Emergency Response Coordination Hub - Complete Blueprint

Project Overview

The Emergency Response Coordination Hub is a web-based platform that enables efficient management of emergency incidents, volunteers, and resources during crisis situations. This demo project will showcase a robust RESTful API with full CRUD operations, real-time coordination features, and role-based access control.

Core Components

1. User Roles & Authentication

Public Users: Can report incidents

Volunteers: Can view and respond to incidents

Coordinators: Can manage incidents, assign volunteers, and oversee operations

Authentication: JWT-based authentication system

Authorization: Role-based access control (RBAC)

2. Incident Management

Create: Report new incidents with type, location, description, severity

Read: View incidents with filtering (status, type, location)

Update: Change incident status (reported → assigned → in progress → resolved)

Delete: Archive resolved incidents (coordinators only)

3. Volunteer Management

Create: Register volunteers with skills, availability, contact info

Read: View volunteers with filtering (skills, availability, location)

Update: Modify volunteer profiles and availability

Delete: Deactivate volunteer accounts

4. Assignment Management

Create: Assign volunteers to incidents

Read: View assignments by incident or volunteer

Update: Change assignment status (assigned → accepted → completed)

Delete: Remove assignments

5. Real-time Features

Live map showing incidents and volunteers

Real-time status updates (using WebSockets)

Notification system for assignments and updates

6. Dashboard

Coordinator Dashboard: Overview of all incidents, assignments, volunteer availability

Volunteer Dashboard: Assigned incidents and status updates

Public Dashboard: Report incidents and view general status

Technology Stack

Backend: Node.js with Express.js

Database: PostgreSQL (with Sequelize ORM)

Frontend: React.js

Authentication: JWT (JSON Web Tokens)

Mapping: Leaflet with OpenStreetMap

Real-time: Socket.io for WebSockets

Styling: Material-UI or Bootstrap

Database Schema

Users Table

CREATE TABLE Users (

id SERIAL PRIMARY KEY,

username VARCHAR(50) UNIQUE NOT NULL,

email VARCHAR(100) UNIQUE NOT NULL,

password VARCHAR(255) NOT NULL,

role VARCHAR(20) NOT NULL CHECK (role IN ('public', 'volunteer', 'coordinator')),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

Incidents Table

CREATE TABLE Incidents (

id SERIAL PRIMARY KEY,

type VARCHAR(50) NOT NULL,

latitude DECIMAL(10, 8) NOT NULL,

longitude DECIMAL(11, 8) NOT NULL,

description TEXT,

severity VARCHAR(20) NOT NULL CHECK (severity IN ('low', 'medium', 'high', 'critical')),

status VARCHAR(20) NOT NULL CHECK (status IN ('reported', 'assigned', 'in progress', 'resolved')),

reported\_by INTEGER REFERENCES Users(id),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

Volunteers Table

CREATE TABLE Volunteers (

id SERIAL PRIMARY KEY,

user\_id INTEGER UNIQUE REFERENCES Users(id),

skills TEXT[], -- Array of skills (e.g., ['first aid', 'rescue'])

availability BOOLEAN DEFAULT true,

current\_latitude DECIMAL(10, 8),

current\_longitude DECIMAL(11, 8),

status VARCHAR(20) DEFAULT 'available' CHECK (status IN ('available', 'assigned', 'unavailable')),

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

Assignments Table

CREATE TABLE Assignments (

id SERIAL PRIMARY KEY,

incident\_id INTEGER REFERENCES Incidents(id),

volunteer\_id INTEGER REFERENCES Volunteers(id),

status VARCHAR(20) DEFAULT 'assigned' CHECK (status IN ('assigned', 'accepted', 'completed', 'declined')),

assigned\_by INTEGER REFERENCES Users(id),

assigned\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

API Endpoints

Authentication

POST /api/auth/register - Register a new user

POST /api/auth/login - Login and get JWT

Incidents

POST /api/incidents - Create a new incident

GET /api/incidents - Get list of incidents (with filtering)

GET /api/incidents/:id - Get incident by ID

PUT /api/incidents/:id - Update incident (coordinators only)

DELETE /api/incidents/:id - Delete incident (coordinators only)

Volunteers

POST /api/volunteers - Register as a volunteer

GET /api/volunteers - Get list of volunteers (coordinators only)

GET /api/volunteers/:id - Get volunteer by ID

PUT /api/volunteers/:id - Update volunteer profile

DELETE /api/volunteers/:id - Deactivate volunteer (coordinators only)

Assignments

POST /api/assignments - Assign a volunteer to an incident (coordinators only)

GET /api/assignments - Get assignments (coordinators see all, volunteers see own)

GET /api/assignments/:id - Get assignment by ID

PUT /api/assignments/:id - Update assignment status

DELETE /api/assignments/:id - Remove assignment (coordinators only)

Dashboard

GET /api/dashboard/stats - Get dashboard statistics (coordinators)

Frontend Pages

Login/Register Page

Dashboard (role-specific)

Incident Report Form

Incident List (with map view)

Incident Detail Page

Volunteer Management (coordinators only)

Volunteer Profile (volunteers only)

Implementation Plan

Phase 1: Backend Setup

Initialize Node.js project with Express

Set up PostgreSQL database connection

Create models with Sequelize

Implement authentication (register, login, JWT)

Implement authorization middleware

Phase 2: Core API Endpoints

Incident management CRUD

Volunteer management CRUD

Assignment management CRUD

Dashboard statistics

Phase 3: Real-time Features

Implement Socket.io for real-time updates

Create WebSocket events for incident/assignment updates

Phase 4: Frontend Development

Set up React app with routing

Create authentication context

Build login/register forms

Develop dashboard components

Create incident reporting and management UI

Implement volunteer management interface

Add map visualization with Leaflet

Phase 5: Testing & Polish

Write unit and integration tests

Improve UI/UX

Add error handling and loading states

Optimize performance

Key Features to Highlight in Demo

Real-time coordination: Show how assignments update instantly across all devices

Role-based access: Demonstrate different views for each user type

Map visualization: Display incidents and volunteers on a live map

Comprehensive CRUD: Showcase all create, read, update, delete operations

Workflow demonstration: Show the complete incident lifecycle from report to resolution

This blueprint provides a comprehensive foundation for building your Emergency Response Coordination Hub demo. The project will effectively showcase your ability to implement a complete RESTful API with CRUD operations while solving a real-world problem.