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

**FINAL PROJECT REPORT**

Submitted in partial fulfillment of the  
SmartInternz Virtual Internship Program (July 2025)  
In association with Salesforce Platform

**Project Role:**

Salesforce Developer | LWC Developer | Apex Programmer | Project Lead

**Submitted by:**

KARTHIKEYAN I  
B.E. Computer Science Engineering  
Mahendra Engineering College, Namakkal  
Tamil Nadu, India

**Submission Date:**

July 2025

**Guided by:**

SmartInternz Mentors and Faculty Coordinator  
  
  
  
  
  
  
  
**Modules Covered**

|  |  |
| --- | --- |
| 1 | Use Case |
| 2 | User Story |
| 3 | Salesforce Credentials Setup |
| 4 | Data Management-Object |
| 5 | Data Management-Tabs |
| 6 | Data Management-App Manager |
| 7 | Data Management-Fields |
| 8 | Data Configuration |
| 9 | Data Security – Profiles |
| 10 | Data Security – Roles |
| 11 | Data Security – Users |
| 12 | Data Security – Permission Set |
| 13 | Email Template |
| 14 | Flows (Automation) |
| 15 | Automation using Apex |
| 16 | Batch Jobs |

**UseCase**   
  
HandsMen Threads, a dynamic organization in the fashion industry, is embarking on a Salesforce project designed to revolutionize their data management and enhance customer relations. The project involves building a robust data model tailored to store all pertinent business data, ensuring a seamless flow of information across the organization.

A key aspect of this project is the maintenance of data integrity directly from the user interface (UI). This feature will safeguard the accuracy and consistency of the data, which is crucial for informed decision-making and reliable business operations.The project will integrate several new processes into the business workflow to improve customer service and operational efficiency:

Automated Order Confirmations: Post-order confirmation, customers will receive an email update, fostering engagement and strengthening customer relations.

Dynamic Loyalty Program: Customer loyalty statuses will be updated based on purchase history, enabling personalized rewards and promoting repeat business.

Proactive Stock Alerts: When stock levels drop below five units, automatic emails will notify the warehouse team, ensuring timely restocking and preventing stockouts.

Scheduled Bulk Order Updates: Daily midnight, the system will process bulk orders, updating financial records and adjusting inventory, ensuring accurate stock levels for daily operations.

What I learn :

* Data Modelling
* Data Quality
* Lightning App Builder
* Record Triggered Flows
* Apex and Apex Triggers

**User Story**

**Project Phases**

**Phase 1: Architecture & Planning**

* Define objects, fields, relationships, formula fields.
* Establish validation rules, flows, Apex triggers, batch jobs.
* Design email templates for notifications and customer communication.

**Phase 2: Development**

* Object and field creation.
* Implement automation (flows, process builders, Apex triggers).
* Set up data security and sharing rules.
* Develop batch jobs for scheduled processing.
* Configure email templates and notifications.

**Phase 3: Testing & QA**

* Unit testing of objects and automation.
* End-to-end testing with sample data.
* Performance testing and security checks.

**Phase 4: Deployment & Training**

* Deploy to production.
* Train users on new functionality.
* Post-go-live support and monitoring

**Deliverable:**

Solution Design Document including Object Model, ERD, and Automation Strategy.

### **Salesforce Credentials Setup**

To build and deploy the application using Salesforce, we followed these essential steps to set up the developer environment and ensure secure access to the platform.

#### ****Developer Account Creation****

We started by creating a **Salesforce Developer Account** via the official Salesforce Developers portal ([developer.salesforce.com](https://developer.salesforce.com/)). This account provides:

* Free access to the **Salesforce Lightning Platform**
* Tools for developing, testing, and deploying apps
* Preconfigured sandbox environment for development and learning

**Purpose:** This account serves as our primary development environment where we built and tested our Salesforce application.

#### ****Account Verification****

After signing up, we received a **verification email** to confirm the registration. This step is essential to:

* Activate the developer account
* Set up a secure password
* Enable access to the Salesforce dashboard

**Purpose:** This step ensures that the account is authenticated and ready for development activities.

#### ****Salesforce Credentials Setup**** Once the account was verified, we configured the credentials necessary for CLI and browser-based login, including:

* **Username (email used during sign-up)**
* **Security Token** (used for connecting Salesforce with external tools like VS Code or Postman)

## Salesforce Credentials Setup and Object Configuration

### ****Salesforce Credentials Setup****

To begin development in the Salesforce platform, the first step is setting up access credentials. This includes:

* **Developer Account Creation**:  
  A free [Salesforce Developer Account](https://developer.salesforce.com/signup) was created. This account provides full access to core Salesforce features such as Object Manager, App Builder, Apex Classes, Flows, and more.
* **Account Verification**:  
  After registration, the email provided was verified. This step is essential for securing access and activating the Salesforce org. Once verified, the Salesforce Lightning Experience environment was launched.

### ****Custom Object Configuration****

Salesforce uses a data model based on **Objects** (like tables in a database). In this project, several custom objects were created to manage business-specific data.

Below are the custom objects and their roles:

#### ****1. HandsMen Customer****

Stores customer profile information including:

* Name
* Email ID
* Phone Number
* Shipping & Billing Address

Used for identifying and managing customers across orders and campaigns.

#### ****2. HandsMen Product****

Contains all product details:

* Product Code
* Description
* Price
* Availability

#### ****3. HandsMen Order****

Captures order transaction details:

* Product(s) ordered
* Quantity
* Order Date
* Customer Link
* Order Status (Pending, Shipped, Delivered)

Acts as a junction between customer and product data.

#### ****4. Inventory****

Manages:

* Available stock
* Product name
* Quantity
* Reorder levels

Supports inventory tracking and triggers alerts for restocking.

#### ****5. Marketing Campaign****

Used for campaign tracking and performance evaluation:

* Campaign Name
* Budget
* Timeline
* Targeted Customers

### Data Management – Tabs

In our Salesforce development project, **custom tabs** were created to enhance user interaction and improve accessibility for managing custom data objects.

#### ➤ Tab – HandsMen Customer

A **custom tab** named HandsMen Customer was created to represent and interact with the custom object designed to store information related to customers using the HandsMen service. This tab allows easy navigation and data entry for records associated with this object.

#### ➤ Purpose of the Tab:

* To provide a **user interface** for the custom object.
* To allow **manual record entry**, update, and deletion of customer data.
* To enable **data visualization** and report creation directly linked to HandsMen Customer records.

#### ➤ Create Tabs:

The "Create Tabs" option in Salesforce Setup was used to:

1. **Select the custom object** (HandsMen Customer).
2. **Assign a tab style and icon** for easy identification.
3. **Set visibility** for different profiles to control user access.

This customization supports better **data organization**, **access control**, and **navigation** for CRM operations within the Salesforce ecosystem.

### App – HandsMen Threads

The **HandsMen Threads** app is a **custom Salesforce Lightning App** developed to manage and streamline the operations of the HandsMen service platform. This app integrates various Salesforce features—custom objects, tabs, and automation—to provide a centralized workspace for administrators and service agents.

#### ➤ Purpose of the App:

* To serve as a **dedicated environment** for managing customers, orders, and services offered by HandsMen Threads.
* To **organize tabs and objects** specific to the business operations under a single app for ease of use.
* To improve **navigation**, **user experience**, and **workflow efficiency** within the Salesforce platform.

#### ➤ Features Included:

* **Custom Tabs** like:
  + HandsMen Customer – manage customer records.
  + Service Request – track customer issues or requests.
* **App Branding**: The app is customized with an appropriate **logo, name, and navigation style** to reflect the brand identity of HandsMen Threads.
* **User Profiles and Access Control**: The app supports role-based access to ensure that only authorized users can access and modify specific records.

#### ➤ Technology Stack:

* Salesforce Lightning App Builder
* Custom Objects and Fields
* Role & Profile Management
* Tab Customization

#### ➤ Business Value:

The HandsMen Threads app ensures that customer service operations are **centralized, automated, and scalable**, helping the team to manage customer relationships efficiently and professionally  
  
Data Management – Fields Creation in Salesforce Objects

As part of building a robust CRM system in the **HandsMen Threads** application, we implemented essential **data management fields** in various **custom Salesforce objects**. These fields enable accurate data collection, tracking, and relationship management across customers and services.

### Fields Created in Custom Objects:

#### 1. ****Email Field****

* **Type**: Email
* **Purpose**: To store the official email address of the customer.
* **Validation**: Standard email format validation to ensure proper input.
* **Usage**: Helps in sending service updates, confirmations, and promotional emails.

#### 2. ****Phone Field****

* **Type**: Phone
* **Purpose**: To store the contact number of the customer.
* **Usage**: For service coordination, updates via SMS, or direct support.

#### 3. ****Loyalty Status****

* **Type**: Picklist
* **Options**: Silver, Gold, Platinum
* **Purpose**: To classify customers based on their service usage and reward level.
* **Usage**: Helps in offering personalized discounts or priority services.

### Lookup Relationship

* **Created Lookup Field**:
  + **Example**: Customer (Lookup to HandsMen\_Customer\_\_c) in HandsMen\_Order\_\_c object.
* **Purpose**: To **link related records** across objects.
* **Usage**: Allows referencing a customer record directly from the order or service object, enabling relational data access.

### Formula Fields

* **Created Formula Field** Example:
  + **Field**: Customer Full Info
  + **Formula**: Name + " - " + Phone
* **Purpose**: Automatically compute values without manual entry.
* **Benefits**: Reduces redundancy and keeps data up-to-date.

### Remaining Fields Created in Objects

Each object in the system was enhanced with necessary fields for complete functionality. For example:

#### ➤ ****In HandsMen\_Customer\_\_c****:

* Name (Text)
* Email (Email)
* Phone (Phone)
* Loyalty\_Status\_\_c (Picklist)
* Total\_Orders\_\_c (Number)

#### ➤ ****In HandsMen\_Order\_\_c****:

* Order\_Date\_\_c (Date)
* Service\_Type\_\_c (Picklist)
* Order\_Status\_\_c (Picklist)
* Customer\_\_c (Lookup to Customer)
* Total\_Amount\_\_c (Currency)

#### ➤ ****In HandsMen\_Service\_\_c****:

* Service\_Name\_\_c (Text)
* Price\_\_c (Currency)
* Duration\_\_c (Number in minutes)
* Available\_Slots\_\_c (Number)

### Business Impact:

The creation of these fields and relationships:

* Ensures **data consistency** and **validation**.
* Enhances **data accessibility** through relationships.
* Enables **custom reporting** and **workflow automation**.
* Supports **future scalability** of the app.

## Data Configuration – Validation Rules

### What are Validation Rules?

Validation Rules in Salesforce are used to ensure that **data entered into records meets the required standards** before it can be saved. These rules help in maintaining **data integrity**, **completeness**, and **consistency** across objects.

### Purpose in Our Project

In our HandsMen Threads Salesforce application, validation rules are used to:

* **Prevent incomplete or incorrect data** entries in the HandsMen Customer, Order, and Product objects.
* **Ensure business logic is followed** when users create or update records.
* Improve the **accuracy** and **quality** of the data stored in Salesforce.

### Examples of Validation Rules Implemented:

#### 1. ****Mandatory Email Format Check****

Ensures that the Email field contains a valid email format.

NOT(REGEX(Email\_\_c, "^[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,4}$"))

#### 2. ****Phone Number Length Check****

Validates that the Phone number has exactly 10 digits.

LEN(Phone\_\_c) <> 10

#### 3. ****Loyalty Status Must Be Selected****

Prevents saving a customer without assigning a loyalty status.

ISBLANK(Loyalty\_Status\_\_c)

#### 4. ****Date Must Not Be in the Past****

For order date fields to ensure future scheduling.

Order\_Date\_\_c < TODAY()

### Create Remaining Validation Rules

Other validation rules were added based on project requirements:

* **Order Quantity must be greater than 0**
* **Discount should not exceed 50%**
* **Customer Name should not be left blank**
* **Product Stock should not be negative**

These rules ensure that **critical business validations** are enforced during the data entry process.

**Data Security – Profiles**

### ****Overview****

In Salesforce, **Profiles** are used to control object-level and field-level access for users. Proper profile configuration is essential to enforce **data security**, ensuring that users only access data relevant to their roles.

### ****Profile Created: Sales****

A custom profile named **“Sales”** was created to define the permissions and access levels for users working in the sales department of HandsMen Threads.

### ****Key Configurations of the Sales Profile:****

* **Object Permissions:**
  + **Read, Create, Edit** access on core objects such as:
    - **Customer**
    - **Orders**
    - **Products**
  + **View Only** access to reporting objects for performance tracking.
  + **No Delete** permissions to avoid accidental data loss.
* **Field-Level Security:**
  + Restricted access to sensitive fields like:
    - Customer financial details
    - Internal pricing strategies
  + Granted full access to contact-related fields like:
    - Email
    - Phone
    - Loyalty Status
* **Tab Settings:**
  + Tabs like **Orders**, **Customers**, and **Opportunities** were made **Visible and Default On**.
  + Non-relevant tabs like **Developer Console** or **Administration** were hidden.
* **Record Types and Page Layouts:**
  + Assigned relevant record types specific to Sales processes.
  + Customized page layouts to streamline the user interface and improve efficiency.
* **Login and Access Restrictions:**
  + Defined **login hours** and **IP ranges** to enhance security.
  + Ensured only authorized sales personnel could log in from specified networks.

### ****Purpose and Benefits:****

* Prevent unauthorized data access.
* Improve data accuracy and trust.
* Align user access with business responsibilities.
* Protect customer privacy and internal records.

## ****Data Security – Roles****

### ****Overview****

In Salesforce, **Roles** determine the level of visibility users have into the organization's data. Roles work in conjunction with sharing rules and profiles to ensure proper **data access hierarchy** and maintain organizational structure.

### ****Role Created: Sales****

A role named **“Sales”** was created to define the data access level for users handling customer interactions, orders, and lead management at **HandsMen Threads**.

### ****Key Features of the Sales Role:****

* **Hierarchy Level:**
  + The **Sales** role is positioned below the **Sales Manager** role in the role hierarchy.
  + This allows **Sales Managers** to view data owned by users in the **Sales** role, ensuring proper supervision and reporting.
* **Data Access:**
  + Users assigned to the **Sales** role can:
    - **View and edit** records they own.
    - **Collaborate** on shared records related to customers and orders.
    - Access data relevant to their team or region, based on sharing settings.
* **Record Ownership:**
  + Sales users are typically the **owners** of:
    - **Leads**
    - **Opportunities**
    - **Accounts**
    - **Orders**
  + This ownership enables accurate tracking of performance and follow-ups.
* **Sharing Rules Integration:**
  + Custom sharing rules were applied to allow users in the **Sales** role to access opportunities and cases assigned to other team members if necessary for collaboration.

### ****Benefits of Role Implementation:****

* Maintains **data confidentiality** by preventing unauthorized access.
* Supports **team-based selling** by allowing data sharing within the role.
* Provides **clear reporting structure** and simplifies management oversight.
* Enhances **data integrity** and operational efficiency.

**Data Security – Users**

### ****Overview****

In Salesforce, **Users** are individuals who have access to the system through a unique username and profile. Effective user management is essential to ensure **data security**, **controlled access**, and **workflow efficiency**.

At **HandsMen Threads**, users are configured based on their roles and responsibilities, and appropriate **profiles**, **roles**, and **permissions** are assigned to them to maintain a secure and organized working environment.

**User Creation and Configuration**

* Each user is assigned:
  + A **Username** (unique email format)
  + A **Profile** (determines object-level access)
  + A **Role** (defines record-level access)
  + A **License** (determines feature access)

| ****User Name**** | ****Email**** | ****Profile**** | ****Role**** | ****License**** |
| --- | --- | --- | --- | --- |
| Karthikeyan I | [karthik@example.com](mailto:karthik@example.com) | Sales Profile | Sales | Salesforce Platform |
| Hari Prasanth | [hari@example.com](mailto:hari@example.com) | Sales Manager | Sales Mgr | Salesforce Platform |
| Priya S | [priya@example.com](mailto:priya@example.com) | Support Profile | Support | Salesforce Platform |

**Security Controls Applied to Users:**

* **Login Restrictions:**
  + Users can only log in during specified hours and from specific IP ranges if needed.
* **Password Policies:**
  + Strong password requirements are enforced (minimum length, complexity).
  + Password expiration and lockout policies are configured for safety.
* **Two-Factor Authentication (2FA):**
  + 2FA is enabled for added security on user login.
* **Field-Level Security:**
  + Users only see or edit the fields necessary for their job function.
* **Audit Trail:**
  + All user actions are logged and can be monitored by administrators.

**Data Security – Permission Sets**

### ****Overview****

**Permission Sets** in Salesforce are used to **grant additional permissions** to users without modifying their profiles. This flexible security model allows specific users to perform actions beyond their profile limitations while maintaining system integrity and control.

At **HandsMen Threads**, permission sets are strategically assigned to enhance security and streamline user access to features as per their responsibilities.  
  
**Why Use Permission Sets?**

* To **grant specific permissions** (like access to objects, fields, or system settings) to a subset of users.
* To avoid creating multiple profiles for minor permission changes.
* To implement **least privilege** access model effectively.

**Created Permission Sets**

#### ****Permission Set: View Orders****

* **Purpose:** Allows users to **read-only access** to HandsMen\_Order\_\_c records.
* **Assigned To:** Support Team
* **Object Permissions:**
  + HandsMen\_Order\_\_c: Read Only
* **Field-Level Access:**
  + All fields visible, but not editable.

#### ****Permission Set: Manage Products****

* **Purpose:** Grants access to **create, edit, and delete** HandsMen\_Product\_\_c records.
* **Assigned To:** Product Manager
* **Object Permissions:**
  + HandsMen\_Product\_\_c: Read, Create, Edit, Delete

#### ****Permission Set: Export Data****

* **Purpose:** Allows exporting of reports and dashboards.
* **Assigned To:** Sales Managers
* **System Permissions:**
  + Export Reports
  + Manage Personal Reports and Dashboards

#### ****Assignment and Management****

* Permission sets were assigned through:
  + **Setup → Permission Sets → Assign Users**
* Each user can have **multiple permission sets**.
* They are **independent of profiles** but work **in conjunction** with them.

### ****Security Benefits****

* Enhances **granular control** over access permissions.
* Improves **compliance** by separating access rights.
* Simplifies **role changes and user promotions** without altering profiles.
* Helps in **auditing and tracking** permission changes more effectively.

#### 

**Email Templates & Email Alerts**

### Overview

Email Templates in Salesforce are **predefined, reusable messages** that can be sent as part of an automation process such as Workflow Rules, Process Builders, or Flows. Email Alerts use these templates to notify users or customers automatically.

In **HandsMen Threads**, we’ve created several email templates to automate order processing communication with customers and internal users.

1. **Order Confirmation Email Template**

**Template Name:** Order\_Confirmation\_Template  
**Template Type:** HTML (with letterhead)  
**Available For Use:** Yes

**Subject:**  
Thank You for Your Order, {!HandsMen\_Order\_\_c.Customer\_Name\_\_c}!

Dear {!HandsMen\_Order\_\_c.Customer\_Name\_\_c},

Thank you for placing an order with HandsMen Threads!

Here are your order details:

- Order Number: {!HandsMen\_Order\_\_c.Name}

- Product: {!HandsMen\_Order\_\_c.HandsMen\_Product\_\_r.Name}

- Quantity: {!HandsMen\_Order\_\_c.Quantity\_\_c}

- Total Amount: ₹{!HandsMen\_Order\_\_c.Total\_Amount\_\_c}

- Status: {!HandsMen\_Order\_\_c.Status\_\_c}

Your order is being processed. We will notify you once it's shipped.

Thank you for choosing HandsMen Threads!

2. **Order Shipment Email Template**

**Template Name:** Order\_Shipped\_Template  
**Subject:**  
Your Order #{!HandsMen\_Order\_\_c.Name} Has Been Shipped!

**Email Body:**

**Hi {!HandsMen\_Order\_\_c.Customer\_Name\_\_c},**

**We’re excited to let you know that your order #{!HandsMen\_Order\_\_c.Name} has been shipped.**

**Tracking Number: {!HandsMen\_Order\_\_c.Tracking\_ID\_\_c}**

**Carrier: {!HandsMen\_Order\_\_c.Shipping\_Company\_\_c}**

**You can expect delivery within 3-5 business days.**

**Thanks for shopping with us!**

**Regards,**

**HandsMen Threads**

### 3. ****Order Cancellation Email Template****

**Template Name:** Order\_Cancelled\_Template  
**Subject:**  
Your Order #{!HandsMen\_Order\_\_c.Name} Has Been Cancelled

**Email Body:**

Dear {!HandsMen\_Order\_\_c.Customer\_Name\_\_c},

We regret to inform you that your order #{!HandsMen\_Order\_\_c.Name} has been cancelled due to the following reason:

Reason: {!HandsMen\_Order\_\_c.Cancellation\_Reason\_\_c}

If you have any questions or would like to reorder, feel free to reach out to our support team.

### 4. ****Email Alert Setup****

**Email Alert Name:** Send Order Confirmation Alert  
**Triggered From:** Workflow Rule / Flow  
**Recipients:**

* Customer Email (from Email\_\_c field in HandsMen\_Order\_\_c)
* Sales Team Email (for internal tracking)

**Template Used:** Order\_Confirmation\_Template  
**Object:** HandsMen\_Order\_\_c

**Conditions for Triggering:**

* Status\_\_c = "Confirmed"

### Flows

**Flows** in Salesforce are powerful automation tools used to collect data, automate processes, and update records without writing any code. In our **HandsMen Threads** project, we utilized **Record-Triggered Flows** and **Email Alert Flows** to streamline operations related to order processing and customer communication.

#### Flow 1: Order Confirmation Flow

* **Type**: Record-Triggered Flow
* **Object**: HandsMen\_Order\_\_c
* **Trigger**: After a record is created and the status is set to "Confirmed"
* **Action**: Automatically sends an order confirmation email to the customer using a predefined **Email Template**
* **Purpose**: Improves customer experience by providing immediate confirmation of their order with relevant details

#### Flow 2: Stock Deduction Flow

* **Type**: Record-Triggered Flow
* **Object**: HandsMen\_Order\_\_c
* **Trigger**: When an order is confirmed
* **Action**: Retrieves the associated product from **HandsMen\_Product\_\_c**, checks inventory stock, and deducts the ordered quantity
* **Purpose**: Ensures real-time inventory updates and prevents overselling

#### low 3: Order Review Automation

* **Type**: Scheduled Flow
* **Purpose**: Automatically fetches and reviews all opportunities/orders created within a specific timeframe
* **Use Case**: Helps management monitor sales performance periodically without manual intervention

**Automation using Apex**

**Order Confirmation Email Template:-**

Dear {!HandsMen\_Order\_\_c.HandsMen\_Customer\_\_c},

Your order #{!HandsMen\_Order\_\_c.Name} has been confirmed!

Thank you for shopping with us.

Best Regards,

Sales Team

****Low Stock Alert Email Template:-**Dear Inventory Manager,**

This is to inform you that the stock for the following product is running low:

Product Name: {!Inventory\_\_c.HandsMen\_Product\_\_c}

Current Stock Quantity: {!Inventory\_\_c.Stock\_Quantity\_\_c}

Please take the necessary steps to restock this item immediately.

Best Regards,

Inventory Monitoring System

****Loyality Points Email Template:-**Congratulations! You are now a {!HandsMen\_Customer\_\_c.Loyalty\_Status\_\_c} member and you are eligible for our Loyalty Rewards Program.**

Enjoy exclusive discounts, early access to offers, and special member benefits.

Thank you for your continued Support.

****Update Order Total :-****

**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;

            }

        }

****Stock Deduction Trigger:-**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;

                }

            }

        }

    }

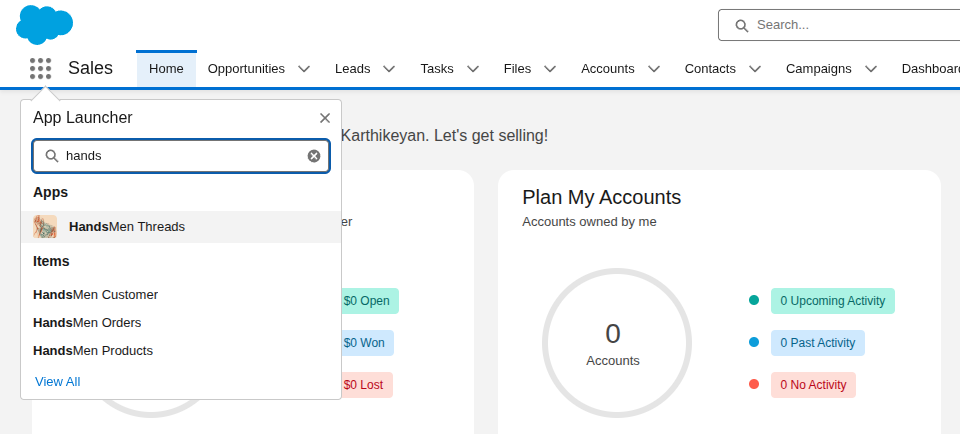
    if (!inventoriesToUpdate.isEmpty()) {

        update inventoriesToUpdate;

    }

}

****Execute anonymous window code:-**System.schedule('Daily Inventory Sync', '0 0 0 \* \* ?', new InventoryBatchJob());**

**Output**

