WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal Homepage: www.ijrpr.com ISSN 2582-7421

AI Pothole and Road Hazard Reporter: An AI-Assisted Web Platform for Pothole Detection, Prioritised Complaint Management and Government Routing

Ms. Gayathri V, Subiksha S, Subitha S, Sudalai Muthu G, Sugitha G

Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology , Coimbatore, Tamilnadu, India

A B S T R A C T

Poor road conditions such as potholes, damaged surfaces, waterlogging and fallen electrical infrastructure endanger road users, yet citizens often lack a simple, trackable way to report them to the right authority. This paper presents AI Road Hazard Reporter, a full-stack web platform that lets citizens report road hazards with photographic evidence, receive an AI-based assessment and follow the complaint until it is resolved. A fine-tuned YOLOv8 model detects potholes in uploaded images and estimates severity from detection confidence and bounding-box area; when the model is unavailable, the system says so instead of producing a fabricated score. Complaints cover sixteen hazard categories, are ranked by a transparent priority score and are matched to a government department by a three-level jurisdiction-aware routing engine backed by a directory of verified contacts for Kerala and Tamil Nadu. A formal complaint letter is generated automatically for every report. Role-based access for citizens, officers and administrators, a twelve-state complaint lifecycle, notifications, citizen feedback with reopening, a public hazard map and an analytics dashboard complete the workflow. The backend is built with FastAPI, SQLAlchemy and PostgreSQL, and the frontend with React, TypeScript and Leaflet. The paper describes the architecture, algorithms and implementation, and discusses current limitations and directions for future evaluation.

Keywords: Pothole detection; YOLOv8; Road hazard reporting; Complaint management; Jurisdiction-aware routing; FastAPI; React

1. Introduction

Road infrastructure defects such as potholes, cracked and uneven surfaces, blocked drains that cause waterlogging, open manholes, fallen trees and damaged electrical lines are a continuing threat to the safety of pedestrians, two-wheeler riders and drivers. Repairing them quickly depends on two things: the authorities must learn that the hazard exists and where it is, and the report must reach the department that is actually responsible for it.

In practice, citizens report hazards through telephone helplines, written applications or generic grievance portals. These channels often expect the citizen to know which department owns the problem, give no objective evidence of severity, and offer little visibility once a complaint has been filed. Authorities, in turn, receive unstructured reports that are hard to verify and prioritise. Advances in deep-learning-based object detection (Redmon, Divvala, Girshick, & Farhadi, 2016) now make it practical to analyse a photograph at the moment of reporting, so that an ordinary photo can be turned into a structured, verifiable and prioritised complaint.

This paper presents *AI Road Hazard Reporter*, a web platform that combines image-based pothole detection, a structured complaint lifecycle and jurisdiction-aware routing in a single workflow for citizens and civic officers. The main contributions are:

An AI-assisted reporting flow in which a YOLOv8 model analyses the uploaded photograph and returns a detection confidence and an estimated severity, with an explicit fallback state when the model cannot be used.

A taxonomy of sixteen road-hazard categories with a transparent, rule-based priority score that also accounts for community confirmations.

A three-level routing engine that recommends the responsible government contact for a given hazard, state and district, together with an automatically generated formal complaint letter.

A role-based complaint lifecycle with twelve states, notifications, citizen feedback with reopening, a public hazard map and an officer analytics dashboard.

The rest of the paper is organised as follows. Section 2 reviews related work, Section 3 describes the system architecture, Section 4 explains the methodology, Section 5 covers the implementation, Section 6 discusses results and limitations, and Section 7 concludes the paper.

2. Related work

Research on automated road-defect monitoring follows two broad directions. Sensor-based approaches mount accelerometers and GPS units on vehicles to detect rough road surfaces; the Pothole Patrol system is an early example that used instrumented taxis for this purpose (Eriksson et al., 2008). Vision-based approaches analyse images instead. Maeda, Sekimoto, Seto, Kashiyama, and Omata (2018) trained deep neural networks on smartphone images to detect and classify several kinds of road damage, showing that photographs taken with ordinary consumer devices are sufficient for the task.

Single-stage detectors of the YOLO family (Redmon et al., 2016) are attractive for citizen-facing applications because they are fast enough for interactive use. The Ultralytics YOLOv8 implementation (Jocher, Chaurasia, & Qiu, 2023) provides a compact nano variant that can be served from a modest backend, and pothole models fine-tuned from it have been shared publicly (Keremberke, n.d.).

Most of this literature concentrates on detection accuracy. The downstream questions of how a detected hazard becomes an actionable complaint, which department should receive it, how urgent it is and how its repair is tracked receive less attention, and general-purpose grievance portals generally rely on the citizen to choose the category and the department manually. The present work addresses this gap by embedding detection inside a complete reporting, routing and tracking workflow.

3. System architecture

3.1 Overview

The system follows a three-tier architecture (Fig. 1). A React single-page application provides the user interface for citizens, officers and administrators. It communicates with a FastAPI backend (Ramírez, 2018) through a REST/JSON interface secured with JSON Web Tokens (Jones, Bradley, & Sakimura, 2015). The backend is organised into route modules for authentication, complaints, contacts and routing, notifications and departments, which delegate to dedicated services for AI inference, priority calculation, routing, letter generation and file storage. Persistent data are kept in a relational database through SQLAlchemy, uploaded evidence images are stored on local disk or optionally on AWS S3, and the YOLOv8 weights are loaded once at start-up and reused for every request.

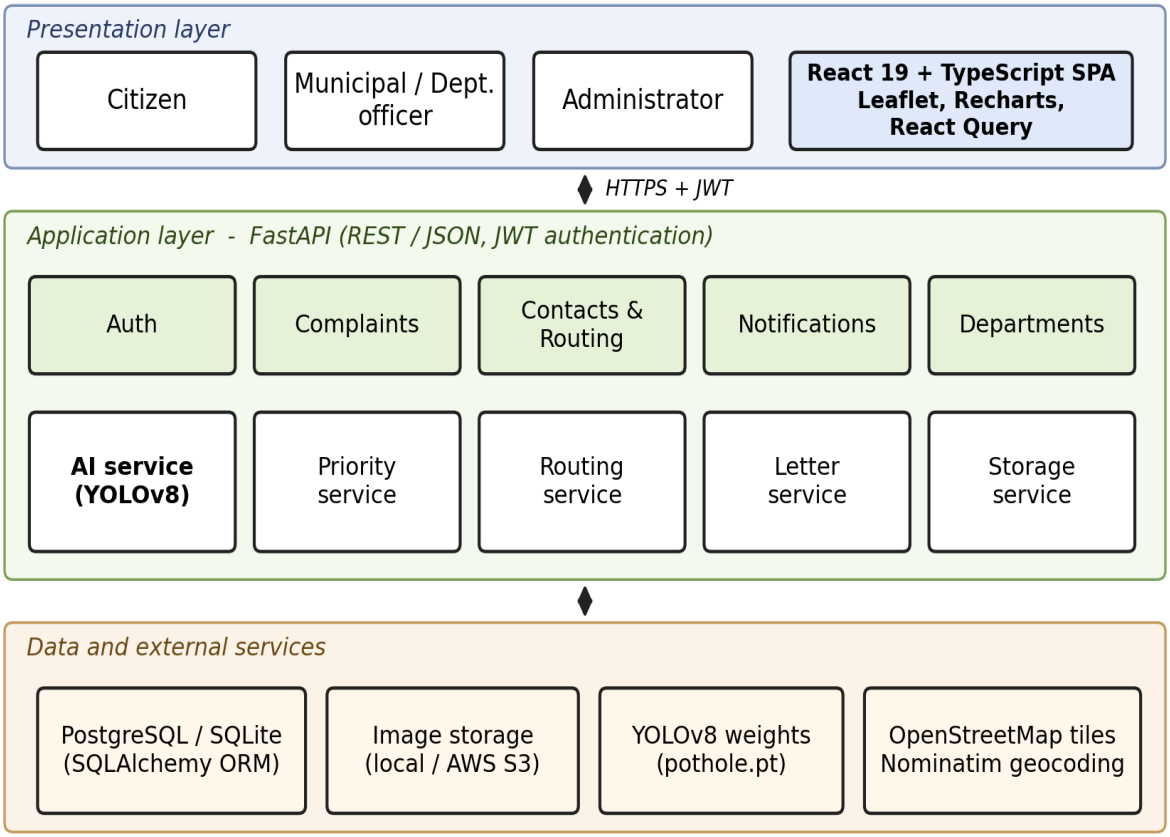


Fig. 1 - System architecture of AI Road Hazard Reporter.

3.2 Technology stack

Table 1 lists the main technologies used in each layer of the system.

Table 1 - Technology stack.

Layer	Technology	Purpose
Frontend	React 19, TypeScript, Vite, Tailwind CSS 4	Single-page application and responsive user interface
Frontend state	React Router, React Query, Zustand, React Hook Form, Zod	Routing, server-state caching, authentication state, validated forms
Mapping	Leaflet, react-leaflet, OpenStreetMap, Nominatim	Hazard map, location picker, reverse geocoding
Analytics	Recharts	Bar and pie charts on the officer dashboard
Backend	FastAPI, Uvicorn, Pydantic	REST API and request/response validation
Persistence	SQLAlchemy, PostgreSQL (SQLite for local development)	Object-relational mapping and data storage
Security	JSON Web Tokens (python-jose), bcrypt (passlib)	Token authentication and password hashing
AI	Ultralytics YOLOv8, PyTorch, OpenCV	Pothole detection and bounding-box extraction
File storage	Local disk, AWS S3 (optional)	Storage of evidence images

3.3 Data model

The relational schema centres on the *Complaint* entity, which stores the citizen-selected category and severity, the AI-suggested category, confidence and severity, the GPS coordinates and address fields, the priority score, the critical flag, the community confirmation count and the assigned department and repair team. Supporting entities include *User*, *ComplaintImage*, *StatusHistory*, *Department*, *RepairTeam*, *ComplaintLetter*, *Notification* and *Feedback*. A separate government-directory subsystem stores *GovContact* records (department name, contact type, state, district, phone, e-mail, website, source URL and verification date), *JurisdictionMapping* rules and *ForwardingAttempt* logs, so that a recommended contact can be traced back to an official source.

3.4 User roles

Five roles are defined: citizen, municipal officer, department officer, repair team and administrator. Citizens can create and view only their own complaints. Officers (municipal, department and administrator roles) can view all complaints and update their status, department and repair-team assignment. Only administrators can create, edit or delete entries in the government contact directory. Demo accounts for the citizen, municipal-officer and administrator roles are created automatically on first start-up for evaluation.

4. Methodology

4.1 Complaint submission workflow

Reporting is a three-step wizard (Fig. 2). In the first step the citizen uploads one or more photographs (JPEG, PNG, WebP or GIF, at most 10 MB each) and may request an AI analysis. In the second step the location is fixed either from the device GPS or by placing a pin on an OpenStreetMap-based map, and the address fields are filled by reverse geocoding through Nominatim. In the third step the citizen confirms the hazard category, severity and description and submits the complaint. On submission the backend stores the images, runs the AI analysis on the first image, computes the priority score, assigns the default department for the chosen category, writes the first entry of the status history and notifies the citizen.

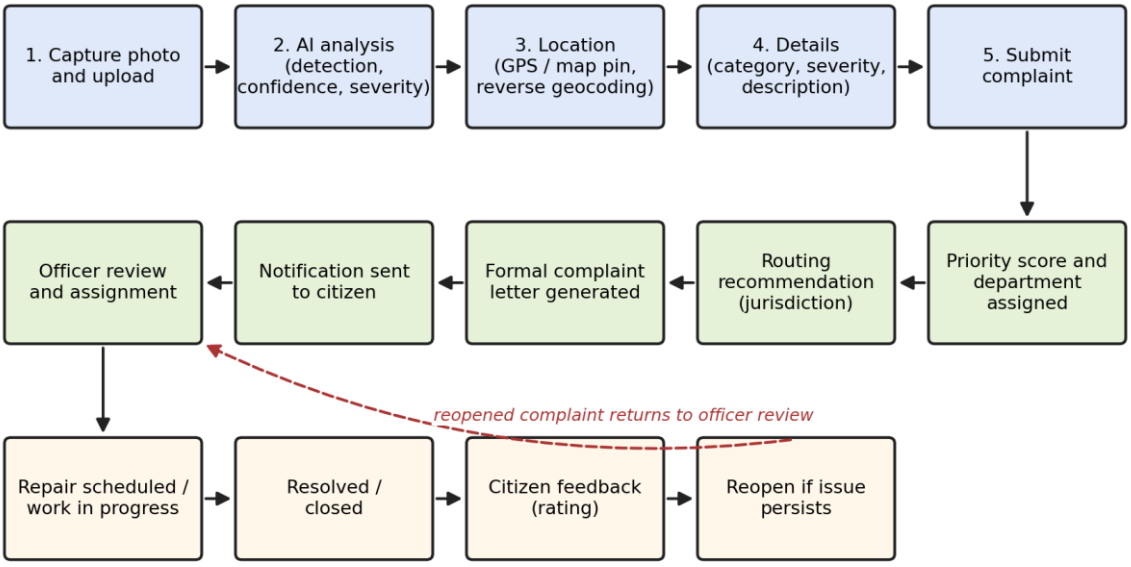


Fig. 2 - Complaint workflow from photo upload to resolution and feedback.

4.2 AI-based pothole detection and severity estimation

Image analysis uses a YOLOv8 model fine-tuned for a single class, *pothole*. The model is loaded once through the Ultralytics library and applied to the uploaded image. Every detected box is reduced to a confidence value c and an area ratio a, the bounding-box area divided by the image area. The detection with the highest confidence is selected and its severity is estimated with the weighted score in Eq. (1).

$$S = 0.6\, c + 0.4 \cdot \min(10\, a, 1) \qquad (1)$$

A score of at least 0.75 is classified as high severity, a score of at least 0.50 as medium, and anything lower as low. The factor 10 in the area term saturates that contribution once a pothole covers about one tenth of the image, so very large boxes do not dominate the score. The response returned to the client contains the category, confidence, estimated severity, the bounding-box coordinates, the number of potholes found and a descriptive message.

Three outcomes are handled explicitly. If potholes are found, the result described above is returned. If none is found, a confidence of 0.0 is returned and the citizen is asked to verify and classify the hazard manually. If the model is unavailable (for example, the weights are missing or the fallback mode is enabled) or inference fails, the response is flagged as a fallback with zero confidence and a clear explanation; the system never fabricates a score. The AI output is stored in fields separate from the citizen's own choices, which keeps the human decision authoritative while preserving the model's suggestion for audit.

4.3 Priority scoring

Each complaint receives an integer priority score that orders the officer's work list. The score combines the base severity, the hazard type, community confirmations and a critical flag (Table 2, Eq. (2)), where S_{sev} is the base severity points, I_{cat} equals 1 for broken cables, fallen poles and open manholes, n is the number of citizen confirmations and I_{crit} is the critical flag. These three hazard types are flagged critical automatically, so they receive both bonuses. Other citizens who observe the same hazard can confirm it, which increases n and recomputes the score, so widely observed hazards rise in the queue.

$$P = \min(\, S_{sev} + 40\, I_{cat} + \min(5n, 30) + 20\, I_{crit}, 200\,) \qquad (2)$$

Table 2 - Components of the priority score.

Component	Condition	Points
Base severity	Low / medium / high / critical	10 / 25 / 50 / 100
Critical hazard type	Broken cable, fallen pole, open manhole	+40
Community confirmations	Each citizen confirmation (counted up to 6)	+5 each, maximum +30
Critical flag	Complaint flagged as critical	+20
Upper bound	Total score is capped	200

4.4 ***Hazard-to-department mapping and routing***

Every hazard category maps to a default department that is used for assignment and as the addressee of the complaint letter (Table 3).

Table 3 - Default department for each hazard category.

Hazard group	Categories	Department
Road surface	Pothole, broken road, uneven road, road crack, broken footpath	Road Maintenance Department
Electrical	Broken cable, fallen pole, damaged streetlight	Electricity Department
Drainage and water	Waterlogging, blocked drain, open manhole	Drainage & Sanitation Department
Waste	Garbage	Sanitation Department
Traffic	Damaged signal	Traffic & Public Works Department
Civic / other	Construction waste, fallen tree, other	Municipal Corporation

Beyond this default, the routing engine recommends a specific government contact (Table 4). It first searches the active jurisdiction-mapping rules for the hazard category (or a wildcard), state and district. Failing that, it relaxes the match to category and state. Failing that, it ranks the verified, active contacts by how well their contact type (for example road/PWD-NH, local body, electricity, or drainage and sanitation) fits the hazard, with extra weight for a matching district and state. If no verified contact exists, the engine says so and asks the user to choose from the directory. Each recommendation carries a confidence label, a plain-language reason and a short list of alternatives. The seeded directory covers Kerala and Tamil Nadu, and each entry has fields for its official source URL and verification date.

Table 4 - Routing levels and confidence.

Level	Matching rule	Confidence
1	Jurisdiction mapping on category + state + district	High
2	Jurisdiction mapping on category + state (district not matched)	Medium
3	Heuristic ranking of verified contacts by type, district and state	Medium (state known) or low
4	No verified contact found; manual selection from the directory	Low (no recommendation)

4.5 Complaint letter generation

For each complaint the platform can generate a formal letter addressed to the officer in charge of the responsible department. The letter is filled from a fixed template with the complaint number, hazard type, severity, address, GPS coordinates, incident date, description and the citizen's contact details. It adds a critical-safety alert when the complaint is flagged critical and lists the requested actions: inspection of the site, corrective action, an acknowledgement with an estimated repair timeline, and a status update on the platform. The letter is stored with the complaint and can be regenerated.

4.6 Lifecycle, notifications and feedback

A complaint moves through twelve states: submitted, acknowledged, under review, verified, assigned, repair scheduled, work in progress, awaiting verification, resolved, closed, rejected and reopened. Officers change the state, add an optional message and choose whether it is visible to the citizen. Every transition is written to the status history together with the previous state and the officer who made the change, and a notification is created for the complainant. After resolution the citizen can submit a rating and comment, and may request reopening, which moves the complaint back to the reopened state.

5. Implementation

5.1 Backend

The backend is written in Python with FastAPI and exposes more than 25 REST endpoints grouped under */api/auth*, */api/complaints*, */api/contacts*, */api/departments* and */api/notifications*. Pydantic schemas validate requests and responses, SQLAlchemy maps the entities to PostgreSQL (SQLite is the default for local development), passwords are hashed with bcrypt and access tokens are signed JSON Web Tokens with a configurable expiry. File uploads are restricted to image types and 10 MB per file. The database URL, secret key, storage credentials, model path and AI fallback switch are read from

environment variables. On start-up the application creates the tables and seeds departments, demo users and government contacts. A smoke-test script loads the model and runs it on a blank image to confirm that inference works.

5.2 Frontend

The frontend is a TypeScript single-page application built with Vite and styled with Tailwind CSS. Its pages are the landing page, login, registration, citizen dashboard, report wizard, public hazard map, complaint detail, notifications, profile, government contacts and the officer dashboard; protected routes enforce authentication. Leaflet renders the hazard map with severity-coloured markers, category and severity filters and links to the complaint details, and the location picker supports both GPS detection and clicking on the map. The officer dashboard uses Recharts to show summary counters and complaints by category and by status, and allows in-place status updates. Forms are handled with React Hook Form and Zod, server state is cached with React Query and the authentication state is held in a Zustand store.

5.3 Model handling

The model weights are kept in *backend/models/pothole.pt* and loaded at import time. A helper script downloads the published weights when the file is absent, and the fallback switch disables inference altogether so that the platform remains usable without a model.

6. Results and discussion

6.1 Functional results

The implemented platform supports the following end-to-end flows across the three seeded roles. A citizen can register, report a hazard with photographs and location, obtain an AI assessment, generate a complaint letter, follow status updates and give feedback. An officer can filter complaints, change their status and view analytics. An administrator can maintain the government contact directory. Table 5 summarises the implemented functionality and the interfaces through which it is exposed.

Table 5 - Summary of implemented functionality.

Module	Capability	Interface
Authentication	Registration, login and profile update with role-based access	/api/auth; Login, Register, Profile pages
Reporting wizard	Three steps: photo with AI analysis, location, details	POST /api/complaints; Report page
AI analysis	Stand-alone pothole analysis with a re-analyse option	POST /api/complaints/analyze-image
Hazard map	Severity-coded public map with category and severity filters	GET /api/complaints/public; Map page
Officer dashboard	Status counters, category and status charts, status updates	PUT /api/complaints/{id}/status
Routing and contacts	Contact directory, routing recommendation, forwarding log	/api/contacts; Government Contacts page
Letter generation	Formal complaint letter for each complaint	POST /api/complaints/{id}/letter
Citizen engagement	Notifications, hazard confirmation, feedback and reopening	/api/notifications; /confirm; /feedback

6.2 Design observations

Several design decisions follow from the goal of a trustworthy civic tool. First, the AI component is advisory: the citizen's own category and severity are stored separately from the model's suggestion, and the model reports when it is unavailable instead of returning an invented value. Second, the priority score is rule-based, so an officer can see exactly why one complaint ranks above another. Third, every recommended contact can be traced to a verified directory entry with a source and verification date, which reduces the risk of sending a complaint to a wrong or outdated number.

6.3 Limitations

The current work has the following limitations. (i) The detector covers potholes only; the other fifteen categories depend on the category selected by the citizen. (ii) No quantitative evaluation of the detector (for example precision, recall or mAP on a held-out test set) was carried out in this work, and the severity heuristic is hand-tuned and has not been validated against assessments by road engineers. (iii) Routing coverage is limited to the seeded contacts for Kerala and Tamil Nadu; elsewhere the engine falls back to lower-confidence recommendations or manual selection. (iv) The platform prepares the letter and records forwarding attempts, but it does not itself transmit complaints into the systems of government departments. (v) Duplicate reports are handled through community confirmation rather than automatic geographic clustering.

7. Conclusion and future work

This paper presented AI Road Hazard Reporter, a full-stack platform that turns a citizen's photograph into a structured, prioritised and correctly addressed complaint and tracks it until resolution. The combination of YOLOv8-based pothole detection with explicit fallback behaviour, a transparent priority score, a three-level routing engine with a verified contact directory, automatic letter generation and a role-based lifecycle forms a complete reporting workflow that is simple for citizens and informative for officers.

Future work includes building a labelled local dataset and evaluating the detector quantitatively, extending detection to further categories such as waterlogging, garbage and open manholes, clustering nearby reports automatically to detect duplicates, delivering e-mail or SMS notifications, offering a mobile or progressive web application, and integrating with official grievance-portal interfaces so that complaints can be forwarded directly.

Acknowledgements

The authors thank Ms.Gayathri V AP/CSE and Sri Shakthi Institute Of Engineering And Technology for the guidance and facilities provided during this work.

References

Eriksson, J., Girod, L., Hull, B., Newton, R., Madden, S., & Balakrishnan, H. (2008). The Pothole Patrol: Using a mobile sensor network for road surface monitoring. *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services (MobiSys '08)* (pp. 29–39).

Jocher, G., Chaurasia, A., & Qiu, J. (2023). *Ultralytics YOLOv8* [Computer software]. https://github.com/ultralytics/ultralytics

Jones, M., Bradley, J., & Sakimura, N. (2015). *JSON Web Token (JWT)* (RFC 7519). Internet Engineering Task Force.

Keremberke. (n.d.). *YOLOv8n pothole segmentation* [Model]. Hugging Face. https://huggingface.co/keremberke/yolov8n-pothole-segmentation

Maeda, H., Sekimoto, Y., Seto, T., Kashiyama, T., & Omata, H. (2018). Road damage detection and classification using deep neural networks with smartphone images. *Computer-Aided Civil and Infrastructure Engineering*, *33*(12), 1127–1141.

Ramírez, S. (2018). *FastAPI* [Computer software]. https://fastapi.tiangolo.com

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 779–788).