Medical Inventory Management
Naan Mudhalvan

Team Members :

RAVI M	911222104034
BALACHANDER S	911222104007
BALAMURUGAN S	911222104008
KARTHIKEYAN K	911222104018

Nan Mudhalvan Guide :

Mrs.S.NITHYA,M.E.,

Medical Inventory Management

Project Overview:

The Medical Inventory Management System is a comprehensive Salesforce application designed to streamline and manage various operational aspects of the medical inventory. It can efficiently maintain supplier details, manage purchase orders, track product details and transactions, and monitor expiry dates of products, thereby improving operational efficiency, data accuracy, and reporting capabilities.

This project is a comprehensive Salesforce application to streamline and manage various operational aspects of medical inventory. The system aims to efficiently maintain supplier details, manage purchase orders, track product details and transactions, and monitor the expiry dates of products. Maintain detailed records of suppliers, including contact information. Catalog product information, including descriptions, stock levels. Monitor and track product expiry dates to avoid using expired items. Comprehensive reports to track supplier performance, and purchase orders.

Introduction:-

The Medical Inventory Management System (MIMS) was conceptualized to address the critical challenges in managing medicines and medical equipment in healthcare institutions. Manual tracking leads to inefficiencies, wastage, and delays, especially when managing large inventories. The system aims to efficiently maintain supplier details, manage purchase orders, track product details and transactions, and monitor the expiry dates of products. Maintain detailed records of suppliers, including contact information. Catalog product information, including descriptions, stock levels.

Problem Statement :-

Hospitals face difficulties such as overstocking, expiry losses, and missing stock information. These issues directly affect patient care quality and financial performance.

Objectives

- To automate the inventory management process.
- To enable real-time tracking and expiry alerts.
- To ensure accurate purchase order management.
- To enhance accountability and decision-making.

Objectives:

Efficient Inventory Control

To streamline and automate the management of medical inventory, ensuring accurate tracking of stock levels and minimizing manual errors.

Supplier and Purchase Order Management

To maintain detailed supplier records and manage purchase orders effectively for timely procurement and supplier performance tracking.

Product Tracking and Expiry Monitoring

To monitor product details, including batch numbers and expiry dates, to prevent the use of expired or low-quality medical items.

Data Accuracy and Centralization

To ensure all supplier, product, and transaction data is accurately recorded and stored within Salesforce for easy access and reliability.

Comprehensive Reporting and Decision Support

To provide real-time reports and dashboards for analyzing inventory trends, supplier performance, and purchase efficiency, supporting informed decision-making.

Detailed Steps to Solution Design:-

Ideation Phase:-

- The idea behind the Medical Inventory Management System (MIMS) emerged from the increasing need for efficient management of medicines, medical equipment, and supplies in hospitals and pharmacies.

- In many healthcare institutions, manual tracking of stock levels leads to delays, wastage, and inaccurate records.

- The goal of this project is to automate and digitize inventory control, enabling real-time tracking, faster purchase decisions, and reduced human error.

Step 1: Developer Account Setup

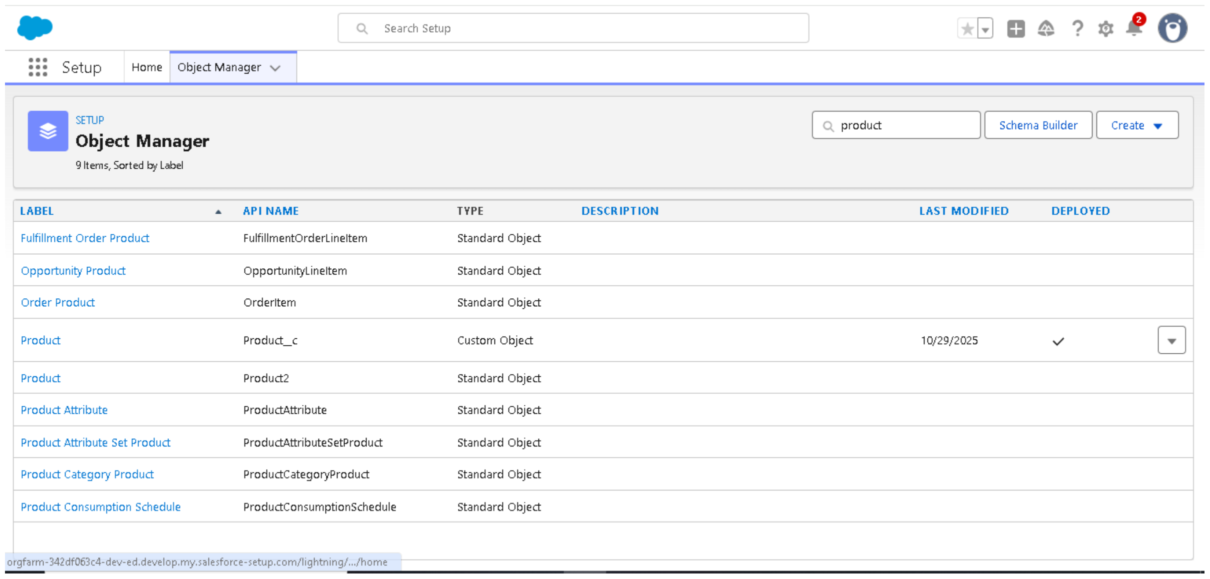
- Registered for a Salesforce Developer account to create a dedicated environment for development and testing.

- Verified the account to unlock full access to Salesforce features, ensuring a smooth setup process.

Step 2: Custom Object Creation

- Used **Salesforce Object Manager** to create custom objects for **Product**, **Purchase Order**, **Order Item**, **Inventory Transaction**, and **Supplier** to manage various aspects of medical inventory.

- Configured fields such as **text**, **numbers**, **dates**, and **relationships** to capture essential details, ensuring each object accurately represents the operational requirements of medical inventory management.



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		10/29/2025	✓
Product	Product2	Standard Object			
Product Attribute	ProductAttribute	Standard Object			
Product Attribute Set Product	ProductAttributeSetProduct	Standard Object			
Product Category Product	ProductCategoryProduct	Standard Object			
Product Consumption Schedule	ProductConsumptionSchedule	Standard Object			

Project Planning Phase:-

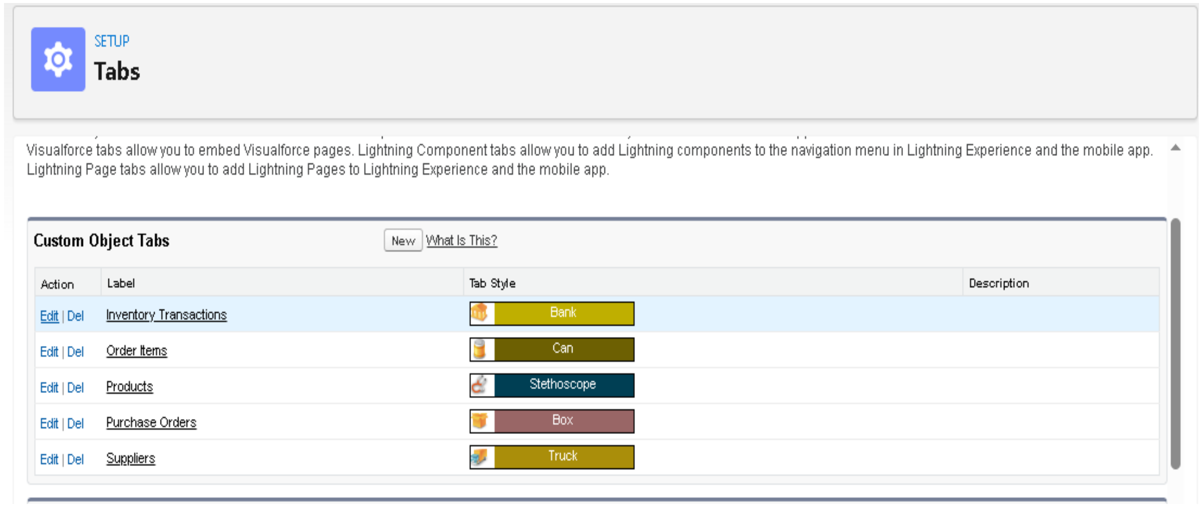
- In this phase, the project’s goals, timelines, and scope were clearly defined.

- A detailed roadmap was created to ensure systematic progress through all stages from design to testing.

Step 3: Custom Tabs for Navigation

- Created custom tabs for objects — **Product, Purchase Order, Order Item, Inventory Transaction,** and **Supplier** using **Salesforce Setup** → Tabs, enhancing navigation and data accessibility.

- Configured tab styles and visibility settings to ensure easy access and seamless management of each object within the application.



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.

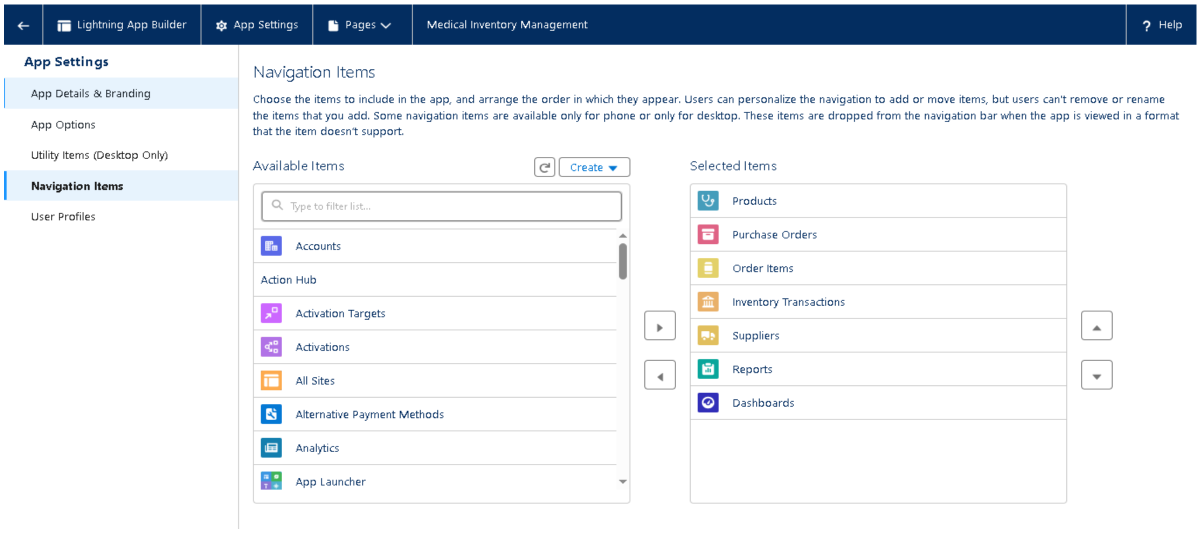
Custom Object Tabs

Action	Label	Tab Style	Description
Edit | Del	Inventory Transactions	Bank	
Edit | Del	Order Items	Can	
Edit | Del	Products	Stethoscope	
Edit | Del	Purchase Orders	Box	
Edit | Del	Suppliers	Truck	

Step 4: Lightning App Development

- Created a **Lightning App** named **Medical Inventory Management** in **Salesforce App Manager**, uploading a relevant image and configuring default app options.

- Added **Products, Purchase Orders, Order Items, Inventory Transactions, Suppliers, Reports, and Dashboards** tabs, assigning access to the **System Administrator** profile for efficient management.



App Settings

- App Details & Branding
- App Options
- Utility Items (Desktop Only)
- Navigation Items
- User Profiles

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.

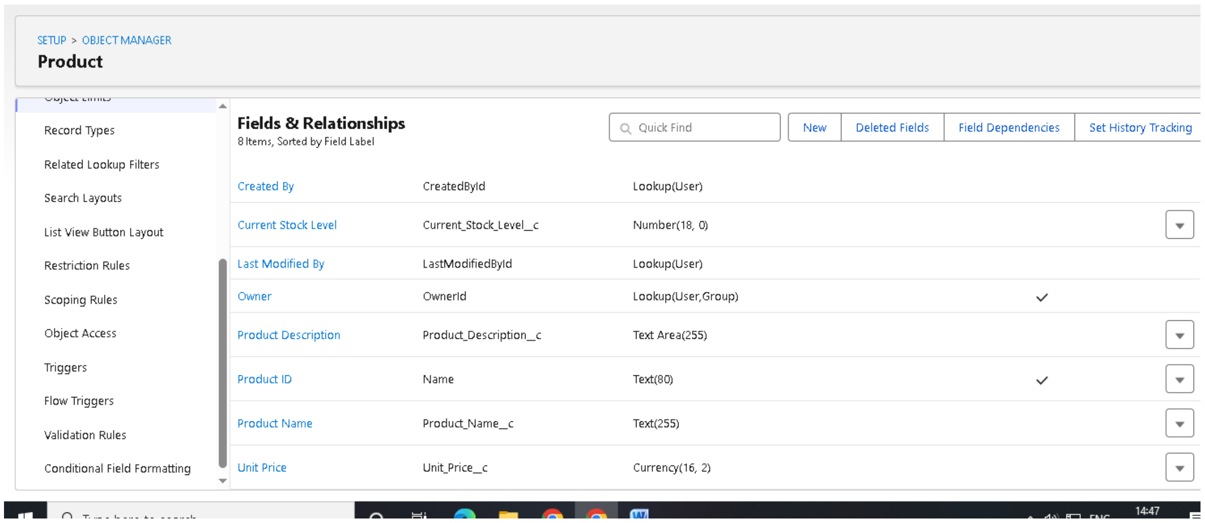
Available Items: Accounts, Action Hub, Activation Targets, Activations, All Sites, Alternative Payment Methods, Analytics, App Launcher

Selected Items: Products, Purchase Orders, Order Items, Inventory Transactions, Suppliers, Reports, Dashboards

Step 5: Field Configuration

- Created **custom fields** for objects — **Product**, **Purchase Order**, **Order Item**, **Inventory Transaction**, and **Supplier** in **Salesforce Object Manager**, defining appropriate data types and relationships.

- Configured fields such as **Text, Number, Currency, Date, Lookup, Master-Detail, Formula, Roll-Up Summary, Picklist, and Text Area** to capture detailed information for efficient management of medical inventory operations.



SETUP > OBJECT MANAGER

Product

- Record Types
- Related Lookup Filters
- Search Layouts
- List View Button Layout
- Restriction Rules
- Scoping Rules
- Object Access
- Triggers
- Flow Triggers
- Validation Rules
- Conditional Field Formatting

Fields & Relationships

8 Items, Sorted by Field Label

Created By	CreatedById	Lookup(User)	
Current Stock Level	Current_Stock_Level__c	Number(18, 0)	
Last Modified By	LastModifiedById	Lookup(User)	
Owner	OwnerId	Lookup(User,Group)	✓
Product Description	Product_Description__c	Text Area(255)	
Product ID	Name	Text(80)	✓
Product Name	Product_Name__c	Text(255)	
Unit Price	Unit_Price__c	Currency(16, 2)	

Tools & Technologies:

1. Frontend: HTML, CSS, JavaScript, Lightning App Builder (Salesforce)
2. Backend: Apex, Salesforce CRM platform
3. Database: Salesforce Objects
4. Version Control: GitHub
5. Project Management: Jira / Trello

Project Design Phase:-

The design phase involved both System Architecture Design and User Interface Design.

System Architecture:

- User Layer: Admin, Inventory Manager, Pharmacist.

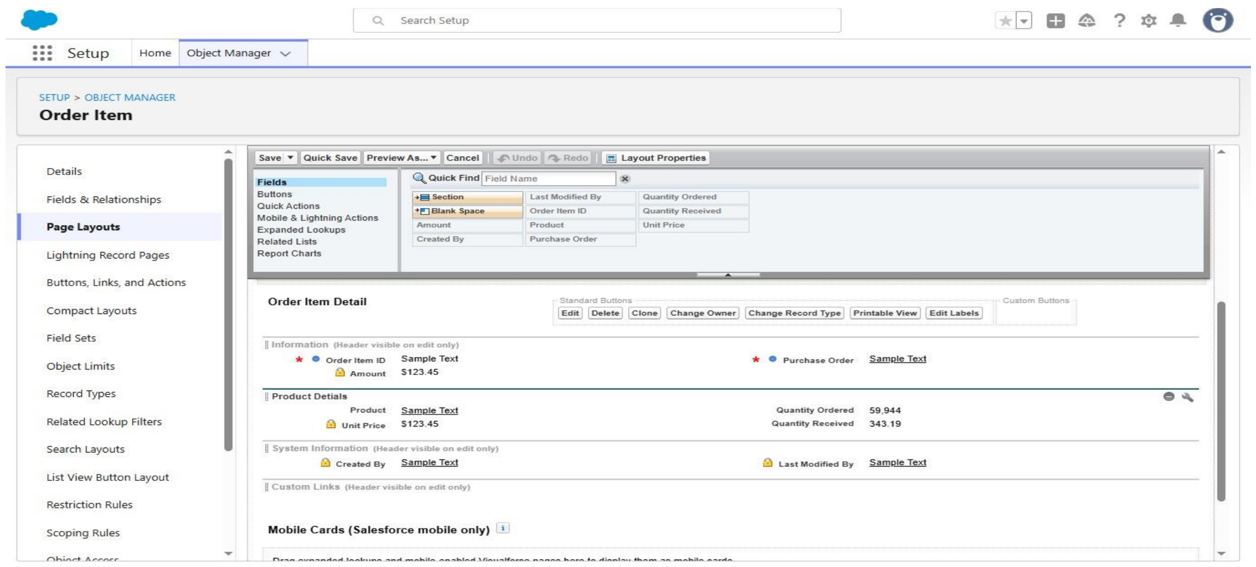
- Application Layer: Handles logic such as stock update, supplier tracking, and order approvals.

- Database Layer: Stores medicine details, supplier data, and purchase orders.

Step 6: Page Layout Configuration

- Edited **page layouts** for all custom objects — **Product, Purchase Order, Order Item, Inventory Transaction, and Supplier** to organize fields and improve usability.

- Customized layouts to ensure important details are easily accessible, enhancing user efficiency and data visibility within the Salesforce app.



Order Item Detail

Information (Header visible on edit only)

Order Item ID	Sample Text	Purchase Order	Sample Text
Amount	$123.45		

Product Detials

Product	Sample Text	Quantity Ordered	59,944
Unit Price	$123.45	Quantity Received	343.19

System Information (Header visible on edit only)

Created By	Sample Text	Last Modified By	Sample Text

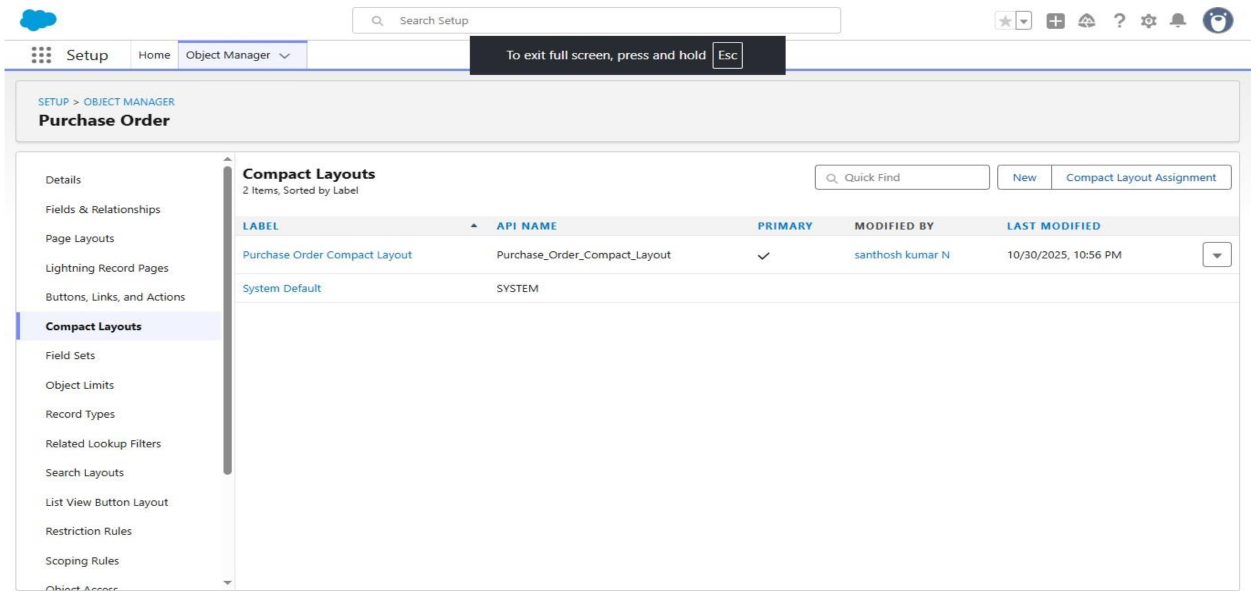
Custom Links (Header visible on edit only)

Mobile Cards (Salesforce mobile only)

Step 7: Compact Layout Configuration

- Created **Compact Layouts** for **Product** and **Purchase Order** objects in Salesforce to display key information in record highlights.

- Configured fields — **Product Name, Unit Price, Current Stock Level** for Product and **Purchase Order ID, Order Date, Total Order Cost, Supplier ID** for Purchase Order, then assigned each as the default compact layout.



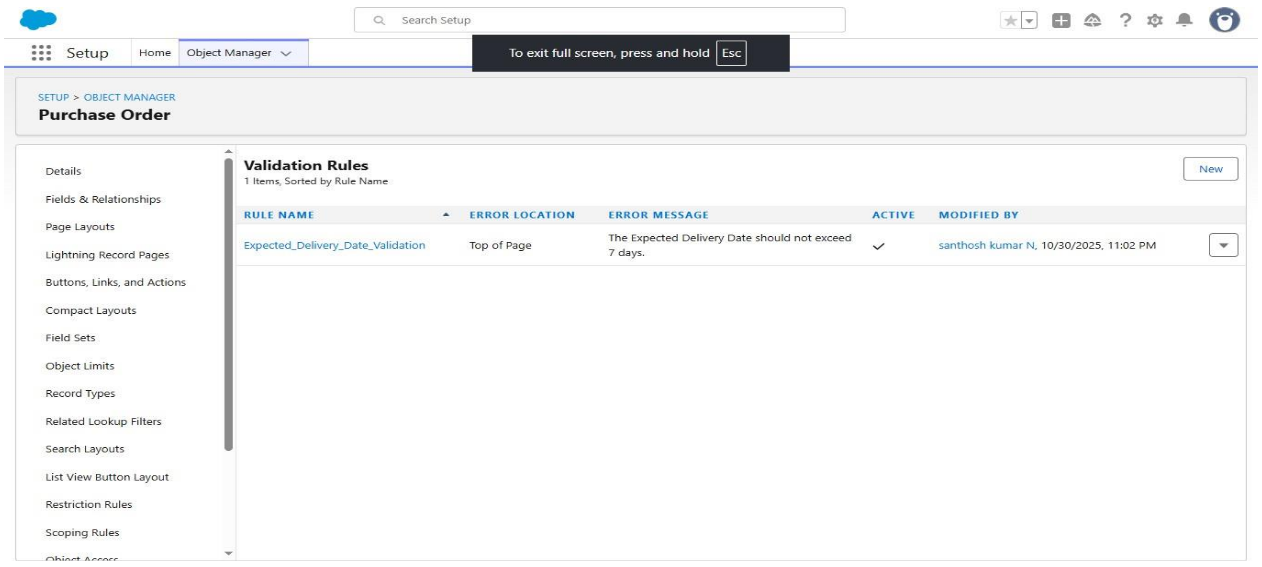
Compact Layouts
2 Items, Sorted by Label

LABEL	API NAME	PRIMARY	MODIFIED BY	LAST MODIFIED
Purchase Order Compact Layout	Purchase_Order_Compact_Layout	✓	santhosh kumar N	10/30/2025, 10:56 PM
System Default	SYSTEM			

Step 8: Validation Rules

- **Navigate and create validation rule:** Go to **Setup** → **Object Manager** → **Purchase Order** → **Validation Rules** → **New**, enter **Rule Name** as *Expected Delivery Date Validation*, select **Active**, and set **Error Condition Formula** to (Expected_Delivery_Date__c - Order_Date__c) > 7.

- **Set error details and save:** Enter **Error Message** as *"The Expected Delivery Date should not exceed 7 days."*, choose **Error Location** as *Top of Page*, and click **Save**.



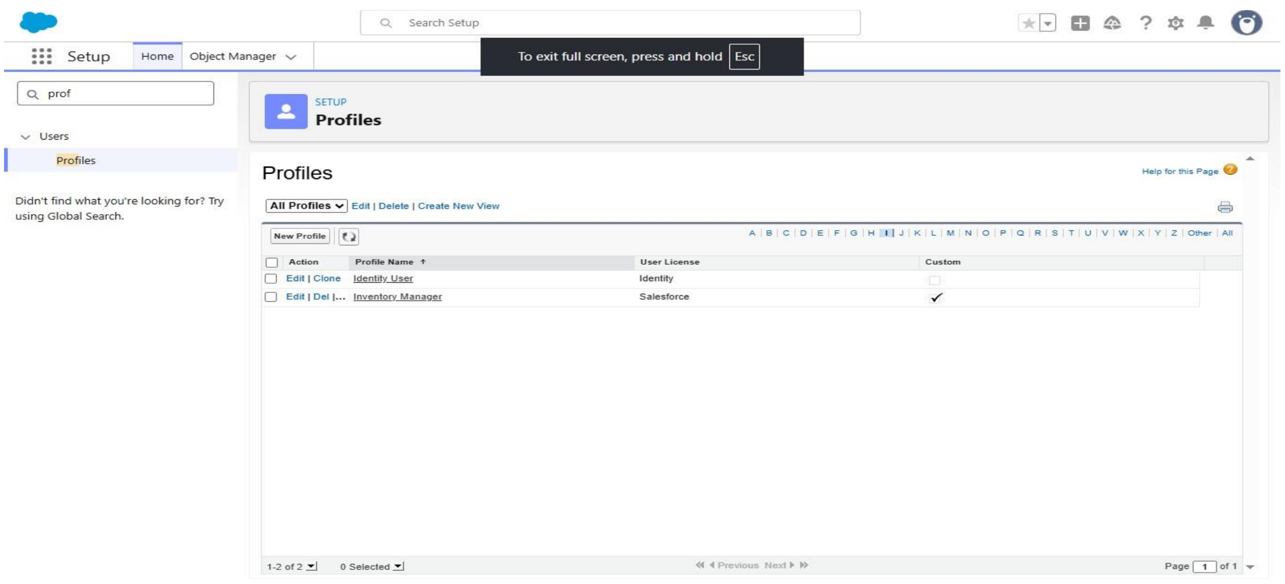
Validation Rules
1 Items, Sorted by Rule Name

RULE NAME	ERROR LOCATION	ERROR MESSAGE	ACTIVE	MODIFIED BY
Expected_Delivery_Date_Validation	Top of Page	The Expected Delivery Date should not exceed 7 days.	✓	santhosh kumar N, 10/30/2025, 11:02 PM

Step 9: Profile Creation

- Go to **Setup** → **Profiles** (search in Quick Find), clone the desired profile (Standard User), enter **Profile Name** as Inventory Manager, and click **Save**. Scroll down to **Custom Object Permissions** and assign access as per the diagram.

- Change password policies by setting **User Passwords Expire In** to Never *Expires* and **Minimum Password Length** to *8*, then click **Save**.


Search Setup
Setup
Home
Object Manager
To exit full screen, press and hold Esc
prof
Users
Profiles
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
1-2 of 2
0 Selected
Previous Next
Page 1 of 1

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

Action	Profile Name ↑	User License	Custom
Edit | Clone	Identity User	Identity	☐
Edit | Del |...	Inventory Manager	Salesforce	✓

Requirement Analysis:-

Functional Requirements:

1. The system must allow adding, editing, and deleting medicines.

2. It must track stock levels and expiry dates.

3. Purchase orders must be auto-generated when stock is below threshold.

4. The admin must manage users and permissions.

5. The system must generate reports of monthly stock usage.

Non-Functional Requirements:

1. The system should be accessible via web browsers.

2. Data should be secure and backed up periodically.

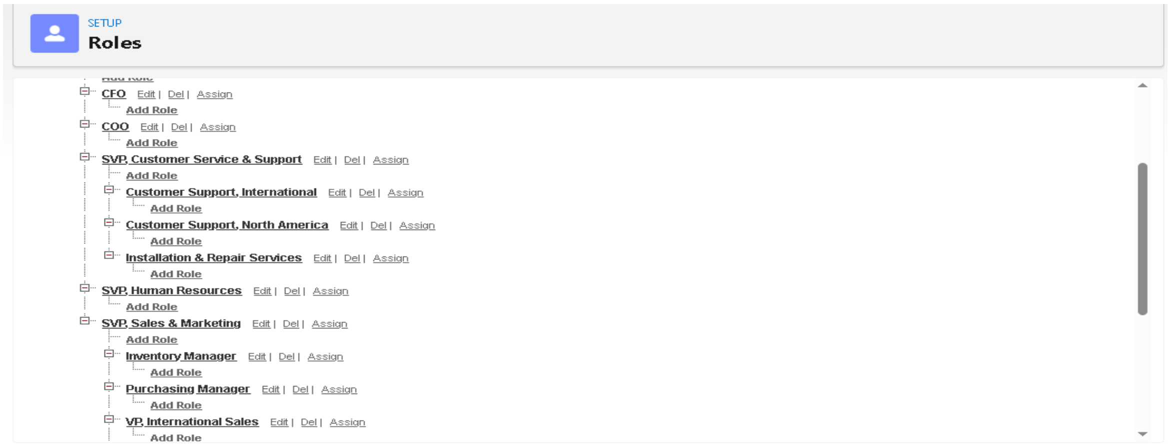
3. The interface should be responsive and easy to use.

4. Performance should support at least 1000 transactions per day.

5. System downtime should be minimal (<2%).

Step 10: Role Creation

- Create a Purchasing Manager Role.
- Create an additional Purchasing Manager Role.



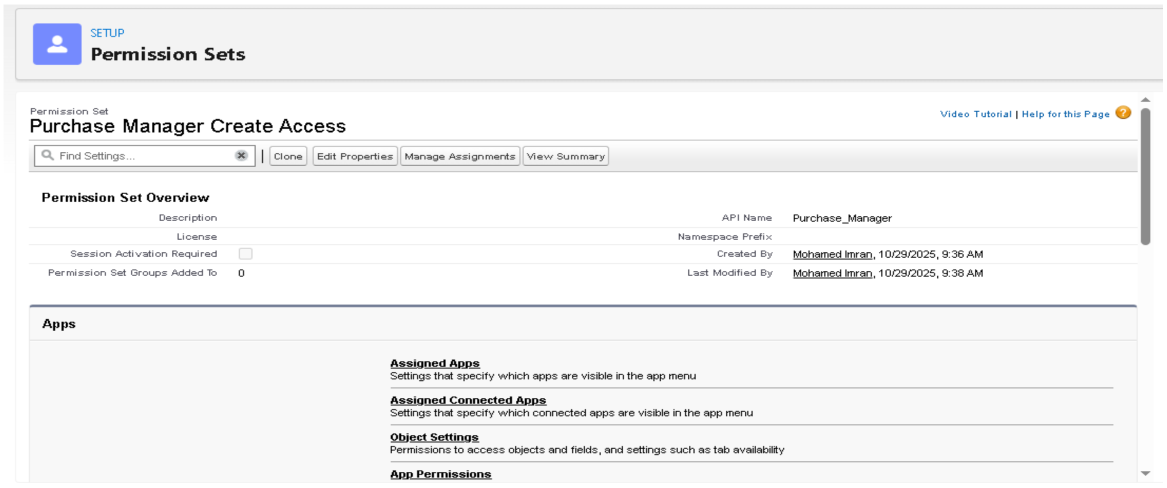
SETUP
Roles

```
CFO  Edit | Del | Assign
    Add Role
COO  Edit | Del | Assign
    Add Role
SVP, Customer Service & Support  Edit | Del | Assign
    Add Role
    Customer Support, International  Edit | Del | Assign
        Add Role
    Customer Support, North America  Edit | Del | Assign
        Add Role
    Installation & Repair Services  Edit | Del | Assign
        Add Role
SVP, Human Resources  Edit | Del | Assign
    Add Role
SVP, Sales & Marketing  Edit | Del | Assign
    Add Role
    Inventory Manager  Edit | Del | Assign
        Add Role
    Purchasing Manager  Edit | Del | Assign
        Add Role
    VP, International Sales  Edit | Del | Assign
        Add Role
```

Step 11: Permission set Creation

- Go to **Setup** → **Permission Sets** → **New**, create the Purchase Manager Create Access permission set, enable **Order Item** tab and **Read/Create** permissions, then save.

- Assign this permission set to user John PurchaseM with **No Expiration Date** via **Manage Assignments → Add Assignments**



SETUP
Permission Sets

Permission Set
Purchase Manager Create Access

Video Tutorial | Help for this Page

Find Settings... | Clone | Edit Properties | Manage Assignments | View Summary

Permission Set Overview

Description		API Name	Purchase_Manager
License		Namespace Prefix	
Session Activation Required	☐	Created By	Mohamed Imran, 10/29/2025, 9:36 AM
Permission Set Groups Added To	0	Last Modified By	Mohamed Imran, 10/29/2025, 9:38 AM

Apps

Assigned Apps
Settings that specify which apps are visible in the app menu

Assigned Connected Apps
Settings that specify which connected apps are visible in the app menu

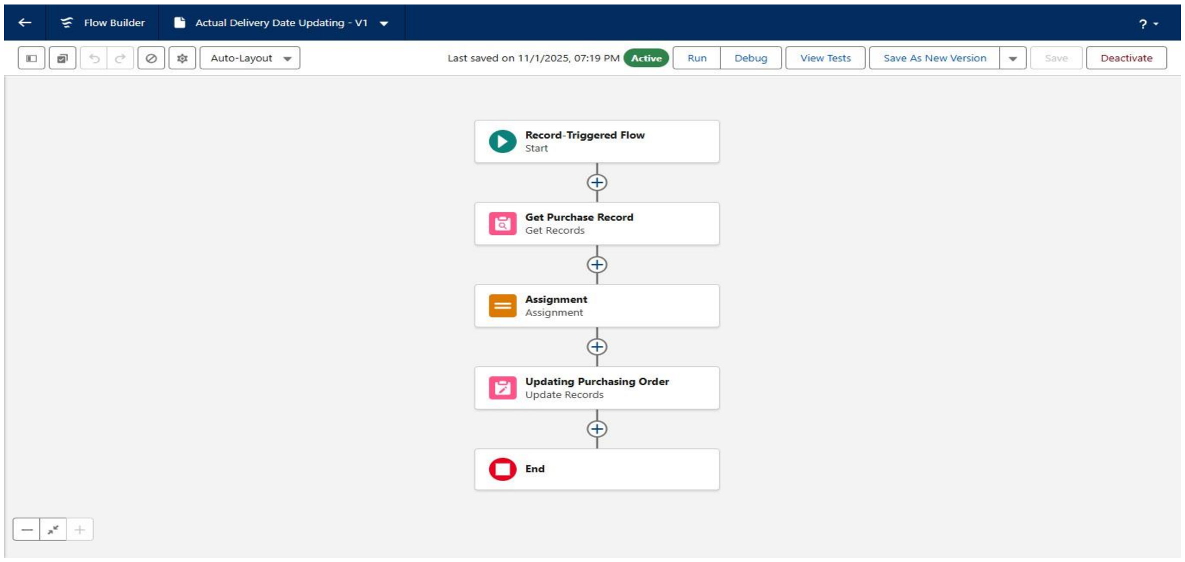
Object Settings
Permissions to access objects and fields, and settings such as tab availability

App Permissions

Step 12: Flow Creation

- Go to **Setup → Flow → New Flow → Record-Triggered Flow**, select **Purchase Order** object with trigger **A record is created or updated**, set fast field updates, then add **Get Records** to fetch the purchase order by Id retrieving the **Order_Date__c** field.

- Create a **Date variable** (ActualDeliveryDate), use an **Assignment** element to set it to Order_Date__c plus 3 days, save the flow as *Actual Delivery Date Updating*, and activate it.



Data Requirements :-

Key data fields include:

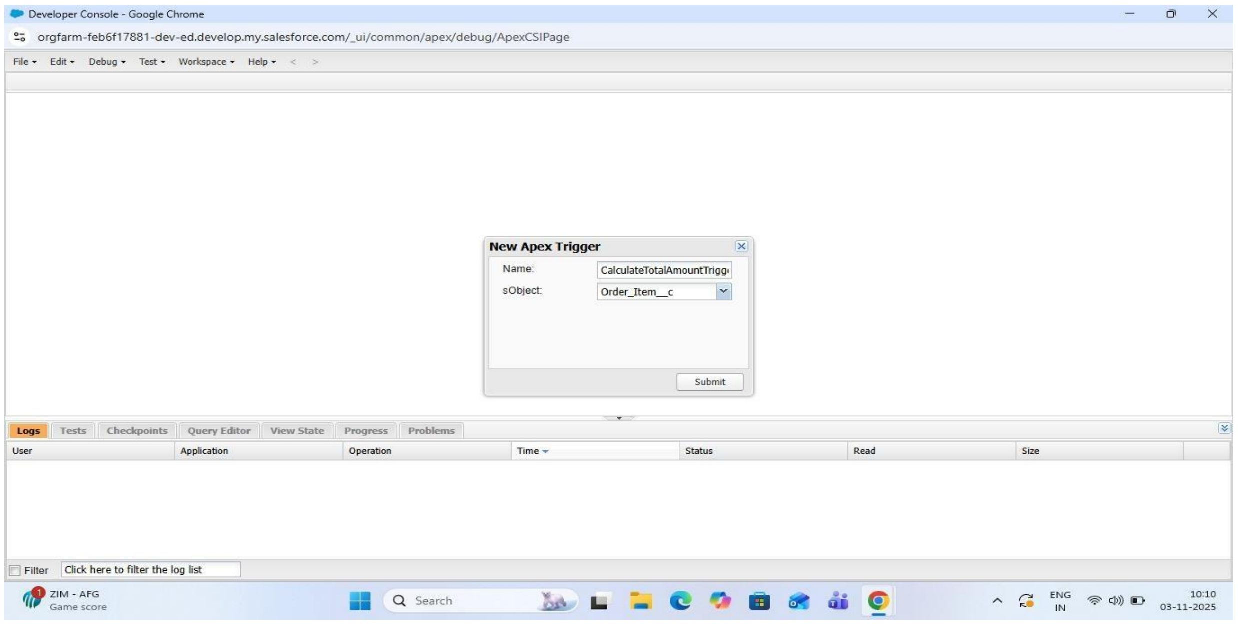
- Medicine Name, Batch Number, Expiry Date, Quantity
- Supplier Name, Contact Number, Address
- Purchase Order ID, Date, Status

Performance Testing:-

Performance testing ensures that the application performs well under expected workloads.

Step 13: Triggers

- Login to Salesforce, open **Developer Console → File → New → Apex Trigger**, create CalculateTotalAmountTrigger on Order_Item__c to call the handler class.
- Then create **Apex Class → CalculateTotalAmountHandler** to calculate and update Total_Order_cost_c on related Purchase_Order_c records based on Order_Item__c changes, and save both.


Developer Console - Google Chrome
orgfarm-feb6f17881-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help
New Apex Trigger
Submit
Logs Tests Checkpoints Query Editor View State Progress Problems
User Application Operation Time Status Read Size
Filter Click here to filter the log list

Name: CalculateTotalAmountTrigg
sObject: Order_Item__c

Program:-

1. Create an Apex Trigger: (Calculate Total Amount 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);
}
```

2. Create an Apex Trigger (Calculate Total Amount Handler)

```
 public class CalculateTotalAmountHandler {


  // Method to calculate the total amount for Purchase Orders based on related Order
Items
```

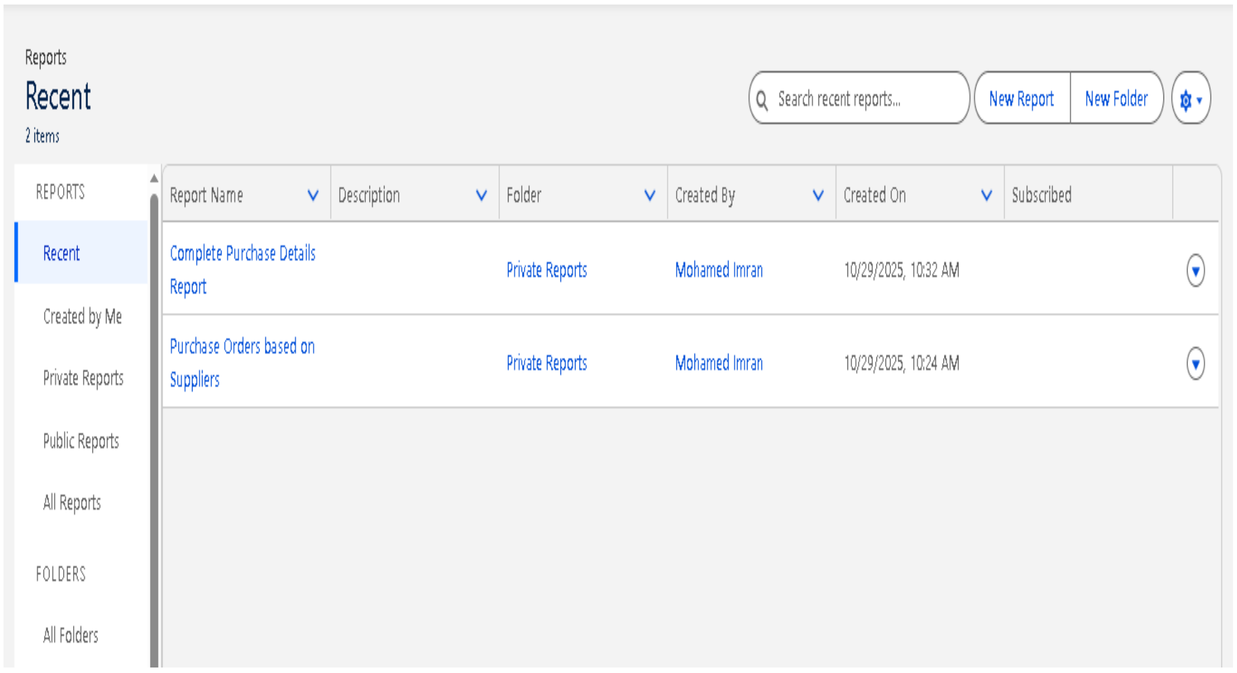
```
  public static void calculateTotal(List<Order_Item_c> newItems, List<Order_Item
c> oldItems, Boolean isInsert, Boolean isUpdate, Boolean isDelete, Boolean isUndelete)
{

    // 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;
            }
        }
    }
}
```

Step 14: Reports Creation

- Create a **Purchase Orders Summary Report** by selecting **Purchase Orders with Order Items**, applying filters, grouping by Supplier ID, Delivery Date, and Purchase Order ID, then adding Product ID, Name, Order Count, Quantity, and Amount.

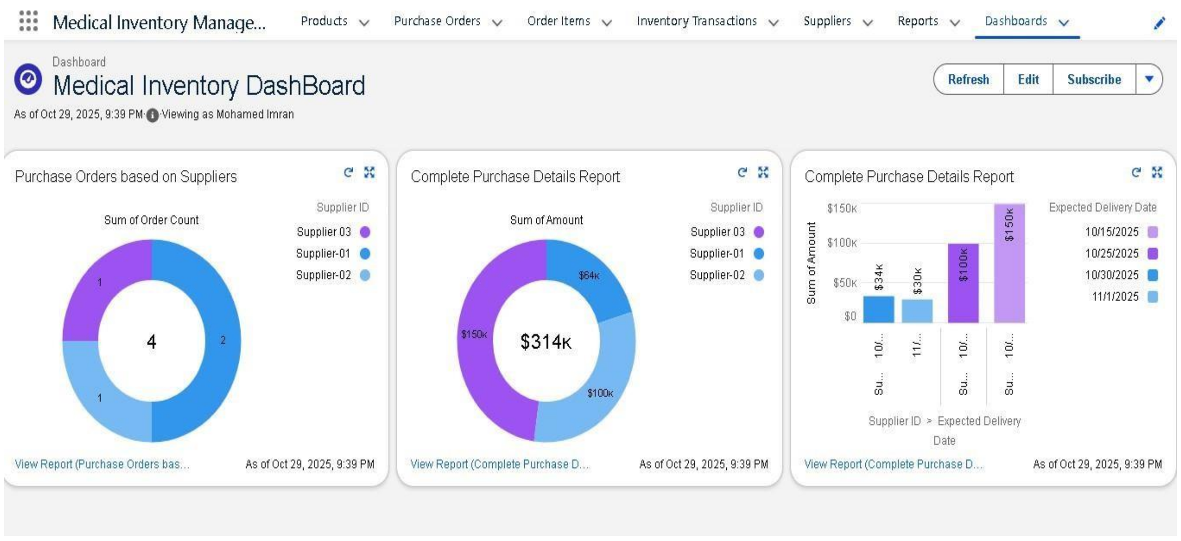
- Create a **Complete Purchase Details Report** by including all relevant columns and saving it as **Complete Purchase Details Report**.



Report Name	Description	Folder	Created By	Created On	Subscribed
Complete Purchase Details Report		Private Reports	Mohamed Imran	10/29/2025, 10:32 AM	
Purchase Orders based on Suppliers		Private Reports	Mohamed Imran	10/29/2025, 10:24 AM	

Step 15: Dashboard Creation

- Create a **Medical Inventory Dashboard** by clicking **Dashboards → New Dashboard**, naming it, and clicking **Create**.
- Add a widget using the **Purchase Orders based on Suppliers** report, choose a chart or table for visualization, then click **Add** and **Save**.



Medical Inventory Manage...
Products
Purchase Orders
Order Items
Inventory Transactions
Suppliers
Reports
Dashboards
Dashboard
Medical Inventory DashBoard
As of Oct 29, 2025, 9:39 PM · Viewing as Mohamed Imran
Refresh
Edit
Subscribe
Purchase Orders based on Suppliers
Sum of Order Count
Supplier ID
Supplier 03
Supplier-01
Supplier-02
4
1
2
1
View Report (Purchase Orders bas...
As of Oct 29, 2025, 9:39 PM
Complete Purchase Details Report
Sum of Amount
$314K
$64K
$150K
$100K
View Report (Complete Purchase D...
As of Oct 29, 2025, 9:39 PM
Complete Purchase Details Report
Sum of Amount
$150K
$100K
$50K
$0
$34K
$30K
$100K
$150K
10/...
11/...
10/...
10/...
Su...
Supplier ID > Expected Delivery Date
Expected Delivery Date
10/15/2025
10/25/2025
10/30/2025
11/1/2025
View Report (Complete Purchase D...
As of Oct 29, 2025, 9:39 PM

Testing Objectives:

- Measure system response time during high traffic.
- Verify system response time for inventory queries.
- Ensure accurate data retrieval and transaction processing.
- Check system behavior under high data load.

Key Scenarios Addressed by Salesforce in the Implementation Project:-

Automated Stock Replenishment – Automatically generates purchase orders when inventory levels fall below thresholds.

Supplier Performance Analytics – Tracks supplier reliability, delivery timelines, and quality metrics.

Real-time Inventory Visibility – Provides instant updates on stock levels across all locations.

Expiry Alerts and Notifications – Sends alerts for products nearing expiry to prevent losses.

Integrated Procurement Workflow – Streamlines the entire purchase process from order creation to receipt.

Customizable Reports & Dashboards – Allows stakeholders to generate tailored reports and visual dashboards for decision-making.

Advantages:-

1. Accurate Inventory Tracking

The system allows real-time monitoring of stock levels, ensuring that medicines and medical supplies are always available when needed.

2. Reduction of Human Error

Automated calculations and record management eliminate the risk of manual mistakes in data entry, stock counting, and expiry tracking.

3. Efficient Resource Management

Helps hospitals and pharmacies plan procurement based on consumption trends, avoiding both shortages and overstocking.

4. Time-Saving Operations

Reduces time spent on manual record-keeping and searching for inventory details.

5. Expiry & Batch Tracking

Automatically alerts when products are near expiry, reducing wastage and ensuring patient safety.

6. User Access Control

Role-based access improves security and accountability by limiting functions based on user type (Admin, Manager, Staff).

7. Detailed Reports & Analytics

Generates reports for decision-making, such as supplier performance, monthly stock usage, and expenditure tracking.

8. Scalable and Customizable

Can be scaled for use in small clinics or large hospitals, and customized according to specific needs.

9. Data Security & Backup

Since the system is built on a secure platform (Salesforce), data integrity and backup are well maintained.

10. Cost-Effective in the Long Run

Reduces manpower requirements and prevents financial loss from expired or misplaced stock.

Disadvantages:-

1. Initial Setup Cost

Developing and deploying the system may require investment in licenses, setup, and user training.

2. Technical Expertise Required

Staff may need basic technical knowledge to operate and maintain the system effectively.

3. Dependency on Internet Connectivity

Since it is a cloud-based system, poor or unstable internet connections can affect real-time access.

4. Maintenance and Updates

Regular maintenance is needed to ensure performance and security, which may incur additional costs.

5. Data Migration Challenges

Transferring old manual records into the system may take time and require careful validation.

6. System Downtime Risk

If the server or cloud platform experiences downtime, users may face temporary unavailability.

7. Resistance to Change

Some staff may be reluctant to shift from traditional manual methods to digital tracking initially.

8. Limited Customization (Platform-Dependent)

Certain features may be restricted depending on the Salesforce platform configuration or license type.

Types of Tests Conducted:-

1. Load Testing: Simulated 100 concurrent users; response time maintained at 1.4 seconds.

2. Stress Testing: Pushed system beyond capacity to identify breaking points.

3. Scalability Testing: Tested performance with 10,000+ records.

4. Data Accuracy Testing: Verified stock and transaction accuracy after updates.

Future Scope :-

- The future enhancements for MIMS include:
- Barcode & QR Code Scanning: For instant product identification and updates.
- AI-Based Forecasting: To predict demand and optimize purchase planning.
- IoT Integration: To automatically monitor storage conditions like temperature and humidity.
- Mobile App Version: For on-the-go stock updates and alerts.
- Integration with Hospital ERP: To unify billing, inventory, and patient data.

Conclusion:-

The Medical Inventory Management System implemented in Salesforce streamlines inventory operations, ensures data accuracy, and enhances decision-making. By automating stock management, monitoring supplier performance, tracking product expiry, and providing real-time reports, the system improves efficiency, reduces errors, and supports effective procurement and inventory control. It enhances accuracy, accountability, and operational speed, ensuring that healthcare facilities can deliver uninterrupted and high-quality patient care.

While the system offers numerous advantages such as accuracy, time savings, and improved efficiency, it also requires proper planning during implementation to overcome challenges like setup cost and training.Overall, the benefits significantly outweigh the disadvantages, making it a valuable tool for modern healthcare institutions. The Medical Inventory Management System digitalizes and simplifies the complex task of managing hospital inventory.By integrating automation and analytics, it improves accuracy, reduces costs, and enhances service delivery.Although it needs initial investment and user training, the long-term benefits like real-time tracking, reduced wastage, and efficient stock control make it an essential tool for modern healthcare systems.