iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.81071

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue IV Apr 2026- Available at www.ijraset.com

Smart Disaster Monitoring and Early Warning System using Real-time Weather Data

Samiksha Desai[1], Harshada Deshmukh[2], Shruti Dwivedi[3], Tejaswini Mane[4], Prof. Mitrakshi Patil[5]
Dept. of Information Technology MGMCET Kamothe, India

Abstract: ***Natural disasters such as floods and heatwaves can cause significant threats to human life, property and environmental equilibrium. Early detection and timely communication can help us to minimize damage and improve response during such critical situations. Traditional disaster management approaches depend on information which can be delayed or inconsistent and may include manual intervention which in result may reduce their effectiveness during critical situations. The proposed system analyzes parameters such as temperature and rainfall to identify potential disaster scenarios and automatically generate alerts. When risk levels exceed predefined thresholds, notification services are triggered to send instant email and message alerts to registered users, enabling faster and earlier preparedness and response. The implementation demonstrates how real-time monitoring along with automated alerts can significantly improve disaster response and help citizens take necessary preventive measures.***

Keywords: Disaster Monitoring System, Real-Time Data, Flood Prediction, Heatwave Detection, Early Warning System, GIS, Multi-Hazard Management, Automated Alerts, React.js, Node.js, MongoDB, Environmental Monitoring, Risk Assessment, Disaster Preparedness

I. INTRODUCTION

Natural disasters such as floods, heatwaves, cyclones, and heavy rainfall are increasing in frequency and intensity due to climate change, urbanization, and environmental degradation. These events pose severe threats to human life, property, infrastructure, agriculture, and the overall environment. Urban areas are particularly vulnerable because of high population density, inadequate drainage systems, and concentrated human activities, which can amplify the impact of extreme weather events [5],[6],[10]. Traditional disaster management systems often rely on manual data collection, historical records, and delayed reporting. This results in slow response times and ineffective communication of warnings, increasing human vulnerability and loss. Moreover, the lack of integration between different sources of data—such as meteorological, hydrological, and environmental datasets—limits the accuracy of risk assessments and predictive insights [1],[2],[11]. As a consequence, emergency responses are often reactive rather than proactive, reducing the overall effectiveness of disaster mitigation efforts and community preparedness[7],[8].

To address these issues, we developed the Disaster Monitoring System, a smart web-based platform for disaster management and alerting. The system leverages modern technologies to provide real-time monitoring and automated communication. It integrates live data collection, predictive analysis, and multi-channel alert delivery to improve emergency preparedness and response. Using the OpenWeatherMap API, the platform collects up-to-date weather information such as temperature, humidity, rainfall, wind speed, and location-specific conditions. When a potential threat is identified, users are promptly notified through multiple channels: emails sent via Nodemailer, desktop notifications for instant alerts, and voice-based notifications using text-to-speech. This approach ensures accessibility for visually impaired users and for situations where users cannot immediately check notifications, enabling timely action in emergencies [3], [4],[12].

This platform provides an interactive and user-friendly dashboard that allows users to monitor and respond effectively to disaster situations. The dashboard provides real-time information about temperature, humidity, rainfall and other vital weather parameters. It also integrates maps to display disaster prone regions which helps the users to take immediate actions. This platform also offers users with a set of precautions that can be taken in case of different types of disasters to reduce their potential risk. This system also provides immediate access to local authorities and helpline numbers for quick support during critical disaster situations. Multi-language support makes the platform accessible to a large number of audiences which eradicates the problem of language barrier[5],[6]. By combining real-time data, automated alerts, multi-language support, and visual analytics, this system significantly reduces human dependency, minimizes response time, and enhances situational awareness, empowering users to make timely and informed decisions [8], [13].

During emergency situations, delayed or inconsistent information can give rise to confusion and may make the situation even more worse. This system overcomes this issue by presenting important weather and risk data in a clear and concise manner to the users. The dashboard displays real-time data using various visual elements like charts, icons, color-coded alerts and interactive maps, which help users quickly identify the potential risks. Alerts and notifications are also delivered using speech synthesis, provide audible warnings for impaired users. Desktop notifications ensure that alerts are received even after the tab is minimized improving system's effectiveness. The platform's design enables it to manage large amounts of real-time data and multiple users simultaneously, making it suitable for both urban and rural environments. By integrating accessibility, automated alerts, and clear visual representation, system provides a reliable and effective solution for early disaster warning and proactive risk management [7],[8],[12].

II. METHODOLOGY

The system has been developed using systematic and organized methods to ensure the system works efficiently and provides information about potential disaster risks. The system uses technologies like real-time alert systems, weather APIs , various data visualization techniques to monitor real-time data and alert users beforehand [8],[12].

A. Requirement Analysis:

In first step, we identified various problems faced by the users like late warnings, lack of information, lack of access to consistent data. Important features include real-time data monitoring, real-time alerts, notifications through emails, early risk prediction and better visualization.

B. System Design:

The system is designed in a structured way with different parts working together. The frontend dashboard is built using React.js to show weather data and alerts. The backend is developed using Node.js and Express to handle system logic and communication. MongoDB is a database that store all information in system .

C. Data Collection and Processing:

The system collects data from live weather information from the OpenWeatherMap API. It monitors and measures real-time data like temperature, humidity and rain. It help in early disaster detection and public safety by identifying danger like floods and heatwave before they occur [3],[13].

D. Risk Prediction and Alert Mechanism:

The system collects data for live weather and checks the risk level. If risk level is high, the system saves the information in the database and automatically alerts are sent via email.

E. Database Management:

The system collects weather and risk data and stores it in MongoDB. The database keeps all information like temperature, humidity, risk level, date and location. This helps view past records, analysis and shows information on charts and dashboards.

F. Data Visualization and Reporting:

The system features user-friendly dashboard that detects live weather data and updates in an easy-to -understand format. It also displays graphs of temperature and humidity using Chart.js includes maps to monitor different location. During disaster it also provide some safety tips, awareness information about the situation users know exactly what action to take. [5],[10].

G. Testing and Validation:

After building the system, testing is done to make sure everything works correctly. Each part of the system is checked, including weather API responses, email alert notifications, and dashboard visuals. The system is tested on different devices and internet conditions to make sure it works smoothly and reliably.

H. Deployment and Maintenance:

In this step, the system is finally deployed on a local server to make it accessible to users. After deployment, the system is monitored regularly for bugs. New features and improvements will be added over time to maintain good performance.

III. RELATED WORK

Author & Year	Focus Area	Key Contribution	Research Gap Identified
Liu et al. (2024)	Remote sensing flood detection	Integrates remote sensing and crowd-sourced data for fine-grained urban flood detection	Limited real-time implementation and dependency on data availability
Li et al. (2026)	SAR-based flood mapping	Uses density-aware neural networks with uncertainty quantification	Region-specific model, limited generalization
Pavithran et al. (2025)	Weather forecasting	Applies LSTM and transformer models for multi-scale weather prediction	Lacks real-time deployment and scalability
Das et al. (2025)	Flood mapping using SAR	Improves flood detection accuracy using Dual-Attention ResUNet	High computational cost and complex implementation
Wahba et al. (2025)	Hazard mapping	Uses ML techniques for improved flood and landslide risk mapping	Requires high-quality datasets and complex tuning

TABLE I – Literature Survey

Recent research in disaster management and environmental monitoring has increasingly focused on the integration of artificial intelligence, remote sensing, and machine learning techniques to improve prediction accuracy and response efficiency. Modern approaches emphasize the use of data-driven models and satellite-based systems to enhance flood detection, weather forecasting, and hazard assessment. These advancements have demonstrated significant improvements over traditional methods, particularly in identifying flood-prone regions and predicting environmental risks.

Several studies have explored the use of deep learning models such as LSTM, transformers, and convolutional neural networks for forecasting and flood mapping. These models provide high accuracy in prediction and pattern recognition by leveraging historical and real-time data. Additionally, remote sensing and SAR image-based techniques have been widely adopted to improve flood detection capabilities. However, many of these approaches are computationally intensive and require specialized datasets, limiting their applicability in real-time systems.

Existing research also highlights the use of multi-source data integration, combining satellite imagery, environmental data, and hydraulic modelling to enhance risk assessment. While these approaches improve prediction accuracy, they introduce complexity in data processing and system integration. Moreover, decentralized systems have been proposed to enable real-time monitoring, but they often lack centralized control and user-friendly interfaces, making them difficult to deploy in practical scenarios.

Another important limitation identified in the literature is the lack of real-time communication and synchronization. Many disaster management systems rely on static or batch-processed data, leading to delays in alerts and response actions. This reduces the effectiveness of early warning systems and increases the risk of damage during disasters. Additionally, several models focus on specific regions or datasets, resulting in limited generalization and scalability.

Security and accessibility are also critical challenges in existing systems. Many platforms do not implement proper authentication or role-based access control, leading to potential data inconsistencies and reduced reliability. Furthermore, user experience is often overlooked, with limited emphasis on interactive dashboards, real-time notifications, and ease of use.

Based on the reviewed literature, it is evident that current disaster management systems face challenges such as high computational requirements, lack of real-time capabilities, limited scalability, and insufficient user-centric design. To address these issues, the proposed Disaster Management System introduces a centralized, web-based platform that integrates real-time data processing, rule-based prediction mechanisms, and interactive dashboards. The system focuses on providing cost-effective, scalable, and user-friendly solutions for monitoring disasters such as floods and heatwaves.

Overall, the proposed system bridges the gap between complex AI-based research models and practical implementation by offering a real-time, efficient, and accessible disaster management solution suitable for real-world applications.

.

IV. SYSTEM DESIGN AND COMPONENT

The Disaster Monitoring System is designed using a modular web-based architecture that integrates frontend, backend, database, and external APIs to deliver real-time monitoring, alert generation, and user support services. The architecture is structured into multiple functional modules, each responsible for specific tasks, ensuring scalability, maintainability, and efficient data processing.

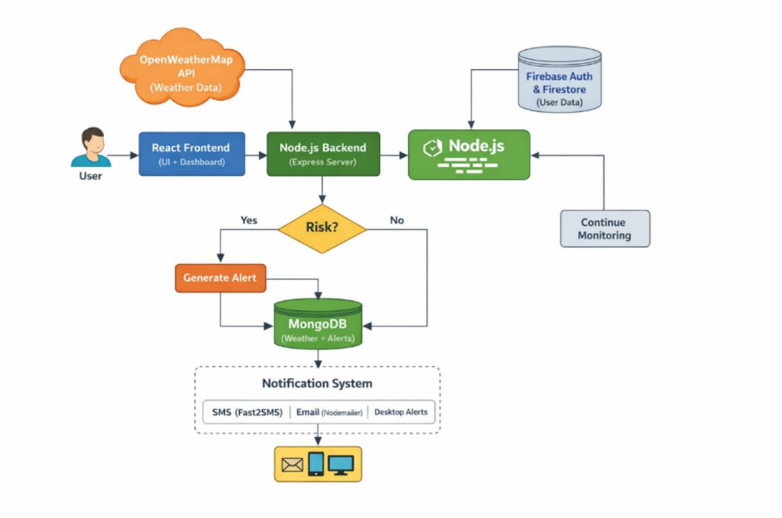


Fig 1 – System Architecture

A. Frontend Layer

The frontend is developed using React.js, providing a dynamic and responsive interface for user interaction. It consists of several modules including authentication service, dashboard module, precaution module, helpline module. Authentication Service manages user sign-up, sign-in, and access control. Dashboard Module displays real-time environmental data, risk indicators, and historical trends through charts and maps. Precaution Module provides guidelines and preventive measures based on detected risk levels. Helpline Module offers users quick access to emergency contacts and local authorities.

B. Backend Layer

The backend is implemented using Node.js with Express.js, acting as the core processing unit of the system. It handles API requests from the frontend, processes real-time weather data, generates alerts, and interacts with the database.

C. Database Layer

MongoDB is used as the primary database to store historical weather data, alerts, user information, and precautionary guidelines. The database supports efficient retrieval and storage, ensuring the system can respond quickly to user requests and generate timely alerts.

D. External API Integration

An external weather API provides live environmental data, including temperature, humidity, rainfall, and wind speed. The backend fetches this data periodically, preprocesses it, and evaluates it against predefined thresholds to determine the risk levels.

E. Server Layer

A Node.js server hosts the backend services and facilitates communication between the frontend, database, and external APIs. This ensures seamless data flow and real-time updates on the user dashboard.

F. Data Flow Overview

Data collected from the external weather API is processed by the backend, analyzed by the rule-based engine, and stored in MongoDB. Alerts and precautionary messages are then transmitted to the frontend modules via the server. Users receive real-time notifications, visual risk indicators, and access to emergency resources through the dashboard interface.

V. IMPLEMENTATION

The Disaster Monitoring System is developed as a web-based platform that enables continuous monitoring of environmental conditions, generates timely alerts, and allows user interaction. This implementation translates the proposed methodology into a practical system using modern web frameworks and APIs.

A. Technology Stack

The frontend is built using React.js to provide a responsive and interactive interface. Data visualization is achieved with Chart.js, while Leaflet.js is used for displaying disaster-prone regions on maps. The backend uses Node.js with Express.js to handle API requests, process incoming data, and manage alert notifications efficiently. MongoDB is employed to store both real-time and historical environmental data, user details, and alert logs. Live weather information is obtained through the OpenWeatherMap API, capturing parameters such as temperature, rainfall, humidity, and wind speed. Notification services include email alerts via Nodemailer, browser-based notifications using the Web Notification API, and audible alerts through a text-to-speech module.

B. Data Flow and Processing

The system regularly collects live weather data from the OpenWeatherMap API . A rule-based engine evaluates these parameters against established thresholds to determine the risk level of floods and heatwaves.

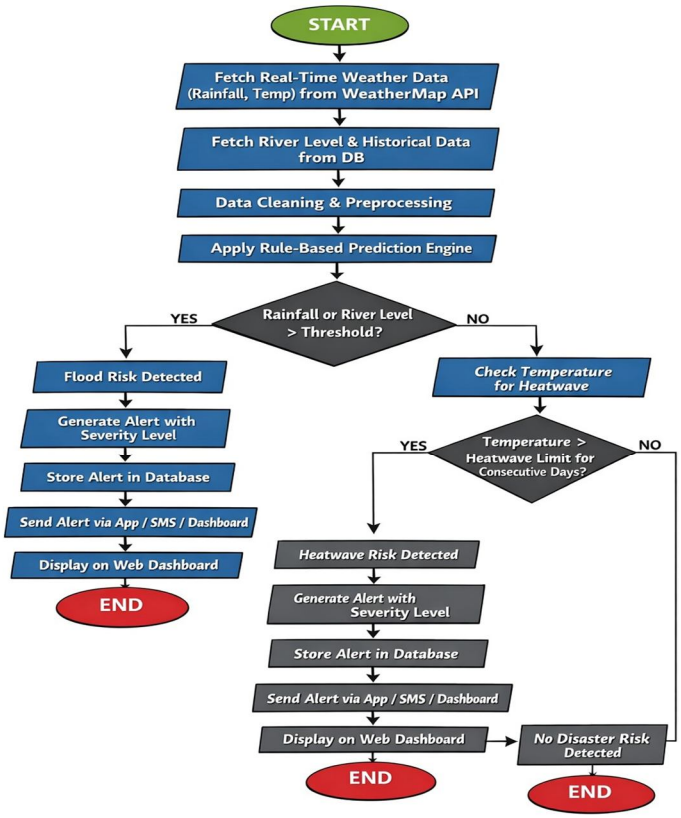


Fig 2 : Data Processing

C. Alert Mechanism

Alerts are automatically triggered when measured environmental conditions exceed predefined safety limits. Each alert is recorded in the database with details including disaster type, severity, location, timestamp, and recommended precautions. The system delivers notifications through multiple channels: emails, real-time dashboard alerts with color-coded severity indicators, and audible warnings, ensuring that users are promptly informed.

D. Dashboard Functionality-

The dashboard provides a responsive interface that updates in real-time with incoming data. Interactive charts and maps allow users to visualize trends and identify at-risk areas. Risk levels are indicated using a color-coded system: green for low, yellow for moderate, and red for high risk. Users can access historical data, safety instructions, and contact information for local emergency services, enabling informed decision-making during disaster events.

E. Deployment

The backend and database are deployed on local or cloud-based servers to support continuous access and multiple concurrent users. The frontend is hosted either on the same server or separately using React build tools. The system has been tested across multiple devices to ensure responsiveness, reliability, and real-time performance, making it practical for real-world disaster monitoring applications.

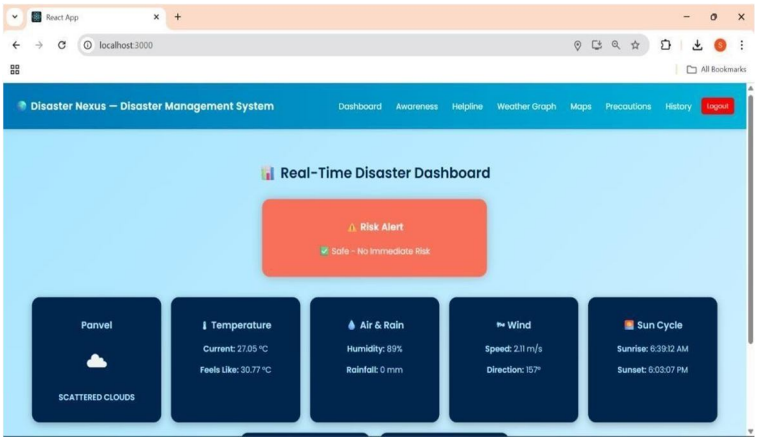


Fig 3 : Dashboard Page

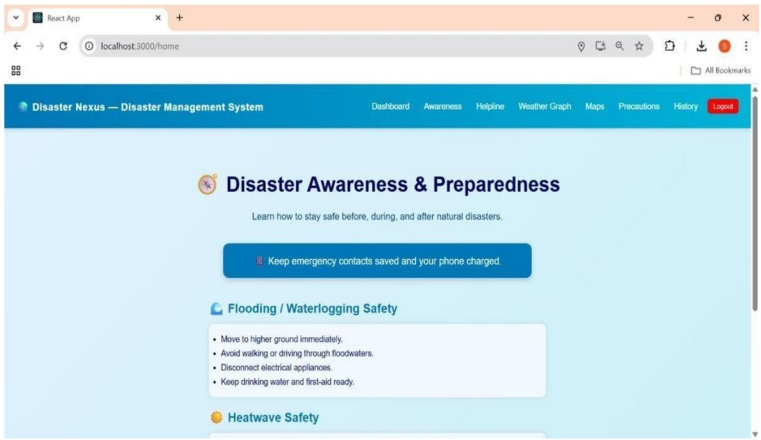


Fig 4 : Awareness page

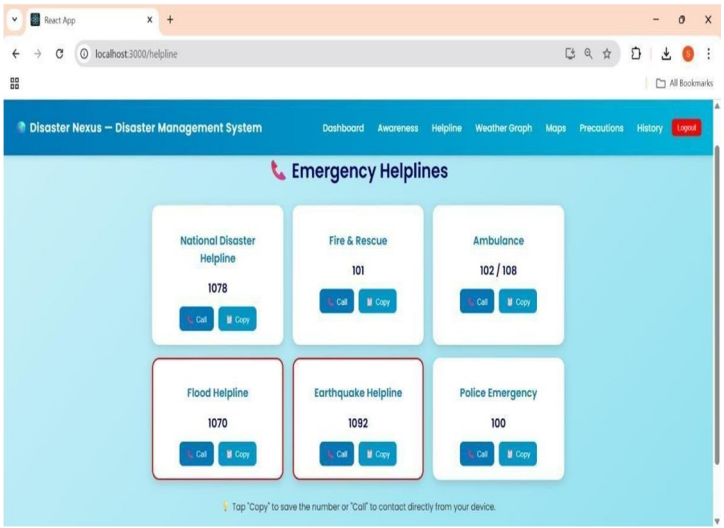


Fig 5 : Helpline Page

Fig 6 : History page

VI. EXPERIMENTAL SETUP

The experimental evaluation of the proposed Disaster Nexus system was conducted to analyze system performance, real-time responsiveness, alert accuracy, and reliability under different disaster scenarios. The evaluation framework integrates real-time data acquisition, rule-based prediction logic, and controlled environmental simulations. The system was tested across multiple disaster conditions such as floods and heatwaves to ensure accuracy, scalability, and consistency of generated alerts and system outputs.

A. Hardware and Software Environment

The system was implemented in a controlled development environment configured as follows:

- Processor: Intel Core i5 (or equivalent multi-core CPU)
- Memory: 8 GB RAM
- Operating System: Windows/Linux
- Programming Language: JavaScript and Python
- Frontend Framework: React.js
- Backend Framework: Node.js with Express
- Database Management: MongoDB
- Authentication: Firebase Authentication
- API Integration: OpenWeatherMap API for real-time weather data
- Notification Services: Nodemailer (Email), Web Notification API, Speech Synthesis

The selected configuration demonstrates that the system can be deployed efficiently in moderate infrastructure environments without requiring high computational resources. The system maintains stable performance under continuous environmental data updates and concurrent user interactions. Derived environmental metrics such as temperature trends, humidity levels, rainfall intensity, and risk levels are computed from incoming API data, ensuring reliable system behavior and accurate disaster prediction.

B. Dataset Description and Preprocessing

The Disaster Nexus system operates on real-time environmental data collected through external APIs and internally generated system data. The dataset includes attributes such as temperature, humidity, rainfall, wind speed, alert status, and historical logs. These records are dynamically updated and stored in the database, enabling continuous monitoring and analysis.

Before processing, the data undergoes preprocessing to ensure consistency and accuracy:

- Removal of incomplete or missing environmental data
- Validation of API responses and parameter ranges
- Categorization of environmental conditions (safe, warning, high risk)
- Aggregation of data into time-based summaries
- Standardization of data formats for uniform processing

Derived system metrics such as risk level, alert frequency, and environmental trends are calculated from the processed dataset.

System response time (Rt) was calculated as:

$$Rt = T_{response} - T_{request}$$

where Trequest represents the time at which data is fetched or user action is performed, and Tresponse represents the time when the alert or dashboard update is displayed. The average response time was observed between 1–3 seconds, demonstrating efficient near real-time performance.

C. Evaluation Strategy

The evaluation strategy focuses on validating the correctness of disaster prediction and the effectiveness of rule-based decision-making. The system was tested using multiple simulated environmental scenarios, including heavy rainfall, high humidity, and extreme temperature conditions.

Key operational logic used for evaluation includes:

Flood Risk Condition (FR):

If Rainfall > Threshold_1 AND Humidity > Threshold_2 → Flood Alert

Heatwave Condition (HR):

If Temperature > Threshold_3 → Heatwave Alert

Risk Classification:

Safe → Warning → High Risk

These rules ensure logical consistency and timely detection of disaster conditions. The system outputs were evaluated based on accuracy of alerts, consistency of predictions, and real-time responsiveness of notifications and dashboard updates.

D. Performance Evaluation

System validation was performed under continuous environmental data updates and real-time monitoring conditions. The system was evaluated for response time, alert generation accuracy, update synchronization, and processing efficiency, confirming its scalability and robustness.

The results indicate that the Disaster Nexus system provides efficient real-time monitoring, accurate disaster prediction, and reliable alert mechanisms. Notifications were successfully delivered through multiple channels, including email, browser alerts, and voice output, ensuring accessibility and rapid response.

Real-time data synchronization ensures that all connected users receive consistent updates without noticeable delay. The system successfully handled multiple concurrent users and continuous API requests without performance degradation, demonstrating its suitability for practical deployment in real-world disaster management scenarios.

VII. RESULT AND ANALYSIS

The results of the experimental evaluation demonstrate that the proposed Disaster Nexus system effectively performs real-time disaster monitoring and alert generation with high reliability and efficiency. The system was tested under multiple simulated environmental conditions, including heavy rainfall, high humidity, and extreme temperature scenarios, to evaluate its prediction accuracy and responsiveness.

The system successfully classified environmental conditions into safe, warning, and high-risk categories based on predefined rule-based thresholds. During flood simulation scenarios, the system accurately generates alerts when rainfall and humidity values exceed the defined limits. Similarly, in heatwave conditions, alerts were triggered when temperature values crossed critical thresholds. The alert generation mechanism showed consistent and correct behavior across all test cases, indicating high logical accuracy of the rule-based model.

In terms of performance, the system achieved an average response time between 1–3 seconds for fetching data, processing conditions, and updating the dashboard. This demonstrates the system's capability to operate in near real-time environments. Continuous API data fetching and processing did not introduce significant delays, and the system maintained stable performance even under repeated and concurrent requests.

The real-time dashboard provides effective visualization of environmental parameters using dynamic charts, enabling users to easily monitor trends in temperature, humidity, and rainfall. The integration of multiple notification channels, including email alerts, browser notifications, and voice-based alerts, ensured that users received timely warnings without delay. This multi-channel communication significantly improves the usability and effectiveness of the system in critical situations. From a scalability perspective, the system handles multiple users and continuous data updates without performance degradation. The backend efficiently processes incoming data and synchronizes updates across all connected clients, ensuring consistency of information. The use of a centralized architecture contributed to better control and coordination compared to decentralized or fragmented approaches. However, the analysis also highlights certain limitations. The system relies on external APIs for environmental data, which may introduce dependency and potential delays in case of network issues. Additionally, the rule-based prediction mechanism, while efficient and lightweight, may not capture highly complex patterns compared to advanced machine learning models. Despite these limitations, the system provides a practical and cost-effective solution suitable for real-time disaster monitoring.

Overall, the Disaster Nexus system demonstrates strong performance in terms of accuracy, responsiveness, usability, and scalability. The results validate that the proposed approach successfully bridges the gap between complex research-based models and practical real-world implementation by offering a reliable, real-time disaster management solution.

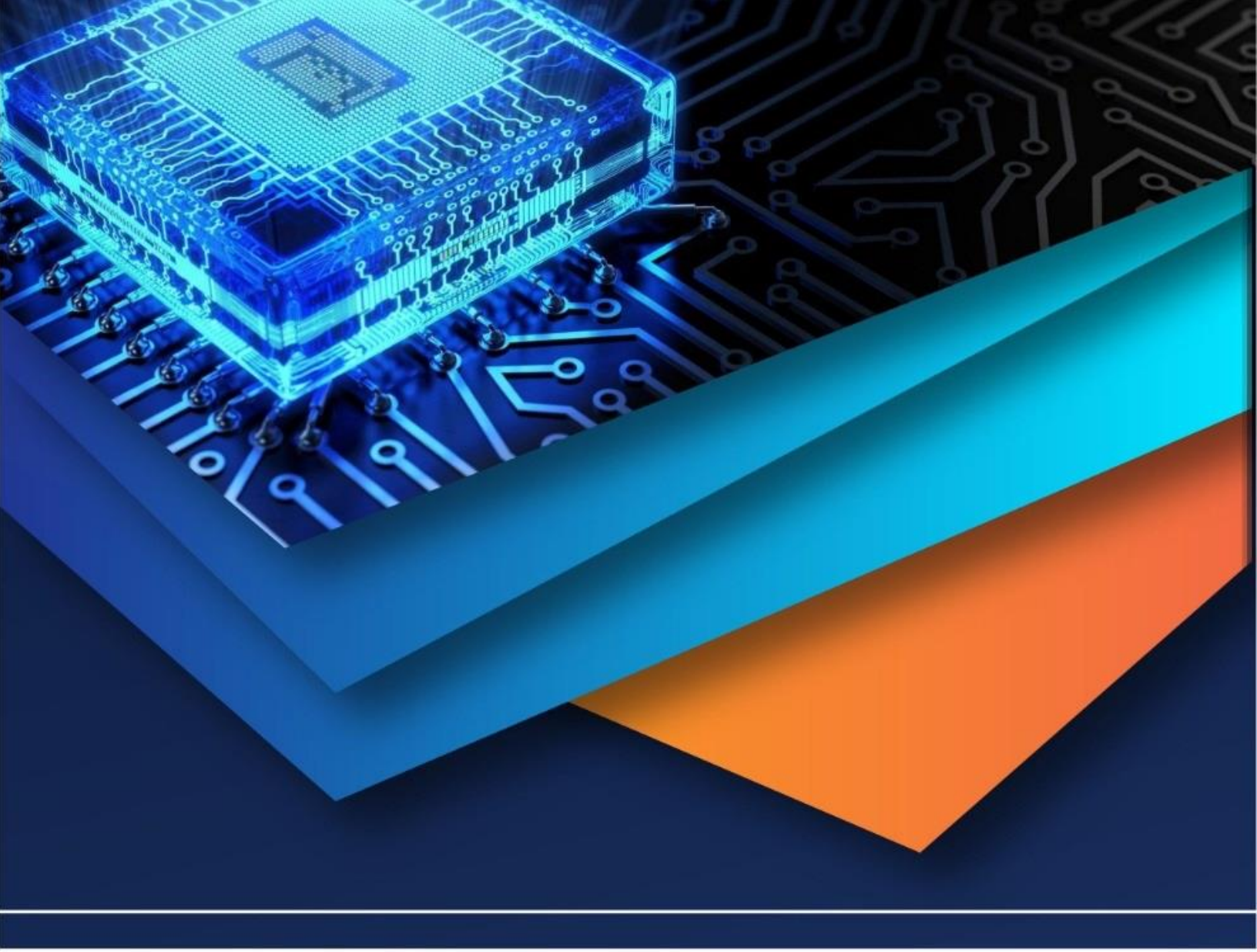
VIII. CONCLUSION AND FUTURE WORK

The Disaster Nexus system provides an effective solution for real-time monitoring of environmental conditions and detection of disaster risks such as floods and heatwaves. By using data from the OpenWeatherMap API, simple threshold-based analysis, and a centralized MongoDB database, the system is able to quickly identify dangerous situations. Alerts are sent through multiple channels including email, dashboard notifications, and desktop alerts, ensuring that users remain informed even if they are not actively using the system. The React.js-based interface offers clear visualization of weather data and risk levels, helping users make better decisions and improving overall public safety. The system also supports continuous monitoring, which helps in early warning and reduces the impact of disasters by giving users enough time to take precautions. The simple and responsive dashboard makes the system easy to use even for non-technical users

The Disaster Nexus system can be further enhanced to improve accuracy and usability. In the future, advanced AI models such as CNN-LSTM could be integrated to improve disaster prediction for floods and heatwaves. The system can also include real-time IoT sensor data like river water levels, soil moisture, and local weather station data for more accurate and location-specific alerts. A mobile application can be developed to support push notifications and SMS alerts, especially useful in areas with poor internet connectivity. Additional features such as interactive maps, heatmaps, and detailed trend analysis can improve visualization and understanding. Personalized alerts based on user location and preferences can increase effectiveness, while integration with government disaster management systems and cloud deployment can improve scalability and coordination. Adding multi-language support and accessibility features will help make the system usable for a wider population.

REFERENCES

[1] D. Liu, J. Li, L. Wang, and A. Plaza, “Integration of Remote Sensing and Crowdsourced Data for Fine-Grained Urban Flood Detection,” IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens., vol. 17, pp. 13523–13532, 2024.
[2] Y. Li, P. Matgen and M. Chini, "Quantifying and Communicating Uncertainty in SAR-Based Flood Mapping via Density-Aware Neural Networks and Conformal Risk Control," in IEEE Transactions on Geoscience and Remote Sensing, vol. 64, pp. 1-20, 2026.
[3] M. S. Pavithran, B. Sreeram, A. V. Pillai and R. Jothi, "Multi-Scale Weather Forecasting Using Deep Learning Architectures With Chennai Climate Data," in IEEE Access, vol. 13, pp. 207303-207319, 2025.
[4] A. Das, S. M. A. Rajin, G. Kah Ong Michael, S. Biswas, N. Billah and R. Khan, "Dual-Attention ResUNet With Masked Focal-Tversky Loss for Robust SAR-Based Flood Mapping," in IEEE Access, vol. 13, pp. 201460-201477, 2025.
[5] M. Wahba et al., "Enhanced Hazard Mapping for Flood and Landslide Risks Using Memetic Programming and Machine Learning Techniques to Support Sustainable Development Goals," in IEEE Access, vol. 13, pp. 182410-182429, 2025.
[6] Q. Yan, J. Yuan, D. Wu and Y. Lin, "Multisource Flood Risk Assessment for Shenyang Townships Using Dynamic Integration of Analytic Network Process and Autoencoder Weights," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 18, pp. 26894-26907, 2025.
[7] E. Filho, M. Saideh, L. Vercouter, J. Silveira, C. Marcon and W. d. Santos, "DADOS: Decentralized Autonomous Disaster Observation System," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 18, pp. 25054-25071, 2025.
[8] A. Raj, A. Shetgaonkar, L. Arora, D. Pradhan, S. S. Girija and S. Kapoor, "AI and Generative AI Transforming Disaster Management: A Survey of Damage Assessment and Response Techniques," 2025 IEEE 49th Annual Computers, Software, and Applications Conference (COMPSAC), Toronto, ON, Canada, 2025, pp. 1834-1840, doi: 10.1109/COMPSAC65507.2025.
[9] N. Fatima et al., "Integrating Machine Learning Models With Probability Distribution Methods for Extreme Flood Risk Assessment," in IEEE Access, vol. 13, pp. 160922-160938, 2025.
[10] Q. Wang et al., "Enhancing Flood Risk Assessment Using Multisensor Remote Sensing Data and Hydraulic Modeling," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 18, pp. 19393-19406, 2025, doi: 10.1109/JSTARS.2025.
[11] M. Raqibul Hasan, M. J. Hossain, M. Waliullah, A. Hannan and M. M. Rahman, "Topological Data Analysis and Wavelet- Unsupervised Machine Learning Approaches to Identifying the Flooding and Non-Flooding Zones," in IEEE Access, vol. 13, pp. 111710-111721, 2025.
[12] J. Teoh, Z. B. Zulkoffli, K. M. Yap and H. S. Chua, "Exploring Generative AI for YOLO-Based Object Detection to Enhance Flood Disaster Response in Malaysia," in IEEE Access, vol. 12, pp. 173686-173699, 2024, doi: 10.1109/ACCESS.2024.
[13] C. Zhang, H. Hou, A. K. Sangaiah, D. Li and W. Li, "Enhancing High-Temperature Prediction via Sixfold Strategy Consensus-Reaching Processes: A Case Study Using FY-3E Spatiotemporal Remote Sensing Satellite Data," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 17, pp. 16377-16391, 2024, doi: 10.1109/JSTARS.2024.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY