**📘 Project Documentation: HandsMen Threads – Salesforce Implementation**

**Overview**

**HandsMen Threads** is a forward-thinking fashion brand focused on premium menswear. To scale operations, improve customer engagement, and streamline inventory/order management, the company initiated a comprehensive Salesforce CRM implementation. This project aimed to create a **centralized, intelligent, and automated business ecosystem** with strong emphasis on **data integrity**, **process automation**, and **customer satisfaction**.

**🧱 Key Objectives**

* Build a **robust and scalable data model** to represent business entities like Customers, Orders, Products, and Inventory.
* Enable **automated workflows** to reduce manual effort and increase accuracy.
* Ensure **real-time communication** with customers and warehouse teams.
* Facilitate **scalable and secure data operations** via Apex and asynchronous processing.

**🗂️ Data Model Design**

The following custom objects were created to model the business entities:

* **HandsMen Customer**
* **HandsMen Order**
* **HandsMen Product**
* **Inventory**

These were interlinked with **lookup/master-detail relationships**, supported by formula fields and validation rules to ensure **referential integrity** and **data quality**.

**✅ Sample Records:**

Screenshots show entries such as:

* **Customers:** customer1, xyz
* **Orders:** O-0001, O-0002, O-0003
* **Products:** Jeans, T-shirt
* **Inventory:** I-0001

**⚙️ Automation Highlights**

**1. Order Confirmation (Record-Triggered Flow)**

When an order record is updated (e.g., status changed to "Confirmed"), an automated email is sent to the customer, enhancing post-purchase communication.

✅ Built using Flow Builder with Run Immediately and Email Alert nodes.  
📸 Refer to attached screenshot of the active flow.

**2. Dynamic Loyalty Program**

* Automatically updates customer loyalty status based on total order value or frequency.
* Enables segment-based marketing and rewards.

**3. Proactive Stock Alerts**

* Inventory levels are monitored.
* If stock drops below a threshold (e.g., 5 units), a **stock alert email** is automatically sent to warehouse staff.

**4. Scheduled Bulk Order Updates (Batch Apex)**

* Every midnight, a batch process:
  + Updates financial reports.
  + Adjusts inventory levels based on orders.
  + Logs transactions for audit.

**🏗️ Development Phases**

**🔹 Phase 1: Architecture & Planning**

* Defined custom objects, fields, and relationships.
* Created Entity-Relationship Diagrams (ERDs) and object model documentation.
* Planned Apex triggers and flow logic.
* Designed branded HTML email templates.

**🔹 Phase 2: Development**

* Created and linked custom objects.
* Implemented:
  + Record-triggered flows
  + Scheduled flows
  + Apex triggers
  + Batch Apex jobs
* Set up **profile-based access control** and field-level security.
* Configured **workflow notifications**.

**✅ 1. Record-Triggered Flows**

**📌 What It Is:**

A **record-triggered flow** is an automation that runs **when a record is created or updated** in Salesforce. It allows declarative (no-code) execution of logic based on field changes or conditions.

**💡 How You Implemented It:**

* **Use Case: Order Confirmation Email**
  + When an order's status is changed to “Confirmed,” a flow was triggered.
  + This flow automatically **sent a confirmation email** to the customer using a **Lightning Email Template**.
  + The flow also logged the action in a custom notification object for tracking.
* **Use Case: Loyalty Program Update**
  + A flow monitored changes in total purchase amount for each customer.
  + Once the threshold (e.g., ₹5000) was met, it **automatically updated the Loyalty Tier** field (Bronze → Silver → Gold).
  + Triggered another flow to **send a congratulatory loyalty upgrade email**.
* **Tools Used:**
  + **Flow Builder**
  + **Email Alert Element**
  + **Decision Element** for condition checks
  + **Update Records Element** to change fields dynamically

**✅ 2. Scheduled Flows**

**📌 What It Is:**

A **scheduled flow** runs **at a specific time or frequency**, such as daily or weekly, to perform background processing tasks across multiple records.

**💡 How You Implemented It:**

* **Use Case: Nightly Bulk Order Processing**
  + A scheduled flow was configured to **run every night at midnight**.
  + It queried all orders placed that day using **Scheduled Path + Get Records**.
  + For each order:
    - Inventory was **reduced based on product quantity ordered**.
    - Financial records were updated with total sales amount.
    - The order status was marked “Processed.”
* **Other Use Cases:**
  + **Daily loyalty tier reassessment** using customer total spend.
  + **Scheduled cleanup or alerts** for incomplete orders or stuck processes.
* **Tools Used:**
  + **Scheduled Flow Trigger**
  + **Loop Elements**
  + **Assignment + Update Records**
  + **Batch logic via loops and collection variables**

**✅ 3. Apex Triggers**

**📌 What It Is:**

**Apex Triggers** are pieces of code that run **before or after record changes**, offering programmatic control that Flows can’t handle—especially for complex, multi-object logic.

**💡 How You Implemented It:**

* **Use Case: Inventory Management**
  + Created a trigger on Order\_\_c object.
  + When an order was confirmed, the trigger:
    - Looked up all ordered products.
    - Deducted quantities from the corresponding Inventory\_\_c records.
    - Handled error cases if stock was insufficient.
* **Use Case: Loyalty Point Calculation**
  + A trigger calculated and updated loyalty points **based on order total**.
  + Applied business logic:
    - ₹1000 = 10 points
    - 50+ points → Tier upgrade

**✅ 4. Batch Apex Jobs (Asynchronous Apex)**

**📌 What It Is:**

**Batch Apex** allows you to process **large datasets asynchronously** in chunks. It’s ideal for nightly operations or data updates that can’t be handled synchronously due to governor limits.

**💡 How You Implemented It:**

* **Use Case: Scheduled Bulk Order and Inventory Processing**
  + Created a class implementing Database.Batchable<SObject>.
  + Every night:
    - The batch job fetched all orders with status “Pending.”
    - It looped through each, marked them as “Processed,” and updated inventory.
    - It also generated a summary log and emailed it to the admin.
* **Use Case: Loyalty Tier Cleanup**
  + Another batch job periodically recalculated loyalty tiers across all customers (e.g., every week).
* **Design Pattern Used:**
  + start → execute → finish methods in batch class
  + Scheduler class with System.scheduleBatch for nightly execution

**🔹 Phase 3: Testing & QA**

* Conducted:
  + Unit tests for triggers and flows.
  + End-to-end flow validation with mock data.
  + Performance testing for batch jobs.
  + Security review to ensure data access compliance.

**🔹 Phase 4: Deployment & Training**

* Promoted configuration from sandbox to production.
* Trained internal teams using hands-on walkthroughs and recorded sessions.
* Post-go-live monitoring using debug logs and user feedback.

**📄 Deliverables**

* ✅ **Solution Design Document**
* ✅ **Entity Relationship Diagram (ERD)**
* ✅ **Automation Strategy Report**
* ✅ **Flow diagrams and screenshots**
* ✅ **Test Case Documentation**
* ✅ **Email Template Repository**
* ✅ **User Training Materials**

**💡 Key Features Demonstrated**

| **Feature** | **Tool/Concept Used** |
| --- | --- |
| Custom Business Objects | Data Modeling in Object Manager |
| Data Quality Enforcement | Validation Rules & Formula Fields |
| UI-Based Automation | Record-Triggered Flows |
| Complex Business Logic | Apex Triggers & Batch Apex |
| Real-Time Communication | Email Alerts |
| Scheduled Data Processing | Asynchronous Apex Jobs |
| Role-Based Data Access | Org-Wide Defaults & Sharing Rules |
| App Configuration | Lightning App Builder |

**🧠 What You Learn from This Project**

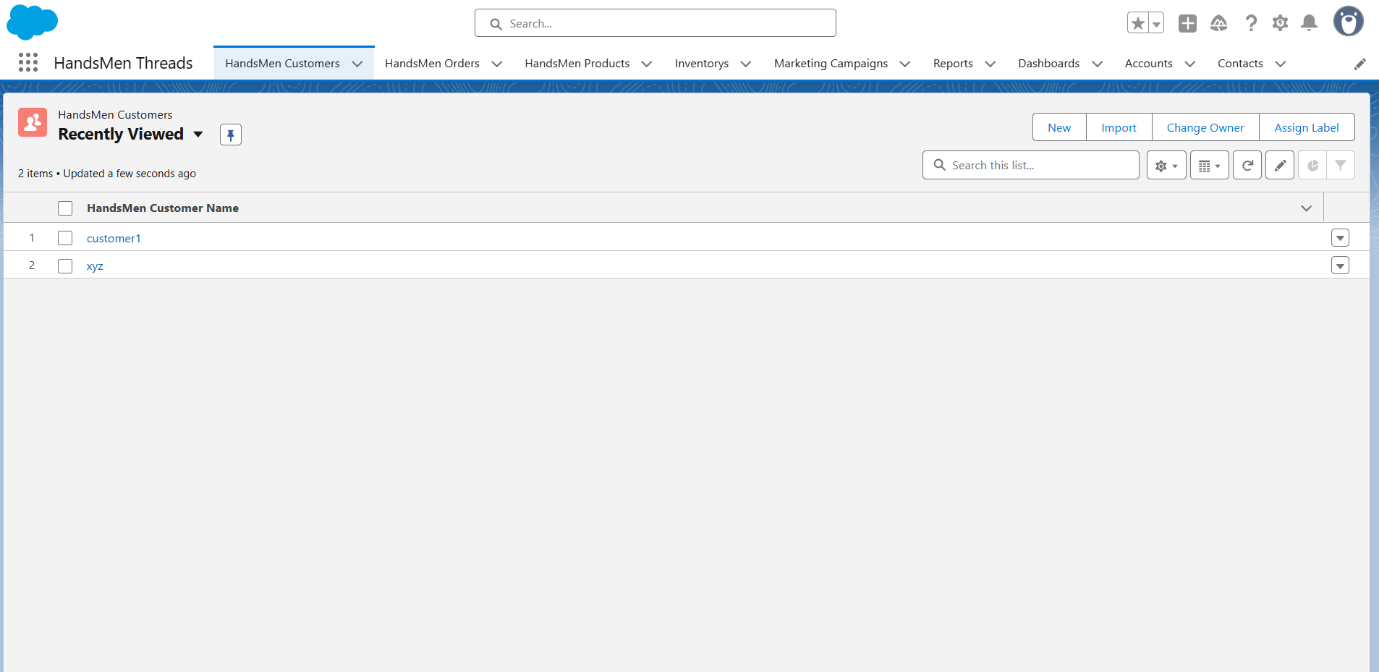
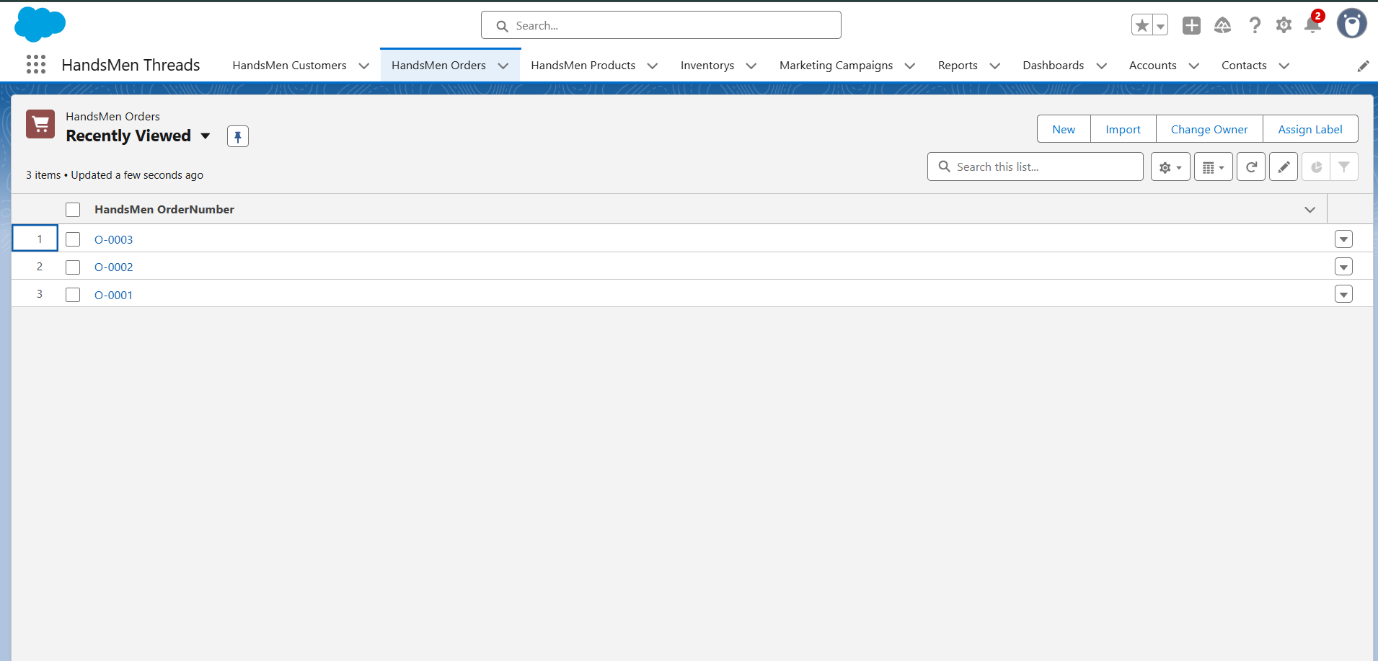
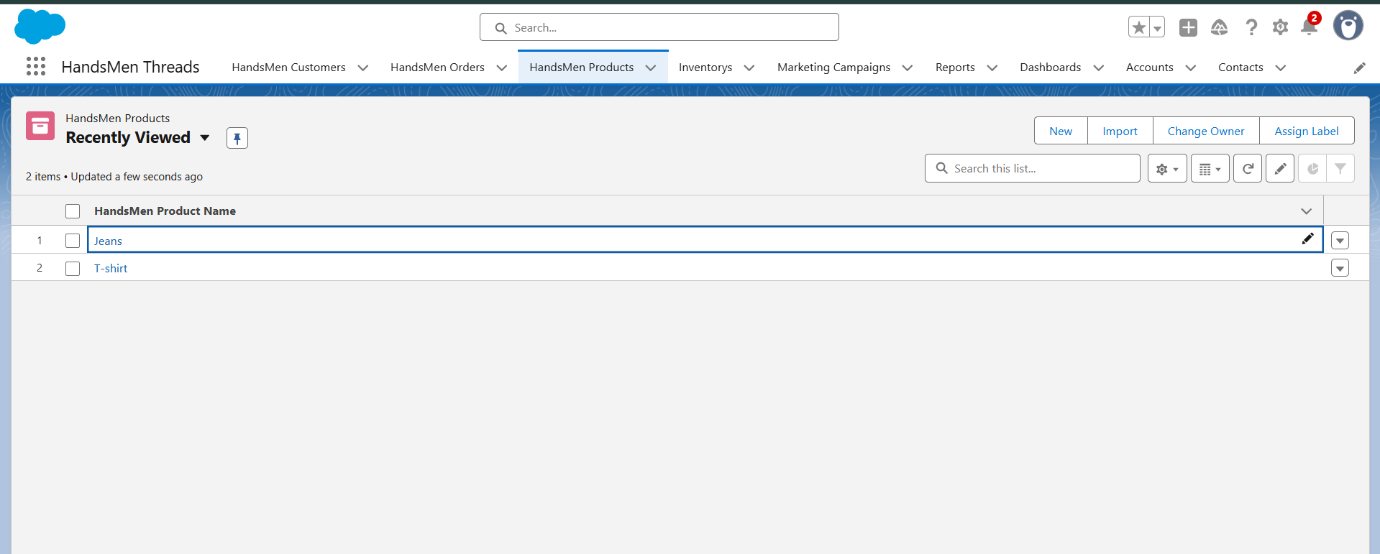
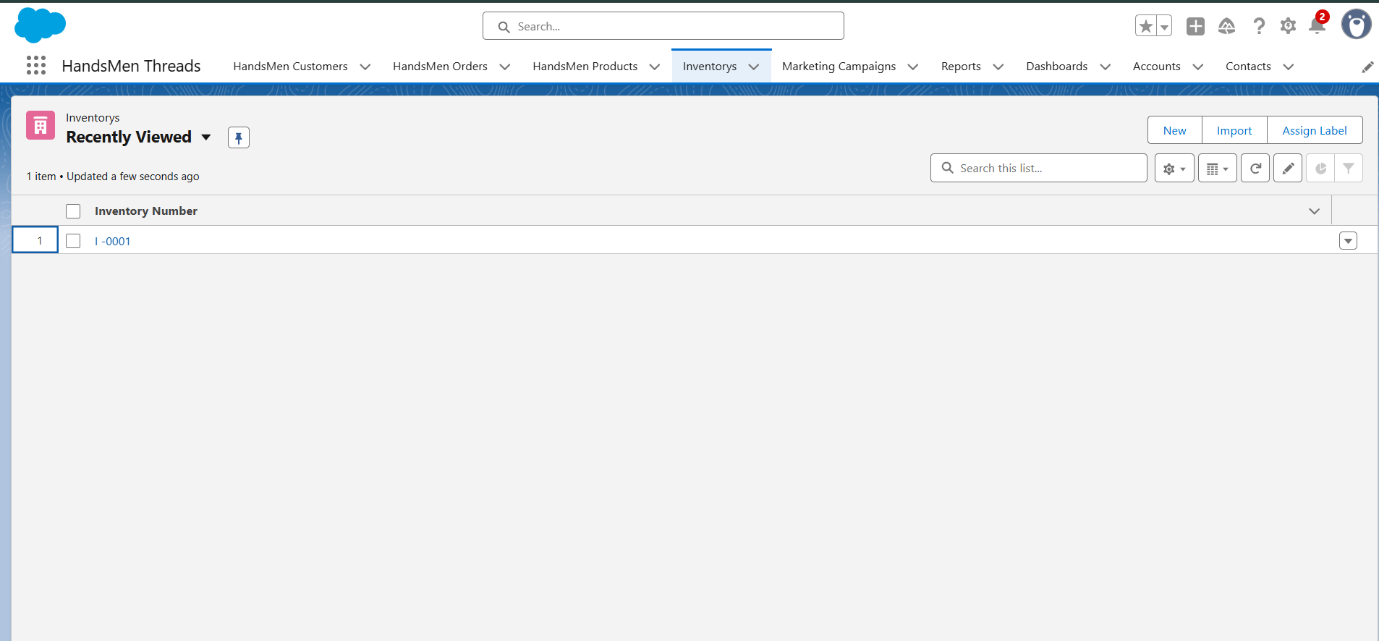
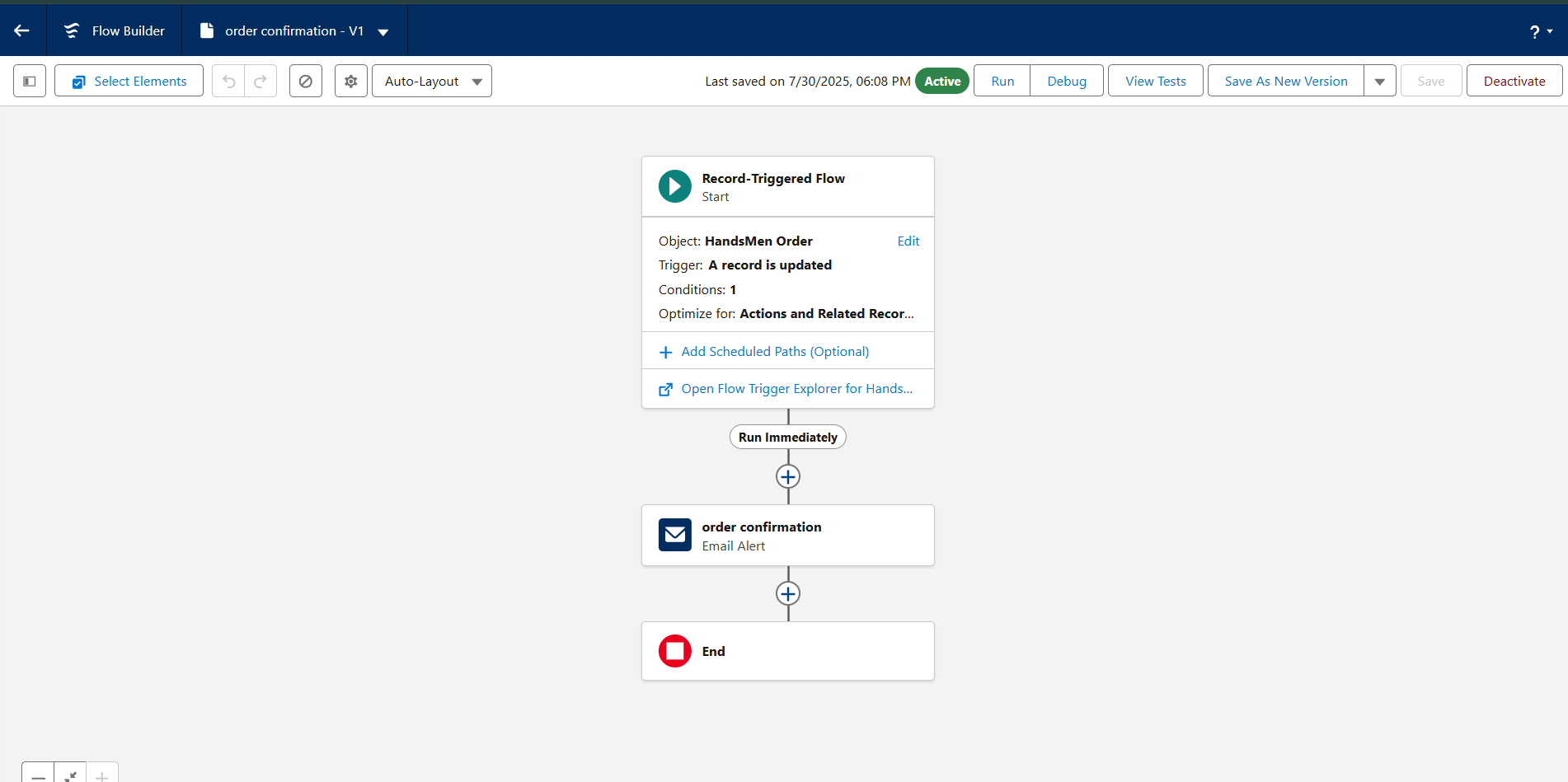
This project exposes developers to real-world Salesforce implementation using:

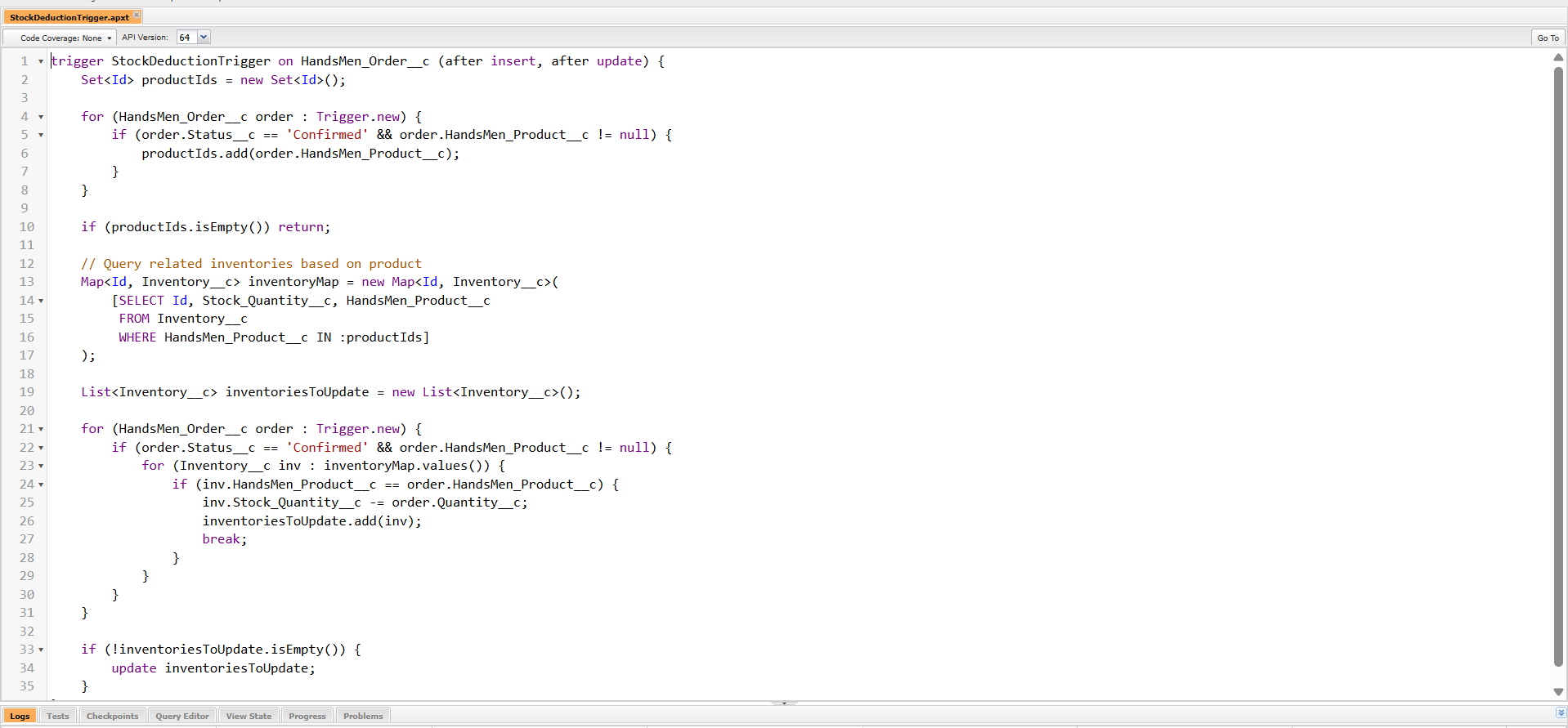
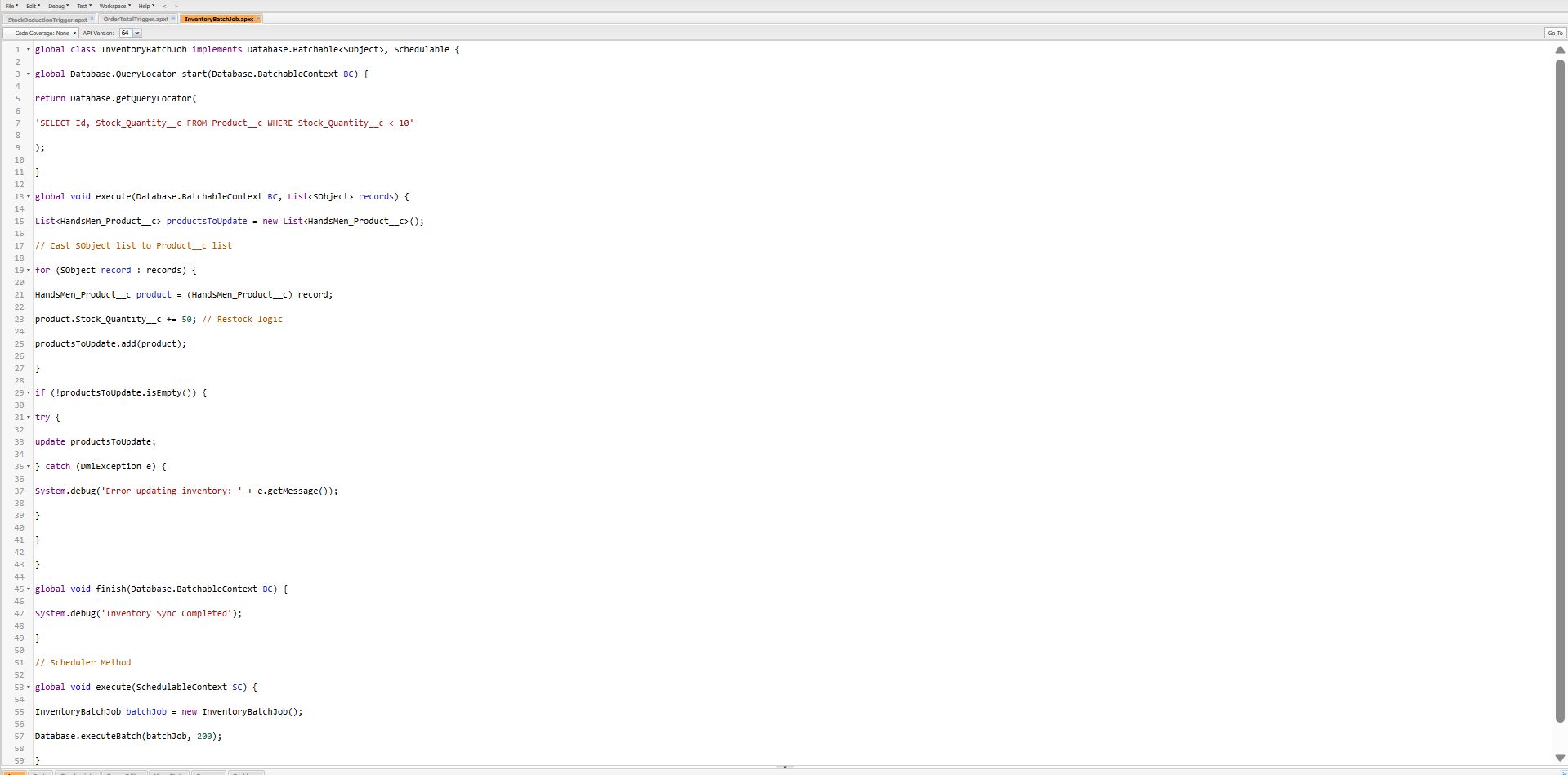
* 🔷 **Data Modeling:** Designing objects and relationships aligned with business goals.
* 🔷 **Lightning App Builder:** Customizing UIs for specific roles.
* 🔷 **Record-Triggered Flows:** Automating routine tasks like notifications and updates.
* 🔷 **Apex Programming:** Writing custom logic for loyalty and inventory processing.
* 🔷 **Asynchronous Apex:** Batch and scheduled jobs for performance and scalability.
* 🔷 **Data Quality Management:** Validations, lookups, formulas to prevent inconsistency.

**📸 Screenshots Overview**

1. **Customers, Orders, Products, and Inventory Lists** – Evidence of object setup and sample data.
2. **Flow Builder: Order Confirmation Email Automation** – Demonstrates real-time automation using record-triggered flows.

*All screenshots represent a live, working solution built in Salesforce Lightning Experience.*





**✅ Conclusion**

The HandsMen Threads Salesforce project is a full-cycle CRM implementation that integrates powerful features like automation, Apex logic, and clean UI design to deliver business transformation. This solution not only simplifies business operations but also strengthens customer satisfaction and operational intelligence — all built on a strong foundation of **data-driven design** and **automation-first principles**.