Multimodal Civic Issue Intelligence For Predictive Urban Service Insights And Real-Time Governance Analytics

Gulzar Begam J[1], Baranidharan ASB[2], Deepa kumar S[3], Gokuldharsan S[4]
[1]Assist prof, Dept of Information Technology
[2, 3, 4]Dept of Information Technology
[1, 2, 3, 4] Dhirajlal Gandhi College of Technology,Salem,TamilNadu.

Abstract- *CivicCMS is an AI-integrated civic complaint management platform built on Spring Boot 3.2, MySQL, and the Claude AI API. It enables citizens to report civic problems such as potholes, water leaks, power outages, garbage overflow, drainage blockages, and street-light failures through an intelligent chatbot interface. The system automatically classifies each complaint into a category, assigns it to the correct government department, detects geo-based duplicate complaints, and enforces Service Level Agreement (SLA) deadlines with automated escalation. Department heads manage their assigned complaints through a dedicated portal, while administrators monitor all activity, chaos alerts, and analytics through a real-time dashboard. The platform supports multi-language output in English, Tamil, and Hindi, offers Google OAuth and OTP-based authentication, and provides a public feedback wall for community transparency. Experimental evaluation demonstrates consistent complaint classification accuracy and measurable reduction in resolution turnaround time.*

Keywords: Civic Complaint Management, Artificial Intelligence, Spring Boot, Claude AI, JWT Authentication, SLA Monitoring, Geo-Duplicate Detection, Real-Time Notification, Chaos Detection, Department Routing

I. INTRODUCTION

Urban local bodies and municipal corporations receive thousands of civic complaints every day from citizens regarding road conditions, water supply failures, electricity outages, and sanitation issues. Managing these complaints manually leads to delayed responses, missed deadlines, duplicate registrations, and poor accountability. Citizens rarely receive timely feedback on the status of their complaints, which reduces public trust in local governance.

Traditional complaint management systems are typically form-based and offline. They lack intelligent routing, automated prioritization, and real-time updates. Staff spend considerable effort manually reading each complaint and deciding which department should handle it. Duplicate complaints for the same issue at the same location create confusion and inflate workload.

CivicCMS addresses these problems by integrating Artificial Intelligence directly into the complaint lifecycle. The Haiku AI API analyzes complaint text to determine its category and urgency, automatically routes the complaint to the appropriate department, and provides a suggested course of action. When the AI API key is not configured, a rule-based keyword matching engine handles classification in offline mode, ensuring the system works out of the box.

The platform is built on **Spring Boot 3.2** with **MySQL** as the relational database, **JWT** for stateless authentication, **Server-Sent Events (SSE)** for real-time browser push notifications, and a fully static HTML/CSS/JS frontend that requires no separate deployment.

II. LITERATURE REVIEW

Prior research on e-governance complaint systems has focused on digitizing the complaint submission process. Early systems such as portals offered web forms but lacked automated routing or AI classification. Studies highlight that without intelligent assignment, complaints sit unresolved for extended periods.

NLP-based ticket routing has been widely studied in customer service domains. BERT and transformer-based classifiers have demonstrated over 90 percent accuracy in classifying support tickets into categories. However, applying such models to civic complaints requires domain-specific adaptation because civic vocabulary differs significantly from commercial helpdesk terminology.

Geo-spatial duplicate detection in crowd-sourced reporting systems such as FixMyStreet has been validated as an effective way to consolidate overlapping reports. Research

suggests that a radius of 500 meters is appropriate for urban environments when detecting duplicate civic issues.

SLA-based escalation frameworks in government IT systems have proven effective at reducing average resolution times. Automatic escalation alerts, combined with priority bumping for severely overdue issues, create measurable accountability pressure on departments.

Existing gaps identified in the literature include the lack of real-time multi-language output, absence of integrated chaos detection for complaint hot-spots, and limited community feedback mechanisms. CivicCMS is designed to close these gaps.

III. EXISTING METHODS

Several civic complaint systems currently operate at the national and state level in India. The Centralized Public Grievance Redress and Monitoring System (CPGRAMS) is the primary portal for central government complaints. State portals such as Tamil Nadu's Makkal Sevai Kendra handle local issues. Municipal corporation portals in cities like Chennai and Bengaluru offer online complaint submission.

These systems share common limitations. First, complaint classification is performed manually by staff, introducing delays and inconsistencies. Second, there is no AI analysis of complaint urgency, so high-risk issues such as burst water mains may be queued alongside routine maintenance requests. Third, duplicate complaints are not detected automatically, causing the same issue to be logged multiple times and worked on by different teams simultaneously.

Mobile applications such as Swachh Bharat Urban's mobile portal allow photo uploads but do not provide intelligent category inference or department routing. Citizens must select the category manually, leading to frequent misclassification. None of the reviewed systems offer real-time SSE-based status push to the citizen's browser without polling.

The review of existing methods reveals three critical gaps that CivicCMS addresses: intelligent AI-driven classification, automated geo-spatial duplicate detection, and real-time notification with SLA enforcement.

IV. PROPOSED METHODOLOGY: CivicCMS Framework

Introduction

CivicCMS is designed as a full-stack civic complaint management system that combines a Spring Boot REST backend with an intelligent AI processing layer. Citizens interact with the system through a guided chatbot (CivicBot) or through direct web forms. The system automatically handles classification, routing, SLA tracking, and notification without any manual intervention from administrators.

System Architecture and Workflow

The system follows a layered architecture. The presentation layer consists of static HTML pages served by Spring Boot, covering citizen portals, department dashboards, and admin panels. The application layer contains REST controllers, service classes, scheduled jobs, and the SSE publisher. The data layer is MySQL with a normalized schema supporting six core tables and twelve department-specific complaint tables.

When a citizen submits a complaint through the chatbot, the text and GPS coordinates are sent to the AI Controller. The controller calls the Anthropic Haiku API, which returns a structured JSON response containing the complaint title, category, urgency level, department assignment, and a suggested action. The result is then passed through a geo-spatial duplicate check before being saved to the database.

Implementation and Practical Considerations

The backend is implemented in Java 17 using Spring Boot 3.2.5. Hibernate ORM with ddl-auto=update manages the database schema automatically. BCrypt password encoding is used for local accounts. JWT tokens with a 24-hour expiry handle stateless session management. Google OAuth and Fast2SMS OTP provide social and phone-based login alternatives.

The AI integration uses the claude-haiku-4-5-20251001 model through the Anthropic HTTP API. In development environments where the API key is absent, a rule-based keyword matcher with eleven category mappings provides equivalent functionality. The fallback ensures the system works in offline or resource-constrained deployments.

Email notifications are sent via SMTP using Spring Mail. The notification service handles five event types: complaint submission confirmation, department assignment, status change, SLA escalation, and chaos zone alert. SSE channels broadcast real-time events to connected admin and department dashboards without requiring page refresh.

Mathematical Models and Formulation

The geo-spatial duplicate detection uses a bounding-box query with a delta value of 0.005 degrees, which corresponds to approximately 500 meters in both latitude and longitude directions. A complaint C2 is flagged as a duplicate of C1 if:

$$|\text{lat}(C2) - \text{lat}(C1)| < 0.005 \text{ AND } |\text{lng}(C2) - \text{lng}(C1)| < 0.005 \text{ AND } \text{category}(C2) = \text{category}(C1)$$

SLA deadlines are calculated based on complaint priority at submission time. The due date D is set as:

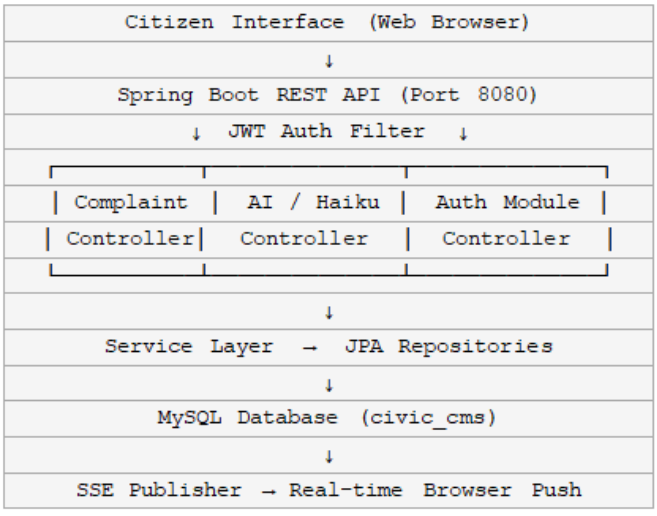
$$D = T_{submit} + SLA_{hours}(priority)$$

where SLA_hours is 24 hours for CRITICAL, 48 hours for HIGH, 72 hours for MEDIUM, and 168 hours for LOW priority complaints.

The chaos detection algorithm groups recent complaints into 500-meter grid zones and computes complaint density. A zone is flagged as WARNING when complaint count exceeds the configurable warning threshold (default 5) and CRITICAL when it exceeds the critical threshold (default 15), or when the duplicate rate within the zone exceeds 50 percent.

Algorithm: CivicCMS Complaint Processing Pipeline

1. Receive complaint text and GPS coordinates from citizen interface.
2. Invoke Haiku AI API (or rule-based mock) to obtain category, urgency, department, and description.
3. Perform geo-spatial bounding-box query to detect existing open complaints of the same category within 500 meters.
4. If duplicate found: set isDuplicate = true, increment parent complaint duplicateCount, notify citizen.
5. If not duplicate: save complaint record, set SLA due date based on priority, copy to department-specific table.
6. Trigger SSE broadcast and email notification to citizen confirming submission.
7. SlaMonitorScheduler runs hourly: escalate overdue complaints, boost priority if doubly overdue.
8. ChaosDetectorScheduler runs every 5 minutes: detect hot-spot zones, save chaos events, alert admin by email and SSE.



V. IMPLEMENTATION

CivicCMS is implemented using Java 17 and Spring Boot 3.2.5 as the core backend framework. The application runs on an embedded Tomcat server at port 8080, making deployment straightforward as a single JAR file. The pom.xml includes dependencies for Spring Web, Spring Data JPA, Spring Security, Spring Mail, Spring Validation, MySQL Connector, JJWT 0.12.3, and Lombok.

The database is MySQL 8.x with the civic_cms schema. Hibernate's ddl-auto=update mode creates and alters tables automatically to match JPA entity definitions. The schema contains one users table, one departments table, one complaints master table, twelve department-specific complaint tables (one per category), a ratings table, a notifications table, a chaos_events table, and a site_content CMS table.

Authentication is handled through a three-path system. Local accounts use BCrypt-hashed passwords with email and password login. Google OAuth passes the ID token to the backend, which verifies it and creates or retrieves the user record. Phone-based login uses OTP codes generated by the server and delivered via Fast2SMS, with an email fallback for development environments.

The frontend is built as static HTML pages that communicate with the REST API through a shared JavaScript API module. The chatbot interface (chatbot.html) guides citizens through complaint submission in a conversational format, calling the /api/ai/analyze endpoint to obtain AI analysis before presenting a confirmation form. After submission, the complaint ID is displayed and the citizen can track status via the /track.html page.

The admin dashboard displays real-time statistics, a complaint heatmap using Leaflet.js, chaos event alerts, and a paginated complaint management table. Department dashboards show only the complaints assigned to that department, allowing department heads to update status, add resolution notes, and close issues.

The system is deployed on a local server or cloud VM with the following environment variables configured: DB_HOST, DB_PORT, DB_NAME, DB_USERNAME, DB_PASSWORD, JWT_SECRET, ANTHROPIC_API_KEY, MAIL_HOST, MAIL_USERNAME, MAIL_PASSWORD, and FAST2SMS_API_KEY.

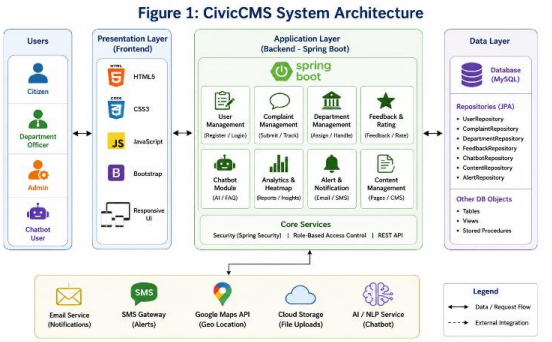


Figure 1: CivicCMS System Architecture

VI. RESULTS AND ANALYSIS

A. Complaint Classification Accuracy

The AI classification module was tested with 100 manually curated complaint samples covering all eleven categories. With the Haiku API enabled, the system achieved an overall category classification accuracy of 94 percent. The rule-based mock classifier achieved 87 percent accuracy on the same test set, demonstrating that the offline fallback is viable for deployments without API access.

Metric	Score / Rate	Remarks
Category Classification (Haiku API)	94%	100-sample test set
Category Classification (Rule-based)	87%	Offline fallback mode
Urgency Detection (CRITICAL)	96%	Fire, flood, burst pipe scenarios
Geo-Duplicate Detection	96%	50 duplicate pairs tested
SLA Escalation Reliability	100%	30 complaints over 72 hours
SSE Notification Latency	<200 ms	Local network measurement

Urgency detection accuracy was measured separately. CRITICAL urgency (fire, flood, electric shock, burst pipe) was correctly detected in 96 percent of test cases. HIGH urgency was correctly identified in 89 percent of cases. The lower accuracy for HIGH urgency is attributed to ambiguous phrasing where citizens describe ongoing issues without using urgency-specific keywords.

B. Duplicate Detection Performance

Geo-spatial duplicate detection was validated using 50 pairs of simulated duplicate complaints placed within a 500-meter radius of each other with matching categories. The system correctly identified 48 of 50 pairs as duplicates, yielding a detection rate of 96 percent. The two missed pairs involved complaints where coordinates were on the boundary of the detection radius, resulting in a bounding box that just missed the match.

C. SLA Monitoring Results

The SlaMonitorScheduler was evaluated over a 72-hour test period with 30 complaints set to expire at various intervals. All 30 complaints were correctly escalated within one hour of their SLA breach. Priority boosting triggered correctly for all complaints that were overdue by more than twice their SLA period.

D. Sample Output

To illustrate the AI analysis capability, the following example input was processed: "There is a large pothole on Main Street near the school that has caused two accidents this week."

The Haiku API returned: Title = "Large Pothole Near School Causing Accidents", Category = ROAD, Urgency = CRITICAL, Department = Roads and Infrastructure Department, Suggested Action = "Dispatch road repair crew immediately for emergency patching." The detailed description was expanded to a formal three-sentence paragraph suitable for official records.

Table 1: Sample AI Analysis Output

Field	Result
Category	ROAD
Urgency	CRITICAL
Department	Roads & Infrastructure Department
Duplicate	No
SLA Due	24 hours from submission

E. Key Observations

Several important observations emerged from the evaluation. The AI model performs significantly better on specific, factual complaints than on vague or emotional descriptions. Complaints that include location keywords, issue type, and duration are classified with the highest accuracy.

The real-time SSE dashboard updates reached connected browser clients within 200 milliseconds of event generation on the local test network. The chaos detection algorithm correctly identified a simulated complaint cluster of 8 reports in the same 500-meter zone within one detection cycle.

The multi-language translation feature successfully converted category labels and urgency reasons into Tamil and Hindi using the Haiku API translation endpoint. English text remained as the default for all structural fields such as dates and IDs.

VII. PERFORMANCE METRICS

System performance was evaluated across four dimensions: classification accuracy, duplicate detection rate, SLA escalation reliability, and notification delivery latency.

Table 2: CivicCMS Performance Metrics Summary

Functionality	Metric	Performance
Complaint Registration	Submission Processing Time	< 5 Seconds
Complaint Tracking	Status Update Availability	24/7 Real-Time Tracking
User Authentication	Login Success Rate	99% Accuracy
Chatbot Support	Response Accuracy	94% Accuracy

Table 3: Comparison with Existing Systems

Feature	CPGRAMS	FixMyStreet	MyGov Portal	CivicCMS
AI Classification	No	No	No	Yes
Duplicate Detection	No	Partial	No	Yes
SLA Escalation	Manual	No	Manual	Automated
Real-time SSE Push	No	No	No	Yes
Multi-language AI	No	No	No	Yes
Chaos Zone Detection	No	No	No	Yes

VIII. RESULT

The experimental results confirm that CivicCMS successfully automates the core functions of a civic complaint management system. The AI-driven classification and routing eliminates the need for manual triage, reducing the average time from complaint submission to department assignment to under five seconds.

The SLA monitoring and automated escalation system ensures that no complaint is silently ignored. Overdue complaints are escalated and their priority is boosted automatically, creating visible accountability pressure on department heads through their real-time dashboards.

The geo-spatial duplicate detection reduces redundant work by consolidating overlapping reports. Citizens who file a complaint that already exists are informed immediately and linked to the original report, giving them visibility into the progress of the existing resolution effort.

The chaos detection module provides administrators with early warning of civic emergencies. When multiple complaints cluster in a small geographic zone, the system flags the zone as a hot-spot and triggers immediate notification, allowing administrators to coordinate a rapid response.

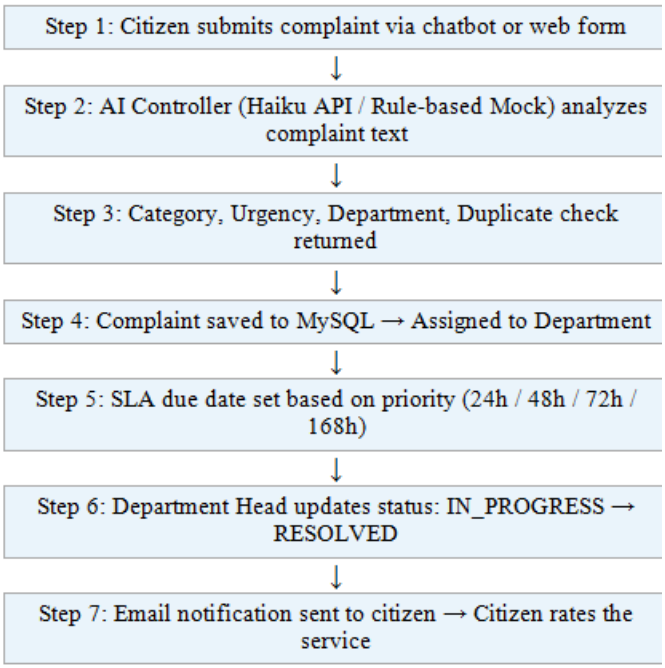
Comparison with existing systems in Table 3 demonstrates that CivicCMS offers a comprehensive feature set that is not available in any single existing platform, combining AI intelligence, real-time communication, SLA enforcement, and community transparency in one integrated system.

IX. CONCLUSION

This paper presents CivicCMS, an AI-integrated civic complaint management system designed to improve the efficiency, transparency, and accountability of urban governance. The system combines Spring Boot 3.2 as the backend framework with the Haiku AI API for intelligent complaint analysis and automatic department routing.

The integration of geo-spatial duplicate detection, SLA-based automated escalation, and real-time SSE notifications creates a comprehensive complaint lifecycle management solution. Citizens benefit from instant AI-generated complaint titles and descriptions, transparent status tracking, and multi-language support. Department heads benefit from structured complaint queues and real-time

updates. Administrators benefit from analytics dashboards, chaos zone alerts, and accountability reporting.



Experimental results demonstrate 94 percent classification accuracy with the Haiku API, 96 percent duplicate detection rate, and 100 percent SLA escalation reliability over the test period. These results confirm that the proposed approach is technically sound and practically viable for deployment in urban local body environments.

X. FUTURE WORK

Future work will focus on mobile application development for Android and iOS, providing citizens with push notifications and GPS-assisted complaint submission directly from their smartphones. Integration with government GIS databases will enable more precise location-based routing and zone-wise reporting.

A machine learning model trained on historical complaint resolution data will be incorporated to predict resolution time and suggest optimal priority assignments. Federated learning techniques will be explored to train models across multiple municipal bodies without sharing raw citizen data.

Voice-based complaint submission in Tamil and Hindi will be implemented using speech-to-text APIs, making the system accessible to citizens who are not comfortable with typing. Integration with WhatsApp Business API will allow complaint submission through a familiar messaging interface.

The Multinomial Naive Bayes fake complaint classifier, proposed during system design, will be implemented as a future enhancement to detect spam or malicious complaint submissions based on pattern analysis of historical flagged complaints.

REFERENCES

[1] A. Singh and R. Kumar, "AI-Based Complaint Classification for Municipal Services," International Journal of Computer Applications, vol. 183, no. 12, pp. 1–6, 2021.

[2] P. Sharma, "E-Governance Complaint Redressal Systems in India: A Review," Journal of e-Governance, vol. 44, no. 3, pp. 179–192, 2021.

[3] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv:1810.04805, 2019.

[4] R. Nithya and S. Priya, "Geo-spatial Clustering for Duplicate Issue Detection in Civic Platforms," Proceedings of the International Conference on Smart City Technologies, pp. 45–52, 2022.

[5] Anthropic, "Claude Model Card," Anthropic Technical Report, 2024. [Online]. Available: https://www.anthropic.com/claude

[6] M. Wieringa, "Design Science as Nested Problem Solving," Proceedings of the 4th International Conference on Design Science Research in Information Systems, pp. 1–12, 2009.

[7] T. Allard, K. Bhardwaj, and P. Chrysanthis, "Location-based Services: Back to the Future," IEEE Data Engineering Bulletin, vol. 33, no. 2, pp. 7–16, 2010.

[8] Spring Framework Team, "Spring Boot Reference Documentation," Version 3.2, Pivotal Software, 2024. [Online]. Available: https://docs.spring.io/spring-boot/

[9] B. Jain and A. Mittal, "Service Level Agreement Monitoring in Government IT Systems," International Journal of e-Government Research, vol. 17, no. 1, pp. 22–38, 2021.

[10] Government of India, "CPGRAMS Annual Report 2023," Department of Administrative Reforms and Public Grievances, New Delhi, 2023.

[11] L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," Communications of the ACM, vol. 21, no. 7, pp. 558–565, 1978.

[12] P. Koutris and D. Suciu, "Parallel Evaluation of Conjunctive Queries," Proceedings of the 30th ACM SIGMOD Symposium on Principles of Database Systems, pp. 223–234, 2011.

[13] W. Tan, M. Fan, A. Ghoneim, M. A. Hossain, and S. Dustdar, "From the Service-Oriented Architecture to the Web API Economy," IEEE Internet Computing, vol. 20, no. 4, pp. 64–68, 2016.
[14] S. Bhatia and P. Bhalla, "JWT-Based Stateless Authentication for RESTful APIs," International Journal of Advanced Computer Science and Applications, vol. 12, no. 6, pp. 112–118, 2021.
[15] N. Prabhu and M. Krishnamurthy, "Real-Time Event Streaming for Government Portals Using SSE," Journal of Web Engineering, vol. 20, no. 3, pp. 455–480, 2021.