

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

Submitted by:

Carl Louie A. Semera

Polytechnic University of the Philippines - San Juan Campus

louie24semera@gmail.com

ABSTRACT

HandsMen Threads, a rising fashion brand specializing in men's apparel, aims to transform its internal operations by adopting Salesforce as its centralized business platform. This project focuses on building a robust data model, ensuring data integrity, and automating critical business processes to enhance customer engagement and operational efficiency. The solution involves designing an accurate data model featuring the custom objects of the project - Customers, Product, Inventory, and Marketing Campaign. The system leverages Salesforce features, such as Data Model design, Lightning App Builder, Flows, Apex, and Asynchronous Apex to streamline order management, customer loyalty tracking, stock monitoring, and bulk order processing.

Through this CRM implementation, HandsMen Threads is empowered with real time insights, improved customer relations, and optimized inventory control.

OBJECTIVE

The Main object this project is to develop a customized Salesforce solution for HandsMen Threads to streamline business operation, and enhance customer satisfaction,

The Project aims to:

- **Design a robust and scalable data model:** which stores all business related information like Customers, Orders, Product, Inventory, Loyalty Status, and Marketing.
- **Data Quality and Integrity:** Through UI validation, Flow, Automation, and Apex Triggers.
- **Enable Real Time Update:** Via Automated email updates, low stock inventory, customer interactions.
- **Deliver Customer Experience:** Target communication, loyalty program, and automation tools.

TECHNOLOGY DESCRIPTION

Apex - Salesforce's programming language, which allows developers to write custom business logic, automated processes, and handle operations that cannot be done with tools. It uses asynchronous operation, batch processing, classes, and triggers which it applied to the project.

Custom App - Collection of tabs in Salesforces for specific business purposes, which organize and display related objects, dashboards, and tools in one whole interface to support a specific workflow. Such as managing sales or orders.

Custom Object - It's a user defined database table in Salesforce, stores business specific data such as Orders, Customers, or Inventory. They can include custom fields, validation rules, relationships, and automations.

E,g

- Customer_c - Stores customer information
- Product_c - Stores product details in Handsmen
- Order_c - Stores orders in HandsMen

Data Modeling - Designing how data is structured and related in Salesforce. Inside of the Data Modeling the following data inside:

- **Customer** - The Individual or Entity purchasing products or services.
- **Order** - Record representing a purchase by a customer.
- **Order Item** - Record representing a product within an order.
- **Product** - An Item or service sold by the (Business) HandsMen.
- **Inventory** - The stocks of products available in the warehouse.
- **Marketing** - Salesforce Functionality or Processes.
- **Loyalty Program** - System that rewards customers based on purchase history.

Data Quality Tools and Rules - Fields that automatically calculate value and requirements based on other fields. The following fields and rules:

- **Formula Fields** - Fields automated calculation in Apex console
- **Validation Rules** - Prevent incorrect or incomplete data entry.
- **Required field** -

Email Alerts - Email alerts can notify customers of order confirmations, update loyalty points, or alert warehouse teams when stock is low.

Flow - Flows automate tasks such as sending emails, updating records, and calculating totals. They can be triggered by record changes, schedules, or user actions.

Profile - Profiles control what objects, fields, and features a user can see or modify. They ensure proper data access and security.

Salesforce - A cloud-based Customer Relationship Management (CRM) platform. Salesforce centralizes sales, marketing, and service operations. It provides tools for data management, automation, analytics, and customer engagement.

EXECUTION OF PROJECT PHASES

Step 0 — Prepare Developer Environment

1. Create Developer Org

- Signed up at developer.salesforce.com/signup using provided details (role = Developer, company = College Name, username = <name>@college.com).

2. Enable My Domain & Dev Settings

- Enabled My Domain in Setup and deployed it to the org for Lightning page testing.
- Turned on Debug Logs and enabled Email Deliverability to “All email” for testing notifications.

3. Create a Sandbox (if using Production)

- Created a sandbox for UAT (if required) and set up a change management plan.

Step 1 — Data Model Implementation (Objects & Fields)

1. Create Custom Objects

- Customer__c (Master record for customers)
- Product__c (SKU, Name, Price, Reorder Level)
- Inventory__c (Lookup to Product, Warehouse Location, Quantity)
- Order__c (Lookup to Customer, Order Date, Status, Total_Amount__c)
- Order_Item__c (Master-Detail to Order, Lookup to Product, Quantity, Unit_Price__c)
- Loyalty__c (Lookup to Customer, Tier, Points)

2. Define Key Fields

- Standard fields (Name, CreatedDate) plus custom fields: Stock_Threshold__c, Loyalty_Points__c, Order_Total__c, Is_Confirmed__c.

3. Set Relationships

- Order_Item__c Master-Detail → Order__c
- Lookup Order_Item__c → Product__c
- Inventory__c Lookup → Product__c

4. Configure Page Layouts & Compact Layouts

- Created page layouts for each object; added related lists (Order Items on Order page), quick actions, and lightning components where needed.

5. Create Record Types / Profiles

- If applicable, created record types (e.g., Retail vs Wholesale Orders) and updated profile permissions.

Step 2 — UI Data Integrity (Validation & UI)

1. Validation Rules

- Example: Order__c — Prevent saving if Order_Total__c <= 0 or if required shipping fields are blank.
- Inventory: Quantity__c >= 0

2. Required & Formula Fields

- Made fields required on page layouts or via validation rules (e.g., Product SKU required).
- Formula: `Order_Total__c = SUM(Order_Item__r.Quantity__c * Unit_Price__c)` (where applicable).

3. Custom Lightning Record Pages

- Used Lightning App Builder to assemble record page with:
 - Highlights panel, related lists, custom Lightning components (if any), and Flow components for inline actions.

Step 3 — Email Templates & Notification Setup

1. Create Email Templates

- Classic/Lightning Email Templates for:
 - Order Confirmation
 - Loyalty Tier Update
 - Low Stock Alert

2. Configure Email Sender

- Set Organization-Wide Email Address for notifications and verified sender.

Step 4 — Flows & Automation (Declarative Work)

1. Order Confirmation Flow (Record-Triggered Flow)

- Trigger: `Order__c` — After Save, when `Is_Confirmed__c` changes to True.
- Actions:
 1. Send Email Alert using the Order Confirmation template (merge fields for customer name, order number).
 2. Create/Update related Order Items processing (if small record set).
- Add Fault paths to log failures (create a Task or Platform Event).

2. Loyalty Update Flow (Record-Triggered Flow)

- Trigger: `Order__c` — After Save.

- Steps:
 1. Query all completed orders for the Customer in the relevant time window (e.g., 12 months) — use Get Records.
 2. Calculate new points or tier using Assignment elements.
 3. Update Loyalty__c record (create if not found).
 4. Send email notifying customers of new tier (Gold, Silver or Bronze).

3. Stock Alert Flow (Record-Triggered Flow)

- Trigger: Inventory__c — After Save.
- Condition: If Quantity__c < 5 AND prior quantity was >= 5.
- Action: Send Email Alert to Warehouse Team (email alert using template).

Step 5 — Apex & Apex Triggers (When Declarative Isn't Enough)

1. Apex Trigger: OrderOrderTotalTrigger

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;

        }

    }

}

}

```

2. StockDeductionTrigger

```

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;
    }
}
}
}

if (!inventoriesToUpdate.isEmpty()) {
    update inventoriesToUpdate;
}
}

```

3. InventoryBatchJob

```

global class InventoryBatchJob implements Database.Batchable<SObject>,
Schedulable {

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

        return Database.getQueryLocator(

            'SELECT Id, Stock_Quantity__c FROM Product__c WHERE Stock_Quantity__c < 10'

        );
    }

    global void execute(Database.BatchableContext BC, List<SObject> records) {

        List<HandsMen_Product__c> productsToUpdate = new List<HandsMen_Product__c>();

        // Cast SObject list to Product__c list

        for (SObject record : records) {

            HandsMen_Product__c product = (HandsMen_Product__c) record;

            product.Stock_Quantity__c += 50; // Restock logic

            productsToUpdate.add(product);
        }

        if (!productsToUpdate.isEmpty()) {

            try {

                update productsToUpdate;

            } catch (DmlException e) {

                System.debug('Error updating inventory: ' + e.getMessage());

            }
        }
    }
}

```

```
}

global void finish(Database.BatchableContext BC) {

    System.debug('Inventory Sync Completed');

}

// Scheduler Method

global void execute(SchedulableContext SC) {

    InventoryBatchJob batchJob = new InventoryBatchJob();

    Database.executeBatch(batchJob, 200);

    }

}
```

REAL WORLD EXAMPLE

Scenario: A Customer Places an Order

1. The customer (name steven) orders a necktie.
2. Order Created → Flow sends email confirmation.
3. The system automatically:
 - Deducts stock from Inventory
 - Recalculates Loyalty Points
 - Updates Customer's Loyalty Tier
4. If product stock becomes less than 5, an automatic email alerts the warehouse team.
5. At midnight, the bulk order update batch runs:
 - Summarizes daily orders
 - Updates financial records
 - Reconciles inventory levels

This ensures:

- Timely updates
- Accurate financials
- Proper warehouse restocking
- Improved customer satisfaction

CONCLUSION

The Salesforce implementation for HandsMen Threads has successfully transformed the organization’s data management and operational workflow. By creating a well-structured data model, we ensured that all business-critical information about customers, orders, products, inventory, marketing, and loyalty programs is stored efficiently and accessible across the organization.

Automation via Flows, Apex triggers, and scheduled batch jobs has reduced manual work, minimized human errors, and accelerated business processes such as order confirmations, inventory updates, and loyalty point calculations. The integration of email alerts and dynamic customer loyalty programs has improved engagement and strengthened relationships with repeat customers, enhancing brand value.

Overall, the project demonstrates how Salesforce can serve as a **robust CRM** and operational platform, enabling HandsMen Threads to operate efficiently while maintaining high standards of customer satisfaction and business intelligence.

RECOMMENDATION

To further enhance system performance, HandsMen Threads should consider:

- Implementing **Salesforce Service Cloud** for customer support.
- Adding **Einstein Analytics** for advanced reporting and forecasting.
- Integrating with **E-commerce platforms** for real-time order sync.
- Using **Mobile App** access for warehouse and field teams.
- Establishing **Periodic Data Audits** for long-term data hygiene.

FIGURES

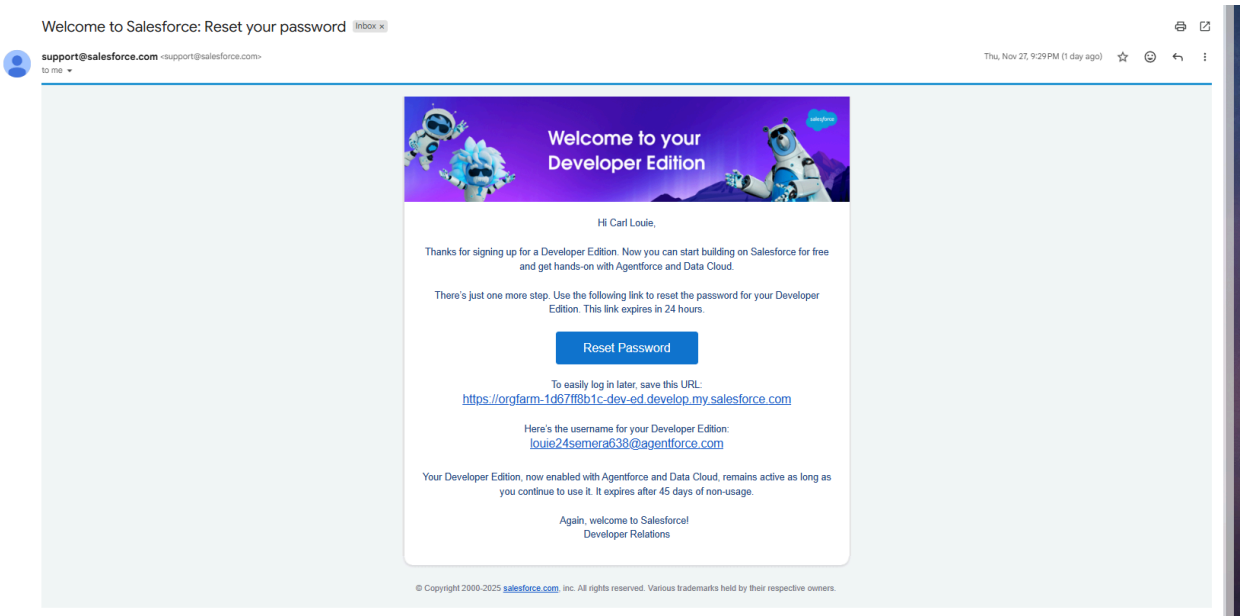


Figure 1: Developer Sign In

SETUP

Object Manager

3 Items, Sorted by Label

Schema Builder

Create

LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED	
HandsMen Customer	HandsMen_Customer_c	Custom Object		11/27/2025	✓	▼
HandsMen Order	HandsMen_Order_c	Custom Object		11/27/2025	✓	▼
HandsMen Product	HandsMen_Product_c	Custom Object		11/27/2025	✓	▼

Figure 2.1: Create Object

SETUP

Object Manager

158+ Items, Sorted by Label

Schema Builder

Create

Individual	Individual	Standard Object			
Inventory	Inventory_c	Custom Object	11/27/2025	✓	▼
Inventory Item Reservation	InventoryItemReservation	Standard Object			
Inventory Reservation	InventoryReservation	Standard Object			
Invoice	Invoice	Standard Object			
Invoice Line	InvoiceLine	Standard Object			
Lead	Lead	Standard Object			
Learning Item	LearningItem	Standard Object			
Legal Entity	LegalEntity	Standard Object			
List Email	ListEmail	Standard Object			
Location	Location	Standard Object			
Location Group	LocationGroup	Standard Object			
Location Group Assignment	LocationGroupAssignment	Standard Object			
Location Shipping Carrier Method	LocationShippingCarrierMethod	Standard Object			
Macro	Macro	Standard Object			
Marketing Campaign	Marketing_Campaign_c	Custom Object	11/27/2025	✓	▼

Figure 2.2: Create Object

SETUP

Tabs

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.

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	Inventories	Building	
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	

Figure 3: Tabs

SETUP

Lightning Experience App Manager

New Lightning AppNew External Client App

28 items • Sorted by App Name • Filtered by All appmenuitems - TabSet Type, App Type

	App Name ↑	Developer Name	Description	Last Modified Date	App Type	Visible in Li...	
1	All Tabs	AllTabSet		11/23/2025, 8:57 PM	Classic		
2	Analytics Studio	Insights	Build CRM Analytics dashboards and apps	11/23/2025, 8:57 PM	Classic	✓	
3	App Launcher	AppLauncher	App Launcher tabs	11/23/2025, 8:57 PM	Classic	✓	
4	Approvals	Approvals	Manage approvals and approval flows	11/23/2025, 8:57 PM	Lightning	✓	
5	Automation	FlowsApp	Automate business processes and repetitive tasks.	11/23/2025, 9:06 PM	Lightning	✓	
6	Bolt Solutions	LightningBolt	Discover and manage business solutions designed for your industry.	11/23/2025, 8:57 PM	Lightning	✓	
7	Community	Community	Salesforce CRM Communities	11/23/2025, 8:57 PM	Classic	✓	
8	Content	Content	Salesforce CRM Content	11/23/2025, 8:57 PM	Classic	✓	
9	Data Cloud	Audience360	Build a thorough and complete understanding of your customers.	11/23/2025, 8:57 PM	Lightning	✓	
10	Data Manager	DataManager	Use Data Manager to view limits, monitor usage, and manage recipes.	11/23/2025, 8:57 PM	Lightning	✓	
11	Developer Edition	Developer_Edition	Welcome to your Developer Edition Org.	11/23/2025, 9:31 PM	Lightning (Managed)	✓	
12	Digital Experiences	SalesforceCMS	Manage content and media for all of your sites.	11/23/2025, 8:57 PM	Lightning	✓	
13	HandsMen Threads	HandsMen_Threads	Give a meaningful description	11/27/2025, 6:18 AM	Lightning	✓	

Figure 4: App 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

11 Items, Sorted by Field Label

Q 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)		

Figure 5: Fields

SETUP > OBJECT MANAGER

HandsMen Order

Details

Validation Rules

Fields & Relationships

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

1 Items, Sorted by Rule Name

New

RULE NAME	ERROR LOCATION	ERROR MESSAGE	ACTIVE	MODIFIED BY
Total_Amount	Total Amount	Please Enter Correct Amount	✓	Carl Louie Semera, 11/27/2025, 7:12 AM

Figure 6: Validation

<input type="checkbox"/>	Edit	Mikaelson_Dan	dmika	louie24semera7812@gmail.com	Marketing	✓	Platform 1
<input type="checkbox"/>	Edit	Mikaelson_Kol	kmika	louie24semera1024@gmail.com	Inventory		Platform 1
<input type="checkbox"/>	Edit	Mikaelson_Niklaus	nmika	louie24semera55@gmail.com	Sales	✓	Platform 1

Figure 7: Users Mikaelsons

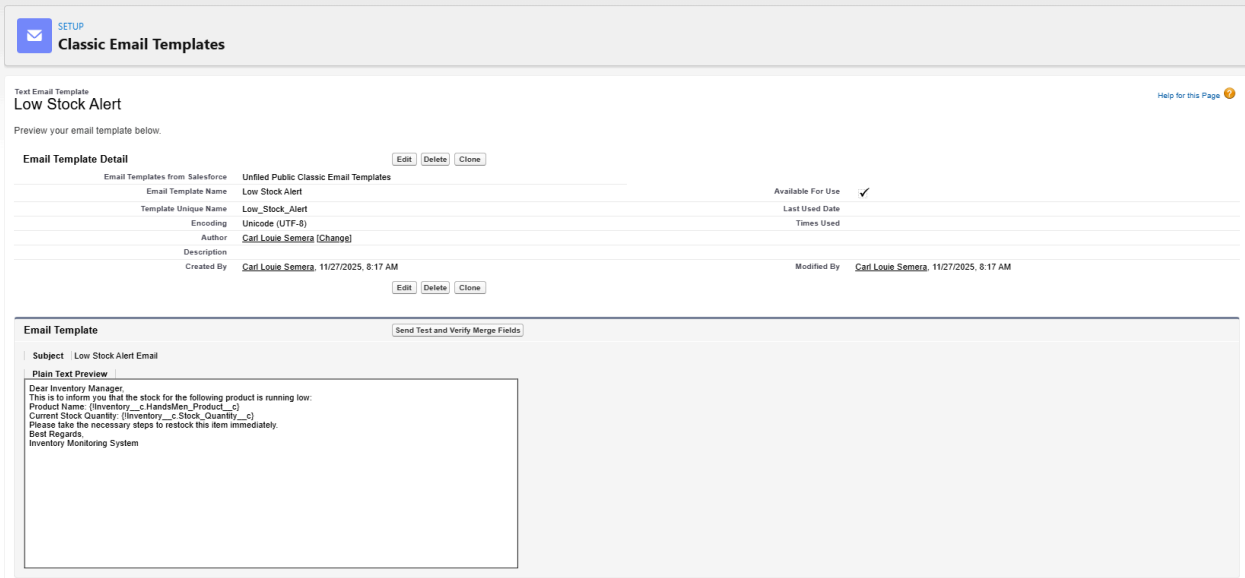


Figure 8.1 Low Stock Alert Template

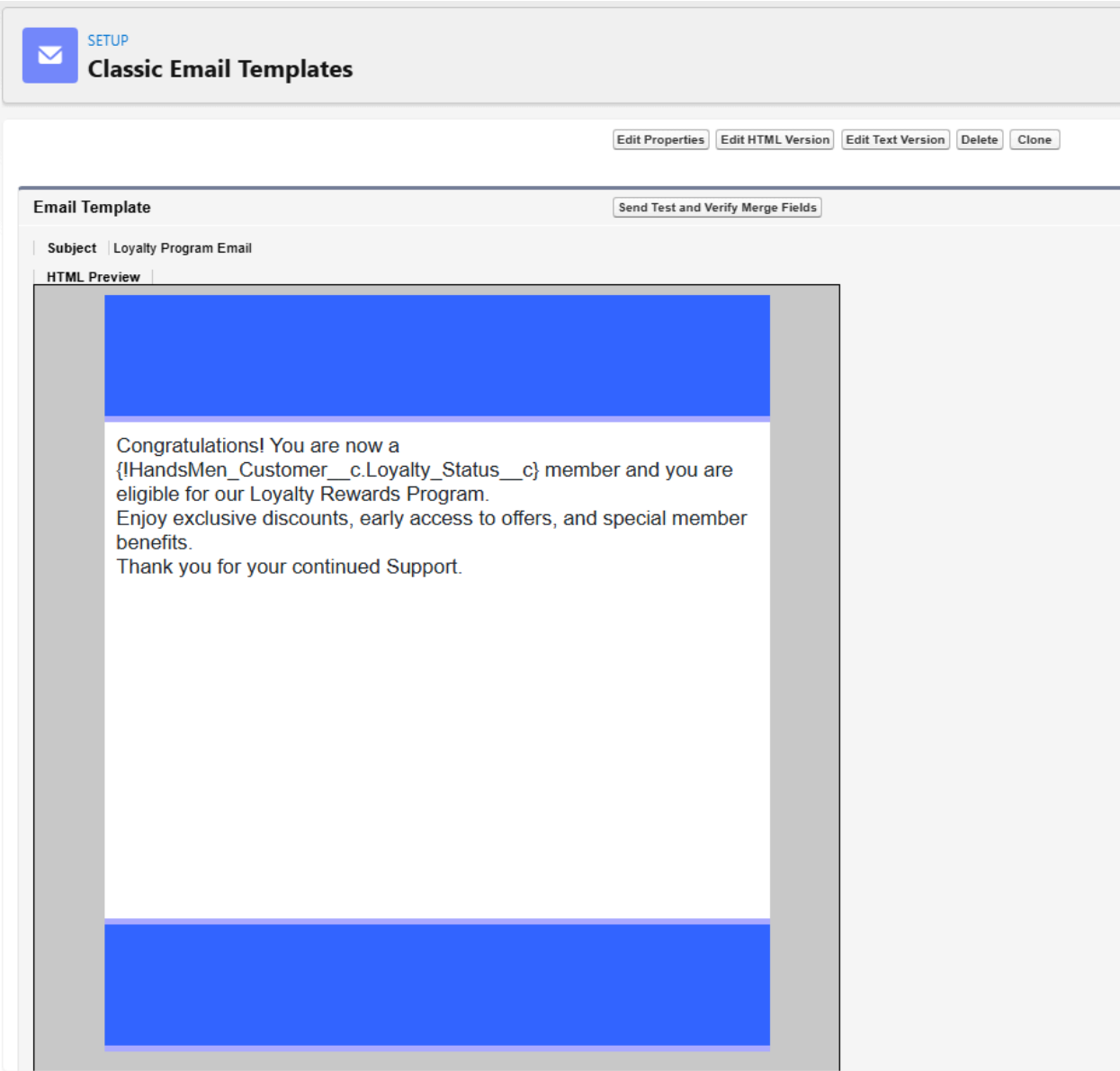


Figure 8.2 Loyalty Email Template

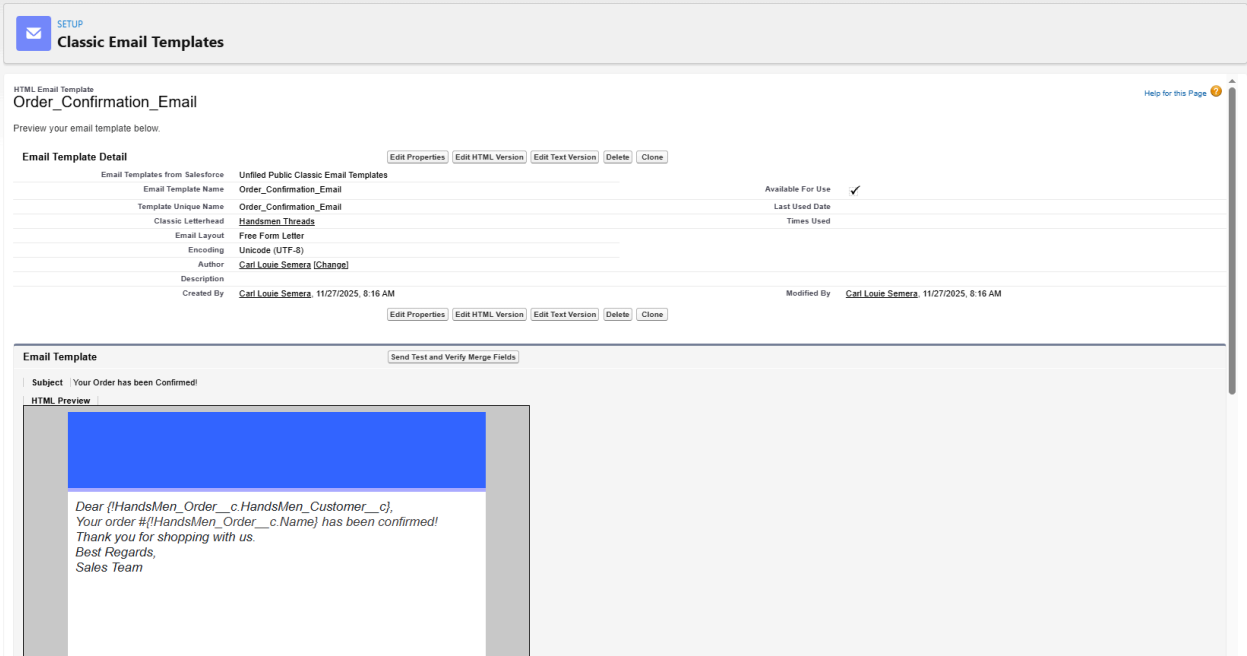


Figure 8.3 Order Email Template

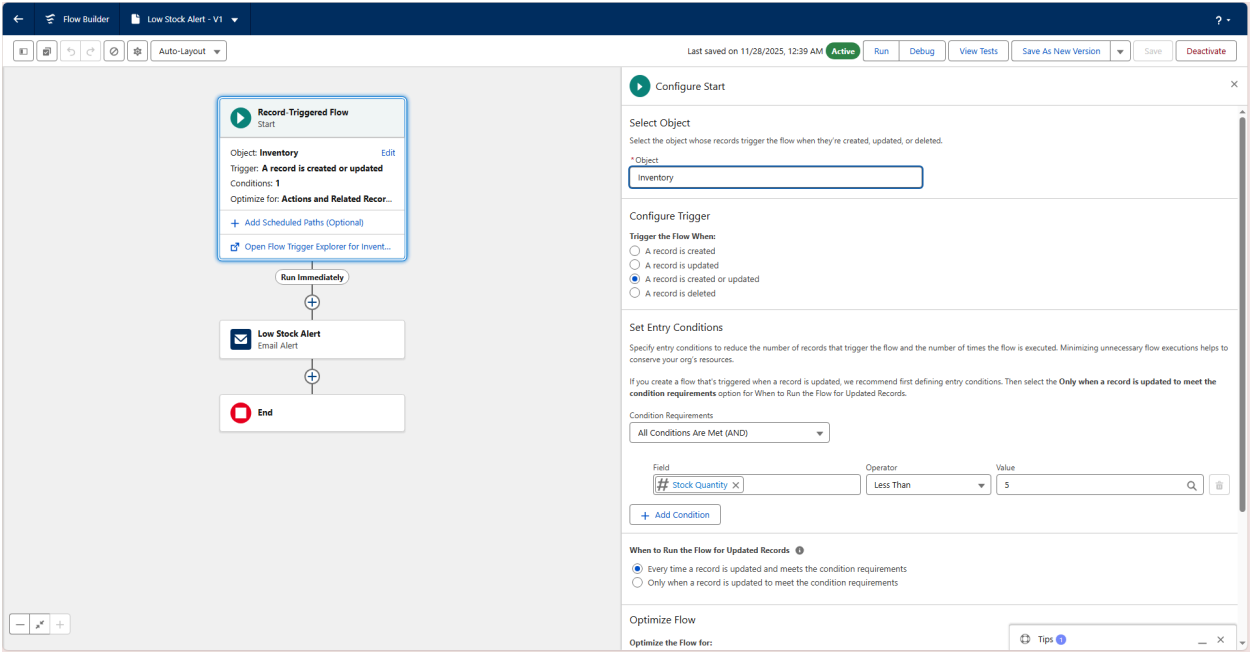


Figure 9.1 Low Stock Alert Flow

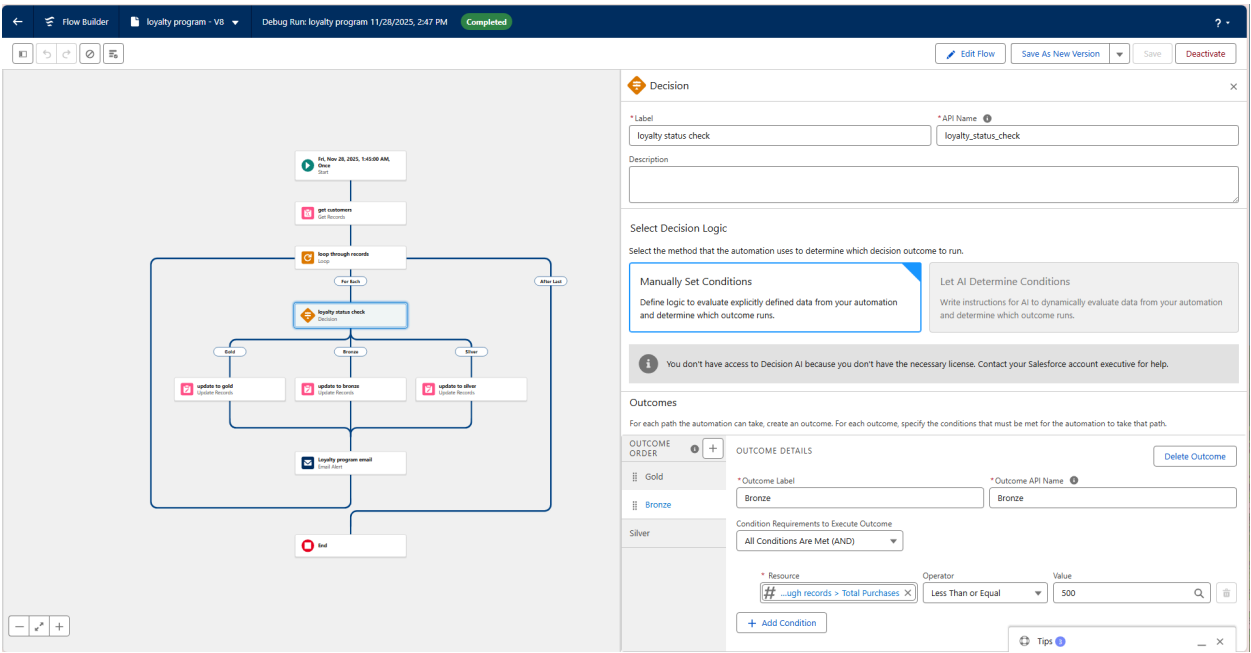


Figure 9.2 Loyalty Flow

Flow Builderorder confirmation - V1

Last saved on 11/28/2025, 12:27 AMActiveRunDebugView TestsSave As New VersionSaveDeactivate

Record-Triggered FlowStart

Object: HandsMen OrderEdit

Trigger: A record is updated

Conditions: 1

Optimize for: Actions and Related Recor...

+ Add Scheduled Paths (Optional)

Open Flow Trigger Explorer for Hands...

Run Immediately

order confirmationEmail Alert

End

Configure Start

Select Object

Select the object whose records trigger the flow when they're created, updated, or deleted.

* Object

HandsMen Order

Configure Trigger

Trigger the Flow When:

☐ A record is created

☒ A record is updated

☐ A record is created or updated

☐ A record is deleted

Set Entry Conditions

Specify entry conditions to reduce the number of records that trigger the flow and the number of times the flow is executed. Minimizing unnecessary flow executions helps to conserve your org's resources.

If you create a flow that's triggered when a record is updated, we recommend first defining entry conditions. Then select the **only when a record is updated to meet the condition requirements** option for When to Run the Flow for Updated Records.

Condition Requirements

All Conditions Are Met (AND)

FieldOperatorValue

StatusXEqualsA ConfirmedX

+ Add Condition

When to Run the Flow for Updated Records ⓘ

☐ Every time a record is updated and meets the condition requirements

☒ Only when a record is updated to meet the condition requirements

Optimize Flow

Optimize the Flow for:

Tips

Figure 9.3 Order Flow

```

OrderTotalTrigger.apex | StockDeductionTrigger.apex | InventoryBatchJob.apex
Code Coverage: None | API Version: 65
1  global class InventoryBatchJob implements Database.Batchable<SObject>, Schedulable {
2
3  global Database.QueryLocator start(Database.BatchableContext BC) {
4
5      return Database.getQueryLocator(
6
7          'SELECT Id, Stock_Quantity__c FROM Product__c WHERE Stock_Quantity__c < 10'
8      );
9  };
10
11 }
12
13 global void execute(Database.BatchableContext BC, List<SObject> records) {
14
15     List<HandsMen_Product__c> productsToUpdate = new List<HandsMen_Product__c>();
16
17     // Cast SObject list to Product__c list
18
19     for (SObject record : records) {
20
21         HandsMen_Product__c product = (HandsMen_Product__c) record;
22
23         product.Stock_Quantity__c += 50; // Restock logic
24
25         productsToUpdate.add(product);
26
27     }
28
29     if (!productsToUpdate.isEmpty()) {
30
31         try {
32
33             update productsToUpdate;
34
35         } catch (DmlException e) {
36
37             System.debug('Error updating inventory: ' + e.getMessage());
38
39         }
40
41     }
42
43 }
44
45 global void finish(Database.BatchableContext BC) {
46
47     System.debug('Inventory Sync Completed');
48
49 }
50
51 // Scheduler Method
52
53 global void execute(SchedulableContext SC) {
54
55     InventoryBatchJob batchJob = new InventoryBatchJob();
56
57     Database.executeBatch(batchJob, 200);
58
59 }
60
61 }

```

Figure 10.1 Apex Batch Inventory

```
OrderTotalTrigger.apxt | StockDeductionTrigger.apxt | InventoryBatchJob.apxc
Code Coverage: None | API Version: 65

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

Figure 10.2 Apex Trigger Order Total

```
OrderTotalTrigger.apxt | StockDeductionTrigger.apxt | InventoryBatchJob.apxc
Code Coverage: None | API Version: 65

1 trigger StockDeductionTrigger on HandsMen_Order__c (after insert, after update) {
2     Set<Id> productIds = new Set<Id>();
3
4     for (HandsMen_Order__c order : Trigger.new) {
5         if (order.Status__c == 'Confirmed' && order.HandsMen_Product__c != null) {
6             productIds.add(order.HandsMen_Product__c);
7         }
8     }
9
10    if (productIds.isEmpty()) return;
11
12    // Query related inventories based on product
13    Map<Id, Inventory__c> inventoryMap = new Map<Id, Inventory__c>(
14        [SELECT Id, Stock_Quantity__c, HandsMen_Product__c
15         FROM Inventory__c
16         WHERE HandsMen_Product__c IN :productIds]
17    );
18
19    List<Inventory__c> inventoriesToUpdate = new List<Inventory__c>();
20
21    for (HandsMen_Order__c order : Trigger.new) {
22        if (order.Status__c == 'Confirmed' && order.HandsMen_Product__c != null) {
23            for (Inventory__c inv : inventoryMap.values()) {
24                if (inv.HandsMen_Product__c == order.HandsMen_Product__c) {
25                    inv.Stock_Quantity__c -= order.Quantity__c;
26                    inventoriesToUpdate.add(inv);
27                    break;
28                }
29            }
30        }
31    }
32
33    if (!inventoriesToUpdate.isEmpty()) {
34        update inventoriesToUpdate;
35    }
36 }
```

Figure 10.3 Apex Trigger Deduction Stocks