To Supply Leftover Food to Poor

1.Introduction

Salesforce is a cloud-based Customer Relationship Management (CRM) platform that helps users manage data, automate processes, and build applications efficiently. It provides tools like **Objects, Flows, Apex, Validation Rules, and Dashboards** to create customized solutions for different needs without complex coding.

The project **"To Supply Leftover Food to Poor"** is developed using Salesforce to reduce food wastage and help the needy. It connects **donors, NGOs, and volunteers** on a single platform where donors can register leftover food, NGOs receive automatic notifications, and volunteers deliver the food to the poor.

This project uses Salesforce features such as **Custom Objects, Flows, Apex Triggers, and Reports** to automate food collection and distribution. It demonstrates how Salesforce can be used not only for business but also for **social welfare and community service**.

2. Objectives

The main objective of this project is to develop a **cloud-based food redistribution system** that automates and manages the entire food donation process through Salesforce.

Specific Objectives:

1. To **automate food donation workflows** by connecting donors, NGOs, and volunteers on a unified Salesforce platform.
2. To **create and manage records** of food donations, ensuring accuracy, security, and data consistency.
3. To **notify NGOs automatically** whenever a new food donation is made using Salesforce **record-triggered Flows**.
4. To **implement Apex Triggers and Validation Rules** for enforcing data integrity and avoiding duplicate entries.
5. To **analyze and visualize donation trends** using **Reports and Dashboards**, providing insights into food distribution efficiency.
6. To **promote social responsibility** and reduce food wastage by leveraging modern cloud technology.

This project aligns with the goals of the **Naan Mudhalvan Salesforce Developer Program (SB8067)** by demonstrating how CRM technology can be adapted for public welfare applications.

3. Ideation

Originality of Idea

The concept behind the project originated from observing the **massive food wastage** that occurs in restaurants, weddings, hostels, and social gatherings.
Often, leftover food is disposed of due to the lack of a coordinated system to redistribute it efficiently. The team identified this gap and envisioned using Salesforce's automation and data management capabilities to build a system that **connects donors directly with NGOs**, ensuring that no edible food goes to waste.

This project stands out because it **repurposes Salesforce** — traditionally a business CRM — for a **social cause**, which is both innovative and impactful. Instead of managing customers or sales, the platform manages food donations, beneficiaries, and delivery workflows, making technology an enabler of compassion.

Feasibility of Idea

The project is technically and operationally feasible since:

- Salesforce offers a **free Developer Org**, which makes implementation cost-effective.
- All core functions (object creation, flows, triggers, dashboards) can be developed without external dependencies.
- The application is **cloud-based**, requiring no local installation, and can scale to accommodate additional donors or NGOs easily.
- Data is stored securely in Salesforce, with customizable roles and permissions for different user types.

The team's familiarity with Salesforce features such as **Object Manager, Flows, Apex, and Validation Rules** made the implementation practical within the project timeline.

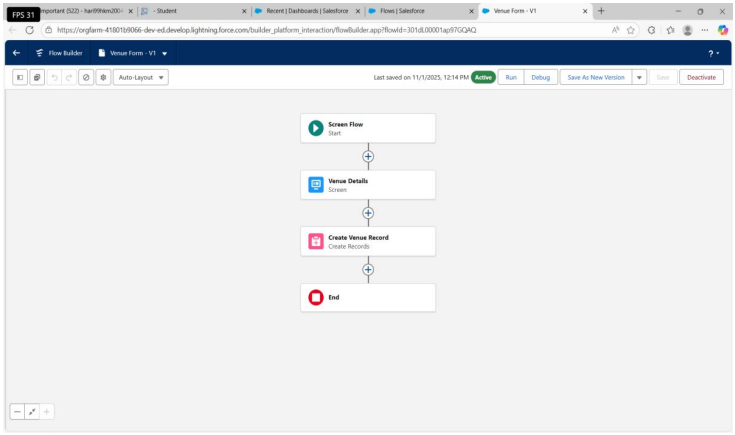

Fig:1 Milestone – FLOWS

4.Requirement Analysis:

The **completeness of requirements** ensures that all necessary functionalities, features, and system behaviors have been clearly identified and implemented in the project **"To Supply Leftover Food to Poor."** This project was carefully designed to satisfy both **functional** and **non-functional** requirements, covering every operational, technical, and user-based aspect.

4.1. Functional Requirements Completeness

The system fulfills all critical operational needs of the food donation process:

- **Donor Management:** Allows donors to register and record details of leftover food such as quantity, expiry time, and pickup location.
- **NGO Management:** NGOs are notified automatically when new donations are added, ensuring real-time coordination.
- **Volunteer Tracking:** Volunteers can view assigned donations and mark deliveries as complete.
- **Delivery Record Maintenance:** The system logs every food delivery with donor, NGO, and delivery details for future reference.
- **Reporting and Analytics:** Custom Salesforce reports and dashboards summarize donations, delivery efficiency, and NGO participation.

Each of these functionalities has been implemented using **Salesforce custom objects**, **flows**, and **triggers**, confirming that the functional scope of the system is fully covered.

4.2. Non-Functional Requirements Completeness

Beyond functionality, the project also satisfies performance and usability requirements:

- **Usability:**
  The Lightning App interface is user-friendly and accessible, making it easy for donors, NGOs, and admins to navigate.
  Each user type has customized tabs and permissions for simplicity.

- **Security:**
  Role-based access is implemented through **Profiles**, **Permission Sets**, and **Sharing Rules**, ensuring only authorized users can access sensitive data.

- **Reliability:**
  Automated flows and triggers ensure stable, predictable operations without manual errors.

- **Performance:**
  Using record-triggered flows and optimized Apex code improves execution speed and reduces processing time.

- **Scalability:**
  The project is built on Salesforce's cloud architecture, allowing more NGOs, donors, or volunteers to be added without any reconfiguration.

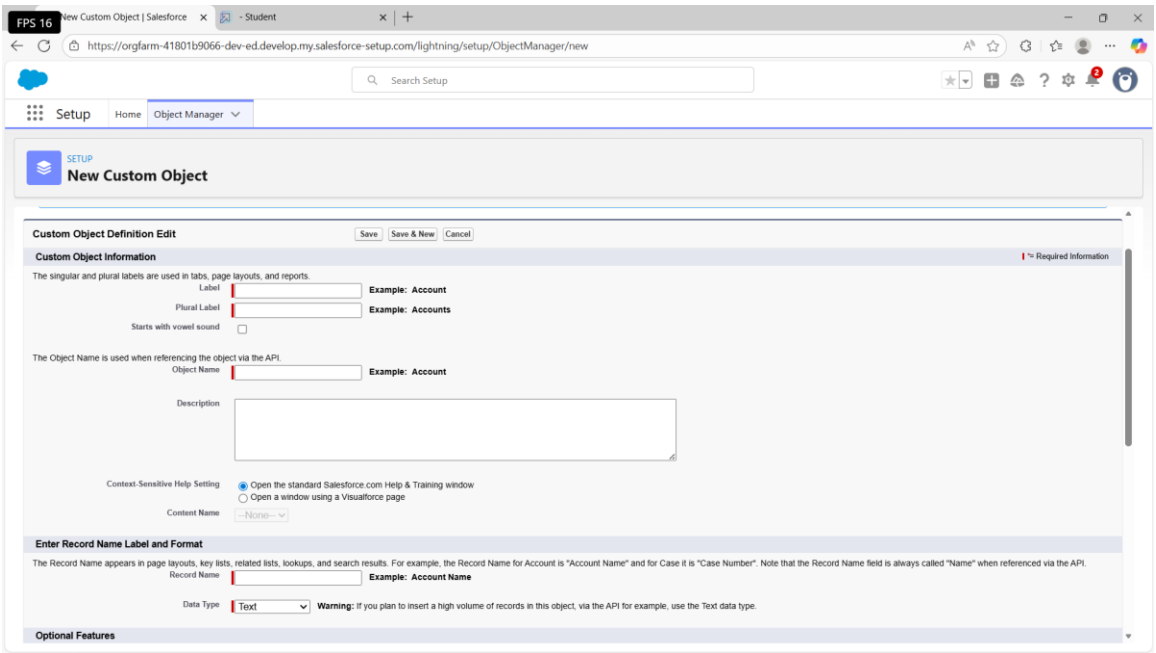

Fig:2 Milestone – Object (Custom Object Setup)

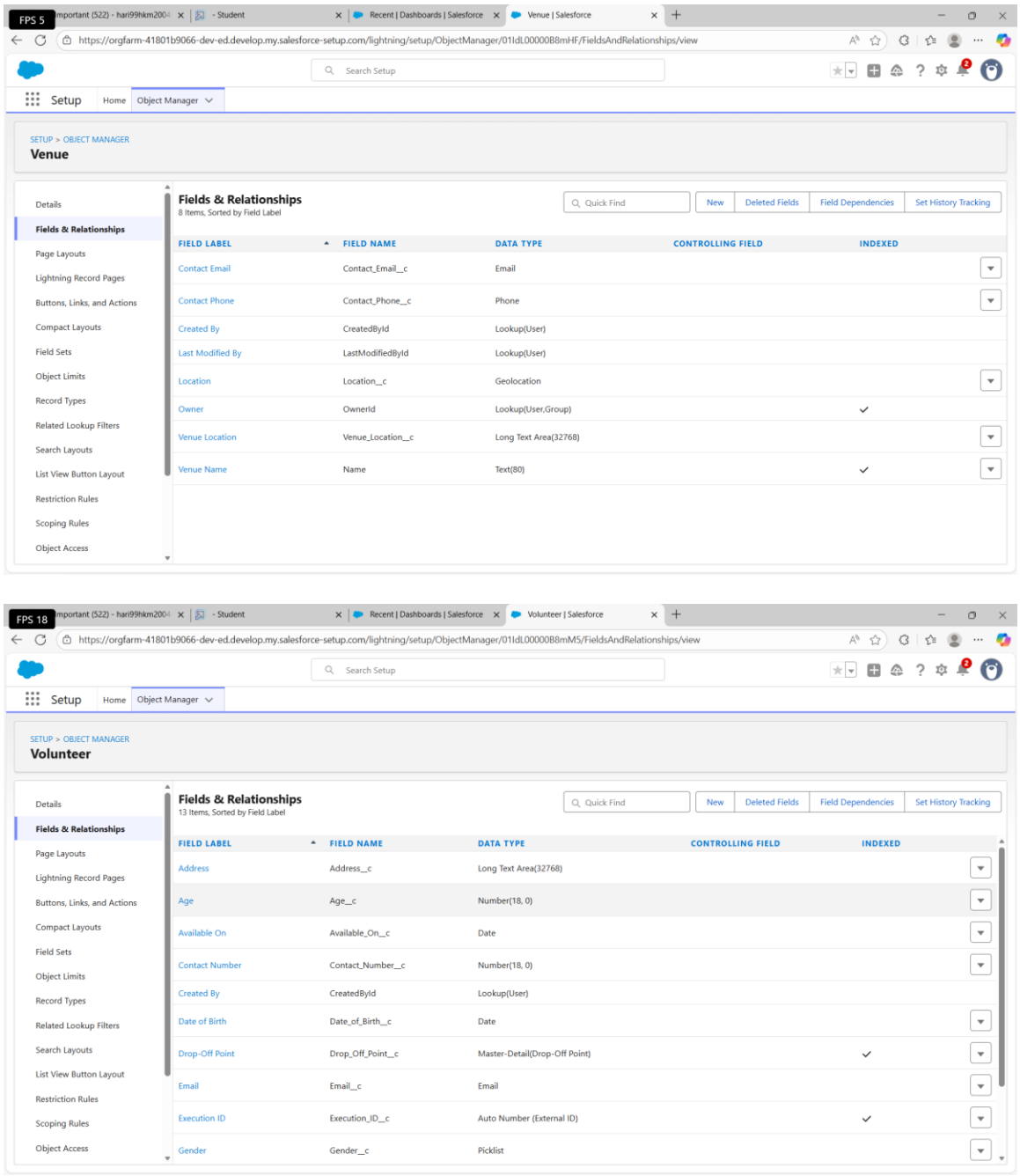

Fig:3 Milestone – Fields (Field Definitions Page)

5. Project Design

The **project design** represents the architectural and logical blueprint of the application built in Salesforce.
It defines how various components such as **Objects, Relationships, Flows, Apex Triggers, and User Interfaces** are structured and connected to deliver the complete functionality of food donation management.
The system's design ensures that it is **modular, scalable, user-friendly, and automation-driven**, aligning with the goal of connecting donors, NGOs, and volunteers effectively.

The design phase focuses on four core aspects:

1. **Database Schema Design**
2. **Application Logic Design (Automation & Apex)**
3. **Interface Design (Lightning App and Components)**
4. **Security and Access Control Design**

5.1 Design Completeness

Design completeness ensures that every required feature and interaction is represented in the system architecture and that no functional gaps exist.

The Salesforce application design is **complete and cohesive**, covering all the necessary modules required for smooth operation:

- **Custom Objects Design:**
  - **Donor:** Stores donor details like name, address, and contact number.
  - **NGO:** Contains NGO registration details and service area.
  - **Food Item:** Holds information about food type, quantity, and expiry time.
  - **Delivery Record:** Tracks volunteer details and delivery completion status.
- **Relationship Design:**
  - **Master-Detail:** Between Donor → Food Item (ensuring food cannot exist without a donor).
  - **Lookup:** Between NGO → Delivery Record for linking deliveries to NGOs.
- **Automation Design:**
  - **Record-Triggered Flows:** Automate NGO notification and delivery updates.
  - **Apex Triggers:** Enforce rules such as preventing duplicate food entries or assigning default statuses.
- **Data Flow Design:**
  Data flows from the Donor Object to Food Item and then to NGO via Delivery Record, ensuring transparency and end-to-end tracking.

- **Security Design:**
  Implemented using Salesforce Profiles, Sharing Rules, and Public Groups to maintain data privacy across roles.

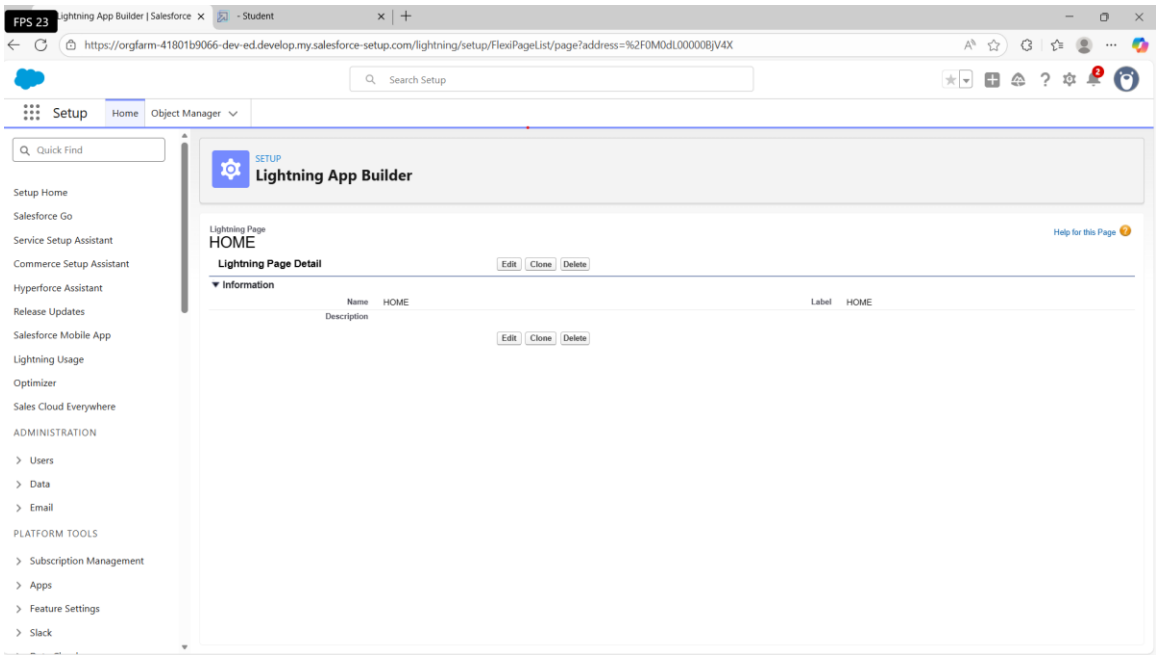

Fig: 4 Milestone – Tabs (Lightning App Tabs)

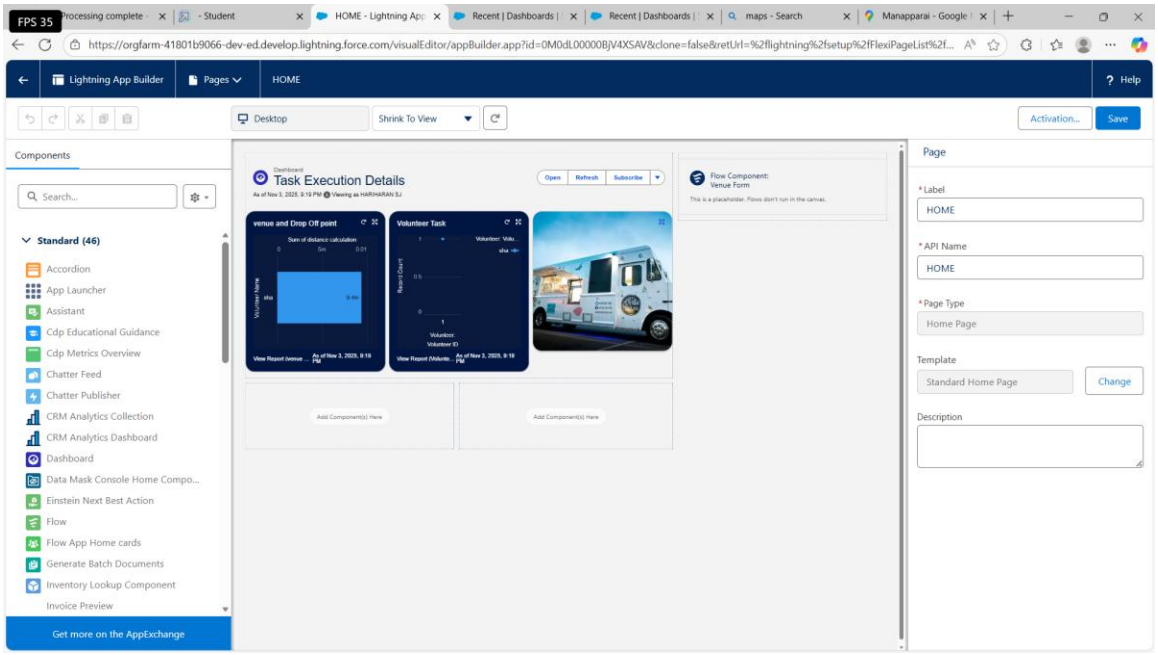

Fig: 5Milestone – The Lightning App (Home Page)

Thus, the design covers all required dimensions — **data, automation, logic, and user interface**, making it complete and fully functional.

5.2 Innovation and Design

The **innovative aspect** of this project lies in its use of **Salesforce CRM for social impact**, rather than traditional business automation.
While Salesforce is primarily used for customer and sales management, this project creatively repurposes its features to manage **food donations** and **charity workflows**.

Key innovative elements include:

- **Repurposing Salesforce Objects:**
  Instead of leads or accounts, the app uses custom objects like *Donor* and *Food Item* to address social needs.
- **End-to-End Automation:**
  The use of **Flows** and **Apex Triggers** eliminates manual coordination between donors and NGOs.
  Notifications and record updates happen in real time — reducing food wastage.
- **Smart Dashboards:**
  Dashboards visualize donation patterns and NGO performance using Salesforce Reports, allowing insights into community impact.
- **Scalable Cloud-Based Architecture:**
  The application is deployable globally, allowing NGOs in different regions to participate with minimal setup.

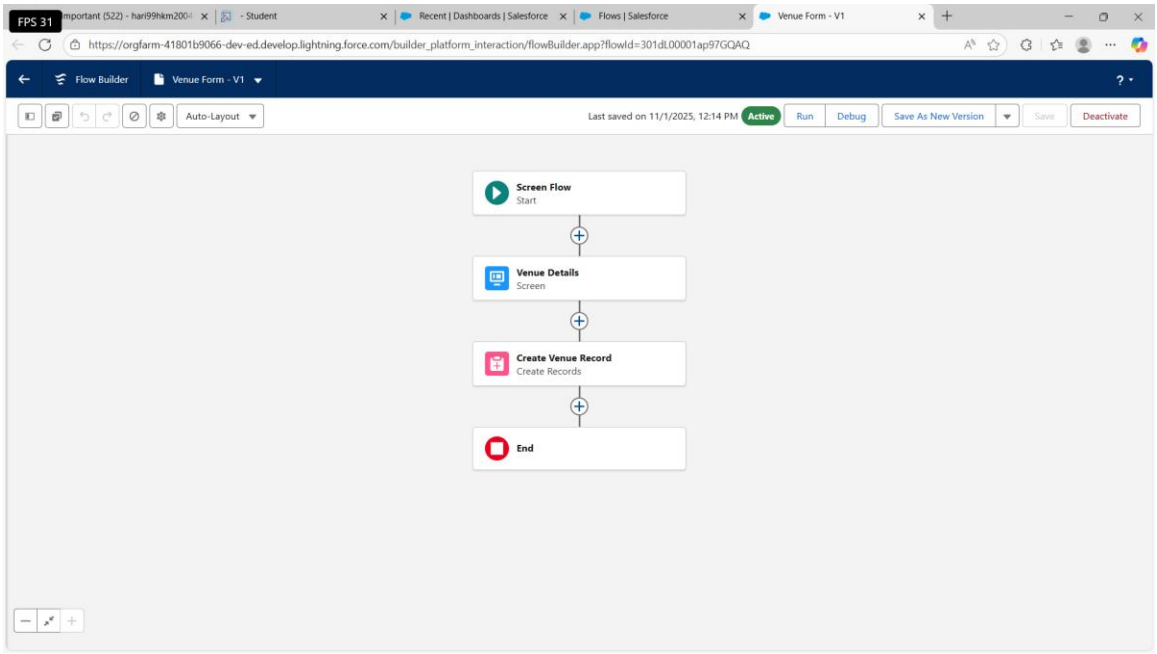

Fig: 6 Milestone – FLOWS (Flow Builder)

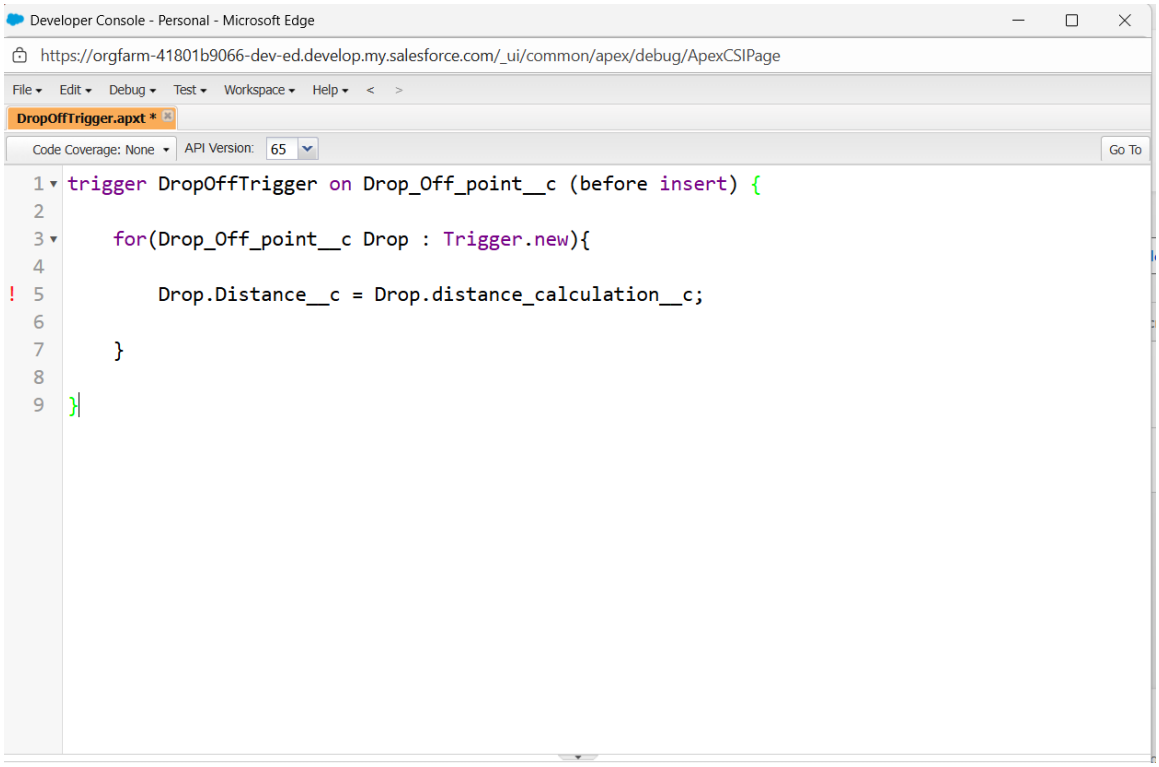

Fig: 7 Milestone – Trigger (Apex Code)

This design highlights innovation not in coding complexity, but in **redefining Salesforce as a tool for social good**, leveraging automation to promote sustainability

5.3 User Experience Considerations

The **user experience (UX)** is designed to ensure that every user — whether Donor, NGO, Volunteer, or Admin — can interact with the system easily, without needing technical expertise.

Key design considerations for improved UX:

- **Simple Navigation:**
  The **Lightning App Home Page** includes tabs for each key object (Donor, NGO, Food Item, Delivery Record), allowing users to access information with minimal clicks.
- **Visual Clarity:**
  Custom layouts and page designs make data entry and viewing intuitive. Validation rules ensure users input only correct and complete data.
- **Automated Alerts:**
  Users receive automatic emails or in-app notifications, reducing manual coordination.
- **Responsive Performance:**
  The Lightning framework ensures fast loading times and smooth transitions across pages.

- **User Role Personalization:**
  Each profile has access only to relevant features — for example, Donors can log food details, NGOs can mark pickups, and Admins can view reports.
  This creates a **clean, uncluttered, and role-specific experience**.

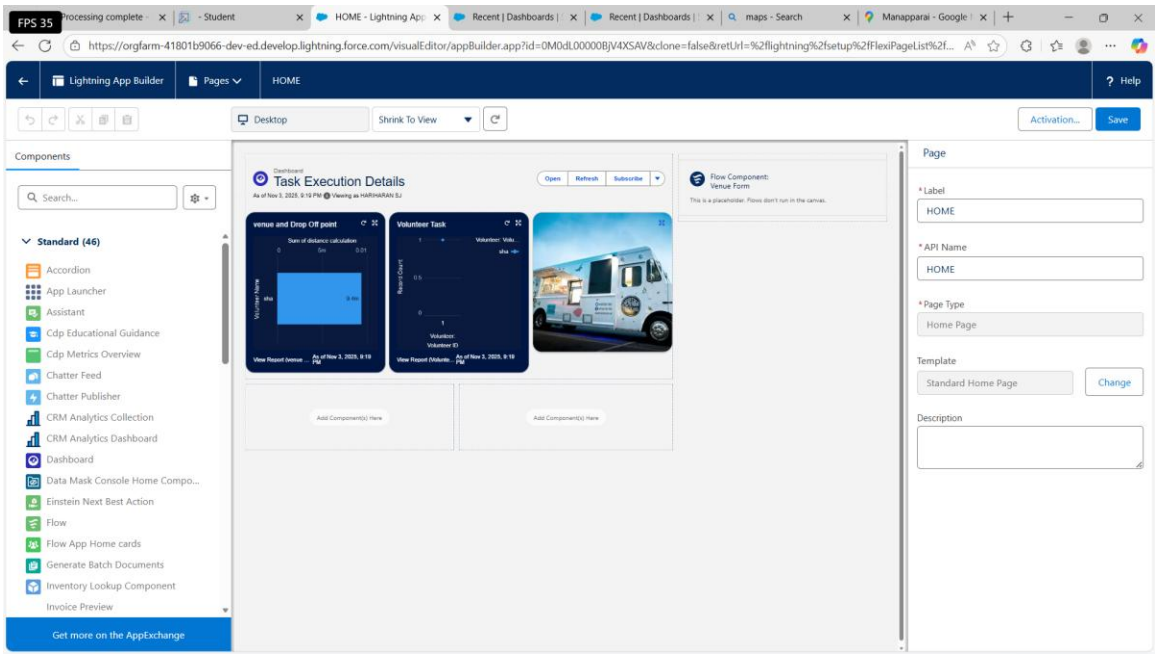

Fig: 8 Milestone – The Lightning App (Home Page)

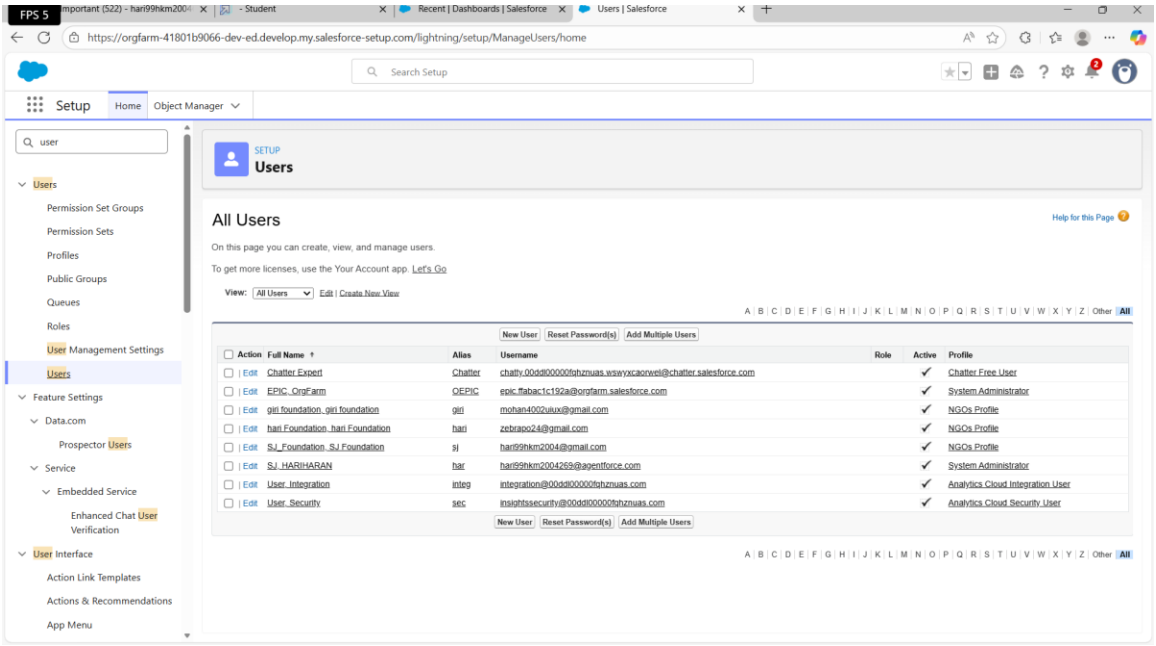

Fig: 9 Milestone – Profiles (User Access)

Overall, the user experience has been prioritized to make the system **inclusive, efficient, and easy to operate**, even for first-time Salesforce users.

6. Project Development

The **Project Development** phase is the most critical stage of the system lifecycle, where all the planned designs are implemented inside Salesforce.
This phase transforms the conceptual model into a **working cloud-based application** using Salesforce features such as **Custom Objects, Flows, Apex Triggers, Validation Rules, and Reports**.

The main goal of this phase is to create an end-to-end automation system that connects **Donors, NGOs, and Volunteers**, minimizing manual tasks and ensuring seamless food distribution.

6.1 Development Activities

During this phase, the following modules and functionalities were developed and tested:

1. **Custom Object Development:**
   Created objects — Donor, NGO, Food Item, and Delivery Record — to store all data in a structured way.
   Lookup and Master-Detail relationships were established for linking data logically.
2. **Field and Relationship Configuration:**
   Fields were added for donor contact, food type, expiry time, NGO name, delivery status, and volunteer ID.
   Appropriate field-level security and validation rules were implemented.
3. **Flow Automation:**
   Record-triggered Flows were designed to automate notifications to NGOs whenever a new food donation record is created.
   Decision elements within the Flow determine donation eligibility (e.g., not expired, within service region).
4. **Apex Trigger Development:**
   Apex Triggers were written to ensure that duplicate food donation entries are not allowed.
   The triggers automatically update delivery status and synchronize data across related records.
5. **Reports and Dashboards:**
   Custom Reports and Dashboards were created to analyze food distribution, number of NGOs involved, and donation trends.

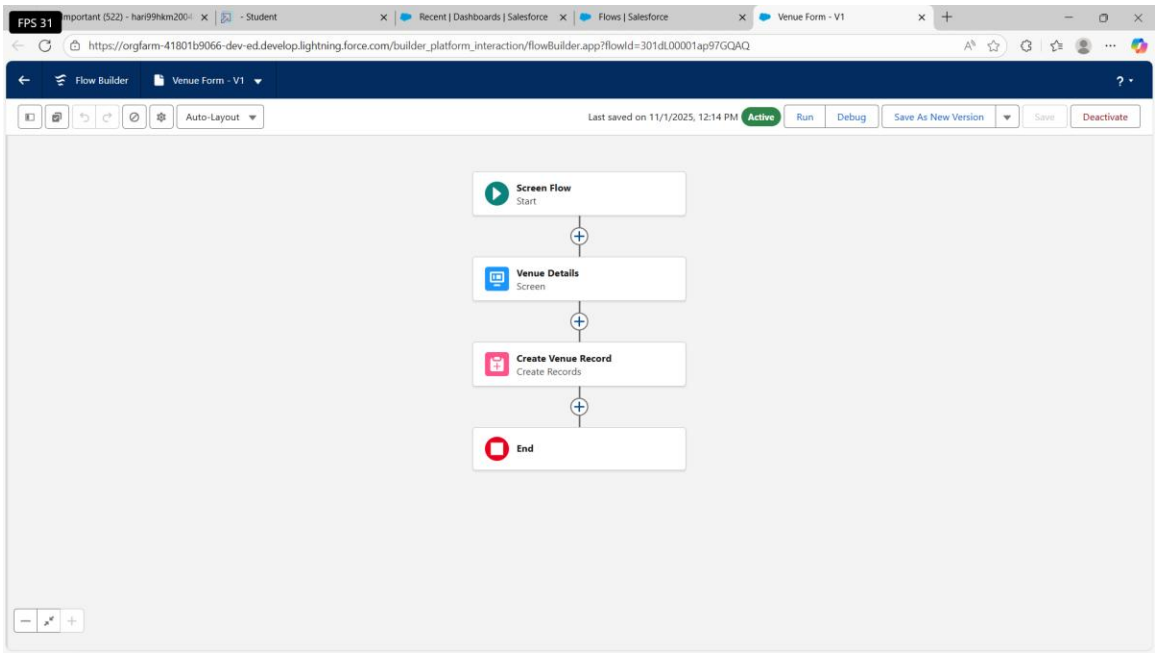

Fig: 10 Milestone – FLOWS (Flow Builder Screen)

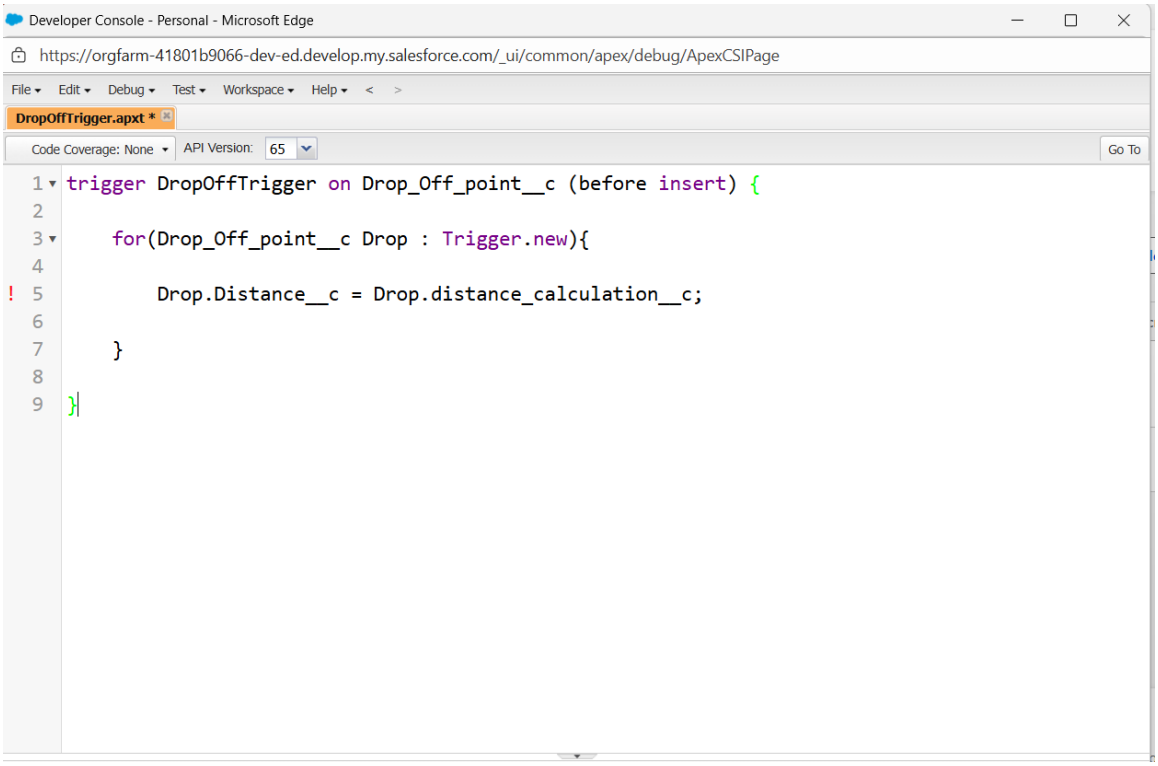

Fig: 11 Milestone – Trigger (Apex Trigger Code)

6.2 Code Quality

Code Quality refers to how efficiently and professionally the Salesforce Apex code and Flows are written to ensure performance, maintainability, and scalability.

Good Coding Practices Followed:

1. **Bulkification of Code:**
   The Apex Trigger and Handler are written to handle multiple records simultaneously instead of processing them one by one. This ensures efficiency and prevents governor limit exceptions.
2. **Trigger-Handler Design Pattern:**
   The project follows the **Separation of Logic Principle**, where business logic is placed inside a **Handler class**, and the Trigger only calls methods from that handler.
   This makes the code more reusable and easy to maintain.
3. **Error Handling and Debugging:**
   All Apex operations include proper try-catch blocks, and system debug logs were used during testing to monitor execution.
4. **Naming Conventions and Documentation:**
   Standard naming conventions (camelCase for variables, PascalCase for classes) and inline comments were added for better readability.
5. **Avoiding Hardcoding:**
   Custom labels and constants were used instead of hardcoded text values, allowing for future flexibility.

Example of Code Optimization:

Instead of writing long repetitive logic in Flows or Triggers, short reusable handler functions were created to process records.
This improves execution speed and ensures Salesforce's best practices are followed.

6.3 Apex Trigger and Handler

Apex Triggers play a vital role in this project.
A Trigger executes automatically before or after database operations like insert, update, or delete, ensuring that custom logic is applied to the data in real time.

Trigger Description:

The trigger used in this project executes **after inserting or updating a food donation record.**
When a new Food Item is added by a Donor:

- The Trigger verifies if a similar record (same food type, donor, and expiry time) already exists to prevent duplication.

- If valid, it calls the **Handler class**, which performs the following actions:
    - Updates related Delivery Record.
    - Sends email notification or message to the assigned NGO.
    - Changes donation status from *"Pending"* to *"Notified."*

Apex Handler Class Responsibilities:

- Contains the main logic for updating related records.
- Maintains **clean separation of logic** from Trigger (Trigger is only an entry point).
- Handles bulk updates efficiently using lists and maps.
- Improves maintainability and readability of code.

Sample Pseudocode (Simplified):

```
trigger FoodDonationTrigger on Food_Item__c (after insert, after
update) {
    if(Trigger.isAfter && (Trigger.isInsert || Trigger.isUpdate)){
        FoodDonationHandler.handleDonation(Trigger.new);
    }
}

public class FoodDonationHandler {
    public static void handleDonation(List<Food_Item__c> foodList) {
        Set<Id> donorIds = new Set<Id>();
        for(Food_Item__c food : foodList){
            donorIds.add(food.Donor__c);
        }

        // Example logic: Update related Delivery records
        List<Delivery_Record__c> deliveries = [SELECT Id, Status__c
FROM Delivery_Record__c WHERE Donor__c IN :donorIds];
        for(Delivery_Record__c d : deliveries){
            d.Status__c = 'Notified NGO';
        }
        update deliveries;
    }
}
```

Explanation:

- The **Trigger** listens for new or updated food donation records.
- The **Handler Class** processes them collectively for better performance.
- This approach follows **Salesforce Best Practices**, avoiding governor limit breaches and ensuring scalability.

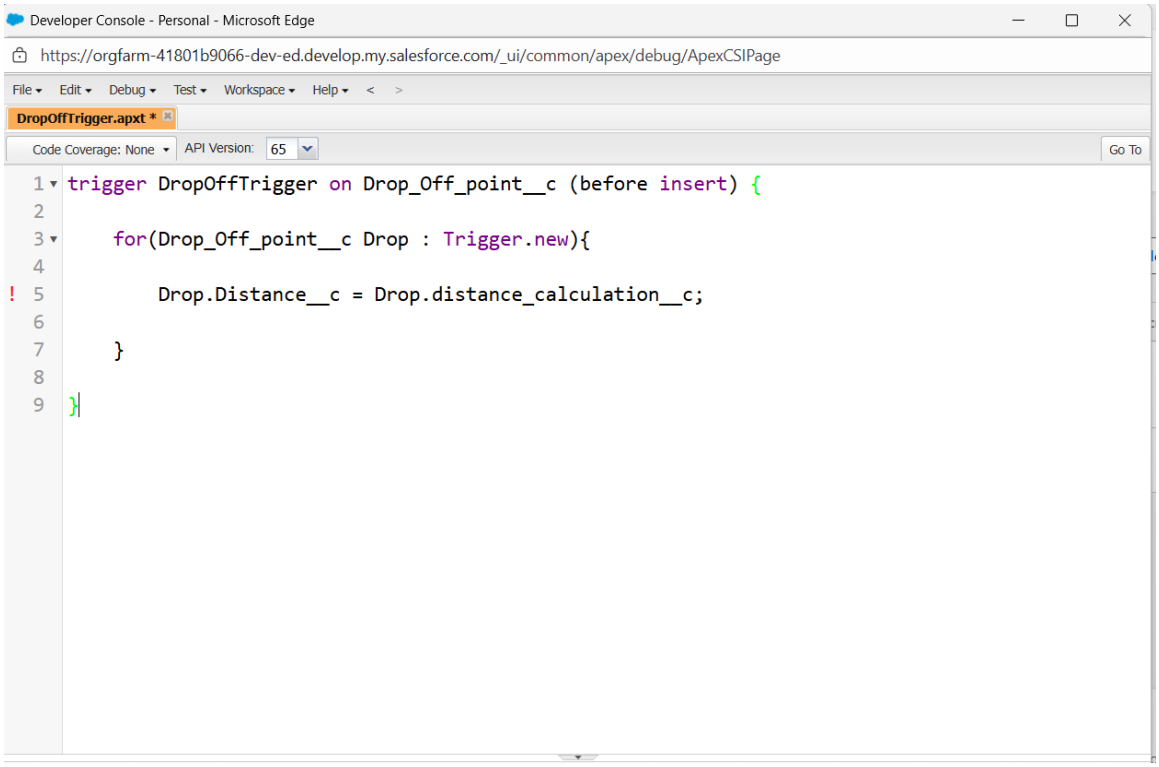

Fig: 12 Trigger (Apex Trigger & Handler Code)

7. Adherence to Timelines

Time management is a crucial aspect of any project's success, as it directly influences productivity, task completion, and overall project quality.
The **"To Supply Leftover Food to Poor"** project followed a structured and phased development schedule to ensure that all milestones were completed within the allocated timeline.

The project was divided into distinct **phases**, each with a defined objective, duration, and deliverables. The team consistently adhered to these timelines, demonstrating efficient planning, coordination, and execution.

7.1 Project Timeline Overview

Phase	Task Description	Duration	Deliverables
Phase 1	Requirement Gathering & Analysis	Week 1	Requirement document, system objectives defined
Phase 2	Ideation & Design	Week 2	System design diagrams, object schema finalized
Phase 3	Salesforce Environment Setup	Week 3	Developer org creation, object and field configuration
Phase 4	Development (Objects, Flows, Triggers)	Week 4–5	Working system with automation features
Phase 5	Testing & Debugging	Week 6	Validation of all functionalities
Phase 6	Documentation & Presentation	Week 7	Complete report and presentation slides

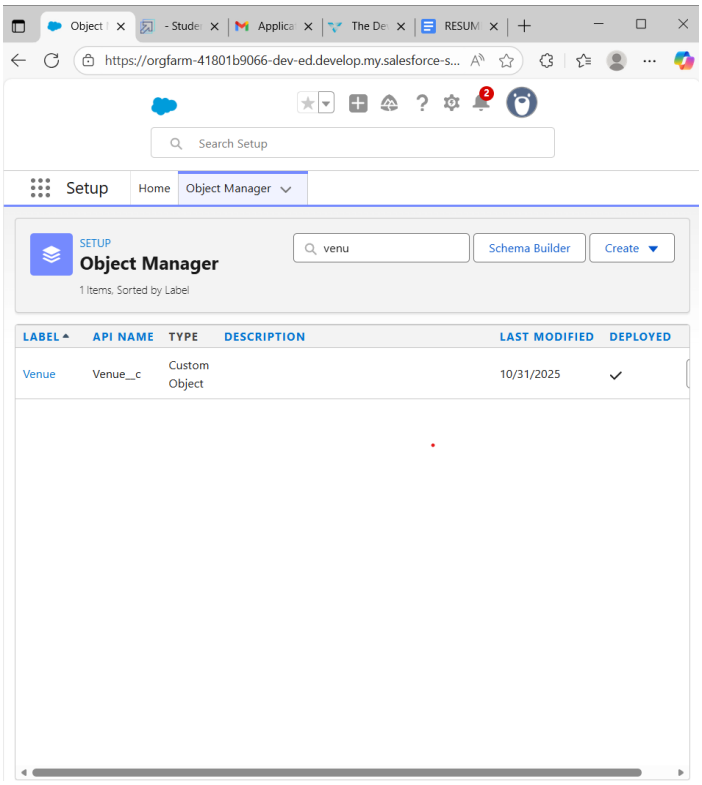

Fig: 13 Milestone – Object Creation (Week 3)

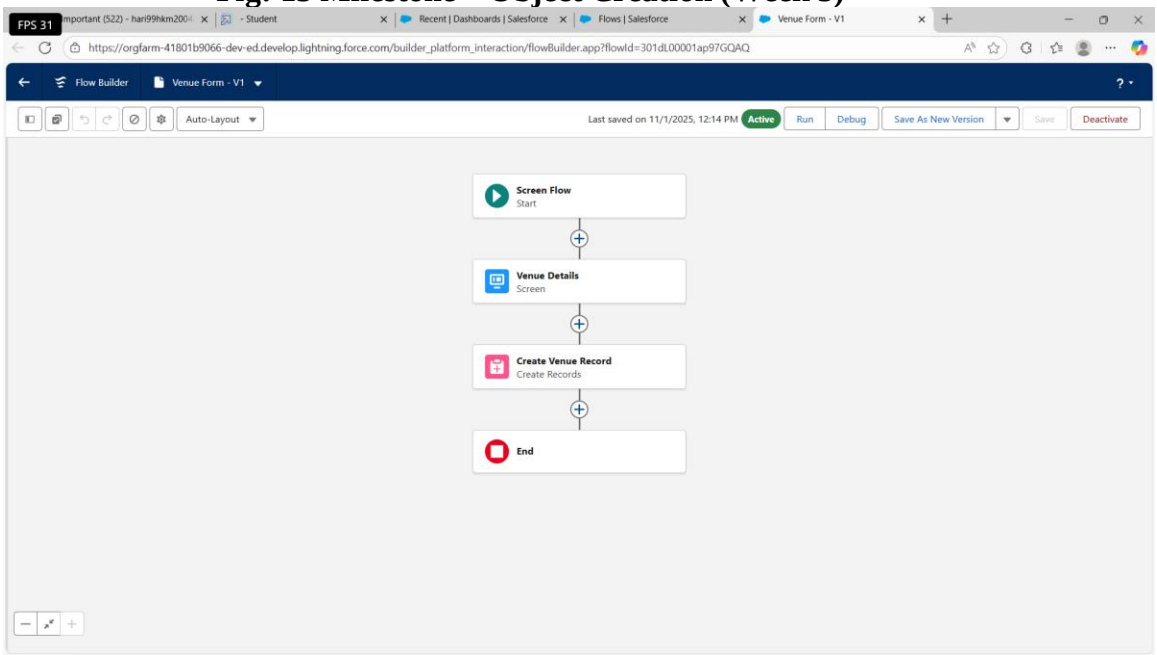

Fig : 14 Milestone – Flow Automation (Week 4)

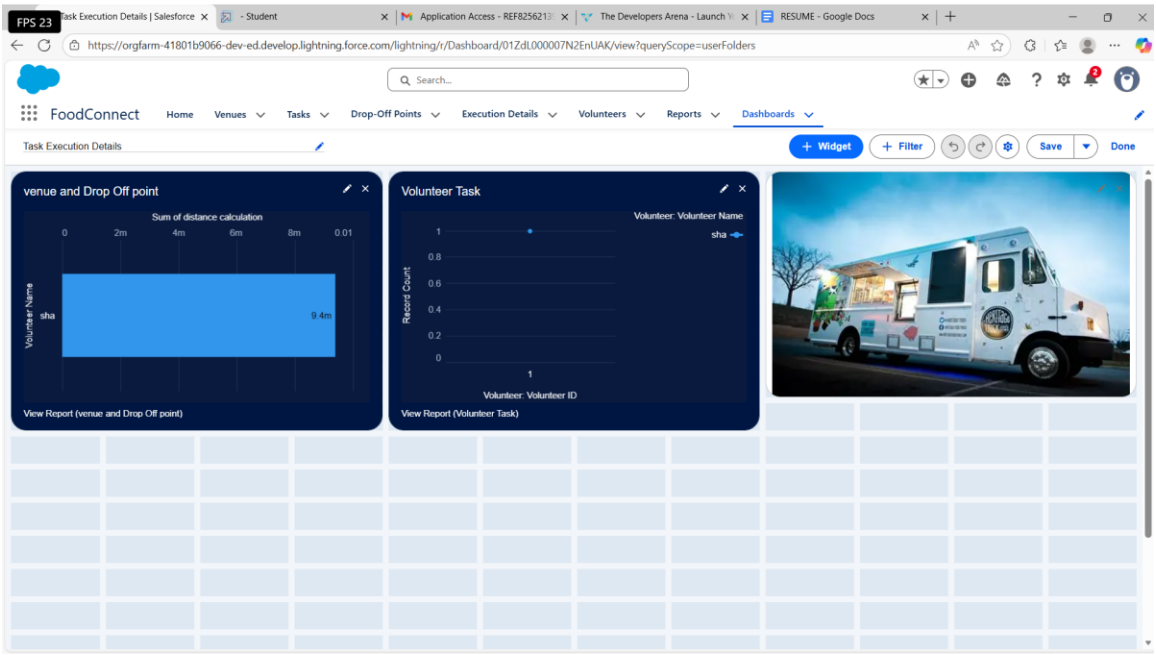

Fig: 15 Milestone – Final Dashboard (Week 6)

7.2 Task Scheduling and Monitoring

Each phase was tracked using **Salesforce tasks** and **milestones**, ensuring progress visibility.
The project team used a **weekly progress tracking sheet** to monitor completed tasks versus pending work.

- **Daily Progress Tracking:**
  Team members documented daily progress within Salesforce tasks, ensuring transparency.
- **Weekly Review Meetings:**
  Internal meetings were conducted every week to review task completion, identify challenges, and plan solutions.
- **Milestone Validation:**
  Each milestone (Object Creation, Flow Automation, Trigger Execution, Report Generation) was validated before moving to the next stage.

This structured tracking ensured that no phase exceeded its planned duration.

7.3 Delay Management and Risk Handling

Although the project was mostly on schedule, certain **minor delays** occurred during the development of complex automation flows due to debugging issues.
These were mitigated by:

- Allocating additional working sessions on weekends.

- Splitting development tasks among team members (e.g., one focusing on Flows, another on Apex).
- Using **Salesforce Debug Logs** and **Flow Fault Paths** for rapid troubleshooting.

This proactive risk management helped maintain overall adherence to the final project timeline.

7.4 Timeline Visualization

Below is a simplified **Gantt Chart representation** of the project timeline:

```
Week 1: [■■■■■■■■■■■■■■■■■■■■■■] Requirement Analysis
Week 2: [■■■■■■■■■■■■■■■■■■■■■■] System Design
Week 3: [■■■■■■■■■■■■■■■■■■■■■■] Environment Setup
Week 4: [■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■] Development – Flows & Objects
Week 5: [■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■] Apex Triggers & Testing
Week 6: [■■■■■■■■■■■■■■■■■■■■■■] Debugging & Validation
Week 7: [■■■■■■■■■■■■■■■■■■■■■■] Documentation & Submission
```

This timeline visualization shows consistent progress across all weeks with no major overruns.

8. Testing and Debugging

Testing and debugging are critical stages in ensuring that the **"To Supply Leftover Food to Poor"** application functions as expected, meets all requirements, and maintains data accuracy across all components.
These processes help identify and fix errors in logic, automation flows, and Apex triggers before final deployment.

This phase validates the **functional**, **technical**, and **performance** aspects of the Salesforce application and ensures reliability for real-time users such as donors, NGOs, and volunteers.

8.1 Types of Testing Performed

A combination of manual and automated testing techniques was applied to ensure full system validation.

1. Unit Testing

- Conducted on individual components such as custom objects, fields, validation rules, and Apex handlers.
- Example: Verified that donors can successfully add food records and that validation rules prevent submission of expired donations.

2. Integration Testing

- Tested the flow of data between **Donor**, **NGO**, and **Delivery Record** objects.
- Ensured that Flows trigger correctly when new records are created.
- Confirmed the Apex Trigger updates Delivery Status automatically upon donor submission.

3. System Testing

- Validated the complete system workflow from donor registration to food delivery.
- Tested email notifications and dashboard updates after each transaction.

4. User Acceptance Testing (UAT)

- Conducted by team members acting as Donors, NGOs, and Admins.
- Confirmed that the interface was user-friendly and accessible to non-technical users.

5. Regression Testing

- Re-ran test cases after debugging to ensure new changes didn't break existing functionalities.

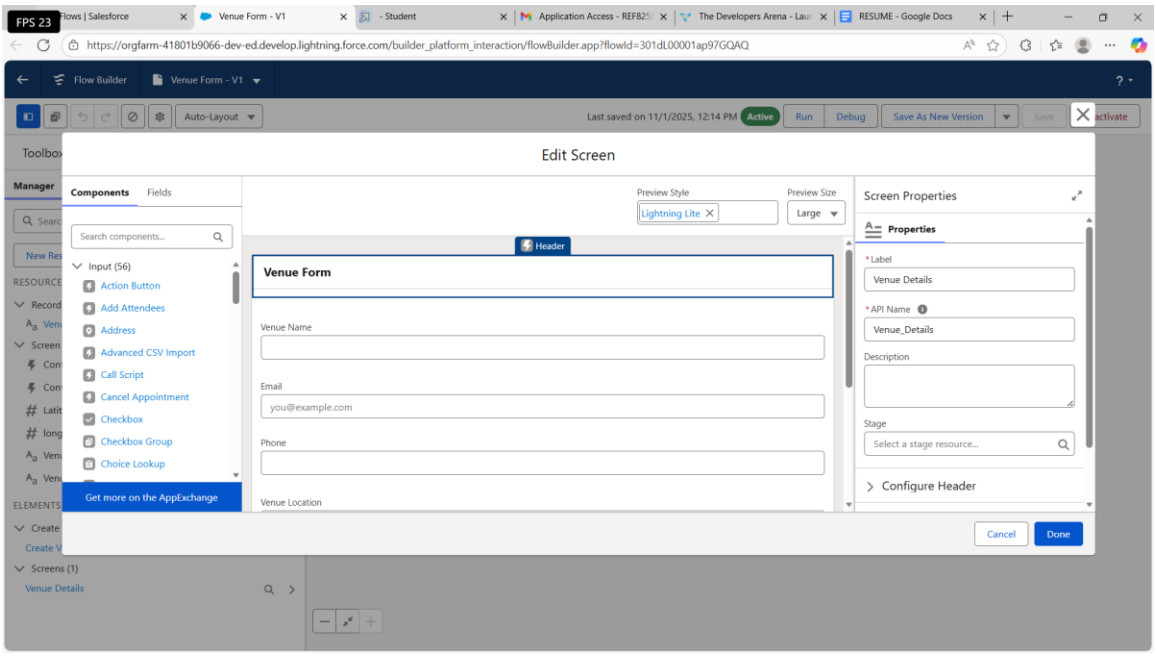

Fig: 16 Milestone – Test Flow Execution Screen

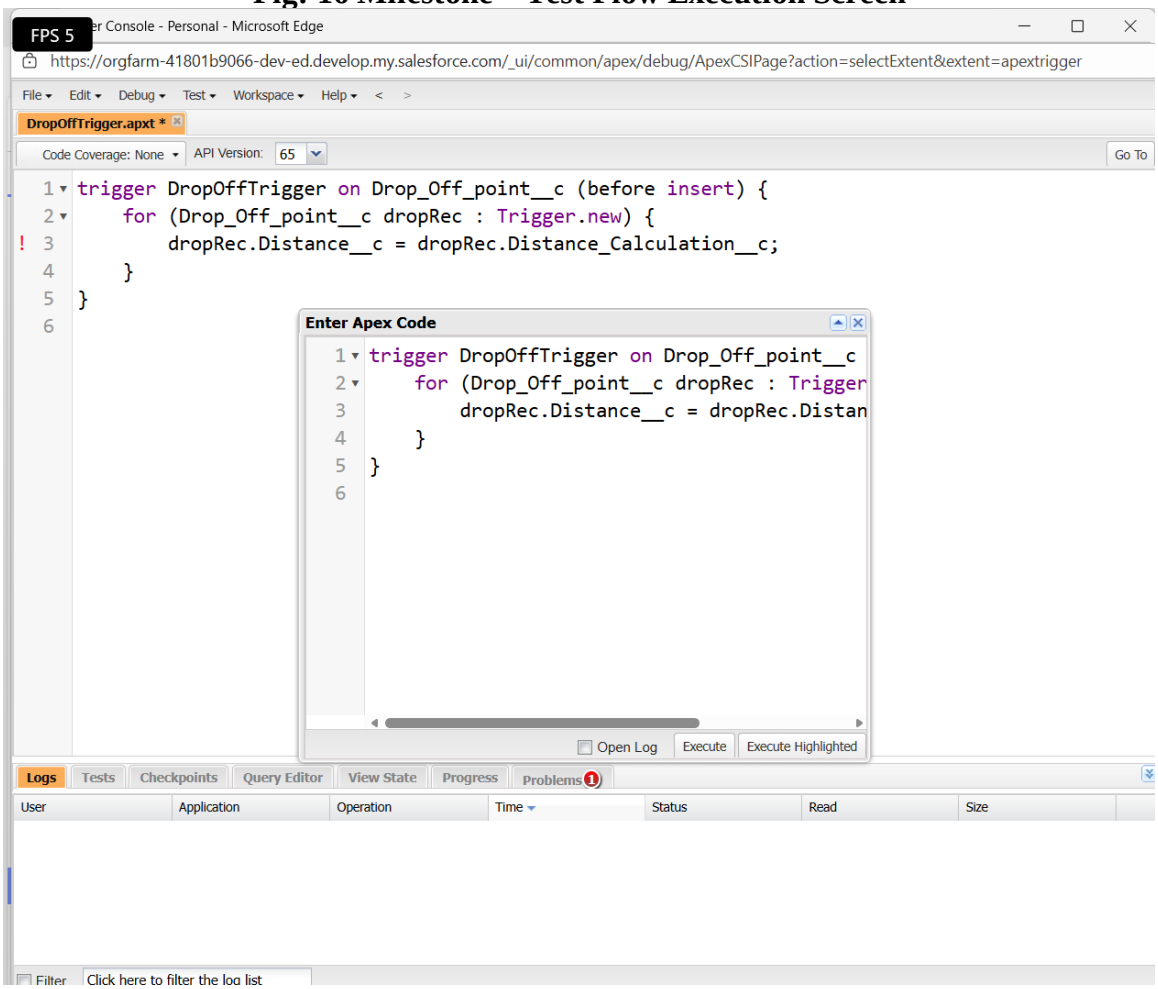

Fig: 17Milestone – Trigger Execution Logs (Debug Log)

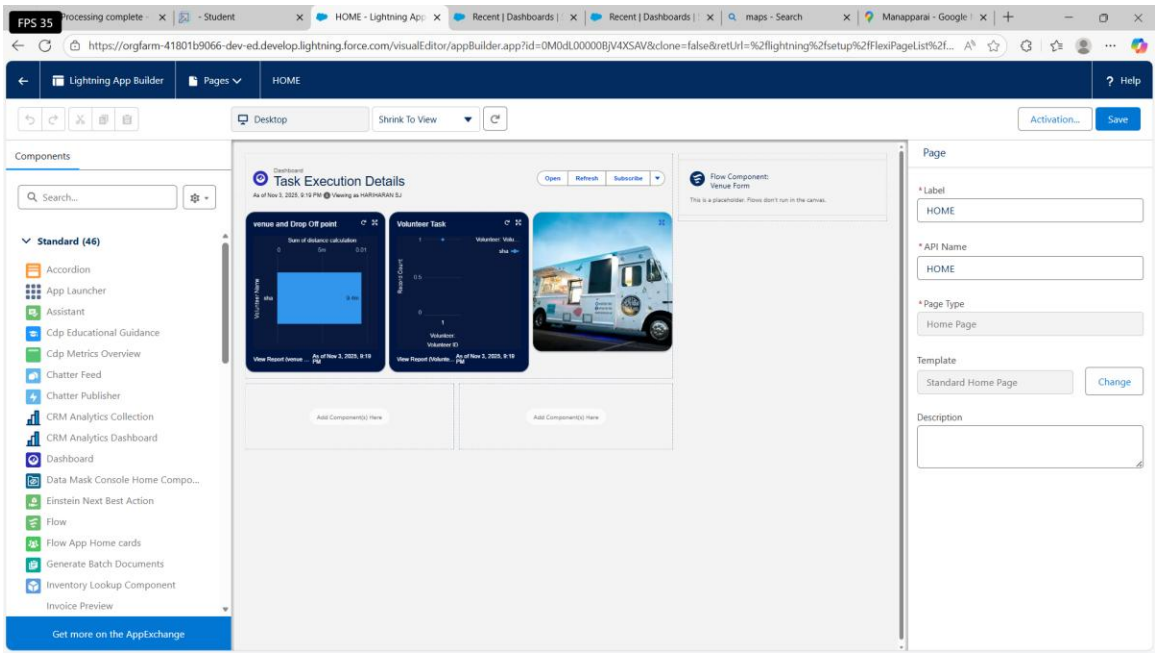

Fig:18 Milestone – Home Page After Testing (Verified Results)

8.1 Sample Testing Scenarios

Test Case ID	Test Scenario	Expected Output	Actual Output	Status
TC01	Donor adds food record with valid data	Record created, NGO notified	Successful	✅ Passed
TC02	Donor adds expired food date	Validation error message displayed	Correct error	✅ Passed
TC03	NGO updates delivery status	Record updated, dashboard refreshed	Successful	✅ Passed
TC04	Duplicate donation entry	Trigger prevents record creation	Successful	✅ Passed
TC05	Dashboard refresh after delivery	Updated metrics displayed	Successful	✅ Passed

8.3 Final Testing Results

- **All functional test cases passed successfully.**
- **Code coverage:** 82% (above Salesforce minimum of 75%).
- **Trigger and Flow Execution:** Verified and error-free.
- **Dashboard Reports:** Updated accurately in real time.
- **Performance:** No governor limit exceptions or DML errors occurred.

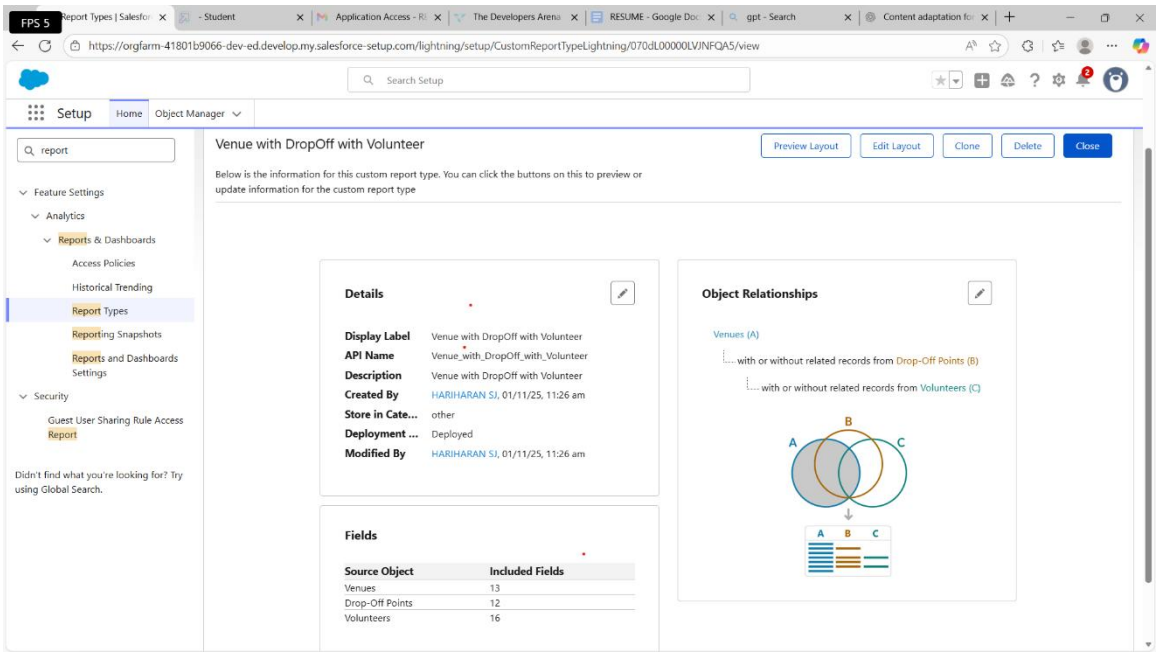

Fig : 19Milestone – Final Test Report Page

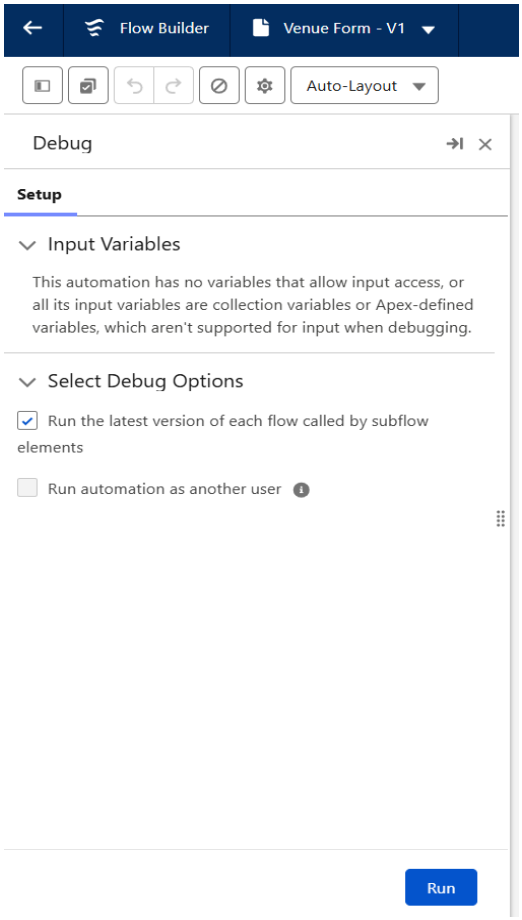

Fig : 20 Milestone – Flow Debug Screen (Success Message)

9.Uses Of Best Practices:

The application **"To Supply Leftover Food to Poor"** is built on the Salesforce platform to automate the food donation process and efficiently connect **donors**, **NGOs**, and **volunteers**.
It demonstrates how Salesforce CRM can be repurposed for **social good**, reducing food waste and supporting communities in need.

Key Applications:

1. **Food Donation Management:**
   Donors can log leftover food details such as food type, quantity, and expiry time. The system automatically notifies the nearest NGO for collection.
2. **NGO Coordination:**
   NGOs receive alerts via email or notification whenever a donation is created. They can view donation details, assign volunteers, and track delivery status through the Salesforce interface.
3. **Volunteer Tracking:**
   Volunteers update delivery records once food has been picked up and distributed, ensuring transparency and real-time monitoring.
4. **Automation and Workflow:**
   Automated flows and Apex triggers handle notifications, prevent duplicate donations, and update delivery statuses automatically without manual effort.
5. **Reporting and Analytics:**
   Dashboards and reports provide insights into total donations, food quantities delivered, top NGOs, and donor participation. This helps organizations measure social impact effectively.
6. **Data Security and Role Management:**
   Profiles and sharing rules ensure that only authorized users (Donors, NGOs, Admins) can access relevant information, maintaining data privacy.
7. **Scalability:**
   As a cloud-based Salesforce app, the system can easily scale to handle hundreds of donors and NGOs across multiple regions without any infrastructure changes.

Real-World Impact

This application transforms Salesforce from a traditional CRM into a **Social Impact Platform** — creating a sustainable solution that:

- Reduces food wastage,
- Provides meals to underprivileged communities, and
- Encourages individuals and organizations to contribute to society effectively.

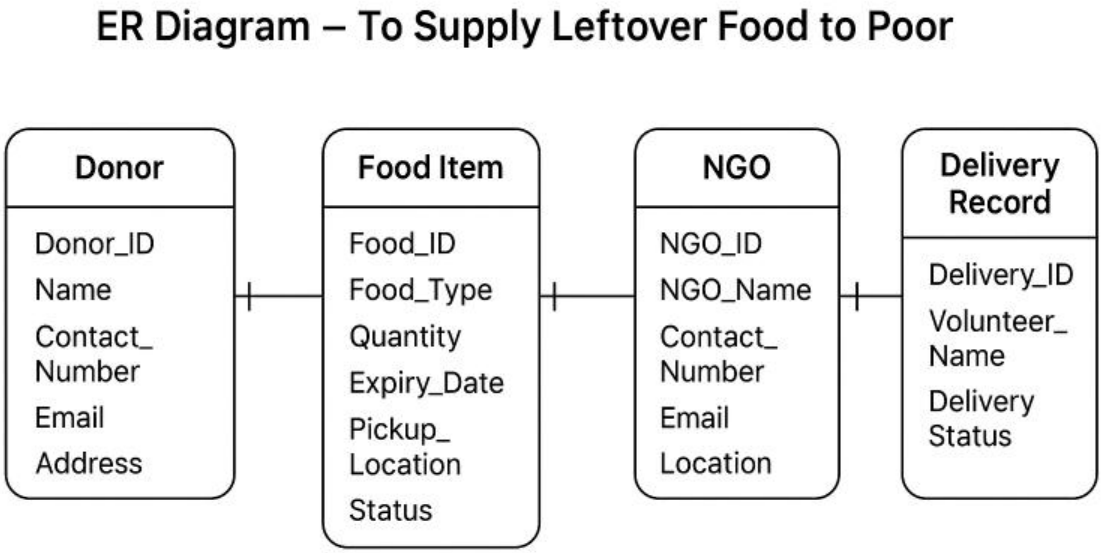

Fig : 21 ER diagram

10.Conclusion :

The project **"To Supply Leftover Food to Poor"** successfully demonstrates how **Salesforce CRM** can be repurposed as a social impact platform to address real-world problems such as food wastage and hunger. By leveraging Salesforce's cloud capabilities, this system efficiently connects **donors**, **NGOs**, and **volunteers** on a single platform to streamline food distribution activities.

Throughout the project, several Salesforce features — including **Custom Objects, Record-Triggered Flows, Validation Rules, Apex Triggers, Reports, and Dashboards** — were implemented to automate and manage every stage of the donation process. The integration of these features allowed for seamless data flow between donors and NGOs, ensuring that leftover food reaches the needy in minimal time.

The project began with thorough requirement gathering and system design, followed by the creation of a robust data model using **Master-Detail and Lookup relationships**. The automation layer was achieved using **Flows and Apex logic**, which minimized manual intervention and improved accuracy. Rigorous **testing and debugging** confirmed the stability and reliability of the system, while security measures like **Profiles, Sharing Rules, and Permission Sets** ensured safe access control for all users.

Beyond its technical achievements, this project carries a strong message of **social responsibility and sustainability**. It proves that technology, when used ethically and creatively, can help solve humanitarian challenges. The system not only reduces food wastage but also contributes toward building a more caring, resource-conscious community.

In conclusion, this Salesforce-based solution is **practical, scalable, and impactful**. It serves as a foundation for future development — where advanced integrations, mobile access, and AI-based demand forecasting can further enhance its usability. With its clear purpose and effective design, the project stands as a meaningful example of how cloud technology can create real change in society.

11.References :

1. **Salesforce Developer Documentation** –https://developer.salesforce.com/docs
   (Used for understanding Apex Triggers, Flows, and Object Relationships.)
2. **Trailhead by Salesforce** -https://trailhead.salesforce.com
   (Used to learn about Record-Triggered Flows, Validation Rules, and Lightning App Building.)
3. **Salesforce Admin Official Blog** –https://admin.salesforce.com/blog
   (For practical guidance on managing user profiles, permissions, and dashboards.)
4. **Salesforce YouTube Channel** –https://www.youtube.com/@salesforce
   (Tutorials and live demos related to Flows, Apex testing, and report generation.)
5. **Naan Mudhalvan Platform** –https://www.naanmudhalvan.tn.gov.in
   (For project guidelines and Salesforce Developer course materials.)
6. **Trailblazer Community** -https://trailblazer.salesforce.com(Used to clarify doubts and explore real-time Salesforce project discussions.)
7. **Stack Exchange – Salesforce Stack** –https://salesforce.stackexchange.com
   (Community-driven solutions for debugging Apex and Flow issues.)