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

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.

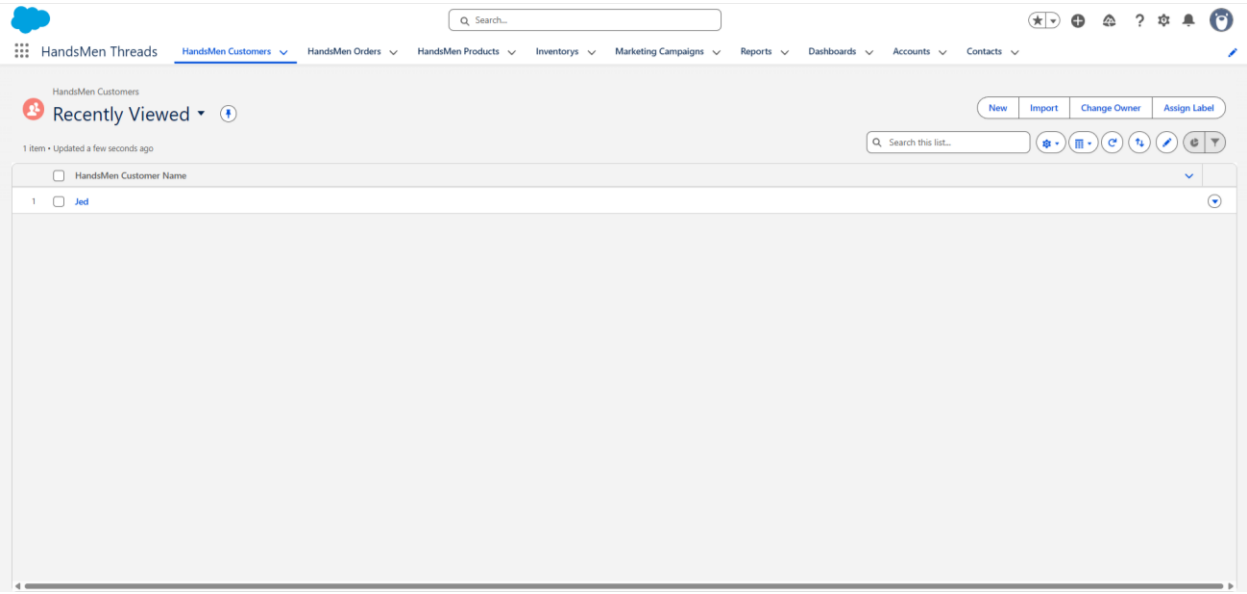
TABS	STYLES
• HandsMen Customers • HandsMen Orders • HandsMen Products • Inventorys • Marketing Campaigns	• *People* (Orange) • *Shopping Cart* (Brown) • *Box* (Yellow) • *Building* (Pink) • *Mail* (Purple)

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.



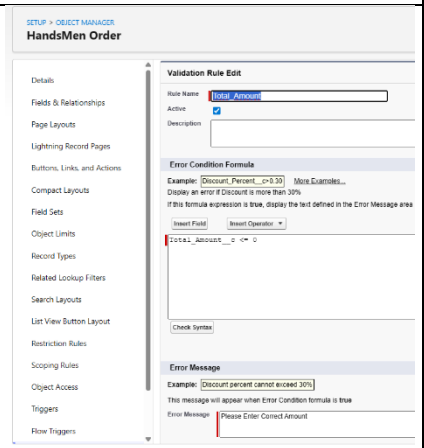
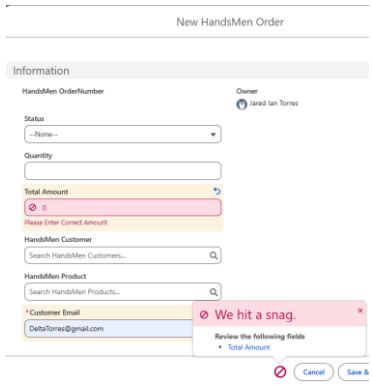
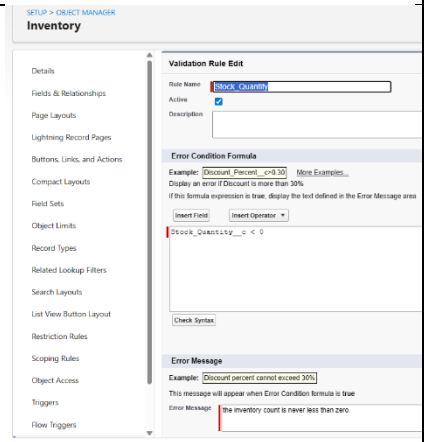
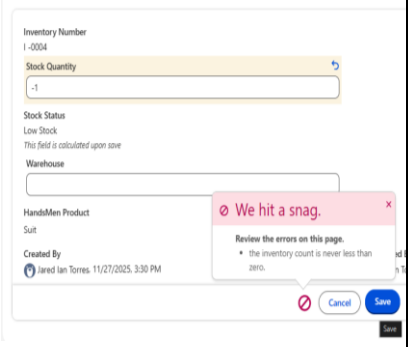
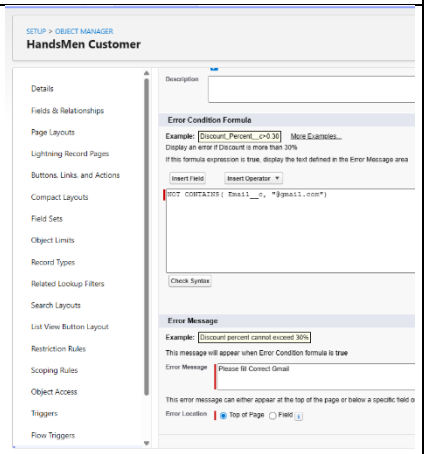
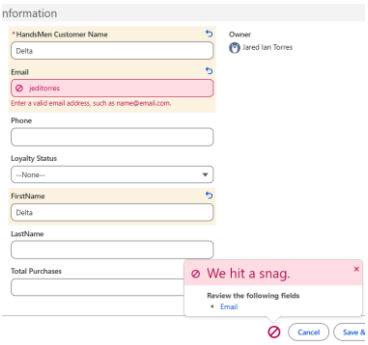
DATA MANAGEMENT - FIELDS

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.

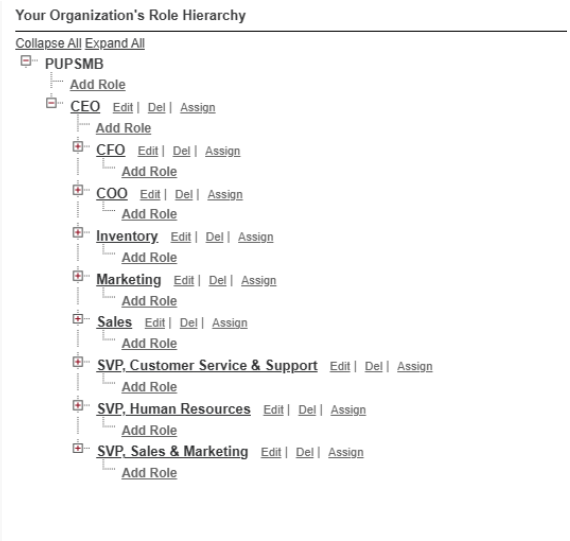
OBJECT	VALIDATION RULE	ERROR
HandsMen Order__c This validation rule will check if the total amount is 0 if its zero error message will pop out		
Inventory__c Same as “Handsmen Order” it will check if the Stock quantity is greater than 0 it will give error messages		
Customer__c If the customer doesn’t input that contains “@gmail.com” it will not be accepted by the system and therefore an error message will pop-up		

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

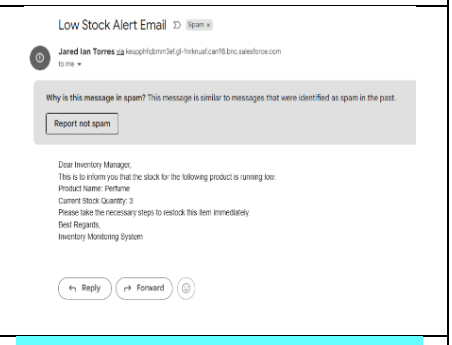
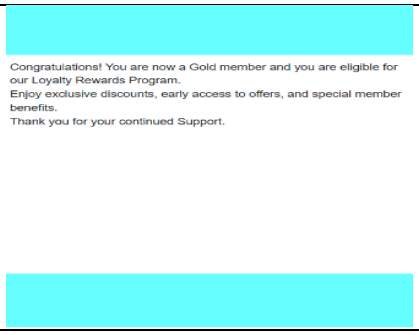
DATA SECURITY - PERMISSION SET

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:

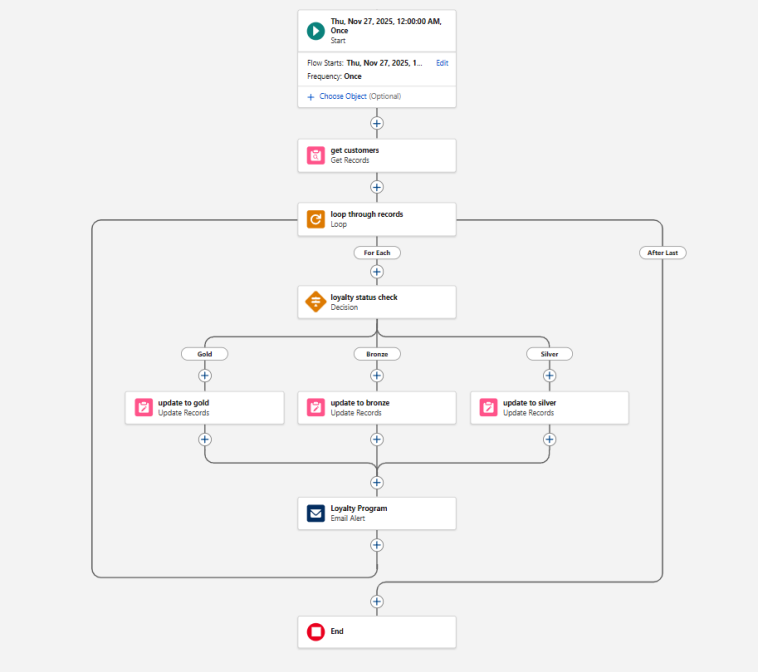
Order Confirmation Email: Send it to HandsMen Customer when an order status is 'Confirmed'.	
Low Stock Alert Email: Sent to the Inventory Manager when Inventory levels dip below the threshold.	
Loyalty Points Email: Sent to customers upon reaching a new loyalty tier.	

FLOWS

The following automation flows were built using Salesforce Flow Builder:

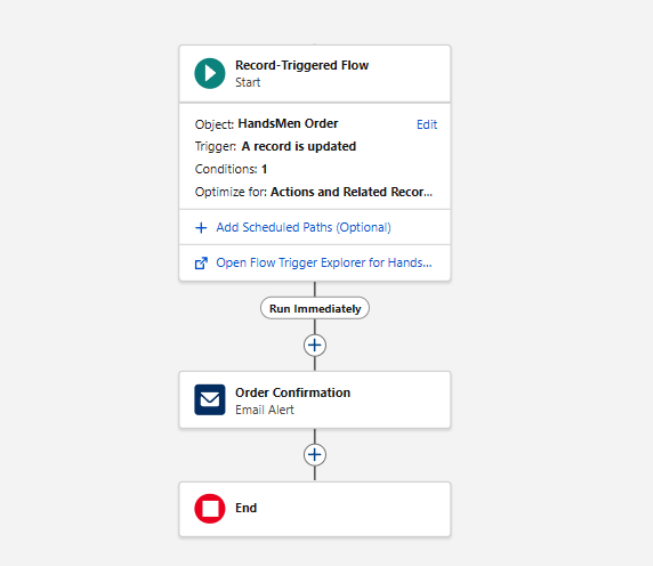
1. Loyalty Program Flow

- **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.



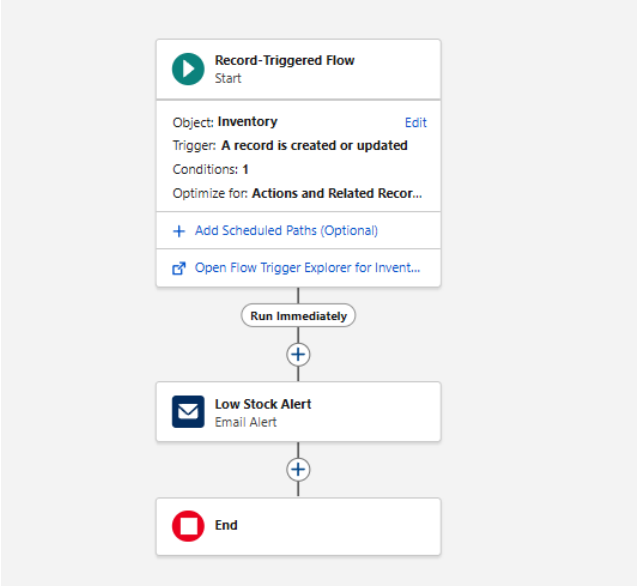
2. Order Confirmation Flow

- **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

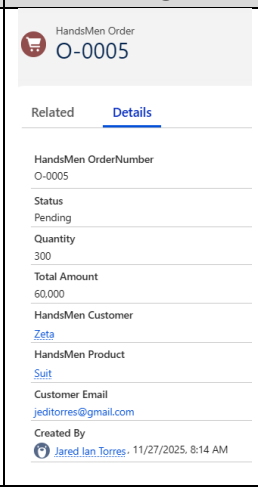
Two Apex triggers were developed to handle complex business logic:

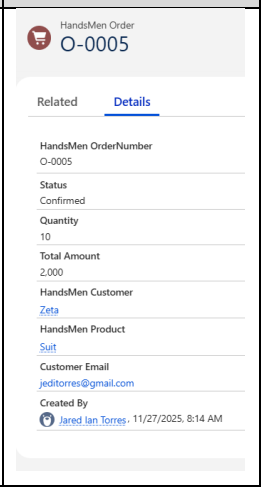
1. Update Order Total (OrderTotalTrigger)

- **Logic:** Think of this as an automatic calculator. Instead of doing math in their heads, staff just pick a product and quantity. The system instantly calculates the total price. It saves time and stops mistakes.

CODE	BEFORE	AFTER

```
trigger OrderTotalTrigger on HandsMen_Order__c (before insert, before update) {
    Set<Id> productIds = new Set<Id>();

    for (HandsMen_Order__c order : Trigger.new) {
        if (order.HandsMen_Product__c != null) {
            productIds.add(order.HandsMen_Product__c);
        }
    }

    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 order : Trigger.new) {
        if (order.HandsMen_Product__c != null && productMap.containsKey(order.HandsMen_Product__c)) {
            HandsMen_Product__c product = productMap.get(order.HandsMen_Product__c);
            if (order.Quantity__c != null) {
                order.Total_Amount__c = order.Quantity__c * product.Price__c;
            }
        }
    }
}
```



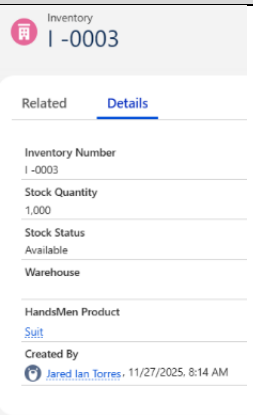


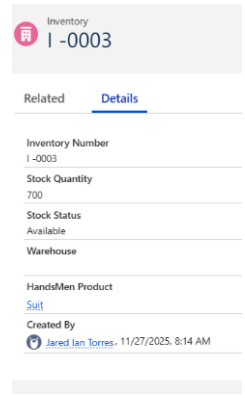
2. Stock Deduction Trigger (StockDeductionTrigger)

- **Logic:** When a sale is confirmed, the system immediately subtracts those items from your stock count. It keeps your inventory numbers perfect without anyone needing to count the shelves.

CODE	BEFORE	AFTER

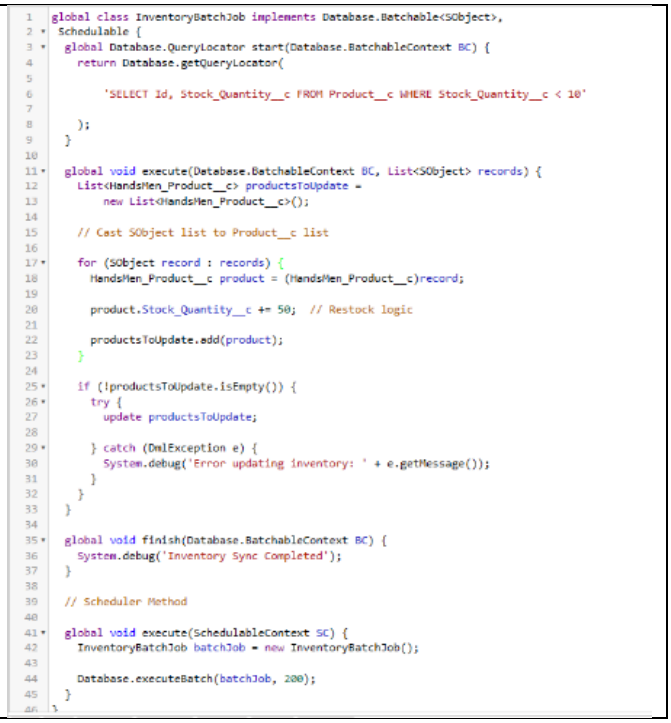
```
Set<Id> productIds = new Set<Id>();

for (HandsMen_Order__c order : Trigger.new) {
    if (order.Status__c == 'Confirmed' && order.HandsMen_Product__c != null) {
        productIds.add(order.HandsMen_Product__c);
    }
}

if (productIds.isEmpty()) return;

// Query related inventories based on product
Map<Id, Inventory__c> inventoryMap = new Map<Id, Inventory__c>(
    [SELECT Id, Stock_Quantity__c, HandsMen_Product__c
     FROM Inventory__c
     WHERE HandsMen_Product__c IN :productIds]
);

List<Inventory__c> inventoriesToUpdate = new List<Inventory__c>();

for (HandsMen_Order__c order : Trigger.new) {
    if (order.Status__c == 'Confirmed' && order.HandsMen_Product__c != null) {
        for (Inventory__c inv : inventoryMap.values()) {
            if (inv.HandsMen_Product__c == order.HandsMen_Product__c) {
                inv.Stock_Quantity__c -= order.Quantity__c;
                inventoriesToUpdate.add(inv);
                break;
            }
        }
    }
}

if (!inventoriesToUpdate.isEmpty()) {
    update inventoriesToUpdate;
}
}
```





BATCH JOBS

To maintain data accuracy, scheduled jobs were implemented:

The InventoryBatchJob acts like a scheduled night-shift worker for your data. Instead of a person manually checking every single product for low stock, this automated process wakes up at a set time (usually at night) and scans the entire inventory list in the background. If it finds any product with fewer than 10 items, it automatically "restocks" it by adding 50 units to the count and then saves the changes. This ensures that your inventory levels are refreshed and accurate by the next morning without any manual effort from your team.	

```apex
global class InventoryBatchJob implements Database.Batchable<SObject>,
Schedulable {
  global Database.QueryLocator start(Database.BatchableContext BC) {
    return Database.getQueryLocator(

        'SELECT Id, Stock_Quantity__c FROM 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>();

    // Cast SObject list to Product__c list

    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);
  }
}
```

CONCLUSION

HandsMen Threads project has successfully transformed how the store operates. By moving from manual tracking to a centralized Salesforce system, we have connected inventory, sales, and customer data into one smooth workflow. The automation we built now handles repetitive tasks like calculating prices and updating stock counts instantly, while our security settings ensure that sensitive data is only seen by the right people. Overall, this system makes the business more efficient, accurate, and customer-focused.

ENHANCEMENTS

To make the system even better, we plan to add:

1. **Einstein Copilot:** To help staff write customer emails instantly.
2. **Mobile Bar Code Scanning:** So, warehouse staff can update stock just by scanning barcodes with their phones.
3. **Direct Payments (E-Pay):** Connecting to GCash or PayPal so customers can pay immediately via email or in the store.