A Secure and Collaborative Rescue Coordination Platform (ResQhub) for Disaster Management

Dr. Mohammed Mujeerulla[1], Zafar Ali Khan N[2] , Shakkeera L[3], Pushpalatha M[4]
Siddaprasad Gwadi[5], Vithal Reddy[6] , Yashwantha M[7]

(1) Associate Professor, School of Computer Science and Engineering, Presidency University, Bengaluru, Karnataka, 560064, India

(2) Professor, School of Computer Science and Engineering, Presidency University, Bengaluru, Karnataka, 560064, India

(3) Professor, School of Computer Science and Engineering, Presidency University, Bengaluru, Karnataka, 560064, India

(4) Assistant Professor, School of Computer Science and Engineering, Presidency University, Bengaluru, Karnataka, 560064, India

(5) (6) (7) Student, School of Computer Science and Engineering, Presidency University, Bengaluru, Karnataka, 560064, India

Abstract— **Disasters, whether natural or man-made, demand rapid, reliable, and coordinated rescue efforts to minimize damage and save lives. Existing solutions such as government portals, SOS applications, and social media tools often fall short due to delayed communication, inefficient resource allocation, and a lack of structured collaboration among multiple agencies. To overcome these challenges, this paper presents ResQhub, a secure and collaborative web-based rescue coordination platform that integrates real-time incident reporting, live location tracking, multi-agency communication, and AI-powered chatbot assistance. The system has been implemented using HTML, CSS (Bootstrap), and JavaScript for the frontend, Node.js with Express for the backend, MongoDB for persistence, and JWT for secure authentication. Unlike traditional systems, ResQhub enriches user-submitted reports through chatbot interactions, ensuring that incidents are described with greater clarity and accuracy before being dispatched to relevant agencies. Rescue teams are automatically matched using proximity-based allocation, while administrators can oversee operations through a dashboard that enables manual overrides, alerts, and audit logging. By combining location-aware technologies with intelligent guidance, the platform significantly enhances situational awareness, reduces duplication of effort, and promotes seamless collaboration between rescue stakeholders. Results from simulated scenarios demonstrate the feasibility and effectiveness of ResQhub, while future enhancements will focus on large-scale deployment, integration with IoT-based sensors, and predictive analytics for disaster preparedness .**

Index Terms—**Disaster Management, Rescue Coordination, Web Application, AI Chatbot, MongoDB, Node.js, Real-Time Systems**

I. Introduction

Natural and man-made disasters such as floods, earthquakes, industrial accidents, and pandemics require timely response and coordinated actions across multiple agencies to minimize loss of life and property. Existing disaster management portals and mobile SOS applications are often limited to providing static information or emergency contact numbers and fail to support real-time multi-agency collaboration, resource allocation, or guided assistance for civilians [6]. These shortcomings result in delayed communication, inefficient use of available resources, and lack of transparency in rescue operations.

To address these issues, ResQhub has been conceived as a secure and collaborative rescue coordination platform. ResQhub provides role-based access for public users, rescue agencies, and administrators. It integrates live location tracking, AI- driven guidance, and a dashboard-based control center to en- sure that incident reports are enriched, verified, and promptly assigned to the most appropriate agencies [9]. By combining real-time data collection, location-aware resource allocation, and chatbot-assisted user interaction, ResQhub seeks to sig- nificantly improve the efficiency and reliability of disaster response.

II. Literature Survey

A thorough review of existing literature is essential to understand the strengths and limitations of current disaster management approaches. Previous research has explored a wide range of techniques, including crowdsourcing, mobile sensing, IoT integration, blockchain systems, and AI-based solutions for early warning and response [8]. These studies provide valuable insights into how technology has been used to improve situational awareness, resource coordination, and communication during emergencies. However, they also highlight persistent challenges such as scalability, data reliability, privacy concerns, and lack of structured collaboration across multiple agencies [7]. By analyzing these works, it becomes clear that no single system fully addresses all the requirements of an effective disaster response platform.

Below are selected survey/case-study works relevant to ResQhub:

- **Crowdsourced Social Media Data for Disaster Response (PetaJakarta)** – Real-world Twitter crowdsourcing for flood mapping; shows strengths in situational awareness and weaknesses in verification and scalability .

- **Crowdsourcing the Disaster Management Cycle** – Examines crowdsourcing use across preparedness, response, and recovery .
- **Mobile Crowdsensing in Disaster Management** – Systematic review of smartphone sensors and GPS for incident reporting .
- **IoT and WSNs in Disaster Management** – Surveys sensor networks for early warning; highlights integration challenges .
- **Blockchain for Emergency Systems** – Demonstrates immutability benefits but notes scalability concerns .
- **Multi-Agency Collaboration Challenges** – Identifies barriers in information sharing that ResQhub aims to resolve .
- **Situation Awareness Models for Emergency Response** – Reviews frameworks informing dashboard and notification design .
- **Social Media + AI Integration for Disasters** – AI techniques to filter and prioritize crisis data.

III. Objectives

ResQhub has been designed with multiple goals in mind. At its core, the platform aims to bridge the gap between citizens and multiple rescue agencies during disasters by providing secure, real-time, and location-aware coordination [6].

The specific objectives are:

- Design a secure web platform for disaster response.
- Enable real-time communication and collaboration among agencies.
- Optimize rescue allocation using location-aware features.
- Enhance user engagement via AI-driven chatbot guidance.
- Ensure transparency and auditability of all actions.

IV. Existing Methods and Drawbacks

A variety of disaster management systems already exist, including government portals, mobile SOS apps, and social media-based tools. While these platforms provide important services, they often lack mechanisms for structured, multi-agency collaboration and guided user support [6]. These shortcomings lead to delays, duplication of effort, and poor situational awareness.

Key drawbacks identified in existing methods:

- Lack of live multi-agency coordination.
- Manual and inefficient resource allocation.
- Absence of automated chatbot guidance for civilians.
- Limited transparency or role-based audit logs.

V. Proposed Methodology

ResQhub integrates multiple modules into a single, unified workflow that streamlines reporting, allocation, and monitoring. The platform combines location-aware resource selection with an AI chatbot to guide users and enrich incident reports before they are dispatched to agencies [2], [3], [5].

The methodology follows these steps:

1) Public user reports incident with live location and optional media.
2) AI chatbot interacts to clarify details and provide immediate guidance.
3) Leaflet library is used for map integration.
4) Agencies receive real-time notifications and update resource status.
5) Agencies can collaborate with other agencies to work faster and help users.

VI. Feasibility Study

ResQhub's architecture and implementation choices make it technically, operationally, and economically feasible for large-scale deployment. The platform effectively combines open-source frameworks, real-time cloud infrastructure, and user-centered design to ensure reliable and affordable performance in emergency response scenarios [8].

Key aspects of feasibility:

- **Technical Feasibility:** The system is built using React for the frontend, Node.js and Express for the backend, and MongoDB for database management. Firebase integration enables secure authentication and real-time data updates. These technologies are proven, scalable, and compatible with cloud hosting environments such as AWS and Google Cloud, ensuring stability and high availability during peak demand.
- **Operational Feasibility:** ResQhub offers an intuitive and accessible interface for all user roles — citizens, agencies, and administrators. Features like live tracking, role-based dashboards, and automated notifications simplify coordination during disaster situations. The platform's modular structure also allows easy onboarding of new agencies or regions without major technical adjustments.
- **Economic Feasibility:** The use of open-source libraries and cloud-based deployment minimizes upfront and maintenance costs. The system requires only basic hosting and API expenses, making it cost-effective for government or NGO adoption. Its architecture also supports gradual upgrades and integration of AI modules in the future without large financial overheads.

VII. System Architecture

The architecture of ResQhub follows a three-tier design consisting of the client layer, the server layer, and the data layer. The client layer includes separate interfaces for public users, rescue agencies, and administrators, all built with React.js to ensure a responsive and modern user experience. These clients communicate with a set of RESTful APIs implemented using Node.js and Express.js on the server layer. Security and authentication are handled by JSON Web Tokens (JWT), providing role-based access control [1], [4]. The server layer also integrates with external services such as Google Maps APIs for geolocation and sendgrid Messaging for push noti- fications. At the data layer, MongoDB stores incident reports, agency information, resource availability, and system logs.This modular architecture allows for seamless scalability, making it easier to onboard additional rescue agencies or expand to new regions without major structural changes. Real-time data synchronization ensures that both users and agencies receive the latest updates instantly, enhancing situational awareness and coordination. Furthermore, the design supports fault tolerance by enabling efficient error handling and retry mechanisms in case of network disruptions. The separation of concerns across layers also simplifies maintenance, testing, and future feature enhancements, such as AI-based predictive alerts or integration with IoT sensors for automated incident detection. Overall, this architecture provides a robust, secure, and flexible foundation for an effective disaster management platform.

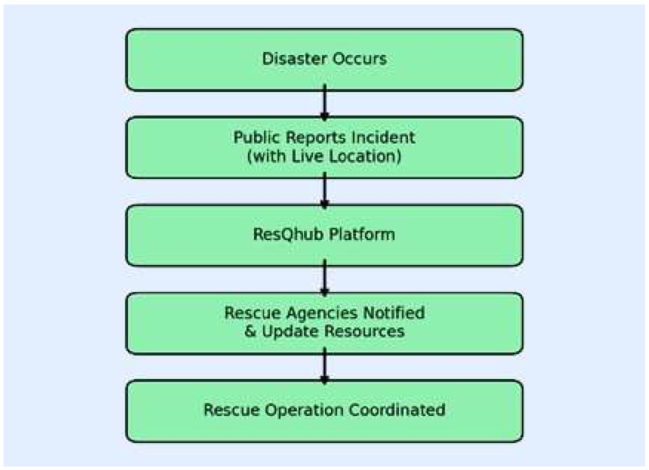

Disaster Occurs
Public Reports Incident (with Live Location)
ResQhub Platform
Rescue Agencies Notified & Update Resources
Rescue Operation Coordinated

Fig. 1: High-level architecture: clients (Public/Agency/Chatbot), REST API (Node/Express), Auth (JWT), DB (MongoDB), Notification and Map services.

VIII. Modules

ResQhub is composed of several interconnected modules that together deliver an end-to-end disaster management workflow. Each module is designed to handle a distinct part of the process while integrating seamlessly with the others.

Main modules include:

- **Public User Module:** Incident reporting, live tracking, chatbot guidance.
- **Agency Module:** Resource management, accept/reject requests, status updates.
- **AI Chatbot Module:** Guided reporting, enrichment, FAQ and safety instructions.
- **Map Integration:** Leaflet library.
- **Notification Module:** Email notifications via SendGrid.

IX. Hardware and Software Requirements

Hardware Requirements:

- Minimum: Intel i5 processor, 8GB RAM, 256GB SSD, internet connectivity.

TABLE I: Comparative Feature Analysis

Feature	Traditional Portals	ResQhub
Live Location Tracking	×	✓
Multi-Agency Collaboration	×	✓
AI Chat-bot Assistance	×	✓
Role-Based Access	Limited	✓
Resource Management	Manual	Automated

- Recommended: Intel i7 processor, 16GB RAM, 512GB SSD for heavier workloads.

Software Requirements:

- Frontend: HTML, CSS (Bootstrap), JavaScript.
- Backend: Node.js, Express.js.
- Database: MongoDB.
- Authentication: JWT.
- API Integrations: Gemini API key.
- Deployment: Render, Netlify.

X. Implementation and Results

The ResQhub platform is implemented using a modern full-stack architecture. The front-end, built with HTML, CSS (Bootstrap), provides distinct dashboards for public users, rescue agencies, and administrators. The backend, built with Node.js and Express.js, exposes secure RESTful APIs that handle incident reports, resource management, and authentication. MongoDB stores data, while JWT ensures secure role-based access and notifications. [2].

During simulated disaster scenarios such as floods and fires, ResQhub demonstrated:

- Reduced time to allocate nearby agencies compared to manual coordination.
- Improved user engagement via the AI chatbot, leading to fewer incomplete reports.
- Real-time updates on resource availability and incident statuses.

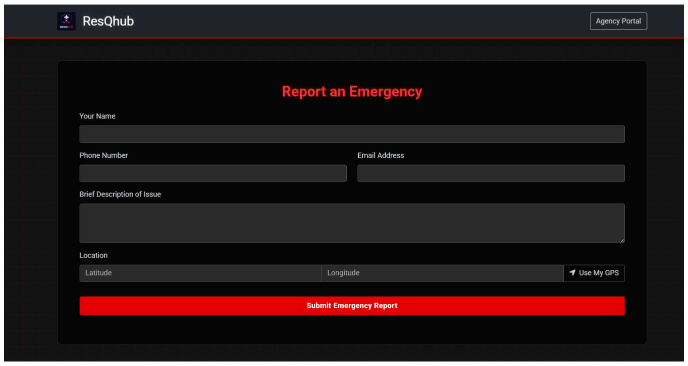
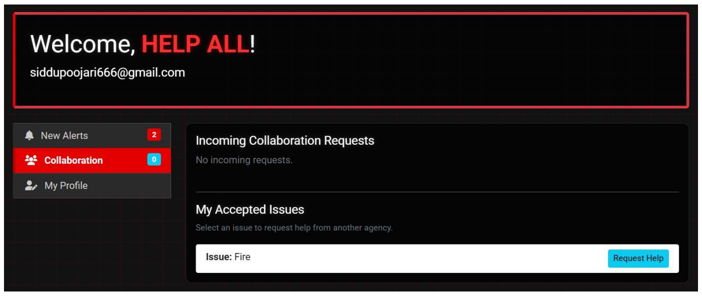


Fig. 2: Homepage of ResQhub to raise help

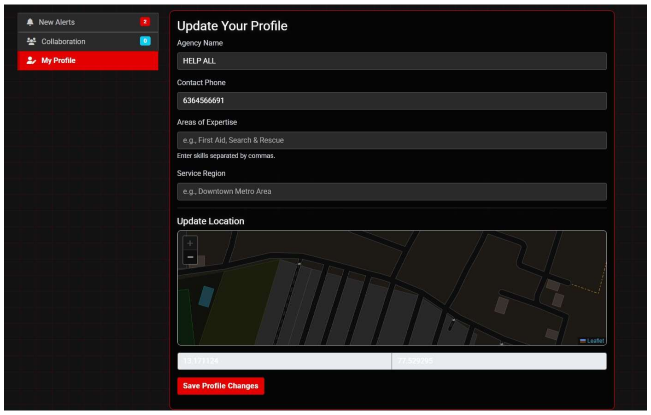

Welcome, HELP ALL!
siddupoojari666@gmail.com

Incoming Collaboration Requests
No incoming requests.

My Accepted Issues
Select an issue to request help from another agency.

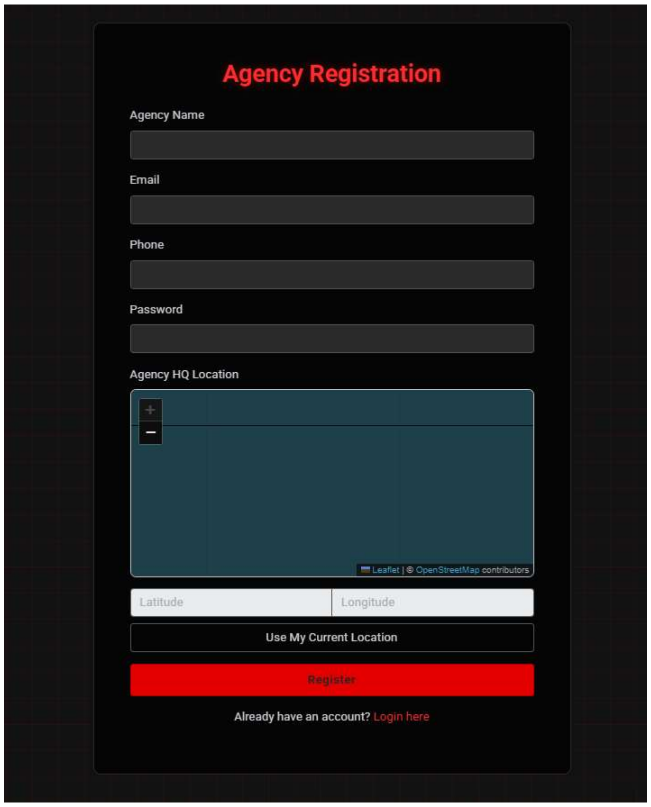
Issue: Fire

Fig. 3: Agencies collaborations



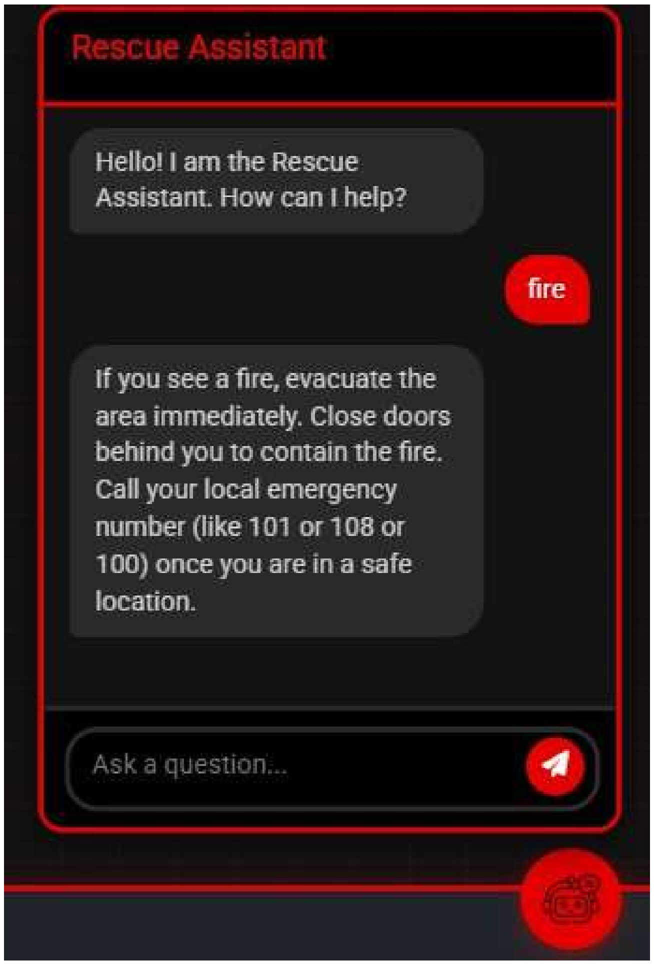
Update Your Profile
Agency Name: HELP ALL
Contact Phone: 6364566691
Areas of Expertise: e.g., First Aid, Search & Rescue
Enter skills separated by commas.
Service Region: e.g., Downtown Metro Area
Update Location
13.171124 77.529295

Fig. 4: Dashboard of agencies.


Agency Registration
Agency Name
Email
Phone
Password
Agency HQ Location

Already have an account? Login here

Fig. 5: Registrations of new agencies.

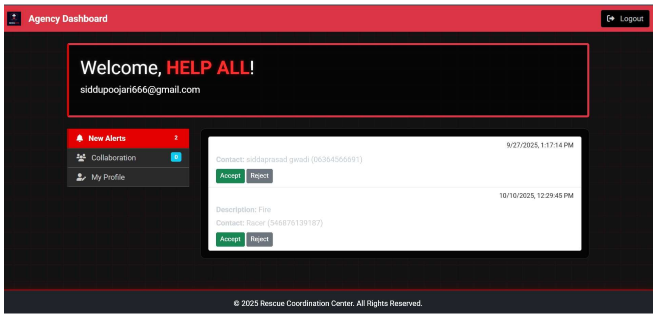

Rescue Assistant

Hello! I am the Rescue Assistant. How can I help?

fire

If you see a fire, evacuate the area immediately. Close doors behind you to contain the fire. Call your local emergency number (like 101 or 108 or 100) once you are in a safe location.

Fig. 6: Chatbot for critical conditions.



Welcome, HELP ALL!
siddupoojari666@gmail.com

9/27/2025, 1:17:14 PM
Contact: siddaprasad gwadi (06364566691)

10/10/2025, 12:29:45 PM
Description: Fire
Contact: Racer (546876139187)

© 2025 Rescue Coordination Center. All Rights Reserved.

Fig. 7: Alerts on ResQhub issues.

XI. Implementation Details

A. *Geospatial Proximity Query (Haversine Formula)*

The core function of identifying nearby agencies is achieved through a geospatial query based on the Haversine formula. The equation computes the great-circle distance between two latitude/longitude points, so it can rank which agencies are closest to a given location on Earth's surface. How it's used For each agency's coordinates, plug user location and agency location into the Haversine formula to get distance in km or miles, then sort by the smallest values to return "nearby" agencies efficiently.

Applying a cutoff radius (e.g., 10 km) filters only agencies within that radius; this works well for geospatial queries because the formula remains accurate and stable even for short distances.

$$d = 2r \arcsin\left(\sqrt{sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos(\varphi_1)\cos(\varphi^2)\sin 2\left(\frac{\Delta\lambda}{2}\right)}\right) \quad (1)$$

where d is the distance between two points on the Earth.

$$\Delta\varphi = \varphi_2 - \varphi_1, \qquad \Delta\lambda = \lambda_2 - \lambda_1, \qquad (2)$$

where φ_1, φ_2 are latitudes and λ_1, λ_2 are longitudes.
Here, r is the Earth's radius. The formula calculates the shortest distance over the Earth's surface (great-circle distance) between two points specified by their latitude (φ) and longitude (λ). where:

- d = Distance between the two points
- r = Radius of the Earth
- lat_1, lon_1 = Latitude and longitude of the first point (emergency)
- lat_2, lon_2 = Latitude and longitude of the second point (agency)

MongoDB Implementation:

```
// Mongoose Query
Agency.find({
  location: {
    $near: {
      $geometry: {
        type: 'Point',

        coordinates: [<longitude>,<latitude>]
        // Emergency coords
     },$maxDistance: 20000 //
     Maximum radius in meters (20 km)
    }
  }
});
```

B. *Password Security (bcrypt Hashing Algorithm)*

Passwords are protected using a one-way cryptographic hash function with a random salt. The choice of cryptographic techniques is informed by recent studies on lightweight ECC and Strobe-based security for IoT systems, ensuring secure handling of sensitive user and agency data [2], [5].

```
// Hashing a new password (Registration)
const salt = await bcrypt.genSalt(10);
const hashedPassword = await bcrypt.hash
(plainTextPassword, salt);

// Verifying password (Login)
const isMatch = await bcrypt.compare
(loginPassword, hashedPasswordFromDB);
```

C. *Authentication (JWT Digital Signature Algorithm)*

User sessions are secured using JWT with HMAC-SHA256.

```
// Creating a token
const payload = {agency:{id:agencyId }};
const token= jwt.sign(payload,
process.env.JWT_SECRET,{expiresIn:'5h'});

//Verifying a token
Const decoded=Jwt.verify(token,
process.env.JWT_SECRET);
req.agency = decoded.agency;
// Access payload data
```

The JWT signature is generated using the following formula:

$$\text{Signature} = \text{HMAC-SHA256}\ \ \text{base64UrlEncode(header)} + \text{"."} + \text{base64UrlEncode(payload)},\ \ \text{secret\_key} \qquad (3)$$

D. *explanation of above Signature formula*

This formula produces a "proof stamp" for the message by combining a secret key with the exact concatenated text formed as Base64URL(header) + "." + Base64URL(payload), then applying HMAC-SHA256 to generate a short, secure signature. The signature is attached to the JWT so that the receiver can verify the integrity and authenticity of the token. The secret key acts like a shared passcode between trusted agency systems. If any field (such as 'agencyid', role, or expiry) is modified, the recalculated signature will not match the original, causing the request to be rejected and preventing forgery or tampering across agency APIs.

Best Practices:

- Encode the header and payload properly, then join with a dot before computing HMAC.
- Send all three JWT components (header, payload, signature) together.
- Use short expiry times and rotate secret keys regularly for enhanced security.

TABLE II: Project Development Timeline

Phase	Duration
Requirement Analysis & Literature Survey	2 weeks
System Design & Wireframes	3 weeks
Frontend Development	4 weeks
Backend Development	4 weeks
Integration & Map API Work	2 weeks
Testing & Scenario Simulations	3 weeks
Deployment & Documentation	2 weeks

XII. Conclusion and Future Work

This paper introduced **ResQhub**, a secure, web-based platform designed to improve the way disaster response and rescue operations are managed. By addressing long-standing issues such as communication delays, unverified reporting, and inefficient resource distribution, ResQhub demonstrates how technology can make rescue efforts faster, more reliable, and more transparent [6], [7]. A key strength of the platform lies in its ability to streamline collaboration across multiple agencies. Instead of operating in silos, different rescue teams can share real-time updates, allocate resources effectively, and maintain accountability through role-based dashboards and audit logs. For civilians, the AI chat-bot ensures that reports are clear, accurate, and supplemented with safety guidance, reducing the risk of incomplete or misleading information. Although the system has been validated through controlled simulations, future development could focus on scaling the platform for real-world deployments. Potential improvements include integrating predictive models for resource forecasting, linking with IoT-based sensor networks for early alerts, and conducting live field trials in collaboration with government and non-governmental disaster response organizations [8]–[10].

References

[1] M. Mujeerulla, M. Preethi, M. S. Khan, et al., Demerits of Elliptic Curve Cryptosystem with Bitcoin Curves Using Lenstra–Lenstra–Lovasz (LLL) Lattice Basis Reduction," *Arab J Sci Eng*, vol. 49, pp. 4109–4124, 2024. doi:10.1007/s13369-023-08116-w.

[2] M. S. Khan, T. M. Chen, M. Sathiyanarayanan, M. Mujeerulla, and S. P. Raja, "Application of Lenstra–Lenstra–Lovasz on Elliptic Curve Cryptosystem Using IoT Sensor Nodes," *Journal of ICT Standardization*, vol. 12, no. 4, pp. 381–408, 2025. doi:10.13052/jicts2245-800X.1242.

[3] Preethi, M. Mujeer Ulla, S. R. Sapna, and R. M. Devadas, "Blockchain modeled swarm optimized Lyapunov smart contract deep reinforced secure tasks offloading in smart home," *MethodsX*, vol. 14, 103305, 2025. doi:10.1016/j.mex.2025.103305.

[4] P. Preethi, M. Mujeer Ulla, G. P. K. Yadav, K. S. Roy, R. A. Hazarika, and K. S. K. Saxena, "Elliptic-Curve Cryptography Implementation on RISC-V Processors for Internet of Things Applications," *Journal of Engineering*, vol. 2024, Article ID 5116219, 9 pages. doi:10.1155/2024/5116219.

[5] M. M. Ulla, Preethi, M. S. Khan, S. L., and S. B. J., "Lightweight Strobe Security Libdisco Scheme for IoT-Based Sensor Data," in *Proc. 2024 8th Int. Conf. on Computational System and Information Technology for Sustainable Solutions (CSITSS)*, Bengaluru, India, 2024, pp. 1–6. doi:10.1109/CSITSS64042.2024.10817029.

[6] M. Kapucu and A. Garayev, "Challenges in multi-agency collaboration in disaster management," *Journal of Contingencies and Crisis Management*, vol. 24, no. 1, pp. 1–10, 2016.

[7] J. R. Endsley and K. Jones, "Situation Awareness in Multi-Agency Emergency Response: Models and Challenges," *Safety Science*, vol. 117, pp. 23–34, 2019.

[8] M. Imran et al., "Edge technologies for disaster management: Social media and AI integration – a survey," *Future Internet*, vol. 12, no. 5, 2020.

[9] H. Wang, S. Li, and J. Chen, "AI-Driven Disaster Warning System: Integrating Predictive Data with Large Language Models," Proc. ACM/IEEE Conf. on Humanitarian Technology, pp. 101–110, 2022.

[10] Y. Liu et al., "Application of Blockchain Technology in Emergency Management Systems: A Bibliometric Analysis," *Applied Sciences*, vol. 11, no. 4, 2021.