Disaster Alert System (DAS)

Summary Document

Assignment 1: Collaborative Software Development

Group 17

January 2026

Contents

1 Project Summary

Project Demo Video: *Click here to view the project demonstration video*

Project Repository: *Click here to access the GitHub repository*

Note: Detailed technical information, system design, testing methodology, and evaluation results are available in the `reports` folder of the repository and in the respective project reports submitted as part of this assignment.

The objective of this project was to design and implement a **Disaster Alert System** capable of delivering real-time alerts to users located in disaster-affected regions. The system was developed as part of Assignment 1, focusing on the core aspects of **usability, reliability, and safety**. The solution aims to ensure timely dissemination of critical information during emergencies while maintaining system stability under high traffic conditions and minimizing false alerts.

1.1 System Overview

The Disaster Alert System was implemented as a web-based application with a modular architecture. The backend was developed using **Python (Flask)** to handle alert logic, user management, and notification workflows. The frontend was developed using **React** to provide a clean, user-friendly interface for alert creation and reception. To ensure portability and ease of deployment, both frontend and backend services were containerized using **Docker**.

A key feature of the system is an interactive **3D world visualization**, which displays disaster-affected regions using exact **latitude and longitude coordinates**. This visualization enables users and administrators to clearly understand the geographical impact of disasters.

The system supports multiple alert delivery mechanisms, including **SMS and email notifications**, ensuring that alerts reliably reach users even if one communication channel fails. Retry mechanisms and fallback strategies were incorporated to improve message delivery success during high-load scenarios.

1.2 Functional Implementation

The system implements the following major functional modules as defined in the requirements document:

- **User Interface and Authentication Module:** Allows users to register, authenticate securely, and manage their profile and location data. Role-based access control ensures that only authorized personnel can create or modify alerts.
- **Alert Management Module:** Enables authorized users to create disaster alerts with attributes such as disaster type, severity level, description, affected region, and optional geographic coordinates. The system validates inputs and prevents duplicate alerts for the same region.
- **Location and Region Processing Module:** Associates alerts and users using validated latitude and longitude values. The module ensures that alerts are delivered only to users within affected regions, which is visually reflected in the 3D world map.

- **Notification Delivery Module:** Sends alerts to affected users using **SMS and email notifications**. Retry mechanisms and multi-channel delivery improve reliability during transient failures.
- **Logging and Audit Module:** Records alert creation, modification, and delivery events with timestamps and user identifiers for traceability and auditing.

1.3 Non-Functional Requirements

The project was executed in a phased manner across four weeks. During the initial phase, requirements were gathered and documented, and the overall system architecture was planned. Development and testing were carried out iteratively, with continuous reviews to ensure alignment with requirements and quality standards. The final phase focused on system integration, documentation, and quality assurance.

- **Usability:** A clean and intuitive web interface, consistent API responses, and descriptive error handling ensure ease of use for both administrators and end users.
- **Reliability:** The system handles high traffic during disaster scenarios, supports retry mechanisms, and degrades gracefully when external services such as SMS gateways are unavailable.
- **Safety:** Strict validation of alert data, geographic boundaries, and severity levels minimizes false positives and ensures accurate alert dissemination.

1.4 Project Planning and Team Execution

The project was executed over four weeks using a phased development approach. In the initial phase, requirements were gathered and documented. Subsequent phases involved implementation, testing, integration, and documentation.

Each team member contributed according to their assigned role. The **Project Manager** coordinated timelines and deliverables. The **Requirements Engineer** prepared the Software Requirements Specification and ensured compliance during development. The **Developers** implemented backend logic, frontend interfaces, alert workflows, and notification services. The **Tester** validated system behavior through frontend and backend testing. The **Quality Assurance Engineer** ensured code quality and standards compliance through reviews. The **Integration Specialist** handled Docker-based deployment and module integration. The **Documentation Specialist** prepared user guides, test reports, and requirement documentation.

1.5 Table of Contributions

Table 1: Team Contributions

Roll	Name	Role	Timeline	Contribution
B23123	Davda James	Project Manager	Week 1-4	Planned the project timeline, assigned tasks, coordinated team activities, monitored progress, and ensured timely completion.
B23197	Avirup Ghosh	Developer	Week 2-3	Code for backend and fronted written in python (Flask) and react respectively, followed instructions given by requirements engineer added each logic along with alert sending interface (SMS).
B23158	Mehul Sharma	Requirements Eng.	Week 1&4	Preparation of Project Requirements as well as making sure that the code follows the correct testing and maintainence guidelines
B23354	Piyush Dwivedi	Tester	Week 3	Testing environment development for both frontend and backend tests.
B23130	Gyan Ratan	Developer	Week 2	Added mail alerts and retries to ensure that alerts reaches to end user reliably.
B23140	Kartik Verma	Quality Assurance	Week 3-4	Responsible for maintaining code quality and reliability through systematic code reviews. Reviewed and merged pull requests to ensure correctness, maintainability, and adherence to project standards.
B23174	Riya Chaudhary	Integration	Week 4	Being the integration specialist, integrated the project by creating dockerfiles for both frontend and backend
B23162	Molik Tyagi	Documentation Specialist	Week 4	As the documentation engineer, added user guide, test reports and requirements docs in the repository to make the overall project more readable.