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

Project Overview:

HandsMen Threads is a Salesforce-based Customer Relationship Management (CRM) solution designed to support a premium men's bespoke tailoring business. The system aims to digitize and streamline key business operations such as client profiling, appointment bookings, tailoring order management, and personalized customer communication.

The CRM addresses the limitations of manual record-keeping by centralizing customer data, tailoring preferences, measurements, and order statuses within a unified Salesforce platform. Through automation and structured workflows, HandsMen Threads enhances customer experience, operational efficiency, and service consistency while supporting the brand's commitment to sophistication and personalization.

Objectives:

The primary objective of this project is to develop a robust and scalable **Salesforce CRM** that enhances customer relationship management and optimizes bespoke tailoring operations. Specifically, the system aims to:

- Improve customer profiling by maintaining accurate and centralized customer records
- Streamline appointment booking and order tracking processes
- Automate notifications and approval workflows to reduce manual intervention
- Provide management with actionable insights through reports and dashboards

By achieving these objectives, the CRM delivers significant business value, including:

- Improved customer satisfaction
- Reduced manual errors
- More efficient and structured workflow management

Phase 1: Requirement Analysis & Planning

➢ **Understanding Business Requirements**

HandsMen Threads operates within the luxury men's fashion industry, where customer experience, accurate order handling, inventory reliability, and loyalty recognition are essential to business success. Prior to the implementation of Salesforce, key business data such as customer records, purchase history, inventory levels, and loyalty status were managed through fragmented and manual processes. This resulted in delayed order confirmations, inconsistent loyalty updates, reactive stock management, and limited visibility into overall business performance.

The organization required a centralized CRM solution capable of managing real-time transactions while ensuring data accuracy and process consistency. Critical business needs identified during this phase included immediate customer communication upon order confirmation, automated loyalty tier updates based on purchase behavior, proactive inventory monitoring to prevent stockouts, and reliable daily processing of bulk financial and inventory adjustments.

To address these requirements, the Salesforce CRM was designed to serve as a single source of truth, enabling structured data storage, automated business logic, and real-time operational visibility across sales, warehouse, and management teams. This approach ensures improved customer satisfaction, reduced operational risks, and a scalable foundation to support future growth and system enhancements.

- **Defining Project Scope and Objectives**

Based on the identified business requirements of HandsMen Threads, the project scope was clearly defined to focus on automating the most critical operations within the luxury men's fashion business. The primary objective at this stage was to ensure that Salesforce would effectively support real-time customer engagement, accurate inventory management, and reliable data processing.

The scope of the project includes:

- Development of custom and standard Salesforce objects to represent Customers, Orders, Products, Inventories, Loyalty Status, and Marketing Campaigns.
- Implementation of automation to support the four core business processes: order confirmation, loyalty tier updates, proactive stock alerts, and scheduled bulk order processing.
- Configuration of role-based security controls to protect sensitive customer, inventory, and financial data.
- Creation of reports and dashboards to provide actionable operational and management insights.

The project scope intentionally excludes external system integrations such as online payment gateways and third-party marketing platforms, which were identified as future enhancements beyond the current capstone implementation.

➢ Design of Data Model and Security Model

The Salesforce data model for HandsMen Threads was designed using custom objects to reflect the company's sales, inventory, and marketing processes. Core objects include **HandsMen Customer**, **HandsMen Order**, **HandsMen Product**, **Inventory**, and **Marketing Campaigns**, connected through clearly defined lookup and master-detail relationships. A one-to-many relationship allows each customer to place multiple orders, while products are linked to inventory records through a master-detail relationship to maintain accurate stock control. Marketing campaigns are associated with customers to support promotional tracking and reporting.

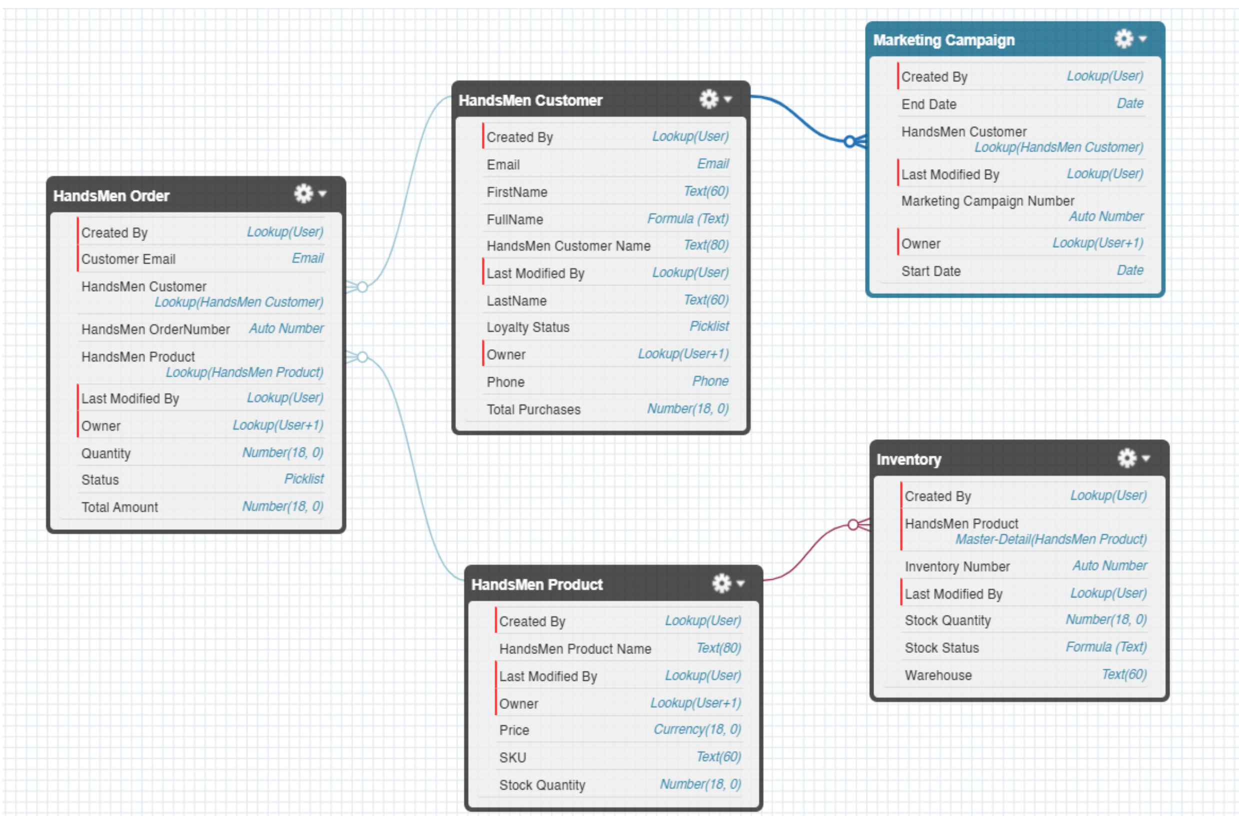


Figure 1. Object Relationship Diagram (ERD) – Salesforce Schema Builder

To protect business and customer information, a role-based security model was implemented using Salesforce profiles, role hierarchies, and permission sets. This design aligns data access with organizational roles and job responsibilities.

- **Profiles and Permission Sets** define user permissions, controlling which objects and fields users can view, create, edit, or delete.
- **Role Hierarchy** mirrors the organizational structure (CEO, Managers, Staff) and governs record visibility, allowing managers to access records owned by their subordinates.

This structured security model ensures data confidentiality and integrity, supports system scalability and automation, and prevents unauthorized access to sensitive information.

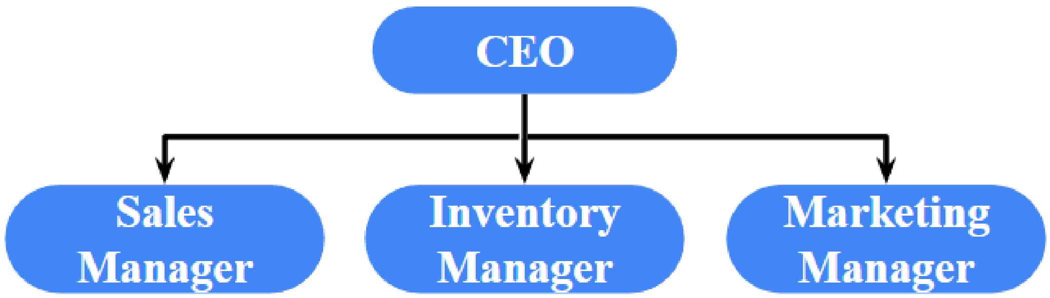


Figure 2. Role Hierarchy Diagram – Salesforce Role Setup

- **Stakeholder Mapping**

Key stakeholders were identified to ensure the CRM meets HandsMen Threads' business needs, with roles and access defined for secure and efficient operations.

- **Developer** – Designs, develops, and documents the CRM.
- **Instructor / Project Advisor** – Provides guidance, sets requirements, and evaluates deliverables.
- **End Users (Staff: Sales, Inventory, Marketing Managers)** – Manage customer records, sales, inventory, and marketing; access is role-based.
- **Customers (Indirect Users)** – Receive automated notifications like order confirmations and loyalty updates.
- **Salesforce Administrator** – Maintains the system, manages security, and implements future enhancements.

- **Execution Roadmap**

A structured execution roadmap was followed to guide the development of the HandsMen Threads Salesforce solution from planning to deployment. Each phase was implemented systematically to align with project objectives.

The roadmap included:

1. **Requirement Analysis and System Design** – Defining use cases, user scenarios, and user stories to map business processes.
2. **Salesforce Backend Configuration** – Setting up credentials, objects, fields, tabs, apps, and user access (profiles, roles, and permission sets).
3. **Automation and Flows** – Creating flows, Apex automation, and batch jobs to streamline business operations.

4. **Email Templates** – Developing automated communication for customer notifications.
5. **Reporting and Dashboards** – Designing insights for sales, inventory, and marketing operations.
6. **Data Testing, Validation, and Security Verification** – Ensuring data integrity, correct permissions, and role-based access.
7. **System Deployment and Technical Documentation** – Final rollout of the CRM along with complete project documentation.

Phase 2: Salesforce Development – Backend & Configurations

➢ Setup Environment & DevOps Workflow

The development of the HandsMen Threads Salesforce CRM was conducted entirely within a **Salesforce Developer Org**, providing a secure environment for configuration, development, and testing. Development followed a structured workflow using **task-based Kanban boards** to plan, execute, and track progress. All changes were implemented and tested in the Developer Org before being marked complete, ensuring systematic development, proper sequencing of tasks, and clear documentation of progress.

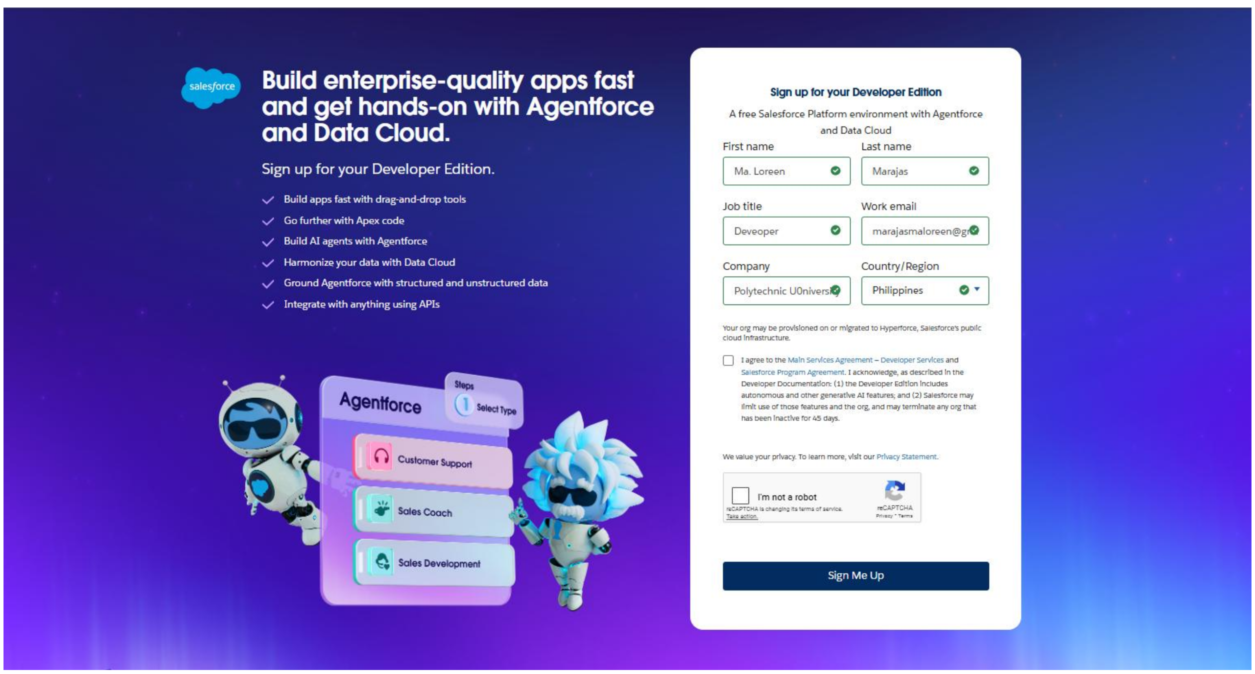


Figure 3.1. Salesforce Developer Org – Sign-Up Page

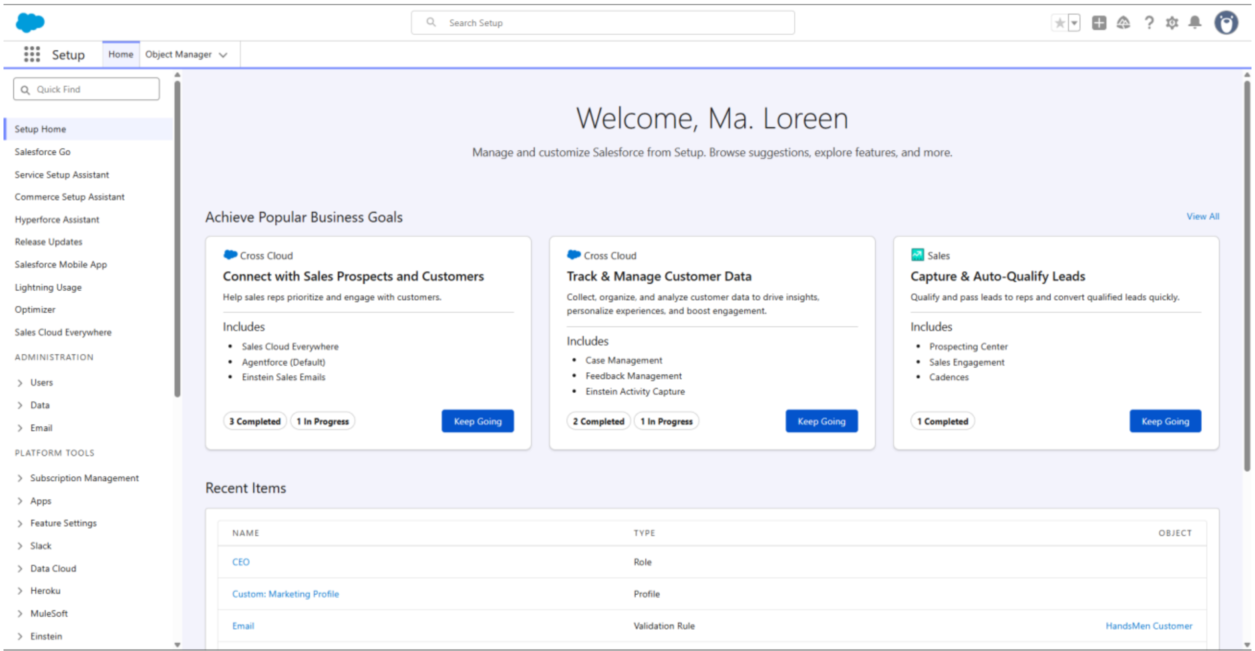


Figure 3.2. Salesforce Developer Org – Logged-In Home Page

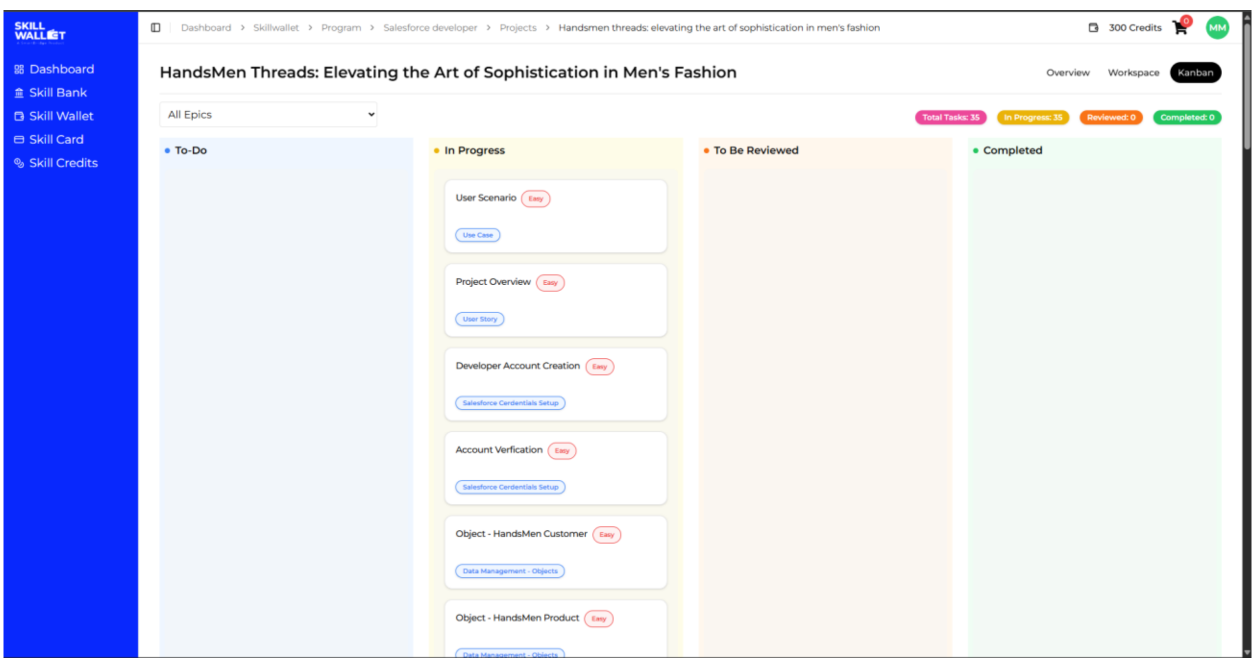


Figure 3.3. Kanban Board – Task Progress for HandsMen Threads Development

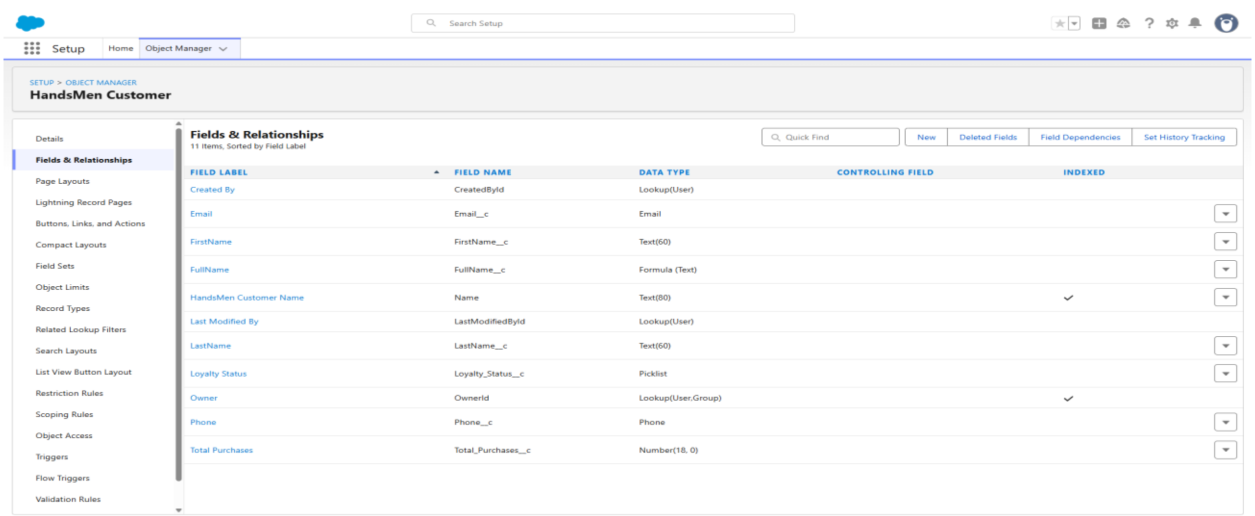
➢ **Customization of Objects, Fields, Validation Rules, Automation (Workflow Rules, Process Builder, Flows, Approval Process).**

- **Custom Objects and Fields**

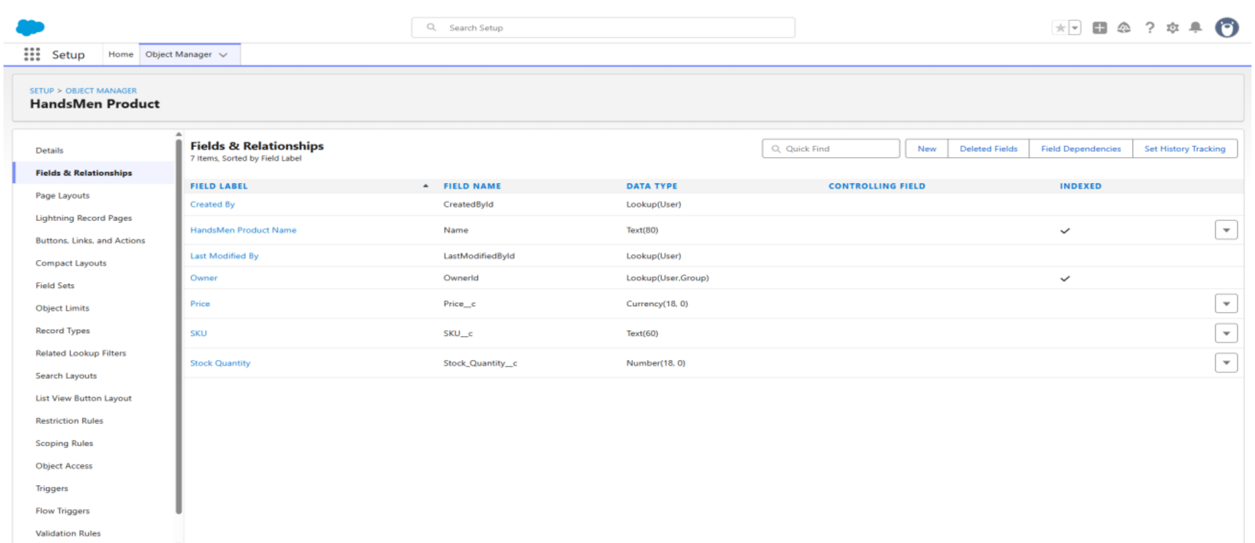
The CRM was designed with custom objects to capture and organize HandsMen Threads' core business data, enabling accurate tracking, reporting, and automation. Relationships and key fields were defined to ensure data consistency across the system.

The five custom objects include:

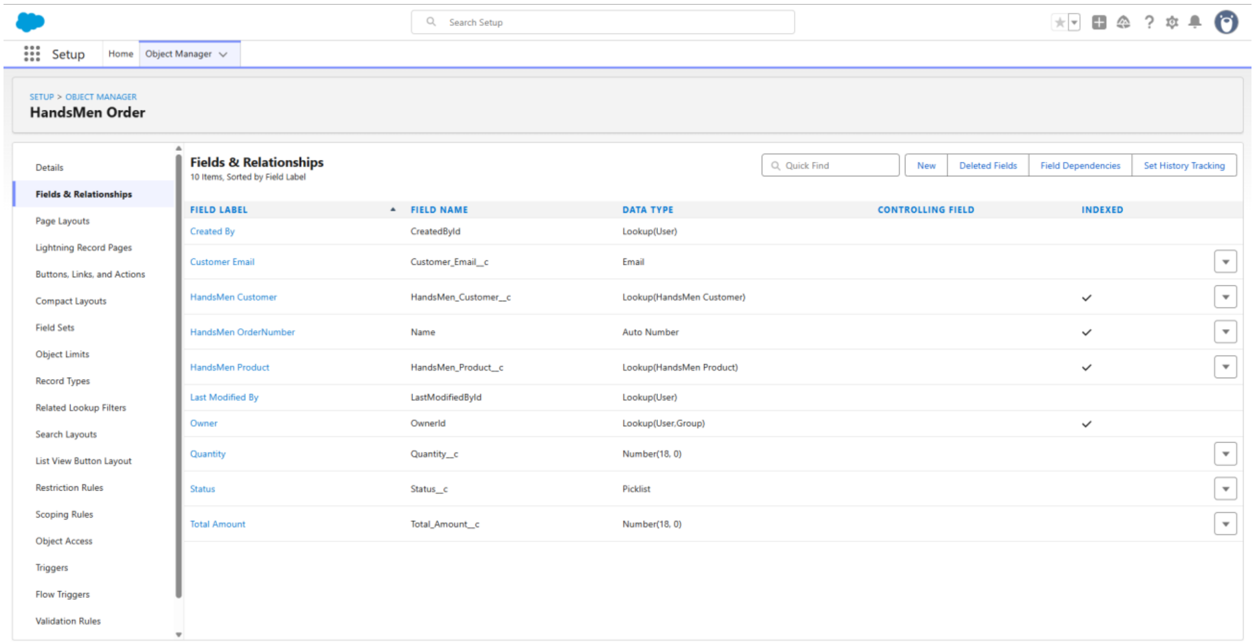
- **HandsMen Customer** – Stores customer details such as email, full name, loyalty status, and total purchases.



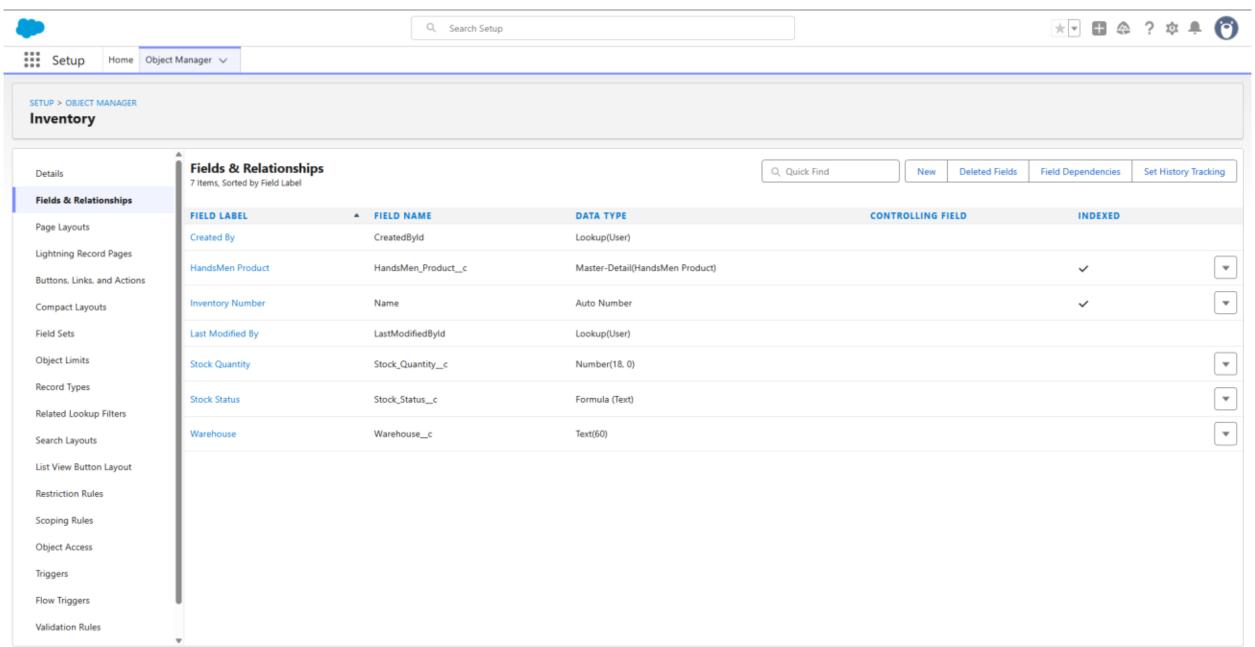
- **HandsMen Product** – Contains product information including SKU, price, and stock levels.



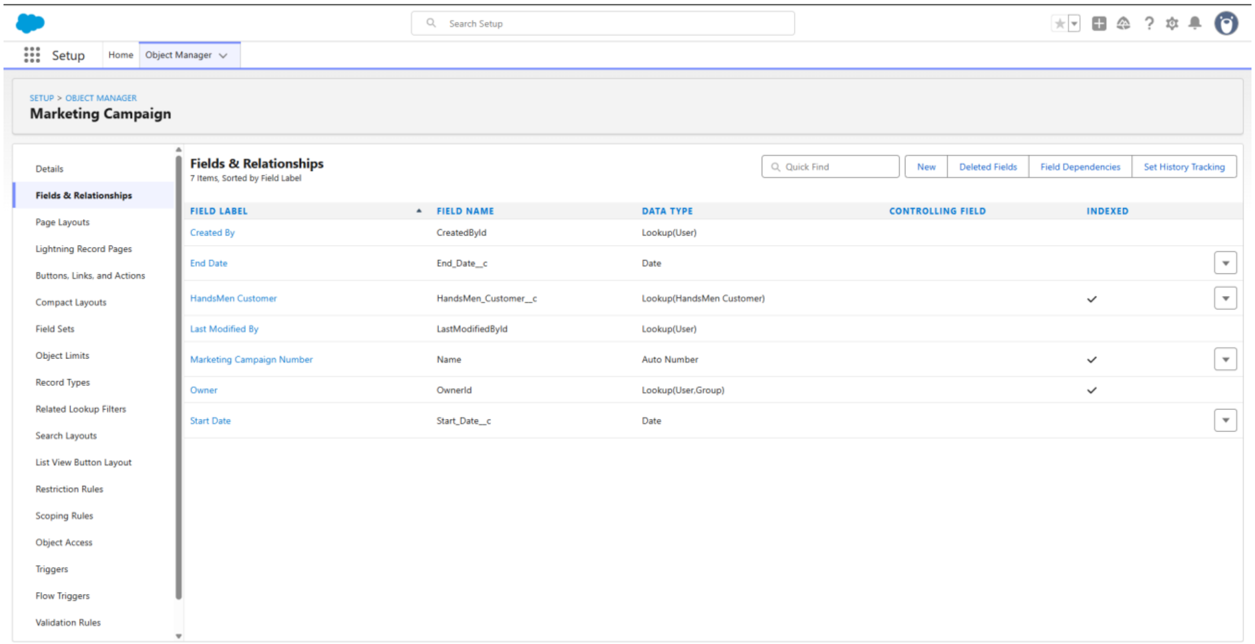
- **HandsMen Order** – Records customer orders with details on quantity, status and total amount.



- **Inventory** – Tracks stock quantities, stock status, and warehouse locations.



- **Marketing Campaign** – Manages promotion details and campaign schedules.



- **Validation Rules**

Validation rules were implemented in the CRM to ensure data accuracy and enforce business logic.

- **HandsMen Customer** – Ensures that email addresses entered are valid Gmail accounts. If an email does not contain "@gmail.com," the system displays the error: *"Please fill Correct Gmail."*

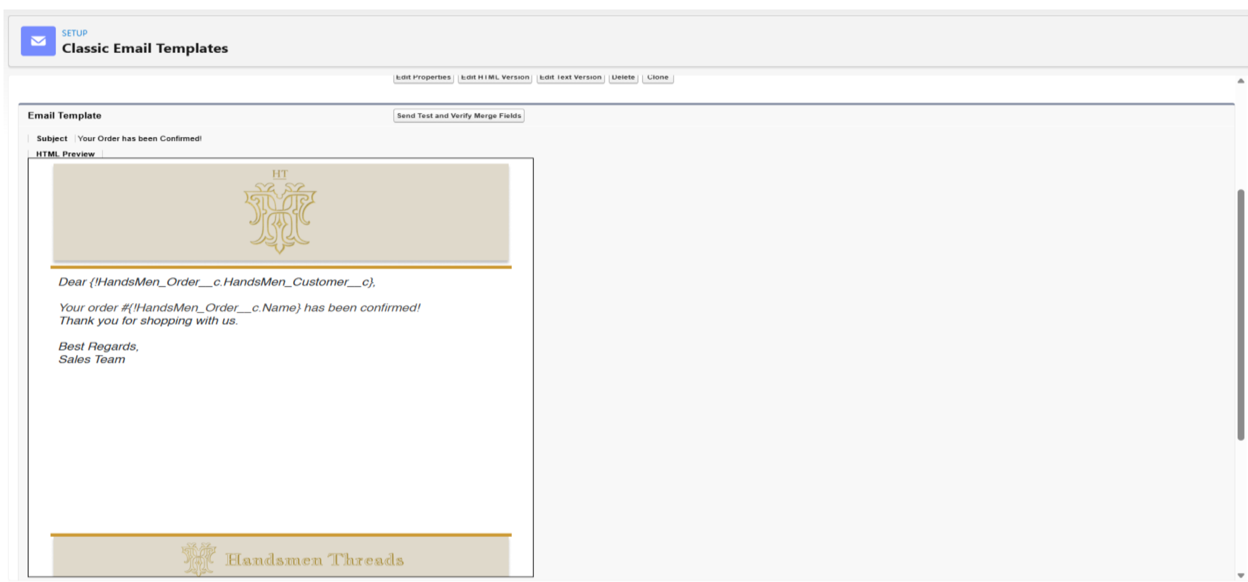
- **HandsMen Order** – Prevents orders from being saved with zero or negative amounts. If `Total_Amount__c` is less than or equal to zero, the error message *"Please Enter Correct Amount"* appears.

- **Inventory** – Ensures that stock levels cannot be zero or negative. If `Stock_Quantity__c` is less than or equal to zero, the system displays *"The inventory count is never less than zero."*

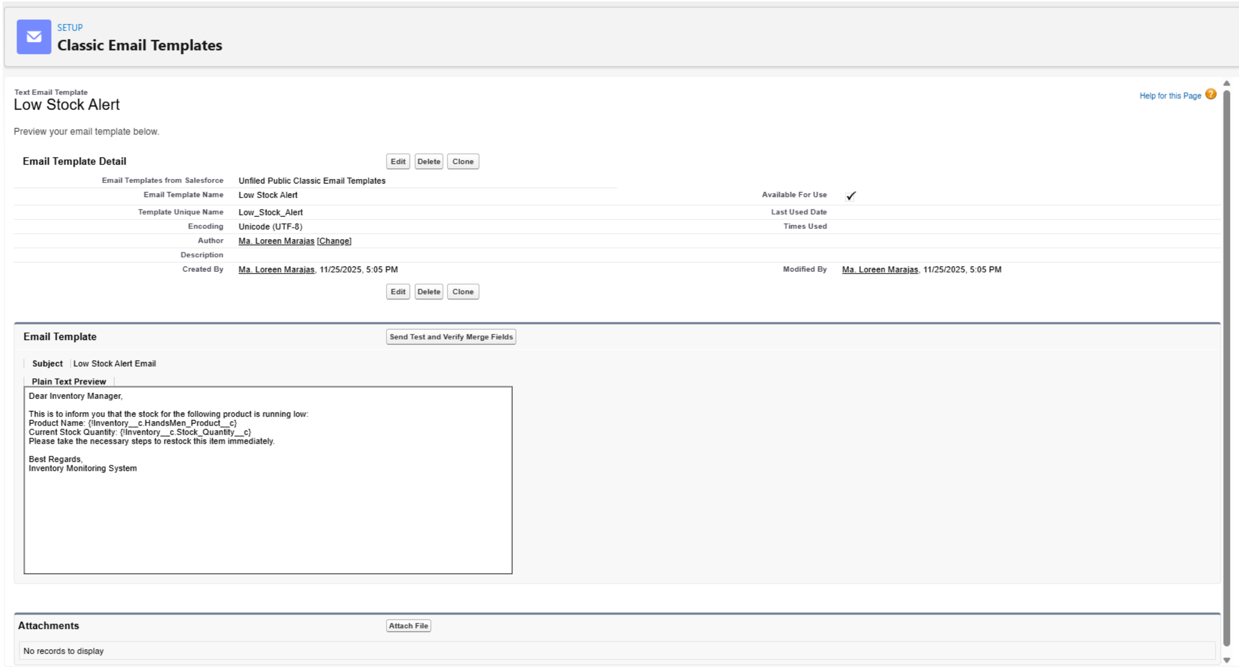
- **Email Templates**

Email templates are predefined message formats that allow automated, consistent communication by dynamically including relevant data. Three email templates were created to automate and standardize communication with customers and staff:

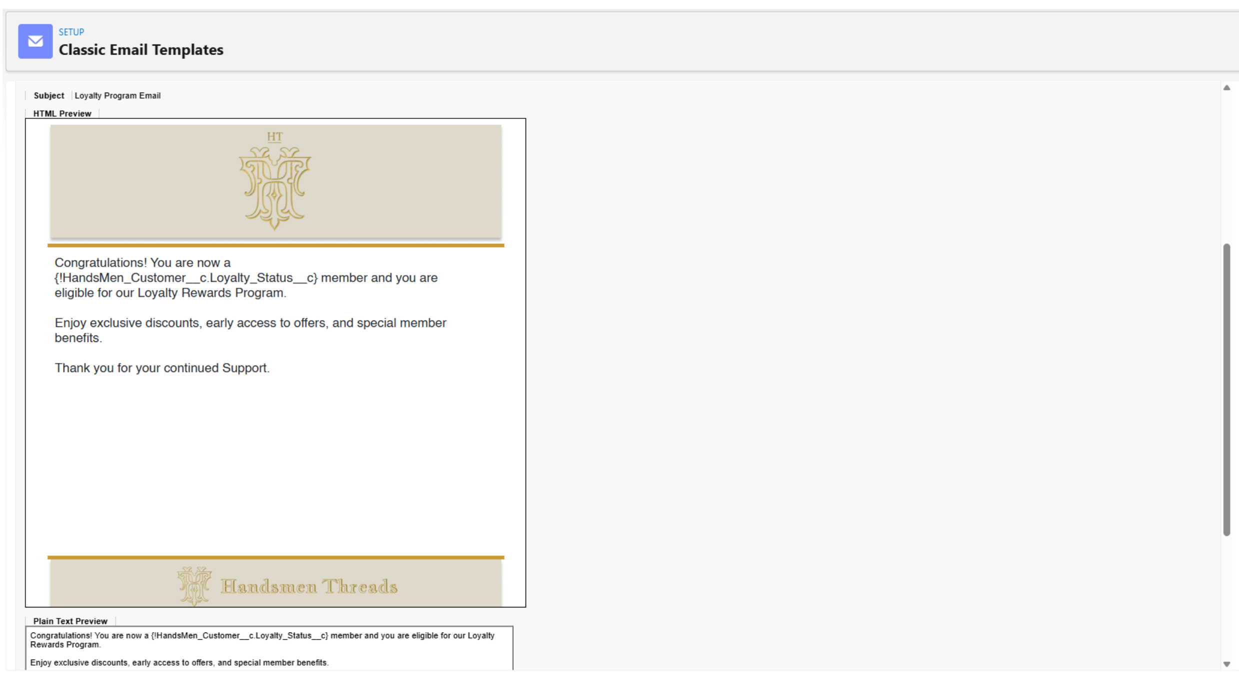
- **Order Confirmation** – Sent automatically when an order is confirmed.



- **Low Stock Alert** – Triggered when inventory drops below 5 units.



- **Loyalty Program Email** – Sent when a customer's loyalty status is updated.

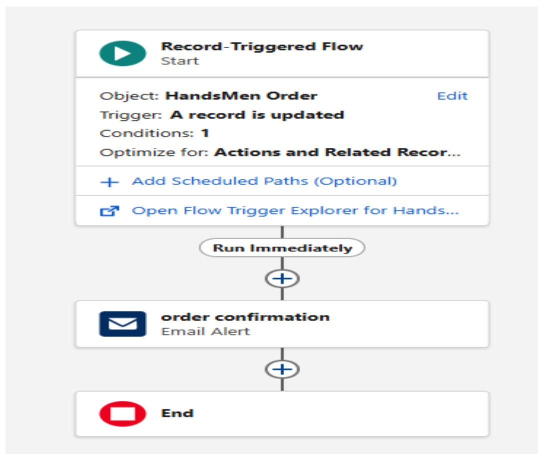


- **Automation**

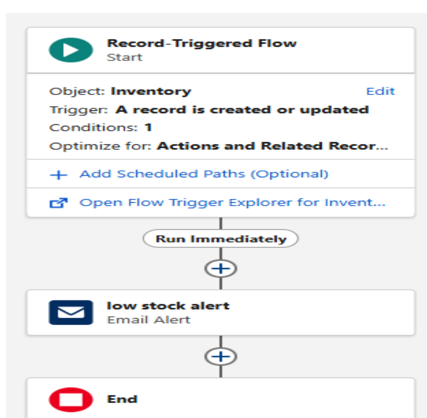
Automation was implemented to streamline routine business processes, reduce manual effort, and ensure consistent execution of critical operations. Salesforce Flows were primarily used to automate tasks such as sending emails, updating loyalty statuses, and monitoring inventory.

Key automated processes include:

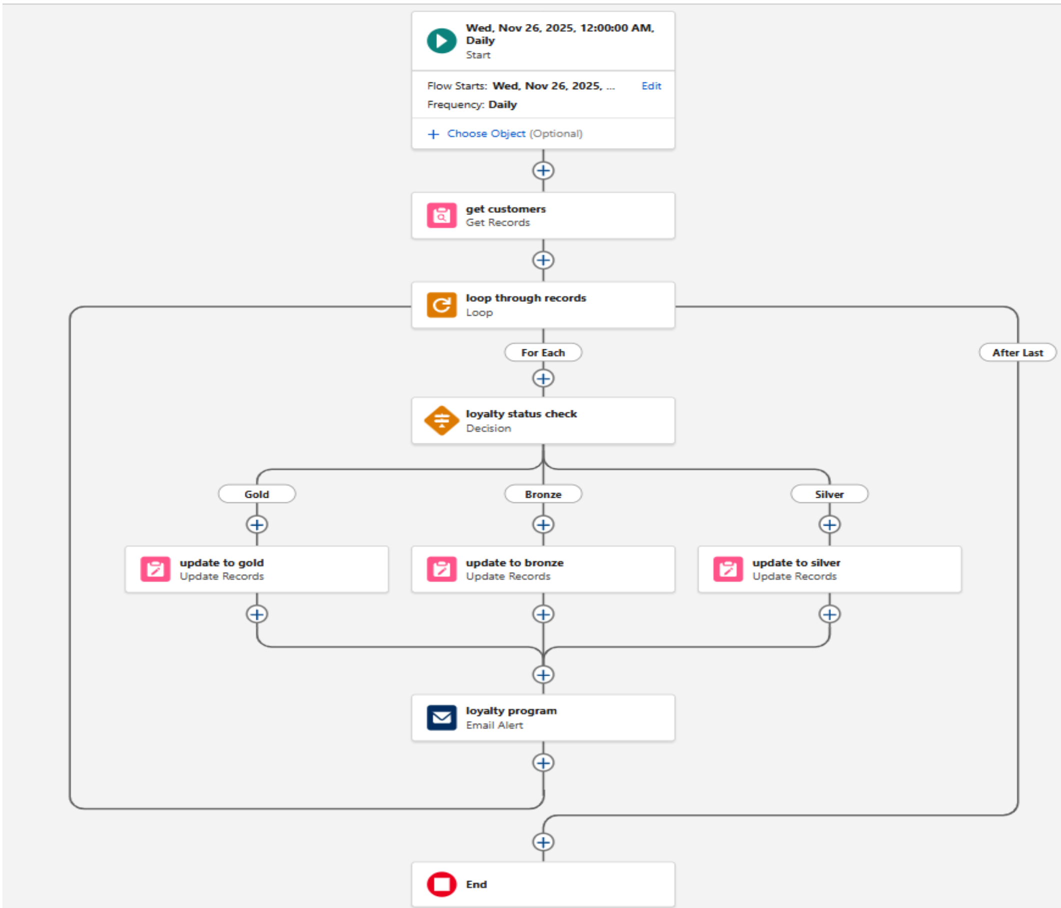
- **Order Confirmation Flow** – A record-triggered Flow on the Order object that automatically sends a personalized confirmation email to the customer when the order status is updated to *Confirmed*.



- **Low Stock Alert Flow** – A record-triggered Flow on the Product or Inventory object that sends an immediate email notification to inventory managers when stock drops to 5 units or below, preventing stockouts.



- **Loyalty Status Flow** – A record-triggered or scheduled Flow that evaluates a customer's total purchase history and updates their loyalty tier accordingly, sending an email notification to inform them of the status change.



➢ **Apex Classes, Triggers, and Asynchronous Apex**

Apex was used to implement complex business logic that could not be fully handled by declarative tools. This ensured **consistent enforcement of business rules**, **data integrity**, and automation of high-volume processes.

Key Apex Components Implemented:

- **Order Validation & Total Calculation**
  - **Trigger:** OrderTrigger – validates order quantity before insert/update
  - **Handler Class:** OrderTriggerHandler – enforces rules for status based on quantity
  - **Trigger:** OrderTotalTrigger – calculates Total_Amount__c as Quantity × Product Price

OrderTotalTrigger.apxt | StockDeductionTrigger.apxt | InventoryBatchJob.apxc

Code Coverage: None | API Version: 65

```
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;
            }
        }
    }
}
```

- **Inventory Management**
    - **Trigger:** StockDeductionTrigger – reduces inventory stock after order confirmation
    - **Handler Class:** StockDeductionHandler – queries inventory and deducts ordered quantities

OrderTotalTrigger.apxt | StockDeductionTrigger.apxt | InventoryBatchJob.apxc

Code Coverage: None | API Version: 65

```
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;
    }
}
```

- **Customer Loyalty Automation**
    - **Trigger:** LoyaltyStatusUpdateTrigger – updates loyalty status after customer purchase changes
    - **Handler Class:** LoyaltyStatusHandler – applies tier rules (Gold/Silver/Bronze) based on total purchases

- **Asynchronous Apex (Batch / Scheduler)**
  - Scheduled batch jobs were developed to handle **bulk order processing** and **automatic inventory restocking** during off-peak hours, ensuring system performance and accurate daily records.

OrderTotalTrigger.apxt | StockDeductionTrigger.apxt | InventoryBatchJob.apxc

Code Coverage: None | API Version: 65

```
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());

}

}
```

Phase 3:UI/UX Development & Customization

➢ Lightning App Setup

A custom Salesforce Lightning App named **“HandsMen Threads”** was created using the **App Manager**. The app consolidates all essential tabs which includes Customers, Orders, Products, Inventory, Loyalty Status, and Marketing Campaigns, into a single workspace. This setup allows users to access all relevant CRM functions efficiently without switching between multiple applications.

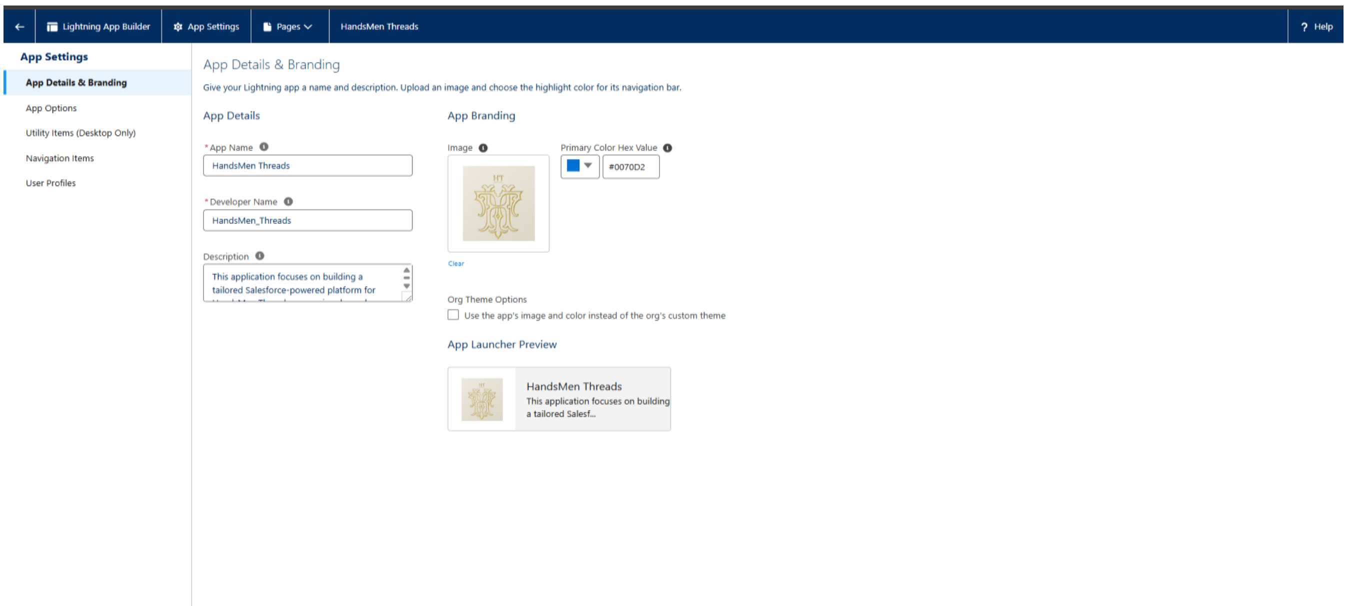


Figure 4.1. HandsMen Threads Lightning App Configuration in Salesforce App Manager

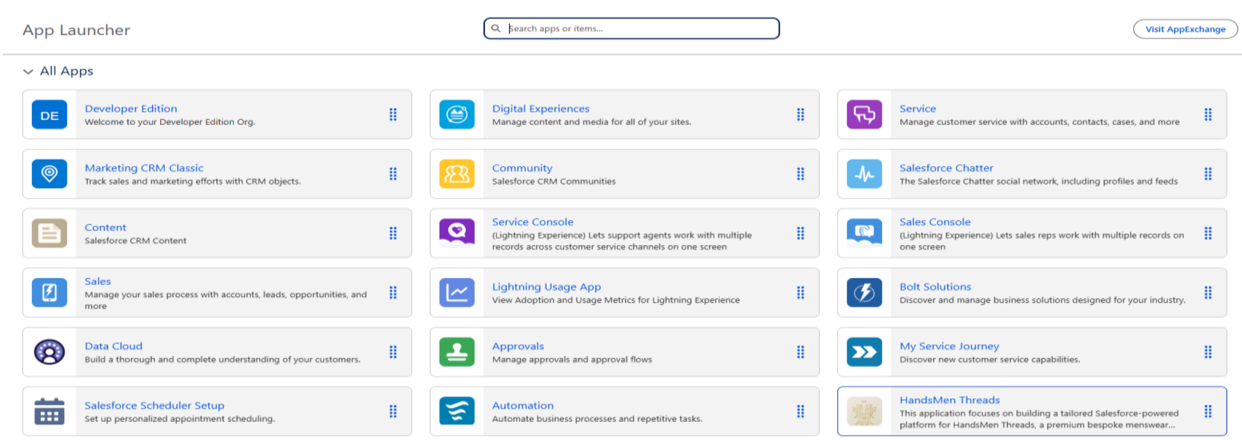


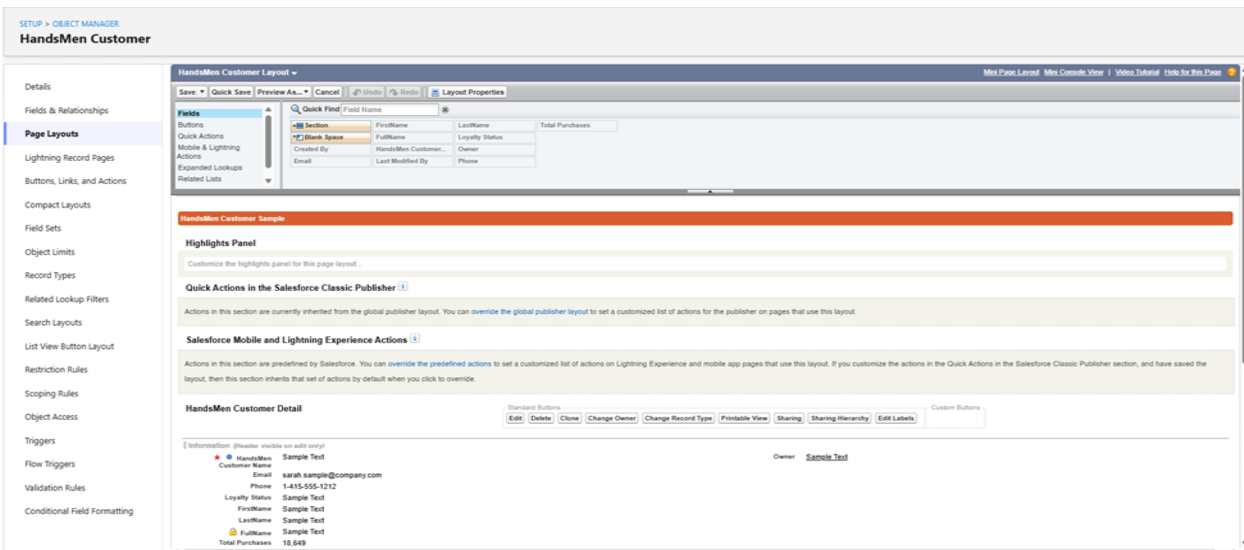
Figure 4.2. HandsMen Threads Application in Salesforce App Launcher

➢ Page Layouts and Dynamic Forms

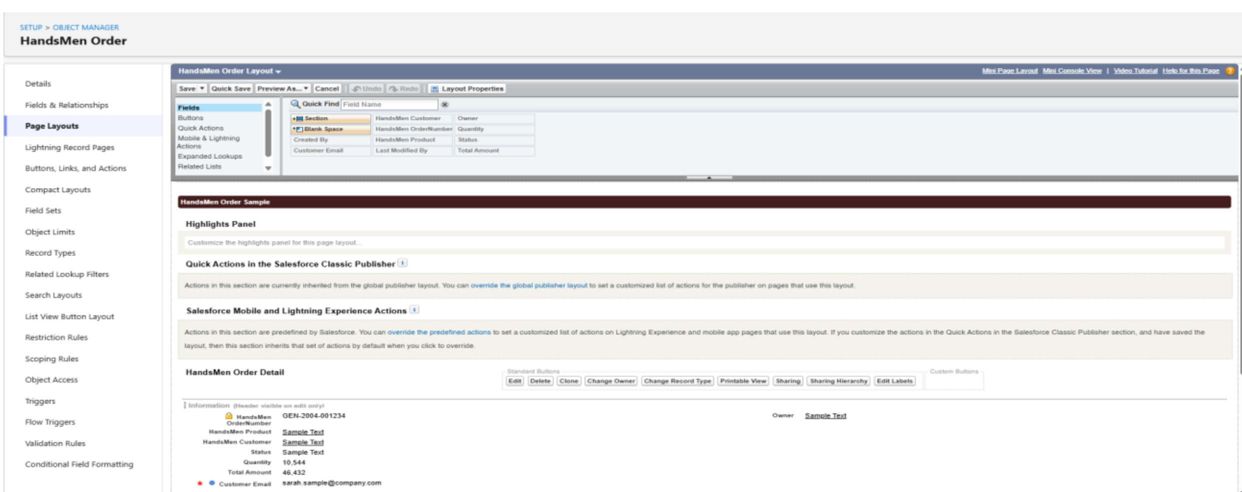
Page layouts were customized using Dynamic Forms and standard layout configurations to improve data visibility and user guidance. Fields were logically grouped and arranged in a simple two-column layout across key objects, ensuring that users can quickly locate and update important information while minimizing data entry errors.

Page layouts and Dynamic Forms were customized for the following Salesforce objects:

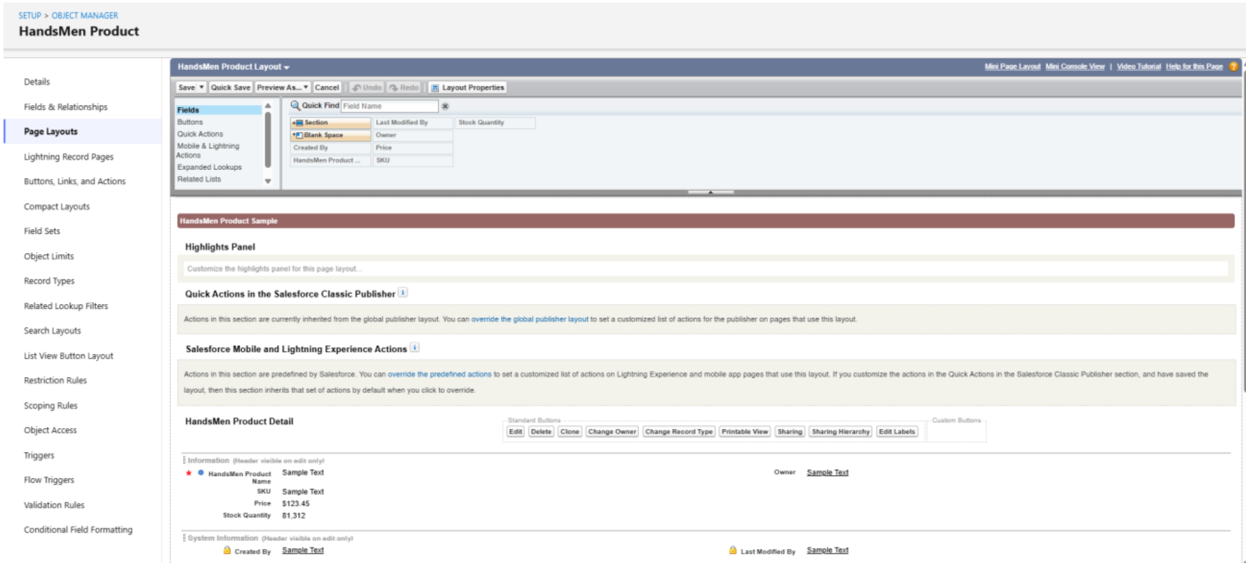
a. HandsMen Customers



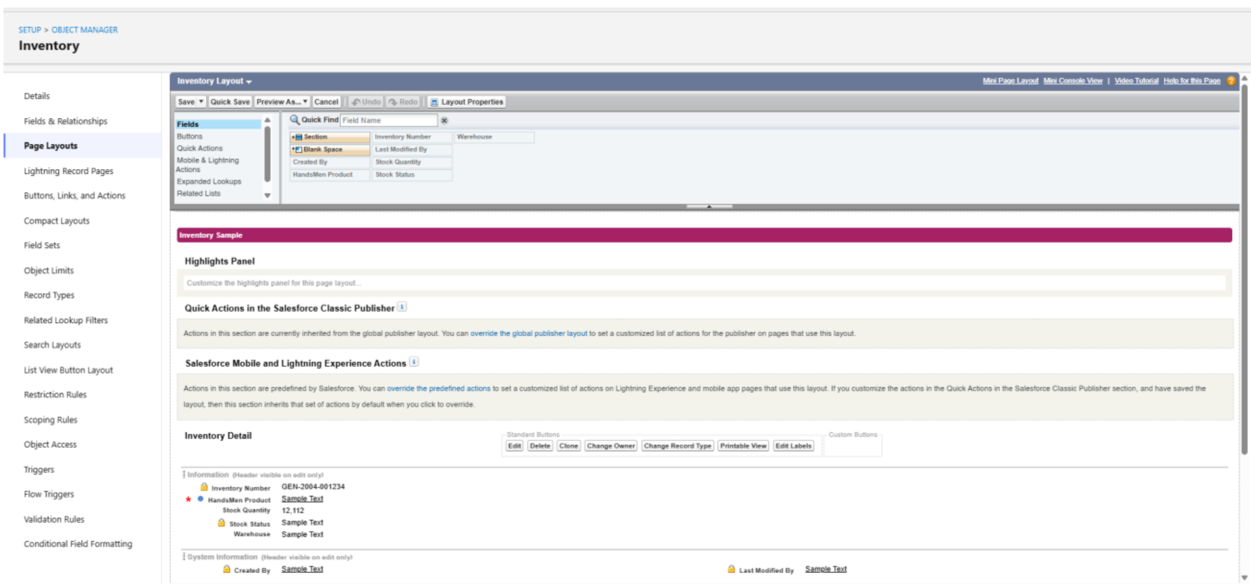
b. HandsMen Orders



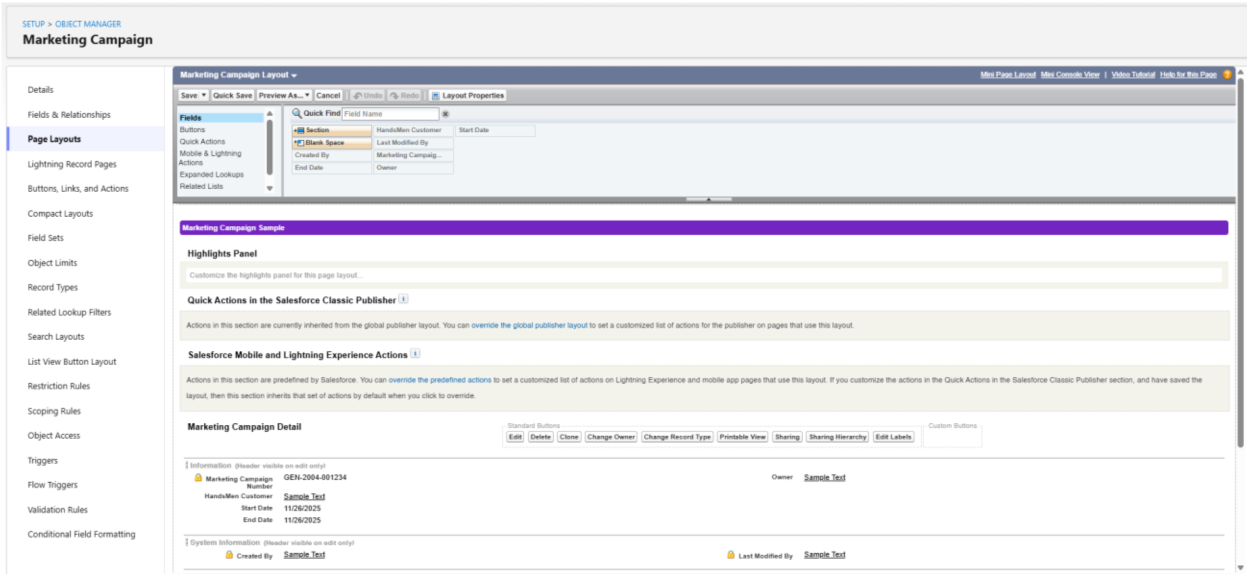
c. HandsMen Products



d. Inventory



e. Marketing Campaigns



➢ User Management

User management was configured to align system access with defined business roles and responsibilities. Roles, profiles, and permission sets were used to control access to CRM data and features, ensuring data security and appropriate user privileges.

Role	**Access Level**
Sales Manager	Full Access to Customers, Orders
Inventory Manager	Read & Edit on Inventory, Products
Marketing Team	Read on Customers, Edit on Marketing Campaigns

Figure 5. User Role and Permission Setup – HandsMen Threads CRM

➢ Reports and Dashboards

Although no custom reports or dashboards were implemented during this phase, the system is fully capable of generating Salesforce reports and dashboards. These can be used to monitor key performance indicators such as order volume, current inventory levels, and customer loyalty distribution, supporting data-driven decision-making for management.

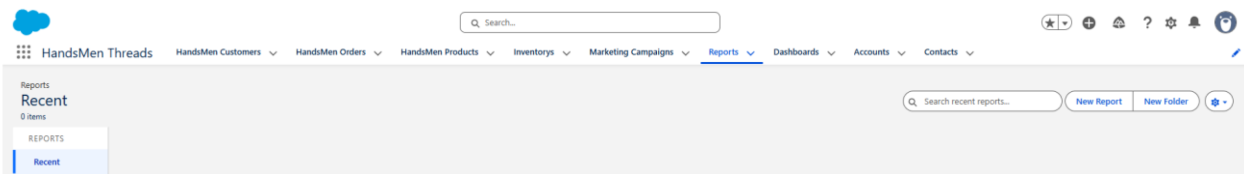


Figure 6.1. Reports Module in Salesforce Lightning App

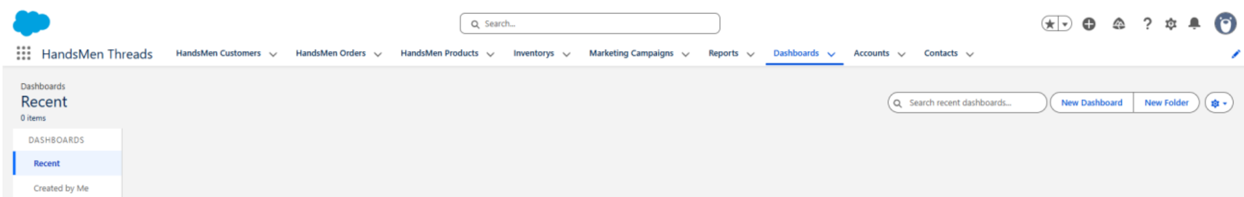


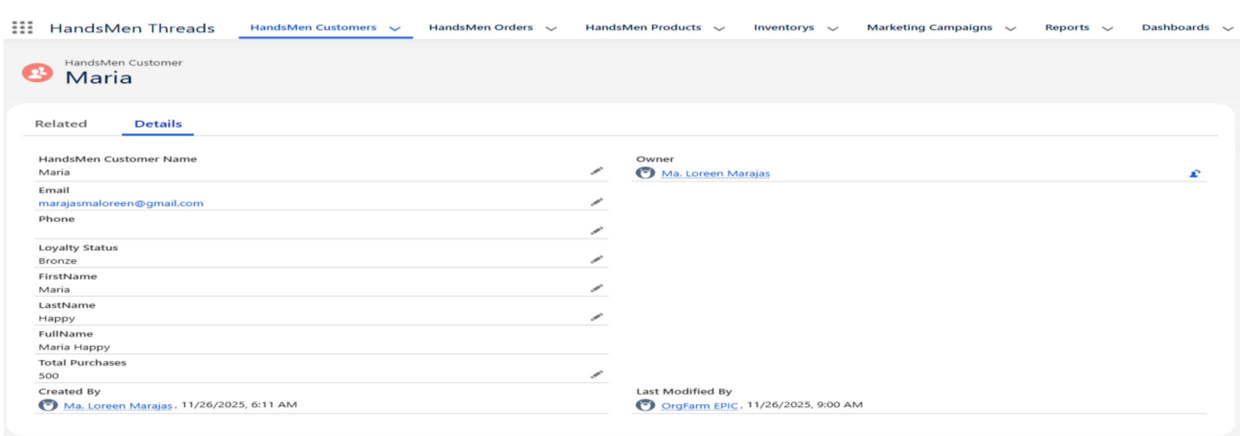
Figure 6.2. Dashboards Module in Salesforce Lightning App

➢ Lightning Pages

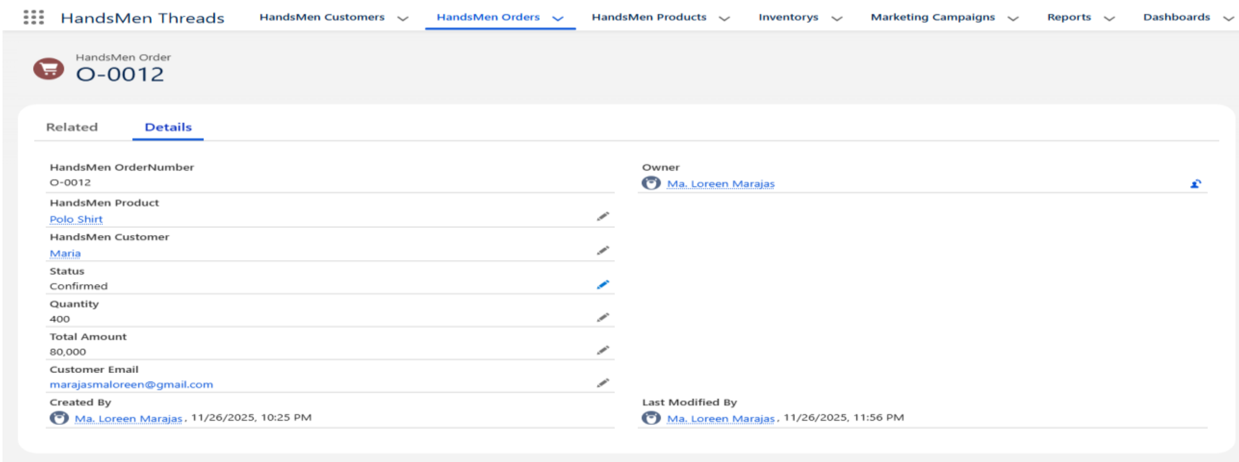
Custom **Lightning Record Pages** were developed using the Lightning App Builder for core objects, including Customers, Orders, Products, Inventory, and Marketing Campaigns. These pages were optimized for clarity and consistency, providing users with a structured view of record details and related information within the Salesforce interface.

Custom Lightning Record Pages were developed using the **Lightning App Builder** for the core Salesforce objects, including:

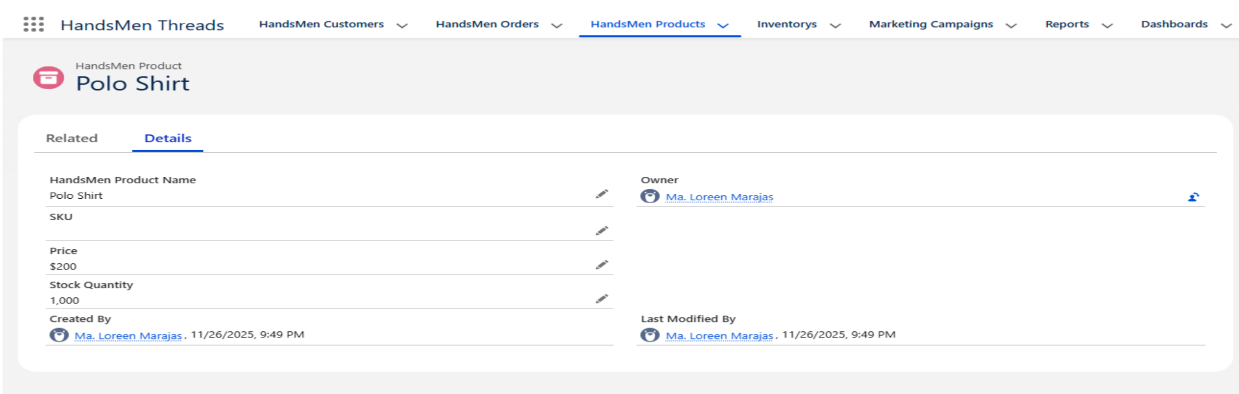
a. HandsMen Customers



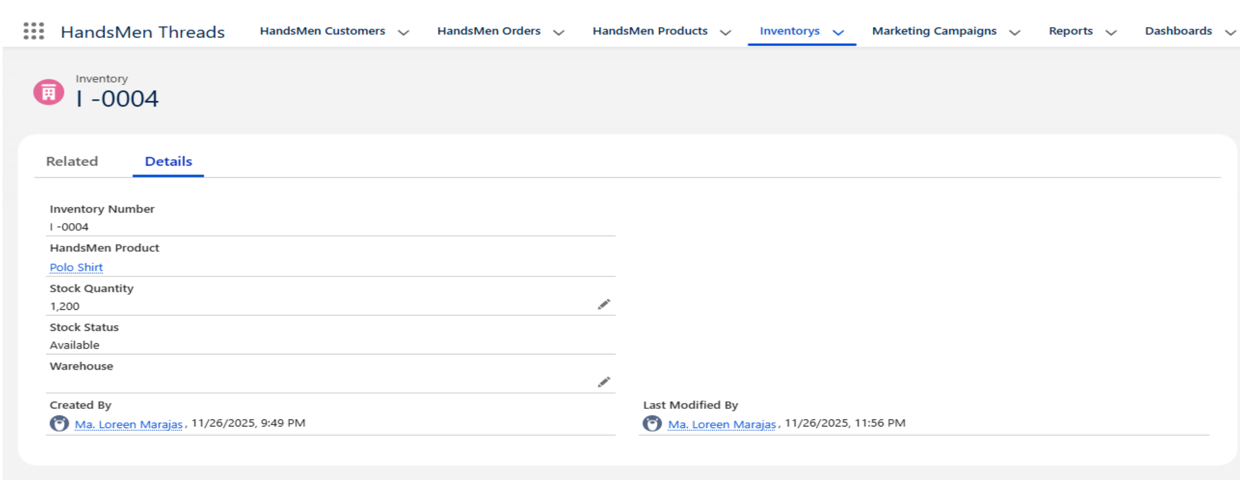
b. HandsMen Orders



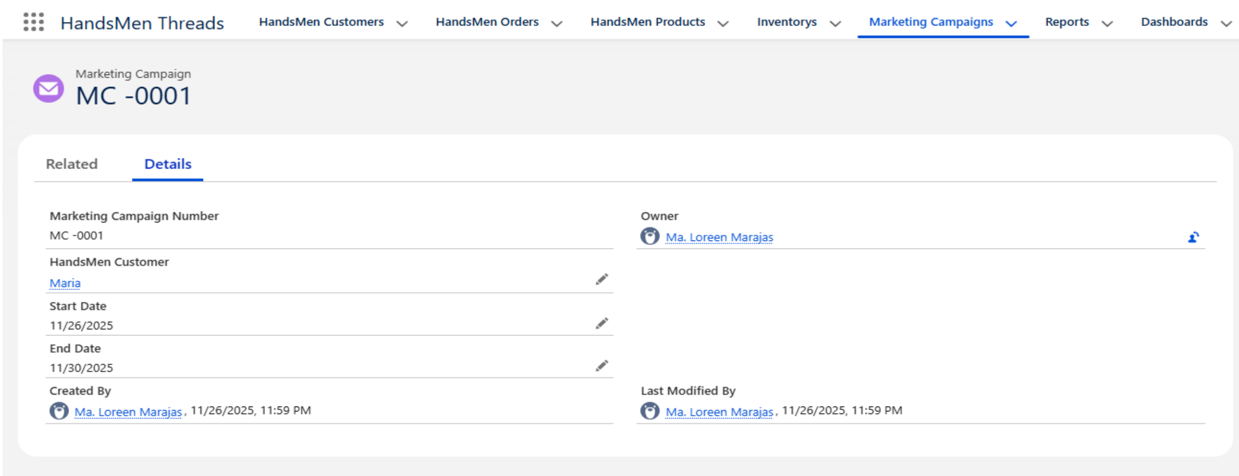
c. HandsMen Products



d. Inventory



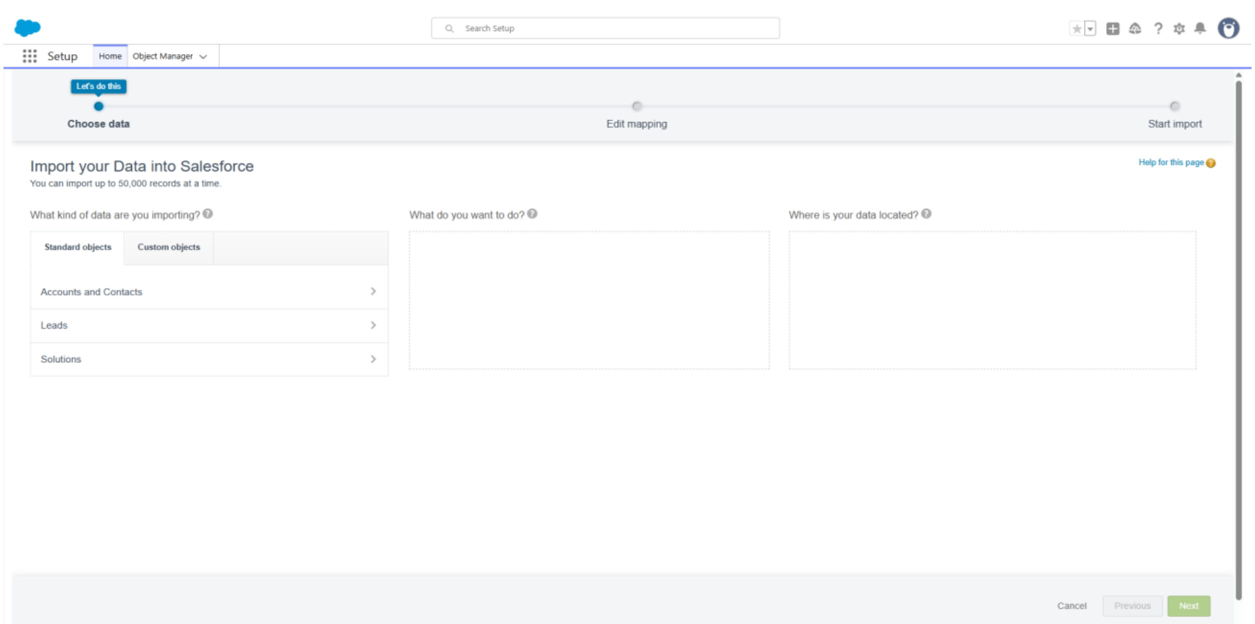
e. Marketing Campaigns



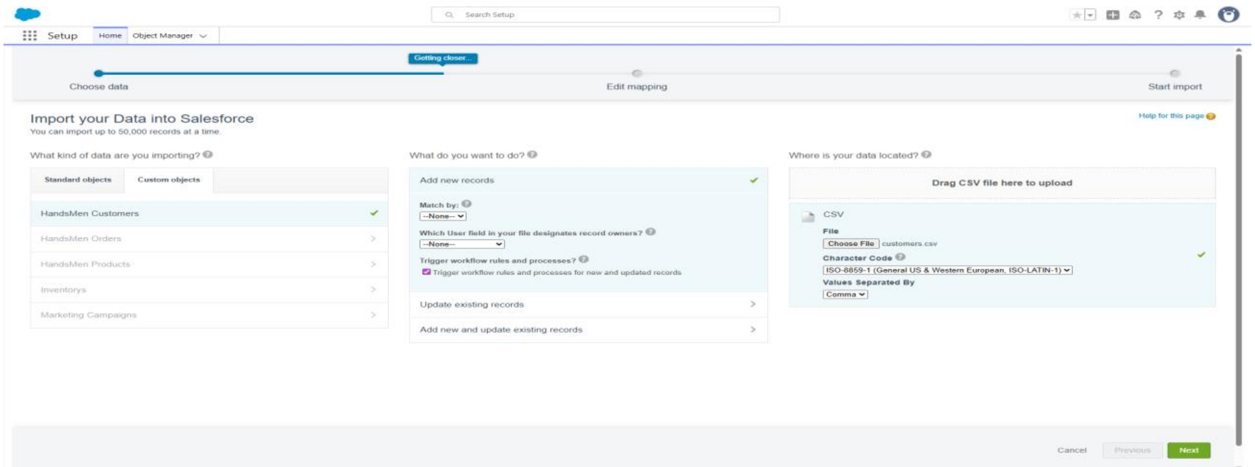
Phase 4: Data Migration, Testing & Security

➢ Data Loading Process

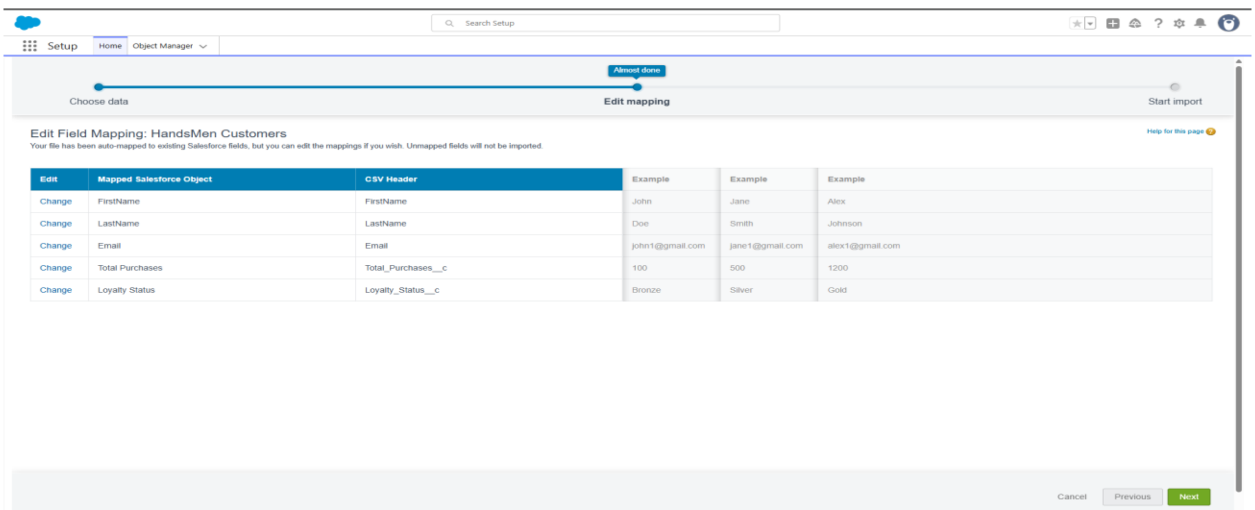
- **Initial Data Migration:** Master data for HandsMen Threads, including Customers, Products, and Inventory records, was migrated using the **Data Import Wizard** for its simplicity and suitability for small to medium datasets (<50,000 records).



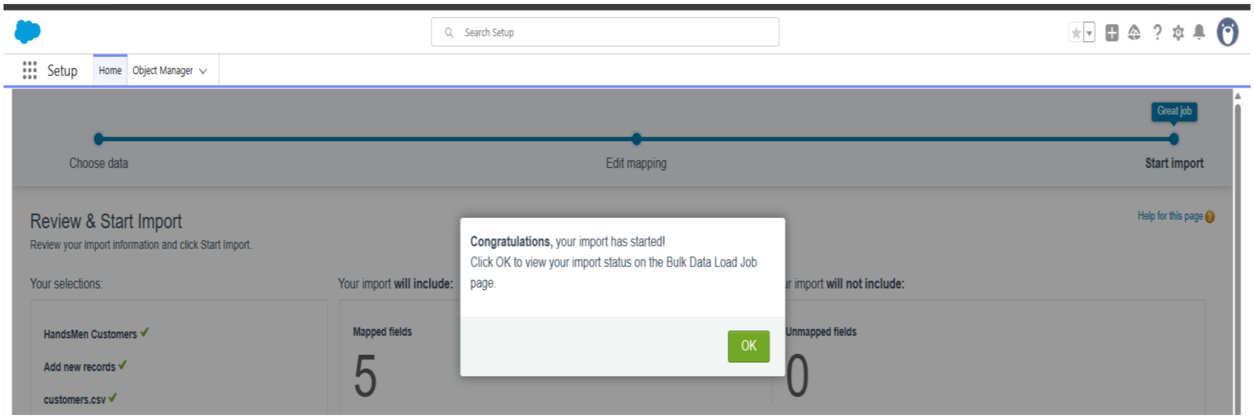
- **Bulk or Ongoing Loads:** For larger volumes or regular updates, the **Data Loader** was designated as the standard tool.

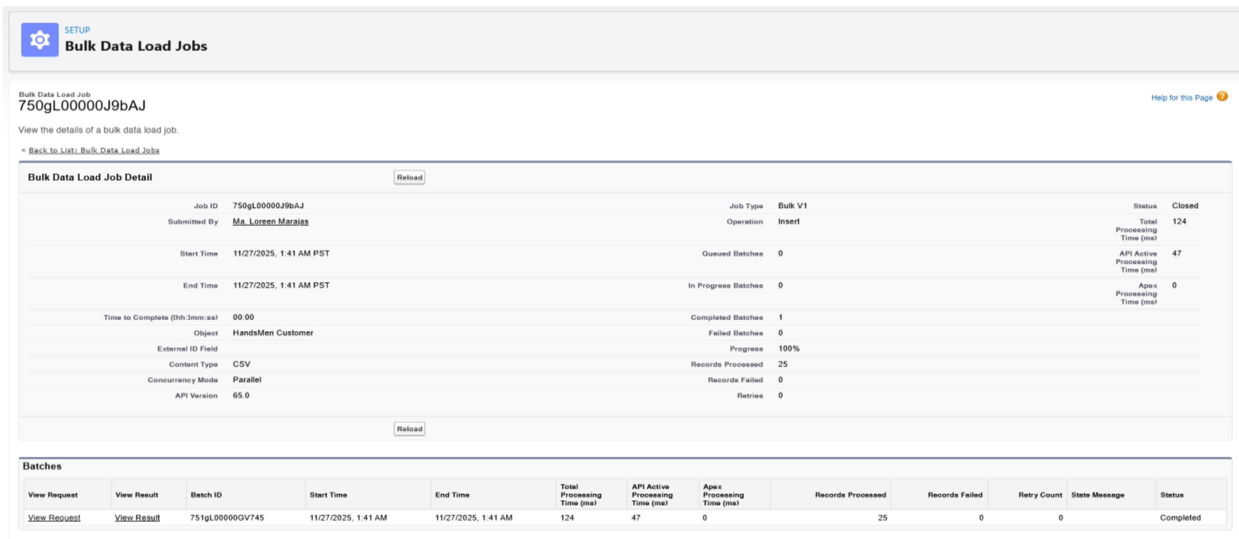


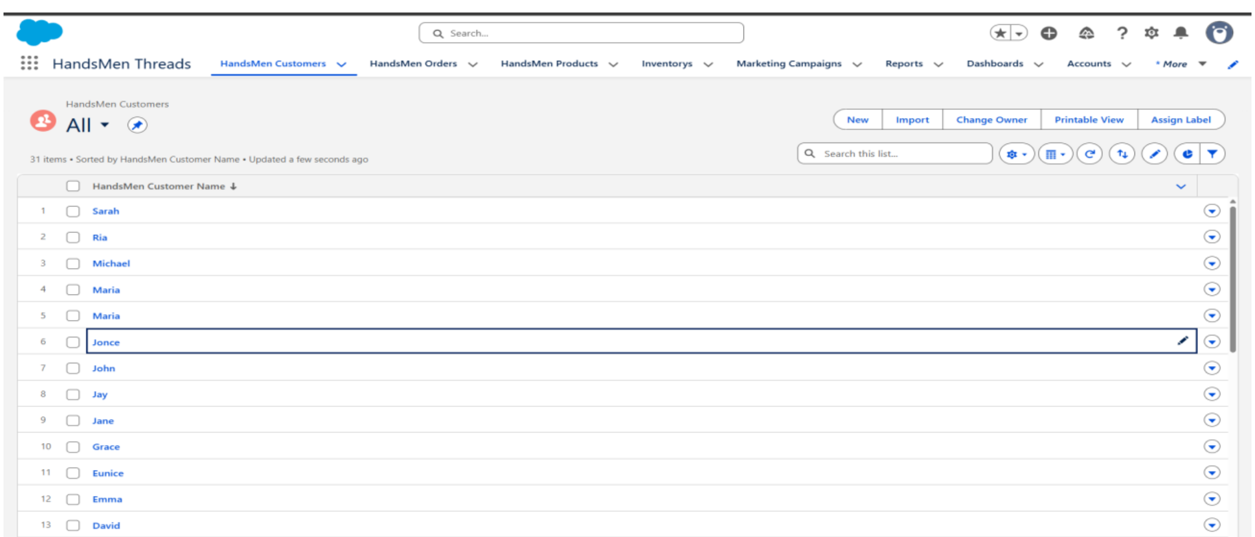
- **Field Mappings:** Carefully configured to ensure data integrity and consistency during migration.



Below are the screenshots of the successful bulk import of customer records for HandsMen Customers:







➢ Field History Tracking, Duplicate Rules, and Matching Rules

Field History Tracking, Duplicate Rules, and Matching Rules were not configured in this version of the system. While these features are useful for monitoring changes to critical fields, preventing duplicate records, and defining criteria for identifying matches, they were not implemented during this phase. These functionalities can be added in future system enhancements to further improve data integrity, auditability, and database quality.

➢ Profiles, Roles, Permission Sets, and Sharing Rules

Custom profiles and a simple role hierarchy were configured to provide appropriate access based on job functions. Sales users were able to manage Customers, Orders, and Products, while Warehouse users could manage Inventory and view Orders. Permission Sets were used to grant additional access to managers without modifying their base profiles. No sharing rules were implemented in this version of the system, meaning all record access was controlled solely through profiles, roles, and permission sets. This configuration ensured users had the necessary access for their responsibilities while maintaining data security.

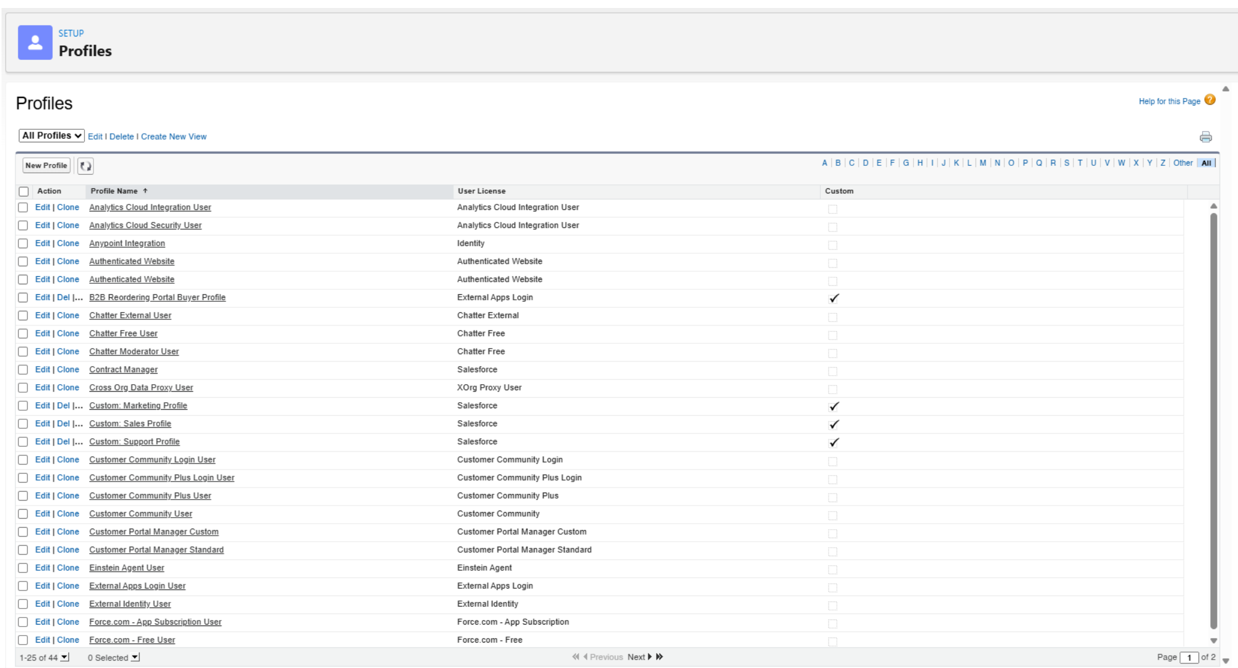


Figure 7.1. Profiles

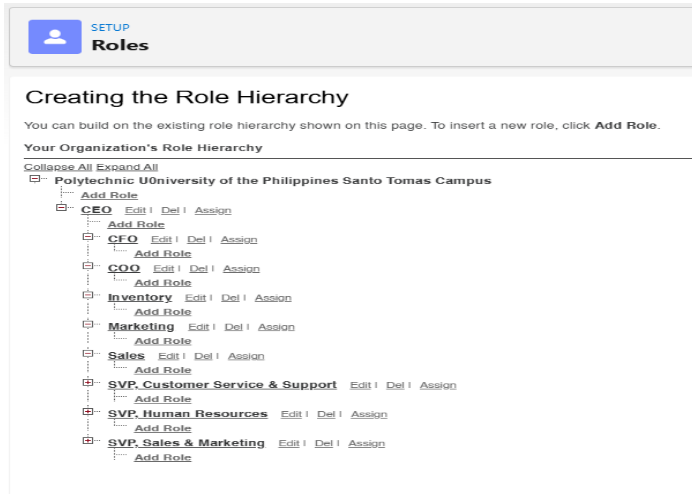


Figure 7.2. Roles and Role Hierarchy

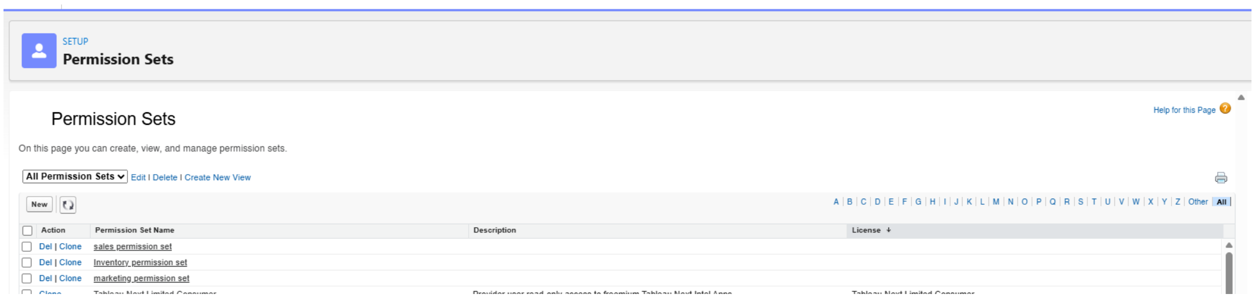


Figure 7.3. Permission Sets

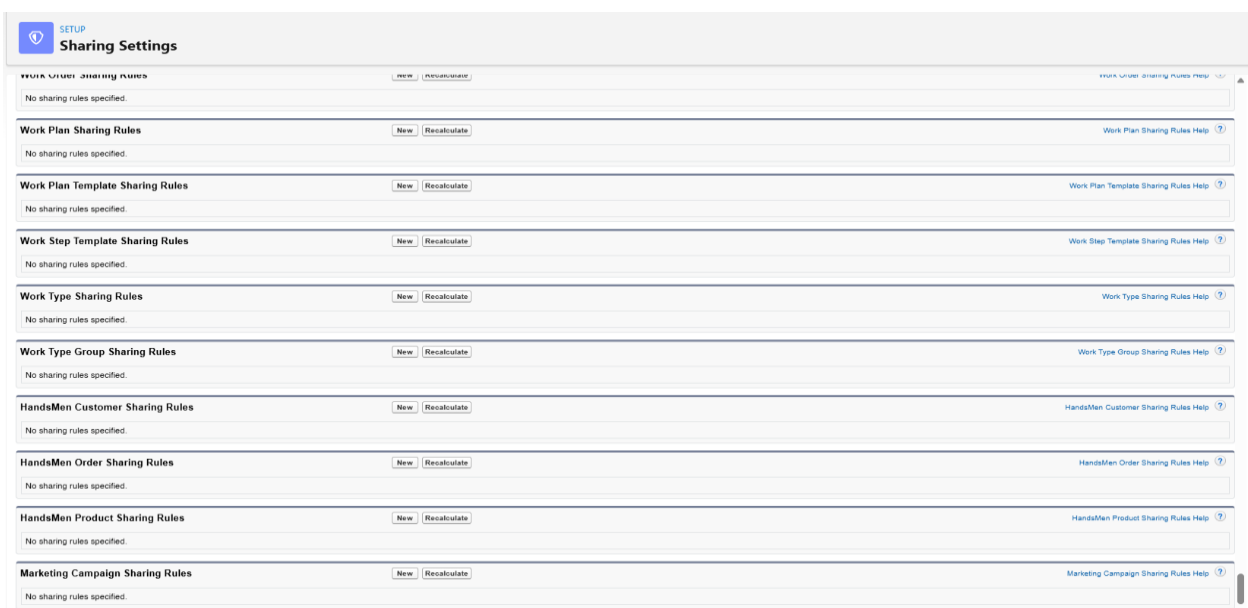
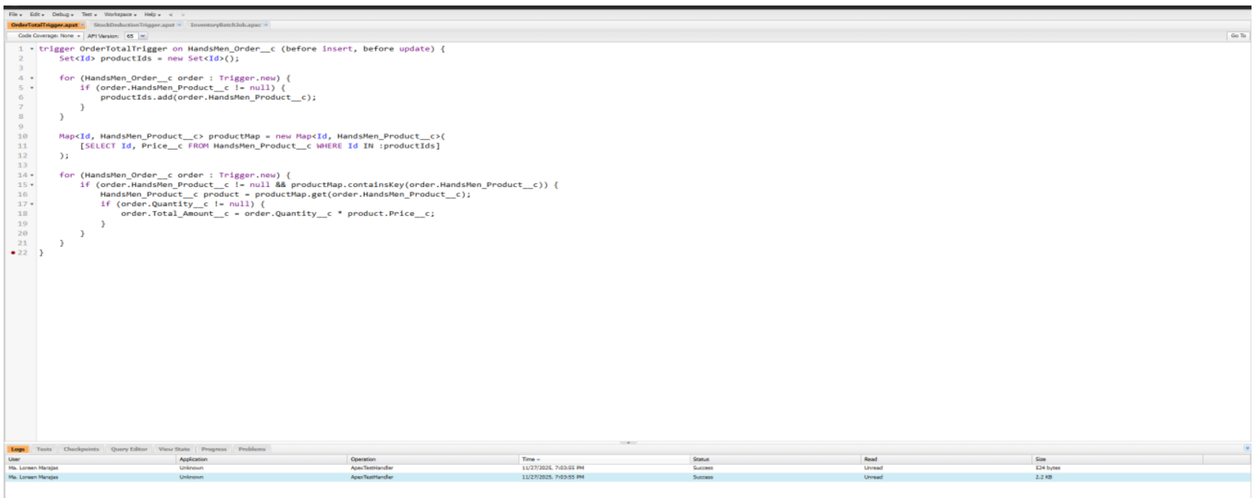


Figure 7.4. Sharing Rules

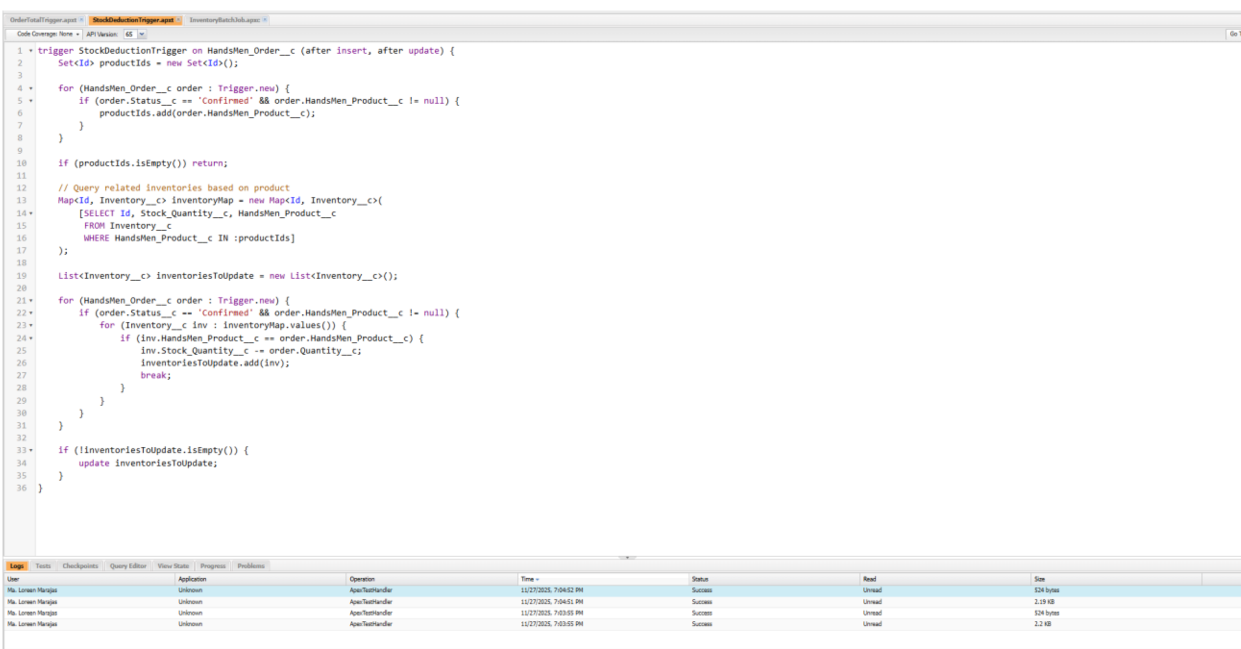
Creation of Test Classes

Apex test classes were developed to validate the custom business logic, ensuring that triggers and automations operate correctly under normal and bulk conditions. These tests also maintain the required 75% code coverage for deployment.

- **OrderTotalTriggerTest:** Inserts sample product and order records to verify that Total_Amount__c is calculated correctly as Quantity__c × Price__c. This test confirms the order total trigger works as expected across multiple records.



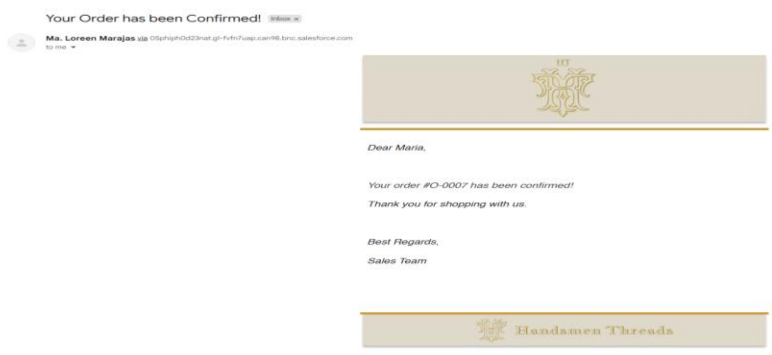
- **StockUpdateTriggerTest:** Inserts inventory and order records, then updates an order status to 'Confirmed'. The test ensures the related Stock_Quantity__c is automatically decremented, validating the stock update trigger



Functional Test Scenarios

1. Automated Order Confirmation

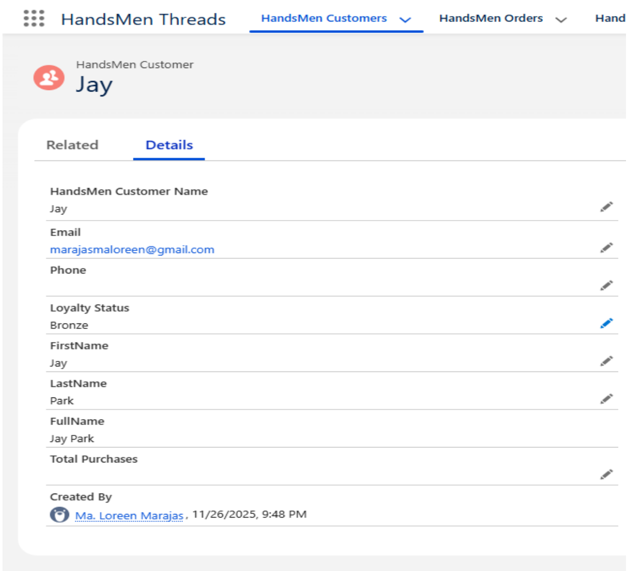
- **Scenario:** Confirm an order for a customer.
- **Steps:**
    1. Open HandsMen Orders tab.
    2. Create a new order with 'Draft' status.
    3. Save, then update status to 'Confirmed'.
- **Expected Outcome:** Confirmation email sent to customer automatically.
- **Observed Outcome:** Email sent successfully; verified in debug logs and customer inbox.

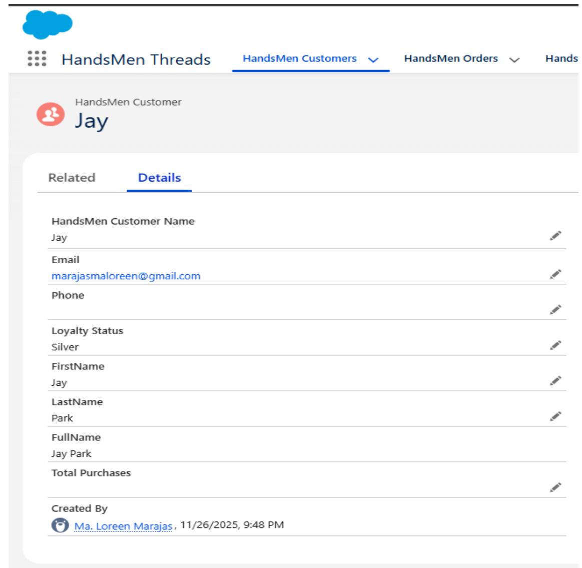


2. Loyalty Program Update

- **Scenario:** Update loyalty status based on purchase amount.
- **Steps:**
    1. Select a customer with Bronze status and $0 total purchases.
    2. Create an order with Total Amount $1,200.
- **Expected Outcome:** Customer status upgrades from Bronze to Gold.
- **Observed Outcome:** Status updated correctly; system automation verified.

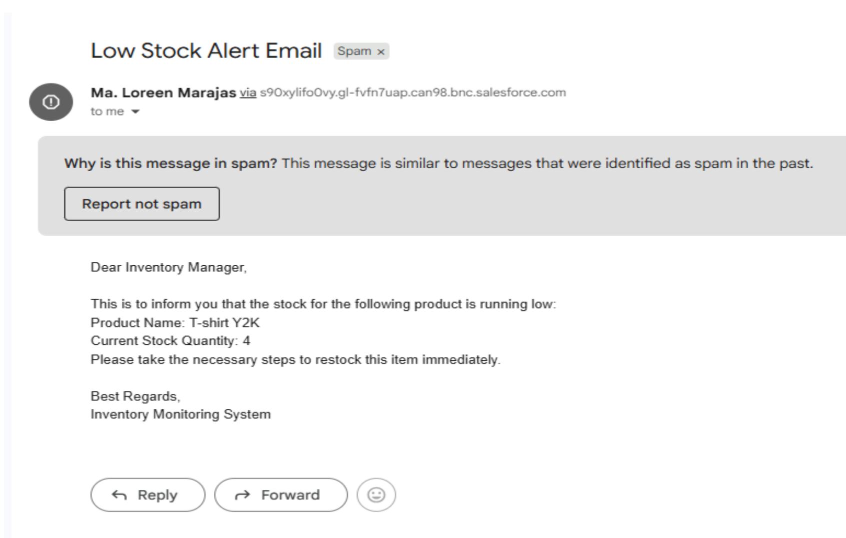
BEFORE: AFTER:





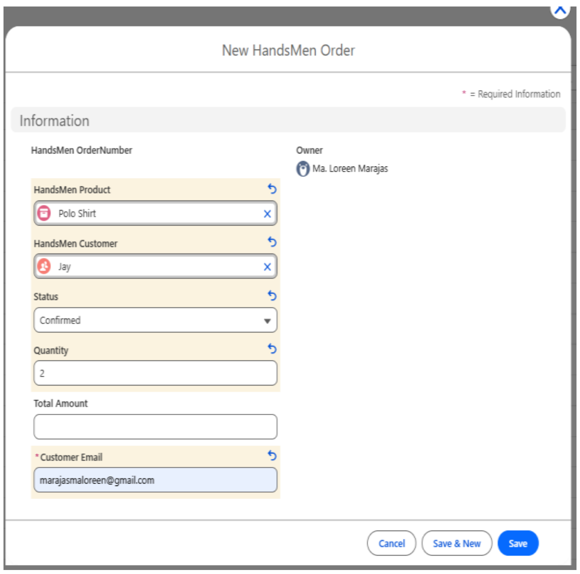
3. Low Stock Alert

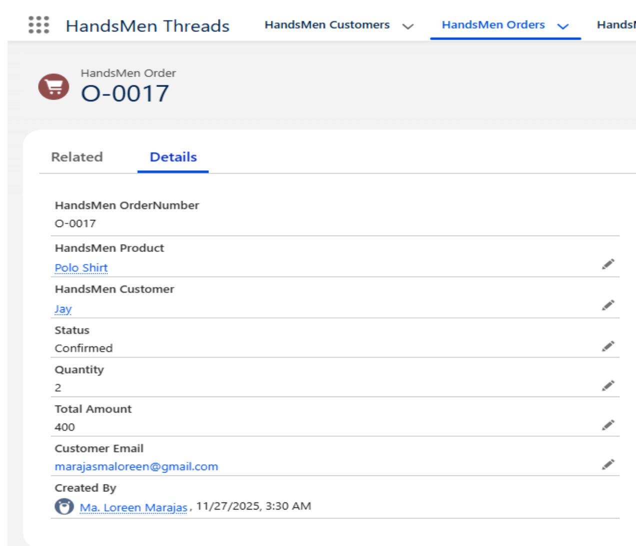
- **Scenario:** Notify inventory manager when stock drops below threshold.
- **Steps:**
  1. Open Inventory record for a product.
  2. Reduce Stock_Quantity__c from 6 to 4.
  3. Save changes.
- **Expected Outcome:** Email alert sent to Inventory Manager.
- **Observed Outcome:** Alert received; team confirmed notification.



4. Order Total Calculation

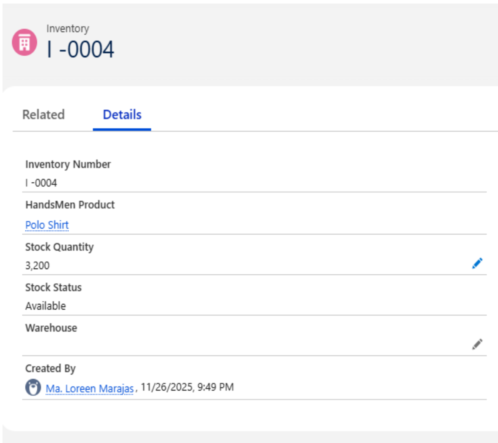
- **Scenario:** Verify automatic calculation of order total.
- **Steps:**
    1. Create a new order.
    2. Select a product (“Jacket” $10) and quantity 50.
    3. Save the record.
- **Expected Outcome:** Total_Amount__c calculates to $500.
- **Observed Outcome:** Field populated correctly upon saving.

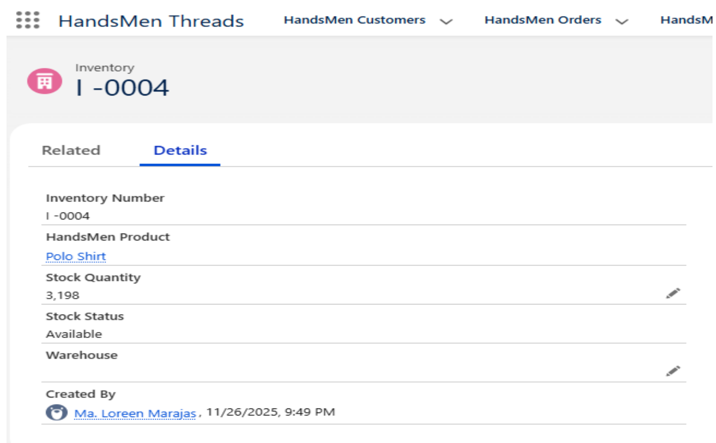




5. Stock Quantity Update

- **Scenario:** Inventory decrement after order confirmation.
- **Steps:**
    1. Check Inventory record (e.g., Stock 100).
    2. Create order with quantity 20 and status 'Confirmed'.
    3. Save record.
- **Expected Outcome:** Stock decreases to 80 automatically.
- **Observed Outcome:** Inventory updated correctly; automation verified.





Phase 5: Deployment, Documentation & Maintenance

➢ Deployment Strategy

The deployment of the HandsMen Threads CRM was planned to ensure a smooth transition from development to production. Change Sets were used to move customizations from the Developer Org to a UAT sandbox for testing. Once approved, the same Change Sets were deployed to the production environment, providing a controlled and traceable deployment process.\

➢ Maintenance and Monitoring

System maintenance focuses on ensuring continuous, reliable operation. Scheduled jobs, workflows, flows, and Apex triggers are regularly monitored using debug logs and system reports. Periodic data backups and user feedback reviews help identify and resolve potential issues proactively. Updates to business logic, objects, or automation are first tested in a sandbox before deployment to production.

➢ Troubleshooting

Troubleshooting follows a structured approach: errors are first analyzed using debug logs and Salesforce error messages, then validation rules, flows, and Apex triggers are reviewed for conflicts or logic issues. Any fixes are tested in a sandbox prior to production deployment, and each step is documented to create a knowledge base for future reference. This approach ensures that the CRM remains reliable, maintainable, and aligned with business needs.

Conclusion

The Salesforce CRM developed for HandsMen Threads successfully integrates and automates critical business processes, including managing customers, products, orders, inventory, and loyalty programs. By leveraging tools such as Custom Objects, Flows, Validation Rules, Email Alerts, and Apex triggers, the system ensures data accuracy, timely updates, and a seamless customer experience. Automation and role-based access controls reduce manual errors, accelerate operations, and provide management with actionable insights for decision-making.

Future Enhancements:

1. **Customer Self-Service Portal:** Allow customers to track orders, update information, and access support resources.
2. **Mobile Inventory & Sales Management:** Enable staff to manage inventory, process orders, and monitor loyalty on mobile devices.
3. **Advanced Analytics & Reporting:** Implement dashboards to visualize sales trends, inventory levels, and customer loyalty metrics.
4. **AI-Driven Insights:** Use predictive analytics to suggest cross-selling opportunities, forecast demand, and analyze customer behavior.
5. **Field History Tracking & Duplicate Rules:** Improve data accuracy and security by auditing changes and preventing duplicates.