

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

## Project Overview:

The Handsmen Threads CRM project transforms the traditional, manual operations of our bespoke tailoring business into a centralized, cloud-based system. By leveraging Salesforce, we aim to streamline client measurement tracking, inventory management, and appointment scheduling, ensuring a seamless experience from the initial fitting to the final suit delivery.

## Objectives:

1. **Operational Efficiency:** Automate manual booking and order tracking to reduce human error.
2. **Data Integrity:** Securely store customer body measurements and preferences in a single digital location.
3. **Inventory Control:** Real-time tracking of fabric usage to prevent material shortages.

## Phase 1: Requirement Analysis & Planning

In this phase, I focused on mapping out the system architecture before implementation. I defined the data structure, business logic, and communication strategies needed to support Handsmen Threads' operations.

### 1. Define Objects, Fields, and Relationships (ERD)

I designed a data model that extends standard Salesforce capabilities. I analyzed the business process to determine necessary Custom Objects.

- **Custom Objects:** HandsMen\_Customer\_\_c (to store customer data), HandsMen\_Order\_\_c (order tracking), HandsMen\_Product\_\_c (product listing), Marketing\_Campaign\_\_c, and Inventory\_\_c.
- **Relationships:** Established Lookup Relationships and Master-Detail relationships between different custom objects to ensure data cascading.

The top screenshot shows the Salesforce Object Manager interface with a list of custom objects:

LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED
Marketing Campaign	Marketing_Campaign_c	Custom Object		11/28/2025	✓
Inventory	Inventory_c	Custom Object		11/28/2025	✓
HandsMen Order	HandsMen_Order_c	Custom Object		11/28/2025	✓
HandsMen Product	HandsMen_Product_c	Custom Object		11/28/2025	✓
HandsMen Customer	HandsMen_Customer_c	Custom Object		11/28/2025	✓

The bottom screenshot shows the Marketing Campaign object details:

Fields & Relationships				
7 Items, Sorted by Field Label				
FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
End Date	End_Date_c	Date		
HandsMen Customer	HandsMen_Customer_c	Lookup(HandsMen Customer)	✓	
Last Modified By	LastModifiedById	Lookup(User)		
Marketing Campaign Number	Name	Auto Number	Activate Windows Go to Settings to activate Windows.	
Owner	OwnerId	Lookup(User,Group)	✓	

## 2. Establish Business Logic Strategy

I planned the automation logic required to enforce business rules:

- **Validation Rules:** defined to prevent errors, such as entering negative values for measurements.
- **Flows & Triggers:** Designed workflows to auto-update inventory when an order is placed.
- **Batch Jobs:** Planned a nightly batch job to scan for orders pending delivery for more than 30 days.

## 3. Design Email Templates

I drafted communication templates to keep customers informed.

- *Order Confirmation Email:* Order Confirmation.
- *Low Stock Alert:* “Low Stock” Notification.

- *Loyalty Program Email:* Either Bronze, Silver, or Gold member based on purchases.

The screenshot shows the Salesforce Setup interface with the 'Email Alerts' page selected. The left sidebar has 'Email Alerts' highlighted. The main area displays a list of three email alerts: 'Low Stock Alert', 'Loyalty Program Alert', and 'Order Confirmation Email Alert'. Each alert includes its description, email template name, object, and last modified date.

Action	Description	Email Template Name	Object	Last Modified Date
Edit   Del	Low Stock Alert	Low Stock Alert	Inventory	11/28/2025
Edit   Del	Loyalty Program Alert	Loyalty_Program_Email	HandsMen Customer	11/28/2025
Edit   Del	Order Confirmation Email Alert	Order_Confirmation_Email	HandsMen Order	11/28/2025

## Phase 2: Salesforce Development - Backend & Configurations

This phase involved the actual construction and configuration of the Salesforce environment based on the architecture defined in Phase 1.

### 1. Object and Field Creation

I built the custom objects and fields in the Object Manager. I configured page layouts to ensure the interface is user-friendly for the tailors and store managers.

The screenshot shows the Salesforce Object Manager page with a list of custom objects. The objects listed are Marketing Campaign, Inventory, HandsMen Order, HandsMen Product, and HandsMen Customer. Each row includes columns for Label, API Name, Type, Description, Last Modified, and Deployed status.

LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED
Marketing Campaign	Marketing_Campaign_c	Custom Object		11/28/2025	✓
Inventory	Inventory__c	Custom Object		11/28/2025	✓
HandsMen Order	HandsMen_Order__c	Custom Object		11/28/2025	✓
HandsMen Product	HandsMen_Product__c	Custom Object		11/28/2025	✓
HandsMen Customer	HandsMen_Customer__c	Custom Object		11/28/2025	✓

Fields & Relationships				
FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Email	Email_c	Email		
FirstName	FirstName_c	Text(60)		
FullName	FullName_c	Formula (Text)		
HandsMen Customer Name	Name	Text(60)		
Last Modified By	LastModifiedById	Lookup(User)		
LastName	LastName_c	Text(60)		
Loyalty Status	Loyalty_Status_c	Picklist		
Owner	OwnerId	Lookup(User,Group)		
Phone	Phone_c	Phone		
Total Purchases	Total_Purchases_c	Number(18, 0)		

## 2. Implement Automation

- Flows:** I utilized Salesforce Flow to handle email scheduling. When a customer confirms an order, the flow automatically sends an email confirmation and updates the customer record.
- Apex Triggers:** Developed an Apex Trigger on the Handsmen\_Order\_c object. When a new order is placed, the trigger checks the available quantity in the Handsmen\_Product\_c inventory. It automatically deducts the ordered quantity from the inventory levels to keep stock accurate, and blocks the transaction if the stock is insufficient.

The screenshot shows the Salesforce Flow Builder interface for a flow named "Order Confirmation - V1". The flow is triggered by "A record is updated" on the "HandsMen Order" object. The flow starts with a "Record-Triggered Flow Start" step, which triggers a "Run Immediately" action. This leads to an "Order Confirmation Email Alert" step, which then concludes with an "End" step. On the right side of the screen, there is a "Configure Start" panel where the condition for triggering the flow is set to "Status Equals Confirmed". Below this, there is a section titled "When to Run the Flow for Updated Records" with two options: "Every time a record is updated and meets the condition requirements" and "Only when a record is updated to meet the condition requirements". The latter option is selected. At the bottom right, there is a "Activate Windows" message: "Actions and Related Records Go to Settings to activate Windows. Update any record" with a "Tips" link.

The screenshot shows the Salesforce IDE interface with the file 'StockDeductionTrigger.apxc' open. The code implements a trigger on the 'HandsMen\_Order\_\_c' object. It iterates through new and updated orders, checking if their status is 'Confirmed' and they have a product assigned. If so, it adds the product ID to a set. After processing all orders, it queries related inventories for these products using a map query. The code is written in Apex.

```
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]
    );
}
```

### 3. Develop Batch Jobs

I developed a Batch Apex class named `InventoryBatchJob` that implements the `Schedulable` interface to ensure inventory stability. This job is scheduled to run periodically to scan for products with low stock levels (less than 10 units). When identified, the system automatically simulates a restocking process by increasing the `Stock_Quantity__c` by 50 units, ensuring that popular fabrics are always available for new orders.

The screenshot shows the Salesforce IDE interface with the file 'InventoryBatchJob.apxc' open. The code defines a global class `InventoryBatchJob` that implements `Database.Batchable<SObject>` and `Schedulable`. It starts by querying products with a stock quantity less than 10. For each record found, it increases the stock quantity by 50. The code is written in 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) {
        for (Product__c record : (List<Product__c>)records) {
            record.Stock_Quantity__c += 50;
            update record;
        }
    }
}
```

### 4. Data Security and Sharing Rules

- Sharing Rules: Created rules to assign “Sales Permission Set” with the “Sales” role, while “Marketing Permission Set” are only assigned with the “Marketing” role.

The screenshot shows the Salesforce Setup interface under the 'Permission Sets' section. The search bar at the top has 'perm' entered. The main area displays the 'marketing permission set' with the title 'marketing permission set'. Below it, the 'Current Assignments' table lists one user assignment: Daniel Mikaelson, who is active, has the Role 'Marketing', is assigned to Profile 'Platform 1', and has the User License 'Salesforce'. The table includes columns for Full Name, Active, Role, Profile, User License, and Expires On.

The screenshot shows the Salesforce Setup interface under the 'Permission Sets' section. The search bar at the top has 'perm' entered. The main area displays the 'sales permission set' with the title 'sales permission set'. Below it, the 'Current Assignments' table lists one user assignment: Niklaus Mikaelson, who is active, has the Role 'Sales', is assigned to Profile 'Platform 1', and has the User License 'Salesforce'. The table includes columns for Full Name, Active, Role, Profile, User License, and Expires On.

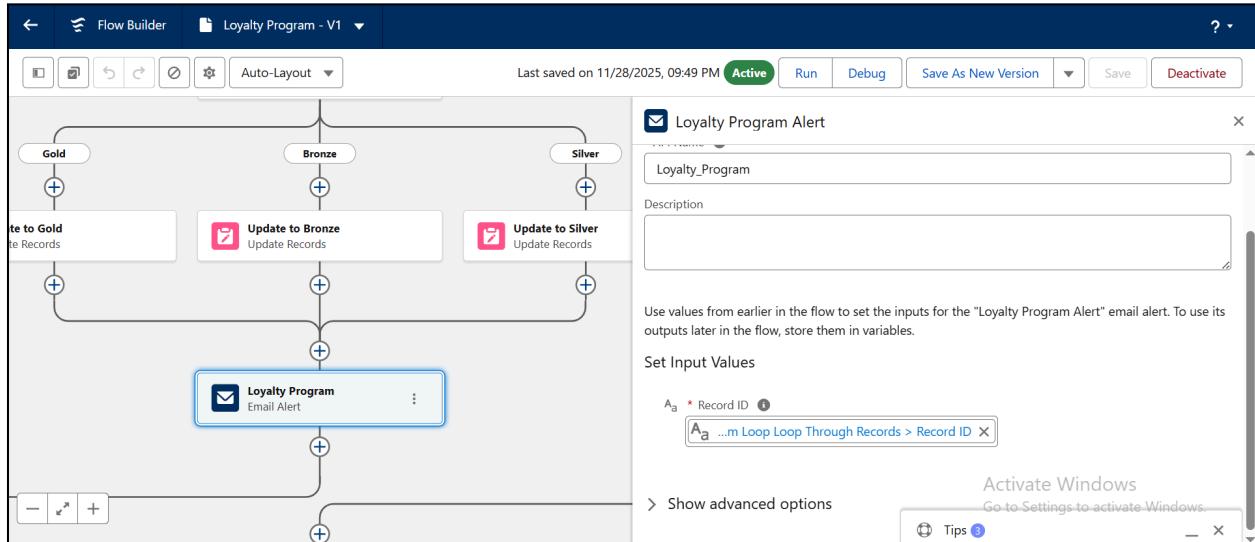
## 5. Configure Email Templates & Notifications

I implemented the HTML email templates and set up Email Alerts within our Flows to send them automatically upon specific triggers.

The screenshot shows the Salesforce Setup interface under the 'Email' section. The search bar at the top has 'classic' entered. The main area displays the 'Classic Email Templates' page, specifically the 'Loyalty Program Email' template. The template subject is 'Loyalty Program Email'. The 'HTML Preview' section contains the following text:

Congratulations! You are now a  
{!HandsMen\_Customer\_\_c.Loyalty\_Status\_\_c} 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.

At the bottom right of the page, there is a message: 'Activate Windows  
Go to Settings to activate Windows.'

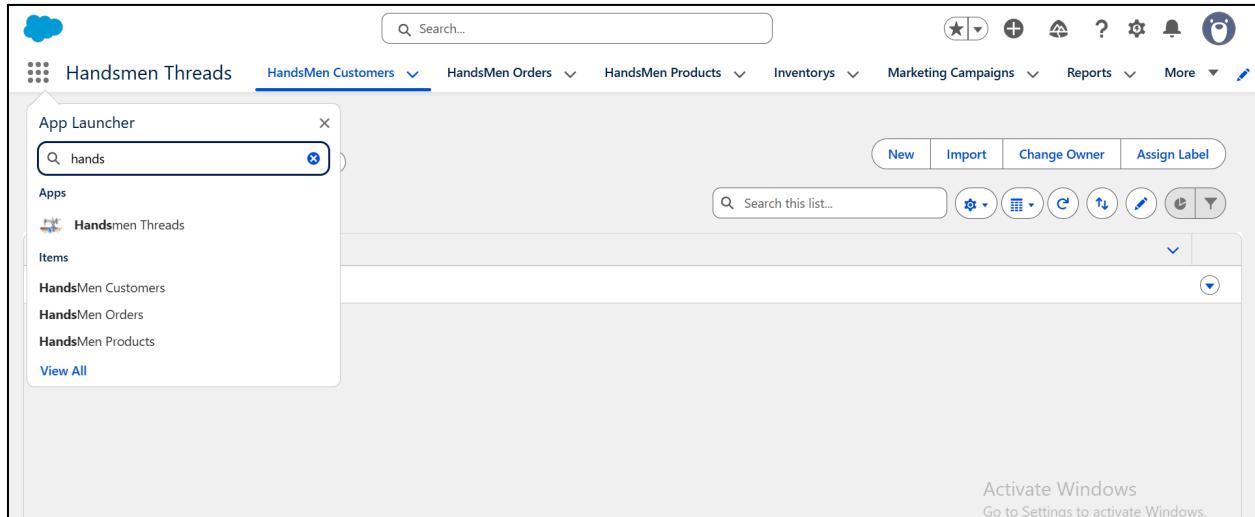


## Phase 3: UI/UX Development & Customization

The screenshot shows the Salesforce Lightning Experience App Manager interface with the following details:

- Setup:** Selected tab.
- Home:** Home tab.
- Object Manager:** Object Manager tab.
- Search:** Search bar.
- App Manager:** Selected under Apps.
- Lightning Experience App Manager:** Main title.
- Table:** List of installed apps (32 items).
 

App Name	Developer Name	Description	Last Modified	App Type
Data Loader Bulk	DataLoader_Bulk	The Data Loader is an easy to use graphical tool that helps you to get your data into Salesforce objects.	11/6/2025, 3:55 PM	Connected (Managed)
Data Loader Partner	DataLoader_Partner	The Data Loader is an easy to use graphical tool that helps you to get your data into Salesforce objects.	11/6/2025, 3:55 PM	Connected (Managed)
Digital Experiences	SalesforceCMS	Manage content and media for all of your sites.	11/6/2025, 3:55 PM	Lightning
Force.com IDE	Forcecom_IDE	The Force.com IDE is a powerful client application for creating, modifying and deploying Force.com ap...	11/6/2025, 3:55 PM	Connected (Managed)
Handsmen Threads	Handsmen_Threads		11/28/2025, 7:49 PM	Lightning
Lightning Usage A...	LightningInstrumentation	View Adoption and Usage Metrics for Lightning Experience	11/6/2025, 3:55 PM	Lightning
Marketing CRM CL...	Marketing	Track sales and marketing efforts with CRM objects.	11/6/2025, 3:55 PM	Classic
My Service Journey	MSIApp	Discover new customer service capabilities.	11/6/2025, 4:03 PM	Lightning
Platform	Platform	The fundamental Lightning Platform	11/6/2025, 3:55 PM	Classic
Playground Starter	Playground_Starter	Get started with your Trailhead Playground.	11/6/2025, 3:55 PM	Lightning (Managed)
Sales	Sales	The world's most popular sales force automation (SFA) solution	11/6/2025, 3:55 PM	Classic
Sales	LightningSales	Manage your sales process with accounts, leads, opportunities, and more	11/6/2025, 3:55 PM	Lightning
Sales Cloud Mobile	SalesCloudMobile	New seller focused mobile first experience	11/6/2025, 3:55 PM	Lightning
Sales Console	LightningSalesConsole	(Lightning Experience) Lets sales reps work with multiple records on one screen	11/6/2025, 3:55 PM	Lightning
Salesforce Chatter	Chatter	The Salesforce Chatter social network, including profiles and feeds	11/6/2025, 3:55 PM	Classic
- Notes:** Go to Settings to activate Windows



## Phase 4: Data Migration, Testing & Security

In this phase, I validated the system architecture to ensure that the custom objects (HandsMen Customer\_\_c, HandsMen Product\_\_c, etc.) function correctly and that data integrity is maintained through formulas and automation.

### 1. Data Quality & Integrity Implementation

To automate data formatting and provide instant visual feedback on inventory levels, we implemented the following formula fields:

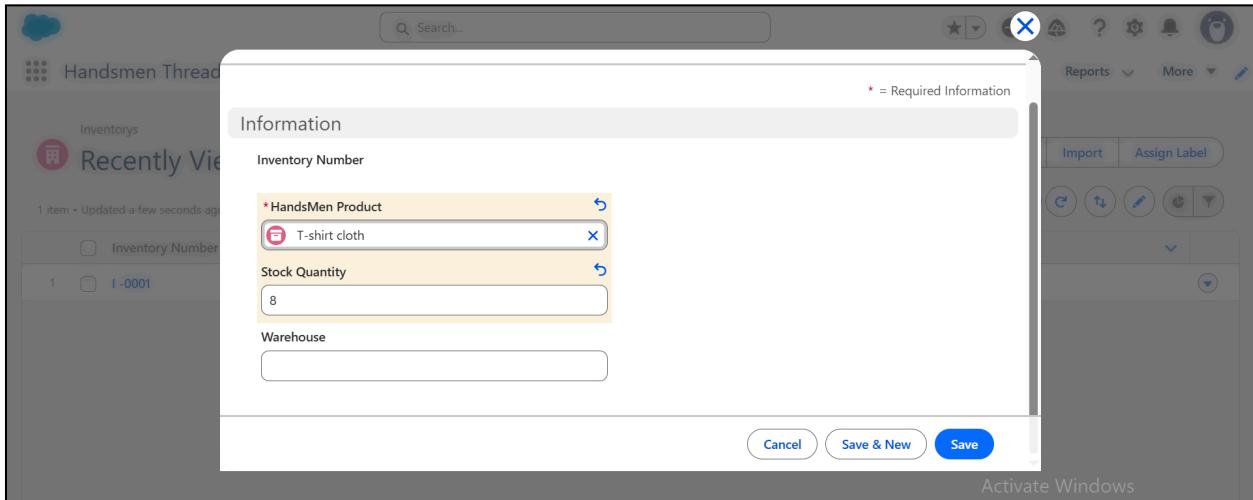
- **Inventory Status Indicator:** I created a formula field Stock\_Status\_\_c on the Inventory\_\_c object. This text formula automatically flags items as “Low Stock” if the quantity drops to 10 or below, allowing the warehouse manager to spot shortages immediately without running a report.
  - *Formula Logic:* IF(Stock\_Quantity\_\_c < 10, “Available”, “Low Stock”)
- **Customer Name Concatenation:** On the HandsMen Customer\_\_c object, I created the Full\_Name\_\_c formula to combine the First and Last names into a single readable field for easier search and reporting.
  - *Formula Logic:* FirstName & " " & LastName

### 3. Functional Test Cases & Results

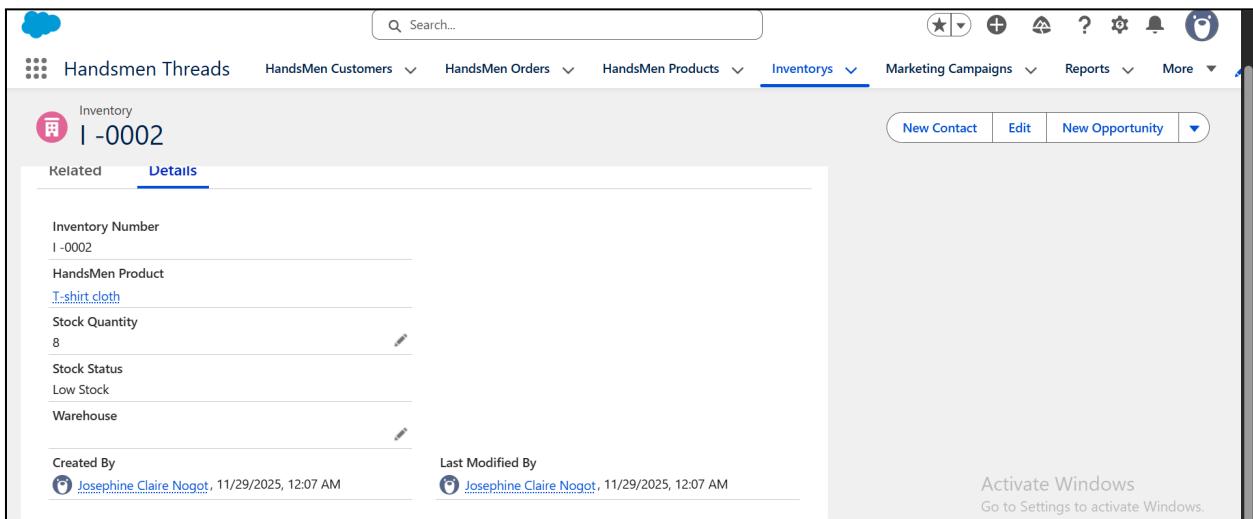
I executed functional tests to verify the formulas and object behaviors defined in the schema.

#### Test Case 1: Inventory Status Formula Logic

- **Objective:** Verify that the Stock\_Status\_\_c field on the **Inventory\_\_c** object correctly updates based on the Stock\_Quantity\_\_c.
- **Step 1 (Input):** User creates a new Inventory record and enters 8 in the Stock\_Quantity\_\_c field.



- **Step 2 (Output):** Upon saving, the Stock\_Status\_\_c field automatically displays “Low Stock”.



## Test Case 2: Customer Name Concatenation

- **Objective:** Verify that the Full\_Name\_\_c formula on **HandsMen Customer\_\_c** works as expected.
- **Step 1 (Input):** User creates a new customer record with First Name: “Daniel” and Last

Name: “Padilla”.

The screenshot shows a Salesforce Lightning Record Edit screen for a customer record. The record ID is 003f00000000000. The form fields include:

- Phone: (empty)
- Loyalty Status: --None--
- FirstName: Daniel
- LastName: Padilla
- Total Purchases: (empty)

At the bottom right are three buttons: Cancel, Save & New, and Save. The Save button is highlighted in blue. A watermark at the bottom right says "Activate Windows Go to Settings to activate Windows."

- **Step 2 (Output):** The system saves the record, and the Full\_Name\_\_c field displays “Daniel Padilla”.

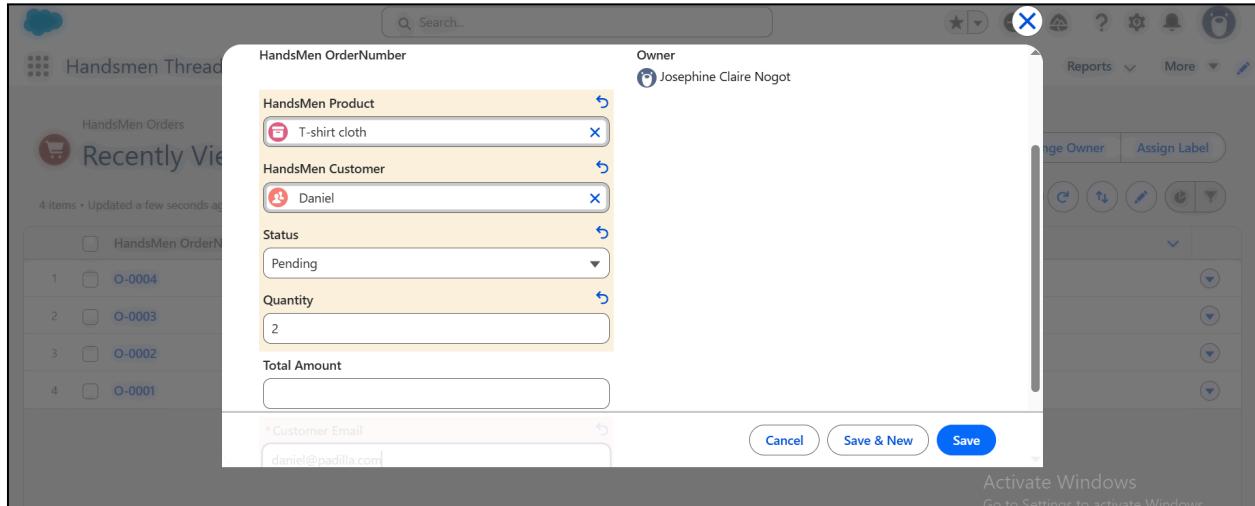
The screenshot shows a Salesforce Lightning Record View screen for the customer record from the previous step. The record ID is now 003f00000000000. The view shows the following data:

Field	Value
Phone	(empty)
Loyalty Status	(empty)
FirstName	Daniel
LastName	Padilla
FullName	Daniel Padilla
Total Purchases	(empty)
Created By	Josephine Claire Nogot, 11/29/2025, 12:09 AM
Last Modified By	Josephine Claire Nogot, 11/29/2025, 12:09 AM

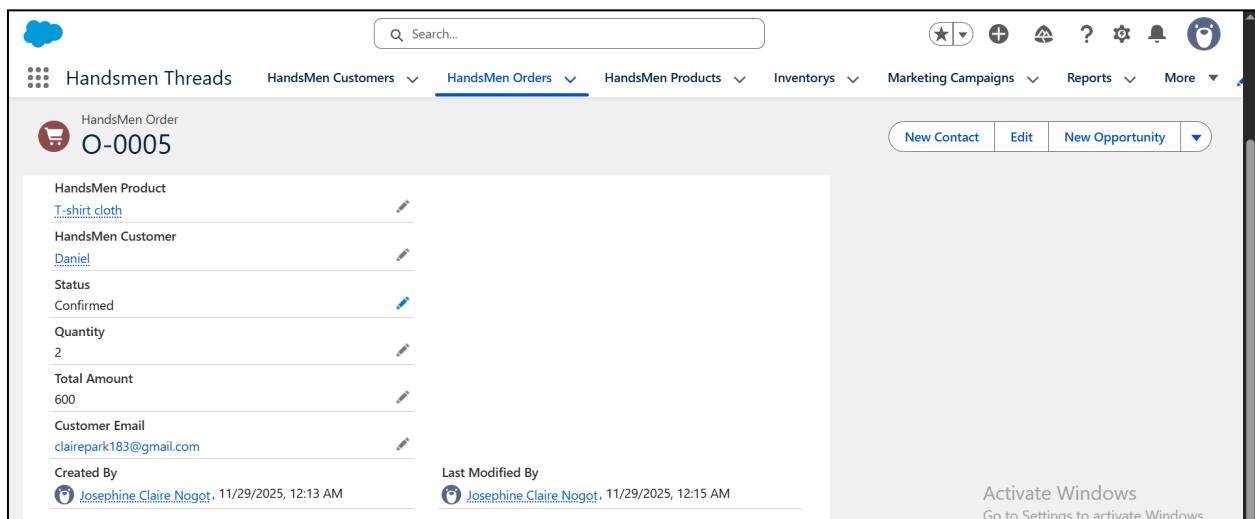
A watermark at the bottom right says "Activate Windows Go to Settings to activate Windows."

### Test Case 3: Order Creation & Status Flow

- **Objective:** Verify that a **HandsMen Order\_\_c** can be created and tracked via the Status picklist.
- **Step 1 (Input):** User creates a new order, linking it to a customer, setting Quantity\_\_c to 2, and Status to “Pending”.



- **Step 2 (Output):** The record is saved successfully. Later, the user edits the Status to “Confirmed”.



## Phase 5: Deployment, Documentation & Maintenance

When a user reports an error (e.g., “I can't save this Order”), I follow this standardized troubleshooting workflow:

1. **Reproduce the Issue:** Log in as the user (using “Login As”) to see if the error is replicable.
2. **Check Validation Rules:** 90% of save errors are due to Validation Rules. We check if the data input violates rules like the “Low Stock” trigger.
3. **Review Debug Logs:** If the error is technical (like an Apex Trigger failure), we turn on Debug Logs for that user to trace the specific line of code causing the exception.
4. **Flow Fault Emails:** We configured the system to send an email to the Admin whenever a

Flow fails, providing immediate context on what went wrong.

## Conclusion

The Handsmen Threads Capstone Project successfully achieved its primary objective: transforming a traditional, manual bespoke tailoring operation into a modernized, digital-first business.

By implementing Salesforce, we addressed the critical pain points of data redundancy and inventory mismanagement. The custom architecture, specifically the HandsMen Customer\_\_c and HandsMen Order\_\_c objects, now provides a “Single Source of Truth” for client measurements and transaction history. The automation we built does more than just save time; the Stock Validation Trigger actively prevents revenue loss by stopping orders for out-of-stock fabrics, while the InventoryBatchJob ensures the warehouse is proactive rather than reactive.

This project demonstrates that Salesforce is not just for large enterprises but can be effectively tailored to small businesses. The system is scalable, secure, and ready to support Handsmen Threads as they expand their customer base, providing a solid foundation for future enhancements like customer self-service portals and automated marketing journeys.