

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

1. Abstract

The HandsMen Threads Salesforce project is designed to modernize and elevate customer engagement and backend operations for a high-end men's fashion brand. Focused on building a data-driven platform, the solution enables automated loyalty management, real-time stock monitoring, and seamless order processing.

Leveraging Lightning components, Apex Triggers, record-triggered Flows, and Batch Apex, the solution enforces business logic at scale and empowers both customers and internal teams with accurate, real-time information. Expected outcomes include improved customer retention through personalized loyalty programs, operational efficiency via automation, and enhanced decision-making using centralized data.

2. Objectives

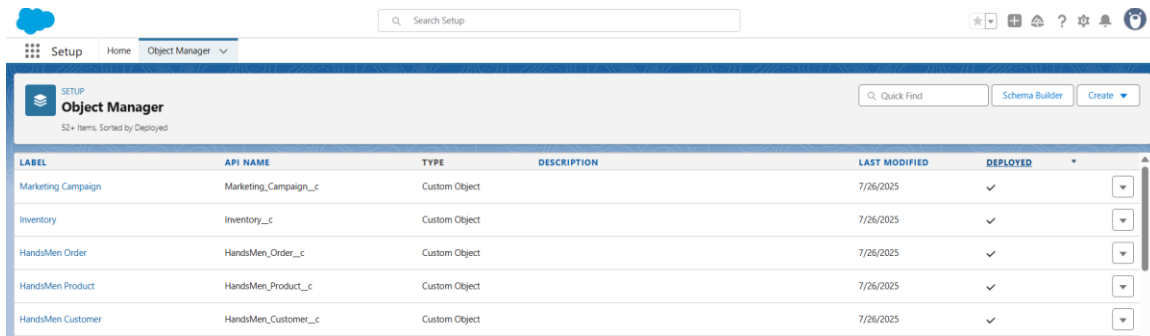
- Automate order total calculations based on product price and quantity.
- Assign loyalty tiers to customers dynamically based on purchase history.
- Trigger real-time updates of total purchases on the customer profile.
- Maintain accurate inventory by linking order data to product stock levels.
- Prevent manual data inconsistencies through UI validation and backend automation.

3. Data Management

3.1 Objects

The solution leverages custom objects that represent core business entities:

- ❖ **HandsMen_Product__c:** Represents catalog items with price and category.
- ❖ **HandsMen_Customer__c:** Tracks customer profile, purchase history, and loyalty tier.
- ❖ **HandsMen_Order__c:** Captures transactions, quantity, and calculated total.
- ❖ **Inventory__c:** Tracks available stock for each product and supports stock alerts.
- ❖ **Marketing_Campaign__c:** Manages promotional campaigns and targeted customer outreach.



Object Manager

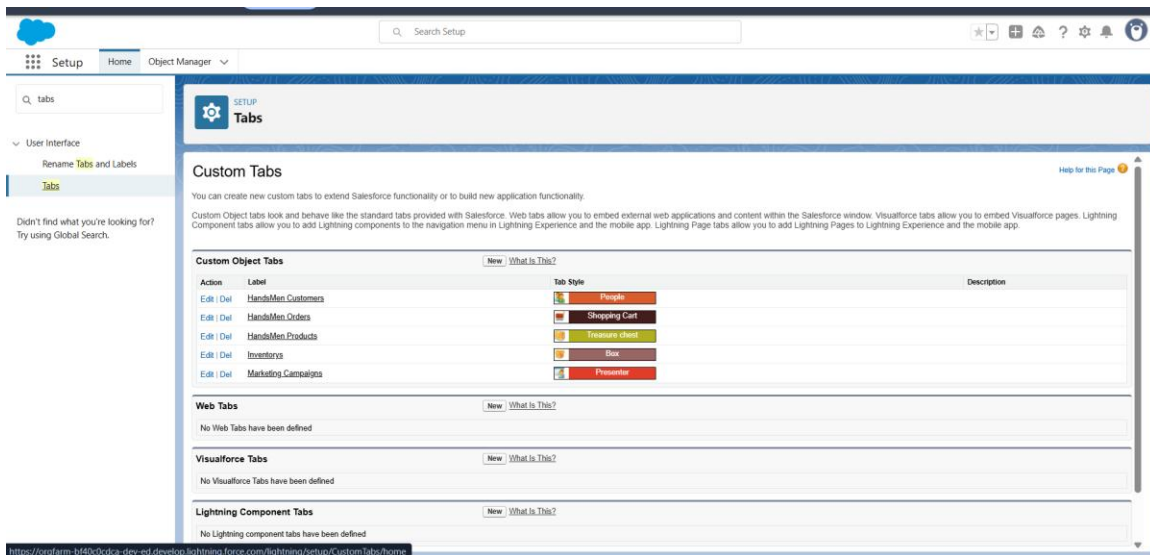
52+ Items. Sorted by Deployed

LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED
Marketing Campaign	Marketing_Campaign__c	Custom Object		7/26/2025	✓
Inventory	Inventory__c	Custom Object		7/26/2025	✓
HandsMen Order	HandsMen_Order__c	Custom Object		7/26/2025	✓
HandsMen Product	HandsMen_Product__c	Custom Object		7/26/2025	✓
HandsMen Customer	HandsMen_Customer__c	Custom Object		7/26/2025	✓

3.2 Tabs

Custom tabs are created for each object, featuring Lightning record pages with:

- Related Lists
- Quick Actions
- Embedded Reports and Dashboards



Custom Tabs

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.

Action	Label	Tab Style	Description
Edit Del	HandsMen Customers	People	
Edit Del	HandsMen Orders	Shopping Cart	
Edit Del	HandsMen Products	Treasure chest	
Edit Del	Inventory	Box	
Edit Del	Marketing Campaigns	Presenter	

Web Tabs

No Web Tabs have been defined

Visualforce Tabs

No Visualforce Tabs have been defined

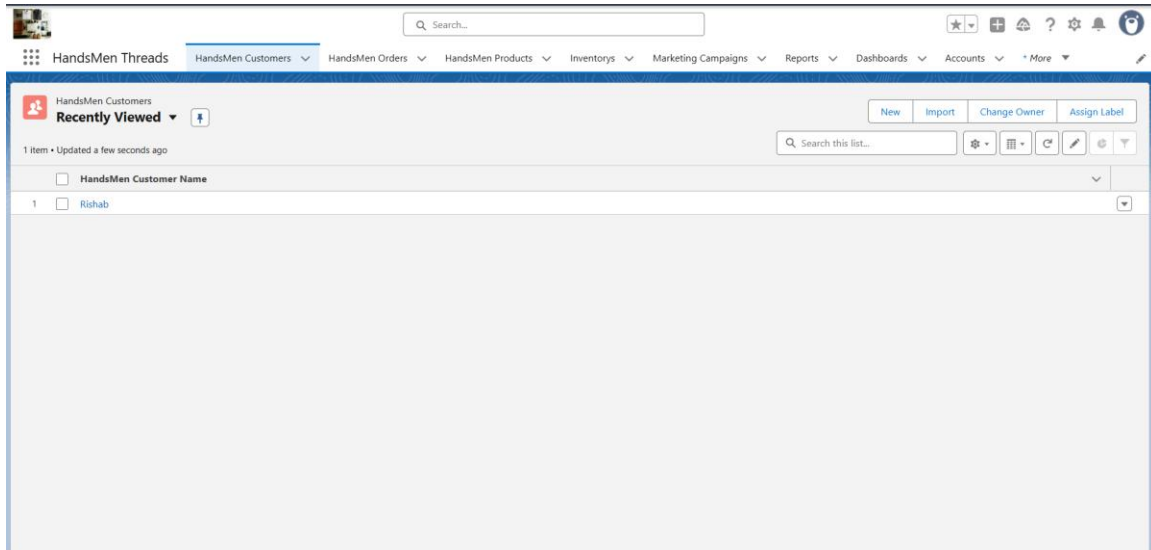
Lightning Component Tabs

No Lightning component tabs have been defined

3.3 App Manager

A dedicated app, "HandsMen Threads ", is configured in Lightning App Builder. The app includes:

- Tabs for Product, Order, and Customer
- Page layouts based on user roles (Sales, Inventory,Marketing, Admin)



3.4 Fields

The filed on the objects are show below:

<i>Object Name</i>	<i>Type</i>	<i>Description</i>	<i>Key Fields</i>
<i>HandsMen_Customer__c</i>	Custom Object	Stores customer details	Name (Record Name), Email (Email), Phone (Phone), Loyalty_Status__c (Picklist: Bronze, Gold, Silver), Total_Purchases__c (Number)
<i>HandsMen_Product__c</i>	Custom Object	Stores product catalog	Name (Record Name), SKU (Text), Price (Currency), Stock_Quantity__c (Number)
<i>HandsMen_Order__c</i>	Custom Object	Stores customer orders	Order_Number (Record Name), Status (Picklist: Pending, Confirmed, Rejection), Quantity__c (Number), Total_Amount__c (Number)
<i>Inventory__c</i>	Custom Object	Tracks inventory levels	Auto Number (Record Name), Warehouse (Text), Stock_Quantity__c (Number)
<i>Marketing_Campaign__c</i>	Custom Object	Manages promotions & campaigns	Campaign_Name (Record Name), Start_Date (Date), End_Date (Date)

New HandsMen Customer

* = Required Information

Information

*HandsMen Customer Name: Armaan

Email: sanchitnegi177@gmail.com

Phone: 8445042254

Loyalty Status: --None--

FirstName: Armaan

LastName: Rawat

Owner: Sanchit Negi

Cancel Save & New Save

Fig 3.4.1 Registering new customer

New HandsMen Product

* = Required Information

Information

*HandsMen Product Name: Jeans

SKU:

Price: \$100

Stock Quantity: 250

Owner: Sanchit Negi

Cancel Save & New Save

Fig 3.4.2 Registering new products

New Inventory

* = Required Information

Information

Inventory Number: I-0002

*HandsMen Product: Shirts

Stock Quantity: 6,000

Warehouse: Kotdwar Zone 1

Cancel Save & New Save

Fig 3.4.3 Adding products to inventory

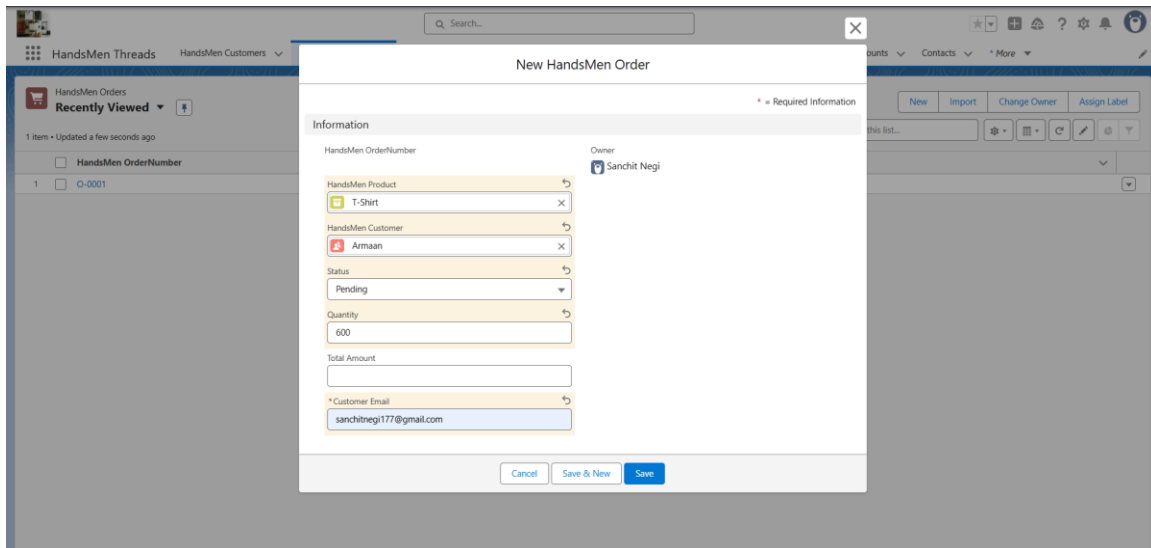
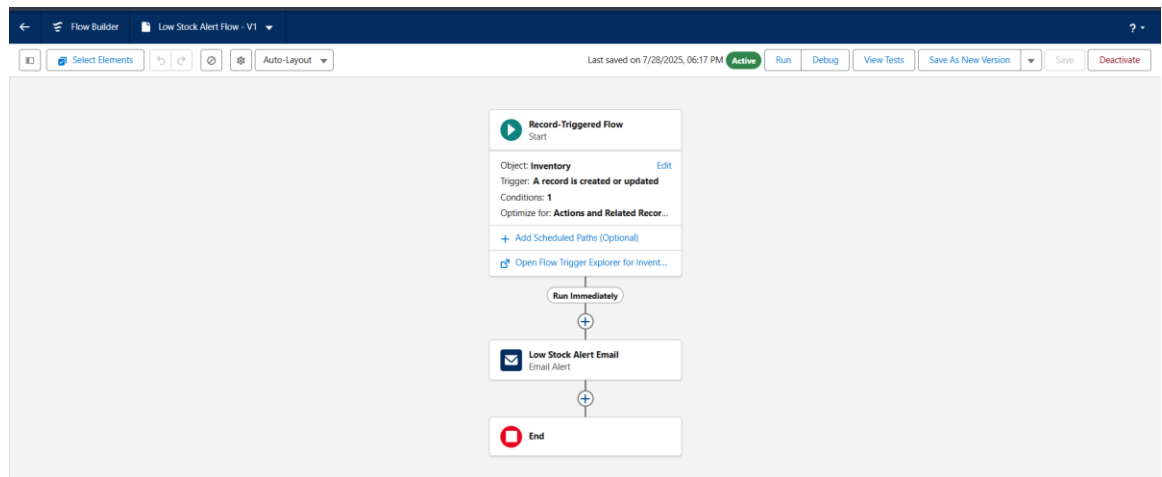


Fig 3.4.4 Placing order

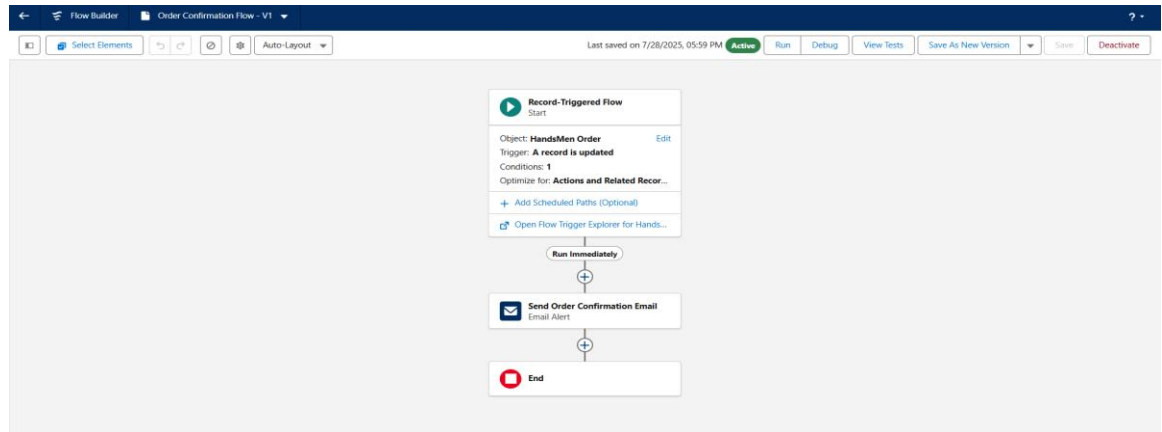
4. Automation

4.1 Flows

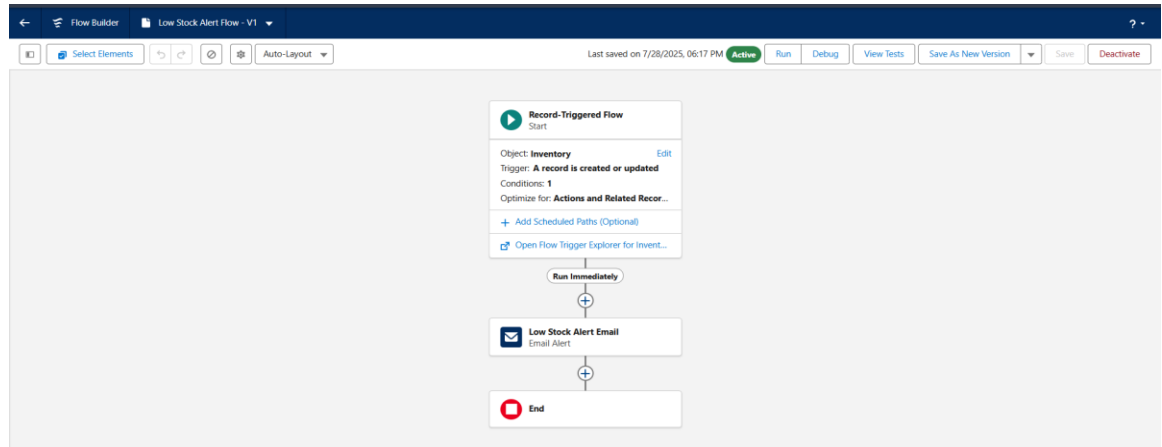
- ❖ **Loyalty Status Update Flow** : Automatically updates a customer's loyalty tier (Bronze, Silver, Gold) when confirmed orders change their total purchase value.



- ❖ **Order Confirmation Flow** : Sends a confirmation email to customers after their order is successfully placed and marked as 'Confirmed'.



- ❖ **Stock Alert Flow** : Sends a notification email to the inventory team when a product's stock quantity drops below 5 units.



4.2 Apex Triggers

- ❖ **Update_Order_Total (before insert/update on HandsMen_Order__c):**
 - Calculates and populates Total_Amount__c

```

1  trigger Update_Order_Total on HandsMen_Order__c (before insert, before update) {
2
3      Set<Id> pids = new Set<Id>();
4
5      for (HandsMen_Order__c order : Trigger.new) {
6          if (order.HandsMen_Product__c != null) {
7              pids.add(order.HandsMen_Product__c);
8          }
9      }
10
11      Map<Id, HandsMen_Product__c> productMap = new Map<Id, HandsMen_Product__c>(
12          [SELECT Id, Price__c FROM HandsMen_Product__c WHERE Id IN :pids]
13      );
14
15      for (HandsMen_Order__c order : Trigger.new) {
16          if (order.HandsMen_Product__c != null && productMap.containsKey(order.HandsMen_Product__c)) {
17              HandsMen_Product__c product = productMap.get(order.HandsMen_Product__c);
18              if (order.Quantity__c != null) {
19                  order.Total_Amount__c = order.Quantity__c * product.Price__c;
20              }
21          }
22      }
23
24  }
  
```

- ❖ **Stock_Deduction (after insert/update on HandsMen_Order__c):**
 - Real-time updates to inventory when orders are placed

```

1  * trigger Stock_Deduction on HandsMen_Order__c (after insert, after update) {
2      Set<Id> pids = new Set<Id>();
3      for (HandsMen_Order__c order : Trigger.new) {
4          if (order.HandsMen_Product__c != null && order.Status__c == 'Confirmed') {
5              pids.add(order.HandsMen_Product__c);
6          }
7      }
8      if (pids.isEmpty()) return;
9      else
10     {
11         Map<Id, Inventory__c> mapOfInventory = new Map<Id, Inventory__c>();
12         [SELECT Id, Stock_Quantity__c, HandsMen_Product__c
13          FROM Inventory__c
14          WHERE HandsMen_Product__c IN :pids]
15         ;
16         List<Inventory__c> actionNeededOnInventory = new List<Inventory__c>();
17         for (HandsMen_Order__c order : Trigger.new) {
18             if (order.HandsMen_Product__c != null && order.Status__c == 'Confirmed') {
19                 for (Inventory__c inv : mapOfInventory.values()) {
20                     if (inv.HandsMen_Product__c == order.HandsMen_Product__c) {
21                         inv.Stock_Quantity__c -= order.Quantity__c;
22                         actionNeededOnInventory.add(inv);
23                         break;
24                     }
25                 }
26             }
27         }
28         if (actionNeededOnInventory.isEmpty()) {
29             update actionNeededOnInventory;
30         }
31     }
32 }

```

- ❖ **LoyaltyStatusUpdateTrigger (after insert/update/delete/undelete on HandsMen_Order__c):**
 - Sums confirmed order totals per customer
 - Updates Total_Purchases__c
 - Dynamically adjusts Loyalty_Status__c

```

1  * trigger LoyaltyStatusUpdateTrigger on HandsMen_Order__c (after insert, after update, after delete, after undelete) {
2      Set<Id> customerIds = new Set<Id>();
3
4      // Collect affected customer IDs based on trigger context
5      if (Trigger.isInsert || Trigger.isUpdate || Trigger.isUndelete) {
6          for (HandsMen_Order__c order : Trigger.new) {
7              if (order.HandsMen_Customer__c != null) {
8                  customerIds.add(order.HandsMen_Customer__c);
9              }
10         }
11     }
12
13     if (Trigger.isUpdate || Trigger.isDelete) {
14         for (HandsMen_Order__c order : Trigger.old) {
15             if (order.HandsMen_Customer__c != null) {
16                 customerIds.add(order.HandsMen_Customer__c);
17             }
18         }
19     }
20
21     if (customerIds.isEmpty()) return;
22
23     // Map to store total amount per customer
24     Map<Id, Decimal> customerTotalMap = new Map<Id, Decimal>();
25
26     // Query confirmed orders grouped by customer
27     AggregateResult[] groupedResults = [
28         SELECT HandsMen_Customer__c, SUM(Total_Amount__c) total
29         FROM HandsMen_Order__c
30         WHERE HandsMen_Customer__c IN :customerIds AND Status__c = 'Confirmed'
31         GROUP BY HandsMen_Customer__c
32     ];

```

```

32     };
33
34     for (AggregateResult ar : groupedResults) {
35         Id custId = (Id) ar.get('HandsMen_Customer__c');
36         Decimal total = (Decimal) ar.get('total');
37         customerTotalMap.put(custId, total);
38     }
39     // Prepare customers for update
40     List<HandsMen_Customer__c> customersToUpdate = new List<HandsMen_Customer__c>();
41     for (Id custId : customerIds) {
42         Decimal total = customerTotalMap.containsKey(custId) ? customerTotalMap.get(custId) : 0;
43         String loyalty;
44
45         if (total >= 1000) {
46             loyalty = 'Gold';
47         } else if (total < 500) {
48             loyalty = 'Bronze';
49         } else {
50             loyalty = 'Silver';
51         }
52
53         HandsMen_Customer__c customer = new HandsMen_Customer__c(
54             Id = custId,
55             Total_Purchases__c = total,
56             Loyalty_Status__c = loyalty
57         );
58         customersToUpdate.add(customer);
59     }
60     if (!customersToUpdate.isEmpty()) {
61         update customersToUpdate;
62     }
63 }

```

4.3 Batch & Scheduled Apex

- **InventorySyncBatch** : Syncs inventory with external system every night at 2 AM.

The screenshot shows the Salesforce Setup page with the 'Scheduled Jobs' section expanded. A message indicates that 1% of the scheduled jobs limit (100 total) is currently used. A table lists several scheduled jobs, with the 'Daily Inventory Sync' job highlighted in red. The table includes columns for Action, Job Name, Submitted By, Submitted, Started, Next Scheduled Run, Type, and Cron Trigger ID.

Action	Job Name	Submitted By	Submitted	Started	Next Scheduled Run	Type	Cron Trigger ID
Manage Del Pause Job	Daily Inventory Sync	Negi_Sanchit	7/28/2025, 7:38 AM	7/29/2025, 2:00 AM	7/30/2025, 2:00 AM	Scheduled Apex	08egL000008gSF
Del	Loyalty_Status_Update_Flow-1	Negi_Sanchit	7/28/2025, 6:17 AM	7/29/2025, 12:02 AM	7/30/2025, 12:00 AM	Scheduled Flow	08egL000008gXW
Del	Metalytics Data Loader Job for Org: 00DgL000007C5H	User_Integration	7/16/2025, 12:58 PM	7/29/2025, 7:50 AM	7/29/2025, 7:50 AM	Autonomous Data Loader Job	08egL000007C5S
	Program Milestone Computation Cron Job	Process, Automated	7/16/2025, 12:58 PM	7/29/2025, 12:00 AM	7/29/2025, 6:59 AM	Program Milestone Computation Cron Job	08egL000007C5Q
	Program Status Update Cron Job	Process, Automated	7/16/2025, 12:58 PM	7/29/2025, 0:01 PM	7/29/2025, 5:00 AM	Program Status Update Cron Job	08egL000007C5R

5. Operational Workflow

1. Customer Registration

- Enter Name and email address
- Total Purchases is auto-calculated
- Loyalty status is auto updated by considering Total Purchases

New HandsMen Customer

Required Information

Information

HandsMen Customer Name: Rishab

Email: sanchitnegi177@gmail.com

Phone:

Loyalty Status: --None--

FirstName: Rishab

LastName: Negi

Total Purchases:

Buttons: Cancel, Save & New, Save

2. Order Creation

- User selects a customer and product
- Enters quantity
- Total Amount is auto-calculated

HandsMen Order

Details

HandsMen OrderNumber: O-0001

HandsMen Product: T-Shirt

HandsMen Customer: Rishab

Status: Confirmed

Quantity: 550

Total Amount: 137,500

Customer Email: sanchitnegi177@gmail.com

Owner: Sanchit Negi

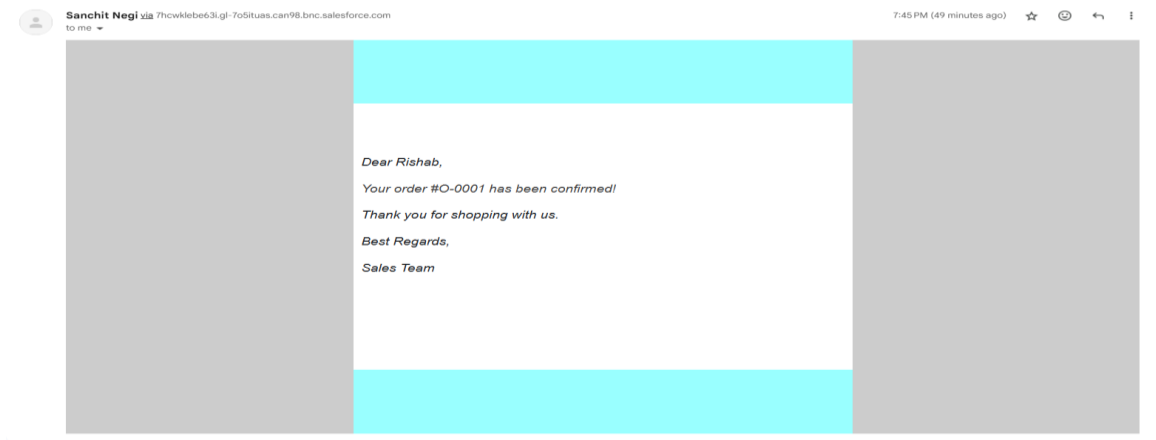
Created By: Sanchit Negi, 7/28/2025, 5:33 AM

Last Modified By: Sanchit Negi, 7/28/2025, 7:10 AM

3. Order Confirmation

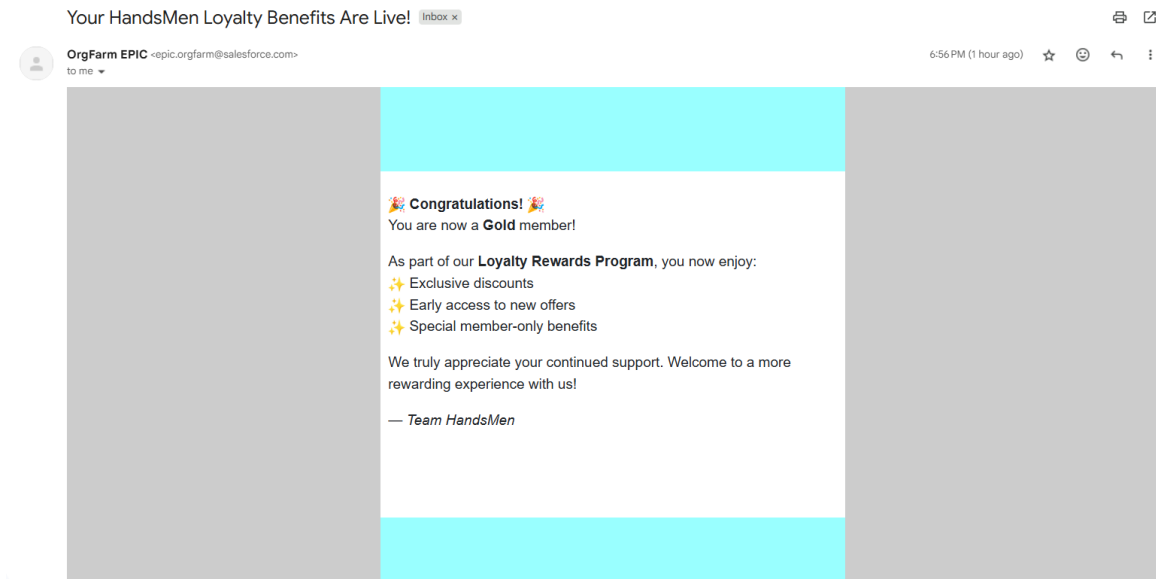
- When set to Confirmed, the order is counted toward customer's Total Purchase
- Loyalty status is re-evaluated and updated accordingly

- Email is sent to customer regarding order confirmation



4. Loyalty Logic

- Gold: > \$1000
- Silver: \$500–1000
- Bronze: < \$500
- Email is sent to customer regarding loyalty tier



6. Conclusion

The **HandsMen Threads** solution blends fashion and functionality by delivering an intelligent CRM backbone. With a custom object model, real-time automation, and clear operational flow, the system reduces manual workload, ensures customer loyalty visibility, and drives informed business decisions.

By automating key business processes such as order total calculation, loyalty tier updates, and inventory tracking, the solution enhances operational efficiency and data integrity across departments. Stakeholders can now gain instant visibility into customer behaviour, product performance, and stock availability — all from a single Salesforce platform.

Ultimately, the implementation empowers HandsMen Threads to scale customer engagement, streamline backend operations, and maintain a competitive edge in the luxury fashion market.