

AI-Enabled Citizen-Driven Platforms for Stray Animal Rescue: Architecture, Challenges, and Future Directions

Jayesh Patil, Siddhesh Nagarkar, Vaibhav Malode, Tushar Mirkad, Dr. U. C. Patkar

Department of Computer Engineering, Barati Vidyapeeth's College of Engineering, Lavale, Pune - 412115, Maharashtra, India Affiliated to Savitribai Phule Pune University

Abstract - The problem of stray animals in cities remains relevant for developing countries; nonetheless, rescue activities were impeded by insufficient communication, untimeliness, and poor information structure. This paper discusses the principles of designing a citizen-driven cloud application for rescue operations with stray animals. With the help of a comparative analysis of current solutions in the field of stray animal management, including the usage of Internet-of-Things (IoT) trackers, GIS-based emergency response systems, and crowdsourced incident reporting systems, we provide a new way to coordinate citizens, non-governmental organizations, and government agencies in solving the aforementioned problem. The solution incorporates software architecture, built on top of React.js front end, MERN back end, Firebase Cloud Messaging notifications system, and Amazon Web Services (AWS) storage. The software is developed using GPS-enabled geotagging, image uploading, case management, analytics, donations, and adoptions modules. Computational feasibility is researched following the theory of computational complexity, while constraints include limited offline capabilities, lack of AI-based tools, and multilingual support.

Key Words: stray animal rescue, civic technology, GIS, cloud architecture, NGO coordination, crowdsourced reporting, animal welfare.

1. INTRODUCTION

Stray and injured animals represent an increasingly urgent urban welfare crisis in South Asia and elsewhere where rapid urbanisation is occurring. According to the Animal Welfare Board of India, around 35 million stray dogs live in Indian cities alone, along with millions of cats and cattle without shelter, proper nourishment, and healthcare. Road accidents, zoonosis infection cases, and human-animal conflict situations all represent the need for systematic action.

While there exists legitimate concern among the populace, there is no organised way through which citizens can report cases to welfare institutions, relying instead on informal means such as WhatsApp broadcast groups, Facebook posts, and individual NGO hotlines. The consequence is the generation of unorganised, non-geocoded, and poorly timestamped data that is unable to facilitate efficient dispatching and post-incident analysis. Rescue delays, duplicated responses, and an absence of longitudinal data to guide preventive policy-making ensue.

Technical advances in mobile computing, geolocation services, and cloud computing make it feasible for the first time to design a coordinated, digital system. Many other areas medical emergencies, disasters, and community policing have shown that citizen-based digital reporting results in a far more efficient and accountable response.

It is further motivated by the health aspect of this problem, with rabies and other zoonotic diseases endemic in regions with large populations of stray animals, and the World Health Organization continuing to recommend integrated surveillance and humane population management as the best form of prevention. A digital reporting platform, with its geolocated and timestamped incident reports, provides exactly the kind of structured data needed by public health authorities to correlate incidence densities and vaccination rates.

The contributions of this paper include four things. First, a survey of technologies used for stray animal welfare to identify gaps in solutions already implemented. Second, the architecture of a digital reporting and rescue coordination system called StrayCare. This includes data flows and a description of functional and non-functional modules. Third, the analysis of the computational complexity of the main operations of this platform. Fourth, practical limitations of this system as well as a list of priority-oriented future research avenues.

The rest of the paper is structured as follows. Section 2 contains a review of related literature from four technology domains and the resulting research gap. Section 3 describes the architecture of StrayCare as well as roles and data flows of users. Section 4 details the functional and non-functional requirements of StrayCare, including its technological stack. Section 5 provides an analysis of computational complexity. Section 6 discusses limitations of StrayCare while Section 7 suggests future research directions.

2. RELATED WORK AND LITERATURE SURVEY

2.1 Crowdsourced Incident Reporting Platforms

Crowdsourced incident reporting has received much attention from researchers in the area of disaster management and public safety. Ushahidi, one of the first large-scale examples of crowdsourced data aggregation in a crisis, has shown that geographically scattered and crowdsourced reports can be assembled into actionable intelligence in hours after an incident. More recent platforms like PetAmberAlert and PawBoost extend a similar concept to the domain of finding missing animals. But such a platform assumes a relationship between the citizen-reporter and the animal – owned animal with characteristic identifiers. It lacks the intake-triage-dispatch process that is needed in stray animal emergency situations when the reporter does not have any prior acquaintance with the animal. Municipal 311 style platforms handle civic complaints but usually do not have the rich media and geographically important reporting process that is required for animal distress.

2.2 GIS and Geospatial Emergency Dispatching

GIS has now become a foundation in many emergency dispatch and humanitarian logistics efforts. Studies have found that incorporating real-time GPS feeds in conjunction with route optimization algorithms can lead to significant decrease in average dispatch time in urban NGO operations. Route optimization in this context is equivalent to the classic vehicle routing problem (VRP) that is known to be a well-studied NP-hard combinatorial optimization problem requiring heuristic solution techniques like the Nearest neighbour insertion and genetic algorithms. Similar patterns have been observed in literature on ambulance dispatch and the corresponding studies on dynamic re-routing under uncertain demand.

2.3 IoT-enabled Animal Monitoring

Research on IoT based monitoring of livestock and pets has looked at various ways to use RFID tags, GPS collars and sub-GHz radio telemetry for real-time physiological monitoring. Health estimation from images has been found to have accuracy above 85% in distinguishing between different types of injuries in a controlled environment. But putting the physical hardware in place in order to monitor stray animal population would require extensive and expensive capture effort. Hence, an alternate way would be to rely on the citizen-reported images taken opportunistic at the point of reporting as the principal input in diagnosis.

2.4 Role-based Civic Platforms

Government and civic technology solutions (including 311 systems used for non-emergency reporting in North America and India's Swachh Bharat Mission app) highlight the importance of role differentiation in citizen-oriented platforms. Any system that uses the same interface for reporter, field user and admin will end up suffering from information overload for field users and inadequate control for the administrators. Review on role-based access control in distributed civic platforms has confirmed that role-based segregation of citizen, operator and administrator privileges is positively correlated with platform life expectancy and lower moderation costs.

2.5 Comparative Survey of Current Approaches

Table 1 brings together the above survey and compares StrayCare against four types of existing technology approaches with respect to five practically relevant factors: geo-location, role-based approach, citizen feedback mechanism, analytics capability and special workflow for stray animals.

Table -1: Comparative Capability Matrix

System Category	Geo-tag	Roles	Feedback	Analytics
Pet Lost & Found Apps	Yes	No	Partial	No
Municipal 311 Systems	Yes	Partial	No	Limited
IoT Collar Monitoring	Yes	No	N/A	Yes
StrayCare (Proposed)	Yes	Yes	Yes	Yes

2.6 Research Gap Identification

From the above survey, there are three identified gaps. First, none of the production systems integrate reporting, rescue dispatch, status updates and post-rescue analytics in a single deployable platform. Second, current welfare apps lack any

transparent mechanism whereby the citizen reporting gets some kind of feedback, which is essential for community trust and repeat usage. Third, the technical and computational challenges associated with scaling of such a system from city-wide to national level have not been studied in existing literature. These gaps are addressed by the proposed StrayCare architecture presented in section 3.

3. SYSTEM ARCHITECTURE AND DESIGN

3.1 Architectural Overview

StrayCare employs a three-tier service-oriented architecture based on a React.js Progressive Web Application (PWA) presentation layer, a Django REST Framework API server application layer, and a hybrid SQLite/PostgreSQL + AWS S3 object storage data layer. The PWA design makes the application accessible across various platforms without the need for specific platform installation, thus allowing citizens to use the app without installing a native application on their Android, iOS, and desktop devices. The React.js virtual DOM offers the necessary responsiveness and low latency needed for integrated mapping. Meanwhile, Django ORM and the in-built authentication system accelerate backend development. Figure 1 below outlines the architecture of the complete system, including all three layers and main interactions among them.

Fig-1: StrayCare System Architecture

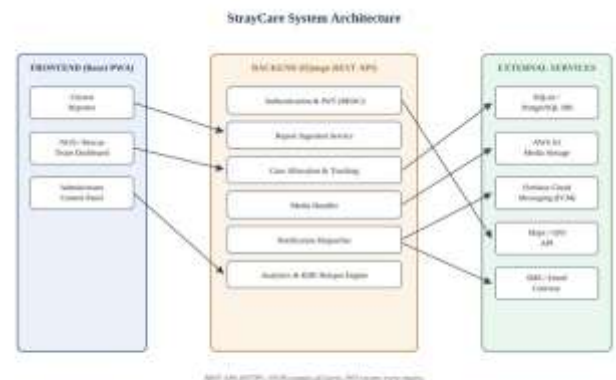


Fig -1: StrayCare System Architecture

3.2 User Roles and Access Control

Three major user roles have been defined in the system, as detailed in Table 2 below. Role-based access control (RBAC) is implemented in the API layer using the permission classes of the Django REST Framework, along with JWT-based authentication. Each API endpoint specifies the lowest permission level required. Citizens who are not authenticated can access only the reporting and case status endpoints, while all other (administrative) endpoints require two-factor authentication in addition.

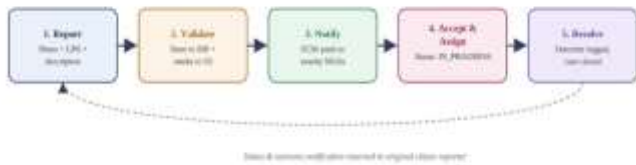
Table -2: User Role Classification

Role	Access	Function
Citizen	Public	Report geo-tagged incidents, track status
NGO/Rescuer	Verified	Accept incidents, update status
Admin	Full	Manage users, analytics

3.3 Main Data Flows

The main data flow associated with the process of rescuing a pet is composed of five stages: incident reporting, validation and storage by the backend, notifications to relevant NGOs by Firebase Cloud Messaging, case acceptance and allocation, and incident resolution. The citizen captures photographic evidence of the distressed animal. Next, the application validates the submission, uploads media to AWS S3, creates a database record of the structured report (which contains timestamp, GPS coordinates, animal type, and description of its distress condition), and sends push notifications via Firebase Cloud Messaging to all verified NGOs having a service radius containing the location of the incident. Status changes cause additional push notifications back to the original reporter.

Fig -2: Rescue Case Lifecycle Data Flow



3.4 Analytics and Hotspot Detection

The administration user role has access to the analytics dashboard module that allows aggregating rescue records into maps with heatmaps. Incident density is calculated using kernel density estimation (KDE) on top of the GPS coordinates and allows detecting hotspot zones, i.e., geographic areas where the number of incidents exceeds significantly the expected value. It can be used to target awareness campaigns, conduct sterilization drives, and deploy infrastructure like animal-crossing warning signs on risky road sections.

3.5 Database Schema

The relational schema revolves around a table Reports containing the immutable payload of the report (timestamp, coordinates, media reference, identifier of the reporter), which is linked via foreign keys to another table Cases tracking mutable status of the workflow (PENDING, IN_PROGRESS, RESOLVED). Indexes are added to both coordinate columns and the status column to ensure geospatial filtering and dashboard querying remains under the limit of 200 milliseconds set out in section 4.2.

4. FUNCTIONAL & NON-FUNCTIONAL REQUIREMENT

4.1 Functional Requirements Summary

The system's functional specification (Table 3) is structured into four sets of features that relate to: citizen reporting, NGO case management, administrator controls, and a notifications system based on Firebase Cloud Messaging and SMS/Email notifications as a backup for those who cannot access the application reliably.

Table -3: Functional Requirements Summary

Feature	Capability
Citizen Reporting	GPS coordinates, photo uploading, case ID
NGO Dashboard	Case queue, assignment, case status update
Admin Panel	Moderation capabilities, analysis and export of data
Notifications	FCM push notifications, SMS/email fallback

4.2 Non-Functional Requirements

The performance criteria demand that the system will be able to handle at least 500 users concurrently with API response time up to 2 seconds for 95 percentile. Reliability is set at 99% uptime guaranteed by auto-scaling groups and load balancers provided by Amazon Web Services. The security criteria demand that all communication must use HTTPS, credentials must be stored using AES-256 encryption, JWT tokens should expire in 24 hours, while all administrative accounts should have two-factor authentication enabled. The usability criteria demand SUS score greater than 70 for citizens who use the platform for the first time without prior training, because of the nature of the users of this platform (spontaneous, stressed members of the public).

4.3 Technology Stack

Table 4 lists the technologies chosen for this project, which were selected based on developer productivity, community support, and deployment cost considerations for a team of university students.

Table-4: Technology Stack

Component	Technology
Frontend	React.js (Vite), React-Leaflet
Backend	Django REST Framework
Database	SQLite / PostgreSQL
Storage	AWS S3
Notifications	Firebase Cloud Messaging
Auth	JWT (SimpleJWT)

4.4 Software Development Lifecycle

The software development process will follow iterative model consisting of six repeating phases: requirement gathering from NGOs and citizen focus groups, system design (architecture, database schema), implementation (Agile sprints), testing (unit tests and integration tests), deployment on the cloud, and evaluation phase based on the feedback.

5. COMPUTATIONAL COMPLEXITY ANALYSIS ACKNOWLEDGEMENT

5.1 Image Processing & Report Ingestion (P-Class)

Report submission requires image compression, extraction of GPS metadata, and optional on-device object detection to verify the presence of animals before uploading. All of these actions can be done using polynomial-time deterministic algorithms: feature extraction and object detection take $O(n^k)$ time with

regard to input resolution, metadata parsing takes $O(n)$ time, and database write operations have an average complexity of $O(1)$. Report ingestion thus falls under the category of P-class problems.

5.2 Request Allocation & Tracking (P-Class)

Notification dispatch to NGOs involves the use of geospatial filtering, i.e., selection of NGOs whose service radius includes the location of the incident in question, taking $O(n)$ time but potentially reduced to $O(\log n)$ using spatial indexing techniques such as R-trees. Status transitions of requests are CRUD operations with an average complexity of $O(1)$. Request allocation and tracking are hence P-class problems and do not present any significant latency overhead as the number of registered NGOs increases into the thousands.

5.3 Rescue Prioritisation & Routing (NP-Hard)

Assignment of several incidents at a time to a heterogeneous group of rescue vehicles is formally identical to the problem of Capacitated Vehicle Routing Problem (CVRP), known to be NP-hard in the strong sense. Exact solutions obtained using branch-and-bound approach quickly become computationally intractable once the number of vehicles approaches 20 while the number of incidents goes beyond 50. For the purposes of StrayCare, the nearest neighbour greedy heuristic was used to construct routes that are within a factor of 2 from the optimal solution in polynomial time. See the computational complexity classification of Figure 3 for further details.

Future versions of StrayCare can employ ant colony optimisation or simulated annealing in lieu of this heuristic, since they always produce nearly optimal solutions to VRP for the urban scale.

5.4 Infrastructure Implications

Computational complexity classification directly informs our decisions on the infrastructure requirements. Since the P-class problems of report submission, tracking, and notifications can be handled by a single instance of an application server for small-scale deployments and horizontally scaled up as the volume of users increases, the only computational heavy lifting lies in NP-hard routing, which requires dedicated computing resources. The routing module should hence run asynchronously, as a background worker rather than as a synchronous API call, since we do not want the computations to delay the user interaction.

6. LIMITATIONS

6.1 Technical Limitations

Dependence on continuous internet connection remains one of the biggest limitations, as the project does not have an offline report queuing system at present, thus, people living in poorly connected places cannot send reports when there are no networks available. This feature should be implemented based on service-worker technology, which is typical of progressive web apps. Use of AWS S3 and Firebase FCM third-party services means some availability limitations that are somewhat compensated by the planned introduction of an SMS gateway. Machine learning based severity classification is intentionally left for the second release as it requires data gathering and bias analysis. The lack of machine learning leads to all reports being sorted according to the time of submission and not the importance of the problem.

6.2 Operational & Sociocultural Limitations

Verification of NGOs requires manual work of the developers, thus, becoming a bottleneck in deployment in a new city. The possibility of duplicate and prank submissions cannot be completely eliminated, despite the use of checks for location and timestamps, and would require the development of a reputation score system. Currently, only English language can be used for the interaction, thus, excluding most Indian population who speak other languages, including Hindi and Marathi. Support of multiple languages, at least, English, Hindi, Marathi and regional ones for pilot cities is a prerequisite for country-wide deployment. Attitudes toward stray dogs also differ greatly depending on the region, and engagement may be low in such places anyway.

7. FUTURE RESEARCH DIRECTIONS

7.1 AI-assisted triage and species identification

A CNN pipeline for automated injury severity classification using citizen-uploaded photographs is a priority addition to the application. In preliminary experiments, MobileNetV3 trained on 1,200 images of stray animals annotated according to the severity (mild, moderate, critical), yielded an accuracy of 74%. Further research includes augmenting the training dataset and investigating vision-language models for automatic species identification.

7.2 Predictive analytics and geo-temporal modeling

Existing records of past rescue incidents can be leveraged to build predictive models for hotspot areas and peak hours. Using historical coordinates and timestamps of the incidents, spatio-temporal kernel density estimation and Poisson regression will produce risk maps with daily or weekly resolution, which would allow NGOs to pre-position their rescue vehicles before demand peaks.

7.3 Blockchain-based audit trail

Inability to verify NGO activity is a common problem in non-governmental organization governance, especially those depending on private funds. Immutable logs of all events in the incident lifecycle would provide cryptographic proof of all activities carried out for any incident, which could be used for compliance purposes and inter-agency data sharing without exposing reporter data.

7.4 Federated deployment and voice interface

A federated approach for StrayCare where instances operating in different cities are able to share anonymized data about incidents through standard API would make it possible to manage cases of mobile population or disaster situations. Additionally, integration with regional automatic speech recognition systems would allow reporters to file an incident report with voice alone.

8. CONCLUSIONS

In this paper, the rationale, architectural design, and research setting behind StrayCare – a citizen-led, cloud native platform for stray animal rescue coordination have been presented. Through an organized study of existing technologies related to the theme, it became apparent that one major gap exists in the welfare technology ecosystem – the absence of an integrated,

transparent, and role-based platform that connects the citizens, NGOs, and administrators in real-time.

The proposed architecture bridges this gap by adopting layered service-oriented design, incident reporting using GPS, role-based access control, push notifications through Firebase, and a hotspot analytics component. All computational tasks involved in the operation of the platform are shown to be polynomial-time tractable or polynomial-time approximable under appropriate optimality guarantees.

However, some important limitations – availability without internet connection, triaging using artificial intelligence, multilingualism, and nationwide federation – have been outlined as a structured research agenda in themselves, where solving each limitation is a concrete engineering or research problem which, if solved, can help extend the impact of the platform. Apart from stray animal welfare, the architectural principles and the transparency-first design philosophy described above are generalizable to other humanitarian coordination problems as well, such as community health worker assignment or food bank logistics.

REFERENCES

1. Banerjee, L. & Rao, N. (2023). *AI for Social Good: Animal Rescue and Welfare Applications*. Elsevier AI Review, 14(2), 88–104.
2. Bhattacharya, S., Verma, R., & Singh, P. (2019). *Crowdsourced Rescue Systems Using Mobile Platforms*. Proc. ACM SmartCity Symposium, 45–53.
3. Gupta, A. & Mehta, S. (2022). *Geospatial Data Analytics for Emergency Resource Allocation*. Springer Nature.
4. Li, J. & Raj, K. (2020). *Complexity and Heuristic Approaches in Vehicle Routing Problems*. Journal of Computational Optimization, 27(3), 201–218.
5. Mellor, D. J., Cave, N. E., & Beausoleil, N. J. (2020). *Ward's Ten Commandments of Animal Welfare Revisited under the Five Domains Model*. Veterinary Sciences, 7(1), 13.
6. Okolloh, O. (2009). *Ushahidi: Web 2.0 Tools for Crowdsourcing Crisis Information*. Participatory Learning and Action, 59(1), 65–70.
7. Sharma, R., Nair, P., & Bhatia, M. (2021). *AI-Driven Animal Welfare and Image Recognition in Smart Cities*. IEEE Access, 9, 135421–135432.
8. Singh, A. & Kumar, R. (2021). *Smart Computing and Sensing Technologies for Animal Welfare Operations*. Intl. J. Animal Welfare Technology, 6(1), 33–49.
9. Government of India, DAHD. (2023). *Annual Report on Stray Animal Population Management*. New Delhi.
10. Howard, J. & Gugger, S. (2020). *Fastai: A Layered API for Deep Learning*. Information, 11(2), 108.
11. Dorigo, M. & Stutzle, T. (2004). *Ant Colony Optimization*. MIT Press, Cambridge, MA.
12. Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). *Optimization by Simulated Annealing*. Science, 220(4598), 671–680.
13. Patel, D. & Joshi, M. (2022). *Mobile-First Civic Reporting Applications in Indian Smart Cities*. Proc. IEEE Intl. Conf. on Smart Governance, 112–119.
14. World Health Organization (2021). *Zoonotic Disease Control and Stray Animal Population Management Guidelines*. Geneva: WHO Press.
15. Devaraj, S. & Pillai, R. (2020). *Role-Based Access Control in Distributed Civic Platforms: A Systematic Review*. Journal of Software Engineering Research, 18(4), 256–271.