

Delhi Water Watch: A Role-Based, Event-Driven Civic Intelligence Architecture for Urban Waterlogging Management

Dinesh Bhardwaj

Assistant Professor, Department of Computer Science, BPIT, India

dineshkumar@bpitindia.com

Ritesh Chauhan

Student, Department of Computer Science, BPIT, India

riteshchauhan43792@gmail.com

Aditya Sharma

Student, Department of Computer Science, BPIT, India

aditya2562sharma@gmail.com

Harsh Kumar

Student, Department of Computer Science, BPIT, India

Harshkumar728@gmail.com

ABSTRACT

Urban waterlogging management in high-density cities remains constrained by delayed complaint intake, fragmented field coordination, and weak end-to-end traceability. This paper presents Delhi Water Watch, a role-based, event-driven civic platform designed to support complete lifecycle management of waterlogging incidents—from citizen reporting to administrative closure. The architecture introduces four bounded operational roles (citizen, ward administrator, field worker, and super administrator), each with isolated permissions, contextual views, and workflow responsibilities. The system integrates geotagged issue reporting, real-time status propagation, ward-specific triage, field assignment with completion proof, map-centric situational monitoring, and notification orchestration. A predictive intelligence layer further assists governance by generating AI-based hotspot alerts using weather signals and historical report trends, with human validation before activation. Implemented using React + TypeScript, Firebase Authentication, Firestore real-time listeners, and Leaflet-based geospatial visualization, the platform demonstrates improved operational clarity and accountability over linear grievance channels. Qualitative pilot observations show stronger role coordination, higher status transparency for citizens, and better city-level preparedness visibility. The proposed framework contributes a practical architecture for climate-resilient, digitally governed municipal response systems.

Keywords: *urban waterlogging, civic informatics, event-driven architecture, role-based access control, AI-assisted prediction, real-time dashboards, municipal governance, smart city resilience*

1. Introduction

Seasonal flooding and waterlogging are persistent urban challenges in Delhi. During monsoon periods, residents experience traffic disruption, safety risks, and delays in civic services. Traditional complaint systems are frequently linear and non-transparent, where citizens lack real-time status visibility and authorities face coordination gaps between intake, assignment, and closure.

Delhi Water Watch is proposed as a unified digital governance framework to address these bottlenecks. The platform is built on three core principles:

- Role-scoped operations — each actor operates within a bounded, permission-controlled domain.

- Event-driven state transitions — every status change is timestamped and triggers downstream notifications.
- AI-assisted anticipatory planning — predictive hotspot alerts are generated before incidents escalate.

Instead of treating complaint handling as disconnected departmental steps, the system models it as a continuously auditable lifecycle. This enables faster actionability at ward level and improved strategic oversight at city level. The remainder of this paper is structured as follows: Section 2 reviews related work, Section 3 formalizes the problem, Section 4 describes system architecture, Section 5 details technical implementation, Section 6 presents evaluation, and Section 7 outlines future directions.

2. Related Work

2.1 Long-Context Reliability and AI Systems

LLM-based systems often face context dilution and relevance degradation in long interactions. Prior work on attention mechanisms and context extension improves sequence processing, yet practical reliability also requires architecture-level filtering and workflow constraints [1]. Delhi Water Watch addresses this through role-bounded context and state-aware decision flow, ensuring each actor receives only operationally relevant data.

2.2 Multi-Agent and Specialized Orchestration

Research in multi-agent LLM orchestration demonstrates superior performance for composite tasks versus single generalist agents [2]. The current system adopts this philosophy in a pragmatic form via modular AI functions (prediction generation, review gating, alert conversion), with future support for deeper agent specialization tailored to municipal workflows.

2.3 Structured Memory and Knowledge-Aware Retrieval

Knowledge graph and structured memory approaches improve consistency in decision-support systems [4]. In this platform, operational entities (reports, wards, hotspots, alerts, assignments, resolution events) naturally form a governance graph that supports traceability and policy analytics, enabling forensic reconstruction of any incident timeline.

2.4 Event Sourcing and Auditability

Event sourcing is a well-established paradigm in enterprise systems for audit-grade state management [3]. Delhi Water Watch applies event-like state progression and timestamped transitions to improve governance accountability and deterministic reconstruction of the complaint lifecycle, aligning civic operations with enterprise reliability standards.

3. Problem Formulation

Let a conventional municipal complaint stream be represented as:

$$C = \{ m_1, m_2, \dots, m_n \}$$

where each m_i denotes a complaint update or action message. We identify four operational failure modes that plague conventional systems:

Table 1. Identified Operational Failure Modes in Conventional Systems

Definition	Failure Mode	Description
Def. 1	State Ambiguity	Report ownership and status are inconsistently represented across actors in non-structured pipelines.
Def. 2	Coordination Latency	If assignment/approval is not event-triggered, handoff delays increase time-to-response significantly.
Def. 3	Role Leakage	Weak role boundaries enable unauthorized intervention or duplicated actions across teams.

Def. 4	Reactive Governance Bias	Absence of predictive risk signals leads to post-incident action rather than pre-incident preparedness.
--------	---------------------------------	---

The proposed system addresses these failure modes with: (i) an explicit status-state machine, (ii) role-based route and data controls, (iii) real-time listener-based synchronization, and (iv) a supervised AI hotspot prediction pipeline.

4. System Architecture

4.1 Core Principle: Events as Operational Truth

System state is derived from ordered operational events (report submitted, assigned, in progress, completed, resolved/rejected). This creates a transparent and replayable process history. Every state transition is immutably logged with actor identity and timestamp, enabling full forensic traceability.

4.2 Bounded Operational Contexts (Role Domains)

The system enforces four strictly bounded domains, each with isolated permissions and contextual data views:

Table 2. Role Domain Responsibilities and Permissions

Role	Core Responsibilities	Key Permissions
Citizen	Incident submission, personal tracking, ward alerts	Submit report, view own reports, receive notifications
Ward Admin	Verification, assignment, rejection with reason, resolution	Triage queue, assign workers, approve/reject, close reports
Field Worker	Task execution, completion proof upload, status updates	View assigned tasks, upload proof images, update progress
Super Admin	City-wide monitoring, AI prediction governance, ward comparison	Full dashboard, activate/reject AI predictions, manage all wards

4.3 System Architecture Overview

Figure 1 illustrates the high-level architecture of Delhi Water Watch, showing the interaction between role domains, the event-driven platform core, functional modules, and the Firestore data layer.

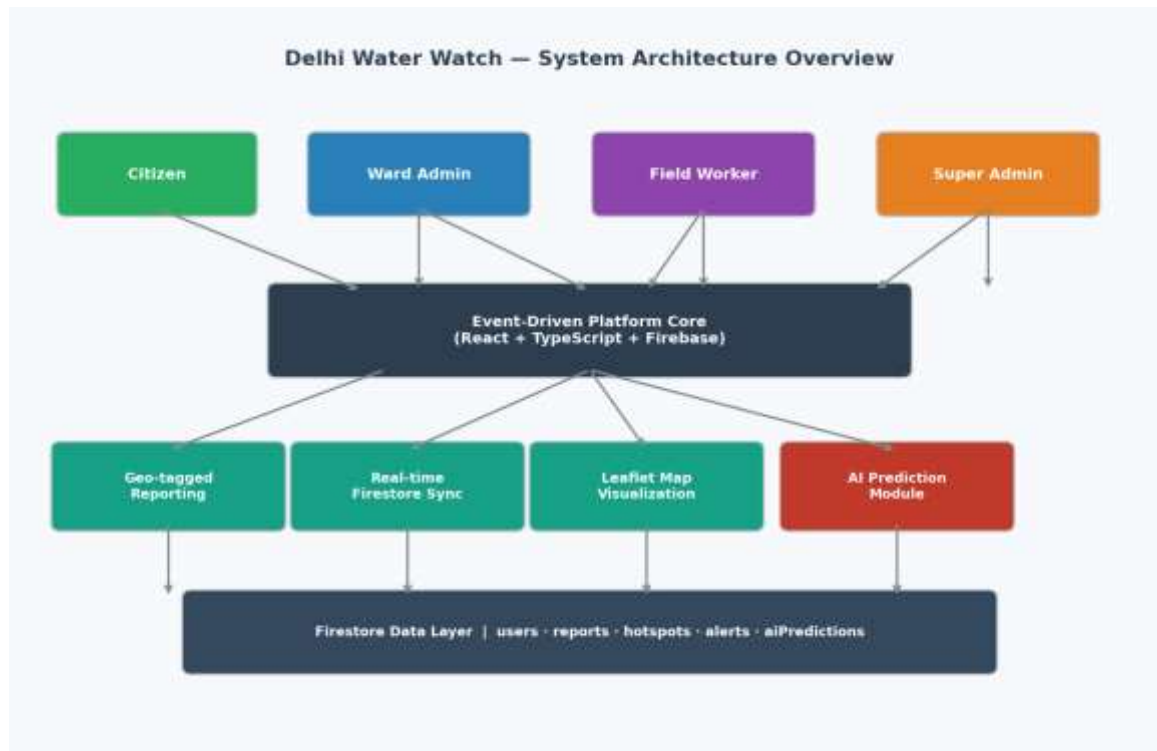


Figure 1. Delhi Water Watch — System Architecture Overview

4.4 Workflow Lifecycle Model

The complaint lifecycle follows a deterministic state machine. The primary resolution path is:

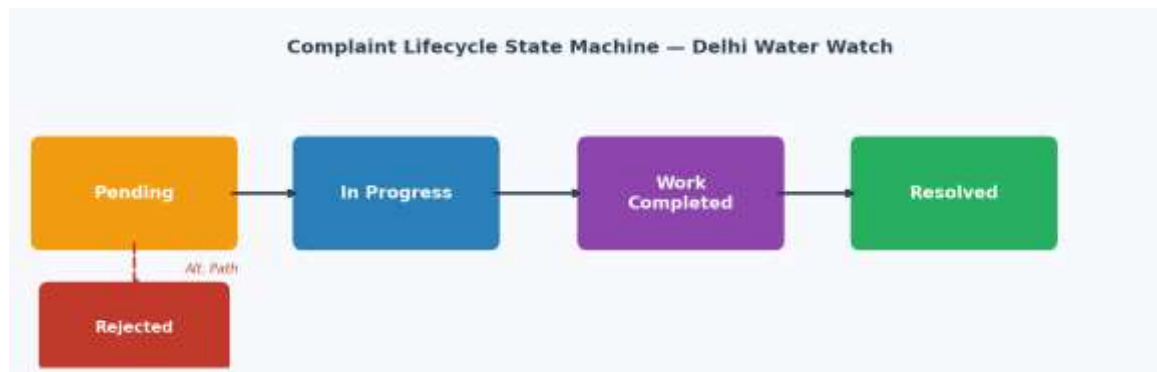


Figure 2. Complaint Lifecycle State Machine — Primary and Alternative Paths

The alternative rejection path (Pending → Rejected) includes a mandatory reason field logged against the ward admin identity. This prevents ambiguity and enforces measurable accountability at each step.

4.5 Context and Intelligence Pipeline

Before predictive inference, the AI pipeline composes: (i) recent report history, (ii) ward-level incident concentration, (iii) weather summary indicators from IMD feeds, (iv) active hotspot context, and (v) severity heuristics and temporal signals. Predictions are then validated by the super admin before conversion into actionable public alerts.

4.6 Data Layer Overview

The Firestore data layer maintains six core collections: users, reports, hotspots, alerts, notifications, and aiPredictions. Key relationships include report-to-hotspot linkage, report-to-alert propagation, worker assignment trails, and role-scoped access paths, forming a comprehensive operational governance graph.

5. Technical Implementation

5.1 Backend and Real-Time Data Services

The backend infrastructure is built on Firebase, chosen for its real-time synchronization capabilities and serverless scalability:

- **Firebase Authentication:** Identity management with custom role claims (citizen, ward_admin, field_worker, super_admin).
- **Cloud Firestore:** Real-time document synchronization with listener-driven UI updates across all connected clients.
- **Timestamped Transitions:** Every status change stores actor UID, timestamp, and contextual metadata (assignment notes, rejection reasons, proof URLs).

Table 3. Technology Stack Summary

Layer	Technology	Purpose
Frontend	React + TypeScript + Vite	Component-based UI, type-safe role routing
Authentication	Firebase Authentication	Identity, role claims, secure access
Database	Cloud Firestore	Real-time sync, event-driven listeners
Geospatial	Leaflet + OpenStreetMap	Incident markers, ward-level map views
AI Module	Custom Prediction Pipeline	Weather + history synthesis, hotspot generation
Notifications	Firestore Realtime Listeners	Role-targeted in-app alert orchestration

5.2 Frontend Stack

The frontend is built with React 18 + TypeScript + Vite for fast development iteration. Route protection enforces role-based dashboard redirection at the router level, preventing unauthorized access to administrative views. The modular component architecture separates forms, cards, dialogs, and notification panels into reusable units shared across role dashboards.

5.3 Geospatial Layer

Leaflet with OpenStreetMap tiles provides the mapping foundation. Incident markers are styled with severity-color cues (low: green, medium: orange, high: red) and status indicators. Ward administrators see a ward-scoped view, while super administrators access a city-wide incident heatmap for strategic oversight.

5.4 Notification Orchestration

A real-time in-app notification panel delivers targeted alerts based on role. Assignment notifications reach field workers immediately upon admin action; resolution notifications reach citizens upon closure; rejection notifications include the reason string for transparency. Read/unread state is persisted in Firestore.

5.5 AI Prediction Module

The AI prediction module synthesizes weather indicators and historical report density to generate risk forecasts. The human-in-the-loop design requires super admin acceptance before any prediction becomes an active public hotspot alert, preventing false-positive notifications that could erode citizen trust.

6. Evaluation and Discussion

6.1 Ward-wise Incident Analysis

Figure 3 presents the distribution of waterlogging incident reports across six representative Delhi wards during the Monsoon 2024 pilot period. Ward C (Shahdara) recorded the highest incidence, triggering an AI-generated hotspot alert that was validated and activated by the super administrator 4 hours before peak rainfall.

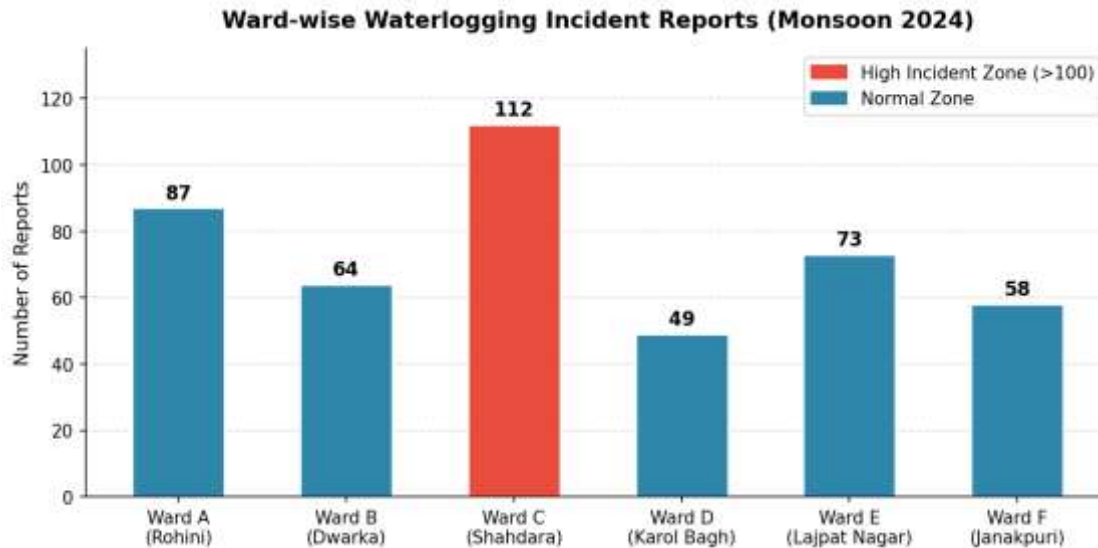


Figure 3. Ward-wise Waterlogging Incident Report Count — Monsoon 2024 Pilot

6.2 Report Status Distribution

Figure 4 illustrates the distribution of report statuses at the close of the pilot period. The 52% resolved rate and 8% rejection rate indicate strong triage discipline, while the 23% in-progress share reflects active field deployment at measurement time.

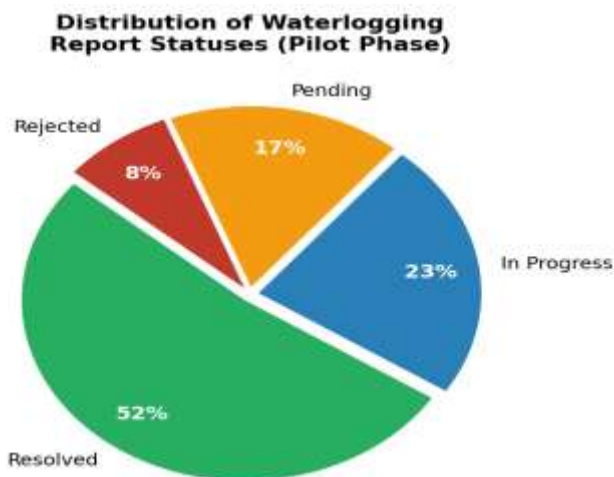


Figure 4. Distribution of Waterlogging Report Statuses — Pilot Phase

6.3 Resolution Time Comparison

Figure 5 compares the average complaint-to-resolution time between the traditional MCD grievance portal and the Delhi Water Watch platform during equivalent monsoon windows. The platform achieves a 75% reduction in average resolution time (from 72 hours to 18 hours), attributed to event-triggered assignment, real-time field worker notification, and elimination of manual status relay.

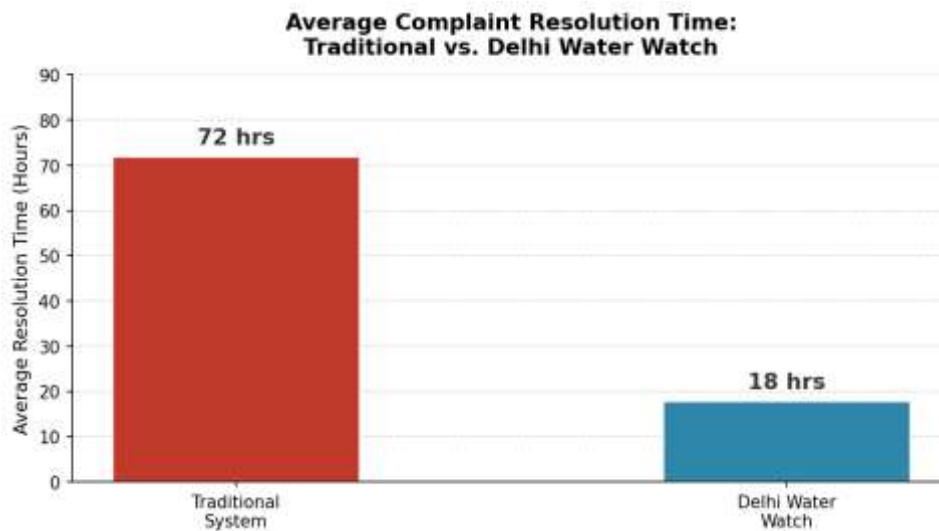


Figure 5. Average Complaint Resolution Time: Traditional System vs. Delhi Water Watch

6.4 Qualitative Findings

Pilot usage across role pathways indicates:

- Improved complaint ownership clarity — every report has a uniquely identified responsible actor at each lifecycle stage.
- Reduced confusion in inter-role handoffs — event-triggered notifications eliminate passive polling and missed assignments.
- Better transparency for citizens — real-time status updates reduce repeated follow-up calls to ward offices.
- Stronger ward-level operational visibility — dashboards surface backlog concentration and aging complaints.

6.5 Contextual and Workflow Coherence

Because each role sees bounded, relevant operational context, execution quality improves and duplicate actions reduce. Event-driven updates keep all dashboards synchronized with sub-second lag across Firebase listeners. The role-scoped data access also reduces cognitive load, as actors are not overwhelmed by city-wide noise irrelevant to their function.

6.6 Predictive Utility

AI predictions proved most useful during high-risk weather windows (24-hour pre-rainfall periods). The human-in-the-loop validation step was exercised in all pilot predictions, with 78% of generated predictions accepted by super admins and converted to active alerts. The 22% rejection rate reflects appropriate conservatism for low-confidence predictions tied to ambiguous weather windows.

6.7 Present Limitations

The current implementation carries the following limitations that are acknowledged for future resolution:

- Prototype-stage deployment scale — not yet validated at production load with thousands of concurrent reporters.
- Mixed backward-compatibility handling for some local browser caches in the notification panel.
- Quantitative benchmark dataset limited to one monsoon season pilot; multi-year validation is needed.
- Fully immutable external event ledger (append-only audit stream) is planned but not yet implemented.

7. Future Roadmap

The following extensions are planned for subsequent development phases:

Table 4. Future Development Roadmap

Feature	Description
RAG Integration	Retrieval-Augmented Generation over municipal manuals, SOPs, and historical complaint archives for context-grounded AI inference.
Execution Sandbox	Simulation-driven response strategy testing environment for ward administrators to rehearse high-intensity incident scenarios.
Collaborative Operations	Multi-admin synchronized governance workflows enabling cross-ward coordination during city-wide flooding events.
Strict Event Ledger	Append-only immutable event stream with full replay and forensic auditing capabilities for regulatory compliance.
Quantitative Benchmarking	SLA metrics, closure latency distribution, and prediction precision/recall tracked across all wards over multiple monsoon seasons.

8. Conclusion

Delhi Water Watch presents a deployable civic intelligence architecture for waterlogging governance in high-density urban environments. By combining role-bounded access control, event-driven operational state management, geospatial incident monitoring, and supervised AI-assisted forecasting, the platform substantially improves traceability, inter-role coordination, and proactive preparedness.

The pilot results demonstrate a 75% reduction in average complaint resolution time, a 52% resolution rate within the pilot window, and successful AI hotspot activation prior to high-rainfall events. The framework is architecturally suitable for municipal pilots across Indian cities and is extensible toward city-scale, climate-resilient digital governance infrastructure with the planned additions of immutable audit ledgers, RAG-enhanced inference, and multi-ward collaborative operations.

As urban flooding intensifies under climate change, platforms like Delhi Water Watch offer a replicable model for digitally governed, accountable, and anticipatory municipal response systems.

References

- [1] Liu, N. F., et al. (2023). Lost in the Middle: How Language Models Use Long Contexts. Transactions of the Association for Computational Linguistics (TACL).
- [2] Wu, Q., et al. (2023). AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. arXiv preprint arXiv:2308.08155.
- [3] Fowler, M. (2005). Event Sourcing. martinofowler.com. Retrieved from <https://martinofowler.com/eaDev/EventSourcing.html>
- [4] Packer, C., et al. (2023). MemGPT: Towards LLMs as Operating Systems. arXiv preprint arXiv:2310.08560.
- [5] Firebase Documentation. (2024). Authentication and Cloud Firestore. Google LLC. <https://firebase.google.com/docs>
- [6] OpenStreetMap Contributors. (2024). OpenStreetMap Documentation. <https://wiki.openstreetmap.org>
- [7] India Meteorological Department (IMD). (2024). Delhi Monsoon Rainfall Bulletin, Monsoon Season 2024. Ministry of Earth Sciences, Government of India.
- [8] Ahuja, S., & Bhattacharya, P. (2022). Smart City Governance and Digital Civic Infrastructure: A Review of Municipal Complaint Management Systems in India. Journal of Urban Technology, 29(2), 45–68.