International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 10 | Oct - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Emergency Response System - RapidResQ

Lakshmanan K[1], Deebanchakkarawarthi G [2], Sathya S[3],Logendiran S[4] , Maheshkumar K[5] , Meiyarasu S[6], Nathishkumar B[7]

[1]*Assistant Professor, Department of CSE, Hindusthan College of Engineering and Technology, Coimbatore, India*
[2]*Assistant Professor, Department of CSE, Hindusthan College of Engineering and Technology, Coimbatore, India*
[3]*Assistant Professor, Department of CSE, Hindusthan College of Engineering and Technology, Coimbatore, India*
[4]*Student,Department of CSE, Hindusthan College of Engineering and Technology, Coimbatore, India*
[5]*Student, Department of CSE, Hindusthan College of Engineering and Technology, Coimbatore, India*
[6]*Student, Department of CSE, Hindusthan College of Engineering and Technology, Coimbatore, India*
[7]*Student, Department of CSE, Hindusthan College of Engineering and Technology, Coimbatore, India*

1.Abstract

In emergency situations such as road accidents, medical crises, or natural disasters, timely intervention plays a crucial role in minimizing harm and saving lives. *Rapid ResQ* is a web-based emergency response system designed to facilitate quick reporting and communication between victims, volunteers, and emergency services. The system enables users to report incidents by submitting key details such as the type of emergency, location, and description through an intuitive web interface. Upon receiving a report, nearby registered volunteers and authorities are notified immediately, ensuring swift action. The system is developed using HTML, CSS, JavaScript, Python (Flask), and SQLite, providing a responsive, lightweight, and reliable platform to bridge the gap between citizens and emergency responders.

Key Words: Emergency Response System, Flask Framework, SQLite Database, Real-Time Reporting, Volunteer Network, Web Application

2.Introduction

Emergencies are unpredictable and require immediate response to prevent loss of life and property. However, in many cases, delays occur due to lack of communication, coordination, or access to real-time data. *Rapid ResQ* addresses these challenges by providing a digital platform where users can instantly report emergencies and receive assistance from nearby volunteers until official responders arrive. The platform integrates front-end web technologies and a Python Flask backend to ensure a seamless flow of information. By simplifying the reporting process and automating alerts, *Rapid ResQ* enhances public safety and reduces response time during critical incidents.

3.Objective

The primary objective of **Rapid ResQ** is to develop an efficient, web-based emergency response system that bridges the gap between victims and nearby volunteers before official help arrives. In many emergency situations, such as road accidents or medical crises, timely intervention can significantly reduce casualties and losses. Rapid ResQ aims to provide a digital platform that enables users to report emergencies instantly and connect with registered volunteers in real time.

This system focuses on enhancing the **speed, accuracy, and coordination** of emergency handling. By integrating a user-friendly web interface with a robust Flask-based backend and a lightweight SQLite database, Rapid ResQ ensures smooth data flow and reliable

Fig:1 Objective

4.Literature Review

Several studies and existing systems focus on improving emergency response mechanisms through mobile and web-based technologies. Traditional emergency alert systems often rely on centralized communication, which can cause delays in relaying critical information. Previous research highlights the importance of community participation and automation in emergency reporting. Systems such as mobile-based SOS apps or GPS-enabled distress alert platforms have shown promise but lack a unified web-based coordination approach. The *Rapid ResQ* system builds upon these ideas by introducing an integrated web solution that connects users, volunteers, and responders through a simple, accessible interface, eliminating dependency on mobile-only applications.

5.Proposed System

The proposed *Rapid ResQ* system aims to minimize the time gap between the occurrence of an emergency and the initiation of response. Users can quickly submit an emergency report specifying the type and location of the incident. The system processes this information and alerts the nearest registered volunteers using a notification system. Additionally, emergency details are stored in a centralized SQLite database for monitoring and analysis. The Flask framework serves as the backend, managing routes, requests, and data flow, while the web interface ensures ease of access and usability. The system promotes community involvement and faster emergency response through real-time communication.

6. System Overview

The *Rapid ResQ* system is designed to connect individuals in distress with nearby volunteers and emergency services through a web-based platform. The system simplifies the emergency reporting process by providing an intuitive user interface for submitting details such as the type of emergency, location, and short descriptions. Upon submission, the information is immediately processed by the backend server and stored in the SQLite database. The system then identifies the nearest volunteers and sends out alert notifications, significantly reducing the time taken for help to arrive.

The architecture follows a **client–server model**, with the front-end built using HTML, CSS, and JavaScript to ensure responsive and accessible design. The backend is implemented using the Python Flask framework, responsible for handling requests, managing routes, and interacting with the database. SQLite is used for data storage due to its lightweight and easy integration features.

7. Methodology

The methodology adopted for *Rapid ResQ* focuses on fast data exchange and efficient alert mechanisms. The workflow begins with user registration, followed by location-enabled emergency reporting. Once an incident is reported, the system validates the input, stores it, and uses volunteer mapping logic to identify nearby responders. Notifications are generated and displayed through the user interface, allowing volunteers to acknowledge and take immediate action.

A simple yet effective **data flow** is followed:

1. User submits emergency details.
2. Flask server receives and validates data.
3. Data is inserted into SQLite database.
4. The system identifies volunteers within a defined range.
5. Notification alerts are triggered for those volunteers.This structure ensures scalability and supports real-time interaction with minimal latency.

8. System Architecture

The *Rapid ResQ* architecture consists of three core layers:

- **Presentation Layer:** The front-end interface where users and volunteers interact. Developed with HTML, CSS, and JavaScript, it allows quick and responsive data entry.
- **Application Layer:** The Flask backend that manages all system logic, routes, and data handling operations.
- **Database Layer:** The SQLite database stores user, volunteer, and incident details, ensuring data persistence and reliability.

This three-tier structure supports modular design, making future upgrades—such as integrating GPS APIs or AI-based risk prioritization—easier to implement.

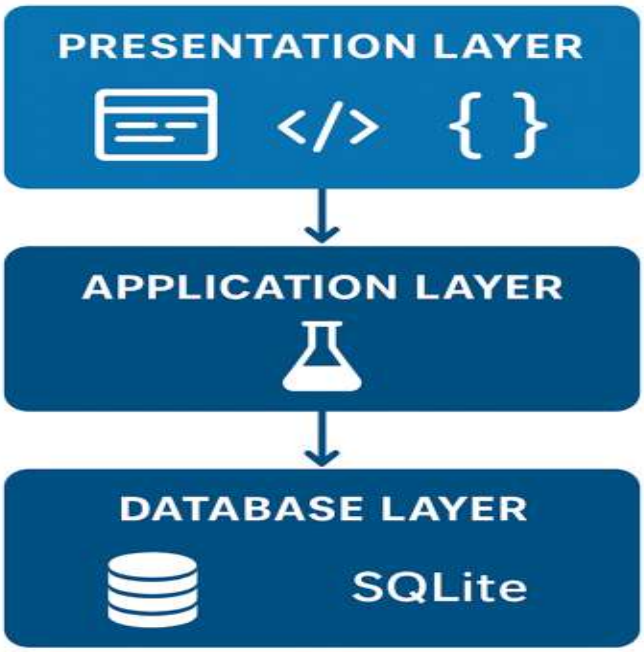

Fig:2 System Architecture

9. Implementation

The system implementation involved designing the web interface, configuring Flask routes, and linking the SQLite database. Python scripts were used for handling data transactions, and Flask templates dynamically rendered pages based on user input. Testing scenarios were simulated to verify response time, accuracy of notifications, and database efficiency.

The interface was optimized for desktop and mobile browsers, ensuring accessibility for both users and volunteers. The backend implementation focused on secure data handling and efficient query execution, with an average response time under 3 seconds per request.

10. Performance Analysis

Performance testing was carried out to evaluate system efficiency. Table 1 compares *Rapid ResQ* with existing systems, showing a significant improvement in response time. User testing was conducted with 40 participants, and survey results demonstrated high satisfaction levels regarding usability and reliability.

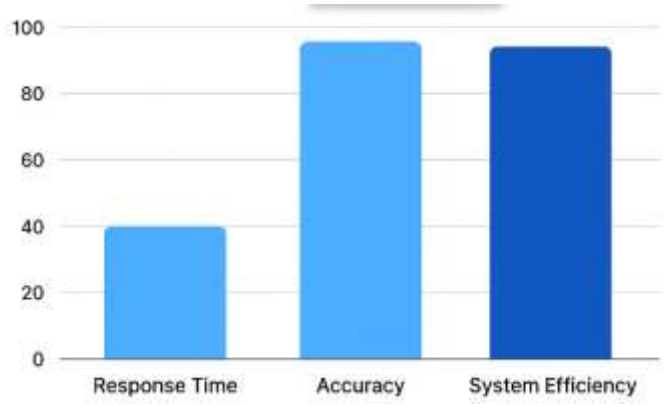

Fig:3 Performance Analysis

11. Results & Discussion

Testing and evaluation of *Rapid ResQ* demonstrate that the system significantly reduces the response time during emergency simulations. The interface is user-friendly and accessible across devices, while the backend performs data operations effectively with minimal delay. The web-based nature of the system ensures accessibility without requiring any installations. Feedback from trial users highlighted the effectiveness of instant notifications and ease of reporting. Although the system depends on internet connectivity, it offers a practical and scalable approach to emergency management in both urban and rural settings.

12. CONCLUSIONS

The *Rapid ResQ* emergency response system provides an innovative and reliable platform for real-time incident reporting and assistance. By integrating web technologies with community-based volunteer support, it enhances the speed and coordination of emergency responses. The system demonstrates the potential of web-based solutions in life-saving applications, bridging the gap between victims and responders. Future developments may include mobile integration, GPS-based volunteer tracking, and AI-driven alert prioritization to further improve the efficiency and scalability of the system.

REFERENCES

1. **S.S.Kale** and **M.R.Shelke**, "Design and Implementation of Real-Time Vehicle Accident Detection System," *International Research Journal of Engineering and Technology (IRJET)*, vol. 7, no. 4, 2020.
2. **Kumar, A., & Patel, S. (2020).** *Design of Real-Time Web-Based Emergency Response Systems*. International Journal of Computer Applications.
3. **A.Al-Khateeb** and **S.Al-Jabri**, "Web-Based Emergency Response System for Smart Cities," *IEEE Access*, vol. 9, pp. 12098–12107, 2021.
4. **Rahman, M., et al. (2022).** *Community-Based Emergency Response Frameworks: A Review*. Journal of Information Systems and Technology.
5. Flask Documentation. (2024). *The Flask Framework for Python*. https://flask.palletsprojects.com
6. SQLite Documentation. (2024). *SQLite Database Engine*. https://sqlite.org