CROWDSOURCED CIVIC ISSUE REPORTING AND RESOLUTION SYSTEM

BUSINESS REQUIREMENTS DOCUMENT

Contents

1. EXECUTIVE SUMMARY SNAPSHOT

This Business Requirements Document (BRD) outlines the need for a mobile-first civic issue reporting and management platform. The system's goal is to provide citizens with a simple way to report everyday municipal problems (e.g., potholes, broken streetlights, overflowing bins) and to enable municipal staff to efficiently triage, route, and resolve those reports.

Currently, existing apps and processes are fragmented, lack automated routing, and force staff to manually sift through unfiltered issue feeds. Poor geolocation, weak analytics, and limited citizen feedback loops contribute to slow resolution times and low trust in government responsiveness.

The proposed solution is a cross-platform mobile application for citizens, combined with a role-based administrative portal for municipal staff and a scalable backend with automated routing, analytics, and APIs for future integration. This will enable faster acknowledgments, transparent communication, and better operational insights for leadership.

2. PROJECT DESCRIPTION

2.1. Project Overview

This project will deliver a mobile-first civic issue reporting platform and a web-based administrative portal that together create a single, auditable workflow for citizens to report everyday municipal problems (e.g., potholes, broken streetlights, overflowing bins) and for municipal staff to triage, assign, resolve, and report on those problems. The solution consists of a cross-platform mobile app (or PWA) for citizens, a role-based web portal for municipal teams, a scalable backend for media and real-time updates, and analytics to inform operational decisions.

2.2. Project Purpose

Primary purpose Empower citizens to report issues (photo, location, text/voice) easily and get reliable feedback on progress.

Operational purpose Centralize and standardize incoming reports so municipal staff can quickly identify, prioritize, assign and resolve tasks instead of manually triaging fragmented channels.

Technical purpose Deliver a resilient, scalable backend and real-time interfaces (mobile + web) that handle multimedia, concurrent users, automated routing, and near real-time status updates.

Governance & accountability purpose Create auditable workflows, SLA enforcement and escalation rules so departments are accountable and citizens can track resolution status.

Data & planning purpose Produce analytics and operational KPIs (report volumes, response times, SLA adherence, hotspots) to inform resource allocation and preventative maintenance.

3. PROJECT SCOPE

3.1. High-level tasks

Discovery & requirements Stakeholder interviews, finalize BRD, define KPIs, confirm wards/zones for pilot, data/privacy rules.

Design (UI/UX & architecture) Mobile flows, admin UX (map + list + Kanban views), API contract, data model, infra design, security/privacy plan.

Implementation - Core MVP

- Citizen mobile app (PWA and/or lightweight native wrappers)
- Admin web portal (role-based, map/list/filters)
- Backend services (API, media store, routing engine, notifications)
- Real-time updates (WebSocket/SSE)
- Analytics dashboard (hotspots, SLA, exports)

Testing & hardening Functional tests, integration, user acceptance testing (UAT), load test for image uploads.

Pilot deployment & training Deploy to pilot ward, train staff, field workers, and support teams; run pilot.

Pilot evaluation & iteration Collect metrics, user feedback, bugfixes, and plan phased city rollout.

Handover & documentation Runbooks, admin guides, API docs, data retention & privacy policies.

3.2. Key Deliverables

1. Project initiation documents & finalized BRD (with KPIs and acceptance criteria)
2. UX prototypes (mobile & admin)
3. Citizen mobile app (PWA + build artifacts) supporting: photo, location pin + manual adjust, text/voice fields, notifications.
4. Admin portal: role-based access, live map (cluster + heatmap), list/kanban view, filters, assignment.
5. Routing engine & rule-config UI (department mapping by location/category/SLA)
6. Backend APIs, media storage, thumbnailing/compression, EXIF handling.
7. Real-time delivery mechanism and notification integrations.
8. Analytics dashboard (trends, hotspots, SLA reports) + export.
9. Field-worker mobile task view (simple checklist, complete photo).
10. Security & privacy controls (RBAC, admin MFA, audit logs, data retention workflows).

3.3. In Scope

1. Design, build and deploy an MVP for one pilot ward/zone that includes the above deliverables.
2. Citizen-facing mobile experience (PWA) for cross-device access.
3. Staff/admin web portal with map visualization and filtered task lists.
4. Routing engine for automated mapping of reports to departments based on location and category.
5. AI/ML based routing engine.
6. Media upload pipeline with compression, thumbnailing, and CDN delivery.

7. Duplicate/nearby-report detection (basic heuristics).
8. Real-time client updates and notifications (push/email/SMS integration for pilot-level volume).
9. API endpoints and webhook support for simple integrations (e.g., export to CSV)

4. ORGANIZATIONAL DRIVERS

4.1. Improve municipal responsiveness and reduce citizen frustration

Current channels are fragmented and slow, leaving many issues unacknowledged or unresolved. A centralized, mobile-first reporting platform will shorten acknowledgement and resolution times and restore citizen trust by providing transparent status updates.

4.2. Reduce manual triage overhead and operational costs

Staff today must scan undifferentiated feeds, manually classify and route reports, and work from spreadsheets. Automating routing, filtering, and assignment will free staff to do field work, reduce human error, and lower administrative costs.

4.3. Enable data-driven planning and better resource allocation

Consolidated, geolocated reports and analytics (hotspots, trends, SLA performance) allow leaders to prioritize recurring problems, plan preventative maintenance, and allocate budget where it has highest impact — improving long-term asset management.

4.4. Increase efficiency of field operations

Filtered task lists for field crews, work-order evidence (photos, checklists), and better geolocation reduce wasted travel and rework, improving crew productivity and reducing operating expense.

4.5. Ensure accountability, traceability and SLA enforcement

The system provides role-based access, audit trails, and escalation rules so departments can be measured and held accountable for response times and outcomes.

4.6. Improve quality and reliability of reports

Mobile-optimized UI, location guidance, and media handling (multi-photo, video, voice) yield higher-quality reports and better decision-making for staff.

4.7. Mitigate risk and meet regulatory/compliance needs

Consistent data retention policies, consent capture, and audit logs simplify public records management and help meet privacy and transparency requirements.

4.8. Future-proof municipal services & integrations

An API-first, scalable architecture enables integration with other municipal systems (ERP, asset management, emergency response) and supports future capabilities (automated image-based classification, vendor integrations).

Outcome

- Faster mean time to acknowledge and resolve issues.
- Reduced time staff spend on manual triage.
- Clear, auditable records for public accountability.
- Better capital planning through aggregated, reliable data.
- Higher citizen satisfaction and increased civic engagement.

Measurable KPIs

- Mean Time to Acknowledge (target: ≤ X hours)
- Mean Time to Resolve (target: ≤ Y days)

- % of incidents resolved within SLA (target: ≥ Z%)
- Reduction in staff triage time (target: % reduction)
- Citizen satisfaction score (post-resolution, target: ≥ N/10)
- Upload success rate and system availability (e.g., 99.9% uptime)

5. CURRENT PROCESS

Several third-party and municipal apps exist today for citizens to report civic issues (potholes, broken streetlights, overflowing bins, etc.). In practice the workflow is:

1. A citizen notices a problem and submits a report using an app, phone, email, social channel, or in-person. Submissions typically include a photo and short description, though completeness varies.
2. Reports are collected in a central backend or forwarded to municipal systems. Many municipalities also provide staff-side apps or dashboards.
3. Staff-side apps/dashboards receive all incoming reports (a full feed), not filtered by department or location. Each departmental user must browse or search the shared list to find the reports relevant to their area or remit.
4. Staff manually review, classify, and identify the responsible department or crew for each report - often relying on local knowledge or manual mapping of addresses/wards.
5. Assignments are done manually (spreadsheets, emails, or direct messages), and status updates (acknowledged, in-progress, resolved) are recorded inconsistently across systems - if recorded at all.
6. Citizens rarely receive timely, consistent feedback; analytics and reporting are produced ad hoc by exporting data to spreadsheets and manually aggregating.

6. CHALLENGES IN THE CURRENT PROCESS

6.1. Staff apps receive unfiltered, undifferentiated feeds

Problem Staff-side apps/dashboards present all reports to everyone rather than automatically routing only the relevant items to the appropriate department or team.

Effect Staff spend time scanning the full feed to find relevant tasks; this increases triage overhead, delays assignment, and causes higher risk of missed or late responses.

6.2. No automated routing by location, department or rules

Problem Reports are not auto-mapped to responsible departments based on location, category, or configurable rules.

Effect Manual classification increases workload and introduces routing errors and delays.

6.3. Poor geolocation accuracy and handling

Problem GPS pins are inaccurate with no reliable pin-adjust or GPS-accuracy metadata.

Effect Mislocated reports require additional staff effort and lead field crews to wasted travel and inefficiencies.

6.4. Weak analytics and reporting

Problem Little to no automated dashboards for trends, SLA performance, or hotspot detection.

Effect Leadership lacks visibility for prioritization and strategic planning.

6.5. Not mobile-friendly or intuitive UI

Problem Citizen and sometimes staff apps are poorly optimized for mobile or have confusing flows.

Effect Fewer quality submissions and higher abandonment; staff usability issues reduce productivity.

6.6. No visual, interactive map in admin panel

Problem Admin/staff interfaces lack a live map with clustering or heatmaps.

Effect Geographic prioritization is slow and error-prone because staff must infer location from list views.

6.7. No configurable priority-area assignment or algorithmic prioritization

Problem Priority cannot be applied automatically using configurable criteria (volume, severity, vulnerable locations).

Effect Inconsistent prioritization and missed high-impact issues.

6.8. Fragmented feedback loop and poor citizen communication

Problem Citizens rarely receive confirmations or consistent status updates.

Effect Reduced trust and increased follow-ups via other channels.

6.9. Duplicate detection and consolidation missing

Problem No automated merging of proximate/duplicate reports.

Effect Inflated workload, skewed analytics, duplicate assignments.

6.10. Manual, error-prone operational practices

Problem Heavy reliance on spreadsheets, manual reassignment, and ad-hoc emailing.

Effect Poor auditability, SLA enforcement gaps, and higher operational cost.

6.11. Limited media handling and upload reliability

Problem Uploads can be slow or unreliable and there’s no robust CDN/thumbnailing.

Effect Weak evidence trail for field crews and poor user experience under load.

7. PROPOSED PROCESS

7.1. Citizen submission (intake)

Actors Citizen (registered), Mobile app / PWA

Trigger Citizen captures an issue and taps Submit.

Steps

1. Citizen attaches one or more photos/videos (or voice note), types/voices a short description, and selects a category.
2. App captures geolocation (lat/lon + GPS accuracy) and allows manual pin-adjust and address/landmark entry.
3. Client performs basic validations (required fields, file size), compresses media if needed, and queues uploads when offline.
4. On successful upload the system returns a report ID and shows a friendly confirmation with expected next steps.

System responsibilities

1. Persist report and metadata (reporter if provided, timestamp, location, media refs, category, device GPS accuracy).
2. Store media in CDN-backed storage, generate thumbnails, and apply EXIF/privacy rules.
3. Create an audit record for the submission event.

Notifications

Immediate in-app confirmation; optional SMS/email copy if configured.

7.2. Ingestion & validation

Actors Backend ingestion pipeline, optional moderation team

Trigger New submission stored.

Steps

1. Backend validates payload and media integrity, runs lightweight heuristics for duplicates/spam, and flags obvious moderation cases.
2. Resolve ward/zone/polygon from provided coordinates and add geo-context (nearest street, ward, asset).
3. Compute initial metadata (nearby report count, basic priority hint from category/time/nearby critical assets) for routing; persist validation logs.

System responsibilities

Maintain spatial lookup tables and basic heuristics. Emit events/messages to routing and staff queues.

Notifications

If flagged for moderation, notify moderators; otherwise proceed silently to routing.

7.3. Routing & filtering

Actors Routing engine, Admin config UI

Trigger Validated report ready for routing.

Steps

1. Routing engine maps the report to the most likely department(s) using ward polygons, category mappings, and admin-configured rules.
2. If multiple departments apply, create linked tasks or send for coordination.
3. Queue the task(s) in the appropriate department feed(s) and tag with metadata (priority hint, nearby duplicates, confidence score).
4. Low-confidence routings surface in a small triage queue for a quick human check.

System responsibilities

1. Provide admin UI for mapping rules and confidence thresholds.
2. Allow webhooks/APIs to push tasks to existing staff systems.

Notifications

Near-real-time notification to the mapped department/supervisor inbox or app.

7.4. Triage & assignment (staff)

Actors Supervisors/dispatchers, staff app/admin portal

Trigger New task appears in department feed.

Steps

1. Staff view tasks in filtered list, map or Kanban view (map clustering/heatmap aids geography-first triage).
2. Supervisor confirms department mapping, merges or links duplicates, adjusts the priority hint if needed, and assigns to a field worker or contractor (single or batch).
3. Assignments can include instructions, checklists, and optional ETA notes.

System responsibilities

1. Support map/list/kanban views, bulk actions, and an audit trail of triage decisions.
2. Allow attachment of scheduling and resource metadata to work orders.

Notifications

Assigned field worker receives a notification; citizen receives an assignment acknowledgment.

7.5. Field execution & verification

Actors Field worker (mobile), supervisor, citizen

Trigger Field worker accepts/claims the task.

Steps

1. Field worker opens task on mobile: route/navigation, checklist, and safety notes.
2. Worker updates status to On-site, captures before/after photos, logs actions taken, and marks Completed with resolution evidence.
3. Supervisor reviews evidence and marks final acceptance if required; otherwise the system can auto-close based on configured acceptance rules.

System responsibilities

1. Store time-stamped events (on-route, on-site, completed) for reporting.
2. Support offline capture and later sync.

Notifications

Citizen receives a resolution update with before/after photos and optional feedback prompt.

7.6. Closure, feedback & re-open

Actors Citizen, supervisor, system

Trigger Task marked completed/resolved.

Steps

1. Citizen receives notification and an opportunity to acknowledge completion, provide a short rating, or re-open the issue.
2. If re-opened or disputed, a follow-up task is created and routed back to the appropriate team for verification.
3. After closure the system enforces retention policy for media and PII according to configuration.

System responsibilities

1. Persist feedback and link it to the report; provide simple re-open workflows.
2. Log final closure and data-retention start date.

Notifications

Closure confirmation to citizen; follow-up notifications if re-opened.

7.7. Escalation & exception handling (lightweight)

Actors Routing engine, supervisors, escalation roles

Trigger Manual flag, repeated re-open, or time-based alerting (informational).

Steps

1. For recurring or repeatedly reopened issues, system flags the problem for supervisor review and possible escalation (coordination across departments or special inspection).
2. Provide a simple escalation path UI where supervisors can raise issues to managers or request specialized teams.

System responsibilities

Maintain flags and history of reopened/repeated issues; provide a clear trail for investigations and follow-ups.

7.8. Analytics, reporting & planning

Actors Leadership, analysts, supervisors

Trigger Periodic review or ad-hoc queries.

Steps

1. Dashboards show volumes, geographic hotspots (heatmap), repeat issues, staff activity (tasks completed, resolution evidence), and citizen feedback trends.
2. Allow exports (CSV/GeoJSON) and ad-hoc drill-down from KPI to case-level detail for planning and budgeting.
3. Provide scheduled summary reports to stakeholders and alerts for unusual spikes.

System responsibilities

Near-real-time aggregation for dashboards, secure exports, and role-based access to analytics.

7.9. Administration, configuration & integrations

Actors System admins, GIS/IT, vendors

Trigger System setup and ongoing changes.

Steps

1. Admins manage categories, ward polygons, department mappings, notification channels, media retention rules, and user roles.
2. Integrate with SMS gateway, auth providers, and third-party systems via APIs/webhooks as required.
3. Provide runbooks, audit logs, and change-tracking for admin actions.

System responsibilities

Secure admin UI, RBAC, versioned configs, and integration endpoints.

7.10. Exception handling & resilience

Examples

1. Media upload failures: background retry + fallback instructions (SMS/upload later).
2. Incorrect geolocation: allow manual pin-adjust and staff geocoding fixes.
3. Moderation need: hold and notify moderators with reason codes.
4. Duplicate false positives: allow manual split/merge.

Resilience features

Offline capture + queued sync, idempotent APIs, queue-based ingestion, CDN-backed media, and health monitoring with alerts.

Data / Status model

Submitted → Validated → Routed → Assigned → On-site → Completed → Closed

Reopened (if citizen/staff requests follow-up) Each transition stores actor, timestamp, notes and media.

8. REQUIREMENTS PRIORITY AND CATEGORIES

8.1. Priority

Value	Rating	Description
1	Critical	The requirement is critical to the project's success. Without fulfilling this requirement, the project is not possible.
2	High	The requirement is high priority re the project's success, but the project could still be implemented in a minimum viable product (MVP) scenario.
3	Medium	The requirement is important to the project's success, as it provides value, but the project could still be implemented in an MVP scenario.
4	Low	The requirement is low priority (i.e., it would be nice to have), but the project's success is not dependent upon it.
5	Future	The requirement is outside of the project's scope and is included as a possible component of a prospective release and/or feature.

8.2. Requirement Categories

ID	Requirement	Priority
RC 1	Functional Requirements	1
RC 2	Non-Functional Requirements	2
RC 3	Data Requirements	1
RC 4	Process / Workflow Requirements	1
RC 5	Integration Requirements	3
RC 6	Compliance & Regulatory Requirements	2
RC 7	Operational Requirements	3
RC 8	Out of Scope Requirements	5

9. FUNCTIONAL REQUIREMENTS

9.1. For Citizens / User Application

1. The system shall provide a cross-platform mobile application (Android, iOS, etc.).
2. The mobile app shall allow users to submit civic issue reports in real time.

   Each report shall include

   - At least one photo.
   - Automatic location tagging (GPS-based).
   - A short text description or voice explanation.
3. The app shall allow users to track the progress of their submitted issues.
4. The app shall send notifications to users at each stage of the issue lifecycle:
   - Confirmation
   - Acknowledgment
   - Resolution
5. The app shall provide a seamless and intuitive user experience (simple, easy-to-use UI).

9.2. For Administrators / Municipal Staff

1. The system shall provide a web-based administrative portal accessible across devices.
2. The portal shall enable staff to view, filter, and categorize incoming reports.
3. The portal shall allow staff to assign tasks to relevant departments.
4. The portal shall allow staff to update issue statuses (e.g., acknowledged, in-progress, resolved).
5. The portal shall enable staff to communicate progress to citizens.
6. The portal shall provide filtering options for issues by:
   - Category
   - Location
   - Priority
7. The portal shall include a live interactive map of the city's reported issues.
8. The portal shall highlight priority areas based on:
   - Volume of submissions
   - Urgency inferred from user input
   - Configurable admin-defined criteria
9. The portal shall include analytics and reporting features, such as:
   - Reporting trends
   - Departmental response times
   - Overall system effectiveness

9.3. For the Backend / System Core

1. The system shall include an automated routing engine to allocate tasks to the correct department based on report metadata (e.g., issue type, location).
2. The system shall support real-time updates across mobile and web clients.
3. The system shall provide APIs for future integrations or extensions.

10. NON-FUNCTIONAL REQUIREMENTS

10.1. Performance & Reliability

1. The system shall be able to handle spikes in reporting (high load of concurrent submissions).
2. The system shall provide quick image uploads for reports, even under high traffic.
3. The system shall provide near real-time updates on both mobile and desktop clients.
4. The system shall ensure responsive performance across devices (fast loading times and smooth interaction).

10.2. Scalability & Resilience

1. The system shall be scalable to support growing numbers of users, reports, and municipalities.
2. The system shall be resilient, maintaining functionality during failures or unexpected surges.

10.3. Usability

1. The mobile application shall provide a seamless, intuitive, and effortless user experience.
2. The administrative portal shall have a powerful yet user-friendly dashboard for staff.

10.4. Extensibility

The system shall provide APIs for future integration or extension into other municipal or third-party systems.

11. ASSUMPTIONS

11.1. Citizen Participation

- Citizens will have access to smartphones or devices capable of running the mobile app (or PWA).
- Citizens will be willing to use the platform to report issues rather than relying only on phone calls or in-person complaints.

11.2. Connectivity & Devices

- Internet connectivity (mobile data/Wi-Fi) will be generally available to citizens and staff for real-time uploads, though the app will support offline capture with later sync.
- Municipal staff and field workers will have access to compatible devices (desktop browsers for portal, smartphones/tablets for mobile app).

11.3. Geospatial Data

- The municipality will provide accurate ward boundaries, zoning maps, and department jurisdiction data to enable automated routing.
- GPS accuracy in most reporting areas will be sufficient for location tagging; where not, manual adjustment will compensate.

11.4. Municipal Staff Operations

- Municipal departments will adopt the admin portal for triage, assignment, and resolution tracking.
- Staff will use the system consistently to update statuses, so citizens receive real-time progress visibility.

11.5. System Integration

- SMS/email gateways and authentication providers will be available for integration.
- Deeper integration with ERP or asset management systems is out of scope for the pilot but can be added later via APIs.

11.6. Legal & Compliance

- The municipality will handle necessary legal approvals regarding data privacy, citizen consent, and record-keeping.
- Citizens consent to share location data and media when submitting reports.

11.7. Operations & Support

- The municipality will provide training and onboarding for staff and field workers.
- A support/helpdesk model will be in place during the pilot to handle technical and process issues.

11.8. Deployment & Infrastructure

- The project will use cloud-based infrastructure with CDN for media, ensuring performance and availability.

12. GLOSSARY

Citizen User: An individual who submits civic issues using the mobile application.

Report: A submission made by a citizen that includes issue details such as photo(s), location, description, and metadata.

Administrative Portal (Admin Portal): A web-based dashboard used by municipal staff to view, filter, assign, and track issues.

Routing Engine: The backend component responsible for automatically mapping citizen reports to the appropriate department based on metadata (location, category, etc.).

Priority Area: A zone highlighted for attention based on criteria such as report volume, severity, or urgency.

Workflow: The sequence of steps from citizen submission → validation → routing → assignment → resolution → closure.

Geolocation: The GPS-based location tagging of an issue report, with optional manual adjustment.

Duplicate Detection: The process of identifying and consolidating multiple reports referring to the same or nearby issue.

SLA (Service Level Agreement): A target metric (e.g., response time, resolution time) used to measure departmental performance.

PWA (Progressive Web App): A lightweight, cross-platform web-based mobile application that works across devices.

API (Application Programming Interface): A set of endpoints that allow integration with external systems, such as municipal ERP or SMS/email services.