Capstone Project Proposal – IS4110

ResQNow

Crowdsourced, Location-Aware Accident Reporting and Emergency Response App

Group No 06 - Information Systems (22/23)

Department of Computing and Information Systems

Faculty of Computing

Sabaragamuwa University of Sri Lanka

11th of January 2026

Approval of Capstone Project

1. Title of the Capstone Project: ResQNow
   Crowdsourced Disaster Response & Accident Reporting App
2. Details of Group members:

Index No	Name with Initials	Email	Mobile No	Signature of Student
22FIS0480	G.P.C. Thushani	22fis0480@ms.sab.ac.lk	+94 775533274	Charitha.
22FIS0482	R.A.C. Ruwanima	22fis0482@ms.sab.ac.lk	+94 758781860	[illegible]
22FIS0483	A.M.G.D. Thawinsa	22fis0483@ms.sab.ac.lk	+94 763732085	[illegible]
22FIS0484	A.T. Kalansooriya	22fis0484@ms.sab.ac.lk	+94 766197566	[illegible]
22FIS0488	K.A. Wijesekara	22fis0488@ms.sab.ac.lk	+94 716053892	Anjana.

3. Name of the Mentor: Mr. G.D.K.C. Weerasooriya
4. Mentor's designation: Manager IT
5. Mentor's Organization: Agro Ventures Plantations (Pvt) Ltd.

For office use only:

Approved/Not approved: Approved

Signature of the Mentor: [illegible]

Date: 2026/01/11

Suggestions if any: —

Table of Contents

ACKNOWDEGDEMENT

We would like to express our sincere gratitude to all the people and organizations that helped us create this project proposal for the ResQNow. The foundation of this ground-breaking work is teamwork and expert consultation with a variety of sources.

Special thanks go out to Prof. S. Vasanthapriyan, who is the coordinator of the capstone project at Sabaragamuwa University of Sri Lanka and the dean of the faculty of computing. We will always be appreciative of the trust placed in our group and for fostering an atmosphere that encourages creativity and scholarly inquiry.

We would especially like to thank the university administration, especially Mrs. W.T. Saranga Somaweera, assistant coordinator of the Faculty of Computing, and Dr. L.S. Lekamge, head of the Department of Computing and Information Systems.

Our internal supervisor, Mrs. R.M.K.K. Wijerathne of the Sabaragamuwa University of Sri Lanka's Faculty of Computing, deserves special recognition. Their consistent scholarly guidance and real-world experience influenced this project's technical viability.

We hold our guide, Mr. G.D.K.C. Weerasooriya, the IT Manager, in the highest regard. His exceptional industry knowledge, which was based on his extensive and long-standing experience in the field, provided invaluable constructive criticism as well as clear, useful guidance for shaping this project.

Our project team members; G.P.C.Thushani, R.A.C.Ruwanima, A.M.G.D.Thawinsa, A.T. Kalansooriya and K.A. Wijesekara deserve special recognition for their collaborative efforts. This success is based on their high level of teamwork, perseverance, and dedication as a whole. We are also grateful to our friends and family for their unwavering support.

We have every confidence that combining these efforts will result in a system that transforms emergency resource planning and real-time disaster preparedness.

Group Details and Contribution

Table 01: Group Details and Contribution

Contribution Area	**K.A. Wijesekara 22FIS0488 (Project Manager)**	**R.A.C. Ruwanima 22FIS0482 (System Analyst)**	**G.P.C. Thushani 22FIS0480 (Backend Developer)**	**A.T. Kalansooriya 22FIS0484 (Frontend Developer)**	**A.M.G.D. Thawinsa 22FIS0483 (QA Engineer / Developer)**
Frontend Development	Designed admin dashboard layout and workflow validation screens	Created wireframes, user flows, and form validation rules	Assisted with frontend API integration points	Developed mobile UI screens (Report, Profile, Map) using React Native	Tested UI components and implemented minor UI fixes
Backend Development	Implemented admin-level APIs and role validation logic	Designed database schema and system workflows (ERD, DFD)	Developed core REST APIs, authentication, verification logic	Integrated frontend with backend APIs	Implemented test endpoints and assisted in bug fixes
Cloud / DevOps	Managed deployment planning and cloud architecture decisions	Analyzed feasibility of cloud services and data storage	Configured AWS services, hosting, and database	Assisted with environment setup and API testing	Conducted deployment testing and monitored system stability
UX / HCI Design	Coordinated usability goals and emergency interaction constraints	Designed user personas and disaster reporting journeys	Ensured backend responses support fast UX	Applied UI/UX principles to mobile and web interfaces	Conducted usability testing and reported UX issues
Information	Defined report	Designed verification	Implemented flagging, misuse	Displayed verification	Tested misuse
Verification & Trust	Credibility policy and escalation rules	Workflow and validation rules	Voting and verification scoring logic	Status and badges in UI	Scenarios and false reporting cases

GIS & Location Services	Oversaw location based feature requirement	Designed location data flow and map interactions	Implemented GPS data handling and location storage	Integrated maps and live incident visualization	Tested map accuracy and location edge cases
Notification & Alerts	Defined alert prioritization and responder routing logic	Modeled notification triggers and system flow	Implemented FCM-based alert dispatch logic	Displayed alert feedback and status updates	Tested alert delivery and failure scenarios
Security & Privacy	Ensured compliance with privacy policies and access rules	Defined RBAC rules and data protection requirements	Implemented JWT authentication and secure APIs	Applied frontend access restrictions	Conducted security and access testing
Testing & Quality Assurance	Reviewed test strategy and milestone validation	Prepared test scenarios based on requirements	Fixed backend issues identified during testing	Fixed UI bugs found during testing	Led functional, integration, and usability testing
Documentation & Research	Prepared proposal, reports, and progress documentation	Created system models, diagrams, and requirement docs	Documented APIs and backend workflows	Documented UI components and usage	Maintained test reports and defect logs

1 Introduction and Objective of the Capstone Project

1.1 Introduction

Natural disasters and auto accidents are examples of emergencies that happen frequently and often call for an immediate response from the appropriate authorities. Crucial information regarding these incidents is frequently dispersed, delayed, or untrustworthy. In order to contact emergency services, victims and witnesses typically use conventional channels like phone calls or social media posts. This can result in a lack of communication, redundant reporting, and ineffective responder coordination. As a result, critical time is lost, response activities are less successful, and occasionally avoidable property and human losses occur. Most of the emergency reporting applications that exist today are powered by manual authority-based verification, which increases response latency in any emergency. Most modern systems have no built-in facilities to gauge the credence of public reports, and therefore they treat all submissions equally without considering their credibility. There is no integrated, trust- aware platform which blends public reporting, community validation and automated responder alerting in real-time. The proposed ResQNow system fills these gaps with the introduction of a crowdsourced verification mechanism combined with automated dispatch logic.

The ResQNow App is suggested as a comprehensive digital platform intended to enhance the information flow during emergencies in order to address these issues. Through a straightforward and user-friendly interface, the system allows regular citizens to actively participate in the emergency response process by reporting accidents or disasters in real time. Important information like the incident type, location, description, and optional photos are included in these reports. These details are then sent to a centralized database and displayed on a real-time map [1].

Through a dedicated dashboard, authorities such as law enforcement, medical facilities, and other emergency responders can keep an eye on these reports, confirm their veracity, and take the appropriate action. A quicker and more effective response is ensured by providing responders with real-time alerts based on their proximity to the reported incident. ResQNow fosters a cooperative ecosystem that facilitates accurate reporting, quicker decision-making, and improved public safety by bringing the public, authorities, and responders together in a single system.

The system's primary function is accident reporting, which facilitates prompt communication between emergency services and citizens. But because of its scalability, it can be expanded in the future to accommodate other disasters like fires, landslides, and floods. Through the use of crowdsourcing, contemporary technology, and efficient data management, the team hopes to contribute to a community that is safer and more responsive.

1.2 Objective of the project

By using a crowdsourced digital platform, the ResQNow App aims to increase emergency response activities' speed, accuracy, and coordination. The general and particular goals that direct the creation and application of the system are described in this section.

1. Provide an intuitive mobile and web interface that enables citizens to register, log in, and promptly report incidents with pertinent information, including type, description, location, and images.
2. Install a centralized incident management system that keeps track of all events that are reported.
3. To establish an automated, proximity-based alert and dispatch mechanism that notifies the nearest verified responders immediately after community-based verification.
4. To incorporate a crowdsourced verification system that allows registered users to confirm or flag incidents, improving data reliability and reducing false or duplicate reports.
5. To protect data privacy and stop unwanted access, create a secure authentication system with contemporary technologies like JWT (JSON Web Tokens).
6. Provide an administrative dashboard that classifies incidents by location, type, and severity to help authorities visualize data.
7. Incorporate cloud-based services for scalability, media storage, and database hosting.
8. To enable responders to acknowledge alerts and update incident status, ensuring accountability and transparent emergency coordination.
9. Before the system is finally released, make sure it is reliable, functional, and easy to use by putting a thorough testing and deployment plan into action.
10. Enable users to participate in real-time emergency data collection to raise public safety awareness and engagement.

1.3 Novelty of the ResQNow Application

The ResQNow system offers a number of innovative features that set it apart from the existing solutions used for emergency reporting. Unlike conventional solutions where the decision-making process is based entirely upon manual checks and governmental procedures, the ResQNow system also uses crowdsourced intelligence and real-time prediction tools to provide a quick and reliable way for emergencies to be responded to through the platform.

1. **Crowdsourced Verification Engine**
   What's unique to ResQNow is the incorporation of the trust-aware verification system, where users are able to up-vote or flag reports of incidents. This generates the Verification Score, which prevents false reports, ensuring the accuracy of reports without overloading the authorities.

2. **Automated Dispatch of Alerts based on Verification**
   The vast majority of emergency apps require the user to share the incident manually. However, the ResQNow emergency service has the advantage of notifying the closest responders only after reaching a certain level of verification in the submitted report.

3. **Real-Time Danger Zone Detection**
   ResQNow operates among the pioneering community-driven services with the ability to identify areas with clusters of incidents and determine Danger Zones. The users who visit such zones receive immediate notifications.

4. **Trust Badges and Reporter Credibility Tracking**
   It keeps an internal credibility score for each user, making valid reporters more credible and lessening noise from fraudulent ones.

5. **Live Map Featuring Multi-Level Real-Time Visualization**
   Instead of simple report listings, ResQNow has:
   - Incident heat maps
   - Severity coloring based on the severity of
   - Danger Zone Overlays
   - Respondent movement tracking It offers an advanced level of situational awareness interface than the conventional reporting app.

6. **Automated Proximity-Based Responder Routing**

   Rather, the response to the distress signal comes through the automatic activation of:

   - nearest verified responders identified
   - routing instructions generation
   - logs all alerts for transparency

7. **Cloud-Native Scalability**

   The whole system is cloud-integrated for real-time functionality, handling thousands of reports at the same time in instant notification.

8. **Community-Focused Emergency**

   ResQNow enables the community to move from observers to participants in the management of disasters and accidents, contributing to a dynamic safety net.

2 Analysis

2.1 Problem and Requirement Analysis

Problem Summary

During accidents, disasters, or emergency situations, authorities often face response delays due to incomplete, duplicated, or unverified information from the public. Citizens rely heavily on informal communication channels such as phone calls and social media, which leads to misinformation, lack of prioritization, and inefficient coordination among responders. Existing systems do not sufficiently integrate trust validation, responder accountability, and centralized real-time monitoring within a single platform. As a result, emergency response efforts may be delayed or misdirected.

This project aims to address these challenges by developing a centralized, crowdsourced incident reporting system that integrates community verification, role-based access, and real-time coordination between citizens, authorities, and emergency responders.

High-level Goals

- Provide a reliable, centralized channel for public incident reporting.
- Visualize incidents on a real-time map with severity and status.
- Enable admins to verify and escalate incidents to responders.
- Send automated alerts to nearby responders and relevant authorities.
- Use crowdsourced verification (upvotes/flags) to reduce false reports.
- Introduce trust-based verification mechanisms to improve the accuracy of reported incidents.
- Enable automated, proximity-based alerting to reduce emergency response time.

Stakeholders

- Citizen Reporters (public mobile/web users)
- Admins / Authority Operators (dashboard users)
- Emergency Responders (police, ambulance, fire teams)
- System Administrators / DevOps
- Third-party services (SMS/email/push providers, map tiles, cloud)

2.2 Feasibility Study

2.2.1 Technical Feasibility

The project is technically feasible because it uses reliable and widely supported technologies required for real-time incident reporting, map-based tracking, and alert notifications. Mobile development frameworks such as React Native or Flutter, along with cloud-based backends, ensure cross-platform availability and scalability. GPS, Google Maps API, and push notification services are already mature technologies, reducing technical risks. The development team has the required skills, and all needed tools, documentation, and cloud infrastructure are readily available.

- Uses widely available technologies (GPS, Maps API, Cloud DB).
- MERN/Flutter stack supports scalability and cross-platform deployment.
- Technologies are well-documented and developer-friendly.
- No complex or custom hardware requirements.
- Internet, smartphone, and location services are already standard.
- The team has prior experience with the required tech stack.

The system extends existing technologies by integrating algorithm-based crowdsourced verification and automated alert dispatch. Cloud-based services support real-time verification scoring and notification triggering without significant technical risk The project is Technically Feasible because required technologies and skills are available, accessible, and easy to integrate.

2.2.2 Operational Feasibility

The system is operationally feasible because it enhances existing emergency response workflows by introducing community-assisted verification and automated coordination. Unlike traditional systems where authorities manually validate each report, the proposed system allows verified community input to support faster decision-making. The user interface is designed for rapid incident reporting under stress, while authorities benefit from real-time dashboards and automated alerts. Clear role separation between public users, administrators, and responders ensures smooth operation without disrupting existing emergency procedures. Easy for public users to submit reports and alerts.

- Simple interface reduces training needs.
- Fits real-world emergency response procedures.
- Supports multiple roles: User, Authority, Responder.
- Improves coordination among stakeholders.
- Reduce reporting delays and response time.
- Community-assisted verification reduces the manual workload on authorities.

- Automated alerts allow responders to act faster without manual coordination.
- Clear role separation supports smooth integration with existing emergency workflows.

The project is Operationally Feasible because stakeholders can adopt and use it effectively with minimal effort and high practical benefit.

2.2.3 Economic Feasibility

The project is economically feasible due to its use of open-source frameworks and cloud services with free or low-cost tiers, while delivering high social value through faster emergency response. The novelty of the system lies not in expensive hardware but in intelligent software design, such as crowdsourced validation and automated dispatch. These features significantly reduce operational costs for authorities by minimizing manual intervention and improving response efficiency. Uses free or low-cost hosting and APIs (Firebase/AWS/GCP).

- No licensing fees due to open-source frameworks.
- No specialized hardware cost.
- Low maintenance and upgrade cost.
- Huge long-term social and economic benefits (lives saved, faster response).
- Cost < Value, so investment is justified.

The novelty of the system is achieved through intelligent software design rather than expensive infrastructure. Automated verification and dispatch reduce long-term operational costs for emergency authorities. The project is Economically Feasible due to low development and maintenance cost compared to high social and practical value.

2.3 Data Flow Diagrams (DFD)

2.3.1 Context Diagram

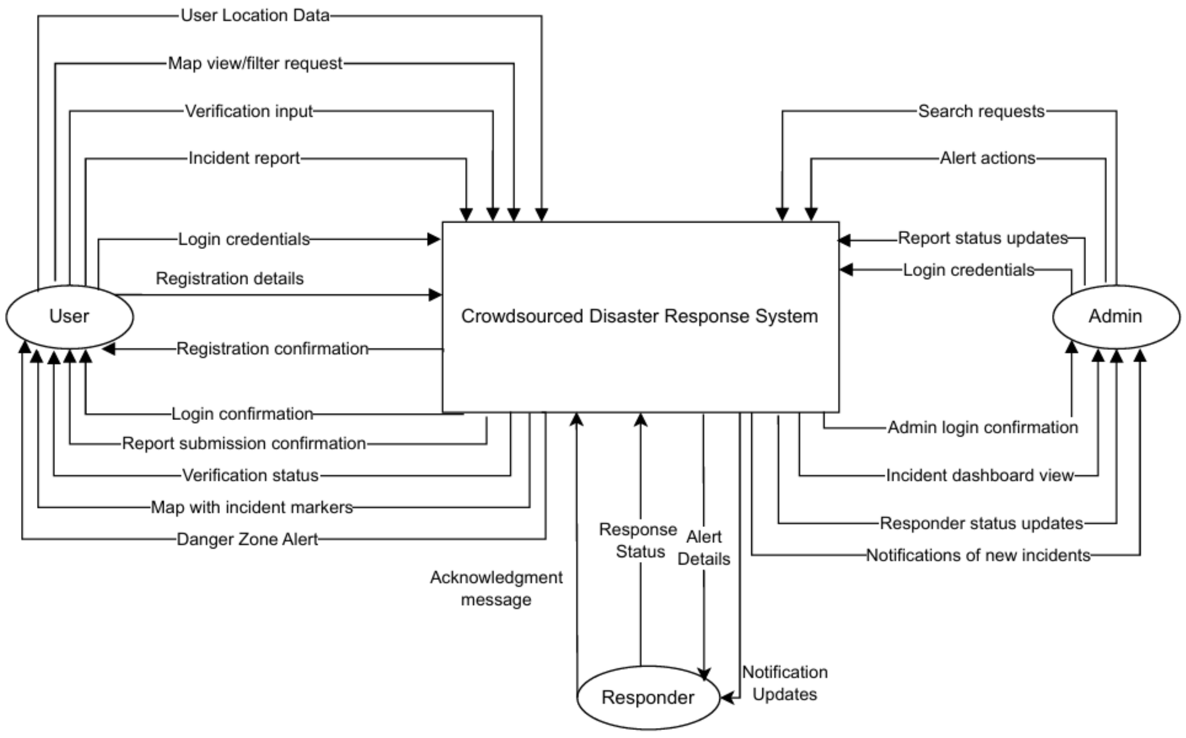

Figure 01: Context Diagram of the ResQNow System

2.3.2 Level 1 Data Flow Diagram

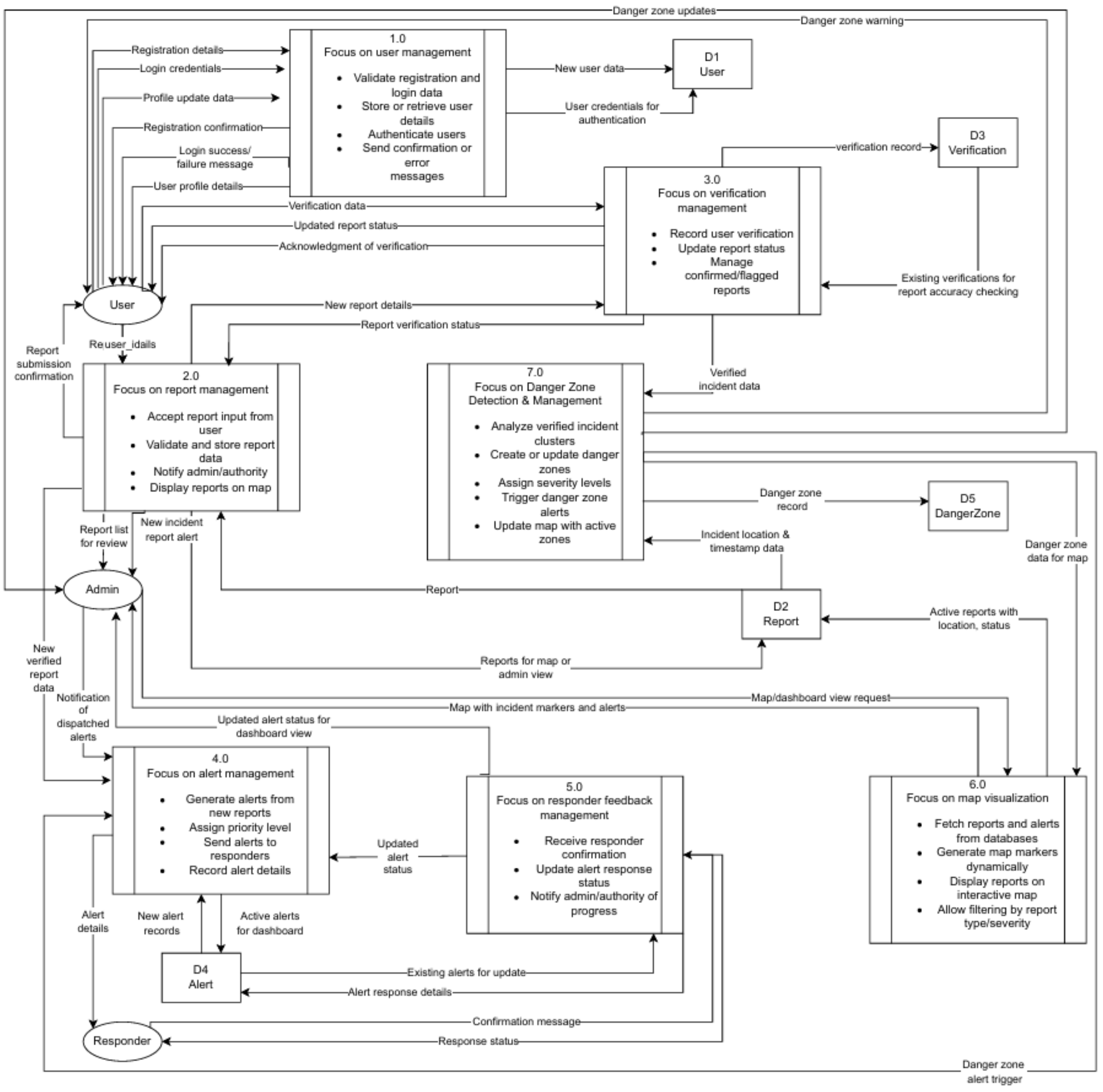
Danger zone updates
Danger zone warning
Registration details
Login credentials
Profile update data
Registration confirmation
Login success/ failure message
User profile details
1.0 Focus on user management
Validate registration and login data
Store or retrieve user details
Authenticate users
Send confirmation or error messages
New user data
D1 User
User credentials for authentication
verification record
D3 Verification
3.0 Focus on verification management
Record user verification
Update report status
Manage confirmed/flagged reports
Verification data
Updated report status
Acknowledgment of verification
Existing verifications for report accuracy checking
User
New report details
Report verification status
Report submission confirmation
Reuser_idails
2.0 Focus on report management
Accept report input from user
Validate and store report data
Notify admin/authority
Display reports on map
Verified incident data
7.0 Focus on Danger Zone Detection & Management
Analyze verified incident clusters
Create or update danger zones
Assign severity levels
Trigger danger zone alerts
Update map with active zones
Danger zone record
D5 DangerZone
Incident location & timestamp data
Danger zone data for map
Report list for review
New incident report alert
Admin
Report
D2 Report
Active reports with location, status
Reports for map or admin view
New verified report data
Notification of dispatched alerts
Map/dashboard view request
Map with incident markers and alerts
Updated alert status for dashboard view
4.0 Focus on alert management
Generate alerts from new reports
Assign priority level
Send alerts to responders
Record alert details
5.0 Focus on responder feedback management
Receive responder confirmation
Update alert response status
Notify admin/authority of progress
Updated alert status
6.0 Focus on map visualization
Fetch reports and alerts from databases
Generate map markers dynamically
Display reports on interactive map
Allow filtering by report type/severity
Alert details
New alert records
Active alerts for dashboard
D4 Alert
Existing alerts for update
Alert response details
Confirmation message
Responder
Response status
Danger zone alert trigger

Figure 02: Level 1 Data Flow Diagram (DFD)

2.4 Entity Relationship (ER) Diagram

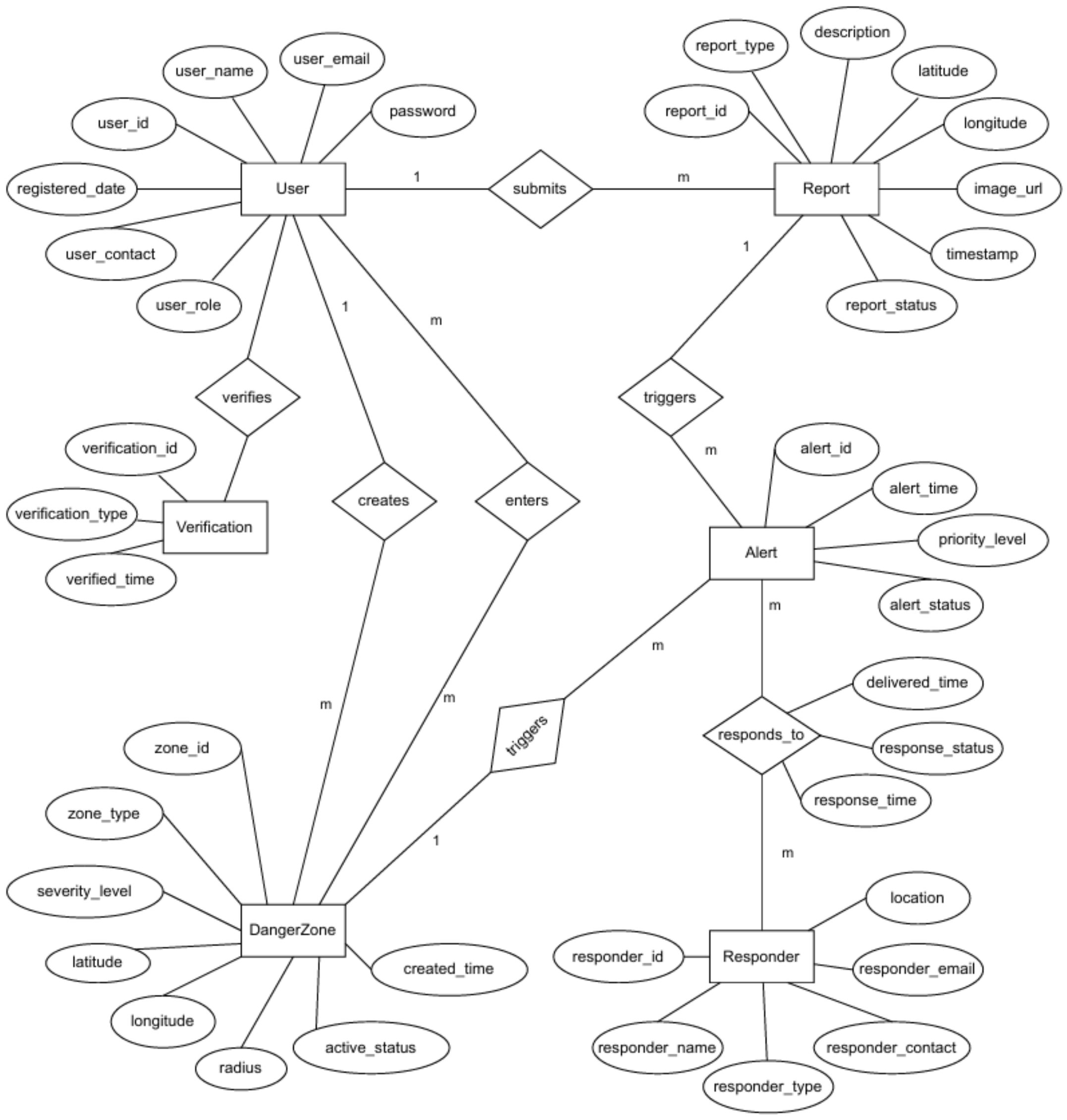

Figure 03: Entity - Relationship (ER) Diagram

2.5 Use Case Diagram

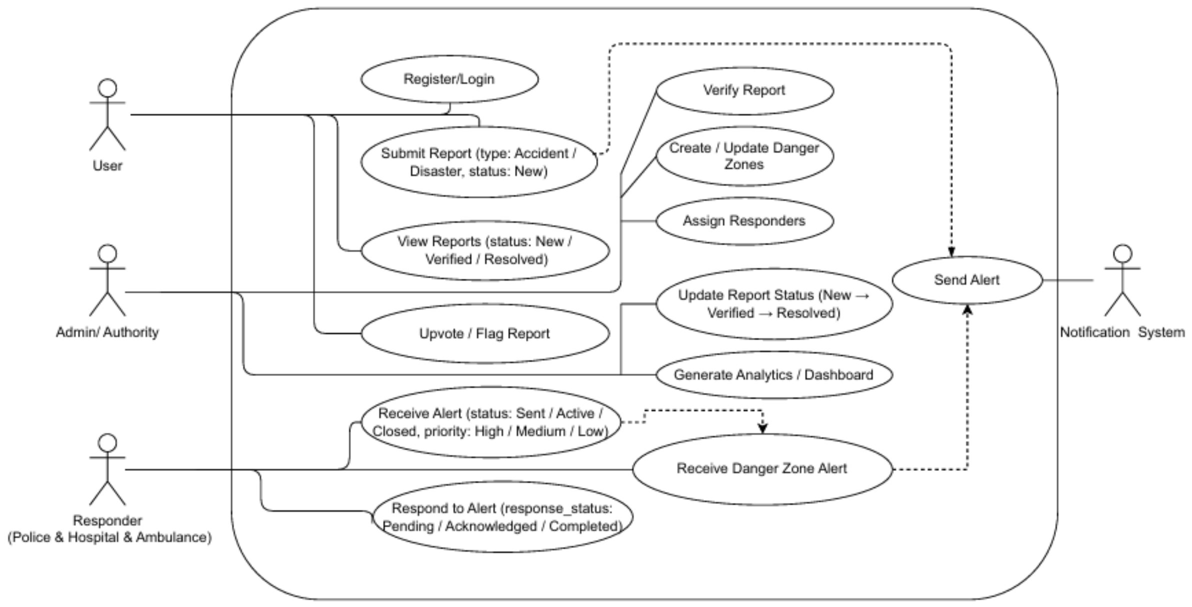
User
Admin/ Authority
Responder
(Police & Hospital & Ambulance)
Notification System
Register/Login
Submit Report (type: Accident / Disaster, status: New)
View Reports (status: New / Verified / Resolved)
Upvote / Flag Report
Verify Report
Create / Update Danger Zones
Assign Responders
Update Report Status (New → Verified → Resolved)
Generate Analytics / Dashboard
Send Alert
Receive Alert (status: Sent / Active / Closed, priority: High / Medium / Low)
Receive Danger Zone Alert
Respond to Alert (response_status: Pending / Acknowledged / Completed)

Figure 04: Use Case Diagram of the ResQNow App

3 Hardware and Software Requirement

3.1 Hardware Requirements

There are two types of hardware requirements: one for end users and one for development and testing.

a) Development Environment

Table 02: Hardware Requirements - Development Environment

Component	Minimum Requirement	Recommended Specification
Processor	Intel Core i3 (10th Gen) or equivalent	Intel Core i5/i7 (11th Gen) or higher
RAM	8 GB	16 GB or higher
Storage	256 GB SSD / HDD	512 GB SSD
Graphics	Integrated graphics	Dedicated GPU (optional for testing maps)
Network	Stable internet connection	High-speed broadband
Devices	Laptop / Desktop	Laptop with mobile testing devices (Android/iOS)

b) User Environment

Table 03: Hardware Requirements - User Environment

Component	Minimum Requirement	Description
Smartphone / PC	Android 8.0+ or Web Browser (Chrome, Edge, Firefox)	For accessing the ResQNow app
Internet Connection	4G / Wi-Fi	To send and receive real-time reports
Location Services	GPS	To share location details

3.2 Software Requirements

The MERN Stack and React Native technologies will be used in the development of the ResQNow App, a cross-platform mobile application with a backend hosted in the cloud. For real-time tasks like reporting, verification, and alerting, this guarantees excellent scalability, dependability, and effective performance.

Table 04: Software Requirements

Software Component	Description	Purpose / Role
Operating System	Windows 10 / 11, Linux, or macOS	Used for development, testing, and deployment environments
Frontend Framework	React.js / React Native	To develop the web site and mobile application for Android and iOS platforms with a single codebase
Backend Framework	Node.js with Express.js [2]	For creating RESTful APIs, managing server-side logic, and connecting with the database
Database	MongoDB Atlas (Cloud-based NoSQL Database) [3]	To store and manage all system data such as user details, reports, verifications, alerts, and responders
Authentication	JSON Web Token (JWT)	For secure login and session management between frontend and backend
Map Integration Library	React Native Maps with Google Maps/ OpenStreetMap [4]	To display real-time accident and disaster reports on a live map
Cloud Hosting Platform	Firebase / AWS / Render	For backend and database hosting, ensuring scalability and accessibility
Version Control System	Git and GitHub	For collaborative code management and version tracking among team members
IDE / Code Editor	Visual Studio Code	For code writing, debugging, and integration testing
Testing Tools	Postman / Jest	For testing API endpoints and validating system performance
Notification Service	Firebase Cloud Messaging (FCM) [5]	To send real-time push notifications to responders and users about incidents

4 Tables, Modules and Structure

4.1 Database Tables / Structures

Table 05: Database Tables / Structures

Collection Name	Key Fields / Attributes	Description / Purpose
Users	user_id, name, email, password, role, contact_number	Keeps track of registered users' details, including their roles (Admin, Authority, and User).
Reports	report_id, user_id, type, description, image_url, location, timestamp, status	Includes the current status and details of every incident that has been reported.
Verifications	verification_id, report_id, user_id, verification_type, verified_time	Keeps track of user confirmations or flags for every report.
Responders	responder_id, name, type, contact, location	Includes information about responders, including ambulance services, police, and hospitals.
Alerts	alert_id, report_id, responder_id, alert_time, alert_status	Keeps track of alerts that are created and forwarded to responders.
DangerZones	zone_id, center_location, radius, severity, created_at, incidents_included, status, and cleared_at	These fields define each danger zone's identity, location, size, severity, creation time, related incidents, and whether it is active or cleared.

4.2 Details of Modules

Table 06: Module Details

Module Name	Key Functions/ Features	Input	Output	Technology
M1- Authentication & User Management	Registration, Login, JWT Token Refresh, Role-Based Access Control (RBAC).	Credentials	User Profile	Node.js, Express, JWT
M2 - Incident Reporting	GPS capture, Camera/Gallery media capture, API POST request, AWS S3 file path generation.	Report Data(Image files)	New Report Document	React Native, Node.js, AWS
M3 - Verification & Status Update	Accepts Vote Type (Upvote/Flag), updates the Verification Score in real-time, triggers status change to "Verified" if threshold is met.	ReportID, VoteType	Update Verification Status	Node.js
M4 - Live Map Dashboard	Downloads live report stream, renders interactive map (via Leaflet), applies filtering, allows Admins to manually set Report status.	Admin status change	Live Map visualization, Updated Report Document	React, MongoDB
M5 - Alert System	Triggers on Verified status. Calculates closest responder, triggers alert (Push Notification) via FCM, logs the event.	Verified Report Document	FCM message	Node.js, AWS
M6 - System Analysis & Design Models	Produces SRS, use cases, DFDs, ERDs, sequence diagrams, and validation of system flow.	Project requirements and stakeholder requirements	UML diagrams, Software Requirement Specification	Draw io, Lucidchart
M7 - Quality Assurance & Testing	Performs unit testing, integration testing, test case design, bug tracking, and regression testing.	Developed system modules, test scenarios, test cases	Test reports, bug reports, validated system	GitHub Issues
M8 - Project Management & Documentation	Planning, scheduling, risk management, progress tracking, report writing, and coordination.	Project plan, milestones, team updates	Project report, progress documentation	MS Word, Google Docs

M9 – Danger Zone Detection & Alerting	Detect clusters of incidents, compute zone radius, assign severity levels, send real-time alerts to users entering zones	Report locations, timestamps, verification scores	Danger Zone creation, automatic alerts, zone updates	Node.js, GeoJSON, Google Maps API, FCM

5 Proposed System

The proposed system is centered around a novel crowdsourced verification mechanism combined with automated emergency dispatch, ensuring accurate reporting, faster response, and improved accountability across all stakeholders.

The system detects Danger Zones, by analyzing clusters of verified incidents and warns users and authorities in proximity. These danger zones are highlighted on the map so as to help avoid risk for people and assist authorities in resource management.

5.1 Functional Requirements

Administrator

- Able to oversee and validate authority accounts, including those for law enforcement and emergency medical services.
- Able to amend or eliminate duplicate or erroneous incident reports.
- Able to keep an eye on all reports, both open and closed, on the dashboard.
- Can alter an incident's status, such as "Verified," "In-Progress," or "Resolved."
- Able to view system analytics, including response times, most frequent locations, and the quantity of reports received each day.

User

- It should be possible for users to register and safely log in with their email address and password.
- Incidents can be reported by users with optional photos or videos, GPS location, type, and description.
- On a real-time interactive map, users can see incidents that have been reported by others.
- Reports can be flagged or verified by users to facilitate crowdsourced validation.
- Reports can be filtered and searched by location, time, or type by users.
- Notifications of nearby verified incidents are available to users.
- Users can see their report history and make changes to their profiles.

Authority

- With validated login credentials, authorities can access the system.
- On the dashboard, they can see and follow incidents in real time.
- To prioritize response, they can filter reports by location, type, or urgency.
- Once they have taken action, they can mark reports as "In Progress" or "Resolved."

- When new verified incidents happen nearby, authorities can get automatic alerts through AWS SNS.

Crowdsourced Validation System (Verification System)

- Based on user flags and upvotes, the backend system automatically determines a verification score for every report.
- A report's status automatically changes to "Verified" once it reaches a predetermined threshold.
- For accuracy and dependability, the system continuously updates the verification status in real time.

Automated Dispatch System & Alert System

- When a report is confirmed, the system automatically sends out alerts via AWS SNS.
- The alert is sent to the appropriate authorities and responders and contains the location, incident type, and description.
- For tracking and reference, every dispatch action is recorded in the database.

Danger Zone Alert System

- The system continuously analyzes reported incidents to identify areas with high-risk activity.
- When more than one incident is verified within a short time and radius, the system declares a "Danger Zone."
- It automatically creates alerts when a user enters or approaches a 'Danger Zone.'
- Manual mark, update, or clear Danger Zones by authorities.
- Danger Zones are featured in the live map with unique warning signs.
- Duration, radius and severity level of Danger Zones is automatically calculated by the system based on: number of incidents, deserve that the frequency of incidents, verified report score.

5.2 Non-Functional Requirements

1. **Performance**
   - To guarantee quick reaction in an emergency, the system should be able to render maps and show fresh reports in less than a second.
2. **Scalability**
   - The backend and cloud infrastructure must be able to manage up to 1,000 concurrent submissions without experiencing system failure.
3. **Availability**
   - The system should have 99.9% uptime and provide 24/7 access for both the general public and authorities.
4. **Security**
   - Encryption is required for all client-server and database communications.
5. **Usability**
   - Even under pressure, users must be able to finish and submit a report using the mobile interface in less than 30 seconds. The dashboard ought to be responsive and easy to use across all of the main browsers.
6. **Reliability**
   - The system must use cloud-based redundancy to automatically recover from network or service outages without losing data.
7. **Maintainability**
   - For future scalability, debugging, and easy updates, the codebase must adhere to modular architecture.
8. **Portability**
   - With only minor platform-specific modifications, the app ought to function flawlessly on both iOS and Android smartphones.
9. **Data Integrity**
   - To guarantee consistency and avoid duplication or tampering, every report and verification record needs to be verified before being stored.
10. **Compliance**
    - When managing user locations and personal information, the system must abide by applicable data protection and privacy regulations.

5.3 Methodology

The Crowdsourced Disaster Response & Accident Reporting App will be developed using the Agile methodology. Agile places a strong emphasis on flexibility, iterative development, and tight coordination between emergency authorities, users, and developers. It enables the team to continuously develop, test, and enhance the system, guaranteeing that the finished product successfully satisfies real-time emergency response requirements.

1. Requirement Gathering and Analysis

Through meetings and conversations with stakeholders, such as administrators, police departments, ambulance services, and the general public, the system analyst and development team will collect specific requirements during this phase.

- Determine the main issues with the current systems for reporting and responding to disasters.
- Both functional and non-functional requirements should be documented.
- Create use cases and user stories for the public, admin, and authority user types.
- Create a product backlog that enumerates and ranks the features of the app, such as dashboard view, alerts, incident reporting, and verification.

2. Planning

- In brief iterative sprints, the team will plan how to deliver the identified features.
- Establish sprint objectives, assignments, and time frames.
- Assign team members front-end, back-end, dashboard, and cloud-related tasks.
- Determine which technologies and tools such as React Native, Node.js, React, AWS, and MongoDB are needed.
- Determine possible risks and solutions while estimating time and resources.

3. System Design

The app's technical framework and user interface will be created at this point.

- Design of the Architecture: Develop a modular and scalable architecture that demonstrates how the database, web dashboard, backend API, and mobile app interact.
- Database Design: To manage massive amounts of real-time data, create MongoDB collections like Users, Reports, Verifications, and DispatchLogs.
- UI/UX Design: To guarantee usability in an emergency, create responsive and user-friendly interfaces for the React authority dashboard and the mobile app.

4. Implementation

Multiple Agile sprints will be used for development, with a working module delivered at the end of each sprint. [6]

- In different sprints, implement the Dashboard, Alert Dispatch, Verification Engine, Incident Reporting, and Authentication.
- Connect APIs so that web and mobile apps can share data.
- Make use of AWS services for automated notifications (SNS), file storage (S3), and cloud hosting.
- After every iteration, conduct sprint reviews to collect feedback and improve features.

5. Testing

Continuous testing will be done at every stage of the development process.

- Unit testing: Confirm that every feature or function operates as intended.
- Integration testing: Examine how modules such as the dashboard and reporting interact with one another.
- System testing: Verify that the entire application functions as intended under real-time circumstances.
- User Acceptance Testing (UAT): Performed with a representative sample of authorities and users to confirm the usability and functionality of the system.

6. Deployment

The application will be launched on the AWS cloud environment following a successful testing phase.

- Users will receive the mobile application for practical testing.
- The dashboard will be linked to MongoDB Atlas and hosted on AWS Elastic Beanstalk.
- Tools for ongoing monitoring will be employed to guarantee system security, uptime, and performance.

7. Review and Feedback

The team will get input from stakeholders after a sprint or significant feature is released.

- Examine comments to find areas for improvement or new needs.
- Revise the backlog and make the necessary changes to the next sprint plan.
- To discuss lessons learned and enhance future procedures, hold sprint retrospectives.

8. Maintenance and Continuous Improvement

The team will enter a maintenance phase following complete deployment.

- Resolve issues after deployment and improve efficiency.
- Include new features like more disaster categories (landslides, fires, and floods).
- Continue to update the app to meet user needs and new technological advancements.

6 Modules Split-up and Gantt Chart References

6.1 Modules Split-up

There are eight primary functional modules that make up the system. To ensure correct integration and performance, each module is developed and tested in turn.

Table 07: Modules Split-up

Module ID	**Module Name**	**Responsibility Scope**
M1	Authentication & User Management	Backend logic, JWT security, role access, API endpoints
M2	Incident Reporting (Mobile Frontend)	UI design, mobile forms, API integration, usability
M3	Verification Engine (Logic)	Verification logic, scoring algorithms, status updates
M4	Live Map Dashboard (Web Frontend)	Map rendering, filters, admin controls
M5	Alert Dispatch System	Proximity logic, notification triggering, logging
M6	System Analysis & Design	Requirements, models, validation, documentation
M7	Quality Assurance & Testing	Test planning, bug tracking, system validation
M8	Project Planning & Coordination	Scheduling, documentation, risk & progress control
M9	Danger Zone Alert System	Risk scoring based on historical incident density and time patterns, danger zone heatmap generation, and preventive user alerts.

6.2 Gantt Chart

Figure 05: Gantt Chart for Project Development Timeline

	No. of weeks														
Module / Task	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
M8 – Project Planning & Coordination	🟩	🟩	🟩	🟩											
M6 – System Analysis & Design		🟩	🟩	🟩	🟩										
M1 – Authentication & User Management				🟩	🟩	🟩									
M2 – Incident Reporting (Mobile Frontend)					🟩	🟩	🟩	🟩							
M3 – Verification Engine (Logic)						🟩	🟩	🟩	🟩						
M4 – Live Map Dashboard (Web Frontend)							🟩	🟩	🟩	🟩					
M5 – Alert Dispatch System								🟩	🟩	🟩	🟩				
M9 – Danger Zone Alert System									🟩	🟩	🟩				
Integration of All Modules											🟩	🟩			
M7 – Quality Assurance & Testing												🟩	🟩	🟩	
Deployment & Configuration													🟩	🟩	
Documentation & Final Submission														🟩	🟩

7 Cost Analysis

Table 08: Cost Analysis

Category	Item / Description	Estimated Cost (LKR)	Remarks
1. Hardware	Laptop / PC (used for development)	0.00	Provided by students
	Smartphones (for testing)	0.00	Personal devices used
2.Software & Tools	Visual Studio Code, Node.js, React Native	0.00	Free / Open-source
	GitHub (Version Control)	0.00	Free for education
3.Cloud & Hosting Services	MongoDB Atlas (Free Tier)	0.00	Free for small-scale testing
	Firebase Cloud Messaging	0.00	Free tier used
	Render / AWS Hosting	1,500.00	Cloud hosting for demo
4. Domain & SSL	Domain name (.com / .lk)	3,000.00	Optional for deployment
5.Internet & Utilities	Internet, electricity, data	3,000.00	Shared group expenses
6. Miscellaneous	Documentation, printing, travel	2,000.00	For submission and review
Total Estimated Cost		**≈ LKR 9,500.00**	

8 Limitations and Future Enhancements

Limitations

The current version of ResQNow has several limitations:

- Requires a live internet connection; offline mode is not supported.
- Reduces accuracy in low-traffic areas by depending on user input for verification.
- Does not directly integrate with the government's emergency systems.
- Notifications via Firebase may experience delays.
- Restricted to reporting accidents rather than major calamities.

These are normal for a first prototype and will be enhanced in subsequent iterations.

Future Enhancements

ResQNow wants to develop into a comprehensive platform for disaster management by:

- Disaster Extension: Assistance in reporting fires, landslides, and floods.
- AI Prioritization: Automatic classification and identification of urgency.
- Government Integration: API connections with law enforcement, medical facilities, and emergency response organizations.
- Multilingual Support: Availability in a variety of geographical areas.
- Offline Mode: After going online, local reports are saved and synced.
- Analytics Dashboard: Planning and hotspot tracking data insights.
- Integrate machine learning prediction to analyze historical incident data and predict emerging Danger Zones before they form.

ResQNow will become a more potent, national emergency response system as a result of these improvements.

9 Conclusion

By offering a centralized platform for real-time reporting and coordination between the public and authorities, the proposed ResQNow – Crowdsourced Disaster Response & Accident Reporting App seeks to improve the effectiveness of emergency response. The system guarantees prompt alerting, precise location tracking, and improved resource allocation during accidents and disasters by utilizing web and mobile technologies coupled with cloud-based infrastructure.

By utilizing the MERN stack and React Native for scalability and cross-platform performance, the project exhibits technical viability. If ResQNow is implemented successfully, it can help create safer, quicker, and better-organized disaster response mechanisms and act as a basis for future emergency management systems at the national level.

References

[1] V. Padghan, "Incident Management in the Cloud Era: Challenges and Opportunities," *Squadcast Blog,* 25 October 2024.

[2] N. Foundation, "Node.js v25.0.0 Documentation," 2024. [Online]. Available: https://nodejs.org/en/docs.

[3] MongoDB, "MongoDB Atlas Documentation," MongoDB Inc, 2025. [Online]. Available: https://www.mongodb.com/docs/atlas/.

[4] M. P. Inc., "React Native - Get Started with React Native," Meta Platforms Inc., 2025. [Online]. Available: https://reactnative.dev/docs/environment-setup.

[5] "Firebase Cloud Messaging," Firebase, 2024. [Online]. Available: https://firebase.google.com/docs/cloud-messaging.

[6] E. Diallo and R. Abdallah and M. Dib and O. Dib, "Decentralized Incident Reporting: Mobilizing Urban Communities with Blockchain," *Smart Cities,* vol. vii, no. 4, pp. 2283-2317, 14 August 2024.