

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

Prepared by: Minette Joy Lorzano

School: Polytechnic University of the Philippines

Project Overview

This project focuses on building a tailored Salesforce CRM for HandsMen Threads, a forward-thinking men's fashion brand. Its main goal is to streamline data management, strengthen customer relationships, and unify operations on a reliable platform. A robust data model and strict UI validations ensure accuracy, while custom objects for Customers, Orders, Products, Inventory, and Campaigns integrate seamlessly. Business processes are automated through flows, email alerts, and Apex code to handle confirmations, loyalty updates, and stock notifications. Data integrity is reinforced with validation rules and role-based access for Sales, Inventory, and Marketing teams. A scheduled Apex batch also monitors low stock for proactive replenishment. Overall, this CRM enhances customer engagement, optimizes workflows, and equips HandsMen Threads with a scalable, future-ready system within Salesforce.

Objectives

The Salesforce CRM for HandsMen Threads aims to create a unified platform that manages sales, inventory, marketing, and customer service data. Centralized validation and controlled access ensure accuracy and reliability, supporting better decisions and smoother operations. Automated processes—such as order confirmations, loyalty updates, and stock monitoring—boost efficiency and agility, while personalized customer communications drive engagement and long-term loyalty. This scalable system not only meets current operational needs but also positions HandsMen Threads for sustained growth, innovation, and measurable value through stronger customer management, streamlined workflows, and optimized inventory control.

Phase 1: Requirement Analysis & Planning

1.1 Understanding Business Requirements

HandsMen Threads needs a centralized CRM to manage sales, inventory, marketing, and customer service. Current challenges include fragmented data, manual processes, and limited visibility across teams. The solution must improve customer engagement, streamline workflows, and support scalable growth in the competitive men's fashion market.

1.2 Defining Project Scope and Objectives

Scope

- Implement Salesforce CRM customized for HandsMen Threads.
- Integrate core business objects: Customer, Order, Product, Inventory, and Marketing Campaigns.
- Automate key processes (order confirmations, loyalty updates, stock notifications).

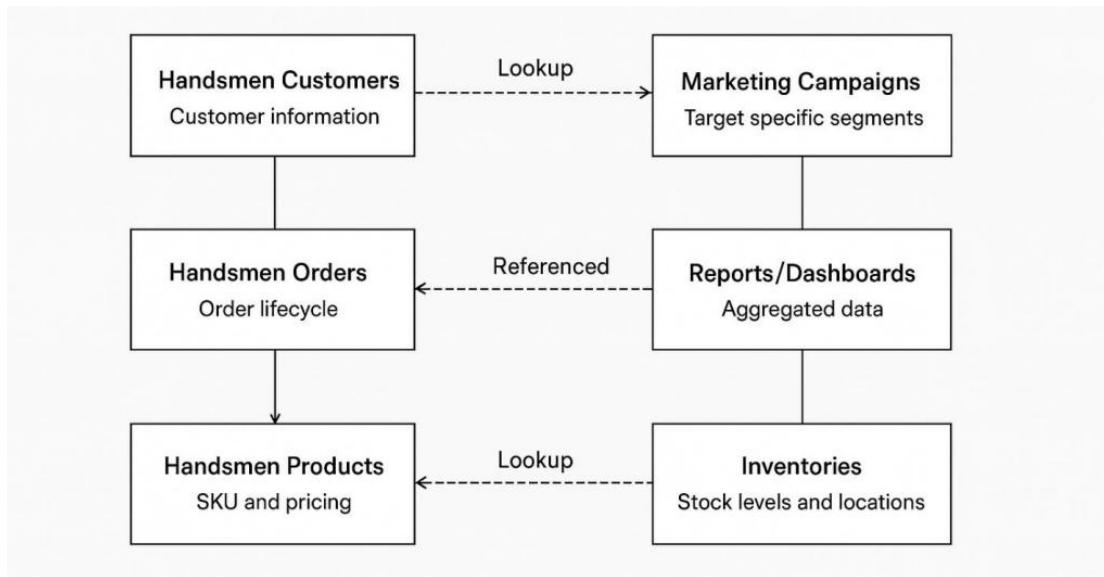
Objectives

- Ensure data accuracy and consistency through validation rules.
- Establish role-based access control for Sales, Inventory, and Marketing teams.
- Enhance customer experience with personalized communications.
- Provide a future-ready, scalable platform for continuous innovation.

1.3 Design Data Model and Security Model

Data Model

Entities: HandsMen Customer, HandsMen Order, HandsMen Product, Inventory, Marketing Campaign. Relationships defined to support reporting and analytics.



Security Model:

- Role-based access (Sales, Inventory, Marketing).
- Validation rules to enforce data integrity.
- Scheduled Apex batch for proactive inventory monitoring.

1.4 Stakeholders Mapping

Internal Stakeholders:

- Sales Team - customer acquisition & relationship management.
- Inventory Team - stock monitoring & replenishment.
- Marketing Team - campaign execution & loyalty programs.

External Stakeholders:

- Customers - improved engagement and service.
- Suppliers - streamlined inventory coordination.
- Management - strategic decision-making with reliable data.

1.5 Execution Roadmap

Week 1-2: Requirements gathering and data model design.

Week 3-4: Back-end development and configurations.

Week 5-6: UI/UX customization and testing.

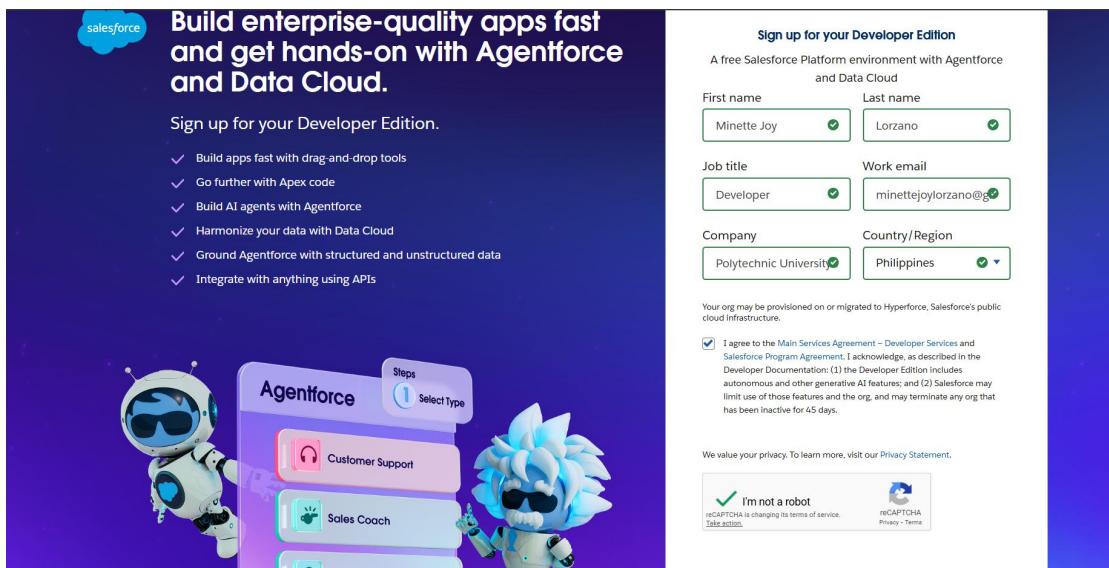
Week 7-8: Data migration, security setup, and deployment.

Ongoing: Maintenance and enhancements.

Phase 2: Salesforce Development - Backend & Configurations

2.1 Setup environment & DevOps workflow

A Salesforce Developer Org was established by registering through developer.salesforce.com/signup. The account was confirmed via email, a password was created, and access to the Salesforce Setup page was successfully enabled.



2.2 Customization of Objects, Fields, Validation Rules, Automation

Custom Objects

Five custom objects were developed to manage essential business data within the Salesforce CRM:

- HandsMen Customer – Stores customer information including contact details and loyalty status.
- HandsMen Product – Tracks product attributes such as SKU, pricing, and available stock.
- HandsMen Order – Records customer orders with details like quantity and order status.
- Inventory – Monitors stock levels and warehouse locations for efficient supply management.
- Marketing Campaign – Captures promotional campaign details, schedules, and targeted activities.

Next Steps:

- Navigated to Setup, clicked Object Manager, and selected Create → Custom Object.
- Entered the object label and name, and enabled options for reporting and search functionality.

The screenshot shows the Salesforce Object Manager interface. At the top, there's a navigation bar with icons for Home, Object Manager, and other setup options. Below the navigation is a breadcrumb trail: SETUP > OBJECT MANAGER. The main title is 'HandsMen Customer'. On the left, a sidebar lists various object configuration options like Fields & Relationships, Page Layouts, and Buttons, Links, and Actions. The central panel is titled 'Details' and contains the following information:

- Description: Stores customer details
- API Name: HandsMen_Customer__c
- Singular Label: HandsMen Customer
- Plural Label: HandsMen Customers
- Enable Reports (checkbox checked)
- Track Activities (checkbox checked)
- Track Field History
- Deployment Status: Deployed
- Help Settings: Standard salesforce.com Help Window

At the bottom right of the central panel are 'Edit' and 'Delete' buttons.

- Saved the custom object

Validation Rules

Order Object: Prevents saving the record if Total_Amount__c <= 0.

Salesforce Order Record Edit Screen:

- Fields:** HandsMen OrderNumber (O-0007), HandsMen Product (Baggy Jeans), Customer (Marinelle), Status (Pending), Quantity (0), Total Amount (0).
- Errors:** A validation error message "ø We hit a snag." is displayed over the Total Amount field, stating "Please Enter Correct Amount".
- Details:** Created By (Minette Joy Lorzano, 11/26/2022) and Last Modified By (Minette Joy Lorzano, 11/26/2025, 5:51 PM).

Customer Object: Validates that the email address contains “@gmail.com”

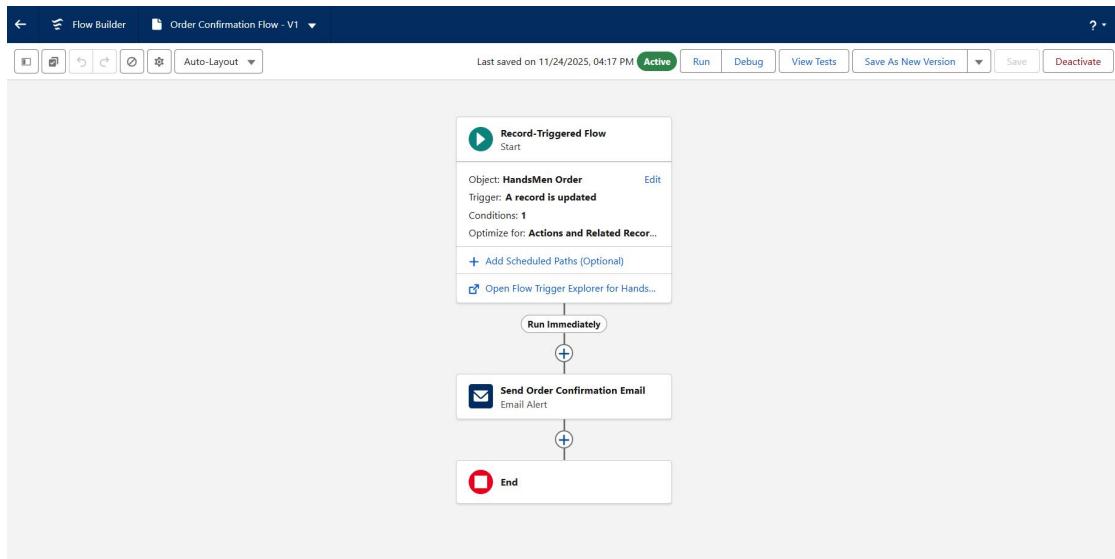
Salesforce Customer Record Edit Screen:

- Fields:** HandsMen Customer Name (Marinelle), Email (minettejoylorzano@xyz), Phone (09109581233), Loyalty Status (None), FirstName (Minette Joy), LastName (Lorzano), Total Purchases (512).
- Errors:** A validation error message "ø We hit a snag." is displayed over the Email field, stating "Email: invalid email address: minettejoylorzano@xyz".
- Details:** Created By (Minette Joy Lorzano, 11/26/2022) and Last Modified By (Minette Joy Lorzano, 11/26/2025, 5:51 PM).

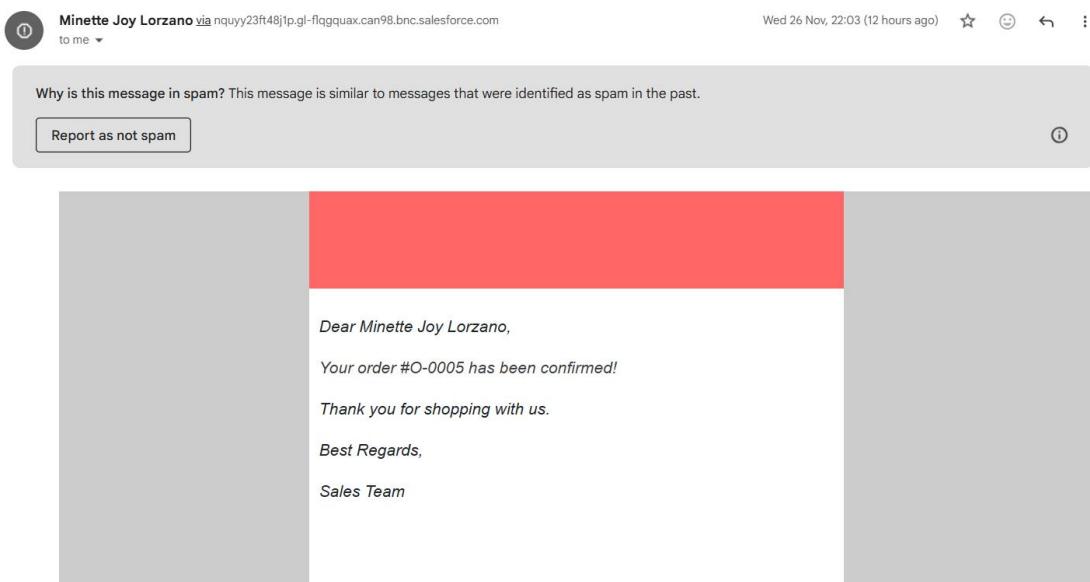
Automation(Flows)

Record Triggered Flows

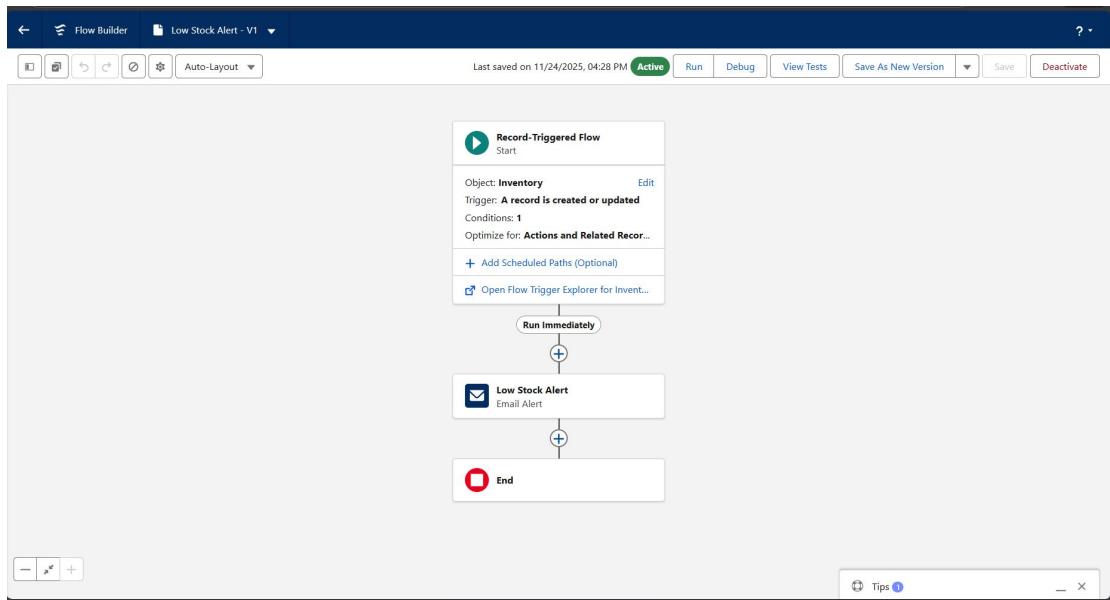
- Order Confirmation Flow



Your Order has been Confirmed! [Spam](#)



■ Low Stock Alert Flow



Stocks are getting low! Spam ×

Minette Joy Lorzano via 5w6g2yo2ga5v.gl-flqgquax.can98.bnc.salesforce.com
to me ▼

Wed 26 Nov, 22:03 (12 hours ago)

Why is this message in spam? This message is similar to messages that were identified as spam in the past.

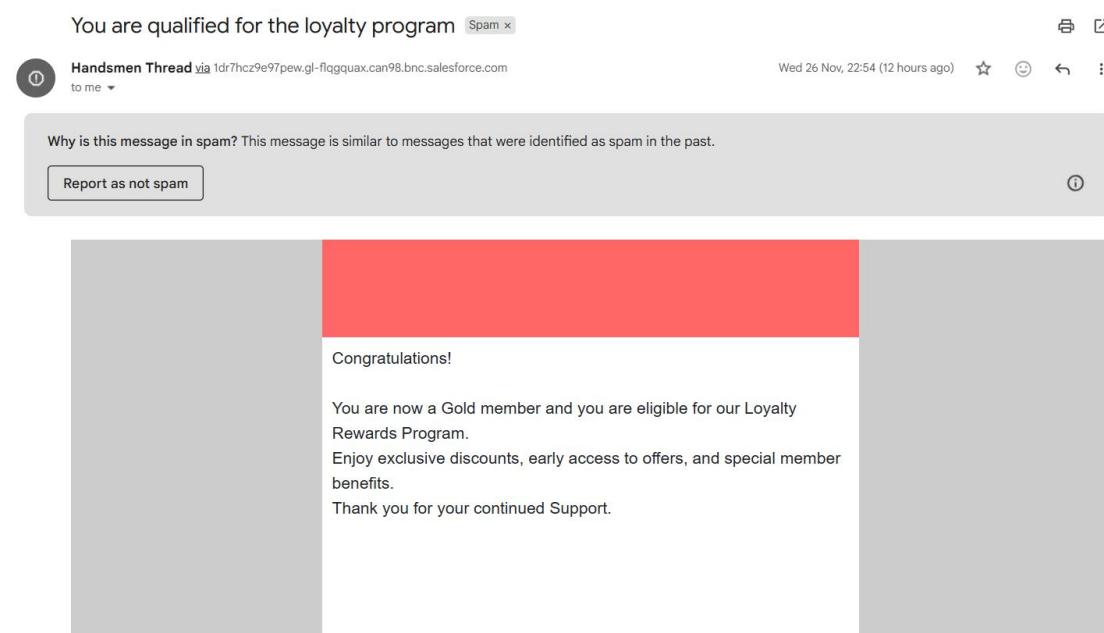
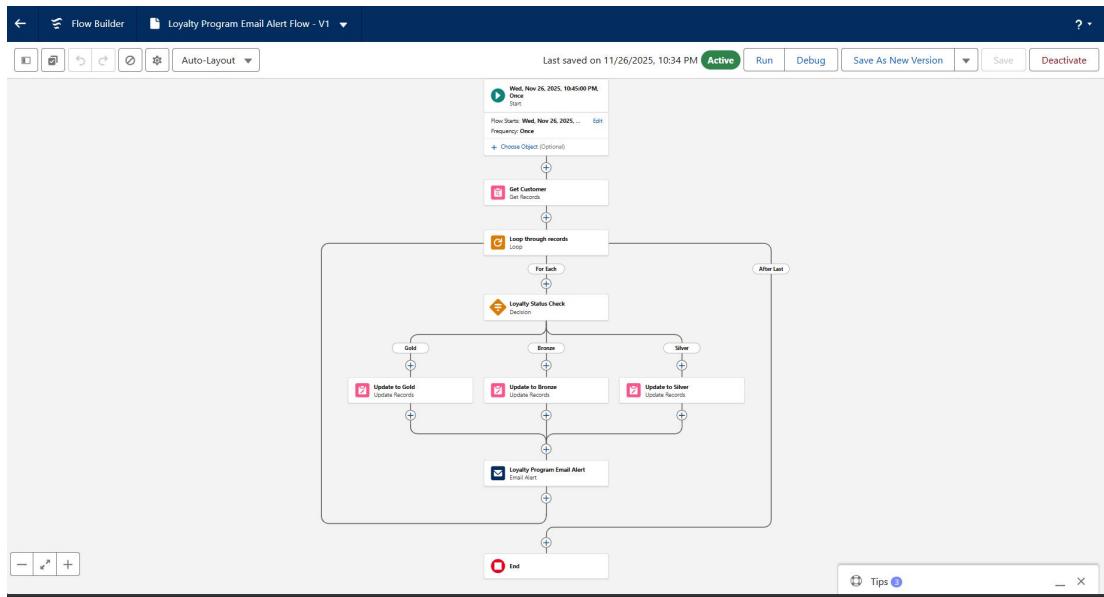
[Report as not spam](#)

Dear Inventory Manager,
 This is to inform you that the stock for the following product is running low:
 Product Name: Baggy Jeans
 Current Stock Quantity: 3
 Please take the necessary steps to restock this item immediately.
 Best Regards,
 Inventory Monitoring System

[Reply](#) [Forward](#)

Scheduled-Triggered Flows

■ Loyalty Program Flow



2.3 Apex Classes, Triggers, Asynchronous Apex Classes

Apex Classes

- Inventory Batch Job - Runs on a scheduled basis to monitor inventory levels and update stock data, ensuring proactive stock replenishment and accurate inventory tracking.

The screenshot shows the Apex Class editor with the tab 'Apex Class' selected. The code is for a class named 'InventoryBatchJob' which implements Database.Batchable<SObject> and Schedulable. It queries for products where Stock_Quantity__c is less than 10, loops through the results, and restocks each product by adding 50 units. It then adds the updated product to a list if it's not empty. The code is as follows:

```

1 global class InventoryBatchJob implements Database.Batchable<SObject>, Schedulable {
2     global Database.QueryLocator start(Database.BatchableContext BC) {
3         return Database.getQueryLocator(
4             'SELECT Id, Stock_Quantity__c FROM Product__c WHERE Stock_Quantity__c < 10'
5         );
6     }
7     global void execute(Database.BatchableContext BC, List<SObject> records) {
8         List<HandsMen_Product__c> productsToUpdate = new List<HandsMen_Product__c>();
9         // Cast SObject list to Product__c list
10        for (SObject record : records) {
11            HandsMen_Product__c product = (HandsMen_Product__c) record;
12            product.Stock_Quantity__c += 50; // Restock logic
13            productsToUpdate.add(product);
14        }
15        if (!productsToUpdate.isEmpty()) {
16
17
18
19
20
21
22
23
24
25
26
27
28
29
}

```

Position: Ln 61, Ch 2 | Total: Ln 61, Ch 1020

Apex Triggers

- Order Total Trigger - Automatically calculates the Total Amount based on the order quantity and unit price.

The screenshot shows the Apex Trigger editor with the tab 'Apex Trigger' selected. The trigger is named 'OrderTotalTrigger' on the 'HandsMen_Order__c' object, running before insert and update. It first collects all product IDs from the order lines. Then, it creates a map where keys are product IDs and values are product prices. Finally, it iterates over the orders and calculates the total amount by multiplying quantity and price for each line item. The code is as follows:

```

1 trigger OrderTotalTrigger on HandsMen_Order__c (before insert, before update) {
2     Set<Id> productIds = new Set<Id>();
3
4     for (HandsMen_Order__c order : Trigger.new) {
5         if (order.HandsMen_Product__c != null) {
6             productIds.add(order.HandsMen_Product__c);
7         }
8     }
9
10    Map<Id, HandsMen_Product__c> productMap = new Map<Id, HandsMen_Product__c>(
11        [SELECT Id, Price__c FROM HandsMen_Product__c WHERE Id IN :productIds]
12    );
13
14    for (HandsMen_Order__c order : Trigger.new) {
15        if (order.HandsMen_Product__c != null && productMap.containsKey(order.HandsMen_Product__c)) {
16            HandsMen_Product__c product = productMap.get(order.HandsMen_Product__c);
17            if (order.Quantity__c != null) {
18                order.Total_Amount__c = order.Quantity__c * product.Price__c;
19            }
20        }
21    }
22
}

```

- Stock Deduction Trigger - Automatically deducts inventory stock levels whenever an order is placed and confirmed.

```

1 * trigger StockDeductionTrigger on HandsMen_Order__c (after insert, after update) {
2     Set<Id> productIds = new Set<Id>();
3
4     for (HandsMen_Order__c order : Trigger.new) {
5         if (order.Status__c == 'Confirmed' && order.HandsMen_Product__c != null) {
6             productIds.add(order.HandsMen_Product__c);
7         }
8     }
9
10    if (productIds.isEmpty()) return;
11
12    // Query related inventories based on product
13    Map<Id, Inventory__c> inventoryMap = new Map<Id, Inventory__c>(
14        [SELECT Id, Stock_Quantity__c, Product__c
15         FROM Inventory__c
16         WHERE Product__c IN :productIds]
17    );
18
19    List<Inventory__c> inventoriesToUpdate = new List<Inventory__c>();
20
21    for (HandsMen_Order__c order : Trigger.new) {
22        if (order.Status__c == 'Confirmed' && order.HandsMen_Product__c != null) {
23            for (Inventory__c inv : inventoryMap.values()) {
24                if (inv.Product__c == order.HandsMen_Product__c) {
25                    inv.Stock_Quantity__c -= order.Quantity__c;
26                    inventoriesToUpdate.add(inv);
27                    break;
28                }
29            }
}

```

Phase 3: UI/UX Development & Customization

3.1 Lightning App Setup

Create and configure a custom Lightning App using Salesforce's App Manager.

The screenshot shows the Salesforce App Manager setup page. On the left, there is a sidebar with various links like Setup Home, Salesforce Go, Service Setup Assistant, etc. The main area has a search bar at the top. Below it, there is a section titled "Lightning App Builder" with a sub-section "Lightning Pages". A message states: "The Lightning App Builder provides an easy to use graphical interface for creating custom Lightning pages for Salesforce Lightning Experience and mobile app. Lightning pages are built using Lightning components—compact, configurable, and reusable elements that you can drag and drop into regions of the page in the Lightning App Builder." Below this, there is a table titled "Lightning Pages" with columns for Label, Name, Namespace Prefix, Description, Type, Created By, and Last Modified By. The table currently shows no results.

3.2 Page Layouts and Dynamic Forms

Customized the page layouts for key objects (Customer, Order, Product, Inventory) to streamline data entry and ensure appropriate visibility based on user roles. Implemented dynamic forms that adjust fields and sections according to user context, improving both usability and data accuracy.

3.3 User Management

User profiles, roles, and permission sets were established to provide controlled access to data and features within the Lightning App, ensuring secure and relevant experiences tailored to each department and responsibility.

Action	Full Name	Alias	Username	Role	Active	Profile
<input type="checkbox"/> Edit	Chatter Expert	Chatter	chatty@00dg0000ffpguax.3mzqzwidfe@chatter.salesforce.com		✓	Chatter Free User
<input type="checkbox"/> Edit	OEPIC_OrgFarm	OEPIC	epic.2cc53935eed4@orgfarm.salesforce.com		✓	System Administrator
<input type="checkbox"/> Edit	Lorzano_Minette_Joy	min	minettejoylorzano548@agentforce.com		✓	System Administrator
<input type="checkbox"/> Edit	Mika_Morko	mnika	mnika@user.com	Inventory	✓	Partner App Subscription User
<input type="checkbox"/> Edit	Mikaelson_Kol	kmika	kmika@user.com		✓	Platform_1
<input type="checkbox"/> Edit	Mikaelson_Morko	mmika01	mmika01@user.com	Marketing	✓	Platform_1
<input type="checkbox"/> Edit	Mikaelson_Niklaus	nmika	nmika@user.com	Sales	✓	Platform_1
<input type="checkbox"/> Edit	User_Integration	integ	integration@00dg0000ffpguax.com		✓	Analytics Cloud Integration User
<input type="checkbox"/> Edit	User_Security	sec	insightssecurity@00dg0000ffpguax.com		✓	Analytics Cloud Security User

The screenshot shows the Salesforce Setup interface with the 'Users' page selected. The left sidebar shows navigation options like 'Permission Set Groups', 'Profiles', 'Public Groups', 'Queues', 'Roles', 'User Management Settings', 'Apps', 'Feature Settings', 'Prospector', 'Service', and 'Embedded Service'. The main content area shows 'Permission Set Assignments' for a user named 'Sales Permission Set' assigned on 11/23/2025. It also lists 'Permission Set Group Assignments', 'Permission Set License Assignments', 'Personal Groups', 'Public Group Membership', and 'Queue Membership'.

Phase 4: Data Migration, Testing & Security

4.1 Data Loading Process

Existing data—including customers, orders, and inventory—was migrated using the Salesforce Data Import Wizard for smaller datasets and Data Loader for bulk uploads. Prior to migration, data was cleansed and mapped to ensure accuracy and consistency.

The screenshot shows the Data Import Wizard interface. It features a 'Recent Import Jobs' section with a table header for Status, Object, Records Created, Records Updated, Records Failed, Start Date, and Processing Time (ms). Below this is a 'Bulk Api Monitoring' button. A large callout box provides tips: 'Clean up your data import file', 'Before you import your data...', 'Make sure your field names match Salesforce field names', and 'Don't import too many records at once'. At the bottom, it says 'Import your data in 3 easy steps!' and 'Launch the Data Import Wizard to import your data.' with four circular icons representing the steps.

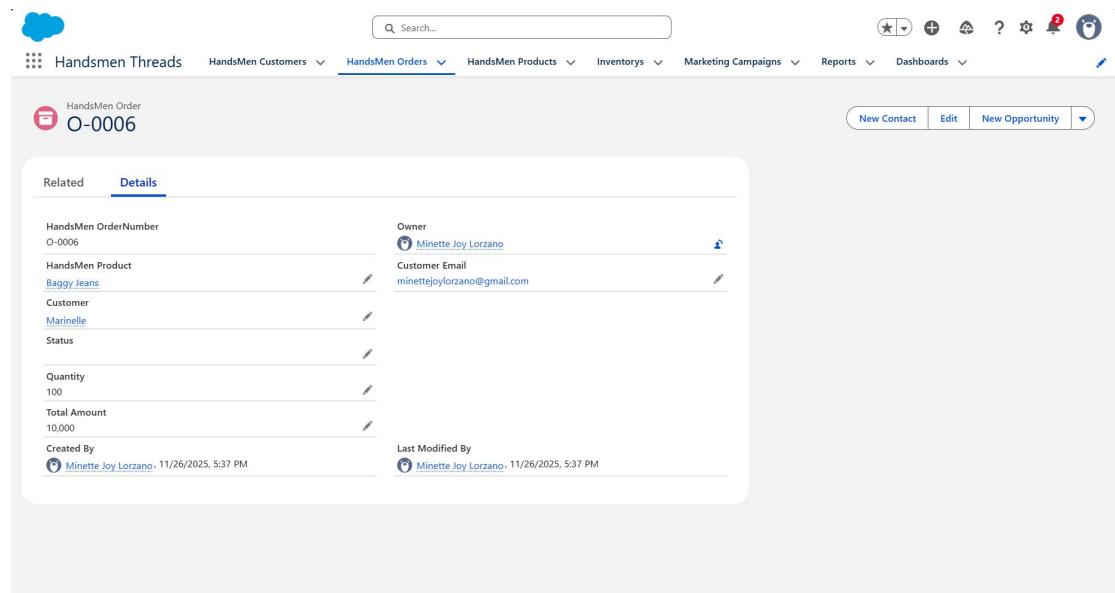
4.2 Field History Tracking, Duplicate Rules, Matching Rules

Tracked Objects & Fields:

HandsMen Orders – Fields tracked: Status, Total Amount

- This object monitors the order lifecycle and financial value.
- Example Scenario: Annie decides to purchase 50 eyeglasses, each priced at \$2.
- A new Order record is created in Salesforce that captures all details of her purchase.

An Apex Trigger automatically calculates the Total Amount, ensuring the computation is consistent and error-free.



The screenshot shows the HandsMen Orders detail page in Salesforce. The page header includes the Salesforce logo, a search bar, and navigation links for Handsmen Threads, HandsMen Customers, HandsMen Orders (selected), HandsMen Products, Inventory, Marketing Campaigns, Reports, and Dashboards. Below the header, there's a toolbar with New Contact, Edit, and New Opportunity buttons. The main content area displays the following fields:

Field	Value
HandsMen OrderNumber	O-0006
HandsMen Product	Baggy Jeans
Customer	Marinelle
Status	(empty)
Quantity	100
Total Amount	10,000
Created By	Minette Joy Lorzano - 11/26/2025, 5:37 PM
Owner	Minette Joy Lorzano
Customer Email	minettejoylorzano@gmail.com
Last Modified By	Minette Joy Lorzano - 11/26/2025, 5:37 PM

HandsMen Products – Fields Tracked: Price, Quantity

- The admin adds products such as eyeglasses, T-shirts, and other items into the Product__c object.
- Each product record includes key details like price, SKU, and other essential attributes.
- The system monitors pricing updates and stock changes to maintain accuracy.

- The Inventory object is automatically updated to reflect stock levels for each product, enabling real-time tracking of item availability.

The top screenshot shows the 'HandsMen Products' page for the 'Baggy Jeans' product. It displays fields such as Product Name, Order, SKU (I-0001), Price (\$100), Stock Quantity, and ownership information (Owner: Minette Joy Lorzano). The bottom screenshot shows the 'Inventory' page for item I-0001, which tracks the same product details and includes additional fields like Stock Status (Low Stock) and Warehouse (Hummy).

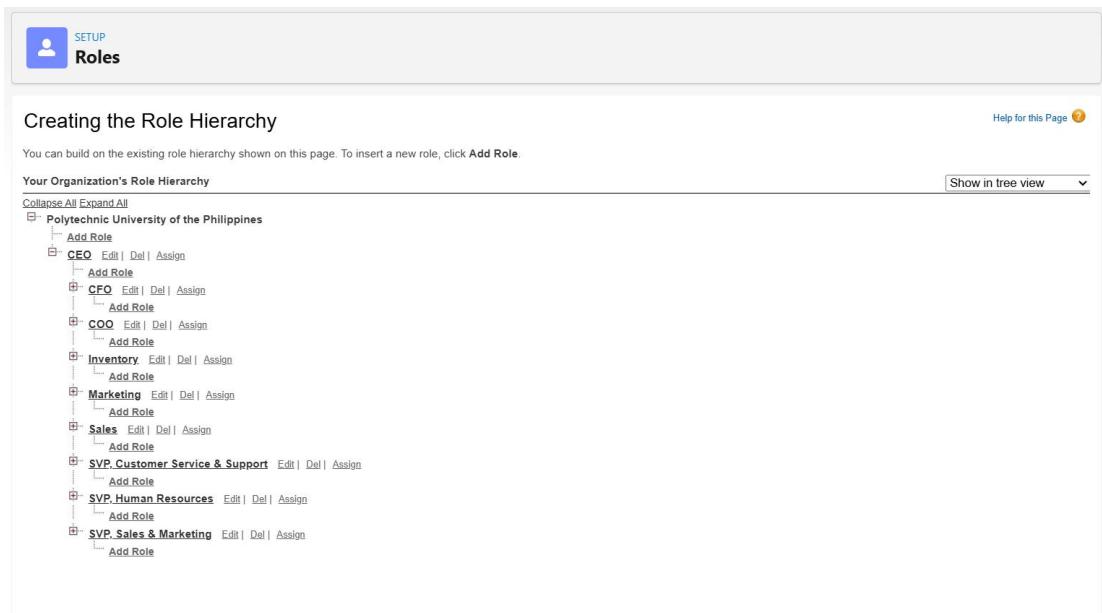
Duplicate and Matching Rules – Implemented to prevent duplicate entries and ensure accurate customer records.

Standard Customer Matching Rule:

- Compares records based on *First Name*, *Last Name*, and *Email*.
- Restricts the creation of new customer records if an existing match is found.

4.3 Profiles, roles, role hierarchy, permission sets, and sharing rules

An integrated security model was aligned with the organizational chart, using profiles, roles, permission sets, and sharing rules to ensure proper access, visibility, and editing rights for Sales, Inventory, and Marketing teams.



The screenshot shows the 'Creating the Role Hierarchy' page in the Salesforce Setup Roles section. At the top, there's a header with a user icon, 'SETUP', and 'Roles'. Below it is a sub-header 'Creating the Role Hierarchy' with a 'Help for this Page' link. A message says 'You can build on the existing role hierarchy shown on this page. To insert a new role, click Add Role.' There are buttons for 'Collapse All' and 'Expand All'. A 'Your Organization's Role Hierarchy' tree is displayed under 'Polytechnic University of the Philippines'. The tree includes nodes for CEO, CFO, COO, Inventory, Marketing, Sales, SVP Customer Service & Support, SVP Human Resources, and SVP Sales & Marketing, each with 'Edit | Del | Assign' options. There are also 'Add Role' buttons next to each node. A 'Show in tree view' dropdown is visible at the bottom right.

Phase 5: Deployment, Documentation & Maintenance

5.1 Deployment Strategy

- The system was deployed using Salesforce Change Sets for configuration and metadata migration between environments.
- For larger or complex updates, Salesforce CLI and DevOps tools were utilized to ensure smooth, controlled releases.
- Deployment followed a staged approach: development → testing → production, with validation at each step to minimize risks.

5.2 System Maintenance & Monitoring

- Regular monitoring of system performance, data integrity, and automation workflows was established.
- Scheduled Apex jobs track inventory levels and ensure proactive replenishment.

- User feedback loops and periodic audits help identify areas for improvement.
- Security updates and role-based access reviews are conducted to maintain compliance and safeguard data.

5.3 Troubleshooting Approach

- Issues are logged and categorized by severity in a centralized tracking system.
- Common problems (e.g., validation errors, automation failures) are addressed through predefined troubleshooting guides.
- Escalation procedures ensure critical issues are resolved promptly, with rollback plans available if needed.
- Documentation of fixes and resolutions is maintained for future reference and knowledge sharing.

Conclusion

The Salesforce CRM implementation for HandsMen Threads successfully delivered a scalable, secure, and automated platform tailored to the company's needs. By consolidating customer, order, product, inventory, and marketing data into a unified system, the project enhanced customer engagement, streamlined workflows, and improved decision-making. Automation and role-based access controls strengthened efficiency and security, while proactive monitoring ensured operational reliability.

This capstone demonstrates how a well-designed Salesforce CRM can transform business operations, positioning HandsMen Threads for sustained growth, innovation, and competitive advantage in the men's fashion industry.