ResQNet: An Integrated Platform for Real-Time Disaster Monitoring and Response

Juhi Goyal[1], Khushi Dangi[2], Khushi Garg[3], Kratika Arya[4], and Kavita Namdev[5]
[1,2,3,4]*Member, Acropolis Institute of Technology and Research*
[5]*Mentor, Acropolis Institute of Technology and Research*

***Abstract*—Disaster management processes, particularly during large-scale emergencies such as floods, earthquakes, and wildfires, are often characterized by fragmented communication systems and a lack of centralized coordination platforms. This paper presents ResQNet, an advanced digital emergency response and disaster management system developed to address these critical challenges. The system is built using modern web technologies including Node.js, Express.js, React.js, and MongoDB, ensuring a scalable, responsive, and user-friendly platform for seamless interaction between civilians, rescue teams, and administrative authorities. At its core, the platform enables real-time disaster reporting, resource allocation, and rescue team coordination through structured data handling and automated workflows. The application processes user inputs, manages disaster-related data, and dynamically updates system records to provide accurate and timely information. Additionally, integrated notification services ensure instant communication through automated email alerts. By combining centralized data management with intuitive design, ResQNet significantly improves emergency response efficiency compared to traditional methods, offering faster coordination, enhanced transparency, and reliable communication during crisis situations. The project demonstrates a practical and scalable approach to disaster management through digital transformation and real-time system integration.**

***Index Terms*—Disaster Management System, Emergency Response, MongoDB, Node.js, React.js, Real-Time Reporting, Rescue Coordination, Resource Allocation, Web Application**

I. INTRODUCTION

The management of disasters and emergency situations is a critical challenge that significantly impacts human lives, infrastructure, and overall societal stability. In a country like India, which is highly prone to natural disasters such as floods, earthquakes, and cyclones, the effectiveness of response systems plays a crucial role in minimizing damage and saving lives. However, the current disaster response framework is often hindered by fragmented communication channels, lack of centralized information systems, and inefficient coordination among civilians, rescue teams, and administrative authorities [1]. Victims and volunteers frequently rely on scattered sources such as news updates, social media, or manual communication, which leads to delays, confusion, and misallocation of resources [2].

The rapid advancement of modern web technologies and real-time communication systems presents a significant opportunity to transform traditional disaster management approaches. Centralized digital platforms can act as integrated hubs, enabling instant reporting, efficient coordination, and transparent resource management [3]. Such systems can ensure that critical information is delivered quickly and accurately to the appropriate stakeholders during emergencies. This research introduces ResQNet, an advanced digital emergency response and disaster management system designed to centralize disaster-related operations and enhance coordination through a unified platform. The system focuses on real-time interaction, automated workflows, and user-friendly interfaces to address the limitations of existing manual and semi-digital systems.

The main contributions of this work are as follows:

- Developed a scalable full-stack system using React.js, Node.js, and MongoDB, integrating real-time data handling with an interactive web-based interface for seamless communication between users, rescue teams, and administrators.

- Implemented a structured disaster reporting and management system that enables users to submit real-time incident data, including location, severity, and resource requirements, ensuring accurate and efficient data processing.
- Created a centralized resource and contribution management framework that ensures transparency and proper allocation of funds, food, medical supplies, and other essential resources during emergencies.
- Designed a coordination mechanism that allows administrators to assign rescue teams based on specialization and availability, supported by instant notification systems for rapid deployment.

The gist of this paper is structured as follows:

- Section 2 reviews related work and highlights gaps in existing disaster management systems.
- Section 3 details the system architecture and workflow processes.
- Section 4 presents the key features and the proposed user interface.
- Section 5 discusses the conclusion and outlines future scope for expansion.

II. LITERATURE RIVIEW

A. Traditional Disaster Management Systems

Traditional disaster management systems, such as those implemented by government authorities, mainly focus on policy-making and coordination frameworks. For example, the National Disaster Management Authority (NDMA) in India provides structured guidelines for disaster response and mitigation [4]. However, these systems lack real-time operational capabilities and direct citizen interaction, as communication is mostly one-way through alerts and advisories.

Similarly, Emergency Operation Centers (EOCs) are used for coordinating disaster response but rely on manual reporting and traditional communication methods [5]. Although effective for internal coordination, they are not accessible to the public and lack centralized digital platforms, leading to delays during emergencies.

B. Digital and Information-Based Systems

With advancements in technology, several digital platforms have been developed to provide disaster-related information and situational awareness. For instance, Google Crisis Response offers real-time alerts, crisis maps, and emergency information to users during disasters [6]. Similarly, platforms like ReliefWeb, managed by the United Nations, serve as global information portals for humanitarian data and reports [7].

While these systems significantly improve information dissemination, they are primarily designed for awareness rather than operational management. They do not support functionalities such as real-time disaster reporting by citizens, resource allocation, or rescue team coordination. Furthermore, they provide limited administrative control for local authorities, making them less effective for ground-level disaster response and management.

C. Research Gap

Existing disaster management systems either focus on policy-level coordination or information dissemination, but lack an integrated, real-time platform for effective response. There is no unified system that combines citizen participation, resource management, and rescue coordination. Additionally, most existing solutions do not provide seamless interaction between multiple stakeholders on a single platform, leading to delays and inefficiencies during emergencies [8].

ResQNet addresses this gap by providing a centralized platform for real-time disaster reporting, resource tracking, and automated rescue team deployment. It ensures efficient communication and coordination among users, administrators, and rescue teams, enabling faster and more effective disaster response.

Table I. Comparative Analysis of Existing Disaster Management System

Product	Key Features	Limitations	Conversational Interface
National Disaster Management Authority (NDMA), India [9]	Disaster policies and coordination frameworks	Not real-time; limited citizen interaction; one-way communication	Not available

Emergency Operation Centers (EOC) [10]	Government emergency response coordination	Manual processes; no public access; no centralized dashboard	Not available
Google Crisis Response [11]	Crisis alerts and maps	No resource management; no team coordination; limited control	Not available
ReliefWeb [12]	Global disaster information platform	Not for local use; no reporting or operational control	Not available

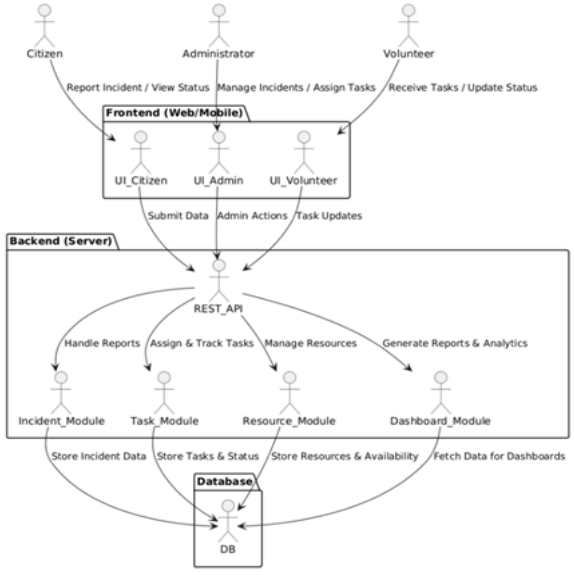
III. METHODOLOGY/ PROPOSED WORK

This document outlines the detailed approach for developing the ResQNet platform. It covers the system architecture, core functionalities, development workflow, and the specific tools and technologies used to build an efficient and scalable disaster management system.

A. System Architecture

As illustrated in Fig. 1, users who use the system include citizens, administrators and volunteers with a dedicated web/mobile frontend to each individual role. These interfaces interact with the backend operations through RESTful APIs that manage critical processes like incident reporting, assigning and tracking tasks, resource management, and generating analytics [13]. The backend consists of modular components such as incident, task, resource, and dashboard modules to perform respective operations. Incident data, task updates, and resource availability are all stored in a central database which is accessed by the application as per need. This organized setup allows for smooth interaction among components, effective data handling, and the scalability of the entire system.

Fig. 1. ResQNet - System architecture diagram



B. Algorithms, Mathematical Models, and Workflow

The core functionality of ResQNet is based on structured data processing, rule-driven workflows, and real-time system interaction, enabling efficient disaster reporting, resource management, and rescue coordination.

1) User Request Ingestion: A user interacts with the system by performing actions such as reporting a disaster, contributing resources, or viewing active incidents (e.g., submitting a disaster report with location and severity details).

2) Preprocessing: The system processes incoming data to ensure consistency and accuracy:

a) Validation: Ensures all required fields (title, type, location, severity) are provided.

b) Data Formatting: Standardizes input data such as location and text fields for uniform storage and retrieval.

3) Processing Logic: ResQNet uses structured backend logic and role-based operations to handle different functionalities:

a) Disaster Reporting: When a user submits a disaster report, the system stores the information along with uploaded files and assigns a unique disaster ID.

b) Resource Contribution Handling: If a user contributes resources (funds, food, medical aid), the system links the contribution to a specific disaster and updates the total resource ledger dynamically.

c) College Type Queries: Administrators assign rescue teams based on availability and specialization. The system updates team status and links them to the respective disaster.

d) Dashboard Data Processing: The system aggregates data such as total disasters, contributions, and active teams to generate real-time analytics for monitoring.

e) Fallback Handling: If any operation fails (e.g., invalid input or missing data), the system returns appropriate error messages to guide the user.

4) Data Management: All system data is stored in a centralized database consisting of collections such as users, disasters, contributions, and rescue teams.

a) Tool: MongoDB is used for efficient data storage, retrieval, and real-time updates. Mongoose ODM enables schema-based modeling and structured data operations.

5) Response Generation: Based on processed data:

- The system generates confirmation responses for actions such as disaster reporting and contributions.
- Dynamic dashboards display updated disaster and resource information.

Response to User: The processed response is sent to the frontend, where it is displayed through an interactive user interface, ensuring real-time feedback and smooth user experience.

C. Tools, Datasets, and Technologies

1) Backend

a) Environment: Node.js is used for scalable server-side execution.

b) Framework: Express.js handles API routing and business logic.

c) Authentication: JSON Web Tokens (JWT) and bcrypt ensure secure user authentication and data protection.

2) Frontend

a) Framework: React.js provides a dynamic and responsive user interface.

b) Design Elements: Modern UI components, routing, and state management ensure smooth navigation and interaction.

3) Dataset and Data Storage

a) Database: MongoDB Atlas stores structured data such as users, disasters, contributions, and rescue teams.

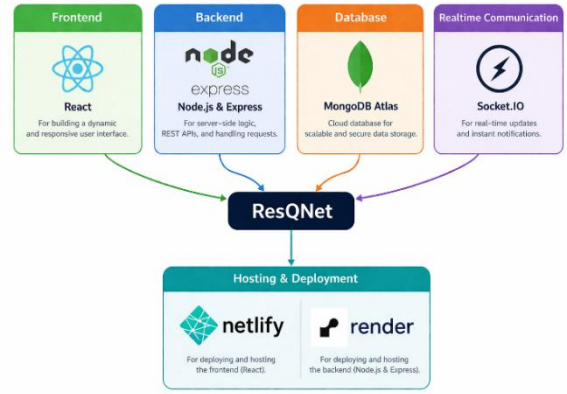
b) Data Handling: Mongoose is used for schema definition and efficient database operations.

4) Workflow and Processing

a) ResQNet follows a structured workflow-based approach to handle disaster events, resource allocation, and rescue coordination.

b) The system processes user actions through REST APIs, applies backend logic, and returns real-time responses, ensuring efficient and reliable operation.

Fig. 2. ResQNet - Tech stack

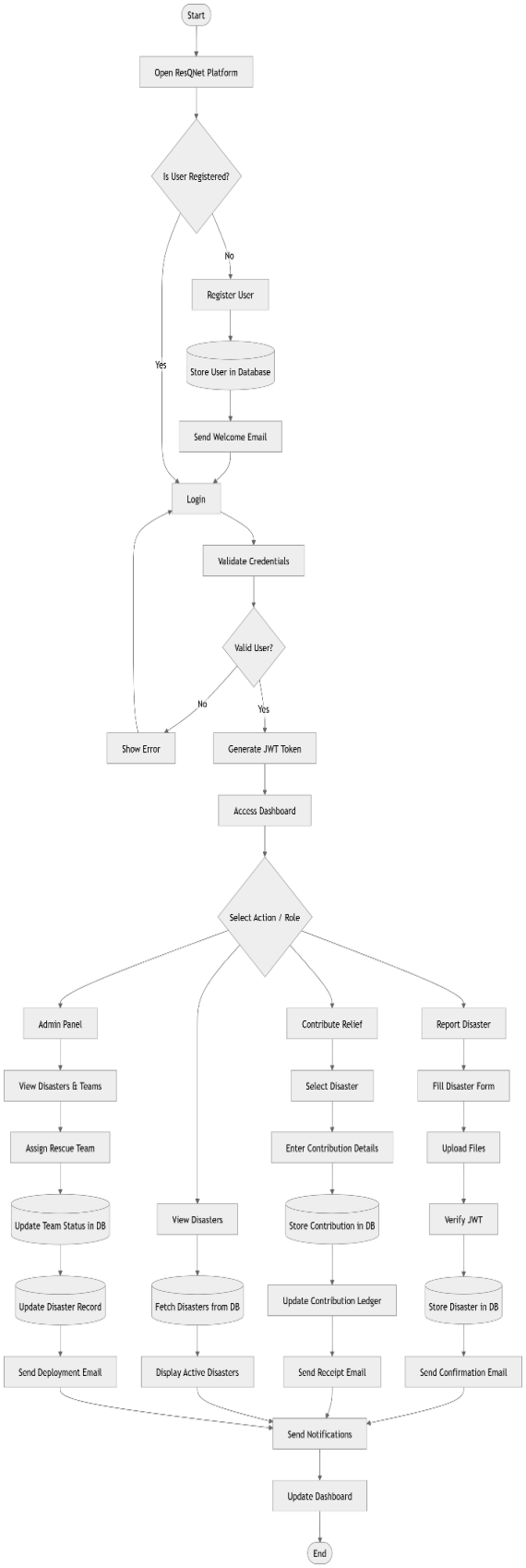


5) Flowchart

In ResQNet, users begin by accessing the platform through the landing page and then proceed to register or log in to the system. If the user is new, they complete the registration process, after which their details are securely stored in the database and a welcome email is sent. Upon successful login, the system generates a JSON Web Token (JWT) to ensure secure authentication and grants access to the dashboard, where users can interact with various functionalities based on their role.

Once inside the system, users can perform multiple actions such as reporting disasters, contributing resources, viewing active incidents, or accessing the admin panel. When a user reports a disaster, the system collects details such as location, severity, and description, along with optional file uploads. The submitted data is validated, verified using JWT, and stored in the database, after which a confirmation email is sent. In the case of contributions, users select a specific disaster, provide resource details, and the system updates the contribution ledger dynamically while sending a receipt email for confirmation.

For administrative operations, administrators can view all reported disasters and registered rescue teams. Based on the severity and requirements of a disaster, the admin assigns appropriate rescue teams. The system then updates the team status and disaster records in the database and sends deployment notifications to the respective teams [14]. Additionally, users can view active disasters, where the system retrieves and displays real-time data from the database in an organized format.



IV. IMPLEMENTATION AND RESULTS

A. Implementation Details

The ResQNet system was developed as a full-stack web application utilizing Node.js and Express.js for the backend and React.js for the frontend, deployed on cloud-based platforms for scalability and accessibility. The application architecture emphasizes modularity and real-time processing by integrating RESTful APIs with a centralized database system. The backend logic was implemented using Express.js, which efficiently handles request processing, authentication, and data operations, while MongoDB Atlas is used for storing and managing structured data related to users, disasters, contributions, and rescue teams.

User requests are processed through a sequence of API calls designed to handle operations such as disaster reporting, resource contribution, and rescue team assignment. Based on these operations, the system retrieves and updates relevant data in the database, ensuring accurate and real-time information flow. Middleware components such as JWT authentication are used to secure endpoints, while bcrypt ensures password encryption. Additionally, Nodemailer is integrated to provide automated email notifications for actions like registration, reporting, and deployment alerts.

The development environment consisted of Node.js running on a system equipped with a modern processor and sufficient memory to support real-time operations, while deployment was carried out using platforms such as Netlify (frontend) and Render (backend), ensuring high availability and performance.

B. Experimental Results

To evaluate the performance of ResQNet, multiple test scenarios were conducted, including disaster reporting, resource contribution, and rescue team assignment. The system demonstrated high efficiency in processing real-time data, with quick response times for API requests and seamless updates across the platform.

The disaster reporting module successfully stored and displayed incident data with minimal delay, while the contribution system accurately tracked and updated resource availability. Rescue team assignment operations were executed effectively, with immediate status updates and notification delivery.

However, performance may vary under extremely high traffic conditions or network constraints, indicating potential areas for improvement such as load balancing, caching mechanisms, and further optimization of API responses.

C. User Interface and Experience (UI/UX)

The user interface was designed with a focus on simplicity, responsiveness, and intuitive interaction. The platform provides a clean and modern interface accessible across desktop and mobile devices.

Dedicated dashboards and pages allow users to easily report disasters, view active incidents, and contribute resources. Administrators are provided with structured control panels for managing operations and assigning rescue teams. The integration of clear visual elements, responsive layouts, and guided navigation ensures a smooth user experience, especially during high-stress emergency situations.

Fig. 4. ResQNet - Reporting a disaster

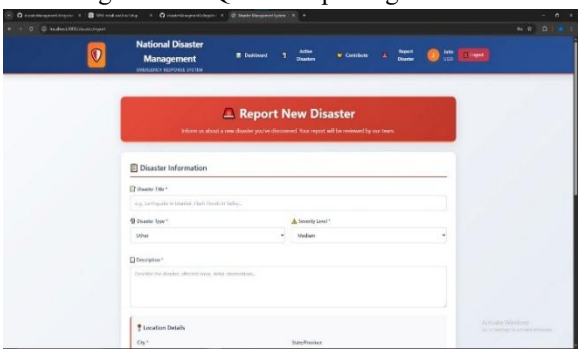
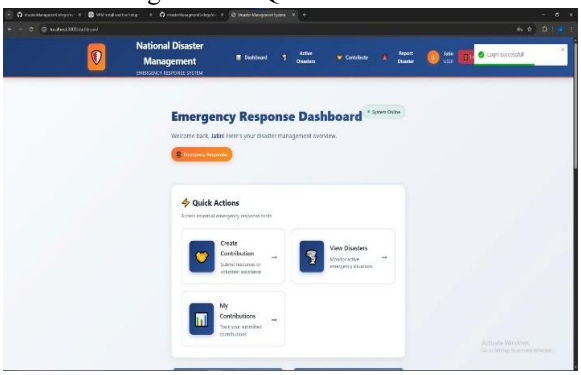


Fig. 5. ResQNet - Dashboard



V. DISCUSSION

A. Figures and Tables

The experimental results demonstrate that the proposed ResQNet system significantly improves the efficiency of disaster management operations compared to traditional manual and semi-digital approaches. The system effectively handles real-time disaster reporting, resource tracking, and rescue team coordination, ensuring faster response times and improved communication among stakeholders. This performance validates the core approach of using a centralized digital platform for managing complex emergency scenarios.

The project proves to be highly effective in real-world conditions as it directly addresses the critical issue of fragmented communication and lack of coordination during disasters. By centralizing information and automating key processes such as reporting, resource allocation, and notification delivery, ResQNet reduces delays and enhances decision-making for administrators and rescue teams.

However, the current system has certain limitations. The platform's performance may be affected under extremely high user traffic or limited network connectivity, which can impact real-time data updates. Additionally, the system currently relies on user-reported data, which may sometimes lack accuracy or completeness. These challenges indicate areas for improvement, such as implementing advanced validation mechanisms, integrating real-time external data sources, and optimizing system scalability.

In the future, the system's robustness can be enhanced by incorporating AI-based predictive analytics for disaster forecasting, integrating live data feeds such as weather APIs, and implementing intelligent resource allocation algorithms. These improvements would further strengthen the system's ability to handle large-scale emergencies efficiently.

V. CONCLUSION AND FUTURE WORK

This research successfully demonstrated the development of ResQNet, an advanced digital platform that modernizes disaster management and emergency response processes. By leveraging real-time data processing, centralized coordination, and scalable web technologies, the system enables efficient disaster reporting, resource allocation, and rescue team deployment. The solution provides a practical, scalable, and user-friendly interface that enhances communication among civilians, administrators, and rescue teams, while significantly improving response efficiency during critical situations.

Future work for this project includes several key areas: 1) Scalability Enhancement: The system can be further optimized to handle large-scale disaster scenarios by implementing load balancing, distributed architectures, and caching mechanisms to ensure consistent performance under high traffic conditions.

2) Integration with External Data Sources: ResQNet can be enhanced by integrating real-time data from external APIs such as weather forecasting systems and early warning services to improve situational awareness and proactive response.

3) Mobile Application Development: A dedicated mobile application can be developed to utilize device GPS for accurate location tracking and provide faster access to emergency services.

4)AI-Based Predictive Analytics: Future improvements include incorporating machine learning models for disaster prediction, risk assessment, and intelligent resource allocation to further enhance system effectiveness.

REFERENCES

[1] S. R. Patel and A. Verma, "Design and development of an integrated disaster management information system," *International Journal of Computer Applications*, vol. 182, no. 25, pp. 14–22, 2023.

[2] K. Tanaka, "Role of IoT and GIS in real-time disaster monitoring and early warning systems," *Journal of Geospatial Technologies*, vol. 27, no. 3, pp. 77–89, 2022.

[3] L. Sharma and R. Bose, "Use of cloud-based platforms for multi-agency coordination during natural disasters," in *Proc. Int. Conf. Smart Cities & Resilient Infrastructure*, 2021, pp. 210–218.

[4] M. Singh, "Flood forecasting and management using sensor networks and data analytics," *Environmental Modelling & Software*, vol. 153, pp. 85–96, 2023.

[5] Chatterjee, D. Roy, and S. Gupta, "AI-enhanced disaster prediction using satellite imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–9, 2022.

[6] P. Mehta and V. Bansal, "Emergency communication systems for rural disaster zones," *International Journal of Disaster Risk Reduction*, vol. 74, pp. 102–118, 2022.

[7] S. Wei, N. Ye, and S. Zhang, "Adaptive algorithms for improving disaster alert accuracy," in *Proc. Int. Conf. Computer Science & Service Systems*, IEEE, 2012.

[8] R. B. Shriram *et al*., "AI-enabled response system for real-time disaster outreach," in *Proc. Int. Conf. Emerging Technologies*, 2020, pp. 95–101.

[9] National Disaster Management Authority, "NDMA Official Website." [Online]. Available: NDMA Official Website

[10] Federal Emergency Management Agency, "Emergency Operations Centers Overview." [Online]. Available: FEMA EOC Overview

[11] Google, "Google Crisis Response." [Online]. Available: Google Crisis Response

[12] United Nations Office for the Coordination of Humanitarian Affairs, "ReliefWeb." [Online]. Available: ReliefWeb

[13] H. Li, L. Jianfeng, and Y. Gangqiao, "Smart response systems for low-resource disaster regions," in *Proc. IEEE Int. Conf. Management Science and Engineering*, 2009, pp. 68–74.

[14] S. Dhamodaran and J. Refonaa, "Application of AI for automated disaster classification," *International Journal of Applied Engineering Research*, vol. 10, no. 76, pp. 353–358, 2015.