# Disaster Management System

SECTION: F

1-NAME: Manas Shetty

1-SRN: PES1UG22CS329

2-NAME: Mudar Pranav

2-SRN: PES1UG22CS362

## INTRODUCTION:

The **Disaster Management System (DMS)** facilitates effective management of disaster scenarios by enabling user registration, disaster reporting, volunteer application, and role-specific functionalities for admins, volunteers, and users. The system provides a centralized platform where NGOs can recruit and assign volunteers, users can apply as volunteers, and admins can manage disasters and approve/reject applications.
The DMS is built with **React** as the front end and **MySQL** as the back end. The application ensures secure login for all roles—users, volunteers, and admins. It is designed to be highly intuitive, ensuring role-specific access control and efficient data processing. Admins have complete control over disaster management, including adding, updating, or deleting disasters, while volunteers can browse disasters and apply to help.
Key features include:

- **User-Friendly Interface**: React ensures smooth interaction and responsive design for all user roles.
- **Role-Based Access Control**: Admins, volunteers, and users have specific access to different features.
- **Secure Data Storage**: All sensitive data is stored securely in a MySQL database.
- **Streamlined Workflow**: The system enables seamless coordination between volunteers and admins during disasters.
- **Performance**: The application ensures quick response times, even during high usage.

The system is secure, ensures efficient disaster handling, and promotes collaboration between all stakeholders.

## IMPLEMENTATOIN:

**Backend (MySQL):**

- **Database**: MySQL is used to store all critical data, such as disaster details, user and volunteer information, admin activities, and application statuses.
- **Data Links**: Tables are linked relationally to ensure data integrity. For example, disasters are linked to volunteers applying to help, and each application is tied to specific admins for approval.

- **Data Security**: MySQL authentication and proper indexing ensure efficient and secure data retrieval.

**Backend Logic (Node.js and Express):**

- **Main Routing**: Node.js (using Express) handles API requests between the front end and back end. Each endpoint corresponds to features like user registration, volunteer applications, and disaster management.
- **Session Management**: Session handling ensures that only authenticated users can access their respective dashboards.
- **Validation**: Input validation is performed server-side to ensure the integrity of data entered by users, volunteers, or admins.

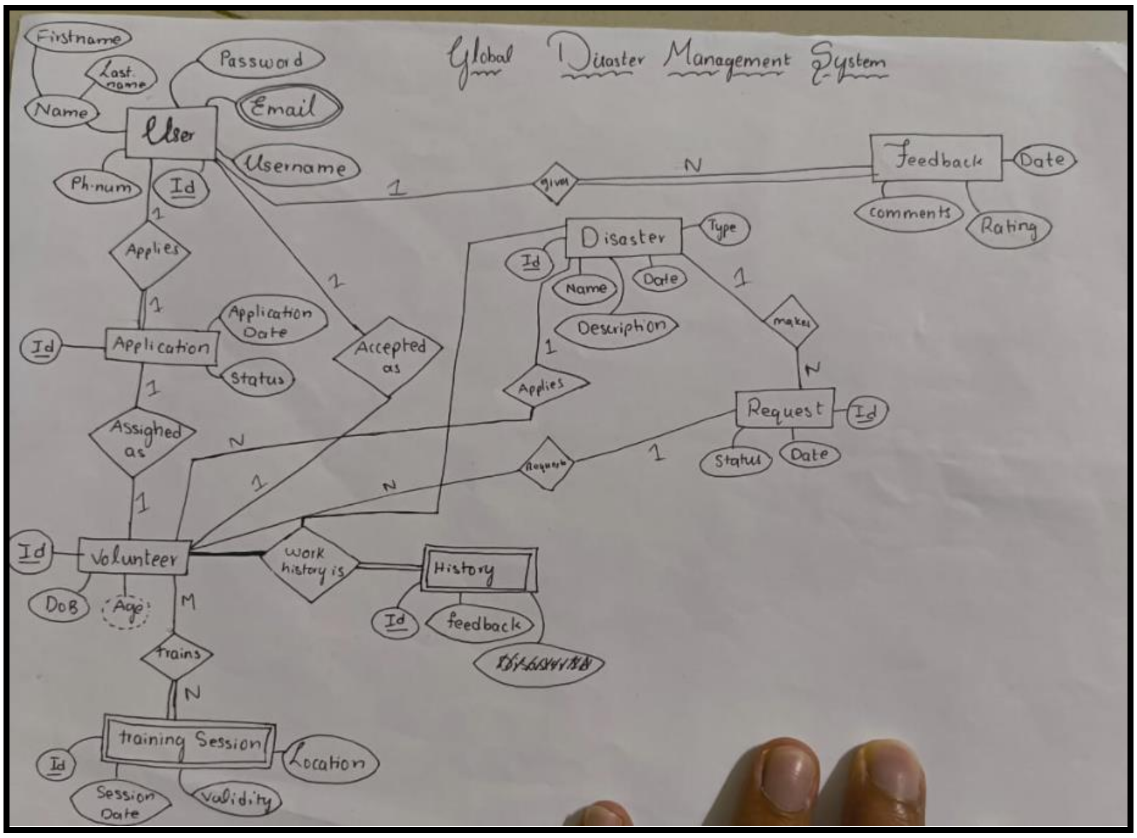
**Frontend (React):**

- **Templates**: React templates provide a dynamic and responsive user interface for all roles. It ensures smooth navigation across dashboards and workflows.
- **Role-Specific Dashboards**:
    - **Admins**: Features include disaster creation, updating disaster details, viewing volunteer applications, and approving/rejecting applications.
    - **Volunteers**: A dashboard for browsing disasters and applying to help.
    - **Users**: Basic access to the system to apply as volunteers.
- **Feedback Notifications**: Real-time updates are provided to users, such as success or error messages during actions like login, application submission, or approval.

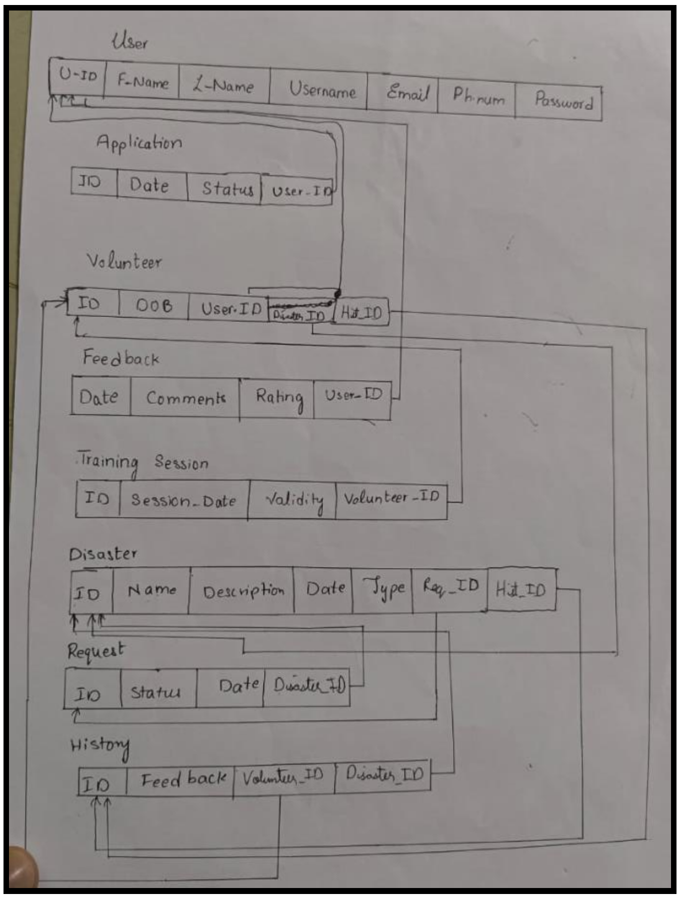
**Additional Features:**

- **Accessibility**: The interface is accessible across devices, ensuring inclusivity for all users.
- **Data Visualization**: Admin dashboards use charts and tables to visualize disaster reports and volunteer applications for easier decision-making.

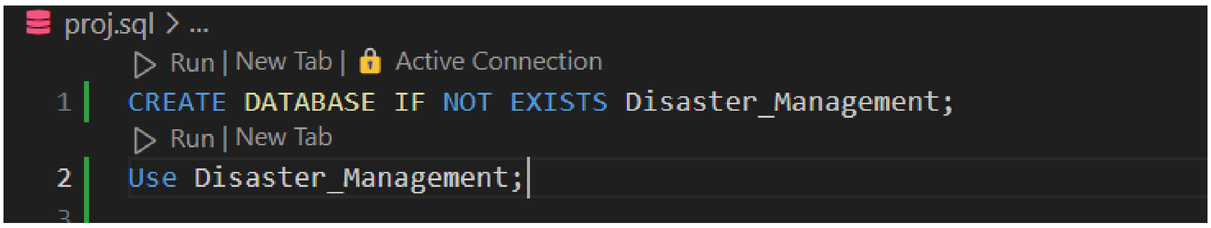
## ER DIAGRAM:



## RELATIONAL SCHEMA:


User
U-ID | F-Name | L-Name | Username | Email | Ph.num | Password
Application
ID | Date | Status | User-ID
Volunteer
ID | DOB | User-ID | Disaster_ID | Hist_ID
Feedback
Date | Comments | Rating | User-ID
Training Session
ID | Session_Date | Validity | Volunteer-ID
Disaster
ID | Name | Description | Date | Type | Req_ID | Hist_ID
Request
ID | Status | Date | Disaster_ID
History
ID | Feed back | Volunteer_ID | Disaster_ID

## DATABASE CREATION:



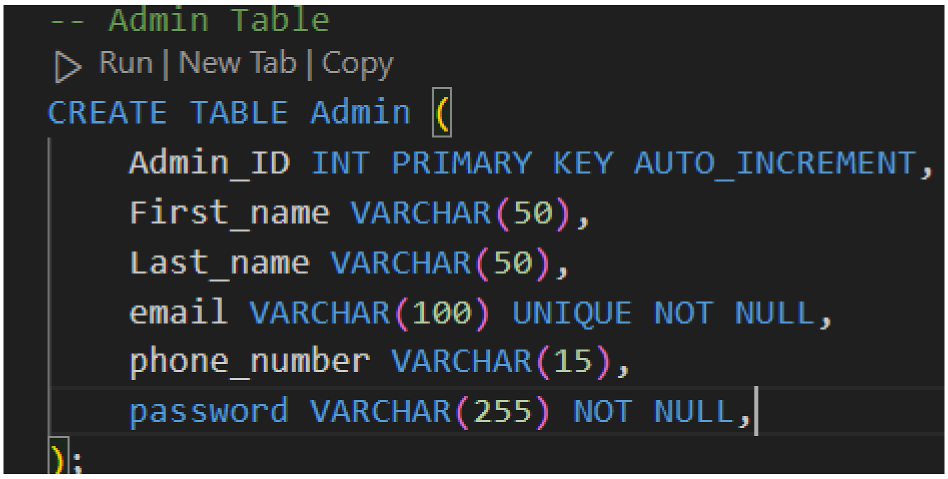
```sql
CREATE DATABASE IF NOT EXISTS Disaster_Management;
Use Disaster_Management;
```

## TABLES:

1)Admin Table:

-- Admin Table

```
CREATE TABLE Admin (
   Admin_ID INT PRIMARY KEY AUTO_INCREMENT,
   First_name VARCHAR(50),
   Last_name VARCHAR(50),
   email VARCHAR(100) UNIQUE NOT NULL,
   phone_number VARCHAR(15),
   password VARCHAR(255) NOT NULL,
);
```

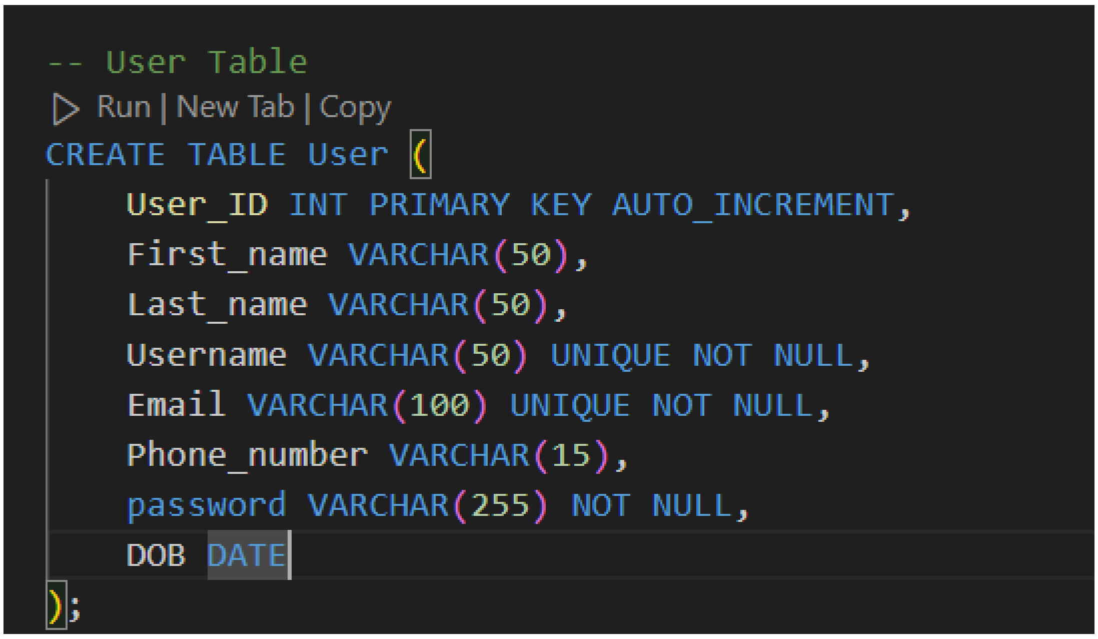


```
-- Admin Table
CREATE TABLE Admin (
    Admin_ID INT PRIMARY KEY AUTO_INCREMENT,
    First_name VARCHAR(50),
    Last_name VARCHAR(50),
    email VARCHAR(100) UNIQUE NOT NULL,
    phone_number VARCHAR(15),
    password VARCHAR(255) NOT NULL,
);
```

2) User Table:

```
-- User Table
CREATE TABLE User (
   User_ID INT PRIMARY KEY AUTO_INCREMENT,
   First_name VARCHAR(50),
   Last_name VARCHAR(50),
   Username VARCHAR(50) UNIQUE NOT NULL,
   Email VARCHAR(100) UNIQUE NOT NULL,
   Phone_number VARCHAR(15),
   password VARCHAR(255) NOT NULL,
   DOB DATE
```

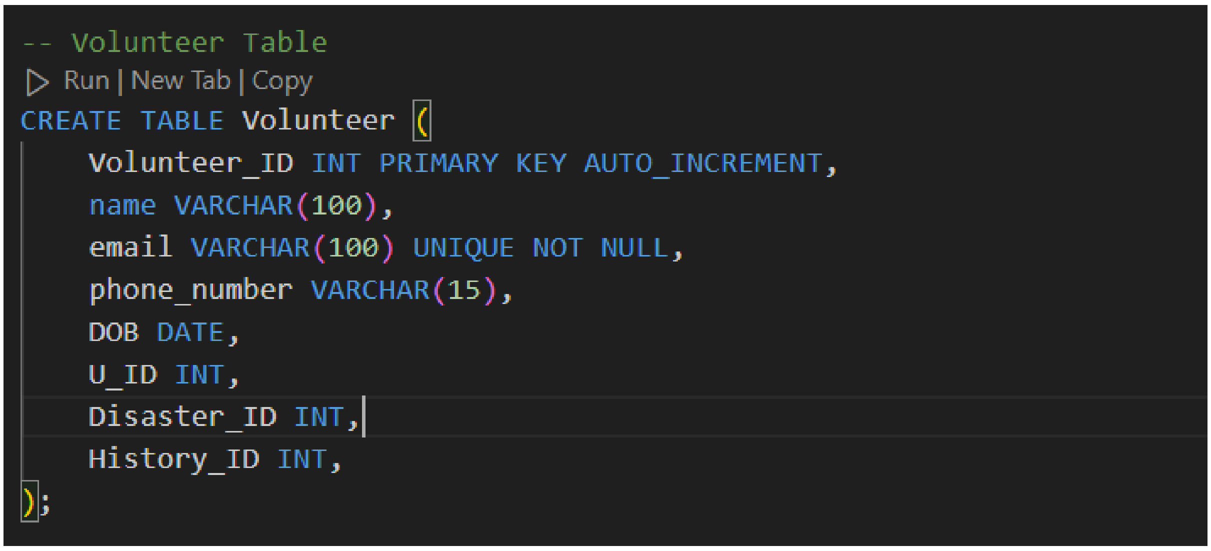
);



```
-- User Table
▷ Run | New Tab | Copy
CREATE TABLE User (
    User_ID INT PRIMARY KEY AUTO_INCREMENT,
    First_name VARCHAR(50),
    Last_name VARCHAR(50),
    Username VARCHAR(50) UNIQUE NOT NULL,
    Email VARCHAR(100) UNIQUE NOT NULL,
    Phone_number VARCHAR(15),
    password VARCHAR(255) NOT NULL,
    DOB DATE
);
```

3) Volunteer Table:

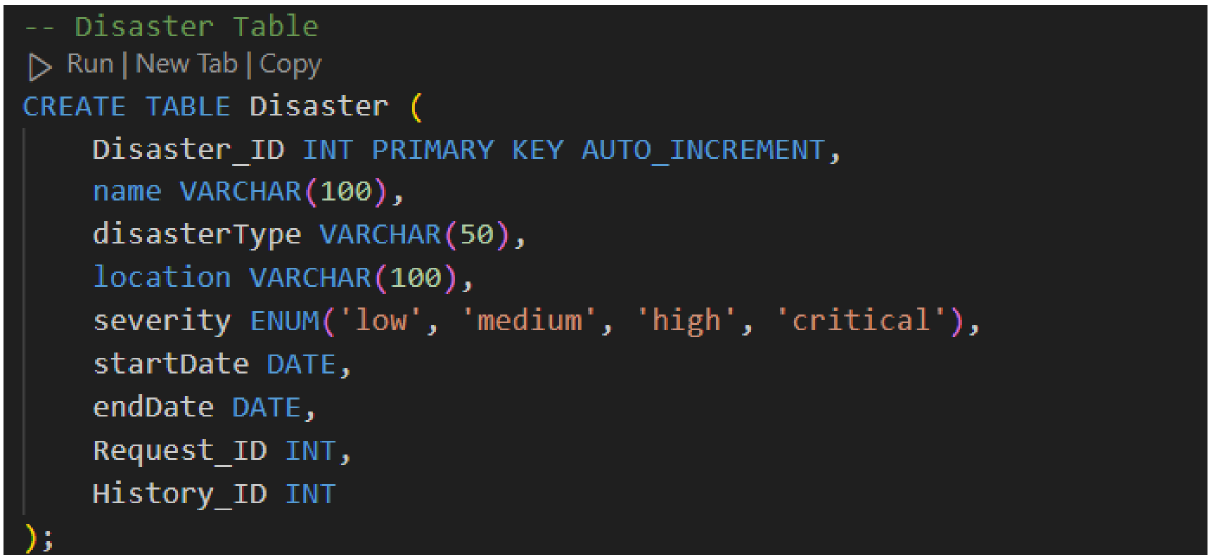
```
-- Volunteer Table
CREATE TABLE Volunteer (
   Volunteer_ID INT PRIMARY KEY AUTO_INCREMENT,
   name VARCHAR(100),
   email VARCHAR(100) UNIQUE NOT NULL,
   phone_number VARCHAR(15),
   DOB DATE,
   U_ID INT,
   Disaster_ID INT,
   History_ID INT,
);
```



```sql
-- Volunteer Table
▷ Run | New Tab | Copy
CREATE TABLE Volunteer (
    Volunteer_ID INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    email VARCHAR(100) UNIQUE NOT NULL,
    phone_number VARCHAR(15),
    DOB DATE,
    U_ID INT,
    Disaster_ID INT,
    History_ID INT,
);
```

4) Disaster Table

```sql
-- Disaster Table

CREATE TABLE Disaster (

   Disaster_ID INT PRIMARY KEY AUTO_INCREMENT,

   name VARCHAR(100),

   disasterType VARCHAR(50),

   location VARCHAR(100),

   severity ENUM('low', 'medium', 'high', 'critical'),

   startDate DATE,

   endDate DATE,

   Request_ID INT,

   History_ID INT

);
```


```
-- Disaster Table
▷ Run | New Tab | Copy
CREATE TABLE Disaster (
    Disaster_ID INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    disasterType VARCHAR(50),
    location VARCHAR(100),
    severity ENUM('low', 'medium', 'high', 'critical'),
    startDate DATE,
    endDate DATE,
    Request_ID INT,
    History_ID INT
);
```

5) Training Session Table:

CREATE TABLE Training_Session (

session_ID INT PRIMARY KEY AUTO_INCREMENT,

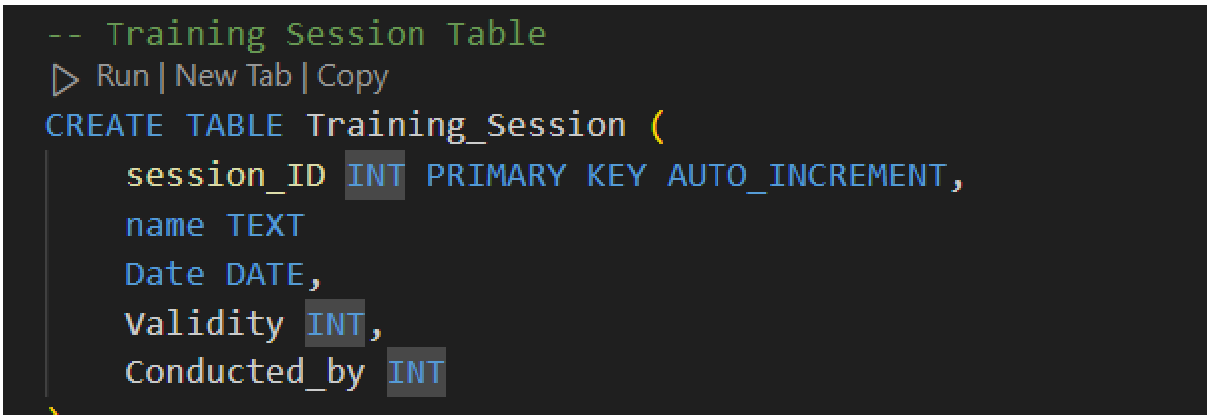
name TEXT

Date DATE,

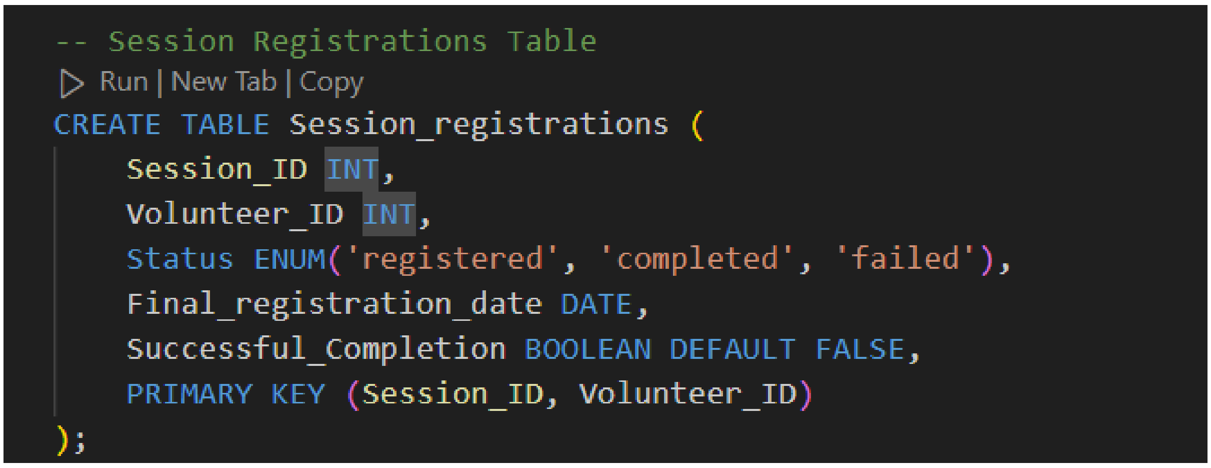
Validity INT,

Conducted_by INT

);

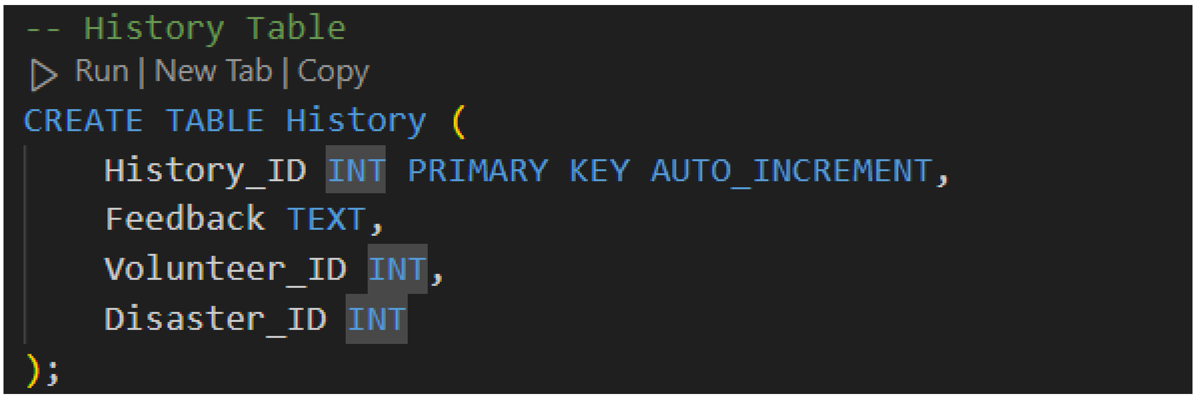

```
-- Training Session Table
▷ Run | New Tab | Copy
CREATE TABLE Training_Session (
    session_ID INT PRIMARY KEY AUTO_INCREMENT,
    name TEXT
    Date DATE,
    Validity INT,
    Conducted_by INT
```

6) Session_Registration Table:



```
-- Session Registrations Table
▷ Run | New Tab | Copy
CREATE TABLE Session_registrations (
    Session_ID INT,
    Volunteer_ID INT,
    Status ENUM('registered', 'completed', 'failed'),
    Final_registration_date DATE,
    Successful_Completion BOOLEAN DEFAULT FALSE,
    PRIMARY KEY (Session_ID, Volunteer_ID)
);
```

7) History Table:

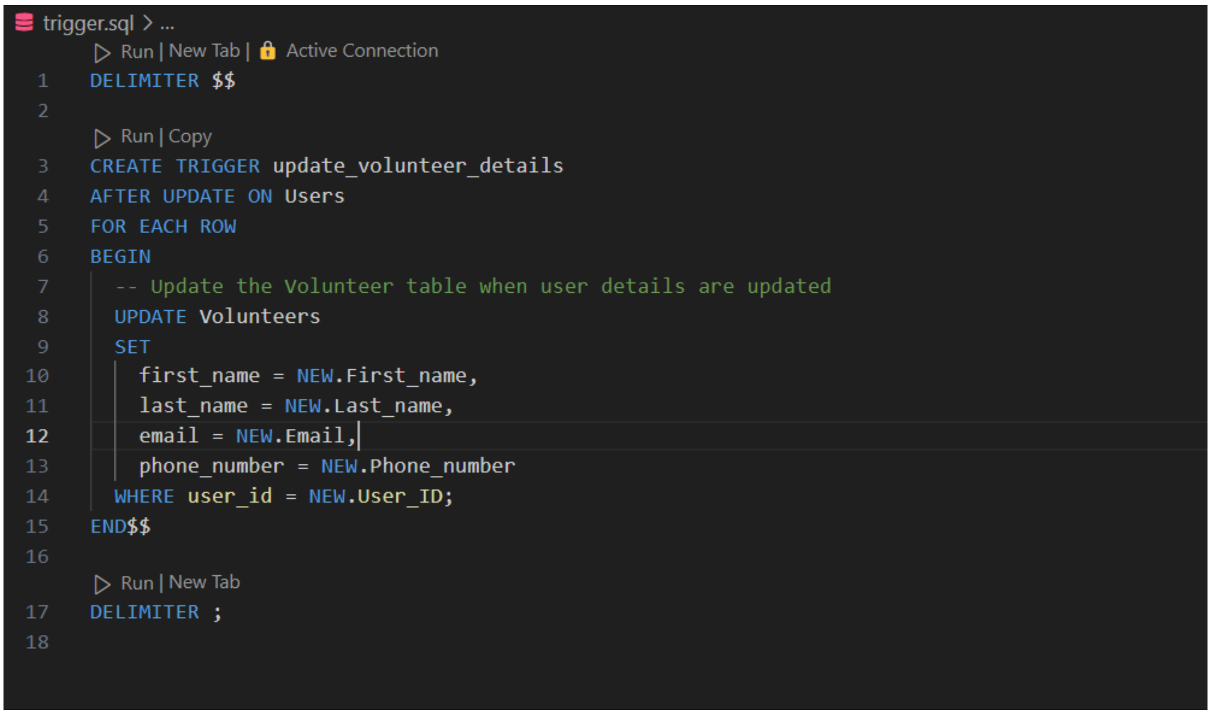


```
-- History Table
▷ Run | New Tab | Copy
CREATE TABLE History (
    History_ID INT PRIMARY KEY AUTO_INCREMENT,
    Feedback TEXT,
    Volunteer_ID INT,
    Disaster_ID INT
);
```

## TRIGGERS:

Our trigger performs a cascading update, when user details are updated, then volunteer details also get updated with the same

```
DELIMITER $$

CREATE TRIGGER update_volunteer_details
AFTER UPDATE ON Users
FOR EACH ROW
BEGIN
  -- Update the Volunteer table when user details are updated
  UPDATE Volunteers
  SET
    first_name = NEW.First_name,
    last_name = NEW.Last_name,
    email = NEW.Email,
    phone_number = NEW.Phone_number
  WHERE user_id = NEW.User_ID;
END$$

DELIMITER ;
```

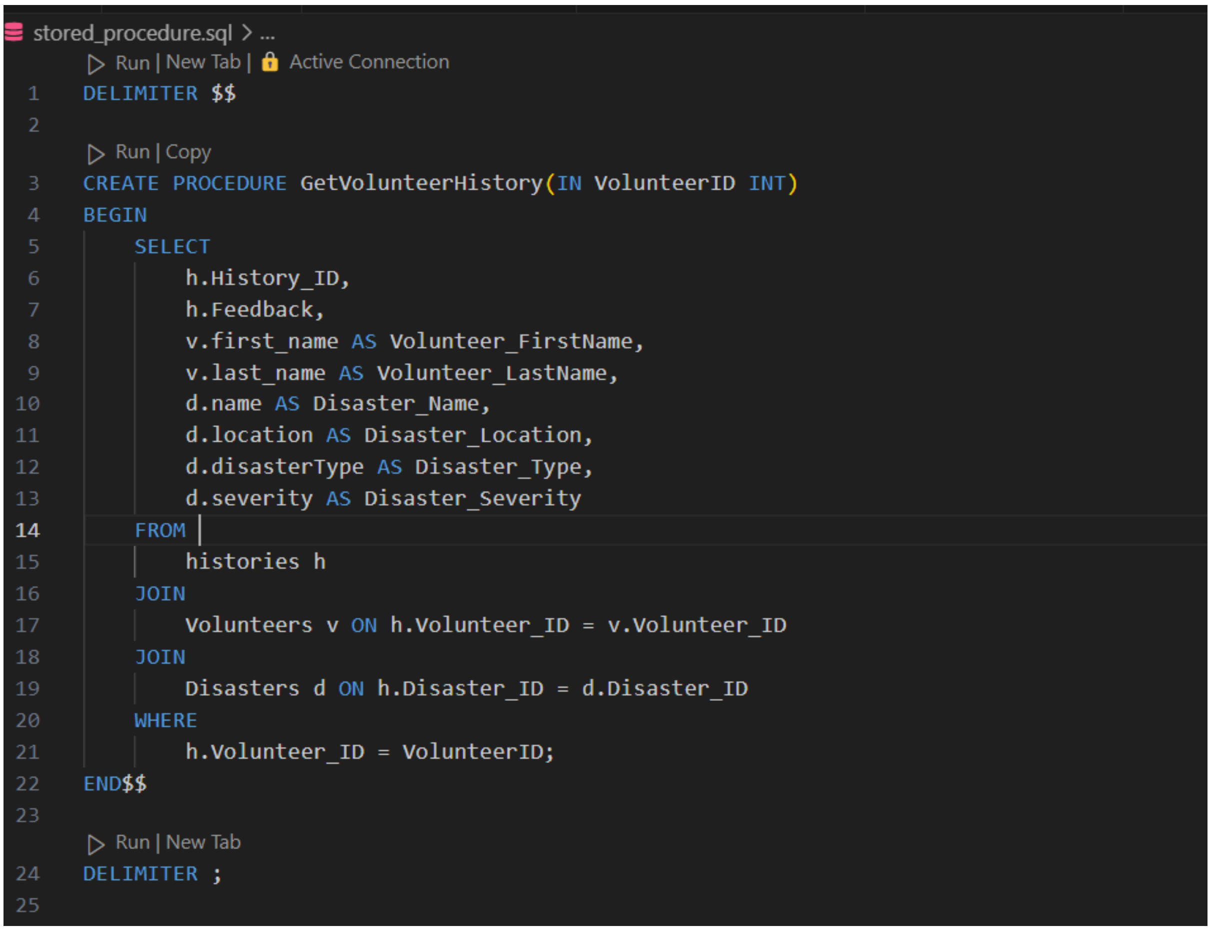


```sql
DELIMITER $$

CREATE TRIGGER update_volunteer_details
AFTER UPDATE ON Users
FOR EACH ROW
BEGIN
  -- Update the Volunteer table when user details are updated
  UPDATE Volunteers
  SET
    first_name = NEW.First_name,
    last_name = NEW.Last_name,
    email = NEW.Email,
    phone_number = NEW.Phone_number
  WHERE user_id = NEW.User_ID;
END$$

DELIMITER ;
```

## PROCEDURES:

Our stored procedure, does multiple table joins to fetch relevant volunteer history and displays it for the admin to review

```
DELIMITER $$

CREATE PROCEDURE GetVolunteerHistory(IN VolunteerID INT)
BEGIN
  SELECT
    h.History_ID,
    h.Feedback,
    v.first_name AS Volunteer_FirstName,
    v.last_name AS Volunteer_LastName,
    d.name AS Disaster_Name,
    d.location AS Disaster_Location,
    d.disasterType AS Disaster_Type,
    d.severity AS Disaster_Severity
  FROM
    histories h
  JOIN
    Volunteers v ON h.Volunteer_ID = v.Volunteer_ID
  JOIN
    Disasters d ON h.Disaster_ID = d.Disaster_ID
  WHERE
    h.Volunteer_ID = VolunteerID;
END$$

DELIMITER ;
```



stored_procedure.sql > ...

▷ Run | New Tab | 🔒 Active Connection

```sql
DELIMITER $$

CREATE PROCEDURE GetVolunteerHistory(IN VolunteerID INT)
BEGIN
    SELECT
        h.History_ID,
        h.Feedback,
        v.first_name AS Volunteer_FirstName,
        v.last_name AS Volunteer_LastName,
        d.name AS Disaster_Name,
        d.location AS Disaster_Location,
        d.disasterType AS Disaster_Type,
        d.severity AS Disaster_Severity
    FROM
        histories h
    JOIN
        Volunteers v ON h.Volunteer_ID = v.Volunteer_ID
    JOIN
        Disasters d ON h.Disaster_ID = d.Disaster_ID
    WHERE
        h.Volunteer_ID = VolunteerID;
END$$

DELIMITER ;
```

## QUERIES:

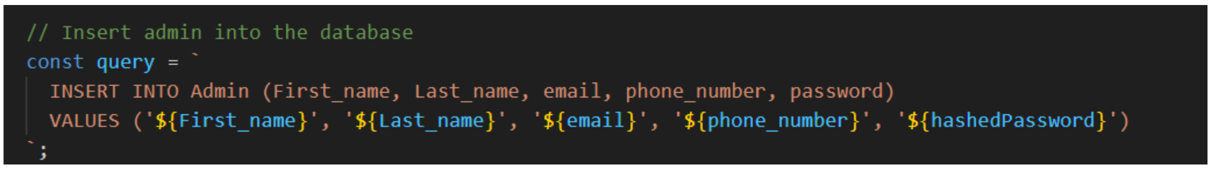
1)

// Insert admin into the database

const query = `

INSERT INTO Admin (First_name, Last_name, email, phone_number, password)

VALUES ('${First_name}', '${Last_name}', '${email}', '${phone_number}', '${hashedPassword}')



```javascript
// Insert admin into the database
const query = `
  INSERT INTO Admin (First_name, Last_name, email, phone_number, password)
  VALUES ('${First_name}', '${Last_name}', '${email}', '${phone_number}', '${hashedPassword}')
`;
```

2)

```
const insertQuery = `
    INSERT INTO User (First_name, Last_name, Username, Email, Phone_number, password, DOB)
    VALUES ('${first_name}', '${last_name}', '${username}', '${email}', '${phone_number}', '${hashedPassword}', '${dob}')
   `;
```

```
// Insert new user
const insertQuery = `
  INSERT INTO User (First_name, Last_name, Username, Email, Phone_number, password, DOB)
  VALUES ('${first_name}', '${last_name}', '${username}', '${email}', '${phone_number}', '${hashedPassword}', '${dob}')
`;
```

3)

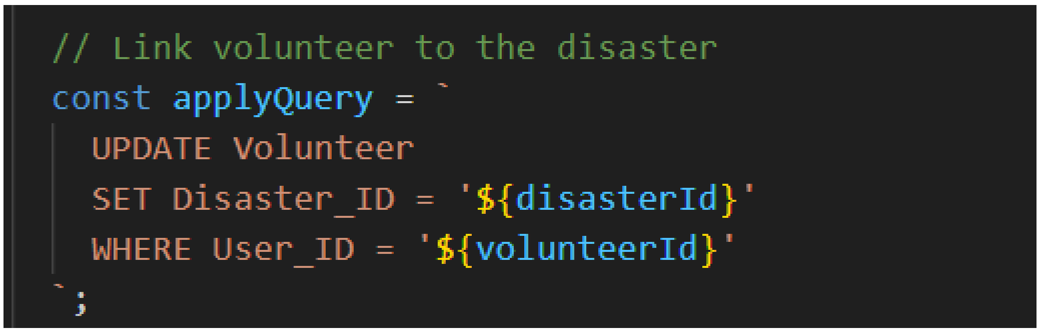
```
// Update user data
  const updateQuery = `
   UPDATE User
   SET First_name = '${First_name}', Last_name = '${Last_name}', Username = '${Username}', Email = '${Email}', Phone_number = '${Phone_number}'
   WHERE User_ID = '${userId}'
  `;
```

```
try {
  // Update user data
  const updateQuery = `
    UPDATE User
    SET First_name = '${First_name}', Last_name = '${Last_name}', Username = '${Username}', Email = '${Email}', Phone_number = '${P
    WHERE User_ID = '${userId}'
  `;
```

4)

```
 // Link volunteer to the disaster
    const applyQuery = `
     UPDATE Volunteer
     SET Disaster_ID = '${disasterId}'
```

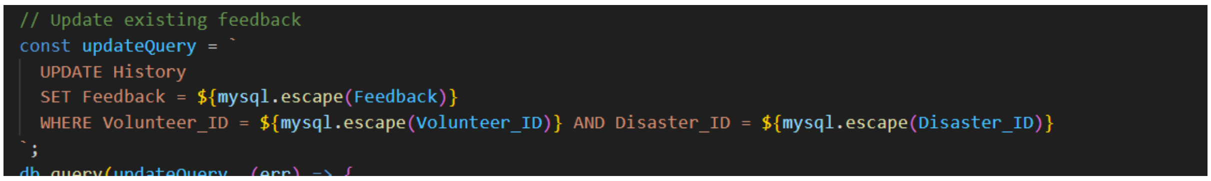
```
        WHERE User_ID = '${volunteerId}'
      `;
```



```
// Link volunteer to the disaster
const applyQuery = `
  UPDATE Volunteer
  SET Disaster_ID = '${disasterId}'
  WHERE User_ID = '${volunteerId}'
`;
```

5)

```
// Update existing feedback
         const updateQuery = `
           UPDATE History
           SET Feedback = ${mysql.escape(Feedback)}
           WHERE Volunteer_ID = ${mysql.escape(Volunteer_ID)} AND Disaster_ID =
${mysql.escape(Disaster_ID)}
         `;
```

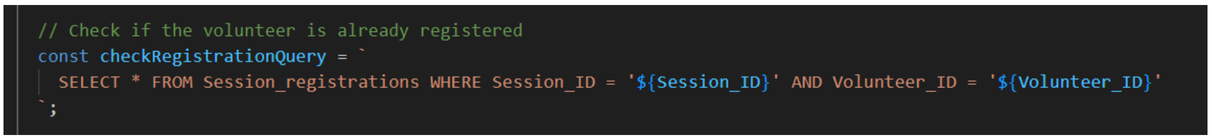


```
// Update existing feedback
const updateQuery = `
  UPDATE History
  SET Feedback = ${mysql.escape(Feedback)}
  WHERE Volunteer_ID = ${mysql.escape(Volunteer_ID)} AND Disaster_ID = ${mysql.escape(Disaster_ID)}
`;
db.query(updateQuery, (err) => {
```

6)

```
// Check if the volunteer is already registered
    const checkRegistrationQuery = `
      SELECT * FROM Session_registrations WHERE Session_ID = '${Session_ID}' AND
Volunteer_ID = '${Volunteer_ID}'
```

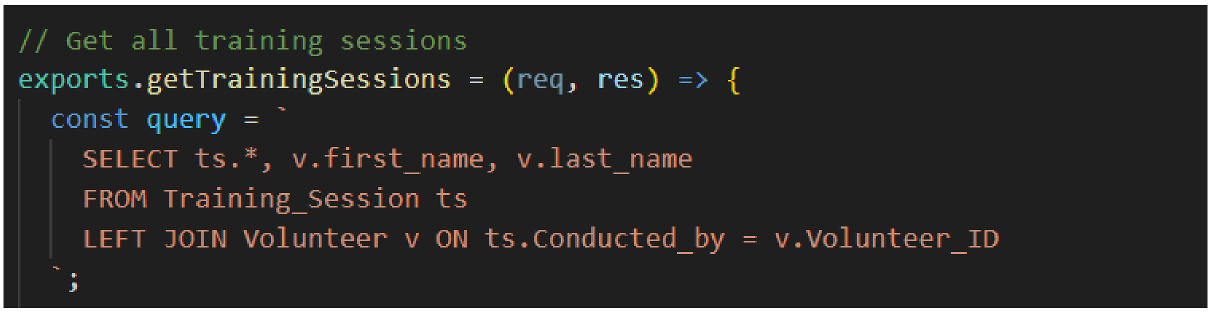
```
    `;
```



```
// Check if the volunteer is already registered
const checkRegistrationQuery = `
  SELECT * FROM Session_registrations WHERE Session_ID = '${Session_ID}' AND Volunteer_ID = '${Volunteer_ID}'
`;
```

7)

```
// Get all training sessions
exports.getTrainingSessions = (req, res) => {
  const query = `
    SELECT ts.*, v.first_name, v.last_name
    FROM Training_Session ts
    LEFT JOIN Volunteer v ON ts.Conducted_by = v.Volunteer_ID
  `;
```

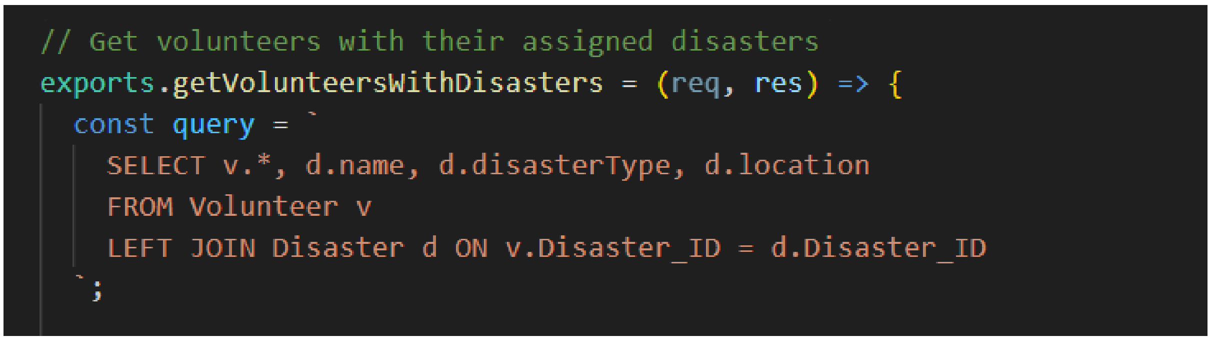


```
// Get all training sessions
exports.getTrainingSessions = (req, res) => {
  const query = `
    SELECT ts.*, v.first_name, v.last_name
    FROM Training_Session ts
    LEFT JOIN Volunteer v ON ts.Conducted_by = v.Volunteer_ID
  `;
```

8)

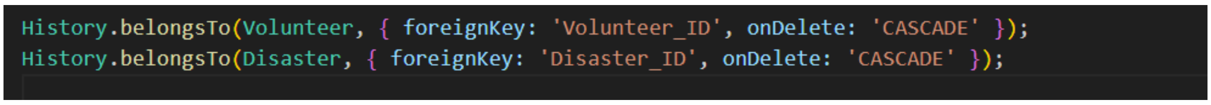
```
// Get volunteers with their assigned disasters
exports.getVolunteersWithDisasters = (req, res) => {
  const query = `
    SELECT v.*, d.name, d.disasterType, d.location
    FROM Volunteer v
    LEFT JOIN Disaster d ON v.Disaster_ID = d.Disaster_ID
```

`;

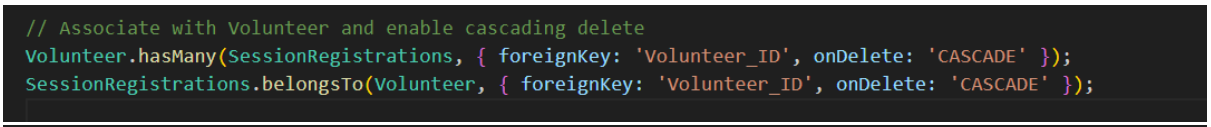


```
// Get volunteers with their assigned disasters
exports.getVolunteersWithDisasters = (req, res) => {
  const query = `
    SELECT v.*, d.name, d.disasterType, d.location
    FROM Volunteer v
    LEFT JOIN Disaster d ON v.Disaster_ID = d.Disaster_ID
  `;
```

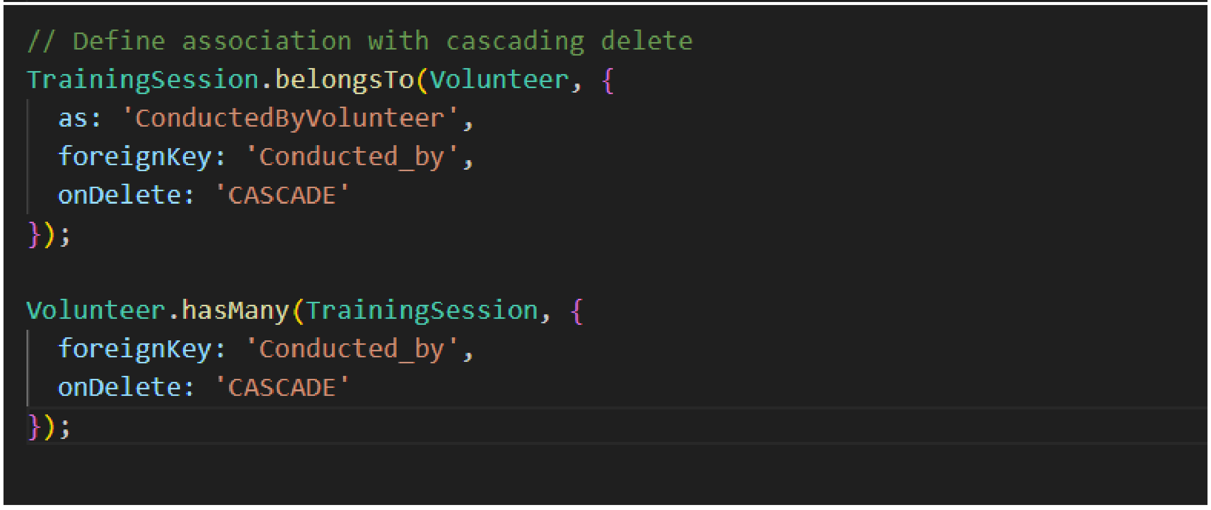
## RELATIONSHIP TABLES:



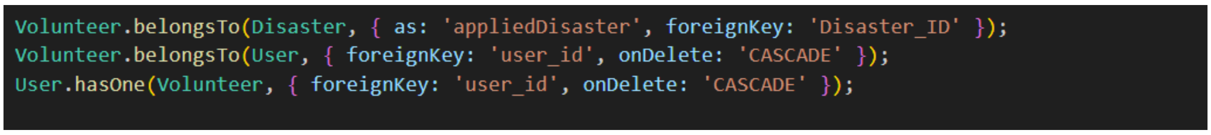
```
History.belongsTo(Volunteer, { foreignKey: 'Volunteer_ID', onDelete: 'CASCADE' });
History.belongsTo(Disaster, { foreignKey: 'Disaster_ID', onDelete: 'CASCADE' });
```



```
// Associate with Volunteer and enable cascading delete
Volunteer.hasMany(SessionRegistrations, { foreignKey: 'Volunteer_ID', onDelete: 'CASCADE' });
SessionRegistrations.belongsTo(Volunteer, { foreignKey: 'Volunteer_ID', onDelete: 'CASCADE' });
```



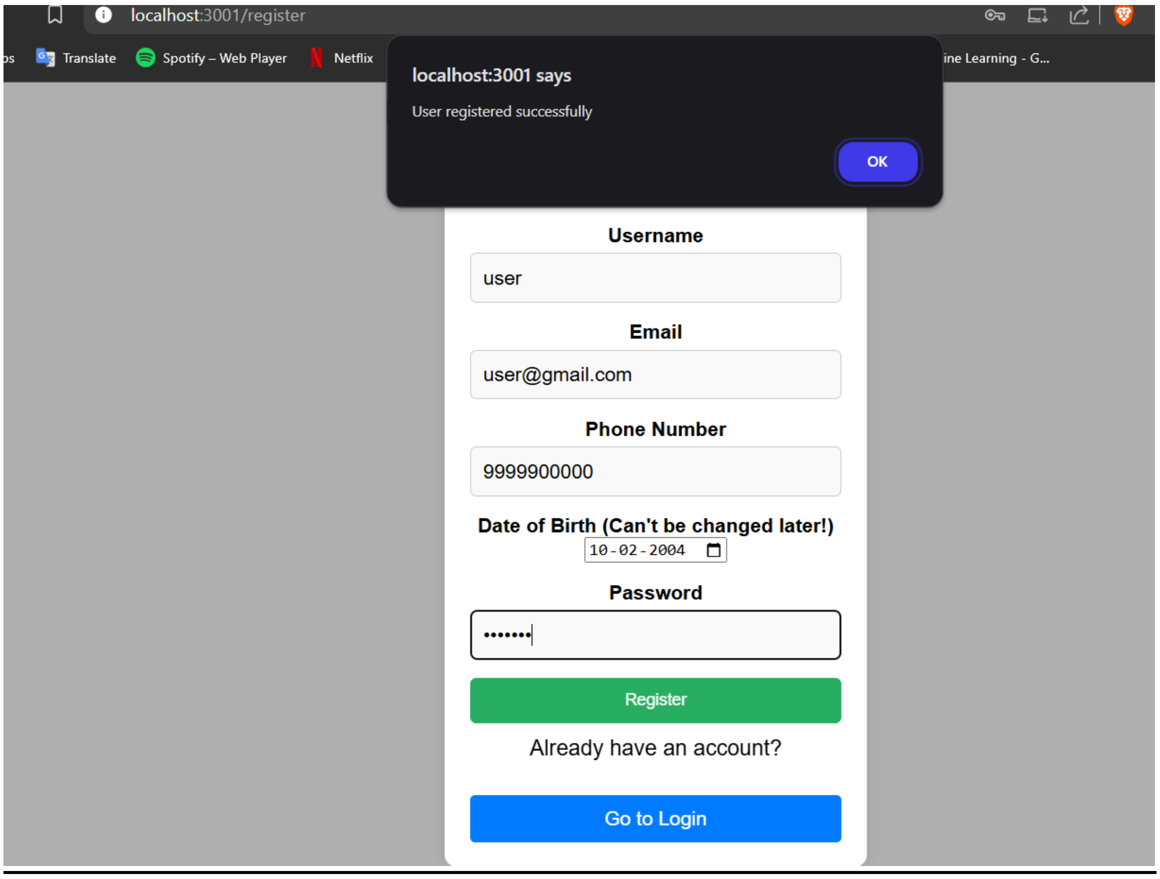
```
// Define association with cascading delete
TrainingSession.belongsTo(Volunteer, {
  as: 'ConductedByVolunteer',
  foreignKey: 'Conducted_by',
  onDelete: 'CASCADE'
});

Volunteer.hasMany(TrainingSession, {
  foreignKey: 'Conducted_by',
  onDelete: 'CASCADE'
});
```



```
Volunteer.belongsTo(Disaster, { as: 'appliedDisaster', foreignKey: 'Disaster_ID' });
Volunteer.belongsTo(User, { foreignKey: 'user_id', onDelete: 'CASCADE' });
User.hasOne(Volunteer, { foreignKey: 'user_id', onDelete: 'CASCADE' });
```

## HTML PART:

### 1) **Registration page:**



localhost:3001 says

User registered successfully

Username

user

Email

user@gmail.com

Phone Number

9999900000

Date of Birth (Can't be changed later!)

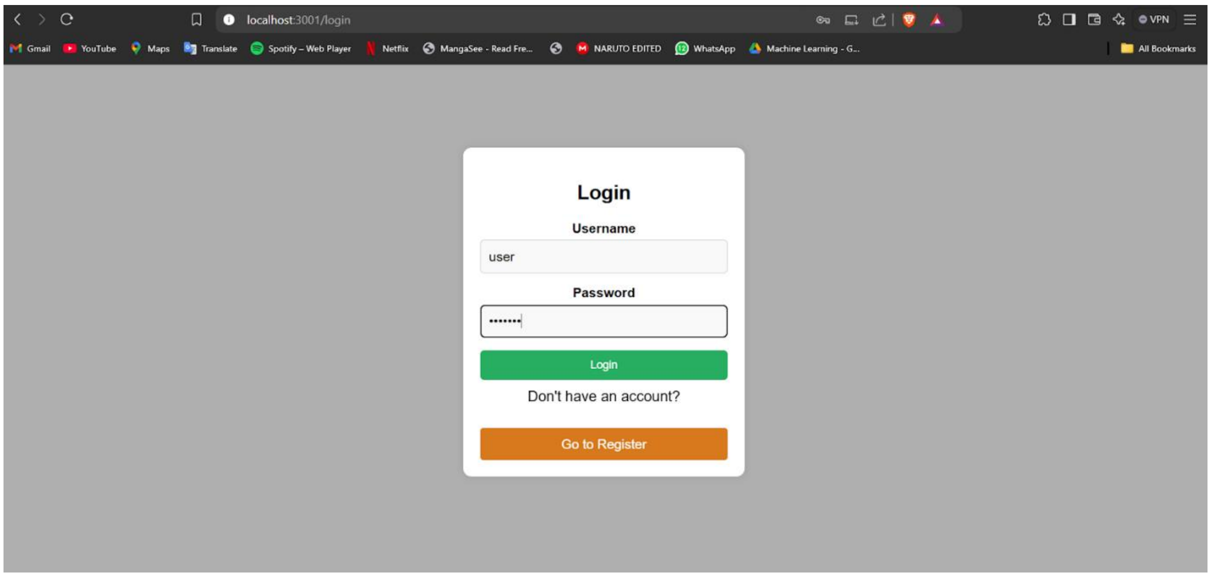
10-02-2004

Password

•••••••

Already have an account?

### **Login page:**


localhost:3001/login

Login

Username

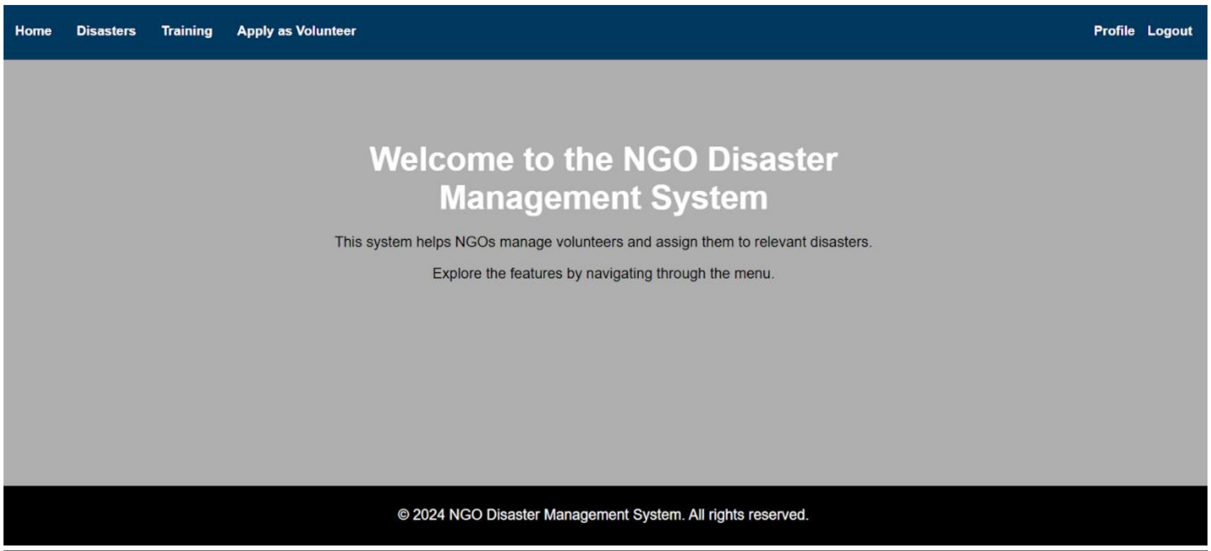
user

Password

Login

Don't have an account?

Go to Register



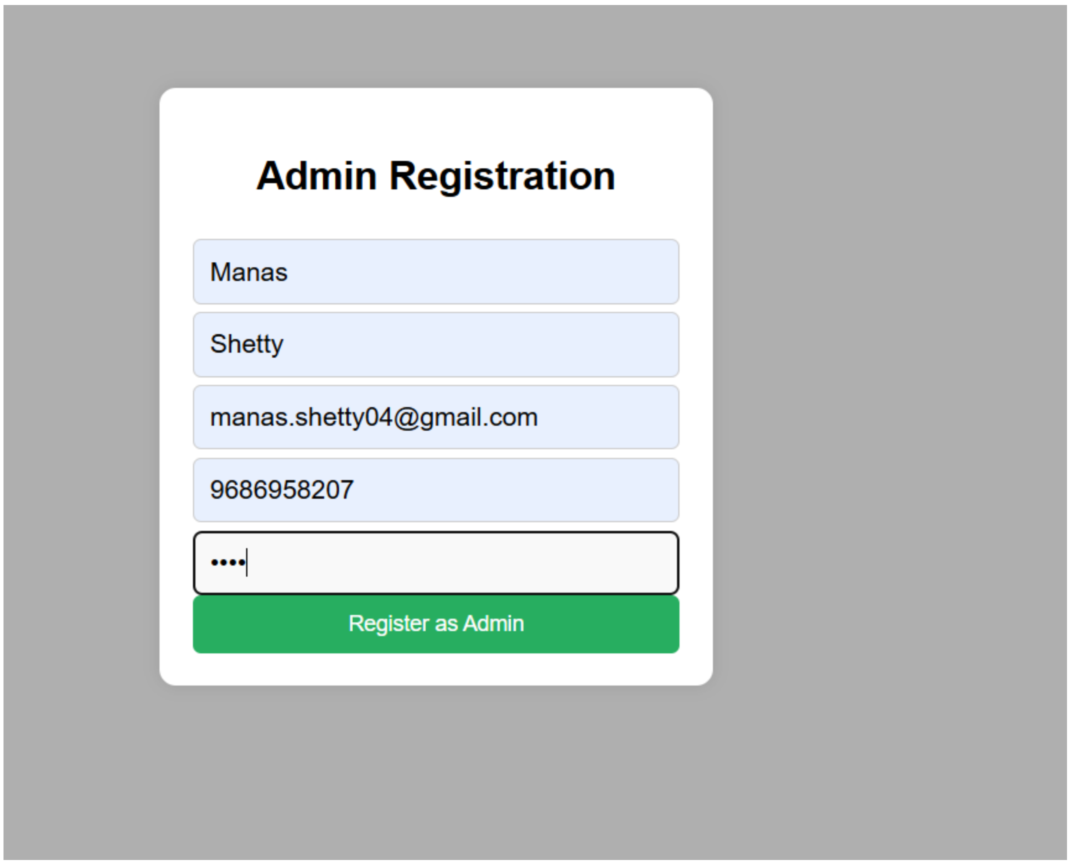
Welcome to the NGO Disaster Management System

This system helps NGOs manage volunteers and assign them to relevant disasters.

Explore the features by navigating through the menu.

© 2024 NGO Disaster Management System. All rights reserved.

## Admin registration and login:



Admin Registration

Manas

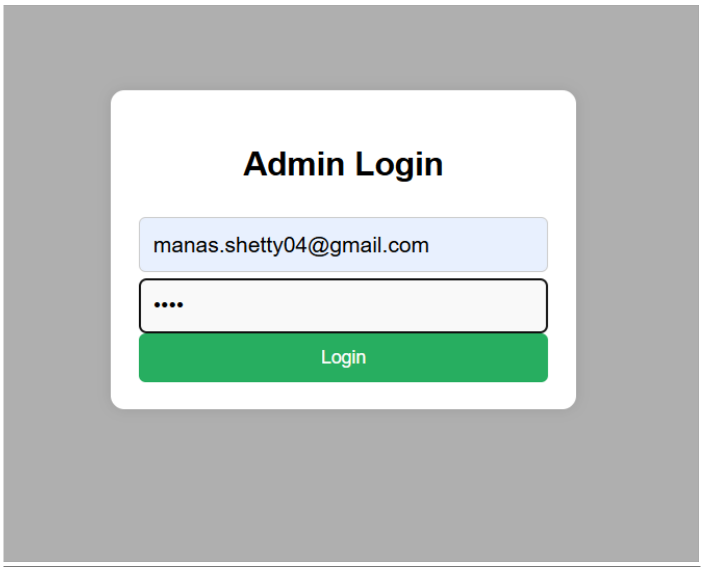
Shetty

manas.shetty04@gmail.com

9686958207

••••

Register as Admin



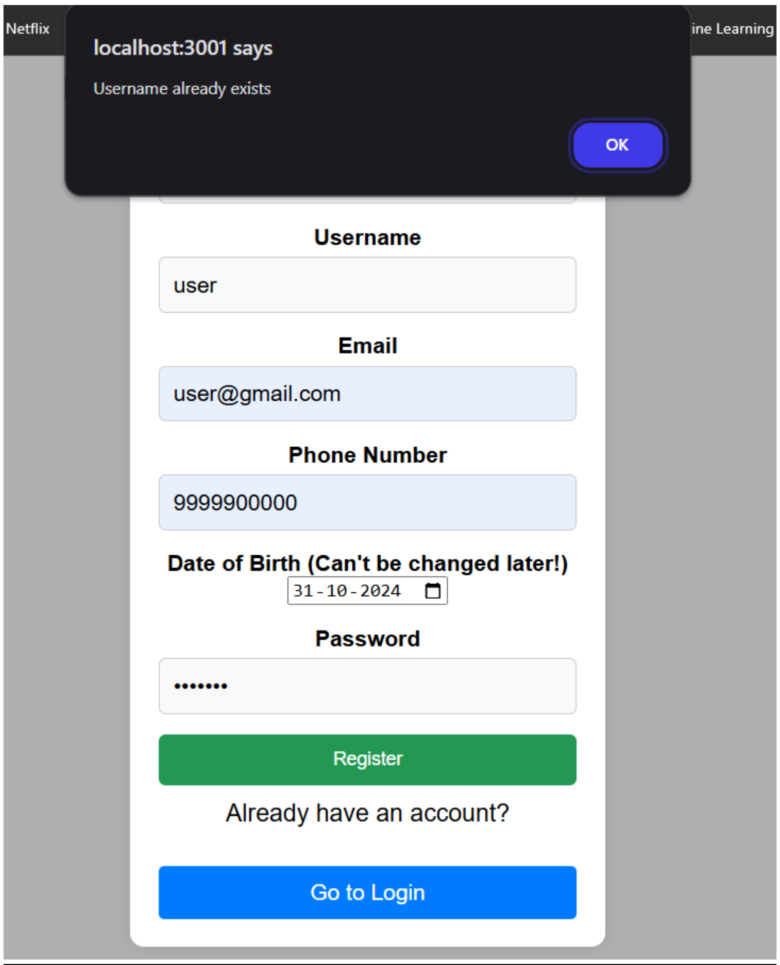
Admin Login

manas.shetty04@gmail.com

••••

Login

## **Existing user:**


Netflix

ine Learning

localhost:3001 says

Username already exists

OK

Username

user

Email

user@gmail.com

Phone Number

9999900000

Date of Birth (Can't be changed later!)

31-10-2024

Password

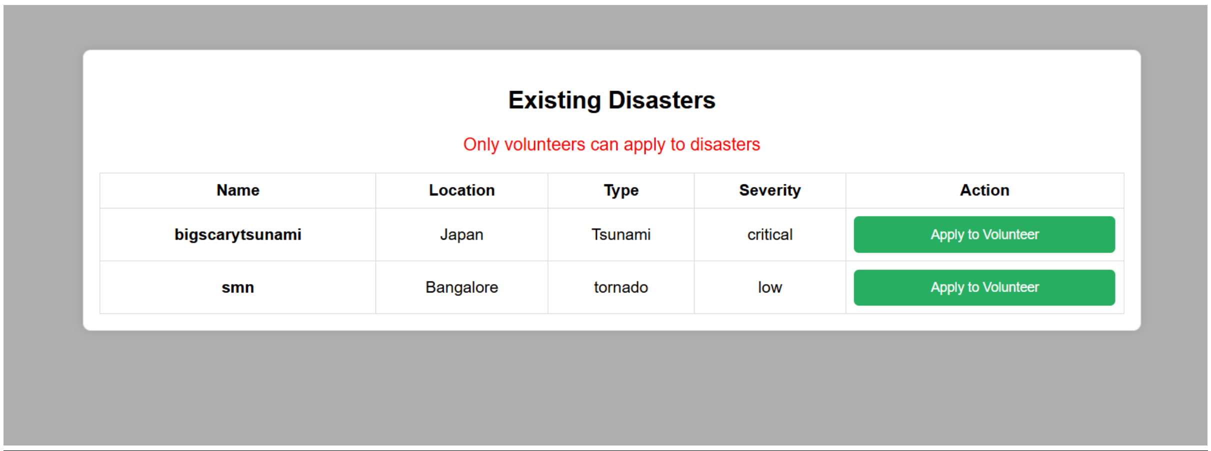
•••••••

Register

Already have an account?

Go to Login

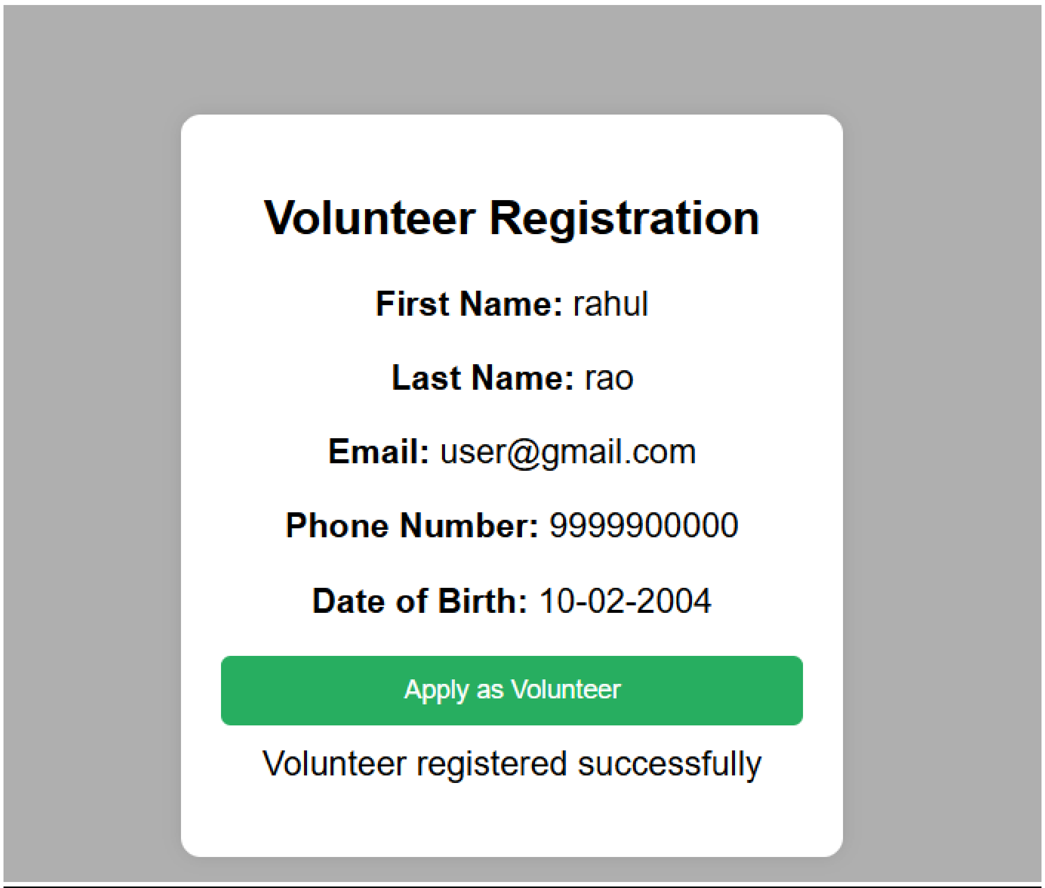
## User viewing disasters:


Existing Disasters

Only volunteers can apply to disasters

| Name | Location | Type | Severity | Action |
|---|---|---|---|---|
| bigscarytsunami | Japan | Tsunami | critical | Apply to Volunteer |
| smn | Bangalore | tornado | low | Apply to Volunteer |

## Registering as Volunteer:



## Volunteer Registration

**First Name:** rahul

**Last Name:** rao

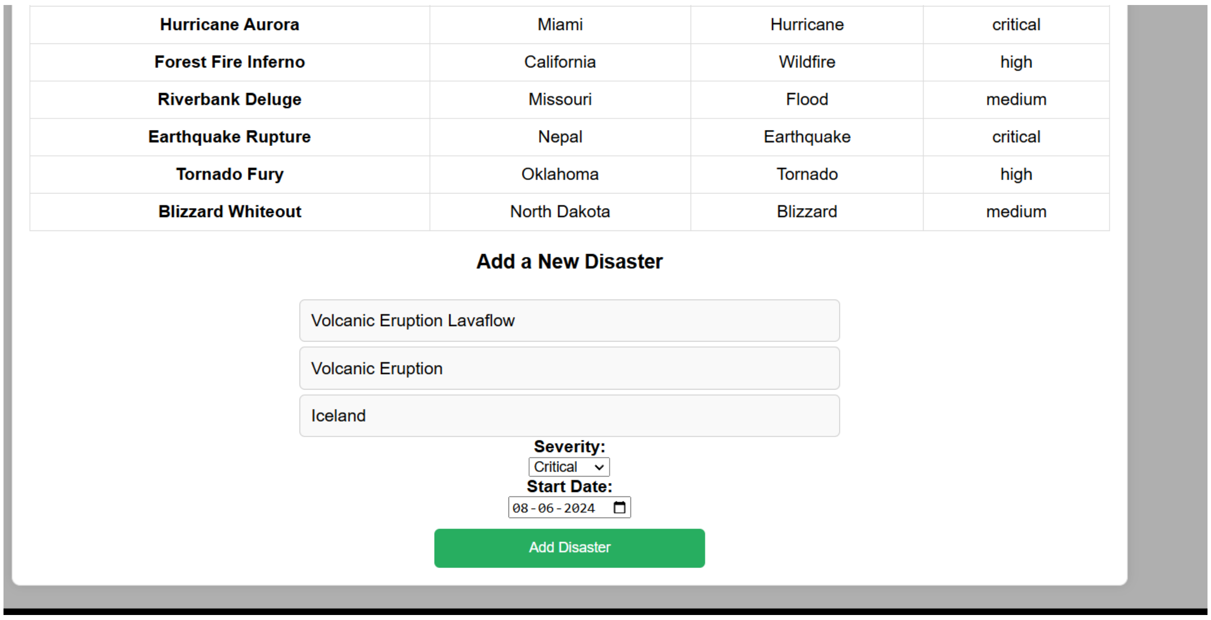
**Email:** user@gmail.com

**Phone Number:** 9999900000

**Date of Birth:** 10-02-2004

Volunteer registered successfully

## Adding Disaster as Admin:



| | | | |
|---|---|---|---|
| **Hurricane Aurora** | Miami | Hurricane | critical |
| **Forest Fire Inferno** | California | Wildfire | high |
| **Riverbank Deluge** | Missouri | Flood | medium |
| **Earthquake Rupture** | Nepal | Earthquake | critical |
| **Tornado Fury** | Oklahoma | Tornado | high |
| **Blizzard Whiteout** | North Dakota | Blizzard | medium |

**Add a New Disaster**

Volcanic Eruption Lavaflow

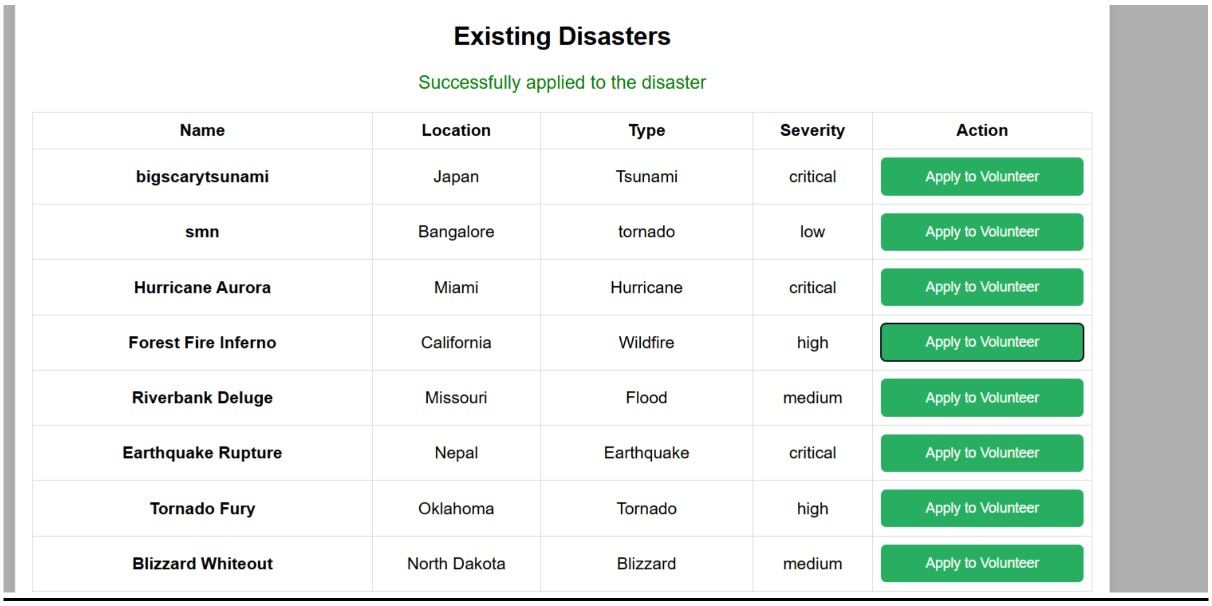
Volcanic Eruption

Iceland

**Severity:** Critical

**Start Date:** 08-06-2024

## Volunteers applying for a disaster:

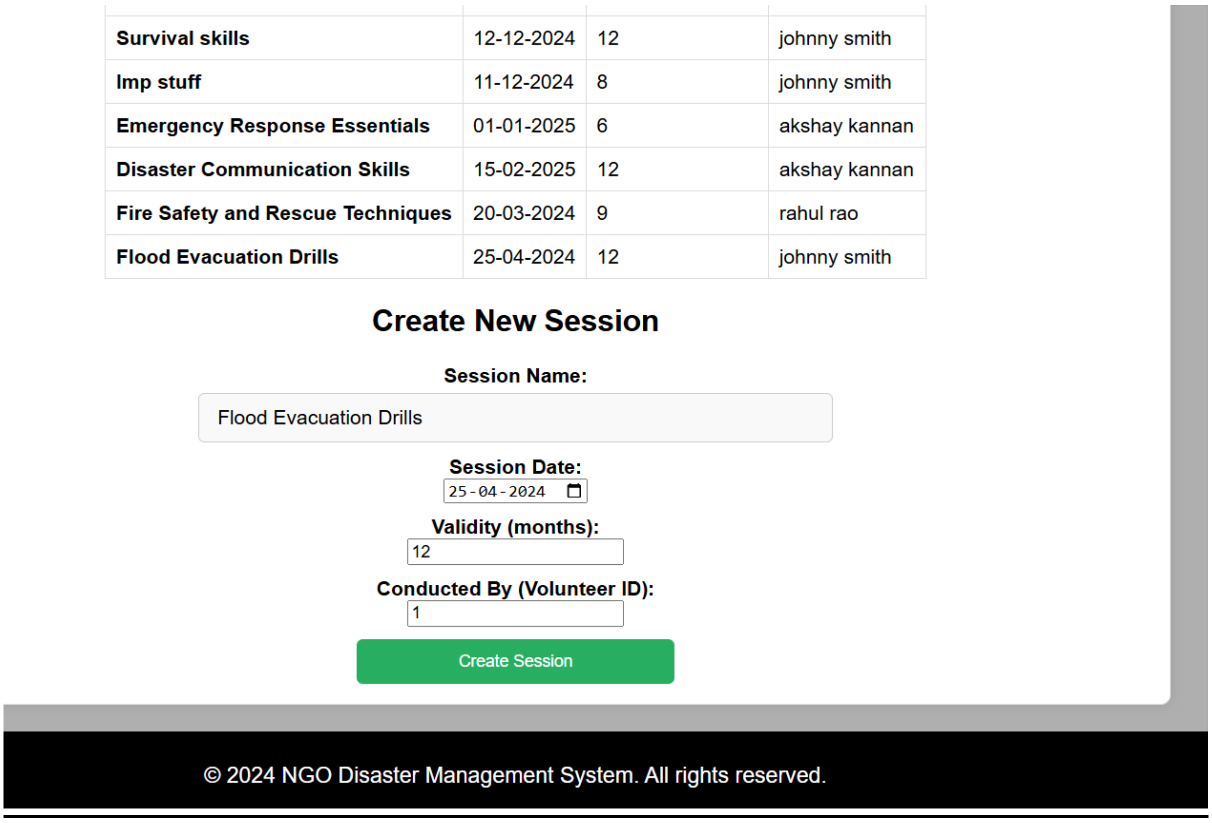


### Existing Disasters

Successfully applied to the disaster

| Name | Location | Type | Severity | Action |
|---|---|---|---|---|
| bigscarytsunami | Japan | Tsunami | critical | Apply to Volunteer |
| smn | Bangalore | tornado | low | Apply to Volunteer |
| Hurricane Aurora | Miami | Hurricane | critical | Apply to Volunteer |
| Forest Fire Inferno | California | Wildfire | high | Apply to Volunteer |
| Riverbank Deluge | Missouri | Flood | medium | Apply to Volunteer |
| Earthquake Rupture | Nepal | Earthquake | critical | Apply to Volunteer |
| Tornado Fury | Oklahoma | Tornado | high | Apply to Volunteer |
| Blizzard Whiteout | North Dakota | Blizzard | medium | Apply to Volunteer |

## Admins can create Training Sessions:



| | | | |
|---|---|---|---|
| Survival skills | 12-12-2024 | 12 | johnny smith |
| Imp stuff | 11-12-2024 | 8 | johnny smith |
| Emergency Response Essentials | 01-01-2025 | 6 | akshay kannan |
| Disaster Communication Skills | 15-02-2025 | 12 | akshay kannan |
| Fire Safety and Rescue Techniques | 20-03-2024 | 9 | rahul rao |
| Flood Evacuation Drills | 25-04-2024 | 12 | johnny smith |

### Create New Session

Session Name: Flood Evacuation Drills

Session Date: 25-04-2024

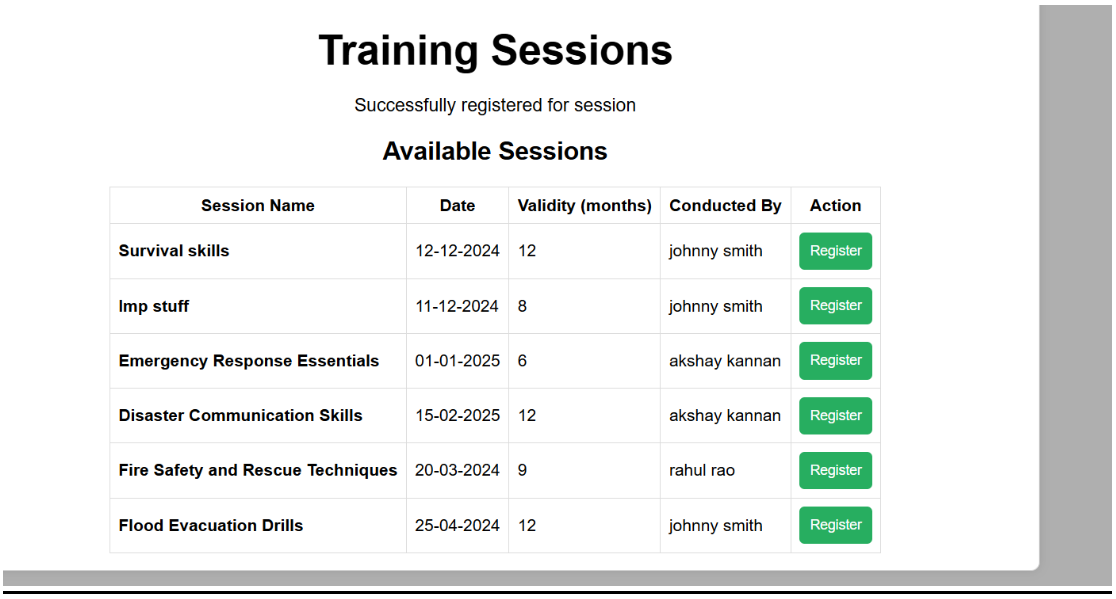
Validity (months): 12

Conducted By (Volunteer ID): 1

Create Session

© 2024 NGO Disaster Management System. All rights reserved.

- **<u>Volunteer registering for a session:</u>**



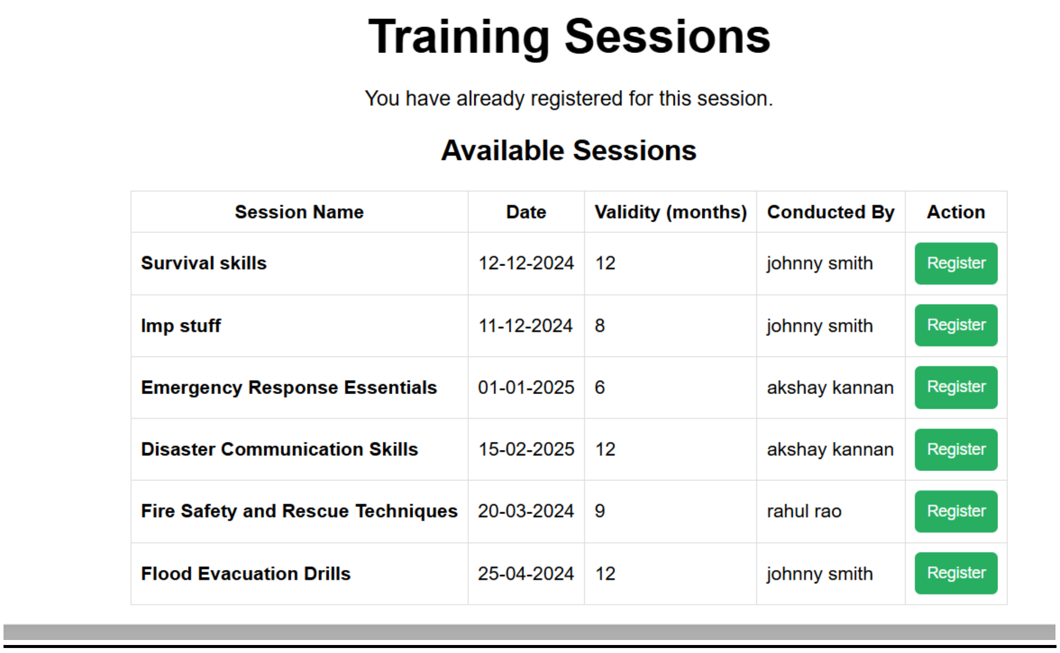
# Training Sessions

Successfully registered for session

## Available Sessions

| Session Name | Date | Validity (months) | Conducted By | Action |
|---|---|---|---|---|
| **Survival skills** | 12-12-2024 | 12 | johnny smith | Register |
| **Imp stuff** | 11-12-2024 | 8 | johnny smith | Register |
| **Emergency Response Essentials** | 01-01-2025 | 6 | akshay kannan | Register |
| **Disaster Communication Skills** | 15-02-2025 | 12 | akshay kannan | Register |
| **Fire Safety and Rescue Techniques** | 20-03-2024 | 9 | rahul rao | Register |
| **Flood Evacuation Drills** | 25-04-2024 | 12 | johnny smith | Register |

**<u>Volunteer registering for a session they already registered for:</u>**



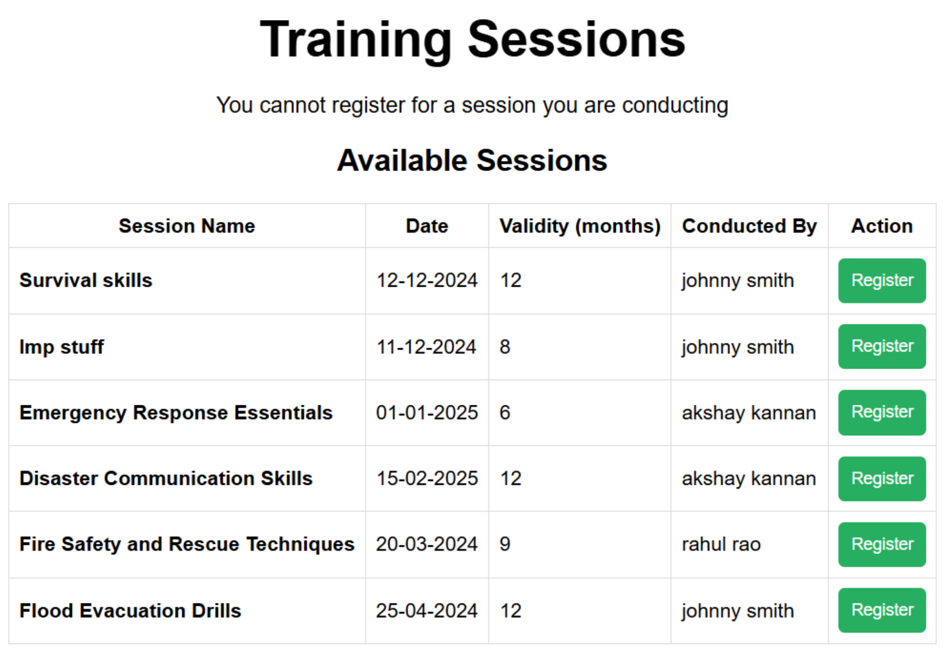
# Training Sessions

You have already registered for this session.

## Available Sessions

| Session Name | Date | Validity (months) | Conducted By | Action |
|---|---|---|---|---|
| **Survival skills** | 12-12-2024 | 12 | johnny smith | Register |
| **Imp stuff** | 11-12-2024 | 8 | johnny smith | Register |
| **Emergency Response Essentials** | 01-01-2025 | 6 | akshay kannan | Register |
| **Disaster Communication Skills** | 15-02-2025 | 12 | akshay kannan | Register |
| **Fire Safety and Rescue Techniques** | 20-03-2024 | 9 | rahul rao | Register |
| **Flood Evacuation Drills** | 25-04-2024 | 12 | johnny smith | Register |

**<u>Volunteer trying to register for a session they are hosting:</u>**



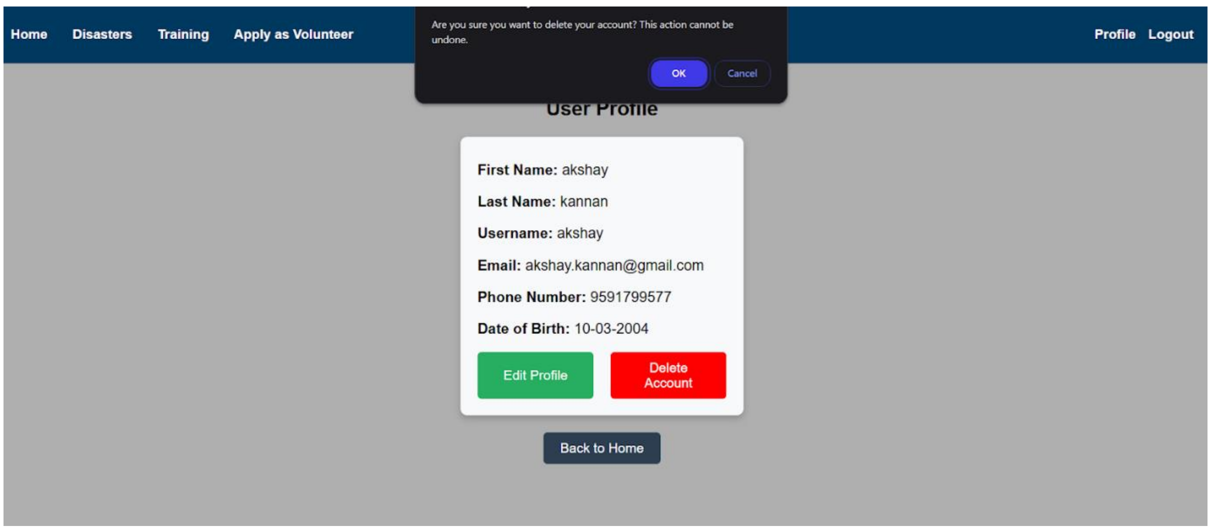
# Training Sessions

You cannot register for a session you are conducting

## Available Sessions

| Session Name | Date | Validity (months) | Conducted By | Action |
|---|---|---|---|---|
| **Survival skills** | 12-12-2024 | 12 | johnny smith | Register |
| **Imp stuff** | 11-12-2024 | 8 | johnny smith | Register |
| **Emergency Response Essentials** | 01-01-2025 | 6 | akshay kannan | Register |
| **Disaster Communication Skills** | 15-02-2025 | 12 | akshay kannan | Register |
| **Fire Safety and Rescue Techniques** | 20-03-2024 | 9 | rahul rao | Register |
| **Flood Evacuation Drills** | 25-04-2024 | 12 | johnny smith | Register |

## Deleting User account:



Are you sure you want to delete your account? This action cannot be undone.

User Profile

**First Name:** akshay

**Last Name:** kannan

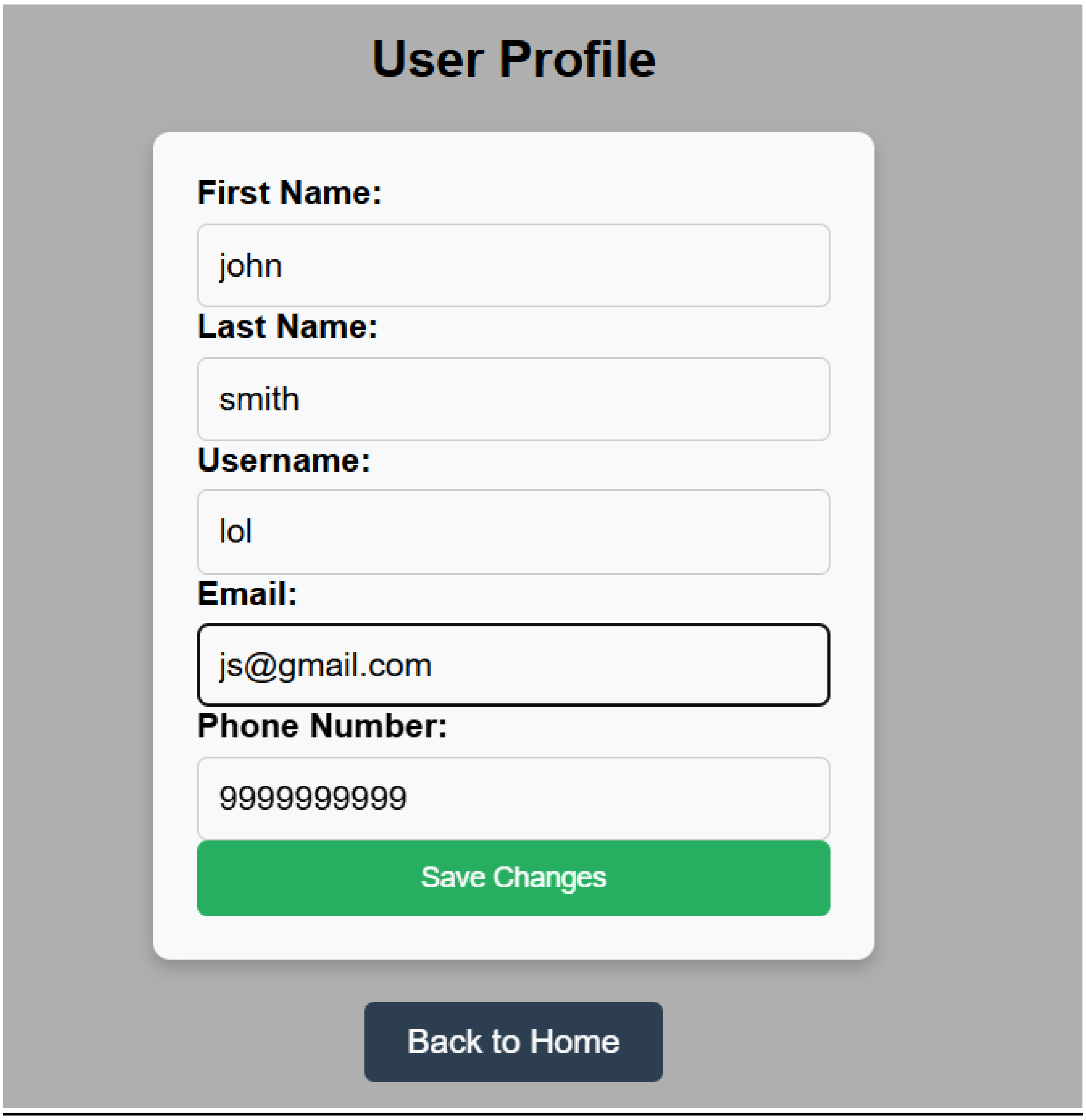
**Username:** akshay

**Email:** akshay.kannan@gmail.com

**Phone Number:** 9591799577

**Date of Birth:** 10-03-2004

## Editing User Profile:



User Profile

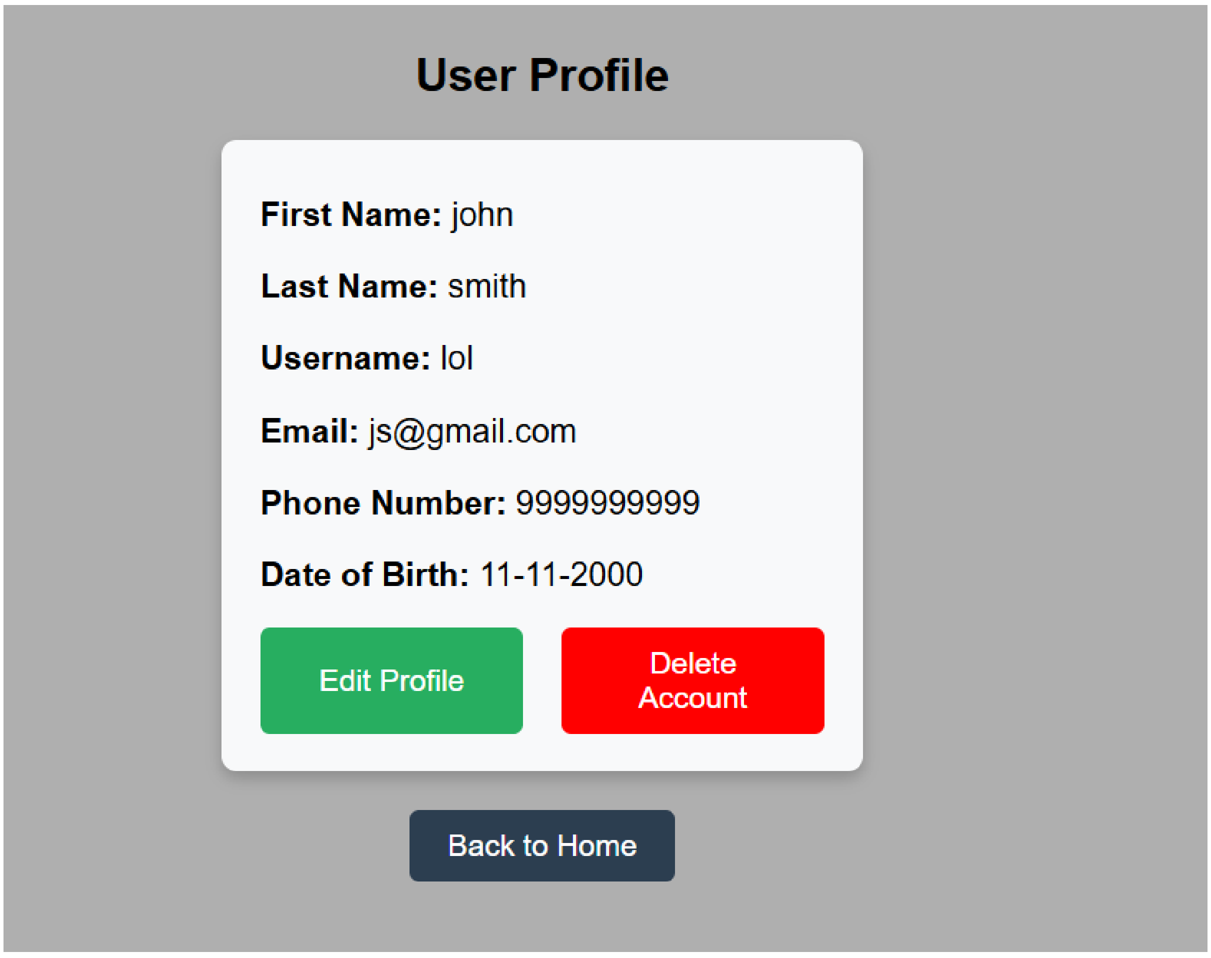
First Name: john

Last Name: smith

Username: lol

Email: js@gmail.com

Phone Number: 9999999999



User Profile

**First Name:** john

**Last Name:** smith

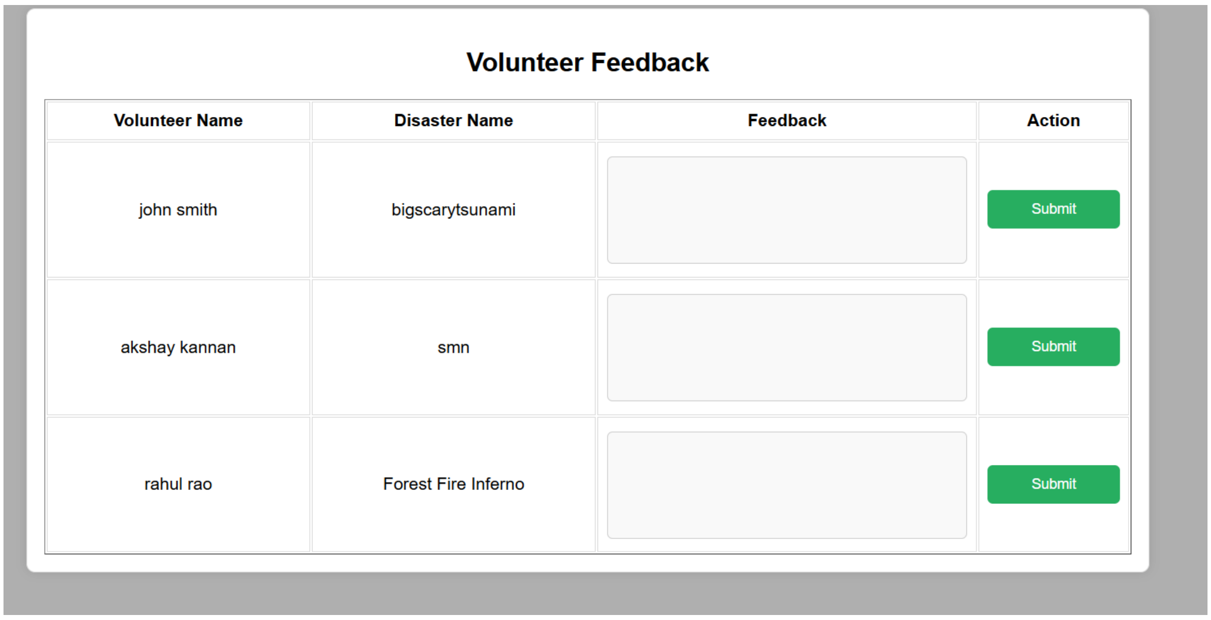
**Username:** lol

**Email:** js@gmail.com

**Phone Number:** 9999999999

**Date of Birth:** 11-11-2000

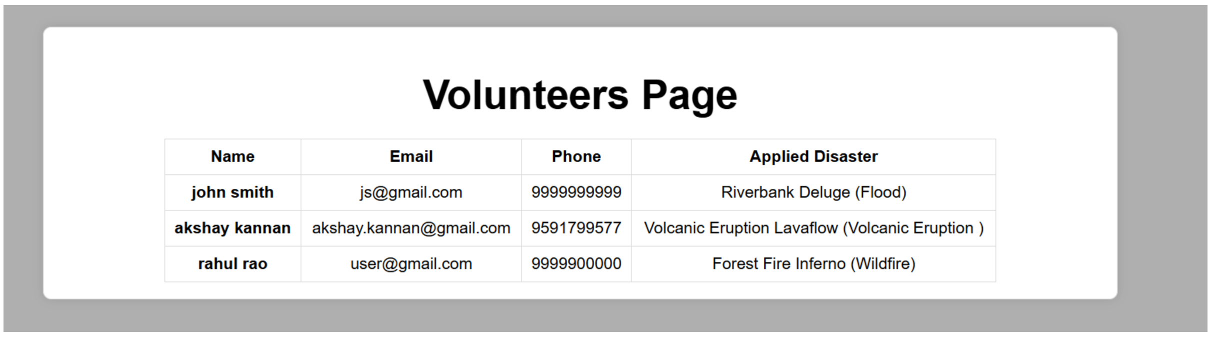
## Admin can give feedback on volunteers:



Volunteer Feedback

| Volunteer Name | Disaster Name | Feedback | Action |
|---|---|---|---|
| john smith | bigscarytsunami | | Submit |
| akshay kannan | smn | | Submit |
| rahul rao | Forest Fire Inferno | | Submit |

## Admin can see list of volunteers and disasters they applied for:



Volunteers Page

| Name | Email | Phone | Applied Disaster |
|---|---|---|---|
| **john smith** | js@gmail.com | 9999999999 | Riverbank Deluge (Flood) |
| **akshay kannan** | akshay.kannan@gmail.com | 9591799577 | Volcanic Eruption Lavaflow (Volcanic Eruption ) |
| **rahul rao** | user@gmail.com | 9999900000 | Forest Fire Inferno (Wildfire) |

# THANK YOU