Title:- HandsMen Threads: Elevating the Art of Sophistication in Men's Fashion (Salesforce CRM Project)

Abstract:

HandsMen Threads, a premium men's fashion brand, is implementing a Salesforce-based Customer Relationship Management (CRM) system to modernize and automate its business operations. The project focuses on organizing customer data, managing products, tracking orders, monitoring inventory, and supporting a dynamic loyalty program.

The system will streamline the entire retail workflow—from order placement to loyalty rewards—while ensuring accuracy, efficiency, and real-time business insights. Salesforce automation tools such as Flows, Apex Triggers, Email Alerts, and Batch Apex are used to enhance operational performance, reduce manual tasks, and improve customer engagement.

This CRM empowers sales teams, warehouse staff, and marketing teams with centralized information, automated notifications, and intelligent dashboards. By integrating structured data models, robust automation, and enterprise security features, the HandsMen Threads CRM system enables the brand to scale operations, improve customer satisfaction, and build long-term loyalty.

Project Overview:

The HandsMen Threads CRM system is a fully customized Salesforce solution designed to support the end-to-end business lifecycle of a premium fashion retailer. It centralizes customer profiles, product catalogs, order details, inventory levels, and loyalty status under one unified platform.

Built using Salesforce Lightning Experience, the CRM incorporates:

- **Custom Objects** for Customers, Products, Orders, Inventory, Marketing Campaign
- **Record-Triggered Flows** for order confirmation emails and stock alerts
- **Apex Triggers** for updating order totals, adjusting stock, and upgrading loyalty tiers
- **Batch Apex** for scheduled restocking
- **Email Alerts** for confirmation and low-stock notifications
- **Dashboards** for sales insights and inventory monitoring
- **Role-Based Access Control** for sales, inventory, and marketing teams

The system eliminates manual processes by automating:

- Order confirmation
- Stock deduction
- Loyalty reward updates
- Low-stock warehouse alerts
- Scheduled stock synchronization

With improved visibility, accuracy, and decision-making, HandsMen Threads can deliver a consistent, personalized, and efficient customer experience.

Objectives:

Business Objectives:

The primary business objective of the HandsMen Threads CRM system is to establish a centralized and efficient platform for managing customers, products, inventory, and orders within a single unified environment. The system aims to automate critical business operations such as order confirmation, inventory deduction, and loyalty point updates, ensuring faster processing and reduced manual effort. A dynamic loyalty program is implemented to encourage repeat purchases, while real-time low-stock alerts help prevent stockouts and maintain optimal inventory levels. Data accuracy is reinforced through validation rules, ensuring clean and consistent records across the CRM. Additionally, dashboards and reports provide meaningful insights for strategic decision-making and performance tracking. Automated backend processes, scheduled batch jobs, and transactional emails further enhance customer engagement and streamline day-to-day operations for the Sales, Inventory, and Marketing teams.

Technical Objectives

The technical objectives of the HandsMen Threads CRM focus on building a scalable, secure, and automation-driven system tailored for fashion retail operations. A robust custom object model is designed to support the relationships between customers, orders, products, inventory, and loyalty data. Record-Triggered and Scheduled Flows are implemented to automate processes such as order status updates, low-stock alerts, and loyalty calculations. Apex Triggers handle complex business logic that requires greater control and efficiency, while Batch Apex enables automated inventory restocking for products that fall below threshold levels. Profile-based security ensures that internal teams

have appropriate access to the data they need, maintaining confidentiality and data integrity. Finally, a Lightning App with structured navigation and intuitive UI components is developed to provide a seamless and user-friendly experience across all modules of the CRM.

Student Outcomes:

- Hands-on experience in designing a CRM for a real business use-case
- Strong knowledge of Salesforce Object Modeling and Relationships
- Expertise in Flows, Apex Triggers, Batch Apex, and Email Alerts
- Understanding of automation for retail operations (orders, stock, loyalty)
- Skills in building scalable and secure Salesforce applications
- Experience in dashboard creation for analytics and insights
- End-to-end Salesforce project documentation skills
- Confidence to work on real-world Salesforce admin/developer roles

System Requirements:

Hardware Requirements:

- Laptop/PC with minimum 4 GB RAM
- Dual-core processor or above
- Reliable internet connection

Software Requirements:

- Salesforce Developer Edition account
- Modern browser (Chrome recommended)
- Data Import Wizard / Data Loader (optional)

Skills Required:

- Salesforce Configuration and Data Modeling
- Security and Access Management
- Apex Triggers, Classes, and Asynchronous Apex (Queueable, Scheduled)
- Flow Builder (Record-Triggered & Scheduled Flows)
- Lightning Web Components (LWC) Development

- Experience Cloud Site Configuration
- Reports and Dashboard Creation

Phases Overview:

Phase No.	Phase Name	Description	Page Range
1	Requirement Analysis & Planning	Understand business workflow, define scope, create architecture	5–11
2	Salesforce Development – Backend & Configurations	Objects, fields, validation rules, triggers, flows, batch apex	12–62
3	UI/UX Development & Customization	Lightning app, tabs, layouts, navigation	63–65
4	Data Testing & Security	Unit testing, UAT, security setup	66–71
5	Documentation & Maintenance	Final documentation preparation, long-term maintenance	72–75

PHASE 1: Requirement Analysis & Planning:

1. Understanding Business Requirements:

Objective:

To analyze the business processes of HandsMen Threads and identify how Salesforce CRM can streamline customer management, product cataloging, order processing, inventory monitoring, and loyalty programs.

Business Context:

HandsMen Threads is a premium men's fashion brand offering shirts, blazers, pants, accessories, and custom-tailored outfits. The company requires a CRM solution to automate processes, eliminate manual errors, enhance customer experience, and maintain accurate inventory and sales tracking.

Approach

The following methods were used to gather and analyze requirements:

Stakeholder Discussions

Discussions with:

- Sales Team
- Warehouse/Inventory Manager
- Customer Relationship Team
- Marketing Team
- IT/Admin Team

Process Observation

Studied manual processes such as:

- How orders are placed
- How stock is updated
- How customers are tracked
- How loyalty rewards are assigned

Research & Documentation

Requirements gathered using:

- ChatGPT
- Google Research
- Salesforce Documentation
- Trailhead Modules
- Real-world retail CRM examples

Key Business Requirements Identified

1. Customer Management

    - Maintain a complete customer profile: Name, Email, Phone, Address
    - Track total purchase amount and loyalty status
    - Automate loyalty upgrades

2. Product & Inventory Management

    - Store product catalog with price, category, and stock
    - Monitor low stock levels
    - Send automatic alerts to warehouse when stock < 5

3. Order Lifecycle Management

    - Create and track orders
    - Auto-calculate total amount
    - Deduct stock when order is confirmed
    - Prevent confirmation if stock is insufficient
    - Send customers a confirmation email

4. Loyalty Program

    - Automatically assign Bronze/Silver/Gold based on total purchases
    - Send email notification on tier upgrade

5. Notifications & Alerts

    - Email alert for order confirmation
    - Email alert for low stock
    - Automated summary processing via Batch Apex

6. Data Integrity & Validation

    - Quantity must be > 0
    - Email should be valid
    - Stock cannot go negative

7. Role-Based Access

- Sales Team → Customer & Order management
- Inventory Team → Stock & Product management
- Marketing → Loyalty & Campaigns
- Admin → Full access

2. Defining Project Scope & Objectives

Project Scope:

HandsMen Customer Module

- Customer details
- Loyalty status
- Purchase history

Product & Inventory Module

- Product catalog
- Stock monitoring
- Auto stock update

HandsMen Order Module

- Complete order processing
- Formula-based total calculation
- Automated status-based behavior

Warehouse Alerts Module

- Low stock email alerts
- Auto-restock with Batch Apex

Email Notification Suite

- Order confirmations
- Loyalty upgrades

Reports & Dashboards

- Sales analytics
- Stock status
- Customer loyalty insights

Objectives Summary:

Operational Objectives

- Reduce manual effort in order handling
- Improve stock accuracy

- Automate customer engagement
- Simplify loyalty tracking
- Provide real-time business intelligence

Technical Objectives

- Build custom objects for all business entities
- Use formula fields, validation, flows, and triggers
- Automate stock and loyalty updates
- Implement role hierarchy & security model
- Create dashboards for decision-making
- Use batch jobs for scheduled processes

3. Gathering & Analyzing User Needs

User Type	Responsibilities
Sales Team	Creates orders, manages customers
Inventory Team	Tracks stock, monitors low stock
Marketing Team	Manages loyalty program & campaigns
Admin	Full control over system configuration

Functional Needs Identified:

For Sales Team

- Create customer profile
- Create orders
- View customer purchase history
- Access product price and stock availability

For Inventory Team

- Update stock levels
- Receive low stock alerts
- Access product catalog

For Marketing Team

- Track loyalty points
- Send loyalty emails

- Run promotional campaigns

For Admin

- Manage users
- Manage permissions
- Monitor system performance

Tools Used for Requirement Analysis

- Google Docs – Documenting requirements
- Miro – Drawing object relationship diagrams
- Figma (optional) – UI wireframing
- ChatGPT – Process clarification
- Salesforce Setup – Metadata mapping
- Excel – CSV template creation

4. Identifying Salesforce Features & Tools Required

Object	Purpose
HandsMen Customer	Customer details & loyalty
HandsMen Product	Product catalog
HandsMen Order	Order details
Inventory	Stock levels
Marketing Campaign	Manages promotions & campaigns

Salesforce Automation Tools Used:

Flows

- Order Confirmation Flow
- Low Stock Alert Flow
- Scheduled Loyalty Update Flow

Apex

- Order Total Calculation

- Inventory Deduction
- Loyalty Upgrade Logic
- Batch Apex Restock

Email Services

- Classic Email Templates
- Email Alerts

Security

- Profiles
- Permission Sets
- Roles
- Sharing Rules

UI

- Lightning App
- Dynamic Forms
- Page Layouts
- List Views

5. Designing Data Model & Security Model:

Data Model Overview

HandsMen Customer

- Customer Name
- Email, Phone, Address
- Loyalty Status
- Total Purchase Amount

HandsMen Product

- Product Name
- Categor
- Price
- Stock Quantity

HandsMen Order

- Customer
- Product
- Quantity
- Total Amount

- Status

Inventory

- Product
- Current Stock
- Low Stock Limit

Marketing Campaign

- Start Date
- End Date

Security Model Overview:

Role Hierarchy

CEO

- Sales
- Inventory
- Marketing

Profiles

- Sales Profile
- Inventory Profile
- Marketing Profile
- System Administrator

Permission Sets

- Permission_Platform_1 (Full access to custom objects)

Sharing Settings

- Customer — Private
- Orders — Private
- Inventory — Private
- Products — Public Read
- Loyalty — Private

Phase 2: Salesforce Development –Backend & Configurations

This phase forms the core functional foundation of the HandsMen Threads CRM system. It includes custom object creation, field configuration, validation rules, automation flows, email templates, Apex triggers, Apex classes, scheduled batch jobs, and security configurations.

All backend processes are built to automate business tasks such as order confirmation, inventory monitoring, and loyalty status management.

Milestone 1: Salesforce Account Setup

Introduction:

Are you new to Salesforce? Not sure exactly what it is, or how to use it? Don't know where you should start on your learning journey? If you've answered yes to any of these questions, then you're in the right place. This module is for you.

Welcome to Salesforce! Salesforce is game-changing technology, with a host of productivity-boosting features, that will help you sell smarter and faster. As you work toward your badge for this module, we'll take you through these features and answer the question, "What is Salesforce, anyway?".

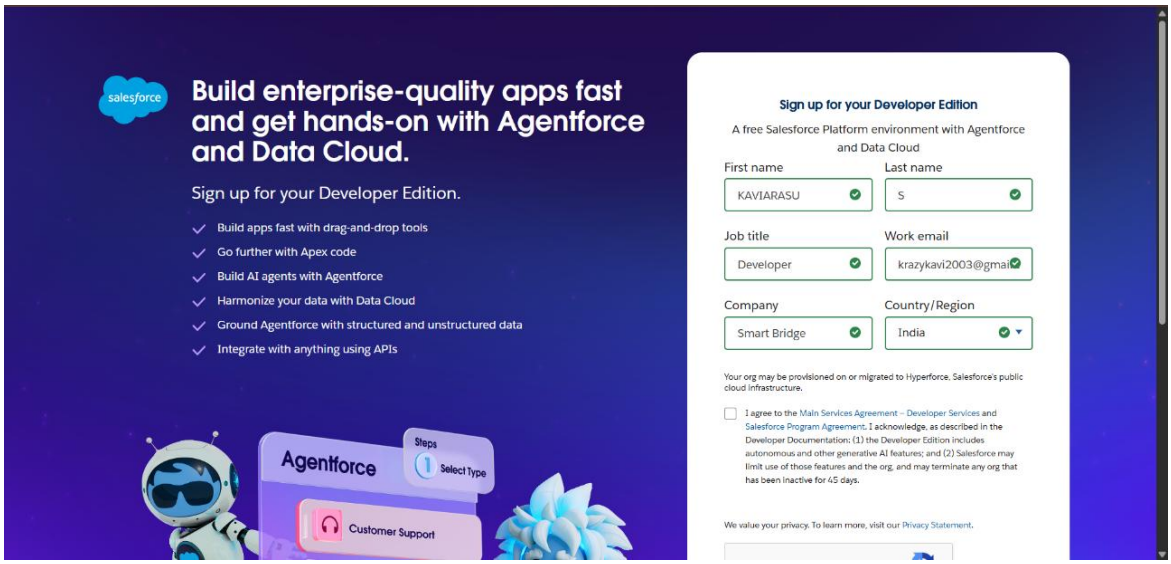
What Is Salesforce?

Salesforce is your customer success platform, designed to help you sell, service, market, analyze, and connect with your customers.

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:

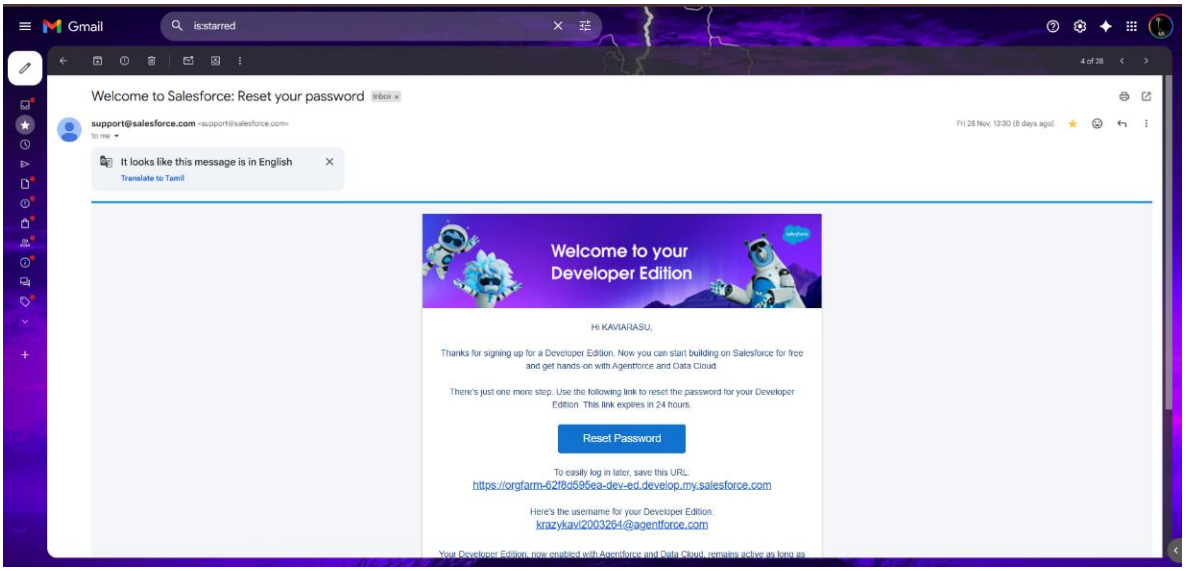


1) First name & Last name
2) Email
3) Job Title: Developer
4) Company: College Name
5) Country: India

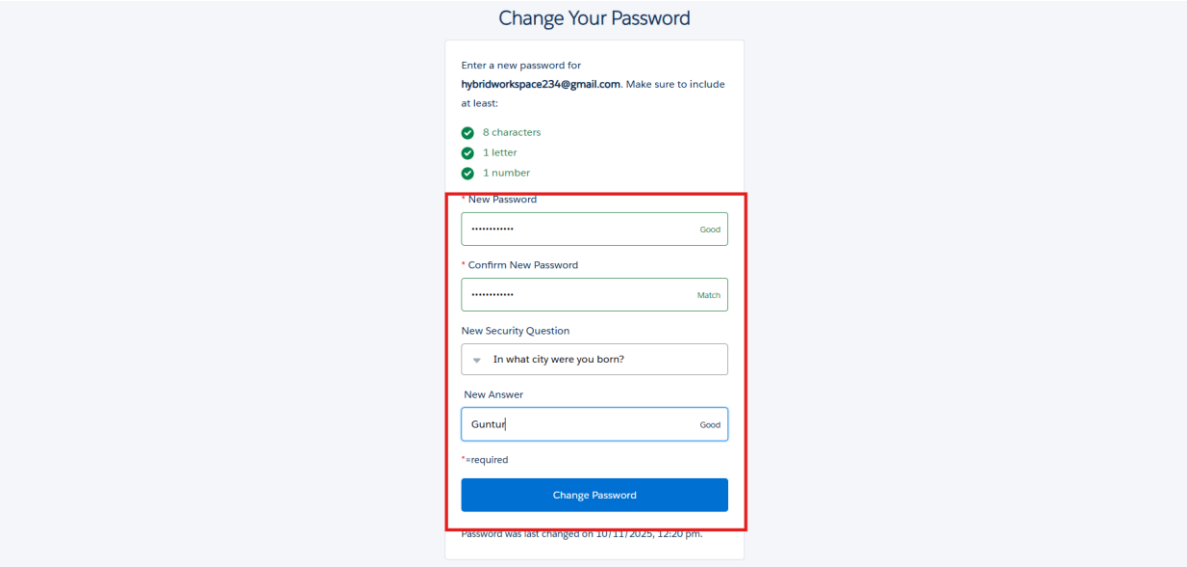
Click on sign me up after filling these.

Activity 2: Account Activation

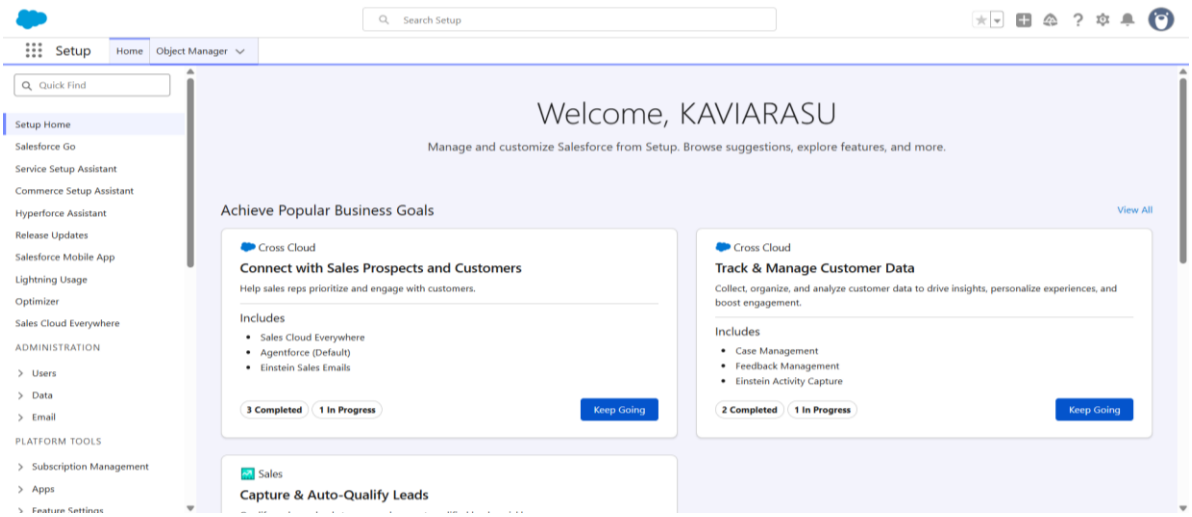
1. Go to the inbox of the email that you used while signing up. Click on the verify account to activate your account. The email may take 10-30mins and sometimes 2 hours.



2. Click on Verify Account
3. Give a password and answer a security question and click on change password.



4. Then you will redirect to your salesforce setup page.



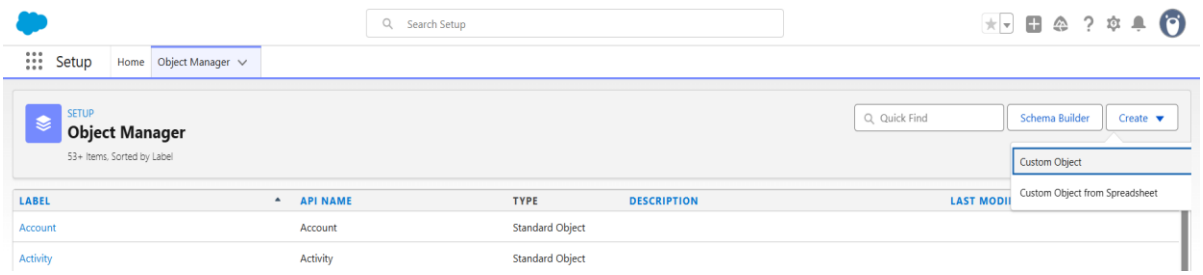
Milestone 2: Custom Object Creation:

HandsMen Threads requires multiple custom objects to store business data. Each object acts as a database table within Salesforce.

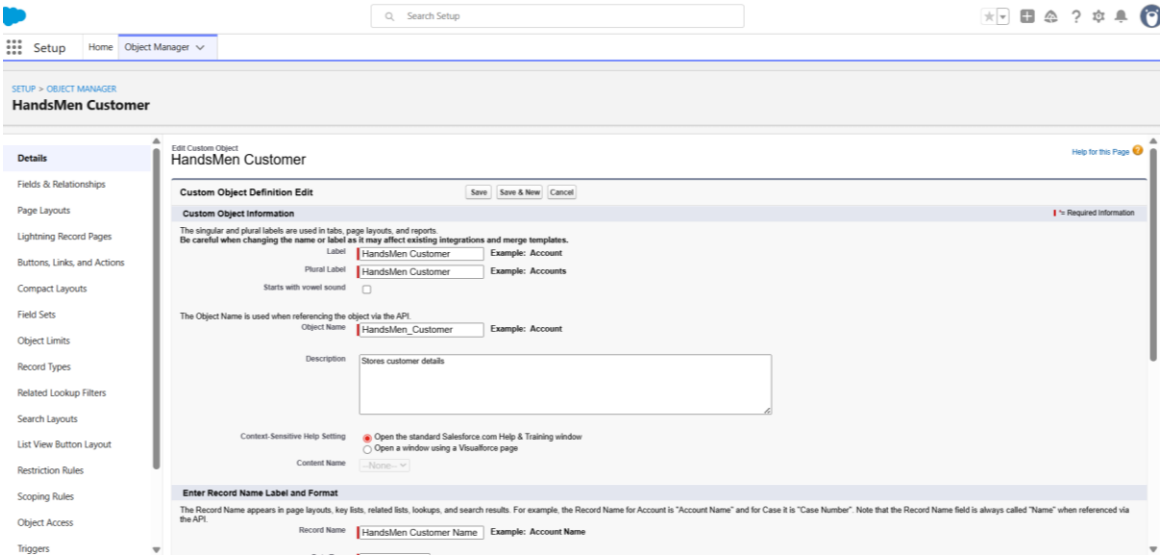
Activity 1: Creating a HandsMen Customer Object

Steps:

1. From the setup page → Click on Object Manager → Click on Create → Click on Custom Object.



2. Enter the label name→ HandsMen Customer
3. Plural label name→ HandsMen Customer
4. Enter Record Name Label and Format
5. Record Name → HandsMen Customer Name
6. Data Type → Text
7. Click on Allow reports,
8. Allow search → Save.



Activity 2: Creating a HandsMen Product Object

Steps:

1. From the setup page → Click on Object Manager → Click on Create → Click on Custom Object.
2. Enter the label name→ HandsMen Product
3. Plural label name→ HandsMen Products
4. Enter Record Name Label and Format
5. Record Name → HandsMen Product Name
6. Data Type → Text
7. Click on Allow reports,
8. Allow search → Save

Activity 3: Creating a HandsMen Order Object

Steps:

1. From the setup page → Click on Object Manager → Click on Create → Click on Custom Object.
2. Enter the label name→ HandsMen Order
3. Plural label name→ HandsMen Orders
4. Enter Record Name Label and Format
5. Record Name → HandsMen OrderNumber
6. Data Type → Auto Number
7. Display Format → O-{0000}
8. Starting Number → 001
9. Click on Allow reports,
10. Allow search → Save

Activity 4: Creating an Inventory Object

Steps:

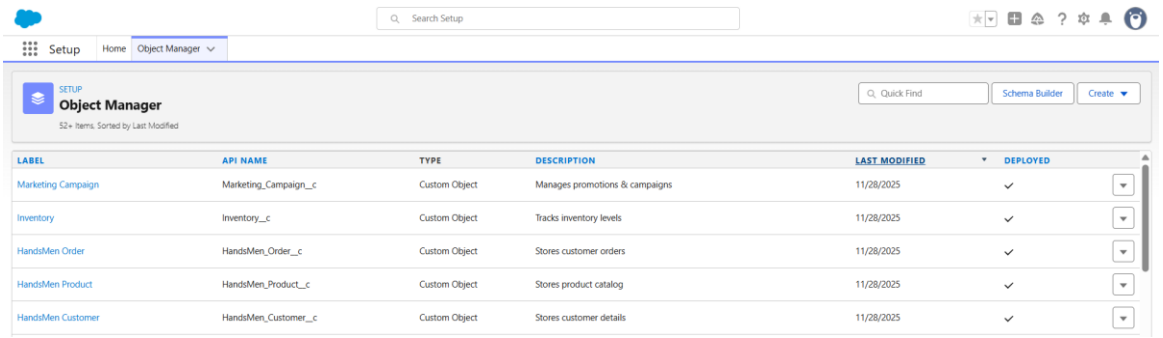
1. From the setup page → Click on Object Manager → Click on Create → Click on Custom Object.
2. Enter the label name→ Inventory
3. Plural label name→ Inventorys
4. Enter Record Name Label and Format
5. Record Name → Inventory Number
6. Data Type → Auto Number

7. Display Format → I -{0000}
8. Starting Number → 001
9. Click on Allow reports,
10. Allow search → Save

Activity 5: Creating a Marketing Campaign Object

Steps:

1. From the setup page → Click on Object Manager → Click on Create → Click on Custom Object.
2. Enter the label name→ Marketing Campaign
3. Plural label name→ Marketing Campaigns
4. Enter Record Name Label and Format
5. Record Name → Marketing Campaign Number
6. Data Type → Auto Number
7. Display Format → MC -{0000}
8. Starting Number → 001
9. Click on Allow reports,
10. Allow search → Save



Milestone 3: Creating Tabs

Activity 1: Creating a tab for All Object

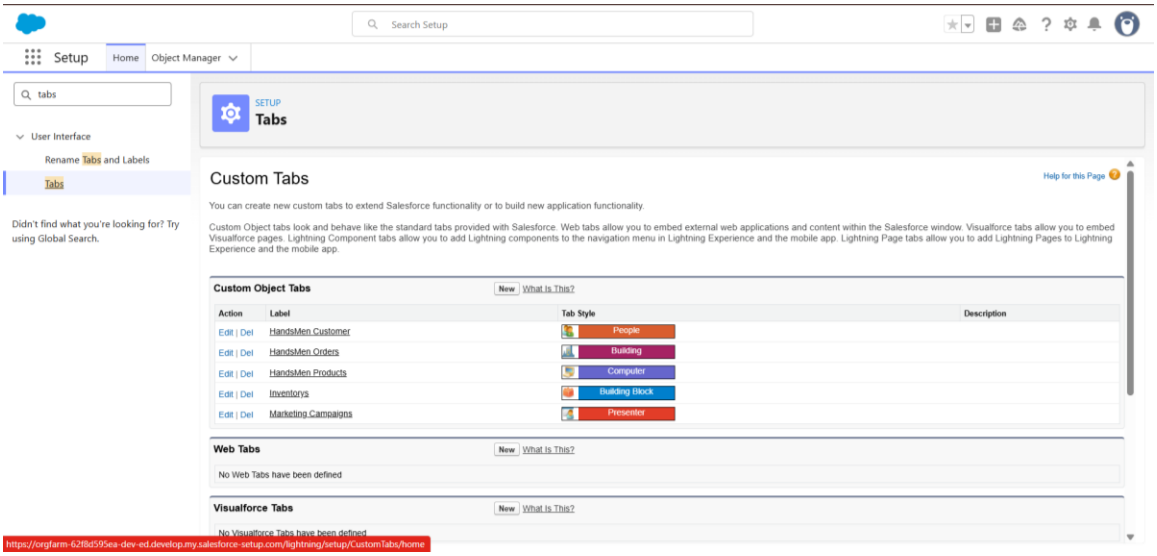
Steps:

1. Setup → Tabs
2. Under **Custom Object Tabs**, click **New**
3. Create tabs for:

- HandsMen Customer
- HandsMen Product
- HandsMen Order
- Inventory
- Marketing Campaign

4. Choose Tab Style → Save

Repeat for each object.

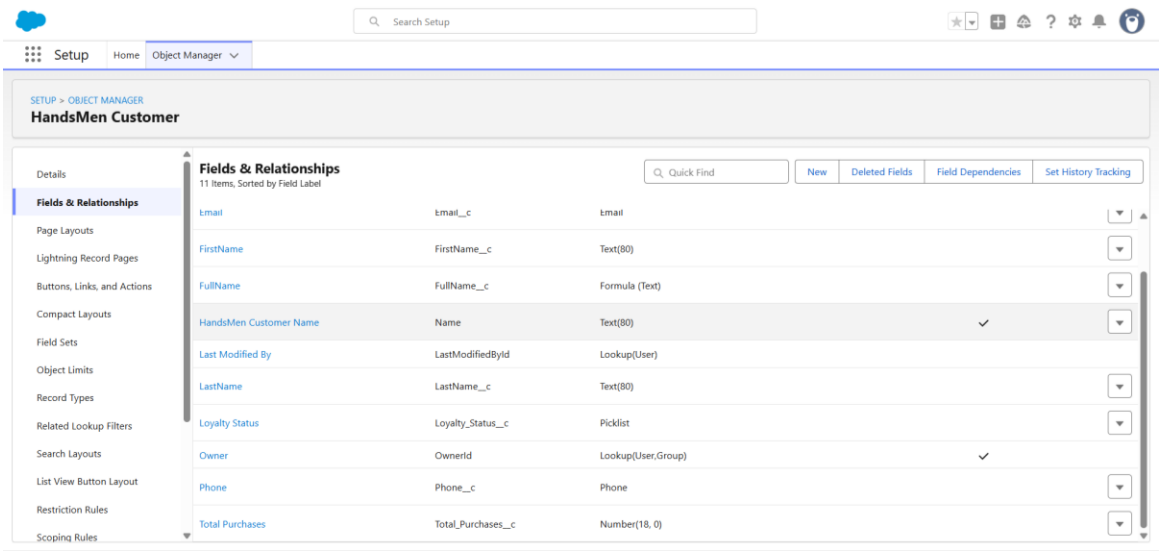


Milestone 4: Fields & Relationships:

HandsMen Customer Fields:

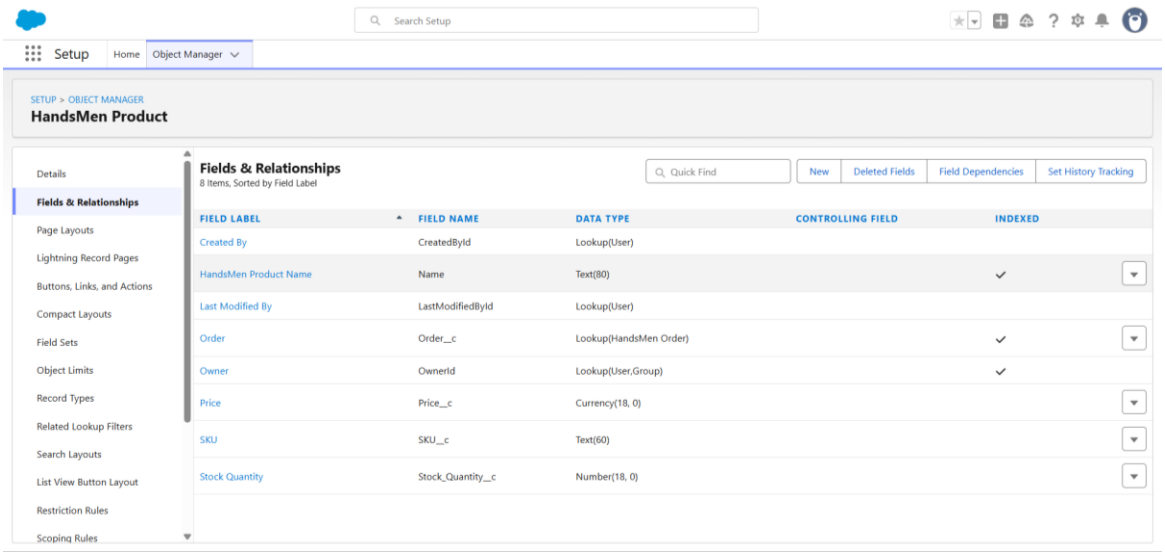
Field Name	Data Type	Required
Customer ID	Auto Number	Yes
FirstName	Text	Yes
LastName	Text	Yes
Customer Name	Text	Yes
Email	Email	Yes
Phone	Phone	Yes

Address	Text Area	No
Loyalty Status	Picklist (Bronze, Silver, Gold)	Yes
Total Purchase Amount	Currency	Yes



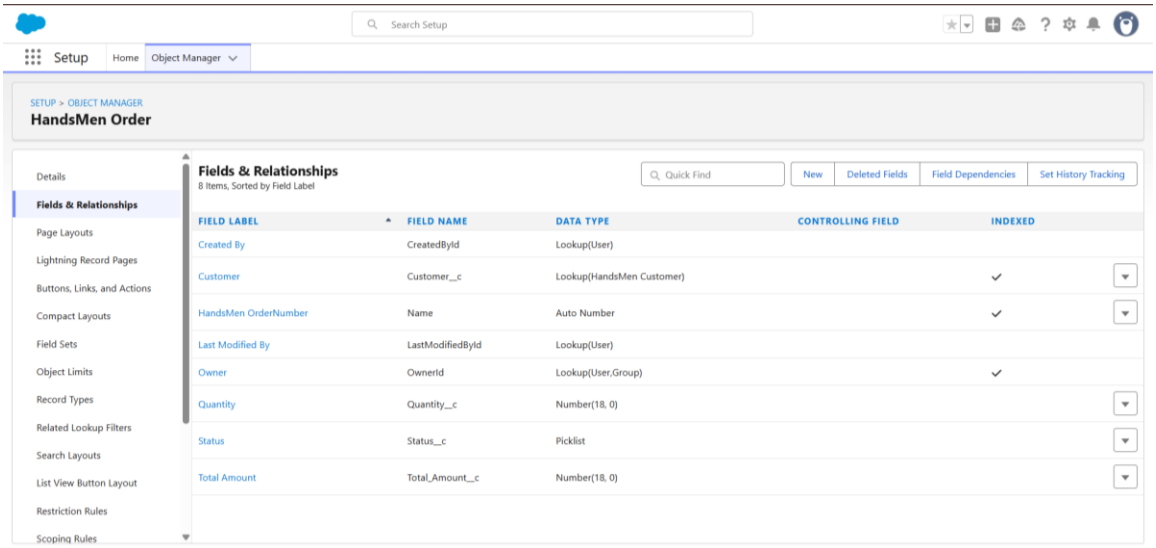
HandsMen Product Fields:

Field Name	**Data Type**
Order	Lookup (HandsMen Order)
SKU	Text
Price	Currency
Stock Quantity	Number



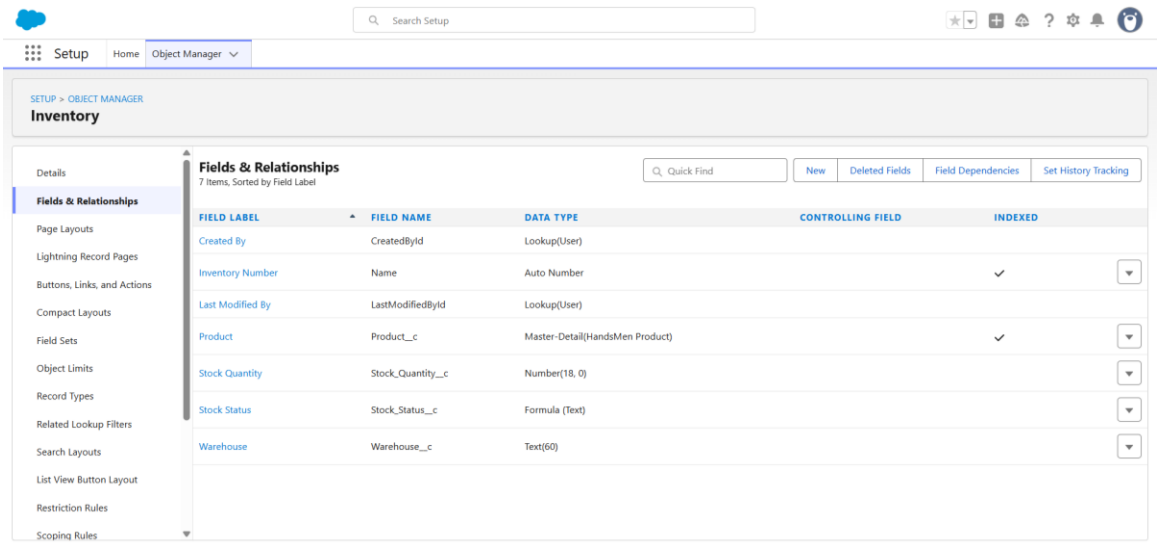
HandsMen Order Fields:

Field Name	Data Type	Description
Order Number	Auto Number	Unique ID
Customer	Lookup (HandsMen Customer)	Customer placing the order
Product	Lookup (HandsMen Product)	Ordered product
Quantity	Number	Order quantity
Status	Picklist (Placed, Confirmed, Rejected)	Order status
Total Amount	Number	Auto-calculated



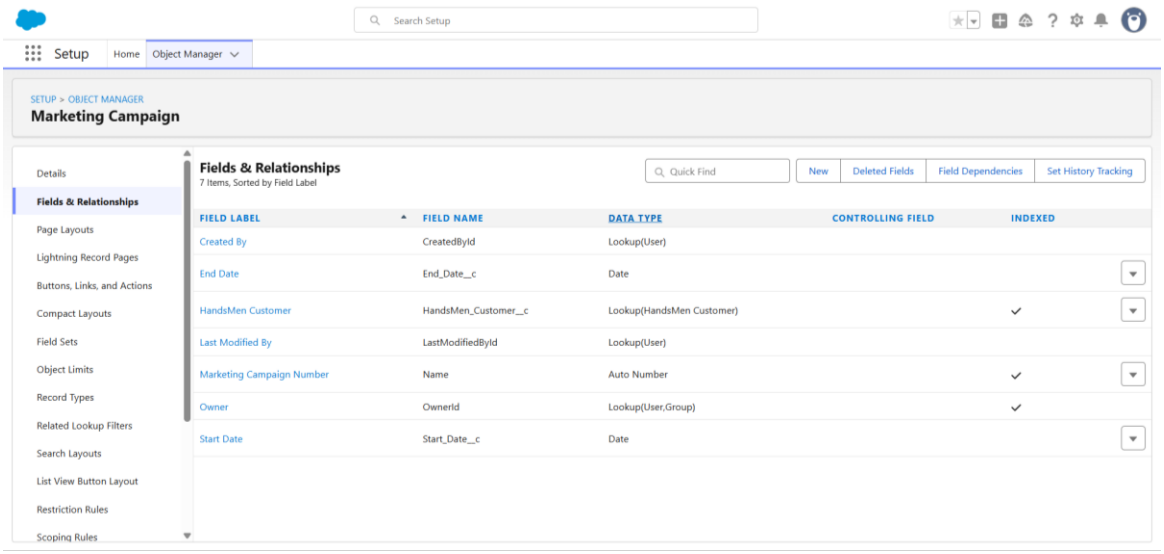
Inventory Fields

Field Name	Data Type
Product	Lookup (HandsMen Product)
Stock Quantity	Number
Stock Status	Formula (Text)
Warehouse	Text



Marketing Campaign Fields:

Field Name	Data Type
HandsMen Customer	Lookup (HandsMen Customer)
Start Date	Date
End Date	Date

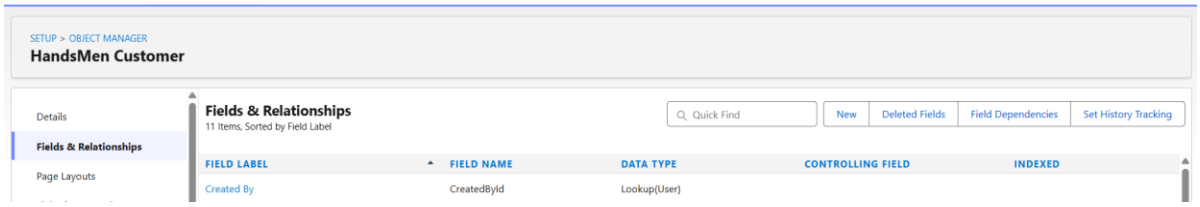


Creation of Formula Fields:

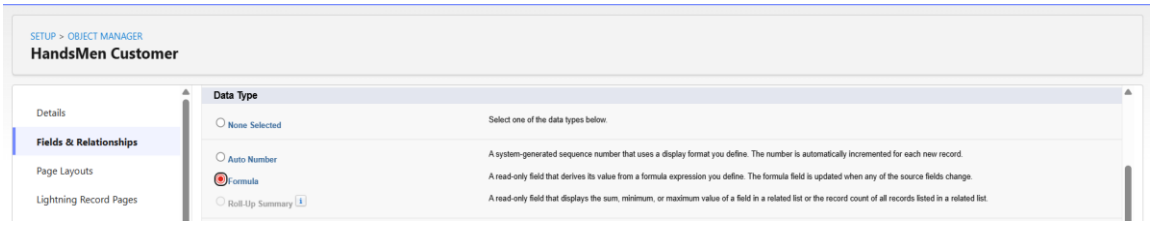
Create Formula Field in HandsMen_Customer__c:

Steps:

1. Go to the setup page → click on object manager → type object name (HandsMen_Customer__c) in the quick find bar → click on the object.
2. Click on fields & relationship → click on New.



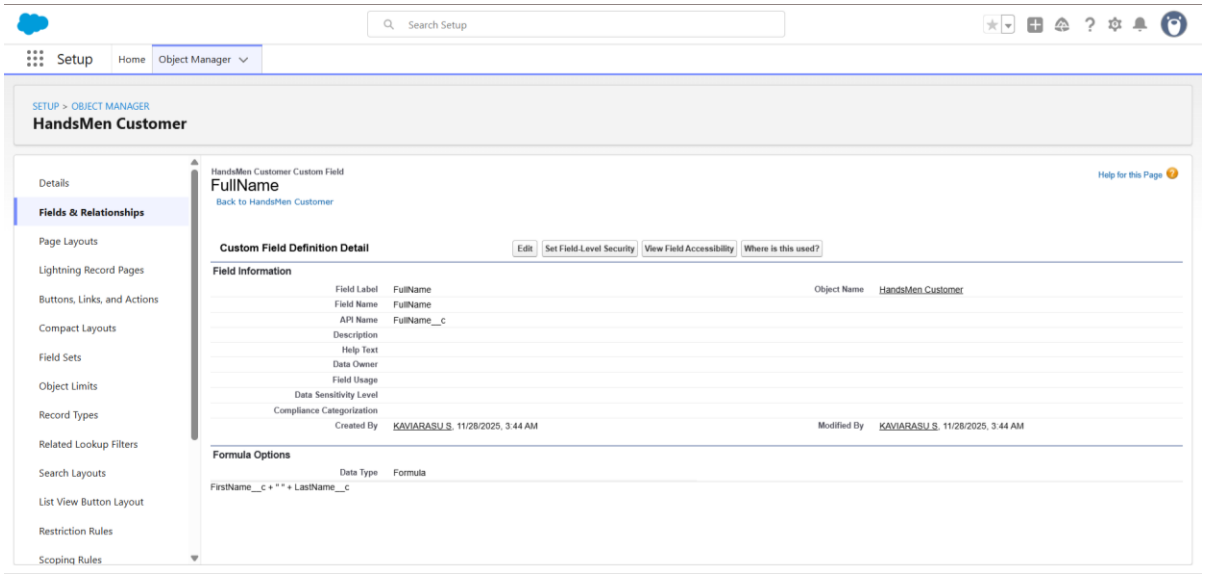
3. Select Data type as “Formula” and click Next.



4. Give Field Label and Field Name as “Full_Name__c” and select formula return type as “Text” and click next.
5. Under Advanced Formula write down the formula and click “Check Syntax” and Next→ Next→ Save & New.

Source Code:

FirstName__c + " " + LastName__c



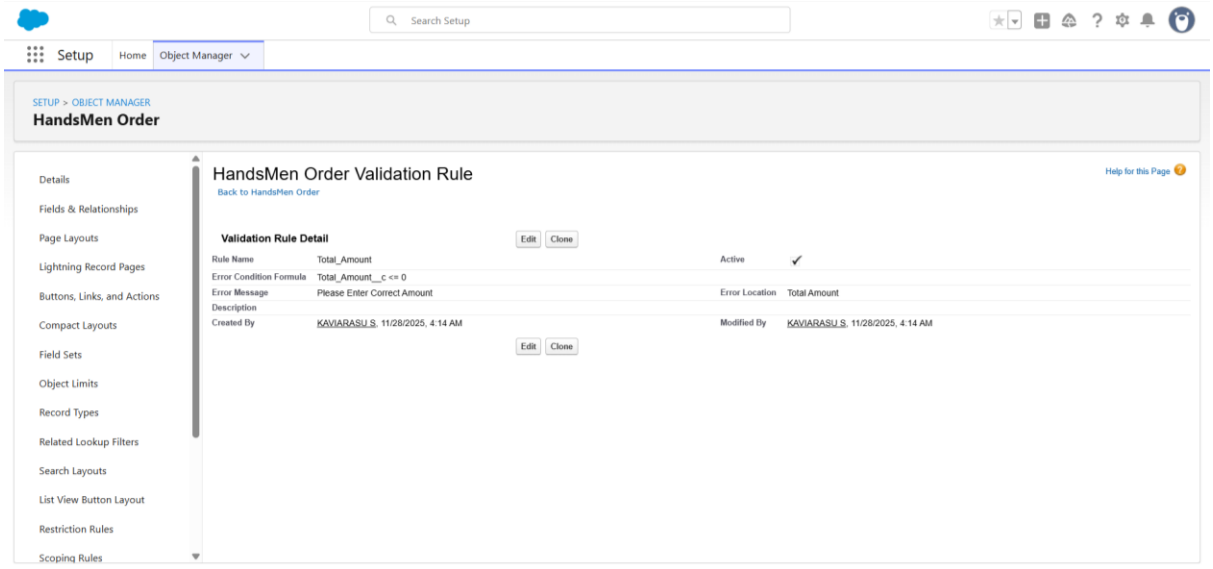
Object	Field	Formula Return type	Formula
Inventory_c	Stock_Status__c	Text	IF(Stock_Quantity__c > 10, "Available", "Low Stock")
HandsMen Customer__c	Full_Name_c	Text	FirstName & " " & LastName

Milestone 5: Validation Rules

Validation Rule 1: Total Amount must be Greater Than Zero

- **Object:** HandsMen Order
- **Rule Name:** Total Amount
- **Formula:**

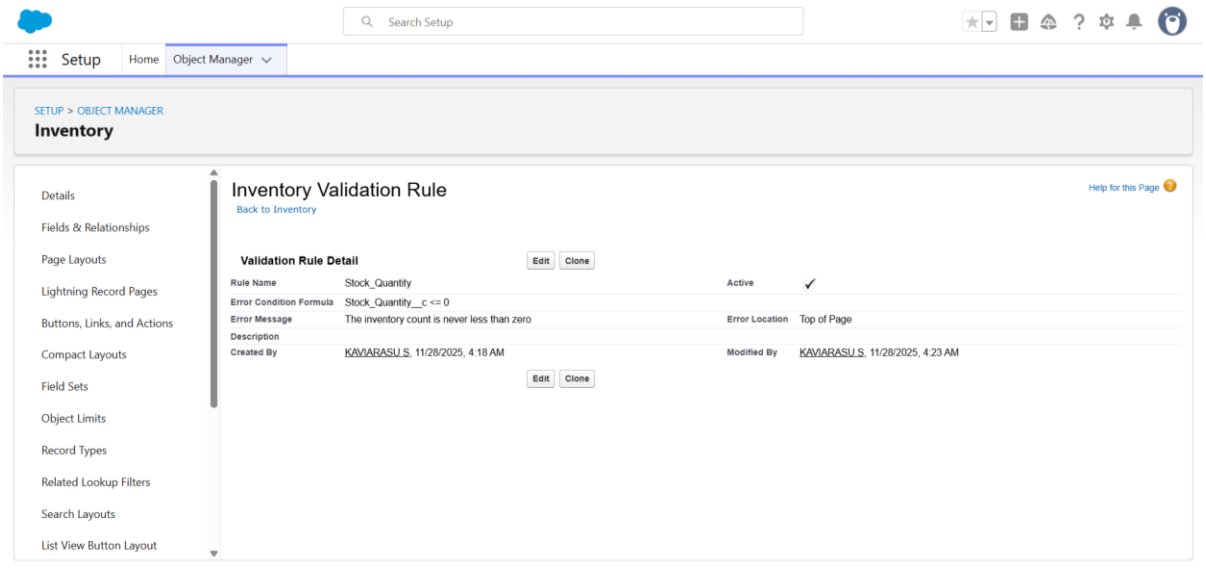
  Total_Amount__c <= 0
- **ErrorMessage:**
  - *Please Enter Correct Amount*
- **Error Location:** Field



Validation Rule 2: Stock Quantity must be Greater Than Zero

- **Object:** Inventory
- **Rule Name:** Stock Quantity
- **Formula:**

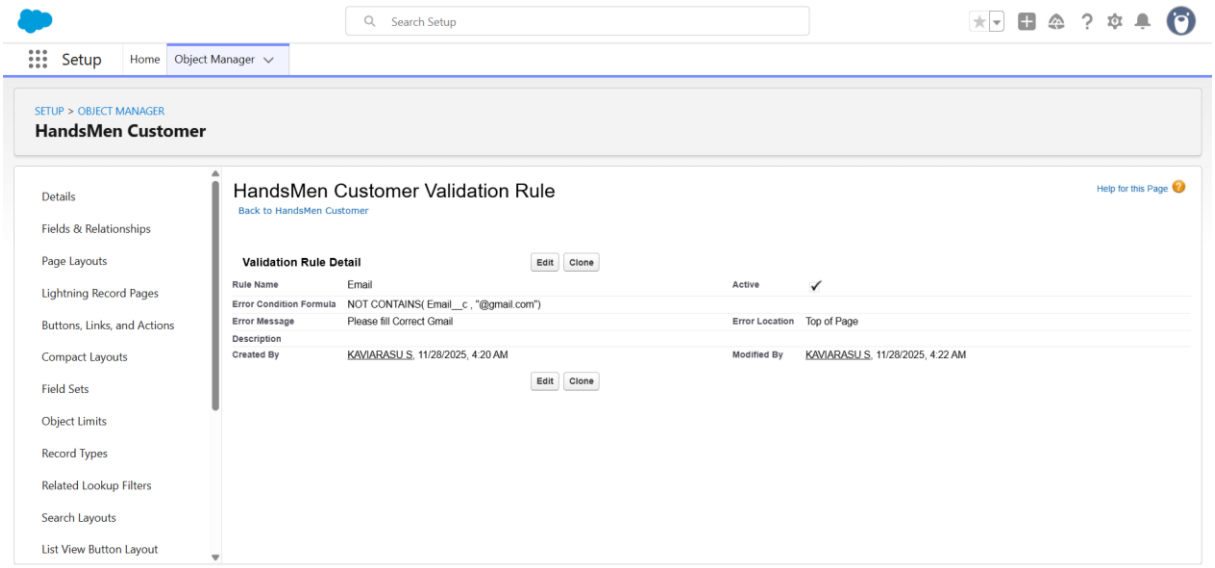
  Stock_Quantity__c < = 0
- **ErrorMessage:**
  - *The inventory count is never less than zero.*
- **Error Location:** Top of Page



Validation Rule 3: Email

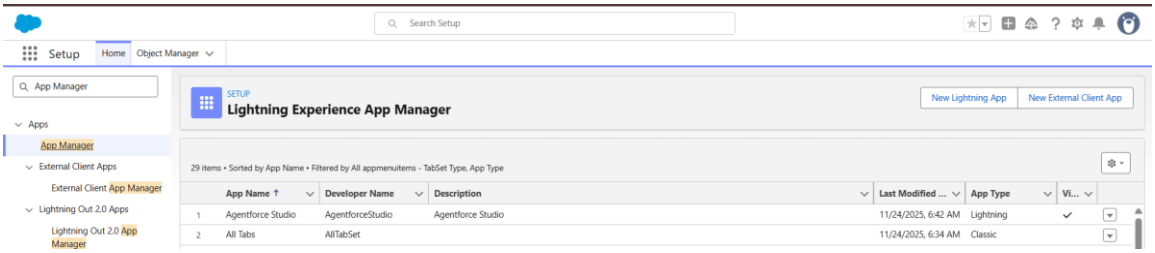
- **Object:** HandsMen Customer
- **Rule Name:** Email
- **Formula:**
- NOT CONTAINS(Email, "@gmail.com")
- **ErrorMessage:**

  Please fill Correct Gmail
- **Error Location:** Top of Page



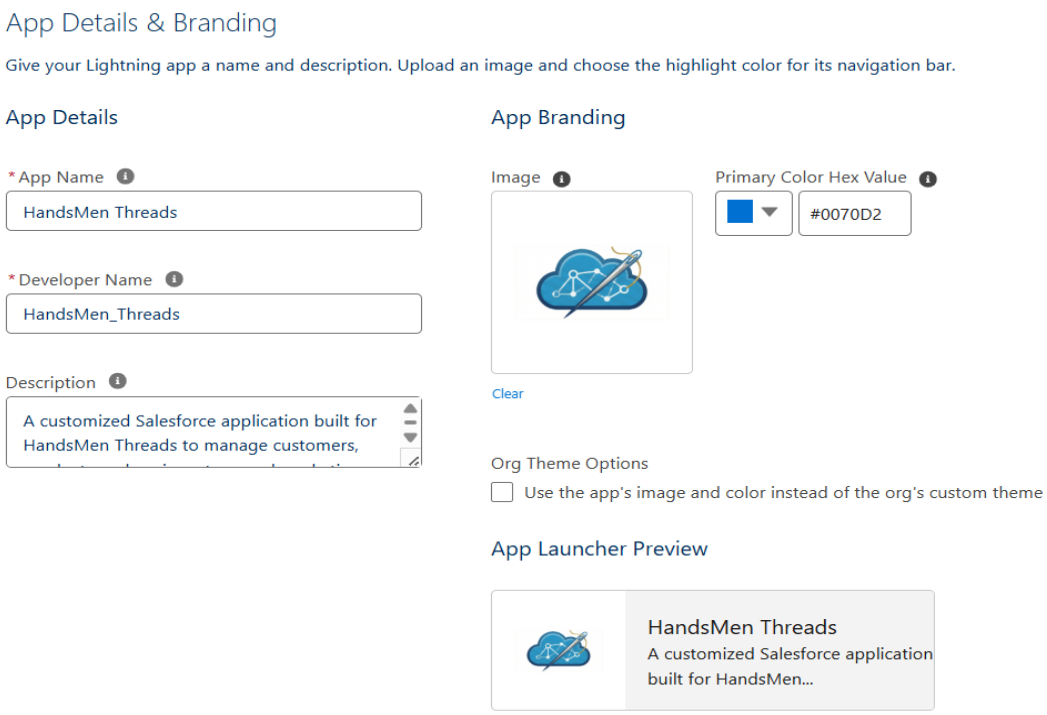
Milestone 6: Lightning App Creation

1. Go to setup page → search “app manager” in quick find → select “app manager” → click on new lightning App.

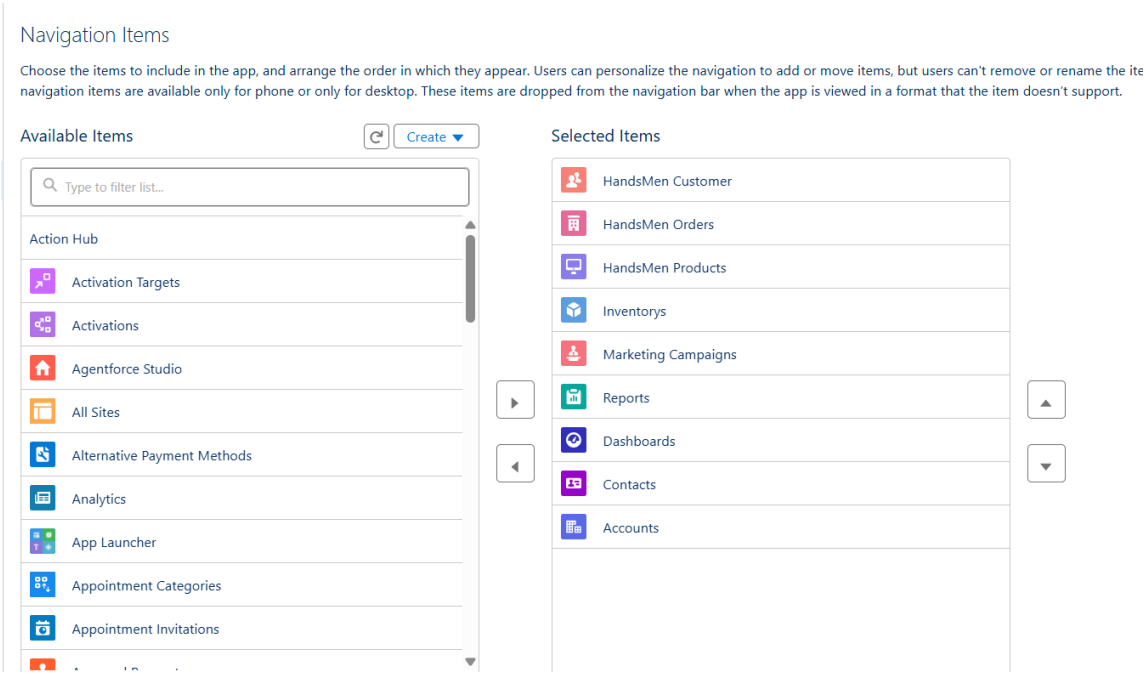


2. Fill the app name in app details and branding as follow

- App Name : HandsMen Threads
- Developer Name : this will auto populated
- Description : Give a meaningful description
- Image : optional (if you want to give any image you can otherwise not mandatory)
- Primary color hex value : keep this default



3. Then click Next → (App option page) keep it as default → Next → (Utility Items) keep it as default → Next.
4. To Add Navigation Items: Search the items in the search bar(HandsMen Customer, HandsMen Order, Inventory, HandsMen Product, Reports, Dashboard, Account, Contact , Marketing Campaign) from the search bar and move it using the arrow button → Next.



Note: select the custom object which we have created in the previous activity.

5. To Add User Profiles: Search profiles (System administrator) in the search bar → click on the arrow button → save & finish

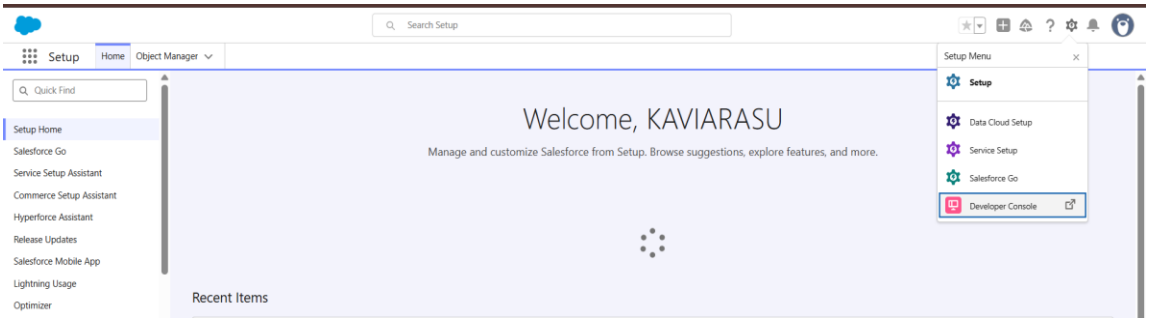
Milestone 7: Apex Triggers & Handler Classes

Apex Trigger 1: Order Total Calculation

Create an Apex Class

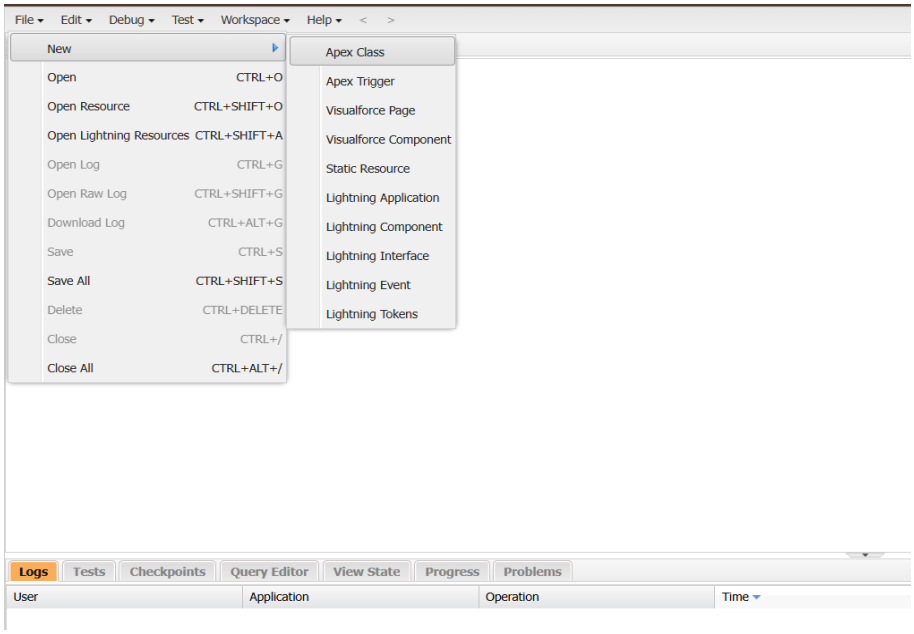
Step 1: Open Developer Console

- Go to Setup → click the gear icon → select Developer Console.
- A new console window will open.



Step 2: Create a New Apex Class

- In Developer Console, click File → New → Apex Class.
- Enter the Apex Class Name as: OrderTotalHandler
- Click OK.

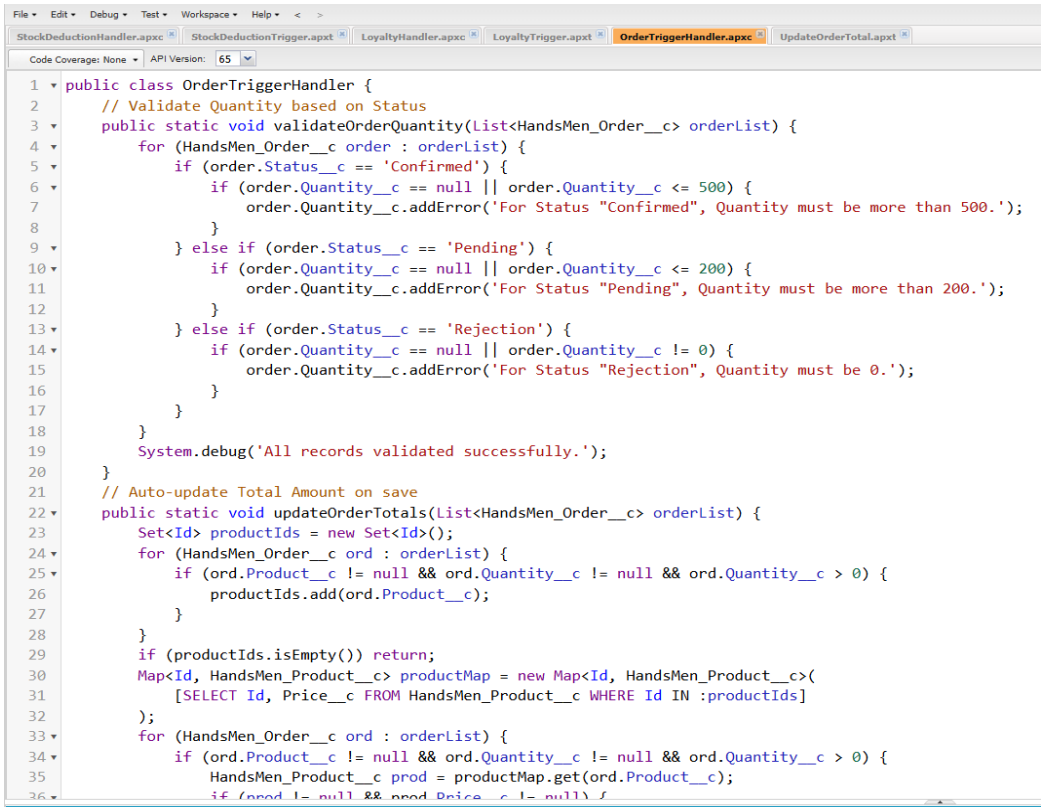


Step 3: Write the Code Logic

```
public class OrderTriggerHandler {
    // Validate Quantity based on Status
    public static void validateOrderQuantity(List<HandsMen_Order__c> orderList) {
        for (HandsMen_Order__c order : orderList) {
            if (order.Status__c == 'Confirmed') {
                if (order.Quantity__c == null || order.Quantity__c <= 500) {
                    order.Quantity__c.addError('For Status "Confirmed", Quantity must be more than 500.');
```

```
            }
        } else if (order.Status__c == 'Pending') {
            if (order.Quantity__c == null || order.Quantity__c <= 200) {
                order.Quantity__c.addError('For Status "Pending", Quantity must be more than 200.');
            }
        } else if (order.Status__c == 'Rejection') {
            if (order.Quantity__c == null || order.Quantity__c != 0) {
                order.Quantity__c.addError('For Status "Rejection", Quantity must be 0.');
            }
        }
    }
    System.debug('All records validated successfully.');
  }
  // Auto-update Total Amount on save
  public static void updateOrderTotals(List<HandsMen_Order__c> orderList) {
    Set<Id> productIds = new Set<Id>();
    for (HandsMen_Order__c ord : orderList) {
      if (ord.Product__c != null && ord.Quantity__c != null && ord.Quantity__c > 0) {
        productIds.add(ord.Product__c);
      }
    }
    if (productIds.isEmpty()) return;
    Map<Id, HandsMen_Product__c> productMap = new Map<Id, HandsMen_Product__c>(
      [SELECT Id, Price__c FROM HandsMen_Product__c WHERE Id IN :productIds]
    );
    for (HandsMen_Order__c ord : orderList) {
      if (ord.Product__c != null && ord.Quantity__c != null && ord.Quantity__c > 0) {
        HandsMen_Product__c prod = productMap.get(ord.Product__c);
        if (prod != null && prod.Price__c != null) {
          ord.Total_Amount__c = prod.Price__c * ord.Quantity__c;
```

```
            } else {
                ord.Total_Amount__c = 0;
            }
        } else {
            ord.Total_Amount__c = 0;
        }
    }
}
```



```apex
public class OrderTriggerHandler {
    // Validate Quantity based on Status
    public static void validateOrderQuantity(List<HandsMen_Order__c> orderList) {
        for (HandsMen_Order__c order : orderList) {
            if (order.Status__c == 'Confirmed') {
                if (order.Quantity__c == null || order.Quantity__c <= 500) {
                    order.Quantity__c.addError('For Status "Confirmed", Quantity must be more than 500.');
                }
            } else if (order.Status__c == 'Pending') {
                if (order.Quantity__c == null || order.Quantity__c <= 200) {
                    order.Quantity__c.addError('For Status "Pending", Quantity must be more than 200.');
                }
            } else if (order.Status__c == 'Rejection') {
                if (order.Quantity__c == null || order.Quantity__c != 0) {
                    order.Quantity__c.addError('For Status "Rejection", Quantity must be 0.');
                }
            }
        }
        System.debug('All records validated successfully.');
    }
    // Auto-update Total Amount on save
    public static void updateOrderTotals(List<HandsMen_Order__c> orderList) {
        Set<Id> productIds = new Set<Id>();
        for (HandsMen_Order__c ord : orderList) {
            if (ord.Product__c != null && ord.Quantity__c != null && ord.Quantity__c > 0) {
                productIds.add(ord.Product__c);
            }
        }
        if (productIds.isEmpty()) return;
        Map<Id, HandsMen_Product__c> productMap = new Map<Id, HandsMen_Product__c>(
            [SELECT Id, Price__c FROM HandsMen_Product__c WHERE Id IN :productIds]
        );
        for (HandsMen_Order__c ord : orderList) {
            if (ord.Product__c != null && ord.Quantity__c != null && ord.Quantity__c > 0) {
                HandsMen_Product__c prod = productMap.get(ord.Product__c);
```

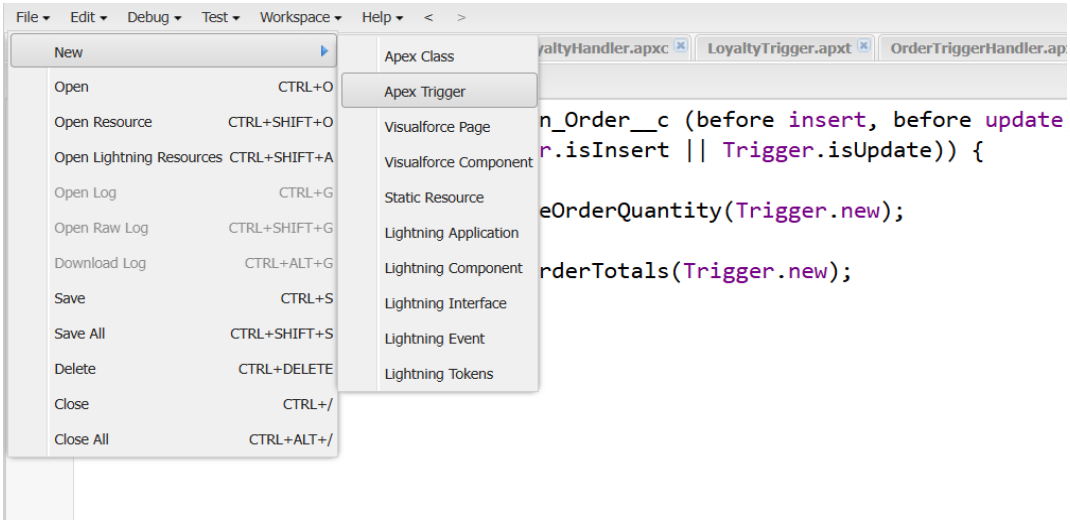
Step 4: Save the Class

- Click File → Save.

Create an Apex Trigger

Step 1: Create a New Apex Trigger

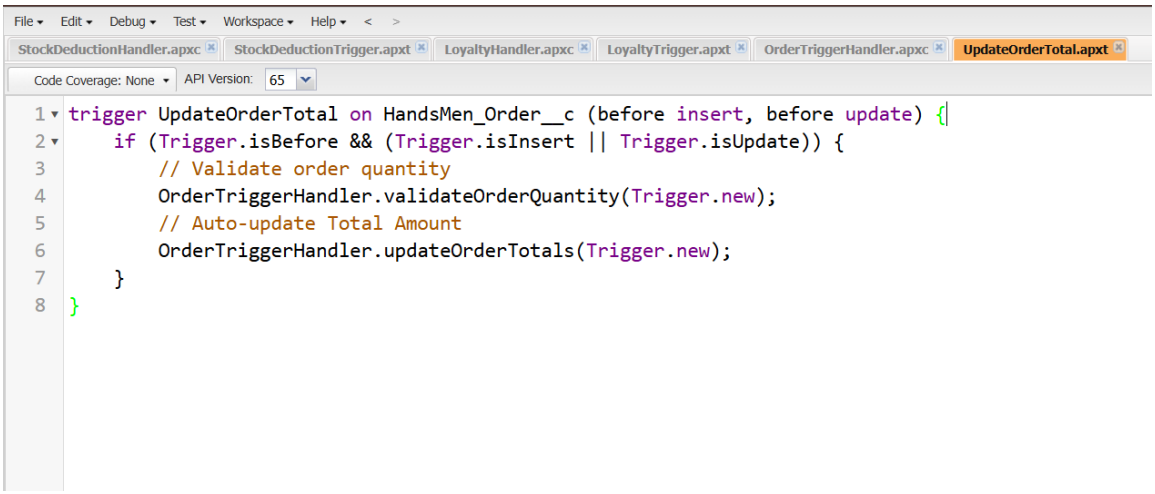
- In Developer Console, click File → New → Apex Trigger.



- Enter:
  - **Trigger Name:** UpdateOrderTotal
  - **sObject:** Select HandsMen_Order__c from the dropdown.
- Click Submit.

Step 2: Write the Trigger Logic

```
trigger UpdateOrderTotal on HandsMen_Order__c (before insert, before update) {
    if (Trigger.isBefore && (Trigger.isInsert || Trigger.isUpdate)) {
        // Validate order quantity
        OrderTriggerHandler.validateOrderQuantity(Trigger.new);
        // Auto-update Total Amount
        OrderTriggerHandler.updateOrderTotals(Trigger.new);
    }
}
```



Step 3: Save the Trigger

- Click File → Save.

Apex Trigger 2: Reduce Inventory When Order is Confirmed

Create an Apex Class

Step 1: Open Developer Console

- Go to Setup → click on the gear icon → select Developer Console.
- A new console window will open.

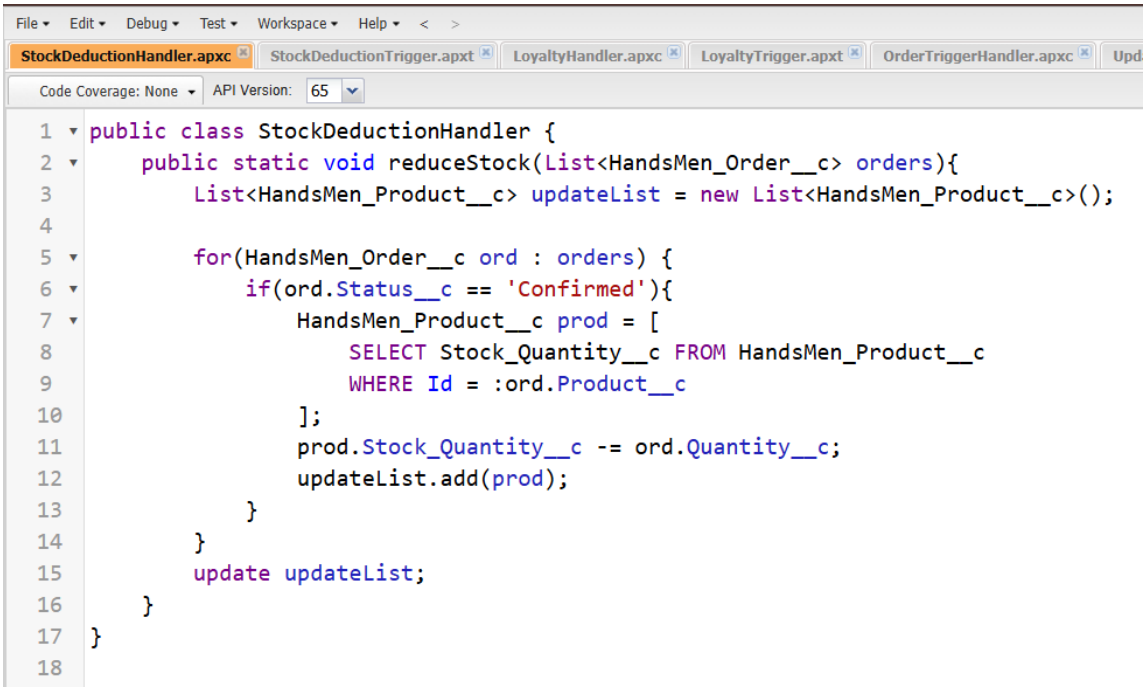
Step 2: Create a New Apex Class

- In Developer Console, click File → New → Apex Class.
- Enter the Apex Class name: StockDeductionHandler
- Click OK.

Step 3: Write the Code Logic

```
public class StockDeductionHandler {
   public static void reduceStock(List<HandsMen_Order__c> orders){
      List<HandsMen_Product__c> updateList = new
List<HandsMen_Product__c>();
      for(HandsMen_Order__c ord : orders) {
         if(ord.Status__c == 'Confirmed'){
            HandsMen_Product__c prod = [
               SELECT Stock_Quantity__c FROM HandsMen_Product__c
               WHERE Id = :ord.Product__c ];
            prod.Stock_Quantity__c -= ord.Quantity__c;
            updateList.add(prod);
```

```
        }
    }
    update updateList;
  }
}
```



Step 4: Save the Class

- Click File → Save.

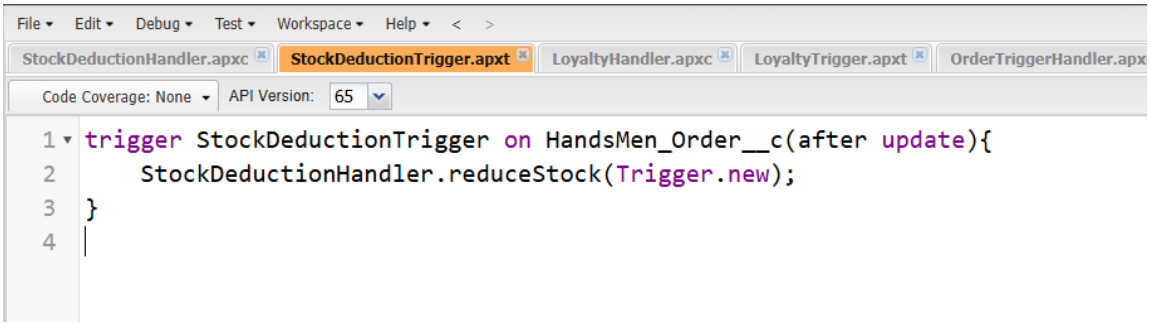
Create an Apex Trigger

Step 1: Create a New Trigger

- In Developer Console, click File → New → Apex Trigger.
- Enter:
  - **Trigger Name:** StockDeductionTrigger
  - **sObject:** HandsMen_Order__c
- Click Submit.

Step 2: Write Trigger Code

```
trigger StockDeductionTrigger on HandsMen_Order__c(after update){
    StockDeductionHandler.reduceStock(Trigger.new);
}
```



Step 3: Save the Trigger

- Click File → Save.

Apex Trigger 3: Loyalty Status Update

Create an Apex Class

Step 1: Open Developer Console

- Go to Setup → Developer Console → new window opens.

Step 2: Create a New Apex Class

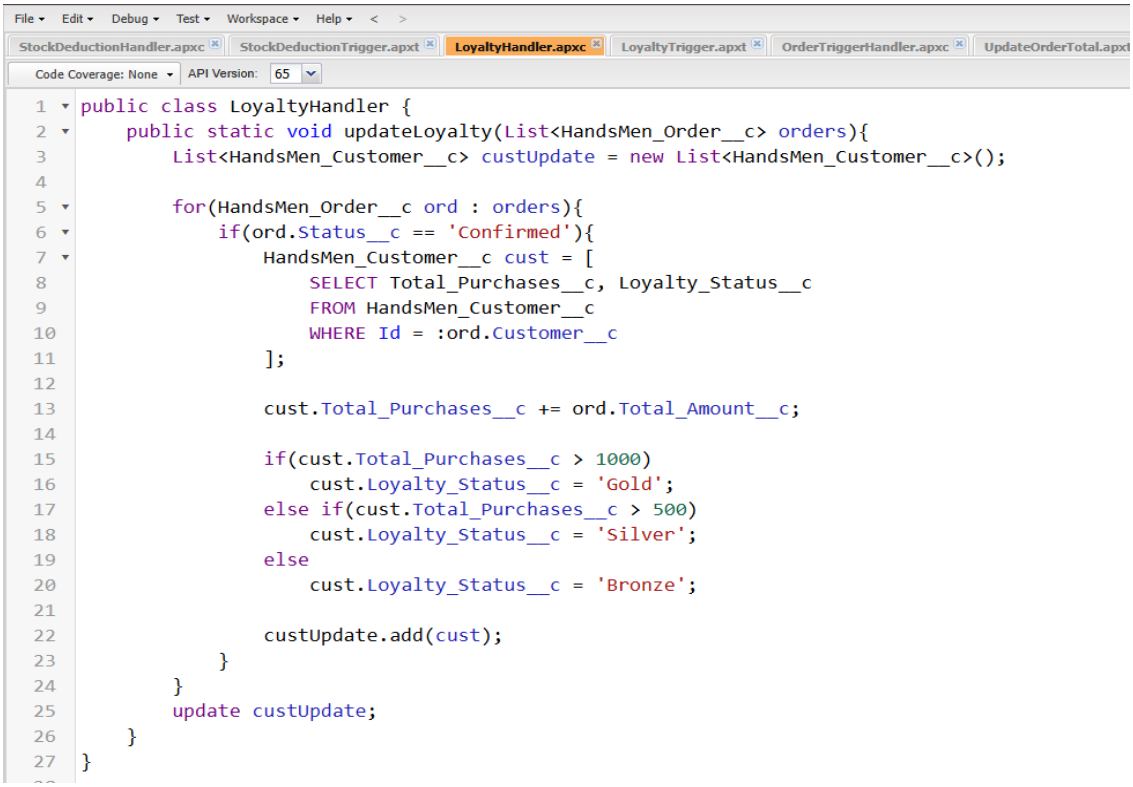
- Click File → New → Apex Class.
- Enter the Class Name: LoyaltyHandler
- Click OK.

Step 3: Write the Code Logic

```
public class LoyaltyHandler {
   public static void updateLoyalty(List<HandsMen_Order__c> orders){
      List<HandsMen_Customer__c> custUpdate = new
List<HandsMen_Customer__c>();
      for(HandsMen_Order__c ord : orders){
         if(ord.Status__c == 'Confirmed'){
            HandsMen_Customer__c cust = [
               SELECT Total_Purchase_Amount__c, Loyalty_Status__c
               FROM HandsMen_Customer__c
               WHERE Id = :ord.Customer__c
            ];
            cust.Total_Purchase_Amount__c += ord.Total_Amount__c;

            if(cust.Total_Purchase_Amount__c > 1000)
               cust.Loyalty_Status__c = 'Gold';
            else if(cust.Total_Purchase_Amount__c > 500)
```

```
                cust.Loyalty_Status__c = 'Silver';
            else
                cust.Loyalty_Status__c = 'Bronze';
            custUpdate.add(cust);
        }
    }
    update custUpdate;
  }
}
```



Step 4: Save the Class

- Click File → Save.

Create an Apex Trigger

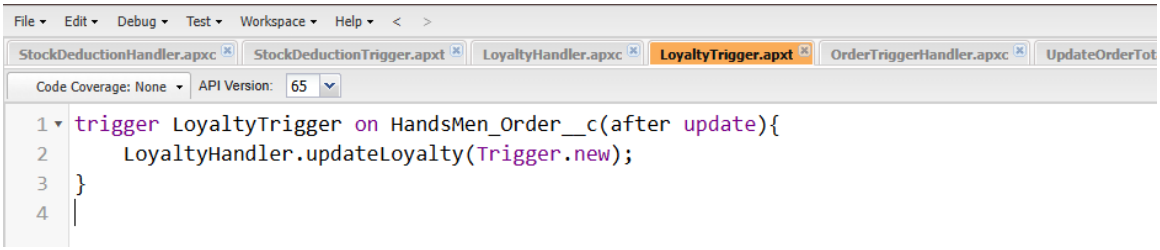
Step 1: Create a New Trigger

- Click **File → New → Apex Trigger**.
- Enter:
    - **Trigger Name:** LoyaltyTrigger

- **sObject:** HandsMen_Order__c
- Click **Submit**.

Step 2: Write Trigger Code

```
trigger LoyaltyTrigger on HandsMen_Order__c(after update){
   LoyaltyHandler.updateLoyalty(Trigger.new);
}
```



Step 3: Save the Trigger

- Click **File → Save**.

Milestone 8: Email Templates

Email Template 1 – Order Confirmation

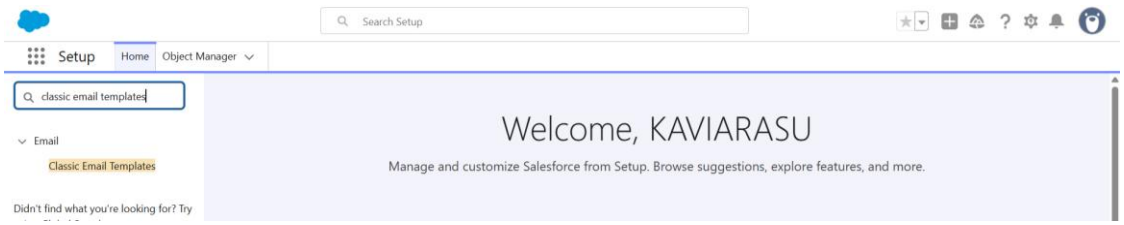
Steps to Create the Order Confirmation Email Template

Step 1: Go to Salesforce Setup

- Click the **Gear I con** → Select **Setup**.

Step 2: Navigate to Classic Email Templates

- In the **Quick Find** box, search **Classic Email Templates**.
- Click on **Classic Email Templates**.

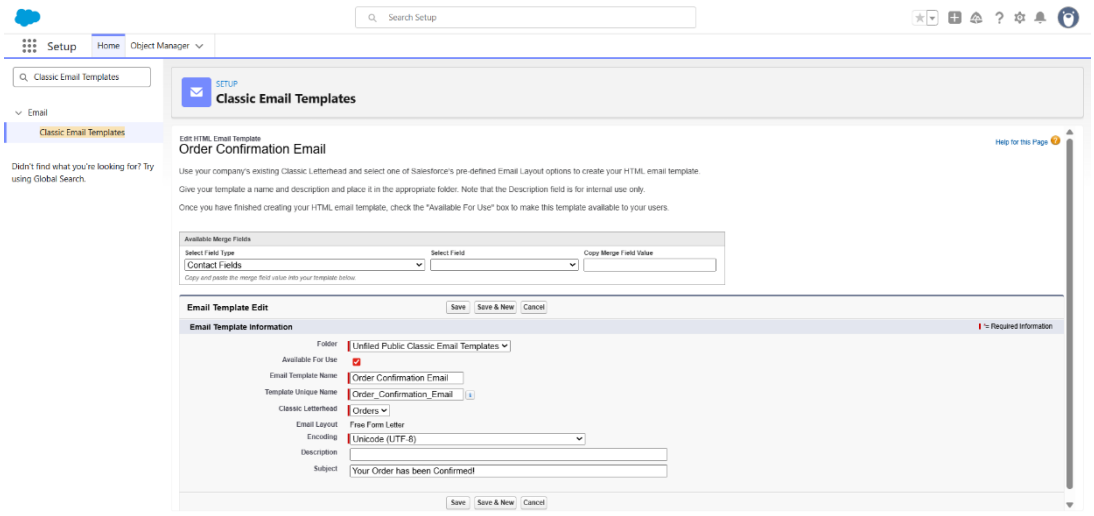


Step 3: Click "New Template"

- Choose **HTML (with Classic Letterhead)**.
- Click **Next**.

Step 4: Fill in Template Details

- **Folder:** Unfiled Public Email Templates
- **Available for Use:** Enable
- **Email Template Name:** Order_Confirmation_Email
- **Encoding:** UTF-8
- **Subject:** Your Order Has Been Confirmed

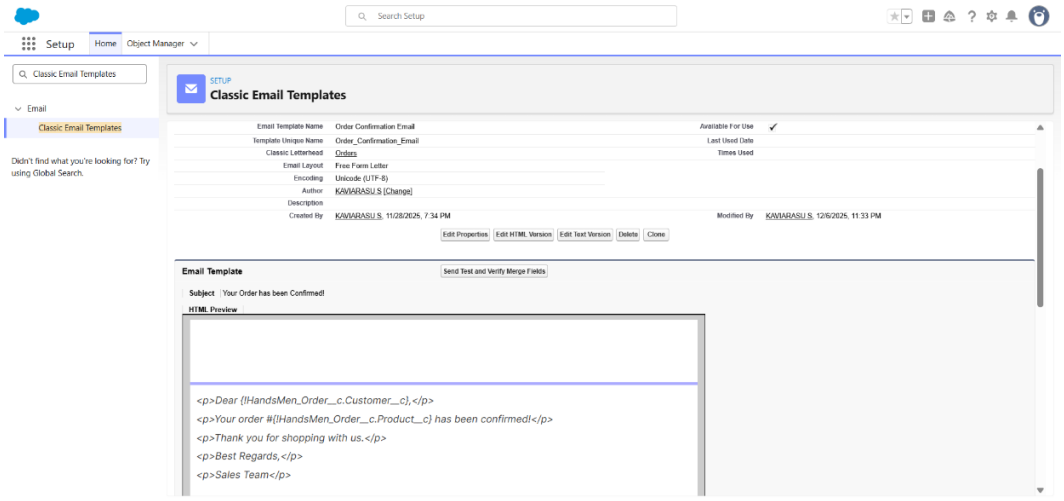


Step 5: Add HTML Body

```
<p>Dear {!HandsMen_Order__c.Customer__c},</p>
<p>Your order #{!HandsMen_Order__c.Product__c} has been confirmed!</p>
<p>Thank you for shopping with us.</p>

<p>Best Regards,</p>

<p>Sales Team</p>
```



Step 6: Save the Template

Email Template 2 – Low Stock Alert

Steps to Create the Low Stock Alert Email Template

Step 1: Go to Salesforce Setup

- Click the **Gear Icon** → Choose **Setup**.

Step 2: Navigate to Classic Email Templates

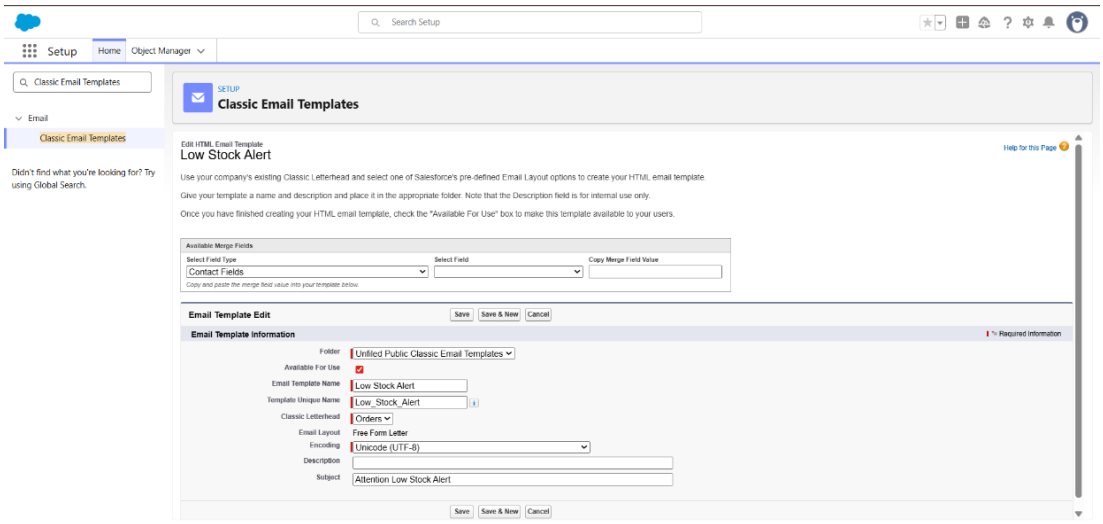
- Search **Classic Email Templates** in Quick Find.
- Click on it.

Step 3: Click "New Template"

- Select **HTML (with Classic Letterhead)**.
- Click **Next**.

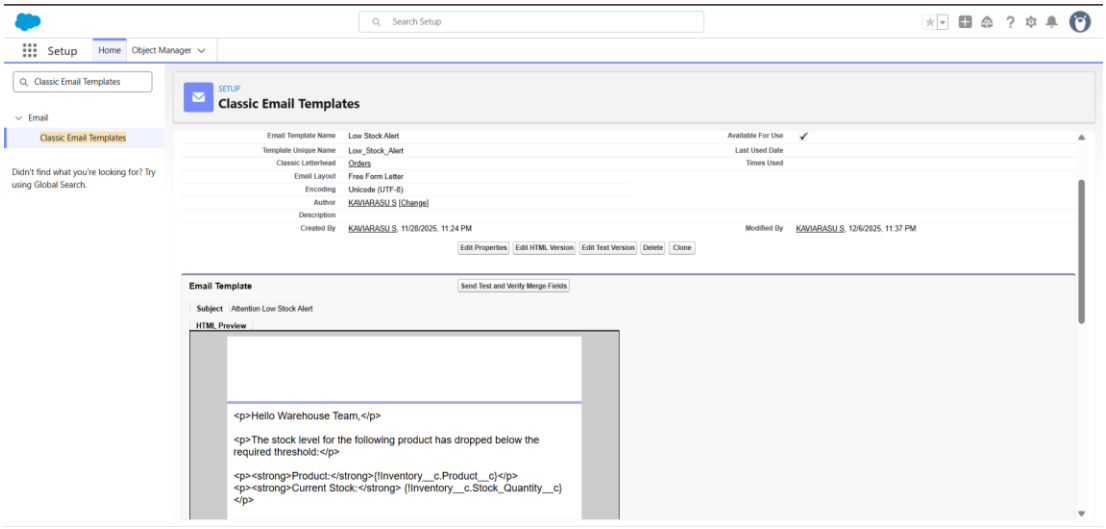
Step 4: Fill in Template Details

- **Folder:** Unfiled Public Email Templates
- **Available for Use:** Enable
- **Email Template Name:** Low_Stock_Alert_Email
- **Encoding:** UTF-8
- **Subject:** Low Stock Alert – Immediate Attention Required



Step 5: Add HTML Body

Low Stock Alert!
Product {!Inventory__c.Product__r.Product_Name__c} is running low.
Current Stock: {!Inventory__c.Current_Stock__c}
Please restock immediately.



Step 6: Save the Template

Email Template 3 – Loyalty Upgrade

Steps to Create the Loyalty Upgrade Email Template

Step 1: Go to Salesforce Setup

- Click the **Gear Icon** → Select **Setup**.

Step 2: Navigate to Classic Email Templates

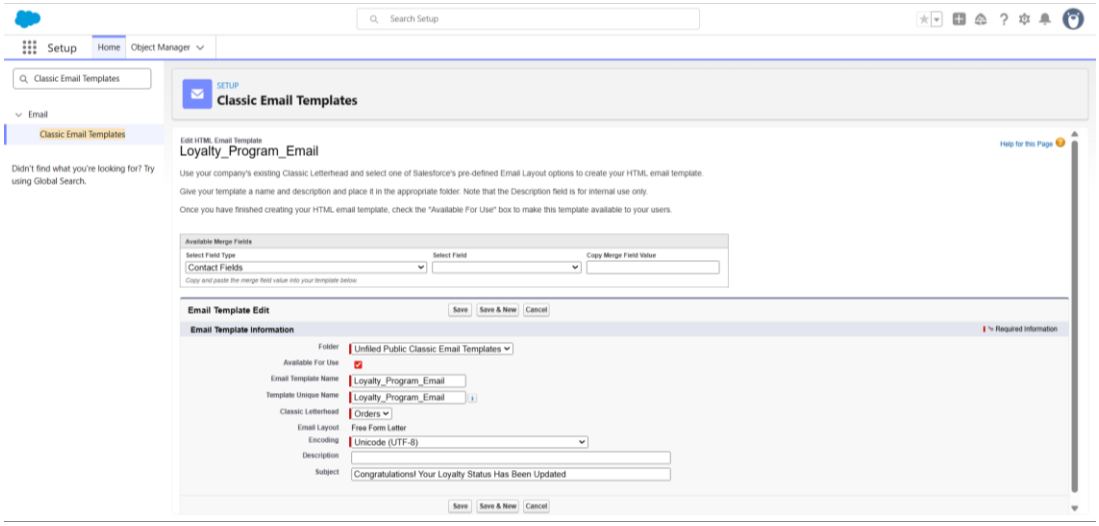
- Search **Classic Email Templates**.
- Click on it.

Step 3: Click "New Template"

- Select **HTML (with Classic Letterhead)**.
- Click **Next**.

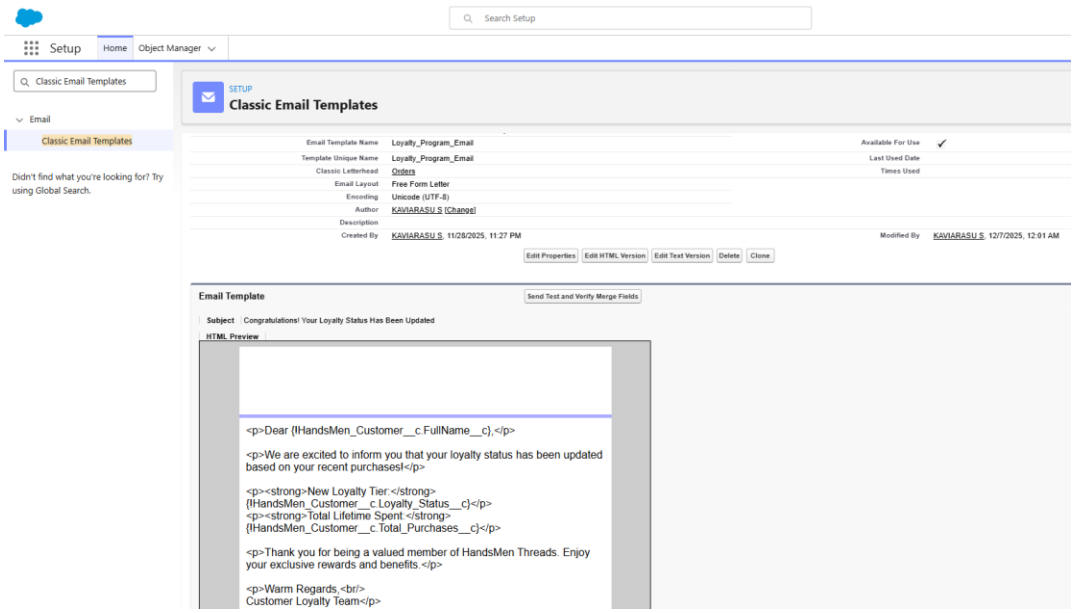
Step 4: Fill in Template Details

- **Folder:** Unfiled Public Email Templates
- **Available for Use:** Enable
- **Email Template Name:** Loyalty_Upgrade_Email
- **Encoding:** UTF-8
- **Subject:** Congratulations! Your Loyalty Status Has Been Upgraded



Step 5: Add HTML Body

```
<p>Dear {!HandsMen_Customer__c.Customer_Name__c},</p>
<p>Congratulations! Your loyalty status has been upgraded to
<strong>{!HandsMen_Customer__c.Loyalty_Status__c}</strong>.</p>
<p>Thank you for being a valued customer.</p>
<p>Regards,<br>
HandsMen Threads</p>
```



Step 6: Save the Template

Milestone 9: Automation Using Flows

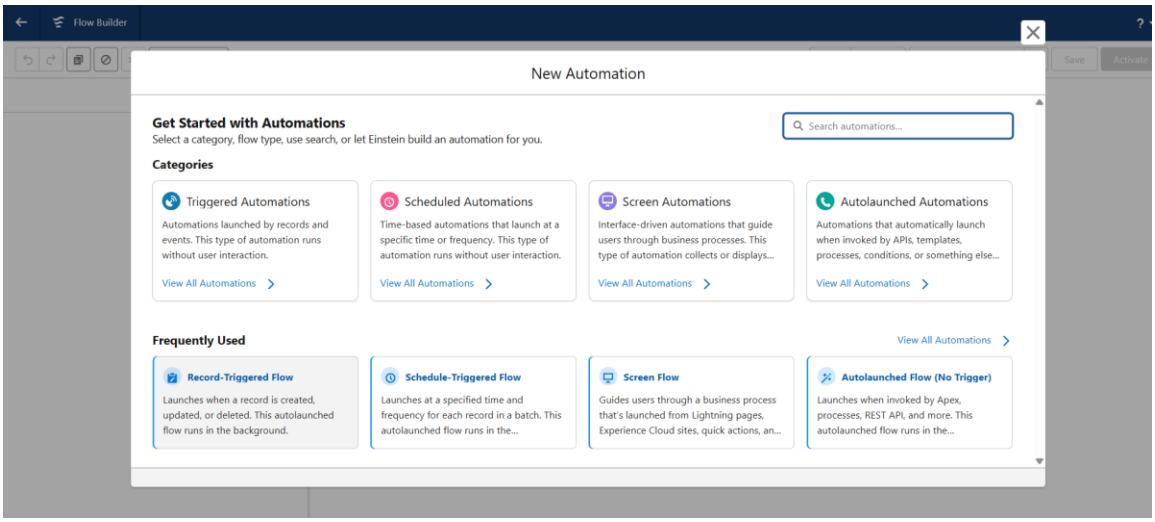
Flow 1: Order Confirmation Email Flow (Record-Triggered Flow)

Step 1: Go to Salesforce Setup

- Click the **Gear Icon** → select **Setup**.

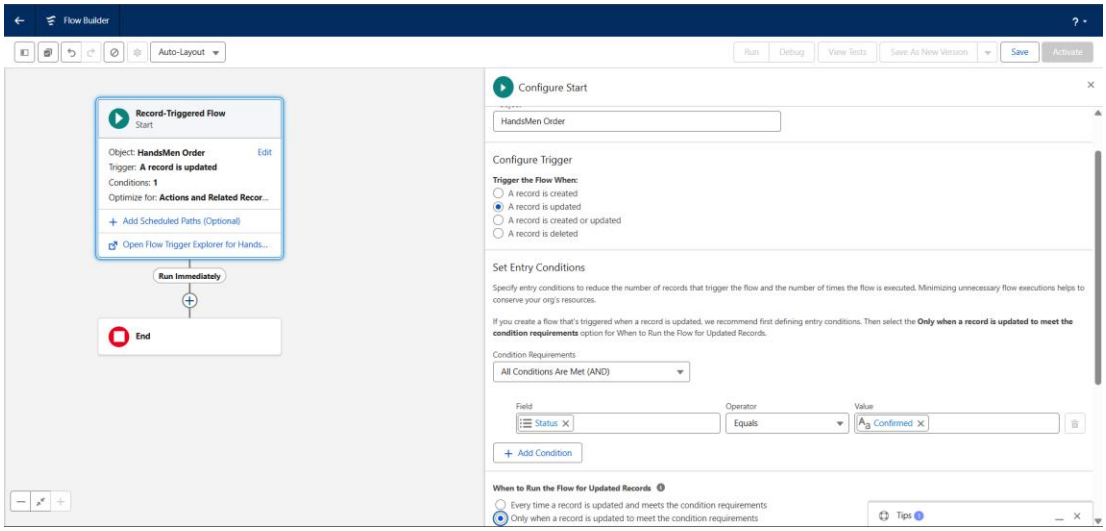
Step 2: Open Flow Builder

- In the **Quick Find** search bar, type **Flows**.
- Click **Flows**.
- Click **New Flow**.
- Select **Record-Triggered Flow** → Click **Create**.



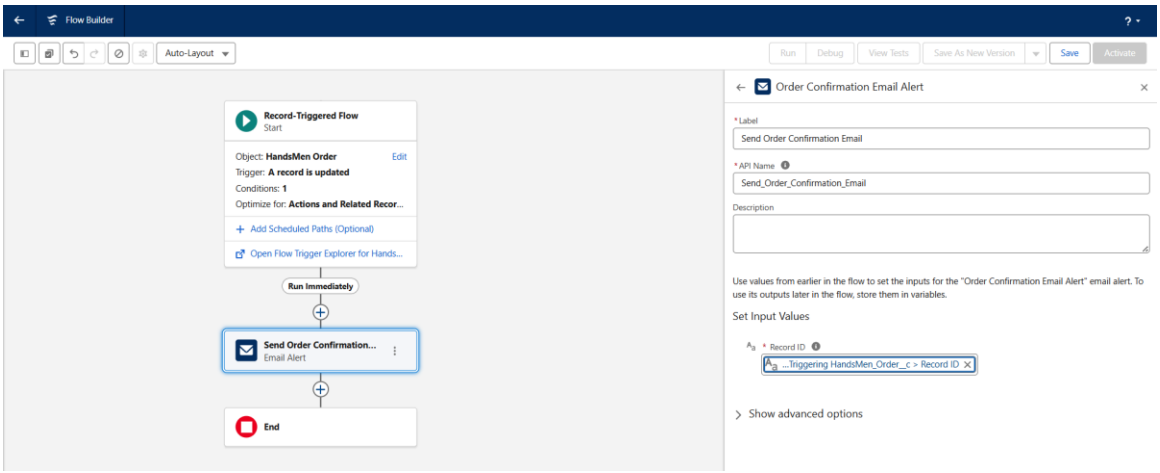
Step 3: Configure Trigger Details

- **Object:** HandsMen_Order__c
- **Trigger:** When a record is **updated**
- **Condition:**
  - Field: HandsMen_Order__c.Status__c
  - Operator: Equals
  - Value: "Confirmed"
- Select: **Only when a record is updated to meet the condition requirements**
- Click **Done**.



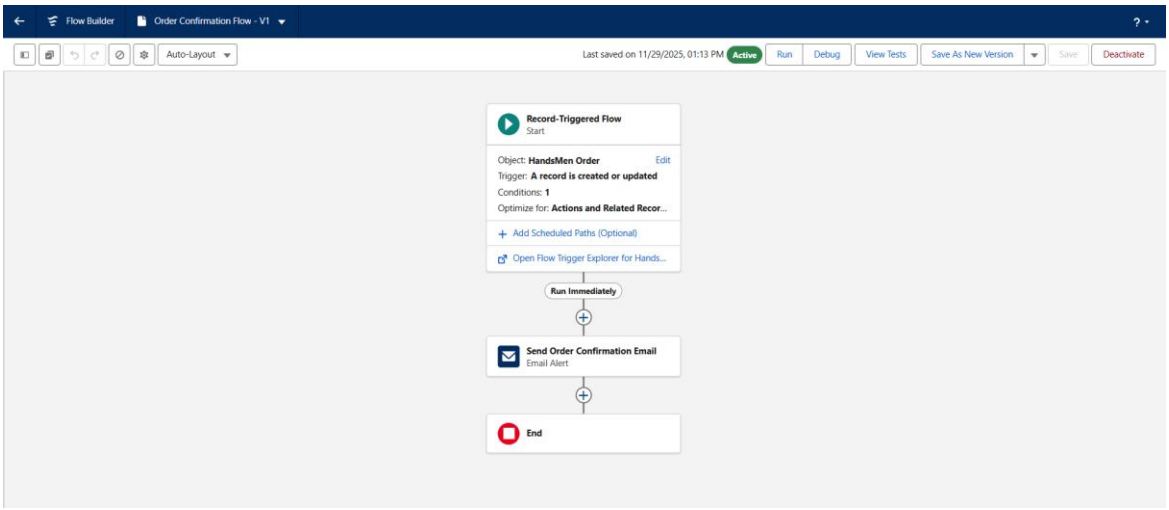
Step 4: Add an Email Alert Action

- Click the **"+"** icon.
- Select **Action**.
- **Action Type:** Send Email Alert
- **Email Alert:** Choose **Order Confirmation Email Alert**
- Fill the details:
  - **Label:** Send Order Confirmation Email
  - **Record ID:** {!$Record.Id}
- Click **Done**.



Step 5: Save & Activate

- **Flow Name:** Order Confirmation Email Flow
- Click **Save**
- Click **Activate**



Flow 2: Low Stock Alert Flow (Record-Triggered Flow)

Step 1: Go to Salesforce Setup

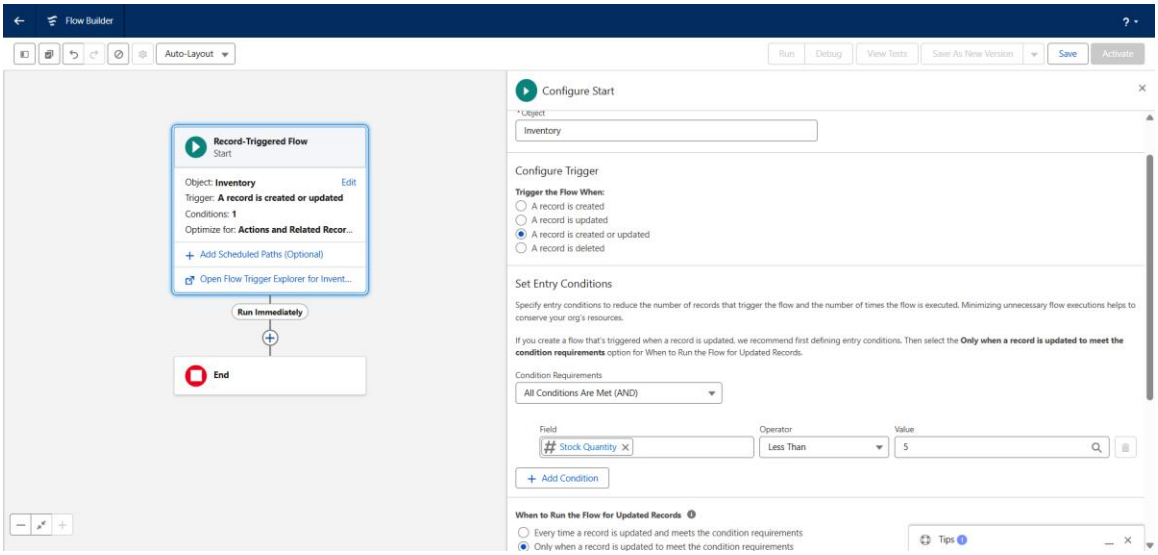
- Click **Setup** → Open **Flows** from Quick Find.

Step 2: Create a New Record-Triggered Flow

- Click **New Flow**
- Select **Record-Triggered Flow**
- Click **Create**

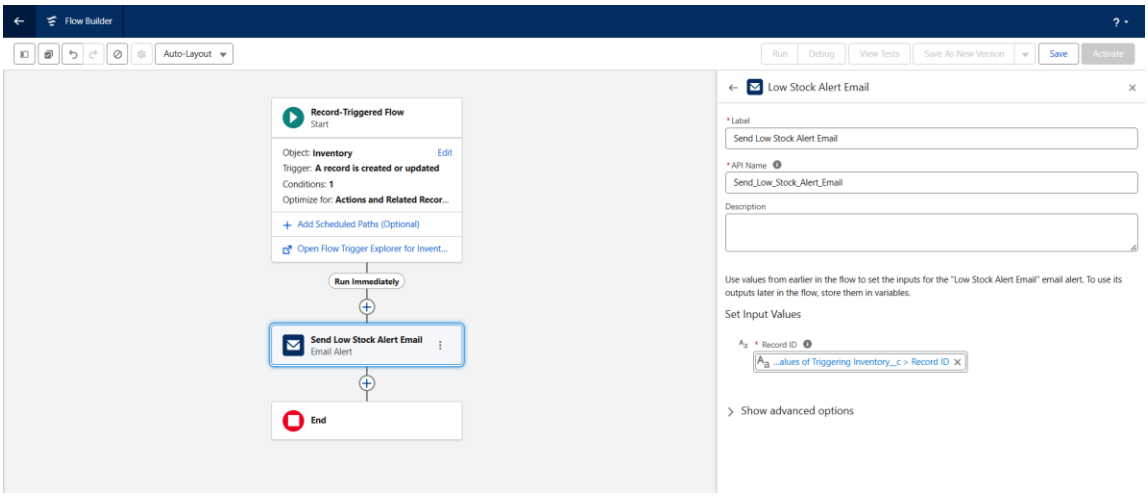
Step 3: Configure Trigger Details

- **Object:** Inventory__c
- **Trigger:** When a record is **created or updated**
- **Condition:**
  - Field: Inventory__c.Stock_Quantity__c
  - Operator: Less Than
  - Value: 5
- Select **Only when a record is updated to meet the condition**
- Click **Done**



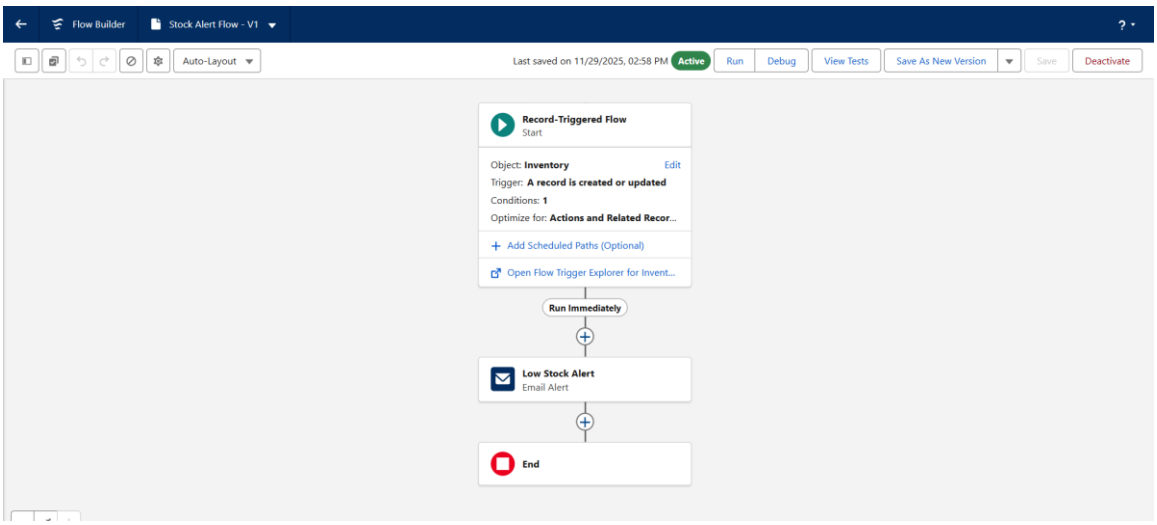
Step 4: Add "Send Email" Action

- Click the **"+"** icon → Select **Action**
- **Action Type:** Send Email Alert
- Select: **Low Stock Alert Email Alert**
- Enter:
  - **Label:** Send Low Stock Alert Email
  - **Record ID:** {!$Record.Id}
- Click **Done**



Step 5: Save & Activate

- **Flow Name:** Low Stock Alert Flow
- Click **Save**
- Click **Activate**



Flow 3: Loyalty Update Scheduled Flow (Scheduled Triggered Flow)

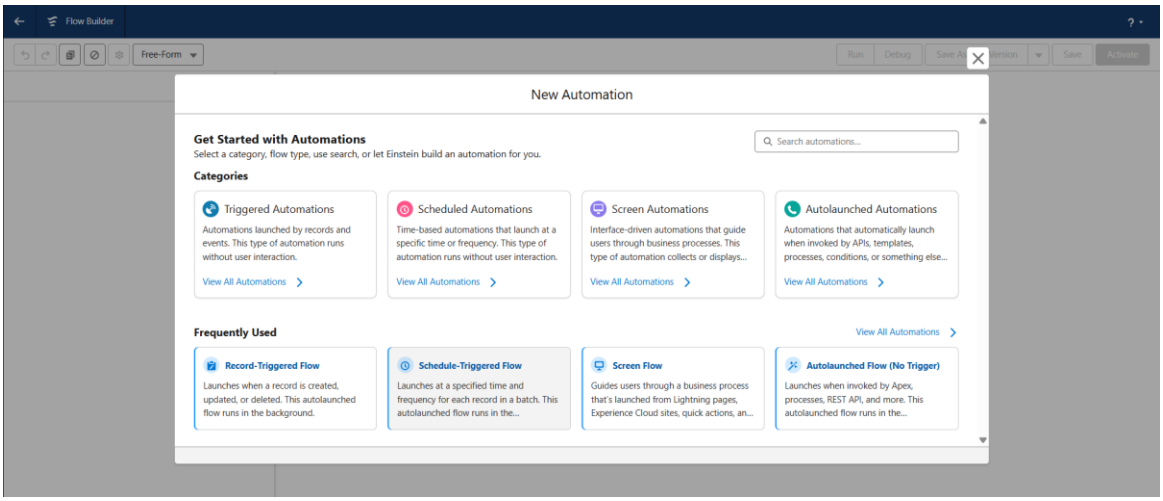
Step 1: Go to Salesforce Setup

- Click the **Gear Icon** → Choose **Setup**

Step 2: Open Flow Builder

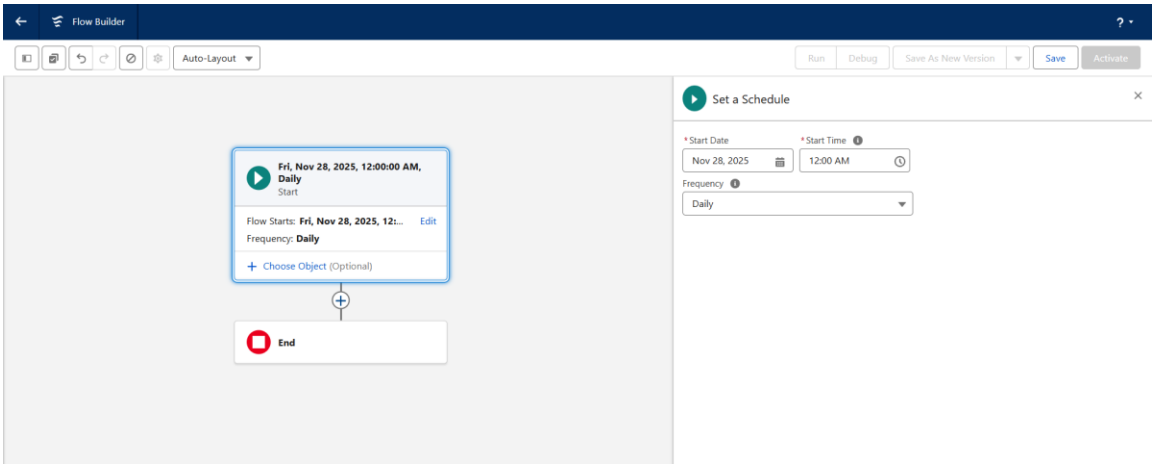
- Search **Flows** → Click **Flows**
- Click **New Flow**
- Select **Scheduled-Triggered Flow**

- Click **Create**



Step 3: Configure Schedule

- **Frequency:** Daily
- **Start Time:** 12:00 AM (Midnight)
- **Time Zone:** Your org default or IST
- Click **Done**



Step 4: Add a Get Records Element

This retrieves all customers whose loyalty must be recalculated.

- Click **"+"** → Select **Get Records**
- **Object:** HandsMen_Customer__c
- **Filter Condition:** None (retrieve all customers)
- **Store All Records:** Yes
- Click **Done**

Step 5: Add Assignment Logic (Optional)

If needed, add logic to process each customer:

- Loop through records
- Recalculate total purchase
- Assign loyalty levels (Bronze/Silver/Gold)

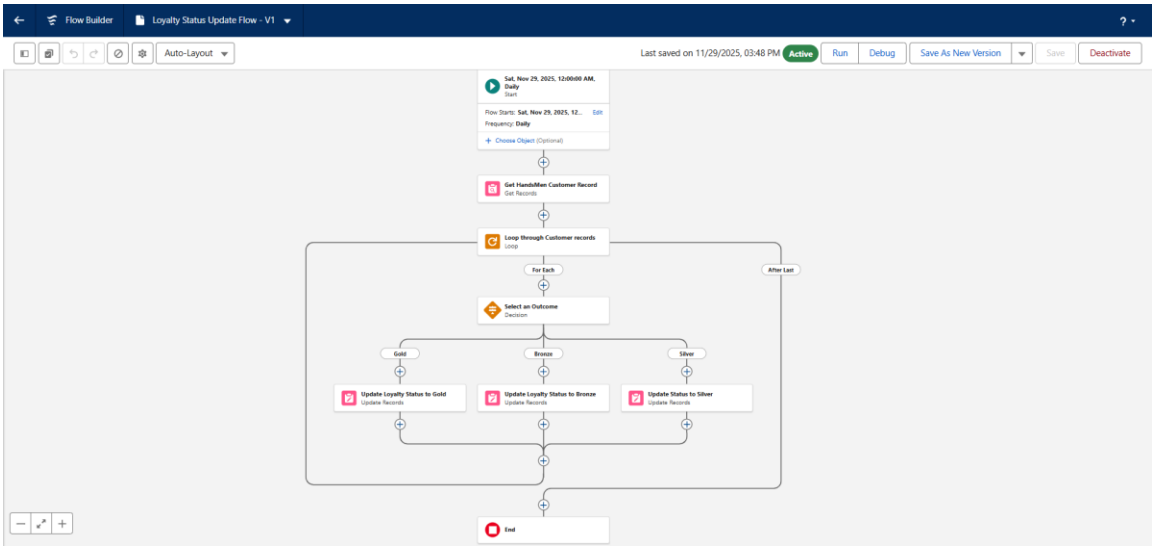
(Or replace with an Apex action if using your handler class.)

Step 6: Add an Update Records Element

- Click **"+"** → Select **Update Records**
- Choose **Records from the Loop / Collection**
- Update loyalty status fields
- Click **Done**

Step 7: Save & Activate

- **Flow Name:** Scheduled Loyalty Update Flow
- Click **Save**
- Click **Activate**



Milestone 10: Batch Apex (Inventory Restock)

Batch Job 1: Inventory

Activity 1: Create the Inventory Batch Apex Class

Step 1: Open Developer Console

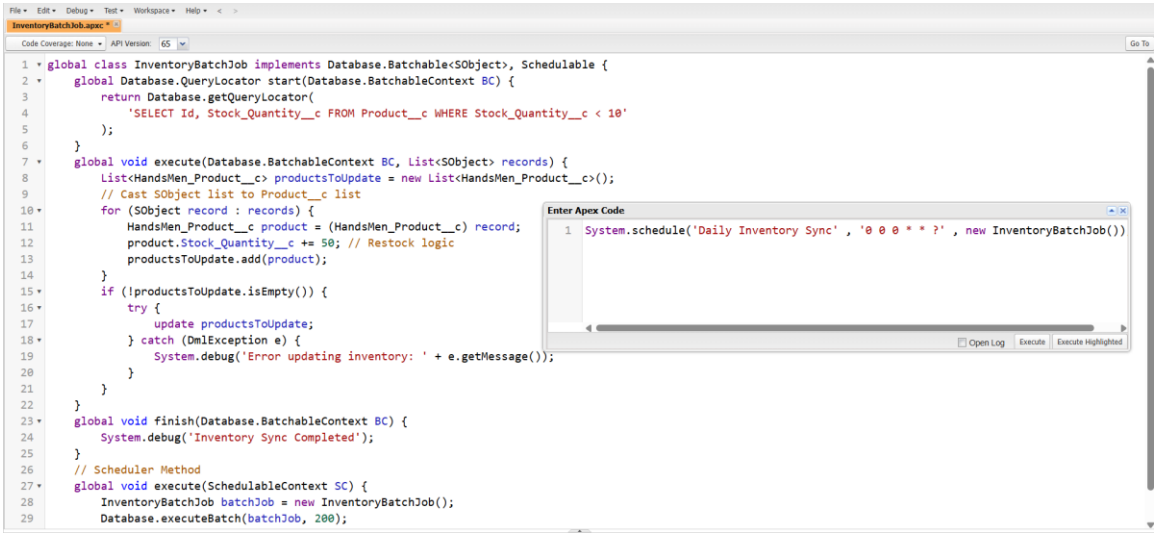
- Go to **Setup** → click **Gear Icon** → select **Developer Console**.
- A new window opens.

Step 2: Create a New Apex Class

- In Developer Console → click **File → New → Apex Class**.
- Enter class name: **InventoryBatchJob**.
- Click **OK**.

Step 3: Write the Code Logic

```
global class InventoryBatchJob implements Database.Batchable<SObject>, Schedulable {
   global Database.QueryLocator start(Database.BatchableContext BC) {
      return Database.getQueryLocator(
         'SELECT Id, Stock_Quantity__c FROM HandsMen_Product__c WHERE Stock_Quantity__c < 10'
      );
   }
   global void execute(Database.BatchableContext BC, List<SObject> records) {
      List<HandsMen_Product__c> productsToUpdate = new List<HandsMen_Product__c>();
      for (SObject record : records) {
         HandsMen_Product__c product = (HandsMen_Product__c) record;
         product.Stock_Quantity__c += 50; // Restock logic
         productsToUpdate.add(product);
      }
      if (!productsToUpdate.isEmpty()) {
         try {
            update productsToUpdate;
         } catch (DmlException e) {
            System.debug('Error updating inventory: ' + e.getMessage());
         }
      }
   }
   global void finish(Database.BatchableContext BC) {
      System.debug('Inventory Sync Completed');
   }
   // Scheduler Method
   global void execute(SchedulableContext SC) {
      InventoryBatchJob batchJob = new InventoryBatchJob();
      Database.executeBatch(batchJob, 200); } }
```



Step 4: Save the Class

- Click **File → Save**.

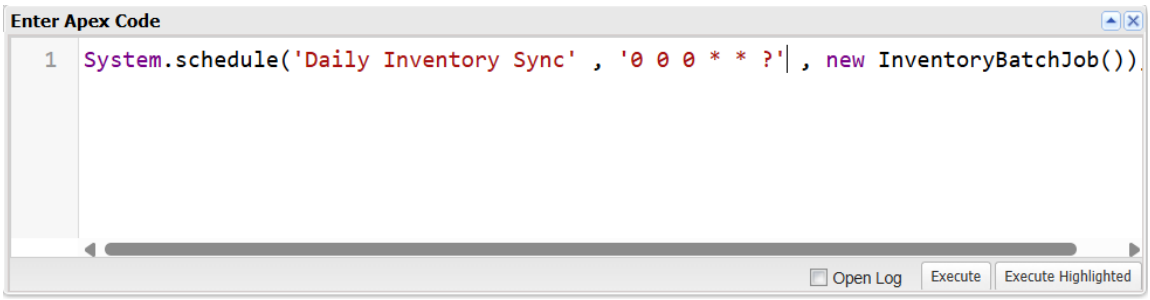
Activity 2: Schedule the Inventory Sync Batch Job (Using Execute Anonymous)

Step 1: Open Execute Anonymous Window

- In Developer Console → click **Debug → Open Execute Anonymous Window**.

Step 2: Paste the Scheduling Code

```
// Schedule Inventory Sync daily at 2:00 AM
System.schedule('Inventory Sync', '0 0 2 * * ?', new InventoryBatchJob());
```



Step 3: Execute

- Click **Execute**.
- The batch job is now scheduled.

Batch Job 2: Loyalty Points Calculation

Activity 1: Create the Loyalty Batch Apex Class

Step 1: Open Developer Console

- Setup → **Developer Console**.

Step 2: Create a New Apex Class

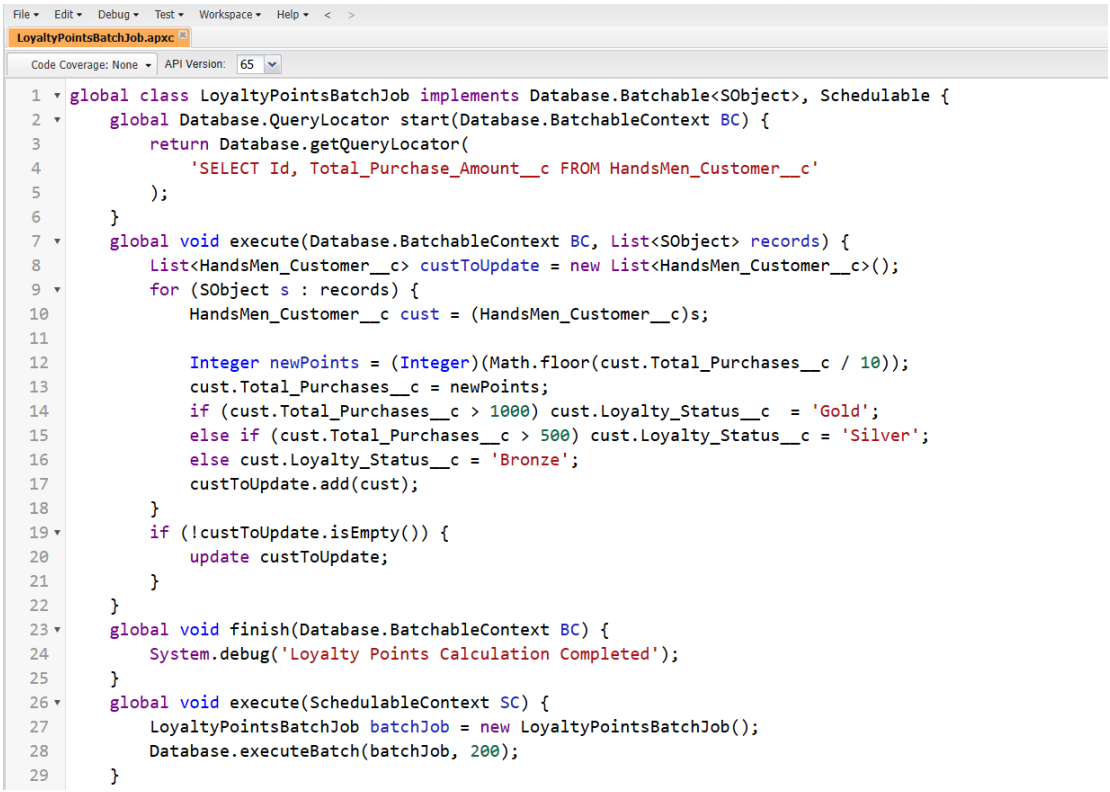
- File → New → Apex Class
- Class Name: **LoyaltyPointsBatchJob**
- Click **OK**

Step 3: Paste the Code Logic

```
global class LoyaltyPointsBatchJob implements
Database.Batchable<SObject>, Schedulable {
   global Database.QueryLocator start(Database.BatchableContext BC) {
      return Database.getQueryLocator(
         'SELECT Id, Total_Purchase_Amount__c FROM
HandsMen_Customer__c'
      );
   }
global void execute(Database.BatchableContext BC, List<SObject>
records) {
      List<HandsMen_Customer__c> custToUpdate = new
List<HandsMen_Customer__c>();
      for (SObject s : records) {
         HandsMen_Customer__c cust = (HandsMen_Customer__c)s;

         Integer newPoints = Math.floor(cust.Total_Purchase_Amount__c /
10);
         cust.Loyalty_Points__c = newPoints;
         if (cust.Total_Purchase_Amount__c > 1000)
cust.Loyalty_Status__c  = 'Gold';
         else if (cust.Total_Purchase_Amount__c > 500)
cust.Loyalty_Status__c = 'Silver';
         else cust.Loyalty_Status__c = 'Bronze';
         custToUpdate.add(cust);
      }
      if (!custToUpdate.isEmpty()) {
         update custToUpdate;
      }
```

```
    }
    global void finish(Database.BatchableContext BC) {
        System.debug('Loyalty Points Calculation Completed');
    }
    global void execute(SchedulableContext SC) {
        LoyaltyPointsBatchJob batchJob = new LoyaltyPointsBatchJob();
        Database.executeBatch(batchJob, 200);
    }
}
```



Step 4: Save the Class

- Click **File → Save**

Activity 2: Schedule the Loyalty Points Batch Job (Using Execute Anonymous)

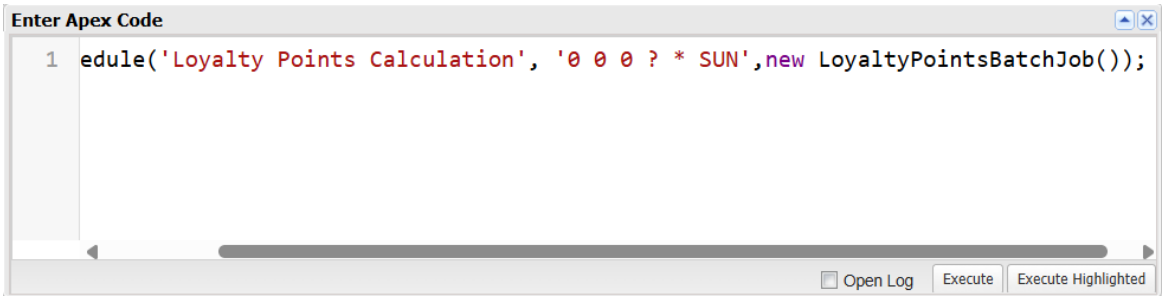
Step 1: Open Execute Anonymous

- Developer Console → **Debug → Open Execute Anonymous Window**

Step 2: Paste the Scheduling Code

// Schedule Loyalty Points Calculation every Sunday at 12:00 AM

System.schedule('Loyalty Points Calculation', '0 0 0 ? * SUN',new LoyaltyPointsBatchJob());



Step 3: Execute

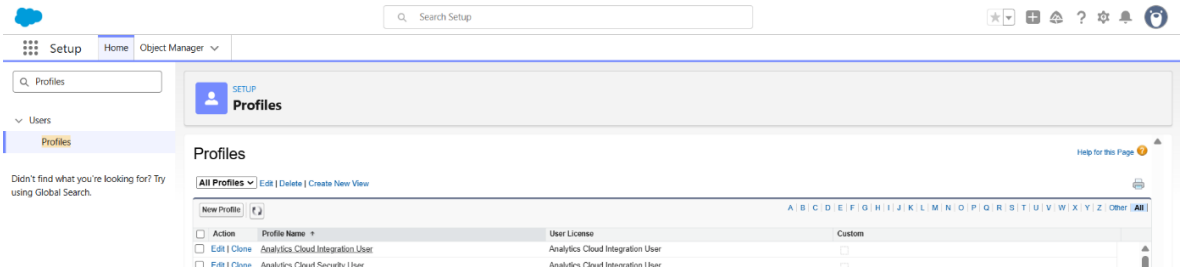
- Click **Execute**
- Weekly batch job is now scheduled

Milestone 11: Profiles, Roles & Permission Sets

1. Profile Creation

Step 1: Navigate to Profiles

1. Go to **Setup**
2. In the **Quick Find** box, type **Profiles**
3. Click **Profiles**

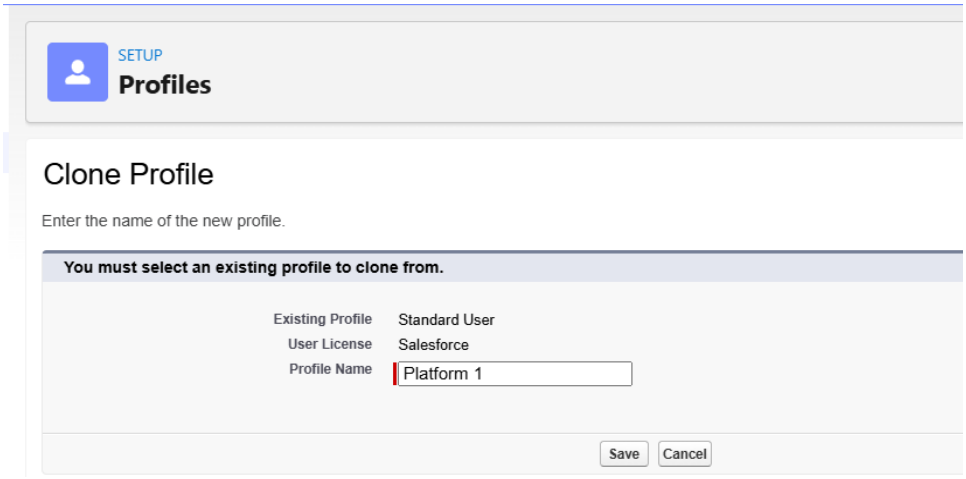


Step 2: Clone a Standard Profile

1. Find the profile **Standard User**
2. Click **Clone**

Step 3: Enter Profile Details

1. **Profile Name:** Platform 1
2. Click **Save**



Step 4: Edit Permissions for the Profile

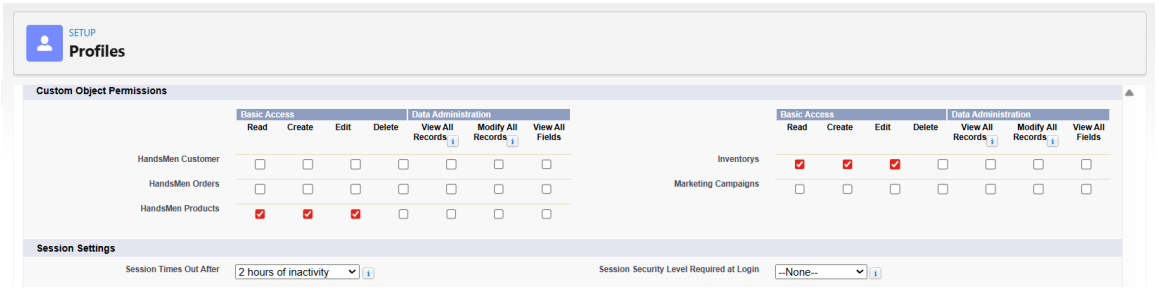
1. You will remain on the newly created **Platform 1** profile page
2. Click **Edit**
3. Scroll down to **Custom Object Permissions**

Step 5: Assign Required Object Permissions

Give the following permissions:

Object	**Permissions**
HandsMen Product	Read, Create, Edit, Delete
Inventory	Read, Create, Edit, Delete

(Select all CRUD checkboxes for these two objects.)



Step 6: Save

1. Scroll down
2. Click **Save**

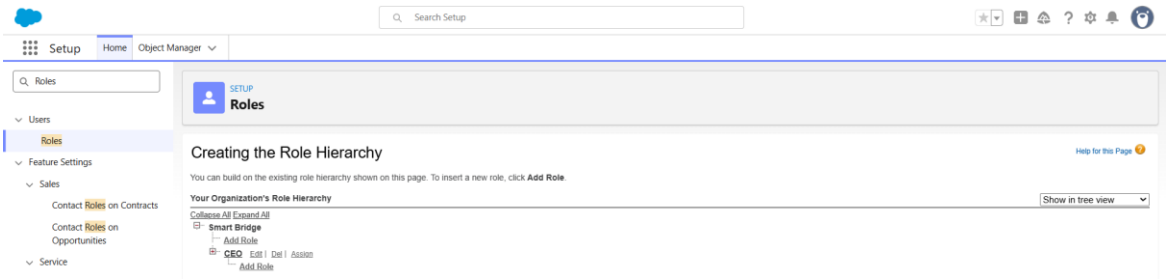
1. Creating Roles

Step 1: Navigate to Roles

- Go to **Setup**
- In the Quick Find box, type **Roles**
- Click **Set Up Roles**

Step 2: Expand Role Hierarchy

- Click **Expand All**

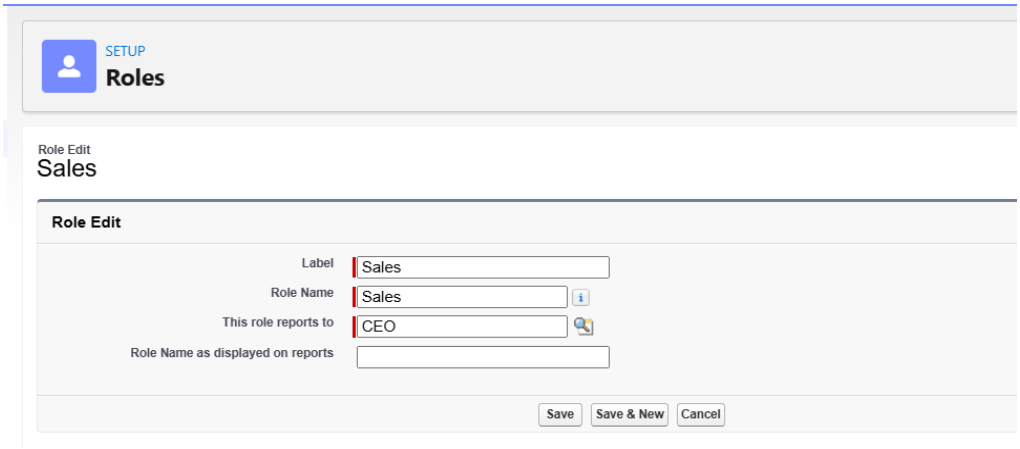


Step 3: Create CEO Role (If not present)

- Click **Add Role**
- Enter:
  - **Label:** CEO
  - **Role Name:** auto populated
- Click **Save**

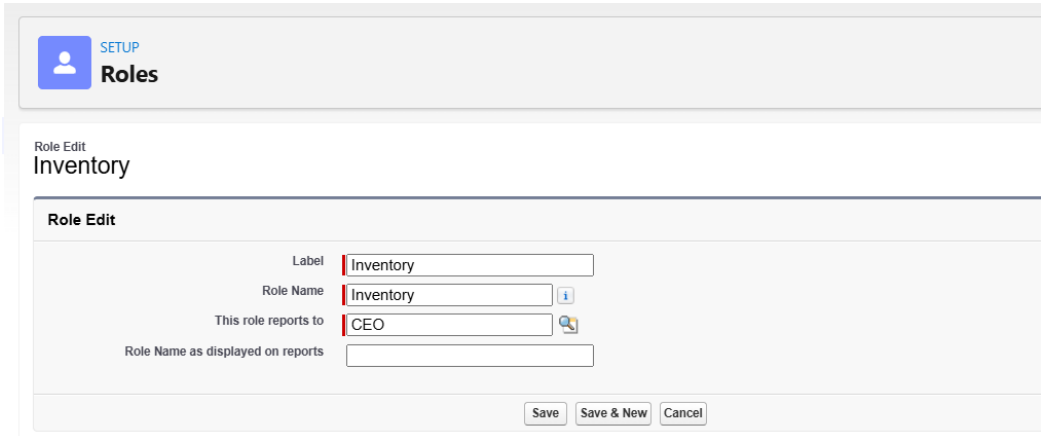
Step 4: Create Sales Role

- Under CEO → Click **Add Role**
- Enter:
  - **Label:** Sales
  - **Role Name:** auto populated
  - **Reports To:** CEO
- Click **Save**



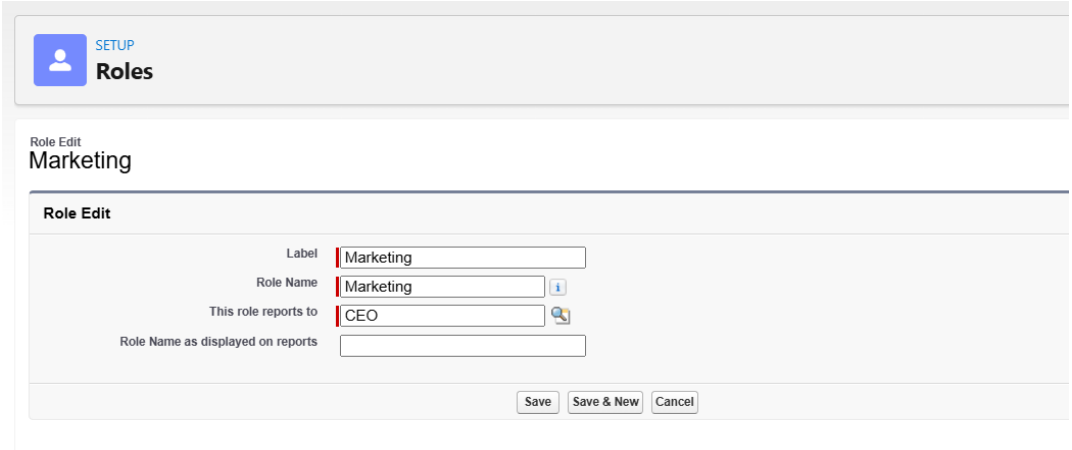
Step 5: Create Inventory Role

- Under CEO → click **Add Role**
- Enter:
  - **Label:** Inventory
  - **Reports To:** CEO
- Save



Step 6: Create Marketing Role

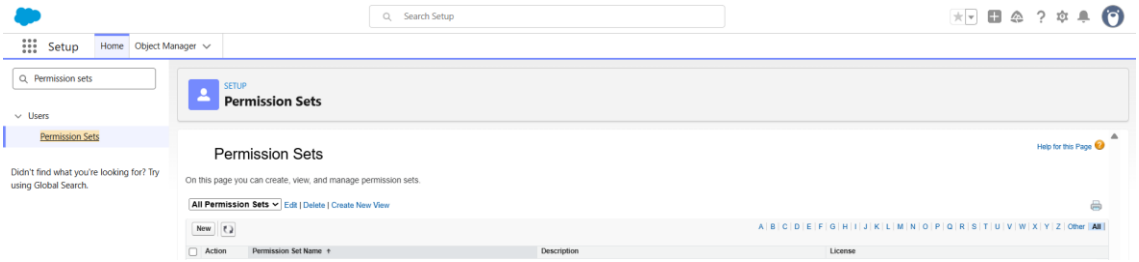
- Under CEO → click **Add Role**
- Enter:
  - **Label:** Marketing
  - **Reports To:** CEO
- Save



2. **Creating Permission Set: Permission_Platform_1**

Step 1: Navigate to Permission Sets

- Go to **Setup**
- In Quick Find type **Permission Sets**
- Click **Permission Sets**



Step 2: Create a New Permission Set

- Click **New**
- Enter:
  - **Label:** Permission_Platform_1
  - **API Name:** auto populated
- Click **Save**

Step 3: Configure Object Permissions

- Under **Apps** → click **Object Settings**

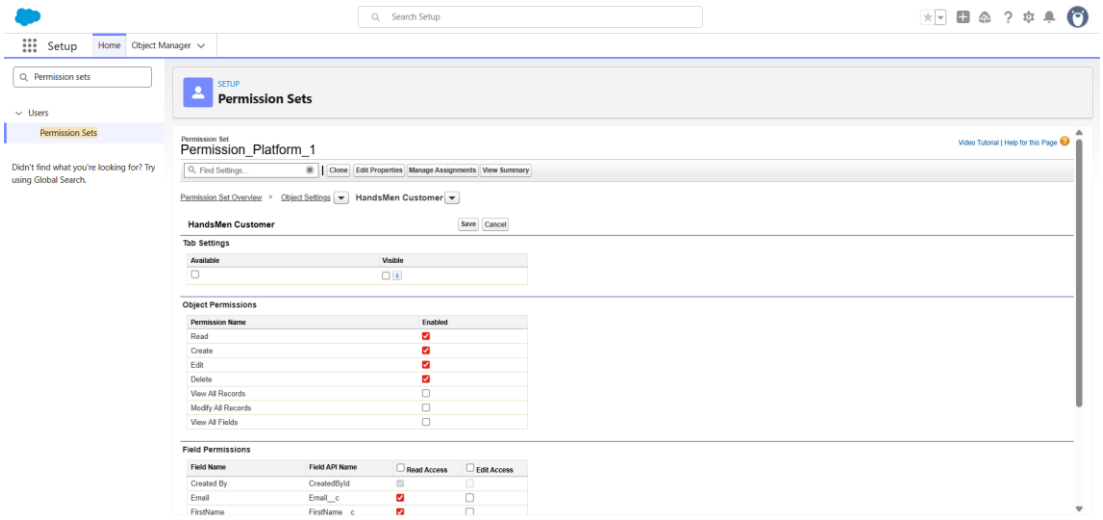
Step 4: Give Full CRUD to Custom Objects

HandsMen Customer

- Click **HandsMen Customer**
- Click **Edit**
- Enable:
  - Read

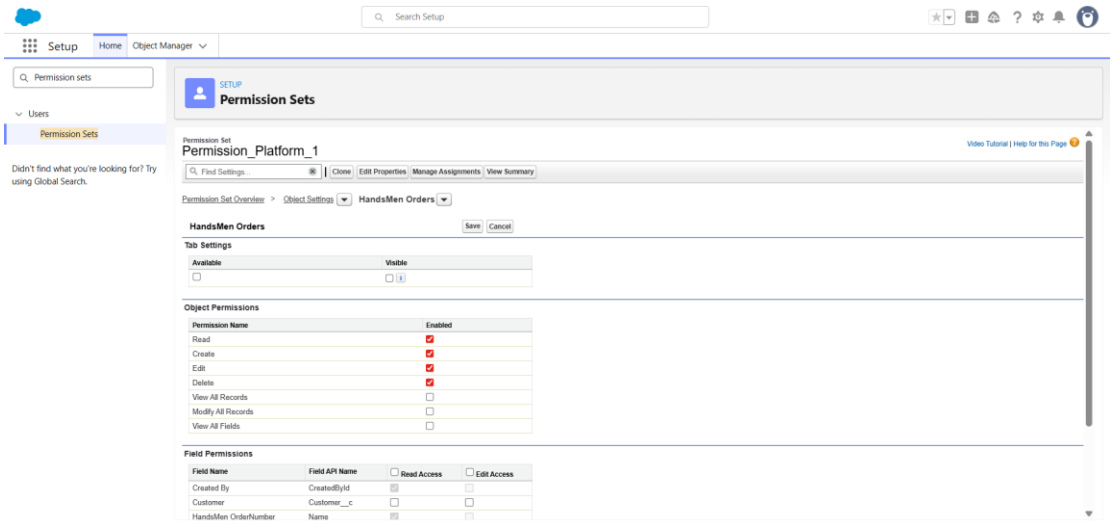
- Create
- Edit
- Delete

- Click **Save**



HandsMen Order

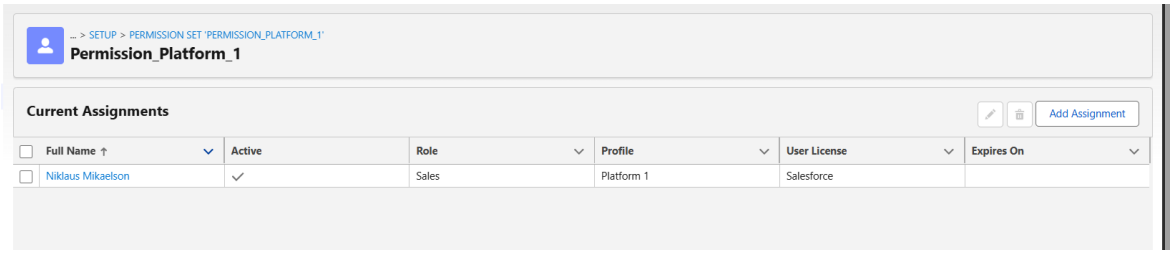
- Go back to Object Settings
- Click **HandsMen Order**
- Click **Edit**
- Enable:
  - Read
  - Create
  - Edit
  - Delete
- Click **Save**



(Repeat for other custom objects if needed.)

Step 5: Assign Permission Set to User

- On the Permission Set page → click **Manage Assignments**
- Click **Add Assignments**
- Select any user who is using the **Sales Profile / Inventory Profile / Platform Profile**
- Click **Next**
- Click **Assign**
- Click **Done**



Data Security Model Summary:

Role	Access Level
Sales Manager	Full access to Customers & Orders
Inventory Manager	Read & Edit on Inventory and Products
Marketing Team	Read Customers, Edit Marketing Campaigns
System Admin	Full access to all objects

This role hierarchy ensures proper data governance and limited access according to business needs.

Milestone 13: Users

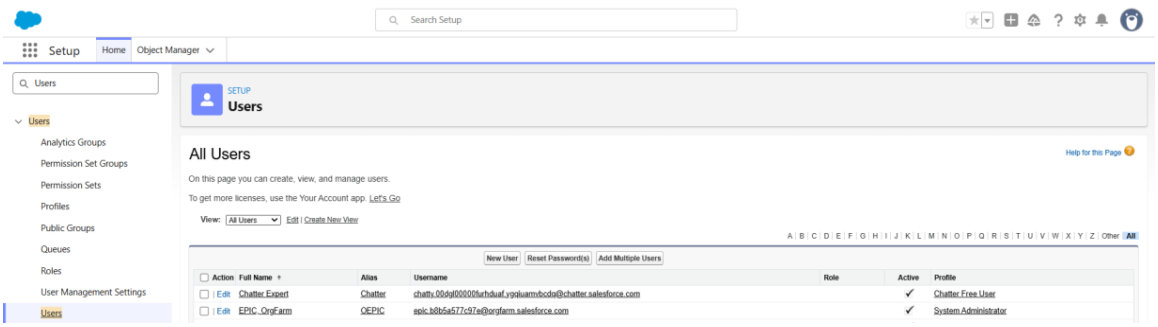
Activity 1: Create User – Sales Team Member

Step 1: Navigate to Users

- Go to **Setup**
- In the **Quick Find** search bar, type **Users**
- Click on **Users**

Step 2: Create a New User

- Click **New User**



Step 3: Fill in User Details

Enter the following information:

- **First Name:** Niklaus

- **Last Name:** Mikaelson
- **Alias:** *(Enter any short alias)*
- **Email:** *(Enter your personal email address)*
- **Username:** Example: niklaus.mikaelson@test.com
- **Nickname:** *(Enter a preferred nickname)*

Role Assignment

- **Role:** Sales

User License

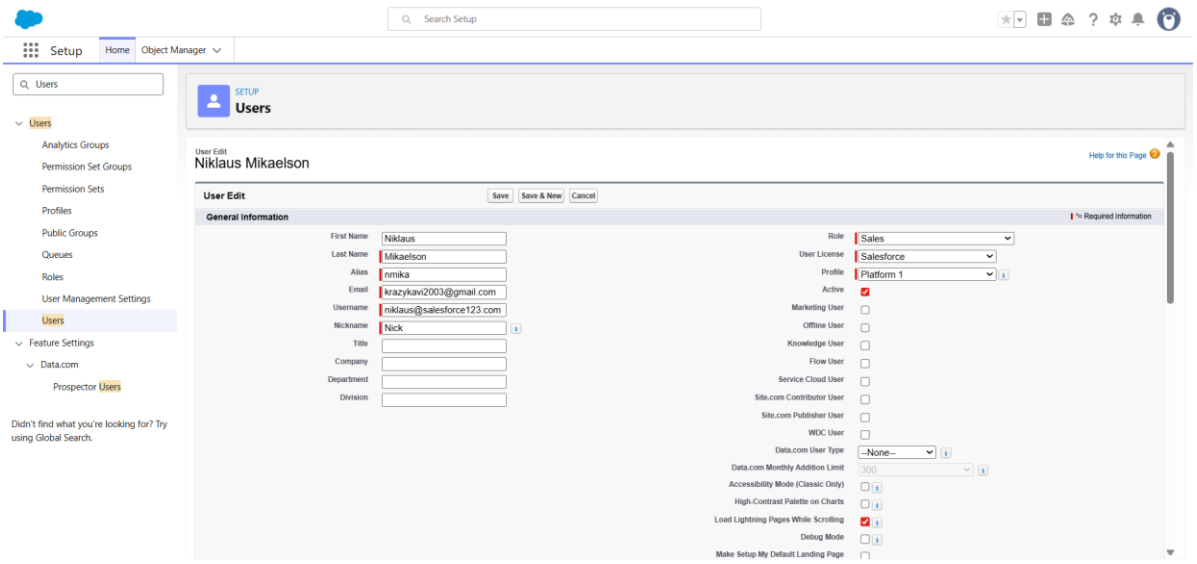
- **User License:** Salesforce Platform

Profile

- **Profile:** Platform 1

Step 4: Save

- Scroll down and click **Save**



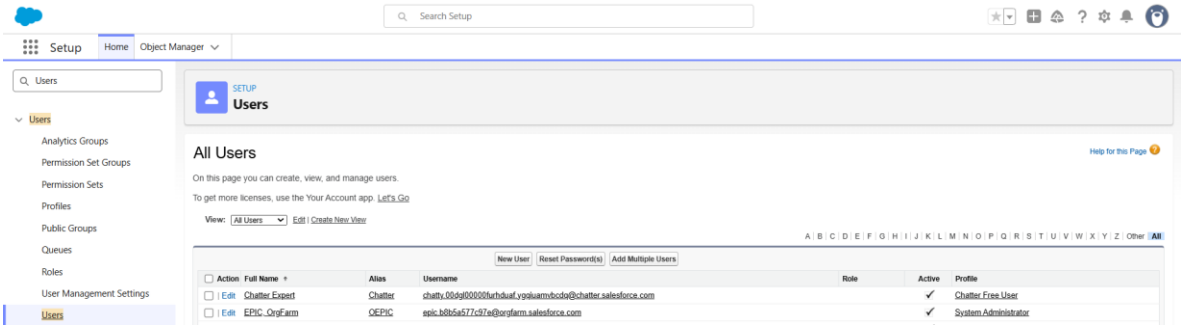
Activity 2: Create User – Inventory Team Member

Step 1: Navigate to Users

- Go to **Setup**
- In **Quick Find**, type **Users**
- Click **Users**

Step 2: Create a New User

- Click **New User**



Step 3: Fill in User Details

Enter the following information:

- **First Name:** Kol
- **Last Name:** Mikaelson
- **Alias:** *(Enter a short alias)*
- **Email:** *(Enter your personal email address)*
- **Username:** kol.mikaelson@test.com
- **Nickname:** *(Enter a preferred nickname)*

Role Assignment

- **Role:** Inventory

User License

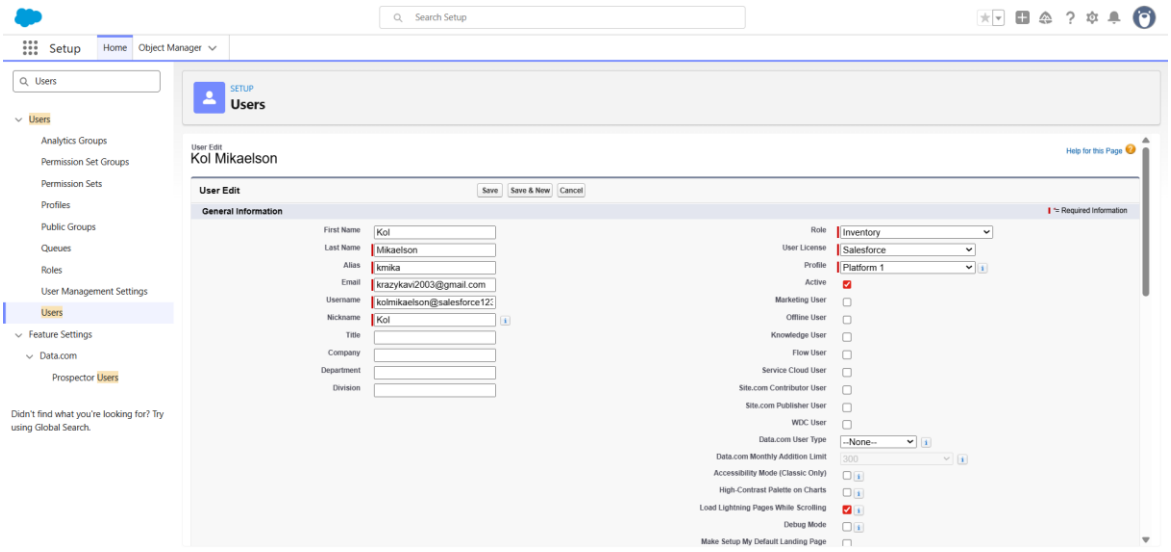
- **User License:** Salesforce Platform

Profile

- **Profile:** Platform 1

Step 4: Save

- Click **Save**


Search Setup
Setup
Home
Object Manager
Users
Analytics Groups
Permission Set Groups
Permission Sets
Profiles
Public Groups
Queues
Roles
User Management Settings
Users
Feature Settings
Data.com
Prospector Users
Didn't find what you're looking for? Try using Global Search.
SETUP
Users
User Edit
Kol Mikaelson
Help for this Page
User Edit
Save
Save & New
Cancel
General Information
= Required Information
First Name
Kol
Last Name
Mikaelson
Alias
kmika
Email
krazykavi2003@gmail.com
Username
kolmikaelson@salesforce12
Nickname
Kol
Title
Company
Department
Division
Role
Inventory
User License
Salesforce
Profile
Platform 1
Active
Marketing User
Offline User
Knowledge User
Flow User
Service Cloud User
Site.com Contributor User
Site.com Publisher User
WDC User
Data.com User Type
--None--
Data.com Monthly Addition Limit
300
Accessibility Mode (Classic Only)
High-Contrast Palette on Charts
Load Lightning Pages While Scrolling
Debug Mode
Make Setup My Default Landing Page

PHASE 3: UI/UX Development & Customization

Phase 3 focuses on building a polished, user-friendly, and highly efficient user interface For the HandsMen Threads CRM system.
This includes:

- Lightning App creation
- Tabs and navigation configuration
- Page Layout customization

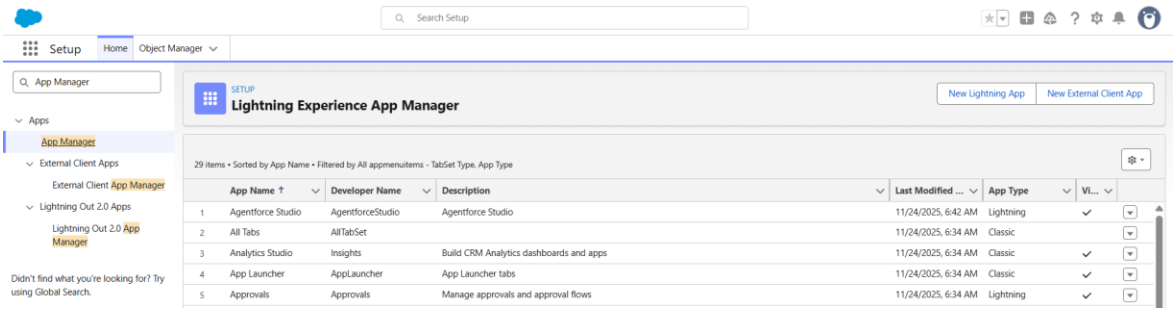
The goal is to ensure that users from different departments—Sales, Inventory, and Marketing—can operate the system smoothly with minimal clicks and maximum clarity.

Milestone 12: Creating the Lightning App

Activity 1: Create the "HandsMen Threads CRM" Lightning App

Steps:

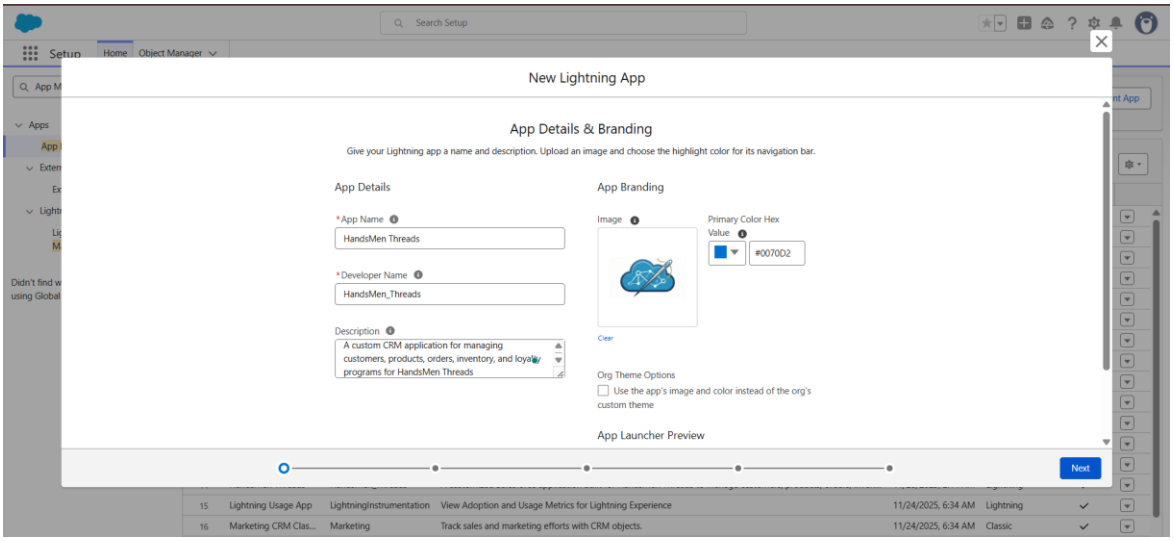
1. Go to **Setup → App Manager**
2. Click **New Lightning App**



App Details & Branding

- **App Name:** HandsMen Threads
- **Developer Name:** Auto-generated
- **Description:**
  A custom CRM application for managing customers, products, orders, inventory, and loyalty programs for HandsMen Threads.
- **Branding Settings:**
  - Logo (Optional): Upload the HandsMen Threads brand image
  - Primary Color: Default Salesforce color

Click **Next**



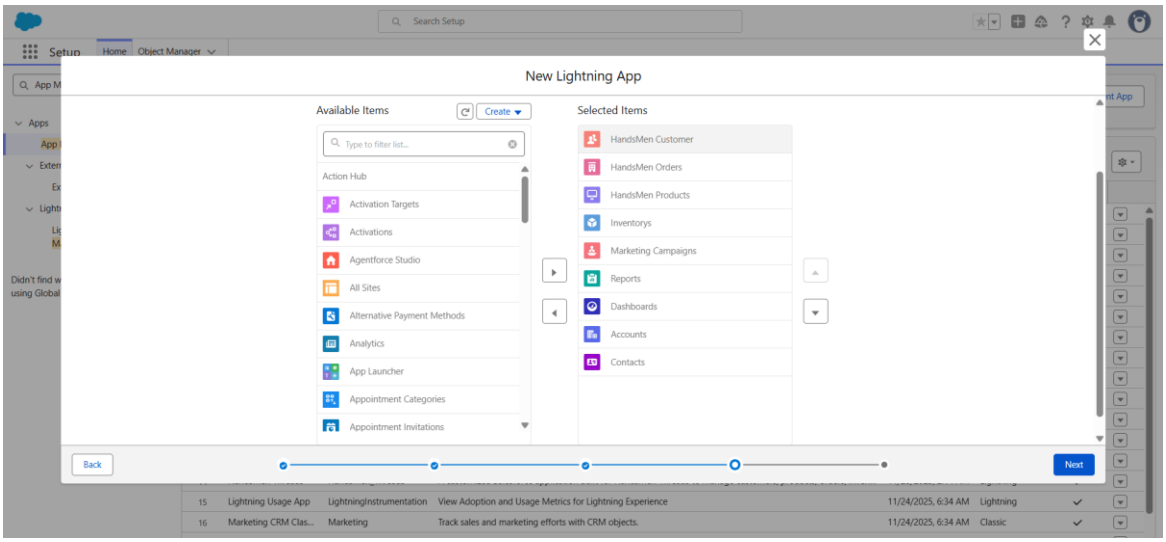
- **App Options**
  Keep all defaults:
    - Setup Utility Bar
    - Auto-Open Tabs
    - App Personalization

  Click **Next**
- **Utility Bar (Keep Default)**
  No additional utilities required
  Click **Next**

Navigation Items

Navigation Item	Object/Feature
HandsMen Customer	Custom Object
HandsMen Product	Custom Object
HandsMen Order	Custom Object
Inventory	Custom Object
Marketing Campaign	Custom Object
Reports	Standard

Dashboards	Standard
Contacts	Standard
Accounts	Standard



Assign to Profiles

Select:

- **System Administrator**
- Sales Profile
- Inventory Profile
- Marketing Profile

Click **Save & Finish**

PHASE 4: Testing & Security

Testing

Testing ensured that all **objects, Flows, Apex triggers, email alerts, batch jobs, and UI components** of the HandsMen Threads CRM performed correctly and integrated seamlessly with one another.

Unit Testing

- Verified Apex triggers and handler classes for:
  - **Order Total Calculation**
  - **Stock Deduction**
  - **Loyalty Status Update**
- Validated record-triggered Flows for:
  - Order Confirmation Email
  - Low Stock Alert
  - Scheduled Loyalty Update
- Checked **best practices and governor limits** (no SOQL in loops, bulkified logic, optimized queries).
- Ensured Apex test classes achieved **required code coverage** for all triggers, handlers, and batch classes.
- Confirmed formula fields and validation rules behaved correctly for different test inputs.

Integration Testing

- Verified relationships between:
  - **HandsMen Customer** → HandsMen Order
  - **HandsMen Product** → Inventory
  - **HandsMen Order** → Loyalty Log
- Tested interaction between:
  - Flows and Apex logic
  - Page layouts, related lists, and dynamic forms
  - Email Alerts triggered through Flow automation
- Simulated complete order lifecycle:
  - Create Order → Confirm Order → Stock updates → Loyalty recalculated
- Ensured data integrity across objects after automation execution.

User Acceptance Testing (UAT)

Conducted with sample users representing:

- Sales Team
- Inventory Team
- Marketing Team

UAT Scenarios Executed:

- Adding new customers
- Creating and confirming orders
- Verifying inventory reduction
- Triggering Low Stock Alert flow
- Auto-updating customer loyalty tier
- Reviewing dashboards and reports

Collected feedback and optimized:

- Page layouts
- Field placement
- Validation messages
- Navigation flow and tab arrangement
- Dynamic Forms visibility rules

UAT Outcome

- System easy to use
- Automation accurate
- Layouts user-friendly
- Dashboard insights clear

Security Implementation

Security configuration ensures that only authorized users (Sales, Inventory, Marketing, and Admin teams) can access the correct level of data within the HandsMen Threads CRM. Salesforce's multi-layered security model—Profiles, Roles, Permission Sets, OWD, Sharing Rules, and FLS—was implemented to protect sensitive business information and enforce proper access control.

Role Hierarchy

The role hierarchy was designed to reflect business reporting structure and control record-level visibility:

- **CEO (Top Level)** – Full visibility into all records
- **Sales** – Access to customer and order records
- **Inventory** – Access to products and stock-related records

- **Marketing** – Access to loyalty-related records

Purpose:

Higher roles inherit visibility from child roles, ensuring leadership can oversee all operations while teams only access their functional areas.

Profiles & Permission Sets

Profiles Created:

- **Sales Profile** – Manage customers & orders
- **Inventory Profile** – Manage products & inventory
- **Marketing Profile** – Manage loyalty logs
- **System Administrator** – Full access

Each profile controls:

- Object permissions (CRUD)
- Tab access
- Page layout access
- App access
- Field-level permission visibility

Permission Set: Permission_Platform_1

Additional access granted to selected users:

- Full CRUD access to:
    - HandsMen Customer
    - HandsMen Product
    - HandsMen Order
    - Inventory
    - Loyalty Log

Used to elevate permissions beyond profile capabilities without creating multiple profiles.

Field-Level Security (FLS)

Sensitive fields were protected to ensure confidentiality and prevent unnecessary exposure.

- Customer personal data (Phone, Email, Address) → Visible to Sales & Admin; limited for Marketing
- Inventory cost/value fields → Visible to Inventory & Admin; hidden from Sales
- Loyalty-related internal fields → Visible to Marketing & Admin only

Purpose:
Ensures users only see fields relevant to their job roles, preventing accidental data access or modification.

Record-Level Security

Record access was enforced using Organization-Wide Defaults (OWD), Role Hierarchy, and Sharing Rules.

OWD Settings

Object	OWD Setting	Reason
HandsMen Customer	Private	Protect customer personal information
HandsMen Order	Private	Limit access to sales team only
Inventory	Private	Inventory data is internal and sensitive
HandsMen Product	Public Read Only	Price & details visible, stock restricted
Marketing Campaign	Private	Confidential customer reward information

Sharing Rules

To allow controlled access:

Sales Team

- Read/Write access to **all Orders**
- Read-only access to **Customers**

Inventory Team

- Read/Write access to **Inventory**
- Read-only access to **Products**

Marketing Team

- Read/Write access to **Loyalty Log**
- Read-only access to **Customers**

Purpose of Sharing Rules

- Provide extended access without changing OWD or profiles
- Allow collaboration between teams without compromising confidentiality

Milestone 14: Performance Optimization

Performance Optimization ensures that the HandsMen Threads CRM operates efficiently, loads quickly, and scales smoothly as data volume and user activity increase. This milestone focuses on improving Apex logic, Flow execution, UI performance, and data model efficiency to provide a seamless user experience.

1. **Apex Optimizations**
   To ensure efficient backend processing, the following Apex improvements were applied:
   No SOQL or DML Inside Loops
   - All queries and DML operations were moved outside loops.
   - Ensured compliance with Salesforce governor limits.

   Bulkified Triggers
   - All triggers were rewritten to handle multiple records at once using Lists, Maps, and Sets.
   - Eliminated single-record processing to avoid performance degradation.

   Reduced CPU Usage
   - Optimized conditional logic.
   - Queried only required fields (SELECT Id, Price__c).
   - Used selective WHERE filters to reduce query cost.

2. **Flow Optimizations**
   Flows were optimized to run efficiently without unnecessary system load.
   Removed Unnecessary Decision Nodes
   - Eliminated repetitive branches.
   - Simplified flow logic for faster execution.

   Combined Update Elements
   - Multiple updates were merged into a single Update action.
   - Reduced the number of DML operations executed by the Flow.

   Used Fault Paths for Debugging
   - All critical elements now include Fault Paths.
   - Errors are logged and displayed clearly to support troubleshooting.

3. **UI Optimization**
   Improvements were applied to make Salesforce pages load faster and display data more efficiently.

Dynamic Forms Enabled

- Only required fields load at runtime.
- Form rendering times reduced significantly.

Clean Layout Arrangement

- Fields grouped logically.
- Removed unnecessary sections to improve usability.

Simplified List Views

- Removed unneeded columns.
- Filters applied to show only relevant records to each team.

4. Data Optimization

The CRM data model was refined to reduce storage waste and boost system efficiency.

Avoided Storing Redundant Fields

- Eliminated duplicate fields and reused shared fields.
- Prevented unnecessary data duplication.

Used Formula Fields Instead of Stored Text

- Calculated values dynamically (e.g., Total Amount).
- Reduced storage usage and improved data consistency.

Phase 5: Maintenance & Documentation

Maintenance, Monitoring & Troubleshooting

Maintenance ensures that customer records, product data, inventory levels, order processing, and loyalty updates remain accurate and functional. The CRM must be continuously monitored to ensure smooth performance and data integrity.

Maintenance Activities

Functional Updates

- Updating product categories and pricing
- Revising stock management rules
- Modifying loyalty tier logic

Record Maintenance

- Cleaning duplicate customer records
- Updating inventory counts
- Removing outdated orders

Automation Maintenance

- Reviewing Flow failures
- Monitoring Apex trigger exceptions
- Checking scheduled jobs & batch apex results

These activities ensure system accuracy and operational reliability.

System Monitoring

Monitoring ensures that workflows, automations, and system access remain stable and error-free.

Monitor Flow Executions

- Setup → Flows → **Paused & Failed Flow Interviews**
- Check for failures in:
    - Order Confirmation Flow
    - Low Stock Alert Flow
    - Scheduled Loyalty Flow

Monitor Apex Jobs

- Setup → **Apex Jobs**
- Track:
    - Batch Apex (InventoryBatchJob)
    - Queueable Apex

- Scheduled Apex

Monitor Login & User Activity

- Setup → **Login History**
- Identify unauthorized or repeated failed logins

Monitor Data Quality

- Duplicate customer report
- Inventory mismatch report
- Orders missing lookup fields

Troubleshooting

Common troubleshooting areas include:

Flow Errors

- Missing field visibility
- Incorrect flow conditions
- Required field not accessible
- Flow trigger conditions not met

Apex Trigger Errors

- Null pointer exceptions
- Invalid or missing lookup records
- Recursive updates
- Incorrect trigger logic

Email Alert Failures

- Email deliverability disabled
- Invalid email address
- Incorrect flow/trigger conditions

All issues are resolved by examining debug logs and updating Flow/Apex configurations.

Project Documentation

Documentation ensures smooth maintenance, future scalability, and clear understanding of system components.

Documentation Includes:

- ER Diagrams & System Architecture
- Object Model (fields, relationships)
- Apex Triggers, Handler Classes, Batch Jobs
- Flow Diagrams & Automation Logic
- Page Layouts & Lightning Record Pages
- Security Model (Profiles, Roles, Permission Sets, Sharing Rules)

- Data Migration Steps
- Test Results (Unit, Integration, UAT)
- Screenshots of UI and dashboards

Benefits of Documentation

- Faster onboarding for new developers
- Easy maintenance for admins
- Troubleshooting reference
- Supports future system upgrades
- Ensures audit and compliance readiness

User Training & Adoption

To ensure smooth system usage, training is provided for Sales, Inventory, and Marketing users.

Training Topics

- Creating customers
- Managing orders
- Updating inventory
- Understanding loyalty updates
- Navigating dashboards and list views

Training Materials

- User Manual (PDF)
- Step-by-step screenshots
- Demo videos
- FAQs and troubleshooting guide

Proper training ensures system adoption and reduces errors.

Conclusion

The HandsMen Threads CRM system provides a robust, automated, and secure platform for managing:

- Customers
- Products
- Orders
- Inventory
- Loyalty Programs

Through the combination of Apex triggers, Record-Triggered Flows, Dynamic Forms, Batch Jobs, and a well-designed Security Model, the CRM reduces manual tasks and improves business efficiency.

Although built inside a Developer Org, the architecture and documentation follow enterprise-grade CRM implementation standards, demonstrating professional Salesforce development and configuration skills.