IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 19, April 2026
Impact Factor: 8.2

Location-based Environmental Cleanup Management System with Automation Framework

Sasthika A, Suguna V, Mohamed Siddiq M N, Rajendra Shreedevi, S. Jayashree
Department of Applied Computing & Emerging Technologies
VELS Institute of Science Technology and Advanced Studies (VISTAS), Pallavarm, Chennai, India

Abstract: *This project presents a Location-Based Environmental Cleanup Management System integrated with an Automation Framework, developed to address the growing challenges of environmental waste management in urban and semi-urban areas. The system leverages geolocation technologies to identify, track, and manage cleanup sites in real time, enabling coordinated efforts between citizens, volunteers, and administrative authorities.*

The proposed system allows users to report pollution incidents by pinning locations on an interactive map. Administrators can assign cleanup tasks to volunteer teams, monitor progress, and receive automated notifications through a real-time communication channel powered by WebSocket technology. The automation framework streamlines repetitive workflows such as task assignment, reminder dispatch, progress tracking, and report generation.

The backend is built using Python 3.11 and the Flask micro-framework, with Flask-SocketIO enabling bidirectional real-time communication. Data persistence is achieved through SQLAlchemy ORM with a relational database backend. Results from functional testing confirm successful real-time task notifications, accurate geolocation pinning, and reliable automation execution.

Keywords: Environmental Cleanup, Location-Based Services, Automation Framework, Flask, WebSocket, SQLAlchemy, Geolocation, Waste Management, Real-Time Notification, Role-Based Access Control.

I. INTRODUCTION

Environmental pollution resulting from improper disposal of solid waste, industrial effluents, and household refuse constitutes one of the most pressing public health and ecological concerns of the contemporary era. Despite growing awareness among citizens and increasing regulatory pressure on municipal bodies, the practical execution of environmental cleanup operations remains fragmented and inefficient in many regions.

The Location-Based Environmental Cleanup Management System with Automation Framework is conceived to address this gap by providing an integrated, data-driven solution for the complete cleanup management lifecycle. The system enables citizens to report environmental hazards through a web interface by submitting geotagged complaints that are automatically plotted on a live administrative map.

At the core of the technical architecture is a Python and Flask backend. Flask-SocketIO extends this foundation with WebSocket support, enabling instant bidirectional communication. The automation framework handles routine administrative tasks programmatically: deadline reminders are dispatched to cleanup teams; unacknowledged tasks trigger escalation alerts; and upon completion, post-cleanup verification requests are sent to reporters.

II. RELATED WORK

Research in location-based service (LBS) platforms has grown substantially over the past decade, driven by the proliferation of GPS-enabled devices and cloud computing infrastructure. Several studies have proposed mobile and web-based platforms for civic issue reporting, environmental monitoring, and crowd-sourced cleanup coordination.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 19, April 2026 Impact Factor: 8.2

Kumar et al. (2021) proposed a mobile crowd-sourcing framework for urban waste detection using image classification and GPS tagging. Their system lacked a real-time notification mechanism and automated task routing, both central to the present work. Zhao and Li (2022) demonstrated IoT sensor networks for environmental quality monitoring, but required significant capital investment and was not easily adaptable to citizen-driven reporting.

Flask-based web applications have been widely adopted in research prototypes for their lightweight architecture and extensive ecosystem. The integration of Flask-SocketIO for real-time updates in cleanup management systems represents a novel contribution of the present work.

III. PROPOSED APPROACH

The proposed system enhances traditional cleanup management by integrating geolocation services, real-time WebSocket communication, and rule-based automation to provide a coordinated and dynamic operational framework.

The system consists of six main components: the Citizen Complaint Portal, the Geolocation and Map Visualization Module, the Task Assignment and Management Module, the Real-Time Communication Module, the Automation Framework Module, and the User Management and Access Control Module.

The complaint portal enables citizens to submit GPS-tagged reports with photographic evidence. The map module visualizes all active incidents on an interactive administrative dashboard. The task module allows administrators to assign cleanup work to volunteers with instant WebSocket notifications. The automation module handles reminders, escalations, and report generation programmatically.

IV. METHODOLOGY

The proposed system is designed using a three-tier Model-View-Controller (MVC) architectural pattern. The methodology is divided into the following key implementation phases.

4.1 Citizen Complaint Submission

The complaint submission form requests geolocation access from the browser. Upon user consent, the navigator.geolocation.getCurrentPosition() API returns latitude and longitude values, which are stored in the database alongside the complaint record. A SocketIO emit() call notifies all active administrator sessions of the new report in real time.

4.2 Task Assignment and Notification

The task assignment route creates a CleanupTask record in the database and emits a new_task WebSocket event to the /tasks namespace. Connected volunteer team interfaces receive this event and render the new task card without requiring a page reload, ensuring instant operational awareness.

4.3 Automation Workflow

The automation worker is a Python daemon thread that evaluates all registered rules at configurable intervals. When trigger conditions are satisfied, corresponding actions are executed: dispatching notifications, updating records, or generating summary reports.

4.4 System Workflow

1. The citizen submits a geotagged complaint through the web interface.
2. The complaint is stored and rendered as a map marker on the admin dashboard.
3. The administrator reviews the incident and assigns a cleanup task to a volunteer team.
4. The volunteer team receives an instant WebSocket notification with full task details.
5. The automation framework monitors deadlines and dispatches reminders as required.
6. Upon task completion, a verification request is sent to the original reporter.
7. Verified completions are archived and included in automated summary reports.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 19, April 2026
Impact Factor: 8.2

V. IMPLEMENTATION

5.1 Tools and Technologies

- Python 3.11 — Primary backend programming language
- Flask — Micro web framework for routing and application logic
- Flask-SocketIO — Real-time WebSocket communication
- Flask-Login — User session and authentication management
- SQLAlchemy ORM — Database abstraction layer
- HTML5 / CSS3 / JavaScript — Frontend interface and interactivity
- Geolocation API — Browser-based GPS coordinate acquisition

5.2 Backend Implementation

The Flask application is instantiated using a factory function pattern in create_app(), which registers all Blueprints and initializes extensions. The run.py entry point calls create_app(), triggers db.create_all() within the application context to initialize the database schema, and launches the SocketIO-enhanced development server on port 2105.

5.3 Real-Time Communication

Flask-SocketIO establishes persistent WebSocket connections with each authenticated client. Events are organized into namespaces corresponding to functional areas. When a state-changing action occurs on the server, a targeted emit() call broadcasts the event to relevant subscribers, eliminating polling overhead.

5.4 Automation Execution

Automation rules are defined as Python configuration objects with trigger conditions and action callables. The daemon thread evaluates all rules at each polling interval and executes any whose conditions are currently satisfied. This design allows new automation rules to be added without modifying the core application codebase.

VI. COMPARISON

Feature	Existing	Proposed
Reporting	Manual/phone	GPS web form
Task Assign	Manual	WebSocket instant
Status Track	None	Real-time
Automation	None	Rule-based
Dashboard	None	Live map
Scalability	Manual limit	Web-scale

VII. RESULT

The proposed system was successfully implemented and tested for functionality and user experience. Complaint submission with GPS geotagging operated correctly in all test cases, with coordinates accurate to within a five-metre radius under standard network conditions.

Real-time WebSocket task notifications were delivered to connected volunteer interfaces with an average latency of 187 milliseconds on a local area network, well within the threshold for perceived real-time responsiveness.

The automation framework successfully dispatched all scheduled reminders and escalation alerts across all test scenarios without manual intervention. User acceptance testing returned a mean satisfaction score of 4.3 out of 5.0 across ten participants representing all three user roles.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 19, April 2026
Impact Factor: 8.2

The administrative map dashboard rendered 500 simultaneous incident markers without perceptible performance degradation. Role-based access control correctly prevented unauthorized route access in all 12 security test cases.

VIII. CONCLUSION

This paper presented a Location-Based Environmental Cleanup Management System integrated with an Automation Framework. The system successfully addresses the fragmented, manual nature of existing environmental cleanup management by providing a unified, real-time, geolocation-enabled platform.

The implementation demonstrates effective real-time communication via WebSocket, accurate GPS-based incident reporting, and reliable automated workflow management. Results confirm improved coordination, accountability, and operational transparency compared to traditional approaches.

The system is positioned for pilot deployment in a municipal context and offers a strong foundation for future enhancements including AI-based waste classification, IoT sensor integration, and mobile application development.

IX. FUTURE SCOPE

Future developments may include the integration of machine learning models for automatic waste classification from submitted photographs, enabling intelligent incident prioritization without manual administrator intervention.

The system can be extended to a mobile platform for wider accessibility. IoT sensor integration will enable proactive environmental monitoring. Advanced analytics dashboards will provide heat maps, team productivity metrics, and historical trend analysis for data-driven policymaking.

REFERENCES

[1]. Kumar, A., Singh, R., & Sharma, P. (2021). Mobile Crowd-Sourcing for Urban Waste Detection. Journal of Environmental Informatics, 37(2), 112–124.
[2]. Zhao, L., & Li, W. (2022). IoT-Based Environmental Quality Monitoring in Smart Cities. IEEE IoT Journal, 9(5), 3341–3352.
[3]. Grinberg, M. (2018). Flask Web Development, 2nd ed. O'Reilly Media.
[4]. Copeland, R. (2010). Essential SQLAlchemy. O'Reilly Media.
[5]. Fielding, R. T. (2000). Architectural Styles and Network-based Software Architectures. Doctoral Dissertation, UC Irvine.
[6]. W3C Geolocation API Specification. (2023). https://www.w3.org/TR/geolocation/
[7]. Fette, I., & Melnikov, A. (2011). The WebSocket Protocol. RFC 6455. IETF.
[8]. Python Software Foundation. (2024). Python 3.11 Documentation. https://docs.python.org/3.11/