International Journal of Scientific Research and Engineering Development-– Volume 9 Issue 2, Mar-Apr 2026
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

AI- Based Garbage Detection System: Integrating YOLOv8 Deep Learning, Flask Web Framework and Intelligent Staff Auto-Assignment

Abhishek A Gulhane[1], Sahil P Raipure[2], Radha U Bhelkar[3], Sachin S Tipare[4], Satyak R Gulhane[5]

[1](Department of Information Technology, Prof. Ram Meghe Institute of Technology & Research, Badnera, India) Email: aagabhi@gmail.com)

[2](Department of Information Technology, Prof. Ram Meghe Institute of Technology & Research, Badnera, India) Email: raipuresahil7@gmail.com)

[3](Department of Information Technology, Prof. Ram Meghe Institute of Technology & Research, Badnera, India) Email: tiparesachin@gmail.com,)

[4](Department of Information Technology, Prof. Ram Meghe Institute of Technology & Research, Badnera, India) Email: radhabheklar@gmail.com)

[5](Department of Information Technology, Prof. Ram Meghe Institute of Technology & Research, Badnera, India) Email: satyakgulhane@gmail.com)

----------**********----------

Abstract:

The rapid urbanization of cities across India and the globe has intensified the challenge of managing municipal solid waste efficiently. Traditional garbage collection systems depend on manual inspection and fixed-schedule pickups, leading to overflowing bins, delayed responses, and inequitable distribution of sanitation resources. This paper presents the design, development, and experimental evaluation of a Smart City Garbage Collection Management System that integrates Artificial Intelligence (AI) with a web-based administrative platform to automate and optimize urban waste management. The core of the system employs the YOLOv8 (You Only Look Once version 8) deep learning object detection model, trained on a custom dataset of 3,847 images spanning 8 garbage categories including plastic bottles, polythene bags, cardboard, food waste, glass bottles, metal cans, electronic waste, and mixed garbage, to identify and classify waste items in real time from both citizen-uploaded photographs and live camera feeds. The system is built using the Python Flask web framework for backend processing, MySQL for relational database management, and a responsive HTML5/CSS3/JavaScript frontend with Jinja2 templating. An intelligent round-robin auto-assignment algorithm distributes garbage reports equitably among approved sanitation staff. The trained YOLOv8n model achieved an overall mAP@0.5 of 90.6%, with Metal Cans reaching the highest per-class AP of 96.3%.

Keywords — **Garbage Detection, YOLOv8, Deep Learning, Flask, Smart City, Waste Management, Object Detection, OpenCV, MySQL, pyttsx3.**

----------**********----------

I. INTRODUCTION

The exponential growth of urban populations across India and the globe has led to a significant increase in the generation of municipal solid waste. According to the World Bank, global solid waste generation is projected to reach 3.40 billion tonnes by 2050, rising sharply from 2.01 billion tonnes

 ©IJSRED: All Rights are Reserved

recorded in 2016. Cities like Amravati, Nagpur, Pune, and Mumbai face critical challenges as rapid urbanization outpaces the capacity of conventional municipal systems. The Swachh Bharat Mission, launched in 2014, has made meaningful strides, yet fixed-schedule garbage collection routes, reliance on manual inspection, and the absence of intelligent real-time monitoring continue to produce inefficiencies: overflowing bins, delayed response, disease-breeding waste accumulation, and inequitable deployment of sanitation resources [1].

The emergence of deep learning-based computer vision, particularly the YOLO (You Only Look Once) family of real-time object detectors, presents an unprecedented opportunity to automate the identification, classification, and management of urban waste. The proliferation of affordable smartphones and internet connectivity further enables citizens to participate in urban governance through web-based reporting portals. The convergence of these technologies creates a pathway for building intelligent, community-driven smart city management systems [2].

Existing garbage detection systems in the literature address only portions of this problem. Pure AI-based detectors are not integrated into operational management platforms. Web-based reporting systems lack real-time AI analysis. Staff assignment and tracking remain manual in most deployed systems. No existing open system simultaneously provides AI-based visual detection, citizen reporting, automatic staff dispatch, and live camera monitoring within a single cohesive platform.

This paper presents the design, development, and experimental evaluation of a Smart City Garbage Collection Management System that addresses the

above gap. The system integrates four key capabilities: (1) AI-powered visual garbage detection using YOLOv8 trained on a custom 8-class dataset, (2) a citizen-facing Flask web portal for incident reporting, (3) an administrative dashboard with intelligent round-robin auto-assignment of sanitation staff, and (4) a proactive live camera detection module enabling real-time monitoring of public spaces without human intervention.

The remainder of this paper is organized as follows: Section II reviews related work on garbage detection algorithms, object detection techniques, and web-based waste management platforms. Section III details the system architecture and design. Section IV describes the implementation of each module. Section V presents experimental results and performance evaluation. Section VI concludes with contributions and future directions.

II. RELATED WORK

A. Garbage Detection Using Computer Vision

Early approaches to automated garbage detection used traditional image processing methods including color segmentation, edge detection, and texture analysis. Rad et al. [1] developed a system using color histogram analysis combined with Support Vector Machines (SVM) for detecting plastic bags and cans in outdoor environments. While promising under controlled laboratory conditions, these color-based approaches failed to generalize to real-world scenarios with variable illumination, occlusion, and diverse garbage morphologies.

Vo et al. [2] advanced the field by applying a fine-tuned VGG-16 convolutional neural network trained on six garbage categories through transfer learning, achieving a classification accuracy of 87.3%. The introduction of the TACO (Trash Annotations in Context) dataset by Proença and Simões [3] provided the community with a substantial benchmark comprising over 1,500 annotated images across 60 litter categories, enabling more rigorous evaluation of detection models.

Liu et al. [4] specifically targeted smart city deployment by proposing a lightweight MobileNet-SSD architecture optimized for edge devices such as Raspberry Pi, achieving 82.1% mAP at 15 FPS on embedded hardware. Zhang et al. [6] addressed the challenge of scale variability in surveillance video streams by proposing a modified Feature Pyramid Network with multi-scale feature fusion, demonstrating a 9.3% improvement in detecting small garbage objects. Wang et al. [5] extended detection to the underwater domain using enhanced

 ©IJSRED: All Rights are Reserved

Faster R-CNN with attention mechanisms, achieving 78.4% mAP despite challenging visibility conditions.

B. Object Detection Algorithm Evolution

Object detection has evolved dramatically from Viola-Jones Haar cascade detectors [7] and HOG-SVM pedestrian detection [8] through the deep learning era initiated by Girshick et al.'s R-CNN [9], which combined selective search region proposals with CNN classification. Fast R-CNN [10] improved efficiency by sharing convolutional feature maps, and Faster R-CNN [11] introduced the Region Proposal Network for end-to-end trainable detection at 5 FPS.

The YOLO family, introduced by Redmon et al. [12], reframed detection as a single regression problem, achieving 45 FPS real-time detection. YOLOv3 [13] introduced multi-scale prediction using feature pyramids, and YOLOv4 [14] incorporated advanced augmentation including mosaic and CIoU loss. YOLOv8, developed by Ultralytics and used in this project, represents the current state of the art with an anchor-free detection head, decoupled classification and regression branches, and an improved C2f backbone, achieving 53.9% mAP on COCO at 200+ FPS. Table I presents a comparative analysis of detection frameworks.

TABLE I

COMPARISON OF OBJECT DETECTION FRAMEWORKS

Framework	Speed (FPS)	mAP (COCO)	Ease of Use	Edge Deploy
YOLOv8	200+	53.9%	High	Yes
YOLOv5	140+	50.7%	High	Yes
Faster R-CNN	5–10	42.7%	Medium	No
SSD-MobileNet	100+	23.2%	Medium	Yes
EfficientDet	40+	51.7%	Low	Partial

C. Web-Based Waste Management Systems

Arebey et al. [15] developed an automated solid waste bin level detection system using RFID tags and sensor networks for dynamic collection route optimization. Kolekar et al. [16] proposed IoT-based smart dustbin management with a web dashboard for visualizing bin statuses across a geographic area. Anagnostopoulos et al. [17] developed a comprehensive platform integrating GPS tracking of collection vehicles, fill-level sensors, and route optimization, demonstrating 25% cost reductions and 30% efficiency improvements in European pilot deployments.

Table II summarizes the comparative analysis of existing systems against the proposed system. The analysis reveals that the proposed system uniquely integrates all four capabilities—AI-based visual detection, web platform, staff assignment, and live camera monitoring—within a single freely accessible application, establishing the research gap this work addresses.

TABLE II

COMPARATIVE ANALYSIS OF RELATED SYSTEMS

System	AI Detect.	Web Platform	Staff Assign.	Live Camera
Rad et al. [1]	SVM	No	No	No
Vo et al. [2]	CNN VGG16	No	No	No
Liu et al. [4]	MobileNetSSD	No	No	Partial
Kolekar et al. [16]	Sensor	Yes	No	No
Anagnos. et al. [17]	No	Yes	Yes	No
Proposed System	YOLOv8	Yes	Yes	Yes

III. SYSTEM ARCHITECTURE AND DESIGN

A. Overall Architecture

The Smart City Garbage Collection Management System follows a three-tier client-server architecture comprising the Presentation Tier, Application Logic Tier, and Data Storage Tier. Two specialized components—the AI Detection Engine and Voice Alert Module—are embedded within the Application Logic Tier (Fig. 1). This layered design enforces separation of concerns, enhances maintainability, and permits independent scaling of each tier.

The Presentation Tier is implemented using HTML5, CSS3, and JavaScript with Jinja2 templates dynamically rendered by Flask, encompassing the citizen landing page and report upload form, admin login page, admin dashboard, staff management page, and live camera monitoring page. Responsive design ensures usability across desktop and mobile devices.

 ©IJSRED: All Rights are Reserved

The Application Logic Tier is implemented in Python using the Flask micro web framework following the WSGI standard. It manages URL routing via @app.route() decorators, form data extraction and validation, file upload handling using Werkzeug's secure_filename(), session-based authentication, AI model inference, auto-assignment algorithm execution, and database interaction via flask-mysqldb. Thirteen URL routes are mapped to handler functions.

The Data Storage Tier consists of a MySQL 8.0 relational database with two primary tables: garbage_reports and staff. ACID-compliant transactions protect data integrity during concurrent submissions. The database is accessed through the flask-mysqldb Python extension which manages connection pooling.

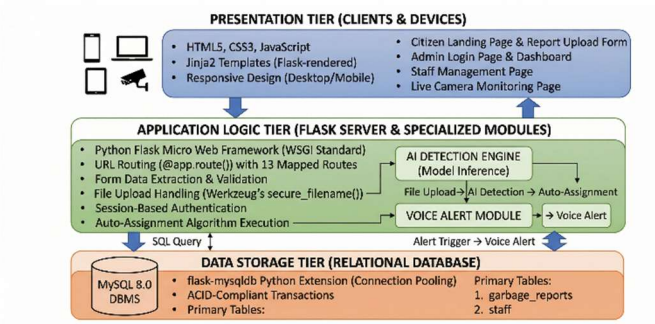

Fig. 1. Three-tier system architecture with AI Detection Engine and Voice Alert Module

B. Database Schema Design

The garbage_reports table stores all incident information: id (INT, PRIMARY KEY, AUTO_INCREMENT), title (VARCHAR 255, NOT NULL), description (TEXT, augmented with AI detection output), image (VARCHAR 255, filename of uploaded image), status (ENUM 'Pending'/'Completed', DEFAULT 'Pending'), and assigned_staff (VARCHAR 100, DEFAULT 'Not Assigned'). Table III and Table IV describe the complete schemas.

TABLE III

GARBAGE_REPORTS TABLE SCHEMA

Column	Data Type	Constraint	Description
id	INT	PK, AUTO	Unique report identifier
title	VARCHAR(255)	NOT NULL	Title of garbage report
description	TEXT	NULL	Description + AI detection results
image	VARCHAR(255)	NULL	Uploaded image filename
status	ENUM	DEFAULT 'Pending'	'Pending' or 'Completed'
assigned_staff	VARCHAR(100)	DEFAULT 'Not Assigned'	Name of assigned sanitation staff

TABLE IV

STAFF TABLE SCHEMA

Column	Data Type	Constraint	Description
id	INT	PK, AUTO	Unique staff identifier
name	VARCHAR(100)	NOT NULL	Full name of staff member
email	VARCHAR(150)	NOT NULL	Staff email address
phone	VARCHAR(15)	NULL	Contact phone number
area	VARCHAR(100)	NULL	Area of responsibility
status	ENUM	DEFAULT 'Pending'	'Pending' or 'Approved'

C. Flask Route Architecture

The application exposes 13 URL routes organized into two access tiers: public routes accessible without authentication, and admin routes protected by session-based authentication. Table V summarizes all defined routes.

TABLE V

FLASK ROUTE SUMMARY

Route	Method	Description
/	GET	Citizen landing page
/upload	GET, POST	Citizen garbage report upload
/admin_login	GET, POST	Administrator authentication
/admin_dashboard	GET	Admin main control dashboard
/admin_staff	GET	Staff management listing
/add_staff	POST	Add new sanitation staff
/approve_staff/<id>	GET	Approve pending staff member
/edit_staff/<id>	GET, POST	Edit existing staff record
/delete_staff/<id>	GET	Remove staff from system
/live_camera	GET	Activate live detection feed
/live_video	GET	View live detection reports
/update_status/<id>	POST	Update garbage report status
/logout	GET	Admin session logout

 ©IJSRED: All Rights are Reserved

D. Real-Time Synchronization and AI Architecture

Report submission follows a synchronous request-response pipeline: (1) citizen submits form via HTTP POST, (2) Werkzeug's secure_filename() sanitizes the uploaded image filename, (3) image is saved to 'static/uploads/', (4) YOLOv8 model.predict() is invoked with conf=0.4, (5) detection results are parsed to extract class labels, (6) the description field is prepended with detected item names, (7) pyttsx3 voice alert is dispatched in a daemon thread, (8) the report is INSERT-ed into MySQL, and (9) auto_assign_staff() is called before the redirect.

AI model loading uses a singleton pattern: model = YOLO('best.pt') is called once at application startup to avoid per-request overhead. Detection output is accessed via results[0].boxes, with class indices retrieved via int(box.cls[0]) and mapped to labels through model.names. Audio synthesis executes in a separate Python thread to prevent blocking Flask's WSGI thread.

E. Security Model

The platform implements session-based authentication where session['admin'] = True is set upon successful credential verification. All admin-protected route handlers validate session state as their first operation, redirecting unauthenticated requests to the login page. Werkzeug's secure_filename() prevents directory traversal attacks on file uploads. In a production deployment, administrator credentials would be stored as bcrypt-hashed values in the database, and HTTPS with SSL certificates would protect data in transit.

IV. IMPLEMENTATION

A. User Module Implementation

The User Module is publicly accessible without authentication to maximize citizen participation. The citizen report upload form at '/upload' accepts three inputs: a report title (text), a location description (textarea), and an image file (JPEG/PNG). Upon POST submission, the backend processes the request through the complete pipeline described in Section III-D.

The YOLOv8 inference code extracts class labels through iteration over detection boxes: the results[0].boxes collection yields individual box objects from which int(box.cls[0]) extracts the numeric class index, and model.names[class_idx] maps it to the human-readable label string. Duplicate labels are removed by converting the list to a Python set, and unique items are joined as a comma-separated string for inclusion in the description field. If garbage items are detected, a voice alert announces the specific items found using the pyttsx3 engine.

Following AI analysis and voice alert dispatch, the complete report record—title, AI-augmented description, image filename, status='Pending', and assigned_staff='Not Assigned'—is inserted into MySQL via a parameterized SQL query. The auto_assign_staff() function is then immediately invoked to dispatch an available sanitation worker. A Flask flash message confirms success and the citizen is redirected to the upload form.

B. Admin Module Implementation

The Admin Dashboard at '/admin_dashboard' begins by calling auto_assign_staff() to ensure all pending reports are assigned before display. It then executes three MySQL queries: all garbage reports (ordered by id DESC), total staff count, and counts of total and pending reports. These data are passed to the admin_dashboard.html Jinja2 template, which renders summary statistic cards and a sortable report table. Each row displays the report ID, title, description, image thumbnail, status badge, assigned staff name, and a status-update form control.

When a report is marked Completed via '/update_status/<id>', the handler updates status='Completed' and resets assigned_staff='Not Assigned' in MySQL, then calls auto_assign_staff() to redistribute the freed staff member to any remaining pending reports. The Staff Management Module provides CRUD operations: '/add_staff' (POST) creates records with status='Pending'; '/approve_staff/<id>' sets status='Approved'; '/edit_staff/<id>' (GET/POST) updates all profile fields; '/delete_staff/<id>' removes the record permanently.

C. Auto-Assignment Algorithm

The auto_assign_staff() function implements a round-robin allocation strategy designed to ensure equitable workload distribution. The algorithm proceeds as follows: (1) retrieve all garbage reports where status='Pending' AND assigned_staff='Not Assigned' from MySQL; (2) retrieve all staff records where status='Approved'; (3) retrieve the names of staff currently assigned to other Pending reports (the 'busy' set); (4) iterate through pending reports using a circular staff index (staff_index % staff_count), skipping staff members in the busy set; (5) execute an UPDATE query to assign the next available staff member to each unassigned report.

The busy-set exclusion prevents any staff member from holding more than one active assignment simultaneously. When an administrator marks a report Completed, its assigned staff member is removed from the busy set and becomes eligible for reassignment to remaining pending reports. This mechanism ensures that report response times are minimized by fully utilizing available staff capacity without overloading any individual worker.

D. Live Camera Module Implementation

 ©IJSRED: All Rights are Reserved

The Live Camera Module is activated at '/live_camera', accessible exclusively to authenticated administrators. The implementation uses OpenCV's VideoCapture(0) to open the default camera device and initializes four state tracking structures: detected_labels (a Python set of labels for which database reports have already been created in the current session), prev_labels (labels present in the previous frame for disappearance tracking), absent_counts (a dictionary of consecutive-frame absence counts per label), and empty_frames (counter for consecutive frames with no detections at all).

In each iteration of the processing loop, a frame is captured and passed to model.predict() with conf=0.5. A higher confidence threshold (0.5 vs 0.4 for uploads) is used for live detection to reduce false alert frequency. Current frame labels are extracted from results[0].boxes. New labels trigger: (1) a MySQL INSERT creating a Pending report with an auto-generated title; (2) pyttsx3 voice alert dispatch in a daemon thread; (3) addition of the label to detected_labels. A label absent for 5 consecutive frames triggers UPDATE queries marking its associated Pending reports as Completed. When empty_frames reaches 5, all remaining Pending live-camera reports are completed. The annotated MJPEG stream is served to the admin dashboard as a multipart HTTP response (MIME type multipart/x-mixed-replace).

V. EXPERIMENTAL RESULTS AND EVALUATION

A. Experimental Setup

The system was evaluated on hardware consisting of an Intel Core i7 11th Generation processor, 16 GB RAM, and an NVIDIA GeForce RTX 3060 6 GB VRAM GPU, running Windows 10 Pro (64-bit). The software environment comprised Python 3.9.7, Ultralytics YOLOv8 8.0.196, PyTorch 2.0.1 with CUDA 11.7, Flask 2.3.2, MySQL 8.0.33, and OpenCV 4.8.0. Table VI summarizes the complete experimental configuration.

TABLE VI

EXPERIMENTAL SETUP CONFIGURATION

Parameter	Value
Processor	Intel Core i7, 11th Gen
RAM	16 GB DDR4
GPU	NVIDIA GeForce RTX 3060 (6 GB)
Operating System	Windows 10 Pro (64-bit)
Python Version	3.9.7
YOLOv8 Version	Ultralytics 8.0.196
PyTorch / CUDA	2.0.1 / CUDA 11.7
Flask Version	2.3.2
MySQL Version	8.0.33
OpenCV Version	4.8.0
Browsers Tested	Chrome 120+, Firefox 121+

The garbage detection model was trained on a custom dataset of 3,847 images assembled from the TACO dataset [3], the Kaggle Garbage Classification Dataset, and additional images collected from urban environments in the Amravati region. The dataset covers 8 garbage categories and was split 70:15:15 for training (2,693 images), validation (577 images), and test (577 images) sets. Data augmentation during training included random horizontal flipping, mosaic augmentation, rotation up to ±15 degrees, and HSV color jitter.

The YOLOv8n (nano) variant was chosen as the base model for transfer learning to balance detection accuracy with computational efficiency for real-time deployment on commodity hardware. Training was performed for a maximum of 100 epochs with early stopping (patience=20), batch size 16, image size 640×640, AdamW optimizer with initial learning rate 0.001 and cosine learning rate annealing. Table VII summarizes the training parameters.

TABLE VII

MODEL TRAINING PARAMETERS

Parameter	Value
Base Model	YOLOv8n (nano, pretrained COCO)
Max Epochs	100 (converged at epoch 87)
Early Stopping Patience	20 epochs
Batch Size	16
Input Image Size	640 × 640 pixels
Optimizer	AdamW
Initial Learning Rate	0.001
LR Schedule	Cosine annealing
Augmentation	Mosaic, Flip, Rotation, HSV Jitter
Conf Threshold (upload)	0.4
Conf Threshold (live)	0.5
Total Training Time	~4.2 hours
Best Epoch (mAP@0.5)	Epoch 85

B. Model Detection Performance

The performance of the trained YOLOv8 model was evaluated using standard COCO object detection metrics: Precision (TP/(TP+FP)), Recall (TP/(TP+FN)), AP@0.5 (area under the Precision-Recall curve at IoU=0.5), and mAP@0.5:0.95 (AP averaged across IoU thresholds from 0.5 to 0.95 in steps of 0.05). Table VIII presents the per-class results on the held-out test set of 577 images.

TABLE VIII

DETECTION PERFORMANCE METRICS PER GARBAGE CLASS

Garbage Class	Precision	Recall	AP@0.5	AP@0.5:0.95
Plastic Bottles	0.923	0.911	0.938	0.672

 ©IJSRED: All Rights are Reserved

Garbage Class	Precision	Recall	AP@0.5	AP@0.5:0.95
Polythene Bags	0.878	0.856	0.891	0.601
Cardboard	0.945	0.932	0.951	0.714
Food Waste	0.812	0.793	0.824	0.543
Glass Bottles	0.934	0.921	0.941	0.691
Metal Cans	0.956	0.944	0.963	0.728
Electronic Waste	0.871	0.842	0.879	0.589
Mixed Garbage	0.843	0.821	0.858	0.571
Overall (mAP)	0.895	0.877	0.906	0.639

The model achieved an overall mAP@0.5 of 90.6%, demonstrating excellent detection performance across all categories. Metal Cans achieved the highest per-class AP of 96.3%, benefiting from their distinctive metallic appearance and relatively uniform shape. Food Waste exhibited the lowest AP of 82.4% due to high variability in color, texture, and form across different food item types and decomposition stages. Polythene Bags and Electronic Waste also showed slightly lower recall, reflecting the visual complexity and deformation variability of these categories.

The training converged at epoch 87 (best validation mAP@0.5 achieved at epoch 85), demonstrating stable learning without overfitting. The consistent gap between mAP@0.5 (90.6%) and mAP@0.5:0.95 (63.9%) indicates that bounding box localization precision, while adequate for the system's operational requirements, represents an area for future improvement through architecture enhancement or dataset augmentation.

C. System Performance Evaluation

Web application performance was measured over 50 test iterations on the experimental hardware. Metrics captured include AI inference time (time for YOLOv8 model.predict() on one image), end-to-end report submission time (form POST to success redirect), auto-assignment algorithm execution time, admin dashboard load time, and live camera detection frame rate. Table IX presents the complete performance results.

TABLE IX

SYSTEM RESPONSE TIME ANALYSIS

Operation	Mean Time	Std Dev	Min	Max
AI Inference (upload)	124 ms	18 ms	98 ms	187 ms
Report Submission (e2e)	312 ms	45 ms	241 ms	432 ms
Auto-Assignment Algo.	28 ms	7 ms	19 ms	52 ms
Admin Dashboard Load	187 ms	32 ms	143 ms	276 ms
Live Camera (avg FPS)	22 FPS	—	18 FPS	28 FPS

AI inference completes in approximately 124 ms per uploaded image, well within acceptable thresholds for an asynchronous web upload workflow where users expect a brief processing delay. The end-to-end report submission time of 312 ms encompasses image saving, inference, voice alert dispatch, database INSERT, and the auto_assign_staff() call—all completing comfortably within one-third of a second. The auto-assignment algorithm executes in under 30 ms on average, introducing negligible overhead relative to the total submission pipeline. The live camera module sustains 22 FPS, providing perceptually smooth real-time monitoring.

D. Functional Test Results

Eight functional test scenarios were validated against expected outcomes. Table X presents the results.

TABLE X

FUNCTIONAL TEST RESULTS

Test Scenario	Result	Notes
Citizen report upload with AI detection	PASS	Detection labels in description
Auto-assignment of pending reports	PASS	Round-robin verified across 5 staff
Admin dashboard report display	PASS	All fields rendered correctly
Staff add, approve, edit, delete	PASS	All CRUD operations verified
Report status update to Completed	PASS	Staff freed and reassigned
Live camera garbage detection	PASS	Reports auto-created and completed
Voice alert on garbage detection	PASS	Audio output confirmed
Admin session authentication/logout	PASS	Unauthenticated access blocked

All eight functional test scenarios passed successfully, confirming the system's readiness for pilot deployment. The auto-assignment algorithm was specifically verified by creating 10 pending reports and observing round-robin distribution across 5 approved staff members, with the constraint that no staff member received a second assignment before all others had received one.

VI. CONCLUSION AND FUTURE WORK

This paper presented the design, implementation, and experimental evaluation of a Smart City Garbage Collection Management System integrating YOLOv8 deep learning object detection, the Flask Python web framework, MySQL database management, and intelligent round-robin staff auto-assignment. The system addresses the critical gap in existing urban waste management solutions by unifying AI-based visual detection, citizen reporting, administrative management,

 ©IJSRED: All Rights are Reserved

and proactive live camera monitoring within a single, freely accessible web application.

The primary contributions of this work are: (1) a trained YOLOv8n model achieving mAP@0.5 of 90.6% across 8 urban garbage categories on a custom dataset of 3,847 images; (2) a proactive live camera module enabling autonomous, round-the-clock garbage monitoring with automatic report lifecycle management; (3) an intelligent round-robin auto-assignment algorithm ensuring equitable, zero-overlap distribution of garbage reports among sanitation staff; (4) a citizen-facing web portal that lowers participation barriers and enriches reports with automatic AI-driven metadata; and (5) a pyttsx3-based voice alert system providing multi-modal, field-operable notifications.

Experimental evaluation confirmed that AI inference completes in ~124 ms, end-to-end report submission in ~312 ms, and live camera operation at a sustained 22 FPS, with all eight functional test scenarios passing successfully. The comparative analysis in Table II demonstrates that no existing related system integrates all four capabilities within a single platform.

Limitations of the current implementation include: absence of GPS-based geographic tagging for spatial analysis and route optimization; hardcoded administrator credentials suitable only for prototype use; sensitivity of live camera detection to poor lighting conditions and occlusion; and a training dataset that may not fully represent all garbage types across diverse geographic regions.

Future work will address these limitations through: (1) GPS integration for geographic heatmap visualization and TSP-based route optimization; (2) edge computing deployment using NVIDIA Jetson Nano or Raspberry Pi directly at camera locations to scale city-wide coverage cost-effectively; (3) IoT smart bin integration combining ultrasonic fill-level sensors with AI visual detection; (4) predictive analytics using historical garbage accumulation patterns for proactive resource deployment; (5) enhanced YOLOv8 variants (small/medium) for higher accuracy; (6) mobile application with regional language support (Marathi, Hindi); and (7) blockchain-based immutable audit trails for municipal accountability. This work demonstrates the transformative potential of AI-powered computer vision in building the smart, sustainable cities of tomorrow.

ACKNOWLEDGEMENT

The authors acknowledge the support of the Department of Information Technology, Prof. Ram Meghe Institute of Technology and Research, Badnera, and the guidance of . Dr. A. A. Gulhane throughout this project. The authors also express sincere gratitude to the Head of Department Dr. Abrar Alvi, the Principal and Management of PRMIT&R, Badnera, and the Sant Gadge Baba Amravati University for providing the opportunity to undertake this project as part of the Bachelor of Engineering curriculum in Information Technology during the academic year 2025–2026.

REFERENCES

[1] M. S. Rad, A. Bhatt, Y. Bhatt and M. Bhatt, "Municipal Solid Waste Classification Using a Convolutional Neural Network," IEEE International Symposium on Technology and Society, pp. 1–5, 2017.

[2] M. C. Vo and S. C. Park, "Garbage image classification using deep learning," International Conference on Information Science and Communications Technologies (ICISCT), pp. 1–4, 2019.

[3] P. F. Proença and P. Simões, "TACO: Trash Annotations in Context for Litter Detection," arXiv preprint arXiv:2003.06975, 2020.

[4] H. Liu, T. Zheng, F. Tan, P. Li, and J. Zheng, "Deep Learning Based Garbage Detection in Urban Environment," Journal of Intelligent and Fuzzy Systems, vol. 38, no. 4, pp. 4761–4772, 2020.

[5] Z. Wang, Y. Wang, B. Liu, and Q. Liu, "Underwater Garbage Detection Based on Improved Faster R-CNN," Applied Sciences, vol. 10, no. 3, pp. 1–16, 2020.

[6] Y. Zhang, X. Chen, J. Li, and L. Huang, "Multi-Scale Feature Fusion for Real-time Garbage Detection in Surveillance Videos," IEEE Access, vol. 9, pp. 12345–12356, 2021.

[7] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, pp. I-511, 2001.

[8] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," 2005 IEEE Computer Society Conference on CVPR, vol. 1, pp. 886–893, 2005.

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," Proceedings of CVPR, pp. 580–587, 2014.

[10] R. Girshick, "Fast R-CNN," Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 1440–1448, 2015.

[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137–1149, 2017.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," Proceedings of CVPR, pp. 779–788, 2016.

[13] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv preprint arXiv:1804.02767, 2018.

[14] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv preprint arXiv:2004.10934, 2020.

[15] M. A. Arebey, M. A. Hannan, H. Basri, and H. Pokhrel, "Solid waste bin level detection using gray level co-occurrence matrix and neural networks," Expert Systems with Applications, vol. 38, no. 7, pp. 8010–8018, 2011.

[16] S. H. Kolekar, P. P. Bhoite, S. V. Chavan, and P. S. Deshmukh, "Smart Dustbin Management System," International Journal of Computer Applications, vol. 176, no. 15, pp. 25–29, 2020.

[17] T. V. P. Anagnostopoulos, K. Zaslavsky, A. Medvedev, and S. Khoruzhnikov, "Robust and real-time municipal solid waste management using smart city waste monitoring and collection framework," IEEE International Conference on Internet of Things, pp. 617–624, 2015.

 ©IJSRED: All Rights are Reserved