Salesforce CRM Project Documentation

Project Title:

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

Author: **John Vincent Del Rosario**

Date: **November 27, 2025**

Project Overview

Handsmen Threads is a Salesforce Customer Relationship Management/CRM solution that designed to streamline a customer engagement and sales processes for a premium men's fashion brand. The CRM focuses on a personalized customer management, order tracking and loyalty programs, ensuring that the brand provide a sophistication both in its products and customer experience.

Objectives

- **Enhancing customer relationship management in terms of tracking preferences and purchased history.**
- **Streamline booking and order processes for bespoke tailoring services.**
- **Provide an actionable insights through dashboards and reports.**
- **Support scalability for future improvements like AI-driven suggestions and chatbot integration.**

Phase 1: Requirement Analysis & Planning

- **Business Requirements** – a customers need a personalized service, efficient order tracking and loyalty rewards.

- **Project Scope –** build a CRM with modules specifically for customer profiles, booking, payments and loyalty programs.
- **Data & Security Model –** custom objects for customers, orders, loyalty points (role-based access for sales staff and admins).
- **Stakeholders –** sales team, tailoring staff, management and IT support.

- **Execution Roadmap –** Requirement gathering, Development, Testing, Deployment lastly is Maintenance.

Phase 2: Salesforce Development – Backend & Configurations

Environment Setup: Sandbox and DevOps workflow established.

Customizations:

- **Custom objects:**

  ***Customer, Booking, Payment, Loyalty Points*.**

- **Validation rules:**

  Ensure accurate customer measurements and payment details.

- **Automation:**

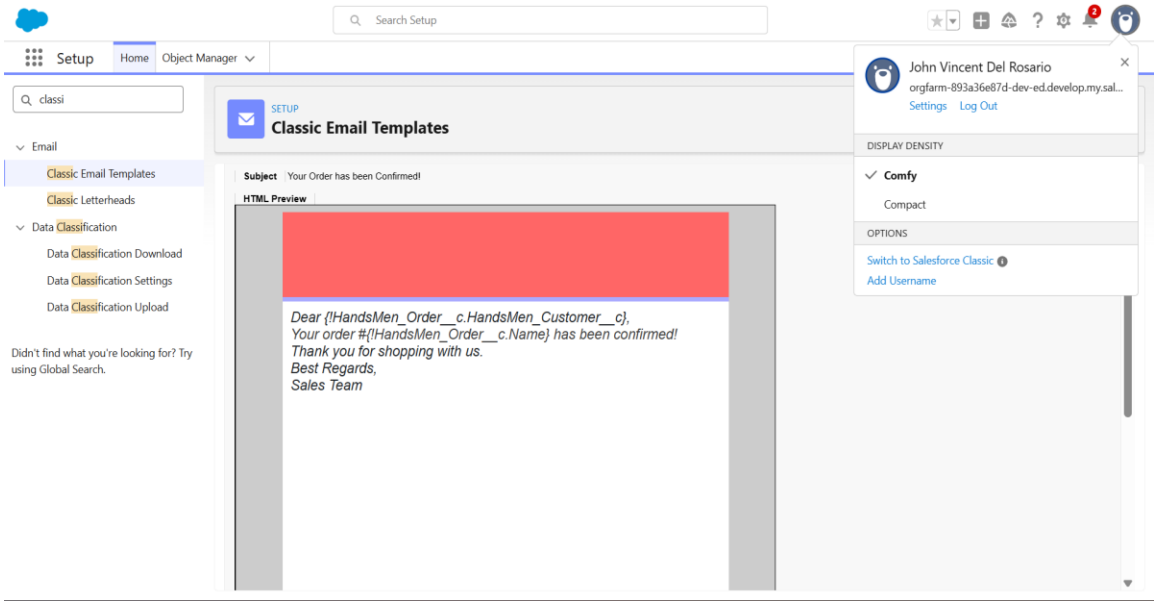
  Flows for loyalty point updates, approval processes for bespoke orders.

- **Apex Development:**
  Triggers for automatic loyalty point calculation.

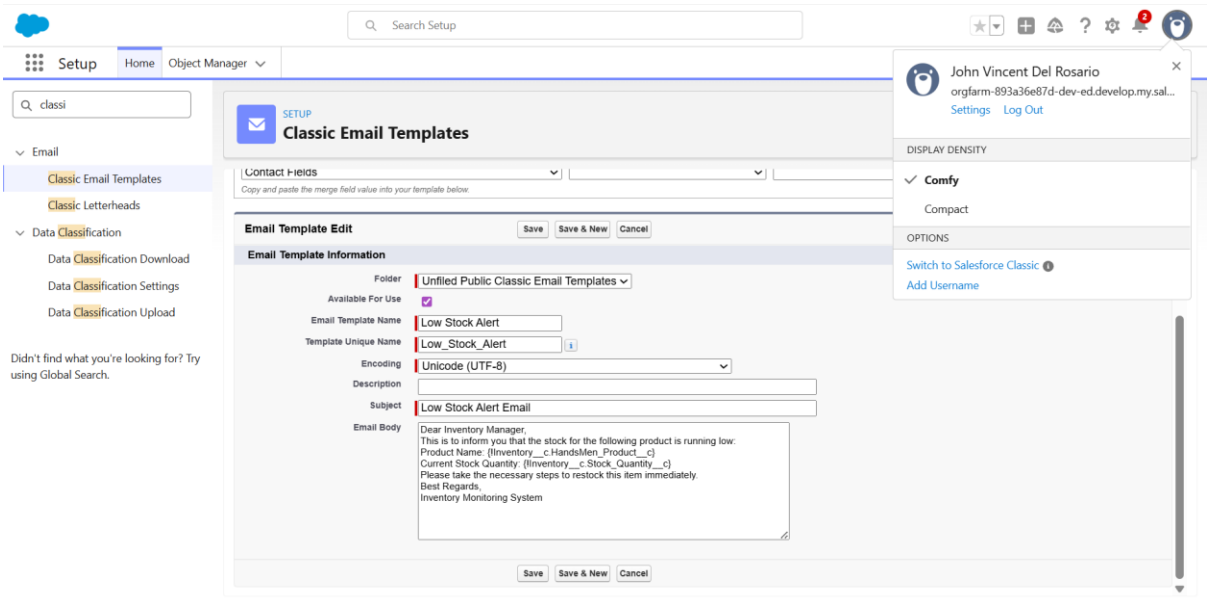
- **Screenshots:**
  (Insert relevant Salesforce setup screenshots).

Here’s the Relevant Screenshots:

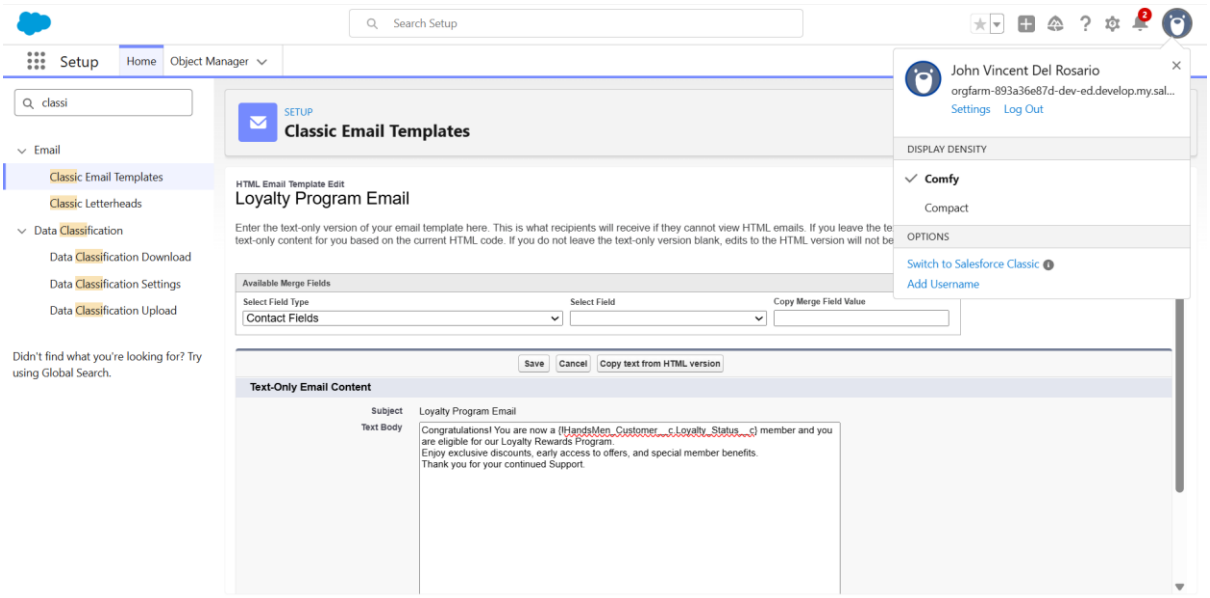
1. ***Order Confirmation Email***



2. ***Low Stock Alert Email***



3. *Loyality Points Email*

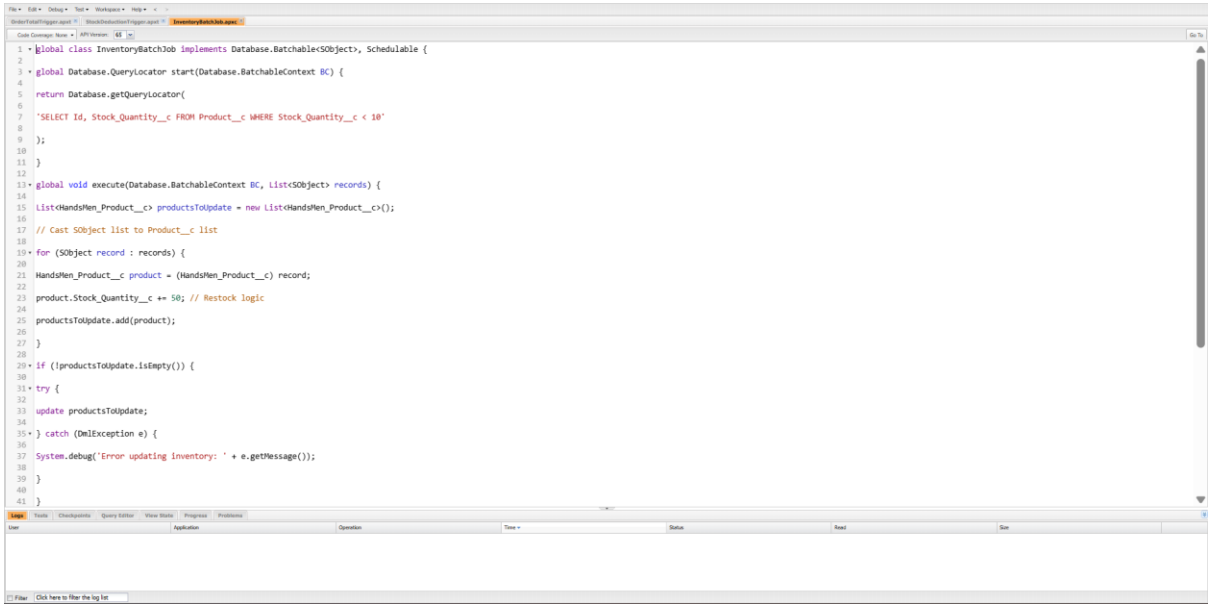


4. *Order Total Trigger*

5. *Stock Deduction Trigger*

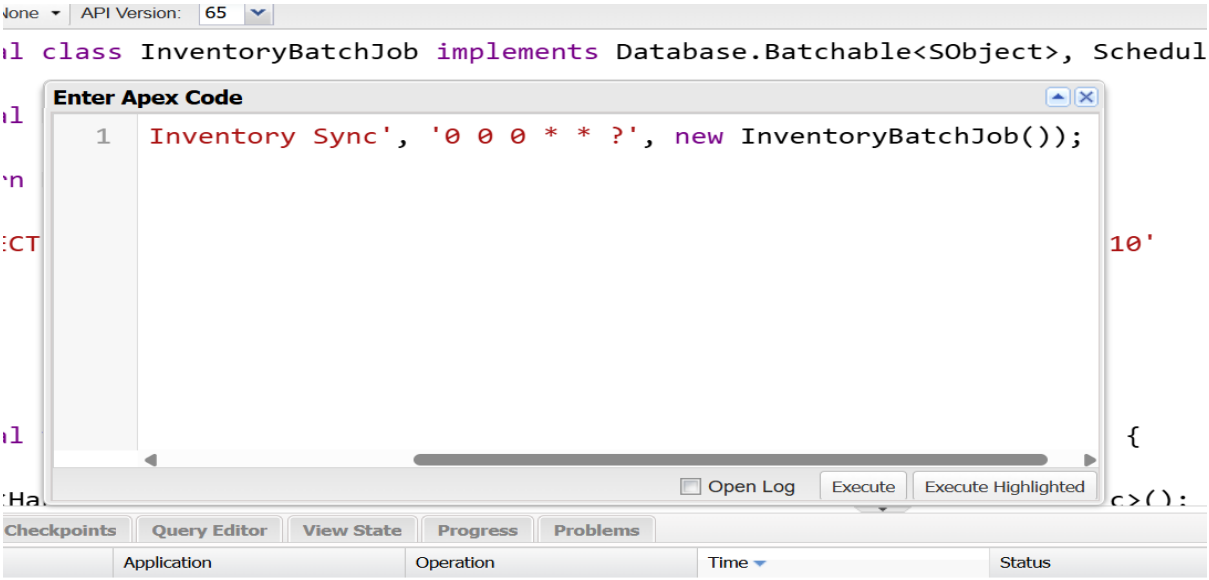
```
trigger StockDeductionTrigger on HandsMen_Order__c (after insert, after update) {
    Set<Id> productIds = new Set<Id>();

    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;
    }
}
```

6. *Inventory Batch Job*

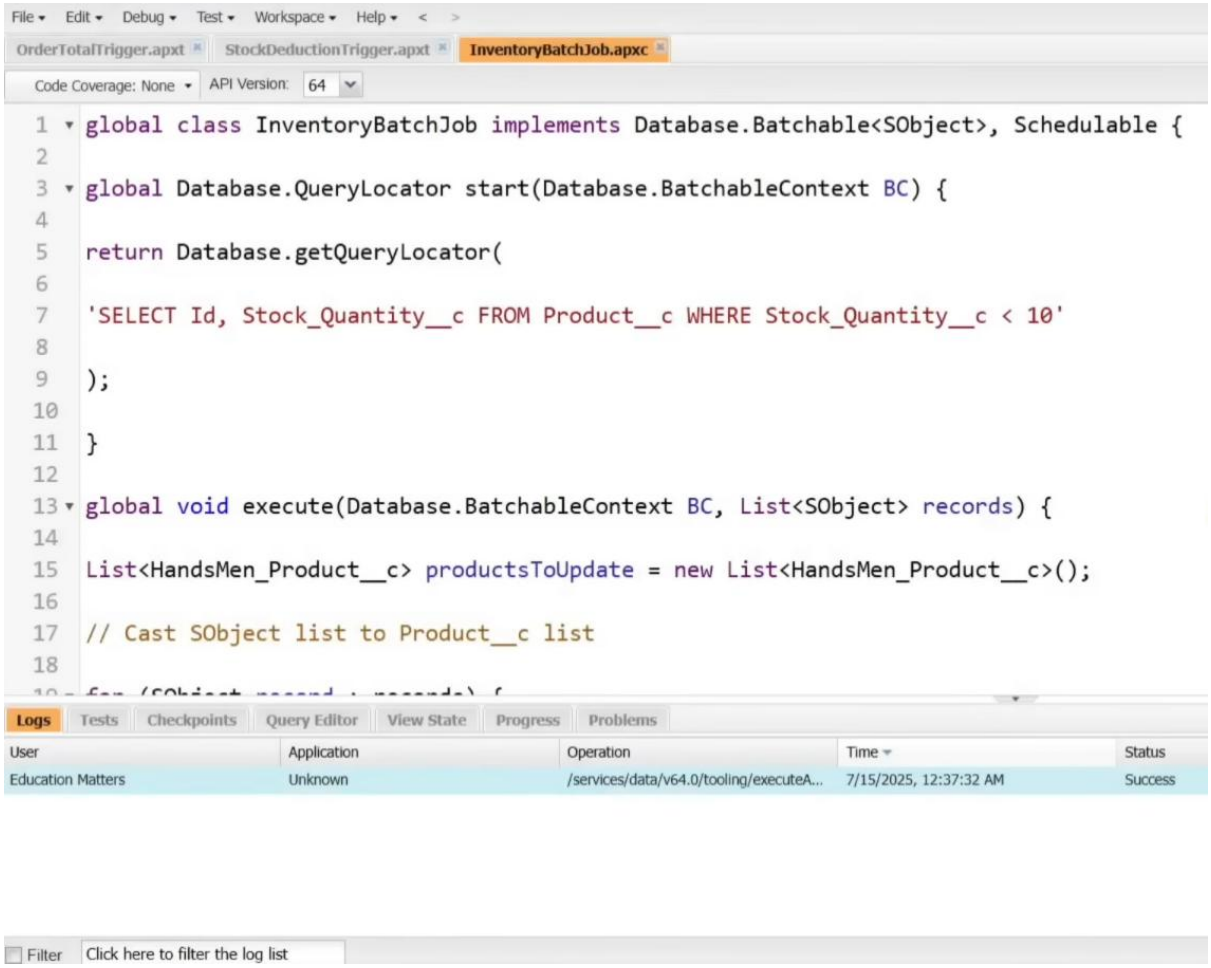


```
global class InventoryBatchJob implements Database.Batchable<SObject>, Schedulable {

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());

}

}
```

7. *Inventory Executing*



8. *Inventory Executing Success*



Phase 3: UI/UX Development & Customization

- **Lightning App:**

  Handsmen Threads CRM app created via App Manager.
- **Page Layouts & Dynamic Forms:**

  Tailored layouts for customer profiles and bookings.
- **Reports & Dashboards:**

  Sales performance, customer loyalty trends, booking pipeline.
- **LWC Development:**

  Optional components for real-time order status.

Phase 4: Data Migration, Testing & Security

- **Data Migration:**

  Imported customer records via Data Loader.
- **Security:**

  Profiles for Admin, Sales, Tailor; role hierarchy for managers.
- **Testing:**

  Test cases for booking creation, loyalty point updates, approval workflows.

Date Model and Key Fields

Handsman Customer

- **FullName**
- **Email**
- **Phone**
- **Loyalty_Status_c**
- **Total Purchases_c**

Handsman Product

- **SKU_c**
- **Name**
- **Price_c**
- **Stock_Quantity_c**

Inventory

- **HandsMen_Product_c(Master Detail)**
- **CreatedById**
- **Warehouse**
- **Stock_Quantity_c**
- **Stock_Status_c**

Handsman Orders

- **Quantity_c**
- **Customer_Email_c**
- **HandsMen_Customer_c (lookup)**
- **Status_c (Pending Confirmed Cancelled)**
- **HandsMen_Product_c (lookup)**
- **Name (Order Number)**