CivicPulse: An AI-Assisted Platform for Municipal Complaint Management

Pranav Srivatsa D M.[1], Shivashankar B N.[2], Manoj Kumar N B.[3], Shravani V.[4], Chaithra P S.[5]
[1,2,3,4]*Student, Dept. of Computer Science and Engineering (Data Science), Vemana Institute of Technology, Bengaluru*
[5]*Assistant Professor, Dept. of Computer Science and Engineering (Data Science), Vemana Institute of Technology, Bengaluru*

Abstract*—Submitting a civic complaint is only the start of the work: someone must route it, assign it, record the response, and tell the resident what happened. CivicPulse brings those steps into one web application. Residents can send a geolocated report with supporting evidence, follow its status, and accept or reject a submitted resolution. Officers work from department queues; administrators see city-wide workloads and ward-level patterns. The prototype uses React, FastAPI, and a Supabasebacked JSONB repository. When configured, a multimodal language model suggests a category, department, and priority. A keyword-based classifier provides a fallback if the model service is unavailable. An end-to-end browser campaign covered 18 frontend capabilities, all marked as working in the demonstration environment. This shows that the main screens and a report to-tracking workflow were connected; it does not establish classification accuracy, performance under load, or improved municipal outcomes.

***Index Terms*—smart city, civic technology, complaint management, artificial intelligence, service-level agreement, role-based access control**

I. INTRODUCTION

A broken streetlight or leaking pipe can be easy to report and difficult to follow. After submission, a complaint still needs an owner, an assignment, a recorded response, and some form of follow-up with the resident. If each handoff happens in a different tool, residents may not know where the case stands and managers may have trouble locating delays. Smart-city research likewise treats participation and technology as only part of the problem; governance and the capacity to act on grievances matter too [1], [2], [3], [4], [5].

We designed CivicPulse to make these handoffs visible. At intake, the system proposes a category, department, and priority; the citizen can review those suggestions before submitting. Officers manage the case, and the citizen can accept or reject the proposed resolution. Automation therefore assists with intake but does not decide whether the work is complete. This design reflects the view of smart cities as a combination of technology, people, and institutions [6], [7], [8], [9]. The current prototype covers seven departments: Water & Sewage, Waste Management, Traffic, Police/Public Safety, Electricity, Roads & Infrastructure, and Parks. Complaints receive one of four priority levels and move through a recorded lifecycle that includes routing, assignment, work in progress, resolution submission, citizen verification, reopening, escalation, rejection, and cancellation.

This paper describes the complaint lifecycle implemented in CivicPulse, the triage design and its rules-based fallback, and the evidence gathered from browser testing. It also identifies what that evidence cannot establish. The work is a prototype description and workflow demonstration; it does not show that CivicPulse has improved city services.

Three questions guide the study: how can one platform make handoffs among residents, officers, and administrators visible; how can intake suggestions remain reviewable and correctable; and what does the current evaluation show, and leave unanswered?

II. RELATED WORK

Smart-city research links successful services to clear responsibilities, coordination between departments, public participation, and the ability to act on collected data [2], [3], [4], [8]. Complaint reporting illustrates the gap between intake and service: a well-designed form can collect reports, but it cannot ensure that a

department accepts the work, completes it on time, or checks the outcome. CivicPulse therefore follows a case beyond submission to citizen verification.

Current reporting channels range from telephone help desks and department websites to general grievance portals and crowdsourced maps. Each is useful in a different way. A common portal simplifies intake; a public map can reveal repeated problems in one location. Yet the resulting data is not automatically representative. Some neighborhoods report more actively than others, so a dense cluster of complaints may reflect access and awareness as well as actual infrastructure failure [18], [19]. CivicPulse keeps duplicate reports and their locations visible, but it's hotspot view is intended to supplement inspections and service records, not replace them.

Three areas of earlier work directly informed the design. Multimodal methods show how text, images, transcripts, and location can contribute complementary evidence [21]. Spatial clustering offers a practical foundation for recognizing groups of nearby incidents [20]. Human-in-the-loop studies, meanwhile, show why an algorithmic score should remain open to review when it affects a real decision [22]. CivicPulse brings these ideas together inside one workflow rather than evaluating them as isolated features.

Public platforms also provide useful points of comparison, although they do not share one mandate or operating model. FixMyStreet centers on place-based reports of local problems; NYC311 organizes a broad catalog of city-service requests; and India's CPGRAMS handles grievances across government organizations. CivicPulse is closer to the municipal issue-reporting end of this spectrum, while the other two demonstrate how request catalogs, agency routing, grievance tracking, feedback, and appeals can be organized at larger administrative scales. Table I compares documented functions. It is not a ranking: different jurisdictions, service responsibilities, and evidence bases make direct performance comparisons inappropriate.

The comparison sets a limit on our novelty claim. CivicPulse is not the first system to accept location-based reports, show maps, or route requests. Its contribution is a prototype that combines multimodal intake suggestions, a deterministic fallback, department queues, priority deadlines, and citizen verification. Whether that combination reduces staff effort or improves resident satisfaction remains to be tested.

TABLE I DESCRIPTIVE COMPARISON OF CIVICPULSE WITH SELECTED PUBLIC COMPLAINT AND GRIEVANCE SYSTEMS.

Dimension	CivicPulse	FixMyStreet	NYC311	CPGRAMS
Primary scope	Seven municipal departments; geolocated reports with text and optional media.	Local street and public-space problems reported to the responsible authority.	City service requests covering a broad catalog of NYC services.	Public grievances addressed by central and state government organizations.
Intake and routing	Citizen reviews model suggestions; department and priority values are validated by the API.	Location and report category help identify the appropriate council or authority.	Request type is selected from a service catalog; the NYC311 app offers AI-based suggestions for relevant service-request forms.	Complaints are routed to the concerned organization; IGMS also flags urgent cases, detects repetitive grievances, and extracts grievance themes.
Progress and response	Role-specific status history, officer assignment, resolution evidence, and citizen verification.	Reports and authority responses can be viewed and updated; available support and council workflow varies.	Request number and status lookup support follow-up; request channels include web, app, and telephone.	Registration-based status tracking, feedback, and an appeal route support follow-up.

Dimension	CivicPulse	FixMyStreet	NYC311	CPGRAMS
Spatial support	Ward maps and a prototype duplicate link for same-department reports within 200 m and 72 h.	Map-based reports expose nearby issues and help residents notice existing reports.	Service-request maps and public request data support geographic exploration.	Grievance dashboards focus on administrative status; street-level incident mapping is not the same service scope.
Operational view	SLA indicators, queues, departmental summaries, and ward hotspots in the prototype.	Authority-side tools depend on the local service deployment.	City agencies manage service requests through NYC311 workflows and public status tools.	Administrative dashboards support pendency and grievance monitoring across organizations.

Reading Note: This table summarizes public-facing descriptions and the CivicPulse implementation record; it is not a feature audit or benchmark. A capability not described in a cited source should not be interpreted as absent from that system. Sources: FixMyStreet [13], NYC311 [14], and CPGRAMS [15].

III. OBJECTIVES AND REQUIREMENTS

We used the following objectives to guide implementation:

1) Provide a short, understandable reporting flow for citizens;
2) Automatically suggest issue type, department, priority, and safety risk;
3) Route each complaint into a department-specific operational queue;
4) Make deadlines and stalled work visible through priority-based SLAs;
5) Preserve a traceable lifecycle with notifications and audit events;
6) Help officers prioritize work using queues, maps, and evidence;
7) Help administrators identify trends, hotspots, and depart-mental performance; and
8) Enforce distinct permissions for citizens, officers, and administrators.

These objectives led to three distinct workspaces. Citizens mainly report and follow issues. Officers need a focused view of the queue owned by their department. Administrators need the city-wide picture. Public pages explain the service, while protected routes expose only the tools allowed for the signed-in role. The objectives imply more than interface functions. A complaint must retain enough context to be actionable without exposing unnecessary personal data; an automated suggestion must be correctable; and a resolution must return to the reporter for verification. The prototype therefore treats traceability, least-privilege access, graceful degradation, and operational visibility as cross-cutting requirements. Table II connects these goals to implemented mechanisms and identifies where the current implementation offers only a starting point.

Several quality attributes remain design targets rather than verified properties. In particular, availability depends on the API, database, and configured model provider; data consistency depends on write behavior in the repository; and accessibility is not established by the use of responsive components alone. The paper uses the terms “implemented,” “tested,” and “proposed” distinctly: implemented denotes behavior present in the code, tested denotes behavior observed in the documented campaign, and proposed denotes work that still requires evidence or implementation.

TABLE II REQUIREMENTS AND THEIR IMPLEMENTATION IN THE CIVICPULSE PROTOTYPE.

Requirement	Implemented mechanism	Boundary or remaining evidence need
Low-friction reporting	Five-step citizen flow for evidence, description, location, analysis, and confirmation.	Completion time, accessibility, and language coverage have not been measured with representative residents.

Requirement	Implemented mechanism	Boundary or remaining evidence need
Reviewable triage	Model suggestions populate editable fields; deterministic rules are used if the model is unavailable.	No labeled corpus establishes routing quality, critical-case recall, or confidence calibration.
Traceable case handling	Tracking ID, status history, event notes, notifications, and audit records.	Audit-log retention, tamper resistance, and operational review procedures need production design.
Department accountability	Department-scoped queues, assignments, SLA deadlines, and escalation states.	Staffing levels, workload policy, and city-specific escalation rules are outside the prototype evaluation.
Citizen verification	Officer submits work notes or after-image evidence; citizen may accept or reopen a resolution.	Verification participation and the reasons for disagreement require a field study.
Privacy by role	API authorization and response filtering limit access to sensitive complaint fields.	No independent penetration test or privacy-impact assessment has been completed.
Operational visibility	Role-specific dashboards, maps, scheduled reports, and complaint summaries.	Current values are computed from demo-scale records; they are not forecasts or validated performance indicators.
Graceful degradation	Rules classifier remains available after missing configuration or model-call failure.	Fallback quality and service availability during prolonged provider outages remain unmeasured.

IV. SYSTEM ARCHITECTURE

The implementation has three main layers. A React client provides separate navigation and pages for citizens, officers, and administrators. Those pages call a REST-style FastAPI service. The service stores users, complaints, notifications, reports, and audit events through a repository built on a Supabase JSONB table [10], [11], [12], [16]. Keeping the interface, business rules, analysis, and storage concerns separate also matches the layered approach commonly used in connected-city systems [17].

The API is the shared source of business rules. Pydantic schemas check authentication, location, media, complaint, assignment, status, resolution, and feedback inputs. It also publishes the allowed departments, wards, priorities, statuses, and SLA windows.

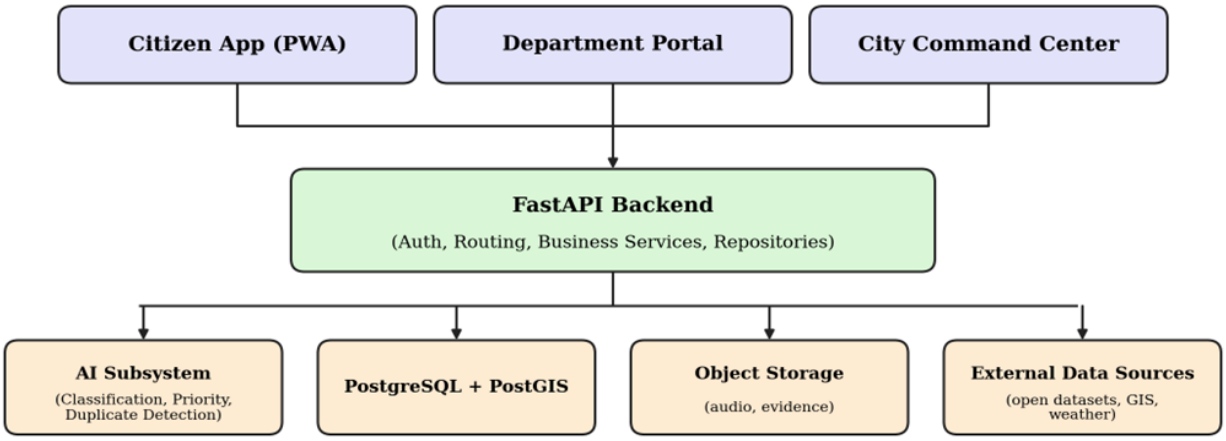

Fig. 1. Conceptual target architecture for the three role-specific applications and their shared services. The evaluated prototype retains the FastAPI and AI layers but uses a Supabase-backed JSONB repository in place of the illustrated PostgreSQL/PostGIS and separate object-storage components

As a result, the three interfaces use the same vocabulary instead of maintaining separate copies of these rules. The storage adapter exposes collection-like operations over Supabase, which kept the application code compact while the data model was still changing. It supports nested-field queries, sorting, projections, and updates without spreading database calls across the service. There is a clear scaling cost, however: several filters currently run in application code. A production version should move common fields and aggregations into indexed, server-side queries, especially for department, status, ward, and SLA views [12].

A. Request path and component boundaries

A normal complaint action follows a consistent path. The browser collects user input and calls a FastAPI route through the shared client library. The API first resolves the bearer token, then validates the request body, checks the user role and (for officers) department scope, and invokes domain logic. A repository operation persists the change; response shaping removes fields the caller should not receive. Notifications and audit events are emitted by the same server-side workflow. This structure keeps policy enforcement out of client-only route guards and makes the backend the authority for state changes. The separation is also a trust boundary. The browser does not receive the Supabase service key or model-provider key and does not write directly to the document table. External model and city-signal requests originate from the backend. The current server fetches weather and air-quality observations from external services and caches the result in process memory; a fallback weather provider is attempted when the primary request fails. That cache is local to one running process, so it is not a distributed cache and can be lost on restart. These integrations enrich the administrative view, but complaint intake and status handling do not depend on them.

The API groups behavior by domain rather than by screen.

For example, both a queue card and a complaint-detail page use the same assignment route. Metadata routes provide enumerations, while complaint routes own creation and lifecycle updates. Analytics routes read complaint records and return aggregate data; they do not create a second authoritative complaint store. This shared service boundary reduces the risk that a citizen page and an officer page encode inconsistent definitions of status or priority.

The repository currently trades query efficiency for a small adapter surface. Its collection wrapper retrieves all rows in a named collection, then applies several nested-field filters, sorts, and projections in application memory. Although the primary table has a collection index and a composite collection/document key, JSON fields such as department, ward, status, and due time are not all first-class indexed columns. This means request cost grows with collection size and can consume application memory. The adapter also calls a synchronous database client from asynchronous route functions; under concurrent load this may block the event loop. The current design is acceptable for a functional demonstrator but should be replaced or profiled before a live municipal service.

Complaint changes, notifications, and audit rows are separate persistence operations. The complaint document stores its status and history together, but the subsequent notification and audit insert are not part of a single transaction with that update. A transient database failure can therefore leave a valid status change without its corresponding notification or audit event. A production implementation should use transactional writes or a durable outbox, and add retry/idempotency behavior for external calls. These are reliability requirements, not capabilities established by the browser evaluation.

V. CORE WORKFLOW AND DOMAIN LOGIC

A. Citizen reporting

Reporting is divided into five short steps: evidence, description, location, analysis, and confirmation. A citizen can attach media, explain the problem in ordinary language, choose a ward and address, and review the suggested routing before submitting. The system then creates a readable tracking number. From the citizen workspace, the reporter can follow the timeline, read notifications, support a nearby complaint, and accept or reject an officer's resolution. This design recognizes both the value of crowdsourced services and the uneven readiness of residents to use them [18], [19].

Location is useful beyond displaying a marker. Officers can see where work is concentrated, and the backend can check whether a new report resembles an active incident nearby. The current rule searches the

same department for unresolved complaints created during the previous 72 hours and located within 200 meters. Distance is calculated with the haversine formula,

$$d = 2R\arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right)}\right) \quad (1)$$

where R is the Earth's radius, ϕ is latitude, and λ is longitude. A match links the reports under a common group instead of deleting either citizen's submission. The rule is deliberately simple for the prototype; a larger deployment could compare it with density-based methods such as DBSCAN [20].

B. AI-assisted triage

When a model key is available, the backend sends the description, optional image, and location context to a multimodal model. The requested response is deliberately narrow: detected issue, department, category, priority, confidence, short reasoning, tags, and a safety flag. The backend does not trust that response blindly. It rejects unsupported department or priority values, clamps confidence to [0, 100], and limits returned text and lists. The use of several evidence types follows established multimodal-fusion principles [21].

Figure 2b presents the intake pipeline. The model assists with issue description, department, category, priority, and safety flag; duplicate grouping is applied separately by the server's time-and-distance rule after the citizen submits the report.

A deterministic classifier provides a fallback. It counts department-specific keywords and checks the description for severity phrases associated with medium, high, or critical cases. When no department keyword is found, it returns Roads & Infrastructure with low confidence. The result is less flexible than the model response, but it keeps complaint submission available during configuration errors or external failures. It also gives the team a reproducible baseline for later model comparisons.

In both paths, the output is advice rather than a final decision. The interface shows the proposed department and priority and lets the citizen review them. Officers can correct the case later as well. A displayed confidence value should not be mistaken for a calibrated probability, so future deployment requires measured error rates and documented override procedures. This approach is consistent with human-in-the-loop research and the NIST AI Risk Management Framework [22], [23].

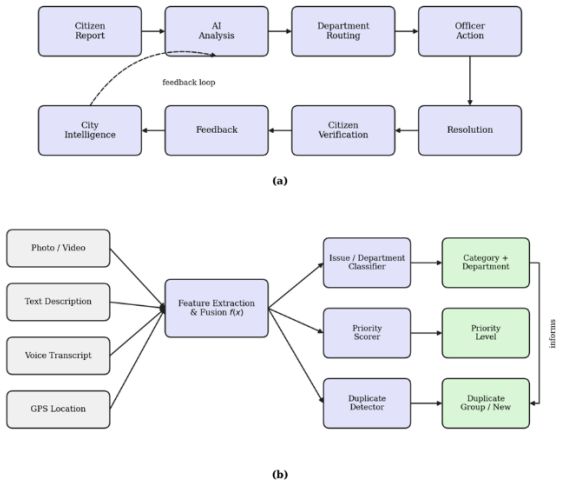

Fig. 2. CivicPulse process models: (a) end-to-end complaint workflow with a city-intelligence feedback loop; and (b) multimodal AI triage followed by a separate time-and-distance duplicate check

C. Model contract and deterministic baseline

In the documented configuration, the backend invokes a Gemini 2.5 Flash model through an integration library when an API key is present. The request includes the citizen's description, optional image, and available address, coordinates, and ward. The service asks for a constrained JSON response rather than an unrestricted narrative. A configured model call is therefore a remote dependency: provider availability, model-version changes, network delay, and data-handling terms all affect the service and must be reviewed before real complaint information is sent to it. The tested interface accepts voice-related media as a transcript, but the analysis route does not establish direct speech recognition from raw audio.

The rules path is intentionally simple enough to reproduce. For description text x and department keyword set K_d, it computes

$$s_d(x) = \sum_{k \in K_d} \mathbb{1}[k \subseteq \text{lower}(x)], \qquad \hat{d} = \arg\max_d s_d(x). \quad (2)$$

If every score is zero, the implementation assigns Roads & Infrastructure as a default route. Otherwise, ties follow the department ordering in the code. Severity words are checked in the fixed order Critical,

High, then Medium; if none match, the default is Low. The safety flag is set by the fallback when a Critical phrase is found. Its confidence is a heuristic based on the number of matched department keywords, capped at 92, with 45 used when no keyword matched. This value is a rule score, not a probability.

The baseline has useful operational properties: it needs no model key, returns quickly, and keeps the reporting path available after common integration failures. It also has predictable weaknesses. Substring matching is insensitive to context, negation, spelling variation, and the difference between a reported hazard and a resolved one. A phrase such as "no exposed wire" can still trigger a safety keyword; multilingual descriptions are not handled by the English keyword maps; and the fixed default department may be wrong. These limitations are reasons to keep the suggestion editable and to measure safety-critical false negatives before relying on it. The model response is not passed straight through to storage. Department and priority values are checked against server enumerations, confidence is clipped to the range [0, 100], and strings and tag lists are length-limited. If JSON parsing fails, a required field is unusable, the provider errors, or configuration is absent, the rules result is returned. These checks constrain format and vocabulary, but they do not validate whether the model's factual explanation is correct. The UI presents the returned confidence and reasoning as triage context; neither should be treated as a verified diagnosis.

An additional distinction is important for future evaluation. The user may edit category, department, and priority after analysis, and the complaint record retains both the AI prediction and the submitted fields. This supports an eventual override analysis: which fields changed, how often, and whether changes improve outcomes. The current test record did not report those quantities. No conclusion about automation accuracy or staff acceptance can therefore be drawn from the successful example alone.

D. SLA and lifecycle management

Priority determines the initial deadline. The windows used by the prototype are listed in Table III.

TABLE III PRIORITY-BASED SLA WINDOWS.

Priority	Deadline	Priority	Deadline
Critical	6 hours	Medium	72 hours
High	24 hours	Low	168 hours

Officers can search their department queue, claim a case, add notes, change its status, and submit before-and-after evidence with a work description. Submitting that evidence does not immediately close the complaint. The citizen first receives a verification request and can either confirm the result or send the case back for more work. Dashboards call attention to overdue items, and the lifecycle history records who performed each significant action.

For a new record created at time t_0, the initial deadline is calculated as

$$t_{\text{due}} = t_0 + \Delta(p), \quad \Delta(p) \in \{6, 24, 72, 168\} \text{ hours} \tag{3}$$

where p is Critical, High, Medium, or Low, respectively. Breach is derived by comparing the stored due time with the current time when complaint lists and analytics are read; it is not a separate immutable event. Resolved, rejected, and cancelled items are treated as terminal for the active-case calculation. The prototype does not define a general pause rule for weekends, jurisdictional closure hours, appeals, or a case awaiting citizen verification. Those details must be agreed with the adopting municipality before the SLA indicator is interpreted as a contractual service target.

Table IV describes the status vocabulary and its role in the workflow. It is a state catalogue rather than a formal transition graph. The current status endpoint checks whether a value belongs to the allowed enumeration but does not enforce every permissible predecessor-successor pair. That means a privileged client can request a status jump that would be disallowed by a stricter production state machine. The citizen verification endpoint has a narrower behavior: confirmation marks the case Resolved, while rejection records feedback and changes it to Reopened.

Retaining a chronological history is more informative than keeping only the latest status. For a supervisor, the history can show whether a case stalled before assignment, during field work, or after a resolution request. For the reporter, it provides a human-readable explanation of movement. In the current document model, each event includes the resulting status, actor, timestamp, and optional note. A production workflow should additionally encode transition permissions, reason codes, time spent in each stage, and rules for correcting erroneous history entries without erasing the audit trail.

TABLE IV COMPLAINT STATUS CATALOGUE AND OPERATIONAL MEANING IN THE PROTOTYPE.

Status	Meaning in the recorded workflow	Typical event or control
NEW	Intake event retained in the history before departmental routing.	Citizen submits a valid complaint.
ROUTED	Complaint has a department and priority and is awaiting assignment or further handling.	Created record is placed in the selected department queue.
ASSIGNED	An officer is associated with the complaint.	Officer claims the case or an administrator assigns it.
IN_PROGRESS	Field work has started.	Department officer records a work-status update.
RESOLUTION_SUBMITTED	Work evidence is available and the reporter is asked to verify it.	Officer submits an after-image or work note.
RESOLVED	Reporter confirmed the submitted result.	Citizen verification marks the result confirmed.
REOPENED	Reporter did not accept the submitted result; more work is expected.	Citizen rejects resolution and may provide a reason.
ESCALATED	Case is flagged for additional attention.	Privileged status update; escalation policy is not formalized.
REJECTED	Case is administratively rejected.	Privileged status update; rejection grounds and appeal policy require governance.
CANCELLED	Case is marked as cancelled.	Privileged status update; citizen-initiated cancellation rules are not established.

VI. ROLE-BASED INTERFACES

The client redirects a user who opens a page outside the permitted role, but this is only a usability control. The API performs its own checks before returning protected data or changing a complaint. The division between citizen, officer, and administrator permissions follows the established role-based access-control model [26].

The three workspaces expose different decisions, not just different menu items. A citizen needs a personal case history and nearby public reports, but should not see another resident's contact details. An officer's view is bounded by department and emphasizes unresolved workload, urgency, location, and evidence required to complete field work. An administrator can compare departments and manage users, so that role has the broadest access and correspondingly requires the strongest pro-visioning controls. This separation reduces incidental exposure and lets each interface foreground the actions most relevant to the user.

TABLE V PRIMARY CAPABILITIES BY ROLE.

Role	Capabilities
Citizen	Report and track issues; review AI output; receive alerts; verify, rate, or reopen resolutions.
Officer	Manage the department queue; assign and update work; submit evidence; view maps and analytics.
Admin	Monitor city KPIs, complaints, hotspots, departments, reports, data sources, and users.

Several interactions cross role boundaries. A citizen's report becomes an officer's queue item; an officer's resolution becomes a citizen's verification task; and an administrator's overview summarizes events generated by both. Each handoff creates a notification and a history entry. These events form the operational explanation for why a complaint changed state. If notifications fail, the underlying case state should still be recoverable from the timeline; if the timeline is incomplete, a notification alone should not be treated as the authoritative record.

VII. IMPLEMENTATION AND DATA MANAGEMENT

A. Frontend organization

The React code is split into public, citizen, officer, and administrator routes. Reusable components handle complaint cards, badges, confidence indicators, timelines, maps, loading states, and report summaries. Each role has its own layout shell: citizen pages favor a simpler mobile flow, whereas officer and administrator pages can use denser dashboards. A shared authentication context tracks the current user, and API calls live in a small client library rather than being repeated inside page components.

We chose a staged report form instead of one long municipal form. A resident can focus first on the evidence, then on the description and location, before seeing any classification fields. Progress is retained while moving between steps. The analysis result fills editable controls, so the citizen can correct an obvious routing mistake before the complaint reaches a department.

B. Backend services and API

The FastAPI service groups routes by task. Authentication routes register users, issue tokens, and return the active profile. Metadata routes provide the valid departments, wards, statuses, priorities, and SLA settings. Complaint routes cover analysis, creation, search, retrieval, assignment, status changes, resolution, verification, community support, and duplicate groups. Other routes serve notifications, maps, analytics, reports, scheduler controls, data-source information, and administrator views.

Request schemas catch missing or malformed fields before the main logic runs. Role checks are attached to the relevant endpoints, while shared functions handle status transitions, notifications, audit entries, duplicate grouping, and the removal of private fields. This avoids subtle differences between two API specification: request schemas, status codes, pagination policy, and versioning should be published separately before third-party clients are supported.

The endpoint grouping has two design consequences. First, permissions must be evaluated against the requested object as well as the role: an officer should not read or change a complaint outside their department, and a citizen should not access a report owned by another account. Second, returning only a subset of rows is not by itself a complete authorization strategy. Query filters, object lookups, mutation routes, and response serialization all need consistent checks. The current code applies several such checks at the complaint endpoints, but a production review should cover the full route inventory and test both allowed and forbidden cases for every role.

TABLE VI CIVICPULSE API DOMAINS AND PRINCIPAL OPERATIONS.

Domain	Representative routes and responsibilities	Principal access scope
Authentication and metadata	/api/auth/register, /api/auth/login, /api/auth/me, /api/meta, /api/departments.	Public registration/login; authenticated profile and metadata.
Intake and discovery	/api/ai/analyze, complaint create/list/detail, nearby complaints, incident groups, and community support.	Analysis requires login; creation and support are citizen-only; lists are role-filtered.
Department operations	Assignment, status updates, resolution submission, citizen verification, and feedback.	Officer/admin for department work; complaint owner for verification and feedback.
Notifications	List, mark-one-read, and mark-all-read operations.	Authenticated users; results are scoped to the current user.
Analytics and mapping	Overview statistics, department performance, map points, and city signals.	Officer views are department-scoped; city-wide views require admin where applicable.
Reports and administration	Scheduled report list/detail, source registry, scheduler status/run, and user listing.	Officer/admin for reports; admin for registry, users, and manual scheduler execution.

C. Complaint representation and persistence

The prototype keeps its logical collections in one Supabase table with a collection name, document ID, and JSONB body. A repository wrapper supplies lookup, filtering, sorting, insertion, update, deletion, and count operations. This was convenient while complaint fields and demo features were changing, and it preserved nested records without frequent schema migrations. It is not the final scaling design. Common query fields—department, status, priority, ward, creation time, and SLA deadline—should eventually be indexed directly, and the database should perform the heavier aggregations.

Table VII lists the main groups of fields in a complaint. Lifecycle events are kept as a history rather than overwritten; each one records the new status, actor, time, and an optional note. The original AI suggestion is also retained beside the submitted values. This makes it possible to study later whether people corrected the automated routing. Resolution evidence and citizen feedback remain linked to the same record, allowing the case to be reconstructed from intake to closure.

TABLE VII CORE COMPLAINT RECORD GROUPS.

Group	Representative fields
Identity	Internal ID, tracking ID, citizen ID, creation time
Issue	Title, description, category, department, priority
Location	Latitude, longitude, address, ward
AI output	Detected issue, confidence, reasoning, tags, safety flag
Workflow	Status, assignment, SLA deadline, lifecycle history
Evidence	Citizen media, before/after images, officer work note
Outcome	Verification, rating, feedback, reopen reason

Notifications belong to individual users and are created for events such as submission, assignment, status changes, verification requests, and escalation. Audit entries serve a different purpose: they record actions that may matter during security or administrative review. A scheduled reporting task aggregates complaints into fixed windows and stores each report. Officers on different shifts can therefore refer to the same snapshot rather than seeing numbers that change every time the page loads.

The repository separates logical collections even though they share one physical table. Table VIII summarizes this arrangement. The pattern lets the application introduce a document type without a new relational table, but it also means that database constraints on nested values are limited. A malformed JSON body may remain structurally valid in the database while violating an application assumption. Schema validation at API boundaries, migration/version fields, and periodic integrity checks are therefore important as the model evolves.

TABLE VIII LOGICAL COLLECTIONS STORED IN THE SHARED DOCUMENT TABLE.

Collection	Purpose
users	Accounts, role, department, and authentication metadata.
complaints	Intake data, location, status history, resolution, and feedback.
notifications	Per-user event notices and read state.
audit_logs	Actor, action, target, and event time.
ai_predictions	Analysis response associated with a requesting user and time.
analytics_reports	Stored city- or department-level report snapshots.
api_sources	Registry of data sources and synchronization metadata.
scheduler_state	Report-cycle timing and execution state.

VIII. SECURITY AND PRIVACY

Passwords are derived with PBKDF2-HMAC-SHA256, a random salt for each user, and 120,000 iterations, using the password-based key-derivation family described in NIST SP 800-132 [24]. The iteration count and minimum password length in the prototype should be reviewed against current deployment policy rather than treated as a compliance claim. A successful login returns a signed JSON Web Token containing the user ID, role, issue time, and a seven-day expiry; the token format is defined by RFC

7519 [25]. On a protected request, the API validates the bearer token, reloads the user, and checks the required role. A signed token provides integrity, not encryption, and the seven-day validity period means the current design has no immediate token revocation mechanism. The response layer removes information that a viewer does not need. Email addresses and phone numbers are excluded from normal complaint responses, while public views also omit citizen names and identifiers. Cross-origin access is controlled through configuration, and deployment secrets—the Supabase service key, JWT secret, and model key—are read from the environment rather than stored in the application source.

The database enables row-level security and is intended to be inaccessible directly to browser clients. The API uses a server-only service-role key, so the database policy does not replace the application authorization checks: compromise of the service key or an incorrectly guarded route can expose the shared document store. The development signing-secret fallback must be rejected at startup in production. Because complaints may contain faces, addresses, voices, and precise locations, a privacy impact assessment is also necessary before real public use.

Code inspection identifies a particularly important privilege-provisioning gap. The public registration request contains a role field, and the backend accepts any recognized value, including officer and admin; only unrecognized values are changed to citizen. Therefore, endpoint role checks alone do not protect a deployment that exposes this registration route: a caller can request a privileged account at registration [16]. The browser test verified that a logged-in citizen was redirected away from officer and administrator pages, but it did not test privileged self-registration. Role provisioning must be restricted server-side (for example, self-registration always creates citizens and staff accounts require an administrator-controlled invitation) before CivicPulse can claim an effective role boundary. Table IX distinguishes implemented controls from unresolved threats. It is a design review based on the source and test record, not a penetration-test result. This distinction matters because security properties are often conditional: a UI may hide a control while its API remains callable, and an API may check a role while account creation lets an untrusted user obtain that role.

TABLE IX SECURITY AND PRIVACY CONTROLS, TOGETHER WITH OPEN RISKS IDENTIFIED IN THE PROTOTYPE.

Threat or asset	Existing control in the demonstrator	Open risk or required production control
Privileged account creation	API routes declare role requirements; client routes also redirect by role.	Public registration accepts client-supplied recognized roles, including administrator; restrict account provisioning before deployment.
Credential compromise	Per-user salt, PBKDF2-derived password hash, signed short-format bearer token.	Review password policy and work factor; add rate limiting, recovery, revocation, and secret rotation.
Cross-role data exposure	Complaint routes apply citizen ownership and officer department checks; responses filter some contact fields.	Test every object and route; define which staff can see reporter identity and precise coordinates.
Database access	Service key is held by the backend; table-level row security is enabled for direct client access.	Service-key compromise bypasses user-level policy; use least privilege, rotate secrets, and monitor privileged actions.
Evidence misuse	Media is validated as part of request schemas and associated with a complaint.	File size/type limits, malware scanning, storage lifecycle, face/voice handling, and deletion workflows are not established.
Audit integrity	Business actions create audit rows with actor, role, action, metadata, and timestamp.	Audit writes are not transactionally coupled to complaint changes and are not shown to be tamper-evident or immutable.
Provider disclosure	Model key remains server-side; rules fallback works when the provider is unavailable.	Text, optional image, address, ward, and coordinates may leave the municipal service; consent, minimization, retention, and vendor review are needed.

Privacy should be treated as part of the workflow, not only as removal of a name from a response. A precise location can identify a household; photographs can include people, license plates, or private property; and voice transcripts may contain names or health and safety details. Before a live pilot, the city should define the purpose and retention period for each data field, whether the reporter can submit anonymously, which staff roles can reveal identity, how correction and deletion requests are handled, and what information is sent to the model provider. Aggregate dashboards should avoid exposing low-count groups or exact coordinates when a ward-level summary is sufficient. The current test data is explicitly seeded and marked as demonstration data in the project. That is suitable for screenshots and functional flow checks, but it cannot stand in for consented resident records or establish that the system is safe with real data. Production readiness therefore requires both a technical security review and a governance decision about data ownership, public-record obligations, incident response, and accountability for automated recommendations.

IX. ANALYTICS AND OPERATIONAL INTELLIGENCE

The analytics pages turn complaint records into working indicators: active cases, priority mix, SLA breaches, resolution rate, average resolution time, reopening, satisfaction, departmental load, and ward hotspots. Officers see their own department, whereas administrators can compare the full city view.

Every 12 hours, the scheduler produces a report containing new, resolved, active, critical, and overdue cases, along with hotspots, repeated issues, and suggested actions. Such a report can support a shift handover or a staffing discussion. It should not be read as a forecast. If residents underreport in one ward or a department updates cases late, the summary will carry the same weakness.

The indicators are operational summaries rather than validated outcome measures. Their definitions matter when an administrator compares wards or departments. Table X documents how the current API computes several of them and what can be inferred from each value. In particular, a "resolution rate" over all stored complaints is not the same as the proportion of eligible cases resolved within a fixed cohort or within their SLA window.

TABLE X CURRENT DASHBOARD MEASURES AND INTERPRETATION CAUTIONS.

Indicator	Prototype calculation	Interpretation caution
Active complaints	Count whose status is not Resolved, Rejected, or Cancelled.	Includes reopened and escalated cases; a deployment should agree how paused or disputed work is treated.
Resolution rate	Resolved count divided by total complaint count, reported as a percentage.	The denominator includes all stored cases; it is not an SLA-compliance rate or a controlled time-to-resolution estimate.
Average resolution time	Mean time from complaint creation to the final history event for records currently marked Resolved.	Reopened cases and missing or malformed timestamps can affect inclusion; use a defined cohort for formal reporting.
SLA breaches	Active cases with a due timestamp earlier than the time of the query.	A live query-time count, not an immutable breach event; it does not model operating hours or pause rules.
Satisfaction	Arithmetic mean of available citizen ratings.	Ratings are voluntary and may be sparse or skewed toward unusually positive or negative experiences.
Ward hotspots	Complaint count grouped by ward, with the six largest groups returned.	Current code counts all complaint statuses; the visualization is not a map of only unresolved need.

Indicator	Prototype calculation	Interpretation caution
Fourteen-day trend	Daily submitted counts and resolved counts based on the last history event date.	A short descriptive series; it should not be interpreted as a forecast or a causal effect.

The platform also retrieves contextual city signals for Bengaluru: temperature, humidity, precipitation, wind speed, weather code, and air-quality values. The weather request uses a primary source and a secondary fallback; the air-quality request can return a city-level value and values associated with configured ward coordinates. Results are cached in the server process for about twenty minutes. This is a contextual panel, not a validated model input: the current implementation does not establish that weather or pollution causes a particular complaint, nor does it join those signals to a complaint outcome for causal analysis.

Hotspot analysis has a participation problem as well as a technical one. A ward with more reports may have more infrastructure faults, greater familiarity with the reporting application, stronger mobile access, or some combination of these factors. Counts should be considered alongside inspections, population and service-area denominators, dispatch records, and outreach in wards with low reporting volume. Where complaint counts are small, an exact map marker can also reveal a reporter's location; aggregation and suppression rules should be designed with municipal privacy staff.

Scheduled reports make a useful snapshot for an operational handover, but their schedule and recipient list need governance. The administrator can trigger a report cycle manually in the prototype, while officers receive department-specific views. A production service should record the aggregation window, generation timestamp, data freshness, and any failed source calls alongside each report. Otherwise, two snapshots can appear comparable even when their input data or refresh status differs.

X. EVALUATION METHODOLOGY

Evaluation needs to answer several different questions. Do the pages and status changes work? Are the routing suggestions accurate? Can the service remain responsive under load? Most importantly, does it help a department resolve complaints more effectively? Passing a browser test answers only the first question. Table XI separates the evidence needed for functional behavior, AI support, performance, and operational outcomes. An evaluation should distinguish four evidence levels.

TABLE XI PROPOSED EVALUATION DIMENSIONS.

Dimension	Measures
Functional	Task completion, transition validity, RBAC, error recovery
AI triage	Accuracy, macro-F1, critical recall, calibration, override rate
Duplicates	Pair precision/recall, cluster purity, false merges
Performance	API latency, analysis latency, throughput, failure rate
Operations	Assignment time, resolution time, SLA compliance, reopen rate
Experience	Completion time, accessibility, satisfaction, abandonment

Component testing asks whether an endpoint rejects invalid data and returns the expected schema. Workflow testing asks whether actors can complete the sequence across pages and roles. Model testing compares predictions with expert labels. Field evaluation asks whether the workflow changes municipal service outcomes relative to existing practice. Evidence at one level does not automatically support claims at the next: a working route does not show that its classifier is accurate, and an accurate classifier does not prove faster resolution.

For an AI study, the evaluation corpus should be sampled from de-identified reports representing all departments, priori-ties, wards, modalities, and supported languages. Two municipal reviewers should independently assign department, category, priority, and safety labels, with disagreements adjudicated before scoring. A held-out test set should be separated from prompt development and rule tuning. Because a hosted model can change over time and may be nondeterministic, the study should record provider, model identifier, prompt version, request date, and whether identical cases are repeated. Report per-class

precision and recall, macro-F1, critical-case recall, calibration error, and the proportion of recommendations changed by citizens or officers. A single accuracy figure can hide poor recall on rare hazards.

For classification, municipal-domain reviewers should label a representative set independently of the system output. The sample should cover each department, priority, ward, input type, and supported language. Otherwise, a large number of routine reports could hide poor performance on rare safety cases. Given true positives *TP*, false positives *FP*, and false negatives *FN*, precision P and recall R are

$$P = \frac{\mathrm{TP}}{\mathrm{TP+FP}} \quad , R = \frac{\mathrm{TP}}{\mathrm{TP+FN}} \quad , F_1 = \frac{2PR}{P+R} \qquad (4)$$

Overall accuracy is not enough when departments appear at different frequencies, so macro-F1 should also be reported. Critical recall deserves its own result: missing one exposed-wire report may matter more than several mistakes on routine maintenance requests. Confidence should be checked with a reliability plot or calibration measure. The override rate is useful too, since frequent citizen or officer corrections would expose a weakness even when a top-line score looks acceptable. Duplicate detection needs a different test set. Reviewers should mark whether pairs refer to the same physical incident. Pair precision shows how often a proposed match is correct, and recall shows how many genuine duplicates are found. The grouped result should also be inspected for false merges and fragmented incidents. Testing location, time, department, text, and image signals separately would show whether a more complex method actually improves on the current 200-meter and 72-hour rule.

Duplicate evaluation requires a different unit of analysis. The test set should contain report pairs sampled across distance and time bands, not only the pairs already retrieved by the 200-meter rule. Reviewers should determine whether each pair refers to the same physical incident. Pair precision and recall should be reported alongside false-merge examples, cluster fragmentation, and the number of reports preserved in a group.

If image or text similarity is added later, ablation comparisons can test whether it improves matching beyond department, time, and distance.

For system performance, load tests should exercise realistic sequences rather than one isolated endpoint. A representative mix would include citizen submission, department queue filtering, complaint-detail retrieval, map-point queries, and analytics generation at several data volumes. Report throughput, error rate, and median, 95th-percentile, and 99th-percentile latency; separate AI-provider latency from internal API time; and record concurrent users, database size, and test hardware. The in-memory repository filtering and synchronous database calls make data volume and concurrency particularly relevant for this implementation.

Performance tests should use concurrent citizen and officer activity, not just one browser session at a time. Useful measurements include median and tail latency for list, detail, creation, and analytics requests; analysis time for both the external model and fallback; notification delay; scheduler duration; and failure rate as load increases. Large datasets are especially important because some repository filtering still occurs in memory. Security tests should cover expired tokens, role boundaries, access to another user's objects, malformed payloads, upload controls, rate limits, and removal of private fields.

Finally, a pilot should compare CivicPulse with the municipality's current intake process. A phased deployment or matched comparison wards could help distinguish product effects from seasonal variation, staffing changes, and shifts in complaint mix. Predefine primary outcomes such as time from submission to assignment, time to first officer action, time to verified resolution, SLA compliance, reopening, and satisfaction. Track participation rates by ward and channel as well: apparent service improvement can coexist with under-reporting if some residents cannot or do not use the platform. Interviews with residents and officers are needed to explain why a metric changed, not merely whether it changed.

A field pilot should then measure what happens after deployment. Relevant outcomes include time to assignment, time to first officer action, resolution time by priority, SLA compliance, verification completion, satisfaction, and reopen rate. The reporting period and sample size should be stated, with uncertainty intervals where they make sense. A comparison with the previous process or a phased rollout would be stronger than a single snapshot, although staffing, seasonality, complaint mix, and policy changes would still complicate causal claims.

XI. FUNCTIONAL EVALUATION

The available test record describes a browser evaluation completed on August 15, 2026. It covers 18 major frontend capabilities: the public landing page, authentication, the three role workspaces, the complete citizen reporting flow, complaint handling, maps, analytics, reports, data-source and user pages, and role-based access control. Each capability was marked as working in the test environment [16].

In the main citizen scenario, the tester completed all five report steps, called the configured analysis endpoint, received a response in roughly two to three seconds, submitted the case, and opened its tracking page. Other checks covered citizen lists and alerts, the officer queue and resolution controls, department and city maps, analytics dashboards, scheduled reports, administrator search, and redirects from unauthorized routes. No critical functional defect was recorded. Two minor console problems remained—an external analytics request failure and chart-dimension warnings—but neither stopped a tested workflow.

Table XII summarizes what the recorded campaign does and does not show. The test record is a structured browser checklist completed on August 15, 2026, rather than a randomized user study or a continuous-integration test suite. Its strongest evidence is that a connected path existed from citizen intake to a tracking page and that role-specific screens rendered against seeded records. The observed AI response of approximately two to three seconds is one successful request and should not be reported as mean or percentile latency

TABLE XII COVERAGE OF THE DOCUMENTED END-TO-END BROWSER CAMPAIGN.

Area	Recorded browser activity	What the observation supports
Public access and authentication	Landing-page navigation, login/register pages, quick-login cards, and manual email/password fields.	Pages and the documented login path were interactive in the demonstration environment; not a credential attack or authentication audit.
Citizen workflow	Five-step report, configured AI call, submission, tracking page, complaint list, and notifications.	One complete report path was observed; no representative usability or classifier evaluation was performed.
Officer workspace	Department queue, case controls, field map, analytics, and reports.	Role-specific screens rendered and supported the recorded interactions against seeded records.
Administrator workspace	City complaint list, map, analytics, intelligence reports, source registry, and user/configuration pages.	Administrative views could display the demonstration dataset; values are not municipal service statistics.
Route boundary	A citizen session was sent away from officer and administrator web routes.	Client-side route redirection worked for this case; direct API authorization and privileged self-registration were not tested.

The test fixtures contain 64 complaints in the administrator list and 11 in the officer queue. The analytics screens display values such as resolution rate, average resolution time, reopened cases, and satisfaction, but those values are generated from seeded records. They demonstrate chart binding and rendering; they are not evidence of actual response time, departmental efficiency, citizen satisfaction, or workload capacity. Treating generated fixtures as an evaluation sample would turn software test data into a false performance claim.

For the same reason, the displayed 98% model confidence is reported only as a UI observation. The evaluation did not compare this value with an expert label, reliability curve, or repeated query. Confidence can be high while a prediction is wrong, and a single selected example cannot estimate the error rate across

departments or languages. The system has no reported baseline comparison against a human dispatcher, keyword-only routing, or an existing municipal portal.

Table XIII presents the numerical evidence alongside its limits. The distinction between functional coverage and measured service quality is central to interpreting the result.

TABLE XIII RECORDED QUANTITATIVE OBSERVATIONS AND UNMEASURED PERFORMANCE DIMENSIONS.

Measure	Observation	Evidence scope
Frontend capability checks	18/18 marked working	One browser campaign; functional demonstration only.
Citizen submission	One complete five-step report	Submitted case and tracking page were opened in the same scenario.
AI endpoint response	About 2–3 seconds	One successful request; no repeated timing sample or percentile.
Displayed confidence	98%	The value shown for the example; not an accuracy or calibration result.
Seeded record counts	64 administrator-list records; 11 officer-queue records	Demonstration dataset size; not a database load test.
Department/ category quality	Not measured	No expert-labeled test set or confusion matrix was reported.
Concurrent throughput	Not measured	No controlled request-rate, multi-user, or stress test was performed.
Operational impact	Not measured	No field deployment compared assignment, resolution, SLA, or satisfaction outcomes.

This result is narrower than a production evaluation. The test used seeded data and documents one browser campaign. It shows that the demonstration build is connected end to end; it does not establish routing accuracy, calibrated confidence, accessibility conformance, concurrent-load performance, database scalability, security resistance, equal usability across groups, or an improvement in municipal outcomes.

XII. DISCUSSION

The shared case record is CivicPulse's answer to the handoff problem. It holds a tracking identifier, location, department, priority, status history, and outcome evidence. Each role sees a different view of that record: residents follow progress, officers manage their department's work, and administrators review activity across the city. In principle, this makes it easier to see where a case is waiting. The browser evaluation did not test whether staff resolved cases faster, so that practical benefit remains a question for a field study. The design is consistent with research that places institutional coordination and public participation alongside technical infrastructure [2], [3], [8].

The triage model can turn a free-text report into a structured suggestion, but it can also be wrong. The rules-based fallback keeps intake available when the provider cannot be reached; it is not an equally safe replacement, since keyword rules miss context and language variation. Before a public pilot, the interface should identify whether a suggestion came from the model or the fallback, record corrections, and provide a clear manual escalation path for high-risk cases.

The browser campaign supports a limited conclusion: the main pages worked in the demonstration environment, and one report was submitted and opened in the tracking view. It does not show that classifications were correct, processing became faster, or complaints were resolved more effectively. Table XIII separates these observations from measures that were not collected. Seeded records can make a dashboard look complete, but they cannot substitute for data from an operating service.

Table I shows that CivicPulse belongs to an existing family of reporting and grievance services. Its contribution is an integration and workflow design, not a new kind of grievance channel or a validated classifier. A useful next comparison would test it against a conventional form with manual routing, keyword-only routing, and an officer queue without AI suggestions. The study should measure staff workload and correction effort as well as model

scores.

Municipal adoption also requires translating software labels into local practice. The seven departments and ten wards in the demonstration are fixed enumerations, not verified city-wide taxonomies or complete administrative boundaries. A city must decide which authority owns cross-cutting issues, how transferred cases are handled, whether a citizen can see internal notes, what deadline pauses are permitted, how appeals work, and which cases must be excluded from a public dashboard. Without these decisions, a technically valid state transition can still be institutionally wrong.

CivicPulse is a non-emergency service-request channel. Its "critical" flag is a prioritization hint, not a promise of immediate dispatch. A public deployment should direct imminently threat to the city's established emergency service, state its operating hours, and preserve an offline way to report safety issues.

XIII. THREATS TO VALIDITY

A. Construct validity

The principal measured outcome is whether a frontend capability was marked as working. That construct captures rendering and interaction in the tested demonstration environment, but not reliability over time, task completion by representative residents, correct backend authorization, or accessibility. Similarly, a displayed model confidence is an interface field; without labels and repeated samples, it is not a measure of predictive confidence quality. Dashboard values generated from seeded complaints are useful for checking charts but not for constructing a service-performance construct.

B. Internal validity

The browser record describes one test campaign and one successful end-to-end report path. The report gives no randomized order, repeated trials, comparison group, or documented control of network and provider conditions. The observed two-to-three-second AI response therefore cannot be attributed solely to CivicPulse's own processing and should not be generalized to other periods or deployments. Likewise, no baseline workflow was measured, so the test cannot show that AI reduces operator time or that the application causes a change in resolution rate.

C. External validity

The test uses a Bengaluru-oriented demonstration configuration, English-language categories, a fixed list of wards, one browser environment, and synthetic records. It does not represent the full diversity of municipal organizations, resident languages, disability access needs, connectivity conditions, device types, or reporting channels. Priority rules and response times are local policy choices; the six-hour Critical deadline, for example, cannot be assumed appropriate for another municipality or department.

D. Data and implementation validity

The source dump documents a single-table JSONB repository, an external model integration, a rules fallback, and a particular snapshot of the frontend and backend. Dependencies, provider behavior, schemas, and test status can change. The PDF's implementation description should therefore be read as version-specific. Some service metrics are computed from status history and the current record state; they may not match a city's official KPI definition. No independent source-data verification or database load characterization was completed.

These threats do not invalidate the functional demonstration. They define its scope. The system can be useful as a prototype and still require a much stronger research design before effectiveness, safety, fairness, or operational readiness is claimed.

XIV. DEPLOYMENT CONSIDERATIONS

A staged rollout would reduce the risk of confusing successful software integration with a validated public service. Table XIV proposes a sequence in which each stage has an explicit purpose and an exit condition. The sequence is a recommendation, not work already completed by the prototype. No expansion should be based solely on aggregate volume or dashboard activity. The deployment owner should review whose reports are missing, whether support requests are concentrated among digitally connected residents, and whether officers have enough capacity to act on additional intake. Public feedback and an appeal path should remain available when the system's category, urgency, or resolution is disputed. These controls connect the technical design to the institutional capacity emphasized in smart-city governance research [1], [2], [5].

TABLE XIV PROPOSED STAGED PATH FROM DEMONSTRATOR TO MUNICIPAL PILOT.

Stage	Main work	Evidence required to proceed
Governance and hardening	Restrict privileged account creation; formalize department roles, status transitions, retention, emergency guidance, provider data handling, and incident response.	Approved policies, threat assessment, secure configuration checks, and independent API authorization tests.
Staff sandbox	Use seeded or consented pilot records with trained municipal staff; expose model suggestions and require human confirmation.	Task-completion observations, override reasons, accessibility review, error handling, and demonstrated auditability.
Shadow operation	Run on a limited incoming sample without allowing automated suggestions to determine service priority; compare suggestions with staff decisions.	Expert-labeled routing and priority results, critical-case recall, calibration, latency distribution, and acceptable privacy review.
Controlled pilot	Enable the workflow in selected departments or wards with a rollback plan and parallel support channel.	Predefined comparison against existing practice, safety monitoring, inclusion measures, operational outcomes, and resident feedback.
Scale decision	Expand only after technical and service outcomes are reviewed with stakeholders.	Sustained capacity, acceptable error and equity measures, funded support, and an accountable owner for continuing oversight.

XV. LIMITATIONS AND FUTURE WORK

The next stage should focus on evidence and reliability rather than adding more screens. The main tasks are:

- constructing an anonymized, expert-labeled complaint set to measure department-routing accuracy, priority recall, safety-critical false negatives, and confidence calibration;
- evaluating duplicate detection against text, time, and spa-tial baselines instead of relying only on same-department proximity;
- moving filtering and aggregation into indexed database queries and load-testing realistic complaint volumes;
- adding automated API, authorization, lifecycle-transition, and regression tests alongside the browser suite;
- conducting accessibility audits against WCAG 2.2 [27], multilingual usability studies, low-bandwidth/mobile testing, and targeted evaluation of voice reporting across Indian languages and accents [28];
- formalizing appeal, correction, and override procedures for automated classifications;
- integrating verified municipal geographic data and production notification channels; and
- measuring operational outcomes such as time to assignment, time to resolution, SLA compliance, duplicate reduction, and citizen satisfaction.

Geographic bias needs particular attention. Complaint counts reflect both physical conditions and the likelihood that residents will use the service. A bright hotspot on the map may show urgent need, strong reporting participation, or a mixture of the two. Studies of civic-reporting platforms have found socio-demographic participation gaps, so resource decisions should combine CivicPulse records with inspections, service history, and outreach in less-represented areas [19].

XVI. CONCLUSION

CivicPulse puts reporting, routing, assignment, resolution, and citizen verification into one case flow. Residents can track a report, officers can manage their queues and submit evidence, and administrators can review activity across departments. The model suggests intake fields; the rules-based fallback keeps the workflow available if the model service fails. Backend role checks and filtered responses provide a starting point for security, but they do not replace an independent security review. The browser tests show that the demonstration workflow operates end to end; they do not establish production readiness. Before a municipal pilot, the system needs evaluation

on representative complaints, independent security and accessibility reviews, realistic load tests, and calibrated model measurements. A limited pilot could then test whether the workflow improves coordination or service outcomes in practice.

REFERENCES

[1] Simonofski, E. Serral Asensio, J. De Smedt, and M. Snoeck, "Citizen participation in smart cities: Evaluation framework proposal," in *Proc. 19th IEEE Conf. Business Informatics (CBI)*, 2017, pp. 227–236, doi: 10.1109/CBI.2017.21.

[2] A. Meijer and M. P. R. Bolívar, "Governing the smart city: A review of the literature on smart urban governance," Int. Rev. Admin. Sci., vol. 82, no. 2, pp. 392–408, 2016.

[3] H. Chourabi et al., "Understanding smart cities: An integrative framework," in Proc. 45th Hawaii Int. Conf. System Sciences, 2012, pp. 2289–2297.

[4] T. Nam and T. A. Pardo, "Conceptualizing smart city with dimensions of technology, people, and institutions," in Proc. 12th Annu. Int. Digital Government Research Conf., 2011, pp. 282–291.

[5] N. V. D. Pradeep and G. Veeraju, "Strengthening grievance redressal systems as a critical component of good governance: The case study of CPGRAMS," Educ. Admin.: Theory Pract., vol. 30, no. 5, pp. 15613–15624, 2024, doi: 10.53555/kuey. v30i5.9989.

[6] R. Kitchin, "The real-time city? Big data and smart urbanism," GeoJournal, vol. 79, no. 1, pp. 1–14, 2014.

[7] S. E. Bibri and J. Krogstie, "Smart sustainable cities of the future: An extensive interdisciplinary literature review," Sustainable Cities Soc., vol. 31, pp. 183–212, 2017.

[8] J. R. Gil-Garcia, T. A. Pardo, and T. Nam, "What makes a city smart? Identifying core components and proposing an integrative and comprehensive conceptualization," Inf. Polity, vol. 20, no. 1, pp. 61–87, 2015.

[9] P. Neirotti, A. De Marco, A. C. Cagliano, G. Mangano, and F. Scorrano, "Current trends in smart city initiatives: Some stylised facts," Cities, vol. 38, pp. 25–36, 2014.

[10] React Team, "React documentation: Quick start." [Online]. Available: https://react.dev/learn. Accessed: Sep. 24, 2026.

[11] S. Ramírez, "FastAPI documentation." [Online]. Available: https://fastapi.tiangolo.com/. Accessed: Sep. 24, 2026.

[12] Supabase, "Managing JSON and unstructured data." [Online]. Available: https://supabase.com/docs/guides/database/json. Accessed: Sep. 24, 2026.

[13] mySociety, "How does FixMyStreet work?" [Online]. Available: https://fixmystreet.org/how-it-works/. Accessed: Oct. 29, 2026.

[14] City of New York, "About NYC311" and "NYC311" mobile application listing. [Online]. Available: https://portal.311.nyc.gov/about-nyc-311/ and https://apps.apple.com/us/app/nyc-311/id324897619. Accessed: Oct. 29, 2026.

[15] Department of Administrative Reforms and Public Grievances, Government of India, "CPGRAMS: About Us." [Online]. Available: https://pgportal.gov.in/Home/AboutUs. Accessed: Oct. 29, 2026.

[16] "CivicPulse project source information dump," unpublished software documentation and functional test record, 2026.

[17] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," IEEE Internet Things J., vol. 1, no. 1, pp. 22–32, 2014.

[18] N. Staletić, A. Labus, Z. Bogdanović, M. Despotović-Zrakić, and B. Radenković, "Citizens' readiness to crowdsource smart city services: A developing country perspective," Cities, vol. 107, Art. no. 102883, 2020, doi: 10.1016/j.cities.2020.102883.

[19] B. Pak, A. Chua, and A. Vande Moere, "FixMyStreet Brussels: Socio-demographic inequality in crowdsourced civic participation," arXiv:1708.02274, 2017.

[20] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining, 1996, pp. 226–231.

[21] P. K. Atrey, M. A. Hossain, A. El Saddik, and M. S. Kankanhalli, "Multimodal fusion for multimedia analysis: A survey," Multimedia Syst., vol. 16, no. 6, pp. 345–379, 2010.

[22] M. De-Arteaga, R. Fogliato, and A. Chouldechova, "A case for humans-in-the-loop: Decisions in the presence of erroneous

algorithmic scores," in Proc. 2020 CHI Conf. Human Factors in Computing Systems, 2020, pp. 1–12, doi: 10.1145/3313831.3376638.

[23] E. Tabassi, Artificial Intelligence Risk Management Framework (AI RMF 1.0), NIST AI 100-1, Jan. 2023, doi: 10.6028/NIST.AI.100-1.

[24] M. S. Turan, E. Barker, W. Burr, and L. Chen, Recommendation for Password-Based Key Derivation: Part 1: Storage Applications, NIST Special Publication 800-132, Dec. 2010, doi: 10.6028/NIST.SP.800-132.

[25] M. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," RFC 7519, May 2015, doi: 10.17487/RFC7519.

[26] D. F. Ferraiolo, J. A. Cugini, and D. R. Kuhn, "Role-based access control (RBAC): Features and motivations," in Proc. 11th Annu. Computer Security Applications Conf., 1995, pp. 241–248.

[27] World Wide Web Consortium, Web Content Accessibility Guidelines (WCAG) 2.2, W3C Recommendation, Dec. 12, 2024. [Online]. Available: https://www.w3.org/TR/WCAG22/

[28] A. Singh, V. Kadyan, M. Kumar, and N. Bassan, "ASRoIL: A comprehensive survey for automatic speech recognition of Indian languages," Artif. Intell. Rev., vol. 53, pp. 3673–3704, 2020.