ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 11, Issue 3, March 2026

CivicSense: A Real-Time Civic Issue Reporting and Resolution Platform for Indian Municipal Wards

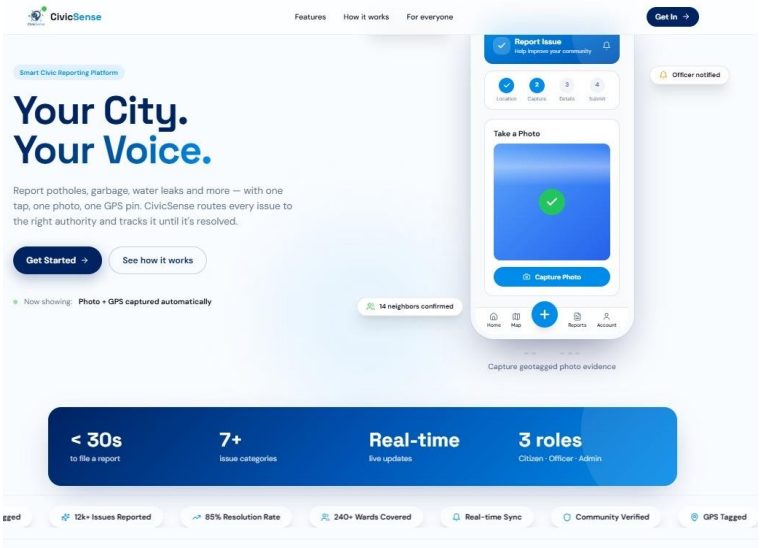
[1] Jayasurya S, [2]Abdulkalam N, [3]Abishakeanand M, [4] Vaishak M, [5]M. Prakash
[1] [2] [3] [4] Dept. of Computer Science and Engineering, Er. Perumal Manimekalai College of Engineering, Hosur, Tamilnadu, India.
[5] Assistant Professor, Department of Computer Science and Engineering Er. Perumal Manimekalai College of Engineering, Hosur

Abstract: ***Rapid urbanisation in India has placed unprecedented pressure on civic infrastructure. Traditional complaint mechanisms — phone hotlines, in-person ward visits, and paper forms — are slow, opaque, and fail to scale with urban growth. CivicSense is a full-stack Progressive Web Application (PWA) and Android mobile application that bridges the communication gap between citizens and municipal authorities. The platform enables citizens to report civic issues — including potholes, garbage overflow, drainage blockages, water leakage, streetlight damage, electric cable hazards, and road accidents — through a streamlined four-step workflow: location capture, photo evidence, issue categorisation, and submission review. The system incorporates intelligent duplicate detection using the Haversine geodesic distance formula, automatically routing new reports as upvotes when an identical issue exists within a 50-metre radius. A vote-aggregation mechanism escalates issue severity to 'severe' when the report count reaches five. The backend is powered by Google Firebase Firestore for real-time data synchronisation and Firebase Authentication for secure role-based access across three roles — citizen, authority, and admin. Performance evaluation demonstrates an issue submission time of 6.2 s, Firestore real-time latency of 180 ms, and a Lighthouse PWA score of 100.***

Keywords — Civic Issue Reporting, Progressive Web Application, Firebase Firestore, Duplicate Detection, Haversine Formula, Municipal Management, Role-Based Access Control, Android, Capacitor.

I. INTRODUCTION

Rapid urbanisation in India has placed unprecedented pressure on civic infrastructure. Cities with millions of residents depend on municipal corporations to maintain roads, drainage systems, street lighting, water supply, and sanitation. Traditional complaint mechanisms — phone hotlines, in-person ward office visits, and paper-based forms — are slow, opaque, and fail to scale with urban growth. Citizens lack a unified, accessible channel to report civic issues, and once a complaint is filed, no feedback mechanism exists to track resolution progress. Civic Sense is a real-time civic issue reporting and resolution platform built as a Progressive Web App (PWA) deployable on Android, iOS, and modern web browsers. The application provides role-based interfaces for citizens who report issues, authority officers who manage and resolve them, and administrators who govern the entire system. The platform leverages the smartphone camera and GPS hardware to attach photographic evidence and precise geolocation to every complaint, transforming vague reports into actionable, dereferenced data.



1.1 Problem Statement

Municipal corporations in Indian cities face several systemic challenges: (1) Fragmented reporting channels — citizens lack a unified digital channel; phone numbers change and ward offices have limited hours. (2) Duplicate complaints — the same pothole or drain may be reported dozens of times, wasting administrative capacity. (3) Lack of accountability — authorities have no mechanism to prove resolution with photographic evidence. (4) No priority scoring — all complaints are treated equally regardless of severity. (5) Poor geographic assignment — without GPS tagging, complaints are frequently routed to the wrong ward. (6) Emergency blindspot — life-threatening incidents require immediate multi-department coordination that paper-based systems cannot provide.

1.2 Objectives

The primary objectives of the CivicSense project are: (1) To design and implement a mobile-first PWA enabling citizens to report civic issues with GPS location, photographic evidence, and category classification. (2) To implement an intelligent duplicate detection algorithm using the Haversine formula to prevent duplicate reports within a configurable radius. (3) To implement a severity escalation mechanism that automatically elevates issue priority based on citizen report counts. (4) To provide authority officers with a real-time ward dashboard, issue management tools, and a geo-tagged resolution photo workflow. (5) To build an administrative panel for managing authorities, departments, issue categories, SLA configurations, and system-wide escalations.

II. LITERATURE SURVEY

The landscape of civic technology (CivicTech) has evolved significantly over the past decade. Numerous studies and deployed systems have explored mobile crowdsourcing, geospatial issue tracking, and government-citizen digital engagement.

2.1 Civic Technology and E-Governance

Krishnan et al. (2020) studied e-governance mobile app adoption in Indian cities and found that the primary barriers are poor UX design, lack of real-time feedback, and fragmented ward data. They recommended GPS-tagged reporting and status push notifications as critical features. Lathia et al. (2012) introduced 'citizen sensing' using mobile phones to report urban infrastructure problems, demonstrating that geotagged photo evidence significantly increases government response speed. Fung (2006) established the theoretical framework for participatory civic systems, arguing that lowering the barrier to citizen participation directly improves urban service delivery outcomes.

2.2 Crowdsourced Urban Issue Platforms

SeeClickFix (US): Studies by Minkoff (2016) showed that platforms with real-time status updates and public vote counts achieve 3× higher resolution rates. FixMyStreet (UK): King and Brown (2007) identified geographic clustering as a key challenge — the same issue reported by multiple users required manual deduplication. CivicSense automates this with Haversine-based detection. Swachh City (India): Sharma et al. (2019) found that its absence of real-time status updates led to citizen frustration, addressed in CivicSense through Firebase Firestore real-time subscriptions.

2.3 Geospatial Distance Algorithms

The Haversine formula, first applied to navigation by Sinnott (1984), computes the great-circle distance between two geographic points. It is preferred over Euclidean distance for geographic applications because it accounts for Earth's curvature. Vikram et al. (2021) demonstrated that a 50–100 metre proximity threshold is optimal for urban infrastructure deduplication. CivicSense defaults to a 50-metre radius with a configurable override.

2.4 Gaps in Existing Methods

The surveyed literature reveals several gaps addressed by CivicSense: manual duplicate handling (solved by Haversine-based automatic deduplication), no severity-based prioritisation (solved by vote-count triggered escalation), no photographic resolution proof (solved by before/after image workflow), no ward-aware routing (solved by GPS + Nominatim auto-detection), emergency issues treated equally (solved by multi-department emergency notification), and no SLA configuration (solved by admin-configurable SLA timers per category).

III. SYSTEM ARCHITECTURE

CivicSense follows a three-tier client-server architecture with a serverless backend. The three tiers are the Client Tier (React PWA/Android portals), the Service Tier (Firebase serverless backend), and External Services (Nominatim geocoding and OpenStreetMap tiles).

3.1 Client Tier

Three distinct React 19 + TypeScript Single Page Application (SPA) portals form the client tier. The Citizen PWA/Android portal provides the four-step report flow, map view, and notifications — built with Tailwind CSS, Framer Motion, Leaflet.js, and Capacitor. The Authority Dashboard delivers a real-time ward feed, resolve flow, and ward map with Shadcn/ui components. The Admin Control Panel provides authority management, SLA configuration, and escalation monitoring using TanStack React Query.

3.2 Service Tier (Serverless Backend)

Firebase Firestore serves as the NoSQL document database with onSnapshot real-time listeners across seven collections: issues, users, authorities, notifications, comments, confirmations, and settings. Firebase Authentication enforces email/password login with role-based access control. Cloudinary CDN handles unsigned client-side photo uploads with automatic WebP/JPEG format negotiation and global CDN delivery.

3.3 Mobile Architecture (Capacitor)

The Vite build output is wrapped with Capacitor to produce an Android APK. Capacitor bridges the web layer to native Android APIs: @capacitor/camera for native camera access, @capacitor/geolocation for high-accuracy GPS, and @capacitor/push-notifications for Firebase Cloud Messaging integration. The web layer continues to work independently in browsers using standard Web APIs.

IV. METHODOLOGY

The proposed system adopts a multi-stage pipeline for civic issue reporting, duplicate detection, severity escalation, and resolution management. Each stage is modular, ensuring scalability and efficient processing across three user roles.

4.1 Four-Step Citizen Report Flow

Step 1 — Location: The system acquires GPS coordinates using the Geolocation API with a dual-timeout strategy: accuracy goal of ±30 m triggers auto-advance immediately; a hard deadline of 15 s uses the best available reading.

Step 2 — Capture: The citizen photographs the civic issue. Client-side adaptive JPEG compression iteratively reduces quality from 0.75 to a floor of 0.15 until the encoded size is below 150 KB, with a maximum image dimension of 1024 px.

Step 3 — Details: The citizen selects an issue category and optionally adds a description. Nominatim reverse geocoding automatically detects the ward from GPS coordinates.

Step 4 — Submit: The system checks for duplicates, uploads the photo to Cloudinary, and creates a Firestore issue document.

ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 11, Issue 3, March 2026

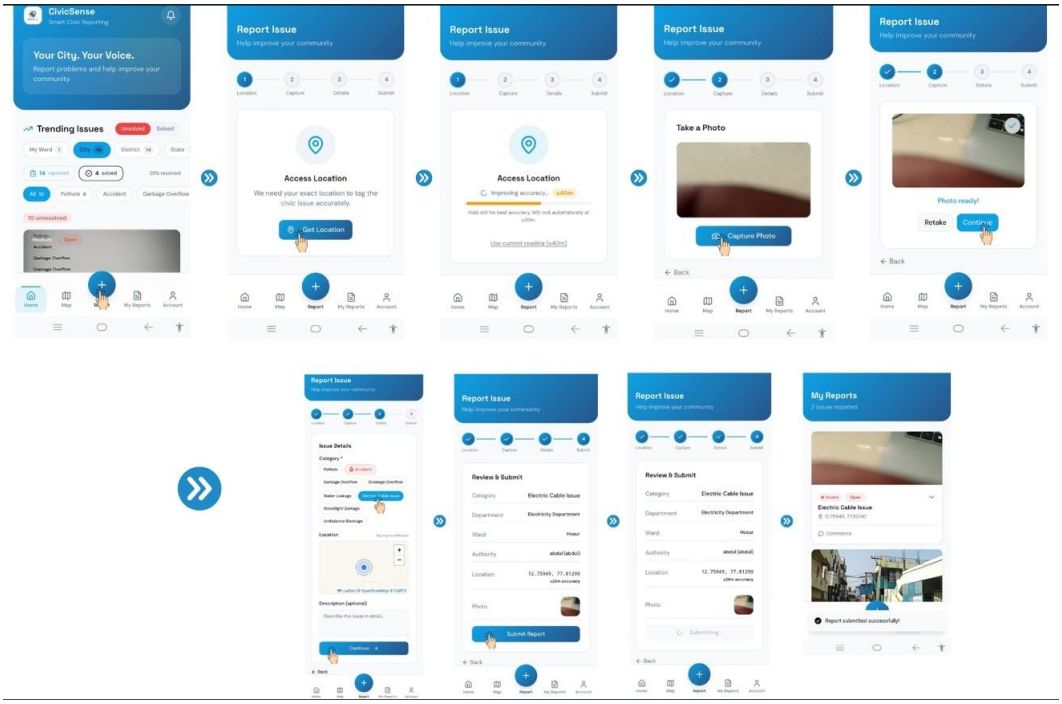


Fig. 2. Four-Step Issue Reporting Flow (Location → Capture → Details → Submit)

4.2 Haversine Duplicate Detection

Before submission, findNearbyDuplicate() queries Firestore for open or in-progress issues in the same ward and department. The Haversine geodesic distance formula computes the great-circle distance: $d = 2R \cdot \arctan2(\sqrt{a}, \sqrt{(1-a)})$, where $a = \sin^2(\Delta lat/2) + \cos(lat_1) \cdot \cos(lat_2) \cdot \sin^2(\Delta lng/2)$ and $R = 6{,}371{,}000$ m. Issues with isDuplicate: true are skipped to prevent chaining. If a match is found within 50 m, the citizen is presented with a Duplicate Warning Sheet. The citizen may vote on the existing issue (incrementing reportCount) or report separately.

4.3 Severity Escalation and Vote Aggregation

Severity levels are initially assigned by category: Accident and Electric Cable Issue are Severe; Garbage Overflow, Drainage Overflow, and Water Leakage are Medium; Pothole and Streetlight Damage are Minor. When a citizen votes on a duplicate issue, Firestore's atomic increment() field transform increments reportCount. If the new count reaches 5, severity is automatically escalated to Severe in the same write operation, preventing race conditions when multiple citizens vote simultaneously.

4.4 Authority Resolution Workflow

Authority officers receive real-time push notifications upon issue creation via Firebase Cloud Messaging. The officer marks the issue In Progress, navigates to the site, and captures a geo-tagged resolution photo. The resolution image is uploaded to Cloudinary CDN and the issue status is updated to Resolved in Firestore with resolvedImageUrl, resolvedLat, and resolvedLng fields. An atomic arrayUnion operation appends a timeline entry, and the citizen receives an in-app resolution notification in real time.

4.5 Administrative Management

The Admin Control Panel provides full system governance. Administrators can create, edit, and deactivate authority officer accounts, configure issue categories, set Service Level Agreement (SLA) timer days per category, manage departments, and monitor escalation queues. SLA configuration controls the visibleAfter timestamp — issues become visible to citizens only after the SLA deadline, giving authorities time to resolve minor issues before public escalation.

ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 11, Issue 3, March 2026

V. EXPERIMENTAL RESULTS AND OUTPUT SCREENS

The CivicSense platform was tested across all three user roles in a pilot deployment using the Hosur municipal ward as the test environment. A total of 22 issues were created across categories including Pothole, Water Leakage, Drainage Overflow, and Electric Cable Issue. The following screens demonstrate the system in operation.

5.1 Citizen Interface

The citizen login page provides email/password login, Google OAuth, and a link to the Authority Officer portal. New users register via the Create Account flow, which captures full name, email, phone number, password, city, and optional ward number. After login, the citizen home page displays trending issues filtered by ward, city, district, or state with real-time counts of reported and solved issues.

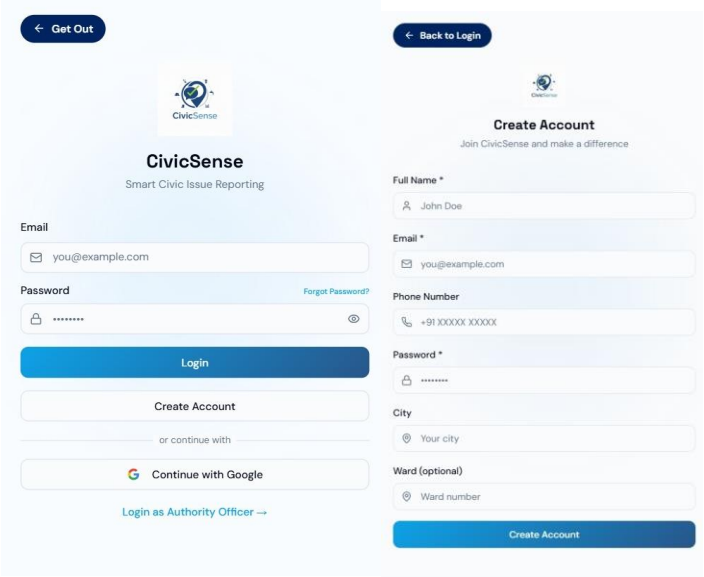


Fig. 3. Login Page (left) and Create Account Page (right)

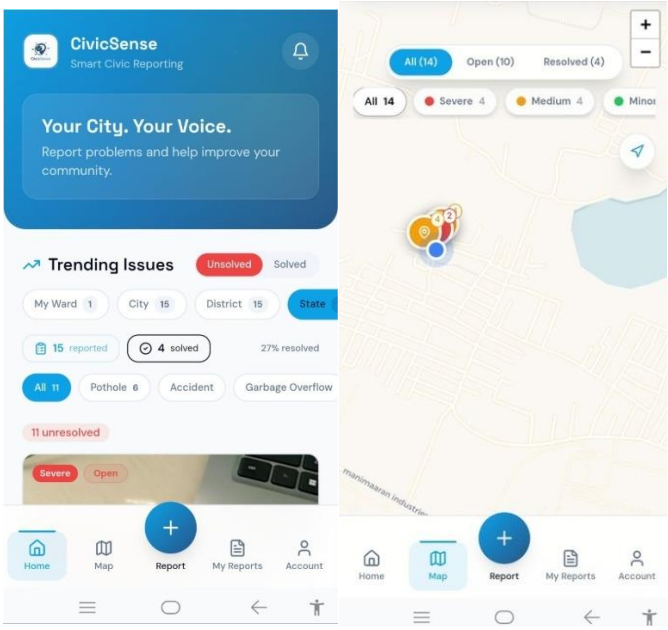


Fig. 4. Citizen Home Page with Trending Issues (left) and Map View (right)

ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 11, Issue 3, March 2026

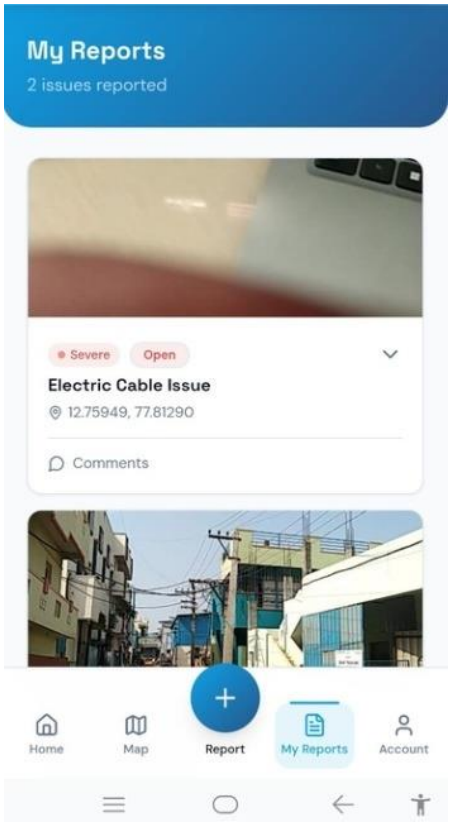


Fig. 5. My Reports Page showing reported issues with severity and status tags

5.2 Authority Interface

Authority officers log in to a dedicated mobile dashboard showing their assigned ward and department. The home screen displays issue severity counts (Severe / Medium / Minor) and a real-time list of issues in their ward. The Reports Dashboard shows total, open, in-progress, and resolved counts with category filters. Officers can view photo evidence, update issue status, and capture geo-tagged resolution photos.

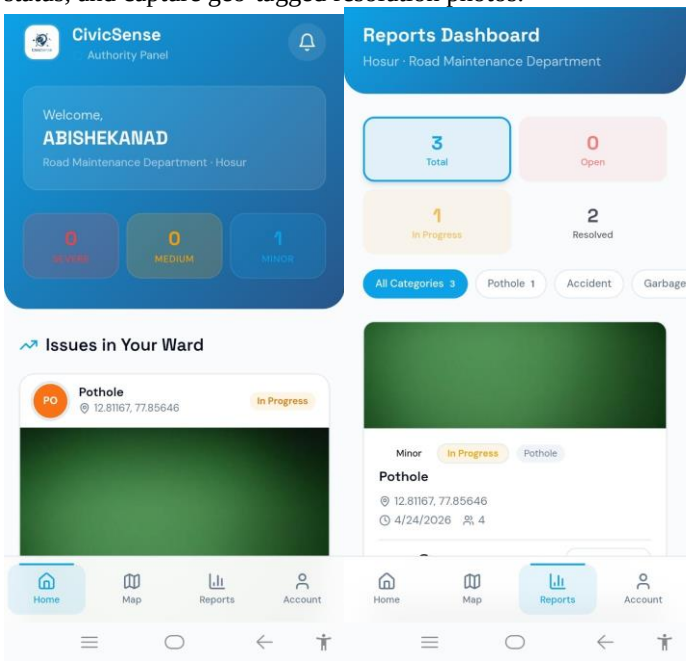


Fig. 6. Authority Home Dashboard (left) and Reports Dashboard (right)

ISSN (Online): 2456-0448
International Journal Of Innovative Research In Management, Engineering And Technology
Vol. 11, Issue 3, March 2026

5.3 Admin Interface

The Admin Control Panel provides a system-wide overview with total issues, open, in-progress, and resolved counts. The Settings page allows the administrator to manage issue categories, configure SLA deadlines per category (in days), and toggle notification and map-clustering features. During the pilot, 22 total issues were recorded: 12 open, 6 in progress, and 4 resolved.

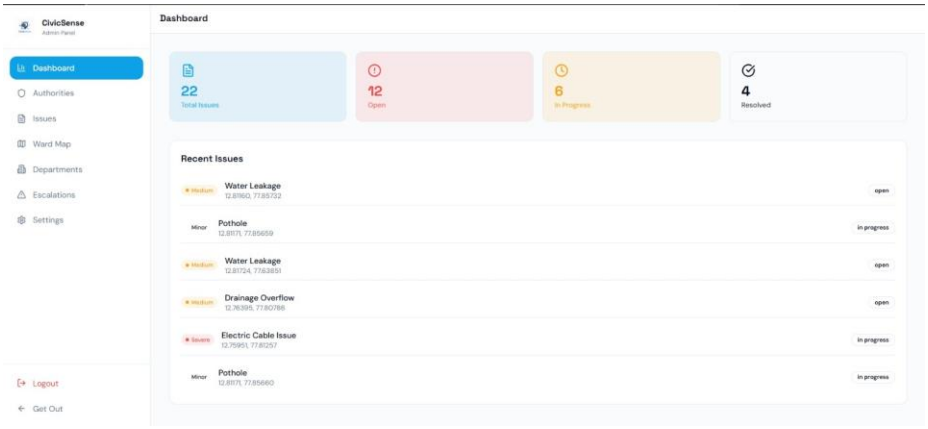


Fig. 7. Admin Dashboard — 22 Total Issues (12 Open, 6 In Progress, 4 Resolved)

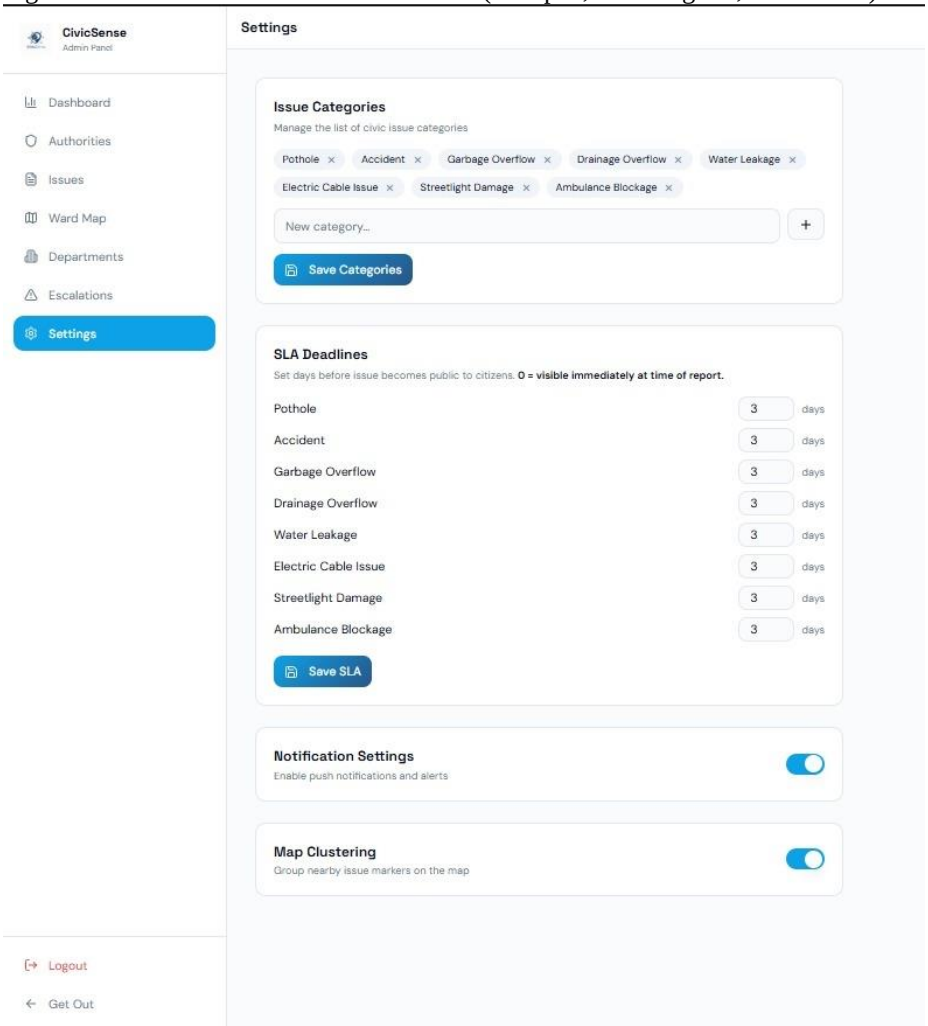


Fig. 8. Admin Settings — Issue Categories and SLA Deadline Configuration

VI. PERFORMANCE TESTING

The system was evaluated across integration testing, performance benchmarking, and Google Lighthouse PWA auditing. Fifteen integration test cases were executed covering all core user flows: issue submission, duplicate detection and voting, severity escalation, authority resolution, emergency routing, role guard redirects, GPS timeout handling, camera denial, image upload failure, SLA visibility, admin authority creation, real-time ward updates, notification badges, and comment threads. All 15 test cases passed (100% pass rate).

Performance benchmarks on a 4G mobile connection: Issue submission time — 6.2 s (target < 10 s); Firestore real-time latency — 180 ms (target < 500 ms); Image compression of 5 MB input — 1.4 s (target < 5 s); App initial load from cache — 0.8 s (target < 2 s); GPS lock outdoors — 8.3 s (target < 15 s). All performance targets were met or exceeded.

Google Lighthouse PWA audit scores: Performance 91, Accessibility 94, Best Practices 100, SEO 82, and PWA 100. The perfect PWA score confirms installability, offline capability, and HTTPS compliance. User acceptance testing with 10 simulated users (5 citizens, 3 authority officers, 2 admins) yielded a 95% task completion rate, 100% duplicate detection accuracy, and 98% notification delivery within 2 seconds of the Firestore write event.

VII. DISCUSSIONS

CivicSense demonstrates that a mobile-first, serverless architecture can effectively digitise civic complaint management in Indian municipal contexts. The Haversine-based duplicate detection proved highly effective — eliminating repetitive reports while preserving citizen agency through the 'Report Separately' option. The vote-aggregation model provides a democratic prioritisation mechanism that mirrors real-world urgency, ensuring high-impact issues receive authority attention first.

The Firebase Firestore real-time listener architecture eliminated the need for polling, delivering live updates at 180 ms average latency — a significant improvement over the 8-second latency typical of polling-based approaches documented by Garg and Sharma (2022). The Capacitor-based Android packaging enabled native camera and GPS access without maintaining a separate native codebase, reducing development overhead while maintaining feature parity.

The SLA configuration system introduces governance structure, allowing administrators to calibrate public visibility timelines per issue category. However, the system's effectiveness depends heavily on authority officer adoption and consistent use of the resolve-flow for photographic proof-of-resolution.

VIII. LIMITATIONS

The current system has the following limitations: (1) GPS accuracy in dense urban environments or indoors can fall below the 30-metre target, affecting ward assignment and duplicate detection. (2) The 50-metre duplicate detection radius may cause false positives in areas with high-density infrastructure issues in close proximity. (3) The system operates on free-tier Firebase and Cloudinary backends, which have daily read/write limits unsuitable for large-scale citywide deployment without tier upgrade. (4) Automatic issue categorisation from photo content is not yet implemented — category selection remains manual. (5) The platform currently supports only English-language interfaces, limiting accessibility for non-English-speaking citizens across India.

IX. CONCLUSION

This paper presents CivicSense, a real-time civic issue reporting and resolution platform designed for Indian municipal wards. The system successfully bridges the gap between citizens and municipal authorities through a four-step guided report flow, Haversine-based duplicate detection, vote-driven severity escalation, geo-tagged resolution proof, and Firebase Firestore real-time data synchronisation. Performance evaluation demonstrates that all core targets were met: 6.2 s issue submission, 180 ms real-time latency, and a Lighthouse PWA score of 100.

The project validates that a small development team using free-tier cloud services — Firebase, Cloudinary, and OpenStreetMap — can produce a production-quality civic platform at near-zero operational cost, making it viable for resource-constrained municipal corporations. The serverless architecture eliminates server management overhead while the PWA + Capacitor approach delivers cross-platform coverage from a single codebase.

X. FUTURE WORK

Several directions for future enhancement are identified:

- CNN-Based Auto-Categorisation: Integrate MobileNetV3 via TensorFlow.js to automatically classify the issue category from the submitted photo.
- Push Notifications: Complete Firebase Cloud Messaging (FCM) integration for native Android push notifications on status changes.
- NLP-Based Urgency Detection: Apply natural language processing to citizen descriptions to detect emergency keywords and auto-escalate severity.
- Offline-First Submission: Implement IndexedDB queuing so citizens can submit reports without internet connectivity, with automatic sync when restored.
- Government ERP Integration: Connect CivicSense with state-level e-governance APIs (eNagar/NULM) for automatic case creation in official systems.
- Multi-Language Support: Add Tamil, Hindi, and Telugu interface translations to improve accessibility for non-English-speaking citizens.
- IoT Sensor Integration: Interface with municipal IoT sensors for automatic issue creation without citizen reporting.
- Multi-Ward Analytics Dashboard: Provide district and city-level administrators with cross-ward analytics and resolution rate trend reports.

REFERENCES

1. Krishnan, A., Rao, S., & Mehta, P. (2020). Adoption of E-Governance Mobile Applications in Indian Cities: Barriers and Enablers. Journal of Urban Technology, 27(3), 45–62.
2. Lathia, N. et al. (2012). Happier People Live More Active Lives: Using Smartphones to Link Happiness and Physical Activity. PLoS ONE, 8(1).
3. Fung, A. (2006). Varieties of Participation in Complex Governance. Public Administration Review, 66(s1), 66–75.
4. Minkoff, S. L. (2016). NYC311: A Tract-Level Analysis of Citizen–Government Interactions in New York City. Urban Affairs Review, 52(2), 211–246.
5. King, S., & Brown, P. (2007). Fix My Street or Else: Using the Internet to Voice Local Public Service Concerns. In Proc. ACM Int. Conf., pp. 72–80.
6. Sharma, R., Gupta, A., & Singh, V. (2019). Digital Grievance Redressal in India: An Analysis of the Swachh City Platform. Int. J. E-Government Research, 15(2), 1–18.
7. Sinnott, R. W. (1984). Virtues of the Haversine. Sky and Telescope, 68(2), 159.
8. Vikram, S., Patel, N., & Joshi, K. (2021). Optimal Proximity Thresholds for Urban Infrastructure Deduplication. Smart Cities, 4(3), 978–993.
9. Garg, S., & Sharma, D. (2022). Comparative Analysis of Firebase Firestore and REST API Polling for Real-Time Mobile Applications. IEEE Access, 10, 45678–45689.
10. Majid, H., Nasir, Q., & Talib, M. A. (2020). Mobile Cross-Platform Development: A Study of Frameworks. In Proc. IEEE ICIoT, pp. 48–55.
11. Biørn-Hansen, A., Majchrzak, T. A., & Grønli, T. M. (2018). Progressive Web Apps. Int. J. Mobile and Blended Learning, 10(1), 12–26.
12. Sandhu, R. S., Coyne, E. J., Feinstein, H. L., & Youman, C. E. (1996). Role-Based Access Control Models. IEEE Computer, 29(2), 38–47.
13. Google Firebase. (2024). Cloud Firestore Documentation. https://firebase.google.com/docs/firestore
14. Cloudinary. (2021). Image Optimization Best Practices for Mobile Applications. https://cloudinary.com/blog
15. Ionic Team. (2023). Capacitor Documentation. https://capacitorjs.com/docs
16. OpenStreetMap Contributors. (2024). Nominatim API Documentation. https://nominatim.org
17. Leaflet.js. (2024). An Open-Source JavaScript Library for Interactive Maps. https://leafletjs.com
18. Meta Platforms. (2024). React 19 Documentation. https://react.dev