

Smart Waste Management and Collection Optimization System Using IoT with Citizen Reporting and Dynamic Route Planning

Nair Akshay Anilkumar, Dr Manoj Singh

Roll No: SMCS2627103

^{1,2}Department of Computer Science, Sies College of Arts, Science and Commerce (Empowered Autonomous), Mumbai, India
Email: akshayfms2526103@ascs.sies.edu.in, manojks@sies.edu.in

Abstract—Municipal waste collection is still largely schedule-driven rather than demand-driven, causing overflowing bins, wasted trips and higher fuel use. This paper presents an integrated smart waste management platform that combines IoT fill-level sensing (ESP32 with an ultrasonic sensor), threshold-based collection scheduling, dynamic route optimization and a citizen reporting portal. A review of twelve published studies shows that existing systems monitor bins or plan routes once per day, while citizen complaints, scheduling and truck management remain separate. A software prototype has been implemented: a FastAPI backend with 46 REST endpoints, a MySQL database, a React administrator panel and an Expo (React Native) client for citizens and drivers, verified with five end-to-end scenarios and 23 automated tests. The sensing, scheduling and routing modules are specified as a roadmap and are not claimed as completed or field-validated; target metrics for a future field trial are defined.

Index Terms—Smart Waste Management, Internet of Things, ESP32, Fill-Level Prediction, Dynamic Route Optimization, Citizen Reporting, FastAPI.

I. INTRODUCTION

Rapid urbanization has increased the volume of municipal solid waste generated every day. Most municipalities still collect waste on predetermined schedules without knowing the actual fill level of bins, so some bins overflow before collection while others are emptied when almost empty. This wastes fuel and manpower, adds traffic congestion and harms sanitation. Citizens also lack a central channel for reporting overflowing garbage or illegal dumping, routes stay fixed even when new incidents occur, and municipal staff lack real-time decision support.

This work proposes a single platform that connects (i) IoT fill sensing, (ii) fill-based collection scheduling, (iii) dynamic re-routing triggered by sensor events and verified citizen complaints, and (iv) a municipality dashboard with driver and citizen applications. The objectives are to monitor bin fill levels in real time, collect only where needed, update routes during the day, and ultimately reduce fuel use and cost. Section II reviews related work, Section III describes the design, Section IV the implementation, Section V the problems encountered, Section VI the results, Section VII the limitations, and Section VIII concludes.

II. RELATED WORK

Twelve studies published between 2017 and 2024 were reviewed [1]–[12]: ten system papers and two surveys. Taken together, they lead to the following conclusions.

IoT monitoring is mature. Every system paper reports ultrasonic or similar fill sensing with cloud or mobile access, from multi-agent simulation [1] and household-level monitoring [2], [7] to ESP32-based municipal apps [4], [5]. Monitoring is therefore not the open problem.

Routing is optimized, but once. Dijkstra-based planning [1], route recommendation [3], knapsack loading [9] and TOPSIS ranking [11] all produce a plan before trucks leave the depot. None re-plans when a bin fills early, a complaint arrives or a truck fails, and several [1], [9], [11] are validated only in simulation.

Citizens and operations are rarely connected. Citizen involvement appears in [4] and [6] but as a separate stream, not as an input to routing, and scheduling is mostly fixed or rule-based [2], [10]. Truck and driver management is absent from all ten systems. The surveys [8] (173 studies screened from 3,732) and [12] independently report that integrated platforms combining IoT, scheduling, routing and citizen interaction remain uncommon.

Evidence quality is uneven. Reported gains such as a 47% improvement in high-priority toxic-waste collection [9] come from simulation, not field trials, and the system papers rarely describe driver workflows, audit trails or role-based access, which a real municipality needs before any optimizer can be trusted.

The gap is therefore not a missing component but a missing *connected workflow*: fill-based scheduling feeding routing, and sensor events and citizen reports triggering re-routing during collection. Fig. 1 summarizes the capability coverage, and this paper targets the fully covered last row.

	IoT monitor	Route optim.	Citizen portal	Dynamic routing	Truck mgmt.
Likotiko '17	Yes	Yes	No	No	No
Saha '23	Yes	Ltd	No	No	No
Ghahramani '22	Yes	Yes	No	Ltd	No
Paul '24	Yes	Ltd	Yes	No	No
Patil '24	Yes	No	Ltd	No	No
Pardini '20	Yes	Yes	Yes	No	No
Vishnu '21	Yes	Ltd	No	No	No
Khan '24	Yes	Yes	No	No	No
Ikram '23	Yes	No	No	No	No
Ali '24	Yes	Yes	No	No	No
Proposed	Yes	Yes	Yes	Yes	Yes

Fig. 1. Capability coverage of the reviewed system papers (surveys [8], [12] excluded).

III. SYSTEM DESIGN AND METHODOLOGY

A. System Architecture

The system has five layers (Fig. 2): smart bins that measure fill level, a communication channel to a server, a backend and database acting as the single source of truth, a scheduling and routing layer that selects bins and plans routes, and the user applications. The backend uses FastAPI with JWT-based role access for Admin, Collector and Citizen users, and MySQL 8.0 accessed through SQLAlchemy 2.0, so every user sees the same operational state.

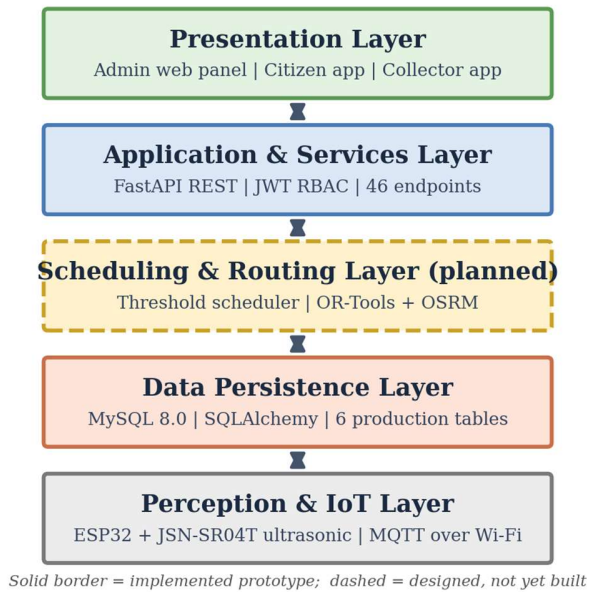


Fig. 2. Layered system architecture.

B. IoT Sensing Module

Each bin carries an ESP32, chosen for built-in Wi-Fi, low power and low cost compared with a Raspberry Pi; unlike an Arduino Uno it needs no extra Wi-Fi module. A waterproof JSN-SR04T ultrasonic sensor measures the echo time to the waste surface, and the known empty depth converts this distance into a fill percentage. Ultrasonic sensing was preferred over load cells (bulky light waste misleads weight), infrared (affected by sunlight and dark surfaces) and cameras (higher cost and compute, kept as future work). An optional DHT22 corrects the speed of sound for temperature and an optional NEO-6M records bin location.

The median of several readings rejects spurious values, and the ESP32 sleeps between readings. Data is published over MQTT, chosen over HTTP for lower power and data use and better tolerance of unstable links. The server rejects messages with an unknown bin, an impossible fill value or a missing timestamp.

C. Collection Scheduling

Once predictions are ready, the scheduler decides which bins should be collected in the next round. A bin is selected if it is expected to be nearly full (for example, above 80%) by the time of the next collection. Bins that are not worth visiting

are skipped, which saves fuel and time. The threshold can be changed by the administrator.

D. Dynamic Route Optimization

Routing is modelled as a capacitated multi-vehicle routing problem solved with Google OR-Tools, which supports truck capacity, multiple trucks and priorities. It builds a route by cheapest insertion and improves it by local search until a time limit. OSRM supplies real road distances. Dijkstra's algorithm alone only finds point-to-point paths, genetic algorithms need more tuning, exact methods become too slow beyond a few dozen stops, and nearest-neighbour is used only as a baseline.

The first plan is made before trucks leave. During collection, re-planning is triggered when a bin fills earlier than expected, an administrator verifies a citizen complaint, or a truck becomes unavailable (Fig. 3). The system re-plans only unvisited bins from each truck's current position; for a single new stop it first finds the truck that can add it with least extra travel and enough space. Each route version is logged for audit. The method will be compared with the fixed daily route and a nearest-neighbour route using distance, trips and overflow count, over identical simulated demand so that differences come from the routing policy alone.

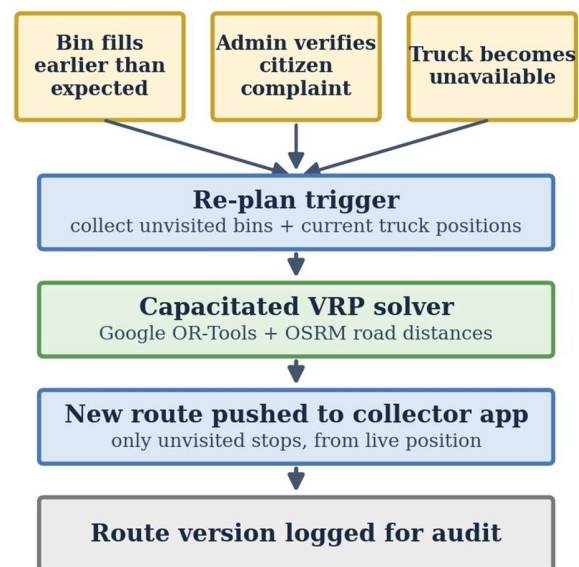


Fig. 3. Event-triggered dynamic re-routing workflow.

E. Database and Security

The production schema has six MySQL tables (users, bins, collections, alerts, citizen_reports, voucher_redemptions); telemetry and routes tables are planned. Users authenticate with JWT tokens and their role limits permitted actions, passwords are stored hashed, all inputs including sensor messages are validated, and citizen photos and locations are used only to handle complaints.

IV. IMPLEMENTATION

The backend runs FastAPI on Uvicorn with SQLAlchemy 2.0, Alembic migrations and Pydantic v2 validation, and exposes 46 REST endpoints covering authentication, administration, bin telemetry, collection dispatch, citizen reporting and analytics. Updating a bin raises an overflow alert automatically, citizen reports enforce a 5-minute cooldown, and pickup completion is an atomic transaction.

Administrator panel. The React 19 panel has six views: an executive dashboard with KPI cards and Chart.js charts, a bin fleet grid with live fill meters, a dispatch center, an alerts center with one-click resolution, user management with driver approval (Pending to Active), and reports with driver performance and CSV export.

Collector app. The Expo app gives drivers an on/off-duty toggle, a task queue scoped to the authenticated driver, Google Maps deep-link navigation and a pickup form where the collected weight is entered; submitting it completes the task, resets the bin to 0% and clears its alert. Task priority is currently set from application records, not from sensor data or route optimization.

Citizen app. Residents can locate nearby bins with live status, report overflow, illegal dumping, odour or damaged bins with text and photo, and earn eco-points (+50 per verified report) redeemable for municipal vouchers, each backed by a unique token.

V. PROBLEMS ENCOUNTERED

Different roles needed different permissions, handled by role-based access and separate workflows; the web panel and apps needed one operational state, solved with a shared backend and database. Physical bins were outside the current phase, so manual or simulated values only demonstrate workflows and no sensor value is field-validated. Likewise, route savings cannot be claimed without GPS, road data and vehicle constraints. These constraints set the rule that advanced modules are added only after the application workflow is stable, and that sensor input is validated before it can trigger alerts.

VI. RESULTS AND ANALYSIS

The prototype was evaluated through end-to-end functional scenarios, API conformance testing and 23 automated Pytest cases covering role boundaries, security and atomic transactions, all of which passed. Table I summarizes the scenarios.

TABLE I
FUNCTIONAL VALIDATION OF THE PROTOTYPE

Scenario	Observed outcome
Admin manages bins	Add, edit, monitor; overflow alert auto-raised at fill $\geq 75\%$
Driver onboarding	New drivers appear as Pending; approval activates mobile access
Citizen report	Accepted, +50 eco-points; duplicates within 5 min blocked
Voucher redemption	Points deducted atomically; 8-character token generated
Pickup completion	Driver sees own tasks, navigates, enters weight; bin resets to 0.0%

The results demonstrate feasibility of the application-layer workflow: administration, collection and citizen reporting share one data path into which future sensor and optimization outputs can plug. They do not show that routes are optimized or that overflow and fuel use are reduced; these are intended outcomes of a future field study defined in Table II.

VII. DISCUSSION AND LIMITATIONS

The prototype supports three claims and no more. First, a single role-controlled data path can serve administrators, drivers and citizens consistently. Second, the controls needed

by later modules already exist: bin updates raise alerts, pickups reset state atomically, and citizen reports carry status and a spam limit, so sensor events and complaints can later enter the same queue. Third, the design choices (ESP32, MQTT, threshold scheduling, OR-Tools) are justified against alternatives, which makes the planned evaluation reproducible.

Its limits are equally clear. No physical sensor has been deployed, so no fill value is field-validated; scheduling and routing savings are unmeasured; and the 25–30% targets in Table II are goals drawn from the design hypothesis, not results. Simulated telemetry will also under-represent real noise such as sensor drift, blocked lids and irregular dumping, so the first field data may force model and threshold changes.

VIII. CONCLUSION AND FUTURE WORK

This paper presented an integrated smart waste management platform and a verified software prototype that unifies municipal administration, driver workflows and citizen reporting, motivated by a gap identified across twelve published studies: routes are planned once and citizens, scheduling and trucks are handled separately. The prototype comprises a FastAPI backend with 46 endpoints, a MySQL database, a React portal and an Expo mobile client, verified with 23 automated tests. The IoT sensing, scheduling and dynamic routing modules are specified but remain future work.

Future work includes deploying ESP32 nodes with a secure telemetry gateway, building a cleaned telemetry store, tuning the scheduling threshold on real data, integrating OR-Tools/OSRM re-routing with live vehicle positions, adding notifications and service-level tracking to citizen reports, and running a controlled field trial against the current workflow.

REFERENCES

- [1] E. D. Likotiko, D. Nyambo, and J. Mwangoka, "Multi-agent based IoT smart waste monitoring and collection architecture," *Int. J. Comput. Sci., Eng. Inf. Technol.*, vol. 7, no. 5, pp. 1–14, 2017.
- [2] S. Saha and R. Chaki, "IoT based smart waste management system in aspect of COVID-19," *J. Open Innov.: Technol. Market Complexity*, vol. 9, no. 2, Art. 100048, 2023.
- [3] M. Ghahramani, M. Zhou, A. Molter, and F. Pilla, "IoT-based route recommendation for an intelligent waste management system," *IEEE Internet Things J.*, vol. 9, no. 14, pp. 11883–11892, 2022.
- [4] L. Paul, R. D. Mohalder, and K. M. Alam, "An IoT based smart waste management system for the municipality or city corporations," *arXiv:2411.09710*, 2024.
- [5] S. C. Patil and M. R. Gidde, "An IoT-based waste management approach for environmental sustainability," *J. Integr. Sci. Technol.*, vol. 12, no. 3, p. 769, 2024.
- [6] K. Pardini *et al.*, "A smart waste management solution geared towards citizens," *Sensors*, vol. 20, no. 8, Art. 2380, 2020.
- [7] S. Vishnu *et al.*, "IoT-enabled solid waste management in smart cities," *Smart Cities*, vol. 4, no. 3, pp. 1004–1017, 2021.
- [8] I. Sosunova and J. Porras, "IoT-enabled smart waste management systems for smart cities: A systematic review," *IEEE Access*, vol. 10, pp. 73326–73363, 2022.
- [9] S. Khan, B. Ali, A. A. K. Alharbi, S. Alotaibi, and M. Alkhatami, "Efficient IoT-assisted waste collection for urban smart cities," *Sensors*, vol. 24, no. 10, Art. 3167, 2024.
- [10] S. Thaseen Ikram, V. Mohanraj, S. Ramachandran, and A. Balakrishnan, "An intelligent waste management application using IoT and a genetic algorithm-fuzzy inference system," *Appl. Sci.*, vol. 13, no. 6, Art. 3943, 2023.
- [11] B. Ali, M. A. Javed, A. A. K. Alharbi, S. Alotaibi, and M. Alkhatami, "Internet of Things-assisted vehicle route optimization for municipal solid waste collection," *Appl. Sci.*, vol. 14, no. 1, Art. 287, 2024.

- [12] B. Esmacilian, B. Wang, K. Lewis, F. Duarte, C. Ratti, and S. Behdad, "The future of waste management in smart and sustainable cities: A review and concept paper," *Waste Manage.*, vol. 81, pp. 177–195, 2018.