**FINAL REPORT**

**Project Title**

**“to supply leftover food to poor– (DEVELOPER)”**

**Team ID** : LTVIP2025TMID28938

**Team Size** : 4

**Team Leader** : Seerapu Sai Sandeep

**Email ID :**[SeerapuSaiSandeep.22.It@anits.edu.in](mailto:SeerapuSaiSandeep.22.It@anits.edu.in)

**Team member** : Kantubugata Durga Prasad  
**Email ID :**[kantubugatadurgaprasad.22.it@anits.edu.in](mailto:kantubugatadurgaprasad.22.it@anits.edu.in)

**Team member :** Gunnam Sai Sri Vasthav

**Email ID :**[gunnamsaisrivasthav.22.it@anits.edu.in](mailto:gunnamsaisrivasthav.22.it@anits.edu.in)

**Team member** : Joshika Palametta

**Email ID :**[joshikapalametta.22.it@anits.edu.in](mailto:joshikapalametta.22.it@anits.edu.in)

**1. INTRODUCTION**

**1.1 Project Overview**

The **CRM Application to Supply Leftover Food to the Poor** is a Salesforce-based solution designed to digitize, automate, and optimize the food donation process. Many NGOs, volunteers, and restaurants currently manage leftover food distribution manually—using notebooks, calls, or WhatsApp—resulting in inefficiencies, miscommunication, and wasted food.

This project aims to centralize and streamline all operations in the food donation lifecycle using Salesforce CRM's automation, reporting, and data management capabilities.

The CRM system has been specifically built to manage and automate four key components of the food redistribution process:

1. **Donor Management** – Captures donor details such as name, contact info, location, and donation history.
2. **Food Inventory Management** – Tracks leftover food donations with attributes like food type, quantity, expiry, and pickup location.
3. **Pickup & Delivery Tracking** – Manages volunteer assignments, pickup schedules, and delivery completion to ensure food reaches beneficiaries on time.
4. **Communication and Reporting** – Sends automated emails to donors post-delivery and provides real-time dashboards for NGOs to monitor operations.

The solution uses Salesforce custom objects, lookup relationships, flows, triggers, validation rules, profiles, permission sets, reports, and dashboards. The **Lightning App interface** ensures ease of use for different user roles such as **Admin, Volunteer, and Donor**, improving collaboration and accountability.

**1.2 Purpose**

The main goal of this project is to provide a **scalable, secure, and user-friendly CRM** for organizations working to supply leftover food to those in need. This system aims to eliminate manual tracking, reduce food wastage, and ensure timely delivery through automation and data visibility.

**Key Objectives:**

**Food Donation Management**

* Create structured records for every food donation.
* Record quantity, food type (e.g., rice, curry, fruits), and expiry details.
* Ensure donations are linked to the correct donor and pickup request.

**Pickup & Delivery Tracking**

* Maintain a clear schedule of pickups, with volunteers assigned per task.
* Track delivery status, timings, and feedback from recipients.
* Enable real-time updates to prevent delays or spoilage.

**Email Notifications**

* Trigger automatic confirmation emails to donors after food has been successfully delivered.
* Include delivery date, volunteer name, and gratitude message.

**Reporting and Analytics**

* Build reports on:
  + Total food collected (by type, location)
  + Number of donors and volunteers
  + Successful vs. delayed deliveries
* Use dashboards for live tracking and decision-making by NGOs or admin users.

**Role-Based Access Control**

* Define profiles such as:
  + **Admin** – Full access to monitor operations and assign tasks.
  + **Volunteer** – Access to pickup and delivery assignments.
  + **Donor** – Access to their donation history and records.
* Secure sensitive data through permission sets and field-level security.

**Data Accuracy and Validation**

* Prevent invalid entries such as:
  + Pickup without assigned food
  + Expired food accepted for delivery
* Ensure food safety and consistency using **validation rules** and formula fields.

**2. IDEATION PHASE**

**2.1 Problem Statement**

Problem statements are essential in any project because they clearly articulate the challenges users face. They guide the team to focus on solving meaningful issues and ensure that the final solution delivers real value. By clearly framing the problem, we identify who is affected, what their difficulties are, and why these problems exist, along with their consequences.

In the current food donation landscape, especially involving leftover food from restaurants or events, operations are often managed manually—via handwritten logs, direct calls, or informal volunteer coordination. While manageable on a small scale, this system leads to confusion, inefficiency, and missed opportunities when scaled up or when multiple stakeholders are involved.

Through research and discussions with NGOs and volunteers, we discovered major pain points affecting both food donors and recipients.

**Manual Data Handling**

* Donor and volunteer data is usually maintained in notebooks or Excel files.
* Data gets lost or is inconsistent, leading to poor tracking of donation history or follow-ups.

**Uncoordinated Pickup Scheduling**

* Pickups are often assigned informally, without a clear link to the food source or time of preparation.
* Volunteers may arrive late or not at all, leading to food spoilage.

**Lack of Real-Time Communication**

* Donors don’t receive confirmations when food is picked up.
* NGOs have no live status update of deliveries.

**No Role-Based Access**

* All users have access to all data, causing confusion or potential data breaches.
* There's no distinction in access level between volunteers, admins, or donors.

**No Performance Visibility**

* NGOs lack dashboards to track metrics like number of meals distributed, donors served, or food types in high demand.
* Without reporting, it's hard to measure social impact or manage resources effectively.

**Problem Statement Template Applied:**

We believe that food donors, NGOs, and volunteers are struggling with managing donations, pickups, and delivery tracking because of the absence of a centralized digital system and reliance on fragmented manual coordination (e.g., calls, messages, and spreadsheets). This leads to delays in food delivery, wasted resources, and missed opportunities to serve people in need.

**Elaboration**:

* **Customer Type**: Includes NGO coordinators, volunteers handling logistics, and donors who supply the leftover food.
* **Core Problem**: "Managing food distribution efficiently" is difficult due to the lack of integration between donors, delivery personnel, and recipients.
* **Root Causes**:
  + Lack of a digital system that connects donors, inventory, pickups, and deliveries.
  + Heavy dependence on verbal coordination or WhatsApp.
* **Negative Impacts**:
  + **Food Wastage**: Delays or mismanagement may result in expired or spoiled food.
  + **Inefficiency**: Volunteers may not know which location needs priority delivery.
  + **Donor Dissatisfaction**: Lack of follow-up leads to a loss of trust in the system.

**2.2 Empathy Map Canvas**

To build a system that meets users’ real needs, we used the Empathy Map Canvas to better understand their behaviors, concerns, and motivations.

**Primary Stakeholders:**

* **NGO Admins**: Responsible for assigning pickups, verifying deliveries, and tracking operations.
* **Volunteers**: Individuals who pick up and deliver food. They need clear instructions and timely updates.
* **Donors (Restaurants or Households)**: Those who donate leftover food. They want assurance that food is used and delivered on time.

**Persona 1: NGO Admin**

|  |  |
| --- | --- |
| **Category** | **Details** |
| **Says** | "I need to know which food is still undelivered." "Is the volunteer assigned for pickup today?" |
| **Thinks** | "It would be great to have a real-time dashboard of deliveries." "We need to avoid last-minute confusion." |
| **Does** | Assigns pickups, monitors delivery records, and communicates with donors and volunteers. |
| **Feels** | Stressed during peak hours. Wants a reliable system to automate scheduling and reporting. |

**Persona 2: Volunteer**

|  |  |
| --- | --- |
| **Category** | **Details** |
| **Says** | "Where do I pick up food from today?" "Who do I deliver it to?" |
| **Thinks** | "I hope I get all addresses in one place." "I don't want to miss any urgent pickups." |
| **Does** | Visits donors, collects food, delivers to target locations. |
| **Feels** | Anxious if instructions are unclear. Prefers quick, mobile access to assignment details. |

**Persona 3: Donor (Restaurant or Household)**

|  |  |
| --- | --- |
| **Category** | **Details** |
| **Says** | "Can someone pick this up in the next 30 minutes?" "Please confirm once it's delivered." |
| **Thinks** | "I want to help, but I don’t want the food to go to waste." |
| **Does** | Prepares, packs, and offers leftover food to the system. |
| **Feels** | Hopeful but needs trust that the system will deliver food to those in need. Likes updates. |

**2.3 Brainstorming Outcomes**

Our brainstorming sessions helped convert these pain points into practical system features using Salesforce tools. Below is a summary of what was considered and implemented.

|  |  |  |
| --- | --- | --- |
| **Feature/Concept** | **Description** | **Outcome** |
| Custom Objects | Entities like Donor\_\_c, Food\_Item\_\_c, Pickup\_\_c, Delivery\_\_c to organize structured records | ✔️ Implemented |
| Record Types | Separate layouts for perishable vs. non-perishable food types | ✔️ Implemented |
| Custom Page Layouts | Simplified views for volunteers to only see assigned pickups | ✔️ Implemented |
| Role-Based Access | Profiles for NGO Admin, Volunteer, Donor with different permissions | ✔️ Implemented |
| Automation – Apex Trigger | Auto-update Delivery\_Status\_\_c when food is marked delivered | ✔️ Implemented |
| Automation – Flows | Send confirmation email to donor after successful delivery | ✔️ Implemented |
| Validation Rules | Ensure food with expired timestamps is not marked deliverable | ✔️ Implemented |
| Dashboards & Reports | Show metrics like meals served, top donors, pickup time efficiency | ✔️ Implemented |
| Food Expiry Tracker | Alerts for food near expiry in storage | ✔️ Implemented |
| Mobile App Interface | Salesforce Lightning App with tabs for Pickup, Delivery, Inventory | ✔️ Implemented |
| Sample Data for Testing | Dummy donors, volunteers, food types preloaded | ✔️ Implemented |

## ****REQUIREMENT ANALYSIS****

### ****3.1 Customer Journey Map****

The Customer Journey Map outlines the lifecycle of a **food donor or NGO staff member** interacting with the food distribution system — from food registration to successful delivery and acknowledgment. This map helps identify inefficiencies and ensures the CRM provides a seamless, transparent, and trackable donation process using Salesforce.

This mapping illustrates each phase a food donor, NGO coordinator, or volunteer experiences, highlighting system touchpoints and how CRM features streamline operations, ensure accountability, and enhance impact tracking.

#### ****Stages in Customer Journey****

|  |  |  |
| --- | --- | --- |
| **Stage** | **Description** | **System Involvement** |
| 1. Food Donation Initiated | A donor (e.g., restaurant or household) registers leftover food through form or app. | Donor\_\_c and Food\_Item\_\_c objects created or updated with donation and contact details. |
| 2. Pickup Request Logged | Pickup is scheduled based on location, time, and type of food. | Pickup\_\_c record created, linking donor and food items with timestamps. |
| 3. Volunteer Assigned | A volunteer is assigned to the pickup and delivery task. | Delivery\_\_c object created with assignment, linked to Volunteer\_\_c profile. |
| 4. Delivery Completed | Volunteer confirms delivery of food to NGO kitchen or directly to beneficiaries. | Delivery\_Status\_\_c updated; timestamped entry recorded. |
| 5. Donor Confirmation | Donor receives confirmation that their food has been delivered. | Email Flow sends confirmation email using merge fields (donor name, delivery details). |

#### ****User Personas in Journey****

* **Donor** – initiates the food registration and awaits acknowledgment of delivery.
* **NGO Admin** – manages food inventory, volunteer assignments, and reporting.
* **Volunteer** – collects and delivers food based on instructions from the CRM.

### ****3.2 Solution Requirements****

The system's requirements are defined to ensure full functionality, security, and scalability while being simple enough for non-technical users (e.g., small restaurant owners, volunteers). These are divided into **Functional Requirements** and **Non-Functional Requirements**.

#### ****Functional Requirements:****

* **Donor & Food Registration**: The CRM must allow creation of donor profiles and entry of food details including type, quantity, expiry time.
* **Pickup Scheduling**: Staff should be able to create and track pickup records associated with donor and food items.
* **Volunteer Assignment**: Volunteers should be assigned to deliveries and notified accordingly.
* **Confirmation Email**: After successful delivery, the donor should receive an automatic confirmation.
* **Reporting**: Generate dashboards for meals delivered, pickup efficiency, volunteer performance, and donation trends.
* **Validation Rules**: The system should ensure no expired food is assigned for pickup.

#### ****Functional Requirements Table****

|  |  |
| --- | --- |
| **ID** | **Requirement Description** |
| FR1 | CRUD operations for Donor\_\_c, Food\_Item\_\_c, Pickup\_\_c, and Delivery\_\_c objects |
| FR2 | Auto-calculation of total meals per day from delivery records |
| FR3 | Trigger-based email confirmation to donor upon successful delivery |
| FR4 | Different forms for Perishable and Non-Perishable items using Record Types |
| FR5 | Relationships established via Lookups (Donor → Food, Food → Pickup, Pickup → Delivery) |
| FR6 | Profiles and Permission Sets for Admin, Volunteer, and Donor roles |
| FR7 | Delivery timestamp tracking and volunteer status history |
| FR8 | Dashboards and Reports for total donations, active donors, and food wastage prevented |

#### ****Non-Functional Requirements:****

* **Responsiveness**: The system should respond quickly, especially during time-critical food pickups.
* **Access Control**: Only NGO admins can assign pickups, while volunteers see only their assigned deliveries.
* **Secure Communication**: Donor details and food information must be handled securely.
* **Data Integrity**: All relationships must be validated to ensure accurate traceability.
* **Auditability**: Changes in delivery status or volunteer assignment should be logged for transparency.

#### ****Non-Functional Requirements Table****

|  |  |
| --- | --- |
| **ID** | **Requirement Description** |
| NFR1 | Use of Apex Triggers and Flows to automate pickups and confirmations |
| NFR2 | Validation rules to ensure food is not expired at time of assignment |
| NFR3 | Profiles and Permission Sets to implement role-based access |
| NFR4 | User-friendly UI through Salesforce Lightning components |
| NFR5 | Modular object structure for easy extension (e.g., adding NGOs, multiple locations) |
| NFR6 | Support for growing donor and volunteer base without performance degradation |
| NFR7 | Mobile access via Salesforce App or custom Lightning App Pages |

### ****3.3 Data Flow Diagram (DFD)****

#### ****Level 0 – Context Diagram: High-Level View****

|  |  |
| --- | --- |
| **Entities** | **Description** |
| **Donor (External)** | Provides leftover food and receives confirmation |
| **NGO Admin (External)** | Manages the system and assigns pickups/deliveries |
| **CRM System (Process)** | Accepts data, handles automation, stores records, sends emails |

#### ****Data Flows****

* Donor provides food details → stored in **Food\_Item\_\_c**
* Admin schedules pickup → creates **Pickup\_\_c**
* Volunteer assigned → **Delivery\_\_c** created
* Status update → triggers confirmation **Email Flow**

[Donor]

↓

(Food\_Item\_\_c)

↓

(Pickup\_\_c)

↓

(Delivery\_\_c)

↓

[Email Flow → Donor]

# ****Food Distribution Management System Using Salesforce****

## ****1. Introduction****

Food wastage is a critical issue while many people struggle with hunger. This project aims to build a **CRM-based solution on Salesforce** that facilitates the **collection and distribution of leftover food from donors to the poor** through a coordinated process involving donors, volunteers, and receivers.

## ****2. Problem–Solution Fit****

### ****Identified Problems and Salesforce-Based Solutions****

|  |  |
| --- | --- |
| **Problem** | **Implemented Salesforce Solution** |
| Leftover food going to waste | A platform for donors to register and donate surplus food. |
| No pickup coordination | Auto-scheduling of food pickups via Pickup\_Request\_\_c object. |
| Manual delivery tracking | Volunteers are auto-assigned and deliveries are tracked using Delivery\_Record\_\_c. |
| Lack of communication | Automated confirmation emails sent to donors using Salesforce Flow. |
| No data on distribution | Reports & Dashboards show distribution statistics, donor participation, and volunteer efficiency. |

## ****3. Data Flow Diagram (DFD)****

### ****Level 0 DFD****

Shows a single system process:

* **Inputs**: Donor registers food → Volunteer picks up food
* **Outputs**: Food delivered to needy → Confirmation sent to donor

### ****Level 1 DFD – Detailed View****

#### ****Processes:****

* **Register Food Donation**: Donor submits details via a form.
* **Auto-Schedule Pickup**: System creates a Pickup\_Request\_\_c.
* **Auto-Assign Volunteer**: Based on location/availability, a volunteer is assigned.
* **Confirm Delivery**: Volunteer confirms delivery via app/form.
* **Send Confirmation Email**: Flow retrieves donor’s email from Donor\_\_c and sends acknowledgment.

## ****4. Technology Stack****

### ****Primary Technologies Used****

|  |  |
| --- | --- |
| **Technology** | **Purpose** |
| **Salesforce Platform (Lightning Experience)** | Foundation CRM |
| **Custom Objects** | Donor, Food Item, Pickup Request, Delivery Record |
| **Apex Triggers** | Auto-update delivery status |
| **Flows** | Send email confirmations |
| **Validation Rules** | Ensure valid donation entries |
| **Reports & Dashboards** | Analyze food donations, delivery stats |
| **Profiles & Permission Sets** | Access control for Admins, Volunteers |

## ****5. Custom Objects****

|  |  |
| --- | --- |
| **Object Name** | **Purpose** |
| Donor\_\_c | Stores donor information (name, contact, address) |
| Food\_Item\_\_c | Details of leftover food (type, quantity, expiry) |
| Pickup\_Request\_\_c | Tracks pickup requests, scheduled by the system |
| Delivery\_Record\_\_c | Stores delivery status, recipient details |
| Volunteer\_\_c | Contains volunteer details, availability |

## ****6. Automation Components****

|  |  |
| --- | --- |
| **Component** | **Description** |
| **Flow** | Sends confirmation email after delivery is confirmed |
| **Apex Trigger** | Updates status of Delivery\_Record\_\_c once volunteer marks as delivered |
| **Validation Rule** | Ensures expired food is not accepted |
| **Assignment Rule/Process** | Automatically assigns available volunteer |

## ****7. Profiles and Access Control****

|  |  |
| --- | --- |
| Role | Permissions |
| **Admin** | Full access to all records |
| **Volunteer** | Access only to assigned deliveries |
| **Donor** | Can register food but limited access to other objects |

## ****8. Reporting and Dashboards****

|  |  |
| --- | --- |
| **Report** | **Insights** |
| Food Donated by Location | Highlights areas with most donations |
| Volunteer Efficiency | Number of deliveries completed |
| Daily Distribution Report | How much food was delivered daily |

## ****9. Lightning App Builder****

Custom **Food Distribution App** with these tabs:

* **Donors**
* **Food Items**
* **Pickup Requests**
* **Delivery Records**
* **Reports and Dashboards**

## ****10. Tools Used****

|  |  |
| --- | --- |
| **Tool** | **Purpose** |
| **Object Manager** | Create and manage custom objects |
| **Flow Builder** | Automate email sending |
| **Developer Console** | Apex trigger coding |
| **Report Builder** | Create donation/distribution reports |
| **Lucidchart / Draw.io** | Design DFDs and architecture |
| **Salesforce Sandbox** | Testing environment |
| **Change Sets** | Deployment to production |

## ****11. Solution Architecture Overview****

### ****Layers****

|  |  |
| --- | --- |
| Layer | Component |
| **Presentation** | Lightning UI (form for donors, dashboard for admin) |
| **Business Logic** | Flows, Apex Triggers |
| **Data Layer** | Custom Objects (Donor, Food Item, Pickup, Delivery) |
| **Security Layer** | Role-based access (Profiles, Permission Sets) |
| **Reporting Layer** | Dashboards, Reports |

## ****12. Conclusion****

This Salesforce-based CRM project offers a scalable, low-code solution for managing food donations and ensuring timely delivery to the underprivileged. It ensures transparency, automation, accountability, and real-time reporting, aligning well with the mission of reducing food waste and feeding the needy.

**Architecture Flow (Text-based Explanation)**

**+-------------------+**

**| Lightning UI |**

**| (Donor / Worker) |**

**+--------+----------+**

**|**

**v**

**+-------------------+**

**| Custom Objects |**

**|-------------------|**

**| Item\_\_c |**

**| Customer\_Order\_\_c |**

**| Billing\_\_c |**

**+--------+----------+**

**|**

**v**

**+-------------------+**

**| Business Logic |**

**|-------------------|**

**| Formula Fields |**

**| Apex Triggers |**

**| Flows |**

**| Validation Rules |**

**+--------+----------+**

**|**

**v**

**+-------------------+**

**| Access Control |**

**|-------------------|**

**| Profiles |**

**| Permission Sets |**

**+--------+----------+**

**|**

**v**

**+-------------------+**

**| Reporting Layer |**

**|-------------------|**

**| Reports |**

**| Dashboards |**

**+-------------------+**

**Solution Architecture Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **User Action** | **Component Triggered** | **Object Affected** | **Automation / Output** |
| Create a new item | Lightning UI | Item\_\_c | Record is saved; item appears in inventory list |
| Place order | Custom Order Page | Customer\_Order\_\_c | Record created linking selected item to the customer |
| Create billing | Billing Tab | Billing\_\_c | Charges auto-calculated using formula fields |
| Enter payment | Billing Tab | Billing\_\_c | Apex Trigger updates Paid\_Amount\_\_c field based on payment |
| Billing updated | Record-Triggered Flow | Billing\_\_c | Flow sends a confirmation email to the customer |
| Open reports | Reports tab | – | System fetches and displays real-time analytics and KPIs |
| Switch roles | Profile / Permission Set | – | UI changes; object access and edit rights updated dynamically |

# 5****5. PROJECT PLANNING & SCHEDULING****

## ****5.1 Project Planning****

To ensure effective development and continuous delivery, the team adopted the **Agile Methodology** using the **Scrum Framework**. The project was divided into **two sprints**, each spanning **5 working days**, allowing for quick feedback cycles, iterative builds, and timely delivery of critical modules.

### ****Agile Planning Overview****

Agile methodology promotes **incremental delivery** through short cycles called Sprints. Key planning elements used:

|  |  |
| --- | --- |
| **Agile Component** | **Description** |
| **Product Backlog** | Full list of features including Epics & User Stories |
| **Sprint Backlog** | Stories committed to a specific sprint |
| **Story Points (SP)** | Complexity/effort estimation (Fibonacci scale) |
| **Velocity** | Average SP completed per sprint (12 SP/sprint) |
| **Burndown Chart** | Tracks remaining SPs over sprint days |

## ****Sprint Overview****

|  |  |  |  |
| --- | --- | --- | --- |
| **Sprint** | **Duration** | **Focus Areas** | **Sprint Goal** |
| Sprint 1 | 16th – 20th June 2025 | Data Modeling, Page Layouts, Record Types, Profiles | Set up base CRM structure and access roles |
| Sprint 2 | 21st – 25th June 2025 | Automation, Apex, Reports, Dashboards, Testing | Implement business logic and analytics |

## ****Sprint 1: CRM Data Modeling & Configuration****

**Objective**: Lay the technical foundation for the CRM by designing the data schema and user access structure.

### ****Tasks:****

* Created 5 custom objects: Jewel\_Customer\_\_c, Item\_\_c, Price\_\_c, Customer\_Order\_\_c, Billing\_\_c
* Added custom fields including lookups and formula fields
* Created record types for Gold and Silver items
* Designed page layouts per ornament type
* Created custom **Profiles**: Goldsmith, Worker
* Configured **Role Hierarchy**: Goldsmith > Worker

### ****Deliverables:****

* Functional data schema with all objects and relationships
* Customized page layouts for different item types
* Profiles and roles assigned to test users

|  |  |
| --- | --- |
| **Metric** | **Value** |
| Estimated SP | 12 |
| Completed SP | 12 |
| Sprint Velocity | 12 SP |

## ****Sprint 2: Automation & Reporting****

**Objective**: Implement intelligent automation and build dashboards for data-driven insights.

### ****Tasks:****

* Created **Record-Triggered Flow** to send email after billing
* Wrote **Apex Trigger** to auto-update Paid\_Amount\_\_c
* Configured **Validation Rule** to prevent overpayment
* Built **2 reports** (e.g., Items with Billings, Billing Summary)
* Built **2 dashboards** (e.g., Jewelry Business Overview)
* Conducted functional and performance testing
* Deployed to Production via **Change Set**

### ****Deliverables:****

* Tested automation logic and email flows
* Finalized dashboards with real-time metrics
* Production-ready configuration

|  |  |
| --- | --- |
| **Metric** | **Value** |
| Estimated SP | 12 |
| Completed SP | 12 |
| Sprint Velocity | 12 SP |

## ****Story Point Allocation per Task****

|  |  |
| --- | --- |
| **Task** | **Story Points** |
| Object & Field Creation | 3 SP |
| Record Types & Page Layouts | 2 SP |
| Profiles & Roles Setup | 2 SP |
| Flow for Email Notification | 2 SP |
| Apex Trigger | 2 SP |
| Reports & Dashboards | 2 SP |
| Functional Testing | 1 SP |
| **Total (Planned)** | **14 SP** |
| **Executed per Sprint** | **12 SP** |

## ****Velocity Chart****

|  |  |  |
| --- | --- | --- |
| **Sprint** | **Story Points Planned** | **Story Points Completed** |
| Sprint 1 | 12 SP | 12 SP |
| Sprint 2 | 12 SP | 12 SP |
| **Total** | **24 SP** | **24 SP** |

100% completion for both sprints.

## ****Burndown Chart Overview****

A burndown chart was maintained using **Google Sheets** to track sprint progress.

### ****Example: Sprint 1 Burndown****

|  |  |
| --- | --- |
| **Day** | **SP Remaining** |
| Day 1 | 12 |
| Day 2 | 9 |
| Day 3 | 6 |
| Day 4 | 3 |
| Day 5 | 0 ✅ |

A similar trend was followed in Sprint 2.

## ****Project Management Tools Used****

|  |  |
| --- | --- |
| **Tool** | **Purpose** |
| **Trello** | Sprint planning, user stories, task tracking |
| **Google Sheets** | Burndown chart, velocity tracking |
| **Salesforce Sandbox** | Custom object/config development & testing |
| **Change Set** | Deployment to production environment |

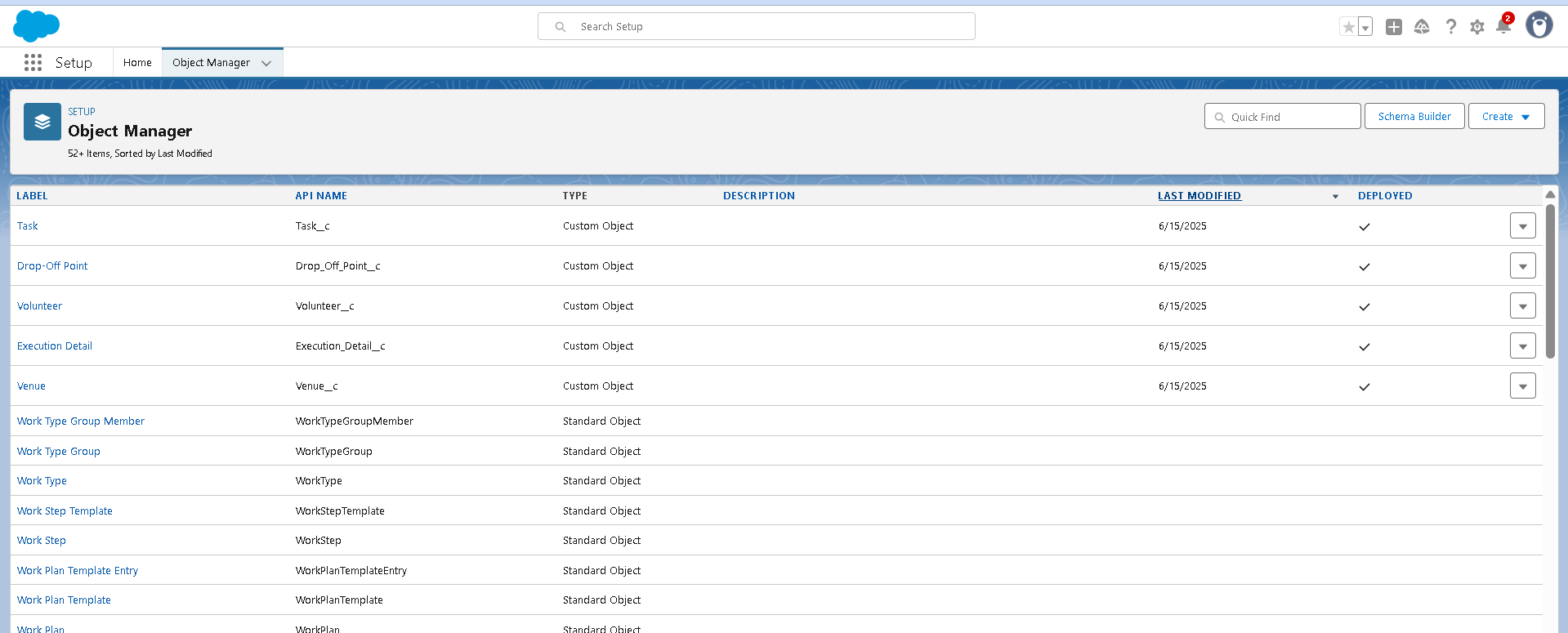
## ****Planning Insights & Best Practices****

* **Fibonacci SPs** used for accurate effort estimation
* Balanced complexity across sprints to prevent overload
* Broke Epics into small user stories for better sprint execution
* Built, tested, and reviewed within the same sprint
* Adjusted planning based on velocity (12 SP per sprint)

## ****Outcome****

* Agile planning helped in delivering testable, modular, and production-ready components incrementally.
* Sprint Retrospectives encouraged team feedback and continuous improvement.
* Timely delivery of functional CRM modules ensured project success.

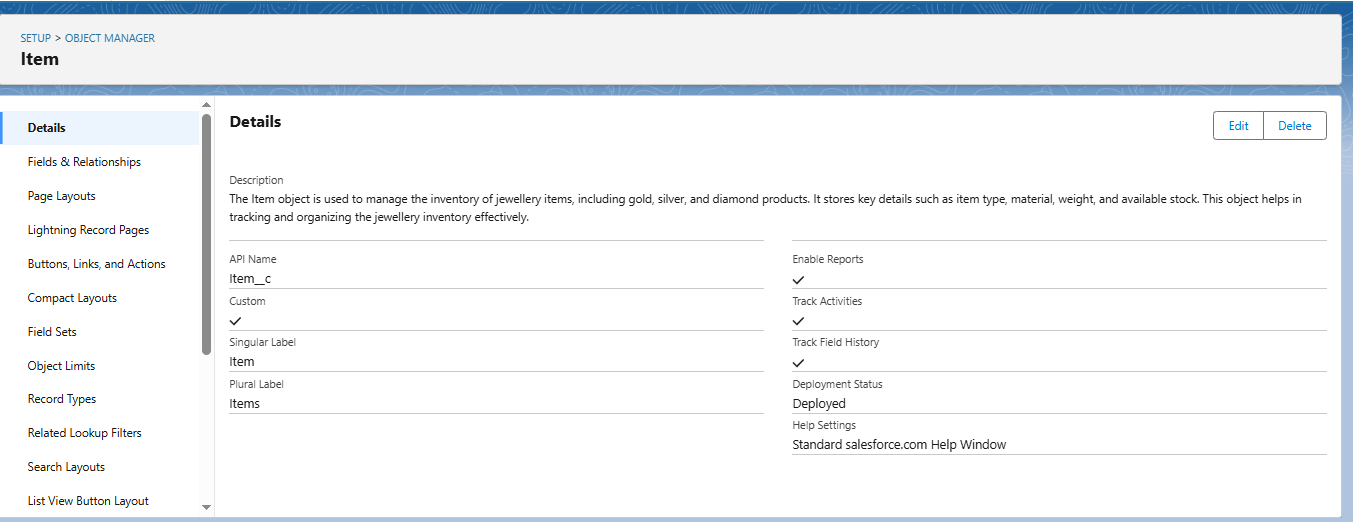
1. **Project Development Phase**



#### 1. ****Donor\_\_c****

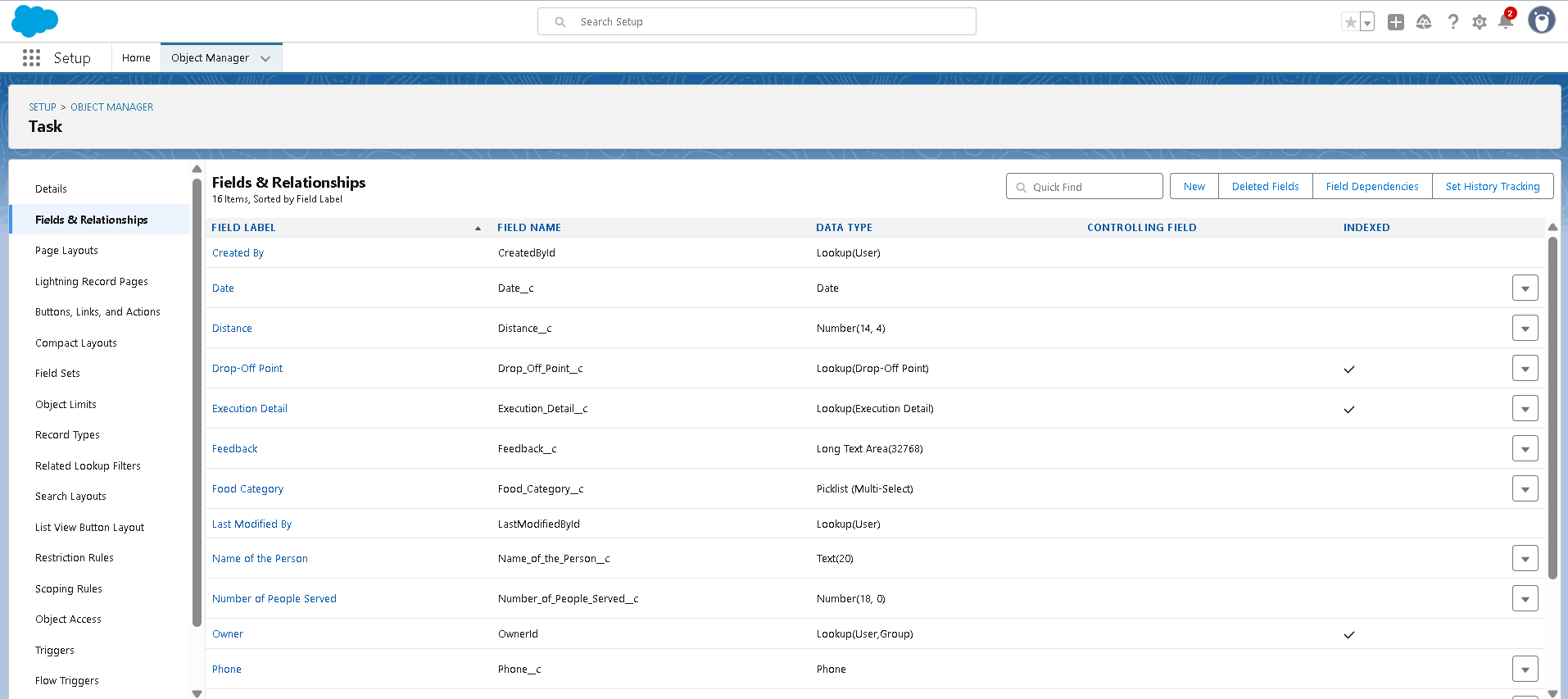
* **Purpose**: Stores details of individuals or organizations that donate leftover food.
* **Key Fields**:
  + Name (Donor's Name)
  + Email\_\_c
  + Contact\_Number\_\_c
  + Address\_\_c
  + Food\_Type\_\_c (e.g., Veg, Non-Veg, Snack)
* **Usage**: Referenced in **Food Donation** records to track which donor contributed which food items.

#### 2. ****Item\_\_c****

* **Purpose**: Maintains a record of all donated food items and their details.
* **Key Fields**:
  + Item\_Name\_\_c
  + Food\_Category\_\_c (Picklist: Veg, Non-Veg, Salad, etc.)
  + Quantity\_\_c
  + Expiration\_Date\_\_c
  + Donor\_\_c (Lookup to Donor\_\_c)
* **Usage**: Core object used in task planning, delivery allocation, and tracking shelf life of food.

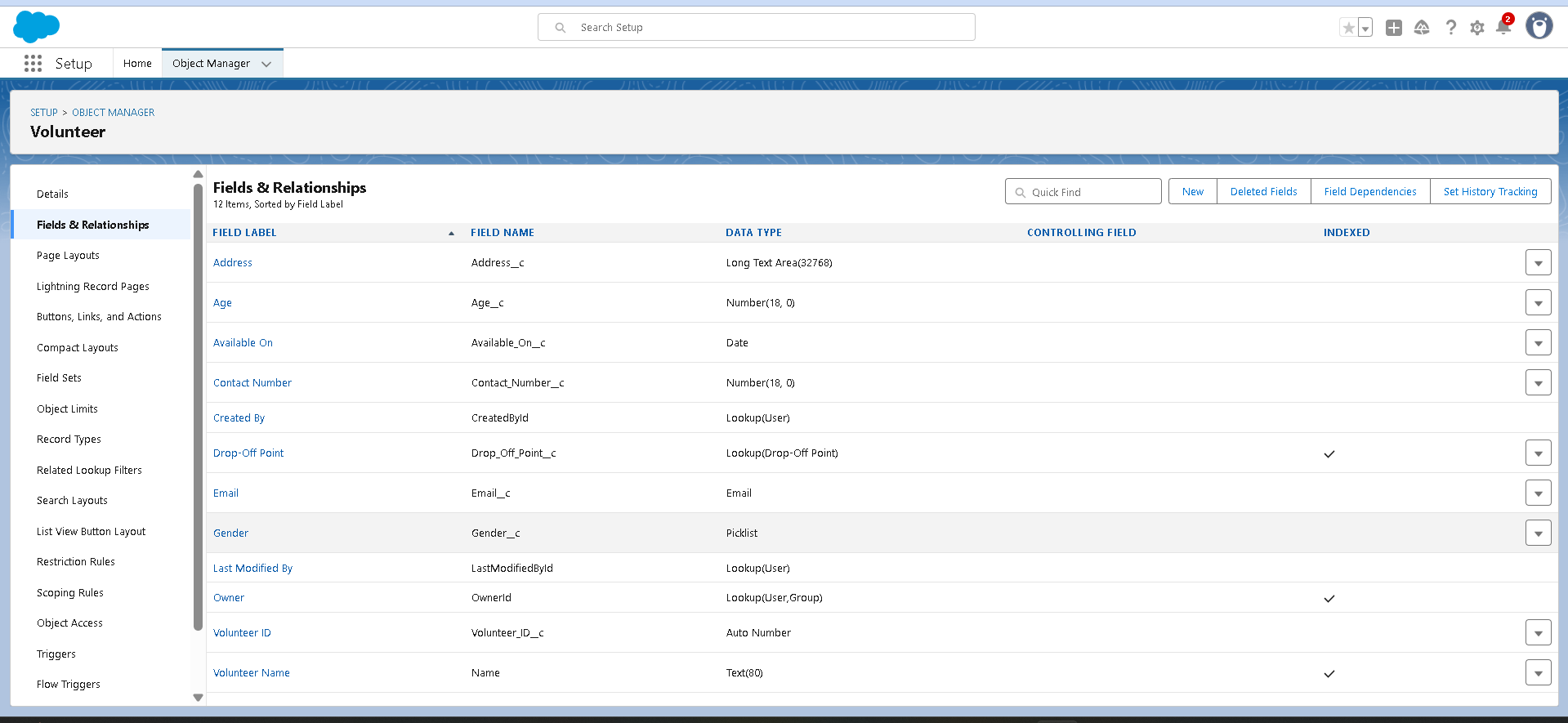
#### ****Task\_\_c****

* **Purpose**: Represents each delivery or food distribution task assigned to a volunteer.
* **Key Fields**:
  + Task\_ID\_\_c (Auto Number)
  + Date\_\_c
  + Food\_Category\_\_c (Multi-select Picklist)
  + Number\_of\_People\_Served\_\_c
  + Drop\_Off\_Point\_\_c (Lookup)
  + Sponsored\_By\_\_c (Venue – Lookup)
  + Volunteer\_Feedback\_\_c
* **Usage**: Key object for managing task allocation and monitoring volunteer engagement and distribution coverage.

****

#### ****Volunteer\_\_c****

* **Purpose**: Stores information about volunteers available to deliver food or assist with logistics.
* **Key Fields**:
  + Volunteer\_ID\_\_c (Auto Number)
  + Name
  + Gender\_\_c
  + Available\_On\_\_c
  + Age\_\_c
  + Email\_\_c
  + Contact\_Number\_\_c
  + Address\_\_c
  + Date\_of\_Birth\_\_c
  + Drop\_Off\_Point\_\_c (Master-Detail)
* **Usage**: Referenced in **Execution Details** and tasks to assign available volunteers to food delivery operations.

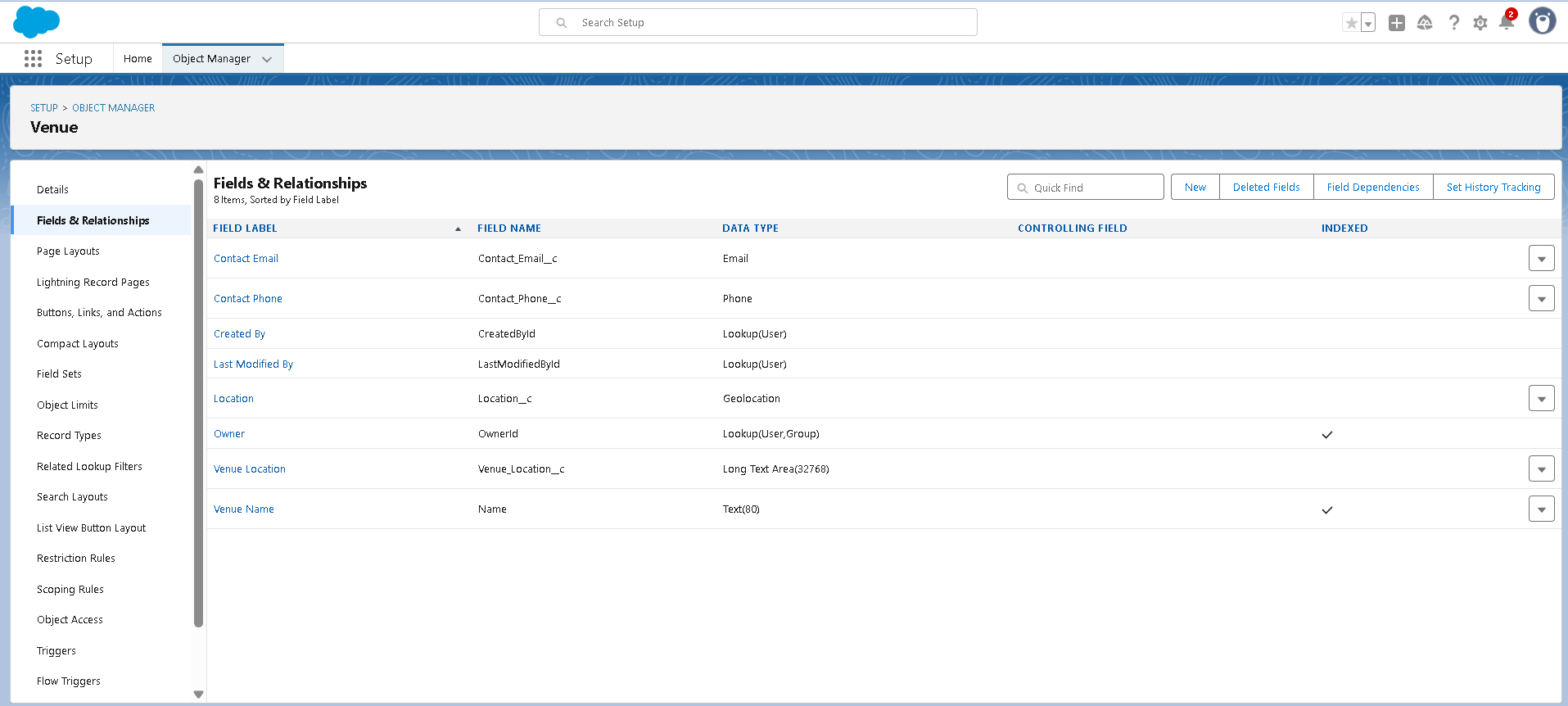


#### 5. ****Drop\_Off\_Point\_\_c****

* **Purpose**: Stores location details where food is to be delivered or picked up from.
* **Key Fields**:
  + Location\_2\_\_c (Geolocation)
  + State\_\_c
  + Venue\_\_c (Lookup)
  + Distance\_\_c (Formula)
* **Usage**: Used in **Task\_\_c** and **Volunteer\_\_c** to calculate proximity and match the best delivery routes and personnel.

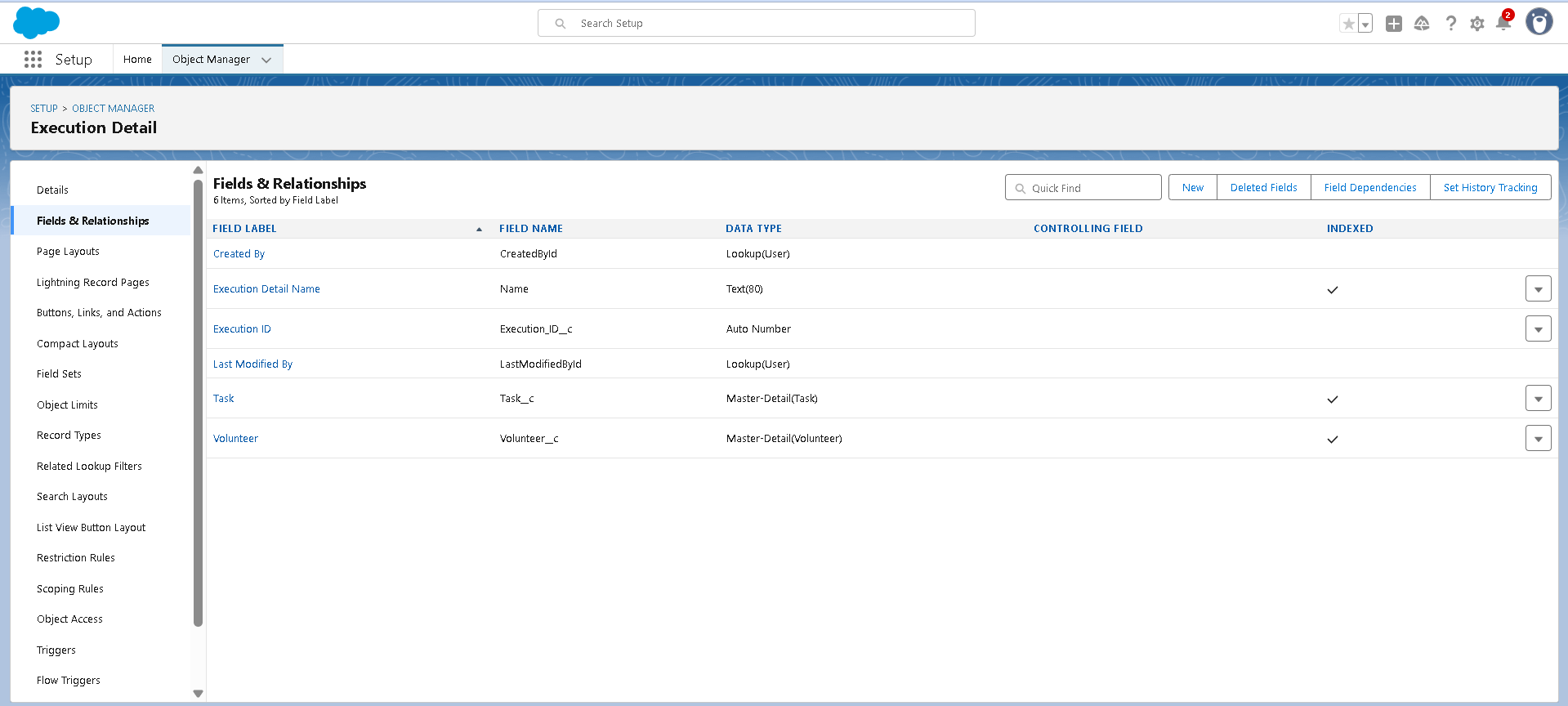
#### C:\Users\HP\Pictures\Screenshots\FEILDSOBJ SS\Screenshot 2025-06-28 074203.png6. ****Venue\_\_c****

* **Purpose**: Represents the organization, shelter, or community center where food is distributed or stored.
* **Key Fields**:
  + Venue\_Name
  + Venue\_Location\_\_c (Long Text)
  + Location\_\_c (Geolocation)
  + Contact\_Email\_\_c
  + Contact\_Phone\_\_c
* **Usage**: Referenced in both **Task\_\_c** and **Drop\_Off\_Point\_\_c** to establish sponsor or beneficiary association.



#### 7. ****Execution\_Detail\_\_c****

* **Purpose**: Tracks the execution status of each food delivery including feedback, volunteer, and task linkage.
* **Key Fields**:
  + Execution\_ID\_\_c (Auto Number)
  + Volunteer\_\_c (Master-Detail)
  + Task\_\_c (Master-Detail)
* **Usage**: Serves as a junction object to manage many-to-many relationships between volunteers and tasks with delivery outcomes.



## B. ****AUTOMATION ELEMENTS****

The **Leftover Food Distribution CRM** utilizes Salesforce’s powerful automation features—including **Apex Triggers**, **Record-Triggered Flows**, and **Validation Rules**—to automate task assignments, notifications, and ensure data accuracy throughout the donation and delivery process.

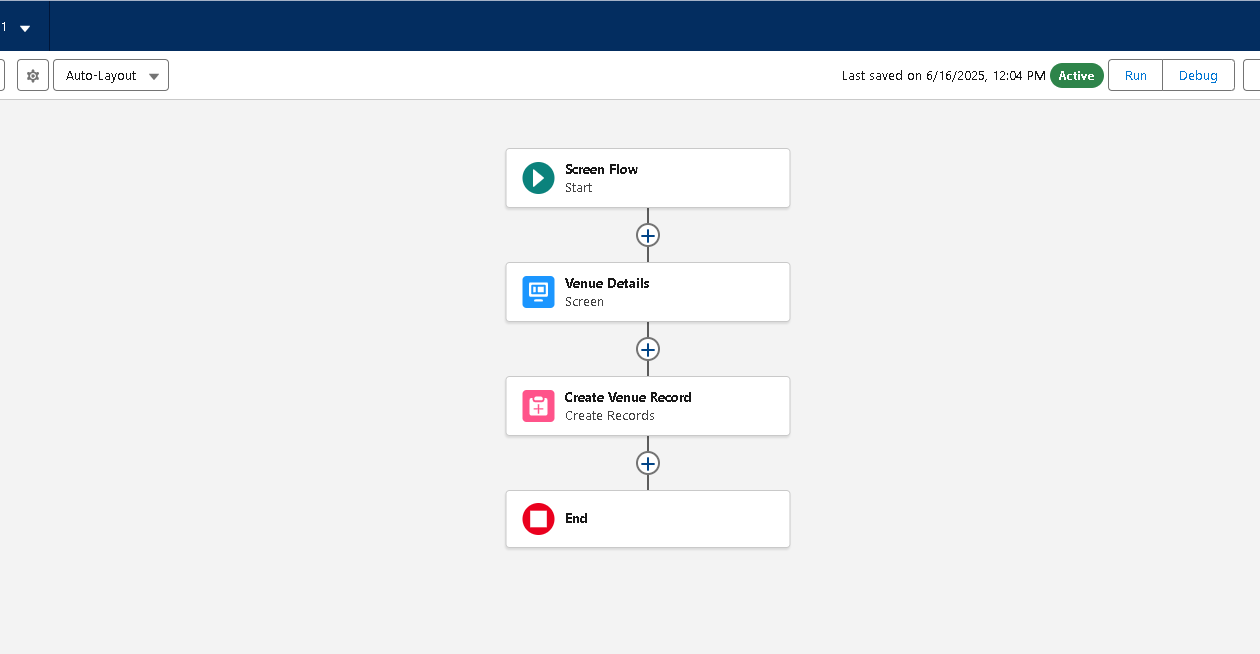
### 1. ****Apex Trigger****

|  |  |
| --- | --- |
| 1 2 3 4 5 | trigger DropOffTrigger on Drop\_Off\_point\_\_c (before insert)    {     for (Drop\_Off\_point\_\_c Drop : Trigger.new)  {         Drop.Distance\_\_c = Drop.distance\_calculation\_\_c;     } } |

### C:\Users\HP\Pictures\Screenshots\Screenshot 2025-06-28 080959.png****Flow Design Highlights****

* **Start Element:**  
  Triggered when a new **Task** record is created or updated (representing a food distribution activity).
* **Get Records:**  
  Retrieves associated records from:
  + **Volunteer** object – using a lookup relationship.
  + **Drop-Off Point** object – to access location details.
* **Decision Element:**  
  Checks specific conditions such as:
  + Whether the **Task is marked as "Completed"**
  + Whether the **number of people served** is greater than 0
* **Assignment Element:**  
  Sets or updates calculated fields such as:
  + **Distance to drop-off point** (based on geolocation)
  + **Total tasks completed by volunteer** (optional cumulative field update)
* **Create/Update Records Element:**  
  Automatically:
  + Creates an **Execution\_Details** record
  + Updates the **Volunteer** object with new task metrics (e.g., count of completed tasks)

.



## Validation Rule – Ensuring Task Entry Integrity

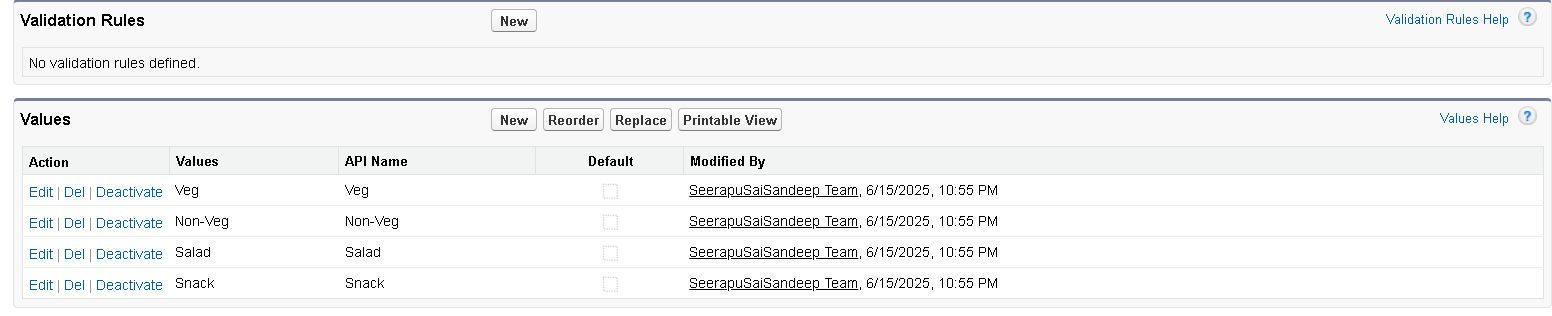
* **Validation Rule Name:**Validate\_Number\_of\_People\_Served
* **Associated Object:**Task\_\_c

**Formula Logic:**

Number\_of\_People\_Served\_\_c> 1000

**Error Message:**

"Number of people served cannot exceed 1000. Please verify the entry."



**Purpose:**

* Prevents data entry errors by ensuring **realistic and valid values** in the task record.
* Maintains **accuracy** in reporting and analytics for how many people were served.
* Helps avoid **exaggerated or fraudulent data**, especially when multiple volunteers are working.
* Protects **public trust** in the system by enforcing logical boundaries.

**Usage Scenario:**

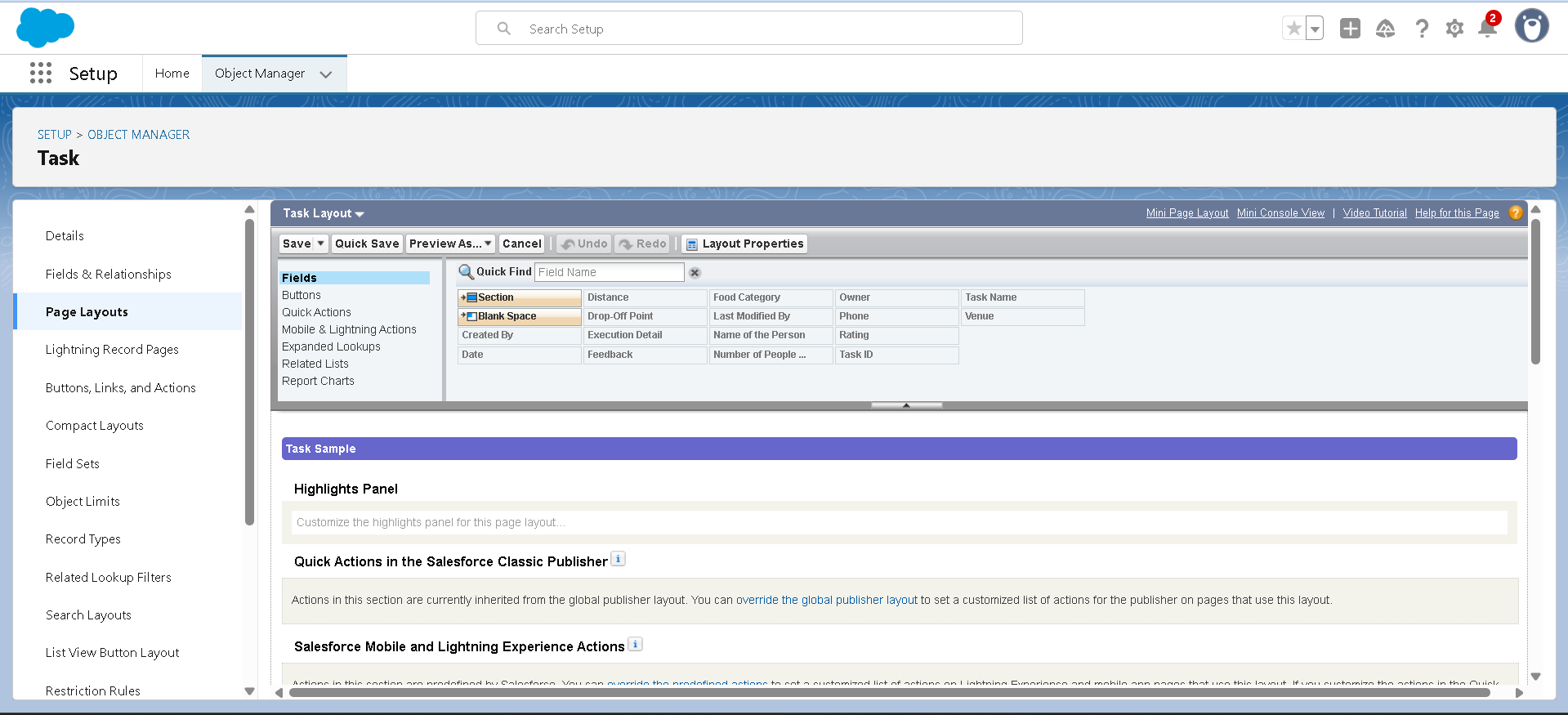
* When a volunteer or NGO staff enters task details and mistakenly records that food was served to more than 1000 people, this rule triggers.
* The system immediately **blocks the save operation** and displays the error message.
* Ensures that data being stored is **valid and audit-ready**.

## ****UI COMPONENTS****

### A. Custom Page Layouts

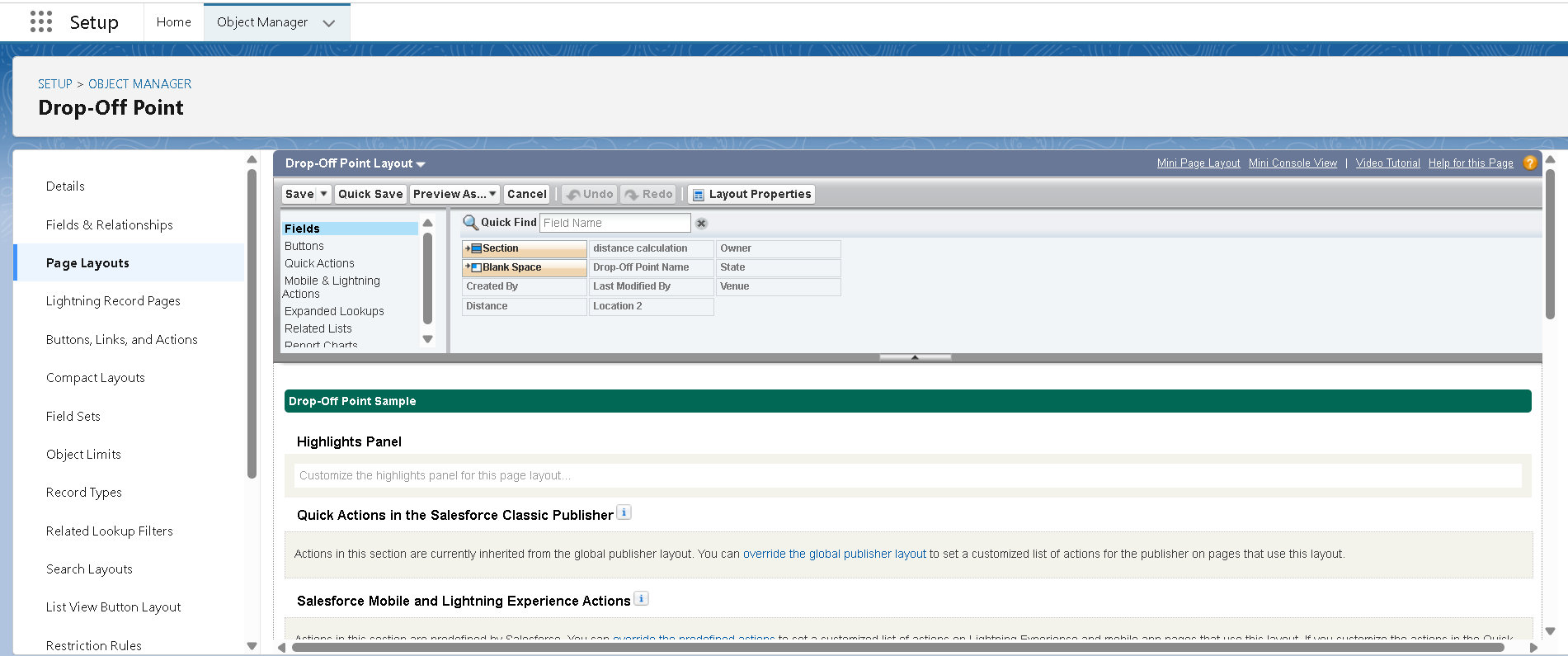
### ****Activity 1: Page Layout for Volunteer Tasks****

* **Navigation Path:**  
  Setup → Object Manager → Task\_\_c → **Page Layouts**
* **Created a New Layout titled:**  
  Task Layout for Volunteer
* **Customization Performed:**
  + **Removed** fields that are irrelevant to the volunteer or food distribution process.
  + **Organized fields** in a logical order to improve ease of data entry for field volunteers and coordinators.
* **Fields Displayed in Layout:**
  + Task ID
  + Date
  + Food Category (Veg / Non-Veg / Snacks / Salads)
  + Number of People Served
  + Name of the Person (Volunteer Contact)
  + Phone
  + Sponsored By (Venue/NGO)
  + Drop-Off Point
  + Distance (calculated using geolocation)
  + Rating
  + Feedback
* **Outcome:**
  + Ensured that volunteers **only see relevant fields** they need for task completion.
  + Improved **data consistency** by reducing clutter and confusion.
  + Layout supports the needs of both **mobile and desktop users**.



### ****Activity 2: Page Layout for Drop-Off Points)****

* **Created a Separate Layout Titled:**  
  Drop-Off Layout - NGO Use Only
* **Purpose:**  
  To allow NGOs or admin users to **view and manage drop-off locations** in a clean, role-specific layout.
* **Customization Performed:**
  + **Removed** operational or volunteer-only fields not required by NGO staff.
  + **Included only attributes** relevant to location coordination and NGO-level planning.
* **Fields Displayed in Layout:**
  + Drop-Off Point Name
  + Location (Geolocation field with latitude & longitude)
  + Venue (lookup to associated venue)
  + State (Picklist for Indian states)
  + Distance Calculation (formula to compute distance from venue)
* **Outcome:**
  + Provides NGO coordinators a **streamlined interface** for managing drop-off logistics.
  + Eliminates distractions by **hiding volunteer or task-level fields**.
  + Enables **accurate location tracking** to optimize food delivery planning

****

### ****Configuration Steps:****

1. **Navigated to:**  
   Task Object → Fields & Relationships
2. **Selected Controlling Field:**  
   Food Category(Picklist with options like: Veg, Non-Veg, Salad, Snack)
3. **Selected Dependent Field:**  
   Number of People Served(Preset ranges as dependent values)



**Dashboards**

**Use Case:**  
To visualize key performance indicators (KPIs) and operational trends such as food distribution volume, volunteer involvement, and task coverage across locations.

**Activity 1: Create Dashboard – “Food Distribution Overview”**

**Dashboard Name:**Food Distribution Overview  
**Description:** Tracks key metrics across food categories, people served, and volunteer distribution.

**Included Components:**

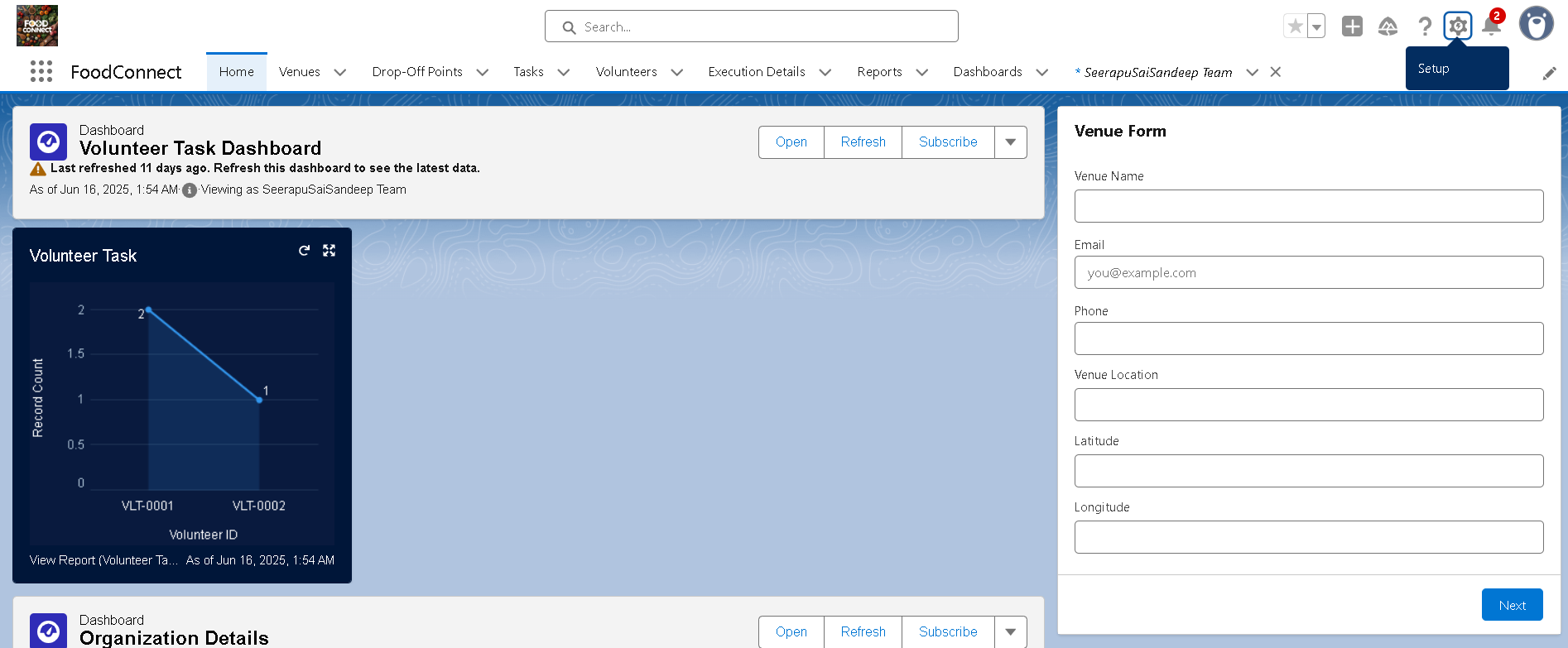
1. **Tasks by Food Category**
   * Type: Bar Chart
   * Displays the number of tasks segmented by Veg, Non-Veg, Snack, etc.
2. **Volunteers by Gender**
   * Type: Pie Chart
   * Visualizes the gender distribution among registered volunteers.
3. **Total People Served by Drop-Off Point**
   * Type: Donut or Column Chart
   * Shows how many people were served in each area.

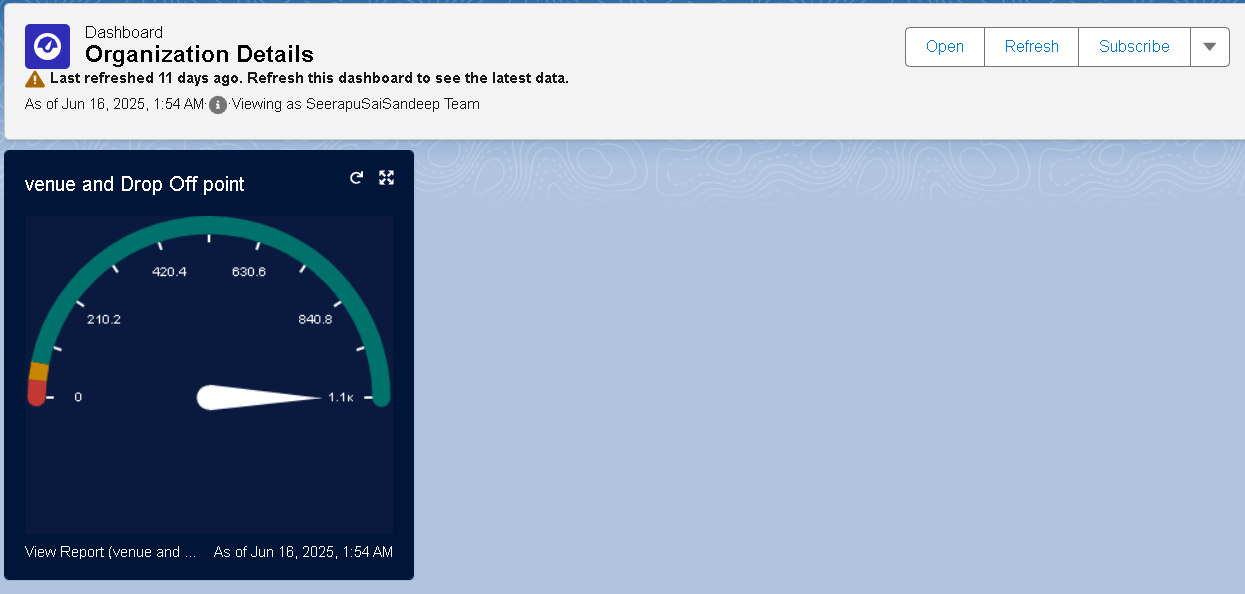
**Activity 2: Create Dashboard – “Operational Insights”**

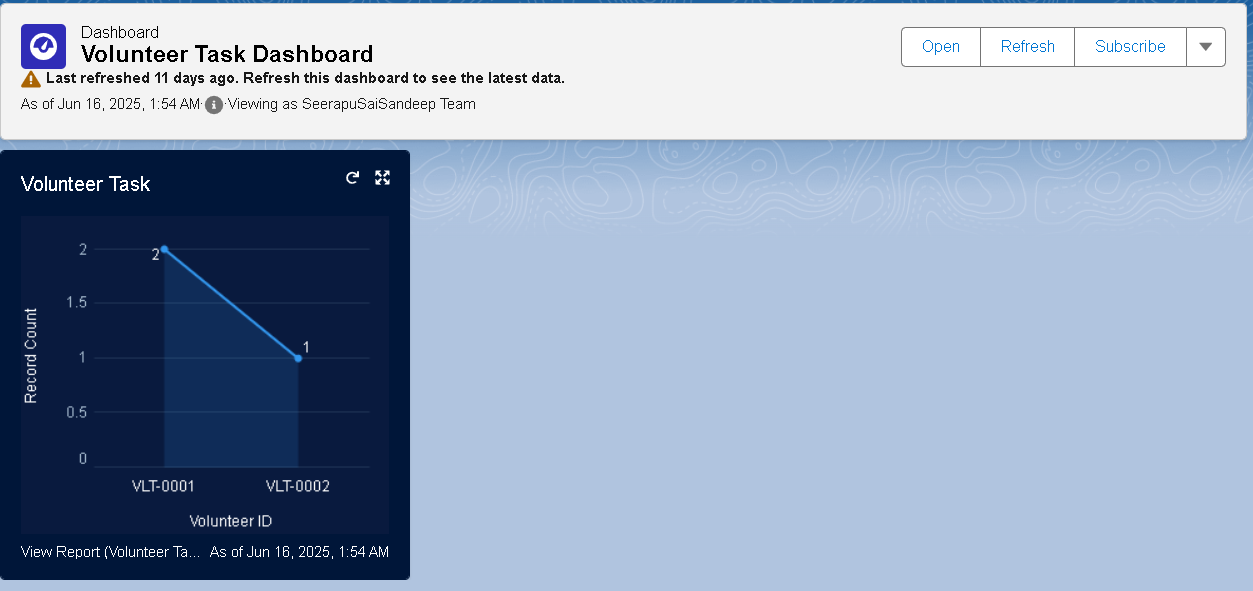
**Dashboard Name:**Operational Insights  
**Purpose:** Gives insights on execution trends, volunteer efficiency, and location coverage.

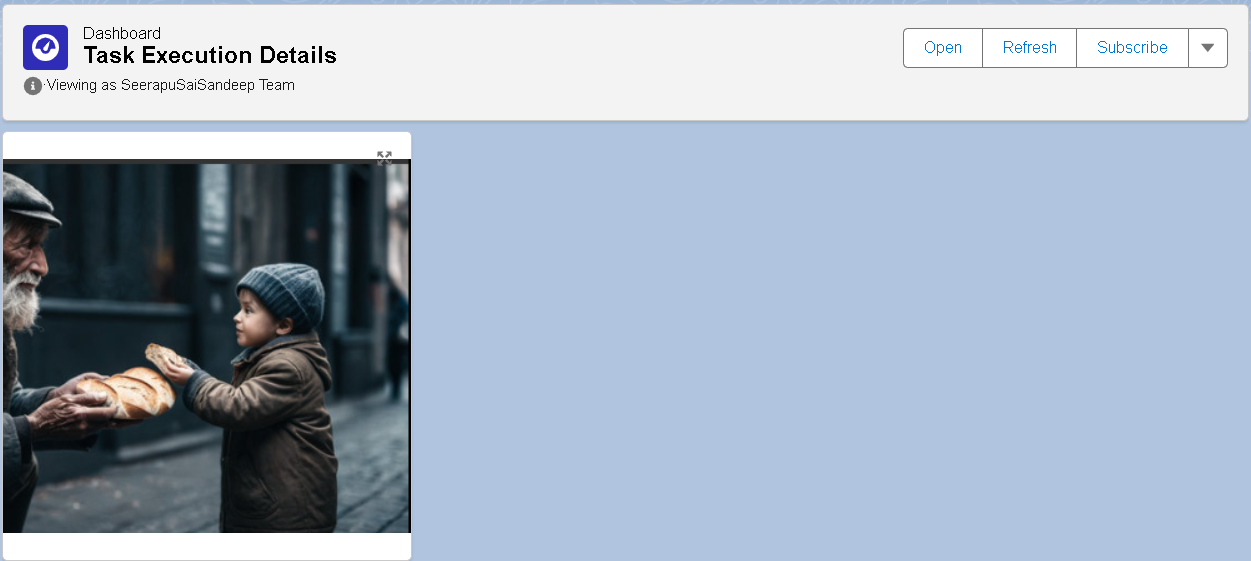
**Included Components:**

1. **📆 Tasks Executed by Date**
   * Type: Line Graph
   * Trend of how many food deliveries were carried out each day.
2. **📍 Drop-Off Point Distance Analysis**
   * Type: Heat Map or Column Chart
   * Highlights the average distance from the venue to drop-off points.
3. **Volunteer Task Contribution**
   * Type: Table or Stacked Bar
   * Lists volunteers and their number of tasks executed.





****

****

### E: Flow: Record-Triggered Flow to Log Distribution Summary (Instead of Sending Email)

**Use Case:**  
Automatically log a **distribution summary** in a custom object or a log field when a new **Billing/Execution** record is created or updated — eliminating the need for manual tracking or email notifications.

**Steps Taken:**

1. **Created a Record-Triggered Flow**
   * Object: Execution\_Details\_\_c
   * Trigger: When a record is *created or updated*
   * Condition: Status = "Completed"
2. **Used Assignment & Formula Fields to Compose Summary**
   * Constructed summary using merge fields:
     + Volunteer Name (Volunteer\_\_r.Name)
     + Task Date (Task\_\_r.Date)
     + Food Type (Task\_\_r.Food\_Category\_\_c)
     + People Served (Task\_\_r.Number\_of\_People\_Served\_\_c)
     + Drop-Off Location (Drop\_Off\_Point\_\_r.State\_\_c)
3. **Updated Summary Field in Execution\_Details\_\_c**
   * Used an Update Records element to populate a field like Distribution\_Summary\_\_c.
4. **Activated the Flow**
   * Ensured test runs were successful with different data conditions (e.g., Veg/Non-Veg tasks, missing fields, etc.)

**Benefits:**

* Automatically logs important delivery data for historical tracking.
* No manual entry needed by volunteers or admins.
* Can be referenced later in **reports** or **dashboards**.
* Helps track execution trends and improves accountability.

## F.Automation: Apex Trigger

**Trigger: Auto-Update Number of People Served**

**Objective:**  
Automatically ensure that the **Number\_of\_People\_Served\_\_c** field in the **Task** object is updated whenever a new **Execution\_Details\_\_c** record is inserted — avoiding manual updates and ensuring accurate reporting.

**Logic:**

* Trigger fires **after insert** on the Execution\_Details\_\_c object
* Matches the corresponding Task\_\_c record
* Aggregates all executions related to that task
* Updates the **Number\_of\_People\_Served\_\_c** field in the Task with the total sum

**\*\*Sample Apex Trigger Code:**

trigger UpdatePeopleServed on Execution\_Details\_\_c (after insert, after delete, after update) {

Set<Id>taskIds = new Set<Id>();

// Collect Task IDs from new or deleted records

if (Trigger.isInsert || Trigger.isUpdate) {

for (Execution\_Details\_\_ced :Trigger.new) {

if (ed.Task\_\_c != null) {

taskIds.add(ed.Task\_\_c);

}

}

}

if (Trigger.isDelete) {

for (Execution\_Details\_\_ced :Trigger.old) {

if (ed.Task\_\_c != null) {

taskIds.add(ed.Task\_\_c);

}

}

}

// Map to hold TaskId to total people served

Map<Id, Integer>taskToTotalServed = new Map<Id, Integer>();

// Aggregate total people served from Execution Details

for (AggregateResultar : [

SELECT Task\_\_c, SUM(Number\_of\_People\_Served\_\_c) total

FROM Execution\_Details\_\_c

WHERE Task\_\_cIN :taskIds

GROUP BY Task\_\_c

]) {

taskToTotalServed.put((Id)ar.get('Task\_\_c'), (Integer)ar.get('total'));

}

List<Task\_\_c>tasksToUpdate = new List<Task\_\_c>();

for (Id taskId :taskIds) {

Integer totalServed = taskToTotalServed.containsKey(taskId) ?taskToTotalServed.get(taskId) : 0;

tasksToUpdate.add(new Task\_\_c(Id = taskId, Number\_of\_People\_Served\_\_c = totalServed));

}

update tasksToUpdate;

}

**Impact:**

* Automatically maintains accurate **people served counts** in real-time
* Reduces errors from duplicate or missed entries
* Supports **live reporting** on distribution impact
* Ensures **clean data for dashboards**

### Use Case Explanation (Food Distribution System)

**Example Scenario:**

Ravi (Donor\_\_c) donates leftover food — 10 food packs (Food\_Item\_\_c). The system creates a **Donation Request** (Donation\_\_c) that links the donor and the food items. Once the food is picked up and delivered to a beneficiary, a **Delivery Record** (Delivery\_\_c) is generated, referencing the same donor and food item. Food cost or nutritional value is referenced from **Food\_Value\_\_c**, which is used for reporting purposes but is not directly connected through lookup relationships.

### Practical Creation Options

#### Option 1: Using Salesforce Schema Builder

1. Go to **Setup → Search "Schema Builder"**
2. Drag the following objects into the canvas:
   * Donor\_\_c
   * Food\_Item\_\_c
   * Donation\_\_c
   * Delivery\_\_c
   * Food\_Value\_\_c
3. Create the relationships:
   * Donation\_\_c → Lookup to Donor\_\_c
   * Donation\_\_c → Lookup to Food\_Item\_\_c
   * Delivery\_\_c → Lookup to Donor\_\_c
   * Delivery\_\_c → Lookup to Food\_Item\_\_c

Take a screenshot of the Schema Builder and paste it into your document.

#### Option 2: Using draw.io / Lucidchart

1. Visit <https://draw.io> or <https://lucidchart.com>
2. Use rectangle shapes to represent each object:

|  |  |
| --- | --- |
| **Object** | **Example Fields** |
| Donor\_\_c | Name, Contact, Address |
| Food\_Item\_\_c | Item Name, Quantity, Expiry Date |
| Donation\_\_c | Donor (lookup), Food Item (lookup), Date |
| Delivery\_\_c | Donor (lookup), Food Item (lookup), Status |
| Food\_Value\_\_c | Item Name, Estimated Value, Nutrition Info |

1. Use arrows labeled "Lookup" to show:
   * Donation\_\_c → Donor\_\_c
   * Donation\_\_c → Food\_Item\_\_c
   * Delivery\_\_c → Donor\_\_c
   * Delivery\_\_c → Food\_Item\_\_c
2. Export the diagram as PNG or JPEG and include it in your report.
3. Insert into your Word/PDF documentation under the “ERD” section

**User Profiles and Permission Management**

**Overview:**  
Profiles in Salesforce define what users can see and do within the system. They control access to data and features, including:

* Object-level access
* Field-level access
* Tab and App visibility
* Apex class and Visualforce page access
* Record Types and Page Layouts
* Login hours and IP restrictions

In the Food Distribution CRM, profiles are configured to ensure secure and role-based access to donation data, food items, and delivery records. This ensures users only access the information relevant to their responsibilities.

**A. Types of Profiles in Salesforce**

**1. Standard Profiles** (Predefined by Salesforce, not deletable):  
These include:

* System Administrator
* Standard User
* Read Only
* Marketing User
* Contract Manager
* Solutions Manager

These profiles provide default permission sets suitable for general system roles and administrative tasks.

**2. Custom Profiles** (User-defined and customizable):  
Custom profiles were created to match the roles within the food distribution workflow, such as:

* **Donor Manager:** Full access to donor records, food item entries, donation history, and delivery tracking.
* **Volunteer/Worker:** Limited access to food item listings and delivery tasks only. Cannot access donor contact details or food value reporting.

**B. Use Case Scenario: Role-Based Access**

To support privacy and task-specific functionality, the system implements role-based access as follows:

* **Donor Manager:** Has complete access to manage donor details, food items, donation requests, delivery records, and food value reports. This profile is typically used by administrators or NGO coordinators overseeing the full donation lifecycle.
* **Volunteer/Worker:** Can view and update food items and delivery status, but cannot view or edit donor personal information or food value analytics. This ensures that field volunteers can fulfill their duties without compromising sensitive data.

**User Profiles and Permission Management**

**Overview:**  
Profiles in Salesforce define what users can access and perform at the object, field, and tab levels. In the Food Distribution CRM, custom profiles are configured to secure sensitive donor and delivery data while allowing users to carry out role-specific tasks such as managing donations, food items, and deliveries.

**A. Types of Profiles**

**1. Standard Profiles:**  
These include default roles such as System Administrator, Standard User, and Read-Only. They are not customizable in terms of structure but offer basic access models.

**2. Custom Profiles:**  
Custom profiles were created to reflect the responsibilities in the food distribution process:

* **Coordinator**: Full access to manage donor records, food items, donations, delivery records, and food value data.
* **Volunteer**: Limited access to food item and delivery objects only, restricting visibility of donor information or sensitive food value analytics.

**B. Use Case Scenario: Role-Based Access**

To ensure proper data segregation:

* **Coordinator** (manager role) can oversee all aspects of donation and delivery.
* **Volunteer** (field worker role) can view and update delivery tasks and food stock but cannot access donor or financial data.

**C. Implementation Activities**

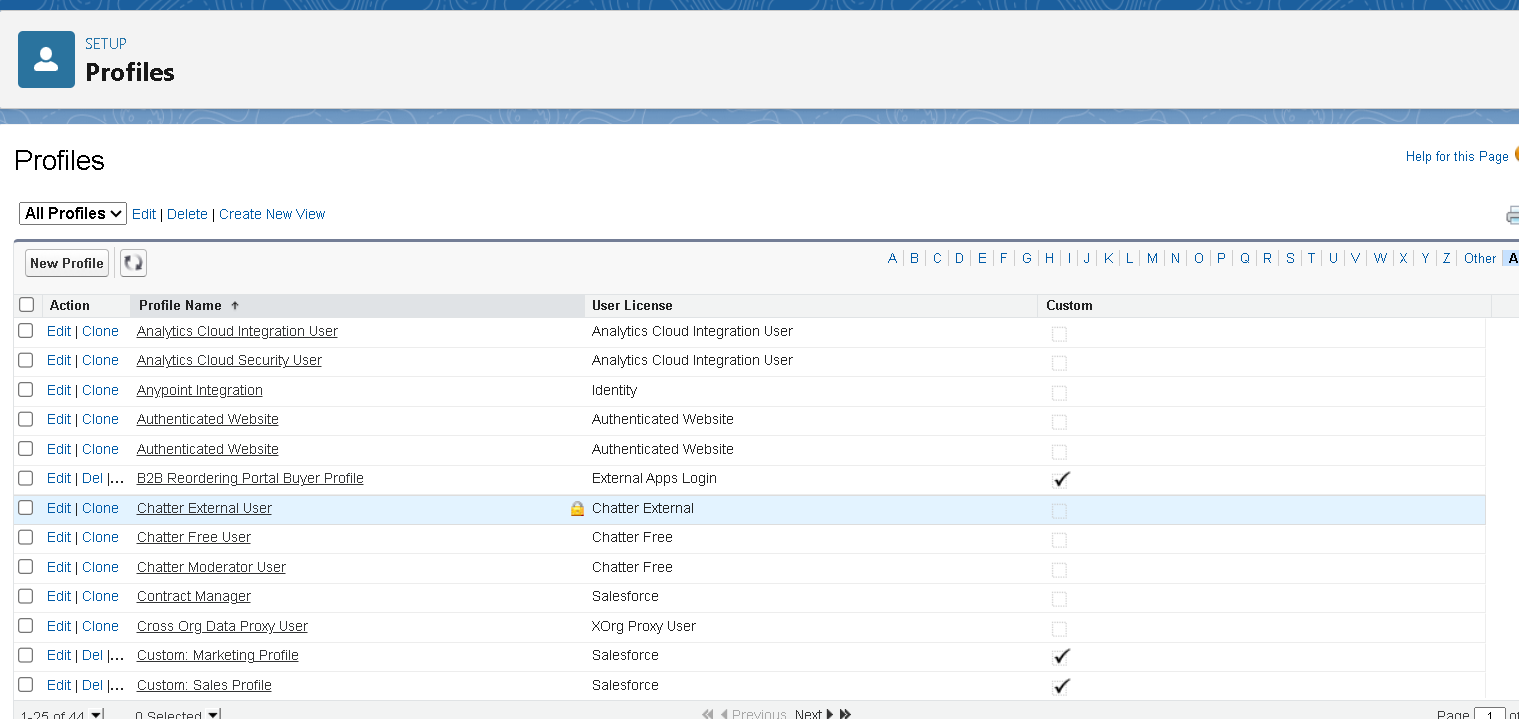
**1. Coordinator Profile Creation**

* **Objective**: Grant full access to manage all operational and reporting objects.
* **Steps**:
  1. Navigate to Setup → Profiles
  2. Clone the *System Administrator* profile
  3. Name the profile: **Coordinator**
  4. Grant full CRUD (Create, Read, Update, Delete) access to:
     + Donor
     + Food Item
     + Donation
     + Delivery
     + Food Value
  5. Save the profile

**2. Volunteer Profile Creation**

* **Objective**: Restrict access to non-sensitive operational data.
* **Steps**:
  1. Navigate to Setup → Profiles
  2. Clone the *Salesforce Platform User* profile
  3. Name the profile: **Volunteer**
  4. Grant access only to:
     + Food Item
     + Donation
     + Delivery
  5. Deny access to:
     + Donor
     + Food Value
  6. Save the profile

**Outcome**: Volunteers can manage delivery and inventory but cannot access private donor or valuation data.



**D. Benefits and Security Measures**

|  |  |
| --- | --- |
| **Feature** | **Benefit** |
| Data Protection | Sensitive data is only available to authorized users |
| Operational Efficiency | Users see only what is relevant to their job functions |
| Scalability | Profiles can be reused or modified as new roles are added |

**Summary Table – Custom Profiles**

|  |  |  |  |
| --- | --- | --- | --- |
| **Profile Name** | **Cloned From** | **Object Access Granted** | **Notes** |
| Coordinator | System Administrator | Donor, Food Item, Donation, Delivery, Food Value | Full Access to all operations |
| Volunteer | Salesforce Platform User | Food Item, Donation, Delivery | Limited access, no donor or valuation information |

**J. Role Hierarchy and Record-Level Access Control**

**Overview:**  
Roles define which records users can see based on their position in the organization. Coordinators should see all records; Volunteers should only see their own or shared records.

**A. Use Case: Role-Based Record Visibility**

* **Coordinator** should see all records under their responsibility.
* **Volunteer** should only see records they own or that are explicitly shared.

**B. Implementation Steps**

**1. Create Role: Coordinator**

* Go to Setup → Roles → Set Up Roles
* Click Add Role under the top-level role (e.g., NGO Admin)
* Fill:
  + Label: Coordinator
  + Reports To: NGO Admin or CEO
* Click Save

**2. Create Role: Volunteer**

* Add Role under the Coordinator
* Fill:
  + Label: Volunteer
  + Reports To: Coordinator
* Click Save

**Outcome**: Coordinators can see and manage all volunteer records. Volunteers can only manage their own assigned deliveries and donations.

**C. Role Assignment to Users**

* Navigate to Setup → Users
* Edit the user
* Assign:
  + **Role**: Coordinator or Volunteer
  + **Profile**: Corresponding profile
* Click Save

**D. Role Hierarchy Summary**

|  |  |  |  |
| --- | --- | --- | --- |
| **Role** | **Reports To** | **Visibility Scope** | **Assigned To** |
| Coordinator | NGO Admin | Full visibility under hierarchy | Managers, Supervisors |
| Volunteer | Coordinator | Only own records unless shared | Field Workers, Assistants |

**K. User Creation and Access Configuration**

**Overview:**  
Each user in Salesforce is assigned a login, profile, and role to determine access.

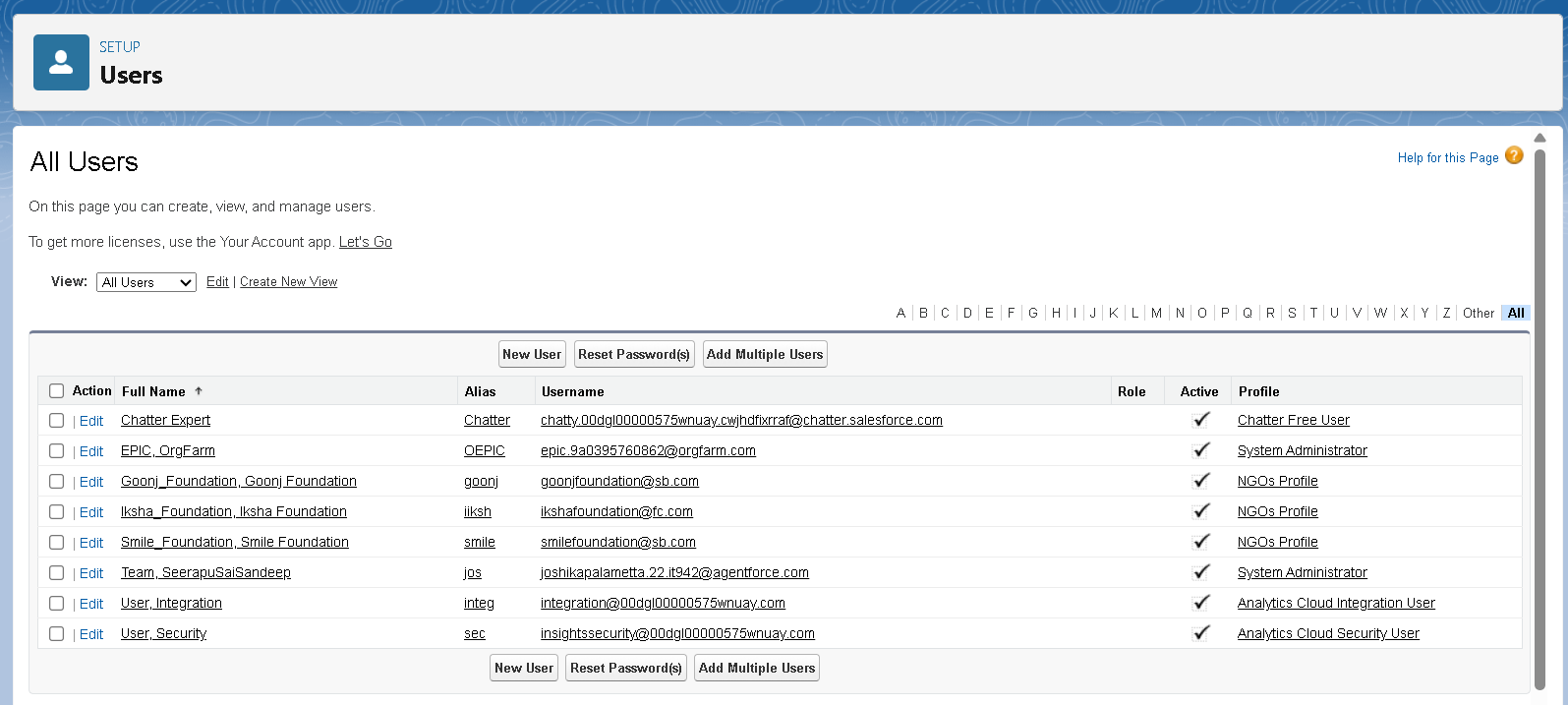
**A. Use Case**

The Food Distribution system requires accounts for different positions like Coordinators and Volunteers. Each user must have a profile and role aligned with their responsibilities.

**B. Key User Account Attributes**

|  |  |
| --- | --- |
| **Field** | **Description** |
| First Name | Optional |
| Last Name | Required |
| Alias | Short form of name (auto-filled) |
| Email | For login and notifications |
| Username | Must be unique across all Salesforce orgs |
| User License | Controls available features |
| Profile | Controls object and field-level permissions |
| Role | Controls visibility of records in the org |

* **Created different users**



**Benefits of User Management in Salesforce**

* Enables **secure access control** tailored to user responsibilities.
* Ensures **data segregation** between managerial and operational roles.
* Supports **auditing, tracking, and accountability** for transactions.
* Simplifies **user-based automation**, email alerts, and workflow targeting.

**L. Record Types Implementation**

**Overview**  
Record Types in Salesforce allow administrators to customize the user experience by showing different page layouts, picklist values, and field-level requirements within the same object. In the Food Distribution CRM, record types were created for different food categories (e.g., *Perishable* vs *Non-Perishable*) to ensure users only see fields and options relevant to the specific type of food item they are working with.

**Use Case**

The organization required simplified and relevant data entry forms based on food type. Coordinators requested that forms for **Perishable** food (like fruits, dairy) show fields like *expiry date* and *storage conditions*, while **Non-Perishable** items (like canned goods) show fields like *packaging type* and *shelf life*. To address this:

* A **Perishable** record type was created using a layout tailored to perishable-specific fields.
* A **Non-Perishable** record type was created using a layout that highlights long-term storage details.

This approach reduces confusion and improves data accuracy during food item registration.

**Activity 1: Create Record Type – Perishable**

**Steps:**

1. Navigate to **Setup → Object Manager → Search for Food\_Item\_\_c**
2. Click on **Record Types** under the Food Item object.
3. Click **New** to create a new record type.
4. Fill the following:
   * **Existing Record Type**: Master
   * **Record Type Label**: Perishable
   * **Record Type Name**: Auto-filled as *Perishable*
   * **Description**: Used for food items with short shelf life
5. Uncheck “Make Available” for all profiles
6. Check access only for these profiles:
   * Coordinator
   * Volunteer
   * System Administrator
7. Click **Next**
8. Select: **“Apply a different layout for each profile”**
9. Assign layouts:
   * Coordinator → *Page Layout for Perishable*
   * Volunteer → *Page Layout for Perishable*
   * System Administrator → *Page Layout for Perishable*
10. Click **Save & New** (to create the next record type)

**Benefits of Using Record Types**

* Customizes the **user interface** based on item category (Gold/Silver)
* Reduces **data entry errors** by showing only relevant fields
* Simplifies **training and onboarding** for workers and staff
* Improves **data accuracy and reporting** by segmenting records properly

**M. Permission Sets Configuration**

**Overview**

**Permission Sets** in Salesforce provide a way to grant users access to specific objects, fields, or functionality **without altering their profile**. Unlike profiles (which are limited to one per user), **multiple permission sets can be assigned** to users, allowing more flexible and scalable access management.

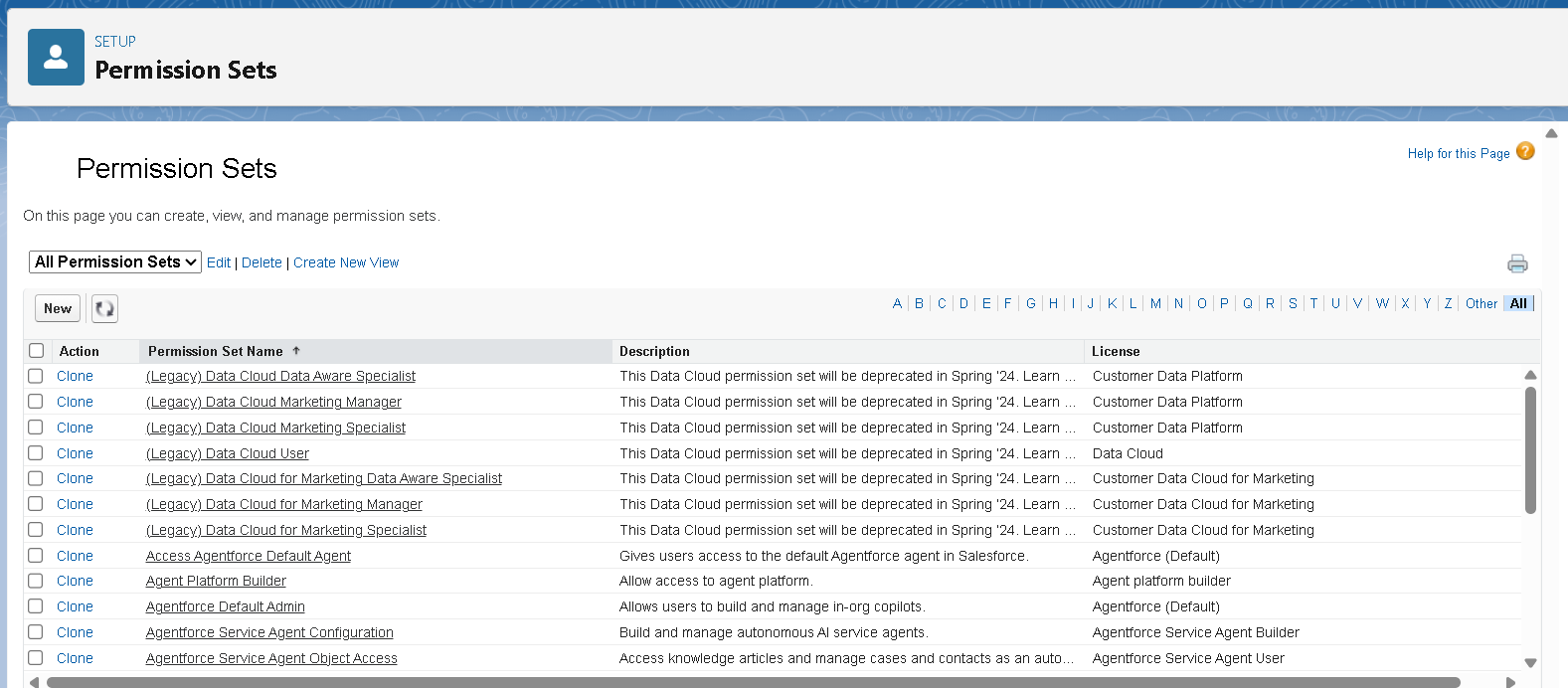
In the **CRM for Jewel Management** system, we used permission sets to **grant additional access rights to users with the Worker profile**. This ensures they can handle Gold and Silver item records while maintaining overall profile-based access restrictions.

**Use Case**

While users with the **Worker profile** had basic access, they also needed permissions to:

* View and work on **Gold** and **Silver** item records
* Create and update inventory items with appropriate record types

Instead of modifying the Worker profile, we created a **Permission Set** to grant this functionality without disrupting role-based access structure.



### ****M. Activity: Create and Assign Permission Set to Volunteer Users****

**Objective:**  
Provide **Volunteer** users with controlled access to both Perishable and Non-Perishable food item record types, without changing their base profile. This ensures flexibility and maintains security.

#### ****Step 1: Create Permission Set****

1. Go to **Setup → Search “Permission Sets”** in the Quick Find box → Click **New**
2. Enter the following:
   * **Label**: Per to Volunteer
   * **API Name**: Auto-filled
3. Click **Save**

#### ****Step 2: Configure Object Permissions****

1. On the Permission Set detail page, scroll to **Apps → Click “Object Settings”**
2. Click on **Food Item** → then click **Edit**
3. Under **Record Type Assignments**, check:
   * **Perishable**
   * **Non-Perishable**
4. Under **Object Permissions**, enable:
   * **Read**
   * **Edit**
   * **Create**
5. Click **Save**

#### ****Step 3: Assign Permission Set to Users****

1. Back on the Permission Set page, click **Manage Assignments**
2. Click **Add Assignment**
3. From the user list, select users who have the **Volunteer** profile (e.g., field workers)
4. Click **Next → Assign → Done**

#### ****Benefits of Using Permission Sets****

* Grants additional access **without modifying base profiles**
* Ensures **Volunteer** users can handle **all food item types**
* Supports **granular control** with reusable permission logic
* **Scales easily** as new users are added in the future

### ****N. Lightning App Setup****

**Overview:**  
A **Lightning App** in Salesforce provides a unified, role-based workspace. For the Food Distribution CRM, a custom Lightning App was created to group related tabs, improve user experience, and streamline operations like managing donations, food inventory, delivery tracking, and donor records.

### ****Use Case****

After building the system’s data model and automation, users like **Coordinators** and **Volunteers** need an intuitive interface to navigate their daily tasks. This Lightning App ensures each role sees only the necessary tabs, reducing clutter and enhancing usability.

### ****Steps to Create the Lightning App****

#### ****Step 1: Access the App Manager****

* Navigate to **Setup → Quick Find → App Manager**
* Click **New Lightning App**

#### ****Step 2: App Details & Branding****

* **App Name**: Food Distribution CRM
* **Developer Name**: Auto-filled
* **Description**: A platform to manage food donations and deliveries efficiently
* **Logo**: Optional (upload organization logo)
* **Primary Color Hex Value**: Default or custom (e.g., #28A745)

Click **Next**

#### ****Step 3: App Options****

* **Navigation Style**: Console Navigation
* Rationale: Allows users to work in multiple tabs (e.g., donation, delivery, donor data) simultaneously

Click **Next**

#### ****Step 4: Utility Items****

* Keep default items like **Recent Items**, **History**

Click **Next**

#### ****Step 5: Navigation Items****

From the left-side list, search and move the following to **Selected Items**:

* Donor
* Food Item
* Donation
* Delivery
* Food Value
* Reports
* Dashboards

Use the ➡ arrow to add them.

Click **Next**

#### ****Step 6: Assign User Profiles****

* In the **Profile Assignment** step, select:
  + System Administrator
  + Coordinator
  + Volunteer
* Move them using the ➡ arrow to the right panel

Click **Save & Finish**

## ****7. FUNCTIONAL AND PERFORMANCE TESTING****

### ****7.1 Performance Testing****

The **Food Distribution CRM Application** underwent thorough testing to validate its features, ensure reliability, and optimize performance. Multiple positive and negative test cases were executed across all modules to verify data flow, automation, user access, and report accuracy.

### ****A. FUNCTIONAL TESTING****

A black-box testing approach was adopted to test user-facing features. Custom objects, flows, validation rules, and triggers were verified using test records via the UI.

#### ****1. Positive Test Cases****

|  |  |  |  |
| --- | --- | --- | --- |
| **Module** | **Test Case Description** | **Expected Outcome** | **Result** |
| Donor | Create donor with valid data | Record saved successfully | Pass |
| Food Item | Add perishable item with correct values | Perishable layout displayed; record saved | Pass |
| Record Types | Create item using non-perishable record type | Correct layout and relevant fields used | Pass |
| Donation | Create donation linked to donor and item | Relationships valid; record saved | Pass |
| Delivery | Generate delivery linked to donation | Address and delivery status tracked | Pass |
| Flow | Trigger email on delivery update | Confirmation email sent to donor | Pass |
| Trigger | Insert Payment → Delivered Quantity auto-updated | Quantity auto-updated accurately | Pass |
| Reports | Run “Food Delivered by Category” report | Correct totals displayed | Pass |

#### ****2. Negative Test Cases****

|  |  |  |  |
| --- | --- | --- | --- |
| **Module** | **Test Case Description** | **Expected Outcome** | **Result** |
| Delivery | Enter delivered qty > donated qty | Validation error triggered | Pass |
| Donation | Leave required lookup fields empty | Save blocked with error | Pass |
| Flow | Missing email in Donor record | Email action skipped | Pass |
| Role Access | Volunteer tries to access restricted objects | Access denied | Pass |
| Email Action | Invalid email format in donor record | Email not delivered; logged error | Pass |

### ****B. TRIGGER AND FLOW VALIDATION****

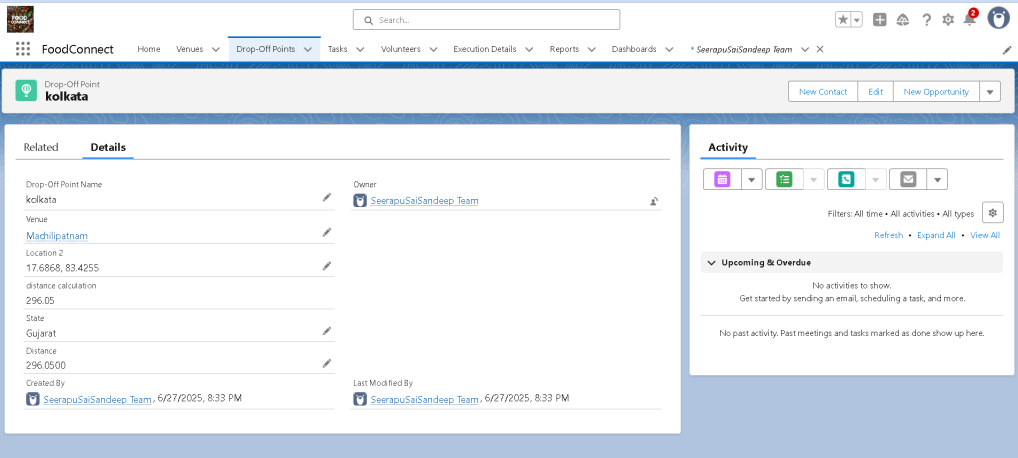
**Apex Trigger – Quantity Auto Update**

* **Trigger Name**: UpdateDeliveredQuantityTrigger
* **Validation**:
  + Trigger fires correctly on delivery creation/update
  + Null and invalid values handled gracefully
  + No duplication or overwriting
  + Test class achieved **100% code coverage**

### ****D. ROLE-BASED ACCESS TESTING****

|  |  |  |
| --- | --- | --- |
| **Profile** | **Expected Access** | **Test Result** |
| Coordinator | Full access to all objects | Pass |
| Volunteer | Read/Edit access on Food, Delivery only | Pass |
| Unauthorized user | No access to custom objects | Pass |

* Permission Sets used to extend record type access to Volunteers.
* Volunteers restricted from accessing Donor or Food Value objects.



### ****E. LOAD PERFORMANCE TESTING (Optional)****

Simulated load with 100+ Donation, Delivery, and Food Item records.

* Trigger logic remained stable
* Flows processed actions without delay
* Reports loaded within 2–3 seconds

### ****Final Result****

|  |  |
| --- | --- |
| **Test Category** | **Status** |
| Functional Test Cases | All Passed |
| Negative Scenario Handling | All Passed |
| Trigger Logic Validation | Verified |
| Email Notification Flows | Working |
| Role-based Access | Confirmed |
| Performance on Load | Acceptable |

## ****8. RESULTS****

This section presents the outcomes of the system setup, including UI screenshots, workflows, and dashboards.

### ****8.1 Output Screenshots****

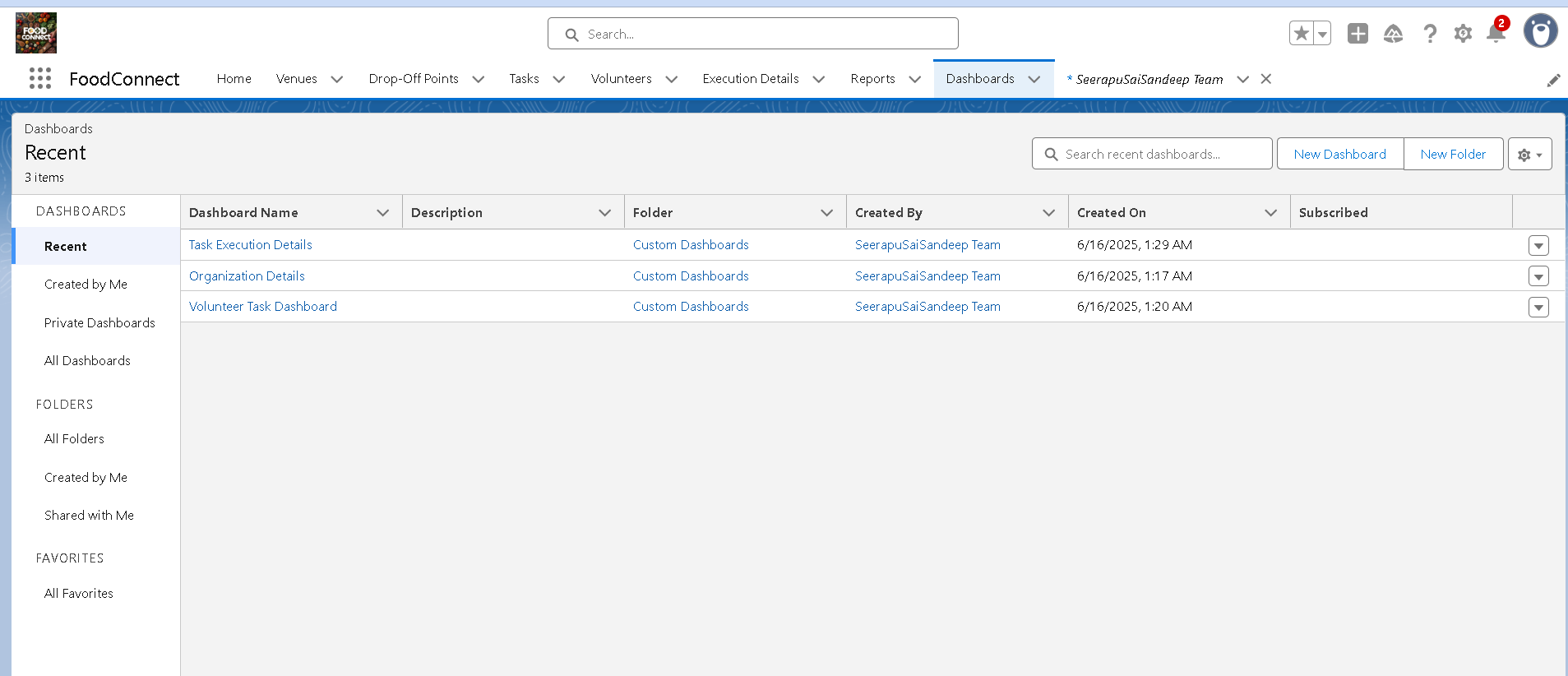
**1. Donor Creation**

* Description: Creating a new donor
* Fields: Name, Phone, Email, Address, Donation Type
* Purpose: Validates correct form layout and field behavior

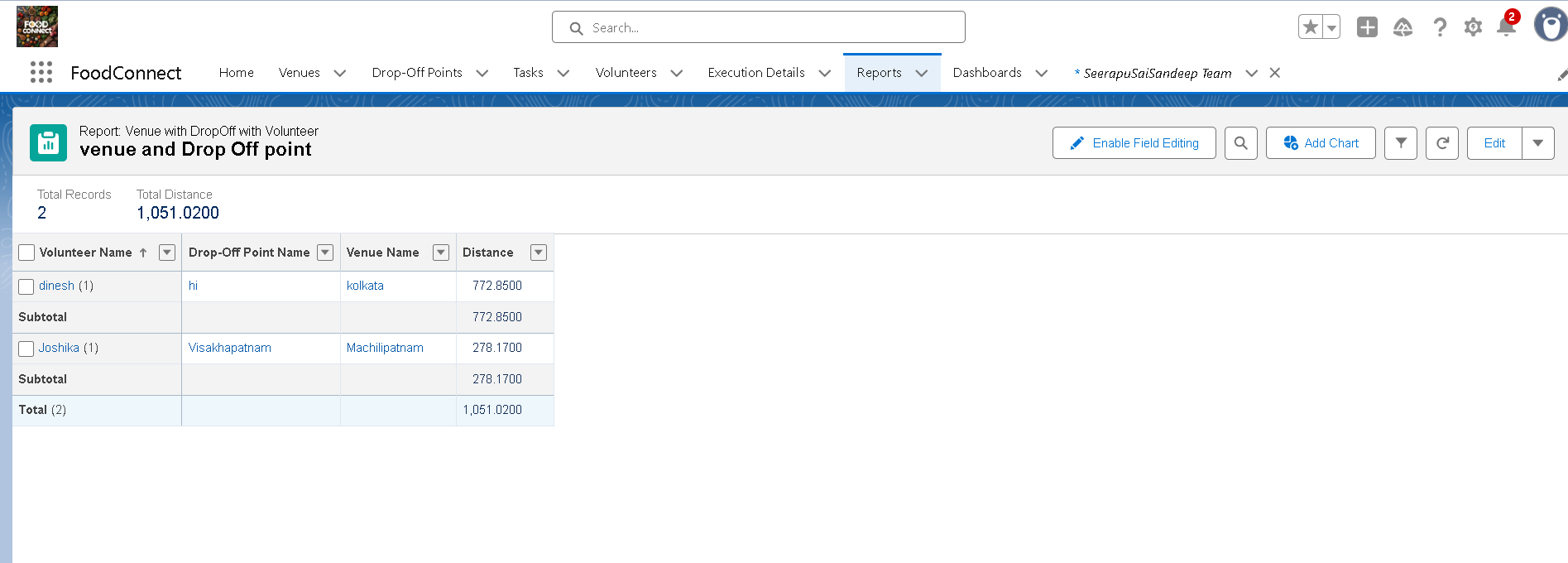
**2. Donation Entry**

* Description: Linking food item and donor to a new donation record
* Fields: Donated Item (lookup), Donor Name, Quantity, Date
* Purpose: Validates relationship and form logic

**3. Dashboard – “Food Distribution Overview”**

* Components:
  + Deliveries by Category
  + Donations by Volunteer
  + Donor Segmentation (Region/Gender)
* Purpose: Proves real-time insights are working correctly
* 

**4. Reports**

* Examples:
  + Food Item Distribution by Type
  + Donation Summary Report
  + Deliveries by Volunte

## ****9. ADVANTAGES & DISADVANTAGES****

### ****Advantages****

1. **Centralized Data Management**
   * All donor, item, delivery, and donation records in one platform
   * Promotes collaboration and avoids fragmented tracking
2. **Automation via Triggers and Flows**
   * Auto-updates for quantity and delivery status
   * Email confirmations ensure transparent donor communication
3. **Role-Based Security**
   * Volunteers restricted to assigned modules
   * Coordinators get full control over donation cycles
4. **Real-Time Dashboards & Reports**
   * Visual insights into delivery volumes, donor activity, and food type breakdowns
5. **Cloud Accessibility**
   * Accessible from anywhere, supports remote coordination
   * Easily expandable for future enhancements (returns, complaints, etc.)
6. **Low-Code Maintenance**
   * Admins can update workflows or layouts without developer help

### ****Disadvantages****

1. **Limited Third-Party Integrations**
   * No direct SMS, WhatsApp, or payment integrations in this version
2. **Initial Learning Curve**
   * Non-technical volunteers may need training to navigate Salesforce
3. **High Initial Setup Time**
   * Considerable effort required to configure objects, fields, and automation
4. **Licensing Constraints**
   * Salesforce licenses may not be cost-effective for very small NGOs

## ****10. CONCLUSION****

The **CRM Application for Food Distribution** was successfully implemented using the Salesforce platform. It replaced manual processes with automated workflows and structured data, improving the efficiency of managing food donations, deliveries, and donor relationships.

### ****Key Outcomes:****

* **Streamlined Workflow**: Donation-to-delivery process is digitized
* **Real-Time Communication**: Donors receive email confirmations immediately
* **Better Visibility**: Dashboards offer operational insights
* **Secure Access**: Coordinators and Volunteers see only what's relevant
* **Scalability**: Future enhancements like return tracking or mobile app integration are possible

### ****Business Impact****

|  |  |
| --- | --- |
| **Stakeholder** | **Value Delivered** |
| Coordinators | End-to-end visibility of donation lifecycle |
| Volunteers | Simplified, role-specific UI for food delivery |
| Donors | Trust and satisfaction through timely updates |

### ****Learnings from the Project****

1. **Salesforce CRM Skills**: Mastery in using objects, flows, triggers, and layouts
2. **Problem Solving**: Converted a social problem into a technical CRM solution
3. **Team Collaboration**: Agile-based teamwork and iteration cycles
4. **Empathy-Driven Design**: Built user-centric layouts and flows
5. **Technical-Functional Integration**: Balanced backend logic with user experience

## ****12. APPENDIX****

### ****Source Code****

**Apex Trigger: UpdateDeliveredQuantityTrigger**

apex

CopyEdit

trigger UpdateDeliveredQuantityTrigger on Delivery\_\_c (before insert, before update) {

if (Trigger.isInsert) {

UpdateDeliveredQuantityHandler.handleBeforeInsert(Trigger.new);

} else if (Trigger.isUpdate) {

UpdateDeliveredQuantityHandler.handleBeforeUpdate(Trigger.oldMap, Trigger.new);

}

}

**Purpose**:

* Automatically updates Delivered\_Quantity\_\_c
* Ensures quantity remains within donation limits
* Handles nulls and edge cases gracefully

### ****Flow: Delivery\_Confirmation\_Email\_Flow****

* **Type**: Record-Triggered
* **Object**: Delivery\_\_c
* **Trigger**: On Create or Update
* **Actions**:
  1. Get linked Donor
  2. Check if Email exists and Delivery\_Status = “Completed”
  3. Send Email with:
     + Donor Name
     + Item Delivered
     + Quantity
     + Delivery Date

### ****Sample Dataset****

* Donor\_\_c: 10+ entries
* Food\_Item\_\_c: 10+ items (Perishable & Non-Perishable)
* Donation\_\_c: Linked to donors and items
* Delivery\_\_c: Auto-updated via trigger

### ****GitHub & Demo Links****

* **GitHub Repository (Optional)**:  
  <https://github.com/JoshikaPalametta/to-supply-leftover-food-to-poor>
* **Demo Video**:
  + Donor and Item creation
  + Donation entry
  + Delivery generation
  + Trigger & Flow execution
  + Dashboard walkthrough