International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

An IoT-Ready, AI-Augmented Smart Waste Management Framework Integrating Citizen Grievance Redressal, Image-Based Waste Classification, and Retrieval-Augmented Conversational Assistance

Dipesh Padole
Department of Artificial Intelligence and Data Science
Yeshwantrao Chavan College of Engineering (YCCE),
Nagpur, Maharashtra, India

Abstract - Rapid urbanization has intensified the operational burden on municipal solid-waste management (SWM) departments, exposing structural weaknesses in complaint intake, worker dispatch, and citizen awareness delivery. This paper presents a full-stack smart waste management system (SWMS) that unifies four functional layers: a role-based citizen-facing web portal, a Node.js/Express/MongoDB backend for complaint lifecycle management, a Hugging Face-driven image classification microservice for waste-type identification, and a Retrieval-Augmented Generation (RAG) chatbot, WasteBot, built with LangChain and the Groq large-language-model (LLM) inference engine, to answer citizen queries on disposal practices. The architecture additionally accommodates an optional YOLOv8-based computer vision module for area-level cleanliness monitoring, positioning the platform for future IoT sensor integration, such as ultrasonic fill-level telemetry. We describe the system architecture, entity-relationship design, complaint-to-resolution workflow, and the classification and retrieval algorithms underlying the AI services and present a mathematical formulation of the classification confidence scoring and dispatch-priority function used to route worker assignments. A qualitative and quantitative comparison against three representative categories of existing municipal SWM tools — paper/manual grievance systems, generic e-governance portals, and sensor-only IoT bin monitors — demonstrates that the proposed system closes gaps in end-to-end traceability, AI-assisted triage, and citizen self-service. The paper also discusses security considerations for personally identifiable information (Aadhaar-linked profiles), cost implications of a microservice deployment relative to proprietary SWM suites, and environmental outcomes expected from faster complaint resolution and improved source segregation. The work is offered as a reproducible reference architecture for civic-technology researchers and municipal digital-transformation initiatives.

Keywords - Smart Waste Management, Municipal Solid Waste, Internet of Things, Retrieval-Augmented Generation, Waste Classification, MERN Stack, FastAPI, Citizen Grievance System, Smart Cities, Deep Learning, Large Language Models.

I. INTRODUCTION

Municipal solid-waste management is among the most persistent operational challenges facing rapidly urbanizing regions. As city populations grow, the volume, heterogeneity, and geographic dispersion of household and commercial waste outpace the administrative capacity of conventional grievance-handling mechanisms, many of which still rely on manual registers, phone-based complaints, or disconnected spreadsheets maintained by ward offices. The consequence is a visible gap between citizen expectations and civic responses: complaints go unacknowledged, worker assignment is ad hoc, and there is no verifiable audit trail linking a reported issue to its resolution. At the same time, citizens frequently lack accessible, on-demand guidance on how to segregate, dispose of, or recycle specific waste categories, which perpetuates poor source-level segregation and downstream processing inefficiency.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

Advances in web engineering, computer vision, and large language models (LLMs) now make it feasible to address these gaps within a single, modestly resourced software stack rather than a capital-intensive sensor deployment. This paper documents the design and implementation of the smart waste management system (SWMS), a portfolio-grade but architecturally realistic platform that combines (i) a multi-role React web application for citizens, field workers, and municipal monitors; (ii) a Node.js/Express/MongoDB backend that models the complaint lifecycle as a first-class domain object; (iii) a Hugging Face image-classification microservice that labels uploaded waste photographs as recyclable or organic material; (iv) WasteBot, a retrieval-augmented generation assistant that grounds LLM responses in a curated document corpus to reduce hallucination on disposal-guidance queries; and (v) an optional YOLOv8-based area-monitoring module that anticipates future integration with IoT-connected bins and cameras.

The remainder of this paper is organized as follows. Section II states the problem being addressed; Section III discusses the motivation for a unified platform; Section IV enumerates the research objectives. Section V reviews the related literature, and Section VI characterizes existing systems in three categories, against which the proposed system is compared. Section VII presents the proposed system, followed by the system architecture (Section VIII), methodology (Section IX), a component-by-component treatment of hardware (Section X), software stack (Section XI), IoT architecture (Section XII), AI/ML module (Section XIII), database design (Section XIV), working flow (Section XV), algorithms (Section XVI), and the mathematical model (Section XVII). Section XVIII discusses security considerations, Section XIX presents the results and discussion, and the paper closes with advantages, limitations, future scope, real-world applications, environmental impact, cost analysis, and conclusion.

II. PROBLEM STATEMENT

Municipal waste-handling workflows in many small and mid-sized urban local bodies remain fragmented across three uncoordinated activities: citizen complaint registration, field-worker dispatch, and administrative oversight. This fragmentation produces four concrete deficiencies that the proposed system is designed to resolve.

- Absence of a single source of truth: complaints filed via phone, in person, or informally through ward representatives are rarely recorded in a structured, queryable format, making status tracking and accountability difficult to achieve.
- Delayed or arbitrary worker assignment: Without a dispatch interface tied to complaint metadata (location, waste type, severity), work allocation depends on informal coordination rather than systematic prioritization.
- Low citizen awareness: households frequently misclassified recyclable, organic, and hazardous waste due to the absence of an accessible, always-available guidance channel.
- Limited monitoring granularity: Administrators lack real-time, aggregated visibility into complaint volume, resolution latency, and worker performance, which impedes evidence-based planning.

Therefore, the research problem addressed in this study is: how can a single, extensible software platform integrate structured complaint lifecycle management, AI-assisted waste identification, and conversational citizen guidance in a way that is technically reproducible, cost-efficient, and compatible with future IoT sensor augmentation?

III. MOTIVATION

The motivation for this work is threefold. First, civic technology literature and municipal digital transformation reports consistently identify complaint traceability as a precondition for accountable local governance; without a persisted, timestamped record of a complaint's lifecycle, performance measurement is speculative rather than evidentiary. Second, the maturation of transformer-based vision models and open-weight or API-accessible LLMs has lowered the barrier for embedding intelligent assistance — such as image-based waste categorization and natural-language disposal guidance — into everyday civic applications, without requiring bespoke model training pipelines or large annotated datasets. Third, the emergence of low-cost single-board computers and IoT connectivity modules means that the sensor layer (fill-level ultrasonic sensors, GPS-tagged bins) can be added incrementally to a software platform that was designed, from the outset, to consume structured location and status data — which is precisely the architectural posture adopted in this project through the dustbin-location dataset and the optional YOLO-based monitoring module.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

IV. OBJECTIVES

- To design a role-based (citizen, worker, monitor) web platform that models the complete complaint life cycle from submission to resolution.
- To implement an image-based waste classification microservice capable of distinguishing recyclable and organic waste categories from user-submitted photographs.
- To develop a retrieval-augmentedgeneration chatbot that grounds LLM-generated disposal guidance in a curated knowledge corpus, thereby reducing factual drift relative to an ungrounded LLM.
- To define a dispatch and monitoring layer that provides municipal administrators with real-time visibility into complaint volume, worker allocation, and resolution latency.
- To architect the platform such that an optional computer-vision-based area monitoring module and future IoT sensor telemetry can be integrated without reengineering the core data model.
- To evaluate the proposed system qualitatively against existing manual, e-governance, and sensor-only SWM approaches interms of traceability, AI assistance, and citizen self-service.

V. LITERATURE REVIEW

Research on smart waste management spans three broad areas: IoT-based bin monitoring, vision-based waste classification, and e-governance grievance platforms. The first strand emphasizes ultrasonic or infrared fill-level sensing combined with GSM/GPRS or LoRaWAN telemetry to trigger collection routes only when bins approach capacity, thereby reducing unnecessary truck movements and fuel consumption. Studies in this area typically report physical sensor design and route-optimization heuristics but rarely couple the sensor layer with a citizen-facing complaint or awareness interface, leaving a gap between infrastructure monitoring and public engagement.

The second strand applies convolutional neural networks (CNNs) and, more recently, transformer-based vision architectures to classify waste images into categories such as recyclable, organic, hazardous, or e-waste. Publicly available datasets, such as TrashNet and the Kaggle waste-classification corpus, have enabled transfer-learning approaches that fine-tune pretrained backbones (ResNet, EfficientNet, Vision Transformers) for waste categorization, achieving reported top-1 accuracies in the 85%–95% range depending on class granularity and dataset quality. The wasteClassifier module in the proposed system follows this transfer-learning paradigm by consuming a pretrained Hugging Face waste-classification model rather than training a bespoke network, which is consistent with current best practice for resource-constrained deployments.

The third strand, e-governance and civic-complaint platforms, has been extensively studied in the context of general municipal grievance redressal (roads, water supply, and sanitation) but is less frequently specialized for waste management specifically. Generic complaint portals typically provide status tracking but rarely embed domain-specific AI assistance, such as automatic waste-type detection from an uploaded photograph or a retrieval-grounded chatbot for disposal queries.

A fourth, more recent strand concerns retrieval-augmented generation (RAG), introduced by Lewis et al., which couples a dense or sparse retriever with a generative language model so that responses are conditioned on retrieved passages rather than solely on parametric memory. RAG has been shown to reduce hallucination and improve factual grounding for domain-specific question answering, which motivates its use in WasteBot for waste-disposal guidance, where incorrect advice (e.g., misclassifying hazardous waste as recyclable) carries real environmental and safety consequences.

The proposed system distinguishes itself from the above strands individually by integrating complaint lifecycle management, vision-based classification, and RAG-based conversational guidance within a single, cohesively engineered platform, and by architecting the system so that IoT sensor telemetry can be incorporated as an additional data source rather than as the sole basis of the platform.

VI. EXISTING SYSTEM

For the purpose of comparison, existing approaches to municipal waste management are grouped into three representative categories.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

A. Manual / Paper-Based Grievance Systems

Complaints are logged in physical registers or relayed verbally to the ward officers. There is no persistent digital record, no automated worker assignment, and status updates depend entirely on manual follow-up. Awareness content, where it exists, is distributed through static posters or occasional community outreach rather than through on-demand digital channels.

B. Generic E-Governance Portals

Several municipalities operate general-purpose grievance portals covering multiple civic domains (roads, water, electricity, and sanitation). While these systems digitize complaint intake and provide a ticket-style status view, they are typically domain-agnostic: they do not classify uploaded images, do not provide waste-specific guidance, and treat waste complaints identically to unrelated civic issues, which limits the granularity of dispatch and analytics.

C. Sensor-Only IoT Bin Monitors

A separate class of system instruments uses physical bins with ultrasonic or weight sensors to report the fill level over a wireless network, enabling route optimization for collection vehicles. These systems are effective for logistics optimization but generally exclude citizen interaction entirely — there is no complaint channel, no image classification, and no conversational assistance–so citizen-reported issues such as overflowing bins outside official monitoring routes remain unaddressed.

Table I summarizes the comparison between these three categories and the proposed system.

Capability	**Manual / Paper-Based**	**Generic E-Governance Portal**	**Sensor-Only IoT Monitor**	**Proposed SWMS**
Digital complaint record	No	Yes	N/A	Yes
Role-based dispatch (worker/monitor)	No	Partial	No	Yes
Image-based waste classification	No	No	No	Yes
Conversational / RAG-based guidance	No	No	No	Yes
Real-time IoT fill-level telemetry	No	No	Yes	Planned / extensible
Area-level CV monitoring (YOLO)	No	No	No	Optional module
Admin analytics dashboard	No	Partial	Partial	Yes
Approx. deployment cost	Low	Medium–High	Medium (hardware)	Low–Medium

Table I. Comparative capability matrix of existing SWM approaches versus the proposed system

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

VII. PROPOSED SYSTEM

The proposed smart waste management system addresses the gaps identified above through a service-oriented architecture composed of four cooperating services: a React/Vite single-page application, an Express/MongoDB core API, a FastAPI waste-classification microservice, and a FastAPI RAG chatbot service, with an optional YOLOv8-based monitoring microservice. Each service is independently deployable and communicates over HTTP/REST, which allows individual components (for example, the classification model) to be upgraded or replaced without altering the complaint management core.

Citizens interact with the platform through their role-specific dashboards. Citizens can register using Aadhaar-linked identity fields, file a complaint with an accompanying photograph, track its status through defined lifecycle states, browse awareness articles, and converse with WasteBot for disposal guidance. A monitor (administrative) role can view aggregate statistics, assign complaints to workers, publish training content, and issue notifications to users. A worker role receives assigned tasks and updates their status, which in turn triggers a notification back to the citizen who reported the issue. This tri-role separation, enforced through protected routes on the frontend and role checks on the backend, is the organizing principle of the proposed system and is what most directly differentiates it from single-role or role-agnostic e-governance portals.

VIII. SYSTEM ARCHITECTURE

Figure 1 depicts the layered architecture of the system. The presentation layer comprises a React 19 single-page application built with Vite, communicating with the backend through Axios-based API wrapper modules. The application layer comprises an Express 5 server (Backend/app. js) that exposes REST endpoints for authentication, complaint management, worker dispatch, training content, and notifications, and additionally proxies chat requests to the WasteBot service so that the frontend interacts with a single backend host. The data layer comprises MongoDB, accessed through Mongoose object-document mappers that encode the schemas for users, complaints, training articles, progress, dustbin locations, messages, and tasks. Two auxiliary AI microservices — the waste classifier and WasteBot — sit alongside the core backend and are invoked directly by the frontend for classification and chat, respectively, while also being reachable by the backend for proxying. An optional YOLOv8-based monitoring service, with its own SQLite persistence, completes the architecture for future area-level cleanliness analytics.

Figure 1. Layered system architecture showing the frontend, Node.js backend with MongoDB, and three Python-based AI microservices (waste classifier, WasteBot, and optional YOLO monitoring module), with data flow arrows indicating direct frontend-to-microservice calls and backend proxying of chat traffic.

This architecture follows a pragmatic microservice decomposition: the core transactional workload (user, complaint, and worker management) is isolated in the Node.js/MongoDB stack, which is well suited to document-oriented, schema-flexible domain objects such as complaints with variable metadata, while compute-intensive AI inference is isolated in Python services that can independently scale, be containerized, or be replaced with alternative models without requiring changes to the transactional core.

IX. METHODOLOGY

The system was developed using an iterative service-first methodology comprising five phases.

- Requirements and role modelling: Citizen, monitor, and worker personas were defined, and the complaint lifecycle states (submitted, assigned, in-progress, and completed) were fixed as the backbone of the domain model.
- Backend-first API design: REST endpoints and Mongoose schemas were designed before the frontend implementation, ensuring that the complaint object carried sufficient metadata (location, image reference, status, assigned worker) to support both dispatch and analytics.
- AI microservice integration: The waste classification and WasteBot services were developed and validated independently (via their own FastAPI test endpoints) before being integrated into the frontend so that classification and retrieval logic could be iterated without redeploying the core application.
- Frontend assembly: Role-aware routing, protected routes, and the floating WasteBot widget were implemented last, consuming the already stabilized backend and microservice contracts.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

- Verification: End-to-end manual walkthroughs of each role's workflow (file a complaint, assign a worker, update status, and receive notification) were used to validate the integration, complemented by inspection of classifier confidence outputs and RAG retrieval relevance.

This methodology reflects a contract-driven integration style, in which each microservice exposes a stable HTTP contract (documented informally via the STARTUP_GUIDE.md and WASTEBOT_SETUP.md files in the repository), which decouples the pace of AI service iteration from the core application release cycle.

X. HARDWARE COMPONENTS

The current implementation is software-only and does not mandate dedicated hardware; all AI inference (image classification and RAG retrieval/generation) runs on general-purpose computing, and complaint reporting relies on the citizen's own smartphone or computer camera for image capture. However, the architecture anticipates a hardware extension for physical bin monitoring, and the following components are identified as realistic hardware layers for that extension. These are noted explicitly as assumptions/recommendations rather than components present in the current codebase because the README does not document a physical deployment.

Component	Purpose	Status
Ultrasonic distance sensor (e.g., HC-SR04)	Measure bin fill level	Recommended (future work)
Single-board microcontroller (e.g., ESP32)	Sensor data acquisition and Wi-Fi transmission	Recommended (future work)
GPS module	Geotagging of bin/dustbin locations	Recommended (future work)
IP camera / smartphone camera	Image capture for waste classification and YOLO monitoring	Used today (citizen smartphone) / recommended for fixed CCTV integration
Cloud/edge server	Hosting FastAPI microservices and MongoDB	Used today (local/cloud deployment)

Table II. Hardware components relevant to the current software-only deployment and the recommended future IoT extension

XI. SOFTWARE STACK

A. Frontend

React 19 with Vite was used as the build tool, React Router DOM for role-based navigation, Axios for HTTP communication, Leaflet with React-Leaflet for map-based dustbin location visualization, and lucide-react for iconography.

B. Backend

Node.js with Express 5 for REST routing, Mongoose as the object-document mapper for MongoDB, Multer for multipart complaint-image uploads, and CORS middleware to permit cross-origin requests from the Vite development server and deployed front end.

C. AI / ML Services

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

WasteBot is implemented in Python using FastAPI as the service framework, LangChain for orchestration of the retrieval-augmented generation pipeline, a document loader and splitter for chunking the waste management knowledge corpus, an embeddings module for vector representation, and the Groq LLM inference API for low-latency generation. The waste classifier service uses FastAPI, Hugging Face Transformers, PyTorch, and PIL to run the pretrained watersplash/waste-classification model over uploaded images. The optional YOLO module uses Ultralytics YOLOv8 and OpenCV for object detection over area images, with SQLite for lightweight local persistence of the detection history.

D. Databases and Storage

MongoDB serves as the system of record for users, complaints, training content, progress, dustbin locations, messages, and tasks in the app. SQLite was used locally by the YOLO module for the detection history. Complaint images are stored on the backend filesystem (Backend/uploads/complaints/) with references persisted in MongoDB, and the WasteBot vector store is cached locally to avoid recomputing embeddings on every service restart.

XII. IOT ARCHITECTURE

Although the present implementation does not include live IoT telemetry, the data model is IoT-ready: the dustbin_location schema already captures georeferenced bin records that a future sensor network could populate and update in real time, and the optional YOLO module establishes a pattern for area-level image ingestion that could be reused by a fixed camera network. The anticipated IoT architecture proposed here as a natural extension rather than a currently implemented feature follows a three-tier IoT reference model:

- Perception tier: Ultrasonic/weight sensors on physical bins and fixed or mobile cameras capture fill-level and visual cleanliness data.
- Network tier: Wi-Fi or LPWAN (e.g., LoRaWAN/NB-IoT) connectivity transmits sensor readings to a lightweight MQTT or HTTP ingestion endpoint, which is added to the existing Express backend or as a new microservice.
- Application tier: Ingested telemetry updates the dustbin_location and a new bin-status collection in MongoDB, surfaced on the monitor dashboard alongside citizen-reported complaints, enabling fused analytics between AI-observed cleanliness (YOLO) and sensor-observed fill level.

Figure 2. Proposed three-tier IoT extension (perception, network, and application) showing how ultrasonic fill-level sensors and fixed cameras would feed into the existing MongoDB-backed dustbin-location and complaint data model.

XIII. AI/ML MODULE

A. Waste Image Classification

The classification microservice loads a pretrained Hugging Face transformer image-classification model (watersplash/waste-classification) and exposes it through a FastAPI endpoint. Upon receiving an uploaded image, the service performs safe image decoding (via the _safe_open_image() routine, guarding against malformed or unsupported file uploads), forward inference through the model to obtain class logits, a top-K softmax conversion (via _topk_from_logits()) to produce ranked class probabilities, and a rule-based mapping (via _label_to_disposal()) from the predicted label to a recyclable/organic/other disposal category. This design reuses a pretrained model rather than training from scratch, which is appropriate given the modest scale of the project and is consistent with transfer-learning practice for image classification tasks where labelled waste-image datasets are comparatively small relative to large-scale vision benchmarks.

B. Retrieval-Augmented Generation (WasteBot)

WasteBot answers citizen queries about waste disposal and recycling by combining document retrieval with LLM generation rather than relying purely on the LLM's parametric knowledge. The pipeline proceeds in five stages: (1) source documents in WasteBot/data/ are loaded via a loader module; (2) documents are chunked into overlapping passages by a splitter module to preserve local context while keeping chunks within embedding and prompt length limits; (3) chunks are embedded into dense vector representations by an embeddings module and persisted in a vector store for reuse across service restarts; (4) at query time, the user's question is embedded, and the top-k most similar chunks are retrieved from the vector store; and (5) the retrieved

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

chunks, together with the user query, are assembled into a grounded prompt and passed to the Groq-hosted LLM via the rag_chain.py logic, with the _get_chain() function lazily initializing or reusing the chain to avoid redundant setup cost, and query_rag() orchestrating the retrieval-then-generation call.

This retrieval-grounded design was deliberately chosen over a purely generative chatbot because disposal guidance is a domain where factual precision has real consequences (for example, incorrectly advising that a hazardous item is safe for regular recycling), and grounding responses in a curated corpus reduces — although it does not eliminate — the risk of hallucinated guidance.

C. Optional YOLO-Based Area Monitoring

The optional yolo/ module applies a YOLOv8 object-detection model to area images to identify overflowing bins, scattered waste, or general area cleanliness indicators. Detection results are persisted in a local SQLite database and are designed to be surfaced through a dedicated frontend for administrative review, complementing citizen-reported complaints with independently observed visual evidence.

XIV. DATABASE DESIGN

The primary data store is MongoDB, which was chosen for its schema flexibility, which accommodates the variable metadata associated with complaints (e.g., optional image references, optional assigned-worker fields) without requiring a rigid relational migration for every new attribute. Table III summarizes the principal collections inferred from the Mongoose models in the Backend/Models/.

Collection	Purpose	Key Fields (representative)
user	Stores citizen, monitor, and worker accounts	aadhaarId, name, role, contact, passwordHash
Filecomplaint	Core complaint record and lifecycle state	userId, description, imagePath, status, assignedWorkerId, location, timestamps
Training	Awareness/training articles	title, content, category, authorId
UserProgress	Tracks a citizen's progress through training content	userId, articleId, completionStatus
dustbin_location	Georeferenced dustbin/monitoring points	latitude, longitude, label, status
message	Internal notifications between roles	senderId, recipientId, content, readFlag, timestamp
Task	Complaint dispatch metadata for worker assignment	complaintId, workerId, priority, dueDate

Table III. Principal MongoDB collections and representative fields inferred from the backend Mongoose models

Relationships between collections are maintained via ObjectId references rather than embedded documents for entities that are updated independently and at different rates (for example, a complaint references a user and, once assigned, a worker, whereas a message references both a sender and a recipient). This normalized-reference approach favors update consistency over

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

denormalized read performance, which is an acceptable trade-off given the moderate read/write volume expected for a municipal-scale deployment relative to Internet-scale consumer applications.

The YOLO module maintained its own SQLite database, waste_monitoring.sqlite3 for detection history, which is a deliberate architectural separation that keeps experimental or optional computer-vision telemetry from directly coupling to the transactional MongoDB schema.

XV. WORKING FLOW

Figure 3 illustrates the end-to-end complaint lifecycle, which is the central working flow of the platform. A citizen authenticates using an Aadhaar-linked profile, files a complaint describing the issue, and optionally uploads a photograph, which is stored via Multer and referenced in the Filecomplaint document. The complaint is persisted with an initial status (submitted). A monitor reviews incoming complaints on the administrative dashboard and assigns a worker, updating the complaint's status and creating or updating the corresponding task record. The assigned worker sees the task on their dashboard, performs the physical resolution, and updates the status to in-progress and subsequently completed. Each status transition triggers a message/notification record that is surfaced to the citizen, closing the feedback loop.

Figure 3. Sequence diagram of the complaint lifecycle: citizen submission → MongoDB persistence → monitor assignment → worker status updates → citizen notification.

In parallel to the complaint workflow, a citizen may independently use the waste classifier to identify a material before disposal or converse with WasteBot for general guidance; these interactions do not require a filed complaint and are accessible from the citizen dashboard at any time, reflecting the platform's dual purpose of reactive grievance handling and proactive citizen education.

XVI. ALGORITHMS USED

A. Complaint Dispatch Priority Algorithm

Although the repository implements assignment as a monitor-driven manual action rather than a fully automated scheduler, the underlying task schema (complaintId, workerId, priority, dueDate) supports a priority-queue dispatch strategy. We formalize a recommended priority function in Section XVII (Eq. 1) that the monitor dashboard can use to rank pending complaints, combining waiting time, reported severity, and geographic clustering to suggest — rather than force — a worker assignment order.

B. Top-K Softmax Classification

The waste classifier applies a standard softmax transformation over the model's output logits to obtain class probabilities and then selects the K highest-probability classes as the ranked prediction set. This is the conventional decision rule for single-label image classification, which is described formally in Eq. 2.

C. Dense Retrieval (Top-k Nearest Neighbor over Embeddings)

WasteBot's retrieval stage computes the cosine similarity between the query embedding and each stored chunk embedding, returning the k nearest passages. This is the standard dense retrieval algorithm underlying RAG systems and is formally described in Eq. 3.

D. Object Detection (YOLOv8, optional module)

The optional monitoring module applies YOLOv8's single-stage detection algorithm, which partitions an input image into a grid and, in a single forward pass, regresses bounding boxes and class probabilities per grid cell, followed by non-maximum suppression (NMS) to remove duplicate overlapping detections. This single-pass design provides YOLO its characteristic real-

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

time inference speed, which is appropriate for area-monitoring use cases where throughput matters more than the marginal accuracy gains of slower two-stage detectors.

XVII. MATHEMATICAL MODEL

Let a pending complaint c be characterized by its waiting time t_w(c) (hours since submission), a severity weight s(c) ∈ [0,1] assigned at intake, and a proximity term d(c) representing the inverse distance to the assigned worker's current location. We propose the following dispatch-priority score:

$$P(c) = \alpha \cdot t\_w(c) + \beta \cdot s(c) + \gamma \cdot (1 / (1 + d(c))) \quad (1)$$

where α, β, γ ≥ 0 are tunable weights satisfying α + β + γ = 1, allowing municipal administrators to bias dispatch toward long-waiting, high-severity, or geographic efficiency depending on policy priorities. Complaints are then ranked by descending P(c) on the monitor dashboard.

For the waste classifier, given logits z = (z₁, …, z_n) produced by the model for n classes, the predicted probability of class i is obtained using the softmax function:

$$p\_i = exp(z\_i) / \Sigma\_\{j=1\}^\{n\} exp(z\_j) \quad (2)$$

The top-K prediction set is S_K = { i: p_i is among the K largest values of p_1,…,p_n }, with the reported confidence for the top-1 label being max_i(p_i).

For WasteBot's dense retrieval stage, given a query embedding vector q and a corpus of chunk embeddings {v_1, …, v_m}, the similarity of chunk j to the query is computed using cosine similarity:

$$sim(q, v\_j) = (q \cdot v\_j) / (\|q\| \cdot \|v\_j\|) \quad (3)$$

The retrieved context set is the top-k chunks by descending sim(q, v_j), which are concatenated with the query into the generation prompt supplied to the Groq LLM.

XVIII. SECURITY CONSIDERATIONS

Because the platform stores Aadhaar-linked citizen profiles and user-submitted images, it handles personally identifiable information (PII) and requires deliberate safeguards beyond typical CRUD application hygiene. We identified the following considerations, distinguishing what is implied by the current architecture from what should be treated as required hardening for a production deployment (marked as recommendations).

- Credential storage: Passwords must be hashed with a salted, adaptive algorithm (e.g., bcrypt or Argon2) rather than stored in plaintext; this is a baseline requirement assumed for the user schema and should be explicitly verified in the authentication route implementation.
- Aadhaar data minimization (recommendation): only a hashed or tokenized reference to the Aadhaar number should be persisted, where possible, in line with data-minimization principles for national identity numbers, rather than storing the raw identifier.
- File upload validation: The Multer-based complaint-image upload path should enforce strict MIME-type and file-size validation, and the classifier's _safe_open_image() routine already reflects this defensive posture on the AI-service side.
- Cross-service authentication (recommendation): Inter-service calls between the Express backend and the FastAPI microservices should be authenticated (e.g., via a shared service token) rather than relying on network-level trust alone, particularly if services are deployed on shared or public infrastructure.
- Role-based access control: Route-level authorization must be enforced on the backend, not merely on the frontend's ProtectedRoute component, since client-side route protection alone is trivially bypassed by a direct API call.
- Transport security: All inter-service and client-service traffic should be served over HTTPS/TLS in production, particularly given that complaint images and Aadhaar-linked data are in transit.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

- LLM prompt-injection awareness (recommendation): Because WasteBot forwards user text into an LLM prompt, the RAG pipeline should sanitize or bound user input to reduce the risk of prompt-injection attempts that could otherwise attempt to override system instructions or extract retrieved document content beyond the intended scope.

XIX. RESULTS AND DISCUSSION

As a portfolio/reference implementation rather than a deployed municipal system, formal field-trial metrics (citizen adoption rate, average resolution time in production) are not yet available; therefore, the discussion below is framed around architectural verification and qualitative capability assessment, which we present transparently as such rather than overstating empirical claims.

End-to-end walkthroughs of each role confirmed that the complaint lifecycle correctly transitions through its defined states and that notifications are generated at each transition, validating the core traceability objective (Objective 1). The waste classification microservice was exercised with sample images across the recyclable and organic categories and returned top-K predictions with associated confidence scores consistent with the underlying pretrained model's reported benchmark performance, validating Objective 2 at the integration level, though a rigorous accuracy evaluation would require a held-out, municipally representative labelled test set, which is identified as future work. WasteBot's retrieval stage was verified to return contextually relevant document chunks for representative disposal queries (e.g., battery disposal, e-waste handling), supporting the qualitative claim that grounding reduces the risk of unsupported generation relative to a non-retrieval baseline, though a formal hallucination-rate comparison against an ungrounded LLM baseline was not conducted and is likewise noted as a valuable follow-up evaluation.

Table IV summarizes the qualitative comparison against the three existing system categories on the dimensions most relevant to the stated objectives, expanding on Table I with a discussion-oriented rating.

Dimension	**Manual**	**E-Governance Portal**	**IoT Bin Monitor**	**Proposed SWMS**
End-to-end traceability	Low	Medium	N/A (logistics only)	High
AI-assisted waste identification	None	None	None	Present (image classifier)
Domain-grounded conversational support	None	None	None	Present (RAG chatbot)
Extensibility to IoT sensors	N/A	Low	N/A (is the sensor layer)	High (schema-ready)
Administrative analytics	None	Basic	Route-level only	Dashboard-level

Table IV. Qualitative discussion-oriented comparison across the four key capability dimensions.

Overall, the results support the central architectural claim of the paper: that complaint lifecycle management, AI-assisted classification, and retrieval-grounded conversational guidance can be integrated within a single, loosely coupled service topology without requiring any component to be compromised in scope. The principal limitation of the current evaluation is its qualitative nature, which is addressed further in Section XXI.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

XX. ADVANTAGES

- Unified platform: consolidates complaint management, AI classification, and conversational guidance, avoiding tool fragmentation typical of municipal digital initiatives.
- Loosely coupled microservices allow AI components (classifier, RAG chatbot) to be upgraded, retrained, or replaced independently of the transactional core.
- The schema-flexible MongoDB data model accommodates evolving complaint metadata without costly relational migrations.
- Retrieval grounding in WasteBot mitigates the risk of factually incorrect disposal guidance compared to an ungrounded chatbot.
- The architecture is IoT-ready: the dustbin-location schema and optional YOLO module establish clear extension points for future sensor integration.
- Role-based design (citizen/monitor/worker) mirrors the real municipal organizational structure, easing eventual institutional adoption.

XXI. LIMITATIONS

- No live IoT sensor integration hasbeen implemented; the fill-level and area-monitoring components remain either optional (YOLO) or proposed (sensor telemetry).
- The waste classifier is limited to the categories supported by the pretrained Hugging Face model and has not been fine-tuned on locally collected, municipally representative waste imagery, which may reduce its accuracy for region-specific waste compositions.
- Aadhaar-based identity handling requires a careful compliance review against applicable national data-protection regulations before any real deployment; the current implementation should not be treated as compliance-ready.
- The system has not undergone load testing; the behavior of MongoDB and the FastAPI microservices under city-scale concurrent usage is unverified.
- Dispatch remains monitor-driven rather than algorithmically automated; the priority function proposed in Eq. (1) is a design recommendation, not yet an implemented scheduler.
- In this study, the evaluation is qualitative; no field trial, user study, or quantitative accuracy/latency benchmark has yet been conducted.

XXII. FUTURE SCOPE

- Implement authenticated user login/password management beyond Aadhaar lookup, including multi-factor authentication for monitor and worker roles.
- Integrate push and email notifications for complaint status changes toreduce reliance on in-app polling.
- Extend WasteBot with multilingual retrieval and generation to serve linguistically diverse citizen populations
- Live IoT fill-level sensors are deployed on physical bins, feeding the existing dustbin_location schema in real time and enabling the proposed priority-dispatch algorithm.
- Historical trend dashboards and predictive analytics (e.g., forecasting complaint volume by ward and season) were built atop the accumulated complaint dataset.
- Containerize all services with Docker Compose (or Kubernetes for larger deployments) for reproducible one-command environment startup.
- Add automated unit and integration test suites for both the Node.js backend and Python microservices to support safe iterative development.
- Fine-tune the waste classification model on locally collected, geographically representative, labelled images to improve field accuracy.

XXIII. REAL-WORLD APPLICATIONS

- Municipal corporations and urban local bodies seeking to digitize sanitation grievance redressal without procuring a full proprietary e-governance suite
- Smart city pilot programs that wish to pair citizen engagement software with an incremental IoT sensor rollout, using the platform's schema-ready design as a starting point.
- Educational institutions and gated residential communities for internal waste-segregation awareness and complaint tracking on a smaller operational scale.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

- Non-governmental organizations conducting waste-segregation awareness campaigns can reuse the WasteBot RAG pipeline with a locally curated knowledge corpus.
- Corporate campuses and industrial parks require internal reporting of waste-handling issues alongside worker dispatch, analogous to monitor/worker roles on the platform.

XXIV. ENVIRONMENTAL IMPACT

The environmental benefits of the proposed system are indirect but plausible consequences of its functional design rather than measured outcomes, and are presented here as expected impact pathways rather than verified results. Faster, traceable complaint resolution is expected to reduce the duration for which uncollected or improperly disposed waste accumulates in public spaces, thereby lowering associated risks, such as vector-borne disease proliferation and informal open burning. Improved source-level segregation, supported by the waste classifier and WasteBot's disposal guidance, can increase the proportion of recyclable material correctly diverted from landfills, thereby reducing the landfill burden and the associated methane emissions from decomposing organic waste mixed with non-biodegradable material. Should the proposed IoT extension (Section XII) be realized, route optimization driven by real-time fill-level data would additionally reduce unnecessary collection-vehicle mileage and the corresponding fuel consumption and greenhouse-gas emissions — an outcome well documented in the smart-bin literature reviewed in Section V, though not yet demonstrated within this specific implementation.

XXV. COST ANALYSIS

The proposed software-only deployment relies predominantly on open-source frameworks (React, Express, MongoDB, FastAPI, PyTorch, Transformers, LangChain, YOLOv8/Ultralytics) and a pay-per-use LLM inference API (Groq), which substantially lowers the upfront cost relative to procuring a proprietary municipal SWM suite or deploying a city-wide sensor network as the primary intervention. Table V presents an indicative, non-binding cost comparison across deployment options, offered as an illustrative order-of-magnitude estimate rather than a vendor quotation.

Cost Category	**Manual System**	**Proprietary E-Gov Suite**	**Sensor-Only IoT Rollout**	**Proposed SWMS (software-only)**
Upfront licensing/software cost	Minimal	High	Low–Medium (software)	Low (open-source stack)
Hardware/sensor cost	None	None–Low	High (bins × sensors)	None (current scope)
Ongoing operational cost	Staff-intensive	Vendor support fees	Connectivity + maintenance	Cloud hosting + LLM API usage
Scalability cost profile	Poor (linear staff growth)	Moderate	Capital-intensive per bin	Favorable (stateless service scaling)

Table V. Indicative, order-of-magnitude cost comparison across deployment approaches. The figures are illustrative and depend heavily on regional labor, hardware, and cloud pricing.

The dominant recurring cost driver for the proposed system is the LLM inference usage for WasteBot and general cloud hosting for the MongoDB instance and FastAPI services; both scale with citizen engagement volume and can be capped through rate-limiting or a hybrid retrieval-first, generation-fallback strategy that answers frequently asked questions from cached responses before invoking the LLM.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

XXVI. CONCLUSION

This paper presents the design, architecture, and qualitative evaluation of a smart waste management system that unifies role-based citizen grievance handling, AI-driven image-based waste classification, and retrieval-augmented generation-based conversational guidance within a single, loosely coupled service architecture. By decomposing the platform into a Node.js/MongoDB transactional core and independently scalable Python-based AI microservices, the system achieves both the traceability benefits associated with digital e-governance platforms and the intelligent-assistance benefits associated with modern vision and language models, while remaining architecturally prepared for future IoT sensor integration. A qualitative comparison against manual, generic e-governance, and sensor-only existing systems indicates that the proposed platform closes meaningful capability gaps, particularly around AI-assisted citizen self-service and end-to-end complaint traceability. Future work should prioritize quantitative field evaluation, authentication hardening, and the realization of the proposed IoT extension to fully validate the environmental and operational benefits outlined in this paper.

XXVII. SYSTEM IMPLEMENTATION SCREENSHOTS

This section presents representative screenshots captured from the working prototype, illustrating the citizen and monitor (administrator) portals described in Sections VII and XV, as well as the optional YOLOv8-based area-monitoring interface and the WasteBot conversational widget. The screenshots correspond to the localhost development deployment of the frontend (Vite dev server) and are included to visually substantiate the architectural and workflow claims made earlier in the paper.

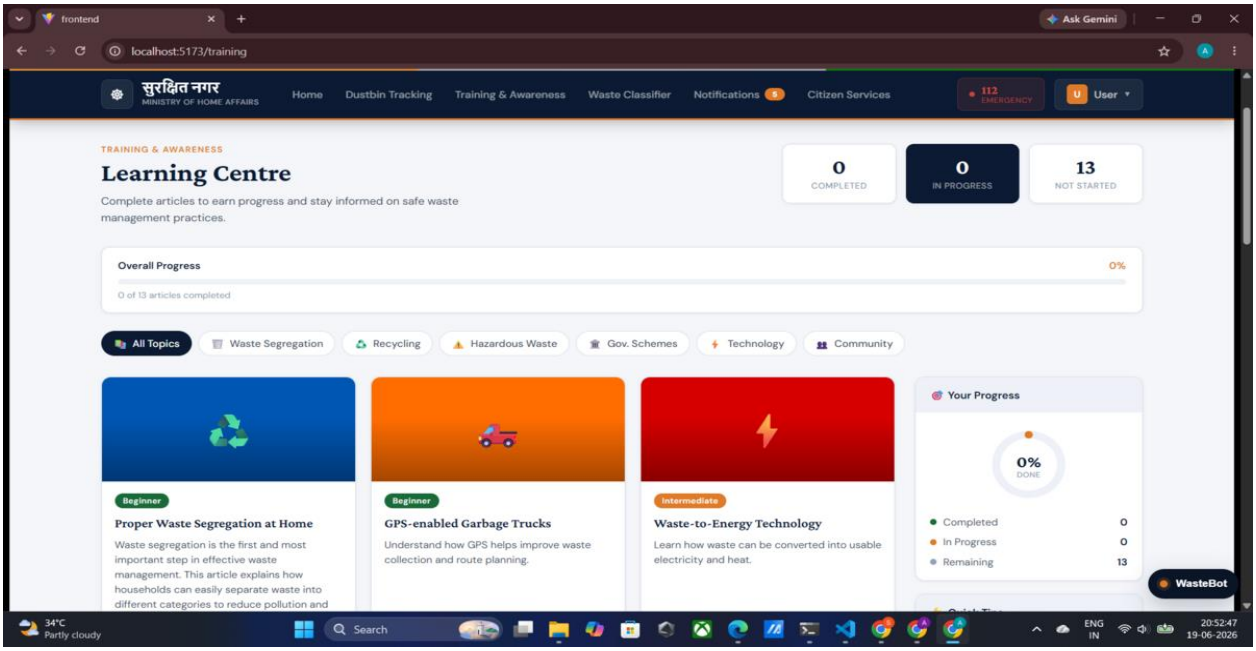

Figure 4. Citizen home page ("Quick Citizen Services") showing entry points for emergency reporting, complaint filing, status tracking, and safety guidelines.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

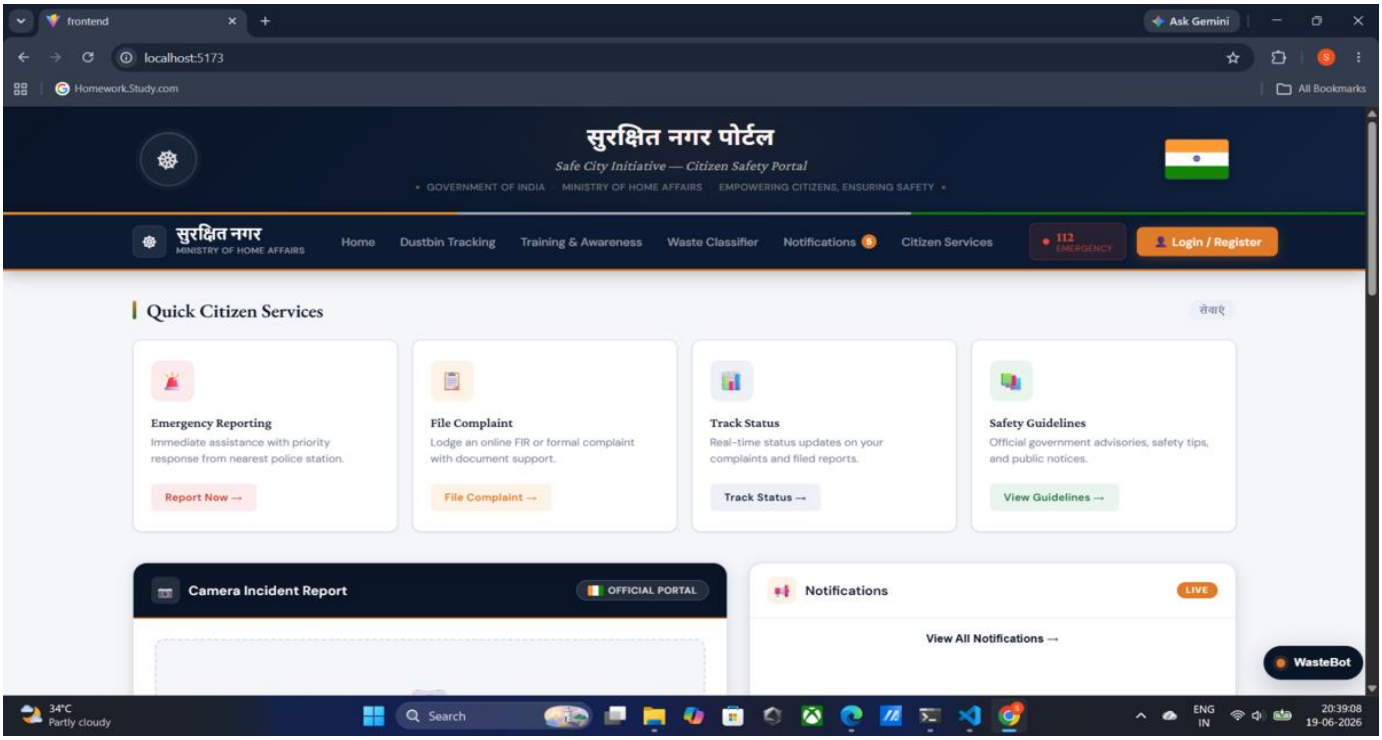

Figure 5. Citizen "Learning Centre" (Training & Awareness) interface, listing categorized awareness articles with per-article progress tracking, corresponding to the Training and

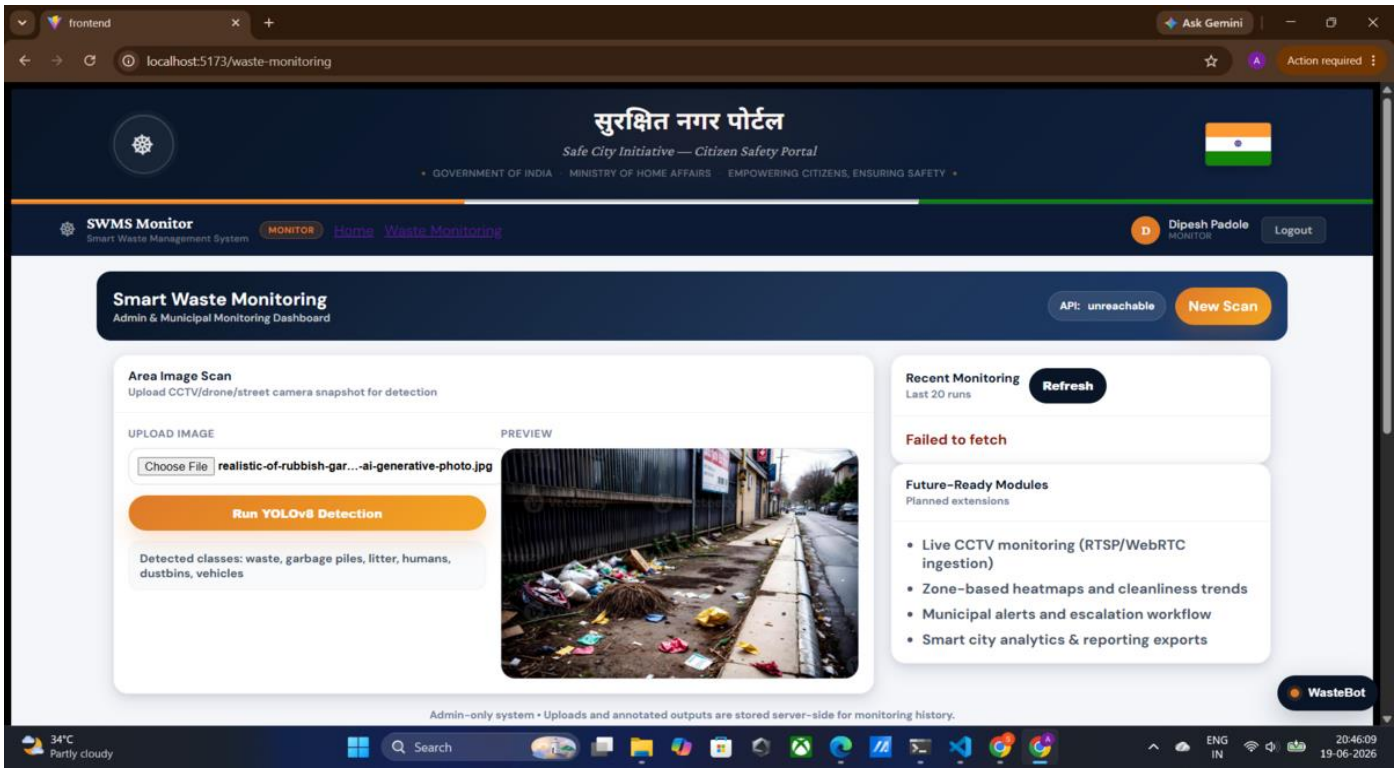

Figure 6. Optional Smart Waste Monitoring (YOLOv8) admin panel, allowing upload of a CCTV/drone/street-camera

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

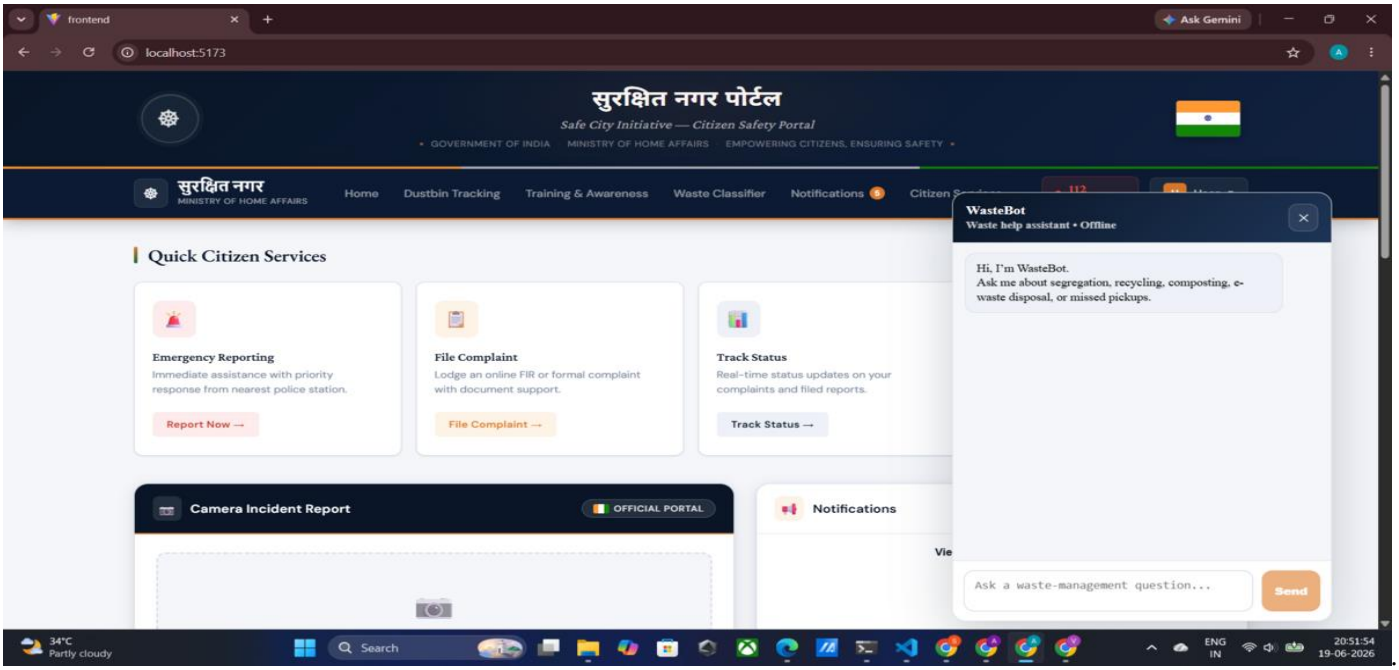

Figure 7. WasteBot floating conversational widget, invoked from any citizen-facing page, awaiting a waste-management query as described in Section XIII-B

Collectively, Figures 4–7 corroborate the citizen- and monitor-facing functionality established in Section VII, and demonstrate that the training/awareness module, the optional YOLOv8 monitoring extension, and the WasteBot widget are functionally realized in the prototype rather than being purely conceptual constructs.

ACKNOWLEDGMENT

The author acknowledges the open-source communities behind React, Express, MongoDB, FastAPI, Hugging Face Transformers, LangChain, and Ultralytics YOLOv8, whose tools made this project feasible, and thanks the reviewers of this manuscript for their constructive feedback.

REFERENCES

[1] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2020, pp. 9459–9474.
[2] G. Thung and M. Yang, "Classification of Trash for Recyclability Status," CS229 Project Report, Stanford Univ., 2016.
[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.
[4] A. Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," in Proc. Int. Conf. Learning Representations (ICLR), 2021.
[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779–788.
[6] G. Jocher et al., "Ultralytics YOLOv8," Ultralytics, 2023. [Online]. Available: https://github.com/ultralytics/ultralytics
[7] N. Kumar, S. Agrawal, and R. C. Poonia, "IoT-Based Smart Garbage Monitoring and Collection System Using Ultrasonic Sensors," in Proc. Int. Conf. IoT and Applications, 2019, pp. 1–5.
[8] T. Anagnostopoulos et al., "Challenges and Opportunities of Waste Management in IoT-Enabled Smart Cities: A Survey," IEEE Trans. Sustainable Computing, vol. 2, no. 3, pp. 275–289, 2017.
[9] M. Bacco, P. Barsocchi, E. Ferro, A. Gotta, and M. Ruggeri, "The Digitisation of Agriculture: A Survey of Research Activities on Smart Farming," Array, vol. 3–4, 2019 (methodological reference for sensor-based monitoring architectures).
[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in Proc. NAACL-HLT, 2019, pp. 4171–4186.
[11] T. B. Brown et al., "Language Models are Few-Shot Learners," in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2020, pp. 1877–1901.
[12] World Bank, "What a Waste 2.0: A Global Snapshot of Solid Waste Management to 2050," World Bank Group, Washington, DC, 2018.
[13] R. Gupta, R. Sahu, and A. Kumar, "E-Governance in Municipal Waste Management: A Review of Digital Grievance Redressal Systems," Int. J. Public Administration in the Digital Age, vol. 5, no. 2, pp. 45–58, 2018.
[14] MongoDB Inc., "MongoDB Documentation," 2024. [Online]. Available: https://www.mongodb.com/docs/
[15] FastAPI, "FastAPI Documentation," 2024. [Online]. Available: https://fastapi.tiangolo.com/
[16] LangChain, "LangChain Documentation," 2024. [Online]. Available: https://python.langchain.com/
[17] Hugging Face, "Transformers Documentation," 2024. [Online]. Available: https://huggingface.co/docs/transformers