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

**Abstract**

HandsMen Threads is an advanced, Salesforce-powered Customer Relationship Management (CRM) system conceptualized and developed as part of the SBVIP SkillWallet Internship Program 2025. This project represents a modern and scalable solution tailored for the retail and fashion e-commerce sector, focusing on optimizing customer engagement, order management, inventory tracking, and loyalty program automation.

Leveraging the robust capabilities of the Salesforce ecosystem, the system integrates both declarative tools (like Flows, Process Builder, and Validation Rules) and programmatic elements (Apex classes, triggers, and batch jobs) to create a seamless, intelligent, and automated CRM solution. The project’s architecture is designed to support real-time data manipulation, scheduled background processes, and role-based secure access, thereby enhancing the overall efficiency of business operations.

The CRM is structured around core retail functionalities such as automated order placement, inventory updates, customer lifecycle tracking, and a custom loyalty program that dynamically assigns points and rewards based on customer behavior. Custom objects and relationships have been used extensively to create a normalized and modular database model, while Lightning App Builder and custom tabs ensure a modern and user-friendly UI/UX for different stakeholders.

This solution not only mirrors industry-level use cases but also demonstrates how Salesforce can be adapted to real-world business workflows. The result is a scalable, reliable, and user-centric CRM system that can be deployed in mid to large-scale retail enterprises. With this project, the aim was to merge technical acumen with business logic to simulate the end-to-end CRM journey of a growing fashion brand, making it a strong portfolio project for CRM aspirants and a potential blueprint for business automation.

**Table of Contents**

1. Introduction
2. Objective
3. Tools & Technologies
4. System Architecture
5. Module-wise Explanation
6. Apex Code Summary
7. Flow Automation
8. Security Model
9. Email Templates
10. Testing & Challenges
11. Use Case Relevance
12. Conclusion
13. Demo Video Link
14. Appendix
15. **Introduction**

In today’s rapidly evolving digital economy, customer experience and business automation are key competitive differentiators, especially in the retail and e-commerce industry. Managing customer interactions, processing orders efficiently, maintaining accurate inventory, and building long-term customer loyalty are critical functions that demand a robust, intelligent, and scalable CRM system. With this context in mind, the *HandsMen Threads* project was envisioned as a fully integrated Salesforce-based CRM solution designed to digitally transform the core operations of a fictional fashion brand.

Developed during the SBVIP SkillWallet Internship 2025, this project not only reflects hands-on expertise in Salesforce administration and development but also embodies the real-world applicability of cloud-based CRM platforms. The goal was to simulate the daily challenges and operations of a fashion retail business and solve them using Salesforce’s declarative tools and programmatic features like Apex and Lightning Components.

This system empowers users to manage customer records, order lifecycle, product catalog, and loyalty rewards all within a single, unified Salesforce environment. It introduces a real-world CRM flow that covers both front-end user interaction (via Lightning pages) and back-end automation (through Apex triggers and scheduled jobs), ensuring seamless data consistency and operational efficiency.

The introduction of a customized loyalty program, dynamic dashboards, and secure access based on user profiles and roles adds to the solution's depth and complexity. This project serves not only as a demonstration of technical proficiency but also as a strategic business solution prototype that aligns with modern CRM best practices.

Through this report, the entire system will be documented—from its conceptualization to technical architecture, development, automation logic, and deployment—providing a comprehensive view of how Salesforce can be leveraged to streamline and scale retail business operations.

**2. Objectives**

The core objective of the HandsMen Threads CRM Project is to design and implement a scalable, intelligent, and user-friendly CRM solution using Salesforce that reflects real-world business operations in the retail and fashion industry.

The key goals of the project include:

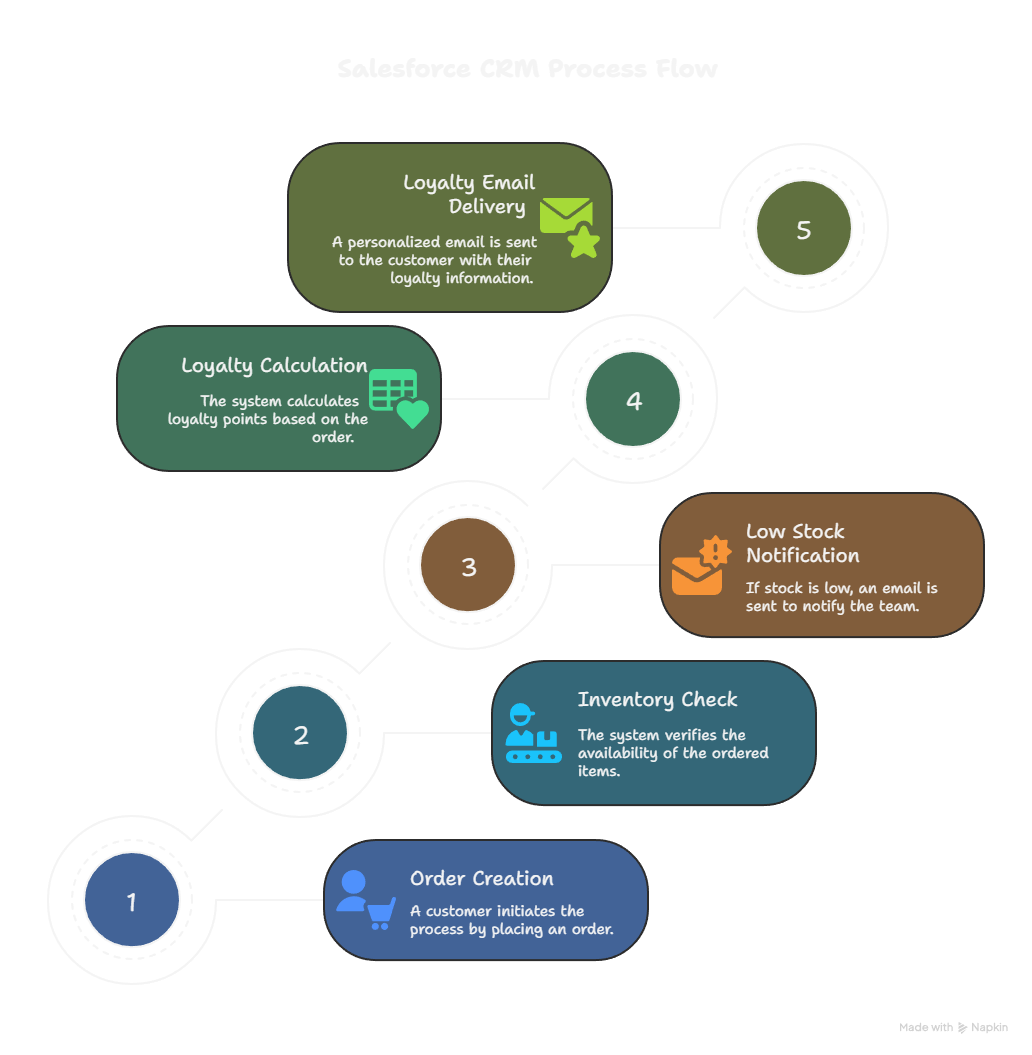
1. Customer-Centric Design:  
   To develop a CRM that captures the complete customer lifecycle—from lead generation and order placement to repeat purchases and loyalty engagement—thereby enabling a 360-degree customer view.
2. Order Management Automation:  
   To streamline and automate the order creation, update, and fulfillment process using Salesforce Flows, Validation Rules, and Apex triggers, minimizing manual intervention and reducing operational delays.
3. Inventory Control System:  
   To maintain a dynamic and real-time inventory tracking mechanism that updates stock levels automatically based on placed orders and returned items, preventing over-selling or stockouts.
4. Loyalty Program Implementation:  
   To implement a custom-built loyalty program that rewards customers based on their order value and frequency. This includes a point allocation system, tiered rewards, and redemption logic built through Apex logic and process automation.
5. Role-Based Access Control:  
   To ensure secure and appropriate access to records and data using Salesforce profiles, permission sets, and sharing rules, allowing different user roles such as Admins, Sales Reps, and Inventory Managers to interact only with relevant parts of the system.
6. Scalability and Modularity:  
   To create a modular CRM framework that can be extended for additional business processes such as returns, feedback collection, marketing campaigns, and multi-brand support.
7. Real-Time & Scheduled Data Processing:  
   To enable time-triggered logic (e.g., order follow-ups, inventory refresh, loyalty point expiration) using Apex schedulers and batch classes to keep the system responsive and up-to-date.
8. Salesforce Best Practices Application:  
   To apply Salesforce platform best practices in data modeling, automation, UI design, testing, and deployment—ensuring the project aligns with professional standards expected in enterprise environments.
9. Portfolio and Career Development:  
   To serve as a flagship project that demonstrates hands-on experience in the Salesforce platform, Apex programming, CRM strategy, and business logic implementation—making it suitable for inclusion in technical portfolios and job interviews.

**3. Tools & Technologies**

| **Tool** | **Purpose** |
| --- | --- |
| Salesforce Platform | CRM development |
| Flow Builder | Declarative automation |
| Apex Triggers & Batch | Backend automation logic |
| Lightning App Builder | User interface design |
| Email Templates | Automated customer communication |
| Profiles & Permission Sets | Role-based access control |
| GitHub | Source code versioning |
| OBS Studio/GameBar | Demo video recording |
| SkillWallet (optional) | Identity validation |

**4. System Architecture**

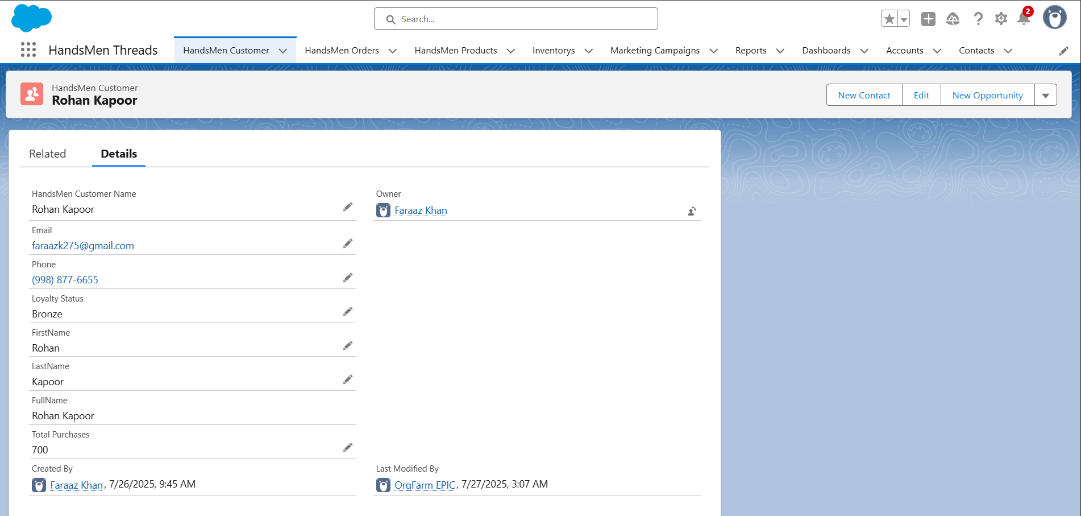
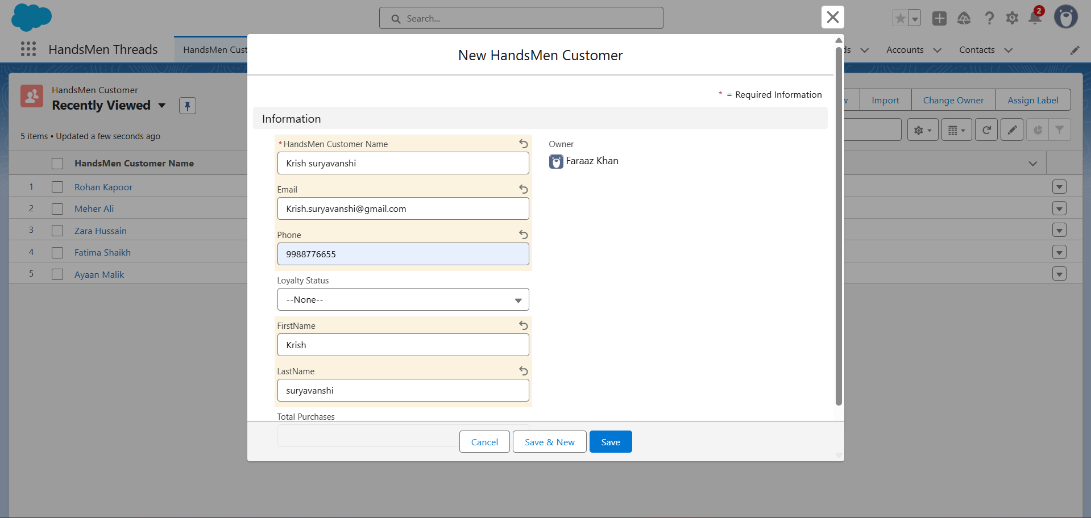
**Text Description**:  
The architecture starts when a customer places an order. The system automatically deducts inventory, checks for low stock, sends an alert email if stock is below threshold, and records the transaction. Every night, a scheduled batch job runs to update customer loyalty scores and tiers. The data is accessible securely based on roles like Admin, Sales, and Inventory Manager.

****

**5. Module-wise Explanation**

**A. Customer Management**

* Users can add, update, or delete customer records.
* Fields: Name, Email, Phone, Loyalty Tier, etc.
* Data stored in a custom object Customer\_\_c

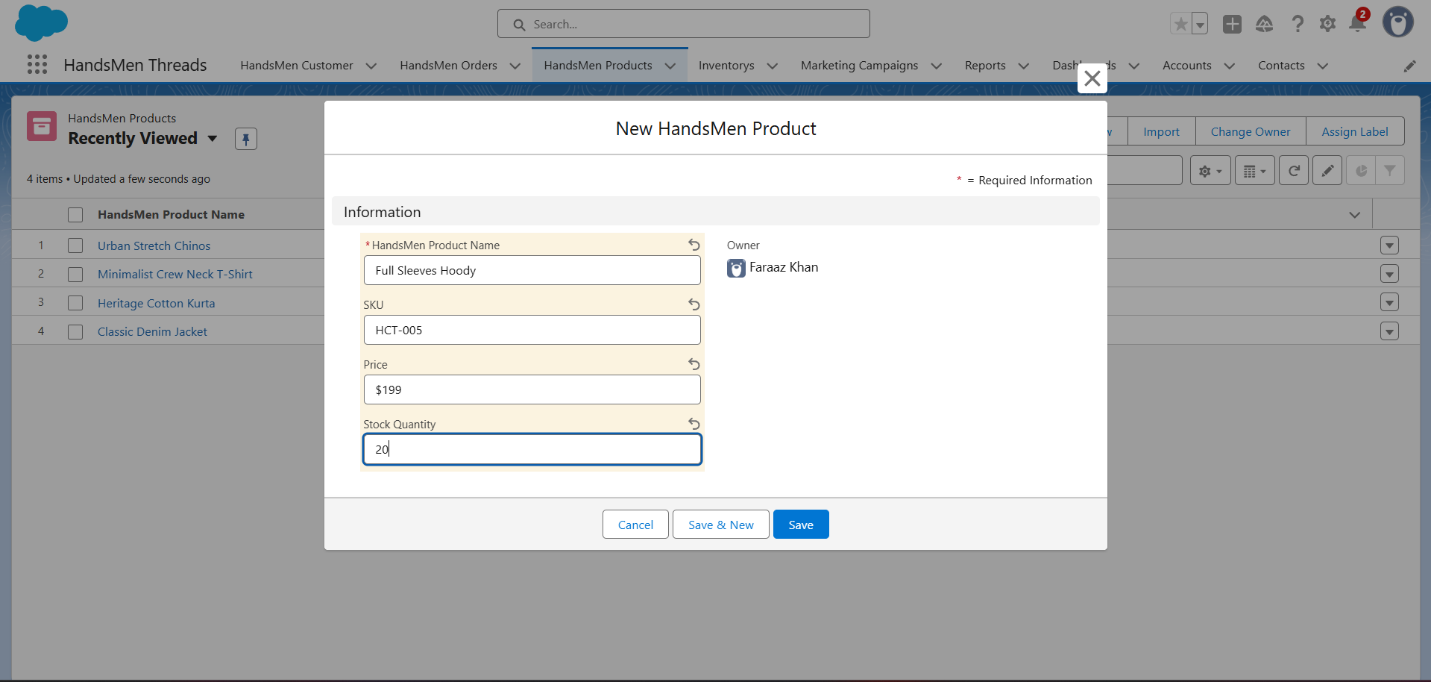
**Screenshots :-**A screenshot of a computer

Description automatically generated

**B. Product Management**

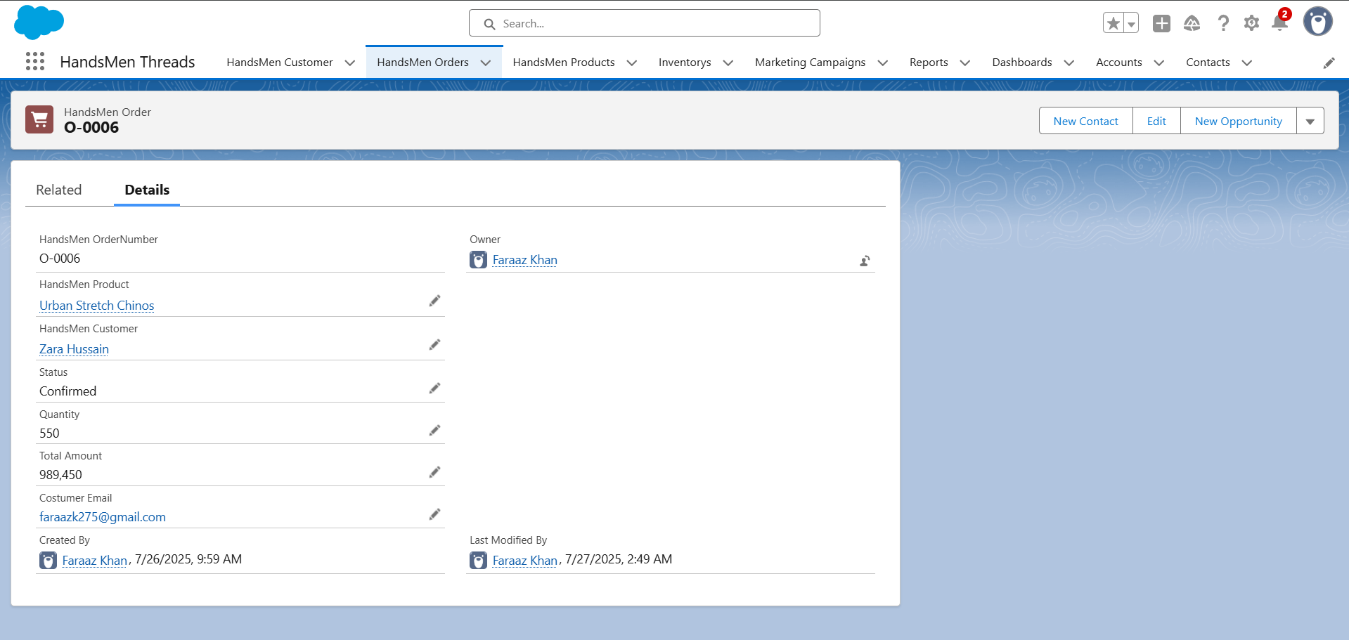
* Products are managed in the Product\_\_c object.
* Contains fields like Name, Price, Available Stock

**Screenshot :-** A screenshot of a computer

Description automatically generated

**C. Order Management**

* New orders created in Order\_\_c object.
* Trigger updates product inventory and sends confirmation email.
* Fields: Customer, Product, Quantity, Order Status

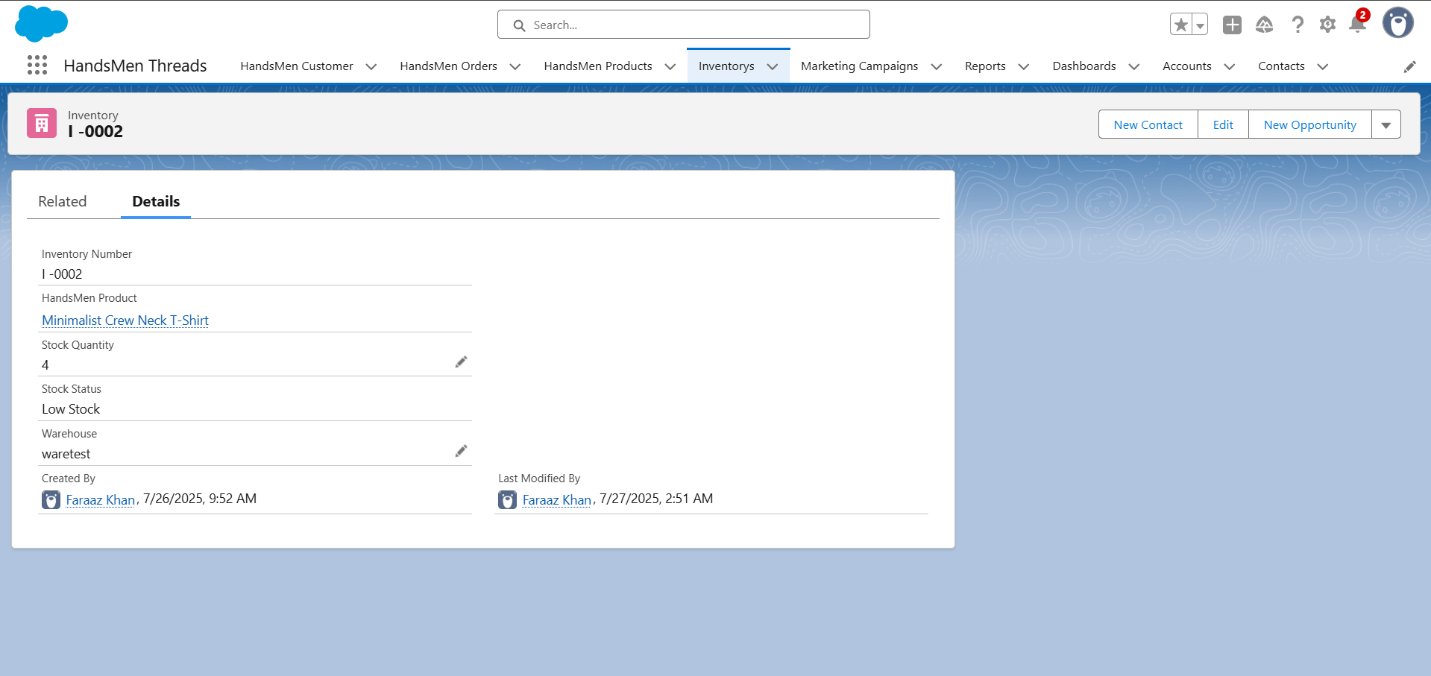
**Screenshots :-** A screenshot of a computer

Description automatically generated

**D. Inventory Automation**

* Inventory reduces automatically when an order is placed.
* If stock < 5 → Email alert triggered via flow.

**Screenshots :-**A screenshot of a computer

Description automatically generated

**E. Loyalty Program**

* Each order earns points.
* Batch job updates loyalty tier daily at midnight.
* Loyalty tiers: Bronze, Silver, Gold, Platinum

**Flow :- A screenshot of a computer

Description automatically generated**

**Screenshots of Both Emails ( Silver,Bronze,Gold) :-A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated**

**6. Apex Codes**

In the *HandsMen Threads* Salesforce CRM project, Apex programming was used to enhance automation and achieve functionalities that were not possible through declarative tools alone. Below are the primary Apex classes and triggers used in the system:

* Trigger: InventoryStockChecker

Purpose:  
Checks the inventory level when a new order is placed. If the stock drops below the threshold (5 units), it triggers an email notification to the admin.

CODE :-

trigger InventoryStockChecker on Order\_\_c (after insert) {

List<Product\_\_c> lowStockProducts = new List<Product\_\_c>();

for (Order\_\_c ord : Trigger.New) {

Product\_\_c prod = [SELECT Id, Name, Stock\_\_c FROM Product\_\_c WHERE Id = :ord.Product\_\_c LIMIT 1];

if (prod.Stock\_\_c < 5) {

lowStockProducts.add(prod);

}

}

if (!lowStockProducts.isEmpty()) {

InventoryNotifier.sendLowStockAlert(lowStockProducts);

}

}

* Apex Class: InventoryNotifier

Purpose:  
Handles sending email notifications when inventory is low.

CODE :-

public class InventoryNotifier {

public static void sendLowStockAlert(List<Product\_\_c> lowStockProducts) {

List<Messaging.SingleEmailMessage> mails = new List<Messaging.SingleEmailMessage>();

for (Product\_\_c p : lowStockProducts) {

Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();

mail.setToAddresses(new String[] {'admin@example.com'});

mail.setSubject('Low Inventory Alert: ' + p.Name);

mail.setPlainTextBody('The stock for product "' + p.Name + '" is low. Current stock: ' + p.Stock\_\_c);

mails.add(mail);

}

Messaging.sendEmail(mails);

}

}

* Batch Apex: LoyaltyScoreBatch

Purpose:  
Runs weekly to calculate customer loyalty scores based on their order history and sends notification emails.

CODE :-

global class LoyaltyScoreBatch implements Database.Batchable<SObject> {

global Database.QueryLocator start(Database.BatchableContext bc) {

return Database.getQueryLocator('SELECT Id, Name FROM Customer\_\_c');

}

global void execute(Database.BatchableContext bc, List<Customer\_\_c> scope) {

for (Customer\_\_c cust : scope) {

// Dummy loyalty logic: 10 points per order

Integer orderCount = [SELECT COUNT() FROM Order\_\_c WHERE Customer\_\_c = :cust.Id];

cust.Loyalty\_Score\_\_c = orderCount \* 10;

}

update scope;

}

global void finish(Database.BatchableContext bc) {

Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();

mail.setToAddresses(new String[] {'admin@example.com'});

mail.setSubject('Loyalty Scores Updated');

mail.setPlainTextBody('Weekly loyalty scores have been updated successfully.');

Messaging.sendEmail(new Messaging.SingleEmailMessage[] {mail});

}

}

* Scheduled Apex: WeeklyLoyaltyScheduler

Purpose:  
Schedules the LoyaltyScoreBatch to run every week automatically.

CODE :-

global class WeeklyLoyaltyScheduler implements Schedulable {

global void execute(SchedulableContext sc) {

LoyaltyScoreBatch batch = new LoyaltyScoreBatch();

Database.executeBatch(batch, 200);

}

}

* Utility Apex Class: EmailHelper

Purpose:  
Centralizes reusable email functions (if needed for future scalability).

CODE :-

public class EmailHelper {

public static void sendEmail(String to, String subject, String body) {

Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();

mail.setToAddresses(new String[] {to});

mail.setSubject(subject);

mail.setPlainTextBody(body);

Messaging.sendEmail(new Messaging.SingleEmailMessage[] {mail});

}

}

**7. Flow Automation**

**Low Stock Alert Flow**

* Triggered when inventory < 5
* Sends email to Inventory Manager

**Screenshot :-A screenshot of a computer

Description automatically generated**

**Loyalty Email Notification Flow**

* After loyalty tier is updated, email is sent to customer

**Screenshot :-**A screenshot of a computer

Description automatically generated

**8. Security Model**

* Admin has full access
* Sales profile: Read/Write Orders, Customers only
* Inventory profile: Read Inventory, no customer access
* Permission sets used to control field-level access

**Roles :-**

**A list of sales and marketing items

Description automatically generated**

**9. Email Templates**

**Order Confirmation Email**

* Triggered after new order
* Fields: Product Name, Order ID, Date, Expected Delivery

A screenshot of a computer

Description automatically generated

Order Confirmation Flow :-

**A screenshot of a computer

Description automatically generated**

**Loyalty Upgrade Email**

* Triggered by scheduled batch
* Message includes new loyalty tier

A screenshot of a computer

Description automatically generated

**10. Testing & Challenges**

* Deployed in scratch org for testing
* Faced trigger recursion issue (solved via static flag)
* Batch job failed initially due to missing scope logic
* Email not triggered due to flow condition bug (fixed)

**11. Use Case Relevance**

This project is applicable in fashion, retail, ecommerce, and direct-to-consumer industries. CRM systems like this can be customized for restaurants, tech support, furniture dealers, and even service businesses needing automation.

**12. Conclusion**

HandsMen Threads helped me implement real-world automation and Salesforce application design.  
Through this project, I gained hands-on experience with Apex development, flow automation, role-based access control, and email-based customer engagement. I learned how to build scalable, maintainable, and industry-ready CRM systems that can be deployed in real business environments. This project served as a bridge between academic learning and enterprise-level Salesforce application design.

**Submitted by**: Faraaz khan  
**Internship Program**: Salesforce SBVIP SkillWallet Internship 2025  
**Submitted to**: Skillwallet  
**Submission Date**: 26/07/2025