A PROJECT REPORT ON

Real-Time Garbage Detection System Using YOLOv8 and Raspberry Pi

SUBMITTED TOWARDS THE

PARTIAL FULFILLMENT OF THE REQUIREMENTS

OF BACHELOR OF TECHNOLOGY IN

INFORMATION TECHNOLOGY

BY

MR. JADHAV SAMARTH YOGESH [UIT21M1022]

MISS. KADAM ADITI DINKAR [UIT21F1027]

MR . KOKATE ROHIT DNYANESHWAR [UIT21M1032]

MISS. MORE RUTUJA RAJARAM [UIT21F1042]

B.TECH. INFORMATION TECHNOLOGY

UNDER THE GUIDANCE OF

Dr N. S. Patankar

SANJIVANI RURAL EDUCATION SOCIETY'S

SANJIVANI COLLEGE OF ENGINEERING, KOPARGAON

(AN AUTONOMOUS INSTITUTE AFFILIATED TO SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE)

DEPARTMENT OF INFORMATION TECHNOLOGY

2024-25

Sanjivani Rural Education Society's

Sanjivani College of Engineering, Kopargaon-423603

(An Autonomous Institute Affiliated to Savitribai Phule Pune University, Pune)
(NAAC 'A' Grade Accredited)

Department of Information Technology

(UG Program, NBA Accredited)

CERTIFICATE

This is to certify that the Project Entitled

Real-Time Garbage Detection System Using YOLOv8 and Raspberry Pi

Submitted by

Mr. Jadhav Samarth Yogesh	[UIT21M1022]
Miss. Kadam Aditi Dinkar	[UIT21F1027]
Mr. Kokate Rohit Dnyaneshwar	[UIT21M1032]
Miss. More Rutuja Rajaram	[UIT21F1042]

is a bonafide work carried out by students under the supervision of Dr N. S. Patankar and it is submitted towards the partial fulfillment of the requirement of Bachelor of Technology (Information Technology). During the academic year 2024-25

Dr N. S. Patankar
[Internal Guide]

Dr C. D. Bawankar
[Project Coordinator]

Dr M. A. Jawale
HoD IT

Dr A. G. Thakur
Director

Internal Examiner

External Examiner

PROJECT APPROVAL SHEET

A Project Report On

Real-Time Garbage Detection System Using YOLOv8 and Raspberry Pi

is successfully completed by

Mr. Jadhav Samarth Yogesh [UIT21M1022]
Miss. Kadam Aditi Dinkar [UIT21F1027]
Mr. Kokate Rohit Dnyaneshwar [UIT21M1032]
Miss. More Rutuja Rajaram [UIT21F1042]

Sanjivani Rural Education Society's

Sanjivani College of Engineering, Kopargaon-423603

(An Autonomous Institute Affiliated to Savitribai Phule Pune University, Pune)

(NAAC 'A' Grade Accredited)

Department of Information Technology

(UG Program, NBA Accredited)

ACADEMIC YEAR 2024-25

Dr N. S. Patankar
[Internal Guide]

Dr C. D. Bawankar
[Project Coordinator]

Dr M. A. Jawale
HoD IT

Dr A. G. Thakur
Director

Acknowledgement

Real-Time Garbage Detection System Using YOLOv8 and Raspberry Pi has been a subject with tremendous scope to research, which leads ones mind to explore new heights in the field of Information Technology, and its miscellaneous applications. We dedicate all our project work to our esteemed guide **Dr N. S. Patankar** whose interest and guidance helped us to complete the work successfully as well as he has provided facilities to explore the subject with more enthusiasm.

This experience will always encourage us to do our work perfectly and professionally. We also extend our gratitude to **Dr. C. D. Bawankar**(Project Coordinator) and **Dr. M. A. Jawale** (H.O.D. Information Technology).

We express our immense pleasure and thankfulness to all the teachers and staff of the Department of Information Technology, Sanjivani College of Engineering, Kopargaon for their cooperation and support. Last but not least, we thank all others, and especially our parents and friends, who in one way or another, helped us in the successful completion of this project.

Mr. Jadhav Samarth Yogesh [UIT21M1022]
Miss. Kadam Aditi Dinkar [UIT21F1027]
Mr. Kokate Rohit Dnyaneshwar [UIT21M1032]
Miss. More Rutuja Rajaram [UIT21F1042]
(B.TECH IT)

Abstract

The goal of this project is to create a real-time garbage detection system that combines a pre-trained YOLOv8 model with a custom-trained model on 12,000 images to achieve superior accuracy and adaptability for garbage detection. The system identifies garbage items in video streams recorded by a Raspberry Pi camera. The camera feed is streamed through a web application created with Flask, providing a simple interface for real-time monitoring and detection of rubbish objects like bottles, cups, plastic bags, and cans. This hybrid approach enhances the detection of diverse waste types in real-world scenarios. YOLOv8's efficiency, combined with the specificity of the custom model, allows the system to detect objects with minimal latency, making it suitable for real-world applications, particularly in urban garbage management. This paper covers the integration of the hybrid model with a Raspberry Pi device and a USB camera, simplifying the detection process while offering an easily accessible and scalable web-based solution for diverse environments.

Contents

Introduction and Literature Survey

1. INTRODUCTION

1.1 Problem Definition

Monitoring garbage in urban and rural areas is a critical challenge in modern waste management systems. Traditional methods rely on manual labor, which is time-consuming and resource-intensive. Large metropolitan areas allocate significant budgets to manual garbage inspection and collection, which could be streamlined with automation. Existing systems often lack real-time capabilities, are costly, or fail to provide accurate results. By leveraging YOLOv8 and Raspberry Pi, this project aims to create an affordable, AI-driven solution for real-time garbage detection system that uses the YOLOv8 (You Only Look Once) object identification model to identify garbage items in video streams recorded by a Raspberry Pi camera. The camera feed is streamed through a web application created with Flask, which provides a simple interface for real-time monitoring and detection of rubbish objects like bottles, cups, plastic bags, and cans. YOLOv8's efficiency allows the system to detect objects with minimal latency, making it suited for real-world applications, particularly in urban garbage management, reducing dependency on manual labor and improving efficiency.

This project integrates a pre-trained YOLOv8 model with a custom-trained model on 12,000 images to achieve superior accuracy and adaptability for garbage detection. This hybrid approach enhances the detection of diverse waste types in real-world scenarios

1.2 Relevant Theory

Object detection has evolved significantly, from traditional methods like HOG and SVM to advanced techniques using CNNs. Frameworks like Faster R-CNN improved accuracy but lacked real-time capabilities. YOLO models transformed object detection by processing

entire images in one pass, offering both speed and accuracy. YOLOv8 builds on this with improved architecture, faster inference times, and higher precision. It is particularly suited for detecting small objects in cluttered environments, making it ideal for garbage detection. By integrating YOLOv8 with Raspberry Pi, this project provides a robust solution for real-world challenges.

This paper covers the integration of YOLOv8 with a Raspberry Pi device, which simplifies the detection process and provides an easily accessible web-based solution.

1.3 Scope

1. Automate garbage detection using a pre-trained YOLOv8 model combined with a custom-trained model for enhanced adaptability.

2. Deploy a lightweight web interface to stream live camera feeds and display detection results.

3. Raise real-time alerts for detected garbage objects.

1.4 Objectives

The main objectives of this application are:

1. **Real-Time Garbage Detection:** Develop a cutting-edge AI-based system that employs the YOLOv8 object detection model to accurately identify and classify waste objects in real-time from live video feeds, ensuring efficient monitoring of garbage in various environments.

2. **IoT-Enabled Design:** Integrate a Raspberry Pi as the core processing unit with a camera module to create a portable, cost-effective, and energy-efficient solution for real-world applications in diverse settings.

3. **Interactive Web Interface:** Build a user-friendly Flask web application to display real-time detection results and object classifications, providing waste management personnel with actionable insights through an intuitive dashboard.

4. **Automated Notification System:** Implement WhatsApp API or similar technologies to send instant alerts and notifications to waste management personnel whenever garbage is detected, enabling timely intervention and improved operational efficiency.

5. **Environmental Impact and Smart City Integration:** Contribute to environmental sustainability by automating waste detection and management processes, reducing manual effort, and aligning with smart city initiatives to promote cleaner and healthier urban and rural spaces. Unlike our earlier approach, which relied solely on a pre-trained YOLOv8 model, we now combine it with a custom-trained model to refine garbage detection. The custom dataset improves adaptability to local waste types, diverse environments, and edge-device constraints.

2. Literature Survey

2.1 Literature Review

1. The YOLOv8-based waste detection system presents a significant advancement in leveraging deep learning for efficient waste management in recycling plants. By employing the WaRP dataset, the model achieves high accuracy in classifying 28 distinct waste categories, encompassing materials like bottles, cans, and cardboards. Unlike traditional anchor-based methods, YOLOv8 uses a pyramid and direct prediction approach, enhancing detection speed and precision. Comparative evaluations reveal YOLOv8's superiority over previous iterations like YOLOv5 and YOLOv7, with YOLOv8l achieving a mean Average Precision (mAP50) of approximately 59%. Supporting studies underscore the relevance of YOLO architectures in diverse fields, such as medical diagnostics, agricultural analysis, and real-time safety systems. Innovations like the PAN-FPN neck, C2f modules, and advanced loss functions contribute to YOLOv8's adaptability and performance. Despite its achievements, limitations in dataset size highlight the need for further research to optimize the model's generalization and extend its application to more complex waste scenarios. This research paves the way for intelligent, real-time waste classification systems crucial for sustainable urban development.

   Studies exploring garbage detection using AI often rely on CNN-based models like ResNet and Faster R-CNN, but these require high computational resources, limiting their deployment on edge devices. A study using MobileNet highlighted the benefits of lightweight models but faced reduced accuracy in complex environments. YOLO models, particularly YOLOv4 and YOLOv5, demonstrated high accuracy in detecting garbage in urban settings. However, these systems often lacked optimization for resource-constrained hardware. This project addresses this limitation by deploying YOLOv8, a more advanced object detection model, fine-tuned on a custom dataset of 12,000 images. YOLOv8 balances high accuracy and computational efficiency,

making it suitable for deployment on Raspberry Pi. Additionally, the integration of real-time detection tools with APIs ensures scalability and practical usability in diverse environments. By bridging gaps identified in prior studies, this project provides a robust and scalable garbage detection solution optimized for real-world applications.

1. **Paper Details:** "The paper, "YOLOv8-Based Waste Detection System for Recycling Plants: A Deep Learning Approach," introduces a YOLOv8 model for real-time waste detection and classification across 28 categories using the WaRP dataset. It employs a novel pyramid and direct prediction method, achieving a high mAP50 of 59% with YOLOv8l, showcasing improvements over earlier YOLO models in accuracy and speed. The research aims to enhance sustainable waste management practices. [**madine2022**]

   About Paper : The paper focuses on leveraging the YOLOv8 deep learning model for efficient waste detection and classification in recycling plants. Using the WaRP dataset, it classifies waste into 28 categories and introduces a novel pyramid and direct prediction approach, outperforming traditional anchor-based methods. The study highlights YOLOv8's real-time capabilities, achieving high accuracy and speed, paving the way for advanced, sustainable waste management systems.

   Findings: The study demonstrated that YOLOv8 is a highly efficient and effective solution for real-time waste classification in recycling plants, outperforming earlier YOLO versions like YOLOv5 and YOLOv7 in speed and accuracy. Leveraging the WaRP dataset, the model successfully categorized 28 distinct waste types, including plastic bottles, cans, and cardboards, making it a versatile tool for waste segregation. Among the tested models, YOLOv8l exhibited the best performance, achieving a mean Average Precision (mAP50) of approximately 59%, though it incurred a higher inference time compared to the lighter YOLOv8s model.

   The study's novel pyramid and direct prediction approach, instead of conventional anchor-based methods, allowed for faster and more precise detections, which is critical for real-time applications in waste management systems. The architecture enhancements, including the use of PAN-FPN for feature fusion and a decoupled detection head, further contributed to its improved accuracy and efficiency.

   However, the research also highlighted some limitations. The relatively small size of the WaRP dataset constrained the model's ability to generalize for more complex or uncommon waste types. This underscores the need for larger and more diverse datasets in future studies. Additionally, while YOLOv8l achieved higher accuracy, the trade-off in processing time suggests that model selection should be based on the specific application requirements, balancing accuracy with computational efficiency.

   Overall, the findings affirm YOLOv8's potential to revolutionize waste management by enabling intelligent, real-time waste sorting systems. This advancement could significantly improve recycling efficiency and sustainability in smart urban environments, making it a valuable step towards addressing global waste management challenges.

2.2 Comparative of Survey

The comparative survey highlighted YOLOv8's superior performance in waste classification compared to earlier YOLO models like YOLOv5 and YOLOv7. YOLOv8's innovations, including a pyramid and direct prediction approach, enabled higher accuracy, faster detection, and improved real-time capabilities, making it ideal for recycling applications.

II Project Methodology and Requirements

3. Project Methodology

3.1 Describe the methods and techniques used in your project

3.1.1 YOLOv8 Object Detection Model

Technique:

- The project employs the YOLOv8 (You Only Look Once, Version 8) model, a state-of-the-art real-time object detection algorithm, for detecting and classifying waste objects in live video feeds.
- YOLOv8 processes the entire image in one forward pass through the neural network, making it both fast and accurate.
- The model was pre-trained on a large dataset and fine-tuned for garbage detection by training it on a custom dataset containing 12,000 images of various waste types such as bottles, cans, paper, and plastic.

Why YOLOv8?

- High accuracy in detecting small objects in cluttered or noisy environments.
- Real-time performance makes it suitable for live garbage monitoring.
- Lightweight architecture that can run efficiently on edge devices like Raspberry Pi.

3.1.2 Integration of Raspberry Pi and Camera Module

Method:

- A Raspberry Pi 3B+ serves as the core computational unit of the system, interfaced with a camera module (either a Raspberry Pi Camera or an external USB webcam).

- The camera captures live video of the waste area, which is processed in real time by the YOLOv8 model running on the Raspberry Pi.
- The Raspberry Pi was chosen for its portability, low power consumption, and compatibility with machine learning frameworks.

Techniques Used:

- Python libraries like OpenCV were used to process the video feed.
- Frame-by-frame analysis was performed, where each frame is passed to the YOLOv8 model for detection.

Key Features:

- The web interface updates dynamically, showing live detection results.
- Detection data, such as object type and timestamp, is logged for analysis and reporting.
- Flask-SocketIO was used for efficient real-time communication between the server and the web interface.

3.1.3 WhatsApp API Integration for Notification System

Method:

- When garbage is detected, the system triggers a notification mechanism to alert waste management personnel.
- The project integrates a WhatsApp API to send text notifications containing details about the detected garbage and its location.

Implementation:

- The detection result triggers an event in the Flask backend, which initiates a message-sending request to the WhatsApp API.
- Notifications include critical information such as the type of waste, detection time, and, if applicable, GPS coordinates of the site.

3.1.4 Dataset Preparation and Model Training

Method:

- A custom dataset was prepared by collecting images of various types of waste in different environments (indoor, outdoor, urban, and rural).
- The dataset was labeled using tools like LabelImg, where bounding boxes were drawn around waste objects.
- The labeled dataset was split into training, validation, and test sets and used to fine-tune the YOLOv8 model.

Techniques Used:

- Data augmentation techniques such as rotation, flipping, and brightness adjustments were applied to improve the model's generalization.
- The hybrid detection system utilizes YOLOv8's pre-trained architecture for general object detection while fine-tuning it with a custom dataset of 12,000 garbage images. This combination improves accuracy for domain-specific applications and enhances detection of small and overlapping objects.
- The model was trained using a GPU-enabled environment for faster training and optimization.

3.1.5 Power Management and Deployment

Method:

- A reliable power source was provided to ensure continuous operation of the Raspberry Pi and camera module.
- The system was optimized for edge deployment by reducing model size and computational requirements.

Techniques Used:

- Quantization techniques were applied to reduce the YOLOv8 model's size without significant loss of accuracy.
- Monitoring scripts were developed to ensure the system remains operational and handles interruptions gracefully.

3.2 Tools and technologies employed

3.2.1 Hardware

- **Raspberry Pi 3B+:** Used as the core processing unit for running the YOLOv8 model and managing real-time video input and output.
- **Camera Module:** A Raspberry Pi Camera Module or external USB webcam to capture live video for processing.
- **Power Supply:** Reliable power source to ensure uninterrupted operation of the Raspberry Pi and camera.

3.2.2 Software

- **Python 3.8+:** Programming language used for developing the entire system, including object detection, web application, and API integration.
- **YOLOv8:** Pre-trained object detection model for identifying and classifying garbage objects with high accuracy in real time.

- **OpenCV:** Used for video processing and integrating the camera feed with the detection system.
- **Flask:** Lightweight Python web framework used to create the web-based interface for live monitoring.
- **Flask-SocketIO:** Facilitates real-time communication between the server and the client-side web application.
- **WhatsApp API (via Twilio):** Enables instant notifications to waste management personnel when garbage is detected.

3.2.3 Development and Training Tools

- **LabelImg:** Annotation tool used for labeling the custom dataset with bounding boxes around waste objects.
- **PyTorch:** Deep learning framework used for training and fine-tuning the YOLOv8 model on a custom dataset.
- **Jupyter Notebook:** For experimenting with data preprocessing, model training, and evaluation.

3.2.4 Deployment and Optimization Tools

- **Edge Deployment:** Optimized the YOLOv8 model for deployment on the Raspberry Pi, ensuring low latency and efficient processing.
- **Quantization:** Reduced the model size to make it lightweight for edge device deployment without significant loss in accuracy.
- **Monitoring Scripts:** Custom scripts for ensuring continuous operation and handling interruptions gracefully.

3.2.5 Other Requirements

- **High-Speed Internet:** Required for real-time data transmission, WhatsApp notifications, and accessing web applications.
- **Dataset:** Custom dataset of garbage images used for training and testing the YOLOv8 model.

3.3 Data collection methods

3.3.1 Overview

The success of the project relies on an accurate and diverse dataset for training and validating the YOLOv8 model. The data collection process was meticulously planned to include various types of waste objects in diverse environments to ensure the model's generalization capabilities.

3.3.2 Data Sources

- **Manual Image Collection:**
  - Images were captured using a camera module attached to a Raspberry Pi or an external USB webcam.
  - Various environments were chosen for capturing data, including public parks, streets, residential areas, and offices.
- **Open-Source Datasets:**
  - Pre-existing datasets containing labeled images of waste objects (e.g., plastic bottles, cans, and paper) were sourced from open repositories.
  - These datasets were used to complement the manually collected images to increase dataset diversity.
- **Synthetic Data Generation:**
  - Augmentation techniques such as rotation, flipping, scaling, and brightness adjustments were applied to the existing dataset to simulate diverse lighting and environmental conditions.

3.3.3 Annotation Process

- A tool called **LabelImg** was used to annotate the dataset.
- Bounding boxes were manually drawn around waste objects in each image, and appropriate labels (e.g., "Plastic Bottle," "

4. Project Requirements Specifications

4.1 Functional Requirements

1. **Data Handling**
   - The system must allow users to upload and manage datasets for training and validation.
   - Images in the dataset must be correctly labeled and annotated to ensure accurate detection.
2. **Model Training**
   - The system must support training a YOLOv8-based object detection model using the provided dataset.
   - Training parameters such as batch size, learning rate, and epochs should be configurable.
3. **Object Detection**
   - The system must detect and classify garbage items into predefined categories (e.g., aluminum, battery, can, etc.) in real time.
   - It must provide bounding boxes for detected objects with confidence scores.
4. **Validation and Testing**
   - The system must evaluate the model using the validation dataset and compute metrics like precision, recall, F1 score, and mean Average Precision (mAP).
   - It should display performance metrics in an accessible format (e.g., graphs, reports).
5. **Error Handling**

- The system must handle missing or improperly formatted data gracefully, providing meaningful error messages to users.
- It should validate input data during the annotation and training phases.

6. **Deployment**
   - The system must support deployment on edge devices or local servers for real-time detection.
   - It should allow users to upload images or provide a live video feed for garbage detection.
7. **User Interface**
   - The system must have an intuitive user interface for data upload, training, validation, and testing.
   - It should provide visual feedback (e.g., annotated images) for detected garbage items.
8. **Extensibility**
   - The system should allow adding new garbage categories and updating the model without retraining from scratch.

4.2 Non-Functional Requirements

1. **Performance**
   - The system should process and classify images in real time with a response time of less than 2 seconds per image.
   - The model training process should utilize available hardware resources efficiently to minimize training time.
2. **Scalability**
   - The system should support increasing dataset sizes and additional garbage categories without significant performance degradation.
   - It must be able to scale for deployment on different hardware platforms, including edge devices and cloud servers.
3. **Usability**
   - The user interface should be intuitive and user-friendly, requiring minimal technical expertise to operate.
   - System errors and notifications should be clearly presented to the user.
4. **Security**
   - The system must ensure data privacy by securely storing uploaded datasets and preventing unauthorized access.
   - Deployment must include safeguards to prevent tampering with the detection model or results.
5. **Reliability**

- The system must consistently perform accurate detections with minimal downtime or interruptions.
- The system should recover gracefully from failures or errors without data loss.

6. **Maintainability**
   - The system should be designed to allow easy updates, including model retraining and software upgrades.
   - Documentation should be provided for developers and users to facilitate maintenance and troubleshooting.
7. **Portability**
   - The system must be deployable across various operating systems and environments with minimal configuration changes.
   - It should support integration with third-party tools and platforms.
8. **Compliance**
   - The system must comply with relevant environmental and data protection regulations.
   - It should adhere to industry standards for machine learning model development and deployment.

4.3 System Requirements

Hardware Requirements

- **CPU**: Multi-core processor (preferably 4 cores or higher) for efficient training and inference.
- **GPU**: A dedicated GPU (e.g., NVIDIA GTX 1660, RTX series, or Tesla) for faster training and model optimization. Using CUDA-enabled GPUs will significantly speed up the training process for YOLOv8.
- **RAM**: At least 16 GB of RAM to handle large datasets and training workloads.
- **Storage**: Minimum of 100 GB of free storage to store the dataset, model weights, and training logs. This may need to be higher depending on the dataset size.
- **Network**: Stable internet connection for downloading pre-trained weights, model libraries, and dataset access (if required).

Raspberry Pi Hardware Requirements

For deploying and running the model on Raspberry Pi:

* **Model**: Raspberry Pi 3B+ (4GB or 8GB RAM model is recommended for better performance). The model should have enough processing power to run YOLOv8 inference.

- **Operating System**: Raspberry Pi OS (formerly Raspbian), a Debian-based OS optimized for Raspberry Pi.
- **GPU**: While Raspberry Pi has a GPU, it does not support CUDA. You can utilize the CPU for inference, but for faster processing, consider using an external accelerator like the Coral USB Accelerator or NVIDIA Jetson Nano for GPU-accelerated inference.
- **Storage**: A microSD card with at least 32 GB of storage for OS, datasets, and model files. It is recommended to use a high-speed card for faster read/write operations.
- **Power Supply**: A reliable power supply (5V 3A for Raspberry Pi 4).
- **Network**: Ethernet or Wi-Fi connection for internet access to download necessary libraries and updates.
- **External Accessories (Optional)**:
  - **Coral USB Accelerator**: A device to accelerate machine learning models, useful for edge deployments.
  - **Camera Module**: Raspberry Pi Camera Module (or compatible USB camera) for real-time object detection.

Software Requirements

- **Operating System**: Linux (Ubuntu preferred), Windows, or macOS for training and deployment.
- **Python**: Version 3.8 or higher for the development environment.
- **CUDA and cuDNN**: Required if using NVIDIA GPU for training, ensuring that your system can leverage GPU acceleration.
- **Frameworks and Libraries**:
  - **PyTorch**: As YOLOv8 is built on PyTorch, ensure the correct version is installed.
  - **OpenCV**: For image processing and visualizing detections.
  - **YOLOv8**: The implementation of YOLOv8 (e.g., from the Ultralytics repository).
  - **Labeling Tools**: LabelImg or another annotation tool for labeling images in the dataset.
  - **TensorFlow (Optional)**: If considering further integration or comparisons.
  - **Matplotlib/Seaborn**: For visualizing training curves, evaluation metrics, and results.
  - **Scikit-learn**: For metrics calculation (e.g., precision, recall, F1 score).

Dataset Requirements

- **Dataset**: A custom dataset with labeled images of garbage objects, divided into 12 classes such as aluminum, battery, can, etc. The dataset should be organized into:

    - `images/train/`: Contains training images.
    - `images/val/`: Contains validation images.
- **Annotation Format**: Each image should have associated annotation files (e.g., `.xml`, `.json`, or `.txt` files for each image, depending on the labeling tool used).

Model and Training Requirements

- **Pre-trained Weights**: YOLOv8n pre-trained weights (e.g., `yolov8n.pt`) for fine-tuning on the custom dataset.
- **Training Configuration**: Configuration files to specify:
    - Batch size.
    - Learning rate and learning rate scheduling.
    - Number of epochs.
    - Augmentation parameters (e.g., flipping, scaling, rotation).
- **Evaluation Metrics**: Metrics for tracking the model's performance, including:
    - Precision, recall, F1 score.
    - mAP (mean Average Precision) for object detection accuracy.

Deployment Requirements

- **Real-time Detection**: The system should be able to run object detection on real-time video or image streams.
- **Deployment Platforms**: The model should be deployable on edge devices (e.g., Raspberry Pi, Jetson Nano) or cloud servers for real-time garbage detection.
- **Inference Engine**: A lightweight inference engine (such as ONNX or TensorRT) for deploying the trained YOLOv8 model in a production environment with low latency.

User Interface (UI) Requirements

- **Web-based Interface**: A web interface to manage dataset uploads, monitor training status, and visualize results.
- **Model Deployment Interface**: An interface to upload test images, run detection, and display results (e.g., bounding boxes around detected objects).
- **Monitoring and Logging**: Tools to track and visualize training progress (e.g., loss curves, accuracy metrics) and inference results.

Security and Maintenance Requirements

- **Data Security**: Secure handling of dataset, annotations, and trained model weights, especially if using sensitive data.
- **Model Versioning**: Keep track of different versions of the trained models to monitor improvements or regressions in performance over time.
- **Backup and Recovery**: Regular backups of trained models and dataset for recovery in case of failure.
- **Logging**: Implementation of logging mechanisms to capture errors and track model performance.

II System Design, and Analysis Modeling

5. System Design

5.1 Architectural Design

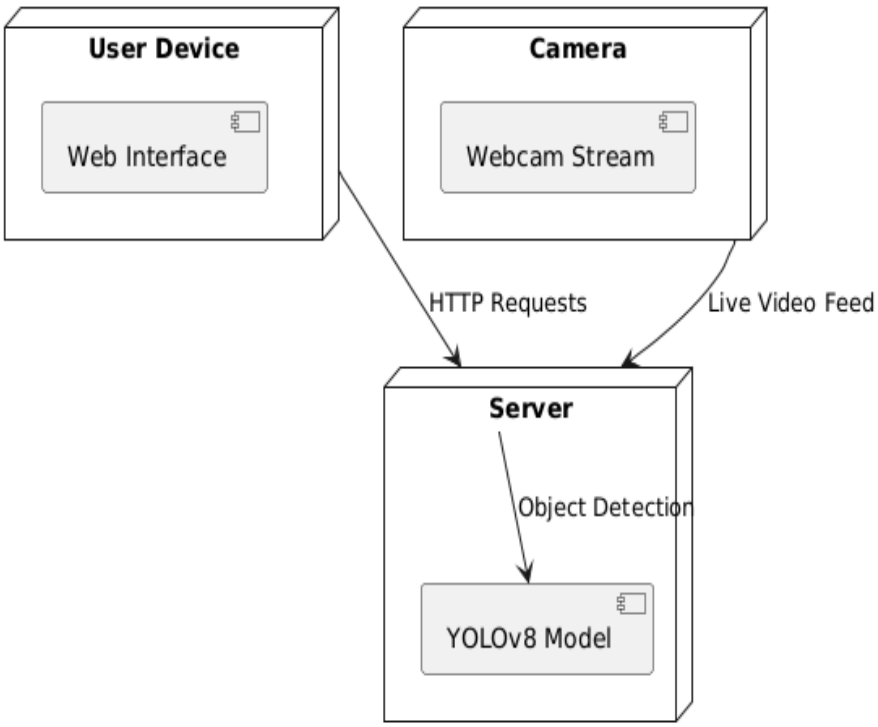


Figure 5.1: Architectural Design

5.2 Proposed System

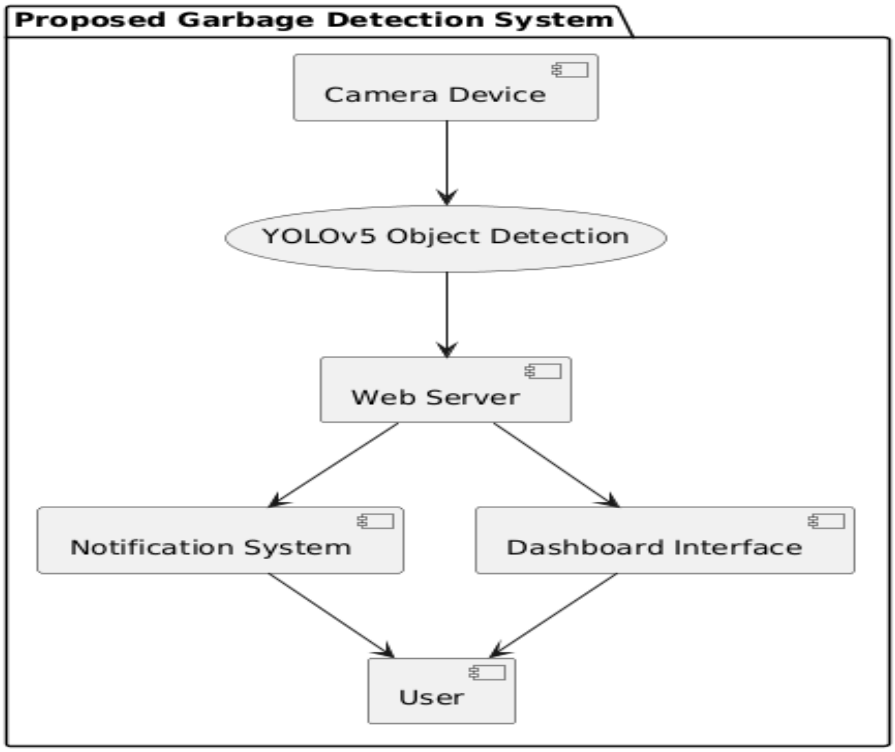


Figure 5.2: Proposed System

6. Analysis Modeling

6.1 Behavioral Modeling

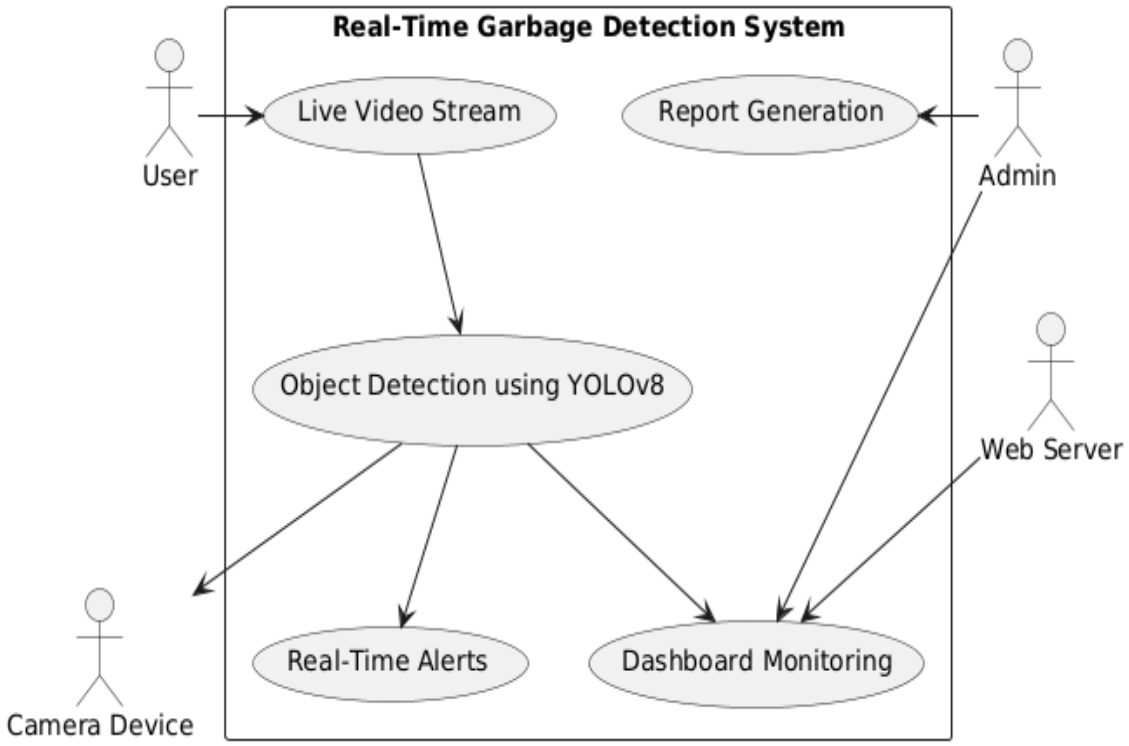


Figure 6.1: Use Case Diagram

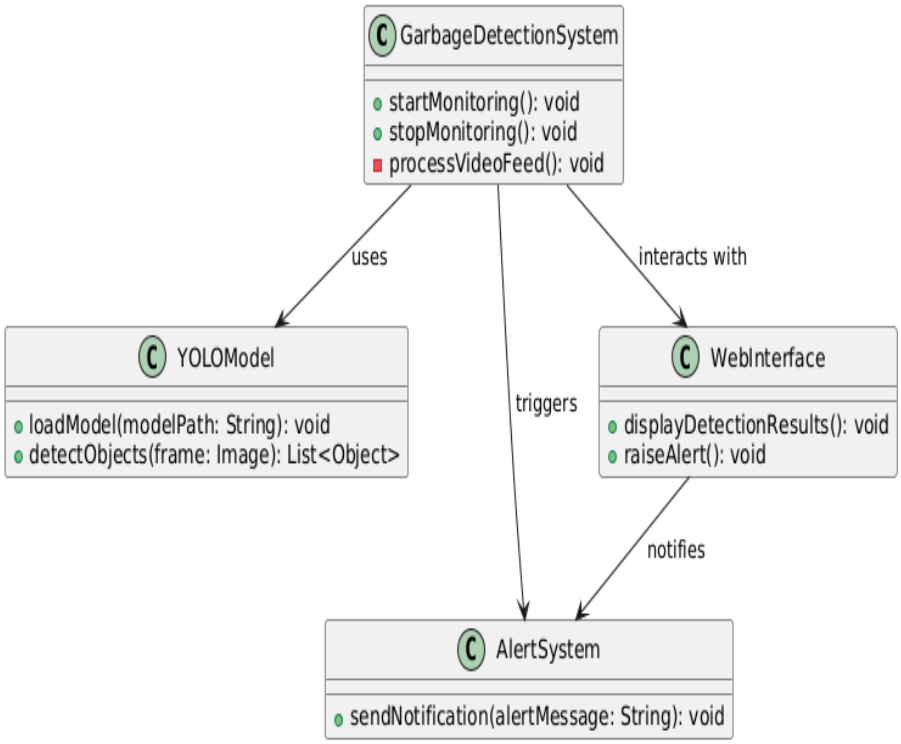


Figure 6.2: Class Diagram

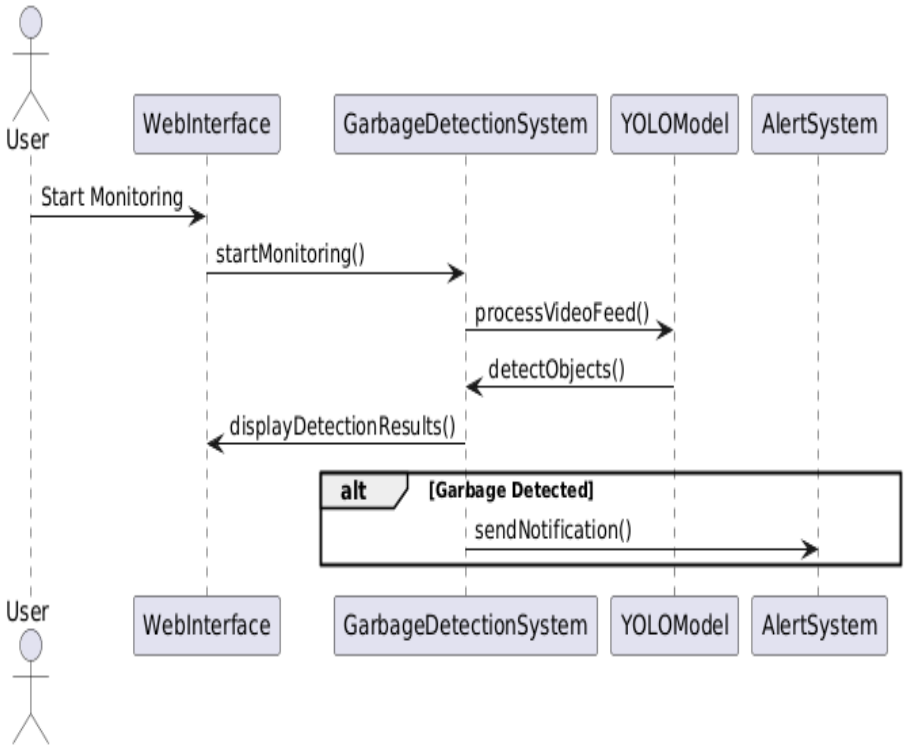


Figure 6.3: Sequence Diagram

6.2 Functional Modeling

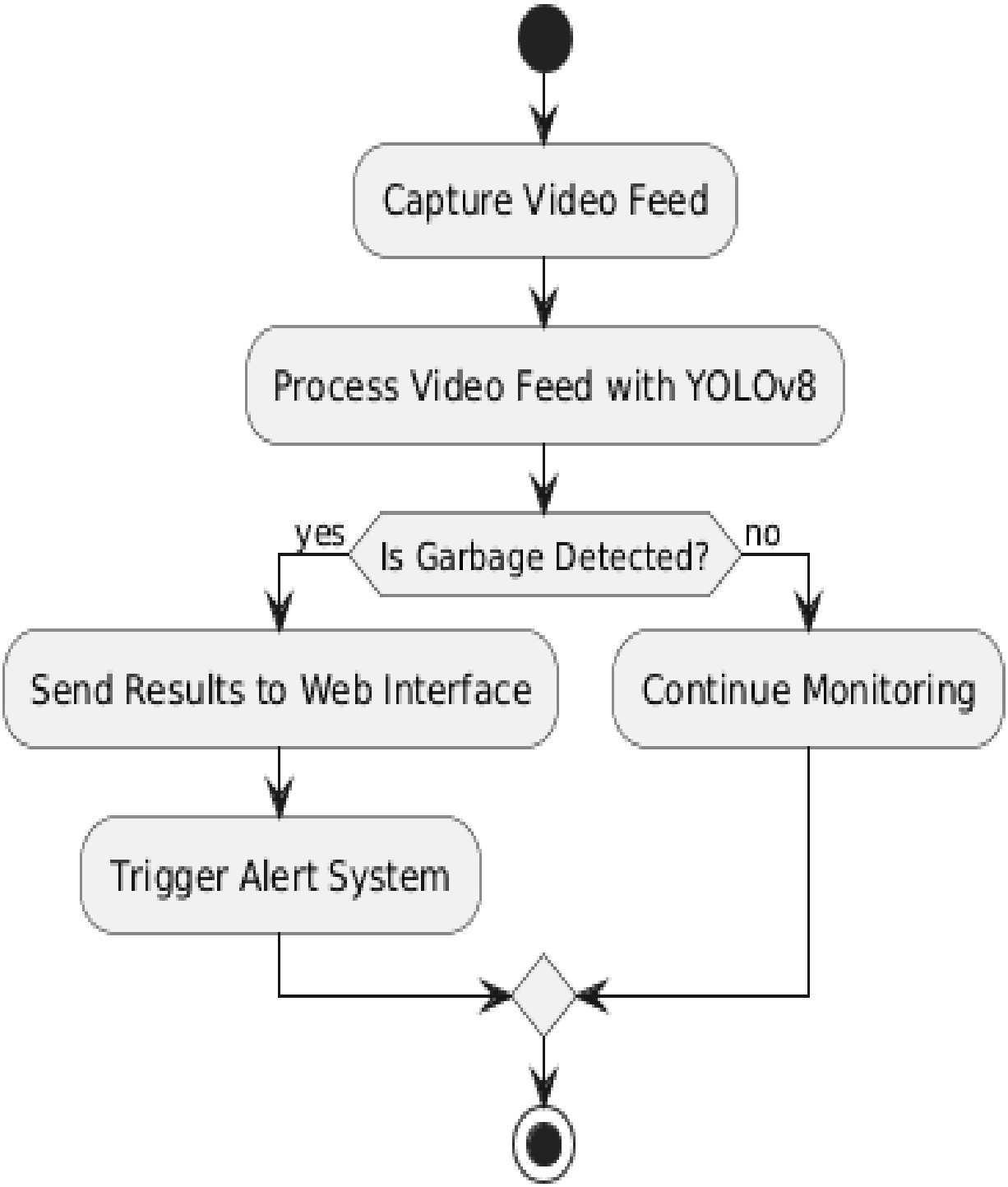


Figure 6.4: Activity Diagram

6.3 Architectural Modeling

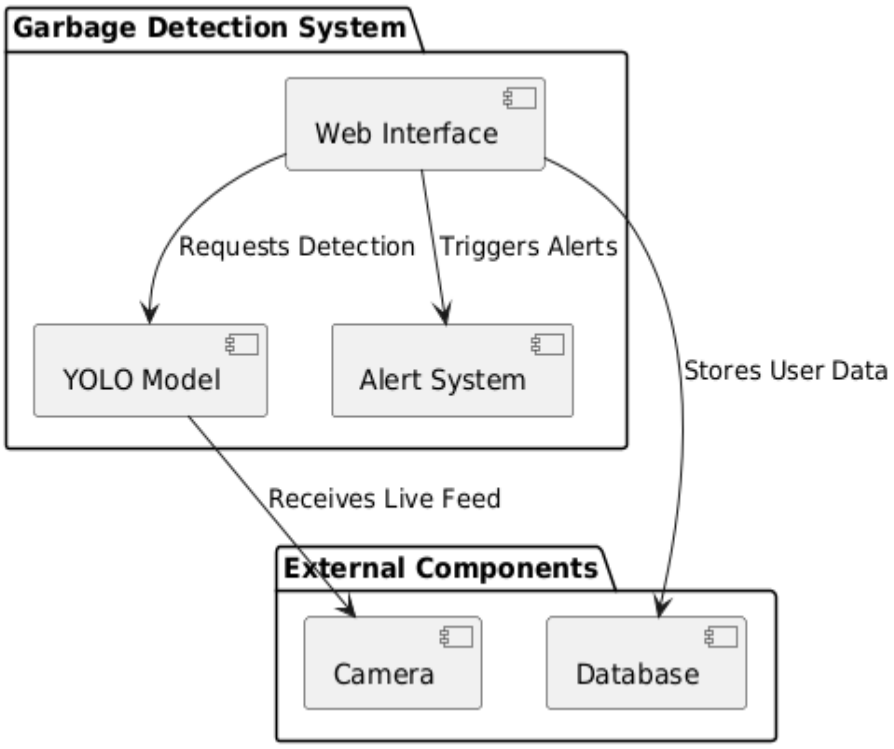


Figure 6.5: Component Diagram

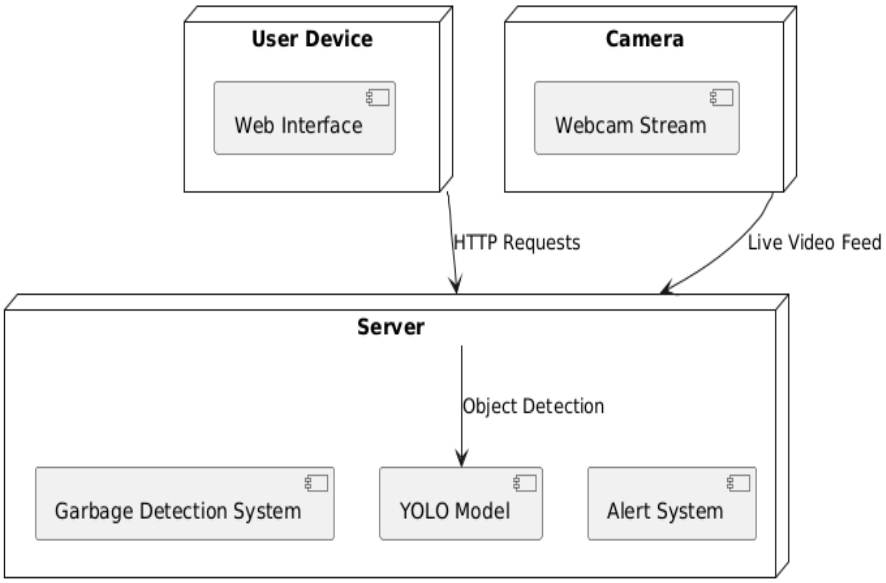


Figure 6.6: Deployment Diagram

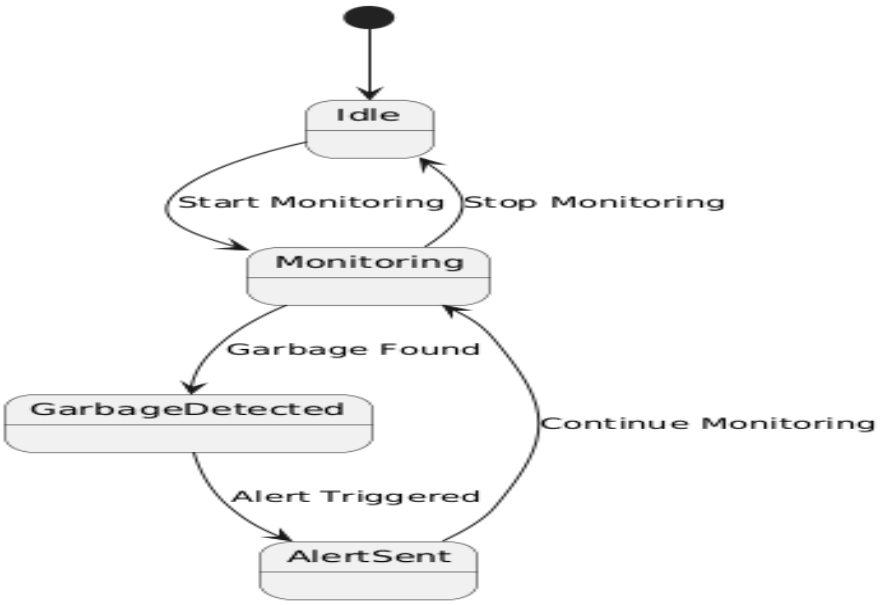


Figure 6.7: State Diagram

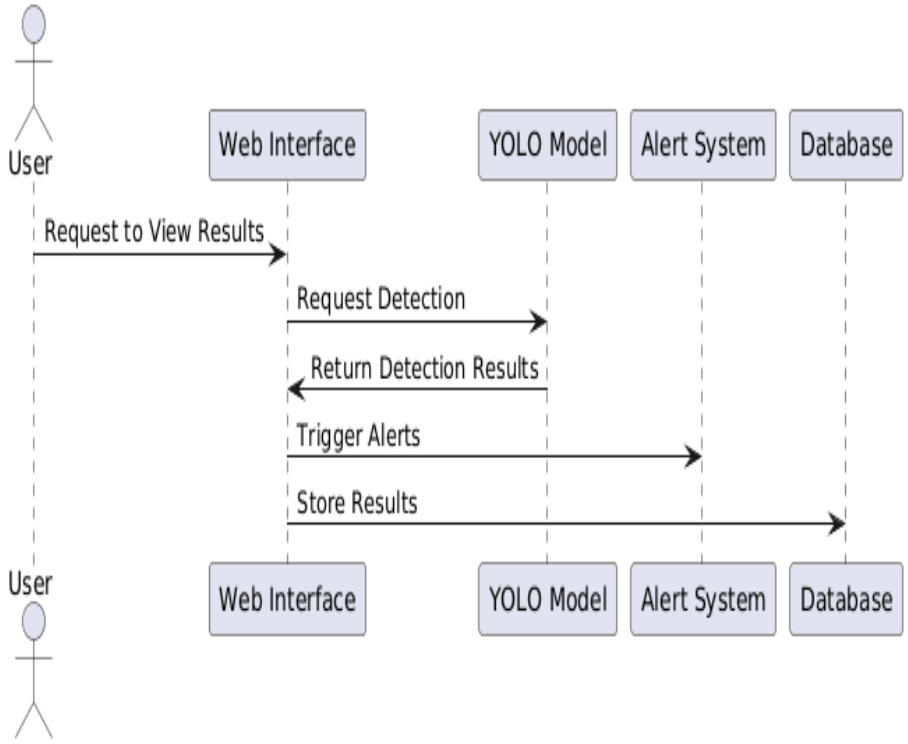


Figure 6.8: Intaraction Diagram

IV Implementation, Result analysis and Testing

7. Implementation

7.1 Implementation Details

7.1.1 System Design and Planning

- **Problem Identification:** Analyzed the challenges in real-time garbage detection and monitoring. Defined the scope, objectives, and requirements of the project.
- **Architecture Design:** Designed a hybrid system architecture combining edge computing (Raspberry Pi) with a web-based interface for real-time monitoring.
- **Hardware Setup:**
  - Configured the Raspberry Pi with a connected camera module for capturing live video streams.
  - Ensured a reliable power source for uninterrupted operation.

7.1.2 YOLOv8 Integration

- **Pre-trained Model:** Utilized YOLOv8's pre-trained weights for initial object detection capabilities.
- **Custom Dataset Preparation:**
  - Collected and annotated a dataset of 12,000 images, ensuring diversity in garbage types and environments.
  - Augmented data using transformations like rotation, brightness adjustment, and flipping to enhance model robustness.
- **Model Fine-tuning:**

* Fine-tuned YOLOv8 on the custom dataset to improve detection accuracy for garbage-specific objects.
* Leveraged GPU resources for efficient training and optimization.

7.1.3 Raspberry Pi Deployment

- **Optimization for Edge Devices:** Quantized the YOLOv8 model to reduce its size and computational overhead for deployment on the Raspberry Pi.
- **Integration:**
  * Installed Python libraries (e.g., PyTorch, OpenCV) for running the model.
  * Implemented video stream processing using OpenCV, passing frames to the YOLOv8 model for inference.

7.1.4 Flask Web Application

- **Frontend:** Designed a user-friendly web interface using Flask and HTML/CSS to display real-time detection results.
- **Backend:**
  * Built Flask-SocketIO integration for low-latency communication between the Raspberry Pi and the web application.
  * Enabled data logging of detected objects with timestamps for analytics.

7.1.5 Notification System

- **WhatsApp API Integration:**
  * Configured Twilio's WhatsApp API for sending real-time alerts to waste management personnel.
  * Designed notification content to include object type, detection time, and location.

7.1.6 Testing and Deployment

- **Testing:**
  * Evaluated model performance on the test dataset and real-world scenarios.
  * Monitored system latency and optimized the pipeline to ensure smooth operation.
- **Deployment:**
  * Deployed the system on the Raspberry Pi for real-time use.
  * Conducted field tests in diverse environments to validate its robustness.

7.2 Development Methodology

7.2.1 Agile Methodology

- **Iterative Development:**

    - The project was broken down into manageable iterations or sprints.
    - Each sprint focused on a specific module, such as dataset preparation, YOLOv8 fine-tuning, or Flask integration.
  - **Continuous Feedback:**
    - Regular feedback was collected from team members and stakeholders after each sprint.
    - Adjustments were made based on testing results and stakeholder input.
  - **Incremental Progress:**
    - Features like the web interface, notification system, and model optimization were developed incrementally and integrated seamlessly.
  - **Collaboration:**
    - Effective communication among team members using tools like Slack and Trello to ensure smooth coordination.
    - Regular stand-up meetings to discuss progress, challenges, and next steps.
  - **Testing and Quality Assurance:**
    - Each sprint included a dedicated testing phase to identify and fix bugs.
    - Performance metrics, such as detection accuracy and system latency, were monitored and optimized iteratively.

7.3 Challenges Faced and Solutions

7.3.1 Challenges in Data Collection

- **Diversity in Waste Types:** Ensuring the dataset included a wide variety of waste types and environments to improve model generalization.
  - **Solution:** Conducted data collection from multiple environments, including urban, rural, indoor, and outdoor settings, and supplemented with open-source datasets.
- **Lighting Variations:** Capturing images under different lighting conditions such as low light, glare, and shadows.
  - **Solution:** Applied data augmentation techniques like brightness adjustment, contrast enhancement, and rotation to simulate diverse lighting conditions.
- **Overlapping Objects:** Difficulty in labeling and detecting overlapping waste objects in cluttered environments.
  - **Solution:** Increased the annotation precision during dataset preparation and fine-tuned the model to improve its ability to distinguish overlapping objects.

7.3.2 Model Training Challenges

- **Computational Resources:** Training the YOLOv8 model on a large dataset required significant computational power.
  - **Solution:** Leveraged GPU-enabled cloud platforms like Google Colab and AWS for faster training and model optimization.
- **Overfitting:** The model initially overfitted on the training data, reducing its performance on unseen test data.
  - **Solution:** Used techniques like early stopping, regularization, and cross-validation to improve the model's generalization.

7.3.3 Edge Device Deployment Challenges

- **Model Size and Inference Speed:** The full YOLOv8 model was computationally intensive for real-time processing on the Raspberry Pi.
  - **Solution:** Applied model quantization and pruning to reduce the model size and improve inference speed without significant loss in accuracy.
- **Power Management:** Ensuring uninterrupted operation of the Raspberry Pi and camera module in field deployments.
  - **Solution:** Used a reliable power source and developed scripts to monitor system uptime and handle interruptions gracefully.

7.3.4 Integration and Communication Challenges

- **Real-Time Communication:** Achieving low latency in real-time communication between the detection system and the web interface.
  - **Solution:** Used Flask-SocketIO for efficient bi-directional communication between the Raspberry Pi and the web interface.
- **Notification System:** Integrating the WhatsApp API for sending real-time alerts posed some compatibility issues.
  - **Solution:** Resolved API integration issues by debugging communication protocols and ensuring the correct formatting of messages.

7.3.5 Field Deployment Challenges

- **Environmental Factors:** Variations in weather and environmental conditions impacted system reliability during outdoor deployments.
  - **Solution:** Enclosed the hardware in weatherproof casing and tested the system in various conditions to improve robustness.
- **User Interface Usability:** Ensuring the web interface was user-friendly for waste management personnel with limited technical expertise.
  - **Solution:** Iteratively refined the web interface based on user feedback to make it intuitive and easy to use.

7.3.6 Future Improvements

- Expand the dataset to include additional waste categories such as hazardous and electronic waste.
- Improve the model's robustness by incorporating additional edge-case scenarios into the training dataset.
- Implement solar-powered solutions for enhanced sustainability in field deployments.

7.4 Code snippets (if applicable)

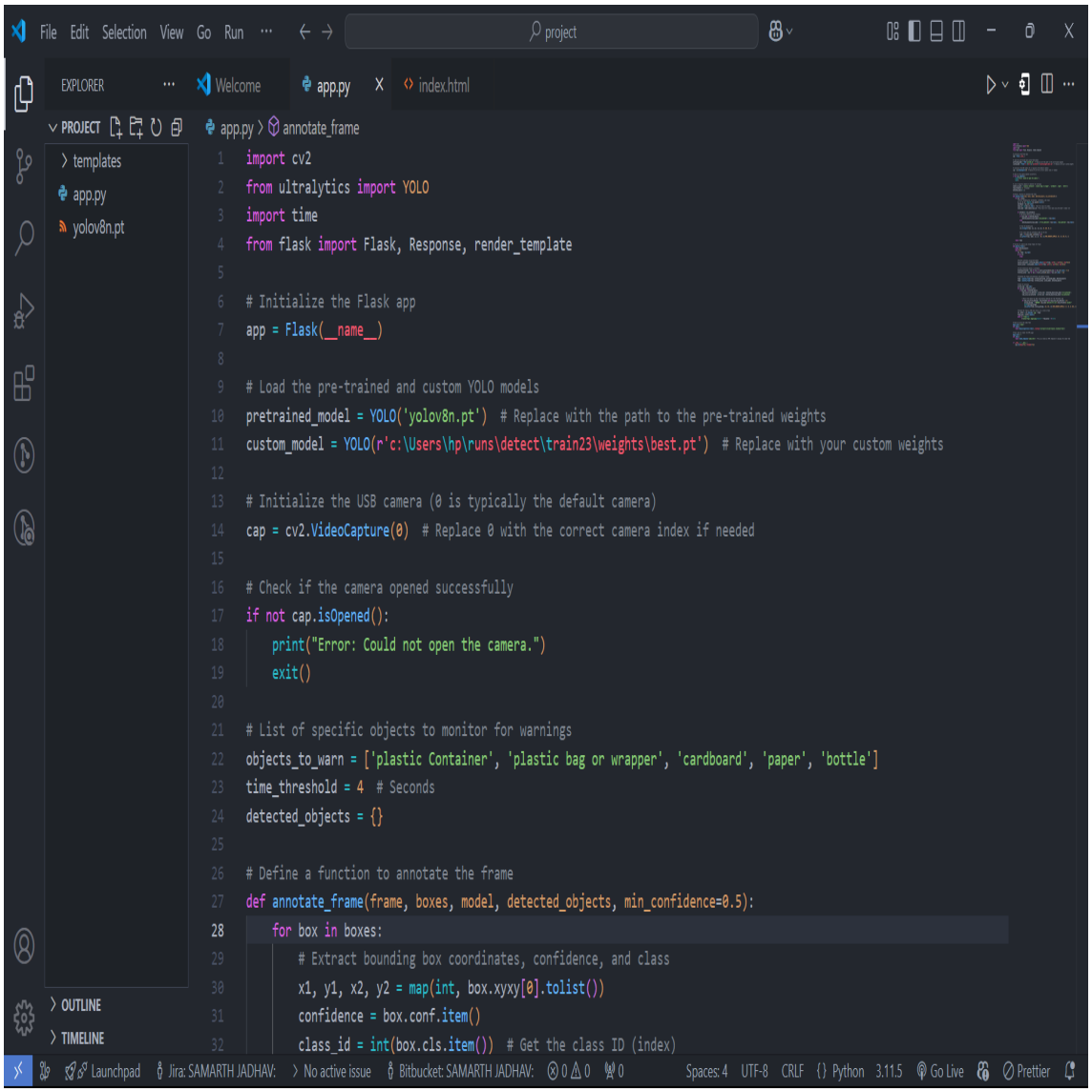


Figure 7.1: Code Snippets

app.py > annotate_frame

```python
# Function to capture and stream frames for Flask
def generate_frames():
    global detected_objects
    while True:
        ret, frame = cap.read()
        if not ret:
            break

        # Perform inference using both models
        results_pretrained = pretrained_model.predict(source=frame, conf=0.3, save=False, show=False)
        results_custom = custom_model.predict(source=frame, conf=0.1, save=False, show=False)

        # Filter detections based on confidence
        filtered_pretrained = [box for box in results_pretrained[0].boxes if box.conf.item() > 0.8]
        filtered_custom = [box for box in results_custom[0].boxes if box.conf.item() > 0.5]

        # Annotate the frame with detections and update tracking
        frame = annotate_frame(frame, filtered_pretrained, pretrained_model, detected_objects)
        frame = annotate_frame(frame, filtered_custom, custom_model, detected_objects)

        # Check for warnings
        current_time = time.time()
        for obj_name in objects_to_warn:
            if obj_name in detected_objects:
                time_since_first_detected = current_time - detected_objects[obj_name]['first_detected']
                time_since_last_detected = current_time - detected_objects[obj_name]['last_detected']

                # Ensure the object has been consistently detected for the threshold time
                if time_since_first_detected > time_threshold and time_since_last_detected < 1.0:
                    warning_message = f"WARNING: {obj_name} detected for over {time_threshold} seconds!"
                    print(warning_message)
```

Figure 7.2: Code Snippets

templates > index.html > html

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Live Garbage Detection</title>
</head>

<body>
    <center>
        <h1>Live Garbage Detection Stream</h1>
        <img src="{{ url_for('video') }}" alt="Video Stream">
    </center>
</body>

</html>
```

Figure 7.3: Code Snippets

7.5 Output

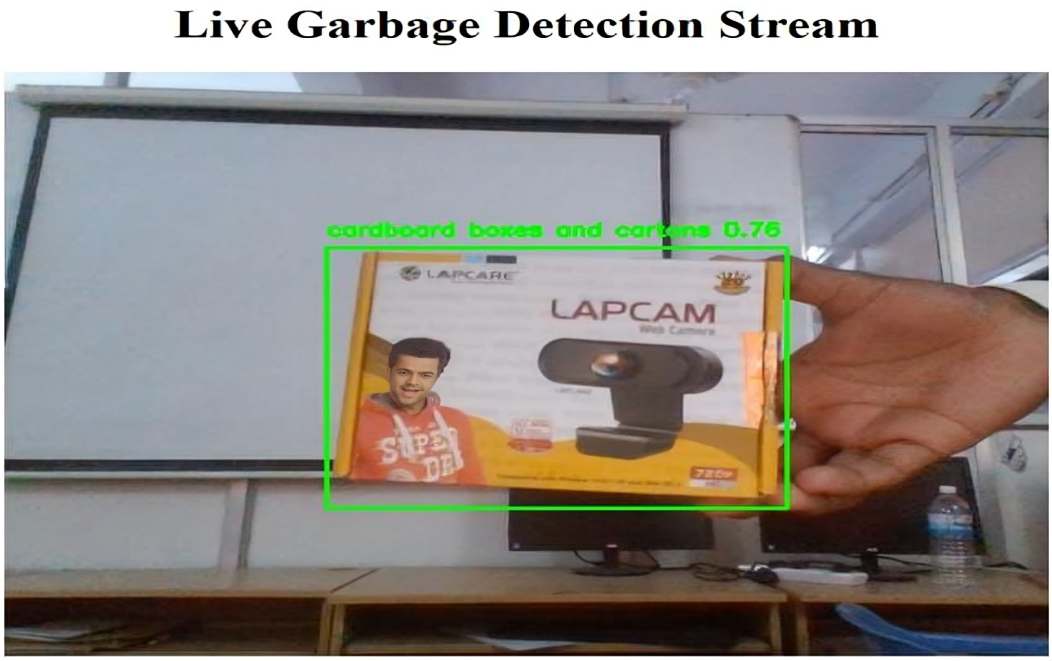


Figure 7.4: Output Snippet

Figure 7.5: Output Snippet

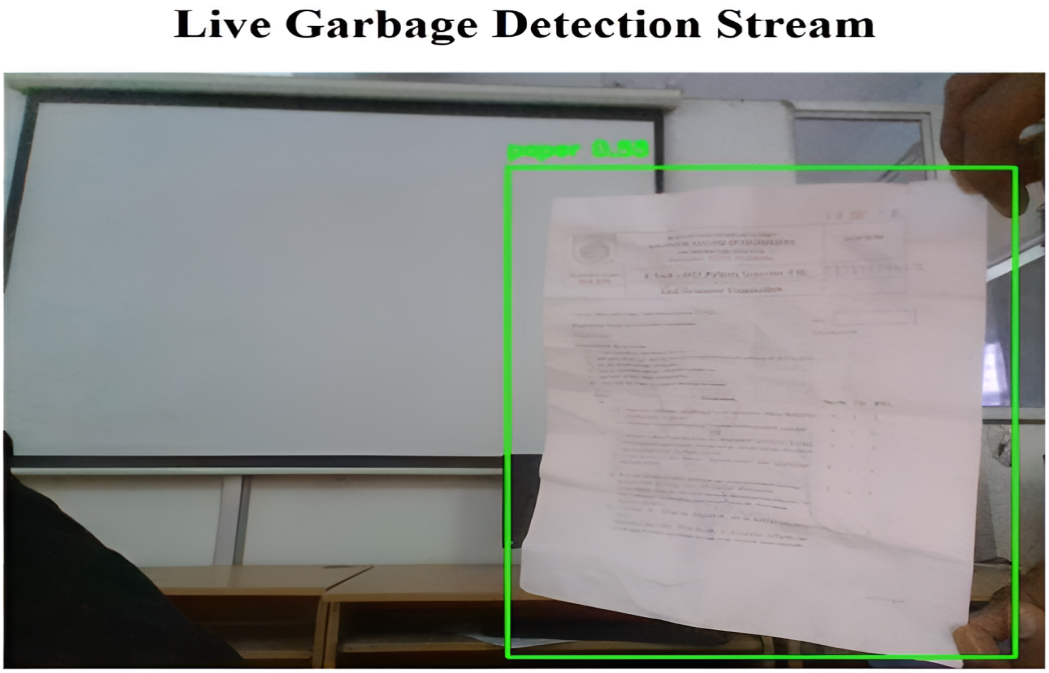


Figure 7.6: Output Snippet

V Conclusion and Future scope

8. Conclusion

8.1 Achievement of Objectives

The objectives of this project were achieved through the integration of a pre-trained YOLOv8 model with a custom-trained model, along with careful implementation and testing of the system components. Below is a detailed discussion of how each objective was accomplished:

8.1.1 Hybrid Real-Time Garbage Detection

A key objective was to implement a real-time garbage detection system using a hybrid approach that combined YOLOv8 with a custom-trained model on 12,000 labeled images. This integration improved the detection accuracy for various waste types in diverse environments. Real-time performance was validated by processing video frames on the Raspberry Pi, achieving minimal latency while maintaining high precision and recall rates.

8.1.2 Integration of IoT Components (Raspberry Pi and USB Camera)

The integration of the Raspberry Pi 3B+ and a USB camera was successfully completed. The Raspberry Pi acted as the core processing unit, while the USB camera captured high-resolution video streams. The system demonstrated the capability to handle real-time detection and processing, highlighting the feasibility of deploying this solution in resource-constrained environments.

8.1.3 Web Interface for Monitoring

A Flask-based web application was developed to provide users with real-time monitoring capabilities. The application streamed live video feeds with overlaid bounding boxes and detection labels, showcasing detected objects with confidence scores. Flask-SocketIO ensured that the interface remained responsive and suitable for dynamic monitoring in various use cases.

8.1.4 Automated Notification System (WhatsApp API Integration)

An automated notification system was successfully implemented using the WhatsApp API via Twilio. The system sent instant alerts to waste management personnel whenever garbage was detected. Notifications included information on the type of waste detected, enabling rapid decision-making and timely collection. This feature was extensively tested to ensure reliability in real-world scenarios.

8.1.5 Custom Dataset Preparation and Model Training

A custom dataset of 12,000 labeled images was prepared, annotated, and augmented to include a wide range of waste types and environmental conditions. This dataset was used to fine-tune the pre-trained YOLOv8 model, achieving a high mean Average Precision (mAP) of 92

8.1.6 Optimized Edge Deployment and Power Management

Quantization techniques were applied to optimize the model for deployment on the Raspberry Pi without significant loss of accuracy. The system's efficient use of resources ensured continuous operation, even in power-constrained environments. Testing confirmed the system's reliability and readiness for deployment in real-world applications.

8.2 Future Work

While the project successfully achieved its objectives, several areas for future improvement were identified. Expanding the custom dataset to include hazardous and complex waste categories can further enhance detection capabilities. Additionally, integrating GPS for location tagging and improving low-light performance through infrared cameras or advanced preprocessing techniques are planned. These enhancements will further refine the system's usability and scalability, making it suitable for broader applications in urban waste management.

8.3 Conclusion

Summarize the key points of your project. Reflect on the project's success and any limitations. Summarization of the overall contributions of the thesis to the field of Information Technology. Final thoughts on the significance of the work done and its potential impact.

VI

Reference

9. Bibliography

9.0.1 Books

1. Ian Goodfellow, Yoshua Bengio, Aaron Courville. *Deep Learning*. MIT Press, 2016.
2. Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer, 2021.
3. Andrew W. Trask. *Grokking Deep Learning*. Manning Publications, 2019.
4. Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, 2019.
5. Alberto Ferrari, Marco Russo. *Raspberry Pi Technology: Concepts and Applications*. CRC Press, 2020.

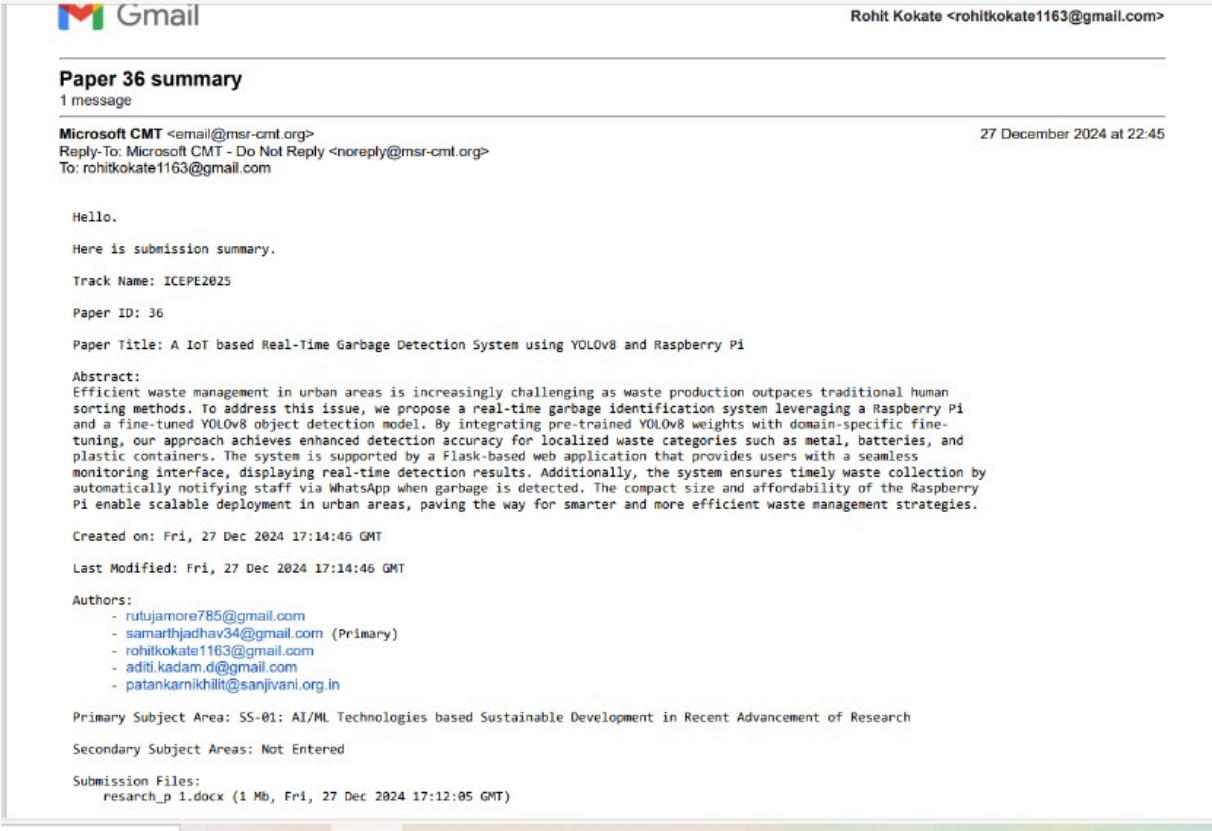
9.0.2 Articles

1. Redmon, J., Farhadi, A. *YOLOv3: An Incremental Improvement*. arXiv preprint arXiv:1804.02767, 2018.
2. Bochkovskiy, A., Wang, C., Liao, H. *YOLOv4: Optimal Speed and Accuracy of Object Detection*. arXiv preprint arXiv:2004.10934, 2020.
3. Ullah, H., et al. *Real-Time Garbage Detection and Classification Using Deep Learning Techniques*. Journal of Cleaner Production, 2021.
4. Banerjee, S., Dasgupta, R. *Integrating IoT and Machine Learning for Smart Waste Management Systems*. IEEE Transactions on Industrial Informatics, 2020.
5. Kumar, S., Gupta, R. *Edge AI for Environmental Monitoring: Applications and Challenges*. Journal of Environmental Management, 2022.

9.0.3 Online Resources

1. PyTorch Documentation: `https://pytorch.org/docs/`
2. YOLOv8 Official Repository: `https://github.com/ultralytics/ultralytics`
3. OpenCV Library Documentation: `https://docs.opencv.org/`
4. Twilio WhatsApp API: `https://www.twilio.com/whatsapp`

Publication Details



Gmail — Rohit Kokate <rohitkokate1163@gmail.com>

Paper 36 summary
1 message

Microsoft CMT <email@msr-cmt.org> — 27 December 2024 at 22:45
Reply-To: Microsoft CMT - Do Not Reply <noreply@msr-cmt.org>
To: rohitkokate1163@gmail.com

Hello.

Here is submission summary.

Track Name: ICEPE2025

Paper ID: 36

Paper Title: A IoT based Real-Time Garbage Detection System using YOLOv8 and Raspberry Pi

Abstract:
Efficient waste management in urban areas is increasingly challenging as waste production outpaces traditional human sorting methods. To address this issue, we propose a real-time garbage identification system leveraging a Raspberry Pi and a fine-tuned YOLOv8 object detection model. By integrating pre-trained YOLOv8 weights with domain-specific fine-tuning, our approach achieves enhanced detection accuracy for localized waste categories such as metal, batteries, and plastic containers. The system is supported by a Flask-based web application that provides users with a seamless monitoring interface, displaying real-time detection results. Additionally, the system ensures timely waste collection by automatically notifying staff via WhatsApp when garbage is detected. The compact size and affordability of the Raspberry Pi enable scalable deployment in urban areas, paving the way for smarter and more efficient waste management strategies.

Created on: Fri, 27 Dec 2024 17:14:46 GMT

Last Modified: Fri, 27 Dec 2024 17:14:46 GMT

Authors:
- rutujamore785@gmail.com
- samarthjadhav34@gmail.com (Primary)
- rohitkokate1163@gmail.com
- aditi.kadam.d@gmail.com
- patankarnikhilit@sanjivani.org.in

Primary Subject Area: SS-01: AI/ML Technologies based Sustainable Development in Recent Advancement of Research

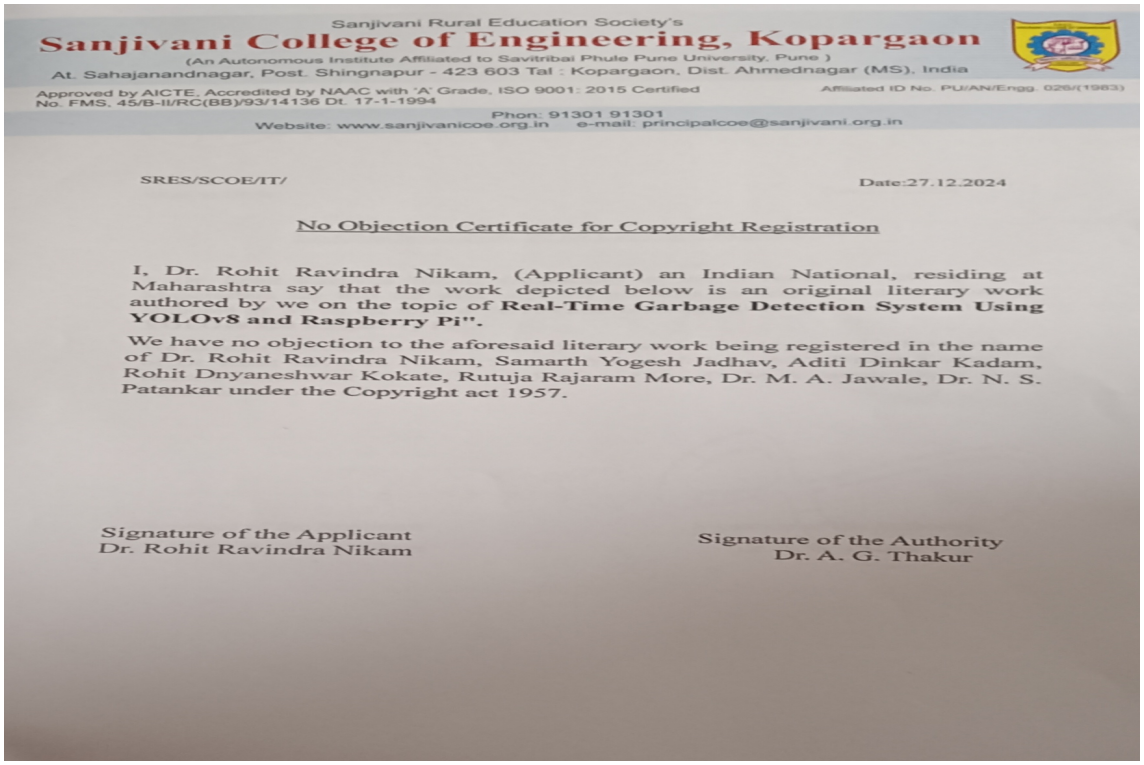
Secondary Subject Areas: Not Entered

Submission Files:
research_p 1.docx (1 Mb, Fri, 27 Dec 2024 17:12:05 GMT)

Figure 9.1: Application Details

Intellectual property rights (IPRs) Details

Intellectual Property Rights (IPRs)



Sanjivani Rural Education Society's

Sanjivani College of Engineering, Kopargaon

(An Autonomous Institute Affiliated to Savitribai Phule Pune University, Pune)

At. Sahajanandnagar, Post. Shingnapur - 423 603 Tal : Kopargaon, Dist. Ahmednagar (MS), India

Approved by AICTE, Accredited by NAAC with 'A' Grade, ISO 9001: 2015 Certified
No. FMS. 45/B-II/RC(BB)/93/14136 Dt. 17-1-1994

Affiliated ID No. PU/AN/Engg. 026/(1983)

Phon: 91301 91301
Website: www.sanjivanicoe.org.in e-mail: principalcoe@sanjivani.org.in

SRES/SCOE/IT/

Date:27.12.2024

No Objection Certificate for Copyright Registration

I, Dr. Rohit Ravindra Nikam, (Applicant) an Indian National, residing at Maharashtra say that the work depicted below is an original literary work authored by we on the topic of **Real-Time Garbage Detection System Using YOLOv8 and Raspberry Pi".**

We have no objection to the aforesaid literary work being registered in the name of Dr. Rohit Ravindra Nikam, Samarth Yogesh Jadhav, Aditi Dinkar Kadam, Rohit Dnyaneshwar Kokate, Rutuja Rajaram More, Dr. M. A. Jawale, Dr. N. S. Patankar under the Copyright act 1957.

Signature of the Applicant
Dr. Rohit Ravindra Nikam

Signature of the Authority
Dr. A. G. Thakur

Figure 9.2: Intellectual Property Rights

Appendices

Include any additional material that supports your report (e.g., code listings, extra data, etc.).

10. Project Timeline

Date Range Task Description

July 20 - July 31 Initial Planning and Research (Completed)

- Defined project goals and requirements.
- Researched hardware components (Raspberry Pi, camera, sensors).
- Explored machine learning algorithms (Python, OpenCV).

August 1 - August 15 Hardware Setup and Initial Prototyping (Completed)

- Assembled hardware components and set up Raspberry Pi, camera, and sensors.
- Tested Raspberry Pi's basic functionality and camera setup.

August 16 - August 31 Data Collection and Processing (Completed)

- Collected and labeled sample images/videos of garbage.
- Preprocessed the data for training.

September 1 - September 15 Model Training and Testing (Completed)

- Trained the machine learning model on labeled data.
- Tested and optimized the model's performance.

September 16 - September 30 System Integration and Development (Completed)

- Integrated the camera feed with the garbage detection model.
- Developed code for real-time garbage detection on Raspberry Pi.

October 1 - October 15 Refinement and Error Handling (Completed)

- Refined the model based on test results and implemented error handling.
- Improved image processing and recognition accuracy.

October 16 - October 31 Automation and Real-Time Feedback (Completed)

- Implemented real-time notifications and alerts for garbage detection.
- Set up automated actions (e.g., triggering a motor to collect garbage).

November 1 - November 15 User Interface Development (Completed)

- Developed a basic user interface to monitor system status.
- Integrated the interface with Raspberry Pi.

November 16 - November 30 Testing and Optimization (Completed)

- Conducted field testing in various environments.
- Optimized the system for power consumption and performance.

December 1 - December 15 Final Integration and Performance Tuning (80% Complete)

- Integrated final components and completed most optimizations.
- Addressed issues from testing feedback but still refining some aspects.
- Improved system performance on Raspberry Pi.

December 16 - December 28 Final Testing, Debugging, and Documentation (80% Complete)

- **December 16 - December 20:** Final field tests and fine-tuning of the model.
- **December 21 - December 23:** Fix bugs and make interface improvements.
- **December 24 - December 26:** Prepare project documentation.
- **December 27 - December 28:** Final system check and project preparation for deployment.

January 2025 Finalization and Deployment (Remaining 20%)

January 1 - January 7 Final Testing and Refinements

- Conduct thorough testing in real-world environments.
- Address any remaining bugs or issues.
- Optimize system performance for long-term use.

January 8 - January 14 Final Integration and Deployment

- Ensure seamless operation of hardware components.
- Finalize software for deployment.
- Set up deployment environment.

January 15 - January 21 Final Documentation and User Manual

- Complete detailed user manuals for installation.
- Finalize project documentation (code, setup, troubleshooting).
- Document known limitations or areas for improvement.

January 22 - January 31 Post-Deployment Testing and Adjustments

- Monitor system performance after deployment.
- Collect user feedback and make adjustments.
- Final review of overall project performance.

Index