Project Report

NGO Donation Management System

---

1. Introduction

The NGO Donation Management System is a web-based application developed to simplify and digitize the fundraising operations of Non-Governmental Organizations (NGOs). The system provides a centralized platform where NGOs can manage fundraising campaigns while donors can easily browse campaigns, register, and make donations.

Built using the Django web framework, the system emphasizes transparency, secure data handling, and scalability. By incorporating role-based access, the platform ensures that administrative functionalities such as campaign creation and management are restricted to authorized users, while donors are provided with a simple and intuitive interface for participation. The project aims to enhance efficiency, accountability, and donor engagement in NGO operations.

---

2. System Architecture

The system is designed using a three-tier architecture, which promotes separation of concerns, modularity, and scalability.

2.1 High-Level Architecture

Presentation Layer

The presentation layer is responsible for user interaction. It consists of web pages created using HTML templates rendered through Django Template Language (DTL). Static resources such as CSS and images are managed using Django's static file handling mechanism. This layer enables users to view campaigns, register, and make donations.

Application Layer

The application layer contains the core business logic of the system. It is implemented using Django views, which handle user authentication, campaign management, user

registrations, and donation processing. This layer acts as a bridge between the user interface and the database.

Data Layer

The data layer manages persistent storage and data retrieval. It uses the Django ORM to interact with a relational database. SQLite is used during development for simplicity, while the system is designed to support MySQL in production environments for better performance and scalability.

2.2 Django MVT Architecture

The system follows Django's Model–View–Template (MVT) architecture:

- **Model: Defines the database schema and manages data operations.**
- **View: Implements application logic and processes user requests.**
- **Template: Displays dynamic content to users through rendered HTML pages.**

2.3 Modular Application Design

To improve maintainability and scalability, the system is divided into modular Django apps:

- **accounts: Handles user authentication and role-based access**
- **campaigns: Manages creation and display of fundraising campaigns**
- **registrations: Handles user registration for campaigns**
- **donations: Manages donation processing and transaction history**
- **pages: Displays static informational pages**

---

3. Database Schema

The database design ensures strong data integrity and efficient relationships among entities. The system consists of four primary models.

3.1 User Model

The User model extends Django's `AbstractUser` to represent all users, including administrators and donors.

Key Fields:
`id`, `username`, `email`, `password`, `is_staff`, `is_superuser`

Constraints:
Usernames and email addresses are unique.

3.2 Campaign Model

The Campaign model represents fundraising initiatives created by administrators.

Key Fields:
`id`, `title`, `description`, `goal_amount`, `start_date`, `end_date`, `is_active`, `image`

Business Rule:
A campaign cannot be deleted if registrations are associated with it.

3.3 Registration Model

The Registration model links users to campaigns.

Key Fields:
`id`, `user_id`, `campaign_id`, `name`, `created_at`

Constraint:
Each user can register only once per campaign.

3.4 Donation Model

The Donation model stores transaction details related to campaign registrations.

Key Fields:
`id`, `registration_id`, `amount`, `payment_status`, `payment_provider`, `transaction_id`, `created_at`

Design Consideration:
Donation records are preserved even if the corresponding registration is removed, ensuring financial data integrity.

---

4. System Flow Diagrams

4.1 User Registration and Login Flow

```
User → Register / Login → Authentication → Role Assignment → Dashboard
```

4.2 Campaign Creation Flow

```
Admin Dashboard → Campaign Creation Form → Validation → Campaign
Stored
```

4.3 Donation Process Flow

```
Donor Login → Browse Campaigns → Register for Campaign → Donate

→ Donation Record Saved → Campaign Total Updated
```

4.4 Donation Sequence Flow

```
Donor → Campaign Page → Registration → Donation Form

View → Donation Model → Database
```

```
Database → Confirmation Page
```

---

5. Key Design Decisions

5.1 Django Framework Selection

Django was chosen for its rapid development capabilities, built-in authentication system, and strong security features such as CSRF and XSS protection.

5.2 Role-Based Access Control

Separate roles for administrators and donors ensure secure access to sensitive functionalities like campaign management.

5.3 Campaign-Centric Donation Design

Each donation is explicitly linked to a campaign, allowing transparent tracking and accurate financial reporting.

5.4 Media Handling

Campaign images are managed using Django's `MEDIA_ROOT`, enhancing donor engagement and system organization.

---

6. Assumptions

- **Database: SQLite is used for development; MySQL is intended for production.**
- **Payment Processing: The system uses simulated payment logic and does not process real-time banking transactions.**
- **Security: Django's default middleware provides sufficient baseline security.**
- **Scalability: The architecture supports future expansion and increased user traffic.**

---

7. Conclusion

The NGO Donation Management System provides a structured and scalable solution for managing fundraising campaigns and donations. Through a modular Django architecture, well-defined database design, and clear system workflows, the platform addresses key operational challenges faced by NGOs. The system is flexible enough to accommodate future enhancements such as real payment gateway integration and advanced reporting features.