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

---

1. Abstract

This document provides a comprehensive overview of the **"HandsMen Threads"** project, a sophisticated fashion management platform developed on the Salesforce Lightning Platform. The project's core purpose is to create a unified and efficient system for managing a bespoke men's fashion business, covering the entire lifecycle from customer engagement and order processing to inventory management and marketing. By architecting a custom data model, implementing robust process automation, and establishing a granular security framework, the HandsMen Threads platform streamlines operations, enhances the customer experience with personalized services, and provides management with clear, real-time visibility into business performance. This solution serves as a foundational asset, designed to scale with the business and elevate its standard of service and sophistication.

2. Objective

The principal objective of the HandsMen Threads project is to engineer a centralized, all-in-one digital solution to manage the complexities of a custom tailoring business. The specific, measurable goals are:

- **To Centralize Data Management:** Establish a single source of truth for all business data, including customer profiles, order histories, product catalogs, inventory levels, and marketing initiatives, thereby eliminating data silos.
- **To Automate Key Business Processes:** Significantly reduce manual data entry and operational overhead by automating critical tasks such as order total calculations, stock level adjustments, and customer status updates.
- **To Enhance Customer Relationship Management:** Deliver a superior customer experience by maintaining a 360-degree view of each client, enabling personalized styling, tracking loyalty, and facilitating targeted communication.
- **To Enforce Data Integrity and Security:** Implement a robust security model to ensure that users can only access data relevant to their roles. Enforce data quality standards through systemic validation rules to maintain the accuracy and reliability of business information.
- **To Provide Actionable Insights:** Enable data-driven decision-making by structuring data in a way that allows for easy reporting and analysis of sales trends, inventory status, and customer behaviour.

3. Technology Description

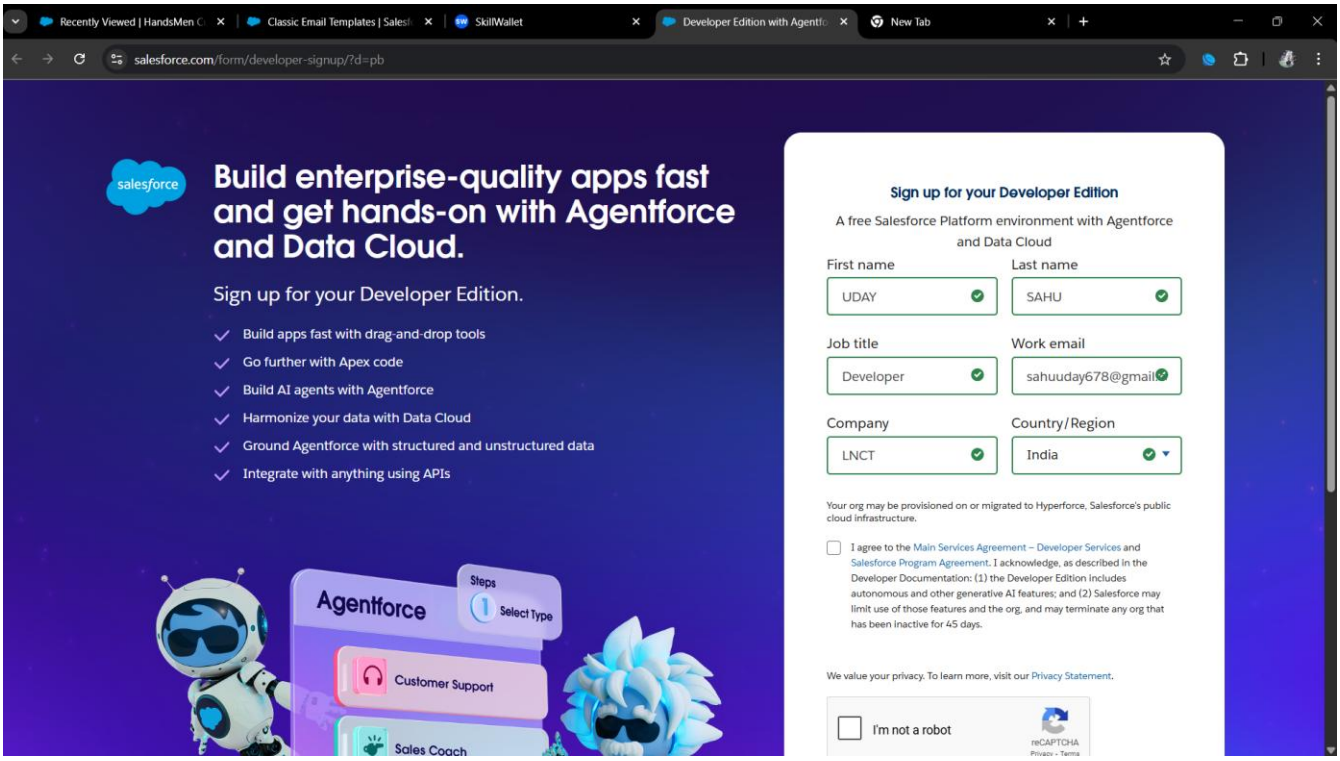
The HandsMen Threads platform is built exclusively using Salesforce technology, leveraging a combination of its declarative (point-and-click) and programmatic capabilities to deliver a highly customized application.

- **Custom Object:** The architectural foundation of the application. Custom objects are created to function as dedicated database tables for storing business-specific information that doesn't fit into Salesforce's standard objects.
- **Custom Tab:** A user interface component that provides a dedicated home page for a custom object, allowing users to easily view, create, and manage records of that object type.
- **Custom App (Lightning App):** A branded collection of tabs, items, and utilities that provides a cohesive and task-specific workspace for users, such as the "HandsMen Threads" app for all business operations.
- **Roles:** A feature of the Salesforce sharing model that defines a user's position in the organizational hierarchy. Roles are primarily used to control data visibility, ensuring that managers have access to the records owned by their subordinates.
- **Fields:** The basic building blocks of an object, used to store individual data points on a record. This includes various types like Text, Number, Currency, Picklist, and Formula fields for calculated values.
- **Permission Sets:** A flexible tool for granting users additional permissions and access beyond what their profile allows. They are used to add specific capabilities (e.g., access to a new object) without changing the user's base profile.
- **Validation Rules:** Formulas that run when a user attempts to save a record. They verify that the data entered meets specified standards and prevent the save operation if the data is invalid, thus ensuring data integrity.
- **Data Configuration:** The overall process of designing and building the data model, which includes defining objects, creating fields, establishing relationships between objects (e.g., Lookups), and setting up page layouts.
- **Flow:** Salesforce's primary declarative automation tool. It is used to build complex, guided processes and record-triggered automations without writing code, such as sending email alerts or updating fields based on specific criteria.
- **Apex:** Salesforce's strongly-typed, object-oriented programming language. It is used to implement complex business logic that cannot be achieved with declarative tools. **Apex Triggers** are scripts that execute in response to record events (like save or delete), while **Batch Jobs** are used for processing large volumes of data asynchronously.

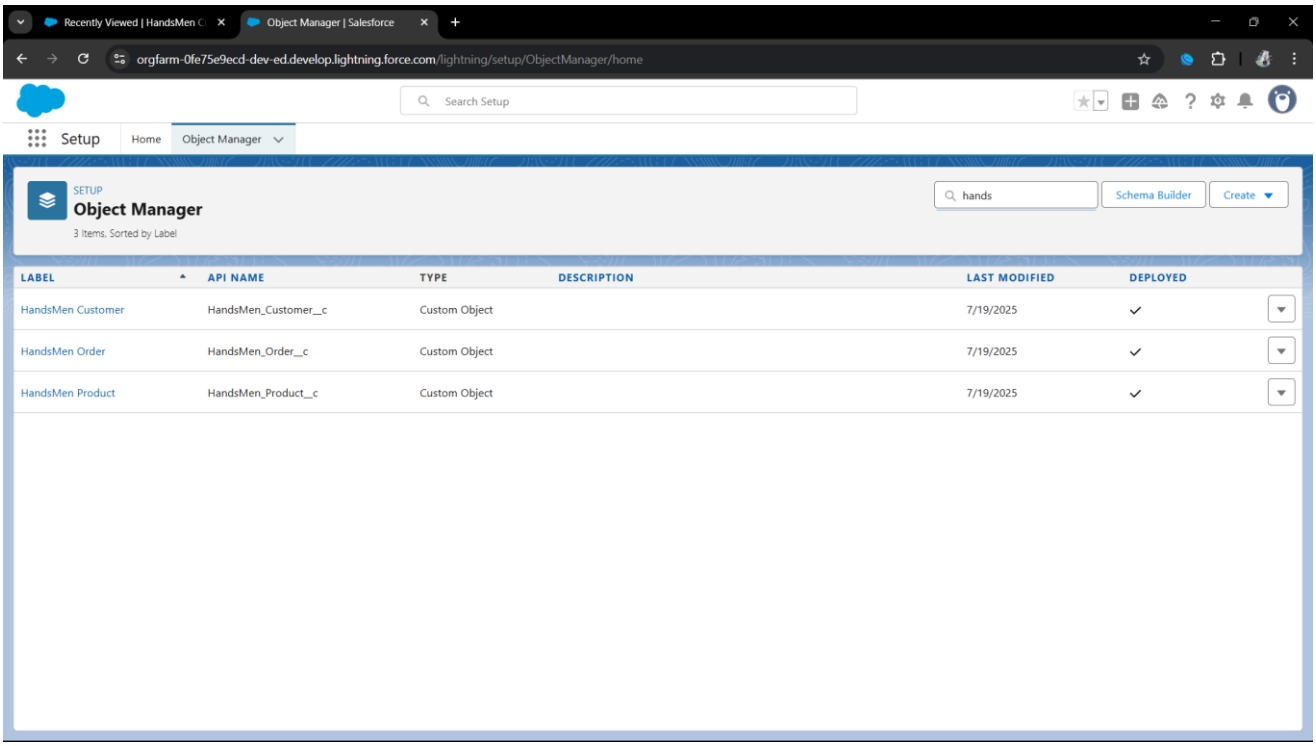
4. Detailed Execution of Project

The project was executed through a systematic, multi-phased approach, building from the foundational data model up to the complex automation layers.

- **Salesforce Credentials Setup:** The project began by establishing a dedicated development environment. A Salesforce Developer Edition org was created, providing a safe, isolated space to build and test all components without impacting a live production environment. This involved signing up for an account and performing the initial security setup.

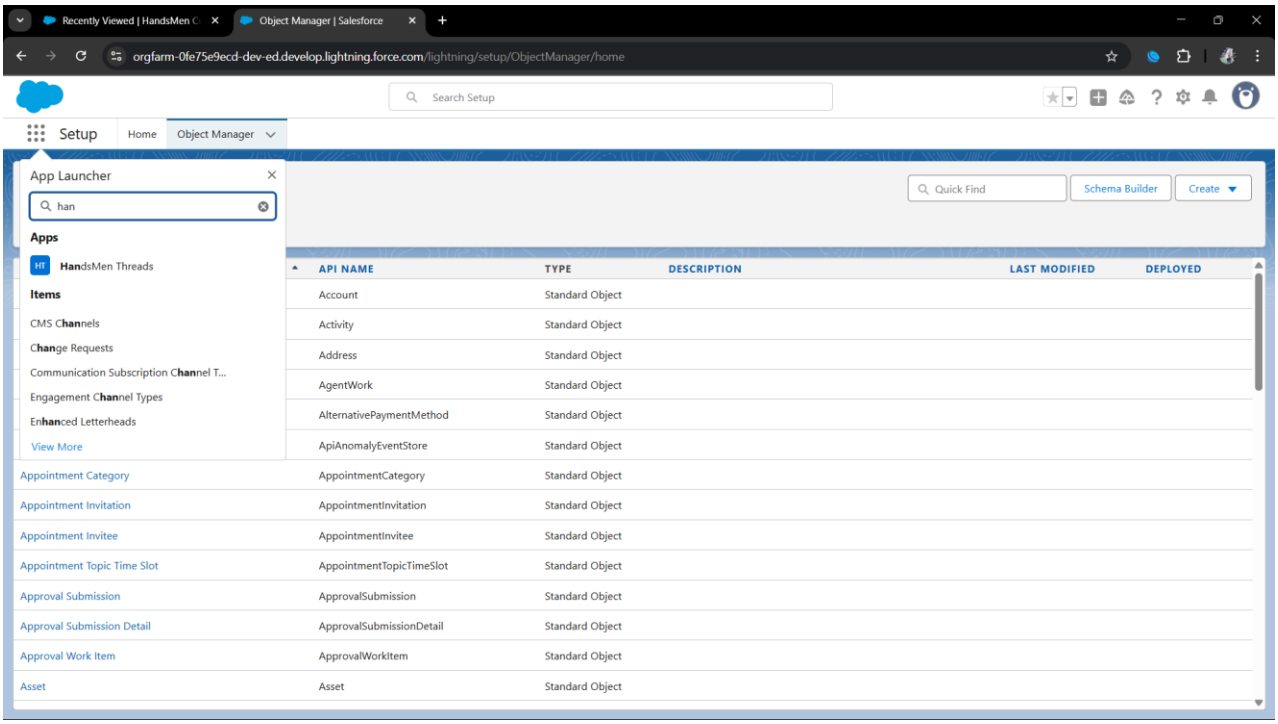


- **Data Management - Objects:** The core data structure was defined by creating five custom objects. This step was critical for tailoring the Salesforce environment to the specific needs of a fashion business:
  1. **HandsMen Customer__c:** To store detailed client information beyond a standard Contact.
  2. **HandsMen Order__c:** To track every bespoke order, its status, and associated details.
  3. **HandsMen Product__c:** To catalog the types of fabrics, suits, and accessories offered.
  4. **Inventory__c:** To manage stock levels of raw materials (e.g., fabric) and finished goods.
  5. **Marketing Campaign__c:** To plan and track the performance of marketing initiatives.



- **Data Management - Tabs:** To make the custom objects accessible to users, custom tabs were created for each. These tabs were then added to the application's navigation bar, allowing users to click a tab to see list views of records and create new ones.

- **Data Management - App Manager:** A dedicated Lightning App named "HandsMen Threads" was constructed. This app acts as a container, grouping all the relevant custom tabs and functionalities into a single, logical workspace, ensuring users have a streamlined and focused user experience.

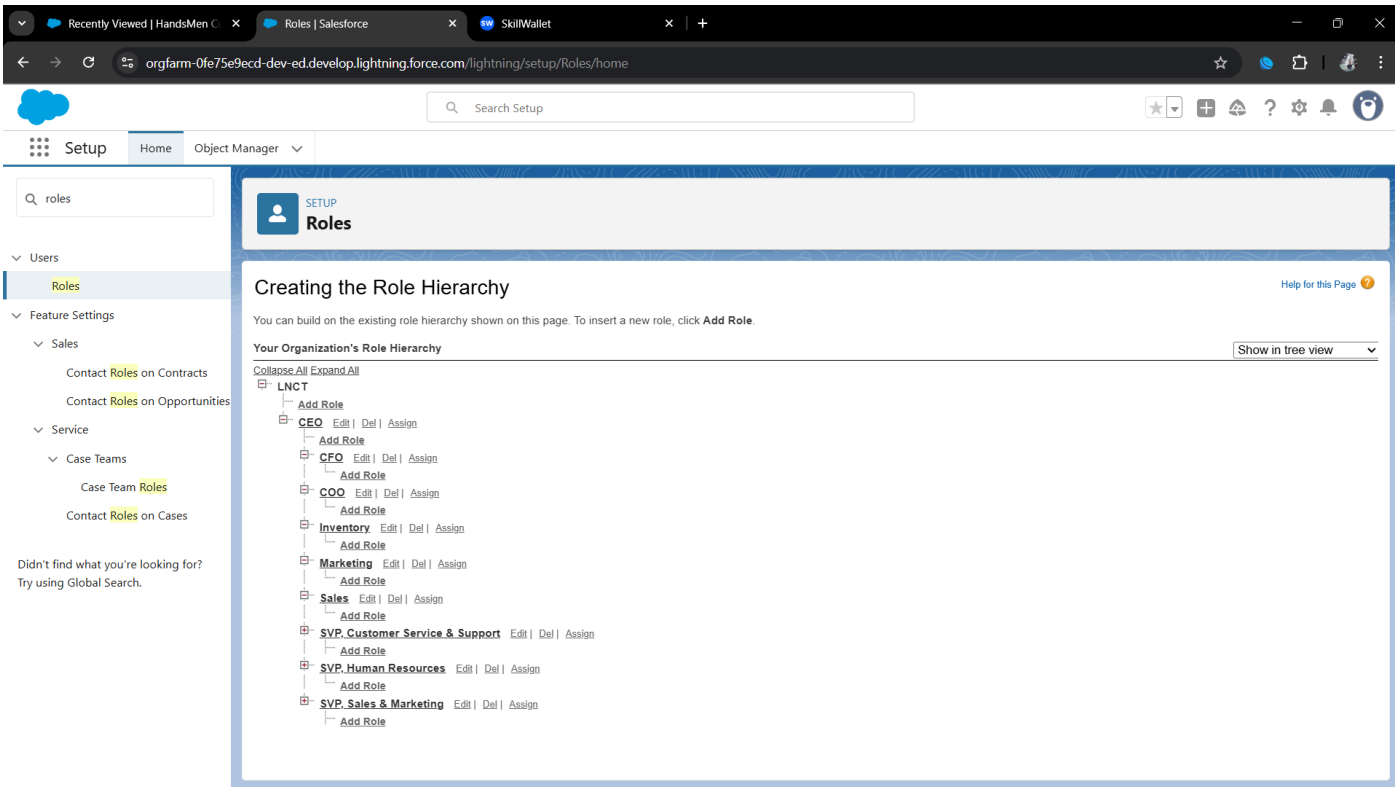


- **Data Management - Fields:** Each custom object was populated with custom fields to capture necessary data points. For instance, a "Loyalty Status" picklist was added to the Customer object, and a "Total Amount" currency field was added to the Order object. Lookup relationships were created to link objects, such as connecting an Order record to the specific Customer who placed it.

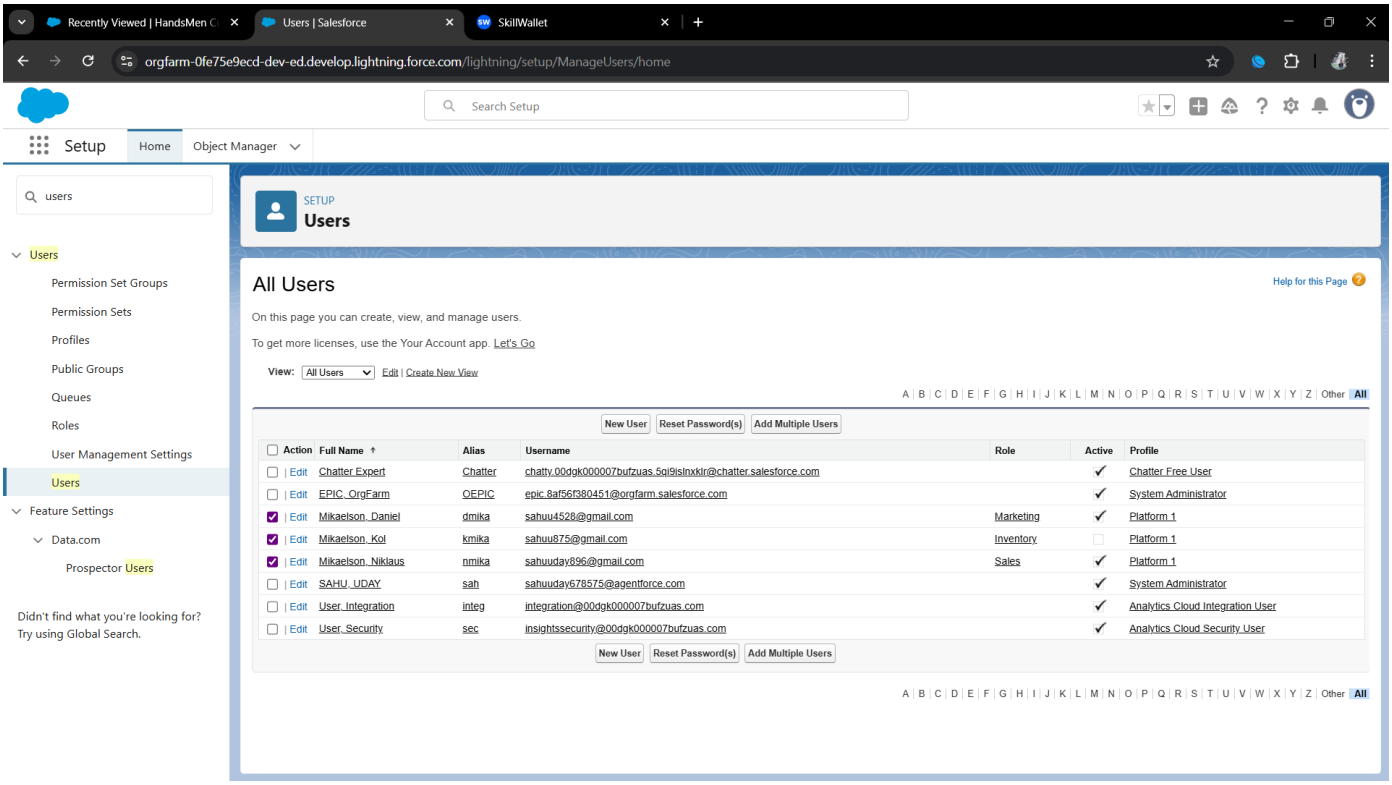
- **Data Configuration:** To enforce data quality, Validation Rules were configured. These rules act as the first line of defense against bad data. For example, a rule was set up to prevent users from saving an order with a negative total amount, ensuring financial data accuracy.

- **Data Security - Profiles:** A custom profile named "Sales" was created by cloning the "Standard User" profile. This follows the security principle of least privilege. Instead of modifying a standard profile, a custom one was created and tailored to grant the exact object and field permissions needed for a sales role.

- **Data Security - Roles:** A role hierarchy was established (CEO > Sales Manager, Inventory Manager, Marketing Manager). This structure is crucial for controlling data visibility. It ensures that a Sales Manager can, for example, view and report on the orders of all sales reps who report to them.

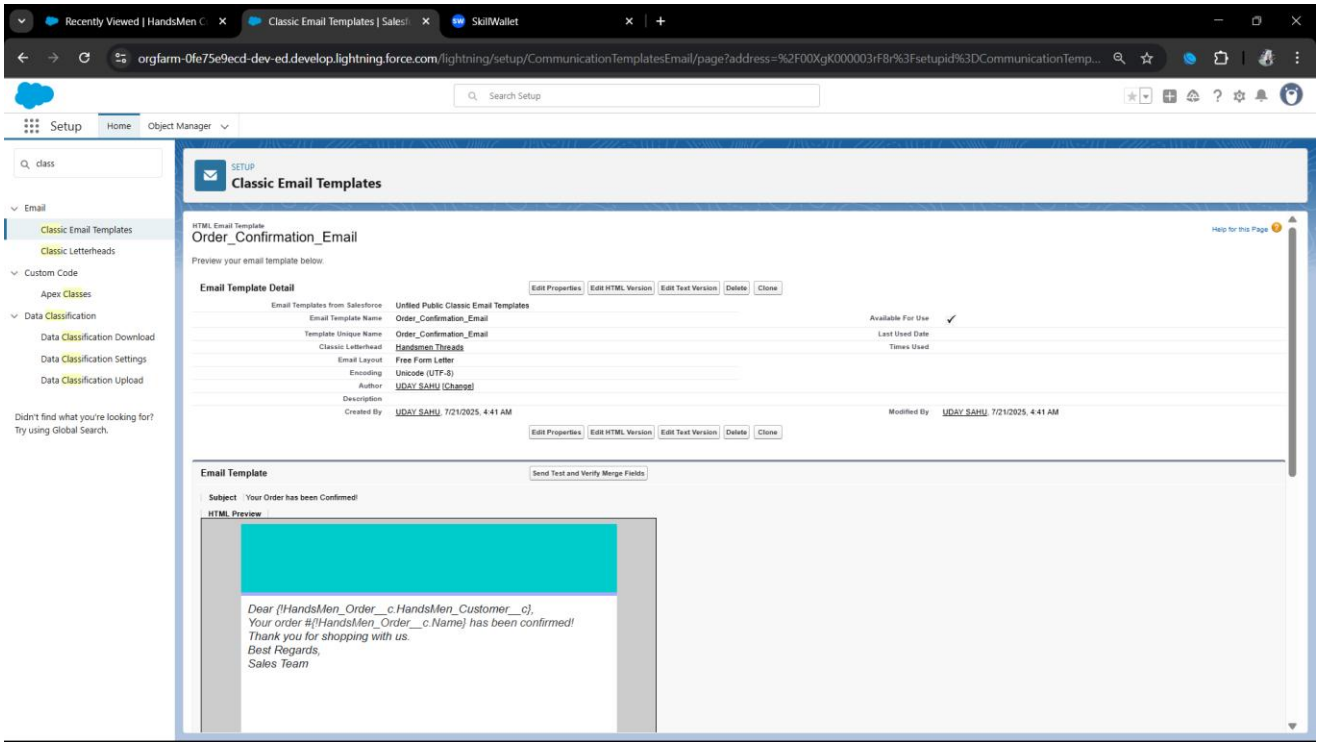


- **Data Security - Users:** Individual user accounts ("Niklaus," "Kol") were created. Each user was assigned the appropriate Profile and Role, which collectively define their access rights and what they can see and do within the application.

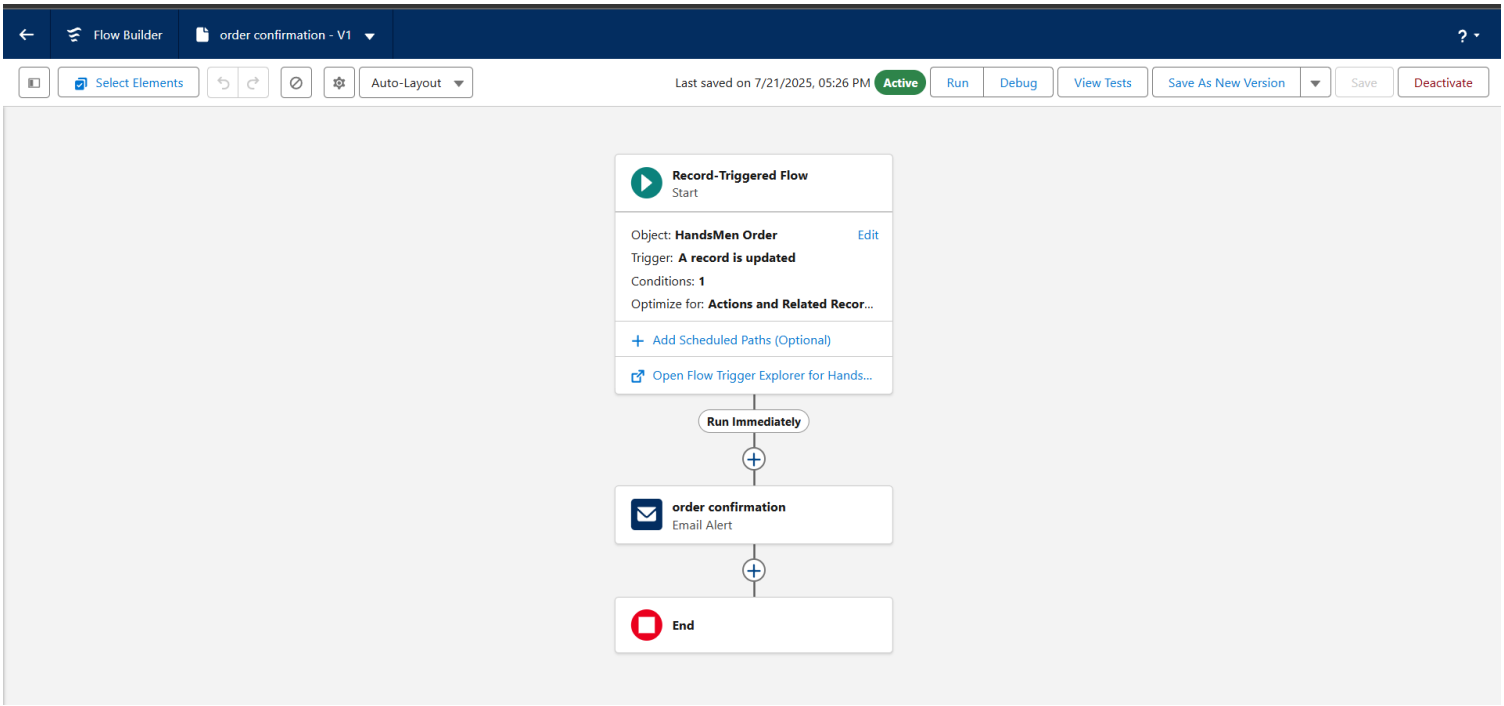


- **Data Security - Permission Set:** A Permission Set named Permission_Platform_1 was created to grant specific permissions (e.g., edit and delete access on Orders) in a modular way. This allows for granting extra permissions to specific users without having to create an entirely new profile for every minor variation in access requirements.

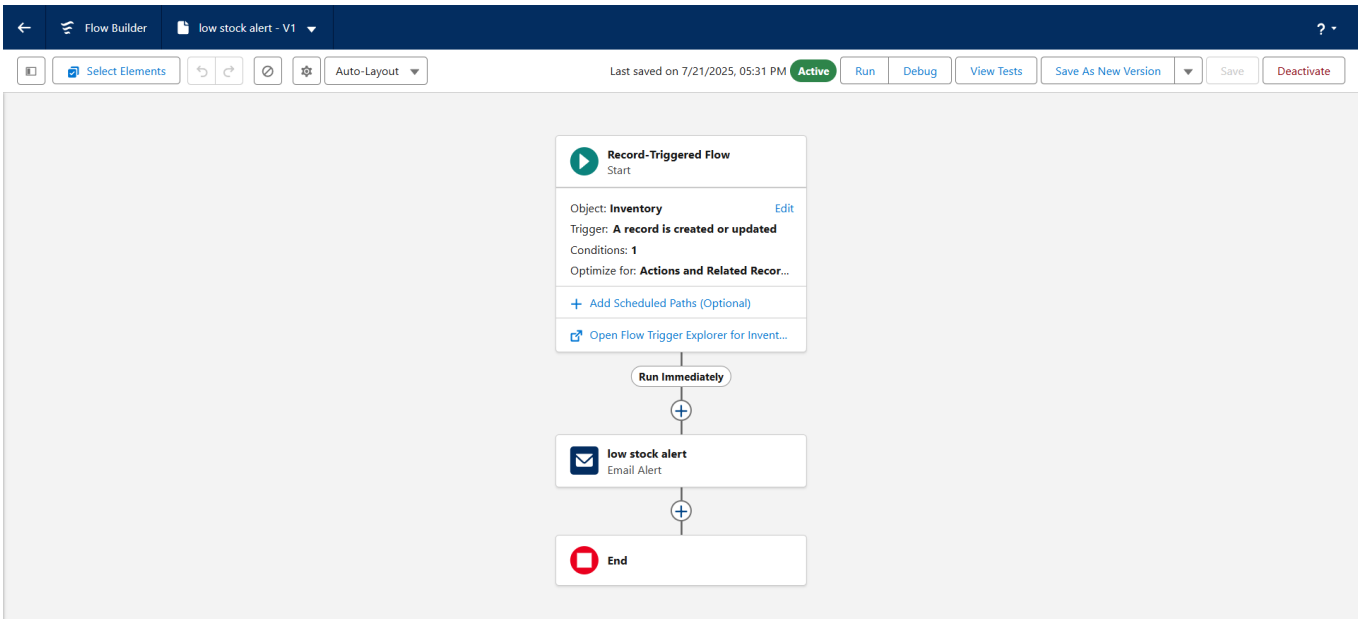
- **Email Template:** Professional email templates were designed for key customer communications. These templates use merge fields to pull data directly from Salesforce records, allowing for personalized messages, such as addressing a customer by name in an order confirmation email.



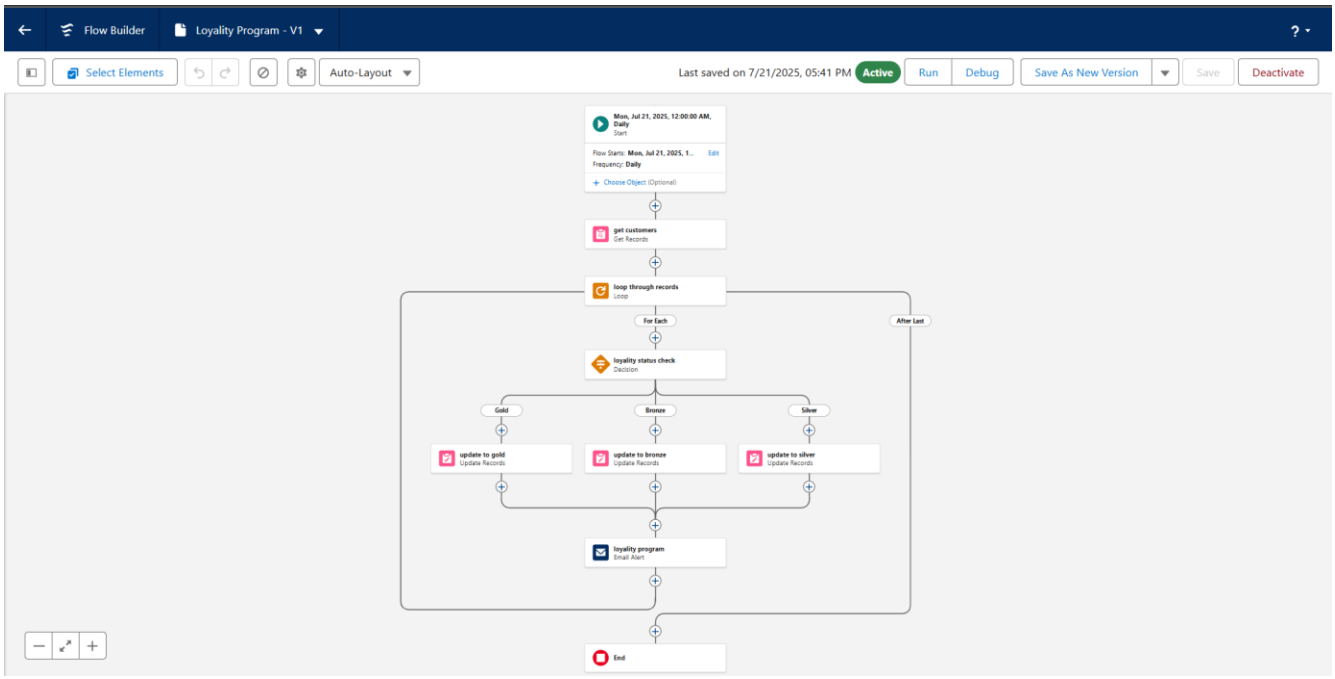
- **Flows:** Declarative automation was built using Flow Builder to handle key business processes. The following flows were implemented:
  - **Order Confirmation (Record-Triggered Flow):** To enhance customer communication, this flow triggers automatically the moment an order's status is changed to "Confirmed." It immediately sends a personalized "Order Confirmation" email to the customer, providing timely updates.



- **Low Stock Alert (Record-Triggered Flow):** To ensure proactive inventory management, this flow activates whenever a product's stock level drops below a specified threshold (e.g., 20 units). It sends an immediate "Low Stock Alert" notification to the inventory team, helping to prevent stockouts.



- **Loyalty Status Update (Scheduled Flow):** This flow runs automatically every day to maintain the customer loyalty program. It assesses each customer's total purchase amount and updates their loyalty status to Gold (>1000), Silver (500-1000), or Bronze (<500) accordingly, ensuring customer tiers are always current.



- **Automation using Apex:** For more complex business logic, **Apex Triggers** were developed. For example, the **OrderTotalTrigger** was written to automatically calculate the total amount of an order by summing up its line items before the record is saved. Another trigger, **StockDeductionTrigger**, was created to automatically deduct stock from inventory when an order is fulfilled.

File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >

OrderTotalTrigger.apxt

Code Coverage: None ▾ API Version: 64 ▾

```
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;
            }
        }
    }
}
```

File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >

StockDeductionTrigger.apxt

Code Coverage: None ▾ API Version: 64 ▾

```
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;
    }
}
```

- **Batch Jobs:** For processing large data volumes efficiently, Batch Apex jobs were created. A weekly batch job calculates loyalty points for all customers, and a daily job syncs inventory levels with an external warehouse system. These run during off-peak hours to avoid impacting system performance for users.

5. Project Explanation with Real-World Examples

The **HandsMen Threads** platform transforms daily operations through its integrated features:

- **Real-World Scenario 1: A New Client Consultation**
  A stylist, logged in as a "Sales" user, meets a new client. They open the HandsMen Threads App and create a new HandsMen Customer record. During the consultation, they create a new HandsMen Order record, linking it to the client. As they add items (e.g., 3 meters of 'Italian Wool' from the HandsMen Product catalog), the Apex trigger instantly updates the Total_Amount__c on the order. The stylist can see from the related Inventory record that the fabric is in stock. Once the client confirms, the stylist changes the order status to 'Confirmed'. This action immediately triggers two automations: the Order Confirmation Flow sends a branded HTML email to the client, and the Stock Deduction trigger reduces the quantity of 'Italian Wool' in the inventory system.

- **Real-World Scenario 2: Proactive Inventory and Marketing**
  The Inventory Manager, who has a different role and visibility, logs in and sees a "Low Stock Alert" email generated by the Stock Alert Flow. They proactively reorder the material. Meanwhile, a Marketing Manager logs in and creates a new Marketing Campaign for a summer collection. They use a report to pull a list of all HandsMen Customer records with a 'Gold' loyalty status. Their security settings allow them to read customer data but not edit it, ensuring data privacy while enabling targeted marketing.

- **Real-World Scenario 3: End-of-Week Reporting**
  On Friday, the CEO logs in. Thanks to the Role Hierarchy, they can view a single dashboard that rolls up all sales data from the Sales Manager's team. They can see the week's total revenue, the number of new customers acquired, and which products are selling best. Every Sunday morning, the Loyalty Points Batch Job runs automatically, ensuring that when the CEO reviews customer data on Monday, all loyalty points are up-to-date without anyone having to perform a manual calculation.

6. Conclusion

The **HandsMen Threads** project represents a successful digital transformation initiative. By leveraging the full spectrum of Salesforce's capabilities—from its flexible data model to its powerful automation and security features—a highly specialized and efficient platform has been constructed. The system effectively addresses the primary business objectives of streamlining operations, enhancing customer relationships, and ensuring data integrity. The resulting application is not merely a system of record but a strategic business asset that automates manual work, provides invaluable insights, and empowers the team to deliver a level of sophistication and service that matches the quality of their products. The platform is robust, scalable, and poised to support the continued growth and success of the HandsMen Threads brand.