**Smart Blood Donation & Donor Engagement System**

**Phase 1: Problem Understanding & Industry Analysis**

**1. Requirement Gathering**

* **Core Need:** Blood banks and hospitals require a centralized system to manage donor records, blood inventory, and urgent requests.
* **Key Requirements:**
  + Maintain donor profiles (blood group, last donation date, eligibility).
  + Automate donation reminders and alerts for eligible donors.
  + Real-time matching of urgent hospital requests with available donors.
  + Track donation history and generate reports.
  + Ensure compliance with medical regulations and privacy standards.

**2. Stakeholder Analysis**

* **Blood Banks & Hospitals:** Primary users, need efficient donor management and quick access to blood in emergencies.
* **Donors:** End users who provide blood; require transparency, reminders, and recognition for their contributions.
* **Volunteers/NGOs:** Help coordinate donation drives and engage communities.
* **Government/Regulators:** Require reports to ensure compliance and transparency in blood donation.
* **System Administrators:** Maintain and manage CRM operations.

**3. Business Process Mapping**

**Current Process (Manual):**

* Hospitals manually contact donors during shortages.
* Donor records are often fragmented across spreadsheets or local databases.
* Delays in matching donors → shortage in critical situations.

**Proposed CRM-enabled Process:**

1. Donor registers in CRM (via hospital/NGO portal).
2. System validates donor eligibility (based on health and donation frequency).
3. CRM sends automated reminders when donor becomes eligible again.
4. Hospitals raise urgent blood requests → CRM matches with eligible nearby donors.
5. Dashboards provide real-time visibility of donor activity, requests, and donation drives.

**4. Industry-specific Use Case Analysis**

* **Healthcare Industry Need:** Reliable, transparent, and quick blood donation systems are critical to saving lives.
* **Challenges Today:** Lack of transparency, fragmented data, emergency delays.
* **Value of CRM in Healthcare Blood Donation:**
  + Centralizes donor data for quick access.
  + Improves trust among donors by ensuring proper tracking and communication.
  + Provides hospitals with efficiency and reduced response time.
  + Enhances community engagement through transparent reporting.

**5. AppExchange Exploration**

On Salesforce **AppExchange**, similar healthcare apps exist (for hospital management, patient tracking, and volunteer engagement). However, very few apps focus specifically on **blood donation management**.

* **Relevant Solutions on AppExchange:**
  + *Volunteers for Salesforce* (for volunteer scheduling).
  + *Health Cloud Apps* (for patient and healthcare data management).
* **Gap Identified:** No focused app that combines **blood donor engagement + hospital urgent request handling + automated reminders**.
* **Opportunity:** Build a unique solution that fills this gap and can later be extended or even published on AppExchange as a custom product.

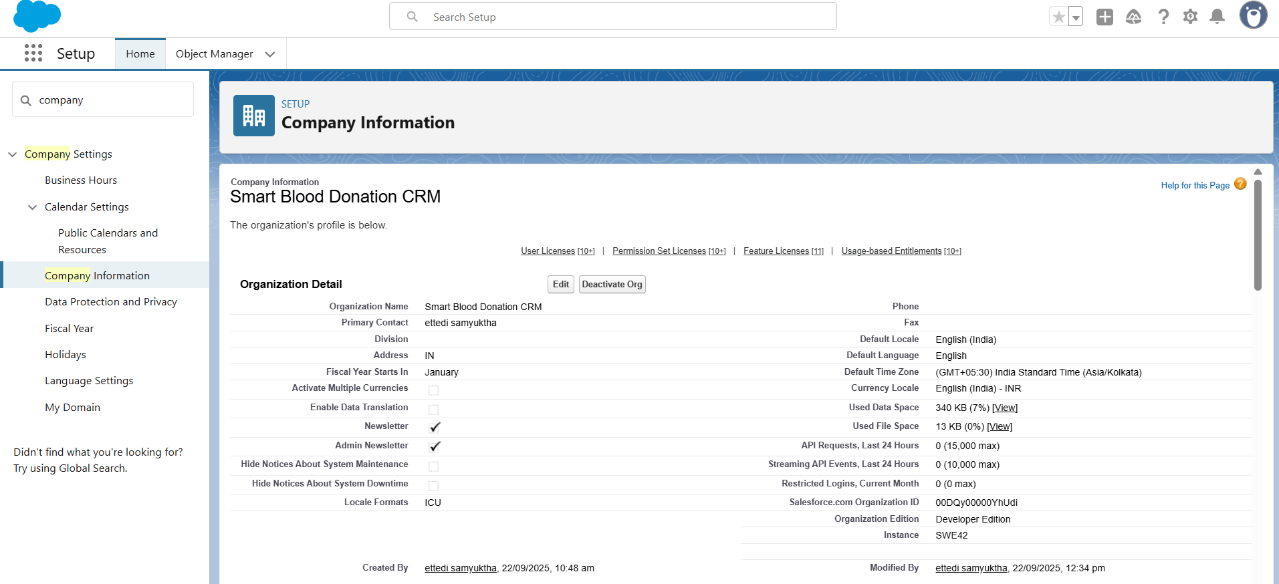
**PHASE 2 - Org Setup & Configuration**

Company Profile Setup

Path: Setup → Company Information → Edit

Name: Smart Blood Donation CRM  
Time Zone: GMT+05:30 Asia/Kolkata  
Locale: English (India)  
Language: English  
Currency: INR

Purpose: Establishes the foundational organizational settings for donor, hospital, and NGO data.

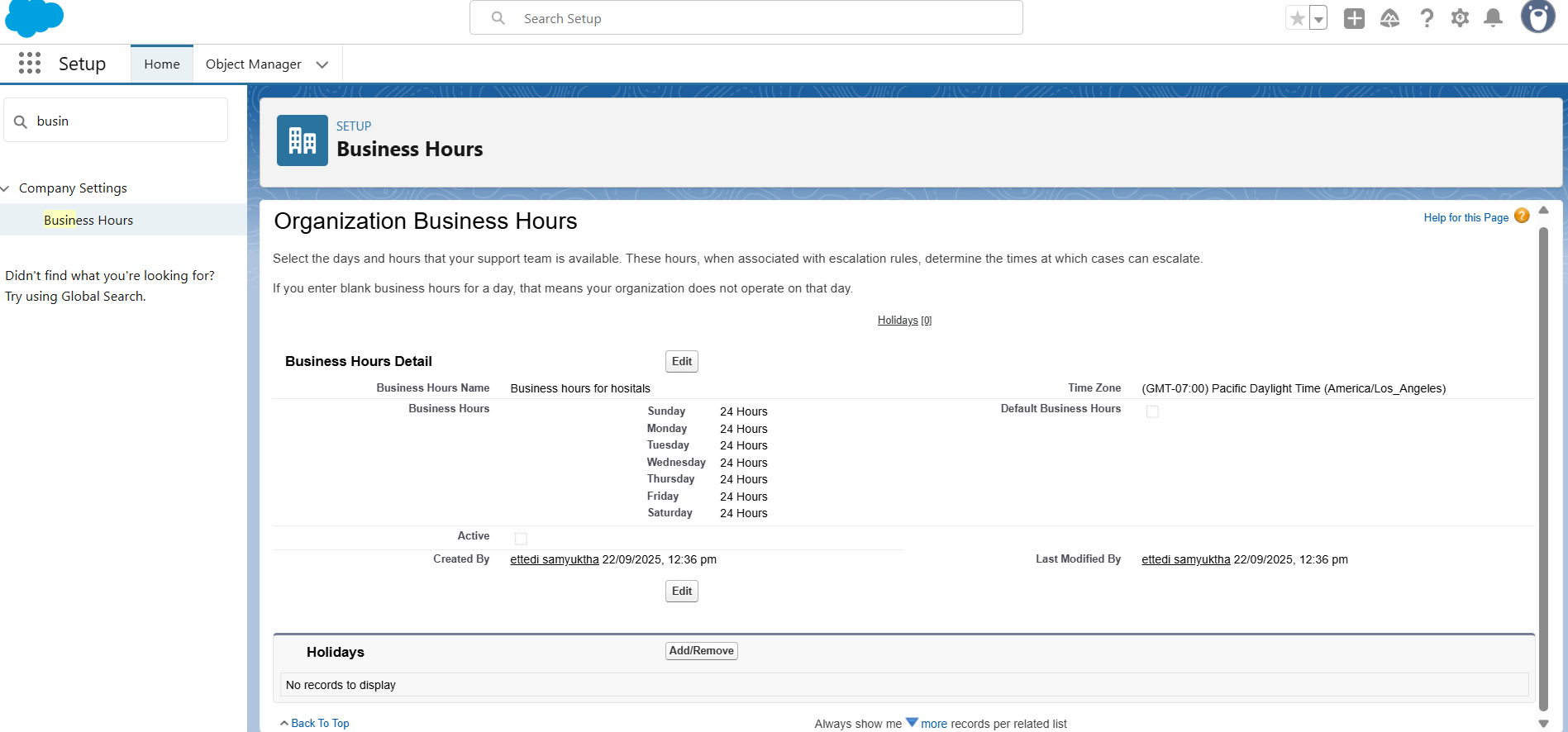


Business Hours Setup

Path: Setup → Business Hours → New

Name: 24x7 Hospital Emergency Hours  
Time Zone: GMT+05:30 Asia/Kolkata  
Working Hours: Mon-Sun 12:00 AM–11:59 PM

Purpose: Hospitals and blood banks operate round-the-clock; emergency requests need 24x7 coverage.

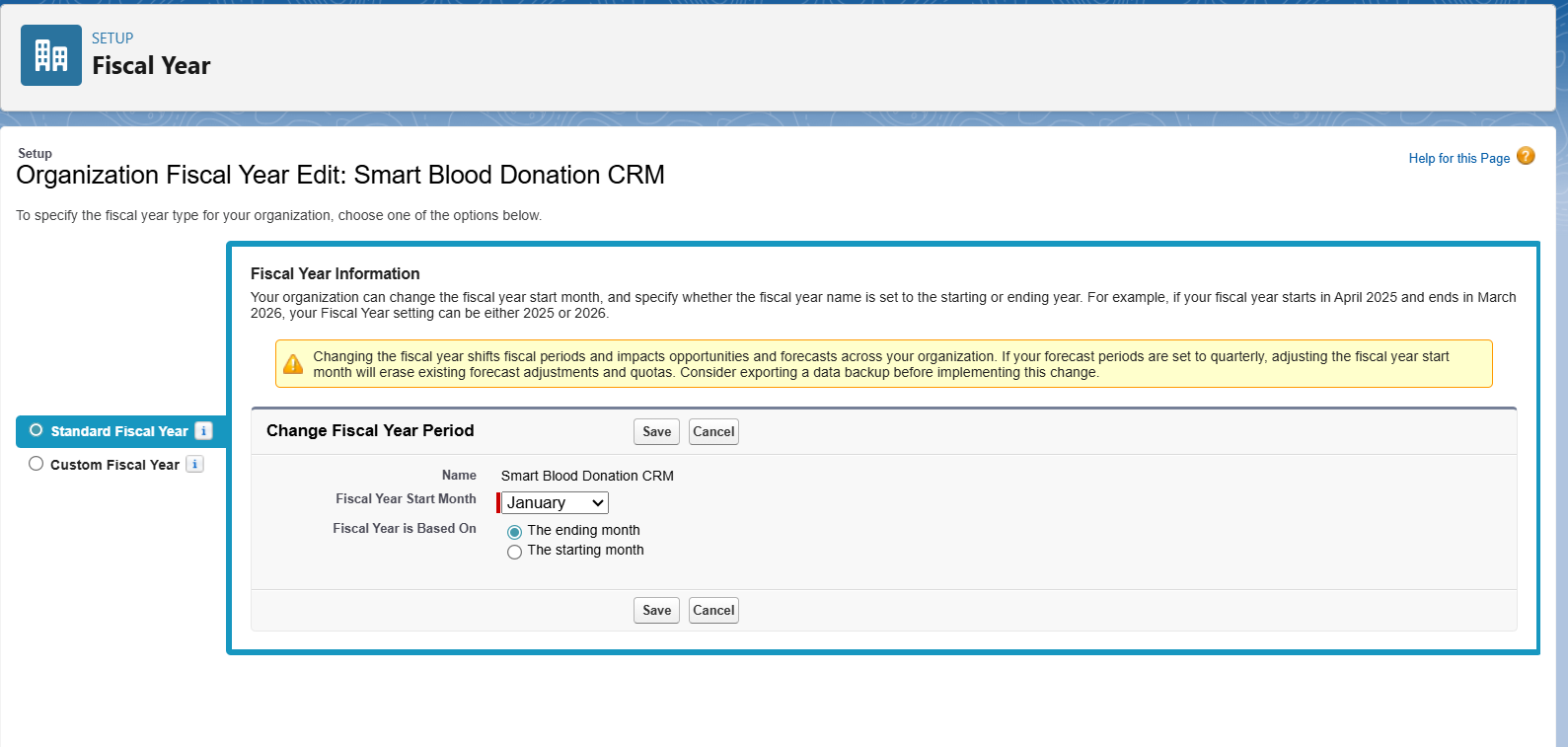


Fiscal Year Setup

Path: Setup → Fiscal Year

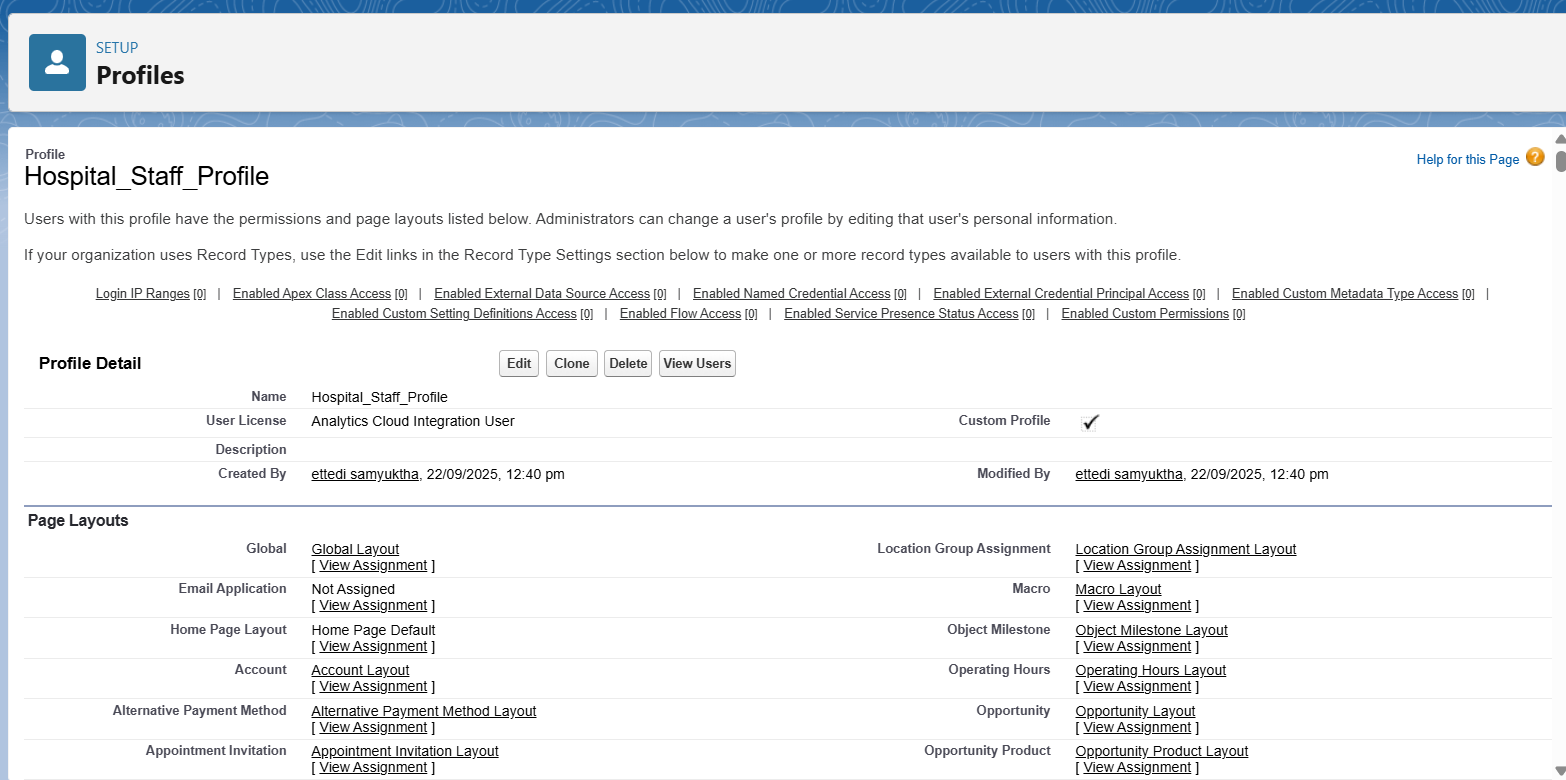
Type: Standard Fiscal Year  
Configuration: Starting month set to January

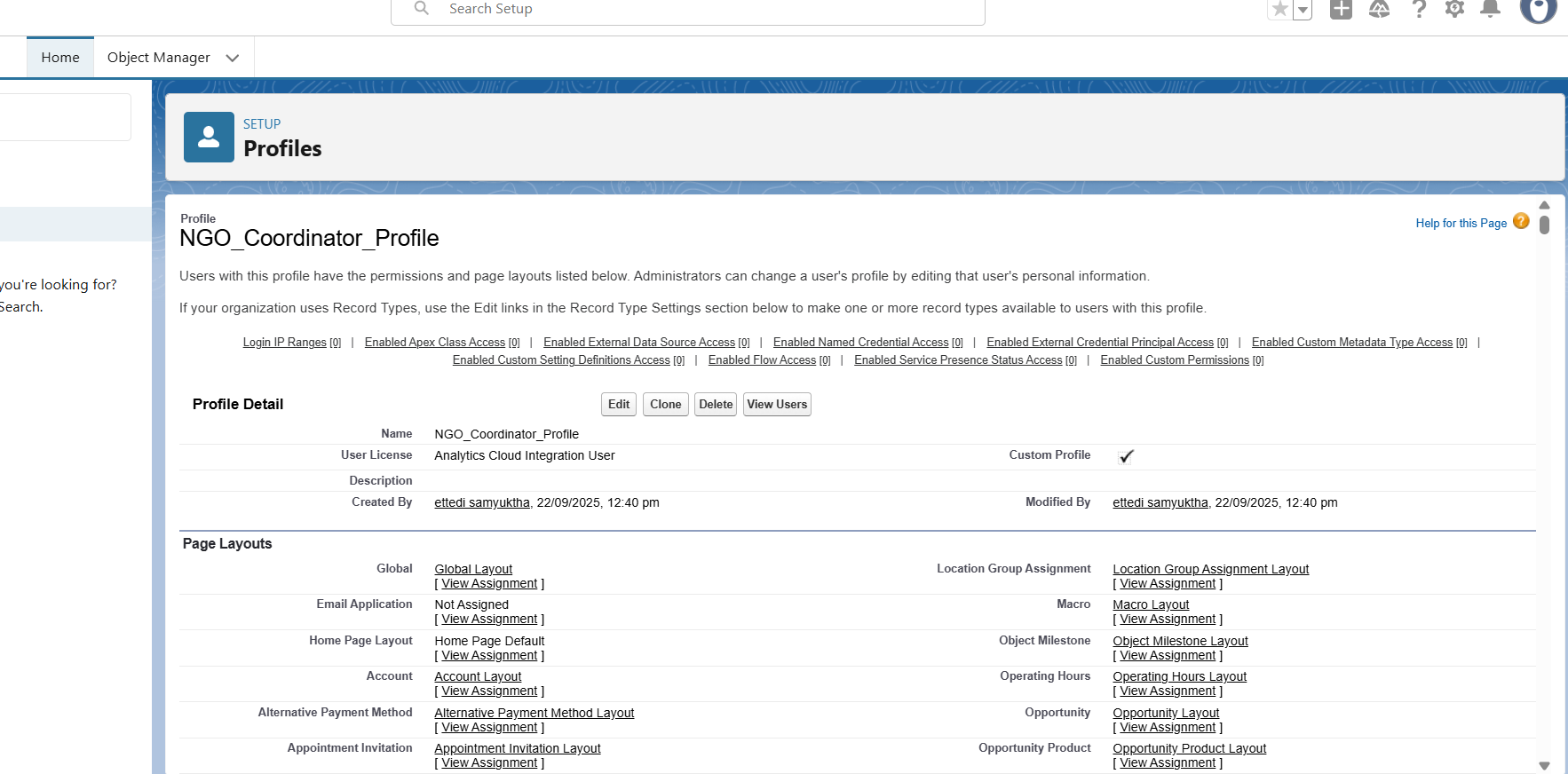
Purpose: Defines reporting periods for donor drives and hospital blood usage analysis.

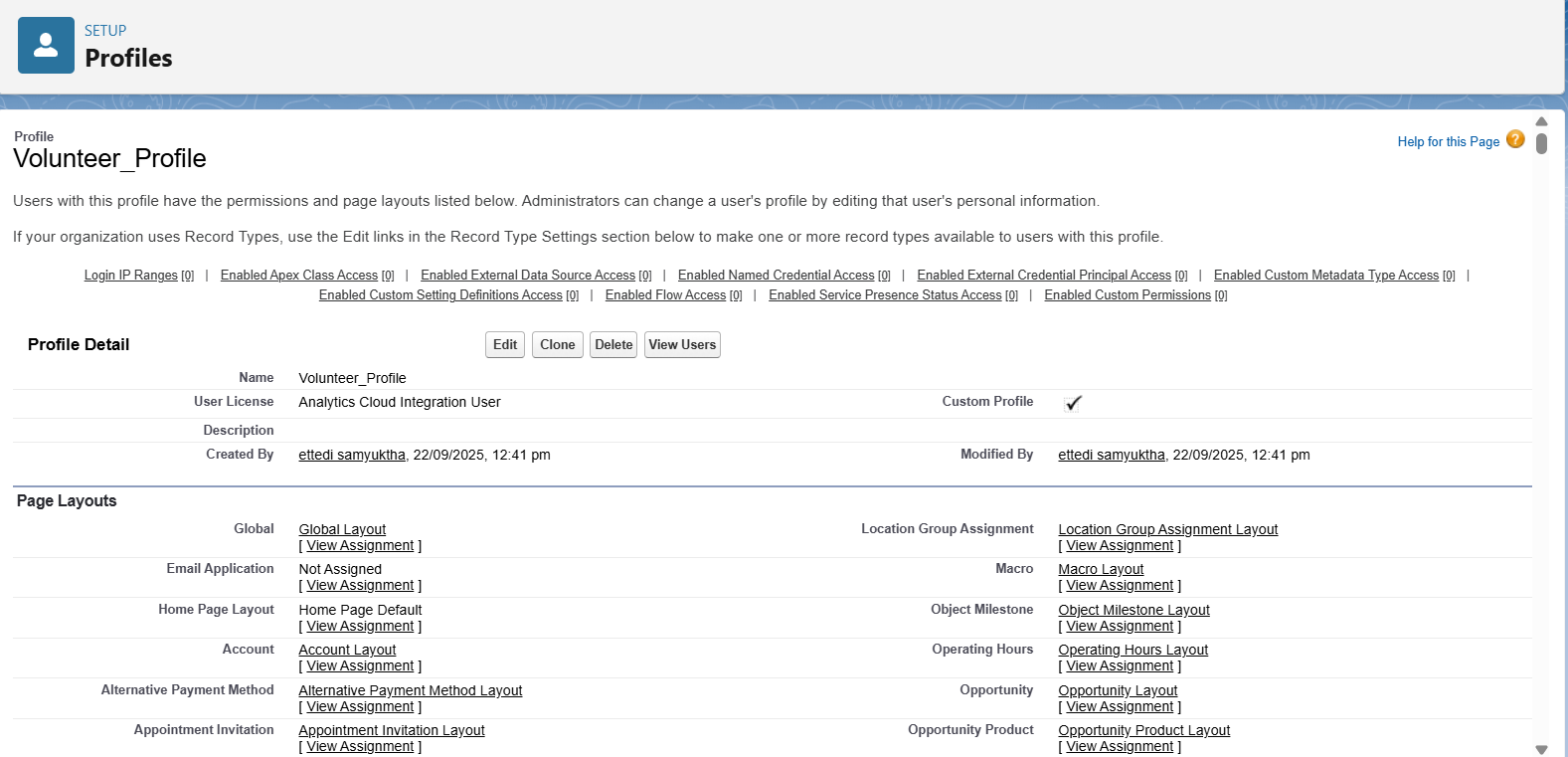


User Setup (Profiles, Roles, Permission Sets, Users)

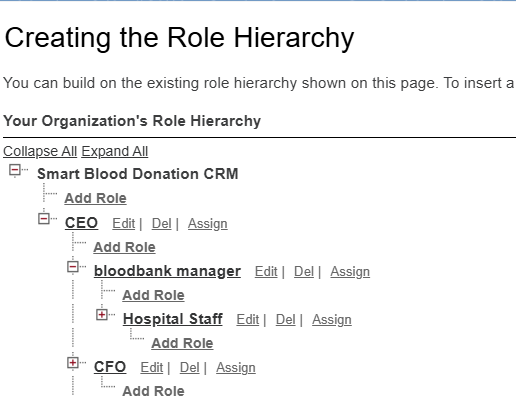
Profiles:  
- Hospital\_Staff\_Profile: For managing donor records and hospital requests.

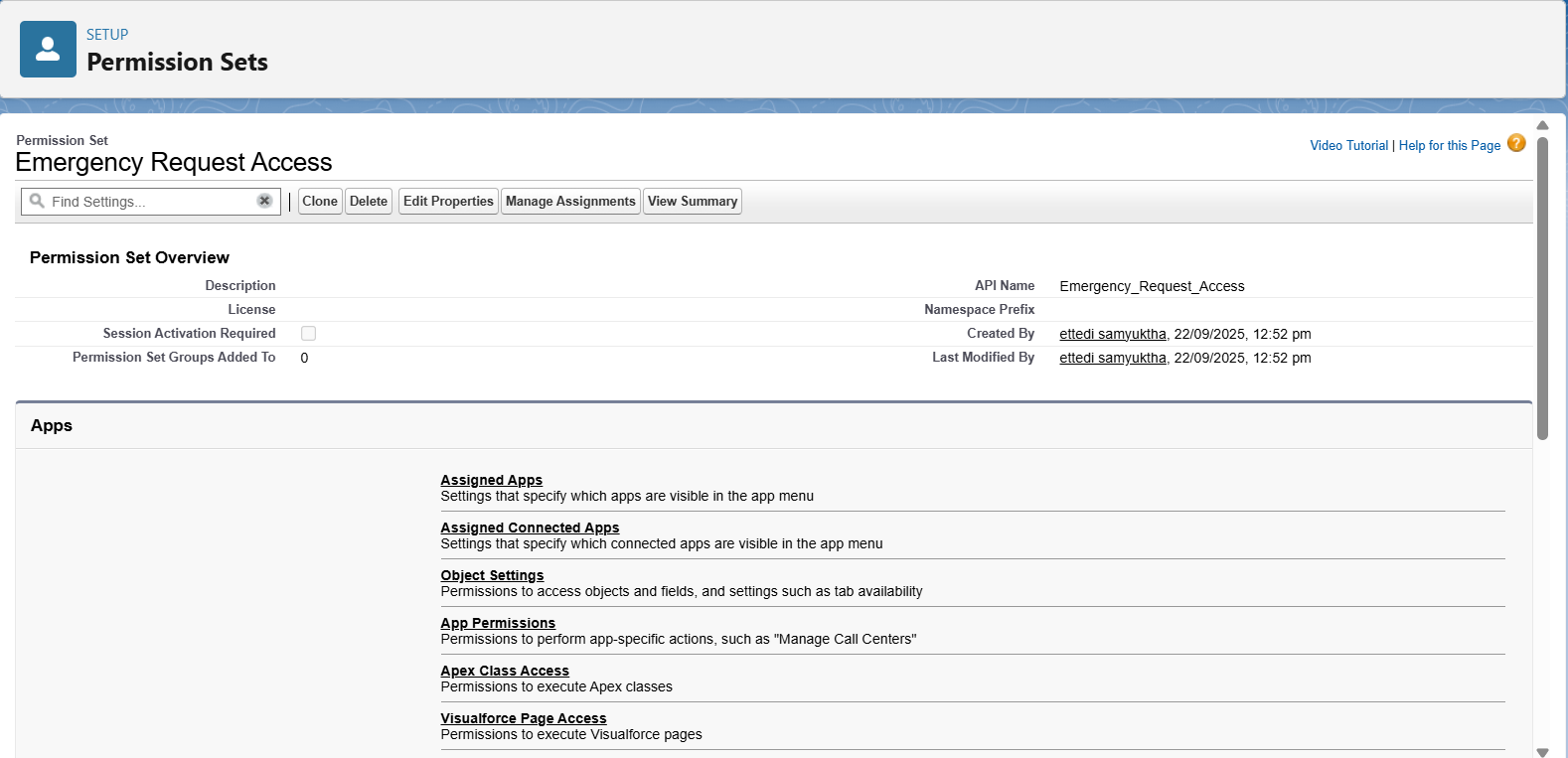
  
- NGO\_Coordinator\_Profile: For managing donation drives and volunteer data.

  
- Volunteer\_Profile: For limited access to participation in donation events.



Roles:  
- CEO → Blood Bank Manager → Hospital Staff → Volunteers



Permission Sets:  
- Emergency\_Request\_Access\_PS: Grants ability to handle urgent hospital requests.  


Users:  
- staff1 → Hospital\_Staff\_Profile, Role: Hospital Staff  
- ngo1 → NGO\_Coordinator\_Profile, Role: NGO Coordinator  
- volunteer1 → Volunteer\_Profile, Role: Volunteer

Role Hierarchy Setup

Path: Setup → Roles → Set Up Roles

Top-Level Role: CEO  
Child Role: Blood Bank Manager  
Child Role: Hospital Staff  
Child Role: Volunteer

Resulting Hierarchy: CEO → Blood Bank Manager → Hospital Staff → Volunteer

OWD & Sharing Rules

* OWD (Org-Wide Default): This will be configured in Phase 3, as the settings need to be applied to the custom objects that will be created then.
* Sharing Rule: This will also be implemented in Phase 3 after the custom objects and OWD settings are in place.

Dev Org Setup

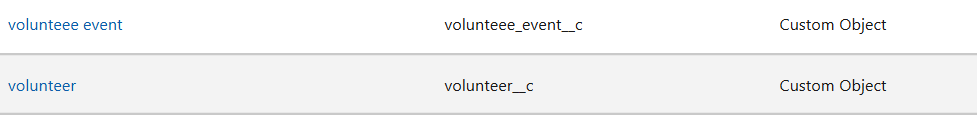
The development environment was prepared to support the project implementation.

* A Salesforce Developer Edition org was set up to build the project.
* A GitHub Repository was created for version control of the source code.
* VS Code and SFDX were configured for developing the Apex backend and future Lightning Web Components.

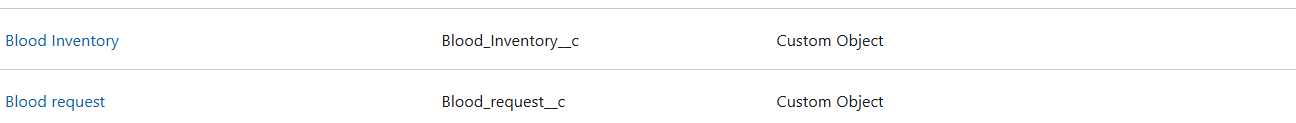
**Phase 3 – Data Modeling & Relationships**

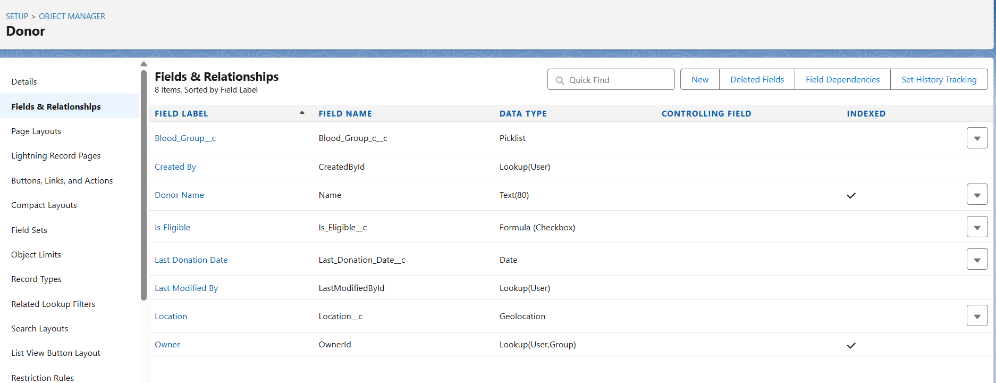
**1. Standard & Custom Objects**

* **Standard Objects:**
  + *User* (for staff, NGO coordinators, volunteers)
  + *Campaign* (for blood donation drives)
  + *Contact* (donors and recipients)
  + *Account* (hospitals, NGOs, organizations)
* **Custom Objects:**
  + *Donor* (blood group, eligibility, donation history)
  + *Blood\_Request* (urgent hospital request, request date, priority)
  + *Volunteer* (NGO/individual volunteers for campaigns)
  + *Donation\_Event* (blood drives and camps, linked to campaigns)

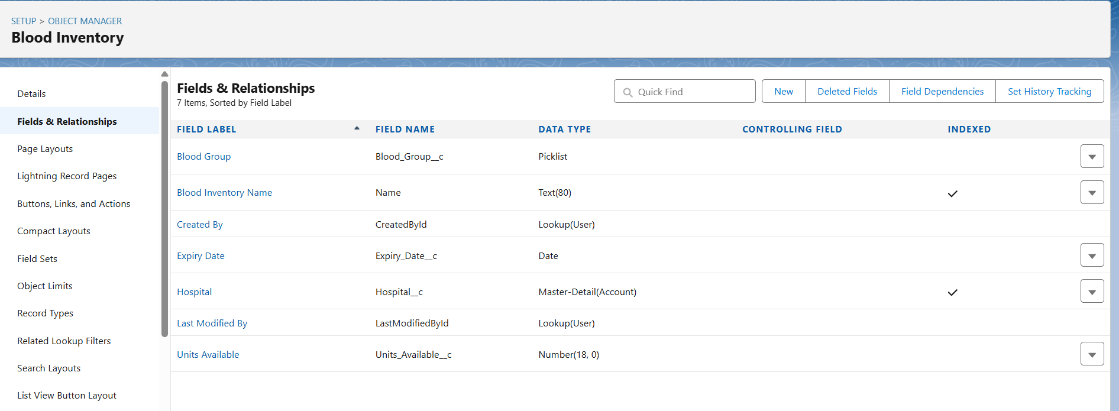
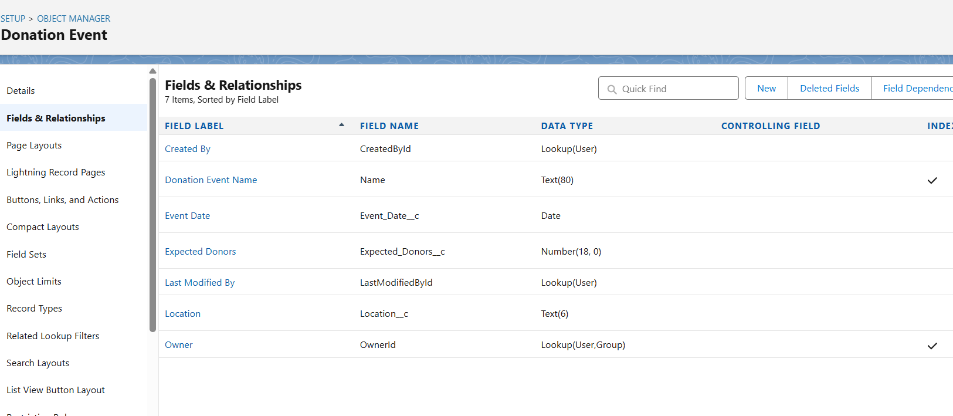


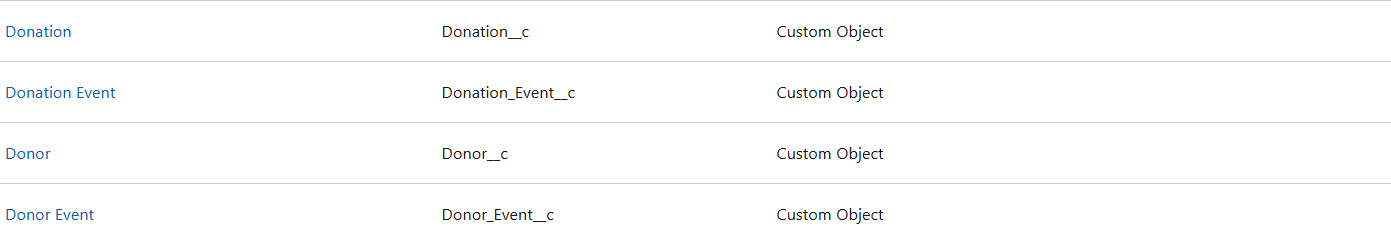
* + *Blood\_Inventory* (available blood units, expiry dates)





**2. Fields**

* **Donor:** Blood Group (Picklist), Last Donation Date (Date), Eligibility Status (Formula), Location (Geolocation).
* **Blood\_Request:** Request Type (Emergency/Planned), Quantity Required (Number), Request Status (Picklist).
* 
* 
* **Volunteer:** Skills (Text), Availability (Date/Time), Hours Contributed (Number).
* **Donation\_Event:** Event Location (Text), Date (Date), Expected Donors (Number).
* **Blood\_Inventory:** Blood Group (Picklist), Units Available (Number), Expiry Date (Date).



**3. Record Types**

* *Donor Record Types:* Regular Donor, Rare Donor (for rare blood groups).
* *Blood\_Request Record Types:* Emergency Request, Planned Request.
* *Volunteer Record Types:* NGO Volunteer, Independent Volunteer.

**4. Page Layouts**

* Customize layouts for usability:
  + Donor Page → Show donation history, eligibility, and contact info.
  + Blood Request Page → Show request status, hospital details, and donor matches.
  + Volunteer Page → Show event participation and contribution hours.

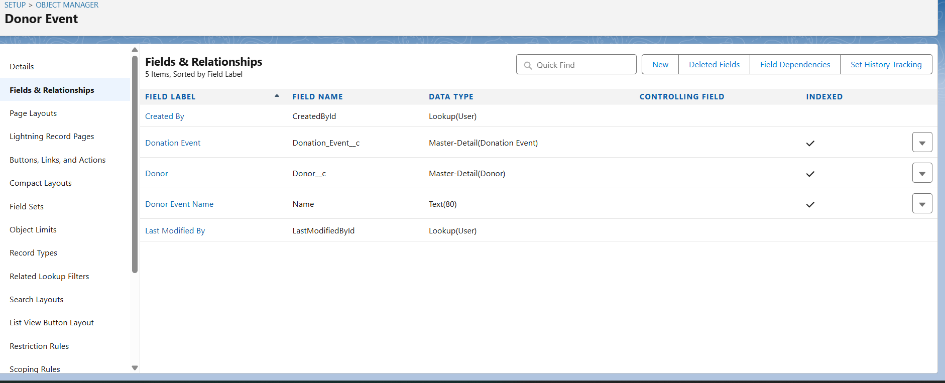
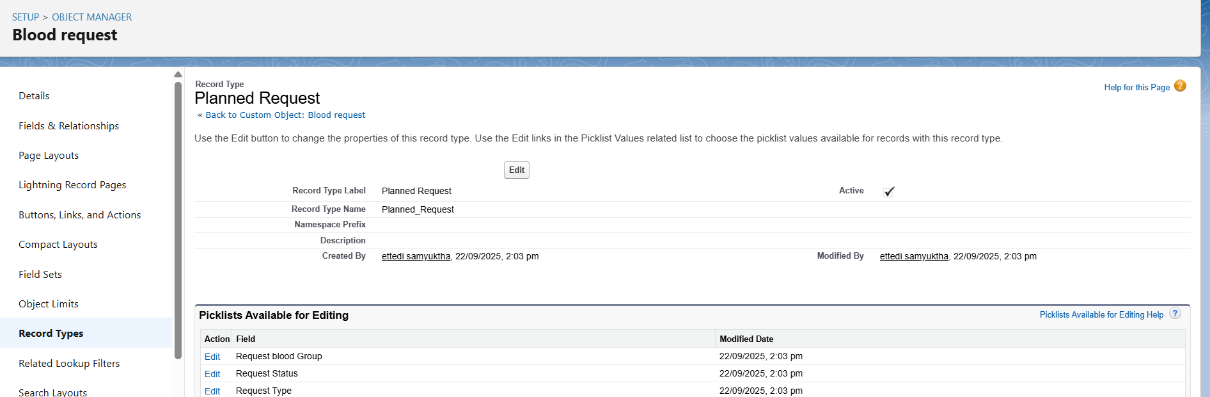
**5. Compact Layouts**

* **Donor Compact Layout:** Blood Group, Last Donation Date, Eligibility.
* **Blood Request Compact Layout:** Request Type, Quantity, Status.
* **Volunteer Compact Layout:** Name, Availability, Skills.

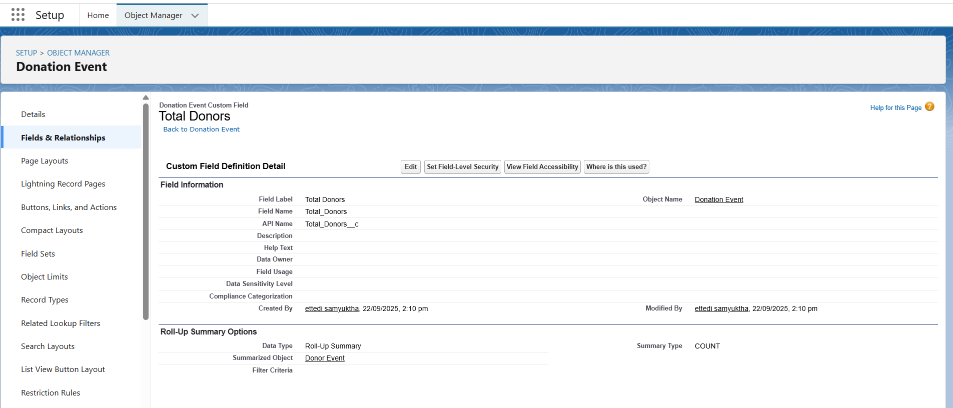
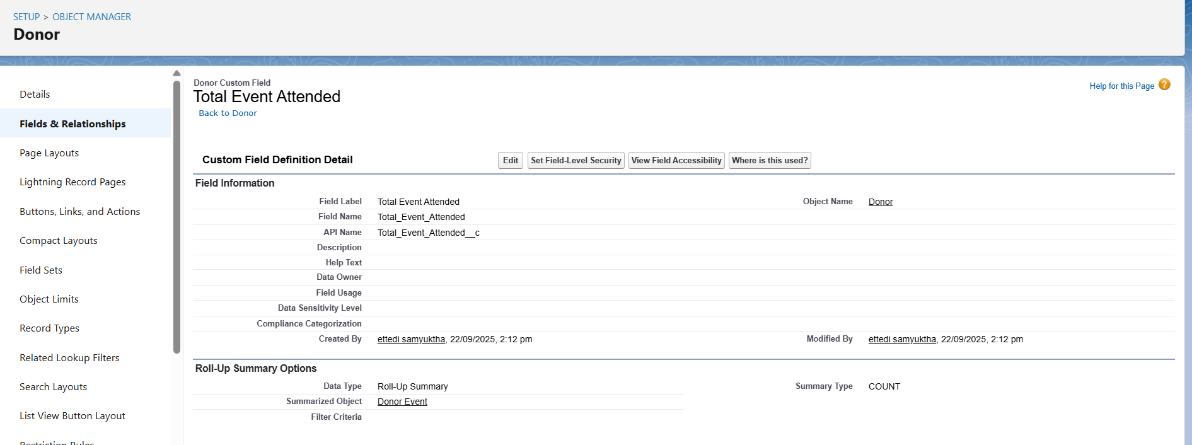
**6. Schema Builder**

* Use **Schema Builder** to visualize object relationships:
  + Donor → Donation\_Event (Many-to-Many via Junction Object).
  + Blood\_Request → Donor (Lookup).
  + Volunteer → Donation\_Event (Junction Object).
  + Blood\_Inventory → Hospital (Master-Detail).

**7. Lookup vs Master-Detail vs Hierarchical**

* **Lookup:** Donor ↔ Blood\_Request (optional relationship, donor may or may not respond).
* 
* **Master-Detail:** Hospital → Blood\_Inventory (ownership & roll-up summaries).
* 
* **Hierarchical:** Used in *User* object (e.g., Staff reporting to Manager).

**8. Junction Objects**

* **Donor\_Event\_\_c:** Junction between Donor and Donation\_Event.
* **Volunteer\_Event\_\_c:** Junction between Volunteer and Donation\_Event.
* 
* Purpose: Allow many-to-many relationships for tracking multiple donors/volunteers in multiple events.
* 

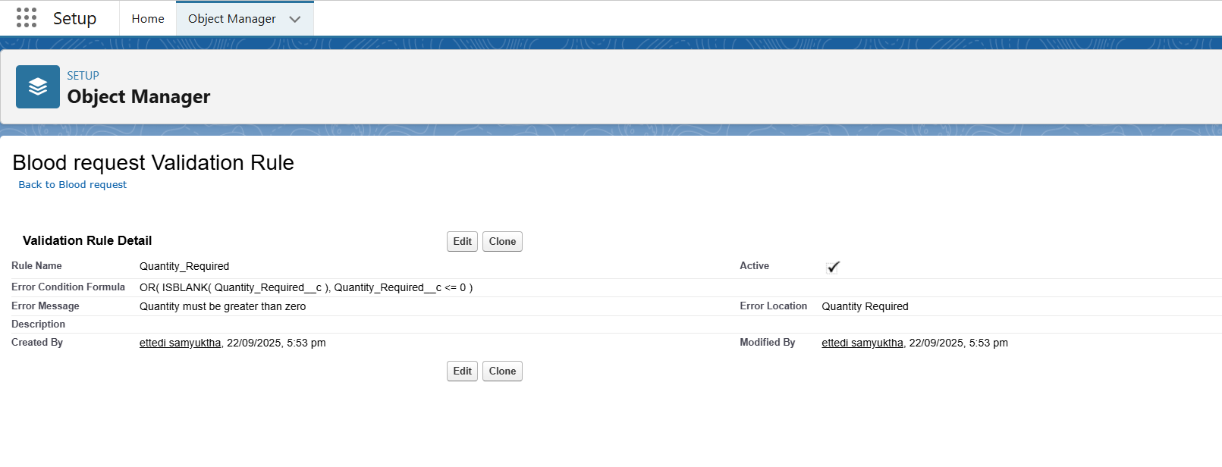
**9. External Objects**

* Integrate with **external hospital systems** or **government blood bank APIs** for:
  + Real-time blood inventory sync.
  + Automatic update of donor eligibility from medical labs.

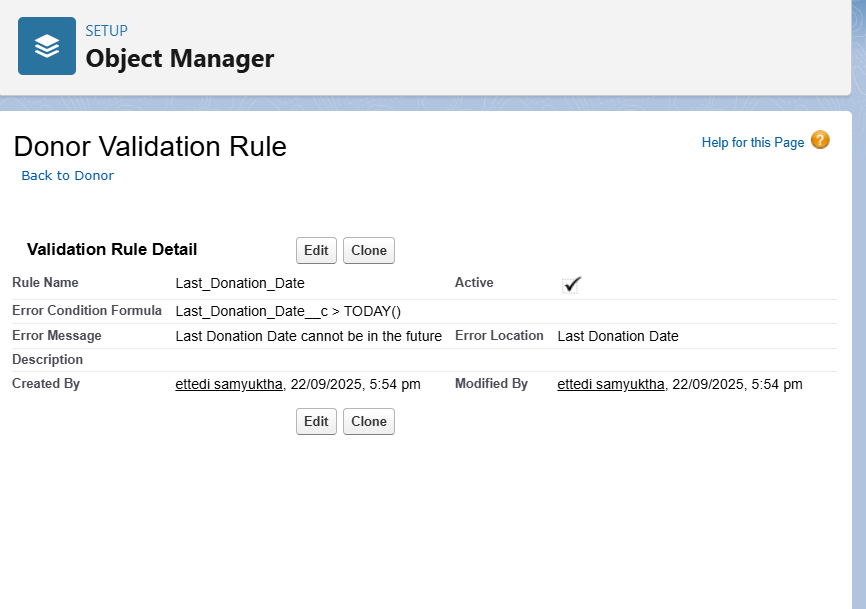
**Phase 4 – Process Automation (Admin)**

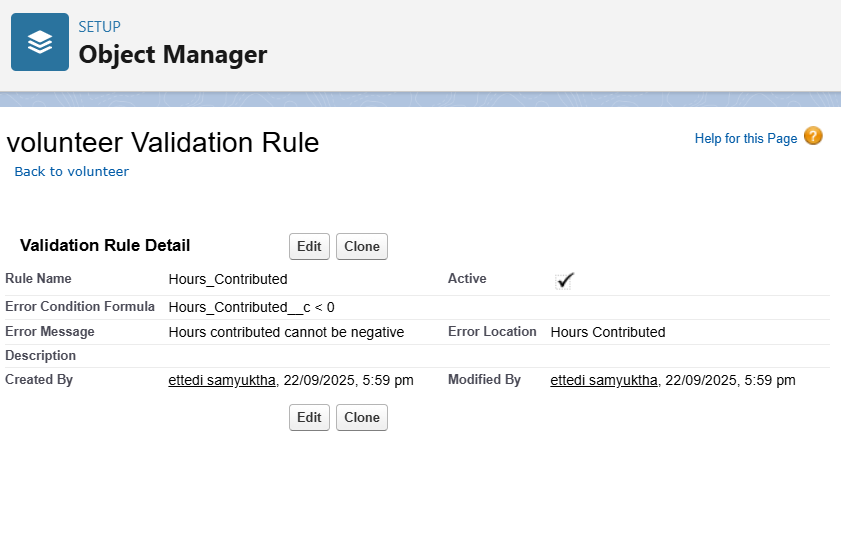
**1. Validation Rules**

* Ensure **data quality and accuracy** before saving records.
* Examples:
  + On Blood\_Request\_\_c: Prevent requests with Quantity\_Required\_\_c <= 0.



* + On Donor\_\_c: Ensure Last\_Donation\_Date\_\_c cannot be a future date.



* + On Volunteer\_\_c: Ensure Hours\_Contributed\_\_c ≥ 0. 

**2. Workflow Rules (Legacy)**

* Though Salesforce recommends **Flows** now, Workflow Rules can still handle simple tasks.
* Example:
* Record-Triggered Flow: Donor Matching for Emergency Requests

**3. Process Builder (Legacy)**

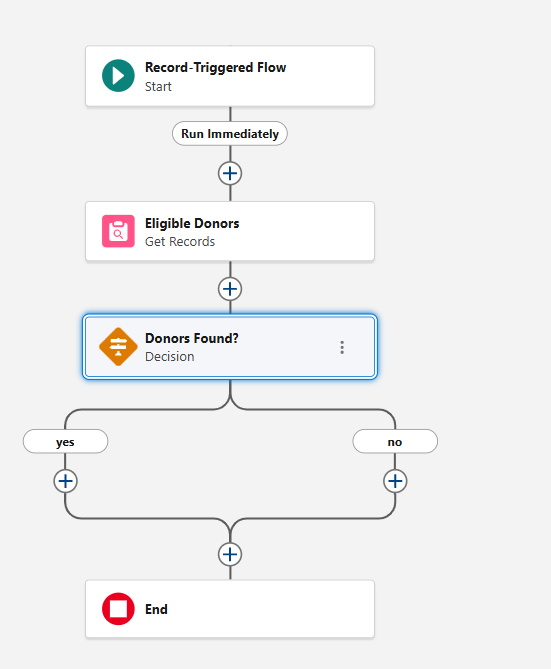
* Used for **if-then automation logic** (also being replaced by Flows).
* Example:
  + When a new **Donation\_Event\_\_c** is created, automatically create related **Tasks** for staff to prepare venue and equipment.

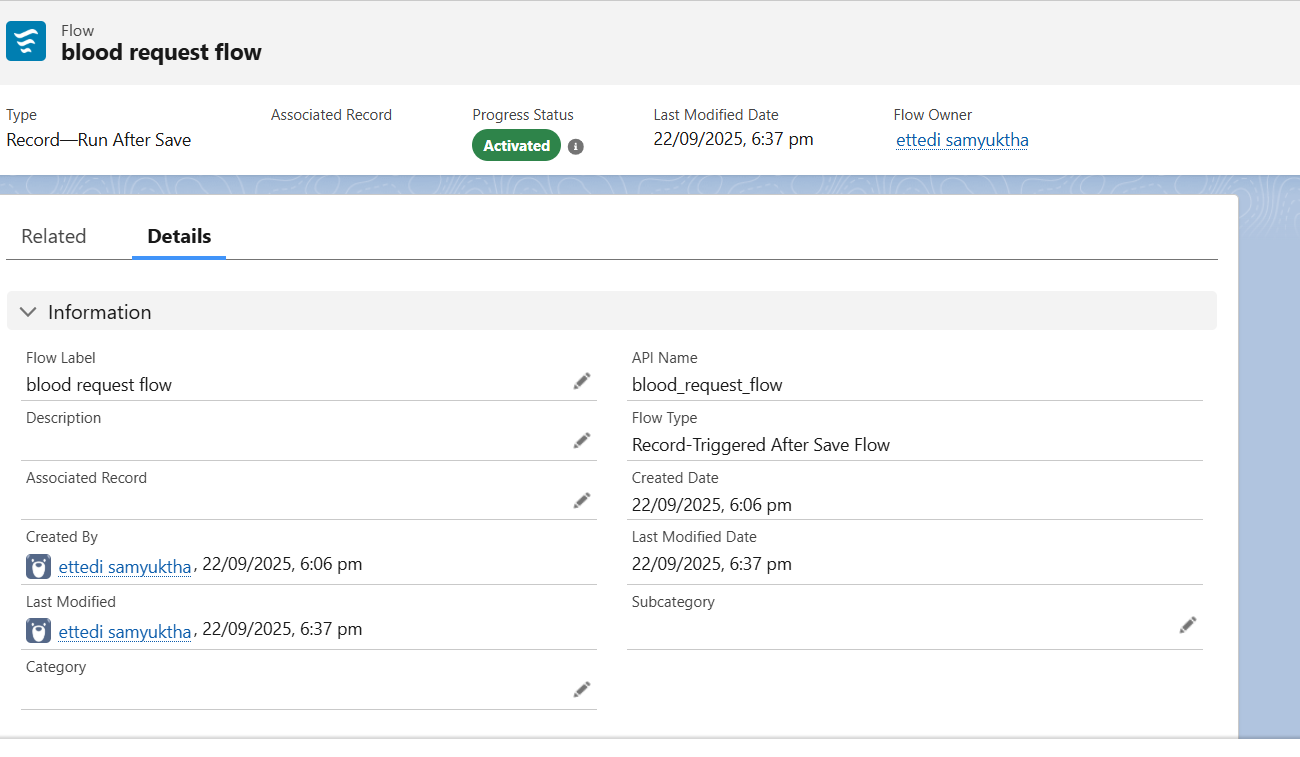
**4. Approval Process**

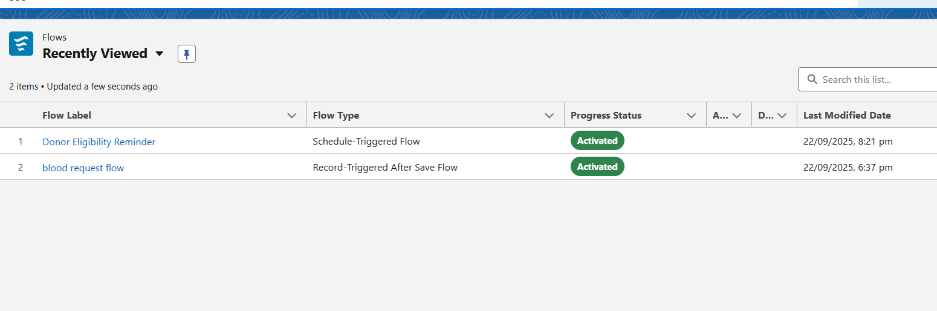
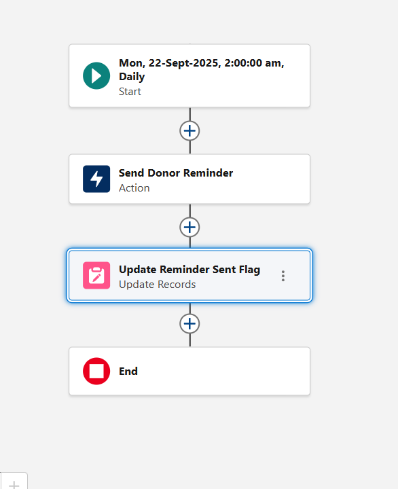
* Manage **multi-step approvals**.
* Example:
  + Large **fund donation requests** by NGOs require approval from CSR Manager → Director.
  + Steps: Submit → Pending → Approved/Rejected.

**5. Flow Builder (Recommended)**

* Main tool for automation going forward.
* Examples for Blood Donation CRM:
  + **Record-Triggered Flow:** When a new **Blood\_Request\_\_c** is logged, auto-match with eligible donors (Donor\_\_c.Is\_Eligible = TRUE AND Blood\_Group = Request\_Blood\_Group).

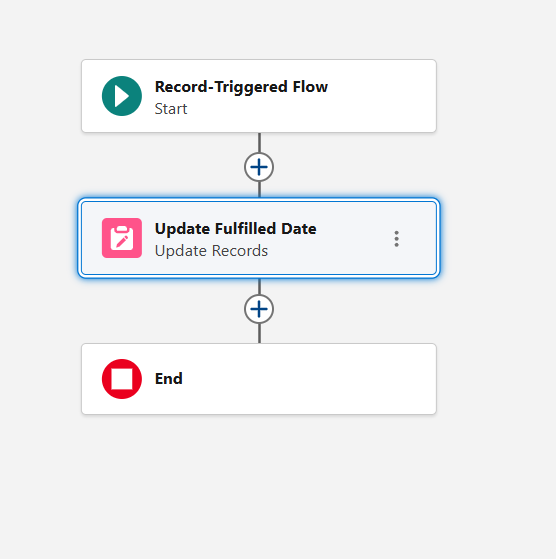


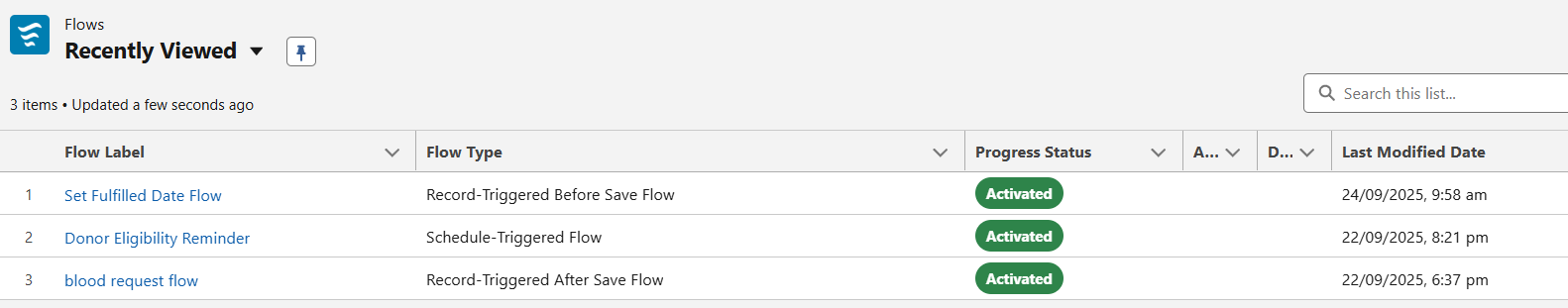
* + **Scheduled Flow:** Send reminders to donors whose eligibility date has just become valid (90+ days since last donation). Run daily at 2 AM → find donors who are eligible to donate again after 90+ days → send reminder → mark as “reminded”.
  + 



**6. Field Updates**

* Automatically update fields based on conditions.
* Examples:
  + When Blood\_Request\_\_c.Status = Fulfilled, update Fulfilled\_Date\_\_c = TODAY().





**Phase 5 – Apex Programming (Developer)**

**1. Classes , Objects &Apex Triggers**

* You want to check donor eligibility, update donor’s last donation date, and assign donors to events.
* Instead of writing this logic in triggers again and again, you put it in Apex Classes.

**DonorService.cls (Service Class):**

Purpose:  
Handles all donor-related business logic in a centralized, reusable way.

* Check donor eligibility: Ensures a donor hasn’t donated in the last 90 days.
* Update last donation date: Updates Last\_Donation\_Date\_\_c after a donation.
* Fetch eligible donors: Returns a list of donors filtered by blood group and location who are eligible to donate.

**2.BloodRequestHandler.cls (Handler Class)**

Purpose:  
Handles the business logic related to Blood\_Request\_\_c operations, called by the trigger.

Responsibilities:

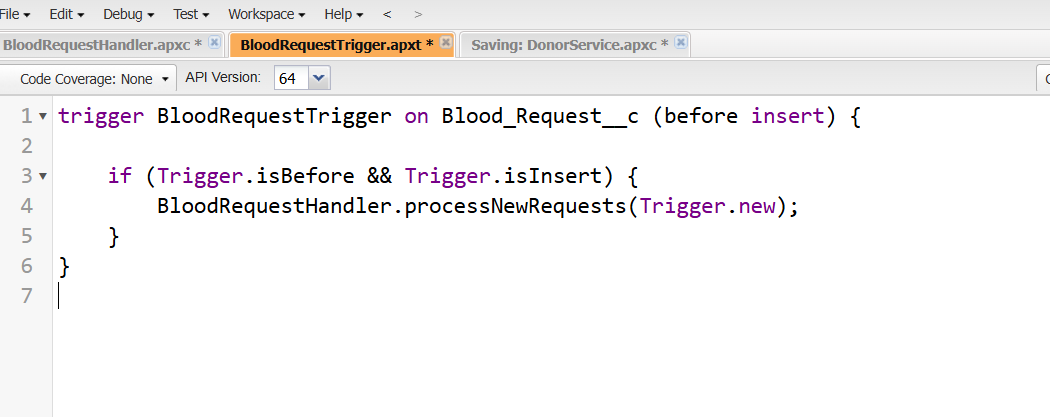
* Processes new blood requests.
* Calls DonorService.fetchEligibleDonors() to get eligible donors.
* Selects a donor (example: first eligible donor) and links it to the blood request.
* Updates the donor’s last donation date using DonorService.updateLastDonationDate().

**3. BloodRequestTrigger.trigger (Trigger)**

Purpose:  
Listens to Salesforce database events (insert, update, delete, etc.) on Blood\_Request\_\_c and calls the handler.

Responsibilities:

* Fires before insert of Blood\_Request\_\_c records.
* Calls BloodRequestHandler.processNewRequests(Trigger.new) to process incoming requests.

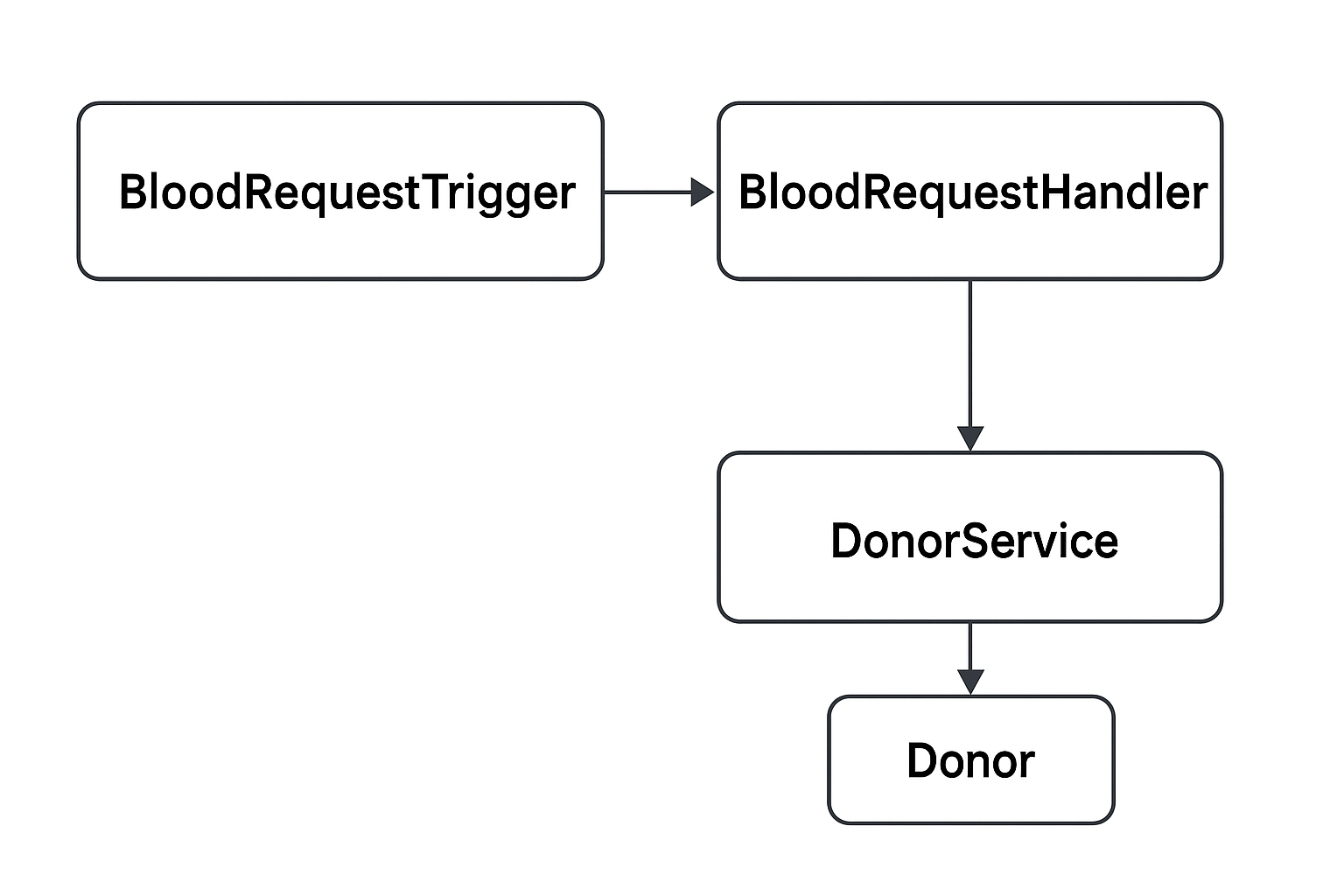


**4.BloodRequestTest.cls (Test Class)**

Purpose:  
Verifies that all classes (trigger, handler, and service) work correctly. Provides Salesforce code coverage for deployment.

Responsibilities:

* Creates test donor data using @TestSetup.
* Inserts Blood\_Request\_\_c records to fire the trigger.
* Validates that:
  + Donor was assigned to the blood request.
  + Donor’s last donation date was updated.
  + Donor eligibility logic works correctly.



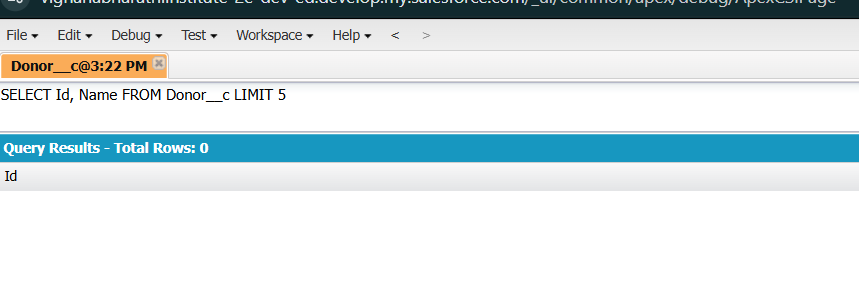
**4. SOQL :**

* Use **SOQL** to fetch donors for matching.
* Example:

SELECT Id, Name

FROM Donor\_\_c

LIMIT 1

* 
* Use **SOSL** for keyword searches (e.g., searching donors by name/email).

**5. Control Statements**

* Use if, for, while for donor/event loops.
* Example: Loop through donors to assign them to requests.

**6. Future Methods**

* Use for **callouts or async updates**.
* Example: Future method to update an external hospital system when a donation is completed.

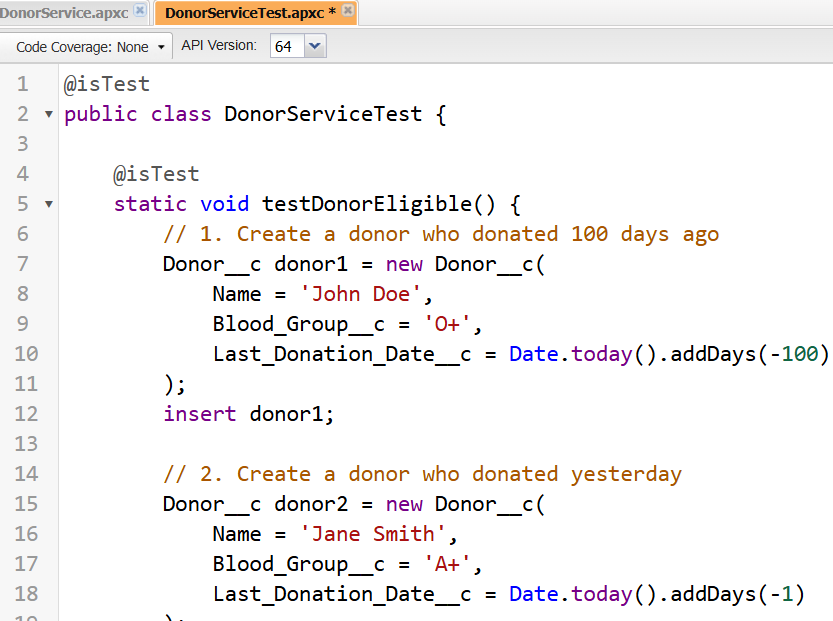
**7. Exception Handling**

* Use try…catch blocks for error handling.
* Store errors in a custom Error\_Log\_\_c object for admins to review.

**8. Test Classes**

* Write **@isTest classes** for every Apex class/trigger.
* Ensure **75%+ code coverage** (mandatory for deployment).
* Example: Test donor eligibility logic with multiple scenarios.

**DonorService.cls** that checks if a donor is eligible (90+ days since last donation).



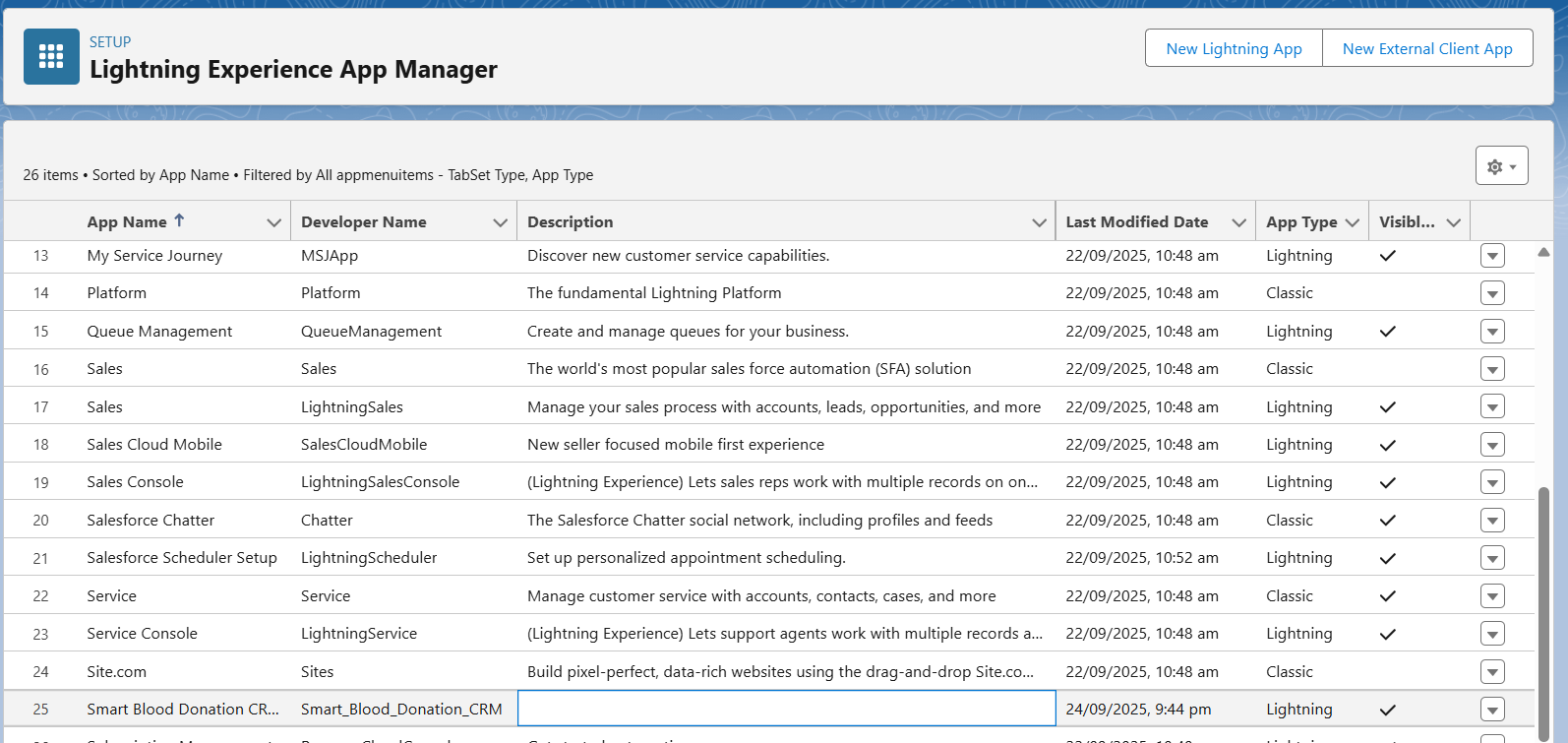
**9. Asynchronous Processing**

* Use **Batch Apex**, **Queueables**, **Future methods**, and **Scheduled Apex** to handle:
  + High-volume donor/event processing.
  + Notifications and integrations.
  + Background calculations (stats, reports).

**Phase 6: User Interface Development**

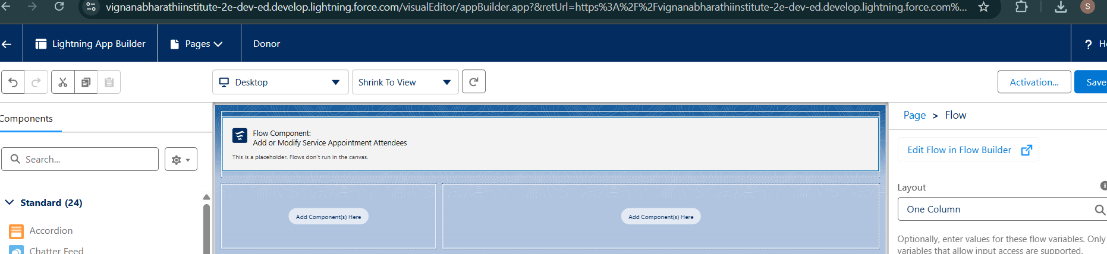
**1. Lightning App Builder**

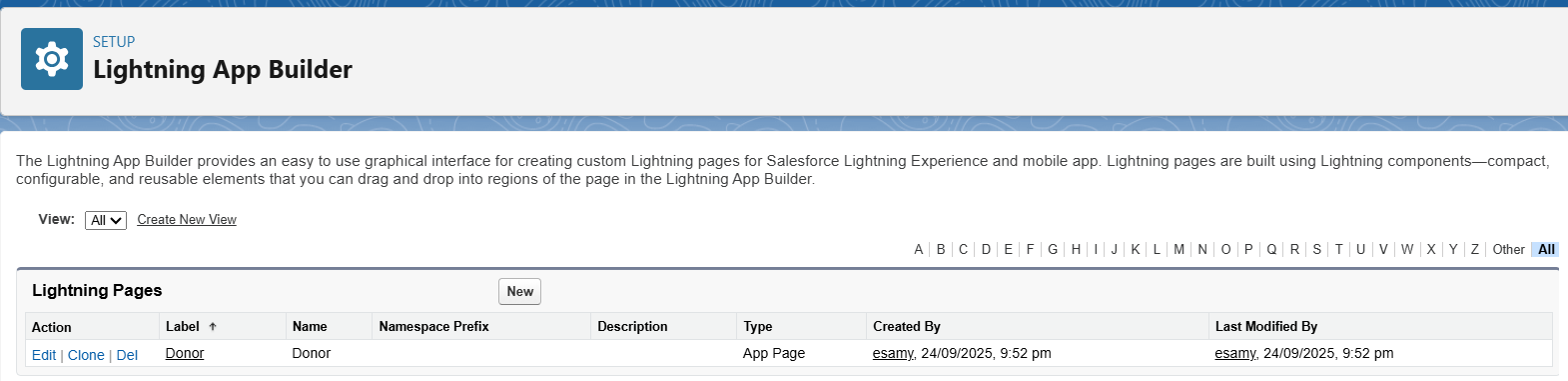
* Use it to design custom apps for different user groups (e.g., Donor Management, Hospital Requests).



**2. Record Pages**

* Customize how a record looks for **Donor\_\_c** or **Blood\_Request\_\_c**.
* **Steps:**
  1. Open a record → click ⚙️ → **Edit Page**.
  2. Add components: Highlights Panel, Related List, Tabs, Custom LWC (like "Eligible Donors Near You").
  3. Save & Activate for specific apps/users.



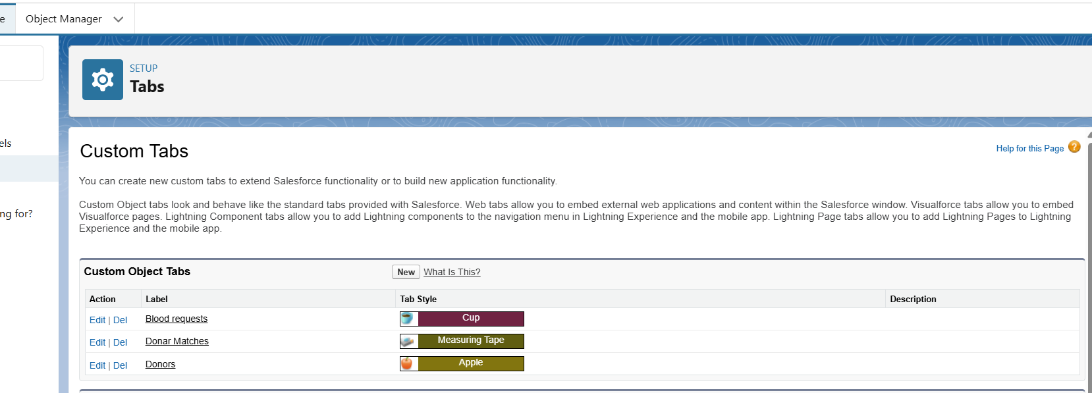


👉 Example: For **Blood\_Request\_\_c**, add a related list of **Donor\_Match\_\_c**.

**3. Tabs**

* Tabs make objects accessible in the app navigation bar.

**Donor\_\_c**, **Blood\_Request\_\_c**, **Donor\_Match\_\_c**.

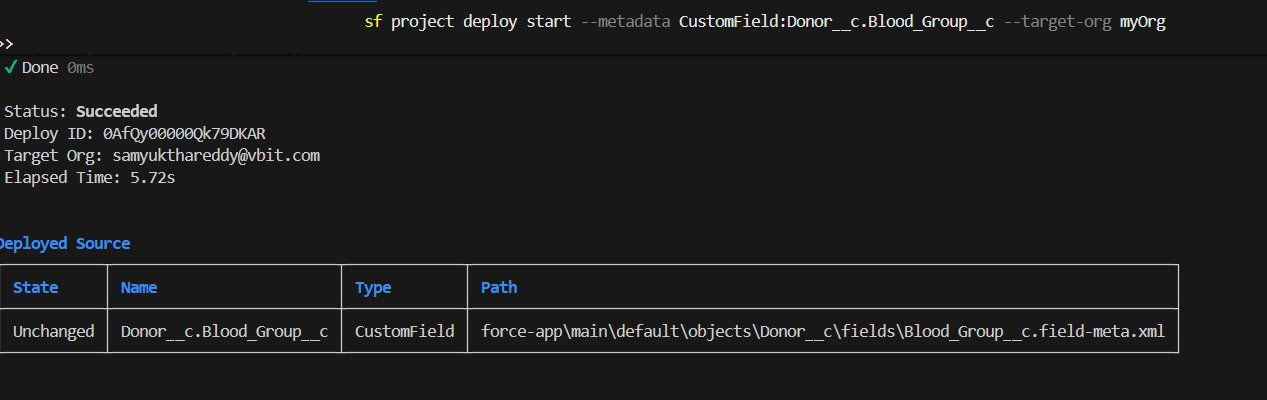


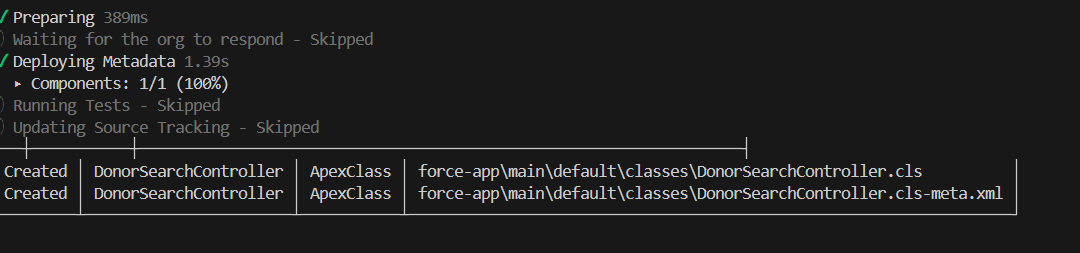
**4. Home Page Layouts**

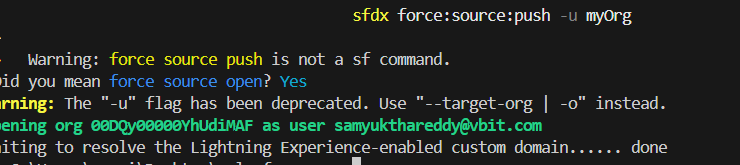
* Create dashboards/quick actions for different roles.
* **Steps:**
  1. Setup → Lightning App Builder → New → Home Page.
  2. Add:
     + Recent Requests
     + Donor Eligibility Reminders
     + Reports & Dashboards
  3. Assign to **CSR Manager App** or **Hospital Staff App**.

**5. LWC (Lightning Web Components)**

Custom components for rich UI.



**

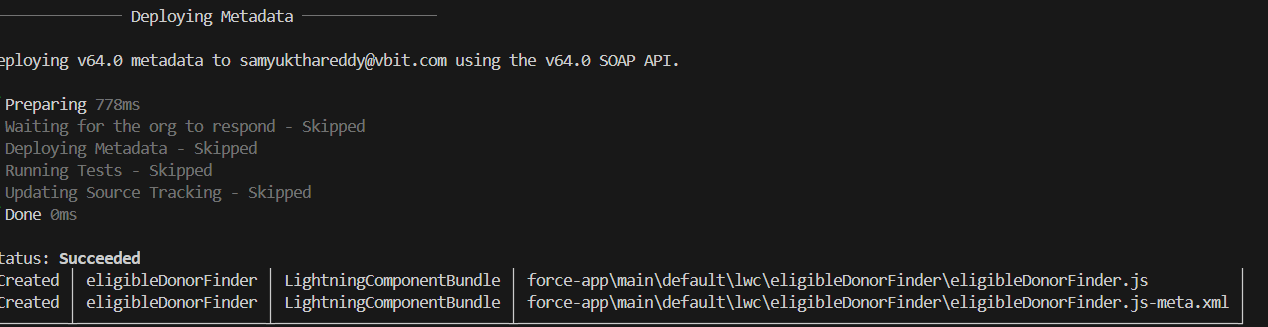
**

**6.** Navigation Service

* Lets you navigate between pages/records inside LWCs.

Example: Navigate from Donor list → Donor record page

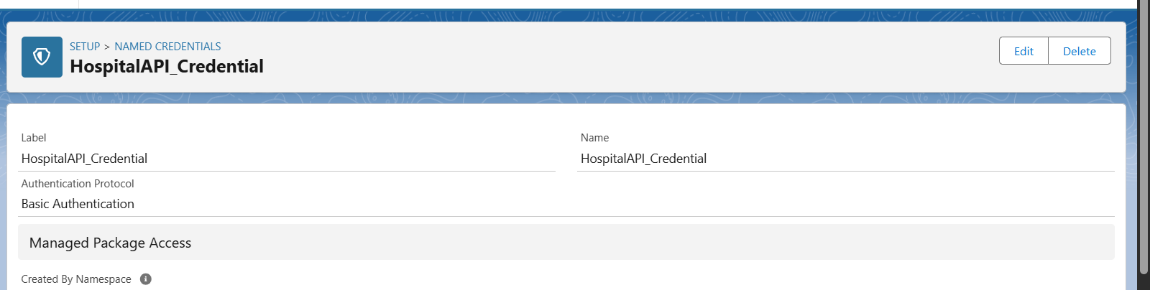
* In Lightning Web Components (LWCs), navigating between pages or records is handled using the Navigation Service, specifically the NavigationMixin. This allows your LWC to programmatically open standard Salesforce pages like record detail pages, object home pages, list views, or even external URLs.LWCs → Build smart custom UI (like donor search).
* Navigation Service → Smooth movement between screens.

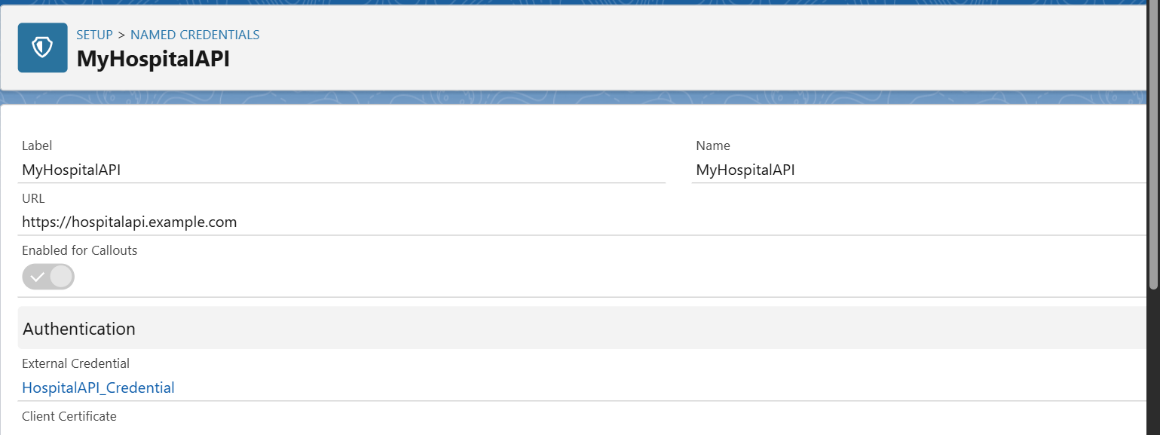


**Phase 7: Integration & External Access**

1.Named Credentials

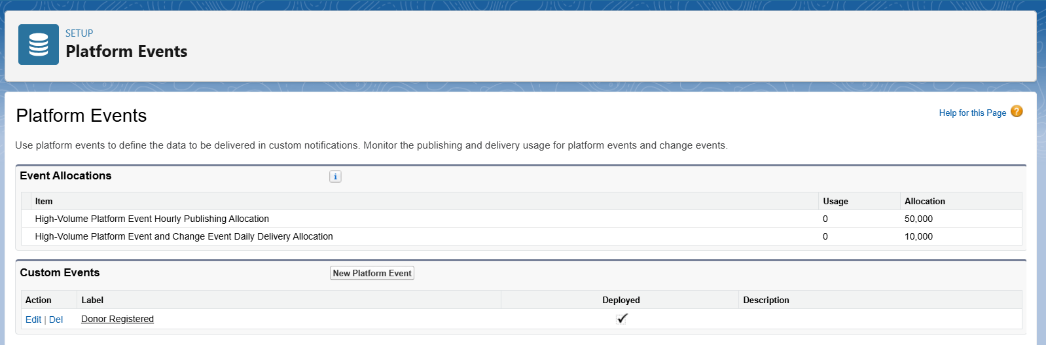
* Purpose: Securely store authentication settings for external systems.
* Use Case: Connect to a REST API without hardcoding usernames/passwords or tokens.
* Example: Access a hospital API to fetch donor blood availability.



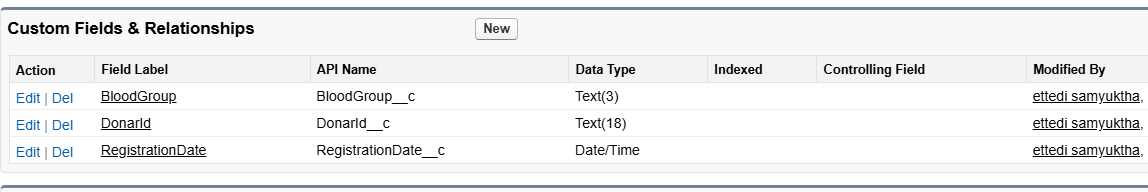


2.Platform Events

* Purpose: Event-driven architecture inside Salesforce.
* Example: Publish a DonorRegistered\_\_e event when a new donor record is added.
  + Define Platform Event object.



* + Publish events from Apex or Flow.



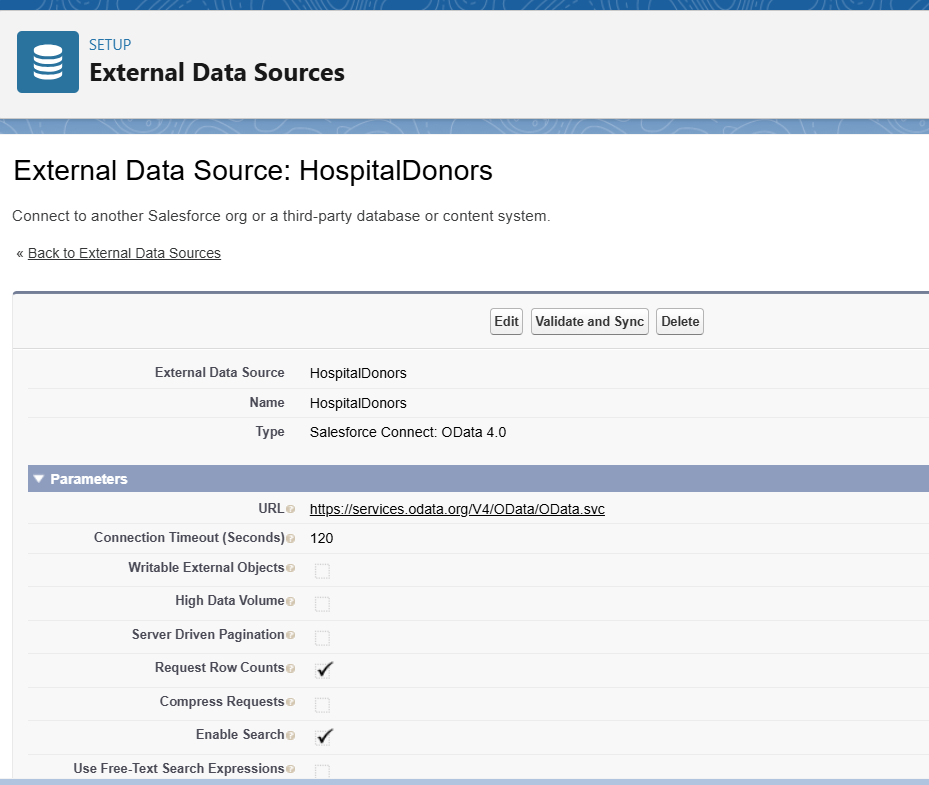
* + Subscribe via triggers, Process Builder, or external systems.

3. Change Data Capture (CDC)

* Purpose: Get real-time updates when Salesforce records change.
* Example: Notify an external blood bank system when a donor’s info changes.
* Key Points:
  + Tracks create, update, delete, and undelete.
  + Can subscribe via CometD or Apex triggers.

4. Salesforce Connect:

* Purpose: Access external data without storing it in Salesforce.



 External table: HospitalDonors with fields DonorId, Name, BloodGroup

 External Object in Salesforce: HospitalDonor\_\_x

 Use in Flow:

* Fetch all donors with BloodGroup = 'O+'
* Send notifications to blood banks

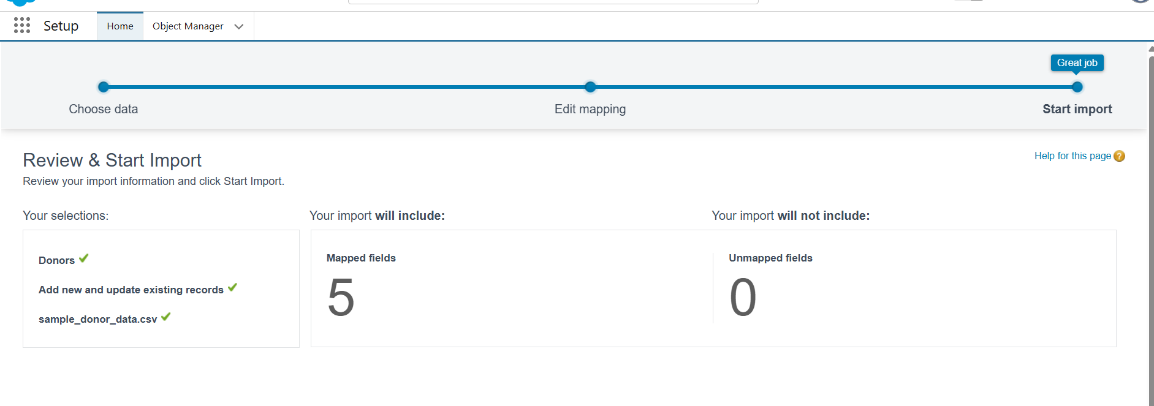
5.OAuth & Authentication

* Purpose: Securely authenticate external systems or users.
* Example Flows:
  + OAuth 2.0 JWT Bearer → API calls from server-to-server.
  + Web Server Flow → Authenticate users via external apps.
* Use Case: Allow a mobile app to access Salesforce donor data without storing passwords.

**Phase 8: Data Management & Deployment**

1. Data Import Wizard

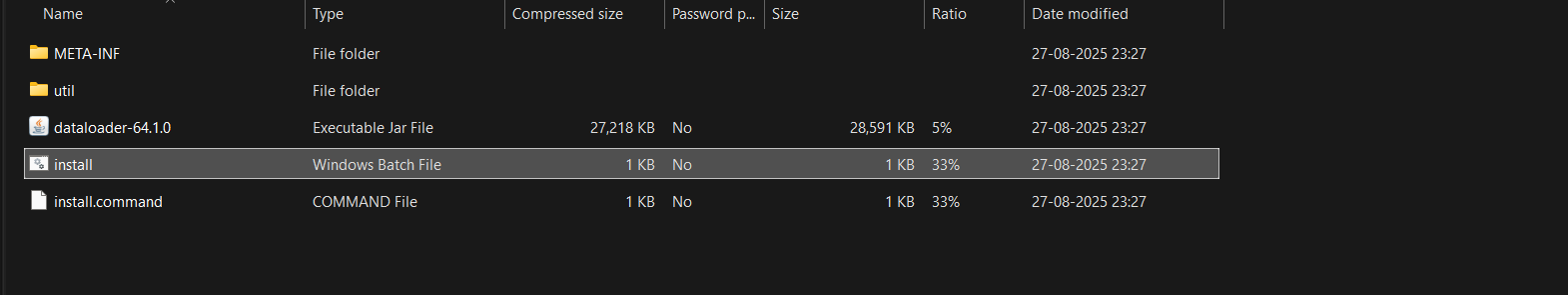
* Purpose: Simple tool for importing small datasets (up to 50,000 records).
* Supports: Accounts, Contacts, Leads, Solutions, Campaign Members, Custom Objects.
* Steps:
  1. Go to Setup → Data Import Wizard.
  2. Select object (e.g., Donor\_\_c).
  3. Upload CSV file.
  4. Map fields → Start Import.



2. Data Loader

* Purpose: Handles large datasets (up to 5 million records).
* Supports: Insert, Update, Upsert, Delete, Export.
* Steps:
  1. Install Data Loader (desktop app).
  2. Log in with Salesforce credentials.
  3. Choose operation (e.g., Insert Donor\_\_c records).
  4. Select CSV → Map fields → Execute.



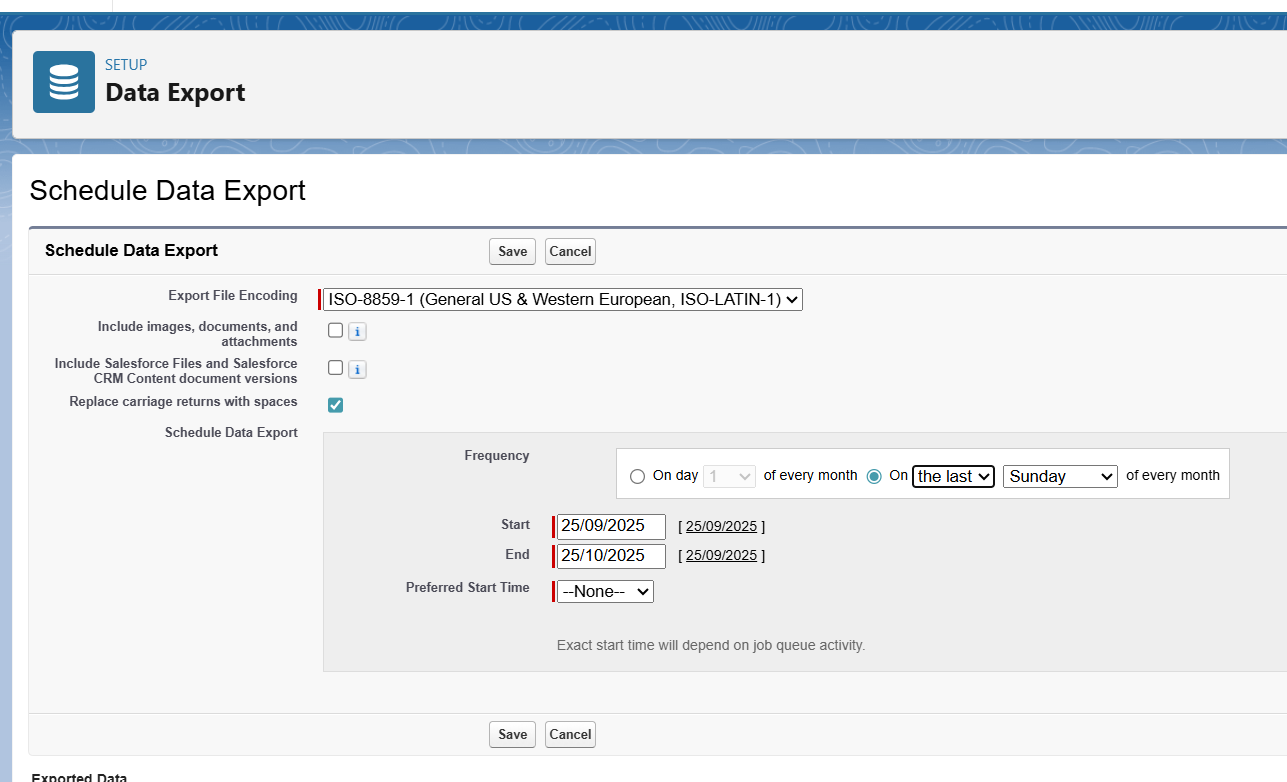


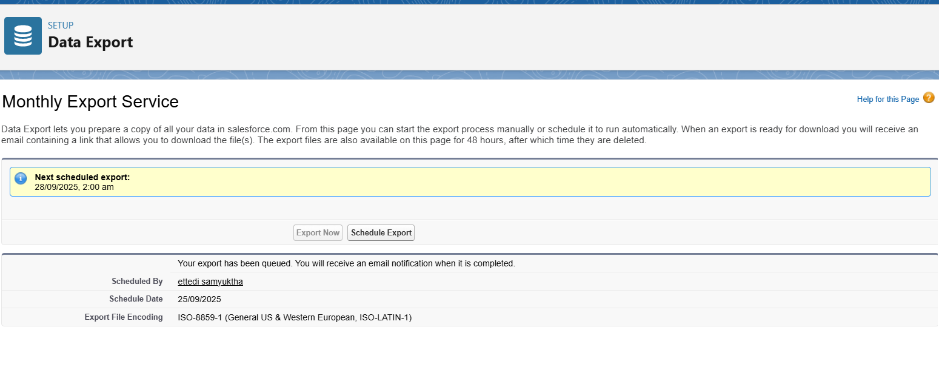
3. Duplicate Rules

* Purpose: Prevent duplicate data entry.
* How to Set Up:
  1. Setup → Duplicate Rules.
  2. Define Matching Rule (e.g., Donor\_\_c.Email = Donor\_\_c.Email).
  3. Create Duplicate Rule → choose Block or Allow with Alert.
* Use Case: Prevent two donor records with same email.

4. Data Export & Backup

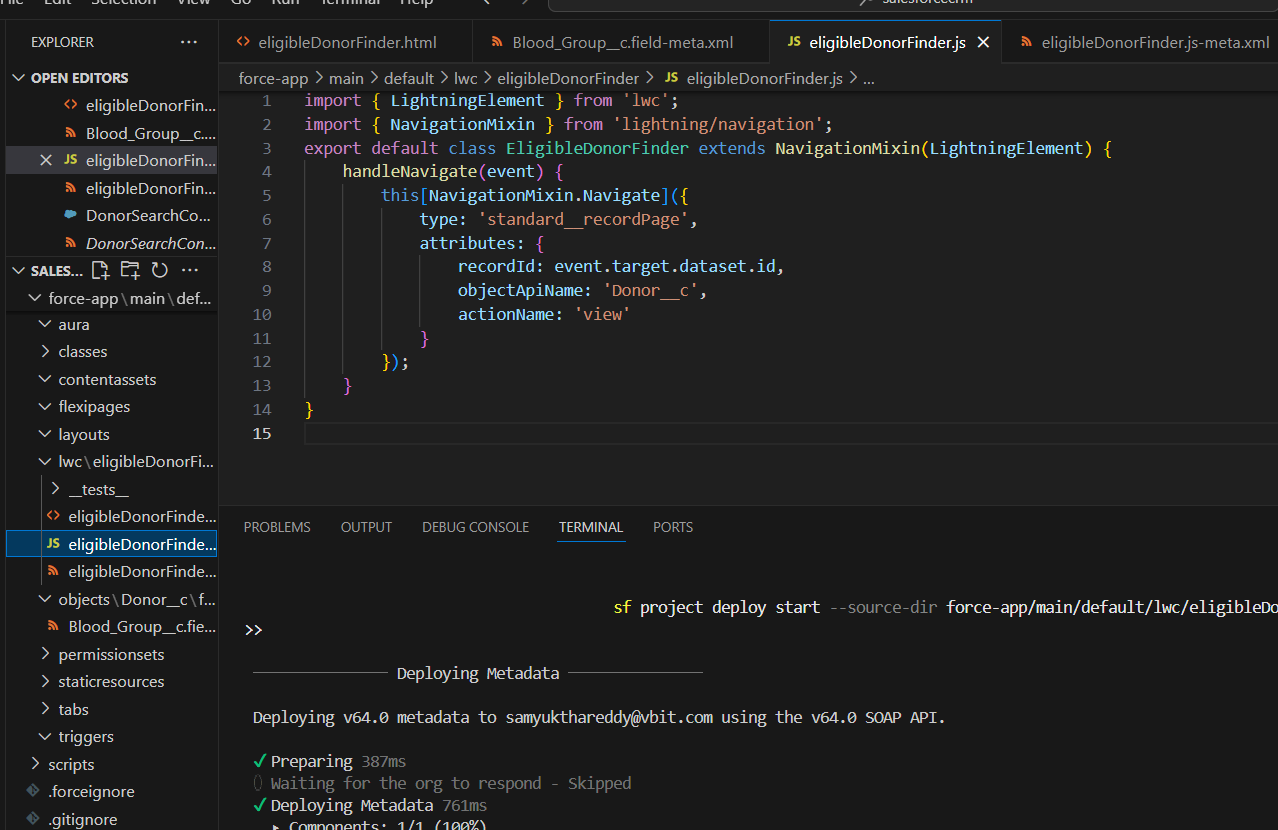
* Purpose: Backup Salesforce data for recovery or offline analysis.
* Options:
  + Data Export Service (Setup → Data Export → Schedule Weekly/Monthly).
  + Data Loader Export (CSV extraction).





5. VS Code & SFDX (Deployment)

* Purpose: Move metadata between orgs (Dev → Test → Prod).



**Phase 9: Reporting, Dashboards & Security Review**

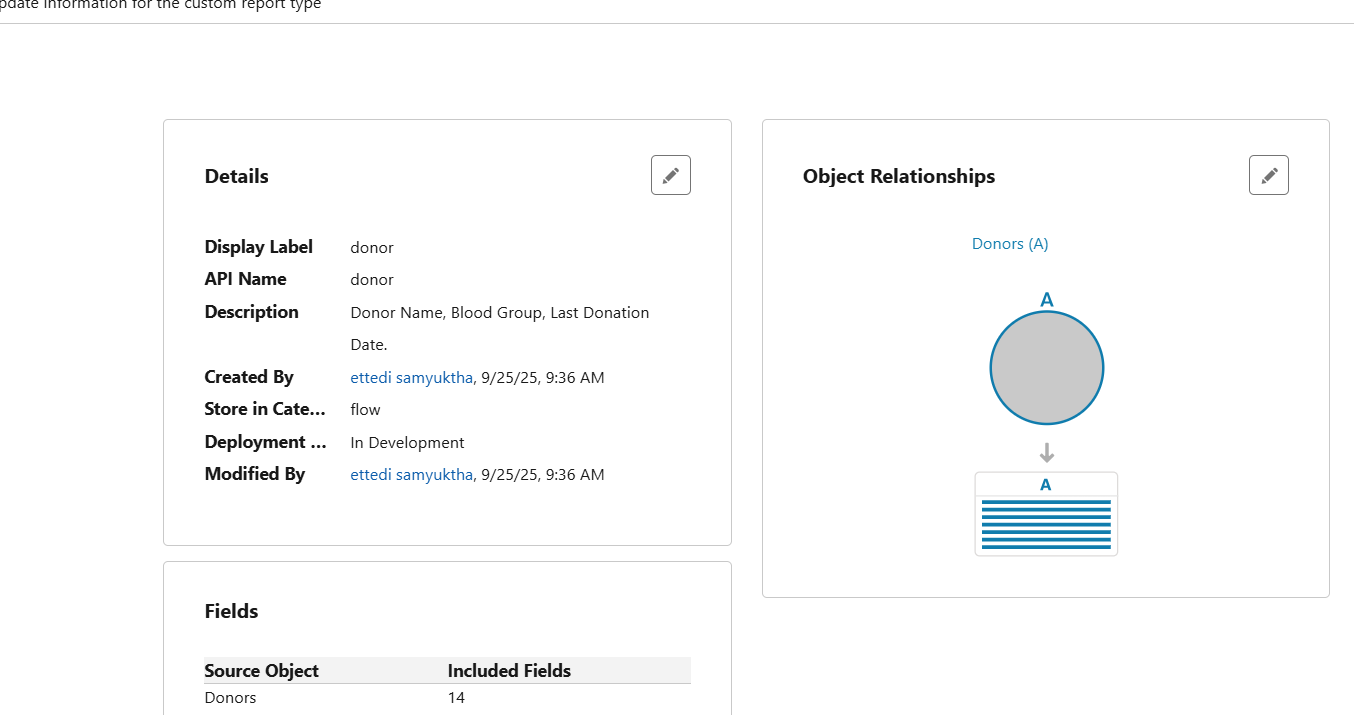
1. Reports

Salesforce offers 4 main report formats:

* Tabular Report → Simple list (like Excel rows).  
  *Example:* List of all donors with their blood group.
* Summary Report → Grouped rows + subtotals.  
  *Example:* Donors grouped by blood group, with counts.
* Matrix Report → Grouped by rows & columns.  
  *Example:* Donors by blood group (rows) vs. location (columns).
* Joined Report → Combine multiple report blocks.  
  *Example:* Donors + Blood Requests in a single view.

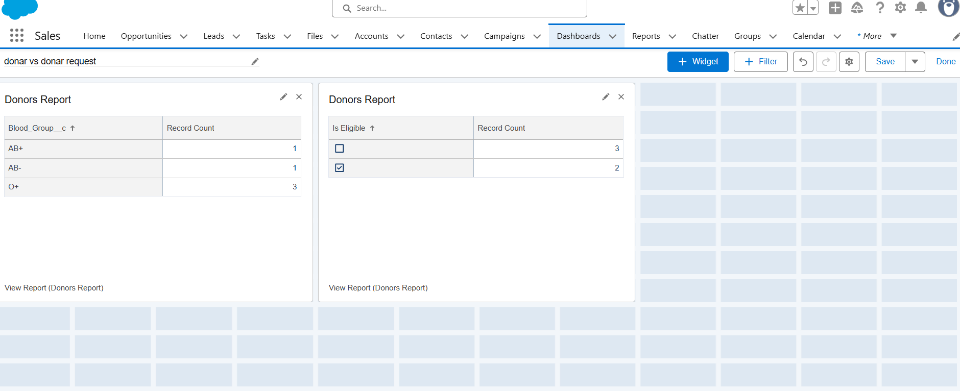
2. Report Types

* Define which objects and fields are available in a report.
* Standard Report Types → Auto-created for most objects.
* Custom Report Types → You choose primary & related objects.  
  *Example:* Donor\_\_c (primary) + Blood\_Request\_\_c (related).



3. Dashboards

* Visual display of multiple reports in one screen.
* Components: Charts, Gauges, Tables, Metrics.  
  *Example:* Dashboard showing:
  + Donor count by blood group
  + Pending requests by city
  + Average donation interval



4. Sharing Settings

* Org-Wide Defaults (OWD): Set baseline (Private, Public Read Only, etc.).
* Role Hierarchy: Higher roles inherit access.
* Sharing Rules: Grant extra access (criteria-based or owner-based).
* Manual Sharing: Record-level sharing.

5. Login IP Ranges

* Restrict logins to trusted IP addresses.
* Configured at Profile level.  
  *Example:* Staff can log in only from hospital’s secure network.

6. Audit Trail

* Tracks changes made in Setup (last 6 months).
* Useful for compliance & debugging.  
  *Example:* Who modified donor sharing rules last week.

**Phase 10: Final Presentation & Demo Day**

1. Project Overview

* Project Name: *Smart Donation CRM*
* Problem Statement: Inefficient management of donors and blood requests in hospitals/NGOs.
* Objectives:
  + Track donors and donations
  + Manage blood requests efficiently
  + Generate reports and dashboards for analytics
* Scope: Features implemented, limitations, and future enhancements.

2. Feedback Collection

| Stakeholder | Role | Feedback / Comments | Action Items |
| --- | --- | --- | --- |
| Dr. Ramesh | Hospital Coordinator | Add email notifications for donors | Future Enhancement |
| Priya Sharma | NGO Admin | Dashboard should filter by city | Implement Dashboard Filters |
| Mentor | Project Guide | Good visualization, add export reports feature | Future Enhancement |