International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 10 | Oct - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Crowdsourced Civic Issue Reporting and Resolution System: An AI-Assisted Web Platform

Lavanya A[1], Sanjhay Tharshan V V[2], Sharveash R[3], Shyam Sundar V[4]

[1]Assistant Professor, Department of Computer Science and Engineering, Adhiyamaan College of Engineering, Hosur, Tamil Nadu, India
[2],[3],[4] UG Scholars, Department of Computer Science and Engineering, Adhiyamaan College of Engineering, Hosur, Tamil Nadu, India

---

Abstract—Municipal service delivery depends on the timely identification, reporting, prioritization, and resolution of local infrastructure problems. Citizens may encounter fragmented complaint channels, uncertainty about the responsible department, and limited visibility into complaint progress.
This paper presents CivicAI, a crowdsourced civic issue reporting and resolution platform described and implemented in the accompanying mini-project report. The platform brings citizen registration, multimedia complaint submission, AI-assisted issue categorization, geographic location tagging, map-based visualization, complaint tracking, and administrative complaint management into a unified web workflow. A report can contain a description, an image, and location information; the AI assistance component is intended to derive structured attributes such as issue category, severity, priority, suggested department, contextual analysis, and recommended action. Citizens can follow a complaint through Submitted, In Review, In Progress, and Resolved states, while authorized administrators review and update reports. The report documents a browser-based interface and code-level integration with Supabase Authentication, PostgreSQL storage, and an analysis function, alongside the proposed use of React, Node.js, MapLibre, and OpenStreetMap. The paper consolidates the system design, modules, data model, implementation workflow, and qualitative observations. It does not claim measured model accuracy or municipal response-time improvements because such quantitative evaluation is not reported in the source project document.

Keywords—CivicAI, civic grievance management, crowdsourcing, issue classification, geolocation, map visualization, complaint tracking, smart governance.

1. INTRODUCTION

Public infrastructure problems such as potholes, accumulated waste, damaged streetlights, water leakage, and blocked drainage directly affect mobility, sanitation, safety, and quality of life. These problems are often first observed by residents, yet the information needed to act on them may be dispersed across phone calls, emails, paper forms, and separate departmental channels. Citizens may not know which authority is responsible, what details are required, or how to determine whether a complaint has been received and acted upon. For administrators, unstructured reports can make it difficult to locate a problem, identify duplicates, prioritize work, and maintain a consistent record of actions.

A crowdsourced reporting platform offers a way to convert local observations into structured service requests. The value of such a platform is not limited to accepting complaints: it must also capture sufficient evidence, associate each report with a place, organize it into a useful category, provide a workflow for administrative review, and communicate progress to the reporter. A map view can reveal the spatial distribution of reported problems, while a dashboard can help staff inspect and manage the incoming workload.

The project report describes CivicAI as a unified web-based system for citizens and administrators. Citizens register or sign in, submit descriptions and photographs, provide voice input where supported, and attach geographic information. An AI-assisted component suggests issue categories and severity-related attributes. The complaint is stored and made available for tracking and map visualization. Administrators can review reports, assign them to departments, and update their status. The system is intended to support a lifecycle with Submitted, In Review, In Progress, and Resolved states.

This paper has four contributions. First, it consolidates the functional requirements and module boundaries of the project into a coherent system description. Second, it explains the citizen-to-administrator workflow and the relationship between reporting, AI assistance, geolocation, storage, and tracking. Third, it documents the code-level data structure and access controls shown in the project report. Fourth, it identifies evaluation limits and practical next steps without inventing performance measurements that were not collected.

2. RELATED WORK

The source project report surveys research on crowdsourced civic reporting, road-damage detection, geotagged issue monitoring, and digital grievance management. The reviewed work indicates that civic platforms commonly combine user-generated reports with location information and centralized complaint records. The report cites CiviX and related crowdsourced systems as examples of digital complaint submission and resolution monitoring [1], [3]. These systems establish the importance of a structured reporting channel, but the quality of the resulting information still depends on the clarity and reliability of citizen submissions.

Computer-vision research provides a second relevant foundation. Ma et al. review road-imaging and pothole-detection methods, while Safyari et al. survey vision-based pothole detection using computer vision and machine learning [2], [4]. The project report also refers to an enhanced YOLO-based pothole assessment approach [5]. These studies motivate the use of image analysis to assist with identifying visible infrastructure defects. However, a general-purpose civic reporting workflow must handle more than road damage: waste, lighting, drainage, leakage, and other categories may require different visual cues and contextual information.

The project report further discusses Open311 as a collaborative model and open standard for civic issue tracking

[3]. Structured service-request formats can support interoperability between citizen-facing applications and public authorities, although actual integration depends on the adoption and maintenance of compatible municipal systems. Other cited work addresses ward-level monitoring, geotagged detection, and citizen complaint management [7]–[10]. The source bibliography also includes related work on crowdsourced civic reporting [10], Android-based pothole petitioning [11], web grievance handling [12], CivicWatch [13], complaint reporting [14], and geotagged complaint resolution [15].

CivicAI follows the common direction of combining citizen-generated evidence, location, categorization, and status monitoring. Its stated distinction is the integration of these functions in one workflow, with AI assistance intended to convert image and text inputs into structured report attributes and with a dedicated administrative interface for review and follow-up. The project report does not provide a controlled comparison against the cited systems; therefore, the present paper treats these as design motivations rather than evidence of superiority.

2.1 Synthesis of the Reviewed Approaches

The literature described in the project report can be grouped into three complementary themes. The first theme is crowdsourced reporting and grievance management. These systems emphasize the conversion of citizen observations into structured service requests, often with a centralized record and a means of following progress. Their main contribution to CivicAI is the workflow model: public participation supplies local observations, while a digital platform organizes those observations for review. Their main operational dependency is the response capability of the authority receiving the report.

The second theme is computer-vision-based infrastructure inspection. Pothole-detection research demonstrates how image analysis can support the identification of visible road defects [2], [4], [5]. Such methods are relevant to CivicAI's AI-assisted analysis component, but they should not be generalized to all civic categories without additional data and validation. Road defects, waste accumulation, damaged lighting equipment, and drainage problems have different visual characteristics and may require category-specific training examples or rules. The project report positions AI as an assistant that suggests categories and severity rather than as an autonomous authority for all civic decisions.

The third theme is geospatial monitoring. Geotagged reporting links a complaint to a physical location and makes it possible to inspect spatial patterns [7]. Map visualization can help administrators see whether reports are concentrated along a road, around a public facility, or within a particular neighborhood. However, a map marker alone does not establish the cause or severity of a problem. It must be interpreted together with the description, evidence, and administrative review. CivicAI combines these themes in a single workflow rather than treating image classification, location capture, and complaint tracking as separate applications.

The reviewed literature also highlights limitations that remain relevant to CivicAI. Image-based systems depend on image quality and representative data. Crowdsourced systems depend on citizen participation and accurate submissions. Location-based systems depend on coordinate quality and clear privacy practices. Grievance platforms depend on institutional adoption and timely updates. The project design brings these dependencies into one application, but it cannot remove them automatically. They must be addressed through interface design, human review, technical validation, and municipal process agreements.

3. PROBLEM DEFINITION AND OBJECTIVES

The problem addressed is the absence of a simple, unified workflow through which residents can submit actionable civic reports and monitor their progress. A useful report should contain a description of the issue, supporting evidence when available, a location, and a record of its processing state. The receiving authority needs a manageable queue of reports with enough context to review, categorize, prioritize, and route them.

The project objectives, as stated in the source report, are to simplify civic issue reporting; use AI-assisted analysis to suggest categories and severity; enable location-based reporting; improve complaint tracking; enhance transparency and accountability; support administrative complaint management; provide map-based visualization; and create a modular web platform that can be extended.

Requirement	System response described in the project
Accessible reporting	Citizen registration and a reporting interface for descriptions, photographs and voice input where available.
Structured issue information	AI-assisted suggestions for category, severity, priority, department and recommended action.
Location awareness	Geographic coordinates associated with reports and a map view of reported issues.
Progress visibility	Citizen-facing status tracking through the complaint lifecycle.
Administrative control	Authorized dashboard for reviewing, assigning, prioritizing and updating complaints.
Extensibility	Modular frontend, service, map and data components with future municipal integration.

3.1 Functional and Non-Functional Requirements

The functional requirements define the operations that the system must make available. A citizen must be able to register, authenticate, create a report, attach optional evidence, associate a location, view submitted reports, and inspect the current status. The AI service must be able to receive the report context and return structured suggestions when available. An administrator must be able to inspect reports, review the suggested classification, assign a department, and update the status. The map component must display the geographic distribution of reports and provide a way to inspect individual markers. These requirements follow the modules and use cases stated in the project report.

The non-functional requirements describe the quality attributes expected of the application. Usability requires a simple and responsive interface that works on common desktop and mobile browsers. Security requires authenticated access, role separation, protected image storage, and database-level restrictions. Reliability requires that a report is not lost when the AI service or map provider is temporarily unavailable. Maintainability requires modular code and

documented configuration. Scalability requires that report retrieval, image storage, and map rendering continue to function as the number of reports grows. The project report lists minimum client and server hardware and software requirements, but it does not provide measured capacity or service-level objectives.

The distinction between functional and non-functional requirements is useful during acceptance testing. A functional test can confirm that a report is created and appears in the citizen dashboard. A security test can confirm that a different citizen cannot retrieve that report. A reliability test can confirm that a report can still be stored when AI analysis times out. A performance test can measure whether a dashboard remains responsive with a representative volume of records. Each requirement should have a measurable acceptance criterion before a production pilot begins.

4. SYSTEM ARCHITECTURE AND DESIGN

CivicAI is organized as a modular web application. The citizen-facing interface provides registration, login, report creation, personal dashboard access, and complaint tracking. Application services validate incoming data, coordinate AI analysis, persist report details, and provide information to the dashboard and map. The administrative interface supports report review, department assignment, and status updates. Location and map components connect a report to geographic coordinates and display issue markers. A persistent data layer stores user-linked complaint information and analysis outputs.

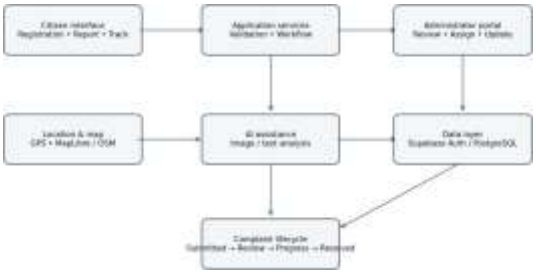


Fig. 1. Logical architecture of the CivicAI workflow, consolidated from the modules described in the project report.

The architecture separates the user interface from the processing and storage responsibilities. A citizen submits a report through the frontend. The application validates the required description and associated user identity, accepts an optional image and location payload, and invokes the analysis capability. The returned structured information is associated with the complaint record. The report can then be displayed in the citizen dashboard, represented geographically, and reviewed by an administrator. Status changes are reflected in the tracking view.

The project report describes React.js, HTML, CSS and JavaScript for the responsive interface; Node.js and Express.js for backend operations; MongoDB for complaint and user records; and MapLibre with OpenStreetMap for geographical visualization. Its code implementation section additionally shows Supabase Authentication, a Supabase Edge Function named analyze-civic-issue, and a PostgreSQL table named civic_reports with row-level security. These are distinct descriptions within the source report. This manuscript therefore distinguishes the proposed technology stack from the code-level implementation evidence and does not imply that MongoDB and Supabase are the same database or that every listed component was deployed together.

4.1 DESIGN PRINCIPLES AND STAKEHOLDER ROLES

The system design is guided by a small set of principles that follow from the problem statement. The first is low-friction reporting: a resident should be able to create a report without needing to understand municipal organizational structures. The second is sufficient context: the report should capture the description, evidence, and location required for an initial assessment. The third is traceability: each report should have a persistent identifier and a visible processing state. The fourth is human oversight: machine-generated labels are recommendations and should not replace an authorized officer's decision. The fifth is modularity: reporting, analysis, mapping, persistence, and administration should be separable so that individual services can be improved without redesigning the entire application.

The primary stakeholder groups are citizens, administrative users, and the municipal departments that ultimately perform or coordinate field work. Citizens need an accessible submission interface, confirmation that the report has been recorded, and understandable progress information.
Administrative users need a queue that can be filtered and reviewed, with access to the original evidence and location. Departmental staff need actionable information and a clear assignment. The project report describes the citizen and administrator roles directly; department-level operational accounts and integrations are considered a future extension rather than a verified deployed feature.

The system boundary is important. CivicAI collects and organizes information about an issue; it does not itself repair a road, remove waste, or replace the public authority responsible for the service. The platform can support the communication and record-keeping stages that precede and accompany field action. Actual resolution depends on municipal staffing, budgets, equipment, work schedules, and formal procedures. This distinction prevents a software prototype from overstating its impact on physical service delivery.

4.2 REPORT LIFECYCLE AND STATE MANAGEMENT

A complaint lifecycle provides a common vocabulary for citizens and administrators. The four states identified in the project report are Submitted, In Review, In Progress, and Resolved. Submitted is the initial state after the application accepts the report. In Review indicates that an administrative user has inspected the submitted information and is deciding whether it is valid, correctly categorized, or requires clarification. In Progress indicates that the responsible unit has begun an action or that the issue has entered an active work queue. Resolved indicates that the authority has recorded

completion. The application should make the meaning of each state visible to avoid ambiguity for users.

State transitions should be controlled rather than treated as arbitrary text edits. A newly created report should begin in Submitted. An authorized administrator may move it to In Review after inspection, then to In Progress when a department or action is assigned, and finally to Resolved after completion is confirmed. A report that cannot be actioned may require a separate rejected, duplicate, or needs-information state in a future version; these states are not claimed as part of the four-state workflow documented in the source report. If such states are added, their definitions and citizen-facing messages should be specified before implementation.

For auditability, a robust implementation should preserve a status event history rather than only overwrite the current status field. Each event could record the report identifier, previous state, new state, actor, timestamp, and optional note. This design would allow the citizen dashboard to show a chronological timeline and would help administrators investigate delays or incorrect updates. The supplied report describes dynamic status updates but does not document this event-history table; it is therefore a recommended enhancement.

4.3 INFORMATION QUALITY AND DUPLICATE HANDLING

Crowdsourced reporting creates a trade-off between broad participation and data quality. A citizen may submit an incomplete description, attach an image that does not show the reported problem, select an inaccurate location, or create a second report for an issue already reported by another resident. The application should guide the user with short prompts that encourage a useful description: what is wrong, what visible impact it causes, and where it can be found.
Required fields should be kept to a minimum so that the reporting process remains accessible.

Image evidence and text descriptions complement one another. A photograph can show the physical condition of a road or public facility, while the text can explain details that are not visible in the image, such as when the issue occurs or whether it creates an obstruction. The AI analysis service may use both modalities, but the original user-provided values should remain available for review. A derived summary must not silently replace the original description because the transformation may omit important context.

Duplicate reports are a likely feature of any public reporting system, especially when a visible problem affects a busy road or neighborhood. A future duplicate-assistance mechanism could compare approximate coordinates, issue category, submission time, and image similarity to flag potentially related records. The system should not automatically discard a report solely because another record is nearby: two different defects can exist close together, and one incident may affect multiple locations. An administrator should make the final merge or duplicate decision and preserve the contributing reports for traceability.

4.4 Data Flow and Component Interaction

The end-to-end data flow can be understood as a sequence of trust boundaries. The browser is the first boundary: it collects a user-entered description, optional image evidence, and location details. The authenticated application service is the second boundary: it verifies the user's session and validates the payload before allowing a report to be created. The AI analysis service is a third boundary: it receives only the information needed for analysis and returns structured suggestions. The database is the persistent boundary: it stores the report and enforces access rules. Finally, the citizen and administrator interfaces read permitted data and present it according to each role.

A separation between original and derived data is valuable. The original description, image reference, and location represent what the citizen submitted. The category, severity, short description, department suggestion, and recommended action represent analysis outputs or later administrative decisions. These should be distinguishable in the schema and interface. If an administrator edits a suggested category, the system should retain the final approved value and, where feasible, the original suggestion. This separation makes the workflow explainable and reduces the risk that an automated recommendation is mistaken for a verified fact.

The report's civic_reports schema places several fields in one relation, which is suitable for a compact prototype. As the application grows, related entities may be separated into tables for status history, departments, attachments, and administrative assignments. A normalized design can reduce repeated department information and support multiple attachments or multiple status events per report. Such a change should be guided by actual query requirements and should preserve a stable report identifier for links, map markers, and notifications.

5. FUNCTIONAL MODULES

5.1 User Registration and Authentication

The registration module enables citizens to create an account using identifying details such as name, email address, phone number, and password. Input validation is intended to check required fields and email format before account creation. The login workflow verifies the supplied credentials and opens the appropriate dashboard. The report describes role-based access for citizens and administrators so that administrative functions are restricted to authorized users. In the code implementation, authentication is connected to Supabase Authentication. The report also notes that passwords should be stored securely using appropriate hashing rather than in plain text; authentication services should handle credential storage rather than application code exposing passwords.

5.2 Civic Issue Reporting

The reporting module is the main data-entry point. A citizen selects or describes the issue, adds a photograph when available, and supplies location information. Voice input is described as an additional input option where supported. Typical examples include potholes, garbage accumulation, water leakage, damaged streetlights, and drainage problems. A clear

description helps an administrator understand what has happened and where attention is required. The application should preserve the original citizen description even when the AI component generates a shorter summary or category suggestion.

5.3 AI-Assisted Issue Classification

The AI component is designed to analyze uploaded images and textual descriptions and suggest structured information. The project report lists issue category, severity, priority level, short description, suggested department, context analysis, estimated resolution time, and recommended action among the fields associated with analysis. These outputs can make a report easier to scan and can support an initial routing decision. They are recommendations, not authoritative determinations: an administrator must be able to review and correct the category, severity, department, or action before operational use.

The report's code section identifies an analysis function named analyze-civic-issue and describes the transfer of an image and user description to the analysis service. The resulting fields are associated with the report record. The source material does not specify the underlying model architecture, training dataset, validation split, confidence calibration, or benchmark results. Consequently, no particular convolutional network, transformer, YOLO variant, or numerical accuracy is attributed to CivicAI in this paper.

5.4 GPS Location Tagging and Map Visualization

The location module associates each complaint with geographic coordinates. This supports accurate identification of the affected place and provides the data required for map visualization. The project report identifies MapLibre and OpenStreetMap as the map technologies considered for displaying complaint locations. On the map, each report can be represented by a marker linked to its category and status. Filters can help users or administrators inspect selected classes of problems and understand the geographical distribution of complaints.

Location capture must account for permission denial, unavailable GPS, inaccurate coordinates, and privacy. The interface should allow the user to review or correct a location before submission. A map marker should be treated as an approximate representation of the reported issue rather than proof that the issue has been independently verified.

5.5 Complaint Tracking

The tracking module provides a status view for complaints submitted by a citizen. The lifecycle described in the project report contains four states: Submitted, In Review, In Progress, and Resolved. The initial state indicates that the report has been received. In Review indicates administrative examination; In Progress indicates that action has been initiated; and Resolved indicates that the authority has marked the complaint as completed. A timestamped status history would improve auditability, although the source report does not provide a complete event-history schema.

5.6 Administrative Dashboard and Department Management

The administrative dashboard provides a consolidated view of incoming reports. Authorized personnel can inspect the description, image, location, and AI-generated suggestions, then categorize the complaint, assign it to a relevant department, prioritize it, and update its status. Department management supports the organizational mapping between issue categories and responsible service units. The complaint-management and analytics views described in the report are intended to help administrators monitor the complaint queue and observe issue distribution.

5.7 DETAILED PROCESSING SEQUENCE

The processing sequence begins when the citizen opens the reporting interface after authentication. The user enters a textual description and may attach an image. If location permission is granted, the browser or device supplies geographic coordinates; otherwise, the application should allow a location to be selected or entered through the available map interface. Before submission, the interface should display a review summary so that the citizen can correct the description, image, or location. The report is then submitted to the application service.

At the service boundary, input validation checks that mandatory values are present and that optional values have acceptable formats. Image uploads should be checked for permitted file type and size, and should be stored through a controlled storage mechanism. The user identity should be taken from the authenticated session rather than trusted from a client-supplied user identifier. These controls reduce malformed submissions and prevent a client from associating a report with another user's account. The source report documents authentication and row-level policies, while these additional validation details are implementation recommendations for a hardened deployment.

The analysis stage receives the description and, where provided, the image reference or image content. The service returns a set of candidate attributes, including category and severity-related fields as documented in the report. The application should handle service errors explicitly: if analysis is unavailable, the citizen's report should not be lost. A safe fallback is to store the original report with analysis marked as pending or unavailable and allow an administrator to classify it manually. This preserves the core reporting function even when the AI component is degraded.

After analysis, the application persists the report and the returned attributes. The citizen dashboard retrieves the user's reports and displays the current state. The map view uses location data to place markers and can expose a summary panel when a marker is selected. The administrative dashboard retrieves reports within the administrator's permitted scope and presents the original evidence together with the suggested classification. An administrator can then correct the category, choose the responsible department, and update the status. The citizen-facing view should reflect only the information appropriate for public display.

5.8 AI OUTPUT REVIEW AND HUMAN-IN-THE-LOOP OPERATION

AI-assisted classification can reduce repetitive data-entry work, but it can also introduce incorrect or overconfident suggestions. For example, an image of a wet road may be interpreted as water leakage when it is actually rainwater; a pile of material may be classified as waste even when it is construction material awaiting collection. Severity is also context-dependent. A small pothole on a low-traffic lane may have a different risk profile from a similar defect on a high-speed road. The report's proposed severity and priority fields should therefore be interpreted as triage support rather than final operational decisions.

A human-in-the-loop process should expose the suggested category, severity, department, and recommended action to the administrator alongside the evidence that produced them. The administrator should be able to accept, edit, or reject each suggestion. If the administrator changes a label, the system can preserve both the AI suggestion and the final human decision for later analysis. Such records can support future dataset curation and help identify systematic model errors, provided that data collection and privacy practices are clearly defined.

A production AI service should also report when it cannot confidently classify an input. A low-confidence or out-of-scope image should be routed for manual review rather than forced into an unrelated category. The project report does not document confidence thresholds or a trained model evaluation, so these mechanisms are proposed design requirements. No claim is made that the current prototype already implements calibrated confidence or automated learning from administrative corrections.

5.9 Proposed Classification Logic and Safe Fallback

The source report describes the AI function by its input and output rather than by a reproducible model specification. At the application level, the classification flow can still be defined independently of the model internals. The function receives the citizen's description and an optional image, prepares the input for the analysis service, and requests a structured response. The response is checked for required fields and allowed values before being stored. If a field is missing or invalid, the application should not silently substitute an arbitrary category. It should preserve the original complaint and mark the incomplete analysis for review.

A generic control sequence is: (i) verify that the authenticated user owns the submission request; (ii) validate text length and image metadata; (iii) store or reference the image securely; (iv) invoke the analysis service with the description and image; (v) parse the returned structured fields; (vi) validate category and severity against configured vocabularies; (vii) persist the report with analysis state; and (viii) return a report identifier and submission confirmation. The control sequence describes a safe application pattern, not a claim about the exact internal implementation of the model.

A failure in the analysis stage should not block civic reporting. Network timeouts, malformed model responses, unsupported image types, and service unavailability are possible. A resilient application can create the complaint with an analysis-pending flag and allow manual classification later. The citizen should receive an accurate acknowledgement that the report was submitted, while the administrator should see that AI suggestions are unavailable. This distinction is preferable to displaying a fabricated classification or reporting a failed submission when the complaint has already been stored.

5.10 Geospatial Representation and Map Interaction

Geographic coordinates connect the report to the physical location of the issue. A map marker provides a quick visual reference, but the underlying record should preserve the coordinate values and any relevant location description. The map should support inspection at different zoom levels and avoid placing too many overlapping markers in a single visual area. Marker selection can open a compact summary containing the issue category, a short description, current status, and the report date, subject to the application's privacy rules.

Map filtering can make the view more useful to different stakeholders. A citizen may want to see reports near a selected area, while an administrator may want to filter by category, status, or department. The project report describes an interactive civic issue map and map-based visualization. It does not specify a clustering algorithm, spatial index, or exact filter implementation. These should be selected based on the volume of records and the map provider's capabilities. For larger datasets, server-side spatial queries and marker clustering may be necessary to keep the interface responsive.

The application should handle coordinate quality explicitly. Browser geolocation can be unavailable or imprecise indoors, and users may deny permission. A map-based manual pin can provide a fallback, but it should be clear that the marker represents the location supplied by the reporter. If location accuracy is available from the device, the application may store it as metadata and warn the user when the reported accuracy is poor. The system should avoid displaying a false impression of survey-grade precision.

5.11 Administrative Triage and Assignment

The administrative dashboard is the point at which a crowdsourced observation becomes a managed service request. An administrator needs to see enough information to determine whether the complaint is understandable, whether the location is usable, whether the category is appropriate, and whether the report belongs to the authority's area of responsibility. The suggested department field can help with this initial decision, but a directory of actual departments and their responsibilities is needed for reliable assignment. The source report describes department management and complaint management screens but does not provide a complete municipal department directory.

Triage can be supported by sorting and filtering, but the rules should be transparent. For example, the interface may allow an administrator to filter by high suggested severity, category, age of report, or current status. A priority score should not be treated as a guaranteed measure of risk unless the scoring method has been defined and validated. An administrator

should be able to override the suggested priority and record a reason when doing so. This provides flexibility for local conditions that an automated model may not understand.

Assignment should be traceable. A report can be associated with a responsible department and, if required, an individual work queue. The system should record when the assignment was made and by whom. If the report is reassigned, the prior assignment should remain visible in the history. These details support accountability and make it possible to calculate operational measures such as time to first review or time spent in each state in a future evaluation.

6. DATA MODEL AND SECURITY

The code implementation in the source report defines a PostgreSQL table named civic_reports. The documented fields include a UUID primary key, a user identifier linked to the authenticated user, image URL, user description, JSON location data, issue category, severity, priority level, short description, suggested department, context analysis, estimated resolution time, recommended action, and creation timestamp. This structure keeps the original report and the derived analysis attributes together for retrieval by the application.

Field	Purpose in the documented schema
id, user_id	Unique report identifier and association with the authenticated user.
image_url, user_description	Reference to submitted evidence and the citizen-provided description.
location_data	JSON geographic/location information for mapping.
issue_category, severity, priority_level	Structured classification and triage attributes.
short_description, context_analysis	Concise summary and additional analysis context.
suggested_department, recommended_action	AI-assisted routing and suggested next action.
estimated_resolution_time, created_at	Estimated duration field and report creation timestamp.

The source code excerpt enables Row Level Security (RLS) on civic_reports and shows policies that allow authenticated users to select and insert rows only when auth.uid() matches user_id. This is a useful baseline for isolating citizen-owned records. The project report does not show a complete administrator policy, audit log, file-storage policy, retention policy, or a full threat model. A production deployment should add carefully scoped administrative authorization, server-side validation, safe image storage, rate limiting, audit trails, backup procedures, and protection against malicious uploads.

6.1 DATA VALIDATION, PRIVACY AND ACCESS CONTROL CONSIDERATIONS

Civic reports can contain personal information, photographs of homes or vehicles, and location details that may reveal a citizen's movements or residence. The application should collect only the information required to process the complaint and should explain how it is used. Public map views should avoid exposing a reporter's name, phone number, email address, or other unnecessary identifying details. If a photograph includes faces, vehicle registration numbers, or private property, a future production workflow may need image redaction or a review process before public display.

Authentication and authorization serve different purposes. Authentication establishes the identity of the user; authorization determines which records and operations that user may access. The RLS policies shown in the project report restrict authenticated users to selecting and inserting reports where the user_id matches the current authenticated identity. This is a useful citizen-level control. However, administrators require a separate, narrowly scoped authorization rule. Simply granting every authenticated user access to all reports would defeat the purpose of row-level security. Administrative privileges should be assigned through a trusted role mechanism and checked on the server or database policy layer.

File storage must be protected independently of database row access. An image URL should not be assumed to be private merely because the associated database record is protected. A secure implementation should define whether image objects are public or private, use time-limited signed access where appropriate, validate file content, and prevent executable or oversized uploads. The source report contains an image_url field but does not fully describe the storage bucket policy; this paper therefore identifies it as an area requiring deployment-level verification.

Retention and deletion rules should be established for reports that contain inaccurate personal information or are no longer needed. Administrative actions should be logged so that status changes and assignments can be traced. Backups should be encrypted and restoration should be tested. These measures are particularly important if the prototype is connected to a public authority or begins handling sensitive location and image data at scale.

6.2 Database Integrity and Query Considerations

The documented civic_reports table uses a UUID as its primary key and a foreign-key relationship from user_id to the authenticated user table. UUID identifiers are suitable for distributed web applications because they can be generated without relying on a single sequential counter. The created_at timestamp provides a basic chronological field. The JSONB location_data field allows location information to be stored flexibly, although a production design may prefer explicit latitude, longitude, and accuracy columns when spatial filtering is required. A spatial database extension or geographic index may be appropriate for neighborhood-scale queries, subject to the deployment environment.

The report fields for category, severity, priority, and department are text values in the shown schema. Text values are simple to evolve during prototyping, but unconstrained strings can lead to inconsistent labels such as multiple spellings for the same category. A controlled vocabulary or lookup table can provide stable identifiers, display labels, and optional descriptions. Constraints can reject invalid states and categories. The set of allowed categories should be documented and versioned so that historical reports remain interpretable when the classification taxonomy changes.

Indexes should be selected from observed query patterns. The citizen dashboard is likely to retrieve reports by user_id and creation time. The administrative queue may retrieve records

by status, category, or department. The map may query by geographic bounds. Indexes can improve these operations, but unnecessary indexes increase write overhead and storage.
Query plans and representative data volumes should be measured before finalizing the index strategy. The project report does not provide a performance profile or index definitions beyond the table and access policies shown.

7. IMPLEMENTATION WORKFLOW

The implementation workflow can be summarized as a sequence of coordinated operations. First, the user authenticates and receives access to the citizen dashboard. Second, the user enters a complaint description, attaches an image if available, and supplies geographic information. Third, the application validates the input and invokes the analysis function. Fourth, the AI-generated fields are returned and stored with the complaint. Fifth, the report is made available in the citizen's dashboard and map view. Finally, an administrator reviews the report, assigns it where appropriate, and updates the status as work progresses.

Step	Input / action	Output / next stage
1. Authentication	Citizen credentials	Authenticated session and dashboard access
2. Submission	Description, optional image, location	Validated complaint payload
3. Analysis	Image and text context	Suggested category, severity and related fields
4. Persistence	User data and analysis result	Stored civic_reports record
5. Presentation	Stored record and coordinates	Dashboard entry and map marker
6. Administration	Review and department decision	Assignment and status update
7. Tracking	Updated complaint state	Citizen-visible progress information

The report's implementation section shows Supabase client configuration through environment variables, calls to Supabase Authentication, invocation of the analyze-civic-issue function, insertion into civic_reports, and RLS policies for user-owned reports. The conceptual architecture also describes Node.js/Express and MongoDB. The final deployment should document the exact configuration actually used, including whether any earlier stack components were replaced during implementation, so that readers can reproduce the system without ambiguity.

7.1 FUNCTIONAL TEST SCENARIOS

The source project report provides descriptive results and screenshots but does not include a formal test matrix with execution outcomes. To make the implementation reproducible, a functional test plan should cover both normal and exceptional paths. The following scenarios define what should be checked during a future verification cycle; they are not presented as completed test results.

Authentication scenarios should include valid registration, duplicate email registration, valid login, invalid password, expired session, and unauthorized access to administrative pages. Reporting scenarios should include a complete report with image and location, a report without an optional image, a missing mandatory description, an unsupported image format, an oversized image, and a network interruption during submission. The expected behavior is that valid reports are stored once, invalid inputs are rejected with a clear explanation, and interrupted submissions do not create misleading success confirmations.

AI scenarios should include representative examples from every supported issue category, ambiguous photographs, low-light images, blurred images, images unrelated to civic issues, and text-only submissions. The service should return a structured response or a controlled failure state. Map scenarios should include permission granted, permission denied, unavailable coordinates, and manual location correction.
Tracking scenarios should verify that a citizen can view only their own reports and that status changes appear in the correct order. Administrative scenarios should verify that only authorized staff can assign departments or change status.

The test log should record the test identifier, environment, input conditions, expected result, observed result, pass/fail decision, and evidence such as a screenshot or server log. Testing should be repeated after changes to authentication rules, database schema, AI service behavior, or map integration. If the application is deployed to multiple environments, the configuration and data fixtures should be versioned so that the same tests can be repeated consistently.

7.2 Error Handling and Operational Observability

A reliable complaint platform should distinguish user-correctable errors from service failures. Missing descriptions and invalid file types can be reported directly in the form.
Authentication expiry should lead to a safe sign-in flow. An unavailable analysis service should leave the complaint recoverable. A database write failure should not be reported as a successful submission. A map tile failure should not prevent the citizen from viewing the underlying report details. These behaviors reduce confusion and help maintain trust when one component is temporarily unavailable.

Operational logs should capture technical events without unnecessarily copying sensitive complaint content. Useful events include authentication failures, report creation attempts, analysis request identifiers, processing duration, storage failures, and administrative state changes. Logs should avoid passwords, secret keys, full private image URLs, and unnecessary personal data. Correlation identifiers can connect a browser request with backend processing and analysis-service logs without exposing the citizen's identity in every message.

Monitoring can track service availability, error rates, processing latency, storage usage, and queue depth. Alerts should be tied to actionable thresholds and routed to an identified maintainer. A backup process should be paired with periodic restore tests; an untested backup is not sufficient evidence of recoverability. These operational practices are not detailed in the supplied project report but are necessary considerations before the platform is used as a dependable public-facing service.

7.3 Reproducibility and Technical Documentation

For another team to reproduce the prototype, the implementation should document the repository structure, runtime versions, dependency versions, environment

variables, database schema, authentication configuration, edge-function deployment steps, and map configuration. The source report includes code excerpts and a database definition, but a complete reproducibility package should also identify the exact frontend and backend versions and explain which technology components are active in the deployed build. This is especially important because the report's design discussion refers to MongoDB and Node.js/Express while the code excerpt demonstrates Supabase Authentication and PostgreSQL.

The AI analysis function should be documented with a stable request and response schema. The documentation should define required and optional input fields, permitted category labels, severity values, timeout behavior, and error responses. Example requests and responses should use synthetic data that does not expose real citizens' information. If the model or prompt is changed, its version should be recorded so that classification results from different releases can be distinguished. A reproducible build should pin dependencies and provide a test configuration that does not rely on production secrets.

A paper describing a software prototype should also make a clear distinction between implemented features, partially implemented features, and proposed features. The project report describes voice input, AI classification, GPS tagging, tracking, map visualization, and administration as modules, and includes code-level excerpts for some backend operations. It does not provide a full feature-by-feature verification record. A future technical appendix should map each feature to its source module, deployment status, and corresponding test evidence. This would strengthen reproducibility and reduce ambiguity for reviewers.

8. RESULTS AND DISCUSSION

The source project report presents the developed system through interface figures for user registration, the user dashboard, issue reporting, complaint tracking, the civic issue map, AI-assisted classification, the administrator dashboard, department management, complaint management, and analytics. The report describes these modules as integrated into a single application. The screenshots and narrative provide qualitative evidence of the available interface areas and intended workflow.

At the functional level, the reported outcome is that users can access a dashboard, submit civic complaints with descriptions and photographs, associate a location, view reports and status information, and use the map view. The administrative interface is described as supporting complaint review, categorization, department assignment, and status updates. The code excerpts substantiate the presence of authentication integration, an analysis-function invocation, a report table, and user-level RLS policies.

The evaluation evidence in the project report is descriptive rather than quantitative. It does not state the number of test users, number of complaint samples, classification accuracy, precision, recall, F1-score, geolocation error, average response time, throughput, task completion time, or user satisfaction. No such values are inferred here. Similarly, the report does not document a live integration with a municipal authority or measured reductions in complaint resolution time. The system should therefore be understood as a developed academic prototype with described modules, not as a validated city-wide deployment.

8.1 Evaluation Plan for Reproducible Validation

A subsequent evaluation should test each module using a documented test set and record expected and observed outcomes. For AI classification, a labeled image-text dataset should be divided into training, validation, and test partitions where model training is part of the implementation. Per-class precision, recall, F1-score, and confusion matrix should be reported, with separate analysis for poor lighting, occlusion, weather, and low-resolution images. For location capture, the measured distance between the reported marker and a verified reference coordinate should be recorded. For the application workflow, successful and failed submissions, authorization attempts, upload failures, and status transitions should be tested.

Evaluation area	Suggested measure	Evidence required
Classification	Per-class precision, recall, F1-score	Labeled and independently reviewed test reports
Geolocation	Coordinate error in metres	Reference coordinates and captured locations
Workflow reliability	Successful/failed submissions and transitions	Repeatable functional test log
Performance	API latency and concurrent request handling	Load-test configuration and raw results
Usability	Task completion and user feedback	Participant protocol and anonymized responses
Security	Access-control and upload validation results	Authorization and security test cases

These are proposed evaluation measures, not results of the present project. They should be completed before making claims about accuracy, reliability, scalability, or improved municipal response times.

8.2 INTERPRETATION OF THE PROTOTYPE EVIDENCE

The interface figures in the project report demonstrate that the application is organized around distinct user tasks rather than a single generic form. Registration and dashboard screens support entry into the system; the reporting screen supports the creation of a complaint; tracking and map screens expose report progress and geography; and administrative screens support management of complaints and departments. The analytics view indicates an intention to summarize the collected reports. This organization is consistent with the project objective of connecting citizen reporting with administrative follow-up.

The database code excerpt provides a more concrete view of the prototype than the high-level architecture alone. It names the civic_reports relation and enumerates the fields used to store citizen descriptions, image references, location data, and AI-related attributes. It also demonstrates user-scoped select and insert policies using the authenticated identity. These details support the claim that the project includes a persistent report model and basic citizen-level data isolation. They do not, by themselves, prove complete security, successful

operation under load, or correct access control for every administrative operation.

The results narrative in the source report describes the system as improving transparency, organization, and complaint monitoring. These are intended benefits that follow from the workflow design. Demonstrating an actual improvement would require a comparison with the previous complaint process, measured across a defined period and user population. For example, a study could measure the proportion of reports with usable locations, the time from submission to first review, the proportion of complaints with status updates, and the rate of duplicate or misrouted reports. Until such a study is performed, the benefits should be stated as design objectives or potential outcomes rather than measured impact.

8.3 Metrics for a Future Field Study

A field study should define its research questions before collecting operational data. One question could examine whether a structured submission form improves the completeness of civic reports. A second could examine whether AI suggestions reduce the time administrators spend on initial categorization. A third could examine whether visible status updates improve citizens' understanding of complaint progress. A fourth could examine whether a map-based view helps administrators identify spatial clusters or repeated infrastructure problems. Each question requires a suitable baseline and a clearly defined outcome measure.

For report completeness, the study could calculate the proportion of submissions containing a usable description, image, and valid location. For administrative efficiency, it could record median time from submission to first review and median time to department assignment. For tracking transparency, it could measure the proportion of reports with at least one status update and the time between status changes. For map usefulness, it could record the proportion of reports that administrators can locate without additional clarification. These metrics should be reported with sample size, observation period, missing-data handling, and appropriate summary statistics.

Any evaluation involving citizens should follow an appropriate consent and privacy process. Personal details should be minimized or anonymized in the analysis dataset. The study should distinguish system processing time from municipal repair time because the latter depends on factors outside the software. Results should also be reported by issue category where sample sizes permit, since a single aggregate measure can hide differences between pothole reports, waste complaints, lighting failures, and drainage problems. No such field-study data is included in the current project report.

9. LIMITATIONS AND FUTURE ENHANCEMENT

The project report identifies cloud-based data management, integration with municipal systems, and continued enhancement as future directions. Additional work is required in several areas. First, AI suggestions should be validated on a representative, locally relevant dataset and should expose uncertainty to administrators. Second, duplicate detection can help identify multiple reports of the same incident. Third, a clear department directory and configurable routing rules can reduce manual assignment. Fourth, notification services can inform citizens when a report is received or its status changes. Fifth, multilingual text and voice support can improve accessibility for residents with different language preferences.

Operational deployment also requires a formal agreement with participating municipal departments, service-level expectations, escalation procedures, and a process for closing or reopening reports. Map interfaces should handle privacy and coordinate precision carefully. Administrative actions should be logged with user identity and timestamps. The system should be evaluated for accessibility, low-bandwidth use, mobile responsiveness, and secure file handling. Cloud deployment, database backups, monitoring, and disaster recovery should be specified before the application is relied upon for public service delivery.

9.1 DEPLOYMENT AND INTEGRATION ROADMAP

A practical deployment roadmap can be divided into stages. The first stage is prototype stabilization: confirm the exact frontend, backend, database, authentication provider, map library, and AI service used in the running version. Remove unused dependencies and document environment variables without exposing secret keys. The second stage is controlled pilot testing with a small group of users and a limited set of issue categories. The third stage is administrative workflow validation with representatives of the departments expected to receive reports. The fourth stage is a wider rollout only after privacy, security, performance, and operational responsibilities have been reviewed.

Municipal integration should be treated as a governed data exchange rather than a simple API connection. The authority must define the categories it accepts, the identifiers used to track service requests, the department responsible for each category, the expected response states, and the mechanism for acknowledging and closing requests. The project report references Open311 as a relevant standard [3]. Compatibility with an external standard can improve interoperability, but the system should not claim Open311 conformance unless its API behavior and data fields have been checked against the applicable specification.

Operational ownership is as important as software readiness. A deployment plan should identify who monitors failed submissions, who handles reports that cannot be classified, who maintains department assignments, and who responds when a citizen disputes a resolution. A public-facing service also needs a published support channel and a process for handling abusive or fraudulent submissions. These procedures are not supplied in the mini-project report and must be agreed with the participating authority before production use.

9.2 ACCESSIBILITY AND INCLUSIVE USE

A civic reporting platform should remain usable by people with different devices, levels of digital literacy, and accessibility needs. The project report describes a responsive

web interface and voice input where available. Further design work should ensure that forms have clear labels, keyboard navigation, readable contrast, and screen-reader-compatible status messages. Error feedback should explain how to correct a problem rather than merely indicating that submission failed. Controls for uploading an image or obtaining location permission should have an alternative path when a device feature is unavailable.

Language support is another important consideration in a multilingual setting. A future version could provide translated interface labels and allow descriptions in local languages. If AI analysis is applied to multilingual text, the system should validate performance separately for each supported language and should not assume that a model trained primarily on English will interpret local place names, colloquial descriptions, or transliterated text reliably. Voice transcription should also be checked for accents, background noise, and domain-specific vocabulary.

Connectivity constraints should be considered for users who submit reports from areas with weak mobile data coverage. The application can reduce unnecessary image size, preserve form data during transient failures, and show a clear upload state. Any offline draft or retry mechanism should avoid duplicate submissions when connectivity returns. These improvements would extend the accessibility goals of the project beyond responsive layout alone.

9.3 Risks and Mitigation Measures

Several risks follow from the nature of crowdsourced civic data. Incorrect reports can consume administrative time; image or text analysis can misclassify a problem; inaccurate coordinates can send staff to the wrong place; and unauthorized access can expose citizen information. The interface can mitigate incomplete reports through guided fields and confirmation before submission. AI errors can be mitigated through administrator review and a manual fallback. Location uncertainty can be exposed to the user and corrected on a map. Access risks can be mitigated through authentication, role-based authorization, RLS, secure file policies, and audit logs.

A further risk is that citizens may interpret a status such as In Progress as a promise of immediate repair. The application should describe status definitions clearly and avoid showing an estimated resolution time as a guaranteed deadline. The estimated_resolution_time field shown in the database schema should be treated as a suggestion or planning value unless a responsible authority has committed to it. Clear language can prevent unrealistic expectations and preserve trust in the reporting process.

A platform can also receive abusive, duplicate, or irrelevant submissions. Rate limits, moderation workflows, and report review can reduce misuse, but controls should not create unnecessary barriers for genuine reports. A moderation process should be transparent and should provide a route for a citizen to correct or clarify a report. The project report does not document a moderation subsystem; it should be considered if the system is opened to a large public user base.

9.4 Sustainability and Maintainability

The long-term usefulness of CivicAI depends on the maintainability of its categories, map services, authentication configuration, and AI analysis function. A category list that is not updated may no longer match the departments' responsibilities. A map library or tile service may change its usage policy. AI model behavior may drift as the types of images and civic conditions change. The system should therefore have named maintainers, documented configuration, version-controlled schema changes, and a release process that includes regression testing.

The codebase should separate environment-specific values from application logic. API keys and service credentials should be stored in protected environment configuration and never committed to a public repository. Database migrations should be versioned so that the schema can be recreated in a new environment. The AI function should have a documented input/output contract and sample test cases. A deployment checklist should include access policies, storage configuration, backup settings, logging, and rollback procedures.

Sustainability also includes the operational cost of image storage, map services, and AI inference. A future deployment should estimate expected monthly report volume, average image size, map request volume, and AI calls per report. Cost controls such as image compression, retention policies, caching, and usage monitoring may be needed. The source report does not include a cost or capacity analysis, so deployment sizing remains an open task rather than an established result.

9.5 Practical Adoption Considerations

The transition from an academic prototype to a service used by residents requires more than technical deployment. A municipal partner must agree on the scope of issues accepted by the platform, the geographic boundary covered, the departments responsible for each category, and the process for acknowledging and resolving a report. Without these agreements, the platform may collect useful observations but leave citizens uncertain about whether their reports have entered an official workflow. The authority should also decide which reports can be made visible on a public map and which details must remain restricted to authorized staff.

A staged introduction can reduce operational risk. During an internal demonstration, the project team can validate the main reporting and administrative flows using synthetic records. During a limited pilot, a small number of users can submit reports for a defined area and a small set of issue categories. The pilot team can then review whether location information is usable, whether the AI suggestions are understandable, whether administrators can process the queue, and whether status updates are communicated clearly. Only after these issues have been addressed should the service be considered for broader public access.

Training is required for both citizens and administrative users. Citizens need concise guidance on what makes a report actionable, how to capture a useful photograph, and how to confirm a location. Administrative users need guidance on

interpreting AI suggestions, correcting classifications, assigning departments, and updating status values. The training material should emphasize that an AI-generated severity or estimated resolution time is not a binding decision. A clear escalation path should be available for urgent hazards that require immediate contact with emergency or municipal services rather than waiting in a routine digital complaint queue.

The platform should not become the only channel through which a resident can report a civic problem. Some citizens may not have a compatible device, reliable internet access, or the ability to use a digital interface. A municipal deployment should consider how digital reports can coexist with existing phone, in-person, or assisted reporting channels. If reports are entered by staff on behalf of residents, the system should record that the submission was assisted and should not imply that the resident personally authenticated or supplied the information. This inclusive approach can help prevent a digital service from excluding the people who may need civic support most.

Finally, public communication should set realistic expectations. The application can acknowledge receipt, display the current processing stage, and provide a record of administrative updates. It cannot guarantee that every issue will be resolved within a particular period unless the responsible authority has established and committed to such a service level. Clear descriptions of the workflow, status meanings, data use, and support contact can help citizens understand what the platform does and where its responsibilities end. These adoption considerations are essential if CivicAI is to move from a project demonstration toward a dependable civic service.

10. CONCLUSION

This paper presented CivicAI, a crowdsourced civic issue reporting and resolution platform documented in the supplied mini-project report. The system brings together citizen authentication, multimedia issue submission, AI-assisted categorization, GPS-based location tagging, map visualization, complaint tracking, and administrative management. Its data model links a complaint to the authenticated user and stores both citizen-provided information and analysis-related attributes. The described workflow provides a foundation for organizing civic observations and improving visibility into complaint progress.

The source report provides module descriptions, interface screenshots, and code excerpts demonstrating Supabase authentication, an analysis function, PostgreSQL report storage, and user-level row security. It also describes a broader technology design involving React, Node.js, MongoDB, MapLibre, and OpenStreetMap; the exact implementation stack should be stated consistently in any final deployment documentation. Since the source does not report quantitative evaluation or a live municipal pilot, performance and real-world impact remain to be measured. Future work should prioritize systematic testing, verified AI performance, administrative security, municipal integration, and accessibility.

ACKNOWLEDGEMENT

The authors thank the Principal, Head of the Department, project coordinator, and faculty members of Adhiyamaan College of Engineering, Hosur, for providing the academic environment and support required to carry out this project.

REFERENCES

[1] M. A. Mugees, M. A. Ali, M. S. Ahmed, and Y. Ashwini, “CiviX – A Crowdsourced Civic Issue Reporting and Resolution System,” International Journal of Creative Research Thoughts (IJCRT), vol. 14, no. 3, pp. g264–g272, 2026. (Bibliographic details reproduced from the project report.)

[2] N. Ma, J. Fan, W. Wang, J. Wu, Y. Jiang, L. Xie, and R. Fan, “Computer Vision for Road Imaging and Pothole Detection: A State-of-the-Art Review of Systems and Algorithms,” Transportation Safety and Environment, vol. 4, no. 4, art. tdac026, 2022.

[3] Open311, “Open311 API: A Collaborative Model and Open Standard for Civic Issue Tracking,” 2010. (The project report does not provide a full publication or URL.)

[4] Y. Safyari, M. Mahdianpari, and H. Shiri, “A Review of Vision-Based Pothole Detection Methods Using Computer Vision and Machine Learning,” Sensors, vol. 24, no. 17, art. 5652, 2024.

[5] R. Rubin, “Pothole Detection and Assessment on Highways Using Enhanced YOLO Algorithm With Attention Mechanisms,” Advances in Civil Engineering, vol. 2025, art. 7911336, 2025.

[6] N. Tiwari, S. Singh, R. Singh, K. Siripuram, and N. Parmar, “AI-Driven Monitoring System for Civic Issues at Ward Level,” International Journal of Creative Research Thoughts (IJCRT), vol. 14, no. 4, pp. b255–b266, 2026.

[7] B. S. Menon, D. Rajeswari, and G. Mudbhari, “Civi-Snap: A Deep Learning Powered Geotagged Civic Issue Detection and Monitoring Framework,” IJCOPE Journal, vol. 2, no. 4, 2026.

[8] C. Navya, P. S. S. Dikshith, P. Sathwik, V. Sneha, and V. Mukthavali, “Citizen Complaint Management System,” American Journal of AI Cyber Computing Management, vol. 6, no. 1, pp. 164–166, 2026.

[9] C. Suganthi, S. Anusuya, V. Dhanusha, and M. Janani, “AI Based Smart Civic Complaint Management System,” International Journal of Engineering & Extended Technologies Research, DOI: 10.15662/IJEETR.2026.0802433, 2026.

[10] S. Pawar, V. Sargar, S. Shinde, J. Kachare, and A. Jadhav, “Crowdsourced Civic Issues Reporting and Resolution System,” International Research Journal on Advanced Engineering Hub (IRJAEH), vol. 4, no. 6, pp. 4268–4273, 2026. (Reference listed in the project report.)

[11] “CrowdSourcing Based Online Petitioning System for Pothole Detection Using Android Platform,” Procedia Computer Science, vol. 87, pp. 316–321, 2016. (Authors are not specified in the project report.)

[12] “A Web-Based Public Grievance Redressal System with Image and Geolocation Support,” IEEE International Conference publication, 2025. (Full author and conference details are not specified in the project report.)

[13] K. H. Kishore, V. L. Sirisha, M. Naveen, S. Hosamani, and S. B. Kulkarni, “CivicWatch: A Community-Driven Platform for Reporting and Resolving Local Infrastructure Issues,” International Journal of Creative Research Thoughts (IJCRT), vol. 13, no. 12, pp. f367–f372, 2025.

[14] N. Sudha, R. Nithya, M. Subasri, and R. Supritha, “Crowdsourced Civic Issue Reporting and Resolution System,” International Journal of Creative Research Thoughts (IJCRT), vol. 14, no. 3, pp. j555–j562, 2026.

[15] N. M. Deore, S. Dhadake, C. N. Bagal, S. Sontake, and A. S. Patel, “A Crowdsourced Complaint Resolution System Using Geotagging and Social Media Engagement,” International Research Journal of Modernization in Engineering Technology and Science (IRJMETS), vol. 7, no. 4, 2025.