A Project Report on

‘SPREAD THE SMILE- DONATION PLATFORM’

By

**Ms. Banshika Gupta Ms. Jyoti Choudhary**

(240310120004) (240310120017)

**Mr. Uday Walia**

(240310120014)

Guide

# Mr. Prashant Kumar

Project Engineer, CDAC Delhi



A Project Report Submitted at

**Centre for Development of Advanced Computing**

In partial fulfillment of the requirements of

**PG Diploma in Advanced Computing [March 2024-August 2024]**

**Centre for Development of Advanced Computing ACT’S, Delhi**

****

**CERTIFICATE**

This is to certify that the project entitled “**SPREAD THE SMILE-DONATION PLATFORM**” is a bonafide work of “**Banshika Gupta (240310120004), Jyoti Choudhary (240310120017)” Uday Walia (240310120014)** submitted to C- DAC Delhi in partial fulfillment of the requirement for the award of the Post Graduate Diploma in Advanced Computing.

Mr. Prashant Kumar ----------------------------

## Guide Project Coordinator

**ACKNOWLEDGEMENT**

Presentation inspiration and motivation have always played a key role in the success of any venture. We express our sincere thanks to Mr. Ankit Khurana, Center for Development of Advance Computing (C-DAC), Delhi.

We pay our deep sense of gratitude to Mr. -----, Project coordinator, C-DAC, Delhi to encourage us to the highest peak and to provide us the opportunity to prepare the project. We are immensely obliged to our colleague for their elevating inspiration, encouraging guidance and kind supervision in the completion of our project.

We feel to acknowledge our indebtedness and deep sense of gratitude to our guide Mr. Prashant Kumar whose valuable guidance and kind supervision given to us.

# CONTENTS

[Abstract 1](#_TOC_250022)

1. [Introduction](#_TOC_250021)
   1. [Purpose 2](#_TOC_250020)
   2. [Scope 2](#_TOC_250019)
   3. Problems Domain 3
   4. [Solution Domain 3](#_TOC_250018)
   5. System Requirement 3
2. [System Requirement Analysis](#_TOC_250017)
   1. Information Gathering……………………………………………………………….5
   2. System Feasibility……………………………………………………………………6
3. [System Analysis](#_TOC_250014)
   1. Use Case Diagram …7
   2. Activity Diagram 9
4. Design
   1. Data Modeling………………………………………………………………………11
      1. ER Diagram/Object Diagram 11

4.1.2 Flow Chart…………………………………………………………………….12

4.1.3 Database……………………………………………………………………….13

1. Implementation

5.1 Frontend code……………………………………………………………………16

* 1. Backend Code……………………………………………………………………26

5.3 Result Analysis…………………………………………………………………..33

1. Testing
   1. Testability………………………………………………………………………..…..46
      1. Observability…………………………………………………………….….46
      2. Decomposibility………………………………………………….….….…..46
      3. Stability…………………………………………………….…………….….46
   2. Testing Method Case…………………………………………………………………46
2. Future Work 49
3. Conclusion 49
4. Appendix 50
5. References 50
   * 1. **Database**

# LIST OF TABLES

* + - 1. Admin 13
      2. Donor 13
      3. Donation Details 14
      4. Recipient 14
      5. Volunteer 15

# ABSTRACT

The **Spread the Smile: Donation Platform** is designed to revolutionize charitable giving in India by addressing the current fragmentation in the donation process. NGOs, donors, and recipients often operate in isolation, leading to inefficiencies and missed opportunities. Spread the Smile aims to unify these efforts by providing a centralized, standardized platform that connects all parties seamlessly.

For donors, the platform offers guidance throughout the donation process, helping them make informed decisions about where their contributions will have the most impact. By providing insights into verified needs and ongoing projects, the platform ensures that donations are meaningful and effective.

On the recipient side, the platform verifies demands and coordinates with volunteers to ensure that aid is delivered quickly and accurately. This not only speeds up the process but also enhances the credibility and trustworthiness of the donation system.

Ultimately, Spread the Smile seeks to bring greater transparency, efficiency, and collaboration to the charitable sector, ensuring that resources are directed where they are most needed and making a significant difference in the lives of those in need.

# INTRODUCTION

The ‘**Spread the Smile: Donation Platform’** aims to transform charitable giving in India by bridging the gaps between donors, NGOs, and recipients. Addressing the inefficiencies caused by fragmented efforts, this platform offers a unified solution that guides donors through a transparent and informed donation process, while ensuring that verified recipients receive aid swiftly through coordinated volunteers. By enhancing collaboration and streamlining operations, Spread the Smile seeks to create a more effective and trustworthy donation system, ensuring that resources are utilized where they are needed most.

## Purpose

The purpose of the ‘**Spread the Smile: Donation Platform’** is to create a unified, efficient system for charitable giving in India. It aims to connect donors, NGOs, and recipients through a single, standardized platform, eliminating fragmentation and improving coordination. By guiding donors in making impactful contributions and ensuring that verified recipients receive aid swiftly through volunteers, the platform seeks to enhance the overall effectiveness, transparency, and credibility of the donation process.

## Scope

The scope of the Spread the Smile: Donation Platform includes:

1. **Integration of Stakeholders:** Bringing together donors, NGOs, and recipients into a single, cohesive system to facilitate efficient interactions and transactions.
2. **Donation Process Management:** Providing tools and resources for donors to make informed decisions, track their contributions, and ensure their impact.
3. **Recipient Verification:** Implementing processes to verify the needs of recipients and ensure that aid is directed to those who genuinely require it.
4. **Volunteer Coordination:** Organizing and managing volunteers to ensure the prompt and accurate delivery of donations.
5. **Transparency and Accountability:** Offering real-time tracking and updates to enhance trust and credibility in the donation process.
6. **Data Analysis and Reporting:** Collecting and analyzing data to assess the effectiveness of donations, identify areas for improvement, and report on the impact of contributions.
7. **Scalability and Adaptability:** Designing the platform to accommodate growth and adapt to changing needs within the charitable sector.

## Problem Domain

The **Spread the Smile: Donation Platform** addresses several critical issues in the charitable sector in India. Currently, the sector suffers from fragmentation, with NGOs, donors, and recipients operating in isolation, leading to inefficiencies and a lack of coordination. There is also a lack of standardization, resulting in inconsistent donation processes and difficulties in tracking contributions. Donors often face uncertainty about where to give and how their donations will be utilized, impacting their confidence in the process. Additionally, the absence of reliable verification systems for recipient needs can lead to misallocation or delays in aid. Inefficient aid delivery further exacerbates these problems, while limited transparency reduces accountability and trust. Spread the Smile aims to solve these challenges by offering a unified platform that enhances collaboration, provides clear guidance for donors, verifies recipient needs, and ensures prompt and accurate aid delivery, thereby creating a more effective and trustworthy charitable ecosystem..

## Solution Domain

It offers a comprehensive solution to the inefficiencies and challenges faced in India’s charitable sector. By integrating donors, NGOs, and recipients into a single, standardized system, the platform eliminates fragmentation and enhances coordination. It standardizes the donation process, providing a consistent and transparent framework for tracking and managing contributions, which simplifies operations and improves accountability. The platform also guides donors with detailed information about recipient needs and projects, increasing donor confidence and engagement. It includes robust mechanisms for verifying recipient needs, ensuring that aid is directed accurately and efficiently. Through coordination with volunteers, it facilitates swift and precise delivery of donations. Additionally, real-time tracking and updates enhance transparency, building trust among all parties involved. Overall, Spread the Smile creates a more effective, transparent, and reliable donation system, addressing existing inefficiencies and maximizing the impact of charitable efforts.

## System Requirements

* + 1. **Hardware Requirements**
       - Processor: i3 and above
       - RAM: 8GB

## Software Requirements

* + - * Client on Internet: Web Browser, Operating System (any).
      * Internet Connection: (any).
      * Data Base : MySQL Workbench
      * Development End: Spring Boot
      * Designing End: React
      * Tools : MySQL Workbench 8.0 CE, VS Code, Spring Tool Site4Browse

# SYSTEM REQUIREMENT ANALYSIS

## Information Gathering

For the **Spread the Smile: Donation Platform**, comprehensive information gathering is essential to develop a solution that effectively meets the needs of all users and stakeholders. This process involves collecting detailed requirements from donors, NGOs, recipients, and volunteers. Donors’ preferences for donation types and communication, NGOs’ needs for support and project details, recipients’ specific needs and verification documents, and volunteers’ availability and task preferences must all be documented.

## Functional Requirements

Functional requirements are those which specify what a system should do.

* + - * Donor, Recipient and Volunteer can Register
      * Donor, Recipient ,Volunteer and Admin can Login
      * Donor can submit the available resources like cloths, food to admin.
      * Recipient can request the resources like cloths, food to admin .
      * Volunteer can help in transportation of the resources assigned by admin.
      * Amin received the resources from recipient and send to the donor by the help of volunteer.

## Non-functional Requirements

Those requirements which are not the functionalities of a system but are the characteristics of the system are called non-functional requirements.

* + - * Secure access of confidential data with the use of encryptions.
      * Full time available.
      * Data Consistency.
      * Data Reliability.
      * Better component design to get better performance at peak time.
      * Flexible service based on architecture will be highly desirable for future extensions.

## System Feasibility

* + 1. **Technical Analysis**
       - It is the study of configuration of the system. It considers the technical requirements of the project.
       - We have used Reactjs, MySQL workbench as the developer tool and spring boot components to develop the project.
       - For storage of data in databases we have used MySQL.

## Economical Analysis

It could also be referred to as cost/benefit analysis. It is used for evaluating the effectiveness of a system. In economic analysis the procedure is to determine the benefits and savings that are expected from a system and compare them with costs.

## Operational Feasibility

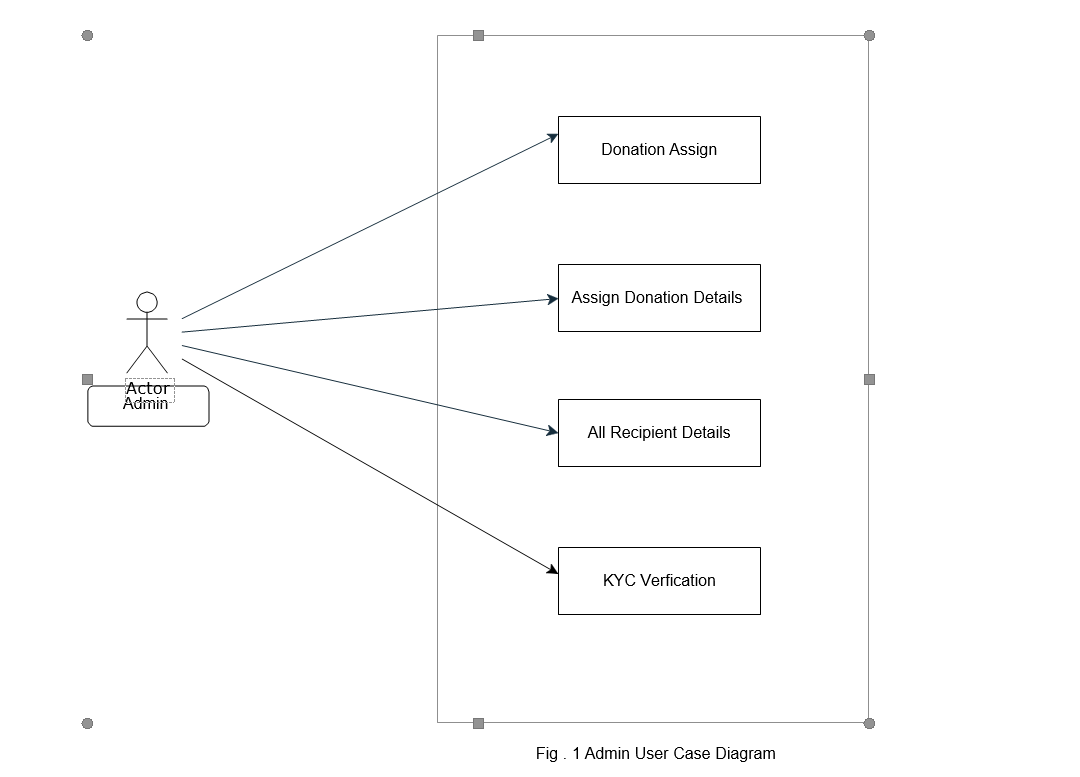
It is dependent on human resources available for the project. This application will be able to handle the all Functionality efficiently. This application will save time and provide an all-time availability to the users. It is secure, time saving and allows users to access the website even from remote locations.

* 1. **Use-case Model**

# SYSTEM ANALYSIS

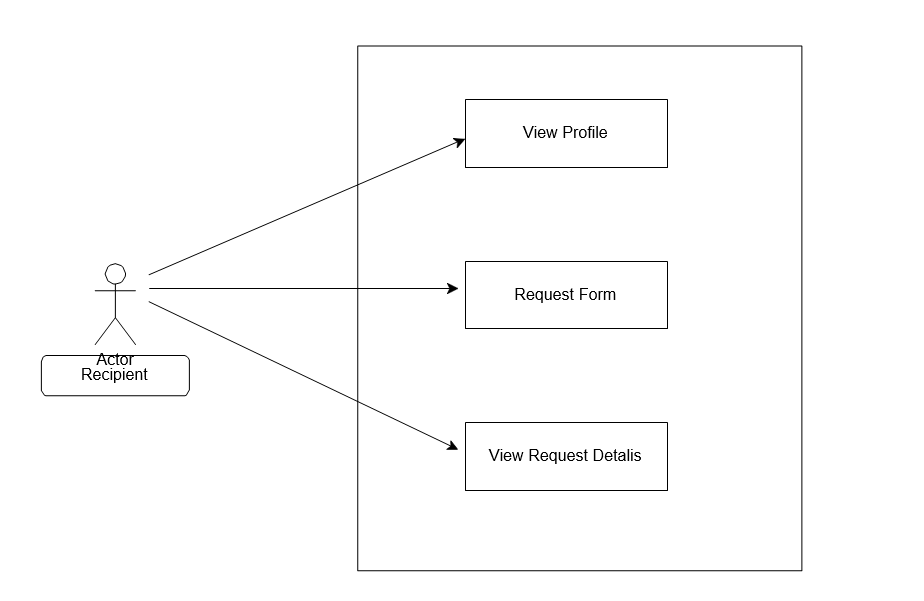
Use Cases are used here to describe the most important sequence of scenarios between the User and the System.

## Admin Use Case:



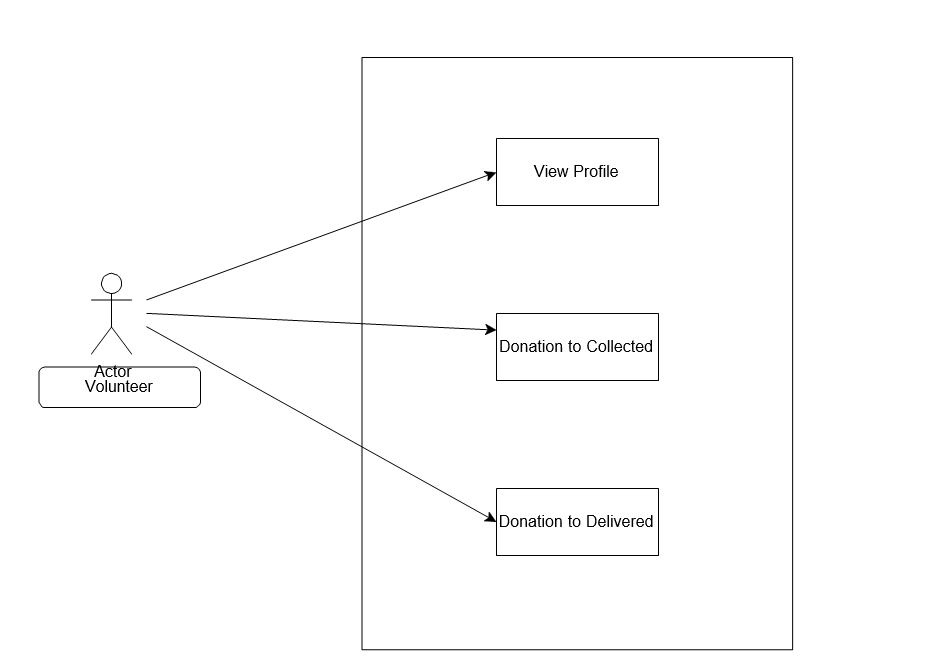
**Fig.3.1.1 Admin Use Case**

## Recipient Use Case:

****

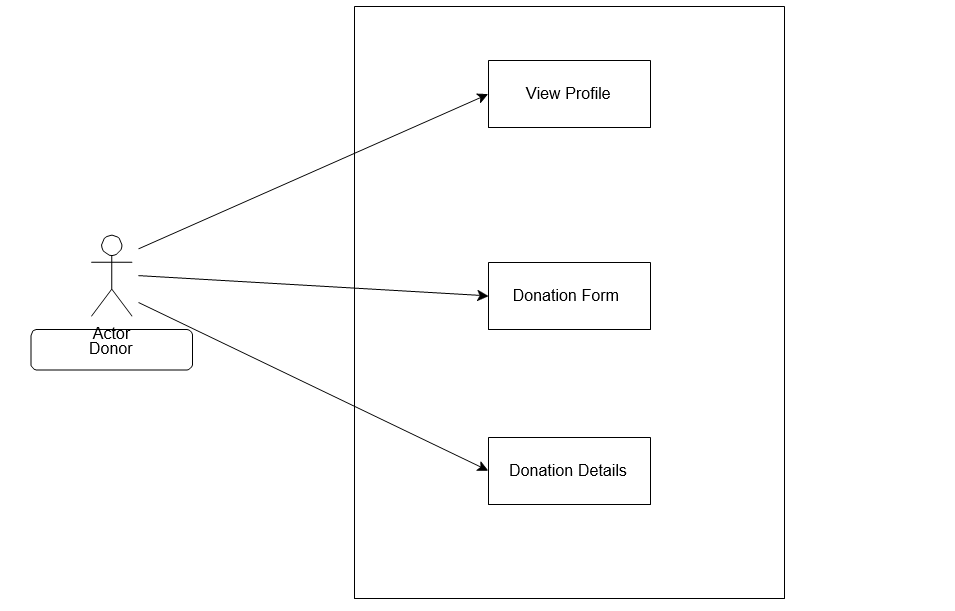
**Fig.3.1.2 Recipient Use Case**

**3.1.3 Volunteer Use Case**



**Fig.3.1.3 Volunteer Use Case**

* + 1. **Donor Use Case**

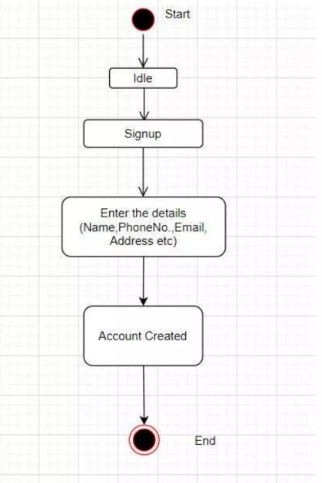
****

**Fig.3.1.4 Donor Use Case**

## Activity Diagrams:

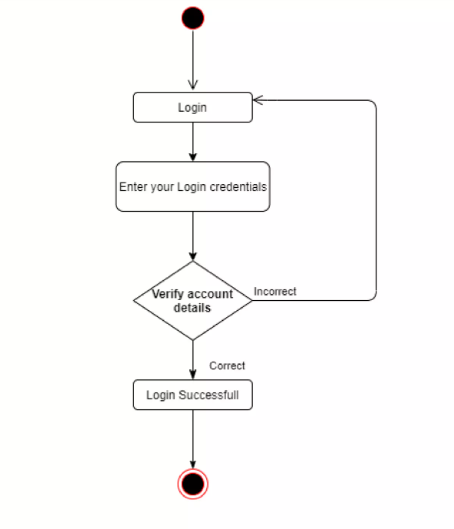
Activity diagram is another important behavioral diagram in UML diagram to describe dynamic aspects of the system. Activity diagram is essentially an advanced version of flow chart that modeling the flow from one activity to another activity.

## Sign Up Activity

****

**Fig.3.3.1 Sign Up Activity**

## Sign in Activity:

****

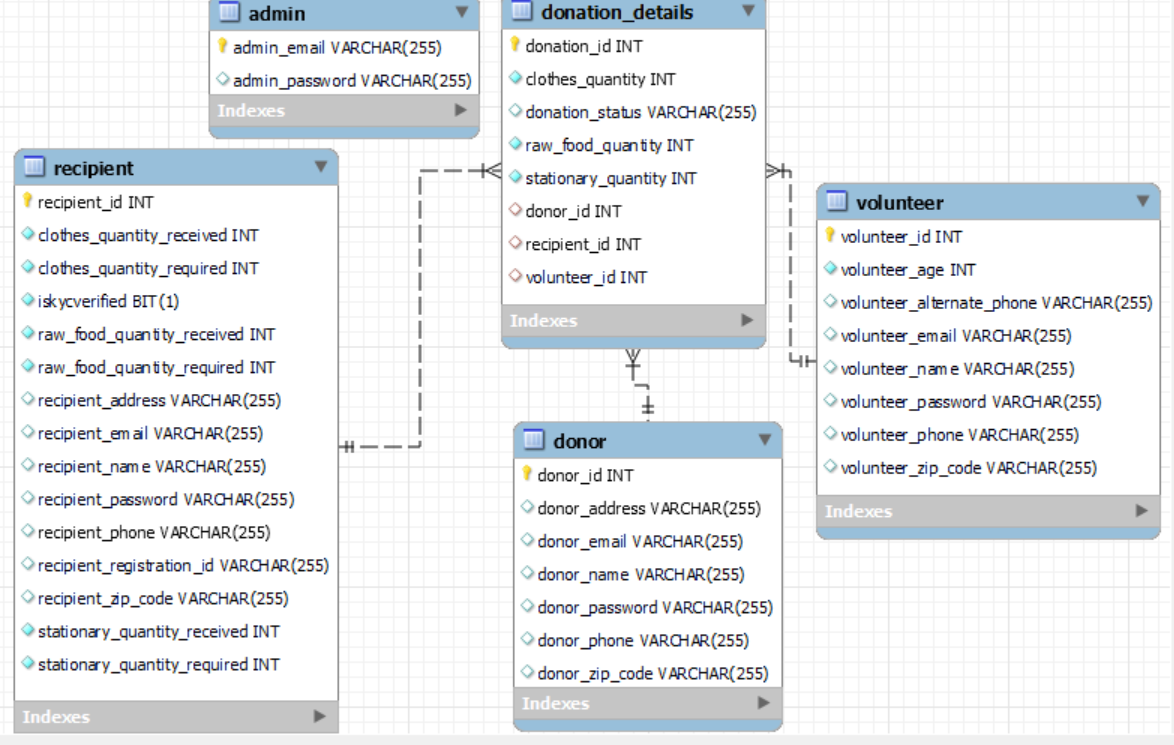
**Fig.3.3.2 Sign in Activity**

## Data Modeling:

# DESIGN

A Conceptual Data Model is an organized view of database concepts and their relationships. The purpose of creating a conceptual data model is to establish entities, their attributes, and relationships.

## ER Diagram:

****

**Fig.4.1.1 ER Diagram:**

## Flow Chart

## 

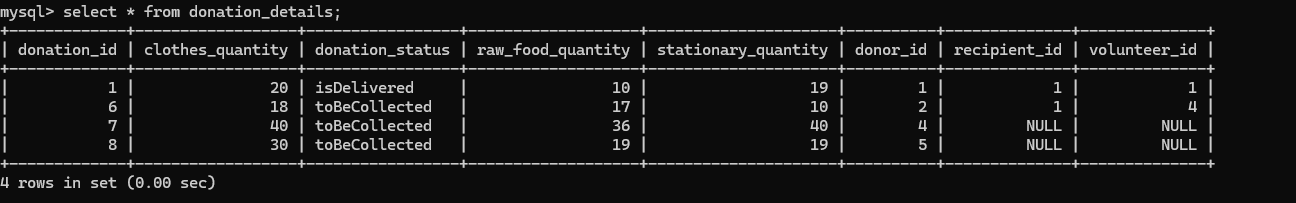
## Database:

|  |  |
| --- | --- |
| Admin\_email | Admin\_password |
| admin@gmail.com | $2a$10$Z1iKCO5hrNs0lHJfwt86oOGARVY03X8ksvICGlQWw2WJLBF0cHh3O |

4.1.2.1 Table for Admin



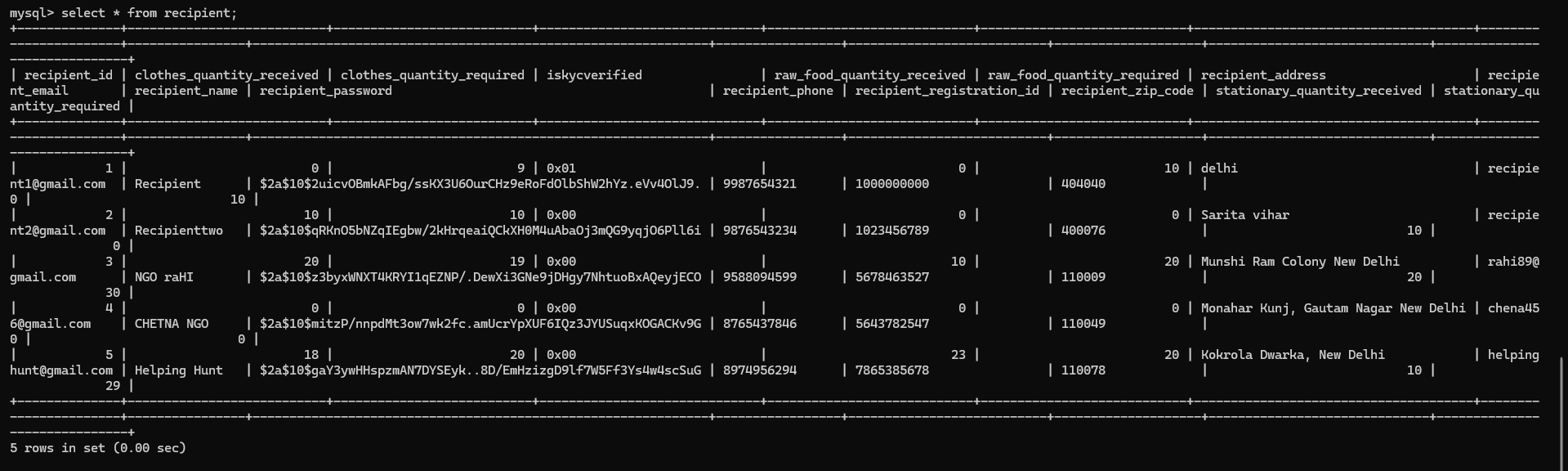
4.1.2.2 Table for Donor



4.1.2.2 Table for Donation Details



4.1.2.4 Table for Volunteer



4.1.2.5 Table for Recipient

# IMPLEMENTATION

## Fronted End Code

## MainHome Page:-

## import { Container, Row, Card, Col } from "react-bootstrap";

## import { NavigationBar } from "./NavigationBar";

## import { CircularProgressbar } from "react-circular-progressbar";

## import "react-circular-progressbar/dist/styles.css";

## import { Link } from "react-router-dom";

## import {

## fetchClothesPercentage,

## fetchRawFoodPercentage,

## fetchStationaryPercentage,

## } from "../services/RecipientApiService";

## import { useEffect, useState } from "react";

## import Fade from 'react-reveal/Fade'; // Importing react-reveal animation

## // Import the image

## import myImage from "../components/images/homeBg.jpg";

## export function MainHome() {

## const [rawFood, setRawFood] = useState(0);

## const [clothes, setClothes] = useState(0);

## const [stationary, setStationary] = useState(0);

## const fetchAllQuantities = async () => {

## let response = await fetchRawFoodPercentage();

## setRawFood(response.data);

## response = await fetchClothesPercentage();

## setClothes(response.data);

## response = await fetchStationaryPercentage();

## setStationary(response.data);

## };

## useEffect(() => {

## fetchAllQuantities();

## }, []);

## return (

## <>

## <NavigationBar />

## <Container className="bg-light text-center mt-0 p-3 rounded">

## <Fade top>

## <h1 className="display-4 text-primary mb-4 font-weight-bold">

## SPREAD THE SMILE-DONATION PLATFORM

## </h1>

## </Fade>

## <Fade top>

## <p className="lead">

## Our mission is to help you make a positive impact on the world by

## enabling you to easily donate to causes you care about.

## </p>

## </Fade>

## <Fade bottom>

## <img src={myImage} alt="A description of the image" />

## </Fade>

## <Container className="mt-4">

## <Row>

## <Col lg={4}>

## <Fade left>

## <Card className="border-0 shadow-lg rounded-lg">

## <div className="p-3 bg-primary text-white">

## <h3 className="text-center mb-0">Food Needs</h3>

## </div>

## <Card.Body className="text-center">

## <div

## style={{

## display: "flex",

## flexDirection: "column",

## alignItems: "center",

## marginTop: 20,

## }}

## >

## <CircularProgressbar value={rawFood} strokeWidth={8} />

## <div

## style={{

## marginTop: 10,

## fontSize: 24,

## fontWeight: "bold",

## color: "blue",

## }}

## >

## {rawFood}% of food needs met

## </div>

## </div>

## <Card.Text className="mt-4">

## Are you aware that 194 million individuals residing in India

## are malnourished and encounter difficulties in obtaining

## healthy and nutritious food? By donating, you can make a

## positive impact on their lives. Your contribution can assist

## us in eliminating hunger and ensuring that no one goes to bed

## without a meal.

## </Card.Text>

## </Card.Body>

## </Card>

## </Fade>

## </Col>

## <Col lg={4}>

## <Fade top>

## <Card className="border-0 shadow-lg rounded-lg">

## <div className="p-3 bg-primary text-white">

## <h3 className="text-center mb-0">Stationery Needs</h3>

## </div>

## <Card.Body className="text-center">

## <div

## style={{

## display: "flex",

## flexDirection: "column",

## alignItems: "center",

## marginTop: 20,

## }}

## >

## <CircularProgressbar value={stationary} strokeWidth={8} />

## <div

## style={{

## marginTop: 10,

## fontSize: 24,

## fontWeight: "bold",

## color: "blue",

## }}

## >

## {stationary}% of stationery needs met

## </div>

## </div>

## <Card.Text className="mt-4">

## A large number of children and adults do not have access to

## basic stationery items such as notebooks, pens, and pencils.

## Your donation can help in providing access to education and

## learning opportunities for those in need. Let's come

## together to create a brighter future for all.

## </Card.Text>

## </Card.Body>

## </Card>

## </Fade>

## </Col>

## <Col lg={4}>

## <Fade right>

## <Card className="border-0 shadow-lg rounded-lg">

## <div className="p-3 bg-primary text-white">

## <h3 className="text-center mb-0">Clothing Needs</h3>

## </div>

## <Card.Body className="text-center">

## <div

## style={{

## display: "flex",

## flexDirection: "column",

## alignItems: "center",

## marginTop: 20,

## }}

## >

## <CircularProgressbar value={clothes} strokeWidth={8} />

## <div

## style={{

## marginTop: 10,

## fontSize: 24,

## fontWeight: "bold",

## color: "blue",

## }}

## >

## {clothes}% of clothing needs met

## </div>

## </div>

## <Card.Text className="mt-4">

## Did you know that approximately 40% of India's population

## lives below the poverty line? Many of these individuals lack

## access to basic necessities like warm clothing. Your

## generous donation to support our cause for clothing drive

## can help provide essential clothing items to those in need.

## </Card.Text>

## </Card.Body>

## </Card>

## </Fade>

## </Col>

## </Row>

## <Fade bottom>

## <Container>

## <Link

## to="/login"

## className="btn btn-primary rounded-pill bg-gradient my-4"

## >

## Donate Now

## </Link>

## </Container>

## </Fade>

## </Container>

## </Container>

## </>

## );

## }

## About US Page:-

## import React from "react";

## import { Container, Row, Col, Image, Card } from "react-bootstrap";

## import { NavigationBar } from "./NavigationBar";

## import Footer from "./Footer";

## import { Fade, Slide } from "react-awesome-reveal"; // Import animations

## const AboutUs = () => {

## return (

## <>

## <NavigationBar></NavigationBar>

## <Container className="bg-light text-center mt-2 p-2 rounded">

## <Fade cascade>

## <h1 className="display-4 text-primary mb-2 font-weight-bold">

## SPREAD THE SMILE-DONATION PLATFORM

## </h1>

## <p className="lead">

## Our mission is to help you make a positive impact on the world by

## enabling you to easily donate to causes you care about.

## </p>

## </Fade>

## </Container>

## <Container className="mt-5 bg-white rounded shadow-lg">

## <Fade>

## <h1 className="text-center mb-5">About Us</h1>

## </Fade>

## <Row className="align-items-center mb-5">

## <Col md={6}>

## <Slide direction="left">

## <Image

## src="https://c0.wallpaperflare.com/preview/287/860/54/team-ethnicity-group-hands.jpg"

## alt="About Us"

## fluid

## />

## </Slide>

## </Col>

## <Col md={6}>

## <Slide direction="right">

## <h2 className="mb-4">SPREAD THE SMILE-DONATION PLATFORM</h2>

## <p>

## In India, we often see NGOs and donors working in silos lacking a

## common standardized platform to handle end-to-end donation

## processes. Spread the Smile-Donation Platform aims to address this critical issue by

## bridging this gap between donors and recipients. In this system,

## donors will be assisted throughout the donation decision-making

## process. Verified recipients’ demands will be fulfilled swiftly

## through volunteers. This platform promises to bring credibility

## and efficiency in the existing system.

## </p>

## </Slide>

## </Col>

## </Row>

## <Row className="justify-content-center">

## <Col md={3}>

## <Fade>

## <Card className="p-3 mb-3 bg-secondary text-white rounded shadow">

## <Card.Body>

## <i className="fas fa-bullseye fa-3x mb-3"></i>

## <Card.Title>Vision</Card.Title>

## <Card.Text>

## The idea is to create self-sustained chapters across the world

## who will look after their local community. And in the process,

## inspire people around us to give back to those who need it

## most.

## </Card.Text>

## </Card.Body>

## </Card>

## </Fade>

## </Col>

## <Col md={3}>

## <Fade delay={200}>

## <Card className="p-3 mb-3 bg-success text-white rounded shadow">

## <Card.Body>

## <i className="fas fa-users fa-3x mb-3"></i>

## <Card.Title>Who We Are</Card.Title>

## <Card.Text>

## The Spread the Smile-Donation Platform is a volunteer-based, zero-funds

## organization that works to get surplus food, stationary,

## clothes from society and the community to serve them to the

## less fortunate people.

## </Card.Text>

## </Card.Body>

## </Card>

## </Fade>

## </Col>

## <Col md={3}>

## <Fade delay={400}>

## <Card className="p-3 mb-3 bg-info text-white rounded shadow">

## <Card.Body>

## <i className="fas fa-bullhorn fa-3x mb-3"></i>

## <Card.Title>Mission</Card.Title>

## <Card.Text>

## The idea is to create self-sustained chapters across the world

## who will look after their local community. And in the process,

## inspire people around us to give back to those who need it

## most.

## </Card.Text>

## </Card.Body>

## </Card>

## </Fade>

## </Col>

## </Row>

## </Container>

## </>

## );

## };

## export default AboutUs;

## User Login Page:-

## import React, { useState, useEffect } from "react";

## import { Button, Col, Container, Form, Row } from "react-bootstrap";

## import { Link, useNavigate } from "react-router-dom";

## import { donorLogin } from "../services/DonerApiService";

## import { NavigationBar } from "./NavigationBar";

## import ReCAPTCHA from "react-google-recaptcha";

## import { motion } from "framer-motion";

## export function UserLoginForm() {

## // Define state for form inputs and errors

## const [donorDetails, setDonorDetails] = useState({

## donorEmail: "",

## donorPassword: "",

## });

## const [verified, setVerified] = useState(true);

## const [loading, setLoading] = useState(false);

## const [donorEmailError, setDonorEmailError] = useState("");

## const [donorPasswordError, setDonorPasswordError] = useState("");

## const [errors, setErrors] = useState("");

## const [isVisible, setIsVisible] = useState(false); // For animation

## const navigate = useNavigate();

## useEffect(() => {

## // Show the form with a fade-in effect

## setIsVisible(true);

## }, []);

## const handleChange = (e) => {

## setDonorDetails({ ...donorDetails, [e.target.name]: e.target.value });

## };

## function onChange(value) {

## console.log("Captcha value:", value);

## setVerified(false);

## }

## const handleSubmit = async (e) => {

## e.preventDefault();

## setErrors("");

## if (!validate()) {

## return;

## } else {

## setLoading(true);

## try {

## let obj = {

## donorEmail: donorDetails.donorEmail.toLowerCase(),

## donorPassword: donorDetails.donorPassword,

## };

## const response = await donorLogin(obj);

## console.log(response);

## if (response.status === 200) {

## if (response.data !== "") {

## if (response.data.adminEmail) {

## localStorage.setItem("admin", JSON.stringify(response.data));

## console.log(JSON.parse(localStorage.getItem("admin")));

## navigate("/adminDonationAssign");

## } else if (response.data.donorId) {

## localStorage.setItem("donor", JSON.stringify(response.data));

## console.log(JSON.parse(localStorage.getItem("donor")));

## navigate("/donorHome");

## } else if (response.data.recipientId) {

## localStorage.setItem("recipient", JSON.stringify(response.data));

## console.log(JSON.parse(localStorage.getItem("recipient")));

## navigate("/recipientHome");

## } else if (response.data.volunteerId) {

## localStorage.setItem("volunteer", JSON.stringify(response.data));

## console.log(JSON.parse(localStorage.getItem("volunteer")));

## navigate("/volunteerHome");

## }

## } else {

## setErrors("Invalid Login Credentials");

## }

## }

## } catch (error) {

## setErrors("An error occurred. Please try again.");

## } finally {

## setLoading(false);

## }

## }

## };

## const validate = () => {

## let isValid = true;

## if (donorDetails.donorEmail.trim() === "") {

## setDonorEmailError("Email is required");

## isValid = false;

## } else if (!/^[a-z]{1}[a-z0-9]{0,}@[a-z]{1,}\.[a-z]{1,}$/i.test(donorDetails.donorEmail)) {

## setDonorEmailError("Email is not valid");

## isValid = false;

## } else {

## setDonorEmailError("");

## }

## if (donorDetails.donorPassword.trim() === "") {

## setDonorPasswordError("Password is required");

## isValid = false;

## } else if (donorDetails.donorPassword.length < 8) {

## setDonorPasswordError("Password must be at least 8 characters long");

## isValid = false;

## } else {

## setDonorPasswordError("");

## }

## return isValid;

## };

## return (

## <>

## <NavigationBar />

## <Container className="bg-light text-center mt-0 p-3 rounded">

## <h1 className="display-4 text-primary mb-4 font-weight-bold">

## SPREAD THE SMILE-DONATION PLATFORM

## </h1>

## <p className="lead">

## Our mission is to help you make a positive impact on the world by

## enabling you to easily donate to causes you care about.

## </p>

## </Container>

## <Container

## fluid

## className="d-flex justify-content-center align-items-center pb-3"

## style={{ backgroundColor: "#f7f7f7" }}

## >

## <motion.div

## initial={{ opacity: 0, scale: 0.9 }}

## animate={{ opacity: isVisible ? 1 : 0, scale: isVisible ? 1 : 0.9 }}

## transition={{ duration: 0.5 }}

## >

## <Form

## onSubmit={handleSubmit}

## className="p-5 rounded"

## style={{

## backgroundColor: "rgba(255, 255, 255, 0.8)",

## boxShadow: "0px 0px 10px rgba(0, 0, 0, 0.1)",

## }}

## >

## <h1 className="mb-4 text-center mt-2 p-3">Login</h1>

## <Row>

## <Col lg={12}>

## <Form.Group className="mb-3">

## <Form.Label>User Email</Form.Label>

## <Form.Control

## type="text"

## placeholder="Enter email"

## name="donorEmail"

## value={donorDetails.donorEmail}

## onChange={handleChange}

## className={

## donorEmailError

## ? "form-control-danger"

## : "form-control-primary"

## }

## />

## {donorEmailError && (

## <Form.Text className="text-danger">

## {donorEmailError}

## </Form.Text>

## )}

## </Form.Group>

## </Col>

## <Col lg={12}>

## <Form.Group className="mb-3">

## <Form.Label>Password</Form.Label>

## <Form.Control

## type="password"

## placeholder="Enter password"

## name="donorPassword"

## value={donorDetails.donorPassword}

## onChange={handleChange}

## className={

## donorPasswordError

## ? "form-control-danger"

## : "form-control-primary"

## }

## />

## {donorPasswordError && (

## <Form.Text className="text-danger">

## {donorPasswordError}

## </Form.Text>

## )}

## </Form.Group>

## </Col>

## </Row>

## <Col className="text-center">

## {errors && <Form.Text className="text-danger">{errors}</Form.Text>}

## </Col>

## <div className="d-flex justify-content-center mb-3">

## <ReCAPTCHA

## sitekey="6LdTH\_UkAAAAAHKD0qtLrAlrb1CFvsYb1C4XpZfD"

## onChange={onChange}

## />

## </div>

## <div className="text-center">

## <Button

## type="submit"

## variant="primary"

## disabled={verified}

## className="mb-3"

## style={{ backgroundColor: "#007bff", borderColor: "#007bff" }}

## >

## {loading ? "Logging in..." : "Login"}

## </Button>

## </div>

## <div className="text-center">

## <Link to={"/forgotPassword"}>Forget password?</Link>

## </div>

## </Form>

## </motion.div>

## </Container>

## </>

## );

## }

## Contact US Page:-

## import React from "react";

## import { Container, Row, Col, Card } from "react-bootstrap";

## import { NavigationBar } from "./NavigationBar";

## import Footer from "./Footer";

## import { Fade, Slide } from "react-awesome-reveal"; // Import animations

## export function Contact() {

## return (

## <>

## <NavigationBar />

## <Container className="bg-light text-center mt-2 p-4 rounded shadow-sm">

## <Fade>

## <h1 className="display-4 text-primary mb-4 font-weight-bold">

## SPREAD THE SMILE-DONATION PLATFORM

## </h1>

## <p className="lead">

## Our mission is to help you make a positive impact on the world by

## enabling you to easily donate to causes you care about.

## </p>

## </Fade>

## </Container>

## <Container

## className="bg-light mt-4 p-5 rounded shadow-sm"

## style={{ marginBottom: "100px" }}

## >

## <Row className="justify-content-center">

## <Col md={8}>

## <Slide direction="up">

## <h2 className="text-center font-weight-bold mb-4">

## Contact Us

## </h2>

## </Slide>

## <Slide direction="up" delay={100}>

## <p className="text-center lead mb-4">

## In India, we often see NGOs and donors working in silos lacking

## a common standardized platform to handle end-to-end donation

## processes.

## <span className="text-primary"> Spread the Smile-Donation Platform</span> aims to

## address this critical issue by bridging this gap between donors

## and recipients. This platform promises to bring credibility and

## efficiency to the existing system.

## </p>

## <p className="text-center lead mb-5">

## If you have any questions, feedback, or want to get involved with

## <span className="text-primary"> Spread the Smile-Donation Platform</span>, please

## don't hesitate to reach out to us.

## </p>

## </Slide>

## <Slide direction="up" delay={200}>

## <Row className="justify-content-center">

## <Col md={6}>

## <Card className="shadow">

## <Card.Body>

## <h2 className="text-primary font-weight-bold mb-4">

## Contact Information:

## </h2>

## <p className="mb-0">

## <i className="fas fa-map-marker-alt mr-2"></i>

## Address: XYZ street, ABC city, India

## </p>

## <p className="mb-0">

## <i className="fas fa-envelope mr-2"></i>

## Email: info@charityxchange.com

## </p>

## <p className="mb-0">

## <i className="fas fa-phone-alt mr-2"></i>

## Phone: +91-1234567890

## </p>

## </Card.Body>

## </Card>

## </Col>

## <Col md={6}>

## <Card className="shadow">

## <Card.Body>

## <p className="text-primary font-weight-bold mb-4">

## For inquiries or partnership opportunities, please

## contact us:

## <br />

## Email:{" "}

## <a

## href="mailto:info@charityxchange.com"

## style={{ color: "black" }}

## >

## info@charityxchange.com

## </a>

## </p>

## <p>Phone: +91-1234567890</p>

## </Card.Body>

## </Card>

## </Col>

## </Row>

## </Slide>

## </Col>

## </Row>

## </Container>

## <Footer />

## </>

## );

## }

## BackEnd code:

## AdminRestController:

1. **package** com.DonationBackend.cntr;
2. **import** org.mindrot.jbcrypt.BCrypt;
3. **import** org.springframework.beans.factory.annotation.Autowired;
4. **import** org.springframework.web.bind.annotation.CrossOrigin;
5. **import** org.springframework.web.bind.annotation.PostMapping;
6. **import** org.springframework.web.bind.annotation.RequestBody;
7. **import** org.springframework.web.bind.annotation.RestController;
8. **import** com.DonationBackend.model.Admin;
9. **import** com.DonationBackend.model.Donor;
10. **import** com.DonationBackend.model.Recipient;
11. **import** com.DonationBackend.model.Volunteer;
12. **import** com.DonationBackend.service.AdminService;
13. **import** com.DonationBackend.service.DonorService;
14. **import** com.DonationBackend.service.RecipientService;
15. **import** com.DonationBackend.service.VolunteerService;
16. @CrossOrigin
17. @RestController
18. **public** **class** AdminRestController {
19. @Autowired
20. **private** DonorService donorService;
22. @Autowired
23. **private** AdminService adminService;
25. @Autowired
26. **private** RecipientService recipientService;
28. @Autowired
29. **private** VolunteerService volunteerService;

32. @PostMapping("/adminRegister")
33. **public** Admin registerAdmin(@RequestBody Admin admin) {
34. // Hash the admin password before saving
35. String hashedPassword = BCrypt.*hashpw*(admin.getAdminPassword(), BCrypt.*gensalt*());
36. admin.setAdminPassword(hashedPassword);
37. // Set the admin email
38. String adminEmail = admin.getAdminEmail();
39. admin.setAdminEmail(adminEmail);
40. **return** adminService.registerAdmin(admin);
41. }
43. //to check password with encrypted password
44. **static** **private** **boolean** checkPass(String plainPassword, String hashedPassword) {
45. **if** (BCrypt.*checkpw*(plainPassword, hashedPassword)) {
46. System.***out***.println("The password matches.");
47. **return** **true**;
48. }
49. **else** {
50. System.***out***.println("The password does not match.");
51. **return** **false**;
52. }
53. }
55. //admin table check for logged in email and password
56. **public** Admin adminCheck(String email, String plainPassword ) {
58. Admin admin = adminService.getById(email);
59. **if** (admin != **null**) {
60. String hashedPassword=admin.getAdminPassword();
61. **if**(*checkPass*(plainPassword, hashedPassword)) {
62. **return** admin;
63. }
64. }
65. **return** **null**;
66. }
68. //donor table check for logged in email and password
69. **public** Donor donorCheck(String email, String plainPassword) {
71. Donor donor=donorService.getByEmail(email);
72. **if** (donor != **null**) {
73. String hashedPassword=donor.getDonorPassword();
74. System.***out***.println(plainPassword);
75. System.***out***.println(hashedPassword);
76. **if**(*checkPass*(plainPassword, hashedPassword)) {
77. System.***out***.println("2");
78. System.***out***.println(donor);
79. **return** donor;
80. }
81. }
82. **return** **null**;
83. }
85. //recipient table check for logged in email and password
86. **public** Recipient recipientCheck(String email, String plainPassword) {
88. Recipient recipient = recipientService.getByEmail(email);
89. **if** (recipient != **null**) {
90. String hashedPassword=recipient.getRecipientPassword();
91. **if**(*checkPass*(plainPassword, hashedPassword)) {
92. System.***out***.println("Recipient");
93. System.***out***.println(recipient);
94. **return** recipient;
95. }
96. }
97. **return** **null**;
98. }
100. //volunteer table check for logged in email and password
101. **public** Volunteer volunteerCheck(String email, String plainPassword) {
103. Volunteer volunteer = volunteerService.getByEmail(email);
104. **if** (volunteer != **null**) {
105. String hashedPassword=volunteer.getVolunteerPassword();
106. System.***out***.println(plainPassword);
107. System.***out***.println(hashedPassword);
108. **if**(*checkPass*(plainPassword, hashedPassword)) {
109. **return** volunteer;
110. }
111. }
112. **return** **null**;
113. }
115. @PostMapping(value= {"/donorLogin"})
116. **public** Object userLogin(@RequestBody Donor donorObj ) {
118. String email = donorObj.getDonorEmail();
119. String plainPassword=donorObj.getDonorPassword();
121. **if**(adminCheck(email, plainPassword)!=**null**) {
122. **return** adminCheck(email, plainPassword);
123. }
124. **else** **if**(donorCheck(email, plainPassword)!=**null**) {
125. **return** donorCheck(email, plainPassword);
126. }
127. **else** **if**(recipientCheck(email, plainPassword)!=**null**) {
128. **return** recipientCheck(email, plainPassword);
129. }
130. **else** **if**(volunteerCheck(email, plainPassword)!=**null**) {
131. **return** volunteerCheck(email, plainPassword);
132. }
133. **else**
134. **return** **null**;
135. }
137. //all the above code is for login check in different roles across all tables
138. @PostMapping(value= {"/checkEmail"})
139. **public** **boolean** checkEmail(@RequestBody Donor donorObj) {
140. String email=donorObj.getDonorEmail();
141. System.***out***.println("emailCheck method");
142. System.***out***.println(email);
143. **if**(adminService.getById(email)!=**null**) {
144. **return** **true**;
145. }
146. **else** **if**(donorService.getByEmail(email)!=**null**) {
147. **return** **true**;
148. }
149. **else** **if**(recipientService.getByEmail(email)!=**null**) {
150. **return** **true**;
151. }
152. **else** **if**(volunteerService.getByEmail(email)!=**null**) {
153. **return** **true**;
154. }
155. **else** {
156. **return** **false**;
157. }
158. }
159. @PostMapping(value= {"/userPasswordUpdate"})
160. **public** **boolean** userPasswordUpdate(@RequestBody Donor donorObj) {
161. String email=donorObj.getDonorEmail();
162. **if**(donorService.getByEmail(email)!=**null**) {
163. Donor donorobj =donorService.getByEmail(email);
164. donorobj.setDonorPassword(donorObj.getDonorPassword());
165. donorService.add(donorobj);
166. **return** **true**;
167. }
168. **else** **if**(recipientService.getByEmail(email)!=**null**) {
169. Recipient recipientobj=recipientService.getByEmail(email);
170. recipientobj.setRecipientPassword(donorObj.getDonorPassword());
171. recipientService.add(recipientobj);
172. **return** **true**;
173. }
174. **else** **if**(volunteerService.getByEmail(email)!=**null**) {
175. Volunteer volunteerobj=volunteerService.getByEmail(email);
176. volunteerobj.setVolunteerPassword(donorObj.getDonorPassword());
177. volunteerService.add(volunteerobj);
178. **return** **true**;
179. }
180. **else** {
181. **return** **false**;
182. }
183. }
185. }

## 5.2.2 DonationDetailsDao:

## package com.DonationBackend.dao;

## import java.util.List;

## import org.springframework.data.jpa.repository.JpaRepository;

## import org.springframework.data.jpa.repository.Query;

## import org.springframework.data.repository.query.Param;

## import org.springframework.stereotype.Repository;

## import com.DonationBackend.model.DonationDetails;

## import com.DonationBackend.model.Donor;

## import com.DonationBackend.model.Volunteer;

## @Repository

## public interface DonationDetailsDao extends JpaRepository<DonationDetails, Integer>{

## @Query(value = "select p from DonationDetails p where p.donor = :donor")

## public List<DonationDetails> selectByDonor(@Param(value = "donor") Donor donor);

## 

## @Query(value = "select p from DonationDetails p where p.volunteer = :volunteer")

## public List<DonationDetails> selectByVolunteer(@Param(value = "volunteer") Volunteer volunteer);

## 

## // //not needed thiis method

## // @Query(value = "select p from DonationDetails p where p.recipient = :recipient")

## // public List<DonationDetails> selectByRecipient(@Param(value = "recipient") Recipient recipient);

## 

## @Query(value = "select p from DonationDetails p where p.volunteer is NULL")

## public List<DonationDetails> selectNotAssignedDonations();

## 

## @Query(value = "select p from DonationDetails p where p.volunteer is NOT NULL")

## public List<DonationDetails> selectAssignedDonations();

## 

## 

## @Query(value = "select p from DonationDetails p where p.volunteer = :volunteer and p.donationStatus = 'toBeCollected'")

## public List<DonationDetails> selectByDonationToBeCollected(@Param(value = "volunteer") Volunteer volunteer);

## 

## @Query(value = "select p from DonationDetails p where p.volunteer = :volunteer and p.donationStatus = 'isCollected'")

## public List<DonationDetails> selectByDonationIsCollected(@Param(value = "volunteer") Volunteer volunteer);

## 

## }

## 5.2.3 DonationDetails:

## package com.DonationBackend.model;

## import javax.persistence.Entity;

## import javax.persistence.GeneratedValue;

## import javax.persistence.GenerationType;

## import javax.persistence.Id;

## import javax.persistence.JoinColumn;

## import javax.persistence.ManyToOne;

## import org.hibernate.annotations.ColumnDefault;

## @Entity

## public class DonationDetails {

## @Id

## @GeneratedValue(strategy = GenerationType.IDENTITY)

## private int donationId;

## @ManyToOne

## @JoinColumn(name = "donorId")

## private Donor donor;

## private int rawFoodQuantity;

## private int clothesQuantity;

## private int stationaryQuantity;

## @ManyToOne

## @JoinColumn(name = "volunteerId")

## private Volunteer volunteer;

## @ManyToOne

## @JoinColumn(name = "recipientId")

## private Recipient recipient;

## @ColumnDefault("'toBeCollected'")

## private String donationStatus;

## public int getDonationId() {

## return donationId;

## }

## public void setDonationId(int donationId) {

## this.donationId = donationId;

## }

## public Donor getDonor() {

## return donor;

## }

## public void setDonor(Donor donor) {

## this.donor = donor;

## }

## public int getRawFoodQuantity() {

## return rawFoodQuantity;

## }

## public void setRawFoodQuantity(int rawFoodQuantity) {

## this.rawFoodQuantity = rawFoodQuantity;

## }

## public int getClothesQuantity() {

## return clothesQuantity;

## }

## public void setClothesQuantity(int clothesQuantity) {

## this.clothesQuantity = clothesQuantity;

## }

## public int getStationaryQuantity() {

## return stationaryQuantity;

## }

## public void setStationaryQuantity(int stationaryQuantity) {

## this.stationaryQuantity = stationaryQuantity;

## }

## public Volunteer getVolunteer() {

## return volunteer;

## }

## public void setVolunteer(Volunteer volunteer) {

## this.volunteer = volunteer;

## }

## public Recipient getRecipient() {

## return recipient;

## }

## public void setRecipient(Recipient recipient) {

## this.recipient = recipient;

## }

## public String getDonationStatus() {

## return donationStatus;

## }

## public void setDonationStatus(String donationStatus) {

## this.donationStatus = donationStatus;

## }

## }

**5.2.4 AdminServiceImpl:**

package com.DonationBackend.service;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.DonationBackend.dao.AdminDao;

import com.DonationBackend.dao.DonorDao;

import com.DonationBackend.dao.RecipientDao;

import com.DonationBackend.dao.VolunteerDao;

import com.DonationBackend.model.Admin;

import com.DonationBackend.model.Donor;

@Service

public class AdminServiceImpl implements AdminService{

@Autowired

private AdminDao adminDao;

@Autowired

private DonorDao donorDao;

@Autowired

private RecipientDao recipientDao ;

@Autowired

private VolunteerDao volunteerDao;

@Override

public Admin getById(String adminEmail) {

Optional<Admin> opt = adminDao.findById(adminEmail);

if(opt.isPresent())

return opt.get();

else

return null;

}

@Override

public Admin registerAdmin(Admin admin) {

return adminDao.save(admin);

}

@Override

public void add(Admin admin) {

adminDao.save(admin);

}

@Override

public Admin getByEmail(String email) {

Optional<Admin> optionalAdmin = adminDao.findById(email);

return optionalAdmin.orElse(null);

}

// to check if registering email is already present

@Override

public boolean doesEmailExist(String email) {

if(adminDao.findById(email).isPresent()) {

return true;

}

else if(donorDao.selectByDonorEmail(email)!=null) {

return true;

}

else if(recipientDao.selectByEmail(email)!=null) {

return true;

}

else if(volunteerDao.selectByVolunteerEmail(email)!=null) {

return true;

}

return false;

}

}

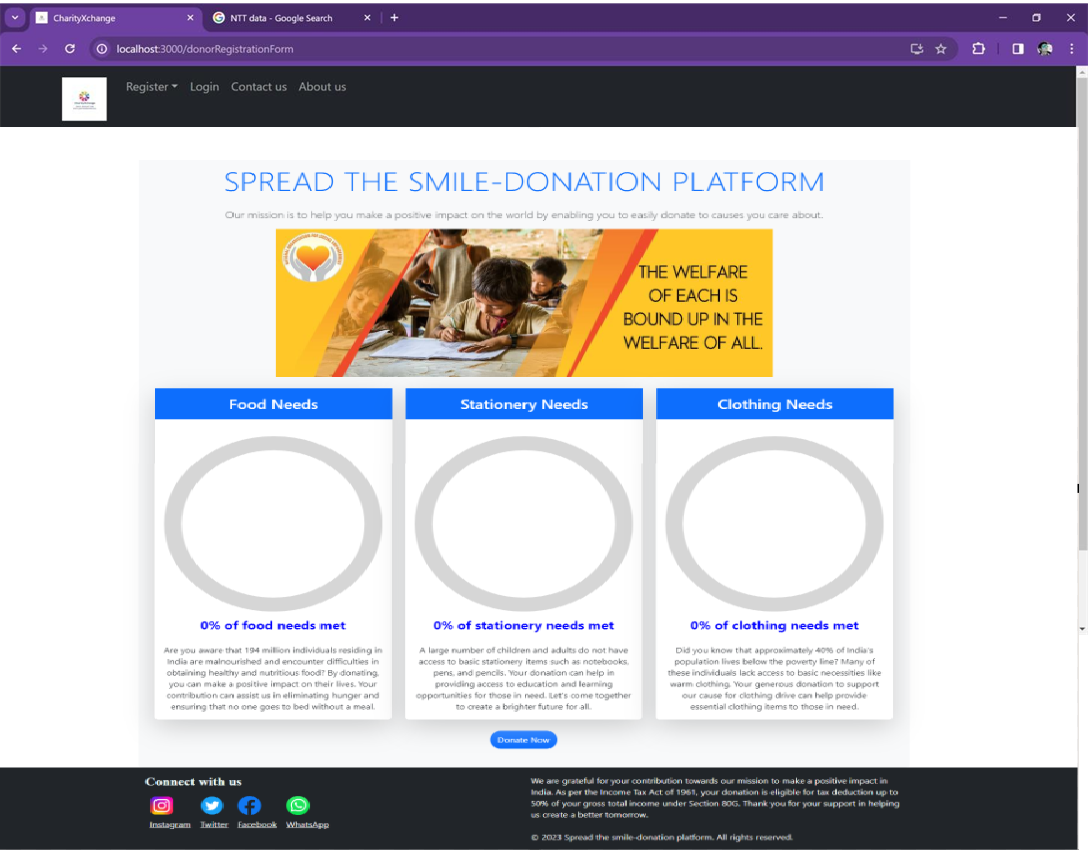
## Result Analysis:-

We have 4 module in our project

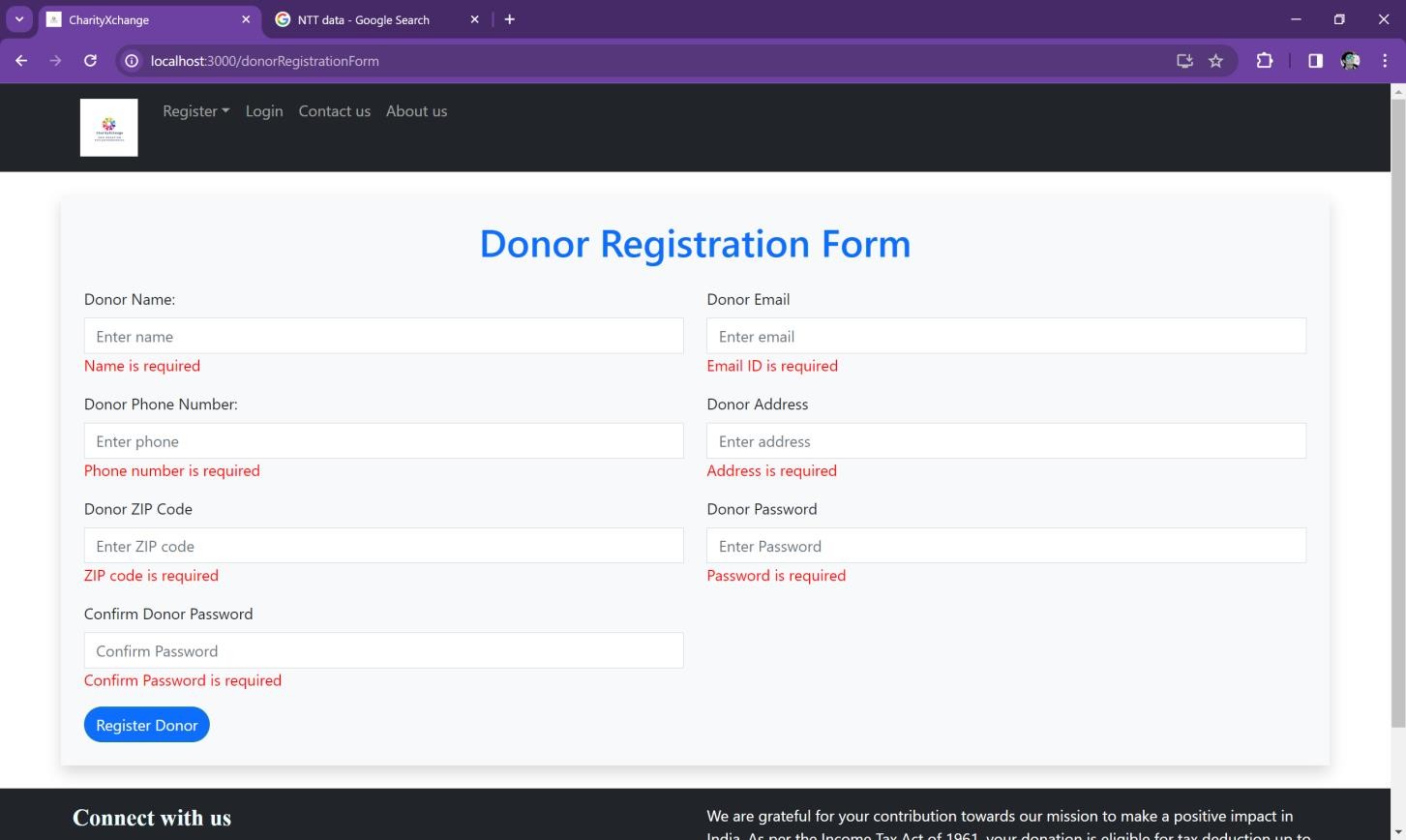
Admin , Donor , Recipient , Volunteer

* + Donor can donate the clothes ,food like that
  + Recipient that is a NGO or old age home who can receive the donation
  + Volunteer is simply the medium of delivery boy who can take donation from donor and give to the recipient.

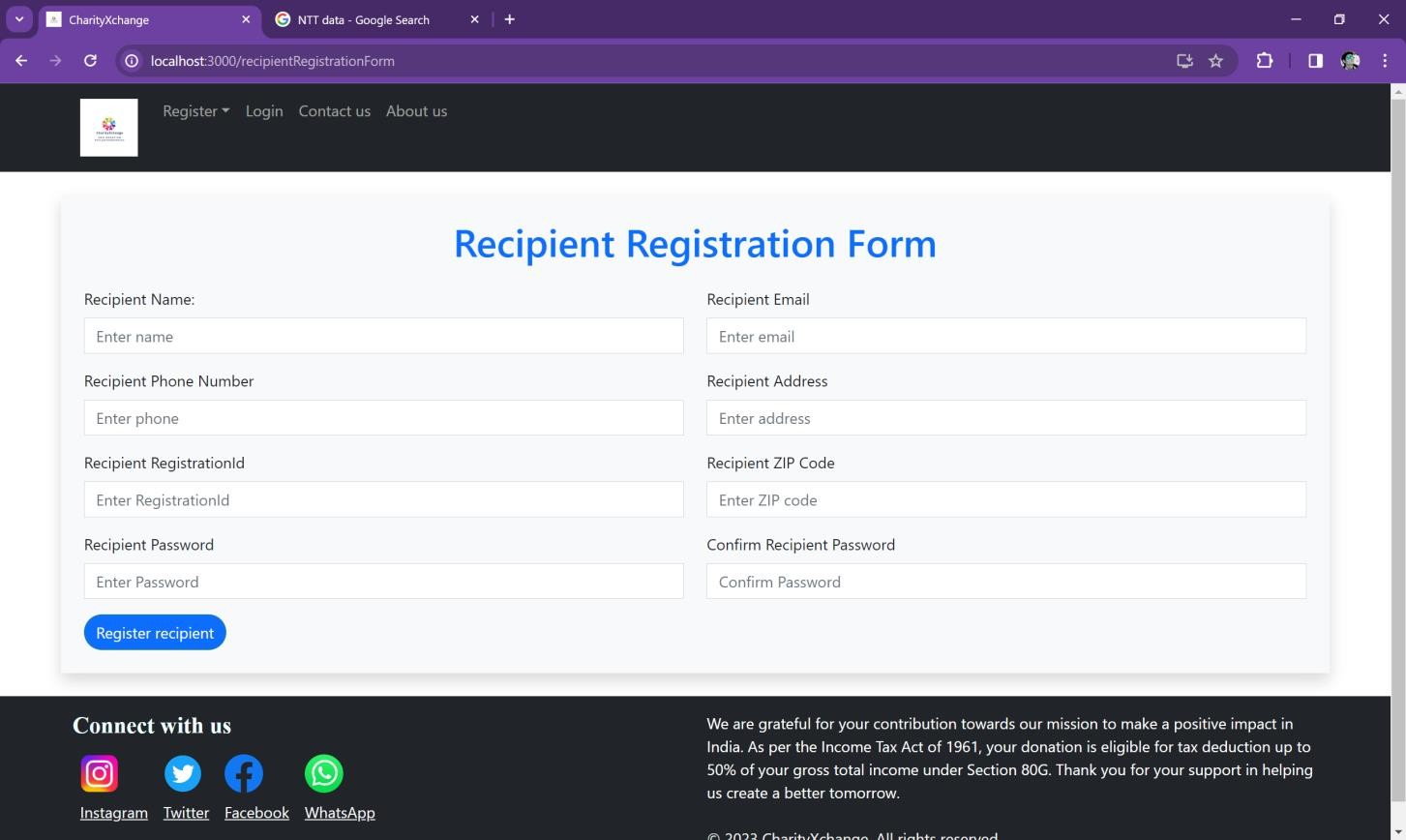
**Home Page**

****

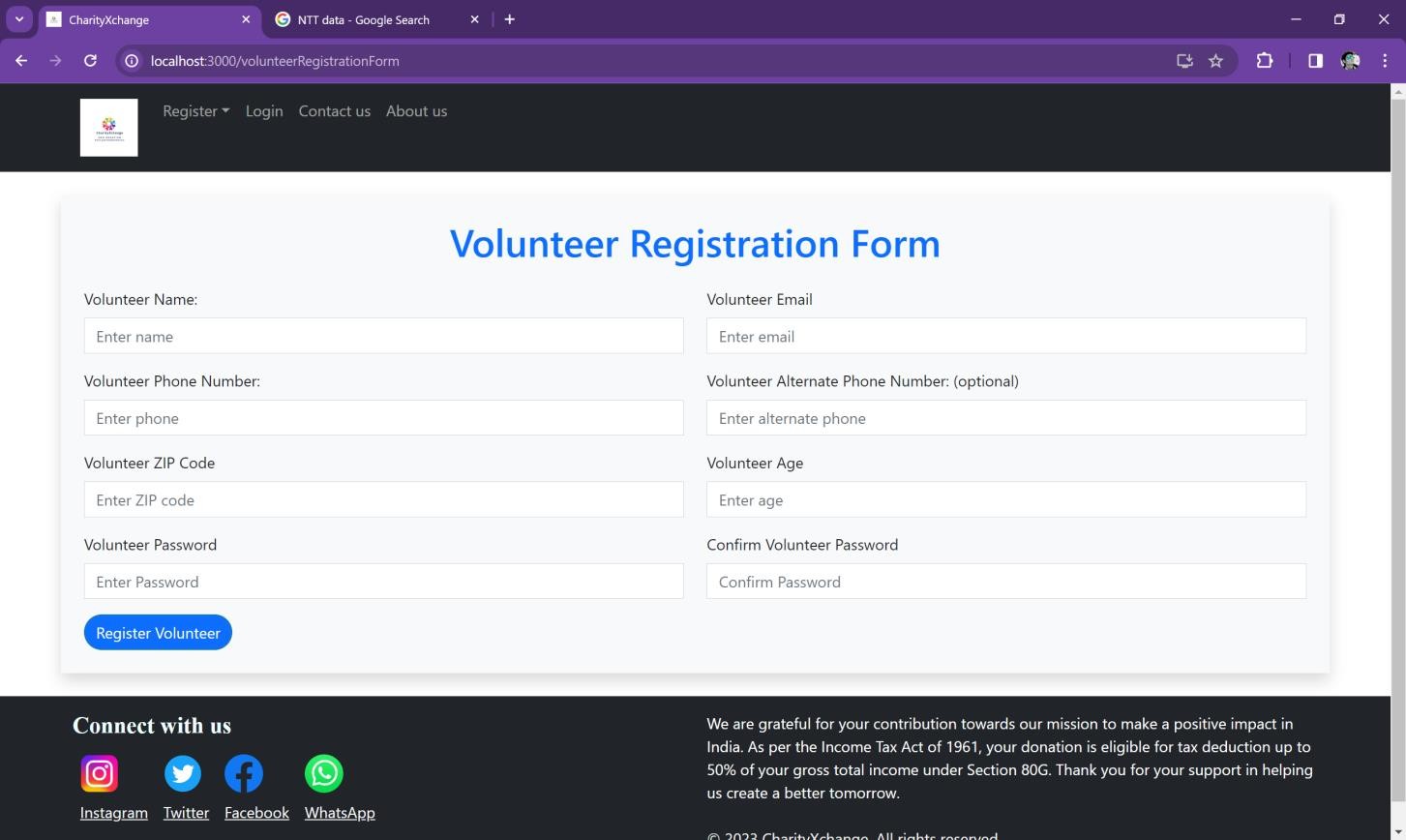
**Donor Registration Form**



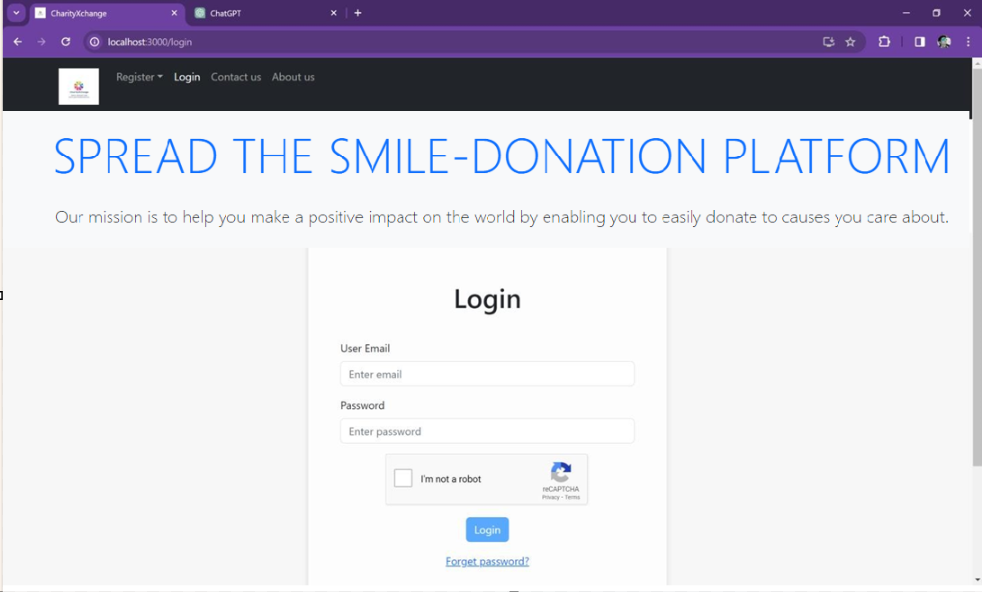
**Recipient Registration form**



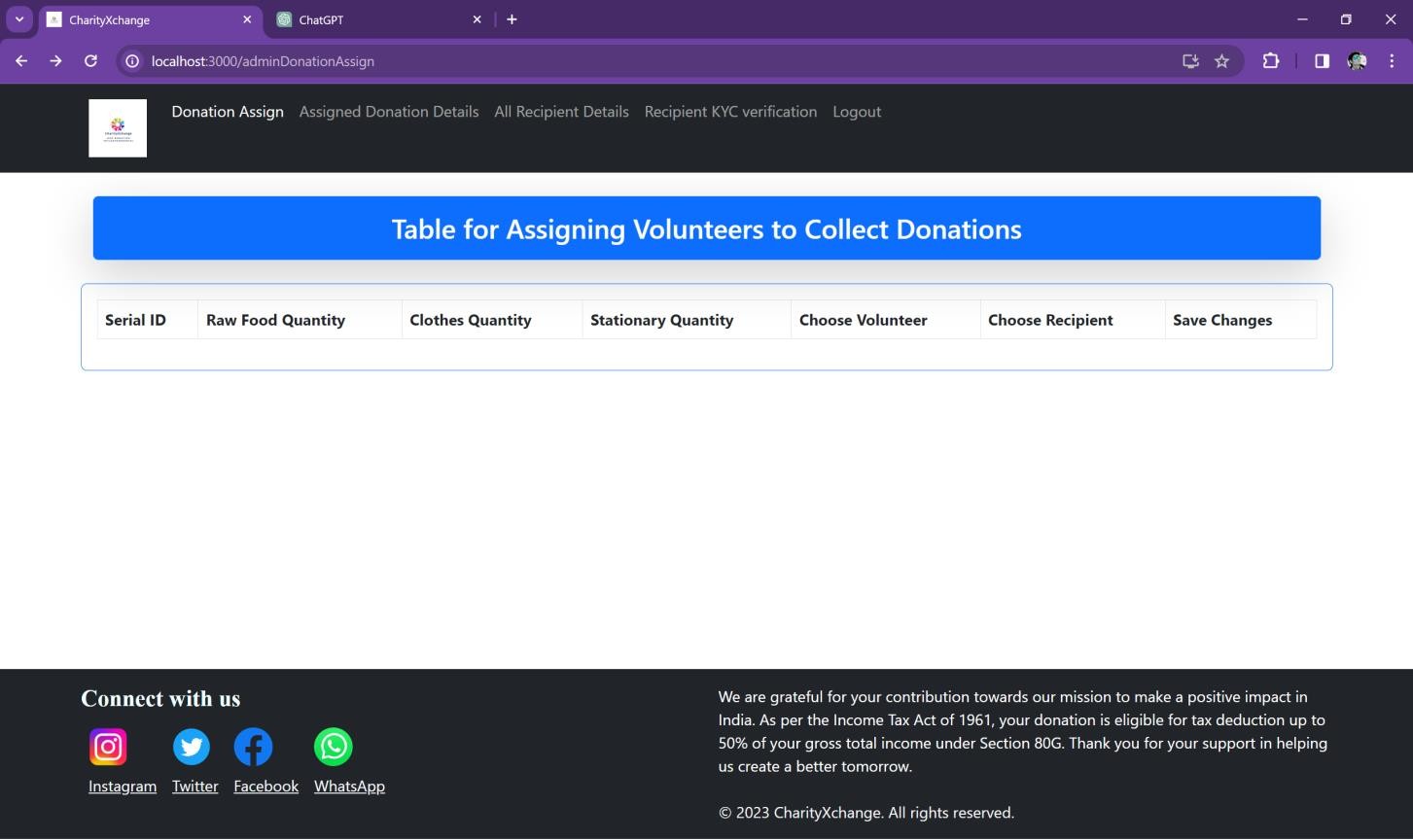
**Volunteer Registration Form**



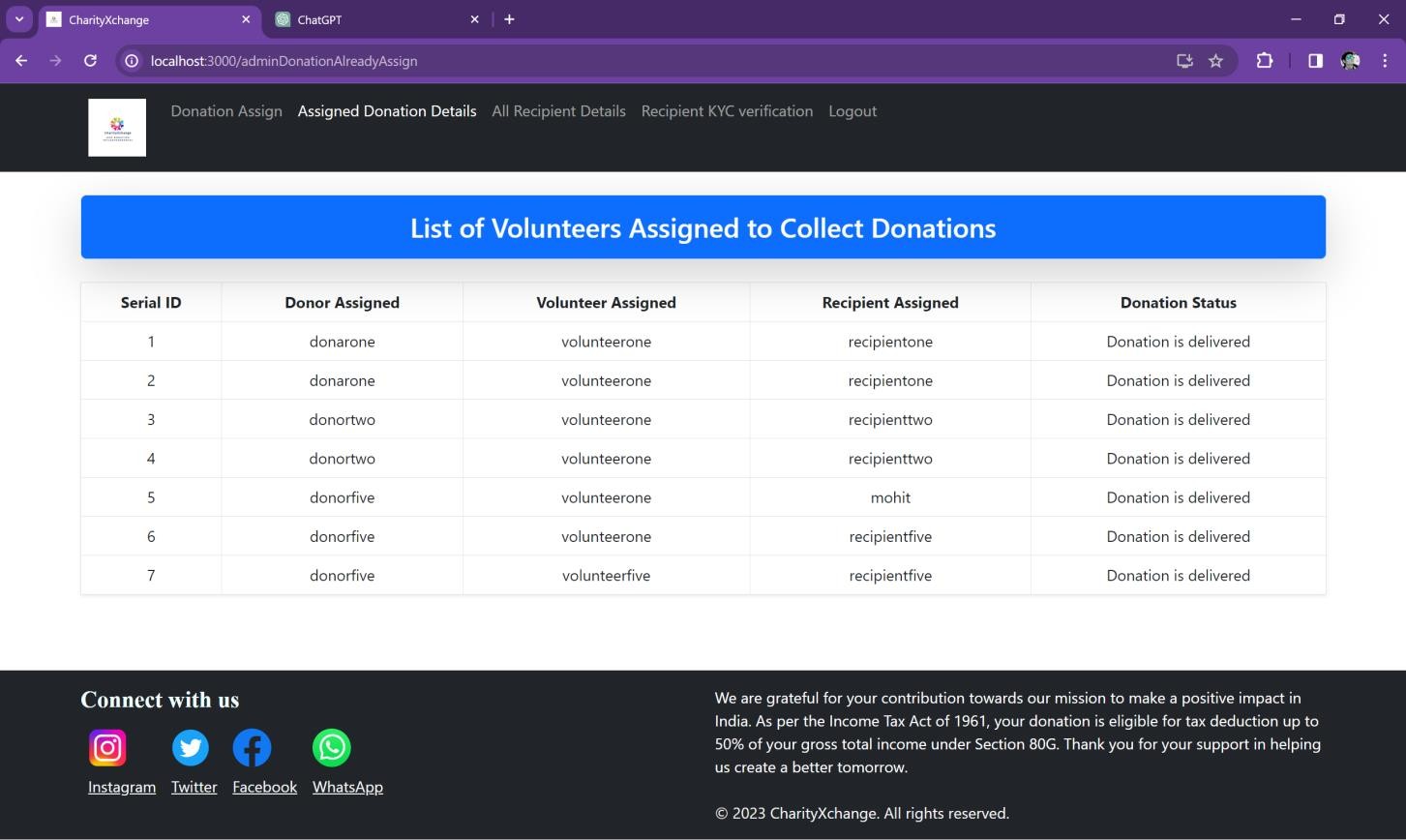
**Admin Login**



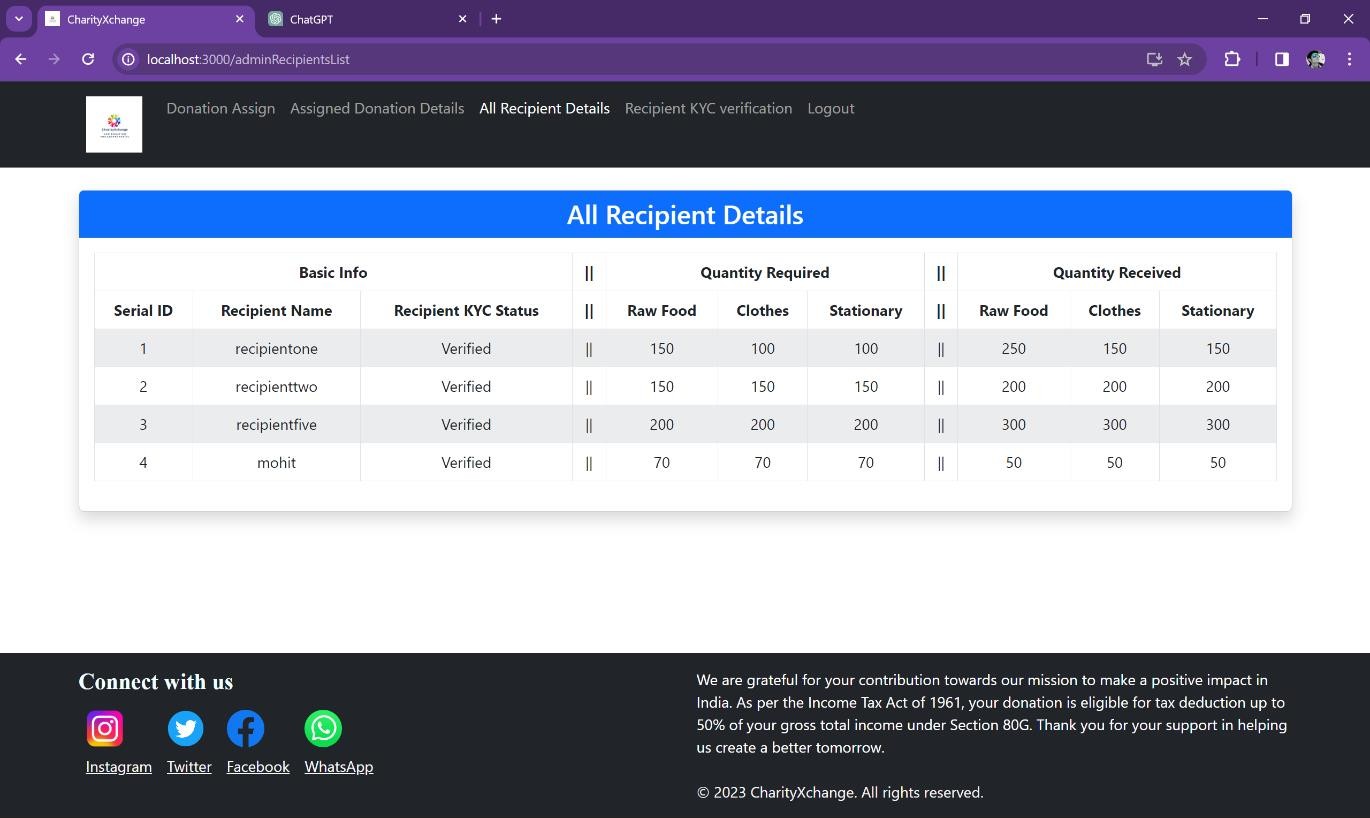
Admin dashboard he can assign volunteer from here to collect donation.



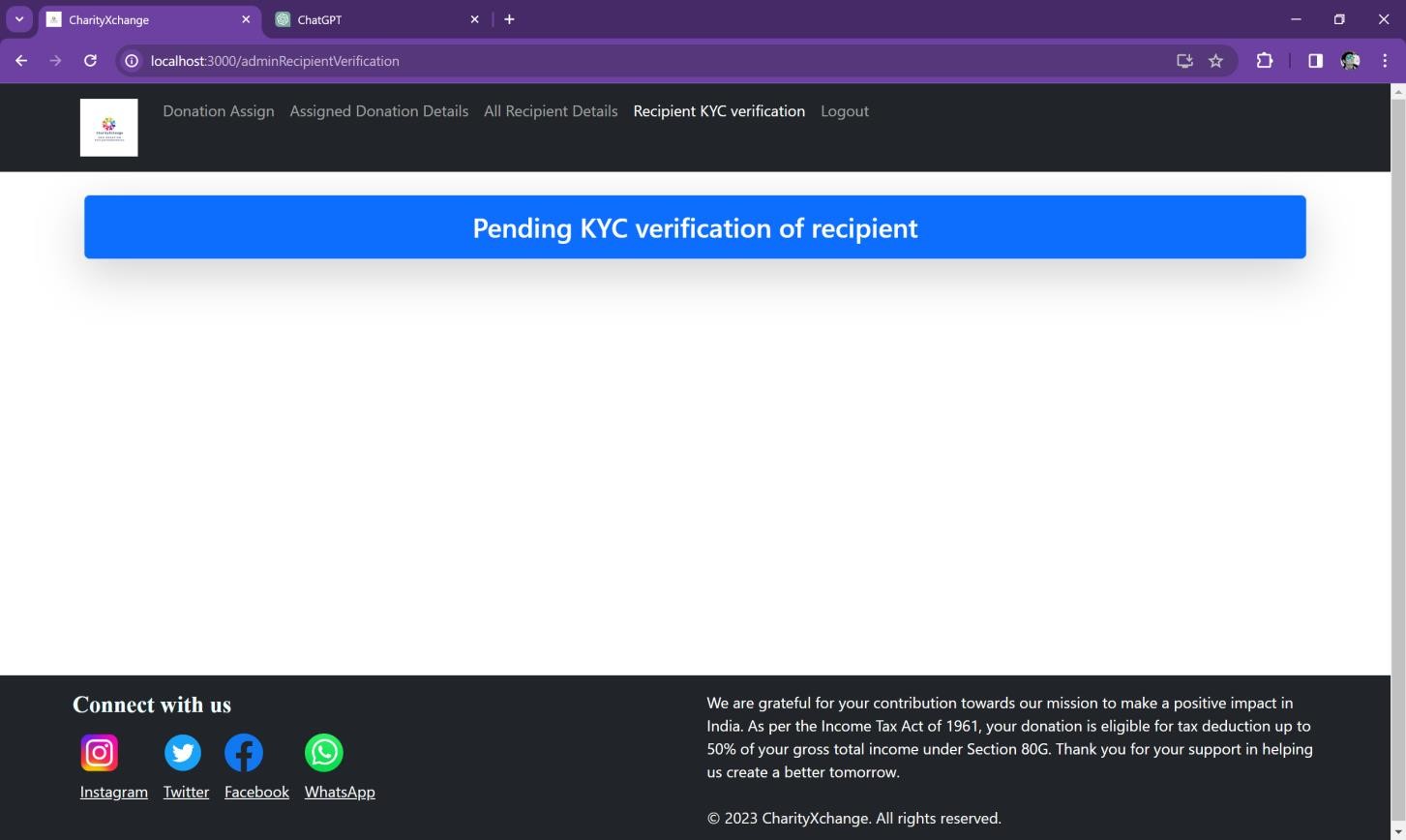
List of volunteer he can see whom to assigned and the status can see.



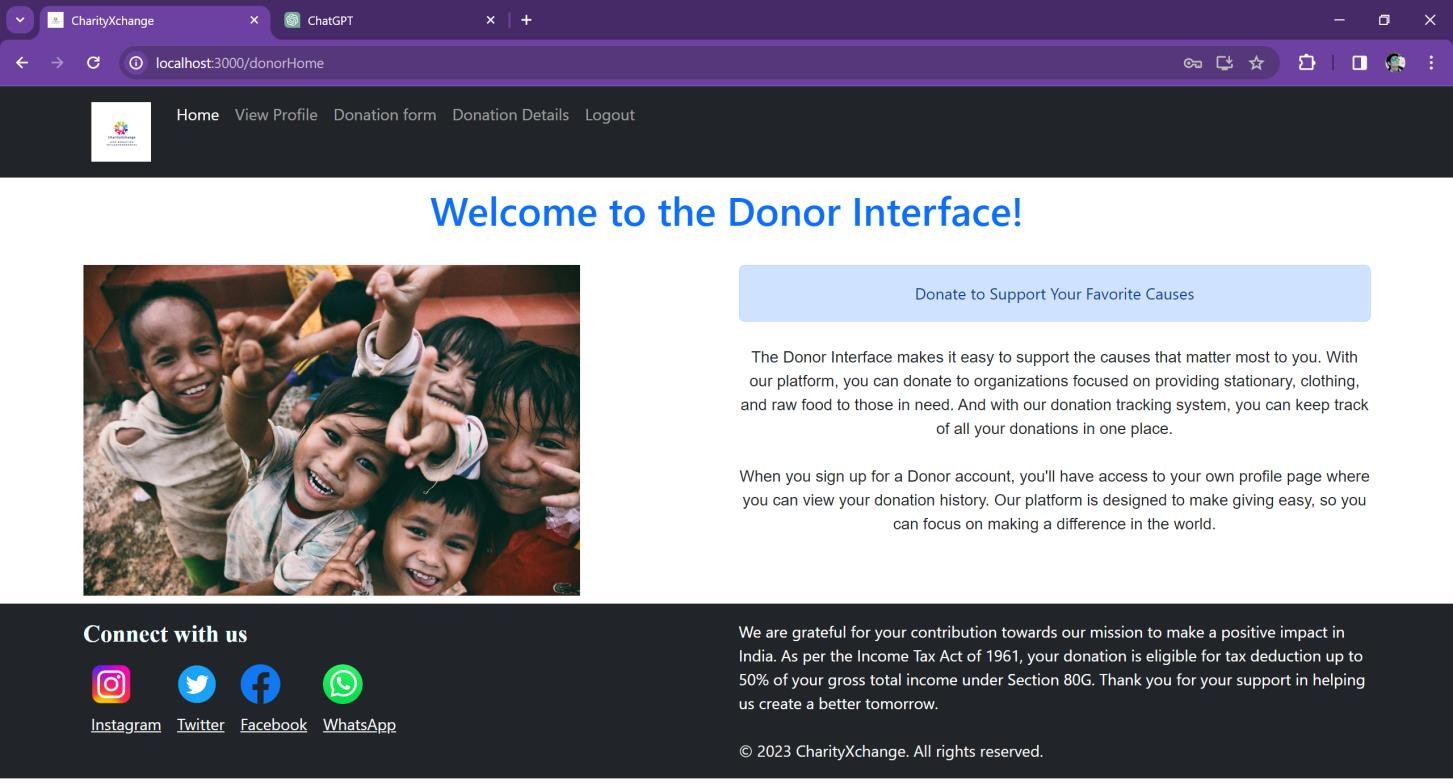
From here he can see the recipient details and the quantity of donation



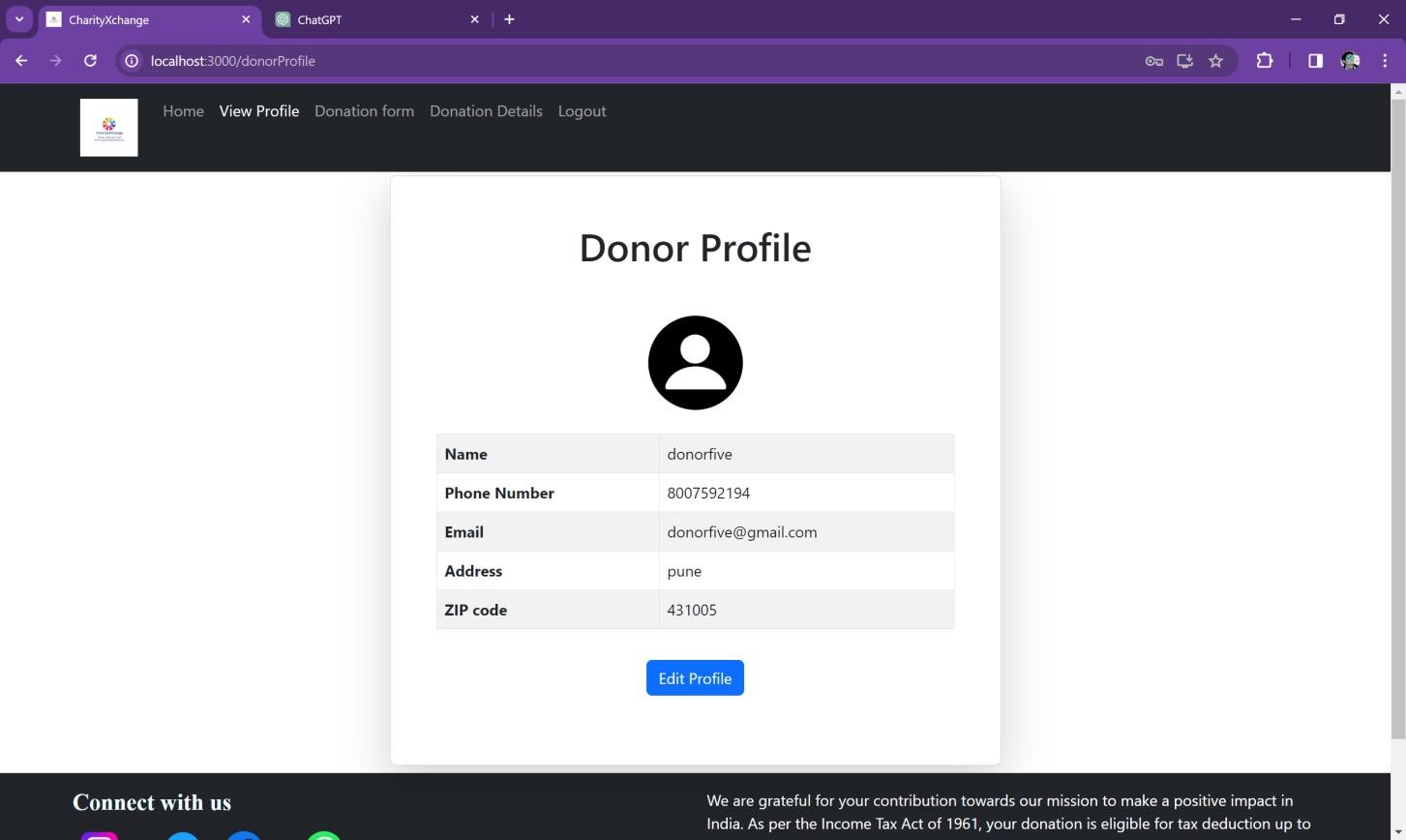
When first time recipient register he need of kyc that is verify by admin.



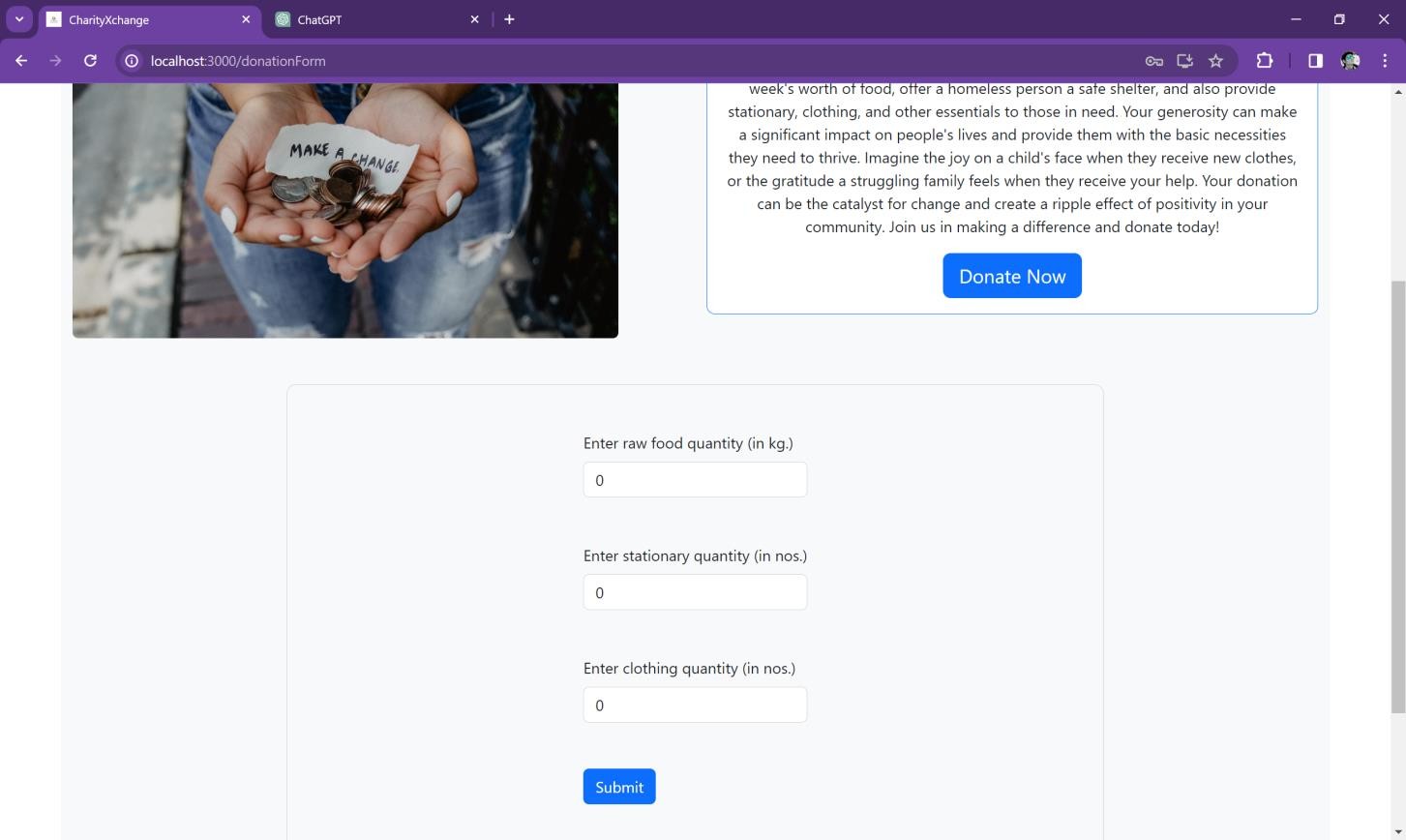
**Donor Dashboard**



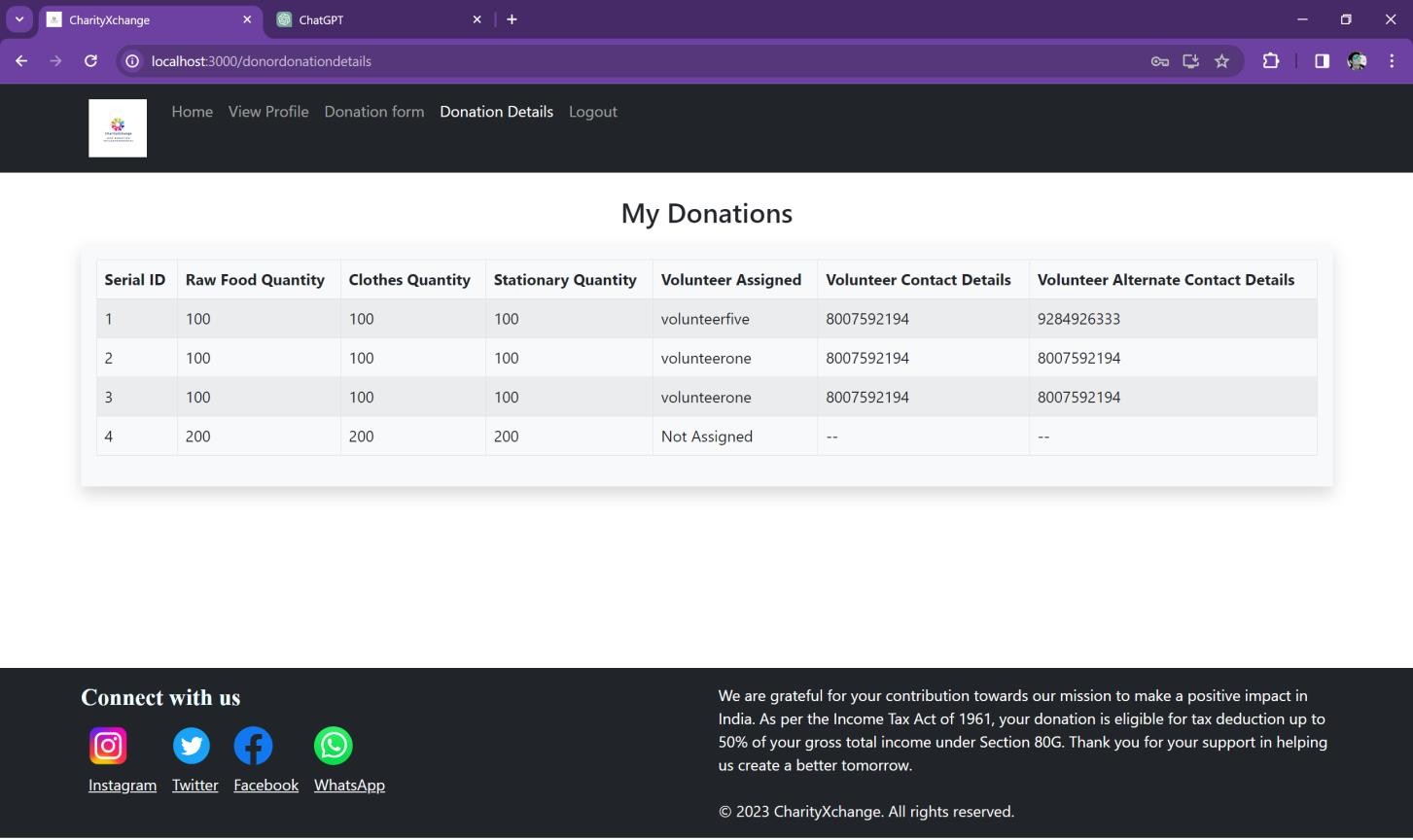
Donor profile



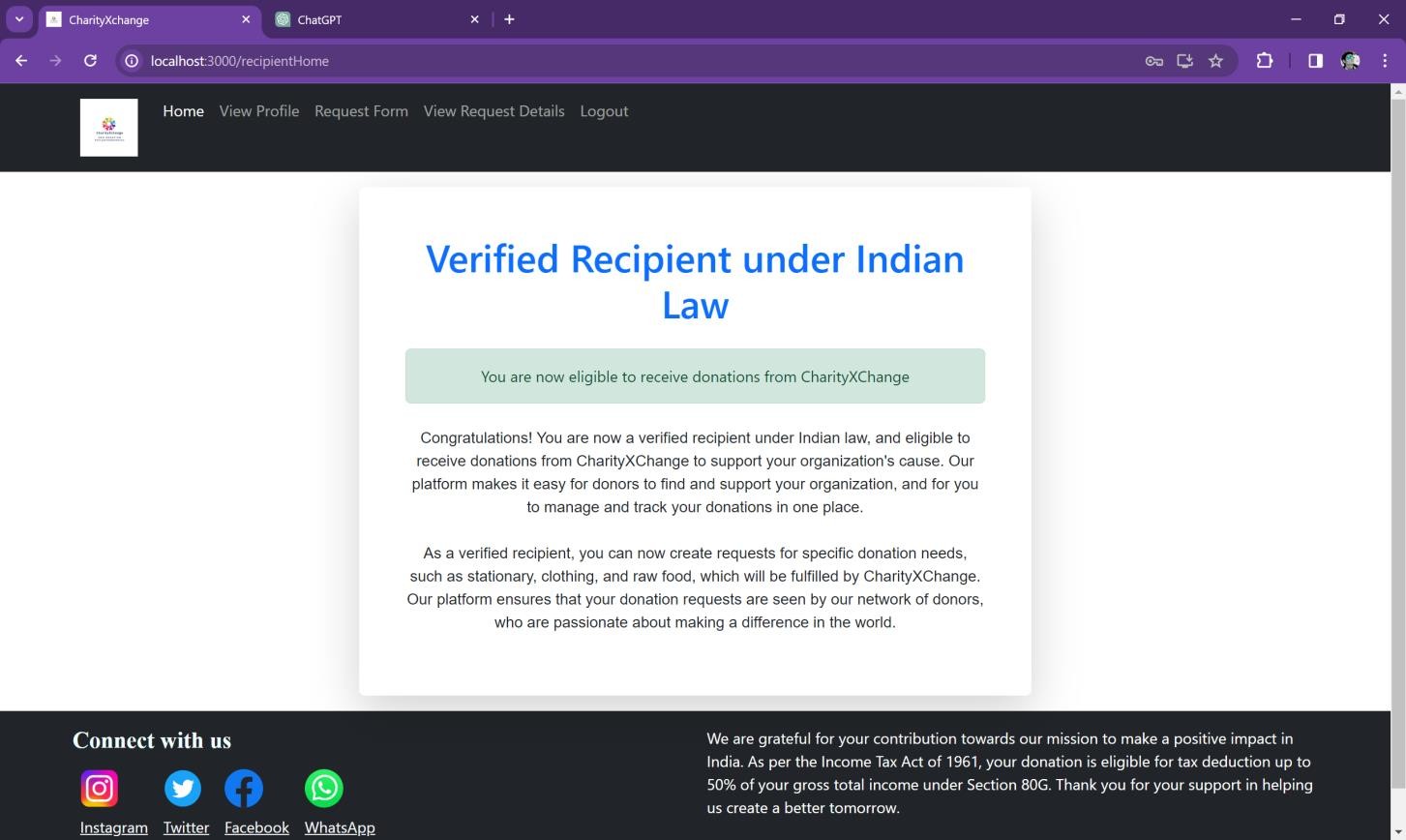
Donation form



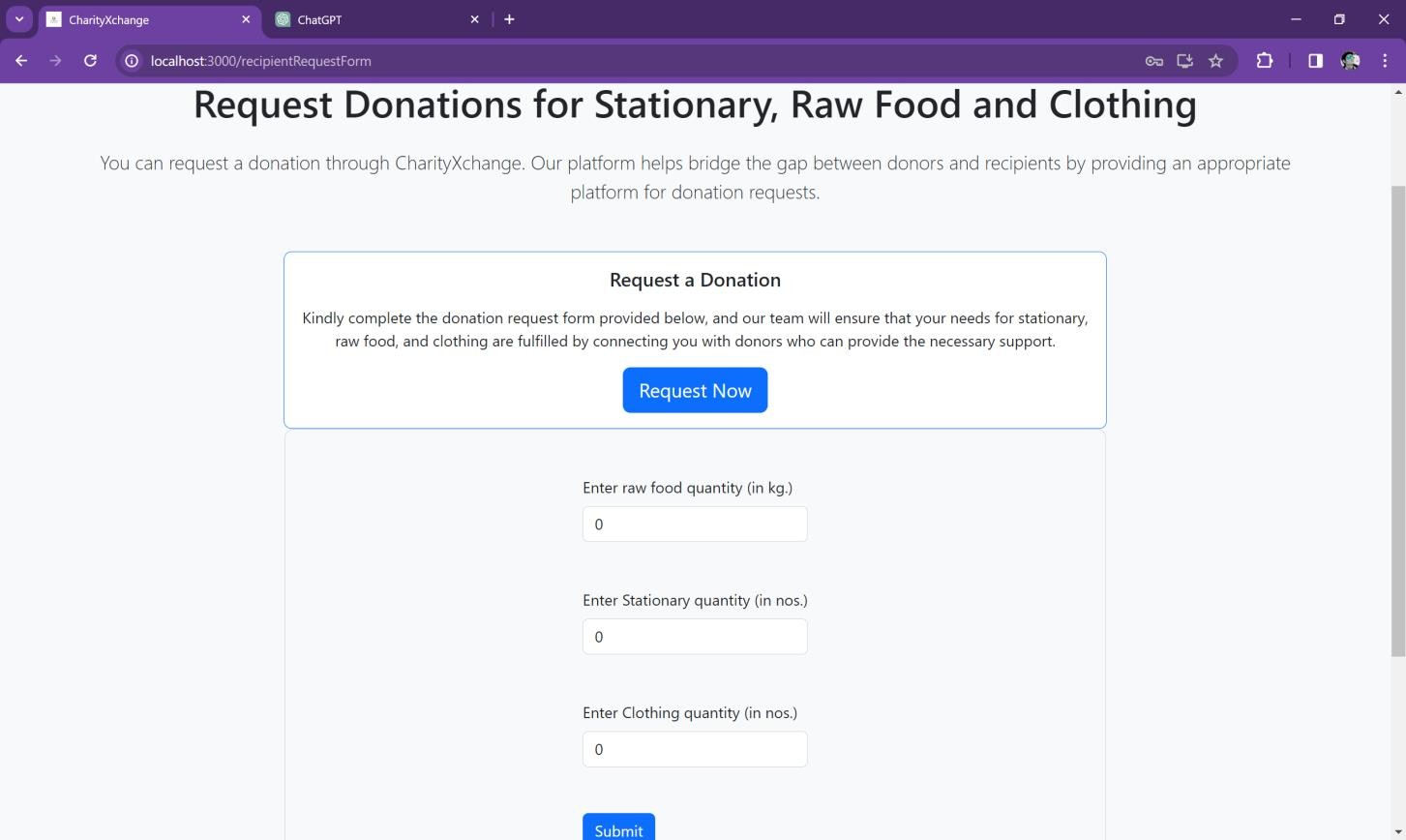
New donation will be added but not assigned yet

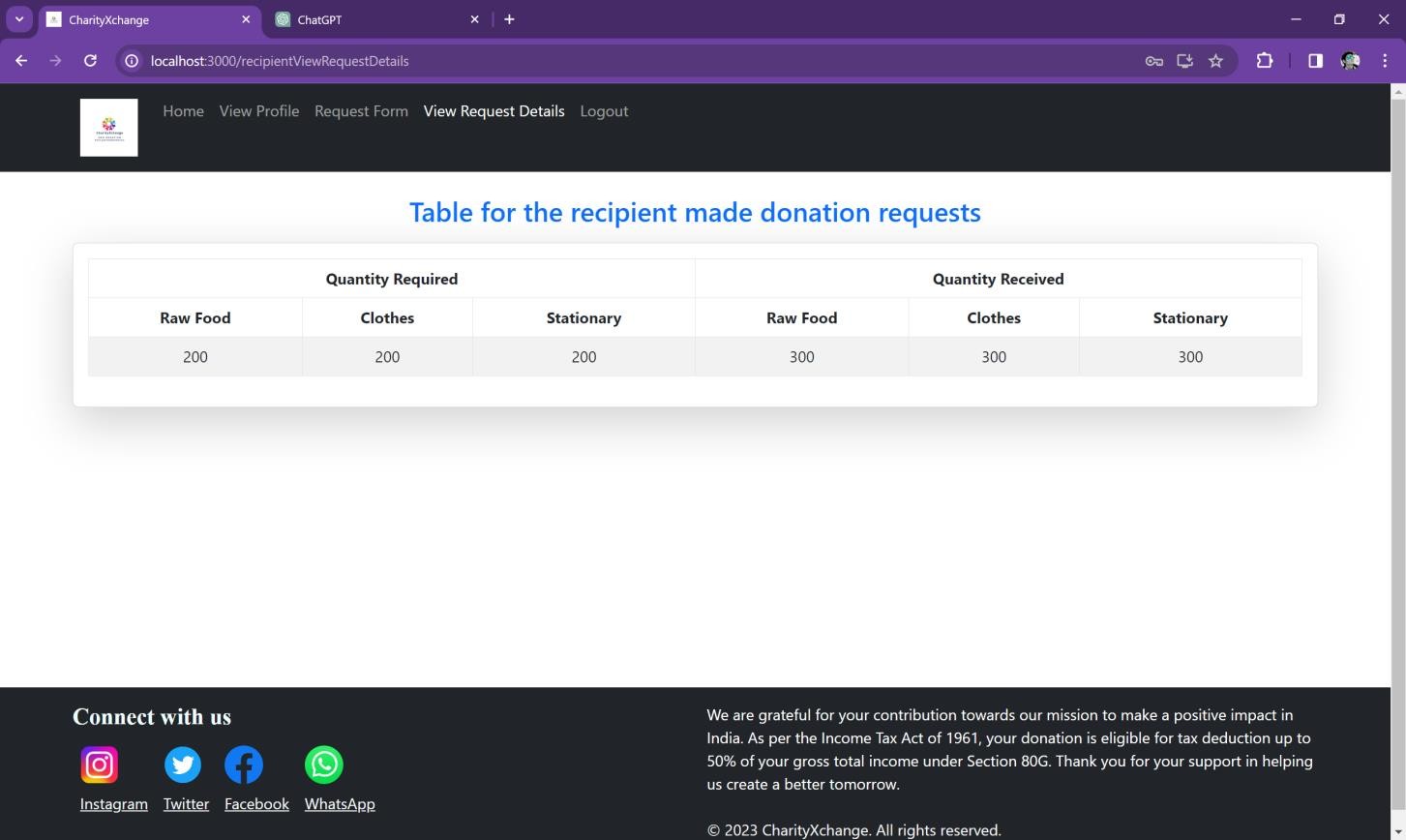


**Recipient Dashbooard**

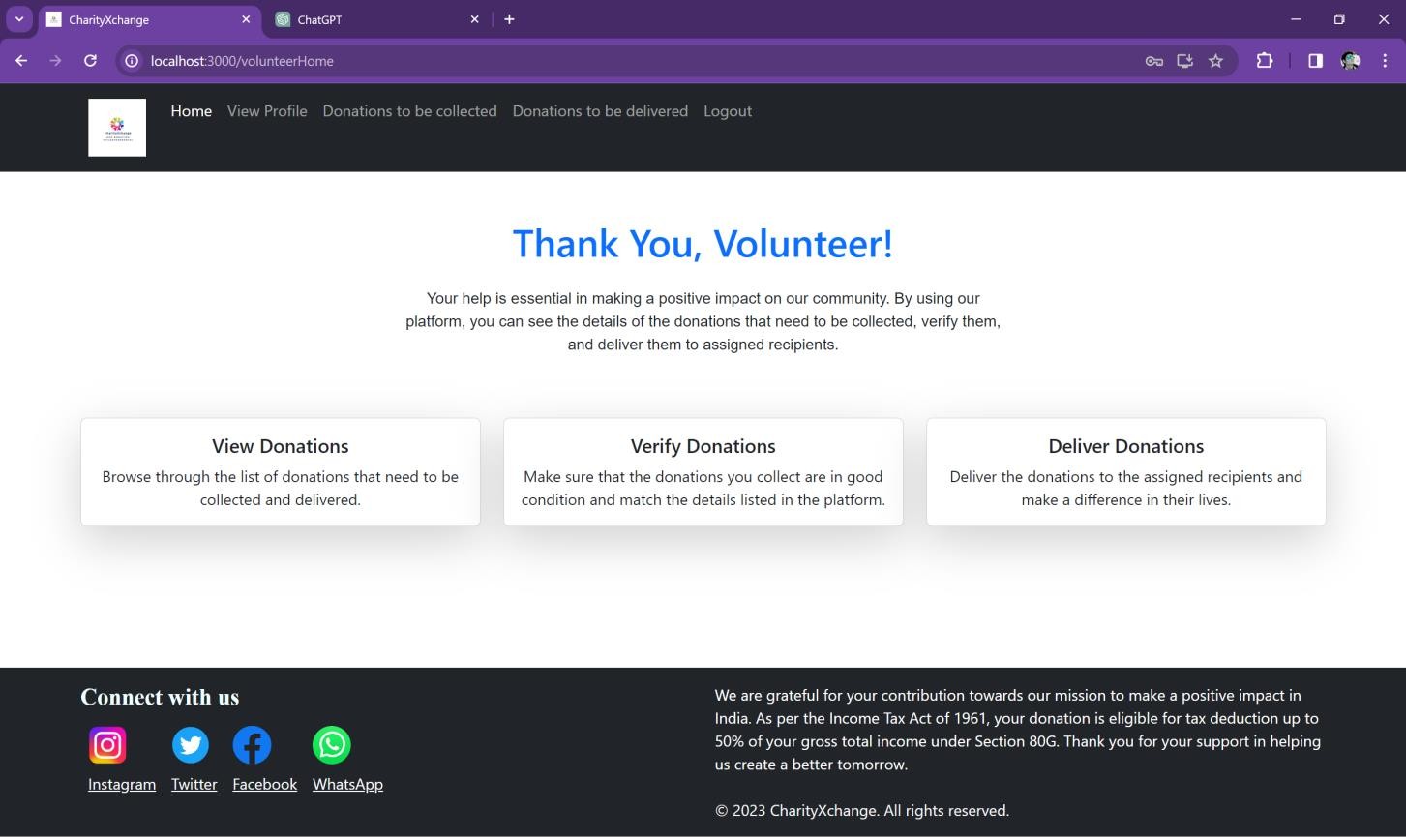


Recipient can request so admin can donate to him when he get donation from donor

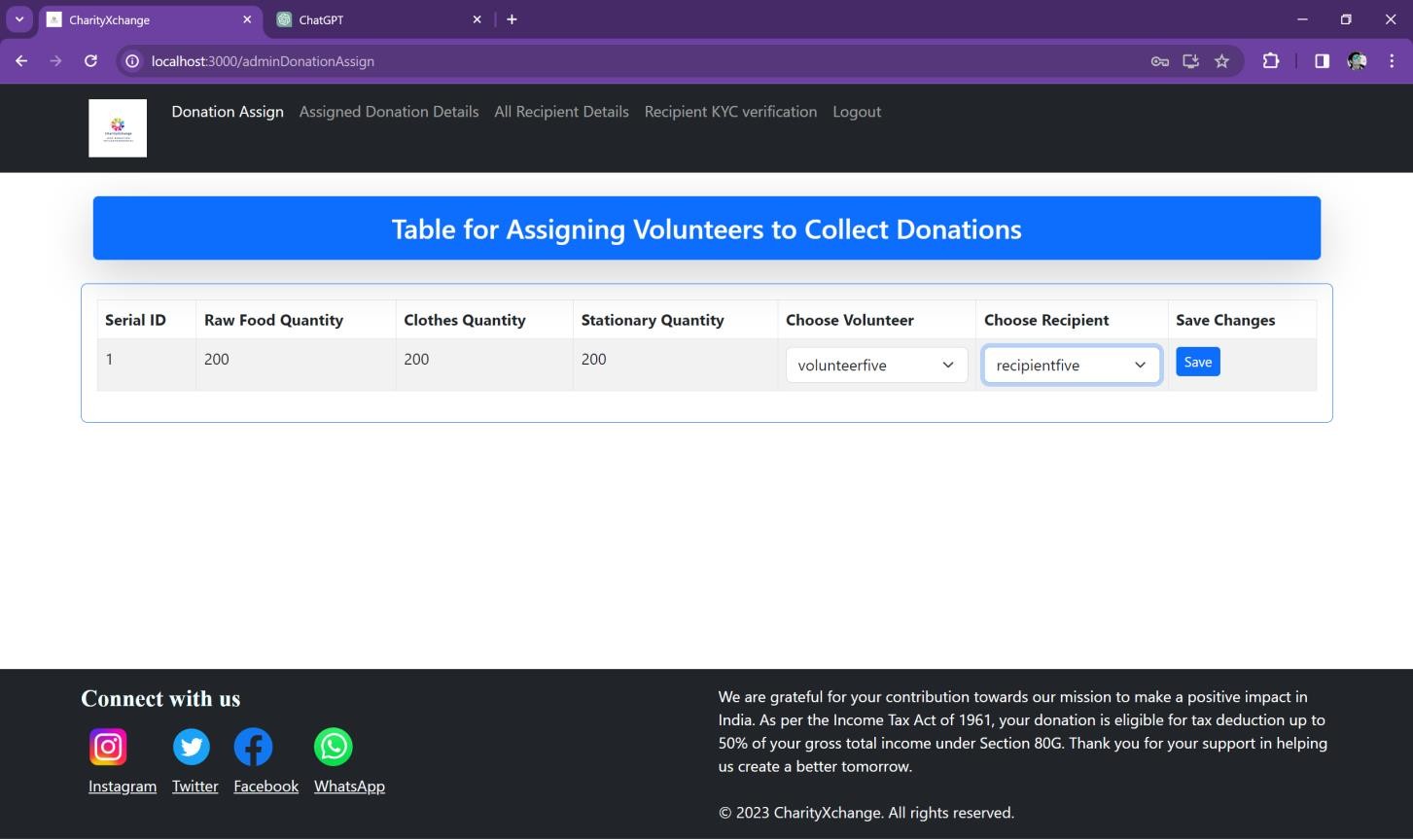




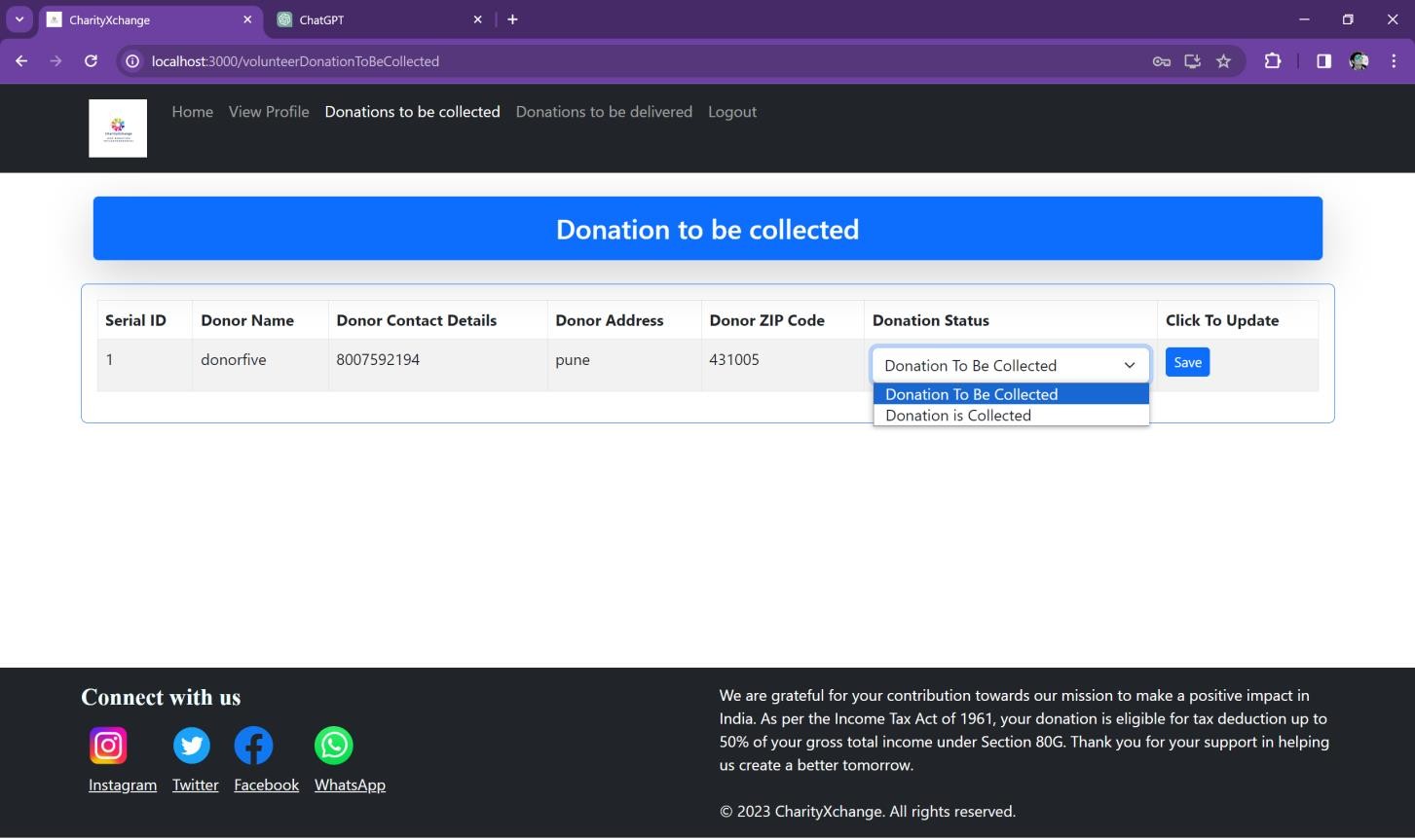
**Volunteer dashboard**



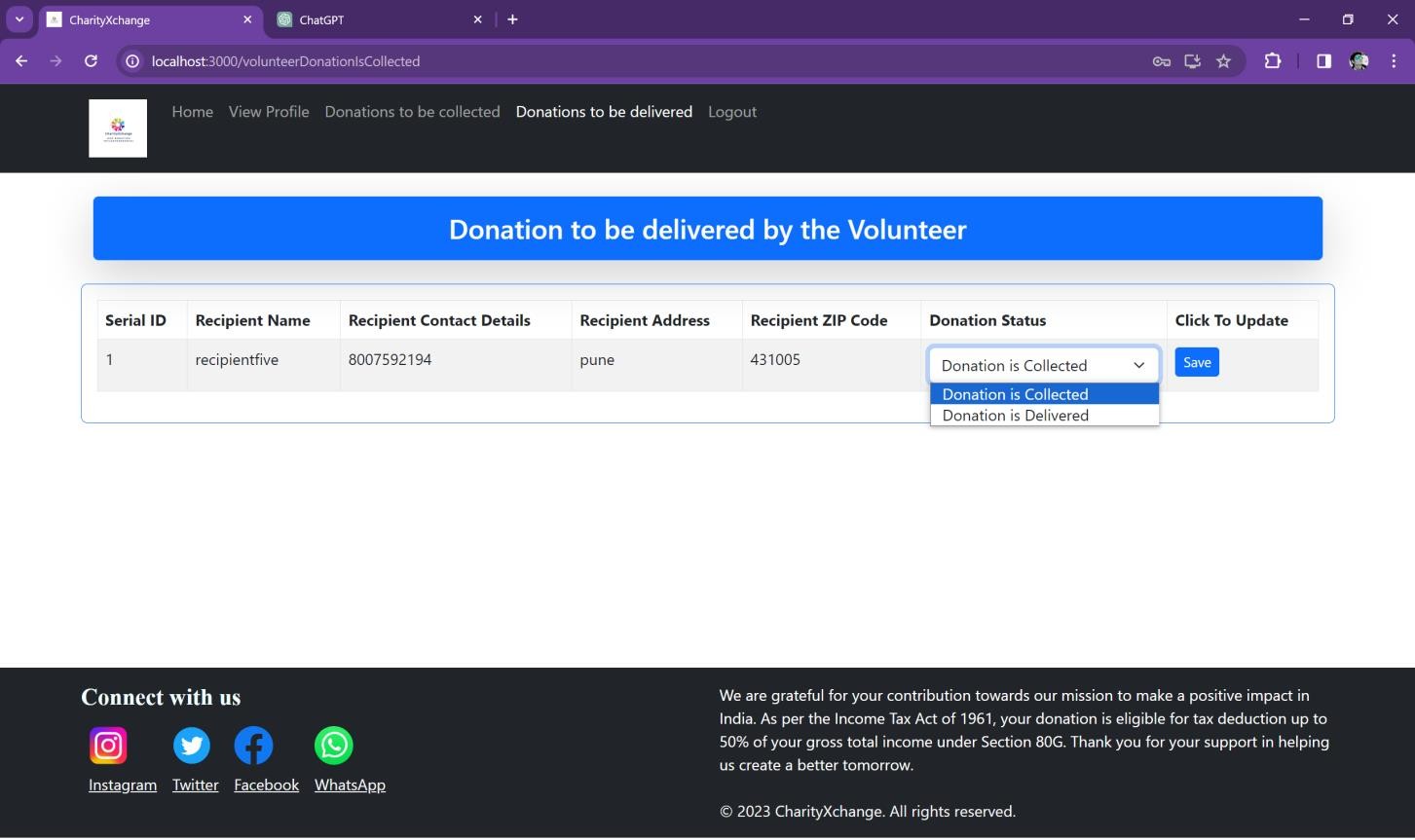
Admin now assign the volunteer



Volunteer go to donor and collect the donation



Volunteer go to recipient and donate



# TESTING

It is a process of ensuring acceptable degree of quality attributes of the software, of finding errors and its purpose is to ensure correctness, robustness, reliability and many other quality attributes.

## Testability

Software testability is the degree to which a software system or module supports testing in a given test context. This cannot be measured directly. A lower degree of testability results in increased test efforts.

## Observability

It refers to how easy it is to observe the behavior of a program in terms of its output, effect on the environment and other hardware and software components. In this the incorrect input and output is easily identified and verified.

## Decomposability

By controlling the scope of testing, problems can be isolated quickly and smarter testing can be performed. The software system is built from independent modules which can be tested independently.

## Stability

In this, fewer the changes, fewer disruptions to testing, that is changes to the software must be infrequent and in a controlled manner. It means that changes to the software do not invalidate existing tests in software testing, due to this the software recovers from failures in software testing.

## Testing Method Used

The Testing method which we have used for our project is black box testing. We used this method because in this approach to testing, the tests are derived on the basis of the requirements or specifications of the module and the internals of the module are not considered. This involves only observation of the output for some input values and there is no attempt to analyze the code which produces the output. The internal structure of the program

is ignored. The tester gives inputs to the system and checks the corresponding outputs. If outputs are not satisfactory, then test has successfully detected a problem with the software.

# FUTURE SCOPE

* **Enhanced Mobile Features:** Develop dedicated mobile apps for iOS and Android to offer a more optimized user experience on mobile devices.
* **Multi-Language Support:** Add support for multiple languages to cater to a broader audience across different regions in India.
* **Advanced KYC Verification System:**  Allow recipients to upload essential documents (e.g., government-issued ID, proof of address, and income certificate). Use AI to automatically verify these documents for authenticity and flag any discrepancies.

**8. CONCLUSION**

At the end it is conclude that, we have made effort on the following points.

* Made statement of the aim and objective of the project.
* We define the problem on which we are working in this project.
* We designed the simple user interface and resolved security issues of the system.
* Finally, system is implemented and tested accordingly to the test cases.
* At personal level we learned to work with team, implement software development technology and principles and good programing practice.

# APPENDIX

## Appendix I

* + GUI: Graphical User Interface
  + Platform dependent operating system which should be compatible with our software.
  + SQL: Structured Query Language

# 10. REFERENCE

* J2EE: The complete Reference, Published by McGraw Hill Education in 2002.
* Google for problem solving
* <https://www.w3schools.com/>
* <https://www.javatpoint.com/spring-boot-tutorial>
* <https://www.javatpoint.com/spring-mvc-tutorial>
* [http://www.google.com](http://www.google.com/)
* [https://www.getbootstrap.com](https://www.getbootstrap.com/)
* <https://khetibuddy.com/>
* <https://enam.gov.in/web/>