



Smart Public Infrastructure Damage Reporting and Prediction System: A Web-Based Platform with Category-Based Rule Priority Classification

Khushi¹, Harsha Bhat T.M², Varun P³, Shreecharan I G⁴, Prof. Raksha K⁵

Student, Department of Computer Science and Engineering, Vivekananda College of Engineering and Technology¹⁻⁴

Guide, Department of Computer Science and Engineering, Vivekananda College of Engineering and Technology⁵

Abstract: Potholes, damaged roads, faulty streetlights and water leakage often remain unresolved because citizens lack a simple reporting channel and authorities lack consolidated data. This paper presents a web-based Smart Public Infrastructure Damage Reporting and Prediction System built with HTML, CSS, JavaScript, PHP and MySQL. Citizens register with OTP verification, submit geotagged complaints with photographs, and track status; administrators manage status, priority and department assignment, and view complaints on an interactive map and analytics dashboard. Source-code tracing shows priority is assigned by a transparent category rule (High for Water Leakage, Pothole and Traffic Signal Failure, otherwise Medium), while several interface features labeled “AI” are fixed values or dormant code. Functional testing of ten cases passed. The paper documents the system as built and outlines steps to close this gap.

Index Terms: infrastructure complaint reporting, civic technology, rule-based prioritization, map visualization, web application, PHP, MySQL.

I. INTRODUCTION

Many public infrastructure defects, such as potholes, damaged roads, faulty streetlights, clogged drains and leaking pipes, reach a municipal office only when a citizen reports them by phone, letter or visit. Reports handled this way are hard to track and aggregate, so defects stay unresolved and officers have no consolidated view of where problems concentrate. Mobile crowdsourcing has been surveyed as a way for cities to obtain citizen-generated data [5], but officers also need a defensible way to decide where to act first. Learning-based urgency estimation [9] needs labeled data and is hard for a small office to audit, and complaint data are biased by uneven reporting behavior [4]. A prioritization method for local bodies should therefore be simple and transparent, and a paper describing it should state the rule that is actually implemented.

Manual complaint handling by phone or office visit gives citizens no way to follow progress and gives authorities no single record of what has been reported, where, and how often. The project therefore pursues four objectives: (i) a simple web channel through which citizens can register, describe a problem, mark its location and upload a photograph; (ii) a central database and administrator dashboard for reviewing, filtering, categorizing and updating complaints; (iii) status tracking so that citizens can follow each complaint from submission to resolution; and (iv) a transparent rule for deciding which problems to handle first, with graphs and a map to show where complaints concentrate.

This paper presents a web-based platform connecting citizens and municipal administrators. It lets citizens report damage with a photograph, location and category and track their complaints; lets administrators manage complaints and view them on a map and dashboard; and assigns each complaint an initial priority through a transparent rule. The contributions are an end-to-end platform, a source-code-level account of how priority is assigned and displayed, a functional evaluation, and an analysis of the gap between the interface’s “AI” labeling and the mechanism actually implemented.

II. LITERATURE REVIEW

Several recent systems provide web or mobile channels for reporting local problems: a civic issue reporting and tracking application [1], CivicEye, an app for civic issue reporting and monitoring in urban governance [2], and a web-based grievance redressal system with image and geolocation support [3]. These concentrate on submission, tracking and map display rather than on how priority is decided. Other work detects or collects road defects through citizens’ smartphones: CommuniSense gathers road-hazard reports in Nairobi [10], a crowdsourced petitioning system detects potholes on Android [8], and sensor-based patrolling monitors road surfaces [6]. These rely on device sensing and focus on roads, whereas the present system covers several damage categories reported manually with photographs.



On prioritization, urgency has been estimated by mining emotions in citizen text [9], which needs labeled data and is hard to inspect. Participation in crowdsourced reporting is uneven [7], and disparities in 311 complaint behavior affect the fairness of data-driven decisions [4], which argues for rules simple enough to be audited. Small municipal systems that state exactly which mechanism assigns priority receive little attention; this paper describes one such rule precisely. In addition to complaint reporting and tracking, effective infrastructure management requires a systematic method for evaluating reported issues and identifying areas that need attention. This helps prioritized maintenance based data.

III. METHODS AND MATERIALS

A. Architecture and Workflow

The system is a three-tier web application (Fig. 1). The presentation layer (HTML, CSS, JavaScript, Leaflet.js, Chart.js) provides the citizen portal, administrator dashboard, and map and analytics views. The application layer (PHP on Apache) handles authentication, validation, complaint management, priority assignment and email notification (PHPMailer 7.0.2, Gmail SMTP). The data layer is a MySQL database plus stored images. As the use cases in Fig. 2 show, a citizen registers, verifies a one-time password (OTP), logs in and submits a complaint with a category, description, location and image. The system validates it, assigns an initial priority and stores it; the citizen tracks its status (workflow in Fig. 3). The administrator reviews complaints and updates their status, priority and assigned department. Functionally the system is organized into six modules: a Citizen module (registration, login, submission and tracking), an Administrator module (review, filtering and updating), a Complaint Management module (validation, storage and status handling), a Database module (central MySQL storage), a Prediction module (priority assignment) and a Visualization module (charts and the map).

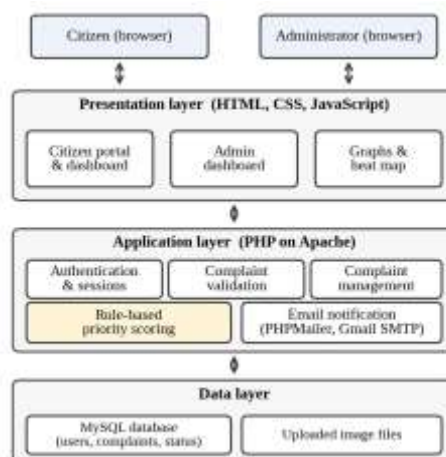


Fig. 1. Three-tier architecture of the system.

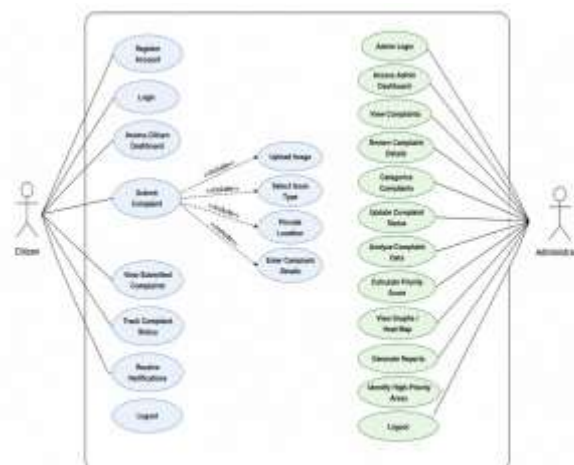


Fig. 2. Use cases of the Citizen and Administrator.



B. System Requirements

The system has two user roles. Citizens register, log in, submit complaints with an image and location details, view their complaints and track their status. Administrators log in to a protected dashboard to view, filter, categorize, update and analyze complaints, and to view complaint images and locations. Non-functional requirements set out in the project report are reliability (complaints must not be lost by database operations), fast response for login, submission, retrieval and dashboard loading, security (authorization before protected modules, session-based access control, input checking and restricted administrator access), scalability of the database design, an interface usable without technical skills, compatibility with common browsers on Windows or Linux, and independent modules for maintainability.

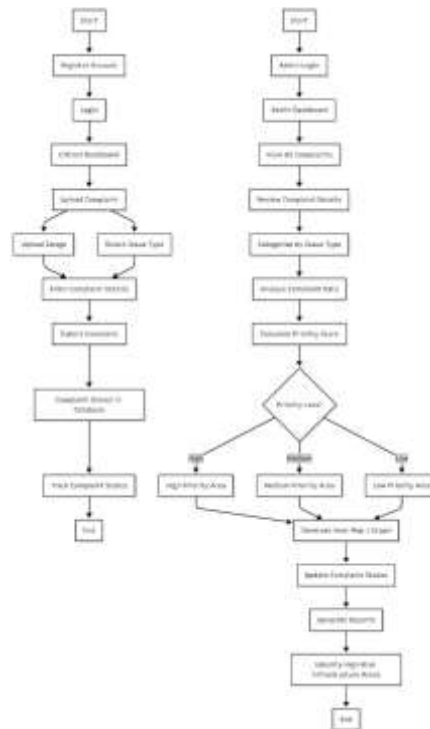


Fig. 3. Workflow of the proposed system.

The software stack is HTML, CSS and JavaScript on the client; PHP on the server for authentication, complaint handling, status updates and email; MySQL for users, complaints, statuses, locations, images and category data; PHPMailer with Gmail SMTP for notifications; and Visual Studio Code, XAMPP (Apache and MySQL) and a web browser for development. No specialized hardware is needed: the recommended minimum is a laptop with an Intel Core i3-class processor, 4 GB of RAM, 10 GB of free storage and a display of at least 1366×768 pixels, with a smartphone used optionally to check responsiveness.

C. Complaint Data and Validation

Each complaint stores a receipt ID, citizen, title, category, description, address, GPS latitude and longitude, image, status, priority, a free-text department label, a confidence value and a timestamp. Before saving, the server checks that required fields and coordinates are present and that any image has an allowed type and size; a failing submission is stored as Blocked with the reason. A valid complaint moves through Submitted, Accepted, Under Review, Assigned, In Progress and Resolved, and every transition is logged. Of the 15 tables in the schema, five are used by the PHP code (users, complaints, complaint_status_history, notifications, repair_images); the rest, including the department, assignment, feedback, session and security-event tables, are defined but never read or written. Registration requires a 4-digit email OTP stored with its expiry on the user record, and authentication uses native PHP sessions.

D. Priority Assignment

Priority is assigned by a rule in the submission handler (php/complaints.php), mirrored in the client-side fallback: if the category is Water Leakage, Pothole or Traffic Signal Failure the priority is High; otherwise it is Medium (Table I). The rule needs no training data and is fully auditable. The system supports three priority levels (High, Medium and Low), but “Low” is never produced by the rule; an administrator sets it, or changes any other value, through a separate priority-update endpoint. No area, ward or population information is used.



TABLE I CATEGORY-TO-PRIORITY MAPPING (AS IMPLEMENTED)

Priority	Categories
High	Water Leakage, Pothole, Traffic Signal Failure
Medium	Road Damage, Broken Streetlight, Drainage, Broken Footpath, Garbage Accumulation, Public Property Damage, Other
Low	Not assigned automatically; set only by an administrator through the priority update

The project report additionally specifies a planned area-level extension that is not part of the code described above. Each issue type would carry a weight (Road Damage 5, Water Leakage 4, Drainage 4, Street Light 3, Garbage 2) and each severity level a weight (High 3, Medium 2, Low 1); the weights of all complaints in an area would be summed into an area score, and the area classified High (80 or more), Medium (40 to 79) or Low (below 40). Under these weights a single complaint contributes between 3 and 8 points, so an area would need at least five complaints to reach Medium and ten to reach High. This design would make the priority depend on complaint frequency and severity as well as category, and it is the main direction for improving the current rule.

E. Visualization and Evaluation Method

A Leaflet.js map plots each complaint as a colored circle at its GPS position (red for High priority or Pothole/Water Leakage, orange for Medium or In Progress, green otherwise); one circle is drawn per complaint, so no density or clustering is computed. A Chart.js dashboard shows category distribution, a monthly trend and a status pipeline. Testing aimed to confirm that every module works as specified, that citizens can register, submit and track complaints, that data and images are stored correctly, and that administrators can authenticate and manage complaints. It covered unit, integration, functional, system and acceptance levels and was source-code-verified: for each module the PHP or JavaScript handler was read to confirm which table or logic path actually executes. Response time and load were not measured.



Fig. 4. Complaint submission screen (citizen).

IV. RESULTS AND DISCUSSION

A. Functional Test Results

All ten test cases in Table II produced the expected outcome for the mechanism actually implemented.

TABLE II FUNCTIONAL TEST CASES AND RESULTS

ID	Scenario	Expected outcome	Result
1	Registration with valid details	Pending account; OTP email sent	Pass
2	OTP verification with correct code	Account activated	Pass
3	Citizen login (verified)	Citizen dashboard shown	Pass



ID	Scenario	Expected outcome	Result
4	Submit complaint with image, category, location	Validated, stored, receipt ID issued	Pass
5	Track complaint status	Current status and history shown	Pass
6	Administrator login	Admin Command Center opens	Pass
7	Admin updates status, priority, department	Row and status history updated	Pass
8	Priority for the three High categories vs. others	High for the three; Medium otherwise	Pass
9	Map and analytics display	Complaints plotted and charted	Pass
10	Submission with missing required fields	Stored as Blocked with a reason	Pass

B. Gap Between “AI” Labeling and Implemented Behavior

Table III compares each interface feature described as AI-driven with what its source code does. In each case the described mechanism is dormant code, a fixed value or a simple rule. The core workflow and the priority rule work end to end, but the system should not currently be described as using image classification, automatic duplicate detection or predictive risk modeling.



Fig. 5. Analytics dashboard (administrator).

TABLE III INTERFACE “AI” LABELS VERSUS TRACED IMPLEMENTATION

Labeled feature	Interface suggests	Source code actually does
“AI Vision Analysis” / confidence	Image analyzed, confidence estimated	Fixed constant sent from client; lookup in ai-engine.js never called
Duplicate detection (“System AI Validator”)	Nearby duplicates blocked	Correct Haversine/50 m function exists but is not invoked



Labeled feature	Interface suggests	Source code actually does
“AI Risk Predictions”	Backend ranks high-risk zones	Queries a predictions table absent from the schema
“Predicted Recurrence Risk” (export)	Forecast of recurrence	Keyword match on category name at export time
Density heat map	Spatial clustering	One fixed-radius circle per complaint, colored by priority

C. Limitations

Testing was functional and small-scale; the system has not been deployed, no user study was run, and performance was not measured. The quality of analysis depends on how accurate and complete the citizen-provided details and locations are, and the system cannot tell that a problem has been repaired unless an administrator updates the complaint status. Priority depends only on category, so complaints in the same category receive the same priority regardless of severity, location or nearby report counts, and complaint volume reflects who reports, not only where damage occurs [4]. The “AI” labels should be corrected before the system is described as AI-based, and the legacy admin/user tables that store plaintext passwords should be retired before real data is handled.

D. Discussion

Compared with phone calls and office visits, the platform gives citizens one place to report and follow a complaint and gives authorities one central record of what has been reported, where and how often. Status tracking makes progress visible to the citizen, and the administrator dashboard, charts and map help officers see which categories and places generate the most reports. The priority rule is deliberately simple: it is easy to explain, needs no training data and can be changed by editing one list, which suits a small municipal deployment. Its simplicity is also its weakness, because it ignores severity, frequency and location, which is why the weighted area-level score in the project report is the most useful next step.

V. CONCLUSION

This paper presented a web-based platform for reporting public infrastructure damage and managing complaints. Citizens can report and track problems, administrators can manage them on a central dashboard, and a map and charts show where complaints concentrate. Tracing the source code shows that priority is assigned by a small, auditable category rule, and that several “AI” features in the interface are dormant code, fixed values or keyword rules. All ten functional tests of the implemented mechanisms passed.

Future work follows the project report and the findings above: a mobile application for citizens; implementing the weighted area-level score and extending it with complaint frequency, population affected and complaint age; ward-wise and district-wise heat map analysis with comparison against historical data; real-time notifications to citizens and alerts to administrators for high-priority complaints; automatic assignment of complaints to the responsible department by category; larger-scale analytics with cloud storage; wiring in the duplicate check and replacing the fixed confidence value with real image classification, or relabeling the dashboard; and a field pilot with a local body.

REFERENCES

- [1]. S. Maskar, P. Patil, R. Chavare, V. Bharmal, and A. J. Shinde, “Civic issue reporting and tracking,” *Int. J. Adv. Res. Comput. Commun. Eng. (IJARCCE)*, vol. 15, no. 4, 2026.
- [2]. Bhuvad, H. Bharnuke, D. Koli, and S. B. Patil, “CivicEye: A smart civic issue reporting and monitoring app for urban governance,” *Int. J. Versatile Res. Anal.*, 2026.
- [3]. “A web-based public grievance redressal system with image and geolocation support,” in *Proc. 2025 Int. Conf. Smart Sustain. Technol. (INCSST)*, 2025.
- [4]. C. E. Kontokosta and B. Hong, “Bias in smart city governance: How socio-spatial disparities in 311 complaint behavior impact the fairness of data-driven decisions,” *Sustain. Cities Soc.*, vol. 64, Art. no. 102503, 2021.
- [5]. X. Kong, X. Liu, B. Jedari, M. Li, L. Wan, and F. Xia, “Mobile crowdsourcing in smart cities: Technologies, applications, and future challenges,” *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8095–8113, 2019.



- [6]. G. Singh, D. Bansal, S. Sofat, and N. Aggarwal, "Smart patrolling: An efficient road surface monitoring using smartphone sensors and crowdsourcing," *Pervasive Mob. Comput.*, vol. 40, pp. 71–88, 2017.
- [7]. B. Pak, A. Chua, and A. Vande Moere, "FixMyStreet Brussels: Socio-demographic inequality in crowdsourced civic participation," *J. Urban Technol.*, vol. 24, no. 2, pp. 65–87, 2017.
- [8]. F. Enigo, T. Vinothkumar, S. Vijay, and K. G. Prabu, "Crowdsourcing based online petitioning system for pothole detection using Android platform," *Procedia Comput. Sci.*, vol. 87, pp. 316–321, 2016.
- [9]. C. Masdeval and A. Veloso, "Mining citizen emotions to estimate the urgency of urban issues," *Inf. Syst.*, vol. 54, pp. 147–155, 2015.
- [10]. D. Santani, J. Njuguna, T. Bills, A. W. Bryant, R. Bryant, J. Ledger, and D. Gatica-Perez, "CommuniSense: Crowdsourcing road hazards in Nairobi," in *Proc. 17th Int. Conf. Human-Comput. Interact. Mobile Devices Serv. (MobileHCI)*, 2015.