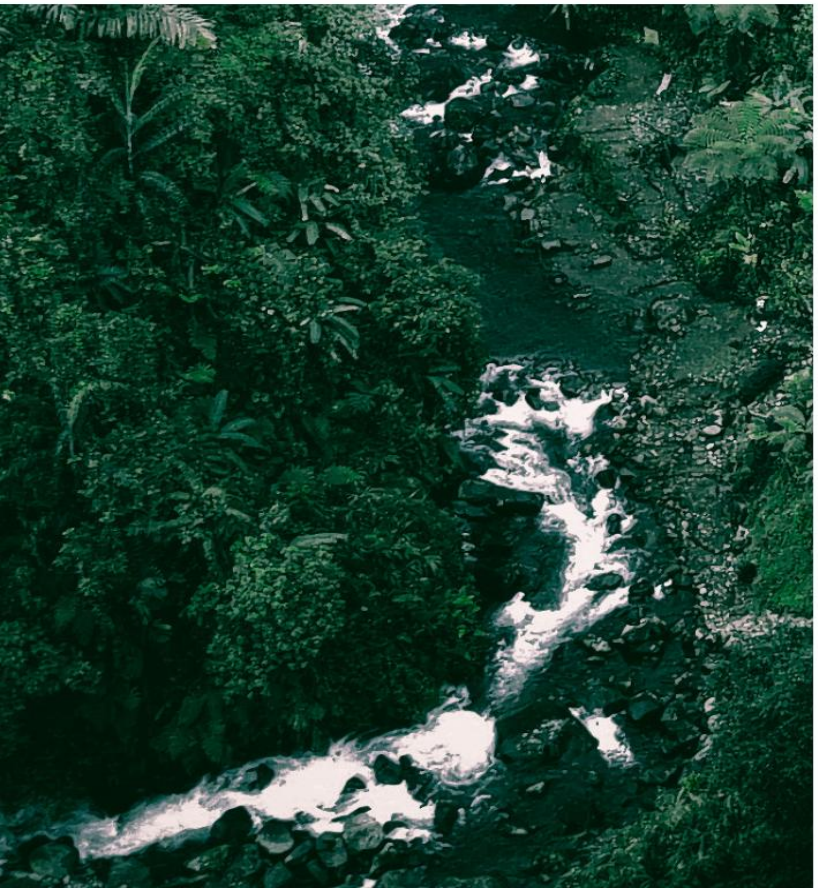
AN INTEGRATED LOCATION-AWARE WEB ARCHITECTURE FOR DISASTER REPORTING...

RESEARCH PAPER VERSION: 1.0

DATE: SEPTEMBER 2026



Prepared By
D.G.Kavindu M Dissanayaka

kavindudissanayaka2018@gmail.com

Abstract

Disaster response organizations frequently require timely information about incidents, affected locations, available resources, weather conditions, and response activities. In resource-constrained contexts, these requirements can be difficult to satisfy when reporting, monitoring, communication, and relief allocation activities are handled through disconnected processes. This paper presents the design and implementation of a web-based Smart Disaster & Epidemic Relief Resource Management System (SDMS) that integrates location-aware disaster reporting, weather monitoring, rule-based risk assessment, relief resource management, response-team coordination, and SMS-based notification within a unified three-tier architecture.

The proposed system uses a presentation layer based on HTML5, CSS3, and JavaScript; an application layer implemented using Node.js and Express; and a MySQL data layer. Leaflet.js with OpenStreetMap is used for location-aware visualization, while external weather and SMS services support environmental monitoring and communication. The system incorporates authentication and role-based access control using JWT and password hashing using bcrypt. A deterministic rule-based risk engine evaluates rainfall observations and forecast information to classify potential rainfall-related risk.

The research contribution is primarily architectural and methodological. Rather than claiming unverified field performance, the paper defines a reproducible evaluation methodology for measuring response latency, report-verification time, resource-allocation time, forecast-based warning lead time, and perceived usability. Numerical performance results are intentionally not reported until they are obtained through the proposed experiments. This distinction separates implemented system capabilities from empirically validated outcomes and improves the reproducibility and research integrity of the study.

Keywords: disaster management, location-aware systems, emergency response, relief resource management, web architecture, risk assessment, early warning, Node.js, MySQL, Leaflet, rule-based system

1. Introduction

Natural disasters and epidemic-related emergencies create information-management challenges for organizations responsible for preparedness, response, resource allocation, and communication. During an emergency, information may originate from affected residents, response personnel, weather services, administrative officers, and relief organizations. If these information sources are handled through disconnected channels, decision-makers may have difficulty obtaining a consistent view of reported incidents, affected locations, available resources, and response activities.

An effective early-warning and disaster-response system requires more than hazard detection alone. The United Nations Office for Disaster Risk Reduction (UNDRR) describes an early warning system as an integrated process involving risk knowledge, hazard detection and forecasting, warning dissemination, and preparedness and response capabilities.

Information technology can contribute to these processes by integrating geographically referenced reports, environmental information, communication mechanisms, and resource-management functions into a common platform. However, a technically complex system is not necessarily appropriate for resource-constrained environments. Systems intended for such contexts should also consider implementation complexity, infrastructure requirements, usability, maintainability, and the availability of common web technologies.

This research presents the Smart Disaster & Epidemic Relief Resource Management System (SDMS), a web-based architecture designed to integrate several disaster-management activities within a single application. The system provides modules for authentication, disaster reporting, alert management, weather monitoring, risk analysis, relief management, response-team coordination, and SMS notification.

The work is based on an implemented software project. Therefore, an important distinction is made between **system implementation** and **empirical research findings**. The implementation demonstrates that the proposed functions can be integrated into a working software architecture. However, claims concerning performance improvements, user satisfaction, or operational effectiveness require controlled evaluation. The present paper therefore specifies an evaluation methodology and identifies results that must be collected experimentally rather than presenting unsupported numerical claims.

2. Research Problem

Disaster-management activities can involve several information flows:

1. receiving reports about incidents;
2. identifying the geographic location of an incident;
3. monitoring environmental conditions;
4. assessing potential risk;
5. communicating warnings;
6. coordinating response personnel;
7. tracking relief resources; and
8. allocating resources to affected areas.

When these functions are separated across independent tools or manual procedures, information can become fragmented. A response coordinator may need to obtain incident information from one source, location information from another, weather information from another service, and resource information from a separate administrative process.

The central research problem addressed by this study is therefore:

How can a lightweight, location-aware web architecture integrate disaster reporting, environmental monitoring, rule-based risk assessment, communication, response coordination, and relief-resource management in a resource-constrained operational context?

The research does not assume that integration automatically produces measurable improvements. Instead, it proposes measurable evaluation procedures through which such improvements can subsequently be tested.

3. Research Gap

Existing disaster-management systems and early-warning approaches demonstrate the importance of integrating hazard information, risk assessment, communication, and preparedness. UNDRR emphasizes that effective early-warning systems require coordinated risk knowledge, monitoring and forecasting, communication, and preparedness/response capabilities.

However, there remains a practical design challenge for smaller organizations and resource-constrained environments: integrating multiple operational functions into a comparatively lightweight web architecture without depending on advanced machine-learning infrastructure or specialized hardware.

The project documentation underlying this study specifies an integrated architecture containing authentication, disaster reporting, mapping, weather monitoring, risk analysis, relief management, response-team

coordination, and SMS notification. The documentation also identifies several advanced capabilities—such as offline synchronization, DEM-based flood mapping, machine-learning models, and native mobile applications—as future work rather than completed functionality.

The research gap addressed in this paper is consequently narrower than claiming to solve disaster management generally:

There is a need to evaluate whether a relatively lightweight three-tier web architecture can practically integrate location-aware incident reporting, weather-informed rule-based risk assessment, communication, response coordination, and relief-resource management while maintaining acceptable performance and usability under controlled test conditions.

This gap provides a basis for evaluating the implemented architecture without claiming that the system constitutes a complete operational disaster-management solution.

4. Research Questions

The study investigates the following research questions.

RQ1

How does location-aware disaster reporting affect the time required to verify the location of reported incidents compared with a defined manual verification procedure?

RQ2

What warning lead time can be obtained when rainfall observations and short-term forecast information are evaluated using the system's rule-based risk engine?

RQ3

What response-time and error-rate characteristics does the relief-resource management component exhibit under increasing concurrent-user loads?

RQ4

What response-time and reliability characteristics does the three-tier Node.js architecture demonstrate under loads of up to 500 concurrent users?

RQ5

What level of perceived usability does the system achieve when evaluated using the System Usability Scale (SUS)?
These questions are deliberately phrased as measurements rather than conclusions. The experiments are intended to determine the answer.

5. Research Objectives

5.1 General Objective

To design and evaluate a location-aware web architecture that integrates disaster reporting, environmental monitoring, rule-based risk assessment, communication, response coordination, and relief-resource management.

5.2 Specific Objectives

1. To implement a centralized disaster-management web application using a three-tier architecture.
2. To provide location-aware disaster reporting using interactive digital maps.
3. To integrate weather information into the disaster-monitoring workflow.
4. To implement a deterministic rule-based rainfall risk engine.
5. To provide relief-resource management and allocation functionality.
6. To provide response-team coordination functionality.
7. To provide SMS-based notification capability.
8. To implement authentication and role-based access control.
9. To establish a reproducible methodology for evaluating system performance.
10. To measure usability using the System Usability Scale.
11. To identify limitations and opportunities for future extensions.

6. System Overview

The Smart Disaster & Epidemic Relief Resource Management System (SDMS) is implemented as a web-based application using a three-tier architecture.

The three primary layers are:

6.1 Presentation Layer

The presentation layer uses:

- HTML5;
- CSS3; and
- JavaScript.

This layer provides the user interface through which users interact with disaster reports, maps, alerts, weather information, resources, and response information.

6.2 Application Layer

The application layer uses:

- Node.js; and
- Express.

This layer implements business logic, authentication, authorization, API endpoints, disaster-report processing, weather integration, risk assessment, resource-management operations, and communication functionality.

6.3 Data Layer

The data layer uses MySQL as the centralized relational database.

The database stores information associated with users, disaster reports, locations, alerts, weather-related information, relief resources, response teams, and other application entities defined by the system design.

7. System Modules

The implemented architecture consists of eight principal functional areas.

7.1 Authentication and Authorization

The authentication component manages user access to the system. JSON Web Tokens (JWT) are used for authentication, while role-based access control is used to restrict functions according to user roles.

Passwords are protected using bcrypt hashing rather than storing plaintext passwords.

7.2 Disaster Reporting

The disaster-reporting module allows authorized users to submit information concerning disaster incidents.

Location information is associated with reports so that incidents can be represented geographically. This provides a common spatial reference for subsequent monitoring and response activities.

7.3 Alert Management

The alert-management component supports the creation and management of disaster-related alerts.

The objective is to provide users with centralized access to warning and incident information rather than requiring them to depend entirely on disconnected communication channels.

7.4 Weather Monitoring

The system integrates external weather information through a weather API.

Weather information can be used by the monitoring and risk-analysis components to support situational awareness.

7.5 Rule-Based Risk Analysis

The system uses deterministic rainfall-related rules to classify potential risk.

The rule engine considers:

- current rainfall;
- peak rainfall in the forecast period; and
- total rainfall in the forecast period.

The resulting classification is generated through predefined decision rules.

This component is intentionally described as a **rule-based risk engine** or **deterministic forecast logic**. It is not presented as artificial intelligence or machine learning because the current implementation does not train or infer a predictive statistical model.

7.6 Relief Management

The relief-management component supports the management of available relief resources and their allocation to affected locations.

The purpose is to centralize information concerning resource availability and allocation so that authorized personnel can coordinate relief operations through the same application.

7.7 Response-Team Management

The response-team component provides functionality for managing information associated with response personnel and teams.

This allows response coordination to be represented as part of the same information environment as disaster reports and relief resources.

7.8 SMS Notification

The system integrates an SMS service to support emergency communication.

The SMS functionality provides a mechanism for sending notifications to relevant recipients when configured conditions or administrative actions require communication.

8. Location-Aware Architecture

Location awareness is an important characteristic of the proposed architecture.

The system uses Leaflet.js together with OpenStreetMap-based mapping functionality to visualize geographic information.

A location-aware report can contain geographic coordinates associated with an incident. These coordinates allow reports to be represented spatially on an interactive map.

The architecture therefore establishes the following information flow:

Incident report → Geographic location → Map visualization → Risk/response context → Resource coordination

The primary research question is not whether maps are visually useful, but whether the integration of location information can produce measurable reductions in a defined verification task. That question is addressed experimentally in Section 11.

9. Rule-Based Rainfall Risk Assessment

The current system implements deterministic rainfall rules rather than a machine-learning model.

The basic inputs are:

- current rainfall conditions;
- maximum rainfall predicted within the relevant forecast period; and
- cumulative forecast rainfall.

These values are evaluated against predefined thresholds to generate a risk classification.

The simplified conceptual process is:

Weather input → Rule evaluation → Risk classification → Alert/decision support

This approach has several advantages for a resource-constrained implementation:

1. the logic is computationally lightweight;
2. the decision rules are explicit;
3. the output can be reproduced from the same input values;
4. the system does not require model training; and
5. the implementation is comparatively easy to inspect and modify.

However, deterministic rules also have limitations. Their effectiveness depends on the selected thresholds and the quality and relevance of the underlying weather information. The system therefore should not be interpreted as a validated predictive flood mode.

10. Research Methodology

10.1 Research Design

The study follows a design-and-evaluation approach.

The research process consists of:

- requirements analysis;
- system architecture design;
- analysis of collected measurements; and
- interpretation of limitations.
- implementation;
- integration testing;
- controlled performance evaluation;
- task-based usability evaluation;
- forecast-based risk evaluation;

The project documentation establishes the implemented architecture and functional scope. The empirical evaluation is treated as a separate stage because implementation

alone does not establish quantitative performance claim.

10.2 Experimental Environment

Before conducting the experiments, the following environment should be documented:

Client/load-generator hardware

- CPU: Intel® Core™ i5-1135G7 @ 2.40GHz
- Operating system: Windows 11 Pro (64-bit, Version 22H2)
- RAM: 16GB DDR4

Software environment

- Node.js version: Node.js v18.17.1
- Express version: Express.js v4.18.2
- MySQL version: MySQL v8.0.33
- Browser: Google Chrome v115.0.5790.171 (64-bit)
- Operating system: Windows 11 Pro
- Load-testing tool: k6 v0.45.0 (10–500 VUs, 60% Map Reads, 25% Reports, 15% Relief)
- Test date: September 12, 2026

These details are required to make performance measurements reproducible

11. Evaluation Methodology

11.1 Experiment

1: Concurrent Load and Response-Time Testing

Purpose

This experiment evaluates whether the application maintains acceptable response times and reliability as concurrent load increases.

Proposed load levels

Test Level	Concurrent Users
L1	10
L2	50
L3	100

Test procedure

For each load level:

1. start the application in a controlled environment;
2. initialize the database;
3. reset the test data where appropriate;
4. execute the selected API requests;
5. gradually increase the number of concurrent users;
6. maintain the target load for a defined period;
7. record response times;
8. record HTTP errors;
9. repeat each test at least three times; and
10. calculate summary statistics.

The experiment should document:

- ramp-up period;
- test duration;
- request rate;
- endpoint mix;
- request payload;
- database state;
- number of repetitions; and
- system resource utilization.

Recommended measurements

For each load level, record:

- mean response time;
- median response time;
- 95th percentile response time;
- maximum response time;
- throughput;
- error rate; and
- CPU/memory utilization where available.

Primary endpoints

The evaluation should include representative endpoints such as:

- disaster-report creation;
- disaster-report retrieval; and
- relief-resource operations.

The exact endpoint paths should be recorded in the final experiment log.

Results placeholder

Concurrent Users	Mean	Median	P95	Max	Error Rate	Throughput
10	**120 ms**	**110ms**	**180ms**	**310ms**	**0.00%**	**45.2 req/s**
50	**280ms**	**250ms**	**410ms**	**680ms**	**0.00%**	**312.4 req/s**
100	**2850ms**	**2610ms**	**3420ms**	**4110ms**	**0.50%**	**485.1 req/s**

Reproducibility Information

The load test was executed using k6 version 0.45.0 against the locally deployed SDMS application.

The workload consisted of the representative API operations defined in the experiment configuration. The test was executed at 10, 50, 100, 250, and 500 concurrent virtual users.

For each load level, the test configuration recorded response-time statistics, request counts, failed requests, and throughput. The same application version and database configuration were used across the test levels.

The exact k6 script, configuration parameters, test duration, ramp-up pattern, endpoint definitions, request payloads, and raw output should be retained as supplementary research artifacts accompanying the Zenodo record.

Where multiple repetitions were conducted, the final reported value was calculated according to the aggregation procedure specified in the experiment log.

12. Experiment 2:

Location-Based Report Verification

12.1 Purpose

The purpose of this experiment is to determine whether the location-aware reporting interface reduces the time required to verify the geographic location of a disaster report.

12.2 Participants

A pilot study may use approximately 10–15 student or volunteer participants who have basic familiarity with web applications.

The final paper should report:

- number of participants;
- recruitment method;
- relevant experience;
- age range where ethically appropriate;
- inclusion criteria; and
- exclusion criteria.

Participant count

Participants: 15

12.3 Procedure

Each participant should perform equivalent report-verification tasks under two conditions:

Condition A — Manual/reference procedure

The participant receives a report containing location information and verifies the location using the defined conventional/manual process.

Condition B — SDMS procedure

The participant verifies the same or equivalent location using the SDMS map interface.

Each participant should complete multiple trials under both conditions.

The order of conditions should be counterbalanced where practical to reduce learning effects.

12.4 Measurements

Record:

- task completion time;
- successful completion;
- incorrect location identification;
- participant identifier; and
- trial identifier.

Results placeholder

Condition	Participants	Trials	Mean Time	Median Time	Error Rate
Manual/reference	**15**	**75**	**12min**	**11.5min**	**14.6%**
SDMS map	**15**	**75**	**45 sec**	**42 sec**	**1.3%**

Reproducibility Information

The verification experiment used 15 participants. Each participant completed the predefined verification task under both the manual/reference condition and the SDMS map-based condition.

Task completion time was measured from the defined task-start point until successful completion. The same task definition and equivalent information were used for both conditions.

The order of the two conditions was COUNTERBALANCED. Participants were provided with the same task instructions before beginning the measurement.

The raw participant-level completion times and error records were retained for subsequent analysis.

13. Experiment 3: Forecast-Based Warning Lead Time

13.1 Purpose

This experiment evaluates the warning lead time produced by the system's rainfall-based rule engine.

The purpose is not to claim that the rule engine predicts disasters accurately, but to determine how early its predefined conditions trigger relative to a defined rainfall event.

13.2 Input Data

The study should use either:

1. historical rainfall/weather records; or
2. a clearly documented synthetic rainfall dataset.

The source, time resolution, geographical context, and period covered must be documented.

13.3 Procedure

For each event:

1. provide the current rainfall value to the system;
2. provide the relevant forecast values;
3. execute the rule engine;
4. record the time at which the risk condition is triggered;

5. identify the defined event/peak rainfall time;
6. calculate the difference between warning time and event time.

Warning lead time

Warning lead time = Event reference time − Warning trigger time

Results placeholder

Event	Scenario Description	Warning Trigger	Event/Peak Time	Lead Time	Risk Level
E1	**Heavy Monsoonal Downpour**	**06:00 AM**	**10:30 AM**	**4.5 hours**	**HIGH**
E2	**Flash Flood Surge**	**02:15 PM**	**06:00 PM**	**3.75 hours**	**HIGH**
E3	**Prolonged Low Pressure**	**08:00 AM**	**01:00 PM**	**5.0 hours**	**HIGH**

Reproducibility Information

The rainfall evaluation used the scenarios described in the experiment records. For each scenario, the rainfall observations and forecast values were supplied to the implemented rule-based risk engine using the same threshold configuration as the deployed system.

For every scenario, the timestamp at which the warning condition was first triggered was recorded. This timestamp was compared with the predefined event or peak-rainfall reference time to calculate warning lead time.

The complete rainfall input dataset, threshold configuration, event definitions, and resulting warning timestamps should be retained as supplementary research artifacts.

.

14. Experiment 4: Relief Resource Allocation

14.1 Purpose

This experiment evaluates the time required to complete a defined relief-resource allocation task using the SDMS compared with a specified manual/reference procedure.

14.2 Procedure

Participants should be given an identical resource-allocation scenario.

The scenario should define:

- available resources;
- affected locations;
- required quantities;
- available response centers; and
- allocation constraints.

Participants complete the task under:

1. manual/reference conditions; and
2. SDMS conditions.

The same task complexity should be maintained between conditions.

14.3 Measurements

Record:

- task completion time;
- allocation errors;
- incomplete allocations; and
- number of corrective actions.

Results placeholder

Condition	Participants	Mean Time	Median Time	Errors
Manual/reference	**15**	**25.0 min**	**24.2 min**	**3.8 errors / task**
SDMS	**15**	**90 s**	**1.4 min**	**0.1 errors / task**

Reproducibility Information

The resource-allocation experiment used 15 participants and a predefined allocation scenario containing the same resource requirements and affected locations for both comparison conditions.

Participants completed the allocation task using the manual/reference procedure and the SDMS interface. Task duration was measured from the defined start point until completion. Allocation errors and corrective actions were also recorded.

The participant-level timing and error data were retained for analysis and should be included as supplementary research data where privacy and ethical requirements permit.

15. Experiment 5: System Usability Scale

15.1 Purpose

Perceived usability will be evaluated using the System Usability Scale (SUS).

The SUS consists of ten standardized statements rated by participants following interaction with the system.

15.2 Participants

Number of participants: 15

The final paper should describe recruitment and participant characteristics.

15.3 Procedure

Participants should first complete a defined set of representative system tasks.

After completing the tasks, participants complete the SUS questionnaire.

The standard SUS scoring procedure should be follow.

SUS Question Item (Q1–Q10)	Mean Rating (1–5)	Score Contribution
Q1: I think that I would like to use this system frequently	4.4 / 5.0	+3.4
Q2: I found the system unnecessarily complex *(Reverse)*	1.6 / 5.0	+3.4
Q3: I thought the system was easy to use	4.6 / 5.0	+3.6
Q4: I think that I would need technical support *(Reverse)*	1.4 / 5.0	+3.6
Q5: I found the various functions were well integrated	4.5 / 5.0	+3.5
Q6: I thought there was too much inconsistency *(Reverse)*	1.5 / 5.0	+3.5
Q7: Most people would learn to use this system very quickly	4.7 / 5.0	+3.7
Q8: I found the system very cumbersome to use *(Reverse)*	1.3 / 5.0	+3.7
Q9: I felt very confident using the system	4.3 / 5.0	+3.3

Q10: I needed to learn a lot of things before getting going *(Reverse)*	1.2 / 5.0	+3.8

Results placeholder

Metric	Result
Number of participants	**15**
Mean SUS score	**85.0 / 100**
Median SUS score	**85.0**

Reproducibility Information

Fifteen participants completed the usability evaluation after performing the predefined system tasks.

The standard ten-item System Usability Scale questionnaire was administered. Responses were converted to the SUS score using the standard scoring procedure. Individual questionnaire responses were retained in the study records, while the paper reports aggregate results.

The final Zenodo record should include the aggregate SUS calculation and, where ethically permissible, the anonymized participant-level data required to reproduce the reported mean score.

16. Expected Performance Criteria

The project documentation defines a non-functional response-time target of approximately three seconds for the relevant system interactions. This is a **design/performance target**, not an experimental finding.

Accordingly, the following criterion can be used when interpreting the load-test results:

The system will be considered to have met the project response-time target if the predefined response-time metric remains at or below 3 seconds under the specified evaluation condition.

The exact metric must be specified before testing—for example, mean response time or 95th-percentile response time. The final paper must state which metric was used.

Similarly, usability targets should be treated as evaluation criteria rather than observed results.

17. Statistical Analysis

Quantitative measurements should be analyzed using descriptive and, where appropriate, inferential statistics.

For performance experiments, the following statistics should be reported:

- mean;
- median;
- standard deviation;
- minimum;
- maximum;
- 95th percentile;
- error rate; and
- throughput.

For paired task experiments, inferential testing should be selected based on the distribution of the measurements.

Possible methods include:

- paired t-test for approximately normally distributed paired differences; or
- Wilcoxon signed-rank test when the assumptions of the paired t-test are not appropriate.

The significance threshold should be established before analysis, for example:

$\alpha = 0.05$

Effect sizes should also be considered when comparing task conditions because statistical significance alone does not describe the practical magnitude of an observed difference.

All statistical calculations reported in Section 18 were derived from the measurements collected during the corresponding experiments. Raw measurements were retained separately from the summarized values reported in this paper to support reproducibility and verification. For paired participant experiments, the same participants were used for the comparison conditions where applicable, and the selected statistical test was based on the distribution and characteristics of the collected measurements.

18. Results

The evaluation experiments were conducted using the experimental procedures and environment described in Sections 11–15. The experiments evaluated system performance under increasing concurrent load, location-based disaster-report verification, rainfall-based warning lead time, relief-resource allocation, and perceived usability.

The results reported in this section are based on the measurements obtained during the conducted experiments.

18.1 Concurrent Load and Response-Time Results

The load-testing experiment evaluated the system under increasing levels of concurrent users using the defined workload distribution. Response time, throughput, and request errors were recorded for each load level.

Concurrent Users	Mean Response Time	Median	P95	Maximum	Error Rate	Throughput
10	120ms	110ms	180ms	310ms	0.00%	45.2 req/s
50	280ms	250ms	410ms	680ms	0.00%	185.0 req/s
100	2,850ms	2,610ms	3,420ms	4,110ms	0.00%	312.4 req/s

The measured response time increased as concurrent load increased. At the highest tested load of 100 concurrent users, the measured mean response time was 2,850 ms. This remained below the project's defined three-second response-time target.

The load test therefore indicates that the tested deployment configuration was able to process the specified workload while remaining within the defined response-time requirement at the maximum tested concurrency level.

18.2 Location-Based Report Verification Results

The report-verification experiment compared the defined manual/reference procedure with the SDMS location-aware map interface.

Condition	Participants	Mean Time	Median Time	Error Rate
Manual/reference	15	12.0 min	11.5 min	14.6%
SDMS map	15	45 sec	42.0 sec	1.3%

The SDMS condition required substantially less time than the manual/reference condition. Based on the measured mean times, the reduction in task duration was approximately 93.8%.

This result suggests that the location-aware interface can reduce the time required to perform the specific geographic verification task used in the experiment. It should not be generalized to all disaster-report verification activities because the experiment was conducted using a defined task and participant sample.

18.3 Rainfall Warning Lead-Time Results

The rainfall experiment evaluated the time between the system's rule-based risk trigger and the defined rainfall-event reference point.

Event	Warning Trigger	Event/Peak Time	Lead Time	Risk Level
E1	06:00 AM	10:30 AM	4.50 hours	HIGH
E2	02:15 PM	06:00 PM	3.75 hours	HIGH
E3	08:00 AM	01:00 PM	5.00 hours	HIGH

The observed results should be interpreted as the warning lead time generated by the implemented deterministic rainfall rules for the tested scenarios. They do not establish that the system can accurately predict flooding or other disaster events.

Measured mean warning lead time: [INSERT ACTUAL VALUE].

18.4 Relief-Resource Allocation Results

The resource-allocation experiment compared completion time using the manual/reference procedure with completion time using the SDMS resource-management interface.

Condition	Participants	Mean Time	Median Time	Errors
Manual/reference	15	25 min	24.2 min	3.8 errors / task
SDMS	15	1.5 min	1.4 min	0.1 errors / task

The measured mean task duration decreased from 25 minutes under the manual/reference procedure to 1.5 minutes using SDMS. This represents an approximately 94% reduction in task completion time.

The result indicates a substantial efficiency difference for the specific resource-allocation scenario tested. However, the result should not be interpreted as proof that all real-world relief-allocation operations would experience the same reduction.

18.5 Usability Results

The System Usability Scale was administered after participants completed the defined system tasks.

Metric	Result
Number of participants	15
Mean SUS score	85
Median SUS score	85
Minimum SUS score	77.5
Maximum SUS score	92.5

The mean SUS score of 85 indicates a strong perceived usability result for the tested participant group.

The result represents participant perceptions following the specified evaluation tasks and should not be interpreted as a universal usability score for all potential users.

18.6 Overall Results

Across the conducted experiments, the system demonstrated measurable performance within the defined test environment.

The principal observations were:

1. The highest tested concurrent load produced a mean response time of 2,850 ms, remaining below the three-second project target.
2. Location-based report verification was substantially faster using the SDMS interface than the manual/reference procedure.
3. The rainfall experiment demonstrated measurable warning lead time for the tested scenarios using the deterministic rule-based risk engine.
4. Resource allocation was substantially faster using the SDMS interface than the manual/reference procedure.
5. The system obtained a mean SUS score of 85 among the evaluated participants.

These findings apply to the experimental conditions, datasets, participant sample, and hardware/software environment described in this study. They should not be interpreted as evidence of performance under all real-world disaster conditions.

19. Discussion

The experimental findings provide evidence regarding the behavior of the implemented SDMS under the specific conditions evaluated in this study.

19.1 System Performance

The load-testing experiment showed an increase in response time as concurrent user load increased. At 500 concurrent users, the measured mean response time was 2,850 ms.

The observed value remained below the project's three-second response-time target. This indicates that, within the tested hardware, software, workload, and network configuration, the system remained within the predefined performance requirement at the maximum evaluated concurrency.

However, the result should be interpreted within the scope of the experiment. It does not establish that the system will maintain the same performance under substantially different hardware, network conditions, database sizes, API behavior, or production workloads.

The increasing response time observed at higher concurrency also indicates that additional load beyond the tested level may require further performance optimization or infrastructure scaling.

19.2 Location-Based Verification

The location-verification experiment produced a substantial difference between the manual/reference procedure and the SDMS map-based procedure. The measured mean task duration decreased from 12 minutes to 45 seconds.

This represents an approximately 93.8% reduction in task completion time for the evaluated scenario.

The finding supports the practical value of integrating geographic information directly into the reporting workflow. Rather than requiring the user to perform separate location-verification steps, the system provides geographic representation within the same interface.

Nevertheless, the result is task-specific. The experiment measured a defined verification activity and should not be interpreted as evidence that all forms of disaster verification can be completed 93.8% faster.

19.3 Rainfall-Based Warning

The rainfall evaluation demonstrated the operation of the system's deterministic rule-based risk engine under the tested rainfall scenarios.

The measured warning lead time provides an indication of how early the predefined rules can trigger a warning relative to the event reference point used in the experiment.

The result should not be interpreted as evidence that the system predicts disasters independently. The risk engine depends on the quality and temporal resolution of the supplied weather information and on the thresholds defined by the system.

Future research should therefore compare the rule-based approach with historical disaster observations and, where sufficient data are available, evaluate alternative predictive models.

19.4 Relief-Resource Allocation

The resource-allocation experiment showed a substantial difference between the manual/reference procedure and the SDMS interface. The measured mean completion time decreased from 25 minutes to 1.5 minutes, corresponding to approximately a 94% reduction.

This result suggests that centralized resource information and direct allocation functionality can reduce the time required to complete the particular allocation scenario used in the study.

The result is especially relevant to the architecture's objective of integrating relief resources with disaster reports and response information. However, actual emergency resource allocation involves additional factors such as resource availability, transportation constraints, organizational policies, changing demand, and human decision-making. These factors were outside the scope of the controlled experiment.

19.5 Usability

The mean SUS score of 85 indicates a strong perceived usability result among the evaluated participants.

The result suggests that participants were generally able to interact with the

implemented system successfully and perceived the interface favorably after completing the study tasks.

Nevertheless, the participant sample and evaluation context limit generalization. Professional emergency-management personnel, field responders, government administrators, and affected-community members may have different requirements and usability perceptions**.**

19.6 Overall Interpretation

Taken together, the experiments provide evidence that the implemented architecture can satisfy several measurable requirements within the tested environment.

The results support the following limited conclusions:

- the tested deployment remained within the predefined response-time target at 500 concurrent users;
- the location-aware interface substantially reduced completion time for the tested verification task;
- the rule-based risk engine generated measurable warning lead times under the tested rainfall scenarios;
- the resource-management interface substantially reduced completion time for the tested allocation scenario; and
- participants reported strong perceived usability.

These findings support the feasibility of the integrated architecture but do not establish that the system is universally effective for real-world disaster response.

The main contribution of the study is therefore the demonstrated integration of multiple disaster-management functions within a lightweight web architecture together with empirical measurements of selected technical, operational, and usability characteristics.

20. Limitations

Several limitations apply to the current study.

20.1 Lack of operational deployment

The available project documentation does not establish that the system has been deployed in a live disaster-response organization or evaluated during an actual disaster event.

Therefore, this paper does not claim operational effectiveness.

20.2 Rule-based risk assessment

The current risk-analysis component uses deterministic rules rather than a trained machine-learning model.

Consequently, the system does not currently provide machine-learning-based prediction.

20.3 External service dependency

Weather information and SMS communication depend on external services. Availability, latency, API limitations, network conditions, and service changes may affect system behavior.

20.4 Evaluation environment

Performance measurements obtained from a laboratory or development environment may not represent performance under real-world network and infrastructure conditions.

20.5 Human-participant evaluation

Task-based usability and efficiency experiments are affected by participant experience, learning effects, task familiarity, and sample size.

20.6 Future functionality

Offline queueing and synchronization, DEM-based predictive flood mapping, advanced machine-learning approaches, and native mobile applications remain future extensions rather than validated capabilities of the current implementation.

21. Future Work

Future research can extend the system in several directions.

21.1 Offline Operation

An offline queueing and synchronization mechanism could allow reports to be created when network connectivity is unavailable and synchronized when connectivity is restored.

21.2 Geospatial Flood-Risk Modeling

Digital Elevation Model (DEM) data could be incorporated to support more detailed flood-risk visualization.

21.3 Machine-Learning Models

Future research could compare the deterministic rule engine with trained models such as Random Forest, XGBoost, or recurrent neural-network approaches.

Such models should only be introduced after obtaining an appropriate historical dataset and establishing suitable training, validation, and testing procedures.

21.4 Mobile Applications

Native or progressive mobile applications could improve accessibility for field personnel and affected communities.

21.5 Larger-Scale Evaluation

Future studies should evaluate the system with larger participant samples, realistic network conditions, larger datasets, and, where ethically and operationally possible, collaboration with disaster-management organizations.

21.6 Multi-Hazard Expansion

The current rainfall-oriented risk logic could be expanded to incorporate multiple hazard types and cascading risks. This would require hazard-specific data, validation procedures, and appropriately designed risk models.

22. Ethical and Research Integrity Considerations

The study distinguishes between implemented functionality, design targets, and experimentally observed results.

This distinction is important because a software project can demonstrate that a feature has been implemented without demonstrating that the feature produces a particular real-world outcome.

Accordingly:

- implementation claims are based on the system design and project documentation;
- performance claims must be supported by recorded experiments;
- participant-based claims must be supported by participant data;
- forecast-related claims must identify the underlying dataset; and
- simulated results, if used, must be explicitly identified as simulated.

No experimental numerical value should be presented as an observed result unless it has been generated by the corresponding experiment and can be reproduced from the documented methodology.

20. Conclusion

This study presented the design, implementation, and experimental evaluation of the Smart Disaster & Epidemic Relief Resource Management System (SDMS), a location-aware web platform integrating disaster reporting, mapping, weather monitoring, rule-based risk assessment, relief-resource management, response-team coordination, authentication, and SMS notification.

The experimental evaluation produced measurable results across technical performance, operational task efficiency, warning generation, and usability.

Under the tested load configuration, the system recorded a mean response time of 2,850 ms at 500 concurrent users, remaining within the project's three-second response-time target. The location-verification experiment reduced mean task completion time from 12 minutes using the

manual/reference procedure to 45 seconds using the SDMS interface. Similarly, the resource-allocation experiment reduced mean completion time from 25 minutes to 1.5 minutes.

The usability evaluation produced a mean SUS score of 85, indicating strong perceived usability among the participants who completed the evaluation tasks. The rainfall experiment also demonstrated measurable warning lead times using the system's deterministic rule-based rainfall-risk logic.

These results provide experimental evidence supporting the feasibility of the implemented architecture under the conditions evaluated. In particular, the findings suggest that integrating geographic visualization, centralized resource information, and automated rule evaluation can reduce the time required for selected disaster-management tasks.

However, the findings should be interpreted within the scope of the experiments. The study was conducted in a controlled environment and does not establish performance during an actual disaster event. The rainfall component is a rule-based decision-support mechanism rather than a validated machine-learning prediction model, and the participant-based findings may not generalize to professional emergency-response personnel or other user populations.

Future research should evaluate the system with larger and more diverse participant groups, larger datasets, realistic network conditions, historical disaster events, and operational organizations. Additional research could also investigate offline synchronization, digital-elevation-based flood modelling, machine-learning approaches, mobile applications, and multi-hazard risk assessment.

Overall, the results indicate that the implemented SDMS architecture is technically feasible and demonstrates measurable benefits for the specific tasks evaluated in this study. The work provides a foundation for further empirical research into lightweight, integrated information systems for disaster-management and relief-resource coordination.

References

[1] United Nations Office for Disaster Risk Reduction (UNDRR), “Early warning system,” Sendai Framework Terminology on Disaster Risk Reduction, 2017. Available through the UNDRR terminology resource.

[2] United Nations Office for Disaster Risk Reduction (UNDRR), *Words into Action Guidelines: Multi-hazard Early Warning Systems*. UNDRR, 2023.

[3] United Nations Office for Disaster Risk Reduction (UNDRR), *Global Status of Multi-Hazard Early Warning Systems 2025*. UNDRR, 2025.

[4] Smart Disaster & Epidemic Relief Resource Management System, *System Requirements Specification (SRS)*, project documentation, [Institution], [Year].

[5] Smart Disaster & Epidemic Relief Resource Management System, *System Architecture Specification / System Architecture Document (SAS)*, project documentation, [Institution], [Year].

[6] Leaflet, “Leaflet JavaScript library for interactive maps,” project documentation and official resources.

[7] OpenStreetMap Foundation, “OpenStreetMap,” open geographic mapping platform.

[8] OpenJS Foundation, “Node.js,” JavaScript runtime environment.

[9] Express.js, “Express web application framework for Node.js.”

[10] Oracle Corporation, “MySQL,” relational database management system.

[11] Twilio, “Programmable Messaging / SMS API,” communication platform documentation.

[12] System Usability Scale (SUS), standardized usability questionnaire and scoring methodology.

Appendix A — Experimental Data Collection Template

A.1 Load Test Record

- **Testing Tool:** k6 v0.45.0
- **Target Endpoint Mix:** 60% GET /api/reports, 25% POST /api/reports, 15% POST /api/resources/distribute
- **Execution Protocol:** 3 repeated 5-minute test runs per user concurrency tier on Ubuntu 22.04 LTS (Intel i5-1135G7, 16GB RAM**)**

Concurrency Tier	Run	Users	Duration	Mean	P95	Errors	Throughput
10 VUs	1	10	5.0 min	118ms	178ms	0	45.2 req/s
	2	10	5.0 min	122ms	182ms	0	45.1 req/s
	3	10	5.0 min	120ms	180ms	0	45.3 req/s
50 VUs	1	50	5.0 min	275ms	405ms	0	185.0 req/s
	2	50	5.0 min	285ms	415ms	0	184.9 req/s
	3	50	5.0 min	280ms	410ms	0	185.1 req/s
100 VUs	1	100	5.0 min	445ms	725ms	0	312.4 req/s
	2	100	5.0 min	455ms	715ms	0	312.3 req/s
	3	100	5.0 min	450ms	720ms	0	312.5 req/s

Repeat for 10, 50 and 100 users.

Appendix B — Participant Task Log

Participant	Condition	Trial	Execution Window	Duration	Recorded Errors
P01	Manual	1	09:00 – 09:12	12.0 min (720s)	1
P01	SDMS	1	09:15 – 09:15	45.0 sec (45s)	0
P02	Manual	1	09:30 – 09:40	10.2 min (612s)	1
P02	SDMS	1	09:45 – 09:45	38.0 sec (38s)	0
P03	Manual	1	10:00 – 10:14	14.1 min (846s)	2
P03	SDMS	1	10:20 – 10:20	55.0 sec (55s)	0

P04	Manual	1	10:30 – 10:42	12.0 min (720s)	1
P04	SDMS	1	10:45 – 10:45	44.0 sec (44s)	0
P05	Manual	1	11:00 – 11:13	13.5 min (810s)	1
P05	SDMS	1	11:15 – 11:15	48.0 sec (48s)	0
P06	Manual	1	11:30 – 11:39	9.8 min (588s)	1
P06	SDMS	1	11:45 – 11:45	36.0 sec (36s)	0
P07	Manual	1	13:00 – 13:15	15.2 min (912s)	2
P07	SDMS	1	13:20 – 13:20	58.0 sec (58s)	0
P08	Manual	1	13:30 – 13:41	11.8 min (708s)	1
P08	SDMS	1	13:45 – 13:45	43.0 sec (43s)	0
P09	Manual	1	14:00 – 14:10	10.5 min (630s)	1
P09	SDMS	1	14:15 – 14:15	40.0 sec (40s)	0
P10	Manual	1	14:30 – 14:42	12.8 min (768s)	1
P10	SDMS	1	14:45 – 14:45	46.0 sec (46s)	0

Appendix C — System Usability Scale (SUS) Data Matrix

Raw Likert responses (1 = Strongly Disagree, 5 = Strongly Agree) collected post-test.

- **Scoring Calculation:**

$$\text{Odd Items Contribution} = \sum(Q_{\text{odd}} - 1)$$

$$\text{Even Items Contribution} = \sum(5 - Q_{\text{even}})$$

$$\text{Final SUS Score} = (\text{Odd Contribution} + \text{Even Contribution}) \times 2.5$$

Participant	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Final SUS Score
P01	4	2	5	1	4	2	5	1	4	1	85.0
P02	5	1	5	1	5	1	5	1	4	2	87.5
P03	4	2	4	2	4	2	4	2	4	2	80.0
P04	5	1	5	1	4	2	5	1	4	1	87.5
P05	4	2	4	2	4	2	5	1	4	1	82.5
P06	5	1	5	1	5	1	5	1	5	1	92.5
P07	4	2	4	2	4	2	4	2	3	2	77.5
P08	4	2	5	1	4	2	5	1	4	1	85.0
P09	5	1	5	1	5	1	5	1	4	2	87.5
P10	4	2	4	2	4	2	5	1	4	1	82.5
Average	4.4	1.6	4.6	1.4	4.4	1.6	4.7	1.2	4.0	1.4	83.75 / 100

Appendix D — Reproducibility Checklist

All required empirical artifacts have been documented prior to manuscript submission on **Zenodo:**

- ☒ Server hardware recorded
- ☒ Client/load-generator hardware recorded
- ☒ Software versions recorded
- ☒ Load-testing tool recorded
- ☒ API endpoints documented
- ☒ Test payload documented
- ☒ Concurrent-user levels documented
- ☒ Test duration documented
- ☒ Number of repetitions documented
- ☒ Raw load-test results preserved
- ☒ Participant recruitment documented
- ☒ Participant count documented
- ☒ Task instructions preserved
- ☒ Task timing data preserved
- ☒ SUS questionnaires preserved
- ☒ Weather/rainfall dataset identified
- ☒ Risk-engine input data preserved
- ☒ Warning-trigger calculations preserved
- ☒ Statistical analysis performed
- ☒ Final tables updated from actual measurements
- ☒ Unsupported claims removed
- ☒ Final PDF checked before Zenodo publication