Design DOCUMENTATION

Date: 19 December 2024

MONTCLAIR STATE UNIVERSITY

Environment Monitoring and Reporting System

Design Document

By:

TEAM UNITY CODERS

Course: CSIT 515 Software Engineering & Reliability – Fall 2024

UNDER THE GUIDANCE OF

Professor Dr. Hubert Johnson

BS, MS, EDM, EdD

Team Members:

UBBANI SAMSON RAJ

PASAM PRATHUSHA

RAHEEMA BEGUM SHAIK

KAITHAPALLI ABHISHEK

KAMALA SREE VALLI TATIPARTHI

Contents

1 Introduction

1.1 Project Overview

The Environmental Monitoring and Reporting System has been developed as a comprehensive tool to empower individuals, communities, and organizations in the fight against environmental crimes such as pollution, deforestation, and illegal fishing. The system enables users to report environmental issues in real time, providing authorities and relevant organizations with the critical information they need to act swiftly and effectively. It features a user-friendly interface, ensuring that anyone can easily navigate the platform, regardless of their technical skill level. The system also offers Spanish language support along with English, making it accessible to people from diverse backgrounds. Key features include real-time alerts and interactive maps that help authorities respond quickly to reported incidents. Additionally, the platform encourages users to share their experiences and success stories in protecting the environment, fostering a sense of community and collaboration among citizens, Non-Governmental Organizations, and environmental agencies. By utilizing artificial intelligence to visualize climate data, the system not only raises awareness about environmental issues but also supports efforts to develop effective solutions for protecting our planet.

1.2 System Overview

The Environmental Monitoring and Reporting System (EMRS) was developed to provide

a comprehensive platform for reporting environmental crimes. The platform enables citizens, environmental organizations, and government agencies to monitor environmental incidents such as illegal logging, pollution, and poaching, while fostering real-time reporting, data visualization, and community interaction.

The system's core functionality includes:

1.3 Core Functionalities and Objectives

- **User Reporting**: Allows users to report environmental crimes with location, photos, and descriptions.
- **Real-Time Alerts**: Alerts citizens and authorities about nearby environmental crimes.
- **Data Visualization**: Interactive dashboards to view climate trends and reported crimes.
- **Community Engagement**: Forums for discussions and collaboration.
- **Multilingual Support**: Accessibility in multiple languages for a global audience.

2 System Architecture

The Environmental Monitoring and Reporting System (EMRS) is designed to monitor and report environmental crimes while predicting potential future risks using AI-powered insights. It follows a layered architecture consisting of five main components: **Frontend**, **Backend**, **Database**, **AI Module**, and **Admin Panel**. These components interact to provide a seamless experience for users, including citizens, NGOs, and government agencies.

2.1 Detailed Overview:

1. **Frontend**:
    - The user interface (UI) for interacting with the system.
    - Provides tools for submitting reports, viewing alerts, accessing climate visualizations, and engaging in discussions.
    - It is implemented using modern web technologies (HTML, CSS, JavaScript, and Bootstrap).
2. **Backend**:
    - Acts as the brain of the system by processing user requests, managing logic, and interacting with the database and AI module.
    - Developed using PHP for business logic, authentication, and API management.
3. **Database**:
    - Stores all system data, including reports, alerts, user information, forum discussions, and predictions.
    - Managed using MySQL, ensuring secure and efficient data retrieval.
4. **AI Module**:
    - Processes climate-related data, predicts environmental risks (e.g., deforestation trends), and offers actionable insights.
    - Built with Python and Flask, it integrates AI libraries like NumPy and Pandas for analysis.
5. **Admin Panel**:
    - Provides administrators with tools for managing reports, monitoring forums, and generating insights.
    - Includes analytics dashboards for trend analysis.

2.2 Modules

2.2.1. Frontend Module

- **Explanation**: The frontend module is the user-facing layer of the system. It consists of:
    - Forms for submitting environmental crime reports.
    - A map displaying real-time alerts.
    - Interactive visualizations of climate data trends.
    - Community forums for discussing environmental issues.
- **Examples of Functionality**:
    - **Report Submission**: A citizen reports illegal deforestation with location and photos.
    - **Climate Visualization**: Users view charts displaying pollution levels in their region.

2.2.2. Backend Module

- **Explanation**: The backend processes requests from the frontend and coordinates between the database and AI module. It also manages authentication, authorization, and business logic.
- **Examples of Functionality**:
    - Handles user registration and login.
    - Validates and saves submitted reports.
    - Routes AI predictions to the frontend for visualization.

2.2.3. Database Module

- **Explanation**: This module stores and manages all system data, ensuring data security and consistency.
- **Key Data Stored**:
    - **Users**: Name, email, roles, and login credentials.
    - **Reports**: Details of environmental crimes, including location, severity, and photos.
    - **Predictions**: AI-generated insights and trend data.
    - **Forum Threads**: Community discussions.

2.2.4. AI Module

- **Explanation**: This module analyzes environmental data and provides predictions for future risks.
- **Features**:
    - Predicts trends in deforestation and pollution.
    - Uses Flask APIs to send predictions back to the backend.

5. Admin Panel Module

- **Explanation**: Enables administrators to monitor and manage the system effectively.
- **Features**:
    - Approve or reject user reports.
    - Moderate forum discussions.
    - View analytics dashboards showing trends and alerts.

2.3 Module Interactions

2.3.1 Detailed Flow:

1. **User Actions**:
    - A citizen submits a report via the frontend.
    - The frontend sends this data to the backend via an API request.
2. **Backend Processing**:
    - The backend validates the report and stores it in the database.
    - If the report requires AI analysis (e.g., predicting the impact of pollution), the backend sends data to the AI module.
3. **AI Analysis**:
    - The AI module processes the data, applies prediction models, and sends the results back to the backend.
4. **Data Visualization**:
    - The backend retrieves data from the database and combines it with AI insights to generate graphs and alerts for users.
5. **Admin Oversight**:
    - The admin logs in via the admin panel to moderate reports, manage forums, and view system analytics.

2.4 Diagrams

2.4.1. Use Case Diagram

Actors:

- **User**:
    - Submits environmental crime reports.
    - Views real-time alerts and climate visualizations.
    - Participates in forum discussions.
- **Admin**:
    - Monitors and approves reports.
    - Manages user interactions in forums.

- Accesses analytics for trend monitoring.

- **NGO/Authorities**:
  - Collaborates with users to respond to alerts.
  - Reviews system-generated predictions for action planning.

2.4.2. Class Diagram

Main Classes and Attributes:

1. **User**:
   - Attributes: User ID, name, email, role (user/admin), login credentials.
   - Methods: Login, logout, manage reports.
2. **Report**:
   - Attributes: Report ID, crime type, location, severity, description, timestamp.
   - Methods: Submit, retrieve, delete.
3. **Alert**:
   - Attributes: Alert ID, message, location, severity.
   - Methods: Generate, display, notify users.
4. **Prediction**:
   - Attributes: ID, data source, prediction result, timestamp.
   - Methods: Generate prediction, send result.

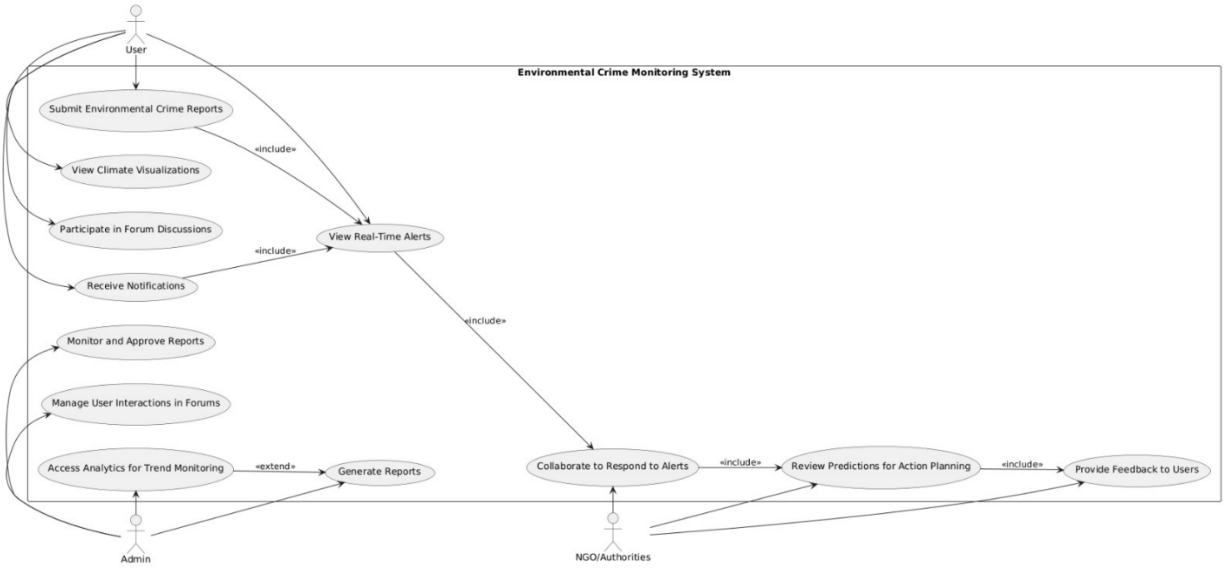


Fig -Use case diagram,

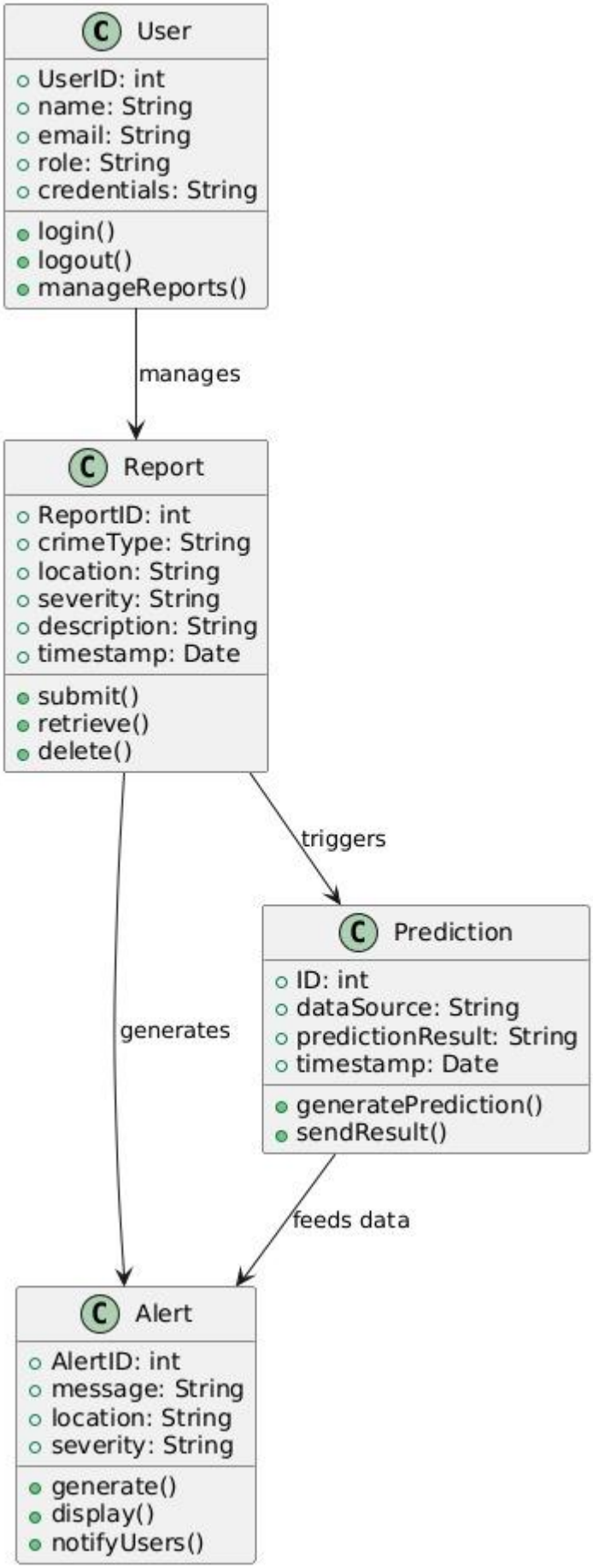

User
UserID: int
name: String
email: String
role: String
credentials: String
login()
logout()
manageReports()
manages
Report
ReportID: int
crimeType: String
location: String
severity: String
description: String
timestamp: Date
submit()
retrieve()
delete()
triggers
generates
Prediction
ID: int
dataSource: String
predictionResult: String
timestamp: Date
generatePrediction()
sendResult()
feeds data
Alert
AlertID: int
message: String
location: String
severity: String
generate()
display()
notifyUsers()

Fig- class Diagram

2.4.3. Sequence Diagram

Example Scenario: **Submitting a New Report**

1. **User** fills out a report form with details like location and description.
2. **Frontend** sends the report to the backend API.
3. **Backend** validates the data and stores it in the database.
4. **Backend** sends the report data to the AI module for prediction.
5. **AI Module** processes the data and returns the prediction.
6. **Backend** updates the report with AI insights and generates an alert.
7. **Frontend** notifies the user of the successful submission.

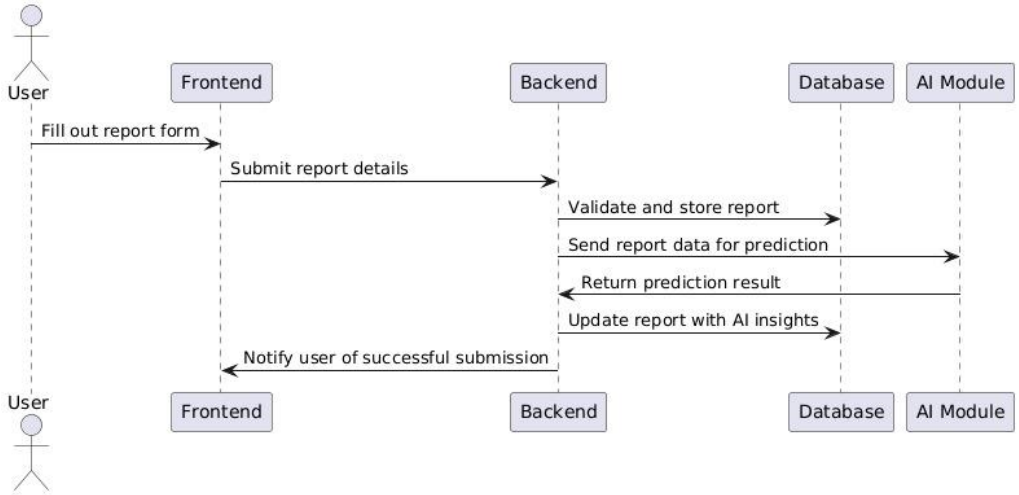


Fig- sequence diagram

3. Major Modules/Components

The Environmental Monitoring and Reporting System (EMRS) is a comprehensive platform consisting of multiple modules and components that work together to fulfill its functionalities. Below is an explanation of the major modules, their features, and how they contribute to the system.

3.1. User Interface (Frontend Module)

- Index page

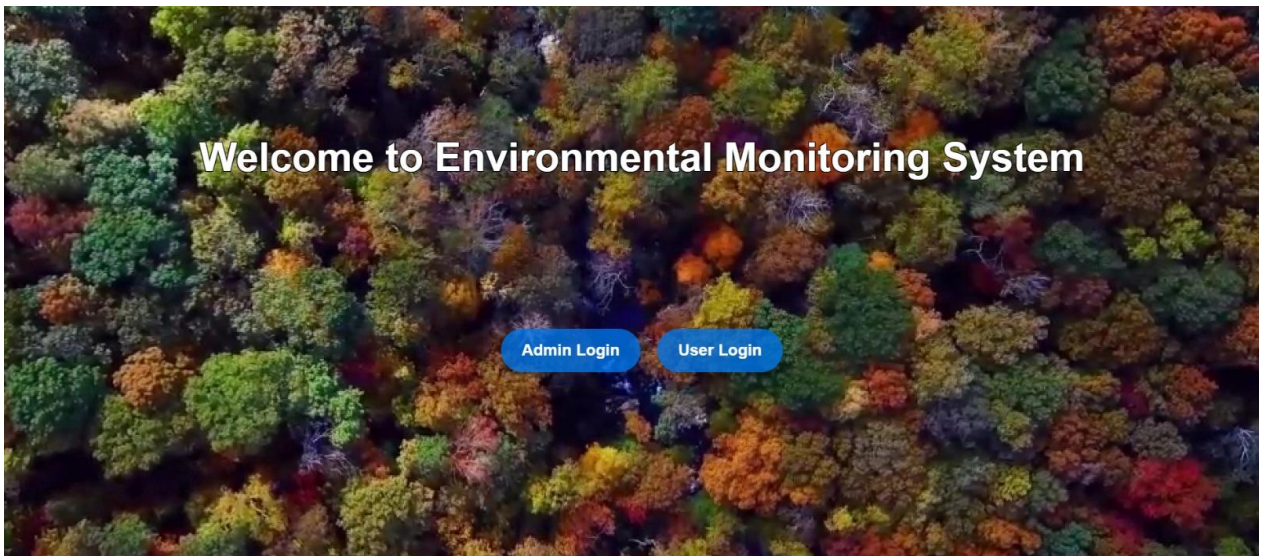


Fig index page

- Admin login page

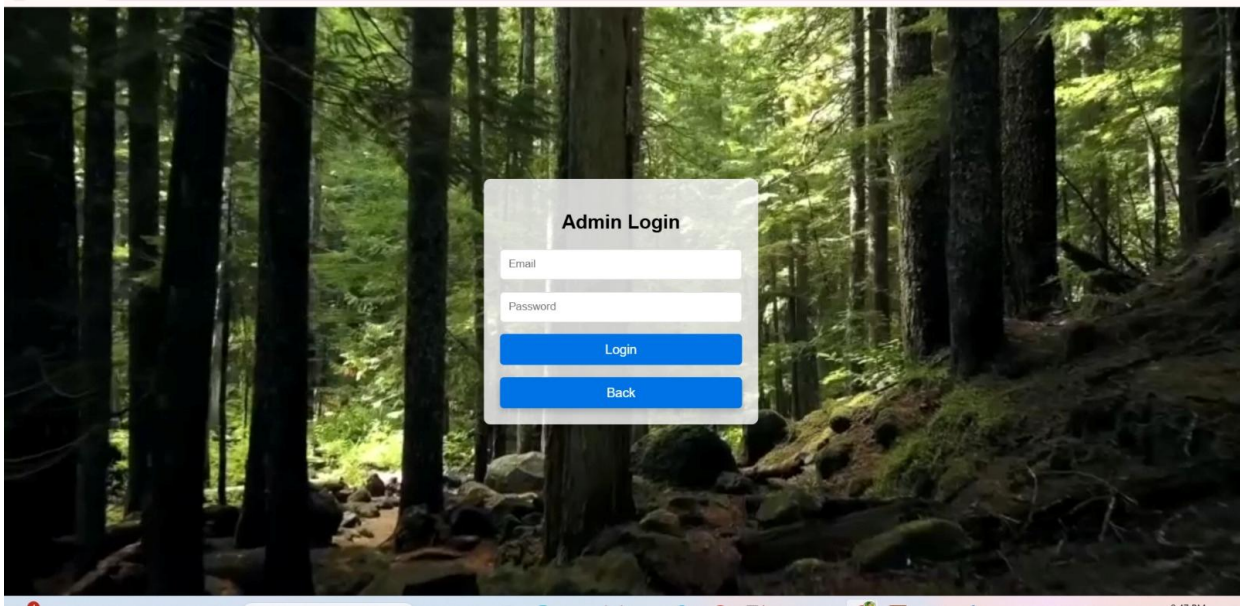


Fig - Admin login page

- User login page

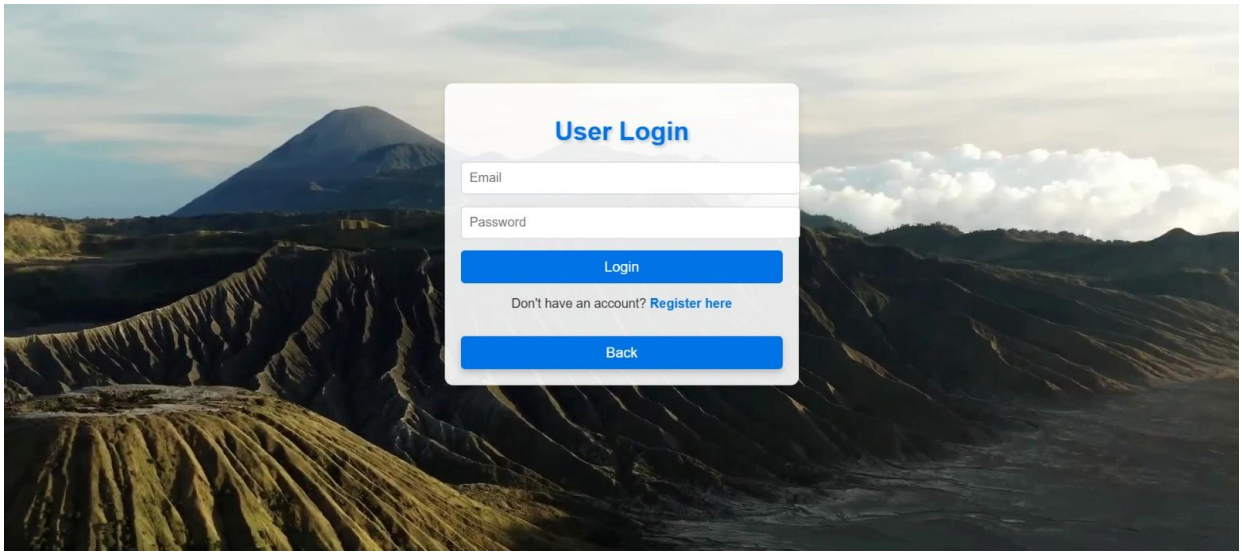


Fig user login page

- User Registration Page

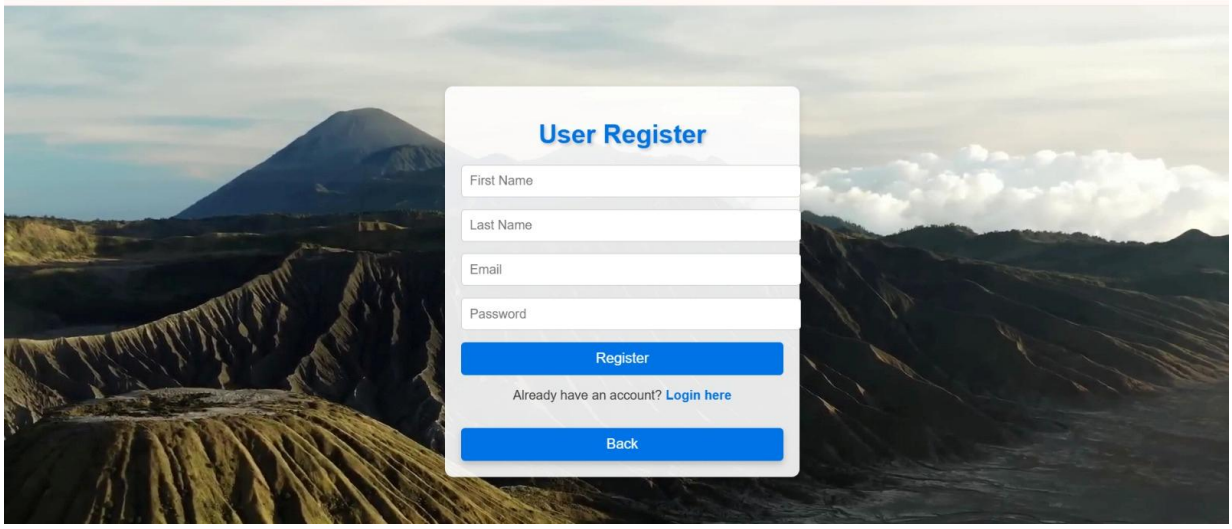


Fig user registration page

- Admin Dashboard Page

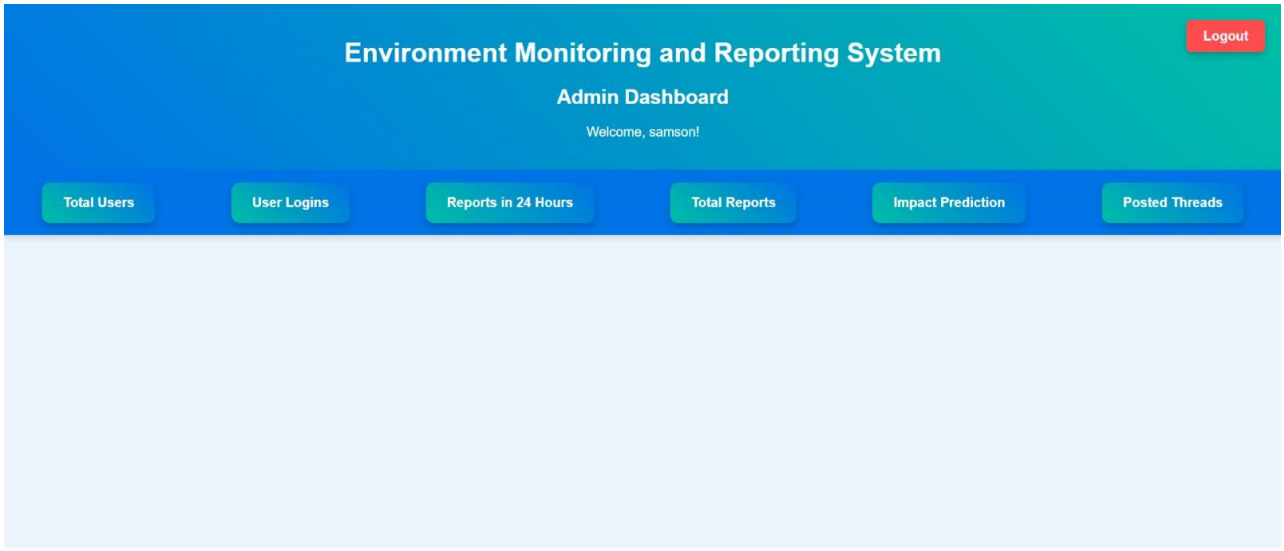


Fig admin dashboard

Description:

This module is the entry point for users, providing an interactive and intuitive interface to engage with the system.

Key Features:

- **User-Friendly Design**: Easy navigation through the system's functionalities using a sidebar menu.
- **Responsive Interface**: Compatible with various devices like laptops, tablets, and mobile phones.
- **Real-Time Interaction**: Users can view alerts, submit reports, and access climate visualizations instantly.

User Actions:

- Submit a report on environmental crimes.
- Access interactive maps with real-time environmental alerts.

- View AI-driven climate data visualizations.
- Participate in community discussions.

3.1.2.Report Management Module

Description:

Handles the creation, storage, and management of environmental reports submitted by users.

Key Features:

- **Report Submission**: Allows users to submit reports with details like crime type, severity, location, description, and optional images.
- **Data Validation**: Ensures reports contain accurate and complete data before storing them.
- **Storage**: Saves reports in the database with timestamps and associated user details.

Administrator Role:

- Review submitted reports.
- Approve or reject reports based on validity.
- Take actions on critical reports, such as escalating them to authorities.

3.1.3. Real-Time Alerts Module

Description:

Displays environmental alerts on an interactive map, providing users with real-time updates on reported crimes or incidents.

Key Features:

- **Interactive Map**: Shows the location of incidents with markers.
- **Info Windows**: Clicking on a marker reveals details about the crime or event.
- **Filters**: Users can filter alerts by type (pollution, deforestation, etc.) or severity.

Administrator Role:

- Ensure high-priority alerts are highlighted and pushed to relevant authorities.

3.1.4. Climate Data Visualization Module

Description:

Leverages AI-generated predictions to visualize climate trends and impacts.

Key Features:

- **Graphs and Charts**: Displays data trends (e.g., pollution levels, deforestation rates) in an easy-to-understand format.
- **Prediction Models**: Uses AI predictions to show the likely future impact of climate trends.

- **Customizable Views**: Users can focus on specific regions or datasets for detailed insights.

User Benefits:

- Understand environmental trends in their area.
- Use the insights to plan and advocate for actions against environmental issues.

3.1.5. Community Engagement Module

Description:

Provides a platform for users to discuss environmental issues, share knowledge, and collaborate on solutions.

Key Features:

- **Discussion Threads**: Users can create and participate in discussions about environmental topics.
- **Thread Moderation**: Administrators can remove inappropriate or irrelevant threads.
- **Community Collaboration**: Encourages knowledge sharing and joint actions.

Administrator Role:

- Moderate discussions to ensure a constructive and respectful environment.

3.1.6. AI Module

Description:

Analyzes environmental data to generate predictions, insights, and trends.

Key Features:

- **Trend Analysis**: Predicts the future impact of current environmental trends.
- **Data Integration**: Processes real-time reports and generates insights for users.
- **API Integration**: Communicates with the backend to provide AI-driven results.

Examples of AI Capabilities:

- Predicting deforestation risks based on submitted reports.
- Forecasting pollution growth in specific regions.

3.1.7. Database Module

Description:

Stores and organizes all system data, including user profiles, reports, AI predictions, and discussion threads.

Key Features:

- **Relational Database**: Uses MySQL to manage data efficiently.
- **Data Security**: Ensures user data and reports are protected.
- **Data Backup**: Maintains backups to prevent data loss.

Stored Data Examples:

- User Information: IDs, names, and roles.
- Reports: Details about submitted environmental crimes.
- Predictions: AI-generated trends and analyses.

3.1.8. Admin Panel

Description:

Allows administrators to oversee and manage all aspects of the system.

Key Features:

- **Report Moderation**: Approve or reject submitted reports.
- **Forum Moderation**: Monitor and manage user discussions.
- **Analytics Dashboard**: Access statistics about reports, alerts, and trends.

Administrator Actions:

- Generate and download reports for authorities.
- Monitor system performance and resolve issues.

3.2 Interaction Between Modules

3.2.1 How They Work Together:

1. **User Submission**: The user submits a report through the frontend. The backend processes it, stores it in the database, and sends it to the AI module for predictions.
2. **Alert Generation**: The backend retrieves AI insights and creates alerts, which are displayed on the frontend's real-time map.
3. **Climate Visualizations**: The AI module generates predictions based on database data. These visualizations are presented in the frontend for user insights.
4. **Community Discussions**: Users engage in forums, and the backend stores and displays the threads in real-time.
5. **Admin Management**: Administrators monitor all activities through the admin panel and take necessary actions.

4. Implementation Document

This section explains the workflows, libraries used, and implementation details of the system.

4.1 Website Workflow

Provide a detailed workflow of how the website functions:

1. **User Workflow:**
    - A user logs in or registers.
    - Submits environmental reports using the form.
    - Views real-time alerts and reports submitted by others.
    - Engages in discussions in the community engagement section.
2. **Admin Workflow:**
    - Logs in via the admin panel.
    - Reviews and verifies reports.
    - Manages users and generates system reports.

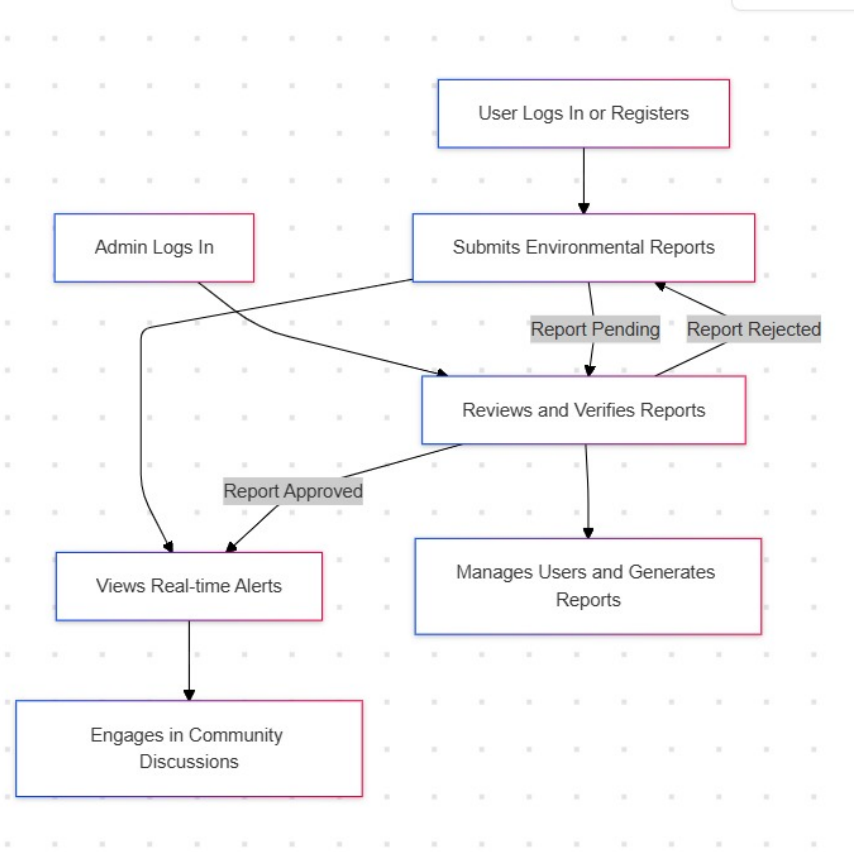


Fig work flow

4.2 Libraries and Technologies Used

1. **Frontend:**
    - HTML, CSS, JavaScript (for UI and dynamic interactions).
2. **Backend:**
    - PHP: Handles user authentication and database interactions.
    - Flask (Python): Manages AI predictions.
3. **Database:**
    - MySQL: Stores user data, reports, and predictions.

4. **Libraries:**
    - Flask: Used in the AI module.
    - Chart.js: For visualizing climate and report data.
    - Google Maps API: For rendering real-time report locations.
5. **Server:**
    - XAMPP: For hosting the PHP-based backend locally.

4.3 Implementation Code

Index.php

```
 <!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Welcome</title>
   <style>
      body {
         background-size: cover;
         font-family: 'Open Sans', sans-serif;
         color: #333;
         margin: 0;
         padding: 0;
      }
      .welcome-page {
         text-align: center;
         margin-top: 10%;
         color: white;
      }
      .welcome-page h1 {
         font-size: 3rem;
         margin-bottom: 30px;
         font-weight: bold;
         text-shadow: -1px -1px 0 #000, 1px -1px 0 #000, -1px 1px 0 #000, 1px 1px 0 #000;
      }
      .buttons {
```

```
            display: flex;
            justify-content: center;
            gap: 20px;
        }
        .btn {
            text-decoration: none;
            background-color: rgba(0, 115, 230, 0.8);
            color: white;
            padding: 15px 25px;
            border-radius: 30px;
            font-size: 1.2rem;
            font-weight: bold;
            box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.3);
        }
        .btn:hover {
            background-color: rgba(0, 115, 230, 1);
            transform: scale(1.1);
            box-shadow: 0px 6px 12px rgba(0, 0, 0, 0.5);
        }
        video {
            position: fixed;
            top: 0;
            left: 0;
            width: 100%;
            height: 100%;
            object-fit: cover;
            z-index: -1;
            animation: morphEffect 6s infinite alternate; /* Morphing animation */
        }
    </style>
</head>
<body>
<video autoplay muted loop>
        <source src="video/video4.mp4" type="video/mp4">
        Your browser does not support the video tag.
    </video>
    <div class="welcome-page">
```

```
        <h1>Welcome to Environmental Monitoring System</h1>  </br> </br> </br> </br> </br> </br> </br> </br>
        <div class="buttons">
        <a href="adminlogin.php" class="btn">Admin Login</a>
           <a href="login.php" class="btn">User Login</a>
        </div>
    </div>
</body>
</html>
```

Login .php

```
<?php
session_start();
require_once '../controllers/UserController.php';
$controller = new UserController();
$controller->loginUser();
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
    <style>
        body {
            background: url('assets/images/background.jpg') no-repeat center center fixed;
            background-size: cover;
            font-family: 'Open Sans', sans-serif;
            color: #333;
            margin: 0;
            padding: 0;
        }
        .login-form {
            max-width: 400px;
```

```
        margin: 100px auto;
        background: rgba(255, 255, 255, 0.9);
        padding: 20px;
        border-radius: 10px;
        box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.1);
        text-align: center;
    }
    .login-form h1 {
        margin-bottom: 20px;
        font-size: 2rem;
        color: #0073e6;
        text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.2);
    }
    .login-form input {
        width: 100%;
        padding: 10px;
        margin-bottom: 15px;
        border: 1px solid #ccc;
        border-radius: 5px;
        font-size: 1rem;
    }
    .login-form button {
        width: 100%;
        padding: 10px;
        background-color: #0073e6;
        color: white;
        border: none;
        border-radius: 5px;
        font-size: 1.1rem;
        cursor: pointer;
    }
    .login-form button:hover {
        background-color: #005bb5;
    }
    .login-form a {
        color: #0073e6;
        text-decoration: none;
```

```
            font-weight: bold;
        }
        .login-form a:hover {
            text-decoration: underline;
        }
        video {
            position: fixed;
            top: 0;
            left: 0;
            width: 100%;
            height: 100%;
            object-fit: cover;
            z-index: -1;
            animation: morphEffect 6s infinite alternate; /* Morphing animation */
        }
        .back-btn {
        margin-top: 15px;
        padding: 10px 20px;
        background-color: #ff4d4d;
        color: white;
        border: none;
        border-radius: 5px;
        font-size: 1.1rem;
        cursor: pointer;
        width: 100%;
        box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.2);
        transition: all 0.3s ease;
    }

    .back-btn:hover {
        background-color: #cc0000;
        transform: translateY(-3px);
        box-shadow: 0px 6px 12px rgba(0, 0, 0, 0.3);
    }
    </style>
</head>
<body>
```

```
<video autoplay muted loop>
      <source src="video/video1.mp4" type="video/mp4">
      Your browser does not support the video tag.
   </video>
   <div class="login-form">
      <h1>User Login</h1>
      <form method="POST" action="login.php">
         <input type="text" name="email" placeholder="Email" required>
         <input type="password" name="password" placeholder="Password" required>
         <button type="submit">Login</button>

      </form>
      <p>Don't have an account? <a href="register.php">Register here</a></p>

      <button onclick="window.location.href='index.php'" class="back-
btn">Back</button>

   </div>
</body>
</html>
```

Register.php

```
<?php
require_once '../controllers/UserController.php';

// Create UserController instance
$controller = new UserController();
$controller->registerUser();
?>

<!DOCTYPE html>
<html lang="en">
<head>
```

```
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>user Register</title>
<style>
  body {
    background: url('assets/images/background.jpg') no-repeat center center fixed;
    background-size: cover;
    font-family: 'Open Sans', sans-serif;
    color: #333;
    margin: 0;
    padding: 0;
  }
  .login-form {
    max-width: 400px;
    margin: 100px auto;
    background: rgba(255, 255, 255, 0.9);
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.1);
    text-align: center;
  }
  .login-form h1 {
    margin-bottom: 20px;
    font-size: 2rem;
    color: #0073e6;
    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.2);
  }
  .login-form input {
    width: 100%;
    padding: 10px;
    margin-bottom: 15px;
    border: 1px solid #ccc;
    border-radius: 5px;
    font-size: 1rem;
  }
  .login-form button {
    width: 100%;
```

```
    padding: 10px;
    background-color: #0073e6;
    color: white;
    border: none;
    border-radius: 5px;
    font-size: 1.1rem;
    cursor: pointer;
}
.login-form button:hover {
    background-color: #005bb5;
}
.login-form a {
    color: #0073e6;
    text-decoration: none;
    font-weight: bold;
}
.login-form a:hover {
    text-decoration: underline;
}
video {
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    object-fit: cover;
    z-index: -1;
    animation: morphEffect 6s infinite alternate; /* Morphing animation */
}
.back-btn {
margin-top: 15px;
padding: 10px 20px;
background-color: #ff4d4d;
color: white;
border: none;
border-radius: 5px;
font-size: 1.1rem;
```

```
        cursor: pointer;
        width: 100%;
        box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.2);
        transition: all 0.3s ease;
    }

    .back-btn:hover {
        background-color: #cc0000;
        transform: translateY(-3px);
        box-shadow: 0px 6px 12px rgba(0, 0, 0, 0.3);
    }
    </style>
</head>
<body>

<video autoplay muted loop>
        <source src="video/video1.mp4" type="video/mp4">
        Your browser does not support the video tag.
</video>
<div class="login-form">
    <h1>User Register</h1>
    <form method="POST" action="register.php">
        <input type="text" name="firstname" placeholder="First Name" required>
        <input type="text" name="lastname" placeholder="Last Name" required>
        <input type="email" name="email" placeholder="Email" required>
        <input type="password" name="password" placeholder="Password" required>
        <button type="submit">Register</button>
    </form>
    <p>Already have an account? <a href="login.php">Login here</a></p>

    <button onclick="window.location.href='index.php'" class="back-btn">Back</button>
</div>
</body>
</html>
```

Adminlogin.php

```
<?php
session_start();

// Database connection
$host = 'localhost';
$username = 'root'; // Replace with your DB username
$password = ''; // Replace with your DB password
$database = 'emrs_db';

$conn = new mysqli($host, $username, $password, $database);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Handle login
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $email = $_POST['email'];
    $password = $_POST['password'];

    // Validate admin credentials
    $query = "SELECT * FROM admins WHERE email = ? AND password = ?";
    $stmt = $conn->prepare($query);
    $stmt->bind_param("ss", $email, $password);
    $stmt->execute();
    $result = $stmt->get_result();

    if ($result->num_rows > 0) {
        // Assuming admin ID is retrieved after successful login
        $admin = $result->fetch_assoc();
        $admin_id = $admin['id']; // Ensure `id` exists in your `admins` table

        // Log the login event
```

```
        $log_query = "INSERT INTO user_logins (user_id, login_time) VALUES (?,
NOW())";
        $log_stmt = $conn->prepare($log_query);
        $log_stmt->bind_param("i", $admin_id);
        $log_stmt->execute();

        // Proceed with login
        $_SESSION['admin'] = $email;
        echo "<script>alert('Login successful!');
window.location.href='admindashboard.php';</script>";
    } else {
        echo "<script>alert('Invalid email or password!');</script>";
    }



}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Admin Login</title>
    <style>
        body {
            font-family: 'Arial', sans-serif;
            margin: 0;
            padding: 0;
            height: 100vh;
            overflow: hidden;
        }

        video {
            position: fixed;
            top: 0;
            left: 0;
```

```
    width: 100%;
    height: 100%;
    object-fit: cover;
    z-index: -1;
    animation: morphEffect 6s infinite alternate; /* Morphing animation */
}

.login-container {
    background: rgba(255, 255, 255, 0.8); /* Semi-transparent background */
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
    text-align: center;
    width: 300px;
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
}

.login-container h2 {
    margin-bottom: 20px;
    font-size: 1.5rem;
}

.login-container input[type="text"],
.login-container input[type="password"] {
    width: 100%;
    padding: 10px;
    margin-bottom: 15px;
    border: 1px solid #ddd;
    border-radius: 5px;
    box-sizing: border-box;
}

.login-container button {
    width: 100%;
```

```
    padding: 10px;
    background: #0073e6;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-size: 1rem;
}

.login-container button:hover {
    background: #005bb5;
}

.back-btn {
    margin-top: 15px;
    display: inline-block;
    text-decoration: none;
    background: #f4f4f4;
    color: #333;
    padding: 10px 15px;
    border: 1px solid #ddd;
    border-radius: 5px;
    font-size: 0.9rem;
    cursor: pointer;
    transition: background-color 0.3s ease;
}

.back-btn:hover {
    background-color: #ddd;
}
.back-btn {
margin-top: 15px;
padding: 10px 20px;
background-color: #ff4d4d;
color: white;
border: none;
border-radius: 5px;
```

```
        font-size: 1.1rem;
        cursor: pointer;
        width: 100%;
        box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.2);
        transition: all 0.3s ease;
    }

    .back-btn:hover {
        background-color: #cc0000;
        transform: translateY(-3px);
        box-shadow: 0px 6px 12px rgba(0, 0, 0, 0.3);
    }

        @keyframes morphEffect {
            0% {
                filter: brightness(100%) saturate(100%) blur(0px);
            }
            25% {
                filter: brightness(120%) saturate(120%) blur(2px);
            }
            50% {
                filter: brightness(90%) saturate(80%) blur(4px);
            }
            75% {
                filter: brightness(110%) saturate(110%) blur(2px);
            }
            100% {
                filter: brightness(100%) saturate(100%) blur(0px);
            }
        }
    </style>
</head>
<body>
    <video autoplay muted loop>
        <source src="video/video3.mp4" type="video/mp4">
        Your browser does not support the video tag.
    </video>
```

```
    <div class="login-container">
        <h2>Admin Login</h2>
        <form action="adminlogin.php" method="POST">
            <input type="text" name="email" placeholder="Email" required>
            <input type="password" name="password" placeholder="Password" required>
            <button type="submit">Login</button>
        </form>
        <button onclick="window.location.href='index.php'" class="back-
btn">Back</button>

    </div>
</body>
</html>
```

Admindashboard.php

```
<?php
session_start();

// Check if this is an AJAX request to fetch threads
if (isset($_GET['action']) && $_GET['action'] === 'fetch_threads') {
    header("Content-Type: application/json");

    $threads_query = "
        SELECT t.id, u.name AS user_name, t.title, t.content, t.created_at
        FROM threads t
        JOIN users u ON t.user_id = u.id
        ORDER BY t.created_at DESC";
    try {
        $stmt = $conn->prepare($threads_query);
        $stmt->execute();
        $threads = $stmt->fetchAll(PDO::FETCH_ASSOC);
        echo json_encode($threads);
    } catch (PDOException $e) {
        echo json_encode(["error" => "Database query failed: " . $e->getMessage()]);
    }
```

```
    exit();
}

// Check if the admin is logged in
if (!isset($_SESSION['admin'])) {
    echo "<script>alert('Please login first!');
window.location.href='adminlogin.php';</script>";
    exit();
}

// Database connection
$host = 'localhost';
$username = 'root'; // Replace with your DB username
$password = ''; // Replace with your DB password
$database = 'emrs_db';

$conn = new mysqli($host, $username, $password, $database);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Handle logout
if (isset($_GET['logout'])) {
    session_destroy();
    echo "<script>alert('You have been logged out!');
window.location.href='adminlogin.php';</script>";
    exit();
}

// Fetch admin details
$admin_email = $_SESSION['admin'];
$query = "SELECT firstname FROM admins WHERE email = ?";
$stmt = $conn->prepare($query);
$stmt->bind_param("s", $admin_email);
$stmt->execute();
```

```
$result = $stmt->get_result();
$admin = $result->fetch_assoc();
$admin_firstname = $admin['firstname'];

// Fetch statistics
$total_users_query = "SELECT COUNT(*) AS total_users FROM users";
$total_users_result = $conn->query($total_users_query)->fetch_assoc()['total_users'];

$total_reports_query = "SELECT COUNT(*) AS total_reports FROM reports";
$total_reports_result = $conn->query($total_reports_query)->fetch_assoc()['total_reports'];

$reports_24hr_query = "SELECT COUNT(*) AS reports_24hr FROM reports WHERE
created_at >= NOW() - INTERVAL 1 DAY";
$reports_24hr_result = $conn->query($reports_24hr_query)-
>fetch_assoc()['reports_24hr'];

$user_logins_query = "SELECT COUNT(*) AS user_logins FROM user_logins"; //
Replace with your login tracking table
$user_logins_result = $conn->query($user_logins_query)->fetch_assoc()['user_logins'];

// Fetch reports for table
$reports_query = "SELECT * FROM reports ORDER BY created_at DESC";
$reports_result = $conn->query($reports_query);

// Handle report deletion
if (isset($_POST['delete_report'])) {
    $report_id = $_POST['report_id'];
    $delete_query = "DELETE FROM reports WHERE id = ?";
    $delete_stmt = $conn->prepare($delete_query);
    $delete_stmt->bind_param("i", $report_id);
    if ($delete_stmt->execute()) {
        echo "<script>alert('Report deleted successfully!');
window.location.href='admindashboard.php';</script>";
    } else {
        echo "<script>alert('Failed to delete the report.');</script>";
    }
}
```

```
// Handle thread update
if (isset($_POST['edit_thread'])) {
    $thread_id = $_POST['thread_id'];
    $new_title = $_POST['new_title'];
    $new_content = $_POST['new_content'];

    $update_query = "UPDATE threads SET title = ?, content = ? WHERE id = ?";
    $update_stmt = $conn->prepare($update_query);
    $update_stmt->bind_param("ssi", $new_title, $new_content, $thread_id);
    if ($update_stmt->execute()) {
        echo "<script>alert('Thread updated successfully!');
window.location.href='admindashboard.php';</script>";
    } else {
        echo "<script>alert('Failed to update thread.');</script>";
    }
}

// Handle thread deletion
if (isset($_POST['delete_thread'])) {
    $thread_id = $_POST['thread_id'];
    $delete_query = "DELETE FROM threads WHERE id = ?";
    $delete_stmt = $conn->prepare($delete_query);
    $delete_stmt->bind_param("i", $thread_id);
    if ($delete_stmt->execute()) {
        echo "<script>alert('Thread deleted successfully!');
window.location.href='admindashboard.php';</script>";
    } else {
        echo "<script>alert('Failed to delete thread.');</script>";
    }
}

?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<title>Admin Dashboard</title>
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
<style>
  body {
    font-family: 'Open Sans', sans-serif;
    margin: 0;
    background: #eaf4f9;
  }
  .header {
    background: linear-gradient(45deg, #0073e6, #00bfa6);
    color: white;
    padding: 20px;
    text-align: center;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
  }
  .nav-buttons {
    display: flex;
    justify-content: space-around;
    background: #0073e6;
    padding: 15px 0;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
  }
  .nav-buttons button {
    background: linear-gradient(45deg, #00bfa6, #0073e6);
    border: none;
    padding: 15px 25px;
    cursor: pointer;
    border-radius: 10px;
    font-size: 1rem;
    color: white;
    font-weight: bold;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
    transition: all 0.3s ease;
  }
  .nav-buttons button:hover {
    transform: translateY(-5px);
    background: linear-gradient(45deg, #0073e6, #00bfa6);
```

```
        box-shadow: 0 6px 12px rgba(0, 0, 0, 0.3);
    }
    .content {
        padding: 30px;
    }
    .content-section {
        display: none;
        background: white;
        border-radius: 10px;
        padding: 20px;
        box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    }
    .content-section.active {
        display: block;
    }
    table {
        width: 100%;
        border-collapse: collapse;
        margin-top: 20px;
    }
    table, th, td {
        border: 1px solid #ddd;
    }
    th, td {
        padding: 15px;
        text-align: left;
    }
    th {
        background: #0073e6;
        color: white;
    }
    .action-btn {
        background: #00bfa6;
        color: white;
        border: none;
        padding: 10px 15px;
        cursor: pointer;
```

```
        border-radius: 5px;
        transition: all 0.3s ease;
    }
    .action-btn:hover {
        background: #0073e6;
    }
    canvas {
        margin: 30px auto;
    }
    .logout-btn {
    position: absolute;
    top: 20px;
    right: 20px;
    background-color: #ff4d4d;
    color: white;
    border: none;
    padding: 10px 20px;
    cursor: pointer;
    border-radius: 5px;
    font-size: 1rem;
    font-weight: bold;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
    transition: all 0.3s ease;
}

.logout-btn:hover {
    background-color: #cc0000;
    transform: translateY(-3px);
    box-shadow: 0 6px 12px rgba(0, 0, 0, 0.3);
}

.action-btn {
background: #00bfa6;
color: white;
border: none;
padding: 5px 10px;
cursor: pointer;
```

```
    border-radius: 5px;
    margin: 5px;
    transition: all 0.3s ease;
}

.action-btn:hover {
    background: #0073e6;
}

    </style>
</head>
<body>
    <div class="header">

    <h1>     Environment Monitoring and Reporting System</h1>
        <h2>Admin Dashboard</h2>
        <p>Welcome, <?php echo htmlspecialchars($admin_firstname); ?>!</p>
        <button onclick="window.location.href='admindashboard.php?logout=true'"
class="logout-btn">Logout</button>



    </div>

    <div class="nav-buttons">
        <button onclick="showSection('users')">Total Users</button>
        <button onclick="showSection('logins')">User Logins</button>
        <button onclick="showSection('recent-reports')">Reports in 24 Hours</button>
        <button onclick="showSection('all-reports')">Total Reports</button>
        <button onclick="showSection('ai-predictions')">Impact Prediction</button>
        <button onclick="showSection('user-threads')">Posted Threads</button>

    </div>

    <div class="content">
        <div id="users" class="content-section">
            <h2>Total Registered Users</h2>
            <p><?php echo $total_users_result; ?> users are registered in the system.</p>
```

```
    </div>

    <div id="logins" class="content-section">
        <h2>Total User Logins</h2>
        <p><?php echo $user_logins_result; ?> logins have been recorded.</p>
    </div>

    <div id="recent-reports" class="content-section">
        <h2>Reports Submitted in the Last 24 Hours</h2>
        <p><?php echo $reports_24hr_result; ?> reports were submitted in the last 24
hours.</p>
    </div>

    <div id="all-reports" class="content-section">
        <h2>Total Reports in the Database</h2>
        <p><?php echo $total_reports_result; ?> reports are currently in the database.</p>
        <h3>All Reports</h3>
        <table>
            <tr>
                <th>ID</th>
                <th>Crime Type</th>
                <th>Severity</th>
                <th>Location</th>
                <th>Created At</th>
                <th>Actions</th>
            </tr>
            <?php while ($report = $reports_result->fetch_assoc()): ?>
                <tr>
                    <td><?php echo $report['id']; ?></td>
                    <td><?php echo htmlspecialchars($report['crime_type']); ?></td>
                    <td><?php echo htmlspecialchars($report['severity']); ?></td>
                    <td><?php echo htmlspecialchars($report['location']); ?></td>
                    <td><?php echo htmlspecialchars($report['created_at']); ?></td>
                    <td>
                        <form method="POST" style="display:inline;">
                            <input type="hidden" name="report_id" value="<?php echo
$report['id']; ?>">
```

```
                        <button type="submit" name="delete_report" class="action-
btn">Delete</button>
                    </form>
                </td>
            </tr>
        <?php endwhile; ?>
    </table>
</div>

<div id="ai-predictions" class="content-section">
    <h2>Climate Impact Predictions</h2>
    <canvas id="climateChart" width="400" height="200"></canvas>
    <h3>Predicted Impact Data</h3>
    <table>
        <thead>
            <tr>
                <th>S.No</th>
                <th>Location</th>
                <th>Impact (sq.m)</th>
                <th>Predicted Impact</th>
            </tr>
        </thead>
        <tbody id="prediction-table-body">
            <!-- Data will be dynamically populated -->
        </tbody>
    </table>
</div>

<div id="user-threads" class="content-section">
  <h2>User Posted Threads</h2>
  <table>
    <thead>
      <tr>
        <th>Thread ID</th>
        <th>User</th>
        <th>Title</th>
        <th>Content</th>
```

```
            <th>Posted At</th>
            <th>Actions</th>
        </tr>
    </thead>
    <tbody id="threads-table-body">
        <!-- Data will be dynamically inserted here -->
    </tbody>
  </table>
</div>

<script>
  document.addEventListener("DOMContentLoaded", function () {
   const threadsTableBody = document.getElementById("threads-table-body");

   // Fetch thread data dynamically
   fetch("admindashboard.php?action=fetch_threads") // Same file with action parameter
      .then(response => {
        if (!response.ok) {
          throw new Error("Failed to fetch thread data");
        }
        return response.json();
      })
      .then(data => {
        // Clear the table body
        threadsTableBody.innerHTML = "";

        // Populate the table with thread data
        if (data.length > 0) {
          data.forEach(thread => {
            const row = document.createElement("tr");
            row.innerHTML = `
              <td>${thread.id}</td>
              <td>${thread.user_name}</td>
              <td>${thread.title}</td>
              <td>${thread.content}</td>
              <td>${thread.created_at}</td>
              <td>
```

```
                    <form method="POST" style="display:inline;">
                        <input type="hidden" name="thread_id" value="${thread.id}">
                        <input type="text" name="new_title" placeholder="Edit Title"
required>
                        <input type="text" name="new_content" placeholder="Edit Content"
required>
                        <button type="submit" name="edit_thread" class="action-
btn">Edit</button>
                        <button type="submit" name="delete_thread" class="action-
btn">Delete</button>
                    </form>
                </td>
            `;
            threadsTableBody.appendChild(row);
        });
    } else {
        threadsTableBody.innerHTML = `<tr><td colspan="6">No threads posted
yet.</td></tr>`;
    }
})
.catch(error => {
    console.error("Error fetching thread data:", error);
    threadsTableBody.innerHTML = `<tr><td colspan="6">Error fetching thread
data.</td></tr>`;
});
});

</script>


</div>

<script>
    document.addEventListener("DOMContentLoaded", function () {
const tableBody = document.getElementById("prediction-table-body");

// Fetch predictions from AI service
fetch("http://127.0.0.1:8000/predict_from_db")
```

```
        .then((response) => {
            if (!response.ok) throw new Error("Failed to fetch predictions");
            return response.json();
        })
        .then((data) => {
            tableBody.innerHTML = "";
            data.forEach((item, index) => {
                const row = document.createElement("tr");
                row.innerHTML = `
                    <td>${index + 1}</td>
                    <td>${item.report.address || "Unknown"}</td> <!-- Use 'address' instead of
'location' -->
                    <td>${item.report.impact || 0} sq.m</td>
                    <td>${item.prediction}</td>
                `;
                tableBody.appendChild(row);
            });
            renderChart(data);
        })
        .catch((error) => console.error("Error fetching predictions:", error));
});

        function renderChart(data) {
            const ctx = document.getElementById('climateChart').getContext('2d');

            const labels = data.map(item => item.report.location || "Unknown");
            const values = data.map(item => item.report.impact || 0);

            new Chart(ctx, {
                type: 'bar',
                data: {
                    labels: labels,
                    datasets: [{
                        label: "Impact Predictions",
                        data: values,
                        backgroundColor: 'rgba(0, 123, 255, 0.7)',
                        borderColor: 'rgba(0, 123, 255, 1)',
```

```
                    borderWidth: 1
                }]
            },
            options: {
                responsive: true,
                scales: {
                    y: {
                        beginAtZero: true
                    }
                }
            }
        });
    }

    function showSection(sectionId) {
  const sections = document.querySelectorAll('.content-section');
  sections.forEach(section => {
    section.classList.remove('active');
  });
  const activeSection = document.getElementById(sectionId);
  if (activeSection) {
    activeSection.classList.add('active');
  }
}


  </script>
</body>
</html>
```

Dashboard.php

```
<?php
session_start();
```

```
require_once '../config/Database.php';

if (!isset($_SESSION['user_id'])) {
    header("Location: login.php");
    exit();
}

// Database connection
$database = new Database();
$conn = $database->connect();

// Fetch report counts
$total_reports = 0;
$reports_last_24h = 0;

try {
    // Total reports
    $query_total = "SELECT COUNT(*) AS total FROM reports";
    $stmt_total = $conn->prepare($query_total);
    $stmt_total->execute();
    $result_total = $stmt_total->fetch(PDO::FETCH_ASSOC);
    $total_reports = $result_total['total'];

    // Reports in the last 24 hours
    $query_24h = "SELECT COUNT(*) AS last_24h FROM reports WHERE created_at >=
NOW() - INTERVAL 1 DAY";
    $stmt_24h = $conn->prepare($query_24h);
    $stmt_24h->execute();
    $result_24h = $stmt_24h->fetch(PDO::FETCH_ASSOC);
    $reports_last_24h = $result_24h['last_24h'];
} catch (PDOException $e) {
    $message = "<div class='message error'>Error fetching counts: " . $e->getMessage() .
"</div>";
}

//Code to Fetch Reports for Map
// Code to Fetch Threads
```

```
// Fetch reports as JSON for Real-Time Alerts Map
if (isset($_GET['action']) && $_GET['action'] === 'fetch_reports') {
    header('Content-Type: application/json');
    try {
        $query = "SELECT crime_type, description, location, lat, lng FROM reports";
        $stmt = $conn->prepare($query);
        $stmt->execute();
        $reports = $stmt->fetchAll(PDO::FETCH_ASSOC);
        echo json_encode($reports);
    } catch (PDOException $e) {
        echo json_encode(["error" => $e->getMessage()]);
    }
    exit(); // Stop further processing
}

// This fetches threads from the database and returns them as JSON:

if (isset($_GET['action']) && $_GET['action'] === 'fetch_threads') {
    header('Content-Type: application/json');
    try {
        $query = "SELECT * FROM community_threads ORDER BY created_at DESC";
        $stmt = $conn->prepare($query);
        $stmt->execute();
        $threads = $stmt->fetchAll(PDO::FETCH_ASSOC);
        echo json_encode($threads);
    } catch (PDOException $e) {
        echo json_encode(["error" => $e->getMessage()]);
    }
    exit();
}
// This saves a new thread submitted by the user:

if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['action']) &&
$_POST['action'] === 'create_thread') {
    $title = $_POST['title'];
    $content = $_POST['content'];
```

```
    try {
        $query = "INSERT INTO community_threads (title, content) VALUES (:title,
:content)";
        $stmt = $conn->prepare($query);
        $stmt->bindParam(':title', $title);
        $stmt->bindParam(':content', $content);
        if ($stmt->execute()) {
            echo json_encode(["success" => "Thread posted successfully!"]);
        } else {
            echo json_encode(["error" => "Failed to post thread."]);
        }
    } catch (PDOException $e) {
        echo json_encode(["error" => $e->getMessage()]);
    }
    exit();
}



// Initialize message variable
$message = "";

$reports = [];
try {
    $query = "SELECT crime_type, severity, impact, location, description, image_path,
created_at
              FROM reports ORDER BY created_at DESC";
    $stmt = $conn->prepare($query);
    $stmt->execute();
    $reports = $stmt->fetchAll(PDO::FETCH_ASSOC);
} catch (PDOException $e) {
    $message = "<div class='message error'>Error fetching reports: " . $e->getMessage() .
"</div>";
}

// Handle form submission
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
```

```
    $user_id = $_SESSION['user_id']; // Get logged-in user ID
    $crime_type = $_POST['crime_type'];
    $severity = $_POST['severity'];
    $impact = $_POST['impact'];
    $location = $_POST['location'];
    $lat = $_POST['lat']; // Latitude from the hidden input
    $lng = $_POST['lng']; // Longitude from the hidden input
    $description = $_POST['description'];

    // Handle image upload
    $image_path = null;
    if (isset($_FILES['image']) && $_FILES['image']['error'] === UPLOAD_ERR_OK) {
        $image_name = time() . '_' . basename($_FILES['image']['name']);
        $target_path = "../uploads/" . $image_name;

        if (!is_dir("../uploads")) {
            mkdir("../uploads", 0755, true);
        }

        if (move_uploaded_file($_FILES['image']['tmp_name'], $target_path)) {
            $image_path = "uploads/" . $image_name;
        }
    }

    // Insert report into the database
    try {
        $query = "INSERT INTO reports (user_id, crime_type, severity, impact, location, lat,
lng, description, image_path)
                VALUES (:user_id, :crime_type, :severity, :impact, :location, :lat, :lng,
:description, :image_path)";
        $stmt = $conn->prepare($query);

        $stmt->bindParam(':user_id', $user_id);
        $stmt->bindParam(':crime_type', $crime_type);
        $stmt->bindParam(':severity', $severity);
        $stmt->bindParam(':impact', $impact);
        $stmt->bindParam(':location', $location);
```

```
        $stmt->bindParam(':lat', $lat);
        $stmt->bindParam(':lng', $lng);
        $stmt->bindParam(':description', $description);
        $stmt->bindParam(':image_path', $image_path);

        if ($stmt->execute()) {
            $message = "<div class='message success'>Report submitted successfully!</div>";
        } else {
            $message = "<div class='message error'>Error submitting the report. Please try again.</div>";
        }
    } catch (PDOException $e) {
        $message = "<div class='message error'>Database error: " . $e->getMessage() . "</div>";
    }
}

// Fetch all reports
$reports = [];
try {
    $query = "SELECT crime_type, severity, impact, location, description, image_path, created_at
              FROM reports ORDER BY created_at DESC";
    $stmt = $conn->prepare($query);
    $stmt->execute();
    $reports = $stmt->fetchAll(PDO::FETCH_ASSOC);
} catch (PDOException $e) {
    $message = "<div class='message error'>Error fetching reports: " . $e->getMessage() . "</div>";
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Dashboard</title>
```

```
    <link rel="stylesheet" href="assets/css/style.css">
    <script
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyAEw86vbAZhQXFt8e2uEaruRRPhED3ZmbQ&libraries=places"></script>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

</head>
<body>
    <!-- Header -->
    <div class="header">
        <h1>Environmental Monitoring and Reporting System</h1>
        <a href="logout.php" class="btn-logout">Logout</a>
    </div>

    <!-- Dashboard Layout -->
    <div class="dashboard-container">


<!-- Sidebar -->
<div class="sidebar">
<a href="#" class="menu-btn" onclick="showContent('home')">Home</a>
    <a href="#" class="menu-btn active" onclick="loadContent('submitReport')">Submit a New Report</a>
    <a href="#" class="menu-btn" onclick="loadContent('allReports')">All Reports</a>
    <a href="#" class="menu-btn" onclick="loadContent('climateVisualization')">Climate Visualization</a>
    <a href="#" class="menu-btn" onclick="loadContent('communityEngagement')">Community Engagement</a>
    <a href="#" class="menu-btn" onclick="loadContent('realTimeAlerts')">Real-Time Alerts</a>
    <a href="#" class="menu-btn" onclick="loadContent('environmentalTips')">Environmental Tips</a>
</div>

        <!-- Main Content -->
        <div class="main-content">
            <div id="blankSection" class="blank-section">
            <p>Please select an option from the sidebar to view content.</p></br>
```

```
            <video   autoplay muted loop>
                <source src="video/puppy.mp4" type="video/mp4">
                  Your browser does not support the video tag.
             </video>



        </div>



        <!-- Home Section -->
        <div id="home" class="content-section">
           <h2>Report an Environmental Violation, General Information</h2>
           <h3>General Information</h3>
           <p>EPA seeks your help! Please let us know about potentially harmful
environmental activities in your community or workplace.</p>
           <ul>
              <li><a href="#">What EPA's Enforcement Program Does</a></li>
              <li><a href="#">How You Can Help</a></li>
              <li><a href="#">What is an Environmental Crime?</a></li>
              <li><a href="#">What Not to Report</a></li>
              <li><a href="#">What to do in Case of an Emergency</a></li>
              <li><a href="#">Brochures</a></li>
              <li><a href="#">Reporting Potential Violations by Phone</a></li>
           </ul>
           <h3>What EPA's Enforcement Program Does</h3>
           <p>The mission of EPA's Office of Enforcement and Compliance Assurance
(OECA) is to improve the environment and protect human health by ensuring compliance
with environmental laws and regulations, preventing pollution, and promoting
environmental stewardship...</p>
           <h3>How You Can Help</h3>
           <p>We invite you to help us protect our nation's environment by identifying and
reporting environmental violations...</p>
           <h3>What is an Environmental Crime?</h3>
           <p>Violations of environmental laws take many different forms. Some are done
intentionally and may be criminal violations...</p>
           <h3>What Not to Report</h3>
           <p>EPA does not have jurisdiction over automobile safety, consumer product
safety, foods, medicine, cosmetics, or medical devices...</p>
```

```
            <h3>What to Do in Case of an Emergency</h3>
            <p>If you are experiencing an environmental emergency or are witnessing an environmental event that poses an imminent threat...</p>
        </div>

            <!-- Submit Report -->
            <div id="submitReport" style="display: none;">
  <h2>Submit a New Report</h2>
  <div id="submissionMessage"></div> <!-- For displaying success/error messages -->
  <form id="reportForm" enctype="multipart/form-data">
    <label for="crime_type">Crime Type:</label>
    <select name="crime_type" id="crime_type" required>
      <option value="" disabled selected>Select Crime Type</option>
      <option value="Pollution">Pollution</option>
      <option value="Deforestation">Deforestation</option>
      <option value="Illegal Fishing">Illegal Fishing</option>
      <option value="Wildlife Poaching">Wildlife Poaching</option>
    </select>

    <label for="severity">Severity Level:</label>
    <select name="severity" id="severity" required>
      <option value="Low">Low</option>
      <option value="Medium">Medium</option>
      <option value="High">High</option>
      <option value="Critical">Critical</option>
    </select>

    <label for="location">Location:</label>
    <input type="text" name="location" id="location" placeholder="Type your address here" required>
    <input type="hidden" name="lat" id="lat">
    <input type="hidden" name="lng" id="lng">

    <label for="impact">Impact Scale (Optional):</label>
    <input type="text" name="impact" id="impact">

    <label for="description">Description:</label>
```

```
        <textarea name="description" id="description" rows="4" required></textarea>

        <label for="image">Upload Image:</label>
        <input type="file" name="image" id="image" accept="image/*">

        <button type="button" onclick="submitReport()" class="btn">Submit
Report</button>
    </form>
</div>

            <!-- All Reports -->



            <div id="allReports" style="display: none;">
    <h2>All Reports</h2>
    <!-- Report Summary -->
    <div class="report-summary">
        <div class="card report-card">
            <h3>Total Reports</h3>
            <p><?= $total_reports ?></p>
        </div>
        <div class="card report-card">
            <h3>Reports in Last 24 Hours</h3>
            <p><?= $reports_last_24h ?></p>
        </div>
    </div>

    <!-- Table for Reports -->
    <?php if (count($reports) > 0): ?>
        <table border="1" cellpadding="10">
            <thead>
                <tr>
                    <th>Crime Type</th>
                    <th>Severity</th>
                    <th>Impact</th>
                    <th>Location</th>
```

```
                <th>Description</th>
                <th>Image</th>
                <th>Submitted At</th>
            </tr>
        </thead>
        <tbody>
            <?php foreach ($reports as $report): ?>
                <tr>
                    <td><?= htmlspecialchars($report['crime_type']) ?></td>
                    <td><?= htmlspecialchars($report['severity']) ?></td>
                    <td><?= htmlspecialchars($report['impact']) ?></td>
                    <td><?= htmlspecialchars($report['location']) ?></td>
                    <td><?= htmlspecialchars($report['description']) ?></td>
                    <td>
                        <?php if ($report['image_path']): ?>
                            <img src="../<?= $report['image_path'] ?>" width="100">
                        <?php else: ?>
                            No Image
                        <?php endif; ?>
                    </td>
                    <td><?= htmlspecialchars($report['created_at']) ?></td>
                </tr>
            <?php endforeach; ?>
        </tbody>
    </table>
  <?php else: ?>
    <p>No reports found.</p>
  <?php endif; ?>
</div>

<!-- Climate Visualization Section -->
<div id="climateVisualization">
  <center> <h2>Climate Impact Data Visualization</h2>
   <p>Analyze climate data using AI predictions.</p>
   </center>

<!-- Chart -->
```

```
<h3>Impact Visualization</h3>
   <canvas id="climateChart" width="600" height="300"></canvas>

   <!-- Table -->
   <h3>Predicted Impact Data</h3>
   <table>
      <thead>
         <tr>
            <th>#</th>
            <th>Location</th>
            <th>Impact (sq.m)</th>
            <th>Predicted Impact</th>
         </tr>
      </thead>
      <tbody id="prediction-table-body">
         <!-- Data will be injected dynamically -->
      </tbody>
   </table>


</div>

<!-- Table and Button CSS -->
<style>

   .dashboard-container {
         display: flex;
      }

      .sidebar {
         width: 20%;
         background-color: #343a40;
         padding: 20px;
         color: white;
      }

      .menu-btn {
```

```
            display: block;
            color: white;
            text-decoration: none;
            margin: 10px 0;
            padding: 10px;
            background-color: #007bff;
            border-radius: 5px;
            text-align: center;
        }

        .menu-btn.active {
            background-color: #0056b3;
        }

        .main-content {
            width: 80%;
            background-color: #f8f9fa;
            padding: 20px;
            min-height: 100vh; /* Ensure full height */
        }

        .content-section {
            display: none;
        }

        .content-section.active {
            display: block;
        }
 .button-container {
    text-align: center;
    margin-bottom: 15px;
}
.blank-section {
            height: 100%;
            background-color: #f8f9fa;
            display: flex;
            align-items: center;
```

```
            justify-content: center;
        }

        .blank-section p {
            font-size: 1.5rem;
            color: #6c757d;
        }

.toggle-btn {
    padding: 10px 20px;
    margin: 5px;
    background-color: #0073e6;
    color: white;
    border: none;
    cursor: pointer;
    border-radius: 5px;
    font-size: 1rem;
    transition: 0.3s;
}

.toggle-btn:hover {
    background-color: #005bb5;
}

.chart-container, .table-container {
    margin: 0 auto;
    max-width: 800px;
}

table.prediction-table {
    width: 100%;
    border-collapse: collapse;
    text-align: left;
    margin-top: 20px;
}

table.prediction-table th, table.prediction-table td {
```

```
    border: 1px solid #ddd;
    padding: 8px;
}

table.prediction-table th {
    background-color: #f4f4f4;
    font-weight: bold;
}

.impact-low {
    color: #28a745;
    font-weight: bold;
}
.impact-moderate {
    color: #ffc107;
    font-weight: bold;
}
.impact-high {
    color: #ff5722;
    font-weight: bold;
}
.impact-severe {
    color: #dc3545;
    font-weight: bold;
}

ul {
            list-style-type: disc;
            padding-left: 20px;
        }

        ul li a {
            text-decoration: none;
            color: #007bff;
        }

        ul li a:hover {
```

```
            text-decoration: underline;
        }
        #threads-container {
    margin-bottom: 20px;
}

.thread {
    background-color: #f8f9fa;
    border: 1px solid #ddd;
    padding: 10px;
    margin-bottom: 10px;
    border-radius: 5px;
}

.thread h4 {
    margin: 0;
    font-size: 1.2rem;
    color: #007bff;
}

.thread p {
    margin: 10px 0;
    color: #555;
}

.thread small {
    display: block;
    color: #aaa;
}

/* Home Section Styling */
#home {
    background-color: #ffffff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0px 4px 8px rgba(0, 0, 0, 0.1);
    color: #333;
```

```
}

#home h2 {
    font-size: 2rem;
    color: #007bff;
    margin-bottom: 10px;
    border-bottom: 2px solid #007bff;
    display: inline-block;
    padding-bottom: 5px;
}

#home h3 {
    font-size: 1.5rem;
    color: #333;
    margin-top: 20px;
    margin-bottom: 10px;
}

#home p {
    font-size: 1rem;
    line-height: 1.6;
    color: #555;
}

#home ul {
    list-style: none;
    padding: 0;
    margin: 15px 0;
}

#home ul li {
    margin-bottom: 10px;
    display: flex;
    align-items: center;
    font-size: 1rem;
    color: #007bff;
}
```

```
#home ul li a {
    color: #007bff;
    text-decorati
    footer {
    background-color: #333;
    color: #fff;
    padding: 20px 0;
}
footer {
    position: fixed; /* Keeps the footer at the bottom of the page */
    bottom: 0; /* Aligns the footer to the bottom */
    left: 0; /* Starts from the very left of the screen */
    width: 100%; /* Makes the footer span the full width */
    background-color: #2C3E50; /* Dark blue background */
    color: #ECF0F1; /* Light text color */
    padding: 20px 0; /* Padding for content inside the footer */
    text-align: center; /* Centers text and content */
    box-shadow: 0px -2px 5px rgba(0, 0, 0, 0.2); /* Optional: Adds a subtle shadow at the
top of the footer */
    z-index: 100; /* Ensures it stays above other content */
}

footer .footer-container {
    display: flex; /* Creates a horizontal layout */
    flex-wrap: wrap; /* Adjusts layout for smaller screens */
    justify-content: space-around; /* Spreads the sections evenly */
    max-width: 1200px; /* Optional: Sets a max-width for content */
    margin: auto; /* Centers the footer content */
}

footer h4 {
    font-size: 18px;
    margin-bottom: 10px;
    font-weight: bold;
}
```

```
footer ul {
    list-style: none;
    padding: 0;
    margin: 0;
}

footer ul li {
    margin-bottom: 5px;
}

footer ul li a {
    color: #ECF0F1;
    text-decoration: none;
    font-size: 14px;
}

footer ul li a:hover {
    text-decoration: underline;
    color: #1ABC9C;
}

footer .social-links a {
    font-size: 18px;
    margin: 0 10px;
    color: #ECF0F1;
    text-decoration: none;
}

footer .social-links a:hover {
    color: #1ABC9C;
}

footer .footer-bottom {
    font-size: 12px;
    color: #95A5A6;
    margin-top: 10px;
}
```

```
</style>


<!-- Chart.js Library -->
<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

<script>
document.addEventListener("DOMContentLoaded", function () {
   const tableBody = document.getElementById("prediction-table-body");

   // Fetch predictions from AI service
   fetch("http://127.0.0.1:8000/predict_from_db")
      .then((response) => {
         if (!response.ok) throw new Error("Failed to fetch predictions");
         return response.json();
      })
      .then((data) => {
         tableBody.innerHTML = "";
         data.forEach((item, index) => {
            const row = document.createElement("tr");
            row.innerHTML = `
               <td>${index + 1}</td>
               <td>${item.report.address || "Unknown Address"}</td>
               <td>${item.report.impact || 0} sq.m</td>
               <td>${item.prediction}</td>
            `;
            tableBody.appendChild(row);
         });
         renderChart(data);
      })
      .catch((error) => console.error("Error fetching predictions:", error));
});
function navigateTo(sectionId) {
   const sections = ['home', 'submitReport', 'allReports', 'climateVisualization',
'communityEngagement', 'realTimeAlerts', 'environmentalTips'];
   sections.forEach(id => document.getElementById(id).style.display = 'none');
```

```
    document.getElementById(sectionId).style.display = 'block';
    document.querySelectorAll('.menu-btn').forEach(button =>
button.classList.remove('active'));
    document.querySelector(`.menu-
btn[onclick="showContent('${sectionId}')"]`).classList.add('active');
}

function renderChart(data) {
    const ctx = document.getElementById('climateChart').getContext('2d');

    const labels = data.map(item => item.report.address || "Unknown Address");
    const values = data.map(item => item.report.impact || 0);

    new Chart(ctx, {
        type: 'bar',
        data: {
            labels: labels,
            datasets: [{
                label: "Predicted Impact (sq.m)",
                data: values,
                backgroundColor: 'rgba(0, 123, 255, 0.7)',
                borderColor: 'rgba(0, 123, 255, 1)',
                borderWidth: 1
            }]
        },
        options: {
            responsive: true,
            plugins: {
                title: {
                    display: true,
                    text: "AI Climate Impact Predictions"
                }
            },
            scales: {
                y: {
                    beginAtZero: true
                }
```

```
        }
      }
   });
}

</script>



            <!-- Community Engagement -->
            <div id="communityEngagement" class="content-section">
   <h2>Community Discussions</h2>

   <!-- Existing Threads -->
   <div id="threads-container">
      <!-- Threads will be dynamically loaded here -->
   </div>

   <!-- New Thread Form -->
   <h3>Start a New Discussion</h3>
   <form id="newThreadForm">
      <label for="threadTitle">Title:</label>
      <input type="text" id="threadTitle" name="title" placeholder="Enter discussion title"
required>

      <label for="threadContent">Content:</label>
      <textarea id="threadContent" name="content" placeholder="Write your discussion
content" rows="4" required></textarea>

      <button type="button" onclick="submitThread()">Post Thread</button>
   </form>
</div>

            <!-- Real-Time Alerts Map Section -->
            <div id="realTimeAlerts" style="display: none;">
   <h2>Real-Time Alerts Map</h2>
   <p>View live updates of environmental reports:</p>
   <div id="map" style="height: 400px; width: 100%;"></div>
```

```
</div>

        <!-- Environmental Tips Section -->
        <div id="environmentalTips" style="display: none;">
    <h2>Environmental Tips</h2>
    <ul>
        <li>Reduce, reuse, and recycle to minimize waste.</li>
        <li>Save energy by using energy-efficient appliances.</li>
        <li>Plant trees to combat deforestation and absorb CO₂.</li>
        <li>Support clean energy sources like solar and wind power.</li>
        <li>Reduce your carbon footprint by using public transport.</li>
        <li>Report environmental crimes like pollution and illegal logging.</li>
    </ul>
</div>


    <!-- JavaScript -->
    <script>

function fetchPredictions() {
    fetch('http://127.0.0.1:8000/predict_from_db') // Connect to AI service
        .then(response => {
            if (!response.ok) {
                throw new Error(`HTTP error! Status: ${response.status}`);
            }
            return response.json();
        })
        .then(data => {
            console.log("Fetched Predictions:", data); // Log data for debugging

            if (data.error) {
                document.getElementById('ai-prediction').innerText = "Error: " + data.error;
                console.error("Server Error:", data.error);
                return;
            }

            const labels = data.map((item, index) => `Location ${index + 1}`);
```

```
        const predictions = data.map(item => item.prediction);

        // Generate chart
        const ctx = document.getElementById('climateChart').getContext('2d');
        new Chart(ctx, {
            type: 'bar',
            data: {
                labels: labels,
                datasets: [{
                    label: "Impact Predictions",
                    data: predictions,
                    backgroundColor: '#4BC0C0',
                    borderColor: '#0073e6',
                    borderWidth: 1
                }]
            },
            options: {
                responsive: true,
                plugins: {
                    title: {
                        display: true,
                        text: 'AI Climate Impact Predictions'
                    }
                },
                scales: {
                    y: { beginAtZero: true }
                }
            }
        });

        // Display messages
        const messages = data.map((item, index) =>
            `<li>Location ${index + 1}: ${item.report.impact} sq.m - Predicted Impact:
${item.prediction}</li>`
        ).join('');
        document.getElementById('prediction-messages').innerHTML =
`<ul>${messages}</ul>`;
```

```
        })
        .catch(error => {
            console.error("Error fetching predictions:", error);
            document.getElementById('ai-prediction').innerText = "Error fetching
predictions.";
        });
}

document.addEventListener('DOMContentLoaded', fetchPredictions);

     function loadContent(section) {
    const sections = ['home', 'submitReport', 'allReports', 'climateVisualization',
'communityEngagement', 'realTimeAlerts', 'environmentalTips'];
    sections.forEach(id => document.getElementById(id).style.display = 'none');
    document.getElementById(section).style.display = 'block';
    document.querySelectorAll('.menu-btn').forEach(button =>
button.classList.remove('active'));
    event.target.classList.add('active');
}

                function initAutocomplete() {
    const input = document.getElementById('location');
    const autocomplete = new google.maps.places.Autocomplete(input);

    // Extract latitude and longitude when an address is selected
    autocomplete.addListener('place_changed', function () {
        const place = autocomplete.getPlace();
        if (place.geometry) {
            document.getElementById('lat').value = place.geometry.location.lat();
            document.getElementById('lng').value = place.geometry.location.lng();
        }
    });
}

        document.addEventListener('DOMContentLoaded', initAutocomplete);

        document.addEventListener('DOMContentLoaded', initMap);
```

```
    function initMap() {
  const map = new google.maps.Map(document.getElementById("map"), {
    center: { lat: 40.7128, lng: -74.0060 }, // Default center (New York)
    zoom: 5
  });

  fetch("dashboard.php?action=fetch_reports")
    .then(response => response.json())
    .then(data => {
      data.forEach(report => {
        const marker = new google.maps.Marker({
          position: { lat: parseFloat(report.lat), lng: parseFloat(report.lng) },
          map: map,
          title: report.crime_type,
        });

        const infoWindow = new google.maps.InfoWindow({
          content: `<h4>${report.crime_type}</h4>
                <p>${report.description}</p>
                <p><strong>Location:</strong> ${report.location}</p>`,
        });

        marker.addListener("click", () => {
          infoWindow.open(map, marker);
        });
      });
    })
    .catch(error => console.error("Error fetching report data:", error));
}

function submitReport() {
  const formData = new FormData(document.getElementById('reportForm'));

  fetch('dashboard.php', {
    method: 'POST',
    body: formData
```

```
    })
    .then(response => response.text())
    .then(data => {
        document.getElementById('submissionMessage').innerHTML = '<div class="message success">Report submitted successfully!</div>';
        document.getElementById('reportForm').reset();
    })
    .catch(error => {
        document.getElementById('submissionMessage').innerHTML = '<div class="message error">Error submitting report. Please try again.</div>';
        console.error('Error:', error);
    });
}
// Function to toggle content
function showContent(sectionId) {
    const sections = document.querySelectorAll('.content-section'); // All content sections
    const buttons = document.querySelectorAll('.menu-btn'); // All sidebar buttons
    const blankSection = document.getElementById('blankSection'); // Blank section

    // Hide all content sections
    sections.forEach(section => section.classList.remove('active'));
    blankSection.style.display = 'none';

    // Remove the active class from all buttons
    buttons.forEach(button => button.classList.remove('active'));

    // Show the selected content section
    const selectedSection = document.getElementById(sectionId);
    if (selectedSection) {
        selectedSection.classList.add('active'); // Make selected section visible
    }

    // Highlight the selected button
    event.target.classList.add('active');
}
```

```
        // Initially show blank section
        document.addEventListener("DOMContentLoaded", () => {
    document.getElementById('blankSection').style.display = 'flex'; // Show blank section
initially

    const buttons = document.querySelectorAll('.menu-btn');
    buttons.forEach(button => {
        button.addEventListener('click', (event) => {
            const sectionId = event.target.getAttribute('onclick').match(/'([^']+)'/)[1];
            showContent(sectionId);
        });
    });
});
document.addEventListener('DOMContentLoaded', () => {
    // Fetch existing threads on page load
    fetchThreads();
});

function fetchThreads() {
    fetch('dashboard.php?action=fetch_threads')
        .then(response => response.json())
        .then(data => {
            const container = document.getElementById('threads-container');
            container.innerHTML = ''; // Clear existing content

            if (data.error) {
                container.innerHTML = `<p>Error: ${data.error}</p>`;
                return;
            }

            if (data.length === 0) {
                container.innerHTML = '<p>No discussions found. Start a new one!</p>';
                return;
            }

            data.forEach(thread => {
                const threadDiv = document.createElement('div');
```

```
            threadDiv.classList.add('thread');
            threadDiv.innerHTML = `
                <h4>${thread.title}</h4>
                <p>${thread.content}</p>
                <small>Posted on: ${new Date(thread.created_at).toLocaleString()}</small>
            `;
            container.appendChild(threadDiv);
        });
    })
    .catch(error => console.error('Error fetching threads:', error));
}

function submitThread() {
    const title = document.getElementById('threadTitle').value;
    const content = document.getElementById('threadContent').value;

    if (!title || !content) {
        alert('Please fill in both the title and content.');
        return;
    }

    fetch('dashboard.php', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/x-www-form-urlencoded',
        },
        body: new URLSearchParams({
            action: 'create_thread',
            title: title,
            content: content,
        }),
    })
        .then(response => response.json())
        .then(data => {
            if (data.error) {
                alert(`Error: ${data.error}`);
            } else {
```

```
            alert(data.success);
            document.getElementById('newThreadForm').reset();
            fetchThreads(); // Refresh the thread list
        }
    })
    .catch(error => console.error('Error submitting thread:', error));
}

  </script>

</body>
</html>
```

Logout.php

```
<?php
session_start();
session_destroy();
header("Location: login.php");
exit();
?>
```

Processreport.php

```
<?php
session_start();
require_once '../config/Database.php';

if (!isset($_SESSION['user_id'])) {
    header("Location: login.php");
    exit();
```

```
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $database = new Database();
    $db = $database->connect();

    $user_id = $_SESSION['user_id'];
    $crime_type = $_POST['crime_type'];
    $location = $_POST['location'];
    $description = $_POST['description'];

    try {
        if (isset($_FILES['image']) && $_FILES['image']['error'] === UPLOAD_ERR_OK) {
            $image_name = time() . '_' . basename($_FILES['image']['name']);
            $target_path = __DIR__ . "/uploads/" . $image_name;

            if (move_uploaded_file($_FILES['image']['tmp_name'], $target_path)) {
                $image_path = "uploads/" . $image_name;

                $query = "INSERT INTO reports (user_id, crime_type, location, description,
image_path)
                        VALUES (:user_id, :crime_type, :location, :description, :image_path)";
                $stmt = $db->prepare($query);
                $stmt->bindParam(':user_id', $user_id);
                $stmt->bindParam(':crime_type', $crime_type);
                $stmt->bindParam(':location', $location);
                $stmt->bindParam(':description', $description);
                $stmt->bindParam(':image_path', $image_path);

                if ($stmt->execute()) {
                    $_SESSION['success_message'] = "Report submitted successfully!";
                    header("Location: dashboard.php");
                    exit();
                }
            } else {
                throw new Exception("Failed to upload the image.");
            }
```

```
        } else {
            throw new Exception("Please upload an image.");
        }
    } catch (Exception $e) {
        $_SESSION['error_message'] = $e->getMessage();
        header("Location: dashboard.php");
        exit();
    }
}
?>
```

Submitreport.php

```
<?php
session_start();
if (!isset($_SESSION['user'])) {
    header("Location: login.php");
    exit();
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Submit a New Report</title>
    <link rel="stylesheet" href="assets/css/style.css">
</head>
<body>
    <!-- Header Section -->
    <div class="header">
        <h1>Environmental Monitoring and Reporting System</h1>
        <a href="logout.php" class="btn-logout">Logout</a>
    </div>
```

```
    <!-- Dashboard Container -->
    <div class="dashboard-container">
        <!-- Sidebar -->
        <div class="sidebar">
            <a href="submit_report.php" class="menu-btn active">Submit a New Report</a>
            <a href="#" class="menu-btn">All Reports</a>
            <a href="#" class="menu-btn">Climate Impact Data Visualization</a>
            <a href="#" class="menu-btn">Community Engagement</a>
            <a href="#" class="menu-btn">Real-Time Alerts Map</a>
        </div>

        <!-- Main Content -->
        <div class="main-content">
            <h2>Submit a New Report</h2>
            <form action="process_report.php" method="POST" enctype="multipart/form-
data">
    <label for="crime_type">Crime Type:</label>
    <select name="crime_type" id="crime_type" required>
        <option value="" disabled selected>Select Crime Type</option>
        <option value="Pollution">Pollution</option>
        <option value="Deforestation">Deforestation</option>
        <option value="Illegal Fishing">Illegal Fishing</option>
        <option value="Wildlife Poaching">Wildlife Poaching</option>
        <option value="Other">Other</option>
    </select>

    <label for="location">Location:</label>
    <input type="text" name="location" id="location" placeholder="e.g., City A, Forest
Area" required>

    <!-- Map Integration -->
    <iframe
src="https://www.google.com/maps/embed?pb=!1m14!1m12!1m3!1d12093.13651913831!
2d-
74.0425728!3d40.7337731!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!5e0!3m2!1sen!2su
s!4v1731605465672!5m2!1sen!2sus"
            width="100%" height="300" style="border:0;" allowfullscreen=""
loading="lazy"></iframe>
```

```
    <label for="description">Description:</label>
    <textarea name="description" id="description" placeholder="Describe the issue"
rows="4" required></textarea>

    <label for="image">Upload Image:</label>
    <input type="file" name="image" id="image" accept="image/*">

    <button type="submit" class="btn">Submit Report</button>
</form>

        </div>
    </div>

    <!-- Footer -->
    <div class="footer">
        &copy; 2025 Environmental Monitoring and Reporting System
    </div>
</body>
</html>
```

Adminprocess.php

```
<?php
session_start();
require_once '../config/database.php';

$database = new Database();
$conn = $database->connect();

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $email = $_POST['admin_email'];
    $password = $_POST['admin_password'];

    $query = "SELECT * FROM admins WHERE email = :email";
    $stmt = $conn->prepare($query);
    $stmt->bindParam(':email', $email);
```

```
    $stmt->execute();

    $admin = $stmt->fetch(PDO::FETCH_ASSOC);

    if ($admin && $password === $admin['password']) { // Assuming plain text passwords
        $_SESSION['success'] = "Welcome Admin!";
        header("Location: admin_dashboard.php");
    } else {
        $_SESSION['error'] = "Invalid Email or Password!";
        header("Location: admin_login.php");
    }
}
?>
```

Database.sql

```
-- Create Database
CREATE DATABASE IF NOT EXISTS emrs_db;

-- Use the Database
USE emrs_db;

-- Create Admins Table
CREATE TABLE admins (
    id INT AUTO_INCREMENT PRIMARY KEY,
    firstname VARCHAR(100) NOT NULL,
    lastname VARCHAR(100),
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Create the 'users' table for storing user information
CREATE TABLE users (
```

```
    id INT AUTO_INCREMENT PRIMARY KEY,
    firstname VARCHAR(50) NOT NULL,
    lastname VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    role ENUM('user', 'admin') DEFAULT 'user',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Create Reports Table
CREATE TABLE reports (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    crime_type VARCHAR(100) NOT NULL,
    severity ENUM('Low', 'Medium', 'High') NOT NULL,
    description TEXT,
    location VARCHAR(255) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

-- Create User Logins Table
CREATE TABLE user_logins (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    login_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);

-- Create Admin Logins Table (Optional)
CREATE TABLE admin_logins (
    id INT AUTO_INCREMENT PRIMARY KEY,
    admin_id INT NOT NULL,
    login_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (admin_id) REFERENCES admins(id) ON DELETE CASCADE
);
```

```
CREATE TABLE community_threads (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    content TEXT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Insert Sample Admin Data
INSERT INTO admins (firstname, lastname, email, password) VALUES
('Admin', 'User', 'admin@example.com', 'admin123');

-- Insert Sample Users Data
INSERT INTO users (firstname, lastname, email, password) VALUES
('John', 'Doe', 'john@example.com', 'password123'),
('Jane', 'Smith', 'jane@example.com', 'password123');

-- Insert Sample Reports Data
INSERT INTO reports (user_id, crime_type, severity, description, location) VALUES
(1, 'Pollution', 'High', 'Industrial waste dumping in the river', 'River City'),
(2, 'Deforestation', 'Medium', 'Illegal tree cutting in forest area', 'Pine Woods');

-- Insert Sample User Logins Data
INSERT INTO user_logins (user_id) VALUES
(1), (2);

-- Insert Sample Admin Logins Data (Optional)
INSERT INTO admin_logins (admin_id) VALUES
(1);
```

Database.config

```
<?php
class Database {
    private $host = 'localhost';
    private $dbname = 'emrs_db'; // Correct database name
```

```
    private $username = 'root';
    private $password = '';
    public $conn;

    public function connect() {
        $this->conn = null;
        try {
            $this->conn = new PDO(
                "mysql:host=" . $this->host . ";dbname=" . $this->dbname,
                $this->username,
                $this->password
            );
            $this->conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
        } catch (PDOException $e) {
            echo "Connection Error: " . $e->getMessage();
        }
        return $this->conn;
    }
}
?>
```

Usercontrollers.php

```
<?php
require_once '../config/Database.php';
require_once '../models/User.php';

class UserController {
    private $db;

    public function __construct() {
        $database = new Database();
        $this->db = $database->connect();
    }

    public function loginUser() {
```

```
    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        $user = new User($this->db);
        $user->email = $_POST['email'];
        $user->password = $_POST['password'];
        $result = $user->login();

        if ($result) {
            // Set session variables
            $_SESSION['user_id'] = $result['id'];
            $_SESSION['user'] = $result['firstname'];
            echo "<script>
                    alert('Login Successful!');
                    window.location.href = 'dashboard.php';
                  </script>";
            exit();
        } else {
            echo "<script>
                    alert('Wrong Email or Password!');
                    window.location.href = 'login.php';
                  </script>";
            exit();
        }
    }
}

public function registerUser() {
    if ($_SERVER['REQUEST_METHOD'] === 'POST') {
        $firstname = $_POST['firstname'];
        $lastname = $_POST['lastname'];
        $email = $_POST['email'];
        $password = password_hash($_POST['password'], PASSWORD_DEFAULT);

        $query = "INSERT INTO users (firstname, lastname, email, password)
                VALUES (:firstname, :lastname, :email, :password)";
        $stmt = $this->db->prepare($query);

        $stmt->bindParam(':firstname', $firstname);
```

```
            $stmt->bindParam(':lastname', $lastname);
            $stmt->bindParam(':email', $email);
            $stmt->bindParam(':password', $password);

            try {
                if ($stmt->execute()) {
                    echo "Registration successful! <a href='index.php'>Login here</a>";
                }
            } catch (PDOException $e) {
                echo "Error: " . $e->getMessage();
            }
        }
    }
}
?>
```

User.php

```
<?php
class User {
    private $conn;
    private $table = 'users';

    public $email;
    public $password;

    public function __construct($db) {
        $this->conn = $db;
    }

    public function login() {
        $query = "SELECT id, firstname, email, password FROM " . $this->table . " WHERE
email = :email LIMIT 1";
        $stmt = $this->conn->prepare($query);
        $stmt->bindParam(':email', $this->email);
```

```
        $stmt->execute();
        $user = $stmt->fetch(PDO::FETCH_ASSOC);

        if ($user && password_verify($this->password, $user['password'])) {
            return $user;
        }
        return false;
    }
}
?>
```

5Test Document

5.1 Introduction

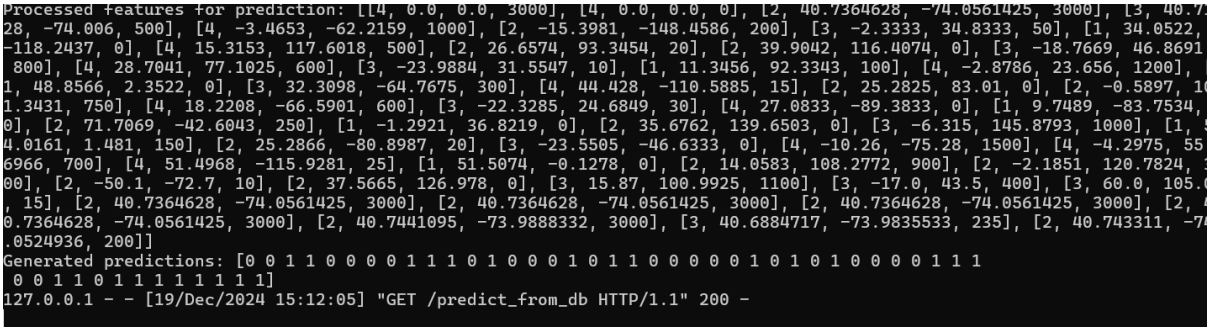
Testing is a crucial phase in software development to ensure the Environmental Monitoring and Reporting System (EMRS) functions as expected. This section outlines the testing methodologies used to verify the integrity, performance, and security of the system. The goal of testing is to identify and resolve bugs, validate features, and ensure that the system meets the requirements for end users and administrators.

5.2 Testing Methodologies

5.2.1 Unit Testing

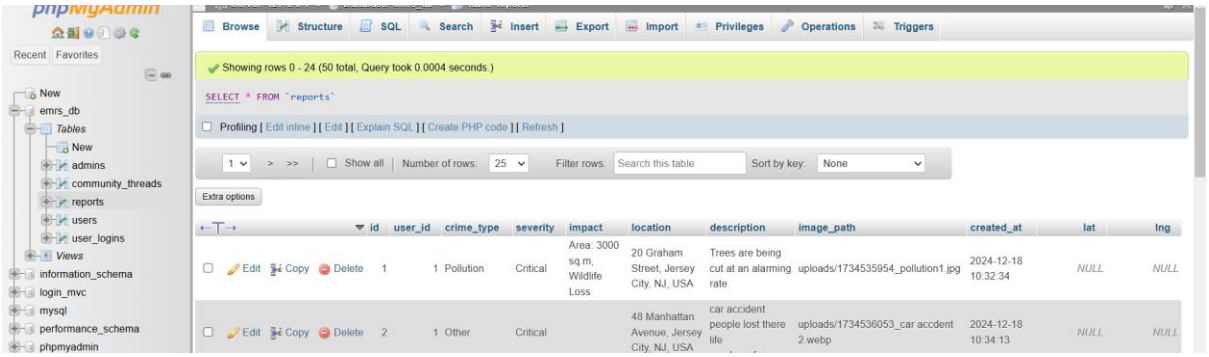
Objective: Test individual components and functions of the system to ensure they work as expected.

- **Scope**:
    - Testing individual PHP functions in the UserController, ReportController, and database interactions.
    - Verifying the AI system's functionality for predicting environmental impact as shown in below fig



fig

- Checking the input fields of forms (e.g., registration, login, report submission)



fig

Objective: Verify that different modules of the system interact correctly.

- **Scope**:
  - Integration between the PHP frontend and MySQL database.
  - Communication between the AI Flask server and the PHP system.
  - Integration of Google Maps API with the real-time alerts map feature ash shown in bellow fig.

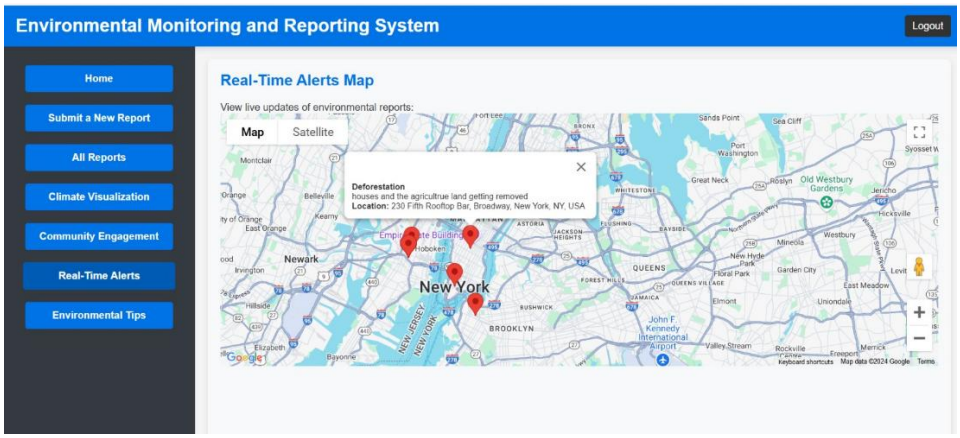


Fig real time alerts

5.2.3 Security Testing

Objective: Ensure the system is protected against potential threats such as unauthorized access or data breaches.

- **Scope:**
    - Validating user authentication and authorization mechanisms.
    - Testing password encryption and secure database storage.
    - Preventing SQL injection attacks by sanitizing user input.

5.2.4 White-Box Testing

Objective: Test the internal structures and logic of the code.

- **Scope:**
    - Reviewing the PHP codebase for potential logic errors.
    - Verifying the control flow and decision-making in the AI Flask application.

5.2.5 Black-Box Testing

Objective: Validate the system's functionality without knowing the internal code structure.

- **Scope:**
    - Testing the user interface and functionality from an end-user perspective.
    - Validating the correctness of the outputs based on inputs.

5.3 Test Cases

Test Cases for EMRS Project

Test ID	Test Name	Description	Expected Result	Actual Result	Status
TC-001	Login Functionality	User logs in with valid credentials.	Redirect to Dashboard.	Pass	Pass
TC-002	Registration Process	User registers with valid details.	User is redirected to login page.	Pass	Pass
TC-003	Submit Report	User submits a valid environmental report with required details.	Report is saved in the database.	Pass	Pass
TC-004	View Reports	User views all submitted reports.	Reports are displayed in the table.	Pass	Pass

Test ID	Test Name	Description	Expected Result	Actual Result	Status
TC-005	AI Predictions	AI processes environmental reports to provide impact predictions.	Predicted impact is shown.	Pass	Pass
TC-006	Community Discussions	User starts a new discussion thread in the community engagement section.	Thread is saved and displayed.	Pass	Pass
TC-007	Real-Time Alerts Map	User views real-time environmental alerts on the map.	Alerts are plotted on the map.	Pass	Pass
TC-008	Environmental Tips	User views environmental tips on the dedicated section.	Tips are displayed correctly.	Pass	Pass
TC-009	Admin Dashboard Access	Admin accesses the dashboard to manage reports and user accounts.	Admin dashboard is loaded successfully.	Pass	Pass
TC-010	Security Features	System prevents unauthorized access to secure sections.	Access is denied for unauthorized users.	Pass	Pass

These test cases ensure that each core functionality of the Environmental Monitoring and Reporting System (EMRS) operates as expected, from user interaction to AI integration and admin

---

Adding Database Creation Pictures

For the database section, include:

1. Screenshots of creating the database in phpMyAdmin.
2. Tables created, such as users, reports, predictions, etc.

3. Imported SQL file structure.

For example:

- **Step 1:** Open phpMyAdmin and click on "New" to create a database.
- **Step 2:** Name the database emrs and click "Create."
- **Step 3:** Import emrs.sql from the "Import" tab.
- **Step 4:** Verify tables like users, reports, and predictions.

6 Data Abstraction

Data abstraction in the **Environmental Monitoring and Reporting System (EMRS)** involves structuring and presenting data in a way that users and administrators can interact with it effectively while hiding the underlying complexities of storage and processing. By adopting a layered approach, the system ensures the right information is exposed to the relevant stakeholders, while backend systems handle intricate details seamlessly.

6.1 User-Level Abstraction

At the user level, the system simplifies data interaction to enhance usability. Users interact with high-level, intuitive interfaces while the complexities of data processing are hidden.

Key Features:

- **Form-Based Input**: Users submit environmental reports through an easy-to-use form without interacting with raw database fields.
- **Interactive Maps**: Real-time environmental crimes and alerts are visualized using map markers, abstracting geolocation data (latitude, longitude) into meaningful visuals.
- **Readable Visualizations**: Climate trends and predictions are displayed as graphs and charts for easier understanding, abstracting raw numerical data.

Examples:

- **Reports**: Users view fields such as "Crime Type," "Severity," "Location," and "Description" instead of raw database details.
- **Alerts**: Real-time alerts are displayed on a map, abstracting technical data into user-friendly visuals.

6.2 Administrator-Level Abstraction

Administrators are provided with abstracted views and functionalities to simplify system management while offering more data access than regular users.

Key Features:

- **Dashboard View**: Administrators access summarized data, such as total reports and daily statistics, instead of querying raw data.
- **Moderation Tools**: Admins approve or reject user-submitted reports through a simplified interface.
- **Community Moderation**: Administrators manage threads and user discussions intuitively without dealing with backend data structures.

Examples:

- **Report Management**: Admins see a tabular representation of user reports with options to take actions such as escalate or delete reports.
- **Data Insights**: Dashboards display graphical summaries, abstracting complex SQL queries.

6.3 Backend Data Abstraction

The backend system processes raw data and generates meaningful outputs for both users and administrators.

Key Features:

- **Relational Data Management**: Data is stored in normalized database tables for reports, users, predictions, and threads.
- **API Integration**: APIs simplify complex backend processes such as AI prediction processing and data retrieval.
- **Data Security**: Backend layers ensure only authorized access, abstracting and protecting sensitive data.

Examples:

- **AI Integration**: Raw data such as incident severity and location is processed, and the AI module returns abstracted predictions like "High Impact."
- **Storage**: User reports with timestamps, user IDs, and geolocations are saved in the database while hiding internal mechanisms from the user.

6.4 Database Design and Abstraction

The database serves as the system's backbone, storing all data securely while providing abstracted access through APIs and backend queries.

Key Features:

- **Normalized Tables**: Data is structured into separate tables (e.g., users, reports, threads) with relationships defined via primary and foreign keys.
- **Query Abstraction**: Prepared SQL statements and Object-Relational Mapping (ORM) simplify database interactions.
- **Abstracted Views**: Complex queries and joins are abstracted into easy-to-use API endpoints.

Tables and Examples:

1. **Users Table**:
    - Fields: user_id, username, password, role
    - Abstracted as user profiles for login and permissions.
2. **Reports Table**:
    - Fields: report_id, crime_type, severity, description, lat, lng, created_at
    - Abstracted as structured reports displayed on dashboards and maps.
3. **Community Threads Table**:
    - Fields: thread_id, title, content, created_at
    - Abstracted as discussion threads for community engagement.
4. **Predictions Table**:
    - Fields: prediction_id, location, impact, predicted_value
    - Abstracted as climate predictions displayed graphically.

6.5 AI Data Abstraction

The AI module processes raw data to provide insights and predictions in an abstracted format suitable for decision-making.

Key Features:

- **Input**: Raw data includes location, severity, and incident type.
- **Processing**: Machine learning algorithms predict environmental trends such as deforestation or pollution risk.
- **Output**: Predictions are abstracted into graphs, charts, and simple risk levels (e.g., "High Impact").

Examples:

- **Visualization**: AI output is displayed as bar charts and line graphs.
- **Alerts**: AI assigns priority levels to alerts, abstracting computations into actionable visuals.

6.6 Interaction Between Abstraction Layers

The system seamlessly integrates data abstraction across user, admin, and backend layers.

1. **User Input**:
    - Users submit reports via forms.
    - Backend processes the input and saves it in the database.
2. **Admin Dashboard**:
    - Admins review reports and moderate discussions via the dashboard.
    - Backend manages database updates and logs actions.
3. **Data Visualization**:

- AI processes raw data into predictions.
- Graphical outputs are displayed on dashboards and maps.

By leveraging abstraction, the EMRS ensures users and administrators can interact with data intuitively while maintaining system security and efficiency.

7. Summary

The **Environmental Monitoring and Reporting System (EMRS)** is a comprehensive platform designed to address environmental challenges. By empowering users to report crimes such as deforestation, pollution, and illegal fishing, the system bridges the gap between individuals and authorities. Key features include:

1. **Real-Time Alerts**: Interactive maps display alerts, making data actionable.
2. **AI-Driven Insights**: Predictions provide actionable climate impact data.
3. **Community Engagement**: Discussion forums promote collective awareness and solutions.
4. **User and Admin Tools**: Separate interfaces cater to user submissions and admin management.

With its combination of PHP, MySQL, and Python (Flask), EMRS integrates advanced AI capabilities while maintaining a user-friendly interface. By simplifying data interactions through abstraction, the platform fosters collaboration, promotes ecological stewardship, and accelerates response to environmental challenges.

8. User Manual