PROJECT TITLE

Medical Inventory Management System

College Name: University College of Engineering, BIT Campus, Trichy

College Code: 8100

Team ID: NM2025TMID00739

Team Size: 4

Team Members:

Team Leader: Abirami P

Email: abiramipalanisamy777@gmail.com

Team Member1: Thilagavathi M

Email: thilagavathimathavan07@gmail.com

Team Member2: Atchaya R

Email: atchayaramesh06@gmail.com

Team Member3: Shanmugeshwaran S

Email: shanmugeshwaransuresh@gmail.com

Medical Inventory Management

Introduction:

Salesforce is a powerful **cloud-based Customer Relationship Management (CRM)** platform designed to streamline business processes, improve collaboration, and enhance customer relationships. With its vast suite of automation, analytics, and customization tools, Salesforce enables organizations to operate efficiently across diverse industries — including healthcare.In this project, Salesforce CRM has been leveraged to design and implement a **Medical Inventory Management System**. The goal is to automate the handling of medical supplies — including medicine stock, supplier management, purchase orders, and expiry tracking — for hospitals and pharmacies.By using Salesforce features such as **Custom Objects, Workflows, Process Builder, Apex Triggers, Validation Rules, Flows, and Dashboards**, this system ensures:

- Real-time inventory visibility
- Automated reorder alerts
- Expiry date monitoring
- Streamlined communication between administrators, suppliers, and pharmacists

This project highlights how Salesforce CRM can transform healthcare operations by improving efficiency, reducing manual errors, and enhancing data-driven decision-making.

Objectives

The primary objectives of the Medical Inventory Management System are:

1. **Centralized Inventory Data**
   Maintain detailed, structured records of medicines, suppliers, purchase orders, and stock movements within Salesforce.
2. **Automated Reorder Alerts**
   Generate notifications when medicine stock falls below a defined minimum threshold, ensuring timely restocking.
3. **Expiry Tracking**
   Continuously monitor expiry dates and send alerts to staff for disposal or replacement of expired medicines.
4. **Supplier Management**
   Record supplier profiles, contact details, payment status, and purchase history to streamline vendor relationships.
5. **Operational Efficiency**
   Minimize manual tracking and data entry through automation and Salesforce Flows.
6. **Role-Based Access Control**
   Secure data by defining roles such as Administrator, Pharmacist, and Store Manager with appropriate permissions.
7. **Data-Driven Insights**
   Generate interactive dashboards and reports for stock movement, supplier performance, and usage trends.

Ideation

Originality of Ideas:

Traditional inventory systems rely on spreadsheets or offline software, leading to data inconsistencies, limited automation, and human errors. This project innovatively integrates **Salesforce CRM** into the medical domain, offering a **cloud-based, automated, and scalable solution** for healthcare inventory operations. It bridges the gap between medical logistics and intelligent CRM systems.

Feasibility of Ideas:

The project is highly feasible using the **Salesforce Developer Edition**, which provides access to tools such as:

- **Custom Objects and Relationships**
- **Apex Classes and Triggers**
- **Flows, Process Builder, and Validation Rules**
- **Reports and Dashboards**
- **Role and Permission Management**

The system is cost-effective, requiring no additional hardware, and can scale easily for **multi-branch hospitals** or **pharmacy chains**.

Requirement Analysis:

Completeness of Requirements

The requirements were gathered through consultation with hospital administrators, pharmacists, and supply managers to ensure all operational needs were covered. Key functional areas include:

- **Medicine Management** — Track details like name, category, batch number, price, stock quantity, and expiry date.
- **Supplier Management** — Store supplier details, contact info, payment terms, and performance history.
- **Stock Tracking** — Log inventory transactions such as purchases, consumption, and returns.
- **Reorder and Expiry Alerts** — Automatically notify staff for restocking or removal of expired items.
- **Purchase Order Management** — Manage purchase orders with automatic total calculations and supplier linkage.
- **Reports and Dashboards** — Provide visual insights into inventory levels, supplier performance, and usage trends.

Project Design:

Design Completeness

The Salesforce configuration includes the following components:

Custom Objects

1. **Medicine (Product)** – Holds details about each medicine.
2. **Supplier** – Manages supplier contact and transaction information.
3. **Purchase Order** – Tracks orders placed with suppliers.
4. **Order Item** – Line items linking medicines to purchase orders.
5. **Inventory Transaction** – Logs stock changes (inflow/outflow).

Relationships

- **Supplier → Purchase Order:** Lookup Relationship
- **Purchase Order → Order Item:** Master-Detail Relationship
- **Medicine → Order Item:** Lookup Relationship

Page Layouts & Tabs

- Custom layouts for each object ensure clarity and ease of use. Tabs organized within a **Lightning App** for unified access.

Lightning App Components

- Home Page with KPIs and charts.
- Tabs for Medicines, Suppliers, Orders, and Reports. Quick Action Buttons for adding new records.

Innovation and Design

- **Lightning App Builder** was used to create a visually appealing and interactive interface.
- **Flows and Validation Rules** ensure data accuracy.
- **Automation Tools** handle reordering, expiry alerts, and total calculations.
- **Dashboards** visualize stock levels, supplier performance, and inventory trends in real-time.

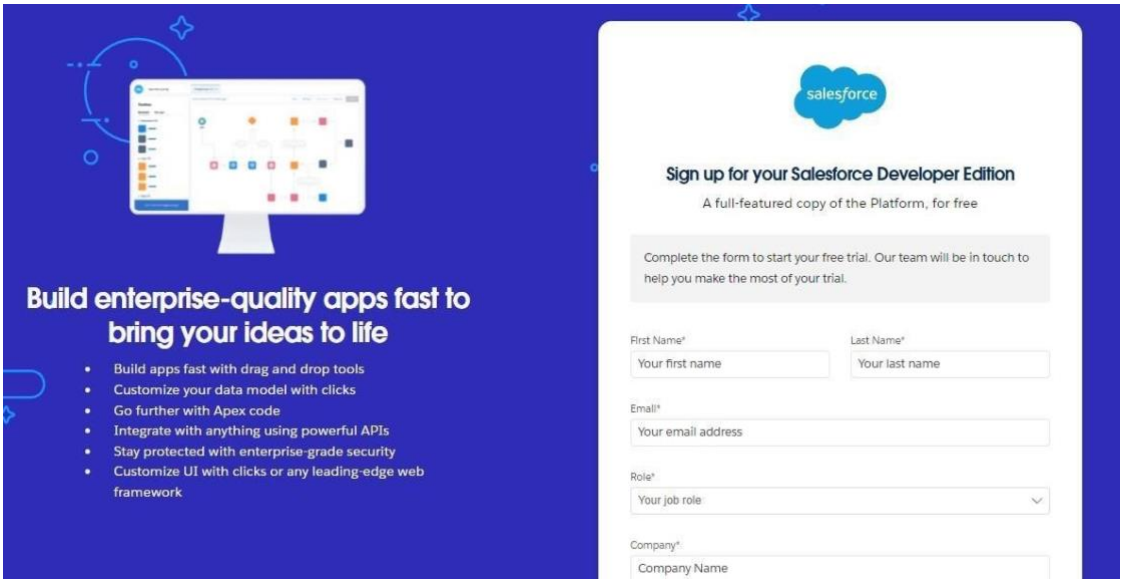
Salesforce Development – Backend Milestone 1: Salesforce Developer Account Creation

Milesstone-1

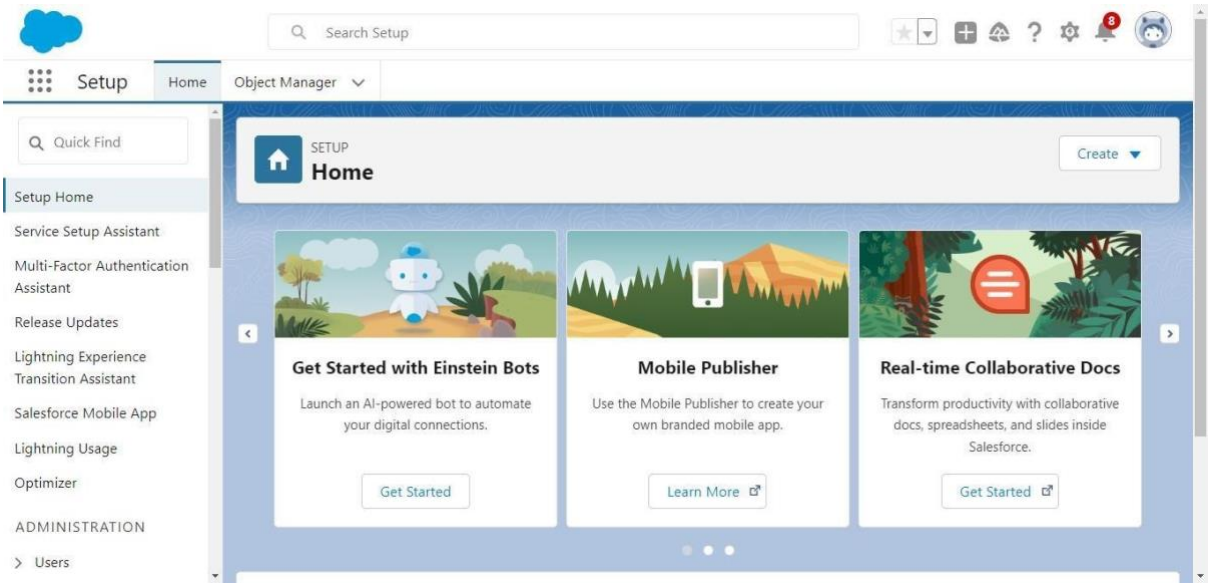
Activity 1: Creating Developer Account:

Creating a developer org in salesforce.

1. Go to https://developer.salesforce.com/signup
2. On the sign up form, enter the following details.
3. Role : Developer
4. Company : College Name
5. County : India
6. Postal Code : pin code
7. Username : should be a combination of your name and company This need not be an actual email id, you can give anything in the format username@organization.com Click on sign me up after filling these.

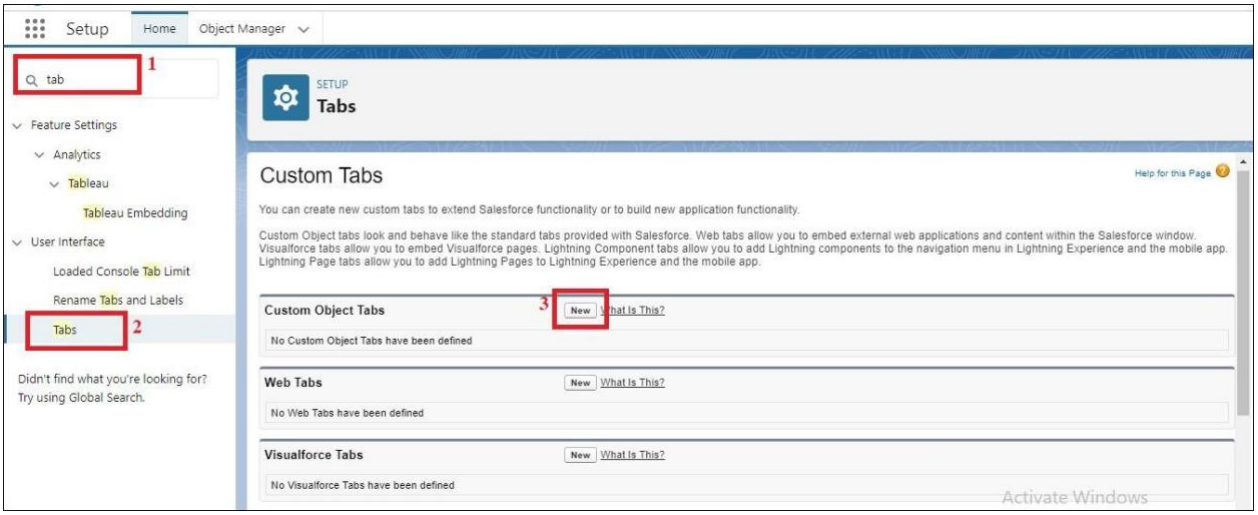


Milesstone-2

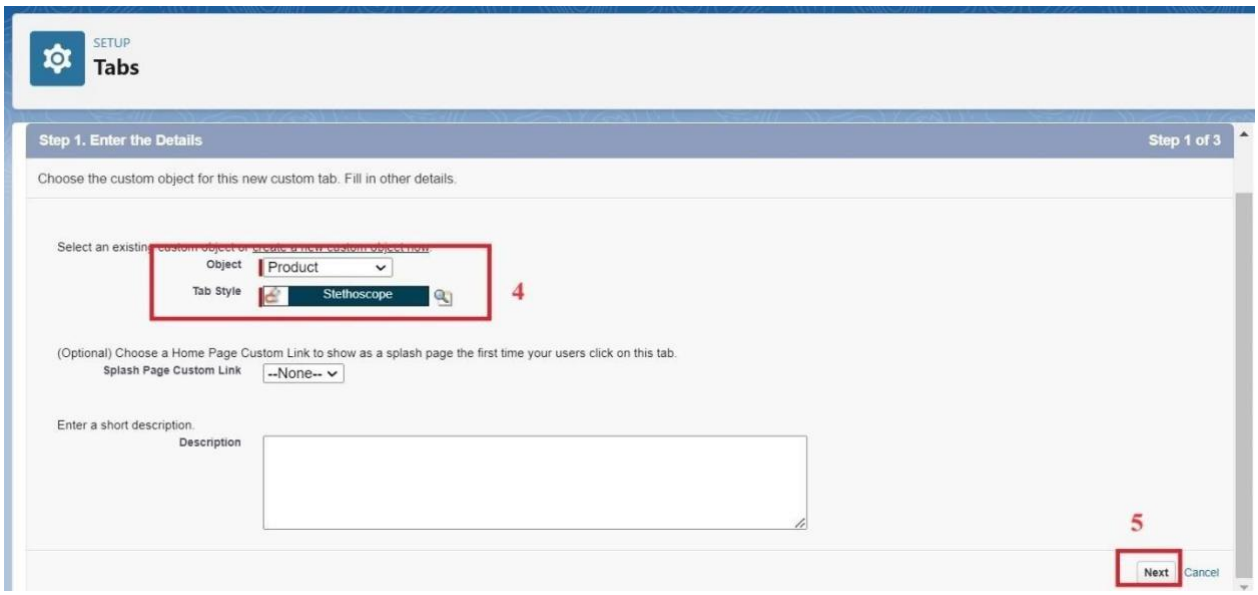


1. Go to the setup page >> type Tabs in Quick Find bar
2. Click on tabs
3. Click on New (under custom object tab).
4. Select Object(Product) >> Select the tab style

5. Click on Next >> (Add to profiles page) keep it as default >> Click on Next (Add to Custom App) uncheck the include tab
.
6. Make sure that the Append tab to user's existing personal customizations is checked.
7. Click save



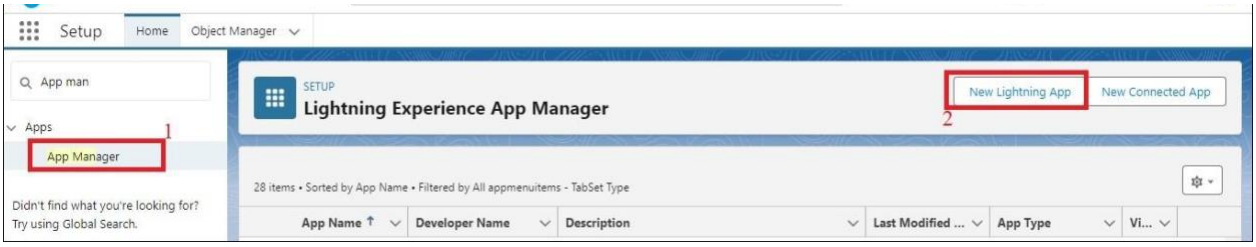
Milesstone-3



Milestone 4- The Lightning App:

Activity 1: Create a Lightning App for Medical Inventory Management

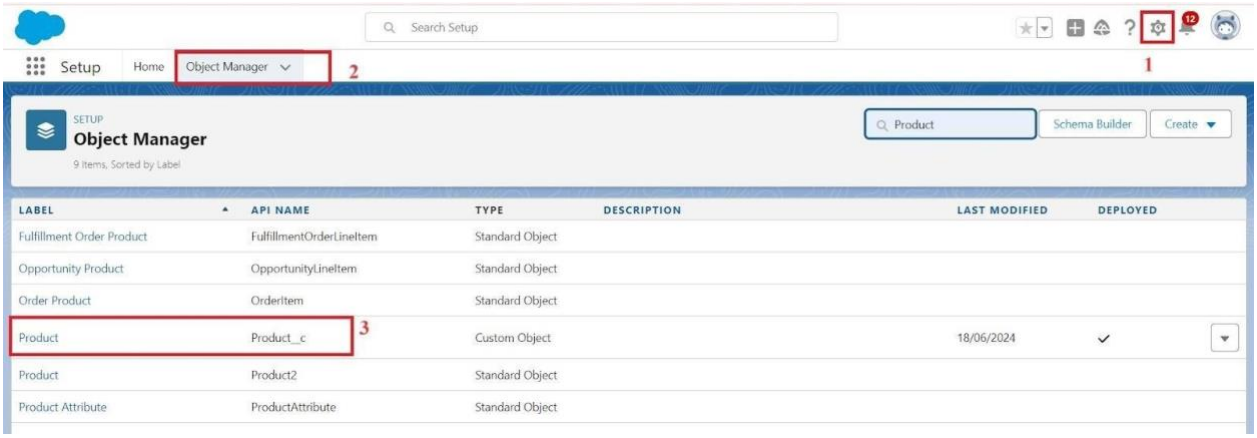
1. From Setup, enter App Manager in the Quick Find and select App Manager.
2. Click New Lightning App.
3. Enter Medical Inventory Management as the App Name >> Click on upload image and add an image related to Medical Inventory then click next
4. Under App Options, leave the default selections and click next.
5. Under Utility Items, leave as is and click Next.
6. From Available Items, select Products, Purchase Orders, Order Items, Inventory Transactions, Suppliers, Reports, and Dashboards and move them to Selected Item and Click Next.
7. From Available Profiles, select System Administrator and move it to Selected Profiles.
8. Click Save & Finish.



Milestone 5- Fields:

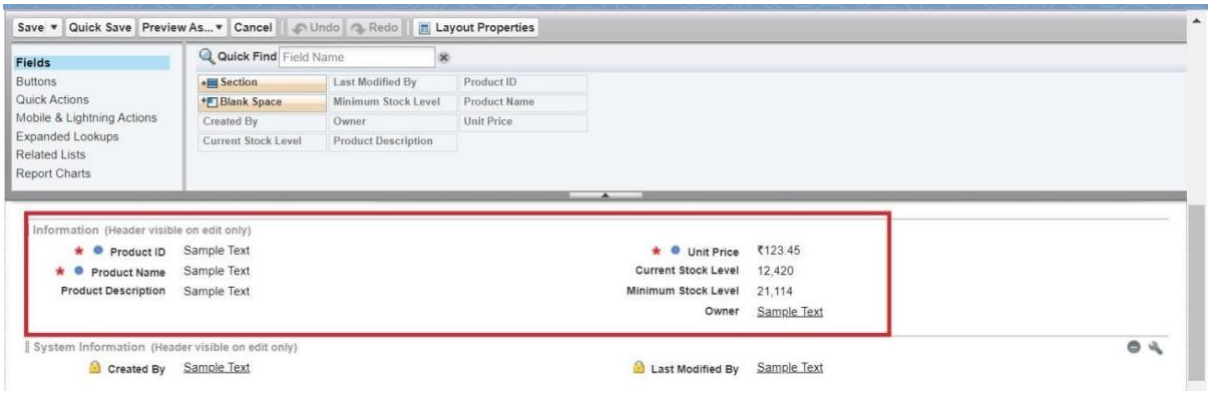
To create fields in an object:

1. Click the gear icon and select Setup. This launches Setup in a new tab.
2. Click the Object Manager tab next to Home.
3. Select Product custom object.
4. Select Fields & Relationships from the left navigation
5. Click on New
6. Select Text field, click Next
7. Enter Field Label as “Product Name” and Length 255.
8. Select Required Field.
9. Click Next, Next, then Save & New.



Milestone 6 -Editing of Page Layouts:

Activity 1: To edit a Page Layout in Product Object:



Milestone 7 - Compact Layouts:

Activity 1: To create a Compact Layout to a Product Object

Skill Tags:

1. Go to setup >> click on Object Manager >> type object name(Product) in quick find box >> click on the Product object
2. Click on Compact Layouts in the sidebar .
3. Click on New.
4. Enter the Label as “Product Compact Layout”.
5. Select the Compact Layout Fields : Select Product name, Unit Price, Current Stock Level.
6. Click Save.
7. Click Compact Layout Assignment.

8. Click Edit Assignment.
9. Choose "Product Compact Layout" from the dropdown.
10. Click Save.

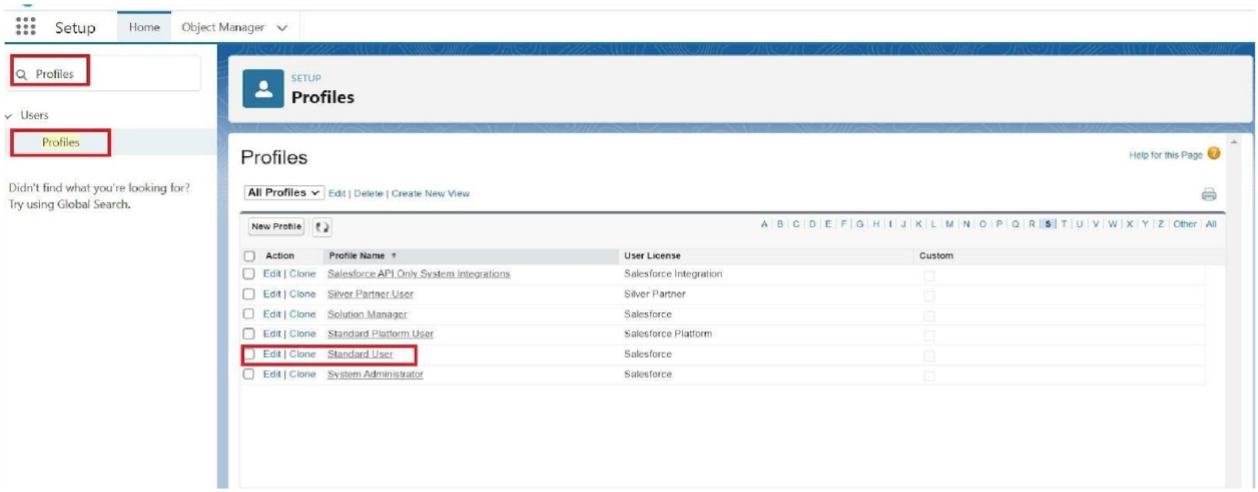
Milestone 9 – Profiles:

Activity 1: To create an Inventory Manager Profile:

Skill Tags:

1. Go to setup >> type profiles in quick find box >> click on profiles >> clone the desired profile (Standard User) >> enter profile name (Inventory Manager) >> Save.

2.



Milestone 10

Activity 1 : Create a Purchasing Manager Role.

Duration: 0.05 Hrs

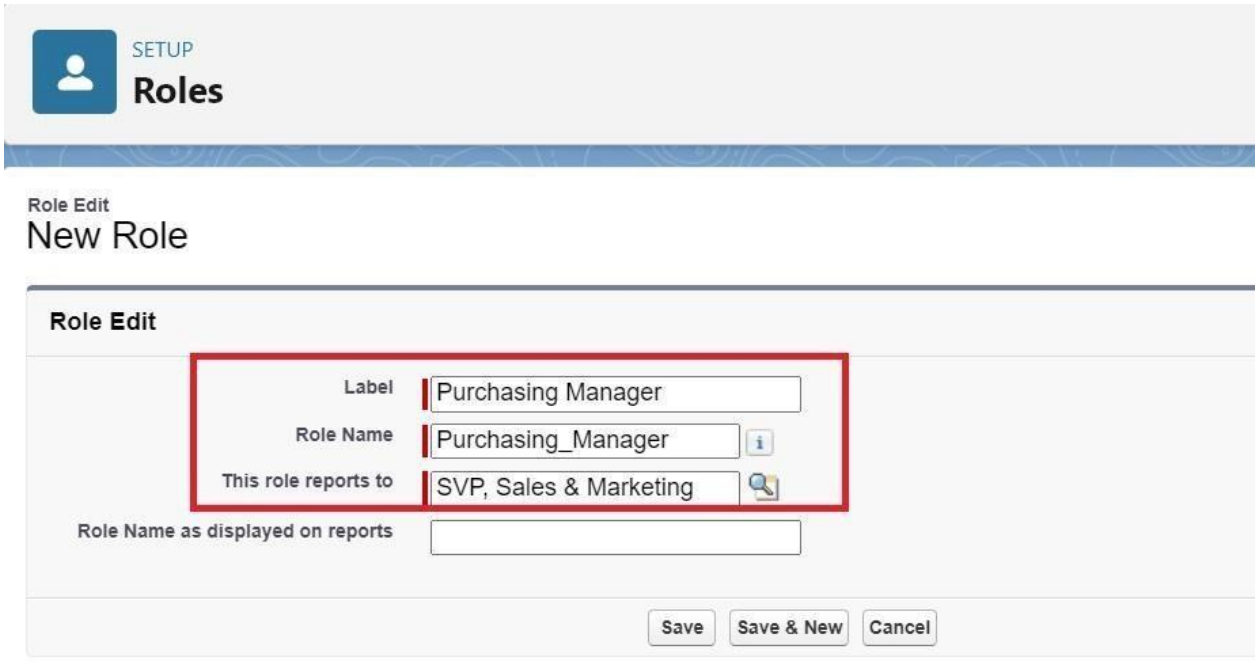
Skill Tags:

1. Go to quick find >> Search for Roles >> click on Set Up Roles.

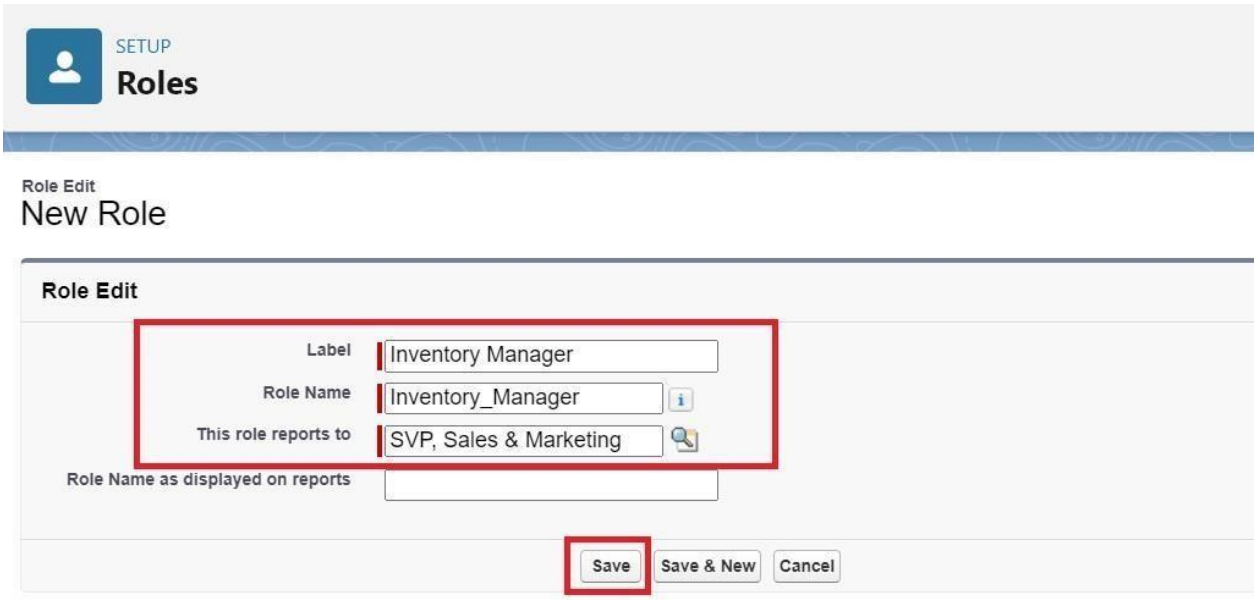
Click on Expand All and click on add role under SVP, Sales & Marketing role.

Give Label as “Purchasing Manager” and Role name gets auto populated. Then click on Save.

3.



Milestone 12 - Permission Sets:



Milestone 14 - Triggers

Activity 1 : Create a Trigger to Calculate total amount on Order Item.

Skill Tags:

Step 1 : Login to Salesforce:
Log in to your Salesforce account with administrative privileges.

Step 2:
i)Navigate to Setup: Once logged in, click on the gear icon ?? (Setup) located at the top-right corner of the page. This will open the Setup menu.

ii)Click on Developer Console: Click on the "Developer Console" option from the Setup menu. This will open the Developer Console in a new browser tab or window.

Step 3:
i) In the Developer Console window, go to the top menu and click on "File". ii)Select New: From the dropdown menu under "File", select "New". iii)Choose Apex Trigger: This will open a new Apex Trigger editor tab. 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);
}
```

Step 4:
i) In the Developer Console window, go to the top menu and click on "File".

ii)Select New: From the dropdown menu under "File", select "New".
iii)Choose Apex Class: Name it as CalculateTotalAmountHandler

```
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;
        }
    }
  }

}
```

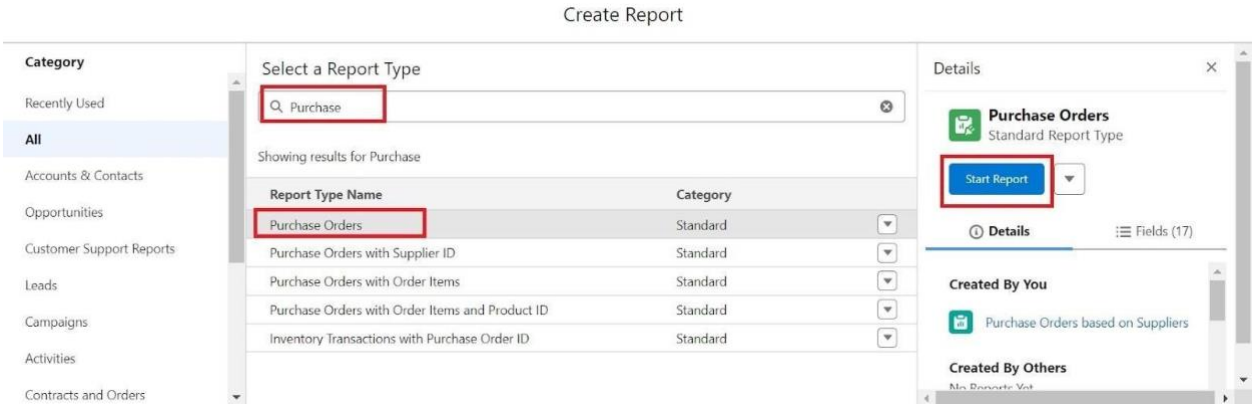
Milestone 15 – Reports:

Activity 1: Create a Purchase Orders based on Suppliers(Summary) Report

Skill Tags:

1. Click App Launcher

2. Select Medical Inventory Management App
3. Click on Reports tab
4. Click on New Report.
5. Click the report type as Purchase Orders Click Start report.

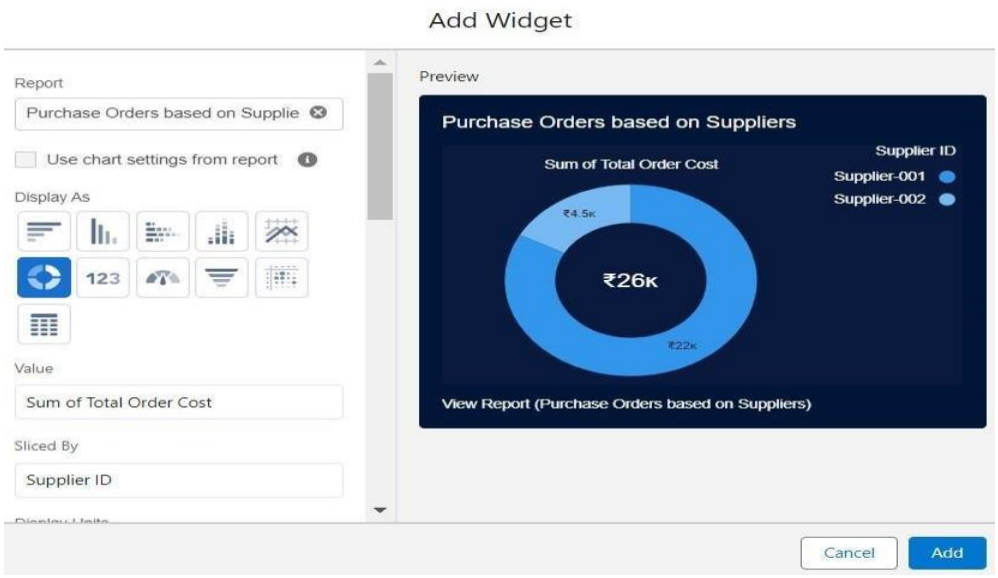


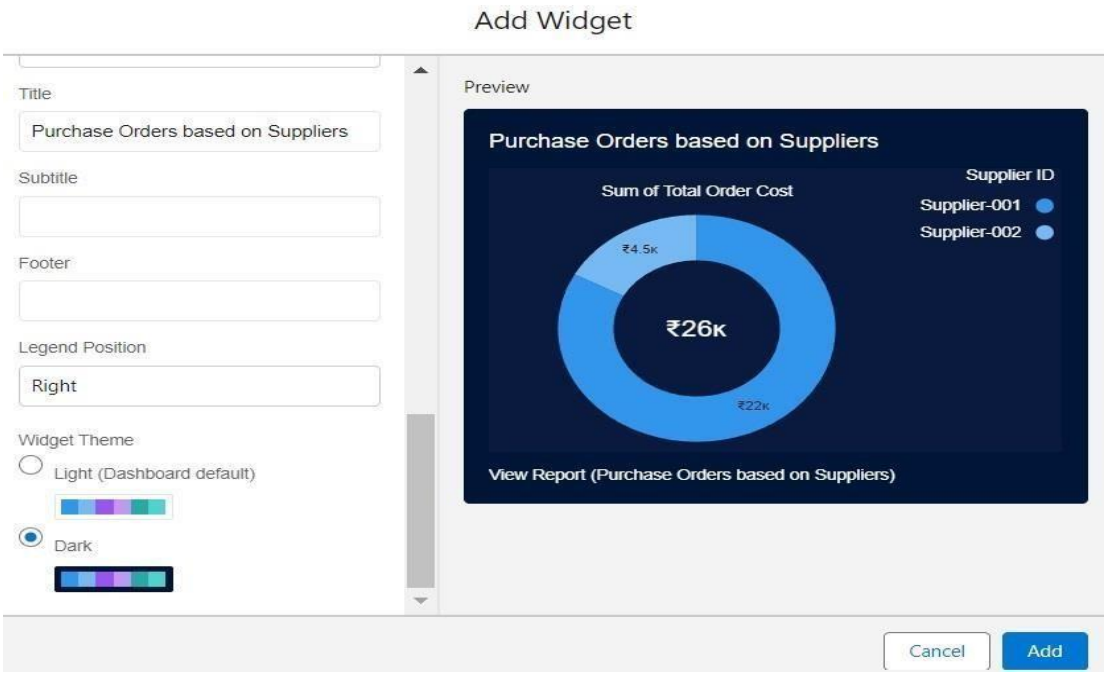
Milestone 16 – Dashboards:

Activity 1: - Create Dashboard

Skill Tags:

1. Click on the Dashboards tab from the Medical Inventory Management application.
2. Click on the new dashboard.
3. Give name - Medical Inventory DashBoard
4. Click create
5. Click on +widget
6. Select the Purchase Orders based on Suppliers Report
7. For the data visualization select any of the charts, tables etc. as per your choice/requirement
8. Click add.

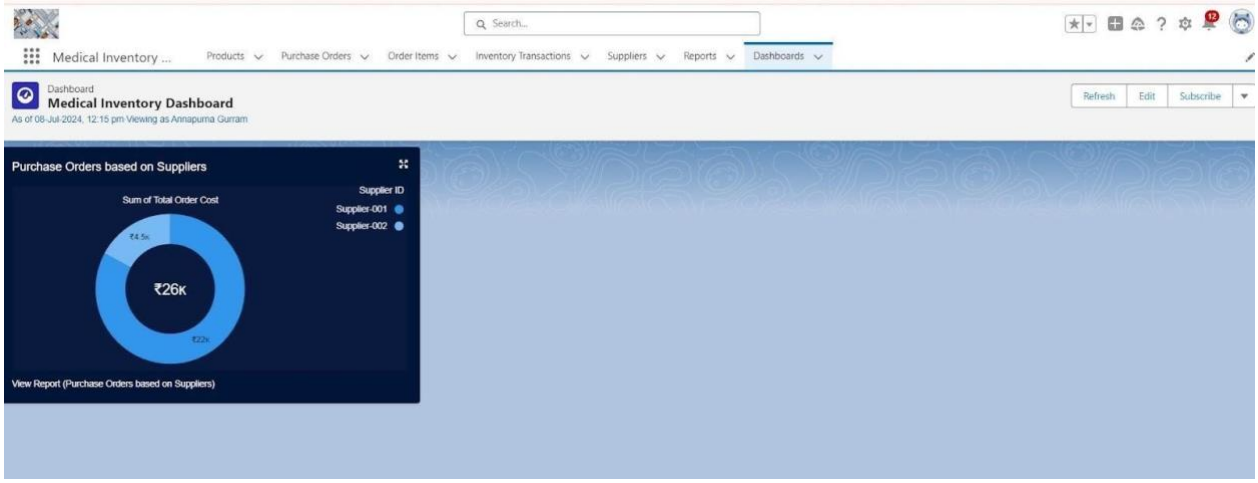

Add Widget
Report
Purchase Orders based on Supplie
Use chart settings from report
Display As
123
Value
Sum of Total Order Cost
Sliced By
Supplier ID
Preview
Purchase Orders based on Suppliers
Sum of Total Order Cost
Supplier ID
Supplier-001
Supplier-002
₹4.5K
₹26K
₹22K
View Report (Purchase Orders based on Suppliers)
Cancel
Add


Add Widget
Title
Purchase Orders based on Suppliers
Subtitle
Footer
Legend Position
Right
Widget Theme
Light (Dashboard default)
Dark
Preview
Purchase Orders based on Suppliers
Sum of Total Order Cost
Supplier ID
Supplier-001
Supplier-002
₹4.5K
₹26K
₹22K
View Report (Purchase Orders based on Suppliers)
Cancel
Add

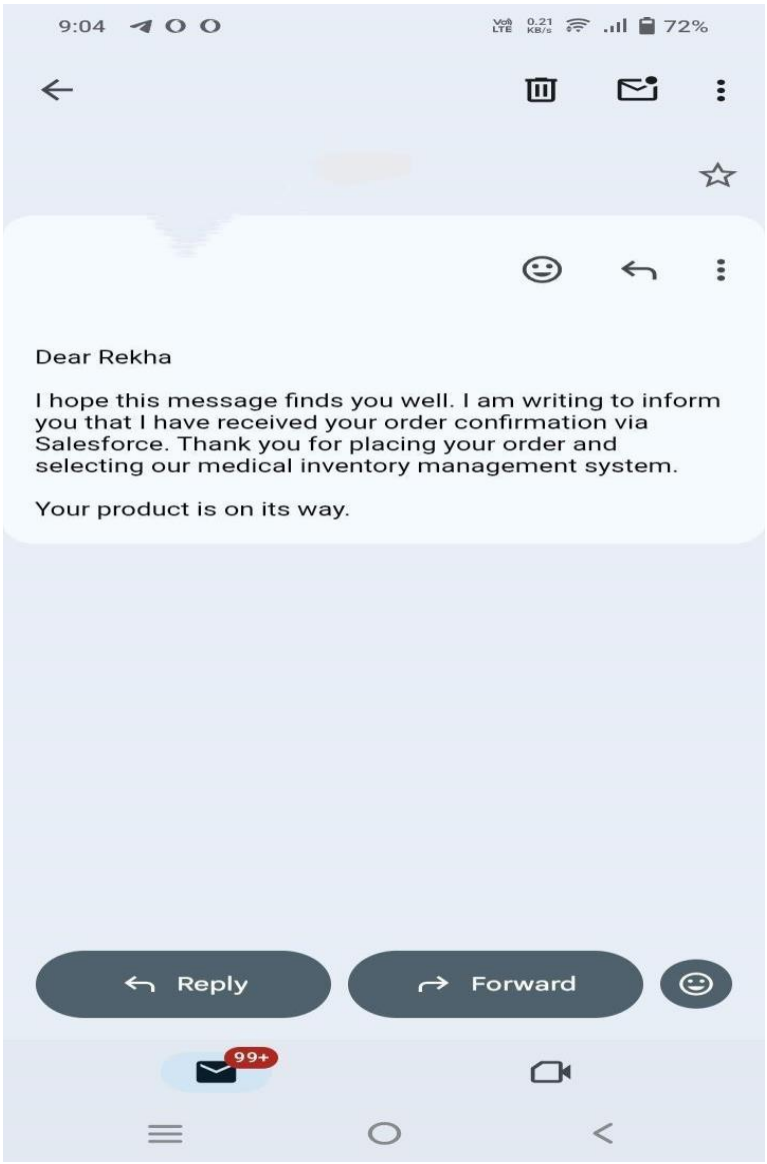
Activity 2: View Dashboard:

Skill Tags:

1. Click on App Launcher on the left side of the screen.
2. Search Medical Inventory Management & click on it.
3. Click on Dashboard Tab.
4. Click on Medical Inventory DashBoard see graph view of records



You will find notification like this and you will get an email check.:



User Experience Considerations

- **Intuitive Navigation:** Simple tabs and clear icons for ease of access.
- **Responsive Layouts:** Optimized for both desktop and mobile Salesforce apps.
- **Role-Specific Interfaces:** Admins manage configurations; pharmacists handle daily stock operations.

- **Dashboards:** Offer color-coded insights for quick decisionmaking.

Project Development

Code Quality

All Apex classes follow Salesforce **best practices**:

- Modular and reusable methods.
- Bulkified triggers for handling multiple records.
- Proper exception handling and logging mechanisms.
- Descriptive comments for readability.

Apex Trigger and Handler

Trigger: `CalculateTotalAmountTrigger`

HandlerClass: `CalculateTotalAmountHandler`

Functionality:

Automatically calculates and updates the total amount in a Purchase Order when related Order Items are inserted, updated, or deleted.

Example Logic Flow:

1. Trigger detects changes in Order Items.
2. Handler retrieves related Purchase Orders.
3. Handler recalculates total amounts.
4. Updated totals are saved automatically.

Adherence to Timelines:

Project phases were completed systematically:

Phase	Activity	Duration	Status
1	Requirement Gathering	1 Week	Completed
2	Object and Field Setup	2 Weeks	Completed
3	Automation and Triggers	2 Weeks	Completed
4	UI Design and Dashboards	1 Week	Completed
5	Testing and Deployment	1 Week	Completed

Testing and Debugging:

Comprehensive testing validated all functionalities:

- Flows and validation rules worked correctly.
- Email alerts were functional (with minor spam filter issues).
- Data integrity and performance were verified.
- Security configurations (roles, profiles, permission sets) were properly enforced.

Use of Best Practices:

- Adopted **Trigger-Handler Design Pattern** in Apex.
- Implemented **Flows and Validation Rules** for process automation.
- Applied **Role-Based Security** for controlled access.
- Designed **Dynamic Dashboards** for visual analytics.
- Followed **Lightning UI Standards** for a modern look and feel.
- Ensured **Scalability** for future features and integrations.

Conclusion:

The **Medical Inventory Management System** effectively showcases how Salesforce CRM can be leveraged to automate healthcare inventory operations.

Key achievements include:

- Automated medicine tracking and expiry monitoring.
- Supplier and purchase order management.
- Real-time stock visibility and analytics.
- Reduced manual workload and data errors.

By combining Salesforce automation tools with robust data design, this system ensures **efficiency, accuracy, and transparency** in medical inventory workflows.

Future Enhancements:

- **Mobile App Integration:** For on-the-go stock management.
- **ERP Integration:** To connect with accounting and billing systems.
- **AI-Based Forecasting:** Predict medicine demand and optimize purchases.

References :

- https://developer.salesforce.com/signup
- https://trailhead.salesforce.com
- https://help.salesforce.com
- https://www.salesforceben.com
- https://admin.salesforce.com