



iJRASET

International Journal For Research in
Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** X **Month of publication:** October 2026

DOI: <https://doi.org/10.22214/ijraset.2026.85227>

www.ijraset.com

Call: ☎ 08813907089

E-mail ID: ijraset@gmail.com

Smart Disaster Platform: A Web-Based System for NGO Verification and Skill-Aware Volunteer Allocation in Disaster Response

Deepa G¹, Dhivya K², Hemavathi R³, Vinodhini T(Mentor)⁴

^{1,2,3}Student, ⁴Assistant Professor, Department of CSE, Adiyamaan College of Engineering, Hosur, India

Abstract: During a disaster, relief agencies must quickly decide which volunteers to send where, while also ensuring that the organisations asking for help are genuine. Informal channels such as phone calls and messaging groups make both tasks slow and error-prone. This paper presents the Smart Disaster Platform, a role-based web application that connects volunteers, non-governmental organisations (NGOs) and an administrator on one system. NGOs register using their NGO-DARPAN identifier and are admitted only after administrator approval; volunteers follow the same approve-before-activate workflow. Approved users can raise disaster requests that state the disaster type, location, severity, number of volunteers needed and the food, water and medicine required. For each request the platform ranks available volunteers with a transparent weighted score built from skill relevance (40%), location match (30%), availability (20%) and fit with the request's resource needs (10%), and records each allocation so a volunteer cannot be assigned twice to the same event. The system is built with React, Node.js/Express and MySQL. We describe the architecture, database design, REST interface and scoring algorithm, and demonstrate the ranking with a worked example. Limitations and the planned extensions are discussed.

Keywords: Disaster management, volunteer allocation, NGO verification, NGO-DARPAN, decision support, role-based access, React, Node.js, MySQL

I. INTRODUCTION

India is exposed to floods, cyclones, earthquakes, landslides and fires, and every large event brings a rush of volunteers and NGOs who want to help [1], [2]. Willingness is rarely the bottleneck. What slows relief is coordination: the coordinating authority must find people with the right skills, close to the affected area and free to travel, and must be sure that the organisations making requests are legitimate. When this is done through phone calls, spreadsheets and messaging groups, it is slow, hard to audit and easy to get wrong.

This paper describes the Smart Disaster Platform, a web application built to address two practical problems. The first is trust: NGOs and volunteers should be vetted before they can act on the platform. The second is matching: given a disaster request, the platform should suggest which available volunteers are the most suitable, and explain why. The main contributions are:

- 1) *Approval-gated onboarding:* NGOs register with an NGO-DARPAN identifier [4] and volunteers with their skill and location; neither can log in until an administrator approves them.
- 2) *Transparent allocation score:* A four-factor weighted score ranks volunteers for each disaster request and returns a short, human-readable reason with every ranking.
- 3) *Complete working prototype:* A three-tier implementation (React, Express, MySQL) with role-based pages for volunteers, NGOs and administrators, a REST interface, and duplicate-allocation protection.

The rest of the paper is organised as follows. Section II reviews related work, Section III states the problem, Section IV presents the architecture, Section V explains the methodology and algorithm, Section VI describes the implementation with a worked example, Section VII discusses limitations and future work, and Section VIII concludes.

II. LITERATURE REVIEW

Volunteers are a valuable but hard-to-manage resource in disasters: they may offer help spontaneously, differ in skills and preferences, and can leave at any time. A recent structured review of 31 articles found that most published models address the assignment of volunteers to tasks, while forecasting the number of volunteers and composing balanced teams have received much less attention [3]. This supports the choice of assignment as the core function of our platform.

International policy also stresses the role of organised community participation and cooperation between agencies in reducing disaster risk [1], and the Disaster Management Act, 2005 provides the legal framework for coordinated response in India [2]. For NGOs, the NGO-DARPAN portal run by NITI Aayog gives each registered organisation a unique identifier and is a natural anchor for verification [4]. Many existing tools handle either volunteer registration or resource requests, but few combine NGO vetting, request management and ranked volunteer suggestions in one lightweight system that small organisations can host themselves. The Smart Disaster Platform targets this gap.

III. PROBLEM STATEMENT AND OBJECTIVES

Given a set of approved, available volunteers and an incoming disaster request, the system must (a) ensure that only vetted users participate, (b) rank volunteers by suitability for that specific request, and (c) record allocations so that coordinators and volunteers see a consistent picture. The objectives are to reduce the time needed to shortlist volunteers, to make the selection logic visible and adjustable, and to keep the system simple enough to deploy with commodity tools.

IV. SYSTEM ARCHITECTURE

A. Overview

The platform follows a three-tier architecture (Fig. 1). A React single-page application provides the user interface; an Express REST service implements the business rules; and a MySQL database stores users, requests and allocations. The client and server communicate through JSON over HTTP, following the REST style [5].

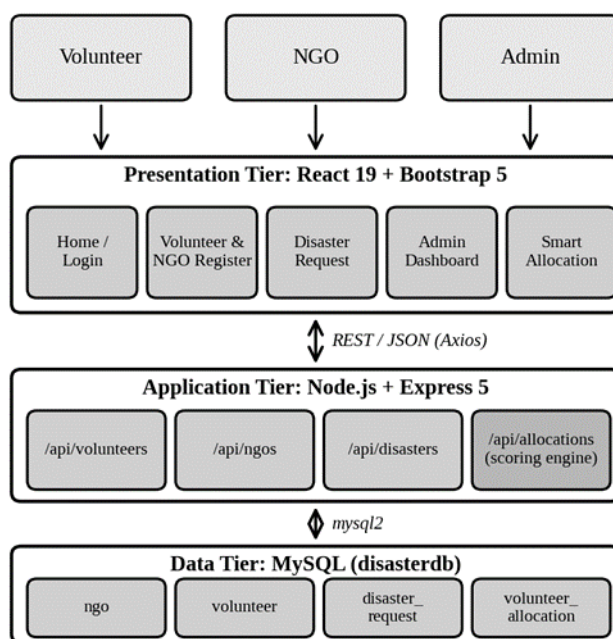


Fig. 1 Three-tier architecture of the Smart Disaster Platform

B. Technology Stack

Table I lists the technologies used. All are open source and widely documented, which keeps the barrier to deployment low.

TABLE I

TECHNOLOGY STACK

Layer	Technology
Front end	React 19, React Router 7, Bootstrap 5, Axios [6], [9], [10]
Back end	Node.js, Express 5, CORS middleware [7]

Layer	Technology
Database	MySQL accessed via the mysql2 driver [8]
Interface	REST over HTTP, JSON payloads [5]

C. Database Design

The schema has four tables. The ngo table holds the organisation's contact details, NGO-DARPAN identifier, registration number, registering authority, sectors and operational area, together with an approval status (Pending, Approved or Rejected). The volunteer table stores name, email, phone, skill, location, availability and approval status. The disaster_request table stores the disaster type, location, severity, number of volunteers needed, quantities of food, water and medicine, and a status. The volunteer_allocation table links a volunteer to a disaster request with a status and timestamp. Email and phone are checked for duplicates at volunteer registration, and the NGO-DARPAN identifier must be unique.

D. Role-Based Access

Three roles are supported. Public visitors can only view the home page and register. Access to other pages is controlled by a protected-route component on the client, as summarised in Table II.

TABLE II
ROLES AND PERMITTED PAGES

Role	Permitted functions
Volunteer	Register, log in after approval, raise disaster requests, view own assignments
NGO	Register, log in after approval, raise requests, view volunteers, run smart allocation
Admin	Approve or reject NGOs and volunteers, view all requests, run and confirm allocations

V. METHODOLOGY

A. Registration and Approval Workflow

Every NGO or volunteer account is created with the status Pending. A login attempt by a pending or rejected account is refused with an explanatory message, and only accounts approved by the administrator can sign in. For NGOs, the administrator reviews the submitted NGO-DARPAN identifier and registration details on the dashboard before choosing Approve or Reject. This human-in-the-loop step keeps the platform simple while still preventing unvetted organisations from acting on it.

B. Disaster Request

An authorised user submits a request containing the disaster type, location, severity (Low, Medium, High or Critical), the number of volunteers required, and the quantities of food, water and medicine needed. New requests are stored with the status Pending.

C. Smart Volunteer Allocation

For a selected request, the platform considers only volunteers who are approved and marked Available. Each is given a score from 0 to 100 using four components, combined as in (1):

$$S = 0.40S_{\text{skill}} + 0.30S_{\text{loc}} + 0.20S_{\text{avail}} + 0.10S_{\text{req}} \quad (1)$$

where each component lies between 0 and 100. The final score is rounded to the nearest integer and volunteers are sorted from highest to lowest.

1) *Required skills*: The disaster type selects a base list of relevant skills (for example, rescue, first aid, medical, swimmer and engineer for floods; fire, rescue and first aid for fires; construction for earthquakes and landslides). The list is extended if the request needs medicine (medical, doctor, nurse), food or water (logistics, distribution), or if the severity is High or Critical (rescue, first aid, medical).

- 2) *Component scores*: The scoring rules are given in Table III. Skill is 100 for an exact match with a required skill, 80 for a partial (substring) match and 30 for any other stated skill. Location is 100 for identical text, 80 when one location name contains the other and 50 otherwise. Availability is 100 for an available volunteer. The requirement component is 100 when the volunteer's skill supports at least one of the request's specific needs (medicine, food or water, high severity) and 50 otherwise.

TABLE III
SCORING RULES FOR THE SMART ALLOCATION SCORE

Component	Weight	Rule (component value)
Skill relevance	0.40	Exact match 100; partial 80; other skill 30; none 0
Location match	0.30	Same 100; contains 80; different 50
Availability	0.20	Available 100
Requirement fit	0.10	Supports a stated need 100; else 50

- 3) *Explanation*: With each ranked volunteer the service returns the four component scores and a short reason such as "Relevant disaster skill, Nearby location, Currently available", so a coordinator can see why a person was suggested and override the suggestion if needed.
- 4) *Confirming an allocation*: The coordinator confirms a suggestion, and the service writes a record to the allocation table with the status Allocated. Before inserting, the service checks whether the volunteer is already allocated to the same disaster and rejects duplicates.

D. REST Interface

Table IV summarises the main endpoints exposed by the Express service.

TABLE IV
MAIN REST ENDPOINTS

Endpoint	Purpose
POST /api/volunteers/register, /login	Volunteer sign-up and sign-in
PUT /api/volunteers/approve reject/:id	Admin decision on a volunteer
POST /api/ngos/register, /login	NGO sign-up and sign-in
PUT /api/ngos/approve reject/:id	Admin decision on an NGO
POST /api/disasters/request	Create a disaster request
GET /api/disasters	List all disaster requests
GET /api/allocations/recommend/:id	Ranked volunteers for a request
POST /api/allocations/allocate	Confirm an allocation
GET /api/allocations/assigned	List assigned volunteers

VI. IMPLEMENTATION AND ILLUSTRATIVE EXAMPLE

A. User Interface

The front end consists of pages for the home screen, login, volunteer registration, NGO registration, disaster request, NGO portal, volunteer listing, administrator dashboard and smart allocation.

The dashboard shows counts of NGOs (total, pending, approved, rejected), volunteers and disaster requests, lets the administrator approve or reject each registration, and lets a coordinator choose a disaster, view ranked volunteers and allocate them with one click. Volunteers see the disasters to which they have been assigned.

B. Worked Example

To illustrate the ranking we apply the implemented rules to a hypothetical flood request in Chennai with High severity, requiring food, water and medicine. Four available, approved volunteers are considered. This example is constructed to explain the algorithm; it is not a field evaluation. The results are shown in Table V and Fig. 2.

TABLE V
EXAMPLE RANKING FOR A HIGH-SEVERITY FLOOD REQUEST

ID	Skill	Location	Component scores (S/L/A/R)	Score
V1	Medical	Chennai	100 / 100 / 100 / 100	100
V2	Logistics	Chennai Central	100 / 80 / 100 / 100	94
V3	Rescue	Madurai	100 / 50 / 100 / 100	85
V4	Cooking	Madurai	30 / 50 / 100 / 50	52

V1 ranks first because the skill is directly relevant, the location is identical and the skill supports the medicine requirement. V2 is a strong second: logistics supports the food and water need and the location is nearby. V3 is a good skill match but far away, which lowers the score. V4 is available but has neither a relevant skill nor a nearby location, so is ranked last. Fig. 2 breaks each score into its weighted contributions.

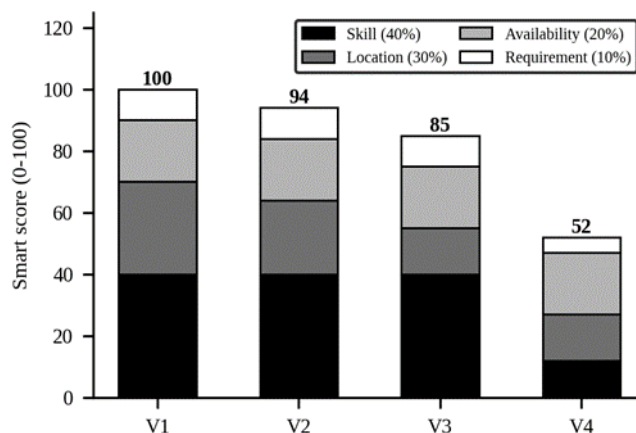


Fig. 2 Weighted contribution of each component to the smart score for the four example volunteers

VII. LIMITATIONS AND FUTURE WORK

The current system is a working prototype and has limitations that we plan to address.

- 1) *Security*: Passwords should be salted and hashed, sessions should use signed tokens with server-side role checks, and database credentials should come from environment variables. These changes are required before production use.
- 2) *Location matching*: Matching uses text comparison of place names. Geocoding volunteers and incidents and using road distance would be more accurate.
- 3) *Skill matching*: Skills are free text matched by keywords. A controlled vocabulary or a semantic similarity measure would reduce missed matches.
- 4) *NGO verification*: Verification is currently a manual administrator decision. A rule-based completeness check on the submitted identifiers, and an automated lookup against the NGO-DARPAN portal if an official interface is available, could assist the administrator.

- 5) *Evaluation*: The weights (0.40, 0.30, 0.20, 0.10) are set by design judgement. Future work should validate them with disaster-management practitioners, compare the ranking against manual allocation on real or simulated incidents, and measure the time saved.

Further extensions include notifications by SMS or email, live status tracking of resources, handling of the required number of volunteers per request, and a mobile client.

VIII. CONCLUSIONS

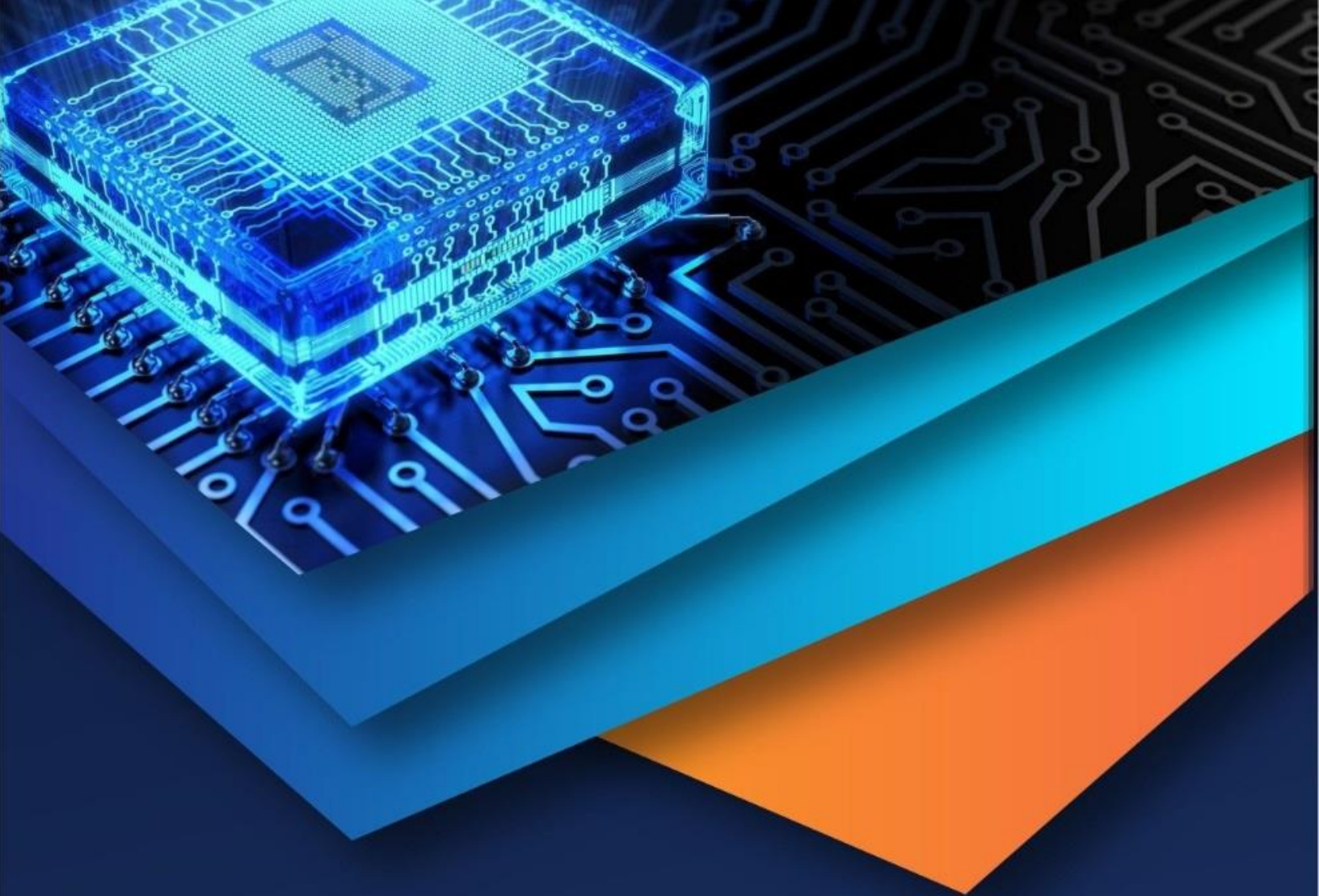
We presented the Smart Disaster Platform, a role-based web system that combines approval-gated onboarding of NGOs and volunteers, structured disaster requests and an explainable weighted ranking of volunteers. The implementation uses common open-source technologies, which makes it easy to deploy and extend. The worked example shows how the ranking behaves, and the discussion identifies the security, matching and evaluation work needed to move from prototype to field use.

IX. ACKNOWLEDGMENT

The authors thank T.Vinothini for guidance and support during this work.

REFERENCES

- [1] United Nations Office for Disaster Risk Reduction, *Sendai Framework for Disaster Risk Reduction 2015–2030*. Geneva, Switzerland: UNDRR, 2015.
- [2] Government of India, *The Disaster Management Act, 2005*, Act No. 53 of 2005, New Delhi, India, 2005.
- [3] F. Hager and M. Reuter-Oppermann, “Managing volunteers in disaster situations – An overview of models for decision-makers,” in *Proc. 21st Int. ISCRAM Conf.*, Münster, Germany, May 2024, doi: 10.59297/h8tk0g80.
- [4] NITI Aayog, “NGO-DARPAN,” Government of India. [Online]. Available: <https://ngodarpan.gov.in/>
- [5] R. T. Fielding, “Architectural styles and the design of network-based software architectures,” Ph.D. dissertation, Dept. Inf. Comput. Sci., Univ. California, Irvine, CA, USA, 2000.
- [6] Meta Open Source, “React.” [Online]. Available: <https://react.dev/>
- [7] OpenJS Foundation, “Express – Node.js web application framework.” [Online]. Available: <https://expressjs.com/>
- [8] Oracle Corporation, *MySQL 8.0 Reference Manual*. [Online]. Available: <https://dev.mysql.com/doc/refman/8.0/en/>
- [9] Bootstrap Team, “Bootstrap.” [Online]. Available: <https://getbootstrap.com/>
- [10] “Axios – promise-based HTTP client for the browser and Node.js.” [Online]. Available: <https://axios-http.com/>



10.22214/IJRASET



45.98



IMPACT FACTOR:
7.129



IMPACT FACTOR:
7.429



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089  (24*7 Support on Whatsapp)