Grievance-Resolution System for a Tier-2/3 City

Project Proposal for UCS503P
Submitted to: Dr. Raghav B. Venkataramaiyer
Thapar Institute of Engineering and Technology

Vidya Vaidyanathan, CSED* Bhanurekha Ganesan, CSED†

October 4, 2026

1 Higher-order goal

. . . secondary framing

This project aims to improve sustainable urban governance by reducing the time and effort required to resolve citizen issues. While the topic is broader than any single municipality, the immediate engineering objective is to translate *efficient public service delivery* into a measurable, deployable software solution.

2 Time-to-value

. . . primary heuristic

- We will deliver meaningful increments quickly by prototyping the core user workflow and validating it with users within a short iteration window.
- We will prioritize fast feedback over perfection by using weekly iterations (and targeting CI-backed automation early) to reduce release friction.
- Success is defined by what can be delivered and measured in the first iteration, then improved based on user feedback.

3 Problem Statement

Citizens in a tier-2/3 Indian city face delays and uncertainty when reporting local issues (e.g., streetlight outages, water leakage, garbage accumulation, road potholes). Common pain points include:

- **Fragmentation:** complaints are often made via multiple channels (calls, social media, paper forms), causing lost context.
- **Low transparency:** citizens cannot easily see the status of their complaint (e.g., "received", "assigned", "in progress", "resolved").
- **Slow coordination:** departments/field teams lack a unified, trackable workflow with clear ownership.

*Roll No: 1024030xxx, Email: vvaidhanathan_be24thapar.edu
†Roll No: 1024030xxx, Email: bganesan_be24thapar.edu

- **Poor data quality:** duplicate or incomplete reports force rework and extend resolution time.

Why this matters now (contextual relevance): In an urban setting with limited staff bandwidth, even small process improvements can noticeably reduce resolution delays and repeated submissions.

4 Proposed Solution *... Software Proposition*

4.1 Overview

Build a **web-based** "Grievance-to-Resolution" system with the following components:

- **Citizen submission portal:** users submit an issue with photo (optional), category, and locality/ward.
- **Automated triage:** basic duplicate detection and completeness checks.
- **Department workflow:** assign issues to relevant department/team and update statuses.
- **Status transparency:** citizens track status and receive notifications when resolved.
- **Admin/ops dashboard:** performance summaries (resolution time, backlog).

4.2 Core workflow

1. Citizen creates a complaint and receives a complaint ID immediately.
2. System validates completeness and suggests possible duplicates (lightweight similarity on text/category and approximate location).
3. Department staff update status (Received → Assigned → In Progress → Resolved).
4. Citizen views status updates and resolution notes.

4.3 Operational constraints for an educational, tier-2/3 setting

- Keep the solution usable on low-to-mid range devices via responsive web design.
- Avoid dependence on advanced research algorithms; focus on workflow, automation, and system correctness.
- Design for deployability using common web hosting and managed databases.

5 Solution Approach *... Engineering focus*

This is an engineering course project, so we focus on scalable administrative and workflow aspects rather than novel algorithm research.

5.1 Duplicate/quality assistance *... non-research*

Duplicate detection will be heuristic-based:

- Normalize text (category + short description keywords).
- Use a simple similarity measure (e.g., token overlap / TF-IDF cosine with threshold) without claiming state-of-the-art.
- Show top candidate duplicates to staff for decision; never fully auto-delete or auto-reject.

5.2 Web architecture *... web-based solution*

A typical 3-tier web architecture:

- **Frontend:** responsive UI for citizens and staff (status pages, submission form, dashboard).
- **Backend API:** authentication, complaint submission, status transitions, assignment logic.
- **Database:** store users, complaints, status history, attachments metadata.

5.3 CI/CD and fast delivery enabler

CI/CD will be used to:

- Automatically build and run tests on every change.
- Produce repeatable deployment artifacts to staging.
- Enable safe and frequent releases of incremental features.

6 Evaluation Criterion *... measurable and attributable*

We define success using measurable metrics tied to the core workflow.

6.1 Primary evaluation metric

Time-to-acknowledgement (TTA): time between complaint submission and "Received/Assigned" status creation in the system.

- Target: median TTA $\leq$ 2 hours during pilot window.
- Attribution: measured from database timestamps (submission vs. first status update by staff).

6.2 Secondary evaluation metrics

- **Resolution time reduction:** reduce time-to-resolved compared to baseline process (even if baseline is collected via survey).
- **Duplicate rate:** percentage of complaints flagged as duplicates and resolved without rework.
- **User clarity:** citizen-reported clarity of status using a simple 1–5 feedback question.
- **Reliability:** availability of staging/production for login and submission during business hours (e.g., $\geq$ 99% uptime in pilot).

6.3 Pilot validation plan *... high-level*

- Recruit 10–20 pilot users (citizens) and 3–5 staff roles (department simulation if real staff unavailable).
- Compare metrics before/after within the iteration window by logging baseline estimates via short interviews or existing process observations.

7 Scalability *... with foresight*

The proposition should scale in both theoretical and practical senses.

1. **Scalable (theoretically):** The system should support growth in number of complaints and concurrent dashboard usage by:
   - decoupling components (frontend/backend),
   - enabling pagination and indexing for common queries,
   - using stateless API design.
2. **Operationally deployable (practically):** The system should run on common web hosting:
   - managed database,
   - containerized or platform-hosted deployment,
   - CI/CD pipeline for staging/production.

8 Engine availability heuristic

A mature set of "engines" (tooling/services) is available to implement this as a web-based project:

- Web framework for REST APIs and templated/reactive pages.
- Managed relational or document database.
- Authentication approach (token-based or session-based).
- Object storage for optional attachments (or local storage for pilot).
- Test framework for unit/integration tests.
- CI runner and staging deployment target.

We will use these mature components to focus on workflow design, correctness, and measurement rather than inventing new infrastructure.

9 Project scope and deliverables *... iteration-friendly*

9.1 Initial deliverable *... first iteration*

- Citizen submission form + complaint ID generation
- Staff workflow: update status and assignment
- Citizen status page: view timeline/history
- Basic logging and timestamps for TTA measurement
- CI: build + backend unit tests + linting
- CD to staging with a single command or pipeline job

9.2 Subsequent deliverables

- Lightweight duplicate suggestions with staff review
- Notification layer (email/SMS optional in later iteration; can start with in-app notifications)
- Admin dashboard for metrics and backlog
- Improved search/filtering and indexed queries for scale readiness

10 Risks and mitigations

- **Data privacy / consent:** minimize collected data; keep attachments optional; store only what is needed.
- **Staff adoption:** provide an easy UI for status updates; keep workflow steps minimal; include shortcuts.
- **Evaluation measurement ambiguity:** instrument timestamps and define clear status transitions before development begins.
- **Operational issues in pilot:** use staging early; ensure CI/CD produces repeatable deployments.

11 Summary

This project proposes a deployable web platform to reduce grievance handling delays in a tier-2/3 city context. It meets the course criteria by emphasizing time-to-value through rapid iterations, implementing CI/CD to support frequent safe releases, and using measurable evaluation criteria (especially time-to-acknowledgement). It remains focused on engineering workflow and scalability readiness rather than research-driven novelty.