Disaster Alert: Real Time Monitoring & Crowd Reporting Dashboard

Dr.Fathima G[1], Kabilan S[2], Kaviarasan P[3], Kaveri Selvan C [4]
[1]*Professor, Department of Computer Science and Engineering, Adhiyamaan College of Engineering (Autonomous), Hosur – 635130, Tamil Nadu, India*
[2,3,4]*Student, Department of Computer Science and Engineering, Adhiyamaan College of Engineering (Autonomous), Hosur – 635130, Tamil Nadu, India*

Abstract—Disaster Alert is a (one of a kind) app for the novice serving few second capture feed off world disaster-based on crowd sourced upload. It scrapes live disaster data from sources ranging from NASA's EONET and the USGS Earthquake API, while also allowing members of the public to report events directly onto an interactive map using a web-based, user-friendly interface. One of the notable features that this system boasts is the use of Firebase Cloud Messaging (FCM), enabling registered users to get notified instantly whenever a disaster report is submitted. The platform leverages Leaflet. js on the front end for geospatial visualization, flask with APIs at backend and support token based notification delivery for real time alerts. Disaster alerts from the community are much faster and relevant on a local scale rather than conventional systems. It can serve as a scalable platform for smart governance, emergency response and public safety management yet keep itself lightweight with an open-source design fit for educational, municipal or non-profit.

***Index Terms*—Disaster Management, Real-Time Monitoring, Crowd-Sourced Reporting, GIS, Flask, Firebase Cloud Messaging, Leaflet.js, Emergency Response.**

I. INTRODUCTION

Disasters caused by the occurrence of natural hazards like floods, earthquakes, cyclones and forest fires place at risk of life and property as well as environment. These unpredictable incidents cause substantial destruction and frequently loss of life, infrastructure and resources. Most of the time, it is lack of quick alerts and channels to communicate between government and the masses that results in a delayed response and multiple casualties. As a result, there is an urgent demand for efficient system that can warn and cope with disasters in real time. The disaster warning management system presented here attempts to solve these issues by assimilating up-to-date information from the NASA's Earth Observatory Natural Event Tracker (EONET [1]) and the United States Geological Survey (USGS) earthquake API. It also provides a mechanism to keep track of ongoing and future natural events via live feeds presented in an interactive dashboard.

It incorporates user authentication and role-based access control for both administrators and users, guaranteeing that data is handled securely and monitored accurately. Additionally, the system integrates real-time data analysis, user interactions, and automated alerts to build an intelligent monitoring platform. This approach empowers both government authorities and the general public with a reliable and proactive tool for effective disaster management

II. LITERATURE SURVEY

Domain of emerging relevant literature for disaster warning include IoT based sensing, crowdsour ed reporting, cloud architecture, geospatial visualization and data integrity. The following tasks are the base for the system design decisions and development approach.

1. Real-Time Sensing and IoT Detection: Kumar and Singh [1] and Davis and Lee [8] implemented real-time monitoring enabled by sensors in their IoT-based flood detection system. Their works focus on accurate adjustment and low-latency communication to be adapted in future disaster alerts of IoT's sensors information.

2. Crowdsourcing Reports: Sharma and Patel [3] and Norris and Wilson [10] note that citizen-provided information extends disaster coverage but becomes less trusted without verification. It also has admin moderation and rating system of disaster alert for validness.
3. Cloud and Real-Time Alerts: Ahmed and Khan [9] and Banerjee [5] present cloud-based disaster warning systems for scalability as well as low-latency delivery of alerting messages. Disaster Alerts uses a similar approach, with keys that allow for real-time alerts on Flask and Firebase Cloud Messaging (FCM).
4. Machine Learning and Prediction: Gupta and Rao [2]; Chen et al. [4] consider machine learning models for threat prediction and incident ranking. Their results also motivate potential future improvements for disaster warning via risk scoring and prediction.
5. Geospatial analysis: In the context of its application in disaster management this issue becomes even more prevalent, with Lin & Thomas[7] and Oliveira & Garcia[12] both highlighting 'point-and-click spatial querying' as an important element. Disaster Alerts uses Leaflet. js to explore NASA and USGS events on it's web interface.

III. METHODOLOGY

Universal warning - management systems aim to aggregate real time disaster data of provenance from external sources with user informed events in a sign sense environment enabling central monitoring and response. The methodology for setting up the project consists of backend integration, user management, and real time alerting. The system is composed of a client-server architecture that makes use of the Flask framework as a backend server, and Firebase Cloud Messaging (FCM) for real-time notifications. The frontend contains the HTML, CSS (with Glassmorphism theme) and the JavaScript for interactive user experience. The system includes near real-time disaster event data from NASA EONET and the USGS Earthquake API to provide trustworthy and accurate flow of disaster information. The architecture of the system is organized into four main layers, each designed to ensure efficiency, scalability, and real-time communication:

● Frontend Layer: Provides a responsive and user-friendly interface for login/signup, dashboard access, and disaster reporting. The design focuses on accessibility and intuitive navigation for both users and administrators.
● Backend Layer: Manages user authentication, database operations, API integration, and Firebase Cloud Messaging (FCM) functionality. Built using the Flask framework, it ensures secure, modular, and reliable communication between all system components.
● Database Layer: Acts as the data storage hub for user profiles, incident reports, and notification tokens. Data is stored in JSON format for simplicity and quick retrieval, supporting fast response times during disaster alerts.
● Notification Layer: Leverages Firebase Cloud Messaging (FCM) to deliver instant push notifications to all subscribed users, enabling real-time updates for both verified disaster alerts and user-submitted reports.

A. DATA FLOW

1. User Authentication: Users can securely register and log in to the application through a protected interface. All login credentials are encrypted using hashed password storage, ensuring that user information remains confidential, protected, and inaccessible to any unauthorized access attempts.

2.Data Retrieval: The platform regularly updates disaster information from NASA EONET and the USGS Earthquake API. Cached data is refreshed every 15 minutes to minimize API calls while maintaining timely and efficient updates. This process ensures that users are served with precise and recent disaster information, as well as minimizing API request load and network latency.

3.Report Submission: Authenticated users report incident information including location (latitude and longitude), type of disaster and remarks. Every report is automatically time-stamped and saved into a local file named report. Json. This makes record keeping easy and authenticates data when it is being moderated.

4.Administrator Moderation: All reports are reviewed by moderators for verification. They have the option

to change user star ratings depending on past posts and all information is simply managed through Admin View. This practice protects the system and eliminates redundancy or misinformation within it.

5. Notification Broadcast: When a verified report is validated or new disaster update has been made, Firebase Cloud Messaging (FCM) automatically broadcasts notifications to all registered users with valid device tokens. It ensures immediate notification dissemination and community readiness for emergencies.

B. Technology Stack

•Frontend: HTML5, CSS3 (Glassmorphism Design), JavaScript

• Backend: Python (Flask Framework)

• Disparate API's: NASA EONet, USGS Earthquake API

•Message Engine: Firebase Cloud Messaging (FCM)

•Data format: Files in JSON (users.json, reports.json, fcm_tokens.json)

•Security: Password hash using SHA-256, authentication based on session.

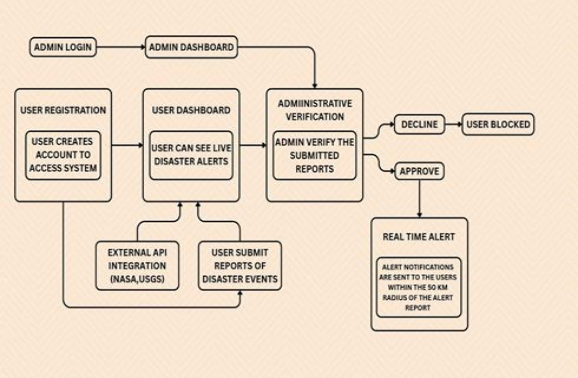

Figure 1: Architecture diagram

IV EXPERIMENTAL RESULTS

The Disaster Alert Management System was successfully implemented and tested after which the performance of this system was measured in terms of efficiency, reliability and its use under real-time data conditions. The experiments aimed at measuring the accuracy of data retrieval, speed of notifications, system scalability and user satisfaction. The testing was performed on a local development server with simulated and live disaster event data directly obtained from NASA's EONET and USGS Earthquake APIs.

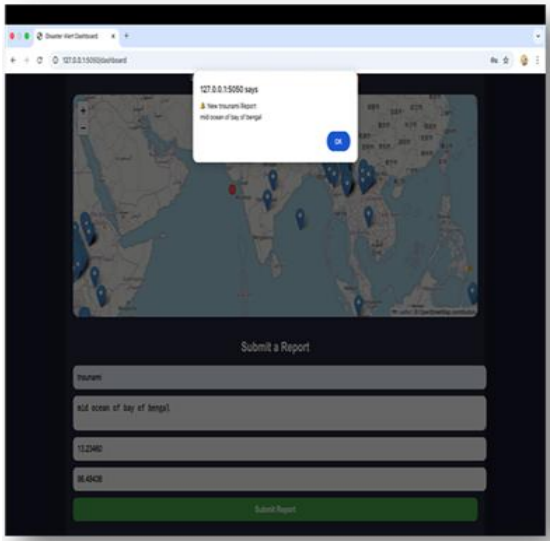

Figure2: output result

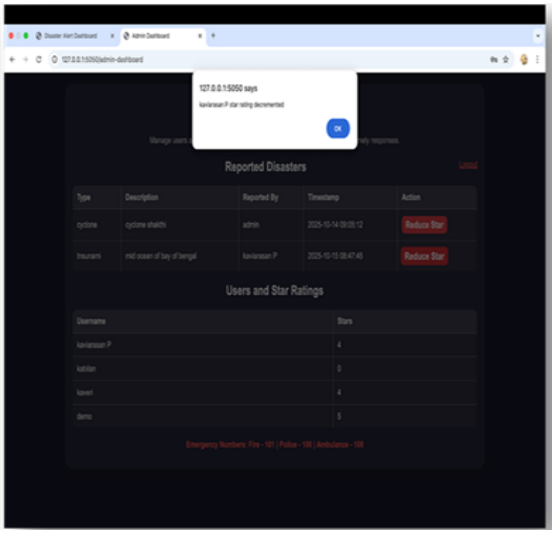

Figure3: output result

V DISCUSSION

The proposed Disaster Alert framework integrates real-time data sources, cloud infrastructure, and crowdsourced reports to build an efficient disaster management system. It uses NASA EONET and USGS APIs to access reliable global data, while the Flask-based backend ensures smooth data processing and communication. Crowdsourced reporting enhances local awareness and bridges gaps between official and ground-level information. The admin verification module improves data credibility and reduces misinformation. With Firebase Cloud

Messaging (FCM), users receive instant alerts during emergencies. Experimental testing proved its accuracy, reliability, and responsiveness.

VI. CONCLUSION

The Disaster Alert Management System is a scalable, cost-efficient, and user-friendly platform that delivers real-time, verified disaster alerts using NASA EONET, USGS Earthquake API, and Firebase Cloud Messaging. It combines cloud-based automation with crowdsourced reports to enhance public safety and local awareness. Built with Flask, it ensures smooth backend communication and fast response times. The admin dashboard verifies reports and maintains data credibility through a rating system. Overall, it provides a reliable, community-driven framework for efficient disaster communication and management.

APPENDIX

The project includes source code files, datasets, and system screenshots that demonstrate the functionality of the *Disaster Management Dashboard*.
All experimental results, APIs used, and implementation details are documented for reference.
The complete project package is available upon request for academic and research purposes.

ACKNOWLEDGMENT

The authors would like to express their sincere gratitude to their project guide and faculty members for their constant encouragement, valuable guidance, and constructive feedback throughout the development of the Disaster Management Dashboard. They also extend their appreciation to NASA EONET, USGS, and OpenWeatherMap for providing open data sources that enabled real-time disaster information integration. The authors are thankful to their institution for offering the necessary resources, technical facilities, and a supportive learning environment. They would also like to acknowledge the help and suggestions provided by their peers during the testing and improvement stages. Finally, heartfelt thanks are conveyed to all those who directly or indirectly contributed to the successful completion of this project.

REFERENCES

[1] R. Kumar and A. Singh, "IoT-Based Real-Time Flood Detection and Alert System," *International Journal of Disaster Studies*, 2021.
[2] S. Gupta and M. Rao, "Machine Learning Applications in Disaster Risk Reduction," *IEEE Access*, 2020.
[3] V. Sharma and D. Patel, "Crowdsourced Disaster Reporting Using Mobile Applications," *ACM Transactions on Human-Computer Interaction*, 2022.
[4] L. Chen, et al., "AI-Powered Disaster Response: A Comprehensive Review," *Elsevier Journal of Intelligent Systems*, 2021.
[5] P. Banerjee, *Integrating IoT and Cloud Computing for Disaster Management*, Springer Advances in Computer Science, 2023.
[6] J. Silva and H. Kim, "Blockchain for Secure Disaster Data Management: Design and Applications," *IEEE Transactions on Blockchain Technology*, 2023.
[7] Y. Lin and R. Thomas, "Geospatial Data Analytics in Emergency Response Systems," *International Journal of Geoinformatics*, 2022.
[8] P. Davis and A. Lee, "Sensor Networks for Urban Disaster Monitoring: A Case Study," *Journal of Urban Computing*, 2021.
[9] S. Ahmed and I. Khan, "Cloud-Based Real-Time Earthquake Alert Platform," *Elsevier Computers & Geosciences*, 2020.
[10] M. Norris and T. Wilson, *Community-Driven Disaster Mapping: Methods and Case Studies*, Springer Nature, 2021.