NGO REGISTRATION AND DONATION MANAGEMENT SYSTEM PROJECT REPORT

By: Shivansh Yadav & Krrish Agarwal
ECE

1. OVERVIEW

This system is a web application for managing NGO member registration and donation transactions. It supports three user roles:

- Supporter: Can register, login, view dashboard, and donate.
- Admin: Has all supporter capabilities plus access to admin dashboard (statistics, user list, donation list, insights, CSV export). Admin accounts require approval by Superadmin before they can login.
- Superadmin: Has all admin capabilities plus the ability to approve or reject admin registrations.

Technology Stack:

- Frontend: React 18 + Vite, React Router, Axios, Recharts
- Backend: Node.js + Express, Mongoose (MongoDB)
- Database: MongoDB (local or Atlas)

2. SYSTEM ARCHITECTURE

Flow:

1. User opens frontend (localhost:5173).
2. Frontend sends API requests to backend (localhost:5000).
3. Backend validates, processes and returns data.
4. MongoDB stores Members, Donations, AuditLogs.

3. DATABASE SCHEMA

Collection: members

Field	Type	Description
_id	ObjectId	Primary key
fullName	String	User's full name
officialEmail	String	Unique, lowercase email
secretHash	String	Bcrypt hashed password
userRole	String	"supporter" | "admin" | "superadmin"
isApproved	Boolean	For admins: false until superadmin approves
registeredAt	Date	Registration timestamp (default: now)

Collection: donations

Field	Type	Description
_id	ObjectId	Primary key
donatedBy	ObjectId	Reference to members._id
amountPaid	Number	Donation amount in INR

orderRef	String	Unique order reference ID
paymentStatus	String	"pending" | "success" | "failed"
transactionTime	Date	Timestamp of transaction

Collection: auditlogs

Field	Type	Description
_id	ObjectId	Primary key
adminId	ObjectId	Reference to admin who performed action
actionPerformed	String	Description of action
timestamp	Date	When the action occurred

4. FLOW DIAGRAMS

4.1 User Registration Flow

1. User fills out registration form (name, email, password, role).
2. Frontend sends POST request to /auth/signup with form data.
3. Backend validates input and hashes password with bcrypt.
4. Backend creates new Member document in MongoDB.
5. If role is "admin", isApproved is set to false (requires superadmin approval).
6. Backend returns 201 Created with success message.
7. Frontend displays success and switches to login mode.

4.2 Login Flow

1. User enters email and password on login form.
2. Frontend sends POST request to /auth/signin.
3. Backend finds member by email in database.
4. Backend compares password hash using bcrypt.
5. If role is "admin", backend checks if isApproved is true.
6. If unapproved admin, backend returns 403 Forbidden.
7. Backend generates JWT token with user ID and role.
8. Backend sets httpOnly cookie with JWT token.
9. Backend returns success with user name and role.
10. Frontend redirects based on role:
    - superadmin -> /superadmin
    - admin -> /admin
    - supporter -> /dashboard

4.3 Donation Flow

1. User navigates to /donate page.
2. User selects preset amount or enters custom amount.
3. User clicks "Proceed to Donate".
4. Frontend sends POST request to /finance/create-order with amount.
5. Backend creates Donation document with status "pending".
6. Backend returns order details (ID, amount, currency).

7. Frontend displays payment simulation screen.
8. User clicks "Simulate Success" or "Simulate Failure".
9. Frontend sends POST request to /finance/update-status.
10. Backend updates donation status to "success" or "failed".
11. Frontend displays confirmation or failure message.
12. User can view donation in dashboard history.

4.4 Admin Approval Flow (Superadmin)

1. Superadmin logs in and is redirected to /superadmin.
2. Superadmin clicks on "Pending Admins" tab.
3. Frontend sends GET request to /admin-portal/pending-admins.
4. Backend queries members with role="admin" and isApproved=false.
5. Backend returns list of pending admin accounts.
6. Frontend displays pending admins in a table.
7. Superadmin clicks "Approve" button for an admin.
8. Frontend sends POST request to /admin-portal/approve-admin.
9. Backend sets isApproved=true for that admin.
10. Backend logs action in AuditLog collection.
11. Frontend removes admin from pending list.
12. Approved admin can now log in successfully.

5. KEY DESIGN DECISIONS

1. Role-Based Access Control
   - Three roles: supporter, admin, superadmin.
   - Middleware (gatekeeper.js) verifies JWT and checks permitted roles.

2. Admin Approval Workflow
   - Admins sign up but cannot log in until superadmin approves.
   - Prevents unauthorized admin access.

3. Superadmin Seeding
   - On database init, a default superadmin is created if not exists.
   - Credentials configurable via environment variables.

4. Cookie-Based JWT
   - JWT stored in httpOnly cookie for security (no XSS exposure).
   - 5-hour expiration; user must re-login after.

5. Sandbox Payment Gateway
   - Donations simulate success/failure for testing.
   - In production, integrate real payment provider (Razorpay, Stripe).

6. CSV Export
   - Admin can export supporter list.
   - Admin can export donation totals grouped by donor.

7. Filtering
   - Donations can be filtered by status (success, pending, failed).
   - Users can be searched by name.

6. ASSUMPTIONS

- MongoDB is running locally or a valid CONNECTION_URI is provided.
- Only one superadmin is needed; additional can be added manually.
- Email uniqueness is enforced at database level.
- Passwords are hashed with bcrypt .
- Frontend and backend run on separate ports during development.

7. DEFAULT SUPERADMIN CREDENTIALS

Email:	superadmin@example.com
Password:	SuperAdmin123!