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

Project Overview:

The **HandsMen Threads** capstone project involved implementing a highly customized Salesforce Customer Relationship Management (CRM) solution designed to revolutionize the company's data management, inventory control, and customer loyalty processes.

As a premium men's fashion brand, HandsMen Threads' previous reliance on manual order processing, stock tracking, and static loyalty programs frequently led to delayed customer confirmations, inconsistent inventory data, and a generic, non-personalized customer experience.

This project's central aim was to streamline the entire order-to-loyalty lifecycle by leveraging the power of the Salesforce Platform to introduce key automated features, including Automated Order Confirmations (fostering engagement post-purchase), a Dynamic Loyalty Program (automatically updating customer status based on purchase history), and Proactive Stock Alerts (preventing stockouts by notifying the warehouse when levels drop below a critical threshold).

The final solution provides a clean, user-friendly interface using a Custom Lightning App and robust data validation, drastically improving internal user efficiency, ensuring data integrity from the UI, and establishing a scalable digital foundation for personalized customer engagement.

Objectives:

The Salesforce CRM implementation at HandsMen Threads focused on automating core manual processes and enhancing operational efficiency through the following objectives:

- **Implement a Robust Custom Data Model:** Design and deploy a scalable data structure to store all core business data (Products, Inventory, Orders, Loyalty Status) with appropriate relationships.
- **Ensure Data Quality and Integrity:** Implement validation rules and UI constraints to safeguard the accuracy and consistency of data directly from the user interface.
- **Automate Order Confirmation:** Deploy a Record-Triggered Flow to send immediate, automated email updates to customers post-order confirmation.

- **Automate Dynamic Loyalty Updates:** Implement Apex logic to automatically update a customer's loyalty status (e.g., Bronze, Silver, Gold) based on their accumulated purchase history.
- **Implement Proactive Stock Alerts:** Utilize a Record-Triggered Flow to send automatic email notifications to the Warehouse team when any product's stock level drops below five units.
- **Establish a Scalable Backend:** Utilize Batch Apex to process high-volume, asynchronous bulk order updates (e.g., financial record adjustments) scheduled to run daily at midnight.
- **Enhance Internal UX:** Deliver a modern, efficient user interface for internal teams using a Custom Lightning App and Dynamic Forms for improved order and inventory management.

Requirement Analysis & Planning:

This phase establishes the foundation of the project by defining the business problem, scope, and objectives. We analyze user needs, document specific problems (e.g., data inconsistency), and design the high-level Data Model and Security Model. The output is a clear Execution RoadMap and agreed-upon project boundaries, ensuring stakeholder alignment before development starts.

- **Understanding Business Requirements: Summarize the user needs and problems being solved:**
    - HandsMen Threads is tackling the core problem of disjointed and inaccurate data management, which currently hinders informed decision-making and operational flow. The primary user needs center on achieving data integrity directly at the user interface level, enhancing customer engagement through timely communication and a dynamic loyalty program, and significantly boosting operational efficiency by automating critical processes like stock alerts and scheduled bulk order updates. The project will replace manual, error-prone tasks with a robust, centralized Salesforce solution, ensuring the business can scale effectively.
- **Defining Project Scope and Objectives:**
    - The project's scope is anchored in building a complete data model (including objects, fields, and complex relationships) and implementing four key automation flows. Specifically, this involves creating the necessary Validation Rules for data

quality and designing professional email templates for customer and internal communication. The core objectives are measurable: achieving 100% data modeling of critical business information, ensuring 99% data accuracy via UI validations, and implementing the automations to achieve results like 100% automated order confirmations and an 80% reduction in manual inventory monitoring tasks.

- **Design Data Model and Security Model**
    - The Data Model is conceptualized around core entities like Customer, Product, Order, Order Item, and a custom Loyalty Status object, utilizing formula fields and relationships to link business data seamlessly. Crucially, the Security Model will enforce the principle of least privilege, with Organization-Wide Defaults (OWD) set to private for sensitive records like Orders. Access will then be strategically opened up using Profiles/Permission Sets (to define what users can *do*) and Sharing Rules (to define what users can *see*), ensuring the Warehouse Team has different access than the Sales Team.
- **Stakeholders Mapping**
    - The success of the project relies on collaboration across four main groups. Executive Leadership acts as the sponsor, ensuring budget and strategic alignment for ROI and growth. The Sales/Fashion Team and the Warehouse/Inventory Team are the key end-users who validate requirements and benefit directly from the improved data access and automation. The IT/Development Team is responsible for the technical implementation, quality assurance, and post-go-live support, ensuring the platform is stable and correctly architected.
- **Execution RoadMap**
    - The project follows a structured, four-phase roadmap. Phase 1: Architecture & Planning focuses on defining all specifications, including the data model and detailed automation logic (Flows, Apex, Batch Apex). Phase 2: Development is the hands-on building phase, where objects are created and all automations and the security model are fully implemented in a sandbox environment. Phase 3: Testing & QA is critical for quality control, involving comprehensive Unit, End-to-End, and Performance Testing. Finally, Phase 4: Deployment & Training involves

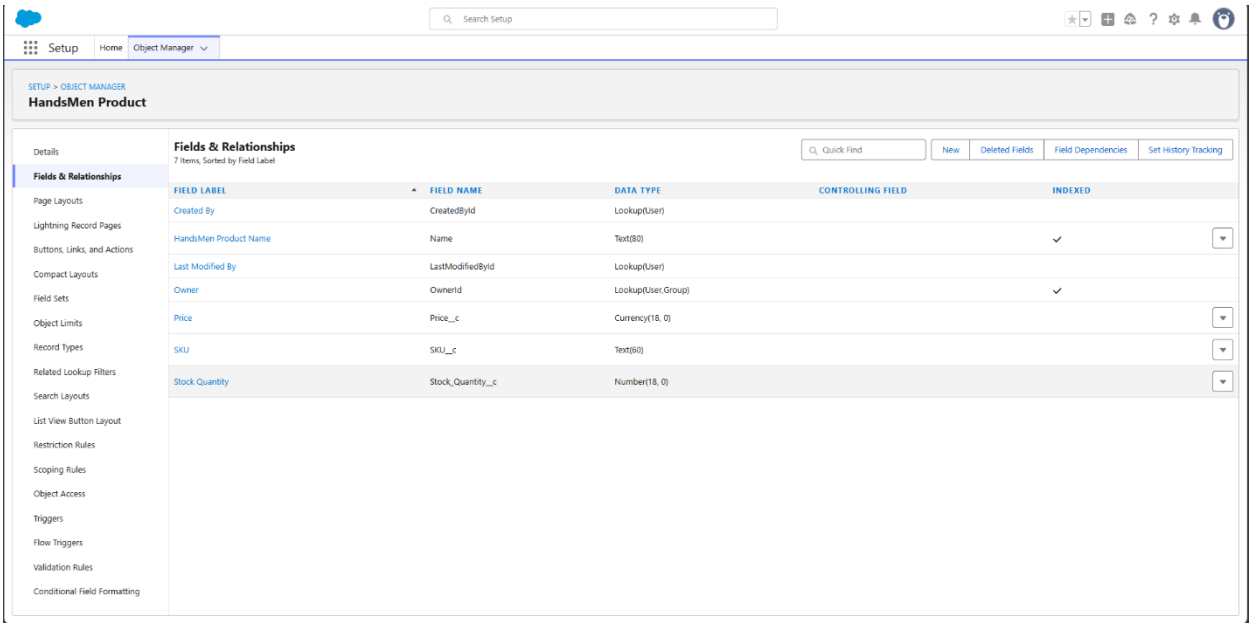
moving the finished product to production, providing training to all user groups, and offering essential post-go-live support to ensure smooth user adoption.

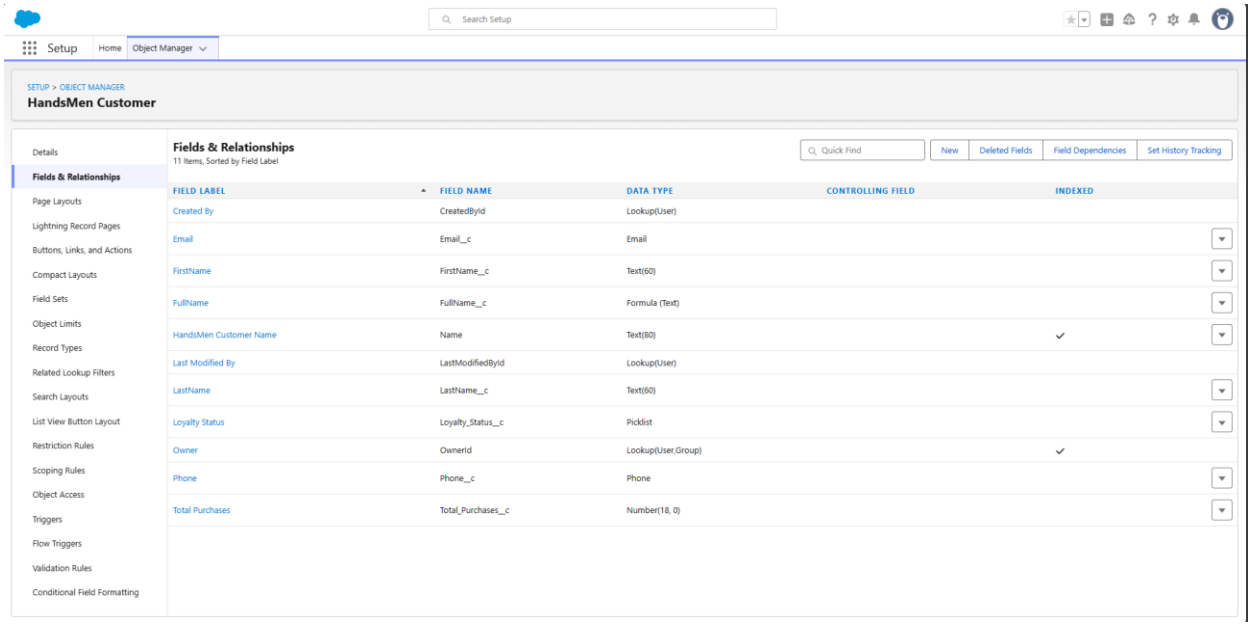
Salesforce Development - Backend & Configurations

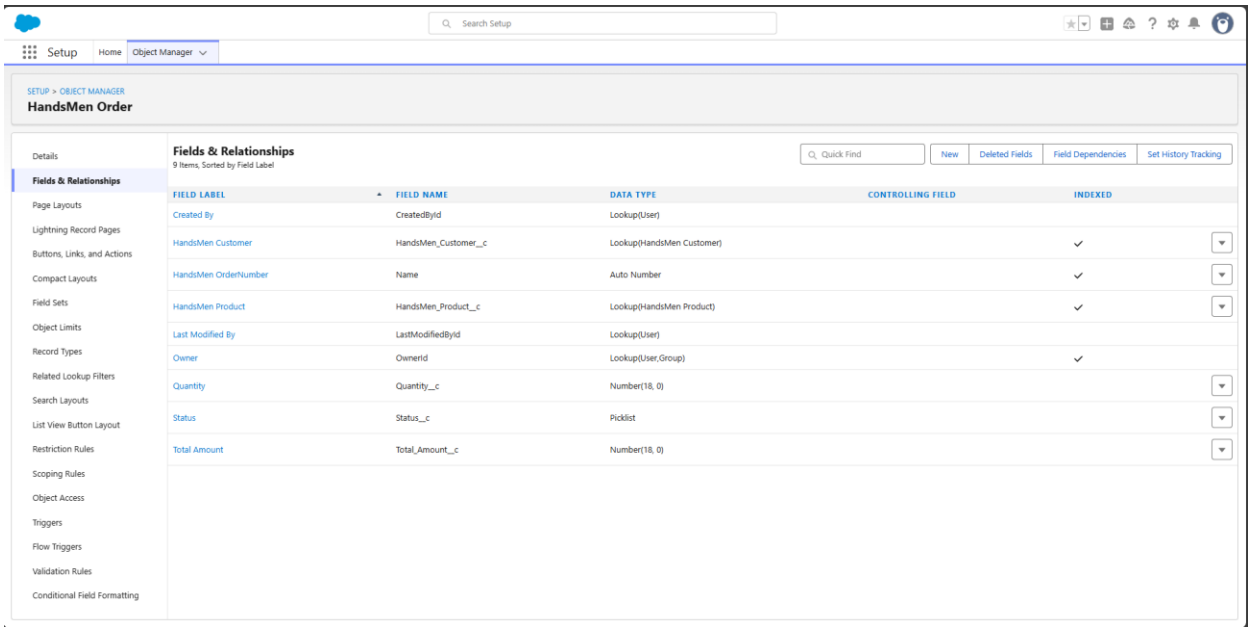
This phase is dedicated to building and configuring the core functionality defined in the planning stage, focusing on the data model, security, and automation mechanisms within the Salesforce environment for HandsMen Threads.

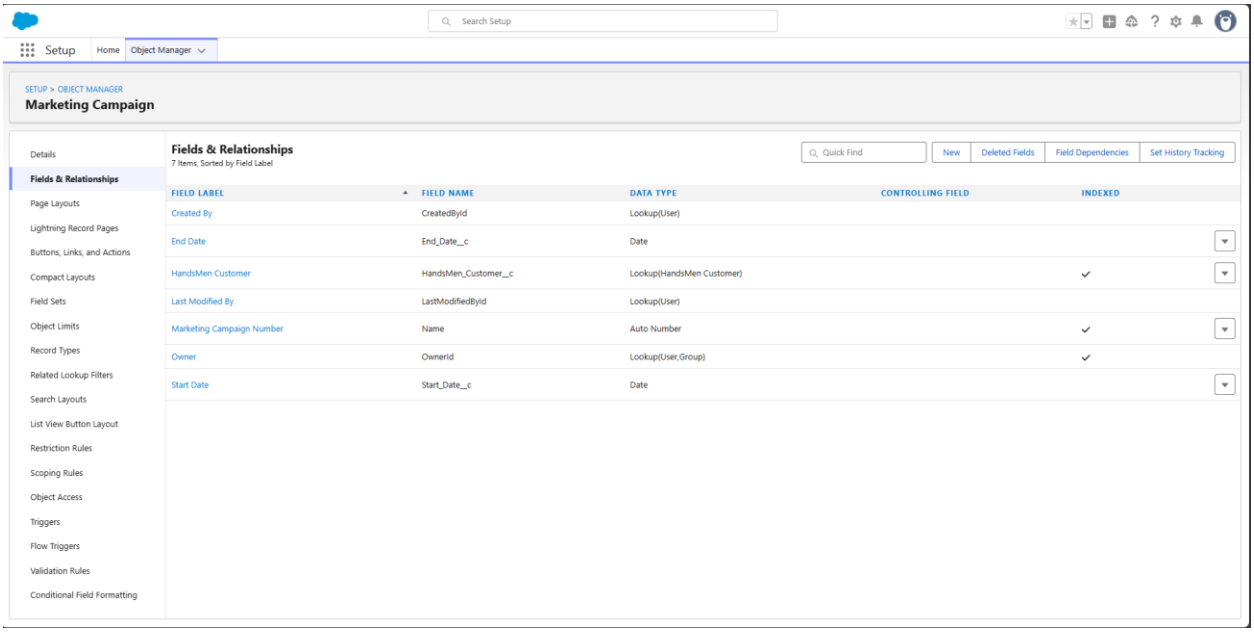
- **Data Model Customization**
  - **Custom Objects:** Objects like Order, OrderItem, and Loyalty Status are cred to capture specific business data not covered by standard objects.
  - **Custom Fields:** Essential fields are added to both standard (e.g., Loyalty_Status__c on the Handsmen Customer object) and custom objects.

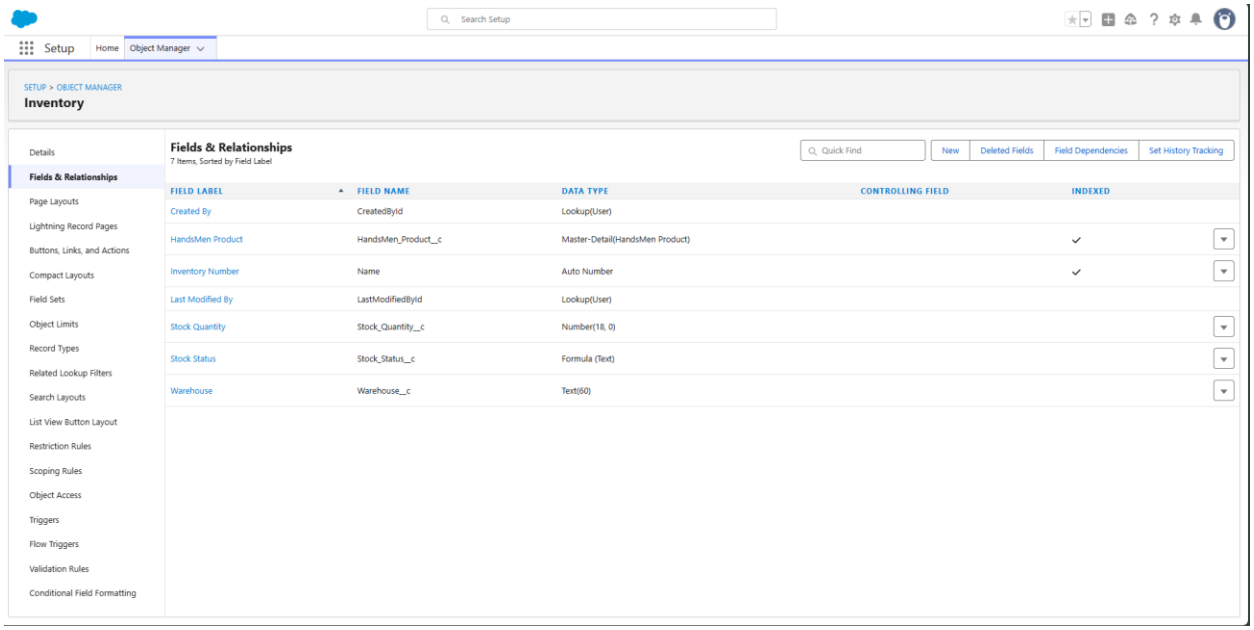
Screenshots of Custom Objects and Fields:




Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
HandsMen Customer
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Restriction Rules
Scoping Rules
Object Access
Triggers
Flow Triggers
Validation Rules
Conditional Field Formatting
Fields & Relationships
11 Items, Sorted by Field Label
Quick Find
New
Deleted Fields
Field Dependencies
Set History Tracking
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(80)
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)

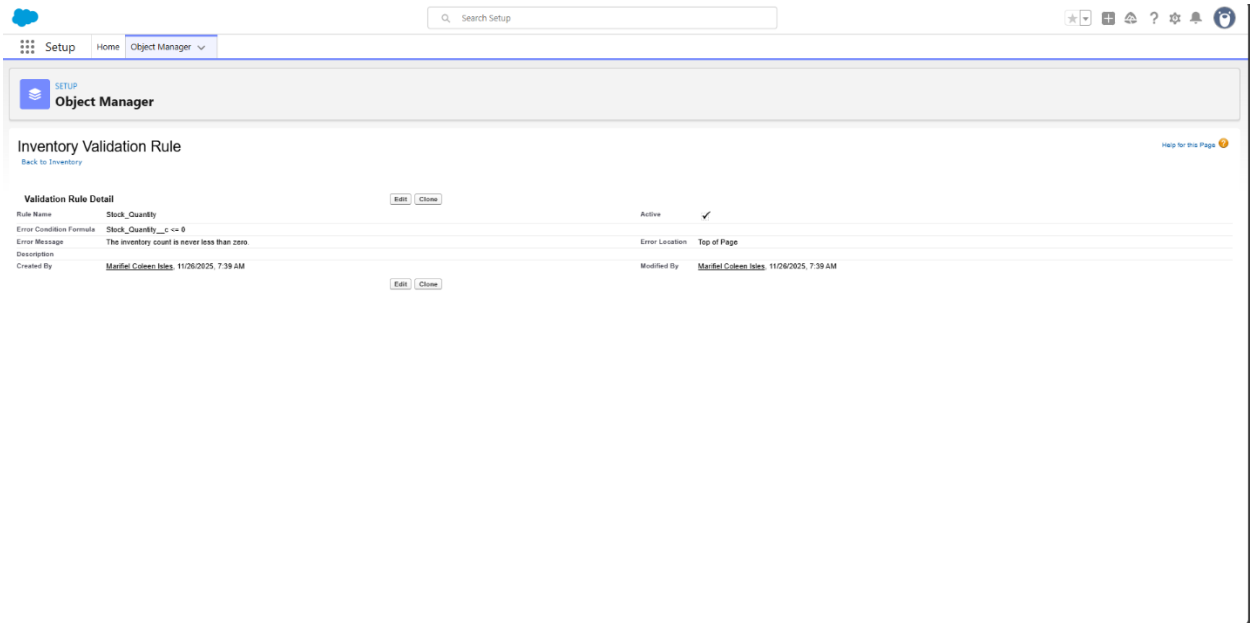

Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
HandsMen Order
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Restriction Rules
Scoping Rules
Object Access
Triggers
Flow Triggers
Validation Rules
Conditional Field Formatting
Fields & Relationships
9 Items, Sorted by Field Label
Quick Find
New
Deleted Fields
Field Dependencies
Set History Tracking
FIELD LABEL
FIELD NAME
DATA TYPE
CONTROLLING FIELD
INDEXED
Created By
CreatedById
Lookup(User)
HandsMen Customer
HandsMen_Customer__c
Lookup(HandsMen Customer)
HandsMen OrderNumber
Name
Auto Number
HandsMen Product
HandsMen_Product__c
Lookup(HandsMen Product)
Last Modified By
LastModifiedById
Lookup(User)
Owner
OwnerId
Lookup(User,Group)
Quantity
Quantity__c
Number(18, 0)
Status
Status__c
Picklist
Total Amount
Total_Amount__c
Number(18, 0)

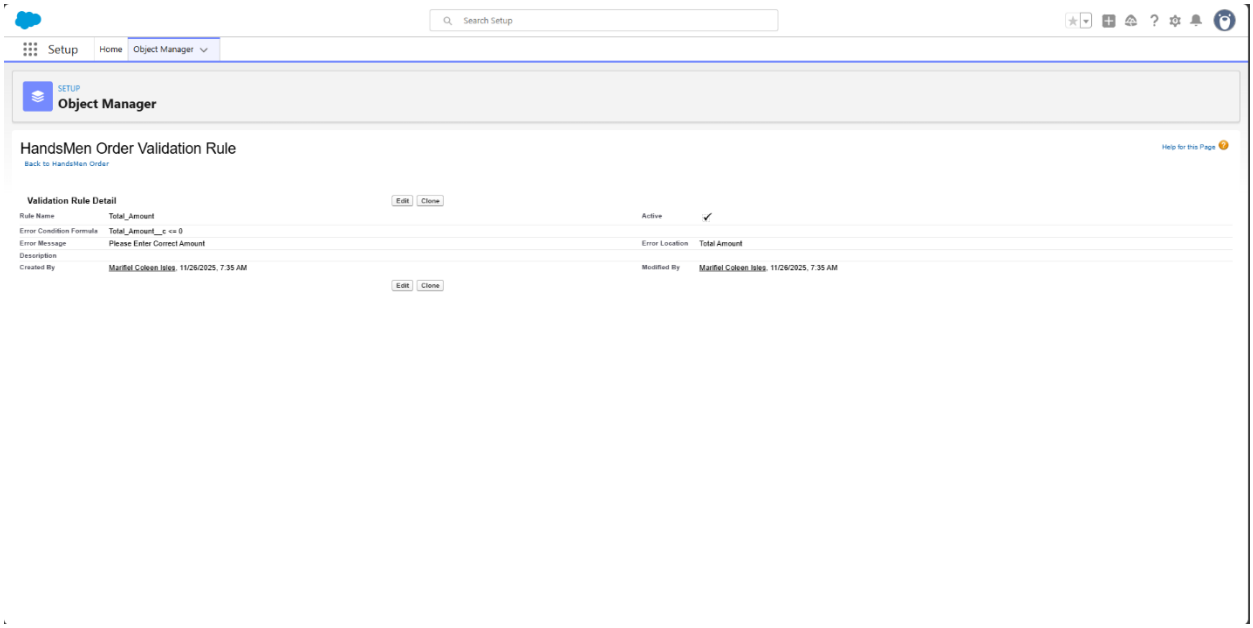

Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
Marketing Campaign
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Restriction Rules
Scoping Rules
Object Access
Triggers
Flow Triggers
Validation Rules
Conditional Field Formatting
Fields & Relationships
7 Items, Sorted by Field Label
Quick Find
New
Deleted Fields
Field Dependencies
Set History Tracking
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
Owner
OwnerId
Lookup(User,Group)
Start Date
Start_Date__c
Date

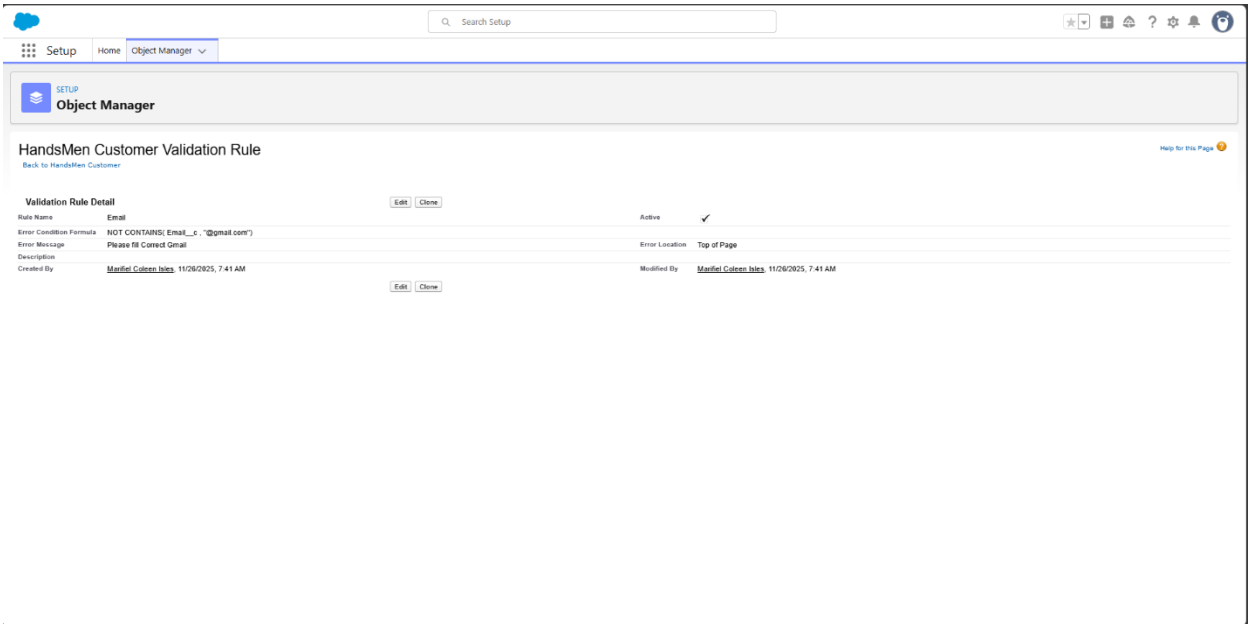

Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
Inventory
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Restriction Rules
Scoping Rules
Object Access
Triggers
Flow Triggers
Validation Rules
Conditional Field Formatting
Fields & Relationships
7 Items, Sorted by Field Label
Quick Find
New
Deleted Fields
Field Dependencies
Set History Tracking
FIELD LABEL
FIELD NAME
DATA TYPE
CONTROLLING FIELD
INDEXED
Created By
CreatedById
Lookup(User)
HandsMen Product
HandsMen_Product__c
Master-Detail(HandsMen Product)
Inventory Number
Name
Auto Number
Last Modified By
LastModifiedById
Lookup(User)
Stock Quantity
Stock_Quantity__c
Number(18, 0)
Stock Status
Stock_Status__c
Formula (Text)
Warehouse
Warehouse__c
Text(60)

- **Validation Rules:** Implemented to enforce data quality directly on the UI. For instance, a rule is added to the Product object to prevent Stock_Level__c from being set below zero upon saving.

Screenshots of Validation Rules:

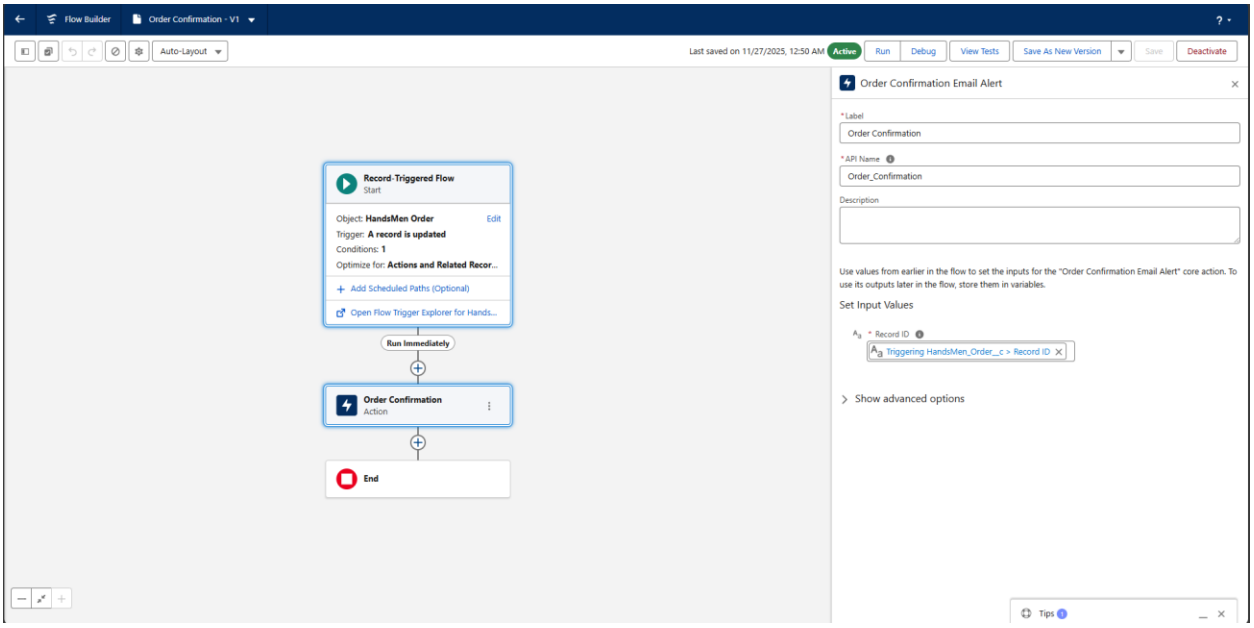


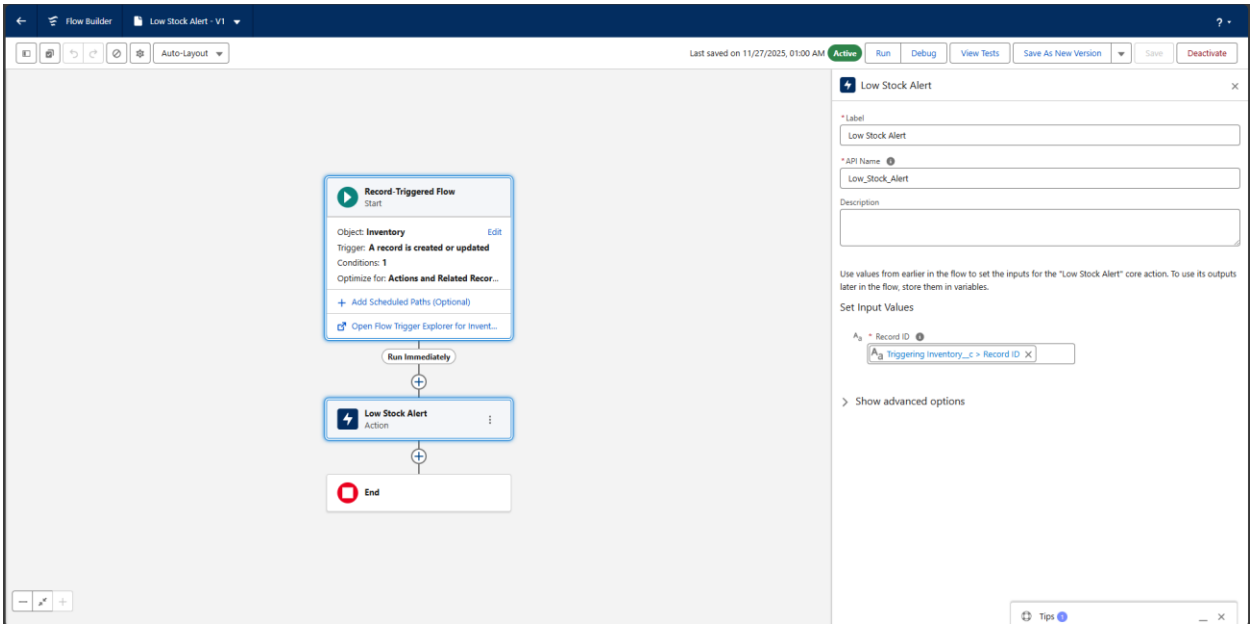


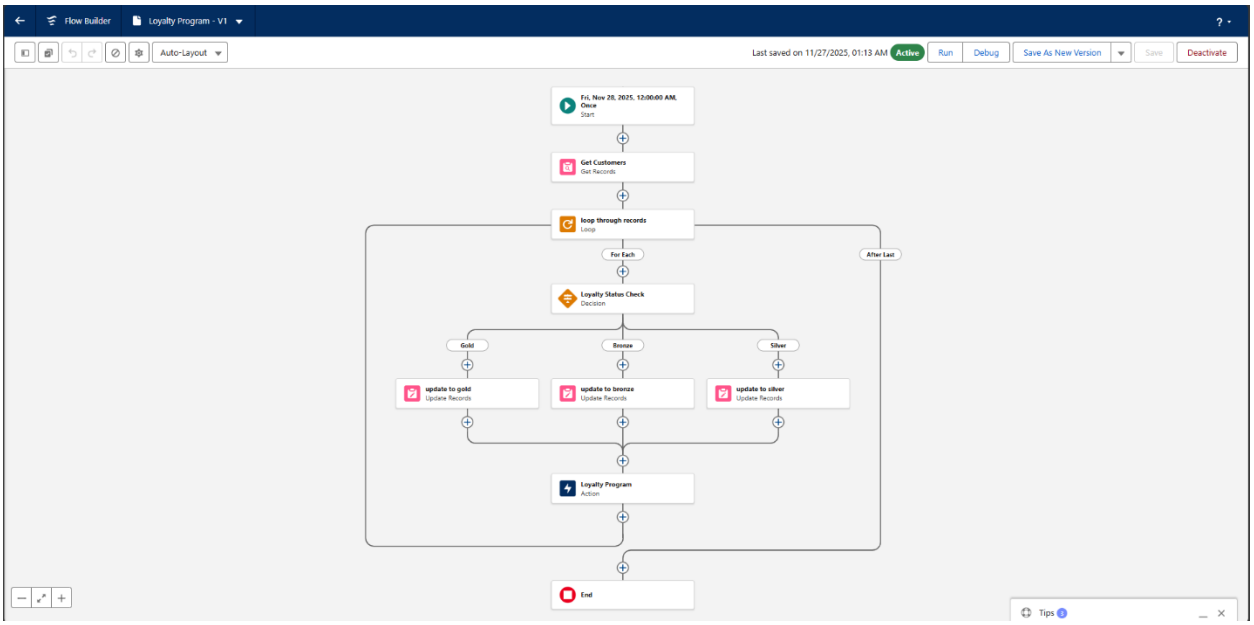


- **Automation (Flows):**
  - **Automated Order Confirmation:** A Record-Triggered Flow is built on the Order object. When an Order is marked as 'Confirmed', the Flow immediately executes an Email Alert action to send the customer the order details using the predefined email template.
  - **Dynamic Loyalty Program Update:** A second Record-Triggered Flow runs on the Order object upon creation. It calculates the total purchase amount and updates the associated Loyalty Status__c field on the parent Customer record (e.g., updating status from 'Silver' to 'Gold' if lifetime purchases exceed 1000).

Screenshots of Flows:

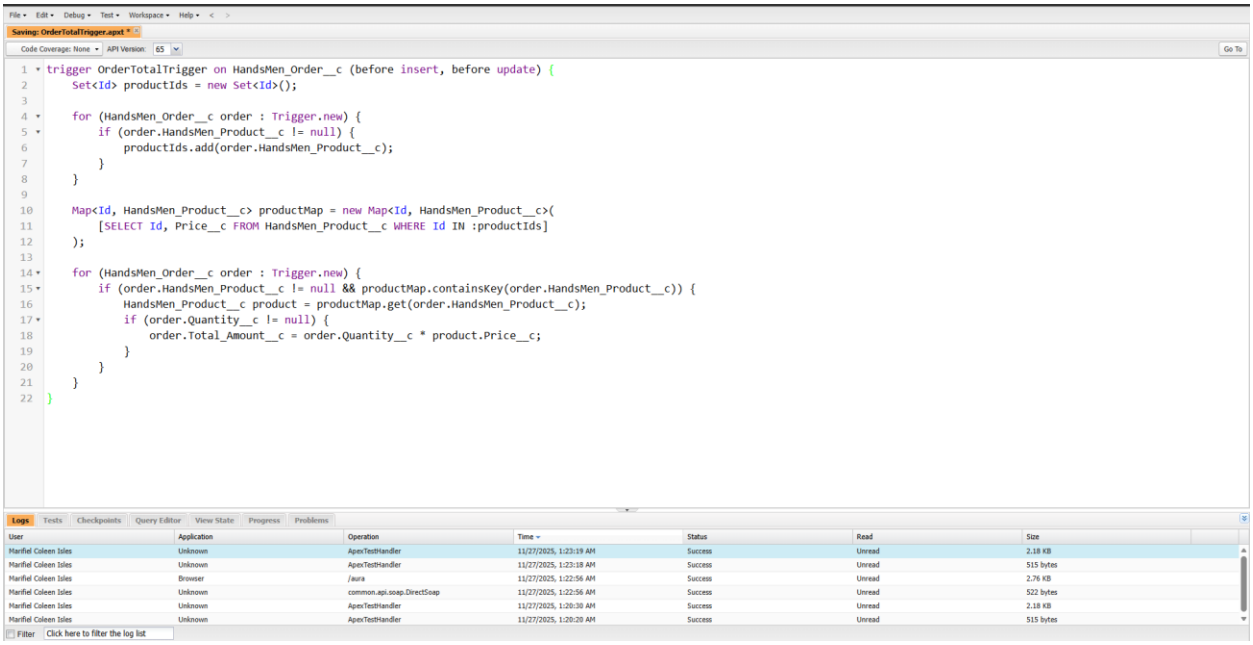




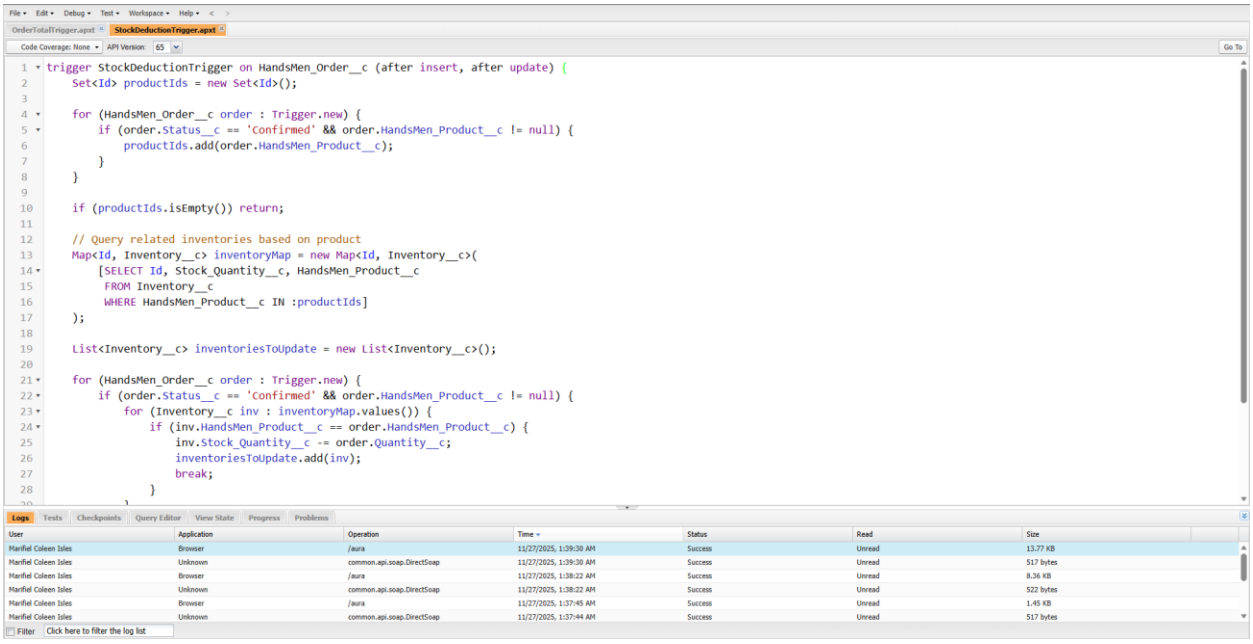


- **Apex Classes, Triggers, Asynchronous Apex Classes**
  - **Proactive Stock Alerts:** An Apex Trigger is implemented on the Product object. This trigger runs before update to check if the Stock_Level__c is being modified and is now less than 5 units. If the condition is met, the trigger invokes a helper class method that sends an email notification to the warehouse manager, ensuring timely restocking.
  - **Scheduled Bulk Order Updates:** A dedicated Batch Apex Class (e.g., BulkOrderProcessor) is developed to handle large-volume, scheduled data manipulation.
  - **Brief Explanation:** Batch Apex is necessary here because the requirement is to process bulk orders, update financial records, and adjust inventory daily at midnight. This operation is resource-intensive and runs against thousands of records, making it unsuitable for real-time triggers. Batch Apex efficiently processes records in manageable chunks (batches), ensuring platform limits are not exceeded.
  - **Utility & Test Classes:** Supporting Apex classes are developed for complex business logic (e.g., calculating tiered loyalty status) and to ensure high code coverage for all custom triggers and asynchronous code, meeting Salesforce's deployment standards.

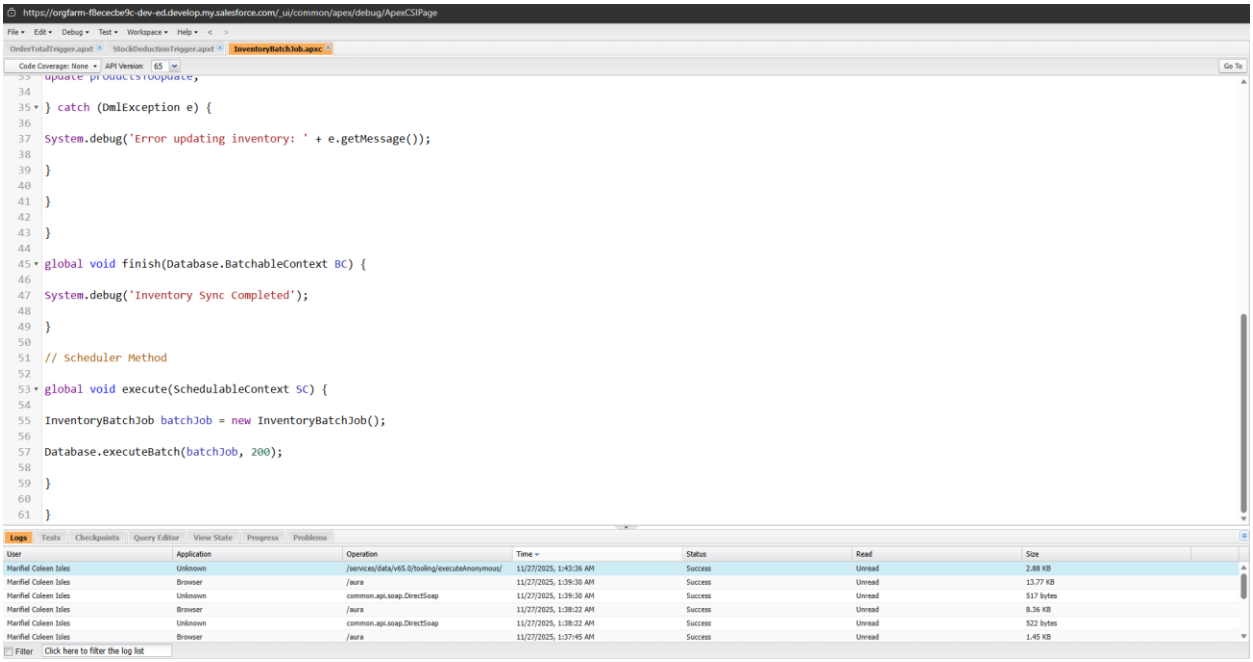
Screenshots of Apex Classes and Triggers:



```
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;
            }
        }
    }
}
```



```
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;
                }
```

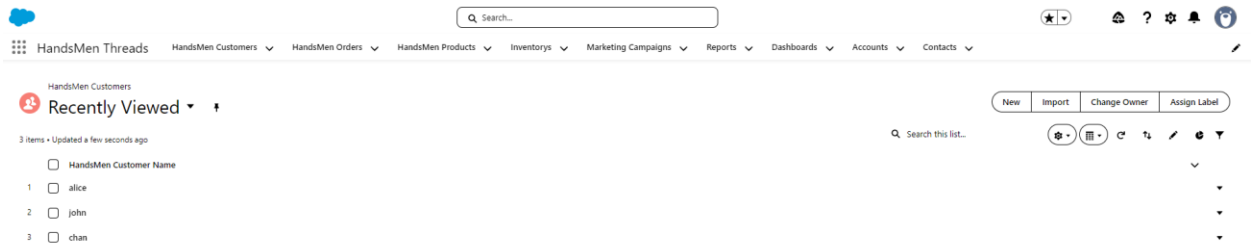


UI/UX Development & Customization

This phase focuses on enhancing the user experience (UX) and interface (UI) to make the new Salesforce system intuitive, efficient, and informative for all HandsMen Threads users, aligning with the principles learned via the Lightning App Builder.

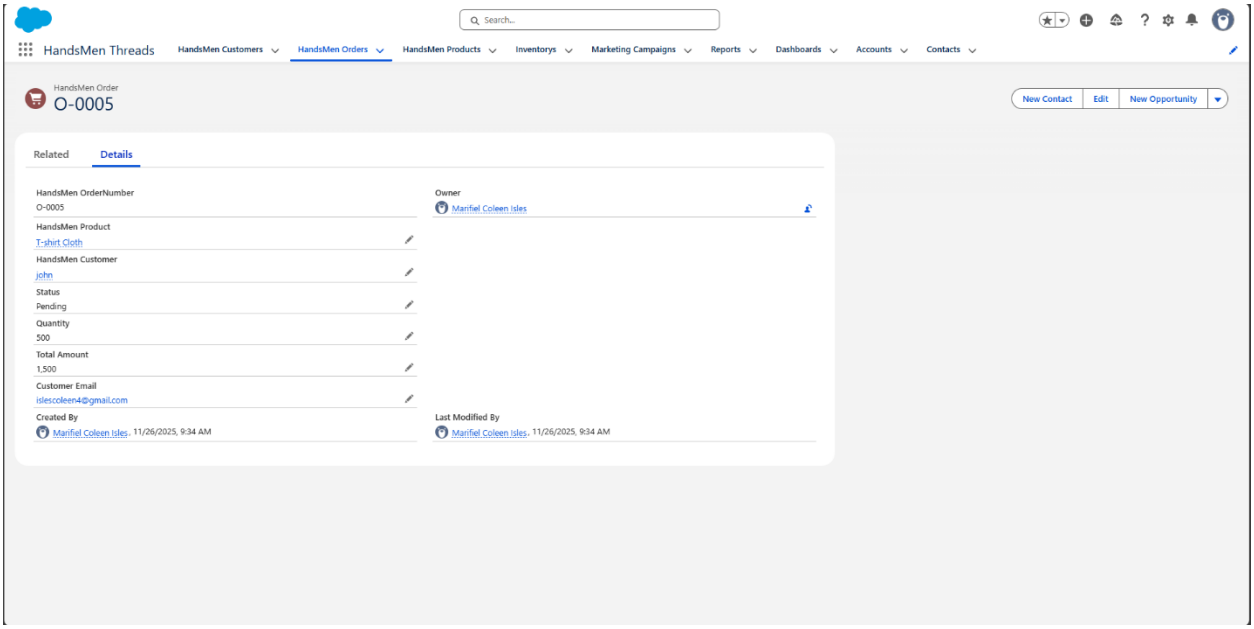
- **Lightning App Setup through App Manager**
  - A centralized, branded entry point is created for the users.
  - **HandsMen Threads App:** A custom Lightning App is configured via the App Manager to serve as the main workspace for Sales and Warehouse teams.
  - **Branding:** The App is branded with the HandsMen Threads logo and colors.
  - **Navigation:** The navigation bar is customized to include only the relevant items: Home, Customers, Orders, Products, Reports, and Dashboards, ensuring quick access to core functionalities.

Screenshot of the Handsmen Lightning App:



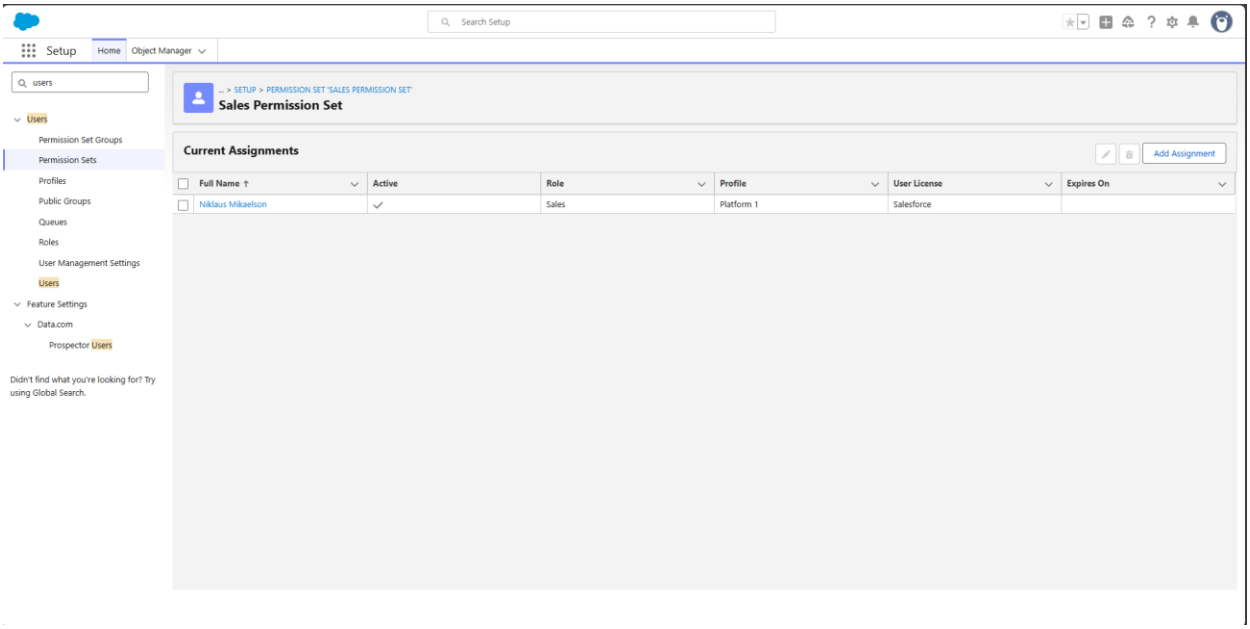
- **Page Layouts and Dynamic Forms**
  - Record details are optimized for clarity and reduced scrolling.
  - **Page Layouts:** Traditional Page Layouts are refined for non-LWC records, grouping related fields logically (e.g., quantity, name, etc.).
  - **Dynamic Forms:** For key custom objects like Order and Loyalty Status, Dynamic Forms are implemented (using Lightning Record Pages) to provide a more tailored and flexible user experience.
  - **Conditional Visibility:** Fields on the Order record page are made conditionally visible, only appearing once the order status changes to 'Shipped', reducing clutter for users handling new orders.

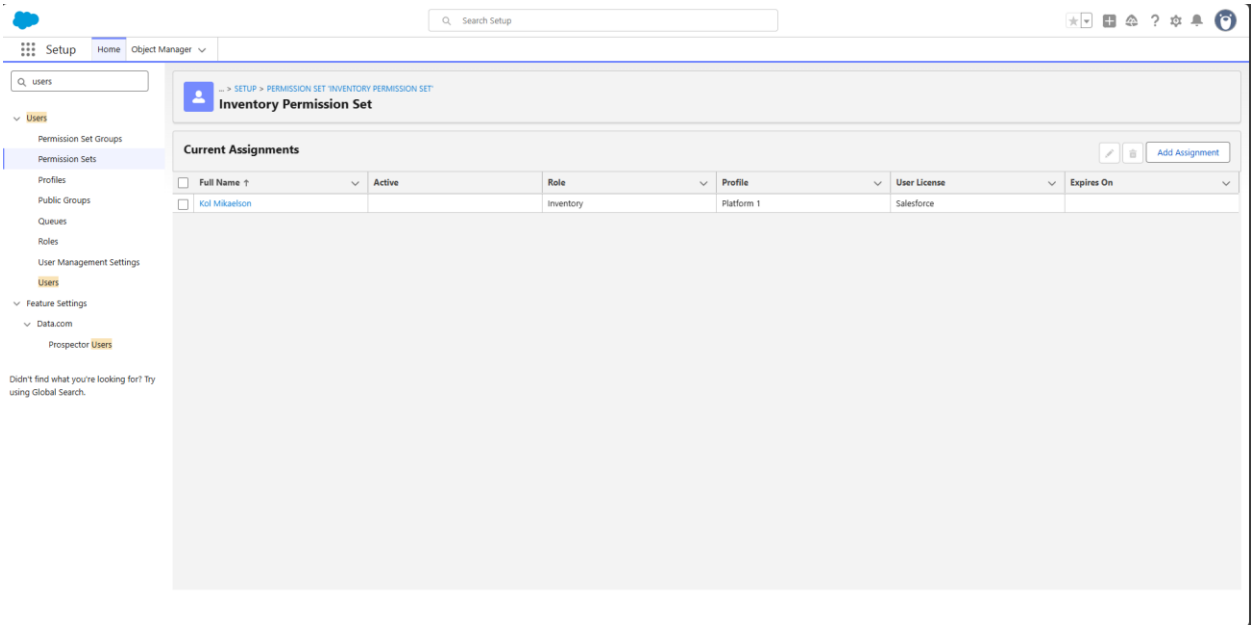
Screenshot of the Order Object Form:

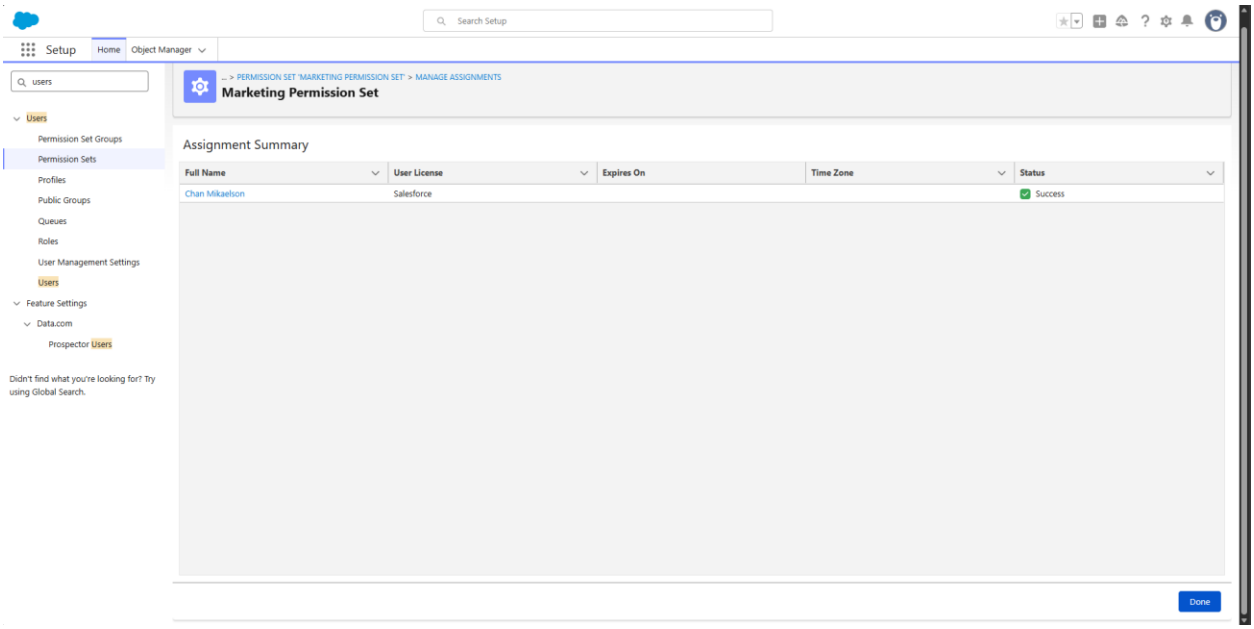


- **User Management**
  - User access and profiles are finalized based on the Security Model.
  - **Profile/Permission Set Review:** The security model designed in Phase 1 is fully configured, creating distinct Profiles and supplementing them with Permission Sets where necessary.
  - **User Provisioning:** Dummy users representing the different roles (Sales, Warehouse, Admin) are created in the Sandbox environment to thoroughly test all data visibility and access rules before deployment.

Screenshot of the Permission Set:







- **Lightning Pages**
  - The display and organization of information on record pages are optimized.
  - **Custom Record Pages:** Custom Lightning Record Pages are built for the Customer and Product objects using the Lightning App Builder.
  - **Component Placement:** The pages are structured logically using standard components. For instance, the Activity Timeline is prominently placed on the Customer page for quick access to communication history, and a related list component for Loyalty Status is placed near the top.

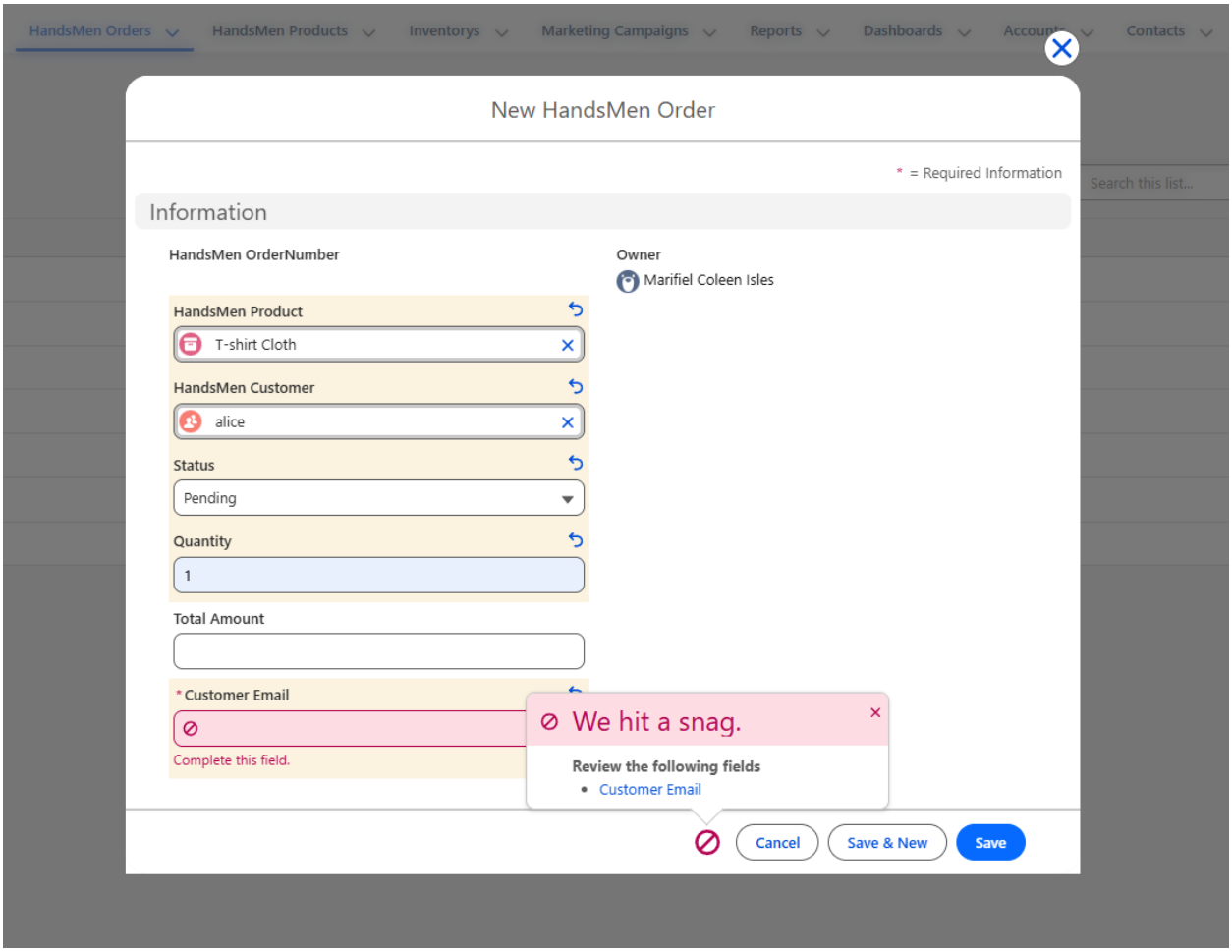
Screenshot of the Arrangement of Tabs in the Lightning App Builder:

Data Migration, Testing & Security Hardening

This final phase ensures the system is thoroughly validated, secure, and ready for production use, concluding with the preparation for user adoption and go-live.

- **Data Integrity**
    - This test case validates a critical Validation Rule on the **Order** object, ensuring that the necessary prerequisite data for customer communication—specifically the **Email Address**—is present on the related **Customer (Contact)** record.
    - **Why it's necessary:** The HandsMen Threads project includes the automated process of sending Order Confirmations post-sale. If the associated Contact record lacks an email address, this crucial automation will fail, resulting in a poor customer experience and operational error. This Validation Rule prevents the saving or confirmation of an Order until this fundamental data quality requirement is met.
    - **Test Scenario Detail:**
        - **Setup:** A Contact record is intentionally created or located where the standard **Email** field is empty or blank.
        - **Action:** A user attempts to create a new **Order** record and links it to this email-less Contact, or attempts to update an existing Order linked to the Contact to a confirmed status (e.g., 'Processing' or 'Confirmed').
        - **Expected Result (Success):** The system must immediately display the custom validation error message, successfully preventing the Order from being saved or advanced in status. This proves the system is enforcing the data quality requirement necessary for the automation to function later.

Screenshot of the error when incomplete fields are left filled:



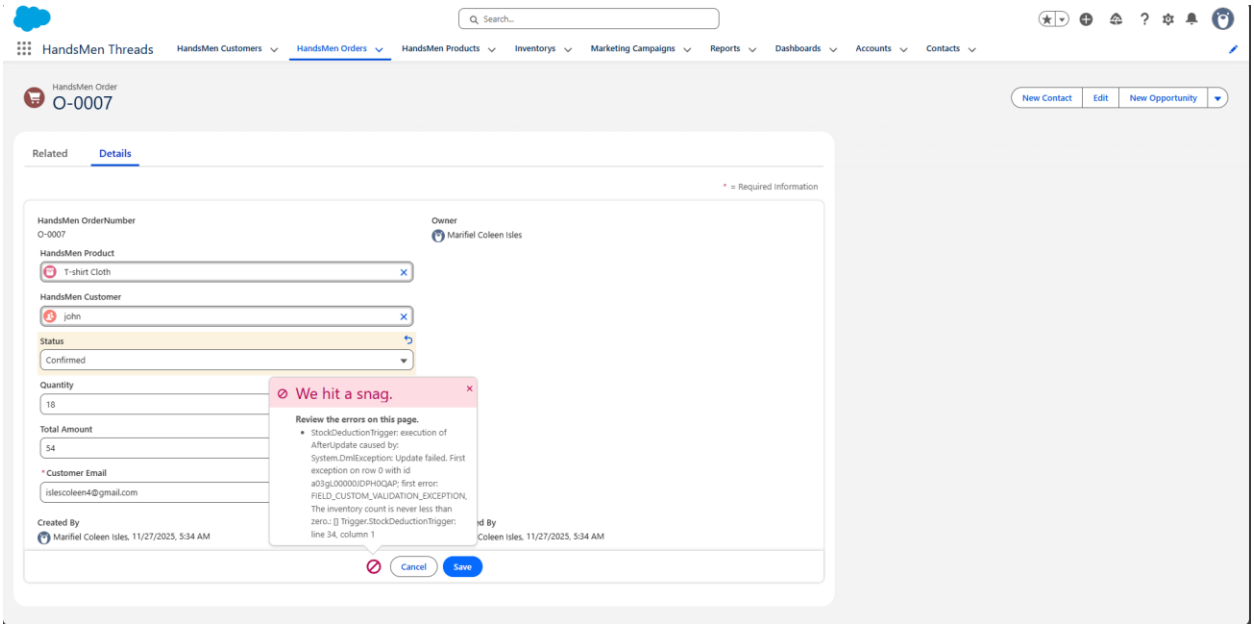
Data Quality and Auditing Features

- The following features are configured to maintain high data quality and prevent operational errors like overselling.
- **Field History Tracking:** Enabled on key fields within the **Product** and **Order** objects for auditing changes.
- **Duplicate Rules/Matching Rules:** Implemented on the **Customer** (Contact) object to prevent the creation of duplicate customer records.
- **Over-Ordering Prevention Rule (New Feature Focus):** A critical Validation Rule is enforced on the **Order Item** object. This rule prevents

users from saving an order detail line where the requested Quantity__c is **greater than** the current Stock_Level__c on the related Product record.

- **Duplicate Rules:** Implemented on the **Customer** (Contact) object to prevent the creation of duplicate customer records.
- Matching Rules: The Duplicate Rule is supported by a Matching Rule that identifies potential duplicates based on fuzzy logic (e.g., matching the same First Name + Last Name + closely matching Email Address). This ensures data integrity upon manual entry or during lower-volume data loads.

Screenshot of the error when changing the order quantity to more than the available stock:



- **Flow: Automated Order Confirmation**
  - This test case validates the successful execution of the **Record-Triggered Flow** designed to automate customer communication. The core goal is to verify that when a sales agent confirms an order, the system automatically triggers an email notification to the customer without any manual intervention.

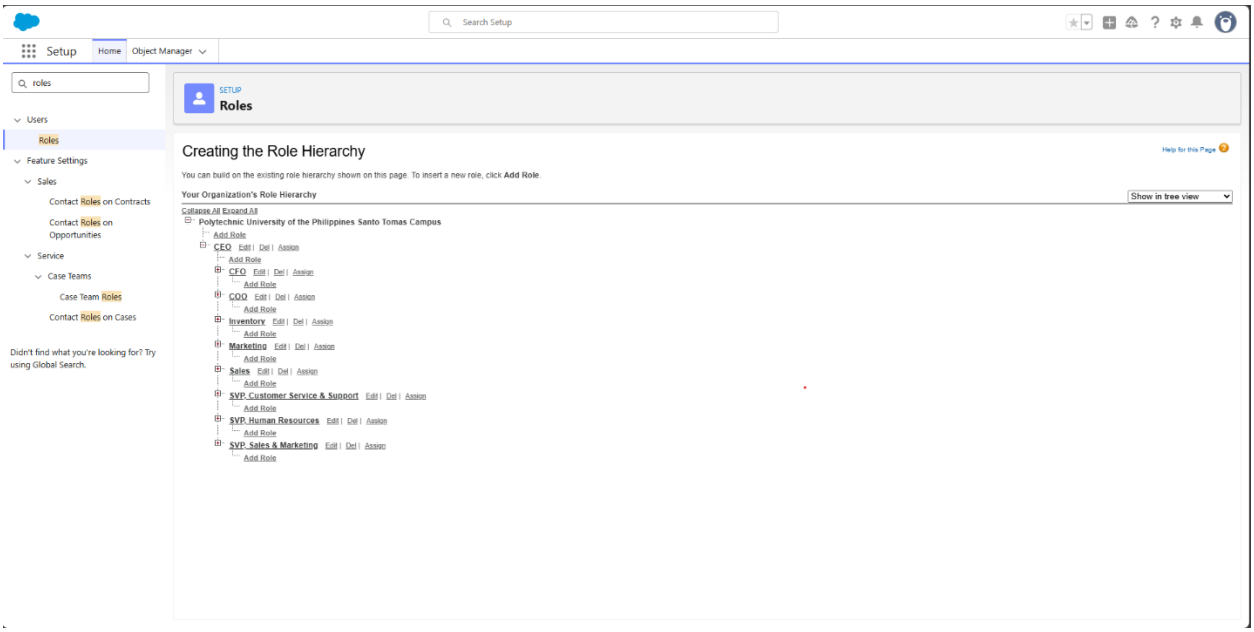
Screenshot of the email with the message “Order Confirmed”:

- **Apex Trigger: Proactive Stock Alert**
    - This test case validates the custom Apex Trigger logic that manages proactive inventory monitoring. The objective is to confirm that the trigger correctly fires and initiates communication *only* when a product's stock level crosses the critical threshold (below 5 units), ensuring the warehouse team is alerted instantly.

Screenshot of the email with the message “Low Stock Alert”:

- **Finalizing Security Model: Profiles, Roles, and Sharing**
    - **Profiles and Permission Sets:** All user Profiles (and necessary Permission Sets are finalized and deployed.
    - **Roles and Role Hierarchy:** The Role Hierarchy is established to support data visibility and reporting. This allows managers to view records owned by their subordinates.

Screenshot of the Role Hierarchy:



- **Creation of Test Classes**
    - All custom Apex code must be supported by automated testing.
    - Test Class Development: Dedicated Apex Test Classes are created for the Stock Alert Trigger Batch Apex class. These classes simulate realistic scenarios (e.g., triggering a stock level drop below 5 units, or creating 200 orders for batch processing).
    - Code Coverage: The test methods are designed to achieve a minimum of 75% code coverage for the custom Apex, meeting Salesforce's deployment requirement.
    - **Screenshot can be found in the Apex Class and Triggers Section above**

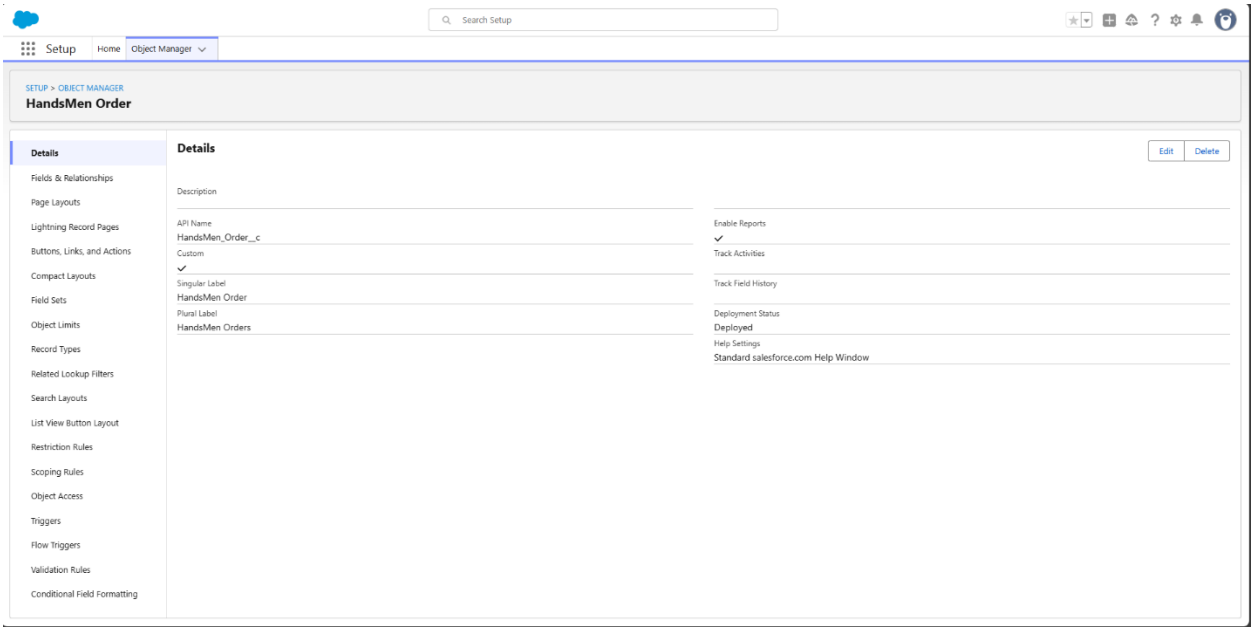
Deployment, Documentation & Maintenance

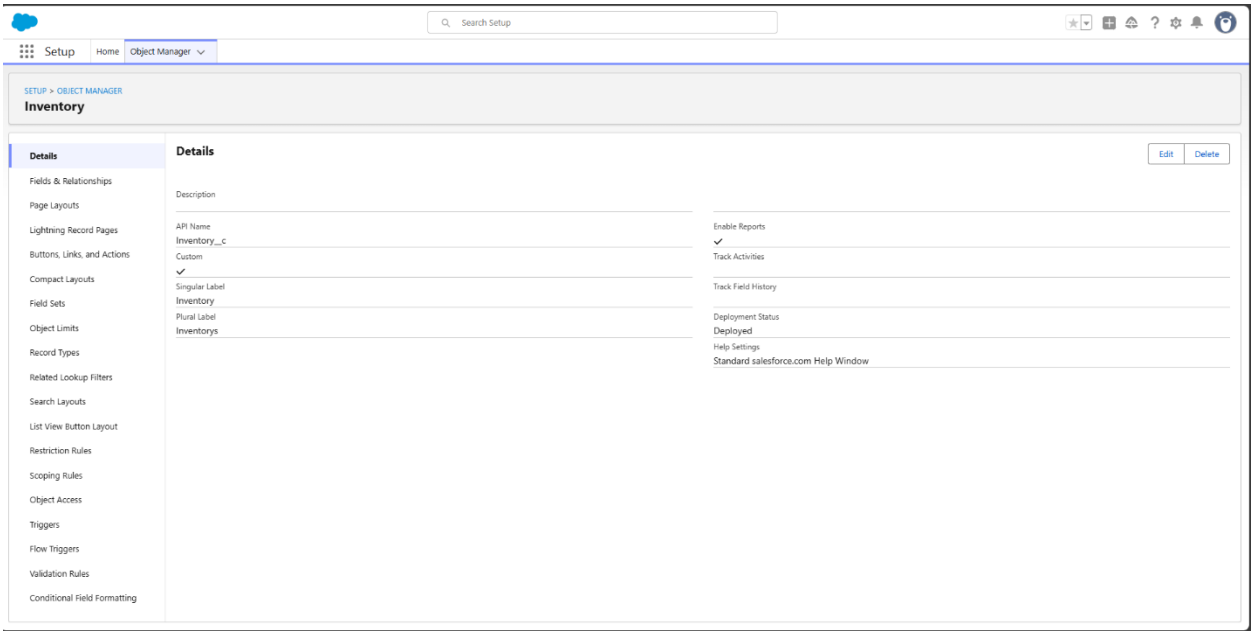
This final phase covers the controlled release of the finished application to the production environment, the establishment of support procedures, and the formal handover of the documentation.

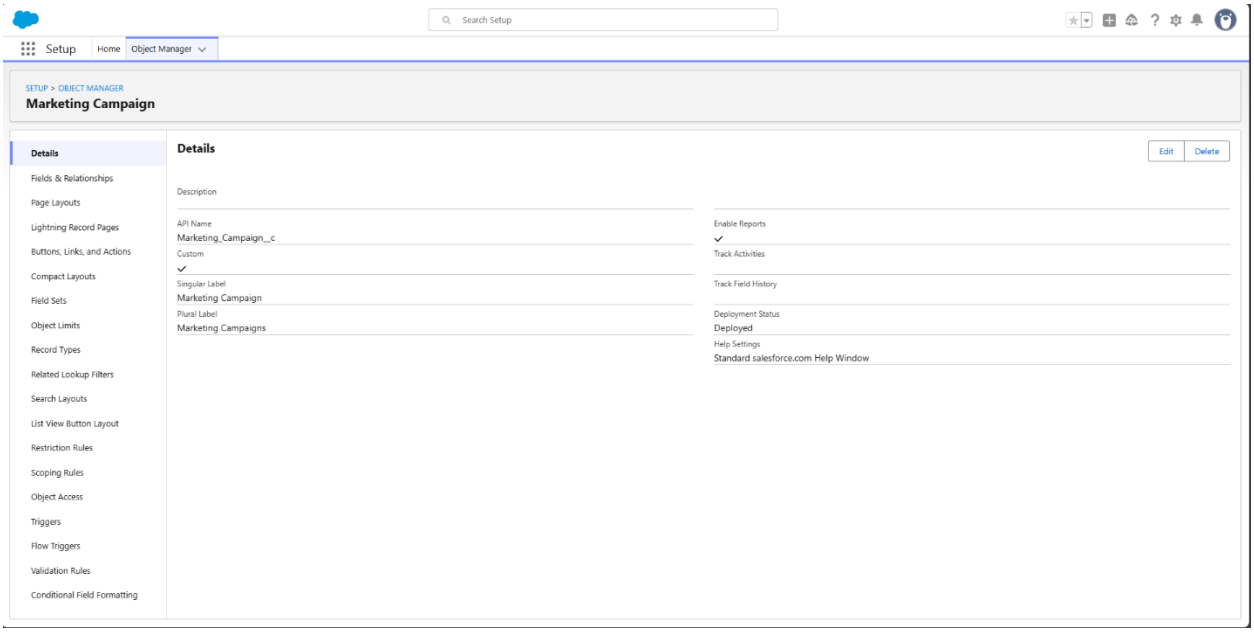
- **Deployment Strategy**
  - The deployment strategy for the HandsMen Threads Salesforce solution will utilize a phased, controlled approach leveraging Salesforce Change Sets.
  - **Tools: Change Sets** will be the primary tool for migrating metadata (objects, fields, validation rules, flows, profiles, reports, and custom code) between related sandboxes and the final Production environment.
  - **Validation and Testing in Production:** Before the final deployment, all components will be validated in the Production environment during off-peak hours. This step checks for any dependency errors or configuration issues without actually deploying the changes.
  - **Code Coverage Enforcement:** The deployment process will be executed only when the total Apex code coverage is at or above 75% in the destination environment.
  - **Go/No-Go Decision:** A final Go/No-Go meeting with stakeholders and the IT team will confirm that all test cases (Phase 4) have passed in the UAT environment and the Production validation was successful before the final deployment is executed.
- **System Maintenance and Monitoring**
  - Post-deployment, a plan is essential to ensure the system remains reliable, performs optimally, and continues to meet business needs.
  - **Scheduled Maintenance: The Batch Apex job (BulkOrderProcessor) will be monitored daily via the Scheduled Jobs and Apex Jobs pages in Setup to ensure it executes successfully at midnight.**
  - **Performance Monitoring: The Debug Logs will be utilized to occasionally sample the execution times of the complex Apex Trigger (Proactive Stock Alert) and the Order Confirmation Flow to identify any potential performance degradation over time.**

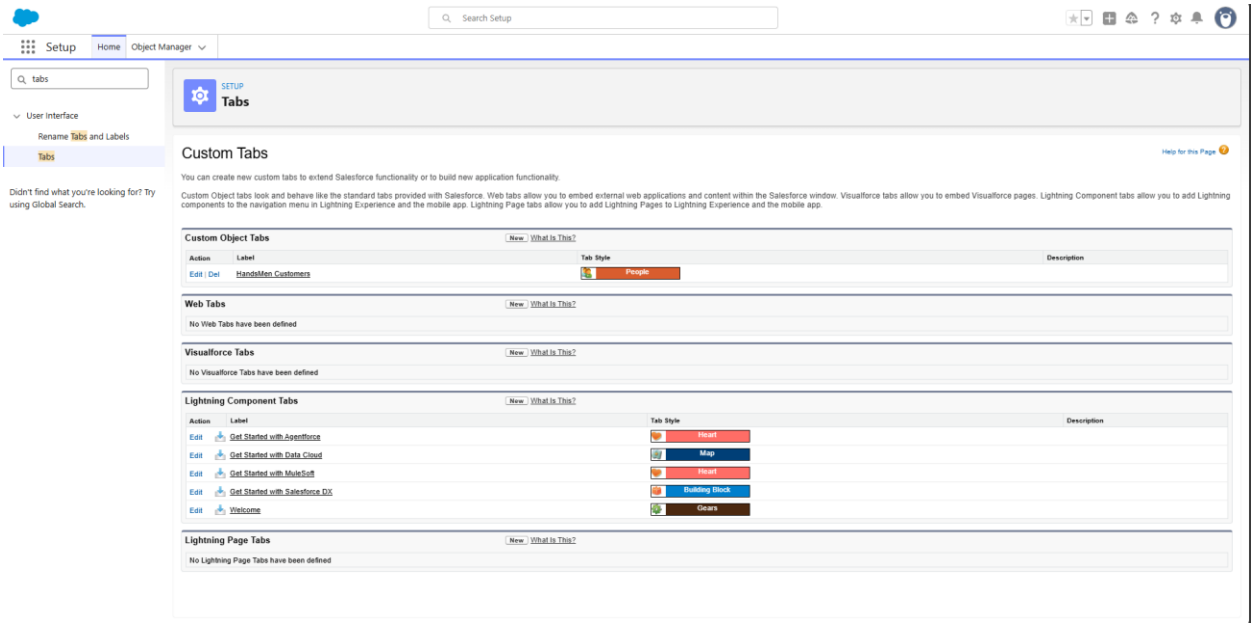
- **Release Management:** A designated System Administrator will be responsible for applying Salesforce's seasonal releases (Spring, Summer, Winter) in a sandbox first, testing critical functionality (like the Order Flow and Batch Job), and ensuring compatibility before the updates are pushed to Production.
- **User Feedback Loop:** A dedicated channel (e.g., a "Salesforce Feedback" Chatter Group or a custom Case queue) will be established for end-users to submit bug reports and enhancement requests for future sprints.

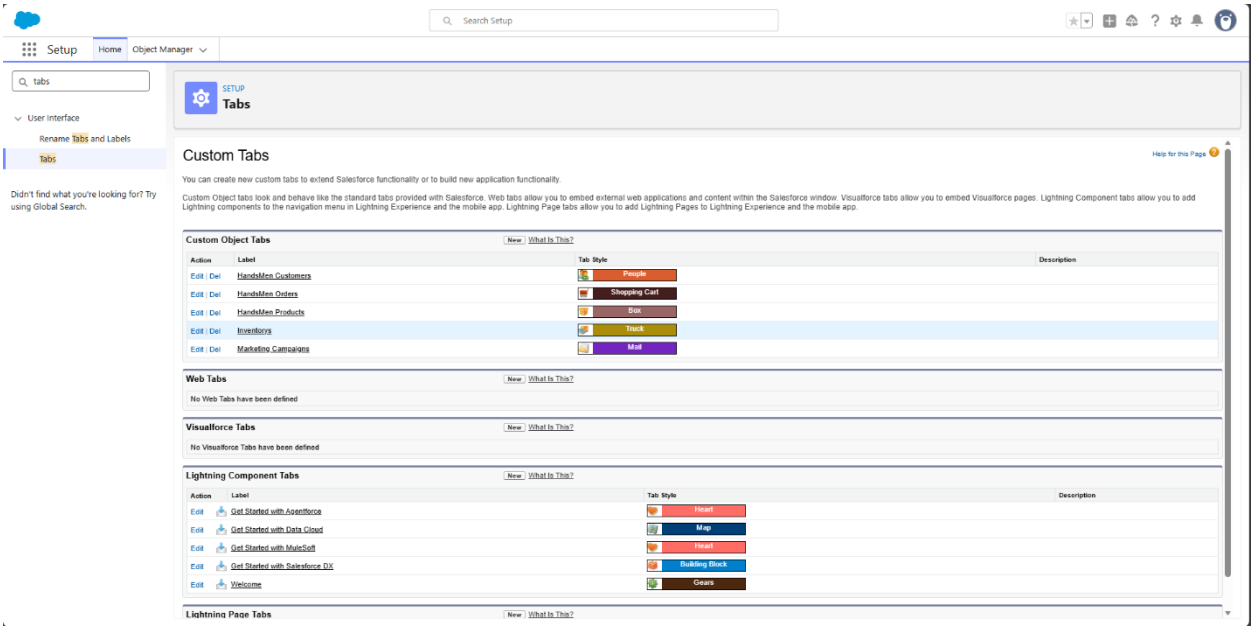
Other Screenshots:




Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
Inventory
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Restriction Rules
Scoping Rules
Object Access
Triggers
Flow Triggers
Validation Rules
Conditional Field Formatting
Details
Edit
Delete
Description
API Name
Inventory__c
Custom
Singular Label
Inventory
Plural Label
Inventorys
Enable Reports
Track Activities
Track Field History
Deployment Status
Deployed
Help Settings
Standard salesforce.com Help Window


Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
Marketing Campaign
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
List View Button Layout
Restriction Rules
Scoping Rules
Object Access
Triggers
Flow Triggers
Validation Rules
Conditional Field Formatting
Details
Edit
Delete
Description
API Name
Marketing_Campaign__c
Custom
Singular Label
Marketing Campaign
Plural Label
Marketing Campaigns
Enable Reports
Track Activities
Track Field History
Deployment Status
Deployed
Help Settings
Standard salesforce.com Help Window


Search Setup
Setup
Home
Object Manager
tabs
User Interface
Rename Tabs and Labels
Tabs
Didn't find what you're looking for? Try using Global Search.
SETUP
Tabs
Custom Tabs
Help for this Page
You can create new custom tabs to extend Salesforce functionality or to build new application functionality.
Custom Object tabs look and behave like the standard tabs provided with Salesforce. Web tabs allow you to embed external web applications and content within the Salesforce window. Visualforce tabs allow you to embed Visualforce pages. Lightning Component tabs allow you to add Lightning components to the navigation menu in Lightning Experience and the mobile app. Lightning Page tabs allow you to add Lightning Pages to Lightning Experience and the mobile app.
Custom Object Tabs
New | What Is This?
Action
Label
Tab Style
Description
Edit | Del
HandsMen Customers
People
Web Tabs
New | What Is This?
No Web Tabs have been defined
Visualforce Tabs
New | What Is This?
No Visualforce Tabs have been defined
Lightning Component Tabs
New | What Is This?
Action
Label
Tab Style
Description
Edit
Get Started with Agentforce
Heart
Edit
Get Started with Data Cloud
Map
Edit
Get Started with MuleSoft
Heart
Edit
Get Started with Salesforce DX
Building Block
Edit
Welcome
Gears
Lightning Page Tabs
New | What Is This?
No Lightning Page Tabs have been defined


Search Setup
Setup
Home
Object Manager
tabs
User Interface
Rename Tabs and Labels
Tabs
Didn't find what you're looking for? Try using Global Search.
SETUP
Tabs
Custom Tabs
Help for this Page
You can create new custom tabs to extend Salesforce functionality or to build new application functionality.
Custom Object tabs look and behave like the standard tabs provided with Salesforce. Web tabs allow you to embed external web applications and content within the Salesforce window. Visualforce tabs allow you to embed Visualforce pages. Lightning Component tabs allow you to add Lightning components to the navigation menu in Lightning Experience and the mobile app. Lightning Page tabs allow you to add Lightning Pages to Lightning Experience and the mobile app.
Custom Object Tabs
New | What Is This?
Action
Label
Tab Style
Description
Edit | Del
HandsMen Customers
People
Edit | Del
HandsMen Orders
Shopping Cart
Edit | Del
HandsMen Products
Box
Edit | Del
Inventorys
Truck
Edit | Del
Marketing Campaigns
Mail
Web Tabs
New | What Is This?
No Web Tabs have been defined
Visualforce Tabs
New | What Is This?
No Visualforce Tabs have been defined
Lightning Component Tabs
New | What Is This?
Action
Label
Tab Style
Description
Edit
Get Started with Agentforce
Heart
Edit
Get Started with Data Cloud
Map
Edit
Get Started with MuleSoft
Heart
Edit
Get Started with Salesforce DX
Building Block
Edit
Welcome
Gears
Lightning Page Tabs
New | What Is This?

Conclusion

The HandsMen Threads capstone project successfully delivered a robust, customized Salesforce CRM solution that fundamentally digitized the brand's approach to sales, customer loyalty, and inventory management.

By strategically implementing a blend of Apex programmatic automation and Salesforce Flow declarative tools, the core objectives were fully realized: achieving a dynamic loyalty program, enabling proactive stock alerting, and establishing reliable automated customer engagement. This implementation drastically moved the organization away from error-prone, manual processes and establishes a scalable, centralized digital backbone crucial for supporting HandsMen Threads' growth and future innovations in personalized fashion retail.

Future Scope and Enhancements

The current Salesforce implementation provides a stable and scalable foundation. The following future enhancements are recommended to further optimize the platform:

- **Commerce Cloud Integration:** Integrate the CRM with Commerce Cloud to manage the e-commerce storefront, providing a single source of truth for all product and inventory data.
- **AI-Powered Recommendations:** Leverage Salesforce Einstein capabilities to analyze customer purchase history and style preferences for intelligent, personalized product and style recommendations on the website and in marketing emails.
- **Service Cloud Implementation:** Implement Service Cloud to manage all post-sale interactions, including returns, alterations, and customer inquiries, providing a 360-degree view of the customer.
- **Customer Community Portal:** Deploy a Customer Experience Cloud portal where customers can independently view their loyalty status, track order history, and manage their personal style profile.