# NGO Donation & Volunteer Management System

---

## Objective of the Project

The objective of the **NGO Donation & Volunteer Management System** is to design and implement a Salesforce-based platform that centralizes the management of donors, donations, volunteers, and events for non-profit organizations. The system aims to simplify donor and volunteer engagement, ensure transparency in financial contributions, and improve operational efficiency through automation, secure data handling, and real-time reporting.

Specifically, this project will:

- Provide NGOs with a unified system to record and manage donor and volunteer data.
- Automate acknowledgement emails, approval workflows for high-value donations, and event capacity tracking.
- Empower managers with dashboards and reports for informed decision-making and accountability.
- Enhance donor trust and volunteer participation by ensuring accurate communication and record-keeping.
- Demonstrate Salesforce capabilities across **Admin, Automation, Apex, and User Interface Development** in a real-world non-profit use case.

## Phase 1: Problem Understanding & Industry Analysis

1. **Requirement Gathering**

We engaged with stakeholders from the NGO domain to identify critical needs in managing donations and volunteers.

➢ Track all donors with their details and donation history.
➢ Record donations with amount, type (one-time/recurring), and status.
➢ Manage volunteers with skills, availability, and event assignments.
➢ Prevent assigning more volunteers than an event's capacity.
➢ Automate donor acknowledgements and approval workflows for high-value donations.
➢ Generate reports for donors, donations, events, and volunteer engagement.

2. **Stakeholder Analysis**

- **NGO Admin** – Manages the entire system, including data access, configuration, and security.
- **Donor Manager** – Handles donor relationships, donation approvals, and reporting.
- **Volunteer Manager** – Manages volunteer registrations, skills, and event assignments.
- **Donors** – Provide contributions, receive receipts and updates.
- **Volunteers** – Offer their time and effort in assigned events.

3. **Business Process Mapping**

Flow: Donor submits donation → Donation is created → Approval required if high-value → Thank-you email sent.

Flow: Volunteer registers → Manager assigns volunteer to event → Event slots updated → Confirmation sent to volunteer.

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

In the non-profit sector, NGOs must maintain donor trust, track volunteer efforts, and demonstrate transparency. Challenges include mismanaged records, delayed communication, and lack of real-time reporting. Our system addresses these by centralizing data, automating processes, and ensuring efficient donor-volunteer engagement.

5. **AppExchange Exploration**

Salesforce AppExchange includes NGO/non-profit management apps like NPSP (Nonprofit Success Pack). While feature-rich, they may be complex for small NGOs. Our custom-built solution balances simplicity with learning outcomes by implementing essential features tailored to donation and volunteer workflows.

# Phase 2: Org Setup & Configuration

1. **Salesforce Editions**

Use a Developer Edition (free) for capstone development. If you have access to Sandboxes in a larger org, use Sandbox for development and Production only for final deployment.

2. **Company Profile Setup**

   **Company Profile:** Configure Company Information (Org Name, default locale, **Timezone: Asia/Kolkata** for your region, default currency INR or USD as required).

3. **Business Hours & Holidays**

   Define NGO working hours (e.g., 09:00–18:00) and list public holidays — helpful for scheduling approvals and SLA logic.

4. **Fiscal Year Settings**

   Keep Standard (Jan–Dec) unless your NGO uses a different fiscal calendar.

5. **User Setup & Licenses**

   **Users & Licenses:** Create test users: **NGO Admin**, **Donor Manager**, **Volunteer Manager**, and sample volunteers/donors. Assign appropriate Salesforce licenses

6. **Profiles**

   Build baseline profiles (Admin = full, Donor Manager = edit donors/donations, Volunteer Manager = manage volunteers/events). Avoid giving elevated permissions unless necessary.

7. **Roles**

   NGO Admin → Donor Manager & Volunteer Manager → Volunteers. Roles control record visibility roll-up for reporting and manager oversight..

8. **Permission Sets**
   Use permission sets to give temporary or extra privileges (e.g., Reports access, Approval permissions) instead of altering profiles.

9. **OWD (Org-Wide Defaults)**

   Secure defaults: **Donations = Private**, **Donors = Public Read Only** (or Private if needed), **Volunteers = Public Read/Write**, **Events = Public Read Only**. (Adjust based on NGO policy.)

10. **Sharing Rules**

Add role-based sharing so Donor Managers can view/edit donation records; add group-based sharing if cross-team access is needed.

11. **Login Access Policies**

    Enforce strong password policies, IP login restrictions if required, and optionally restrict login hours for non-admin users.

12. **Dev Org Setup**

    If you plan to use CLI & scratch orgs, enable **Dev Hub**; otherwise Developer Console is fine for this capstone.

13. **Sandbox Usage**
    NILL

14. **Deployment Basics**

    Document the deployment approach (change sets, unmanaged packages, or source deploy). Maintain a metadata backup and use version control (Git/GitHub).

- P2_CompanyInfo.png — Company Information (Timezone, Currency).
- P2_Users_Profiles_Roles.png — Users list + Role Hierarchy + Profiles.
- P2_PermissionSet.png — Donation Approval Permission Set.
- P2_OWD_SharingSettings.png — Org-Wide Defaults and a Sharing Rule for Donations.
- P2_AppManager_NGOManager.png — NGO Manager app configuration (navigation items).
- P2_DevHub_Settings.png — Dev Hub enabled (only if used).

# Phase 3: Data Modeling & Relationships

1. **Standard & Custom Objects**

   **Standard:** Contact (can be used for donors/volunteers if needed, but in our case we designed Custom objects for clarity).

   **Custom:**

   - **Donor** – Stores donor details (name, contact, type).
   - **Donation** – Tracks each donation transaction, including amount and status.
   - **Volunteer** – Captures volunteer details, skills, and availability.
   - **Event** – Represents NGO events with date, location, and capacity.
   - **Volunteer Event Assignment** – Junction object to manage many-to-many relationship between Volunteers and Events.

2. **Fields**

   - **Donor:** Name, Email, Phone, Donor Type (Individual / Corporate).
   - **Donation:** Amount, Date, Type (One-time / Recurring), Status (Pending / Approved), Lookup to Donor.
   - **Volunteer:** Name, Email, Skills, Availability.
   - **Event:** Event Name, Date, Location, Capacity, Status.
   - **Volunteer Event Assignment (junction):** Lookup to Volunteer, Lookup to Event.

3. **Record Types**

   **Donation:** Could be split into record types for "Individual Donation" vs "Corporate Donation."

   **Event:** Could be record-typed into "Fundraising Event" vs "Community Service Event."

4. **Page Layouts**

   - **Donor Layout:** Shows personal details and list of related Donations.
   - **Donation Layout:** Shows details of amount, type, status, and linked Donor.
   - **Volunteer Layout:** Shows volunteer details and assigned events.
   - **Event Layout:** Displays event details, available slots, and related volunteers (via junction).

5. **Compact Layouts**

- ❖ **Donor:** Name, Email, Donor Type.
- ❖ **Donation:** Amount, Status, Date.
- ❖ **Volunteer:** Name, Skills, Availability.
- ❖ **Event:** Event Name, Date, Status.

6. **Schema Builder**

- ❖ Donor → Donation (One-to-Many).
- ❖ Volunteer ↔ Event (Many-to-Many via Volunteer Event Assignment).
- ❖ Donation has a lookup to Donor; Event connects to Volunteers via junction

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

**Donor ↔ Donation:** Lookup (a donation belongs to a donor, but donor can exist independently).

**Volunteer ↔ Event:** Many-to-Many via junction object.

**Hierarchical:** Could apply if we track Volunteer reporting hierarchy, but not required for this scope.

8. **Junction Objects**

**Volunteer Event Assignment** is essential because:

- One volunteer can attend multiple events.
- One event can have multiple volunteers.

9. **External Objects**

External objects could be integrated if NGOs store donor/payment data in external systems (like PayPal or bank databases). This project uses only internal objects, but future scope could include integration with external donation/payment platforms.

# Phase 4: Process Automation (Admin)

1. **Validation Rules**

   - **Business Need:** Ensure donation entries are valid.
   - **Rule Example:** If **Donation End Date < Start Date**, show error: *"End Date must be later than Start Date."*
   - **Project Context:** Prevents wrong data entry during fundraising campaigns or donation drives.

2. **Workflow Rules** (legacy)

   **Business Need:** Notify NGO staff when a new donor record is created.

   **Example:** Auto-send email to Donor Relations Officer when a donation is logged.

   **Project Context:** Though legacy, it shows how early automation kept teams updated.

3. **Process Builder** (legacy)

   **Business Need:** Update donation status automatically.

   **Example:** If donation is received, set **Donation Status = Received**.

   **Project Context:** Earlier we'd use Process Builder, but in our project we'll use **Flow Builder**.

4. **Approval Process**

   **Rule:** If **Donation Amount > ₹50,000**, send record to **NGO Director for Approval**.

   **Steps:**

   1. Donor pledges ₹75,000.
   2. Record enters approval process.
   3. Director reviews and approves.
   4. On approval → donor gets confirmation + finance team is notified.

5. **Flow Builder**

A. **Record-Triggered Flow:**

- Trigger: New donation record.
- Action: Auto-calculate **Total Donation Value (if recurring donations)**.
- Project Example: ₹2000/month pledge × 12 months = ₹24,000 auto-filled.

B. **Screen Flow (Form for Volunteers/Staff):**

- Build a guided form to log donations or volunteer sign-ups.
- Fields: Donor Name, Contact, Donation Amount, Donation Type (one-time/recurring).
- Project Example: NGO staff can quickly add donation entries during field events.

6. **Email Alerts**

**Rule:** After Director approval → send donor an email.

**Content Example:**

> Subject: *Thank You for Your Support!*
> Body: *Dear [Donor Name], we are grateful for your generous donation of ₹75,000. Your support helps us continue our mission.*

7. **Field Updates**

- ❖ **Business Need:** System updates record status automatically.
- ❖ **Rule:** After approval → **Donation Status = Confirmed**.
- ❖ **Project Example:** Reduces manual effort for staff.

8. **Tasks**
Assign follow-up activities for NGO staff.

- ✓ **Automation:** After donation approval → create Task for Donor Relations Officer:
- ✓ **Subject:** "Send Thank-You Letter"

- ✓ **Due Date:** Within 2 days of approval
- ✓ **Assigned To:** Staff responsible for donor engagement
- ✓ **Project Example:** Ensures every donor gets acknowledged.

9. **Custom Notifications**
   Keep NGO team updated instantly.

   **Automation:** After approval → send in-app notification:

   - Message: *“Donation #D123 approved. Follow up with donor immediately.”*

   **Project Example:** Team sees notification in Salesforce without waiting for email.

# Phase 5: Apex Programming (Developer)

## 1. Classes & Objects

- **Use Case:** Create a **DonationService** class to handle reusable donation logic.
- **Example:** Method to calculate total donation value from recurring pledges.

- .

```
public class DonationService {
    public static Decimal calculateTotal(Decimal amount, Integer months) {
        return amount * months;
    }
}
```

## 2. Apex Triggers

- **Use Case:** On **Donation Insert** → prevent duplicate donations for the same donor on the same date.
- **Logic:** If donor already made a donation today → throw error.

## 3. Trigger Design Pattern

- **Best Practice:** Use a **DonationTriggerHandler** class.
- **Reason:** Keep trigger code clean → delegate logic to handler.

## 4. SOQL & SOSL

- **Use Case:** Query available volunteers.
- **SOQL Example:**

```
List<Volunteer__c> availableVolunteers = [SELECT Id, Name FROM Volunteer__c WHERE Status__c = 'Available'];
```

- **SOSL Example:** Search donor by phone/email.

## 5. Collections (List, Set, Map)

- **Use Case:** Store multiple **Donor IDs** in a Set to avoid duplicate processing.
- **Example:** If 3 records of same donor come in bulk → Set ensures only 1 is processed.

## 6. Control Statements

- **Use Case:** IF donor has already pledged for the same campaign → block entry.
- **Logic:** Throw error *“This donor already pledged for this campaign.”*

## 7. Batch Apex

- **Use Case:** Night job to mark **pledges overdue** if donations not received by due date.
- **Example:** Every night → run batch job to update status of overdue pledges.

## 8. Queueable Apex

- **Use Case:** Process **bulk donation receipts** asynchronously.
- **Example:** If 500 donors pledge during an event, queue job to send discount coupons / thank-you notes without blocking main transaction.

## 9. Scheduled Apex

- **Use Case:** Every morning → email NGO Director a list of **today’s donation events** or **volunteer assignments**.
- **Example:** At 7:00 AM → “10 Volunteers assigned for Health Camp today.”

## 10. Future Methods

- **Use Case:** Call external **payment gateway API / CSR portal** asynchronously to confirm donation status.
- **Example:** @future(callout=true) → send donor info to partner’s system.

## 11. Exception Handling

- **Use Case:** Catch errors if donation duplicate is attempted.
- **Example:**

```
try {
  // insert donation
} catch (DmlException e) {
  System.debug('Error: ' + e.getMessage());
}
```

## 12. Test Classes

- **Use Case:** Create mock donor and donation records → insert donation → assert trigger prevents duplicate donations.
- **Requirement:** Salesforce requires **75% code coverage** for deployment.

## 13. Asynchronous Processing

- **Batch Apex:** For large donation updates (night jobs).
- **Queueable Apex:** For real-time but async processing (sending receipts).
- **Future Methods:** For external API callouts (insurance/payment gateways).
- **Project Context:** Together they handle NGO bulk data + external integrations smoothly.

# Phase 6: User Interface Development

## 1. Lightning App Builder

- **App Name:** *"NGO Donor & Volunteer CRM"*
- **Purpose:** Provides one workspace for Donor Management, Campaigns, and Volunteer tracking.

## 2. Record Pages

- **Donation Record Page:** Show related Donor details and Payment History.
- **Campaign Record Page:** Display pledged donations, assigned volunteers, and campaign updates.

## 3. Tabs

- Add dedicated tabs for:
  - **Donors**
  - **Donations**
  - **Campaigns**
  - **Volunteers**
- Makes navigation simple for NGO staff.

## 4. Home Page Layouts

- Display key dashboards:
  - *Total Donations this Month*
  - *Active Campaigns*
  - *Upcoming Volunteer Events*

## 5. Utility Bar

- Add quick "New Donation" action.
- Field staff can log donations during field campaigns without leaving their current page.

## 6. LWC (Lightning Web Component)

## 7. Apex with LWC

- **Use Case:** LWC calls an Apex method to create a **new donation record**.
- **Benefit:** Faster and more dynamic than standard UI.

## 9. Wire Adapters

- **Usage:** Fetch live donor or volunteer data.
- **Example:** @wire adapter auto-refreshes list of “Active Donors” whenever a new record is added.

## 10. Imperative Apex Calls

- **Use Case:** On *“Pledge Now”* button click → Apex runs to insert Donation.
- **Why Imperative:** Runs only on demand, not pre-fetched.

## 11. Navigation Service

- **Flow:** After new donation record created → user is redirected to that donation’s record page.
- **Benefit:** Staff can instantly verify and edit the donation details.