

SafePaws - A MERN Stack-Based Web Platform for Centralized Stray Animal Rescue Coordination

**Dr. Pragati Fatinge¹, Hansika Butoliya², Khushboo Pingle³,
Manish Dubey⁴, Guddi Singh⁵, Princi Singh⁶**

¹Guide, Dept. CSE, G.H. Rasoni College of Engg.&Mgmt., Nagpur, India

^{2,3,4,5,6}Dept. of Computer Science& Engg., G.H. Rasoni College of Engg.&Mgmt., Nagpur, India

Abstract

Stray animal rescue in Indian cities often depends on phone calls, forwarded messages, and social-media posts. This makes it difficult to know whether a case has already been reported, who is responding, and whether the animal has been safely assisted. SafePaws presents a centralized web platform that brings citizens, rescuers, NGO coordinators, and administrators into one shared rescue workflow. A citizen can submit animal details, photographs, location, and emergency information; coordinators can review and assign cases; rescuers can update progress; and authorized staff can verify completion before closure. A working prototype was configured with demonstration data for the Nagpur metropolitan area and used to check reporting, assignment, tracking, duplicate screening, notifications, evidence review, and summary views work together. The prototype contains more than 100 rescue cases, over 40 rescuer records, and six partner organizations. These figures describe the demonstration environment and do not represent city-wide operational performance. The study therefore focuses on the design, implementation, and workflow integration of the proposed platform and identifies the field evaluation needed before wider deployment.

Keywords - MERN stack, GPS geolocation, geospatial indexing, REST API, role-based access control, duplicate-report detection, evidence verification, MongoDB, React.js, Node.js, Express.js

INTRODUCTION

Stray-animal rescue in an urban setting involves more than locating an animal in distress. Once an injured, sick, abandoned, or trapped animal is noticed, the report must reach someone who can respond, the case must be assigned, and the progress must remain visible until the intervention is completed. In many communities, these steps still depend on phone calls, messaging groups, social-media posts, and personal contacts. Without a shared record, different people may act on the same incident while another case receives no response. The literature reviewed for SafePaws shows that digital systems already support animal reporting, adoption, shelter activities, and community participation, but these functions are often distributed across separate applications or focused on a single part of the process [1]-[5].

A second difficulty is the quality of location information. A landmark or verbal description may be understandable to a local volunteer but can be imprecise for a responder who does not know the area. SafePaws therefore treats location as a primary part of the rescue record. During reporting, the web

application requests permission to use the device location service. With permission, the browser can obtain the current position using the available positioning sources on the device, including GPS and network-assisted positioning, and returns a latitude-longitude pair. Those coordinates are attached to the rescue request and stored with the case. The user can also adjust the marker on the map when the automatically detected point does not exactly match the animal's observed location. This gives the coordination workflow a precise geographical reference for nearby-case discovery and assignment.

SafePaws is organized around four operational participants: citizens, rescuers, NGO coordinators, and administrators. Citizens create reports; NGO coordinators review and assign cases; rescuers carry out field interventions and update progress; and administrators supervise users, organizations, and case records. The platform is designed to keep one rescue request intact as it moves through these roles instead of creating disconnected records at each stage. This case-oriented view follows the gap identified in the project literature survey, where existing work commonly emphasizes adoption, shelter management, reporting, tracking, or community information sharing rather than a complete intake-to-closure rescue workflow [1]-[10].

The contribution of this work is a design and implementation study supported by a working prototype. The study evaluates whether the proposed workflow can be implemented as a coherent system and whether the main operational functions can be exercised together. It does not treat the seeded demonstration records as evidence of city-wide rescue impact. The principal contributions are:

- A single workflow connecting citizen reporting, organizational triage, field response, and administrative supervision.
- A controlled rescue lifecycle that records permitted status changes and preserves a time-stamped case history.
- An interpretable method for identifying likely duplicate reports using location, time, animal type, and reported condition.
- A verification step that separates a rescuer's completion claim from final case closure.
- A two-mode implementation that supports both demonstration data and persistent backend services.
- A prototype evaluation approach that clearly separates implementation characteristics from real-world performance claims.

The rest of the paper is organized as follows. Section II reviews related work and identifies the research gap. Section III presents the system architecture. Section IV describes the main data entities and rescue lifecycle, followed by the duplicate-report method in Section V and evidence verification in Section VI. Sections VII and VIII discuss analytics and the frontend structure. Section IX presents the all six interface screenshots of the working SafePaws prototype, while Section X describes the evaluation scope. Limitations, conclusion, and the ethics and data statement follow.

RELATED WORK

A. Animal Rescue and Welfare Platforms

Recent studies show a growing use of digital tools for animal welfare. Tukadiya et al. describe PawConnect as a system that combines rescue reporting with adoption and lost-pet functions, along with location tracking and notifications [1]. Ikhsan et al. present a modular animal-welfare application intended to improve communication, transparency, and volunteer coordination [2]. Erin et al. focus on modernizing shelter adoption through a mobile application, while Wang et al. examine intelligent identification and monitoring for stray-animal management [3], [4]. Efe et al. propose a location-aware application covering

several stray-animal activities, including tracking, adoption, vaccination, and shelter management [5]. Together, these studies show that digital reporting, location information, and shared records can support animal-welfare operations.

B. Shelter, Adoption, and Community-Oriented Systems

A related group of studies concentrates on shelter administration, adoption, matching, and community participation. Jagtap's Shelter Soul work brings together shelter records, volunteer coordination, donations, matching, and analytics [6]. Torres et al. use content-based filtering within an animal adoption and rescue setting [7], while Gatmaitan et al. describe the design and assessment of a web-based pet-adoption application [8]. Chan and Dalim present a mobile application for animal conservation and rescue reporting [9], and Wee et al. describe a web-based system for shelter and rescue activities [10]. These studies reinforce the value of centralized information, but their primary emphasis differs from the operational coordination problem addressed by SafePaws.

C. Research Gap

Across the reviewed literature, the emphasis is generally on one or more of the following: reporting, adoption, shelter administration, animal identification, or community information exchange [1]-[10]. SafePaws addresses a narrower but important gap by treating an individual rescue request as a managed case. The same record follows the incident from initial reporting and location capture through organizational review, rescuer assignment, field-status updates, evidence submission, verification, and final closure. The objective is not to replace existing welfare applications, but to provide a structured coordination layer that keeps these operational steps connected.

D. Location and Duplicate Reporting

Location also matters because more than one citizen can report the same animal. When each message is treated as an independent incident, organizations may spend resources on duplicate requests while another case remains unattended. SafePaws therefore uses geographical and temporal proximity as important signals for duplicate screening, with animal type and reported condition providing additional context. The method is intentionally explainable so that a person reviewing the report can understand why an earlier case was considered potentially related.

SYSTEM ARCHITECTURE

A. Overview

SafePaws follows a three-layer client-server structure. The user-facing layer contains the public pages, authenticated dashboards, reporting forms, notifications, analytics views, and map screens. The service layer receives requests, checks access rights, validates incoming data, applies rescue-state rules, calculates duplicate-report scores, and manages evidence verification. The data layer stores user, organization, rescue, assignment, history, notification, and evidence information. The separation makes the workflow easier to maintain and also allows the interface to operate with demonstration records before a persistent backend is connected.

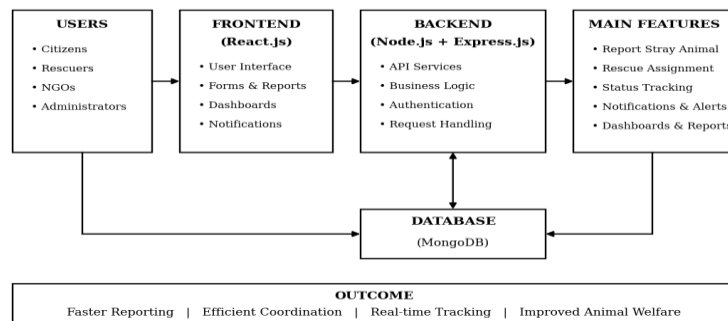


Fig. 1. Block diagram of the SafePaws system architecture.

The block diagram shows how the four user groups connect through the frontend to the backend services and the central database. The main functions - reporting, rescue assignment, status tracking, notifications, and dashboards - are supplied through this shared path. In practical terms, the diagram emphasizes that the rescue case is the common object connecting citizens, field responders, partner organizations, and administrators.

Development was carried out incrementally. The first phase used bundled demonstration data to build and inspect the main screens. The second connected the frontend to the service layer and database, added authentication, and enabled image storage. The final phase added duplicate screening, evidence verification, notifications, analytics, and role-scoped views. This sequence allowed the workflow to be exercised before all external services were required.

B. Technology and Service Layer

The implementation uses a MERN-oriented full-stack design. React with TypeScript is used for the frontend, including routing, responsive interfaces, dashboards, charts, and map views. Node.js and Express.js provide the application services, while MongoDB is used for persistent storage. Signed authentication tokens, role checks, request validation, a geospatial index, and cloud image storage support the operational requirements. These technologies are implementation choices; the main contribution is the way they are combined around a complete rescue workflow.

An adapter boundary separates backend documents from the data structures expected by reusable interface components. Identifiers, role values, nested references, location arrays, and rescue-history records are normalized before presentation. This reduces coupling between storage and interface code and allows the same user-facing components to work with both the demonstration mode and the connected deployment.

C. Access Control and Request Handling

The application is divided into domains such as authentication, reports, assignments, evidence, notifications, analytics, and administration. A request is accepted only after the authenticated user and the requested operation have been checked. Validation is performed before data is persisted, and invalid rescue-state changes are rejected by the server. This design is important because hiding a button in the user interface is not sufficient protection if a client can construct a request directly.

Access is scoped by role. Citizens can work with reports they created, rescuers can see cases assigned or eligible for them, NGO users operate on cases linked to their organization, and administrators have broader management rights. The same restrictions are applied to service requests rather than being limited to navigation in the frontend.

DATA MODELS AND RESCUE LIFECYCLE

A. Data Model Design

The data model centers on six entities that together preserve the operational history of a rescue. The RescueReport acts as the main case record and is linked to user, organization, assignment, history, and notification information. Table I summarizes the role of each entity and the fields that are most relevant to coordination and traceability.

Entity	Purpose	Important fields
User	Identity and role management	Role, status, availability
Rescue Report	Main rescue case	Animal data, location, status, timestamps
Organization	NGO profile and verification	Verification status, service area
Rescue Assignment	Case-rescuer relationship	Report, rescuer, assigning organization
Rescue History	Immutable event log	Actor, previous/new status, timestamp
Notification	Role-targeted updates	Recipient, type, read status, reference

TABLE I. SIX PRIMARY ENTITIES AND THEIR ROLES

The table highlights the separation between the current case state and the history of what happened to that case. In particular, RescueHistory and Notification prevent important coordination events from being reduced to a single editable status field, while RescueAssignment records who was asked to act on a report.

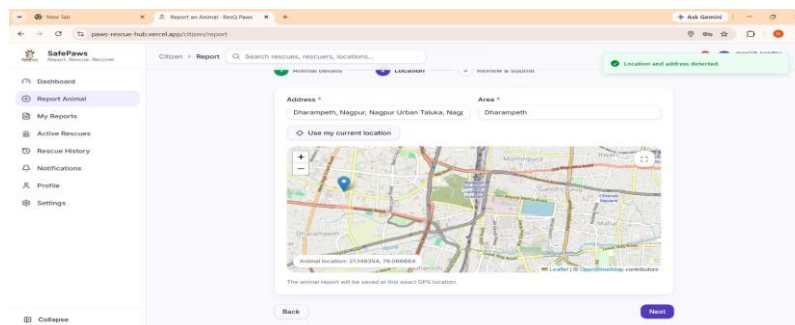


Fig. 2. Location capture during the “Report an Animal” workflow.

Fig. 2 shows the location step after the animal details are entered. With location permission, the current position is shown on the map and saved with the case; the marker can be adjusted before submission when needed.

B. Rescue Lifecycle

Each case moves through seven defined states: REPORTED, ASSIGNED, ACCEPTED, IN_PROGRESS, RESCUED, CLOSED, and CANCELLED. The first six form the normal route from report creation to verified completion, while CANCELLED is used when a case is no longer actionable. A status change is accepted only when it is valid for the current state. The server records the actor, previous status, new status, time, and optional notes for each accepted transition and rejects invalid requests.

The event history also supports reproducible duration measures. Response time is derived from assigned time minus reported time; acceptance time from accepted time minus assigned time; rescue duration from rescued time minus start time; and total case duration from closed time minus reported time. Keeping the timestamps as events allows these measures to be computed from the same records used for case tracking.

$$\begin{aligned} T_{\text{response}} &= T_{\text{assigned}} - T_{\text{reported}} \\ T_{\text{acceptance}} &= T_{\text{accepted}} - T_{\text{assigned}} \\ T_{\text{rescue}} &= T_{\text{rescued}} - T_{\text{started}} \\ T_{\text{case}} &= T_{\text{closed}} - T_{\text{reported}} \end{aligned}$$

where each T term denotes the recorded timestamp for the corresponding lifecycle event. These differences allow response, acceptance, rescue, and total case duration to be derived directly from the audit history.

C. Role-Based Rescue Flow

The citizen workflow begins with a guided report containing the animal type, number of animals, condition, emergency level, photographs, description, and location. After submission, the case enters REPORTED. The location can come from the current device position or be selected manually on the map, so the workflow remains usable when automatic positioning is unavailable or needs correction.

NGO coordinators review incoming reports and can assign a rescuer using the reported location, rescuer availability, and current workload as operational signals. Rescuers can accept assigned work and update the case while traveling to and assisting the animal. After the field intervention, the rescuer submits completion evidence. The case can then be moved to RESCUED, but final closure remains a separate review decision for an authorized coordinator or administrator.

DUPLICATE REPORT DETECTION

A. Problem Definition

Duplicate reporting occurs when more than one citizen describes the same animal or incident within a relatively small geographical and time window. Treating every submission as a separate case can lead to repeated assignment work. SafePaws therefore does not silently delete or merge reports. Instead, it identifies plausible duplicates and shows an explanation that allows the citizen or coordinator to decide what to do next.

B. Scoring Method

The detector combines four signals. Location proximity can contribute up to 45 points, animal-type agreement contributes up to 35 points, time proximity contributes up to 15 points, and matching reported condition contributes 5 points. The maximum score is therefore 100. These weights are heuristic prototype settings and are not presented as statistically calibrated classifier parameters.

For a candidate report at distance d metres and an age difference h hours, the location and time components decay linearly as the distance or time difference grows. The implementation uses a 500 m spatial radius and a 48 h reporting window. The complete score is the sum of the four components, so the reason for a warning can be traced back to concrete properties of the two reports rather than an opaque output.

The duplicate detector can therefore be expressed mathematically as follows. The variables d and h represent the spatial distance in metres and the reporting-time difference in hours, respectively.

$$\begin{aligned} S_D &= S_L + S_T + S_A + S_C \\ S_L &= 45 \times \max(0, 1 - \frac{d}{500}) \end{aligned}$$

$$S_A = \begin{cases} 35 & \text{if animal type matches} \\ 0 & \text{otherwise} \end{cases}$$

$$S_T = 15 \times \max(0, 1 - \frac{h}{48})$$

$$S_C = \begin{cases} 5 & \text{if reported condition matches} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Class}(S_D) = \begin{cases} \text{Likely Duplicate} & \text{if } S_D \geq 75 \\ \text{Possible Duplicate} & \text{if } 50 \leq S_D < 75 \\ \text{No Warning} & \text{if } S_D < 50 \end{cases}$$

Here, S_D is the final duplicate score; S_L , S_A , S_T , and S_C are the location, animal-type, time, and condition components; d is the distance between the two report locations; h is the difference between their reporting times; 500 m is the configured spatial radius; and 48 h is the configured time window. The weights 45, 35, 15, and 5 sum to 100

Illustrative calculation: consider two reports that are 100 m apart, were submitted 6 h apart, describe the same animal type, and record the same condition. The component scores are:

$$S_L = 45(1 - \frac{100}{500}) = 36$$

$$S_T = 15(1 - \frac{6}{48}) = 13.125$$

$$S_A = 35, S_C = 5$$

$$S_D = 36 + 13.125 + 35 + 5 = 89.125$$

Since $89.125 \geq 75$, the report pair falls within the likely-duplicate range defined for the prototype.

C. Operational Interpretation

The prototype uses three operational ranges. Scores at or above 75 generate a stronger duplicate warning; scores from 50 up to but not including 75 indicate a possible duplicate; and scores below 50 do not trigger a warning. These thresholds are intended to support human review rather than replace it. The current weights and thresholds should be recalibrated using a labelled collection of genuine duplicate and non-duplicate report pairs before any accuracy claim is made.

D. User Experience

When duplicate candidates are found, the reporting interface can show the earlier case and the signals that produced the score. The citizen may acknowledge the warning and continue when the cases are different, or return to the existing case when it is the same incident. This approach reduces unnecessary suppression of legitimate reports while still bringing possible duplication to the reviewer's attention.

RESCUE EVIDENCE VERIFICATION

A. Motivation

A rescue should not be considered closed merely because the field rescuer marks an intervention as complete. SafePaws separates the rescuer's completion claim from the final closure decision by requiring evidence and an authorized review. This creates an auditable checkpoint between field action and case closure.

B. Workflow

After completing the intervention, the rescuer can submit photographs, an animal-condition description, treatment notes, and, when available, the completion location. The case moves to RESCUED and the evidence enters a pending review state. An NGO coordinator or administrator can verify the submission, which closes the case, or reject it with a reason so that the rescuer can revise and resubmit. Each submission and review decision remains in the evidence history.

The evidence requirement creates two separate assertions: the rescuer reports that the field work has been completed, while an authorized reviewer confirms that the material provided is sufficient for closure. The workflow improves traceability without assuming that a photograph alone proves the quality of treatment or the animal's long-term condition.

C. Evidence and Notifications

Evidence records contain the submitted photographs, notes, optional completion coordinates, submitter information, review state, reviewer information, and rejection reason. Related notifications are generated for the appropriate participants: a new submission creates a review task for the organization or administrator, verification produces a confirmation for the rescuer and the original reporter, and rejection returns the case to the rescuer together with the recorded reason.

ANALYTICS ENGINE

A. Overview

The analytics layer derives summaries from the rescue cases and their history rather than relying on manually typed dashboard numbers. This keeps the reporting layer connected to the same records used by the operational workflow. The prototype can summarize total reports, active rescues, completed cases, emergency levels, animal types, and time-based trends, while lifecycle timestamps provide the basis for duration calculations.

B. Metrics and Role Scoping

Distribution views can group cases by animal type, rescue status, and emergency level. Monthly counts can be derived from reported and completed timestamps, and response time or rescue duration can be computed from the recorded lifecycle events. These values should be read as operational summaries of the available records. In the present prototype, they describe the seeded demonstration environment rather than city-wide rescue performance.

Analytics results are also filtered according to the authenticated role. NGO users receive information associated with their organization, rescuers see their assigned cases, citizens see their own reports, and administrators can view the system-wide dataset. This preserves the same data-access boundary used by the transactional workflow.

FRONTEND ARCHITECTURE

A. Routing and Navigation

The frontend uses role-specific route groups so that users are directed to the screens relevant to their work. Reporting, assignment, evidence submission, analytics, notification, and administration views are exposed only to roles that are permitted to use them. This keeps the primary rescue path visible without presenting every user with unrelated controls.

B. State and Data Handling

A centralized frontend store maintains authentication state, rescue-report collections, notifications, and the current operating mode. The application can switch between demonstration data and connected service mode according to the configured backend endpoint. A shared adapter layer normalizes database identifiers, role values, nested references, geographical coordinates, and timeline records before those values reach reusable components.

This separation allows the same reporting, map, dashboard, and timeline components to work with seeded data during development and with persistent records in the connected configuration. It also reduces the amount of database-specific logic that has to be repeated across interface components.

C. Component Organization

Shared components are combined with role-specific workflows for reporting, assignment, evidence submission, administration, notifications, analytics, and duplicate comparison. The map views are reused for reporting and field-response tasks, while the rescue timeline provides a common representation of case progress to the roles that are allowed to see it.

IMPLEMENTATION AND PROTOTYPE RESULTS

A. Prototype Deployment

A working SafePaws prototype was configured for the Nagpur metropolitan area and populated with demonstration records for the four supported user roles. The environment contains six partner NGO organizations, more than 40 rescuer records, and more than 100 rescue cases distributed across lifecycle states and emergency levels. The geographical coordinates in the dataset are sufficient to exercise map display and location-dependent functions. These records are for workflow demonstration and testing; they are not presented as evidence of actual field participation.

The deployment was used to exercise the complete path from reporting through closure. Citizens can submit and track reports, rescuers can discover and act on eligible cases, NGO coordinators can triage and assign work, and administrators can manage users and organizations while inspecting the case history. The following screenshots are taken from the working SafePaws prototype and are placed next to the parts of the workflow they illustrate.

B. Public Home Screen

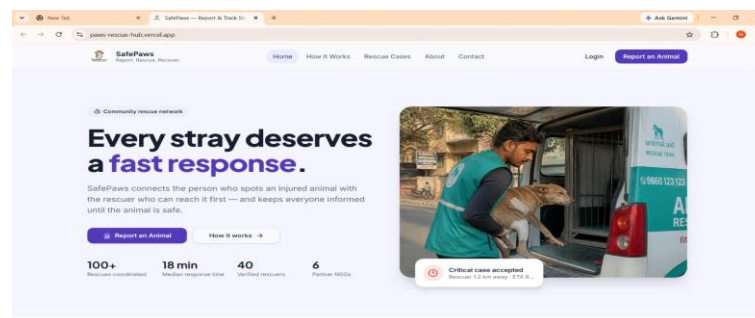


Fig. 3. Home page of the SafePaws platform.

The home screen is the public entry point to the platform. It introduces the reporting and tracking workflow and displays summary information from the seeded demonstration environment. The visible figures provide a quick view of the configured workload, while the live-status card demonstrates how an active rescue can be highlighted for users entering the system.

C. Citizen Authentication

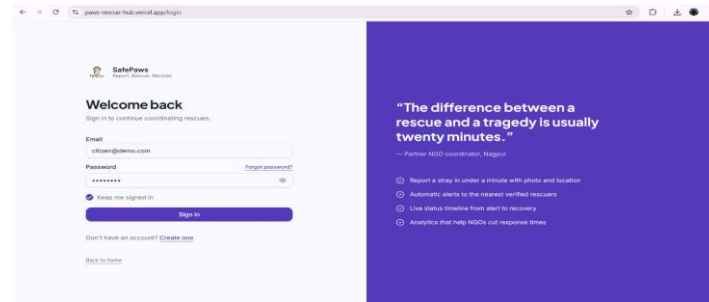


Fig. 4. Citizen login screen.

The login screen provides the entry point for authenticated users. After a successful sign-in, the system routes each account to the dashboard that corresponds to its assigned role. This keeps citizen reporting separate from rescuer, NGO, and administrator operations and prevents users from being presented with functions outside their scope.

D. Rescue Report Form

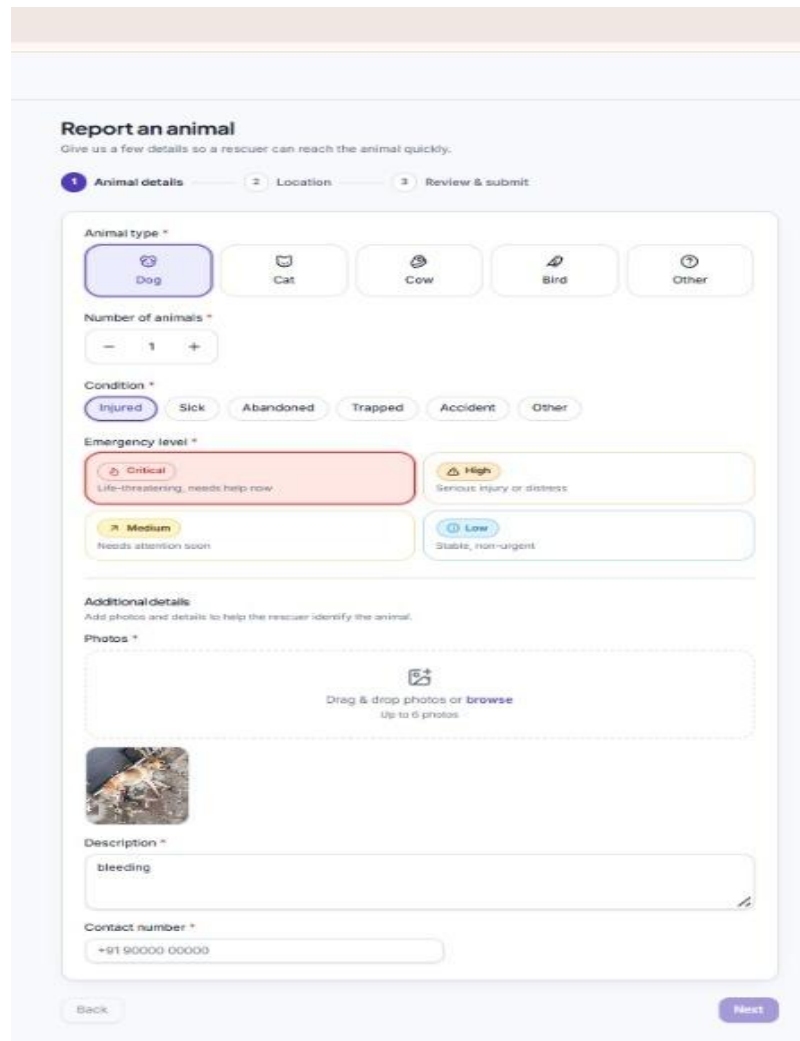


Fig. 5. "Report an Animal" form

The reporting form gathers the information needed to start a rescue case in a consistent format. It asks for the animal type, number of animals, condition, emergency level, photographs, a short description, and a

contact number. Structuring these fields avoids the missing details and ambiguity that can occur when a rescue request is communicated only through an informal message.

E. Geographical Location Capture

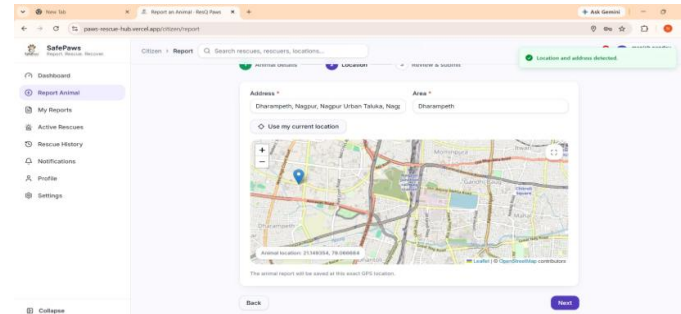


Fig. 6. Location step of the "Report an Animal" form.

The location step is used to attach the animal to a precise point on the map. The citizen can use the current device position or adjust the marker before submitting the case. In the screenshot, the selected point is displayed with its latitude and longitude, demonstrating the same geographical input that later supports nearby-case discovery, assignment, and duplicate screening.

F. Submission Confirmation

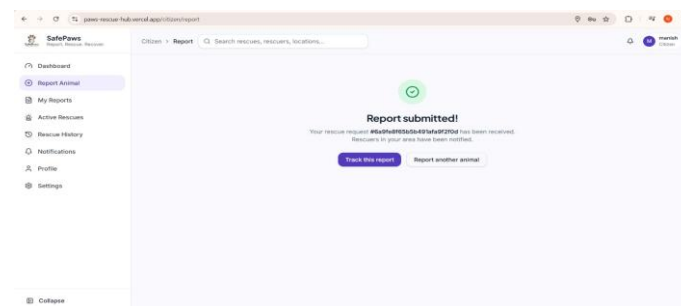


Fig. 7. Report submission confirmation screen.

After submission, the interface confirms that the report has been received and provides a unique request identifier. This screen marks the handover from the citizen-facing reporting flow to the operational workflow. From this point, the case can be reviewed, assigned, updated, and tracked by the authorized users involved in the rescue.

G. Rescue Status Timeline

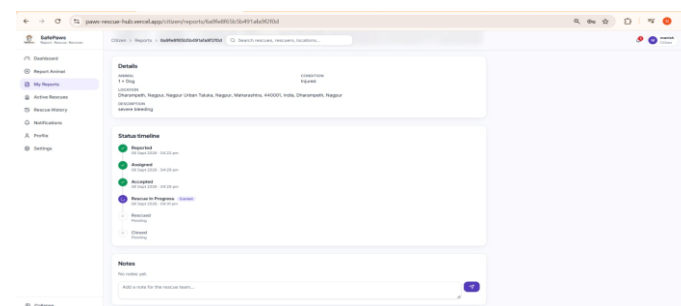


Fig. 8. Status timeline of a tracked rescue report.

The status timeline presents the rescue as a sequence of recorded events rather than as a single manually edited label. The example shows an injured-dog case reported in the Dharampeth area of Nagpur, followed by assignment, rescuer acceptance, and movement into the rescue-in-progress stage while later stages remain pending. This view gives the reporter and authorized coordinators a common history of the case.

H. Prototype Configuration Table

Table II summarizes the principal configuration characteristics of the deployed demonstration environment. These values are descriptive implementation details, not controlled measurements of field performance. In particular, the response-time value comes from seeded data and should not be interpreted as the response time of a real rescue service.

Metric	Value
Total rescue cases in demo data	100+
Median response time in seeded data	18 minutes
Rescuer records in demo data	40+
Partner organization (NGO) records	6
API endpoint count	35+
Backend data models	6 schemas
Geospatial index type	2dsphere
Duplicate-detection radius	500 metres
Duplicate-detection time window	48 hours

TABLE II. PROTOTYPE CONFIGURATION CHARACTERISTICS

The table demonstrates the configured scale of the prototype and the parameters used to exercise the main location-aware and duplicate-screening features. It also provides a compact description of the implementation footprint without presenting the demonstration records as a representative sample of Nagpur rescue activity.

I. Feature Coverage

At the workflow level, the prototype covers the complete operational path. Citizens can create reports, receive duplicate warnings, follow progress, and view notifications. Rescuers can discover eligible cases, accept work, update the case, submit evidence, and interact with the map. NGO coordinators can triage cases, assign rescuers, verify evidence, manage duplicate warnings, and review analytics. Administrators can manage users and organizations, inspect the audit trail, and view system-wide summaries.

EVALUATION METHODOLOGY

The present evaluation is intentionally limited to prototype verification and descriptive configuration characteristics. The project materials do not provide a labelled duplicate-report dataset, a controlled response-time baseline, measured precision or recall, a sustained field-deployment dataset, or a statistical comparison against another rescue workflow. Accordingly, the paper does not present those quantities as measured research outcomes.

A future usability evaluation should recruit representative participants under the applicable ethical procedures and measure task completion time, task success, error rates, and a standardized usability measure for the citizen, rescuer, NGO, and administrator tasks. A separate performance study should examine service response time, duplicate-query latency, image-upload time, concurrent-user throughput, and database behavior using median and 95th-percentile values.

For duplicate detection, a labelled set of report pairs should be classified as duplicate, non-duplicate, or excluded. Precision, recall, and F1-score can then be reported together with confusion-matrix counts, while different warning thresholds can be compared. Functional testing should also cover authentication, role

separation, report validation, permitted and rejected state transitions, assignment and reassignment, evidence submission and rejection, case closure, notifications, and duplicate-warning overrides.

LIMITATIONS

The main limitation is the absence of sustained field deployment. The prototype has been exercised primarily with demonstration records, so the current material cannot establish how the system behaves with live rescue demand or how it changes response time and rescue outcomes. The duplicate-score weights and thresholds are heuristic and require calibration against labelled reports. The quality of location data, descriptions, and photographs also affects assignment and duplicate screening. SafePaws provides coordination support rather than veterinary diagnosis, emergency medical advice, or guaranteed rescuer availability. Finally, the current configuration is specific to Nagpur and would need to be reconfigured for another city.

CONCLUSION AND FUTURE WORK

SafePaws demonstrates how a centralized rescue workflow can connect the steps that are often spread across calls, messages, and social-media posts. The prototype brings together citizen reporting, precise location capture, organizational triage, rescuer assignment, controlled status changes, duplicate screening, evidence review, notifications, and role-specific summaries within one operational model. The block architecture separates user interaction, service logic, and data storage, while the rescue history provides a common record of case progress.

The six interface screenshots show a continuous path from sign-in and report creation to location capture, submission confirmation, and status tracking. The 100+ rescue cases, 40+ rescuer records, and six organization records demonstrate the configured scale of the prototype, but they do not constitute evidence of real-world rescue performance. Establishing improvements in response time, duplicate reduction, or rescue outcomes requires field data, a defined baseline, labelled duplicate-report pairs, controlled usability testing, and appropriate privacy and ethical safeguards.

Future work should prioritize calibration of the duplicate-detection method, controlled usability and performance testing, and a supervised pilot with participating animal-welfare organizations. Additional directions include offline field support, in-app communication, route optimization, multilingual interfaces, and carefully governed machine-learning assistance after sufficient operational data have been collected.

ETHICS AND DATA STATEMENT

The prototype uses demonstration records and does not include identifiable citizen, rescuer, or animal medical data. Any future study involving participants, photographs, precise locations, or treatment records should follow applicable institutional and local ethical requirements. Personally identifiable information should be minimized, access-controlled, and excluded from public rescue views unless there is an appropriate and documented basis for disclosure.

REFERENCES

1. U. Tukadiya, H. Mohan, V. Ramalingam, E. Fernandes, and F. Prakash, "PawConnect: Smart Animal Rescue and Pet Adoption System," SSRN Electronic Journal, 2026.
2. N. M. Ikhsan, M. Md-Arshad, and A. Abdul-Samad, "Design and Implementation of a Transparent and Modular Animal Welfare System Using React Native and Firebase," International Journal of Innovative

Computing, 2026.

3. K. Erin, J. Altay, and B. Sari, "Modernization of Adoption Processes in Animal Shelters: A User-Friendly Mobile Application Approach," *Journal of Smart Systems Research*, 2026.
4. Z.-Y. Wang, Y.-H. Sheu, B.-K. Yang, C.-J. Chen, and C.-W. Chen, "Intelligent TNVR Ear-Tag Recognition and Monitoring System for Stray Animal Management," *International Journal of Engineering and Technology Innovation*, 2026.
5. H. S. Efe, G. Yıldız, and Ö. Yakut, "Be My Home in Smart Cities: A Mobile Application for Stray Animal Tracking, Adoption and Shelter Management," *EJONS International Journal*, 2025.
6. Y. D. Jagtap, "Shelter Soul: Bridging Shelters and Adopters Through Technology," *arXiv Preprint*, arXiv:2506.12739, 2025.
7. L. J. F. Torres et al., "PAWrfect Match: A Web-Based Animal Adoption and Rescue System Using Content-Based Filtering Algorithm," *Library Progress International*, 2024.
8. J. E. Gatmaitan et al., "iPET: The Design, Development, and Assessment of a Web-Based Application for Pet Adoption," *Formosa Journal of Computer and Information Science*, 2024.
9. L. Y. Chan and C. S. C. Dalim, "Development of Mobile Application for Animal Conservation: Animal Rescue," *Applied Information Technology and Computer Science*, 2023.
10. M. Wee, H. Hafit, and B. Leong, "Development of Web-Based System for Animal Shelter and Rescue in Johor State," *Applied Information Technology and Computer Science*, 2022.