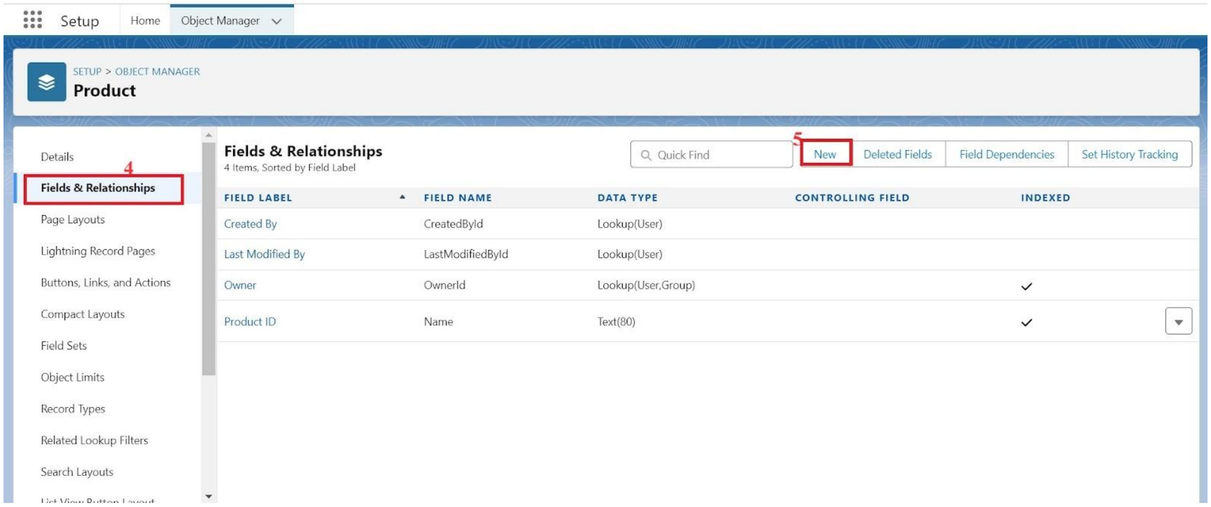
SETUP

Object Manager

9 Items, Sorted by Label

Product

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			



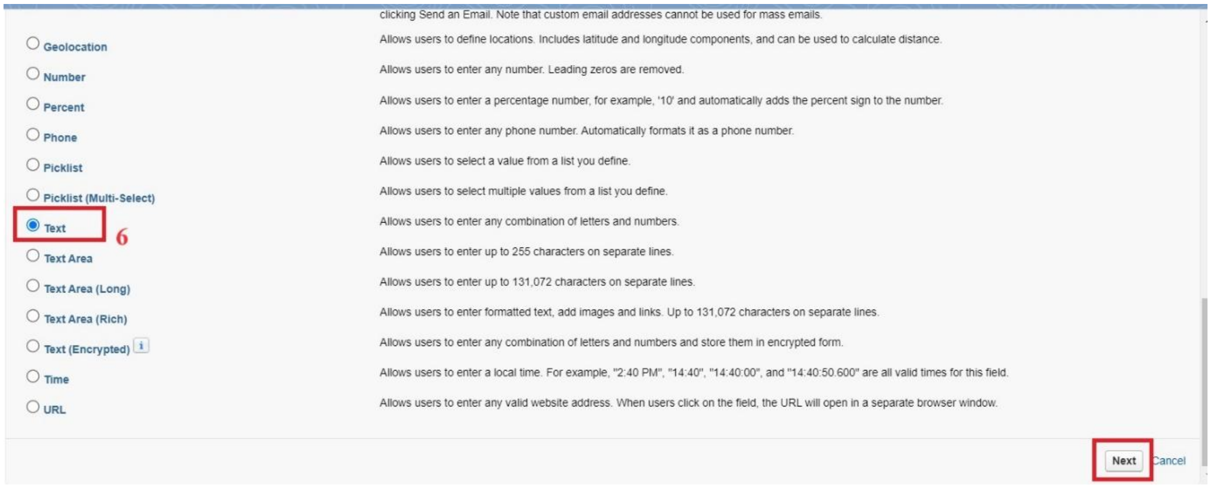
SETUP > OBJECT MANAGER

Product

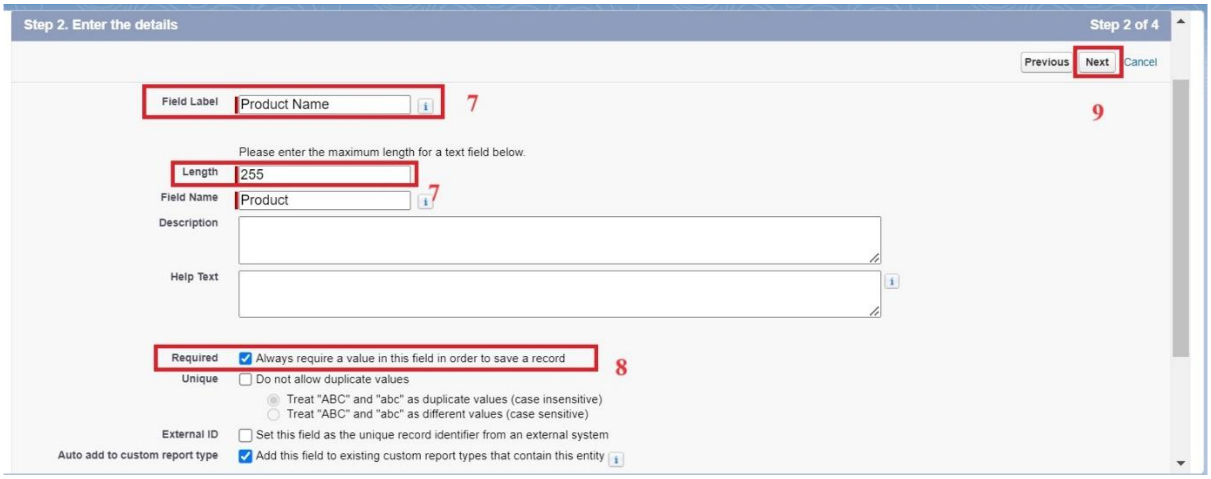
Fields & Relationships

4 Items, Sorted by Field Label

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)		✓



	clicking Send an Email. Note that custom email addresses cannot be used for mass emails.
○ Geolocation	Allows users to define locations. Includes latitude and longitude components, and can be used to calculate distance.
○ Number	Allows users to enter any number. Leading zeros are removed.
○ Percent	Allows users to enter a percentage number, for example, '10' and automatically adds the percent sign to the number.
○ Phone	Allows users to enter any phone number. Automatically formats it as a phone number.
○ Picklist	Allows users to select a value from a list you define.
○ Picklist (Multi-Select)	Allows users to select multiple values from a list you define.
● Text	Allows users to enter any combination of letters and numbers.
○ Text Area	Allows users to enter up to 255 characters on separate lines.
○ Text Area (Long)	Allows users to enter up to 131,072 characters on separate lines.
○ Text Area (Rich)	Allows users to enter formatted text, add images and links. Up to 131,072 characters on separate lines.
○ Text (Encrypted)	Allows users to enter any combination of letters and numbers and store them in encrypted form.
○ Time	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.
○ URL	Allows users to enter any valid website address. When users click on the field, the URL will open in a separate browser window.



Step 2. Enter the details   Step 2 of 4

Field Label: Product Name

Please enter the maximum length for a text field below.

Length: 255

Field Name: Product

Description:

Help Text:

Required: ☑ Always require a value in this field in order to save a record

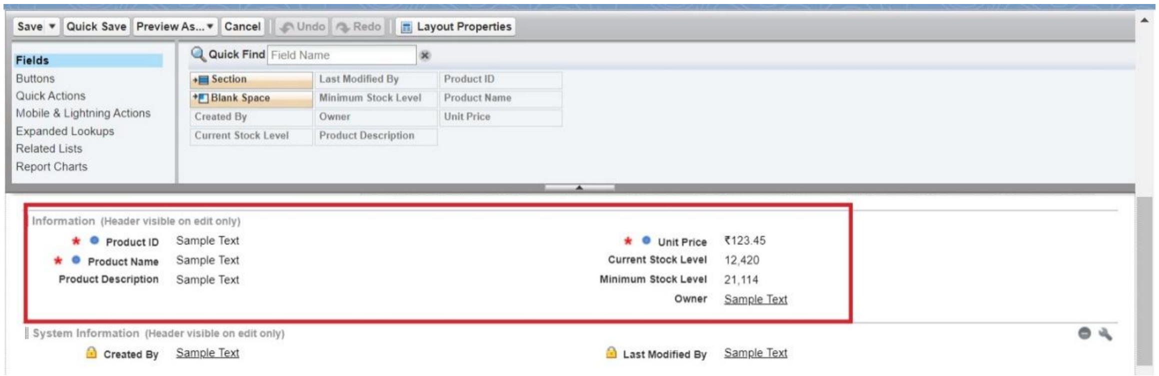
Unique: □ Do not allow duplicate values

○ 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



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)

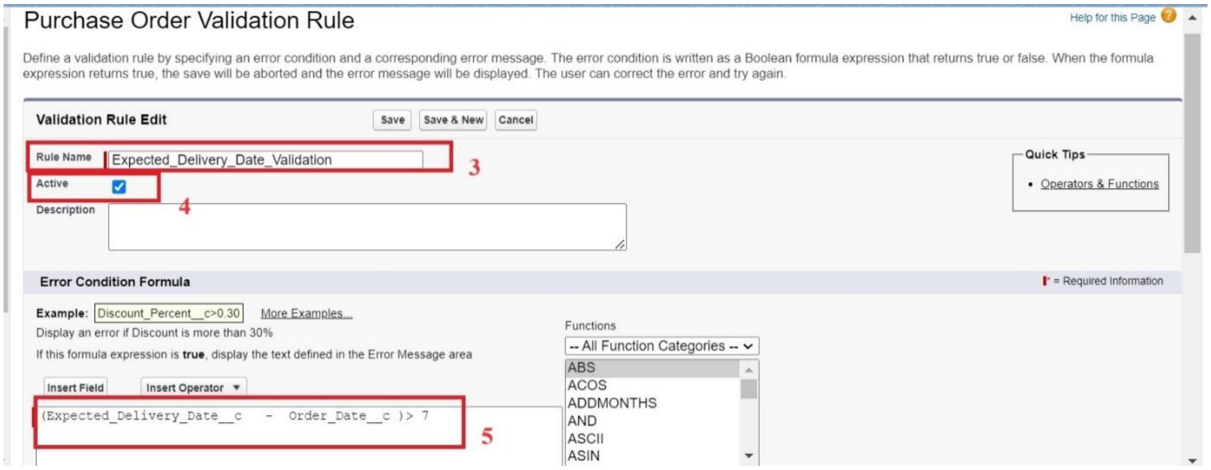
Product ID	Sample Text	Unit Price	₹123.45
Product Name	Sample Text	Current Stock Level	12,420
Product Description	Sample Text	Minimum Stock Level	21,114
		Owner	Sample Text

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:
(Expected_Delivery_Date__c - Order_Date__c) > 7
Error: “The Expected Delivery Date should not exceed 7 days.”



Purchase Order Validation Rule

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

Rule Name: Expected_Delivery_Date_Validation

Active: ☑

Description:

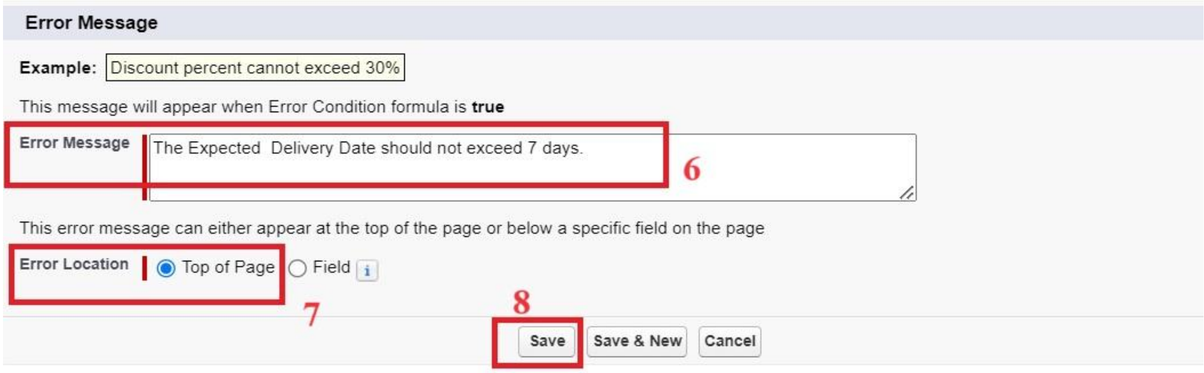
Error Condition Formula

Example: Discount_Percent__c>0.30

Display an error if Discount is more than 30%

If this formula expression is **true**, display the text defined in the Error Message area

```
(Expected_Delivery_Date__c   -   Order_Date__c )> 7
```



Error Message

Example: Discount percent cannot exceed 30%

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

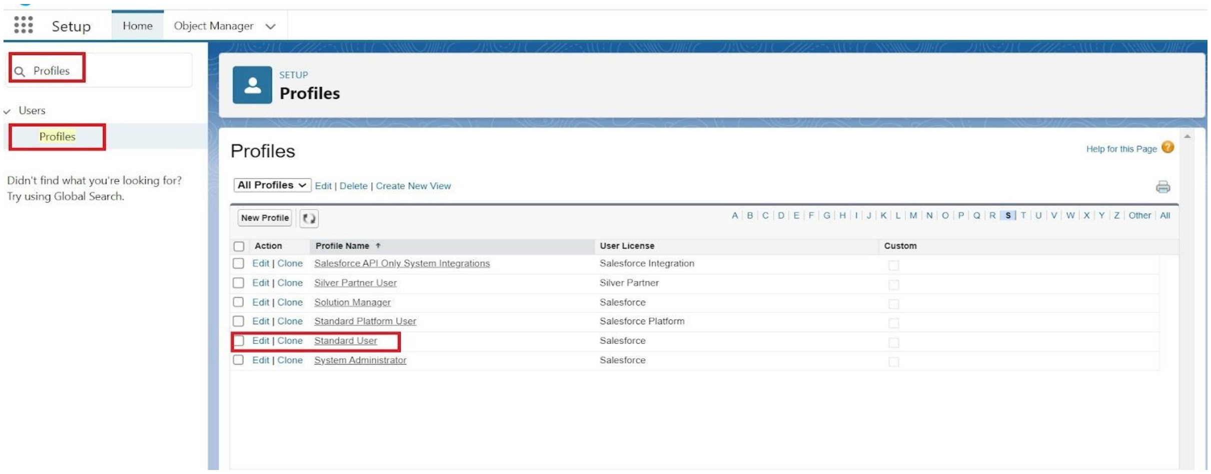
Error Message: The Expected Delivery Date should not exceed 7 days.

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

2.6 Profiles, Roles, and Permission Sets

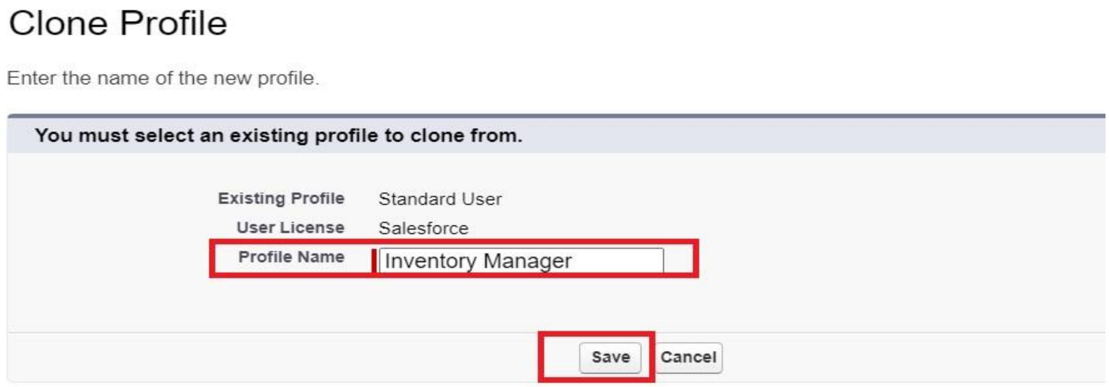
- Created a profile: Inventory Manager.
- Created a role: Purchasing Manager.
- Created permission sets for Purchase Manager Create Access.


Setup
Home
Object Manager
Profiles
Users
Profiles
Didn't find what you're looking for? Try using Global Search.
SETUP
Profiles

Profiles

Help for this Page
All Profiles
Edit | Delete | Create New View
New Profile
A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | Other | All

□	Action	Profile Name ↑	User License	Custom
□	Edit | Clone	Salesforce API Only System Integrations	Salesforce Integration	□
□	Edit | Clone	Silver Partner User	Silver Partner	□
□	Edit | Clone	Solution Manager	Salesforce	□
□	Edit | Clone	Standard Platform User	Salesforce Platform	□
□	Edit | Clone	Standard User	Salesforce	□
□	Edit | Clone	System Administrator	Salesforce	□


Clone Profile

Enter the name of the new profile.

You must select an existing profile to clone from.

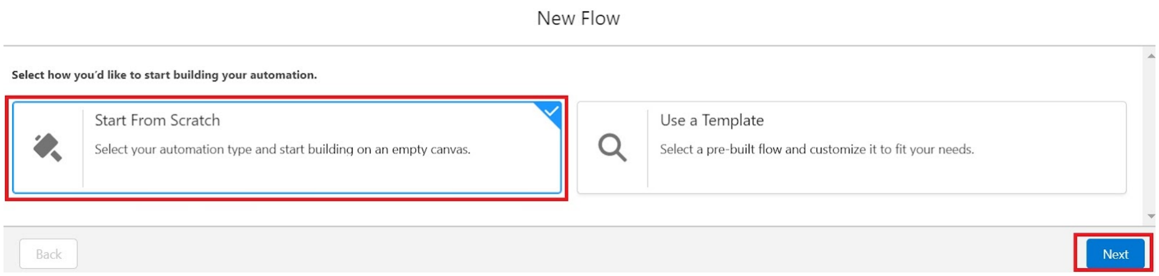
Existing Profile: Standard User

User License: Salesforce

Profile Name: 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.

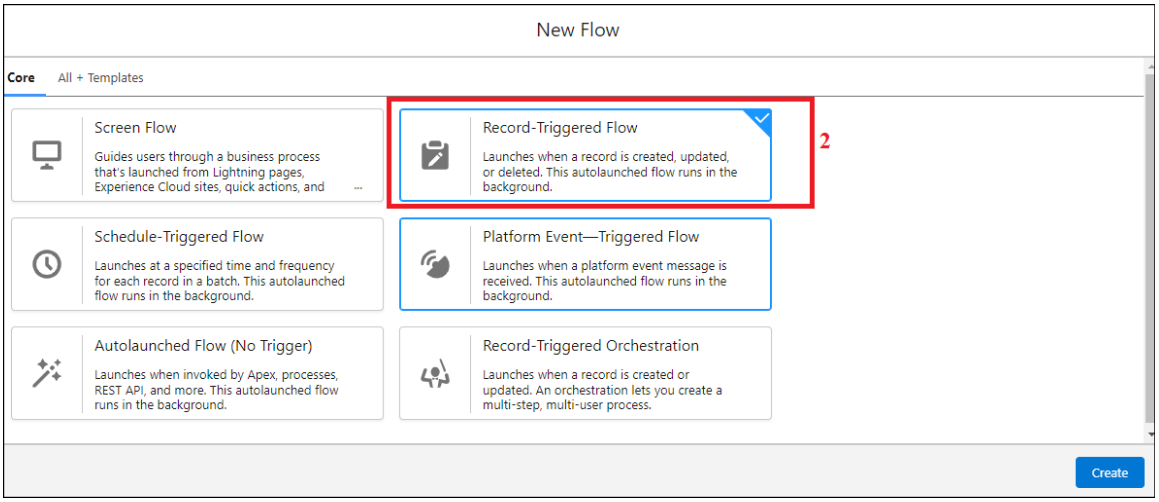


New Flow

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.



New Flow

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

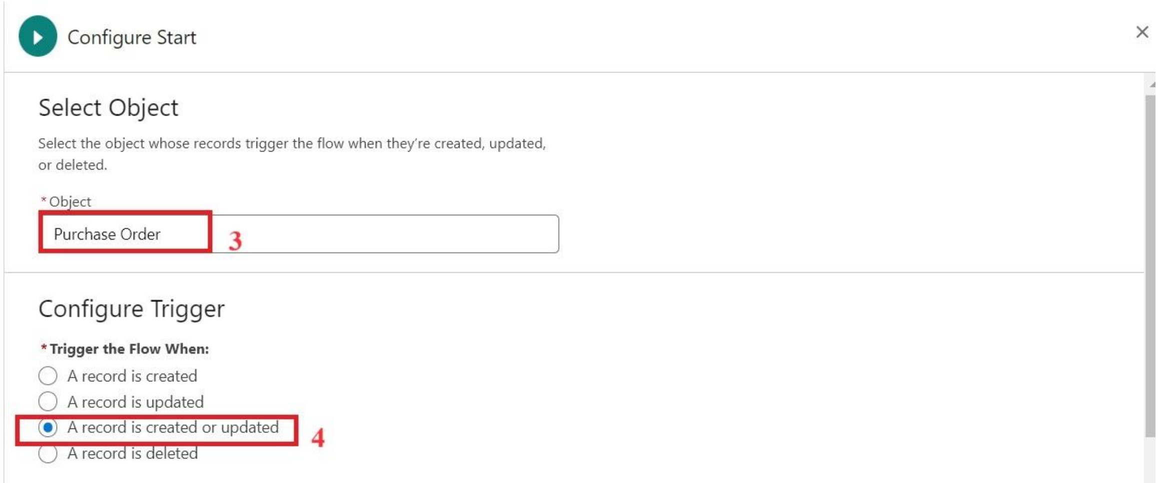
Record-Triggered Flow
Launches when a record is created, updated, or deleted. This autolaunched flow runs in the background.

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.



Configure Start

Select Object

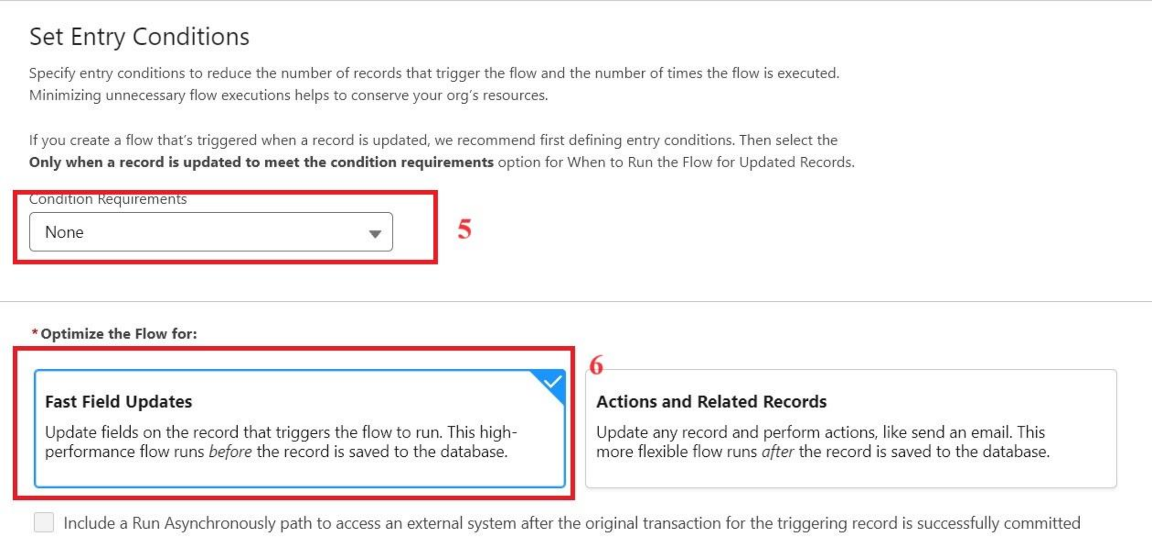
Select the object whose records trigger the flow when they're created, updated, or deleted.

* Object
Purchase Order

Configure Trigger

* Trigger the Flow When:

○ A record is created
○ A record is updated
● A record is created or updated
○ A record is deleted



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

* **Optimize the Flow for:**

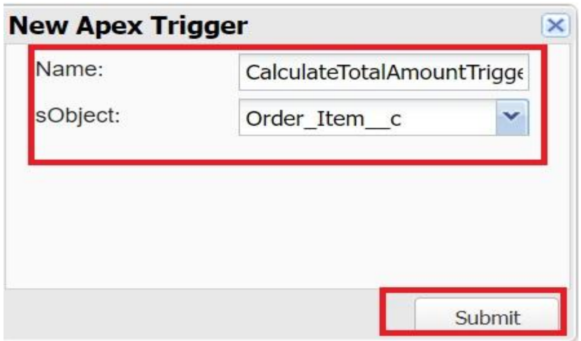
Fast Field Updates ✓
Update fields on the record that triggers the flow to run. This high-performance flow runs *before* the record is saved to the database.

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

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;
            }
        }
    }
}
```