PROJECT REPORT: NGO Registration and Donation Management System (UnityFund)

Submitted by: Ashi Agrawal & Utkarsh Bhardwaj **Date:** January 20, 2026

Video Demonstration:
https://drive.google.com/file/d/1iJeDBrNyFtJXldhYY_2yJjk0-bHM7cDP/view?usp=sharing

---

1. Abstract

The "UnityFund" project is a web-based platform designed to bridge the gap between donors and Non-Governmental Organizations (NGOs). It solves the problem of trust and complexity in charitable donations by providing a transparent, easy-to-use interface. The system features secure user authentication, a dashboard for viewing NGO causes, and a real-time payment gateway (PayHere) integration. The application is built using the MERN stack (MongoDB, Express, React, Node.js) to ensure high performance and scalability.

2. Introduction

2.1 Problem Statement

Current manual methods of donating to NGOs are inefficient. Donors often lack transparency regarding where their money goes, and NGOs struggle with maintaining donor records. Additionally, integrating secure online payments is technically difficult for smaller organizations.

2.2 Proposed Solution

UnityFund provides a centralized digital platform where:

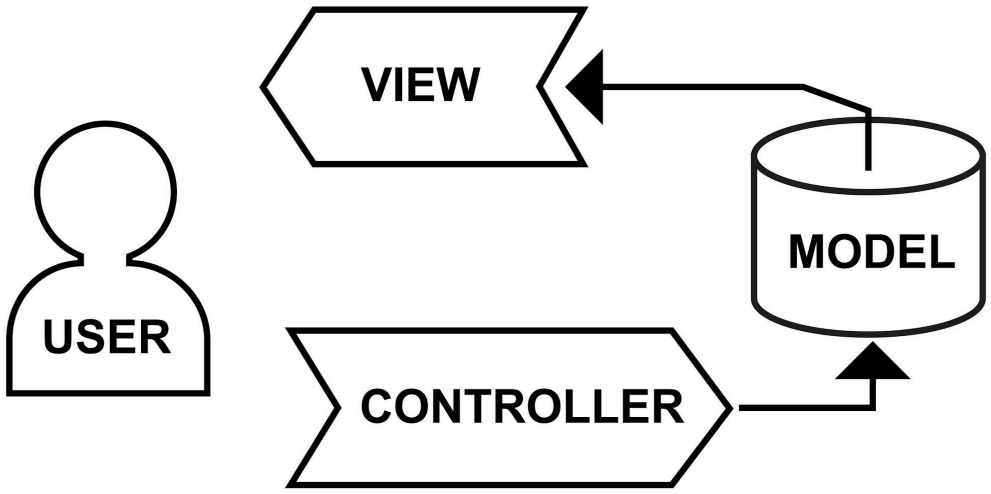
- **Donors** can register, view causes, and donate securely.
- **Admins** can manage users and view total funds raised.
- **Payments** are processed instantly using a secure gateway.
- **Transparency** is maintained through digital transaction records.

3. System Architecture

The project follows the **MVC (Model-View-Controller)** architecture:

- **Frontend (View):** Built with **React.js** and **Tailwind CSS**. It provides a responsive user interface for donors to interact with.
- **Backend (Controller):** Built with **Node.js** and **Express.js**. It handles API requests, authentication, and payment processing logic.

- **Database (Model): MongoDB Atlas** (Cloud) is used to store user data and donation history securely.



MODEL - VIEW - CONTROLLER PATTERN

4. Technology Stack

- **Frontend:** React.js, Vite, Tailwind CSS
- **Backend:** Node.js, Express.js
- **Database:** MongoDB Atlas
- **Payment Gateway:** PayHere (Sandbox Environment)
- **Tunneling Tool:** ngrok (Required for connecting local backend to public payment gateway)

5. Modules Description

1. **Authentication Module:** Secure Sign-up and Login using JWT (JSON Web Tokens).
2. **Donation Module:** Allows users to select donation amounts and currency (LKR).
3. **Payment Module:** Integrates PayHere API. It uses a secure checksum hash to verify transactions and prevent fraud.
4. **Admin Dashboard:** A special protected area where Administrators can see a list of all users and donation statistics.

6. User Manual (Installation & Setup Guide)

This section describes the steps required to execute the UnityFund application on a new machine. Since sensitive configuration files (such as .env) and dependency folders (node_modules) are not included in the source code distribution, they must be set up manually by following the steps below.

6.1 Prerequisites Ensure the following software is installed on the target machine:

- **Node.js:** Version 16 or higher.
- **Git:** For version control.
- **ngrok:** Required to expose the local server to the payment gateway.
- **Terminal:** Ability to open 3 separate terminal windows simultaneously.

6.2 Installation Steps

Step 1: Transfer or Clone the Code Copy the project folder to your local machine or clone the repository using Git.

Step 2: Install Dependencies The project relies on external libraries that must be installed fresh. Open a terminal and run the following commands for both the backend and frontend:

- **For Backend:** cd backend npm install
- **For Frontend:** cd ../frontend npm install

6.3 System Configuration (Critical)

The application requires environment variables to connect to the database and payment gateway.

Step 1: Create Environment File Navigate to the **backend** folder and create a new file named **.env**.

Step 2: Add Credentials Copy and paste the following configuration into the .env file.

(Note: You must obtain the actual connection strings and API keys from the project owner).

PORT=5000
MONGO_URI=mongodb+srv://<username>:<password>@cluster.mongodb.net/UnityFund
JWT_SECRET=your_secure_jwt_secret PAYHERE_MERCHANT_ID=your_merchant_id
PAYHERE_SECRET=your_payhere_secret NODE_ENV=development

Step 3: Database Access Ensure the MongoDB Atlas Network Access is set to **"Allow Access from Anywhere" (0.0.0.0/0)**. This ensures the database does not block the new machine's IP address.

6.4 Execution Workflow (How to Run)

To run the full system, you must use **three separate terminal windows**.

Terminal 1: Start the Backend This starts the API server on port 5000.

- Command: cd backend npm run dev

Terminal 2: Start Payment Tunneling This exposes the local backend to the public internet so PayHere can send payment receipts.

- Command: ngrok http 5000
- **Action:** Look for the "Forwarding" line on the screen and **copy the HTTPS URL** (e.g., https://random-id.ngrok-free.app).

Terminal 3: Start the Frontend Before running the frontend, you must link it to the tunnel created in Terminal 2.

1. Open the file: **frontend/src/pages/DonationPage.jsx**
2. Find the variable **notify_url** and paste the new ngrok URL you just copied.
3. Start the client application with the command: cd frontend npm run dev

The application will now be accessible in your browser at: **http://localhost:5173**

6.5 Admin Access Guide

To access Admin features, the user role must be updated directly in the database:

1. Register a new user account on the website.
2. Open your MongoDB Atlas database and find the **users** collection.
3. Locate the new user and change the **role** field from "user" to **"admin"**.
4. Log out and log back in to the website to access the Admin Dashboard.

7. Testing Results

7.1 Payment Integration Test

- **Objective:** Verify that the database updates after a successful payment.
- **Method:** Used PayHere Sandbox credentials to make a dummy payment.
- **Result:** The `notify_url` (via ngrok) successfully received the callback, and the user's donation status changed from "Pending" to "Success" in the database.

7.2 Security Testing

- **Authentication:** Confirmed that users cannot access the Admin Dashboard without valid Admin credentials.
- **CORS Policy:** Verified that the backend only accepts requests from authorized frontend origins.

8. Project Workflow

1. User registers or logs in
2. User selects donation amount
3. Payment gateway processes transaction
4. Payment status is verified

5. Donation data is stored in database
6. Admin can view donation records

9. Database Design

Main Collections:

- **Users** – Stores user details and roles
- **Donations** – Stores donation amount, user, and status

Each record is uniquely identified and securely stored.

10. Security Features

- JWT-based authentication
- Encrypted environment variables
- Secure payment verification
- Role-based access control
- Protected admin routes

11. Future Enhancements

- Email confirmation after donation
- NGO verification system
- Donation analytics and reports
- Mobile application
- Multiple payment gateway support
- Multi-language support

12. Conclusion

UnityFund successfully provides a reliable and transparent donation management system. By integrating modern web technologies and a secure payment gateway, the platform ensures trust, efficiency, and ease of use for both donors and NGOs.

The project demonstrates real-world implementation of full-stack development concepts and serves as a strong foundation for future enhancements.