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

---

Table of Contents

---

1. Abstract

This report details a Salesforce implementation project undertaken for HandsMen Threads, a dynamic organization in the fashion industry. The primary goal of this initiative was to revolutionize the company's data management practices and enhance customer relationship management. The project centered on building a robust, custom data model within Salesforce to store all pertinent business data, including customer, product, order, and inventory information. A significant emphasis was placed on maintaining high data integrity directly from the user interface to ensure accuracy and consistency, which are critical for informed decision-making.

Key deliverables included the automation of several core business processes. An automated order confirmation system was developed to immediately update customers via email, fostering better engagement. A dynamic loyalty program was implemented to update customer statuses based on purchase history, enabling personalized marketing and rewards. To prevent stockouts, a proactive alert system was configured to notify the warehouse team when inventory levels for any product drop below a specified threshold. Furthermore, a scheduled batch process was created to handle bulk order updates nightly, ensuring financial records and inventory levels are consistently accurate for the start of each business day. This project showcases proficiency in Salesforce data modeling, automation using Flows and Apex, and data quality management.

2. Objective

The main objective of the HandsMen Threads project was to design and deploy a comprehensive Salesforce solution to streamline operations, improve data accuracy, and elevate the customer experience. The fashion industry is fast-paced, and the ability to manage data efficiently and interact with customers effectively is a significant competitive advantage. Prior to this implementation, the organization faced challenges with disparate data sources, manual processes, and a lack of a unified view of the customer. This project aimed to address these challenges by leveraging the full power of the Salesforce platform.

The core objectives were broken down into four key process improvements:

1. **Enhance Customer Communication and Engagement:** In the competitive fashion market, post-purchase communication is crucial for building brand loyalty. The project aimed to implement an automated system for sending order confirmation emails. As soon as an order's status is updated to "Confirmed" in Salesforce, the customer receives a personalized email. This not only keeps the customer informed but also reinforces their purchase decision and strengthens their relationship with the HandsMen Threads brand.

2. **Foster Customer Loyalty through Personalization:** The project sought to create a dynamic, data-driven loyalty program. By tracking customer purchase history (`Total_Purchases__c`), the system automatically assigns a loyalty status—Bronze, Silver, or Gold. This segmentation allows HandsMen Threads to tailor marketing campaigns, offer personalized rewards, and make customers feel

valued. This automated process replaces a manual, and often inconsistent, method of tracking customer loyalty, ensuring fairness and timely status updates.

3. **Prevent Stockouts and Optimize Inventory Management:** Running out of popular items can lead to lost sales and customer dissatisfaction. A critical objective was to create a proactive stock alert system. By monitoring the `Stock_Quantity__c` field on inventory records, the system automatically triggers an email alert to the warehouse team whenever stock for an item drops below five units. This ensures that the procurement and restocking processes are initiated in a timely manner, minimizing the risk of stockouts and optimizing inventory turnover.

4. **Ensure Data Accuracy through Scheduled Processes:** To maintain accurate financial records and inventory levels, the project included the development of a scheduled nightly batch job. This asynchronous process runs at midnight to handle bulk order updates, adjusting inventory counts and flagging records for financial reconciliation. This ensures that when the business day begins, all stakeholders are working with up-to-date, accurate data, which is fundamental for reliable reporting and operational planning.

3. Technology Description

The implementation for HandsMen Threads leveraged a suite of core Salesforce technologies to build a tailored and efficient CRM solution.

Salesforce

Salesforce is a cloud-based Customer Relationship Management (CRM) platform that provides a single source of truth for all business data. It allows organizations to manage customer data, track interactions, and automate business processes. For this project, a Salesforce Developer Edition org was used as the foundation for building the custom solution.

Data Model: Custom Objects, Apps, and Tabs

- **Custom Objects:** To store business-specific data, several custom objects were created. These are analogous to tables in a database. The objects created were `HandsMen Customer`, `HandsMen Product`, `HandsMen Order`, `Inventory`, and `Marketing Campaign`, each with its own set of custom fields and relationships (Lookup and Master-Detail) to create a relational data model.
- **Custom Apps:** A custom Lightning App named "HandsMen Threads" was built to provide users with a tailored user interface. This app bundles all the relevant custom objects, tabs, reports, and dashboards into a single, cohesive navigation experience, improving user efficiency.
- **Tabs:** Custom tabs were created for each custom object, such as "HandsMen Customer" and "HandsMen Product." These tabs allow users to easily access, view, and create records for that object directly from the app's navigation bar.

Security and Access: Profiles, Roles, and Permission Sets

- **Profiles:** Profiles control what users can see and do within the application (Object-Level Security and Field-Level Security). A custom profile named "Platform 1" was created by cloning the standard user profile. It was then configured to grant specific create, read, update, and delete (CRUD) permissions on the new custom objects.
- **Roles:** The Role Hierarchy was established to control the visibility of records. Roles like Sales, Inventory, and Marketing were created under the CEO role. This ensures that users can only see their own records and the records of the users below them in the hierarchy, protecting data confidentiality.
- **Permission Sets:** To grant additional, specific permissions to users without changing their profile, a permission set named "Permission_Platform_1" was created. This was used to grant CRUD permissions on the Customer and Order objects to select users, providing a flexible and scalable security model.

Data Integrity: Validation Rules

Validation rules are crucial for ensuring the quality and accuracy of data entered by users. Several validation rules were implemented:

- On the `HandsMen Order` object, a rule ensures the `Total_Amount__c` is greater than zero.
- On the `Inventory` object, a rule prevents `Stock_Quantity__c` from being less than or equal to zero.
- On the `HandsMen Customer` object, a rule was created to ensure the `Email` field contains "@gmail.com" for data consistency.

Communication: Email Templates and Email Alerts

- **Email Templates:** HTML email templates were designed for standardized communications. Templates like "Order_Confirmation_Email," "Low Stock Alert," and "Loyalty Program Email" were created with merge fields to personalize the content with data from Salesforce records (e.g., customer name, order number).
- **Email Alerts:** Email Alerts are the mechanism that sends an email template to specified recipients. An alert was configured to send the "Order_Confirmation_Email" to the related customer when an order is confirmed.

Automation: Flows and Apex

- **Flows:** Salesforce Flow is a powerful tool for declarative process automation. Three key flows were built:
    - **Record-Triggered Flow (Order Confirmation):** This flow runs when an `HandsMen Order` record is updated to have a status of "Confirmed" and automatically sends the order confirmation email alert.
    - **Record-Triggered Flow (Stock Alert):** This flow activates when an `Inventory` record is updated and the `Stock_Quantity__c` drops below five, sending an email alert to the inventory team.

    - **Schedule-Triggered Flow (Loyalty Status Update):** This flow runs daily to loop through all `HandsMen Customer` records. It uses a decision element to check the `Total_Purchases__c` and updates the `Loyalty_Status__c` field accordingly (Gold, Silver, or Bronze).
- **Apex and Apex Triggers:** For more complex business logic that cannot be handled by flows, Apex, Salesforce's proprietary programming language, was used.
    - An **Apex Trigger** (`OrderTrigger`) was written on the `HandsMen_Order__c` object to execute on `before insert` and `before update` events.
    - This trigger calls an **Apex Class** (`OrderTriggerHandler`) which contains the logic to validate the `Quantity__c` based on the order's `Status__c`, enforcing different quantity requirements for "Confirmed," "Pending," and "Rejection" statuses.
- **Asynchronous Apex (Batch Apex):** To process a large number of records without exceeding governor limits, a **Batch Apex** class (`InventoryBatchJob`) was developed. This class is designed to find all products with stock quantities below 10 and update their stock by a set amount. The `Schedulable` interface was also implemented, allowing this batch job to be scheduled to run at a specific time, such as daily at midnight.

4. Detailed Execution of Project Phases

The project was structured into four distinct phases to ensure a smooth and successful implementation.

Phase 1: Architecture & Planning

This initial phase focused on laying the groundwork for the entire project. Key activities included:

- **Defining the Data Model:** The custom objects (`HandsMen Customer, HandsMen Product, HandsMen Order, Inventory, Marketing Campaign`) were defined, along with their fields and data types.
- **Establishing Relationships:** The relationships between objects were mapped out, determining where to use Lookup vs. Master-Detail relationships to connect orders to customers and products, and inventory to products.
- **Designing Automation:** The logic for all automation was planned. This included defining the trigger criteria and actions for the record-triggered flows, the schedule for the loyalty status flow, and the business rules for the Apex trigger and batch job.
- **Creating Communication Assets:** The content and merge fields for the three email templates (Order Confirmation, Low Stock Alert, Loyalty Program) were designed.

Phase 2: Development

This was the core implementation phase where the planned architecture was built in the Salesforce org.

- **Object and Field Creation:** All custom objects and their respective fields (Email, Phone, Loyalty Status, SKU, Price, etc.) were created as per the design. This also included creating formula fields like `Full_Name__c`.
- **Implementation of Automation:**
    - The three Salesforce Flows (Order Confirmation, Stock Alert, Loyalty Status Update) were built and activated.
    - The `OrderTriggerHandler` Apex class and the `OrderTrigger` were coded in the Developer Console to enforce advanced validation logic.
- **Data Security Setup:**
    - The "Platform 1" profile was created and configured with the correct object permissions.
    - The role hierarchy (CEO, Sales, Inventory, Marketing) was built.
    - New users were created and assigned the appropriate profile and role.
    - The "Permission_Platform_1" permission set was created to grant additional access where needed.
- **Batch Job Development:** The `InventoryBatchJob` Apex class was developed to handle asynchronous inventory updates and scheduled to run daily.
- **Email Configuration:** The classic email templates and their corresponding email alerts were configured and made available for use by the automation tools.

Phase 3: Testing & QA

Before deploying the solution, it underwent rigorous testing to ensure it was functioning as expected and was free of bugs.

- **Unit Testing:** Each component was tested individually. This included creating records to test validation rules, triggering flows by updating records, and writing Apex test classes to verify the logic in the trigger and batch job.
- **End-to-End Testing:** The team performed testing with sample data that mimicked real-world scenarios. For example, creating a customer, placing an order, confirming it, and verifying that the confirmation email was sent and the loyalty status was updated correctly after the scheduled flow ran.
- **Performance and Security Checks:** The performance of flows and batch jobs was reviewed to ensure they were optimized. Security settings were double-checked to confirm that users with different profiles and roles could only access the data they were permitted to see.

Phase 4: Deployment & Training

The final phase involved moving the solution to the live production environment and preparing the end-users.

- **Deployment:** The completed and tested configuration was deployed from the development environment to the production org using Salesforce deployment tools (e.g., Change Sets).
- **User Training:** End-users from the sales, inventory, and marketing teams were trained on the new functionality. This included how to use the "HandsMen Threads" app, manage customer and order records, and understand the new automated processes.

- **Post-Go-Live Support:** After deployment, a support period was established to monitor the system, address any issues that arose, and provide assistance to users as they adapted to the new platform.

5. Project Explanation with Real-World Example

To illustrate how the implemented solution works in practice, let's follow a complete customer journey.

Scenario: A new customer, **Niklaus Mikaelson**, browses the HandsMen Threads website and places an order for a new shirt.

1. **Order Placement and Data Entry:**

    - A sales representative creates a new `HandsMen Customer` record for Niklaus Mikaelson, entering his name and email. The `Full_Name__c` formula field automatically populates to "Niklaus Mikaelson". His initial `Loyalty_Status__c` is Bronze.
    - Next, a new `HandsMen Order` record is created. The system auto-generates an `OrderNumber` (e.g., O-0023). The order is linked to Niklaus's customer record and the specific `HandsMen Product` record. The order's initial `Status` is "Pending" and the `Quantity__c` is 300.
    - The `OrderTrigger` fires on insert. It calls the `OrderTriggerHandler` class, which checks the quantity. Since the status is "Pending" and the quantity (300) is greater than 200, the record saves successfully.

2. **Order Confirmation and Customer Communication:**

    - After payment is verified, a user updates the `Status` of the order from "Pending" to "Confirmed".
    - This update immediately triggers the **Order Confirmation Flow**.
    - The flow executes its action, which is to send an email. It uses the "Order Confirmation Email Alert" to send the "Order_Confirmation_Email" template to the email address on Niklaus's customer record. He receives an email that says, "Dear Niklaus Mikaelson, Your order #O-0023 has been confirmed!".

3. **Inventory Management:**

    - The confirmed order reduces the `Stock_Quantity__c` on the related `Inventory` record. Let's say the quantity drops from 12 to 9.
    - Later, another order for the same product is confirmed, and the `Stock_Quantity__c` drops to 4.
    - Because the quantity is now below 5, the **Stock Alert Flow** is triggered.
    - This flow sends an email alert using the "Low Stock Alert" template to the warehouse team, notifying them that the product is running low and needs to be restocked.

4. **Loyalty Program and Scheduled Updates:**

    - The value of Niklaus's purchase is added to his `Total_Purchases__c` field on his customer record.
    - That night, at midnight, the **Loyalty Status Update Flow** runs as scheduled.
    - The flow loops through all customer records, including Niklaus's. It checks his `Total_Purchases__c`. If his total purchases now exceed the threshold for the "Silver" tier, the flow automatically updates his `Loyalty_Status__c` field from "Bronze" to "Silver".

5. **Nightly Batch Processing:**

    - Also at midnight, the **InventoryBatchJob** runs. It queries for all products where the stock quantity is low (less than 10).
    - It finds the shirt that Niklaus ordered (which is now at a quantity of 4).
    - The batch job executes its logic, adding 50 units to the `Stock_Quantity__c`, simulating a daily restock of low-inventory items. The process is completed efficiently without impacting system performance during business hours.

This end-to-end example demonstrates how the custom objects, automation, and security features work together to create an efficient, intelligent, and responsive system for HandsMen Threads.

6. Screenshots

The following screenshots document key steps in the setup and configuration process of the HandsMen Threads Salesforce implementation.

Figure 1: Creating a New Custom Object This screenshot shows the setup menu for creating a new custom object, in this case, the `HandsMen Customer` object.

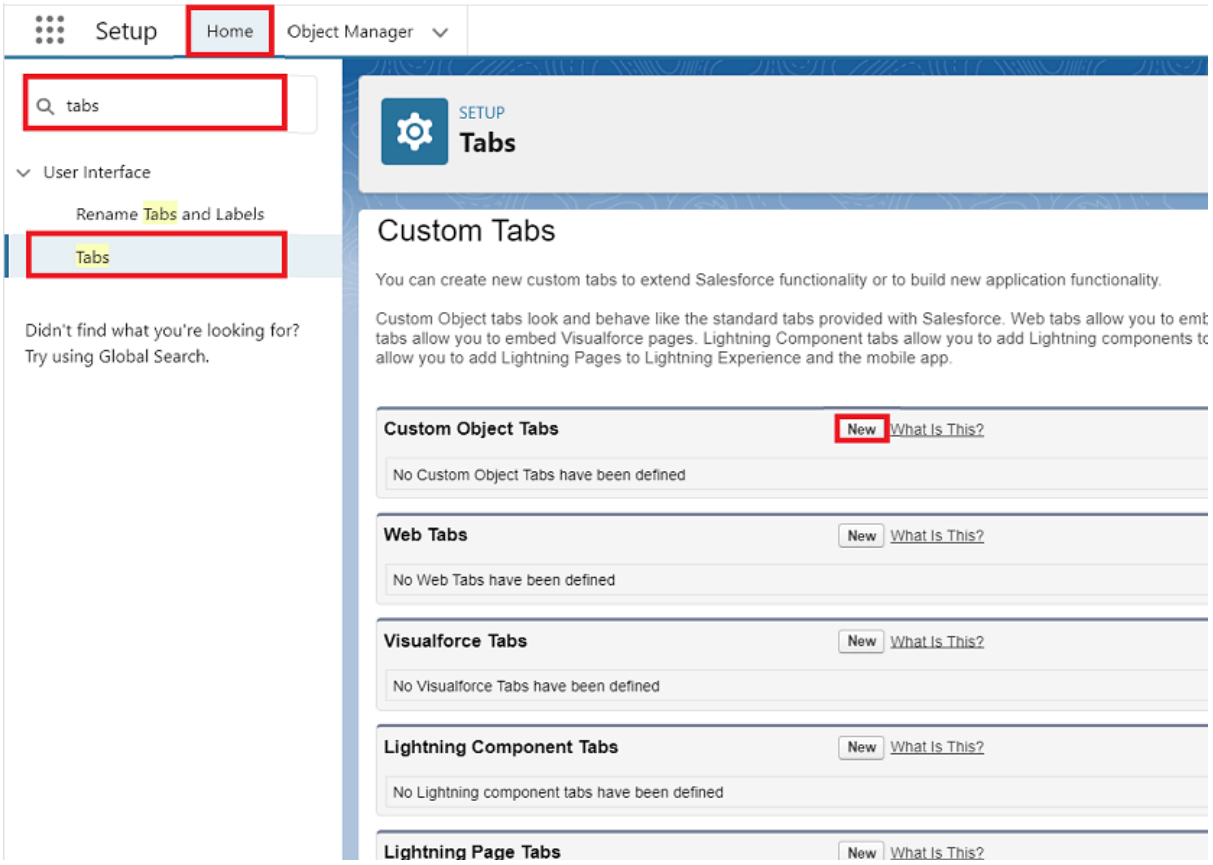


Figure 2: Creating a Custom Tab This image displays the configuration page for creating a new tab for the `HandsMen Customer` object, including the selection of the object and tab style.

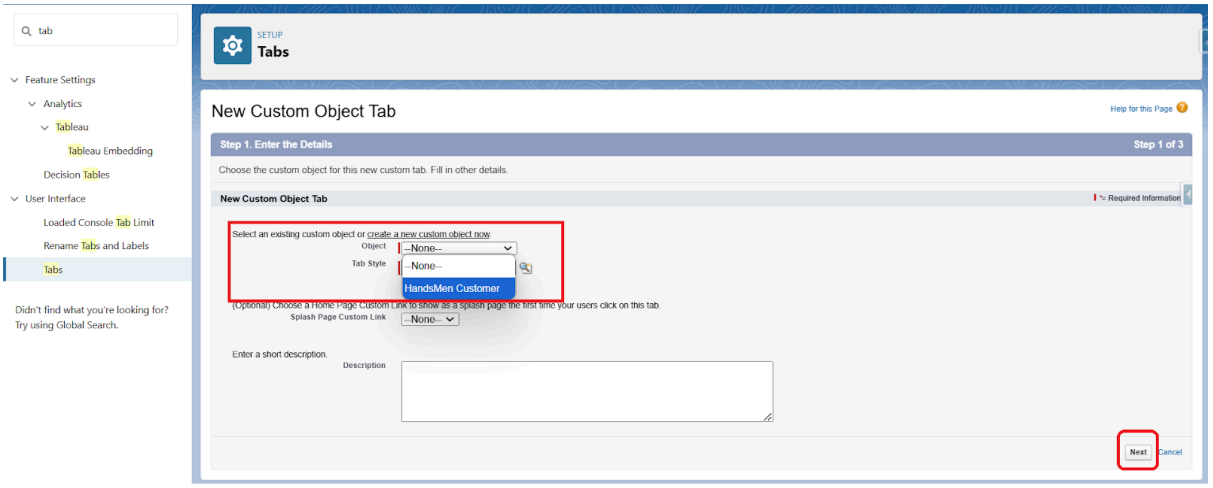


Figure 3: Creating a New Lightning App The App Manager screen where the new "HandsMen Threads" Lightning App is initiated.

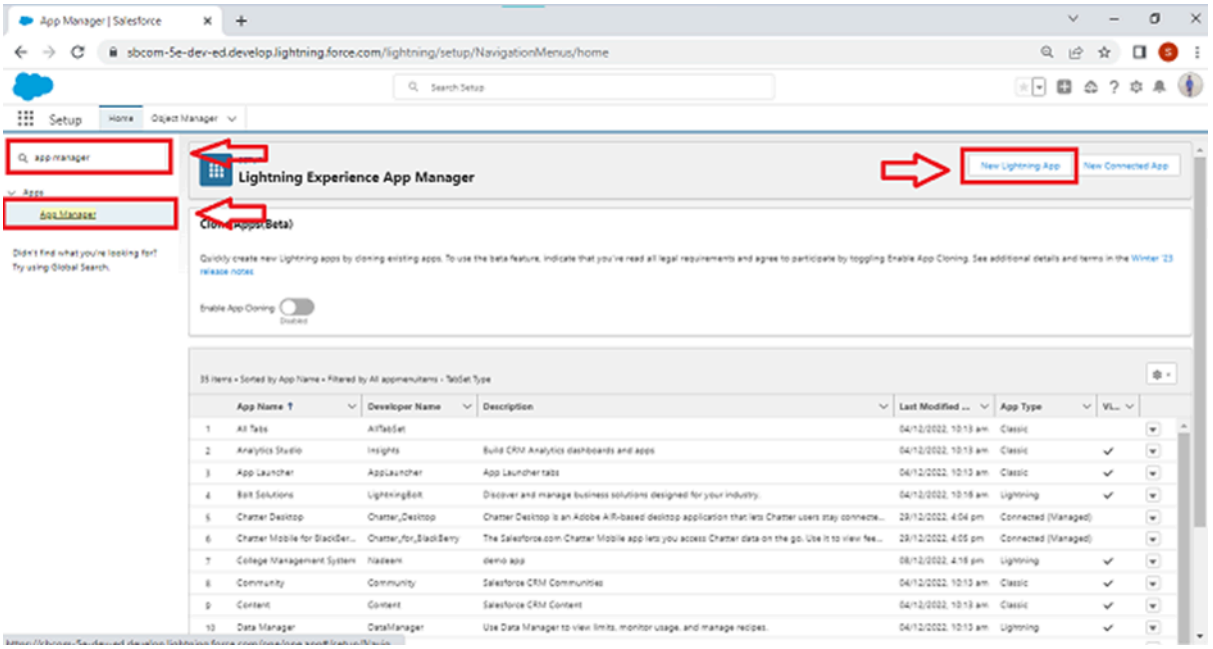


Figure 4: Adding Navigation Items to the App This shows the selection of tabs (e.g., HandsMen Customer, HandsMen Order) to be included in the navigation menu of the custom app.

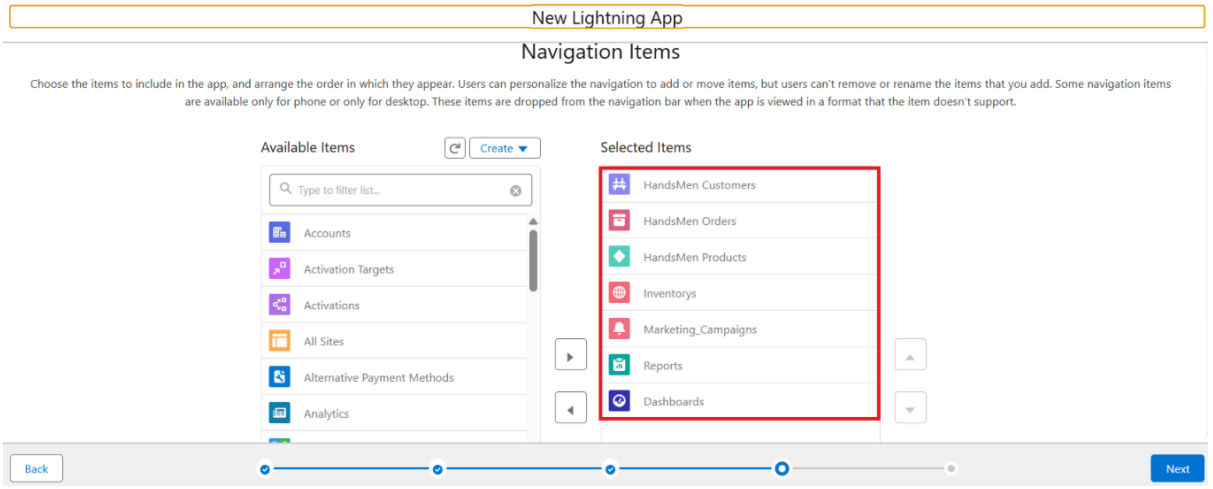


Figure 5: Assigning the App to User Profiles The final step of app creation, where the app is made visible to specific user profiles, in this case, the System Administrator.

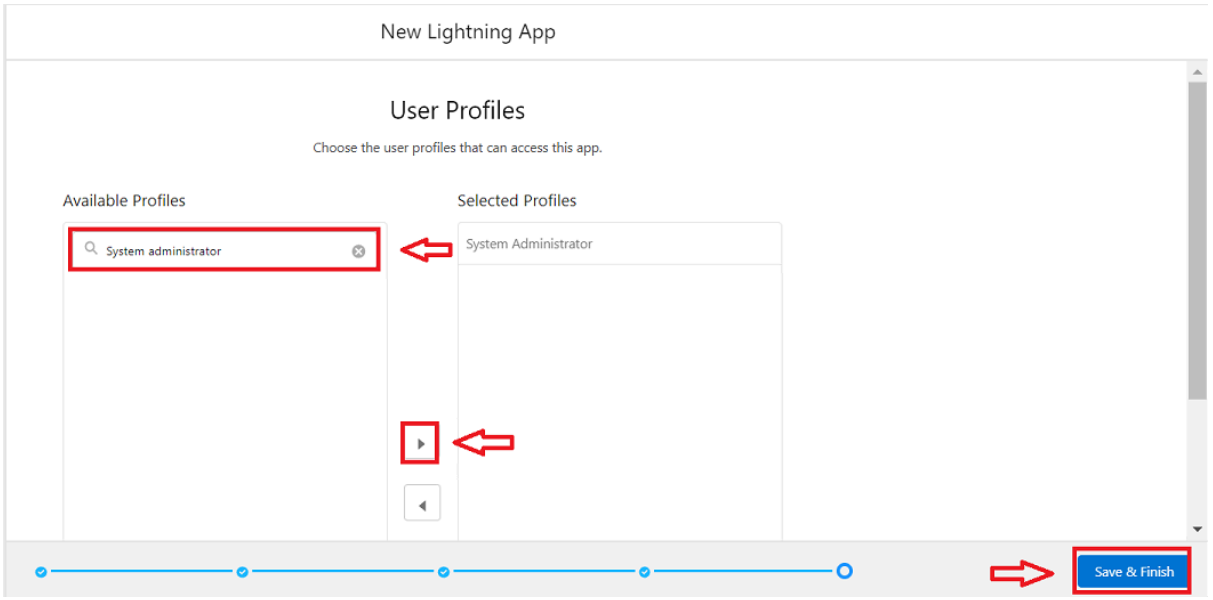


Figure 6: Creating a Picklist Field The setup for the "Loyalty Status" picklist field on the `HandsMen Customer` object, with Gold, Silver, and Bronze as the defined values.

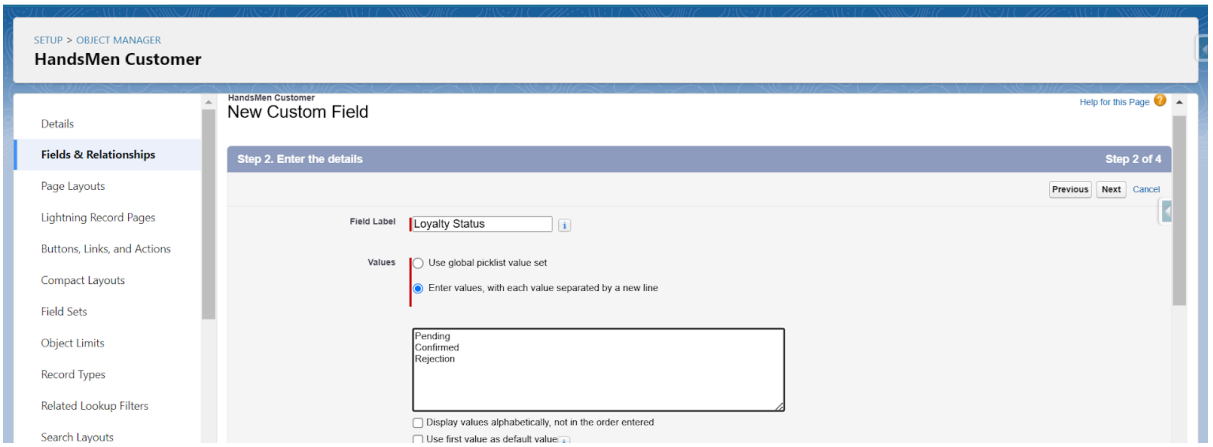


Figure 7: Creating a Validation Rule The configuration screen for a validation rule on the `Total_Amount__c` field, ensuring the value entered is greater than zero.

```
app = Flask(__name__)
```

Figure 8: Record-Triggered Flow Configuration The initial setup for the Order Confirmation flow, triggered when an `Order__c` record is updated.

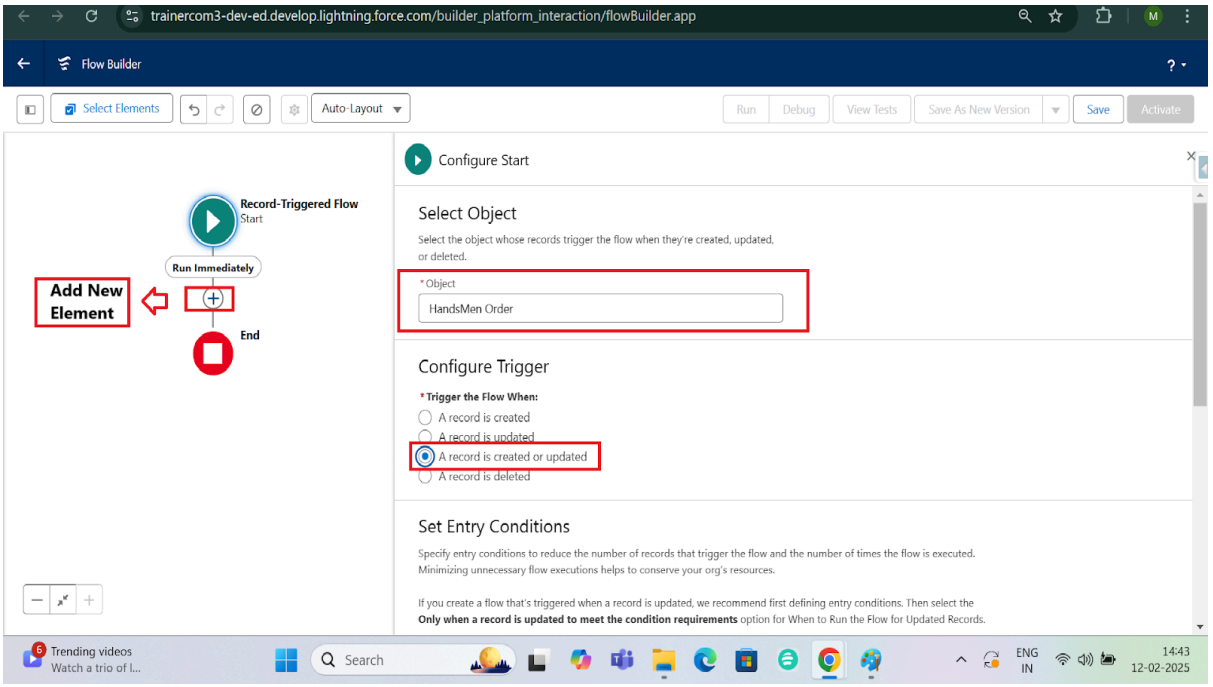


Figure 9: Adding an Action to a Flow The "Send Email Alert" action being added to the flow, with the Record ID correctly mapped.

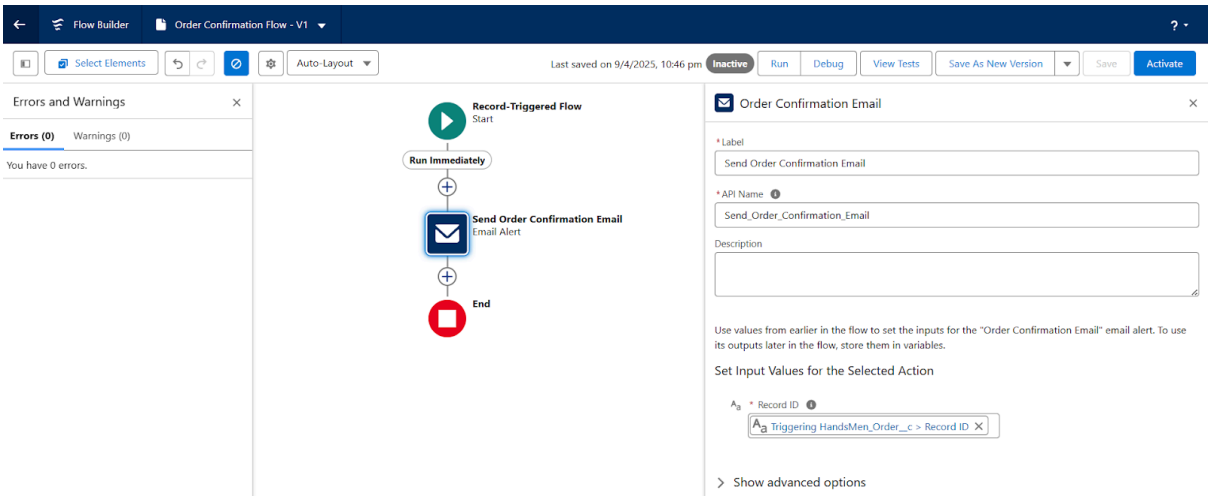
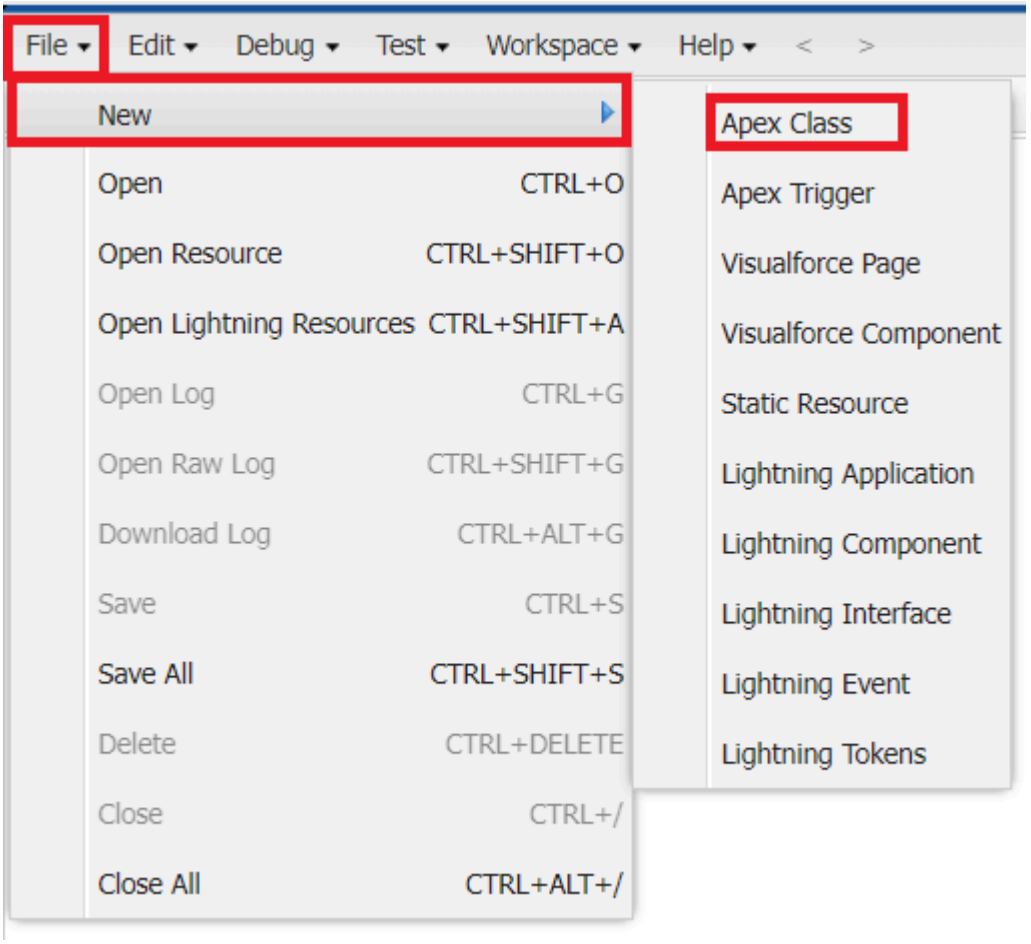


Figure 10: Creating an Apex Class in Developer Console The Developer Console interface, showing the creation of the `OrderTriggerHandler` Apex class.



7. Conclusion

The Salesforce implementation for HandsMen Threads successfully achieved its strategic objectives, delivering a powerful, scalable, and automated CRM solution. By building a custom data model, the project established a single source of truth for all customer, order, product, and inventory data, eliminating data silos and improving overall data quality. The meticulous application of validation rules ensures a high degree of data integrity, which is foundational for reliable reporting and business intelligence.

The automation of key business processes has significantly enhanced operational efficiency and customer engagement. The automated order confirmations, dynamic loyalty status updates, and proactive low-stock alerts have not only reduced manual effort and potential for human error but have also created a more responsive and personalized customer experience. Furthermore, the implementation of asynchronous Apex for nightly batch processing ensures that large data volumes can be handled efficiently without compromising system performance during peak business hours.

This project demonstrates a comprehensive understanding and practical application of core Salesforce technologies, including data modeling, Lightning App Builder, advanced automation with both Flows and Apex, and a robust security framework. The resulting

solution provides HandsMen Threads with the technological foundation needed to manage its business effectively, foster customer loyalty, and scale its operations for future growth in the competitive fashion industry.