

**A Capstone Title:**

**HandsMen Threads:  
Elevating the Art of  
Sophistication in Men's  
Fashion**

**Submit by:**

*Jared Ian Torres*

## PROJECT OVERVIEW

HandsMen Threads is a Salesforce application built for a high-end men's clothing store. Its main goal is to link inventory tracking with customer service in one place. Instead of using separate lists for products, sales, and stock, the salesforce puts everything into a single system in the cloud. This makes the daily work much smoother. Style consultants can now check if an item is available instantly and see a customer's shopping history and loyalty status at the same time.

## OBJECTIVES

The main purpose of this project is to improve how the store runs for everyone involved. For store managers, the goal is to track inventory in real-time across different warehouses, so products never run out of stock. For sales contacts, the system is designed to automatically calculate order totals and send email confirmations, which helps avoid manual mistakes. Finally, for the marketing team, the objective is to group customers by their loyalty level Bronze, Silver, or Gold so they can send special offers that keep customers coming back.

## DATA MANAGEMENT - OBJECTS

The following Custom Objects were created to meet the data architecture requirements:

1. **HandsMen Customer:** Stores detailed client information, including contact details and loyalty status.
2. **HandsMen Order:** Represents sales transactions, linking customers to the products they purchase.
3. **HandsMen Product:** Acts as the master catalog for all items, storing pricing, SKU, and descriptions.
4. **Inventory:** Manages the stock levels for each product, ensuring real-time availability tracking per warehouse.
5. **Marketing Campaign:** Tracks promotional activities and the duration of marketing initiatives.

## DATA MANAGEMENT - TABS

Custom tabs were created for each of the objects above with distinct styles to ensure easy navigation for the end-users.

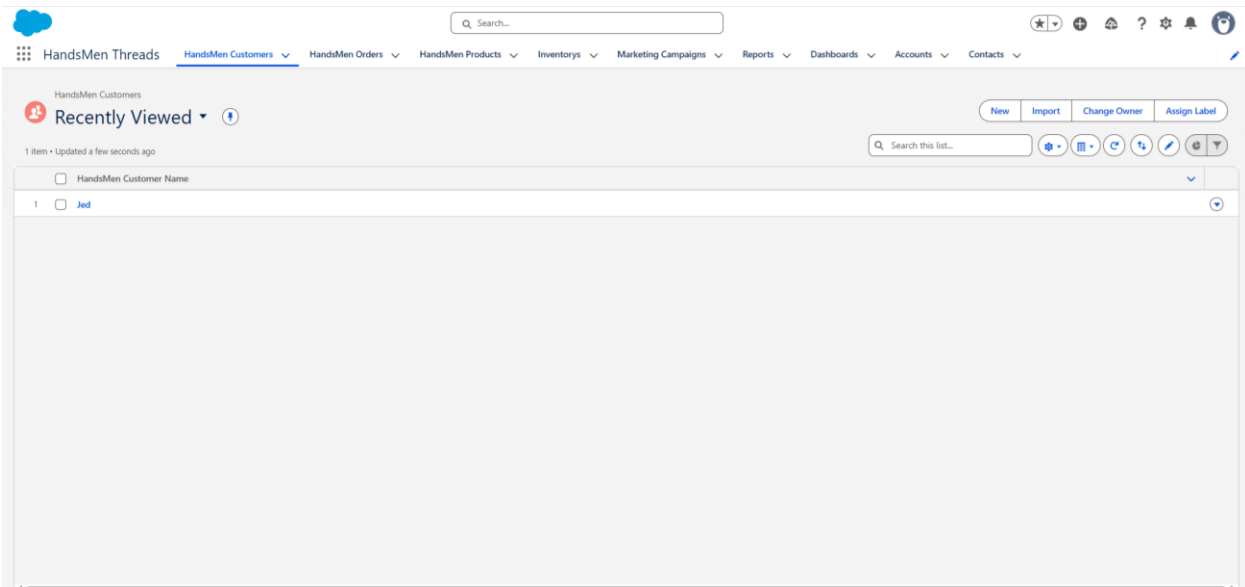
TABS	STYLES
<ul style="list-style-type: none"><li>HandsMen Customers</li><li>HandsMen Orders</li><li>HandsMen Products</li><li>Inventorys</li><li>Marketing Campaigns</li></ul>	<ul style="list-style-type: none"><li><i>People</i> (Orange)</li><li><i>Shopping Cart</i> (Brown)</li><li><i>Box</i> (Yellow)</li><li><i>Building</i> (Pink)</li><li><i>Mail</i> (Purple)</li></ul>

## DATA MANAGEMENT - APP MANAGER

**App Name:** HandsMen Threads

**Description:** A custom Lightning App designed to house all related objects and tabs in a single, branded environment.

**Navigation Items:** Home, HandsMen Customer, HandsMen Order, HandsMen Product, Inventory, Marketing Campaign, Reports, Dashboards.



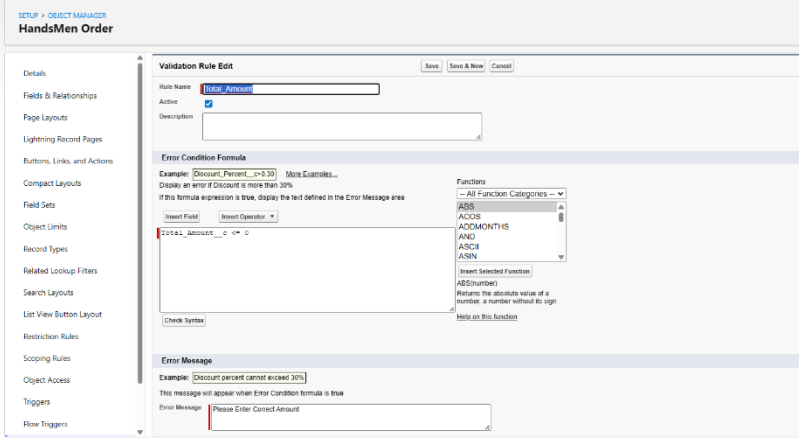
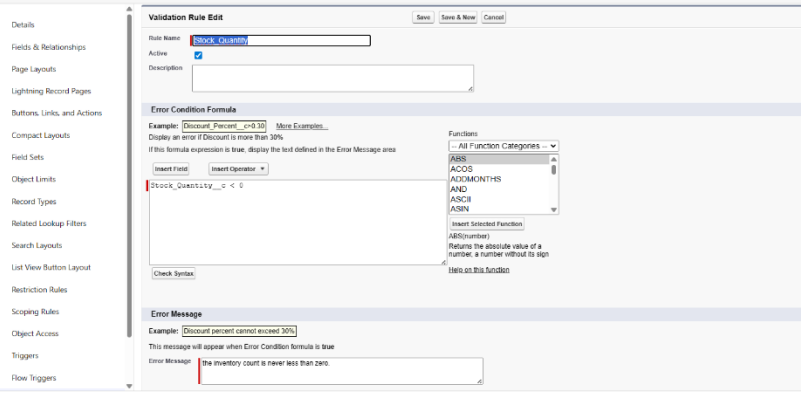
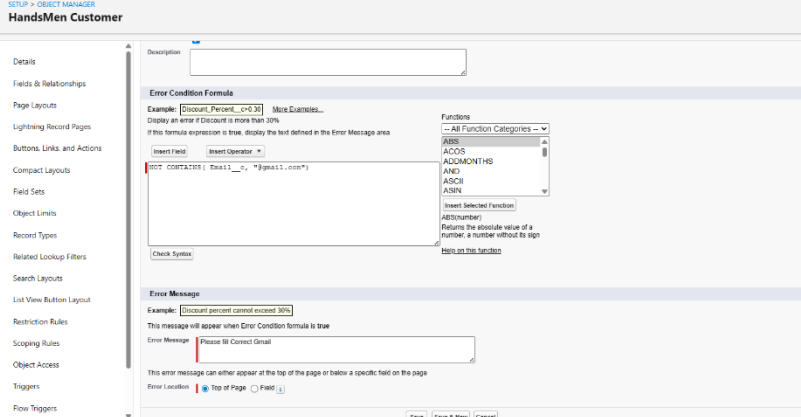
## **DATA MANAGEMENT - FIELDS**

Key custom fields were configured to capture specific business data, as verified in the Object Manager:

- **HandsMen Order:**
  - Customer Email (Email): Captures the buyer's email for notifications.
  - Status (Picklist): Tracks order progress (Pending, Confirmed, Rejection).
  - Quantity (Number): The number of items purchased.
  - Total Amount (Number): Calculated cost of the order.
  - HandsMen Customer (Lookup): Connected to “HandsMen Customer Object”
  - HandsMen Product (Lookup): Connected to “HandsMen Product Object”
- **HandsMen Product:**
  - Price (Currency): Unit cost of the item.
  - SKU (Text): Stock Keeping Unit identifier.
  - Stock Quantity (Number): Stock count.
  - HandsMen Product Name (Text): Name of the product.
- **Inventory:**
  - HandsMen Product (Master-Detail): Links inventory directly to the product catalog.
  - Stock Quantity (Number): Current count in a specific location.
  - Warehouse (Text): Specifies the physical location of the stock.
  - Stock Status (Formula): Automates status (Available, Low Stock).
- **Marketing Campaign:**
  - Start Date (Date): When the campaign begins.
  - End Date (Date): When the campaign concludes.
  - HandsMen Customer (Lookup): Connected to “HandsMen Customer Object”

# DATA CONFIGURATION

Validation Rules were implemented to ensure data integrity before saving records.

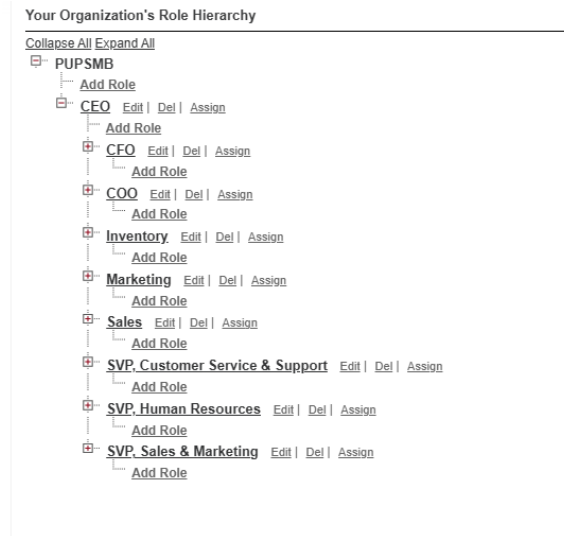
OBJECT	VALIDATION RULE
<p><b>HandsMen Order__c</b></p> <p>This validation rule will check if the total amount is 0 if its zero error message will pop out</p>	 <p>Validation Rule Edit</p> <p>Rule Name: <b>Total_Amount</b></p> <p>Active: <input checked="" type="checkbox"/></p> <p>Description:</p> <p>Error Condition Formula</p> <p>Example: <code>Discount_Percent__c &gt; 30</code> More Examples...</p> <p>Display an error if Discount is more than 30%</p> <p>If this formula expression is true, display the text defined in the Error Message area</p> <p>Insert Field: <b>Total_Amount__c</b> Insert Operator: <b>&lt;=</b></p> <p>Check Syntax</p> <p>Error Message</p> <p>Example: <code>Discount percent cannot exceed 30%</code></p> <p>This message will appear when Error Condition formula is true</p> <p>Error Message: <b>Please Enter Correct Amount</b></p>
<p><b>Inventory__c</b></p> <p>Same as “Handsmen Order” it will check if the Stock quantity is greater than 0 it will give error messages</p>	 <p>Validation Rule Edit</p> <p>Rule Name: <b>Stock_Quantity</b></p> <p>Active: <input checked="" type="checkbox"/></p> <p>Description:</p> <p>Error Condition Formula</p> <p>Example: <code>Discount_Percent__c &gt; 30</code> More Examples...</p> <p>Display an error if Discount is more than 30%</p> <p>If this formula expression is true, display the text defined in the Error Message area</p> <p>Insert Field: <b>Stock_Quantity__c</b> Insert Operator: <b>&lt;</b></p> <p>Check Syntax</p> <p>Error Message</p> <p>Example: <code>Discount percent cannot exceed 30%</code></p> <p>This message will appear when Error Condition formula is true</p> <p>Error Message: <b>the inventory count is never less than zero</b></p>
<p><b>Customer__c</b></p> <p>If the customer doesn’t input that contains “@gmail.com” it will not be accepted by the system and therefore an error message will pop-up</p>	 <p>Validation Rule Edit</p> <p>Rule Name: <b>Email</b></p> <p>Active: <input checked="" type="checkbox"/></p> <p>Description:</p> <p>Error Condition Formula</p> <p>Example: <code>Discount_Percent__c &gt; 30</code> More Examples...</p> <p>Display an error if Discount is more than 30%</p> <p>If this formula expression is true, display the text defined in the Error Message area</p> <p>Insert Field: <b>NOT CONTAINS (Email__c, "@gmail.com")</b> Insert Operator: <b>&amp;</b></p> <p>Check Syntax</p> <p>Error Message</p> <p>Example: <code>Discount percent cannot exceed 30%</code></p> <p>This message will appear when Error Condition formula is true</p> <p>Error Message: <b>Please fix Correct Gmail</b></p> <p>This error message can either appear at the top of the page or below a specific field on the page</p> <p>Error Location: <input checked="" type="radio"/> Top of Page <input type="radio"/> Field</p>

## DATA SECURITY - PROFILES

Users are assigned the **Platform 1** profile to grant them access to the custom application features, while System Administrators retain full access.

### Data Security - Roles

A Role Hierarchy was established under the root **PUPSMB (Company Name)** to control record visibility:



## DATA SECURITY - USERS

Specific users were created and assigned to Roles and Profiles to validate the security model:

1. **Mikaelson, Kol:**

- **Role:** Inventory
- **Profile:** Platform 1

2. **Mikaelson, Niklaus:**

- **Role:** Sales
- **Profile:** Platform 1

3. **Torres, Delta:**

- **Role:** Marketing
- **Profile:** Platform 1

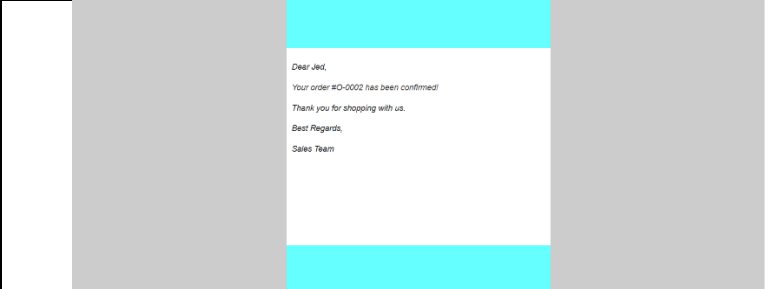
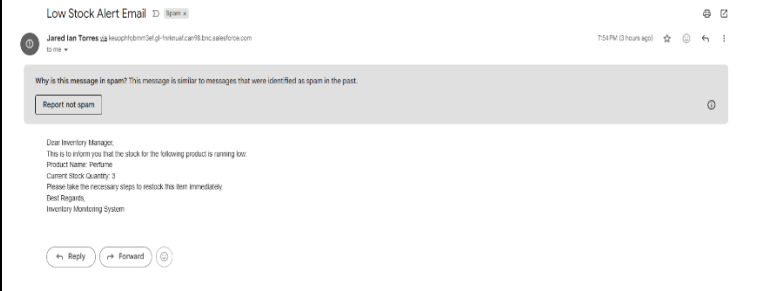
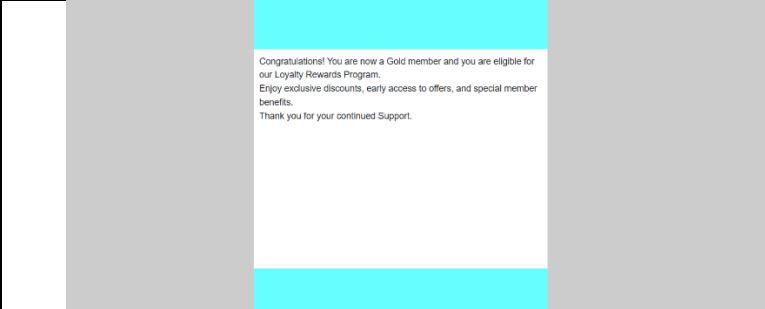
## DATA SECURITY - PERMISSION SET

Permission Sets were created to extend specific functional access to users beyond their profile settings:

1. **Sales Permission Set** - Niklaus Mikaelson
  - **Purpose:** Grants specific access required for sales processing.
2. **Inventory Permission Set** - Kol Mikaelson
  - **Purpose:** Grants specific access required for stock management.
3. **Marketing Permission Set** - Delta Torres
  - **Purpose:** Grants specific access required for campaign management.

## EMAIL TEMPLATE

The following email templates were configured to automate communication:

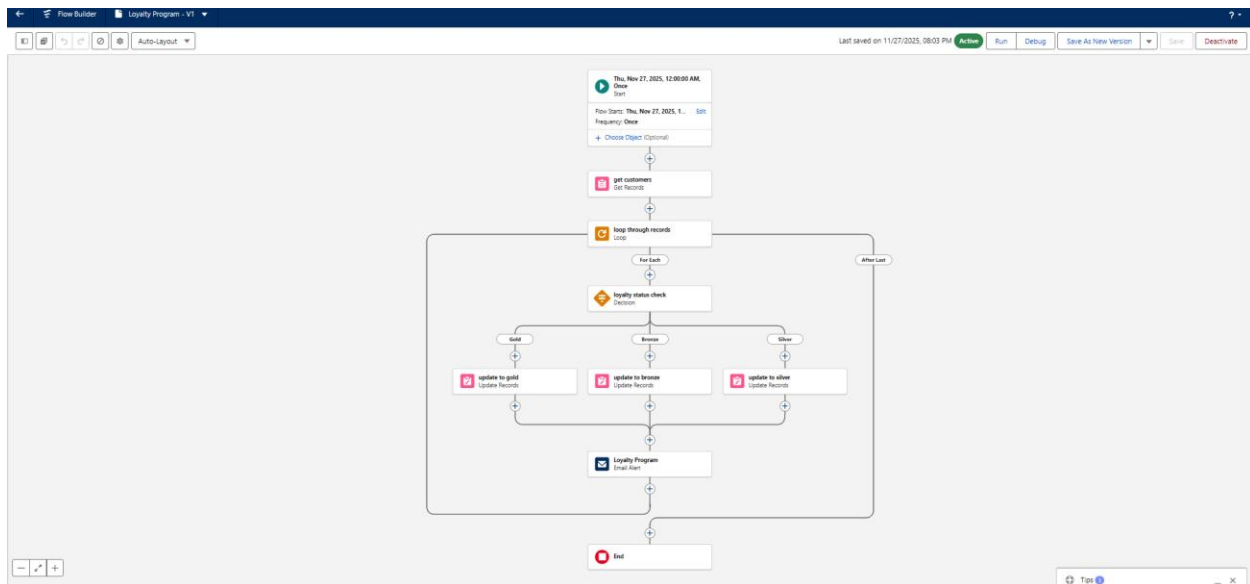
<b>Order Confirmation Email:</b> Sent to HandsMen Customer when an order status is 'Confirmed'.	 A preview of an order confirmation email. It features a light blue header and footer. The main body is white with a light blue border. The text reads: 'Dear Jax, Your order #O-0002 has been confirmed! Thank you for shopping with us. Best Regards, Sales Team'.
<b>Low Stock Alert Email:</b> Sent to the Inventory Manager when Inventory levels dip below the threshold.	 A preview of a low stock alert email. It shows a standard email interface with a header from 'Jared Ian Torres' and a subject 'Low Stock Alert Email'. The body text reads: 'Dear Inventory Manager, This is to inform you that the stock for the following product is running low: Product Name: Perfume Current Stock Quantity: 3 Please take the necessary steps to restock this item immediately. Best Regards, Inventory Monitoring System'.
<b>Loyalty Points Email:</b> Sent to customers upon reaching a new loyalty tier.	 A preview of a loyalty points email. It features a light blue header and footer. The main body is white with a light blue border. The text reads: 'Congratulations! You are now a Gold member and you are eligible for our Loyalty Rewards Program. Enjoy exclusive discounts, early access to offers, and special member benefits. Thank you for your continued Support.'

# FLWS

The following automation flows were built using Salesforce Flow Builder:

## 1. Loyalty Program Flow

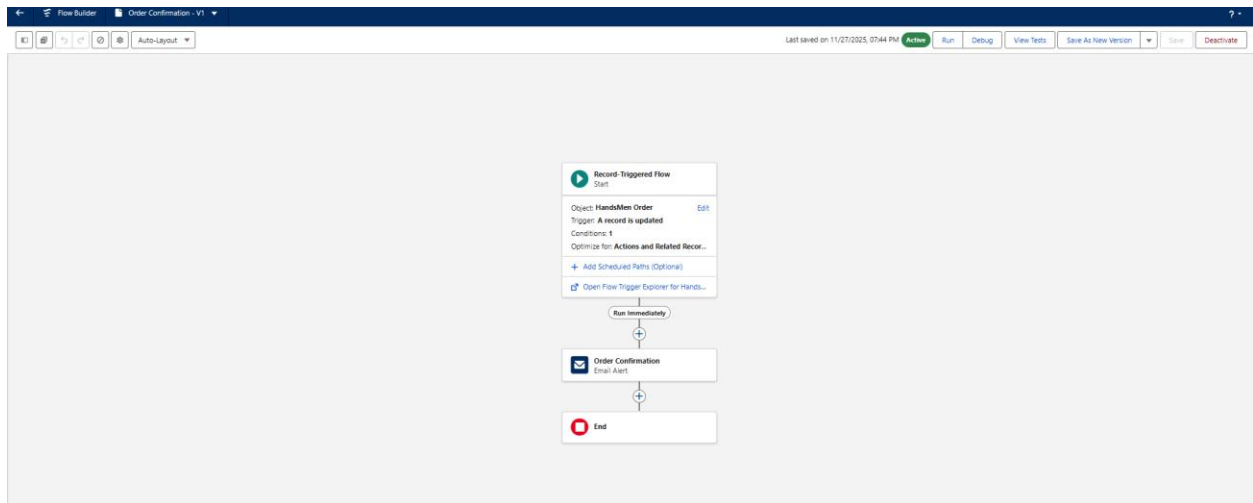
- **Type:** Schedule-Triggered Flow (Runs Once/Daily).
- **Logic:**
  1. **Get Records:** Fetches all Customer records.
  2. **Loop:** Iterates through each customer.
  3. **Decision:** Checks the customer's Total purchase.
  4. **Update Records:** Updates Loyalty Status to "Gold", "Silver", or "Bronze" based on the decision.
  5. **Action:** Sends the "Loyalty Program" email alert.



## 2. Order Confirmation Flow

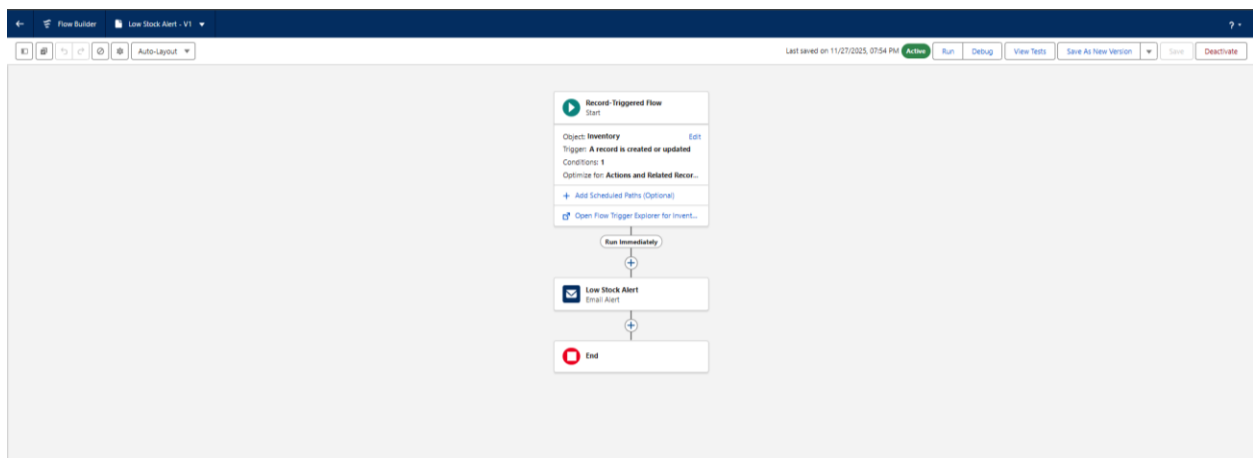
- **Type:** Record-Triggered Flow (After Update).
- **Object:** HandsMen Order.
- **Trigger:** Runs when a record is updated.
- **Action:** Sends the "Order Confirmation" email alert to the customer's email address.





### 3. Low Stock Alert Flow

- **Type:** Record-Triggered Flow (After Create or Update).
- **Object:** Inventory.
- **Trigger:** Runs when an Inventory record is created or updated.
- **Action:** Sends the "Low Stock Alert" email to the store manager if stock is low.

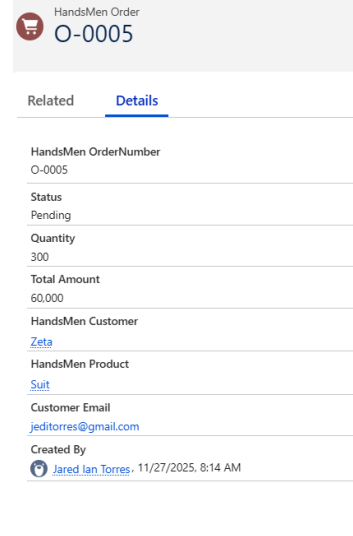
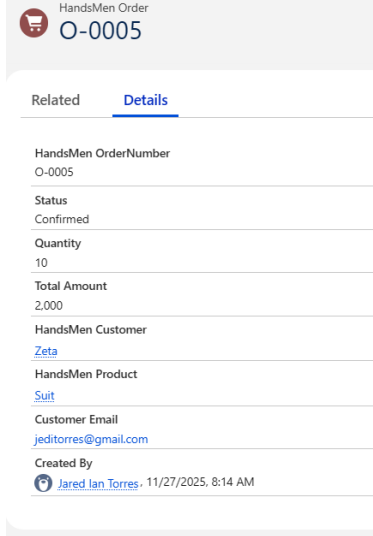


# AUTOMATION USING APEX

Two Apex triggers were developed to handle complex business logic:

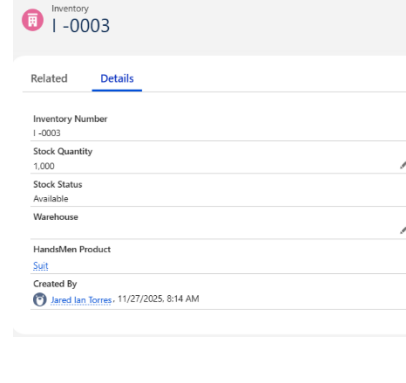
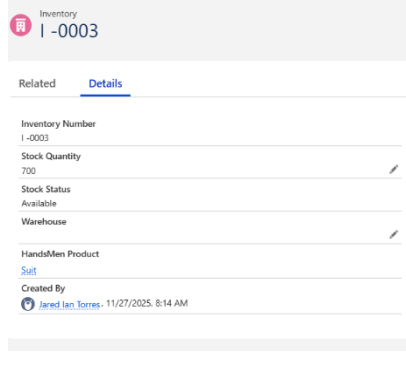
## 1. Update Order Total

- **Trigger:** OrderTotalTrigger
- **Logic:** Automatically calculates Total\_Amount\_\_c on HandsMen Order by multiplying Quantity by the related HandsMen Product Price.

CODE	BEFORE	AFTER
<pre>1 • trigger OrderTotalTrigger on HandsMen_Order__c (before insert, before 2 •   Set&lt;Id&gt; productIds = new Set&lt;Id&gt;(); 3 4 •   for (HandsMen_Order__c order : Trigger.new) { 5 •       if (order.HandsMen_Product__c != null) { 6 •           productIds.add(order.HandsMen_Product__c); 7 •       } 8 •   } 9 10  Map&lt;Id, HandsMen_Product__c&gt; productMap = new Map&lt;Id, HandsMen_Product__c&gt;(); 11  [SELECT Id, Price__c FROM HandsMen_Product__c WHERE Id IN :productIds]; 12 13  for (HandsMen_Order__c order : Trigger.new) { 14  •   if (order.HandsMen_Product__c != null &amp;&amp; productMap.containsKey(order.HandsMen_Product__c)) { 15  •       HandsMen_Product__c product = productMap.get(order.HandsMen_Product__c); 16  •       if (order.Quantity__c != null) { 17  •           order.Total_Amount__c = order.Quantity__c * product.Price__c; 18  •       } 19  •   } 20  } 21 22 }</pre>	 <p>HandsMen Order O-0005</p> <p>Related Details</p> <p>HandsMen OrderNumber O-0005</p> <p>Status Pending</p> <p>Quantity 300</p> <p>Total Amount 60,000</p> <p>HandsMen Customer Zeta</p> <p>HandsMen Product Suit</p> <p>Customer Email jeditorres@gmail.com</p> <p>Created By Jared Ian Torres · 11/27/2025, 8:14 AM</p>	 <p>HandsMen Order O-0005</p> <p>Related Details</p> <p>HandsMen OrderNumber O-0005</p> <p>Status Confirmed</p> <p>Quantity 10</p> <p>Total Amount 2,000</p> <p>HandsMen Customer Zeta</p> <p>HandsMen Product Suit</p> <p>Customer Email jeditorres@gmail.com</p> <p>Created By Jared Ian Torres · 11/27/2025, 8:14 AM</p>

## 2. Stock Deduction Trigger

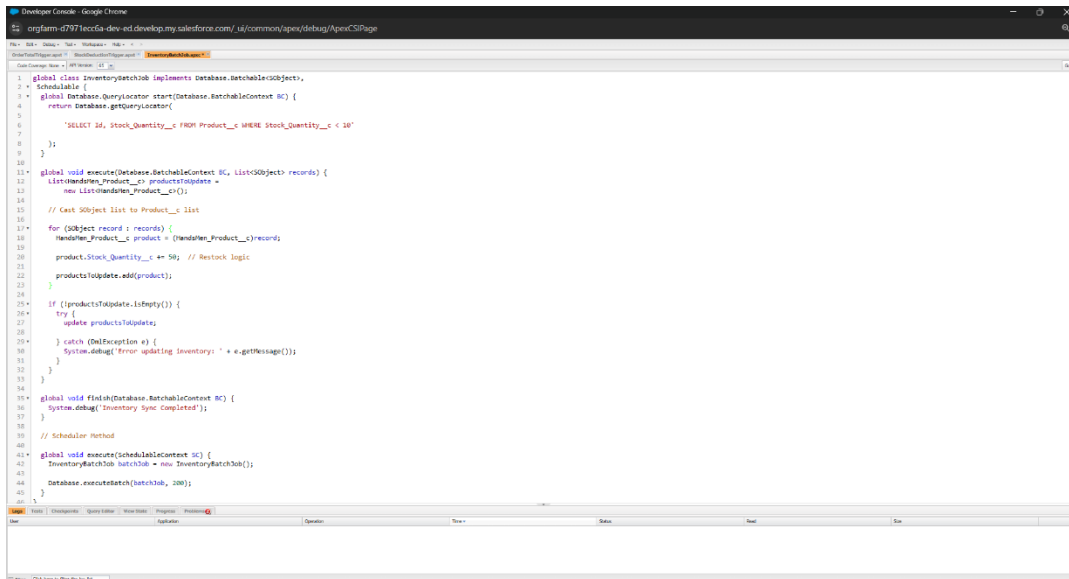
- **Trigger:** StockDeductionTrigger
- **Logic:** When an HandsMen Order is finalized, this trigger automatically subtracts the order quantity from the related Inventory record.

CODE	BEFORE	AFTER
<pre>1 • trigger StockDeductionTrigger on HandsMen_Order__c (before update, before 2 •   Set&lt;Id&gt; productIds = new Set&lt;Id&gt;(); 3 4 •   for (HandsMen_Order__c order : Trigger.new) { 5 •       if (order.Status__c == 'Confirmed' &amp;&amp; order.HandsMen_Product__c != null) { 6 •           productIds.add(order.HandsMen_Product__c); 7 •       } 8 •   } 9 10  [SELECT Id, Stock_Quantity__c FROM Inventory WHERE Id IN :productIds]; 11 12  List&lt;Inventory&gt; inventoryList = new List&lt;Inventory&gt;(); 13 14  for (Inventory inv : inventoryList) { 15  •   if (inv.Stock_Quantity__c &gt;= order.Quantity__c) { 16  •       inv.Stock_Quantity__c -= order.Quantity__c; 17  •   } else { 18  •       inv.Stock_Quantity__c = 0; 19  •   } 20  } 21 22  update inventoryList; 23 24  for (HandsMen_Order__c order : Trigger.new) { 25  •   if (order.Status__c == 'Confirmed' &amp;&amp; order.HandsMen_Product__c != null) { 26  •       HandsMen_Product__c product = [SELECT Id, Stock_Quantity__c FROM HandsMen_Product__c WHERE Id = :order.HandsMen_Product__c]; 27  •       product.Stock_Quantity__c -= order.Quantity__c; 28  •       update product; 29  •   } 30  } 31 32 }</pre>	 <p>Inventory I-0003</p> <p>Related Details</p> <p>Inventory Number I-0003</p> <p>Stock Quantity 1,000</p> <p>Stock Status Available</p> <p>Warehouse</p> <p>HandsMen Product Suit</p> <p>Created By Jared Ian Torres · 11/27/2025, 8:14 AM</p>	 <p>Inventory I-0003</p> <p>Related Details</p> <p>Inventory Number I-0003</p> <p>Stock Quantity 700</p> <p>Stock Status Available</p> <p>Warehouse</p> <p>HandsMen Product Suit</p> <p>Created By Jared Ian Torres · 11/27/2025, 8:14 AM</p>

# BATCH JOBS

To maintain data accuracy, scheduled jobs were implemented:

- **Job Name:** Daily Inventory Sync
- **Purpose:** Runs nightly to resolve inventory counts and flag discrepancies.
- **Execution:**



```
1  global class InventoryBatchJob implements Database.Batchable<Object>,
2  > {
3  >     global Database.QueryLocator start(Database.BatchableContext BC) {
4  >         return Database.getQueryLocator(
5  >             'SELECT Id, Stock_Quantity__c FROM Product__c WHERE Stock_Quantity__c < 10'
6  >         );
7  >     }
8  >
9  >
10 >     global void execute(Database.BatchableContext BC, List<Object> records) {
11 >         List<Inventory_Product__c> productsToUpdate =
12 >             new List<Inventory_Product__c>();
13 >
14 >         // Cast SObject list to Product__c list
15 >
16 >         for (SObject record : records) {
17 >             Inventory_Product__c product = (Inventory_Product__c)record;
18 >
19 >             product.Stock_Quantity__c += 50; // Restock logic
20 >
21 >             productsToUpdate.add(product);
22 >         }
23 >
24 >         if (!productsToUpdate.isEmpty()) {
25 >             try {
26 >                 update productsToUpdate;
27 >             } catch (DmlException e) {
28 >                 System.debug('Error updating inventory: ' + e.getMessage());
29 >             }
30 >         }
31 >     }
32 >
33 >
34 >     global void finish(Database.BatchableContext BC) {
35 >         System.debug('Inventory sync completed');
36 >     }
37 >
38 >     // Scheduler Method
39 >
40 >     global void execute(SchedulableContext SC) {
41 >         InventoryBatchJob batchJob = new InventoryBatchJob();
42 >
43 >         Database.executeBatch(batchJob, 200);
44 >     }
45 >
46 > }
```