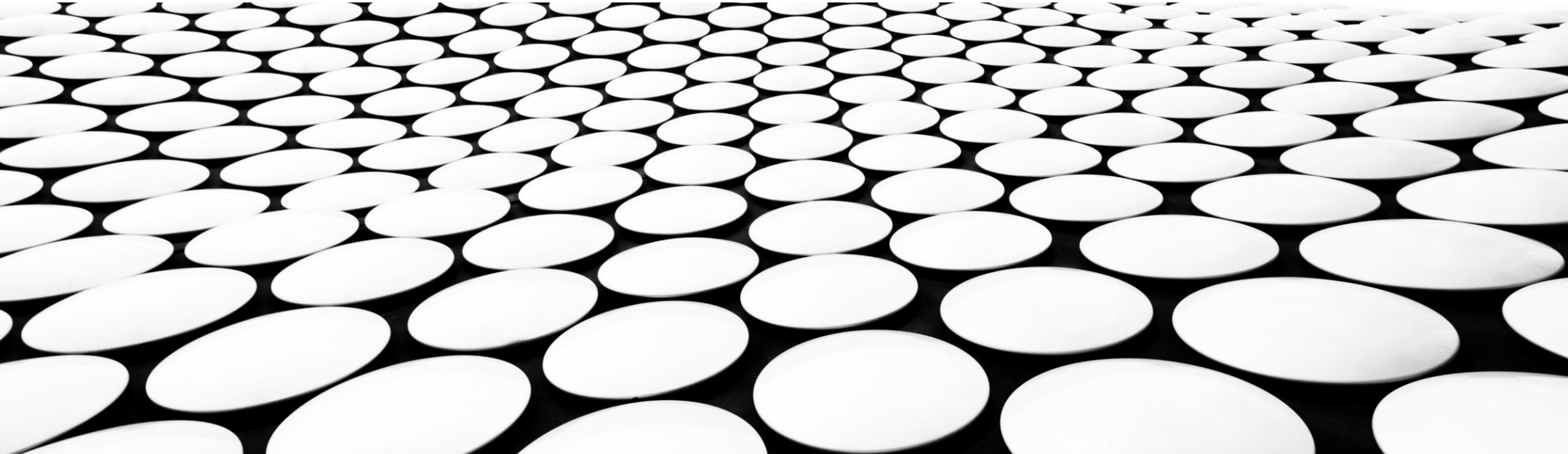
SWACCHHAI: LIVE WASTE SEGREGATION

BY:

AMISHA PATEL

HEMIL SHAH



THE PROBLEM

- The people who are living in backward or rural areas, they tend to have very less knowledge about the Segregation of Waste. They dump all their wastes at a single place, and thereby suffer from various diseases, and other problems.
- For this Problem, we have created SwacchhAI, a Waste Segregation System, which can classify from real-time data. This project would help them segregate their wastes properly, without any support of any person.
- Also, Real-time data would be a positive side of this project, as it does not require the user to take a picture and upload it to the app. The waste can be displayed in front of the camera, and they are good to go.

$$2x - 3x$$

The user can save their time drastically, by this project. They save their time of clicking photos and uploading them. They just need to display the waste in front of the camera.

IDEA BRIEF

- *OBJECTIVE:*

Building SwacchhAI, a Waste Classification Tool, that provides real-time, accurate, and time saving output, classifying the wastes in real-time through the camera of the user.

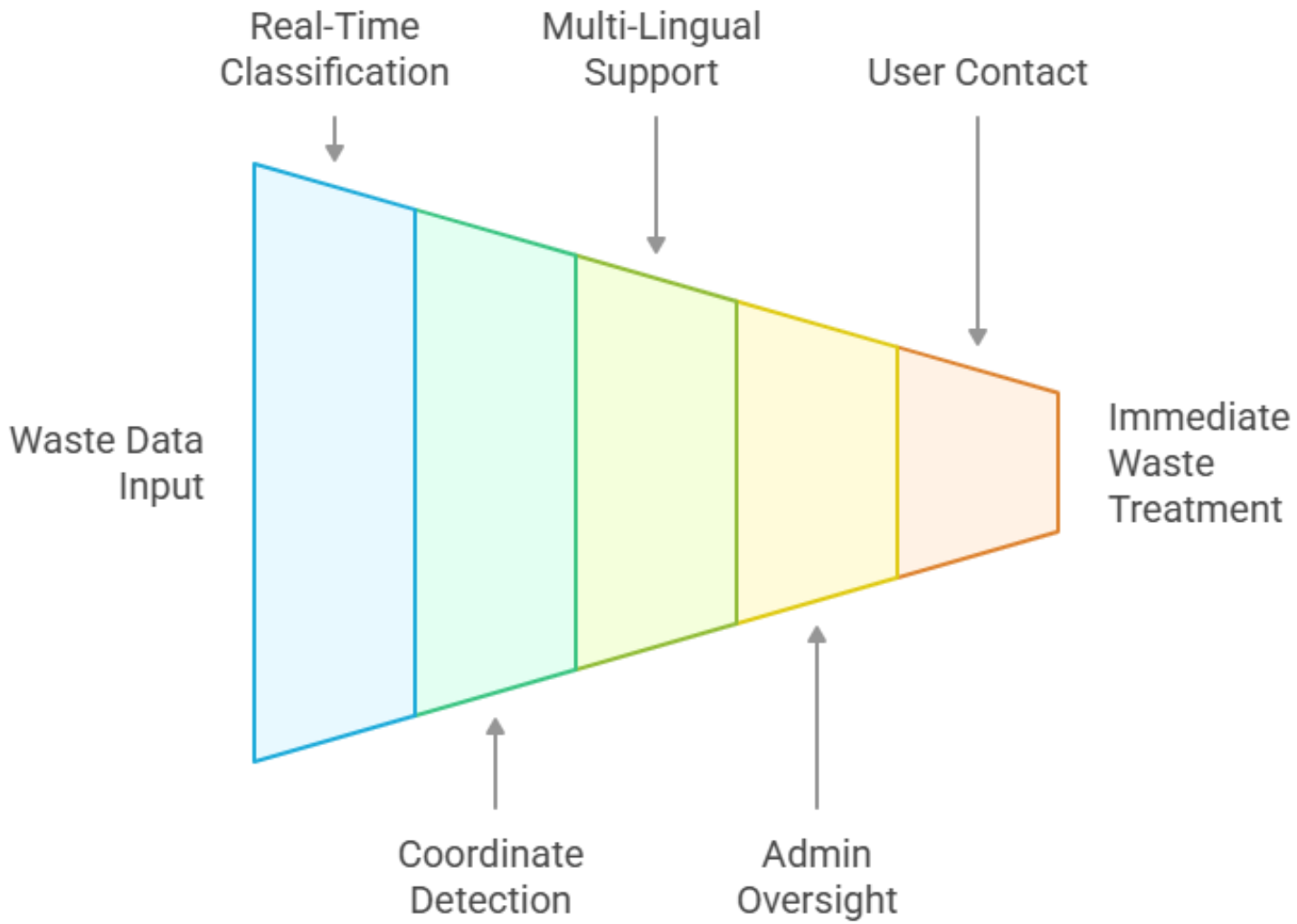
- *The following features will be included:*

- Real-Time Waste Classification
- More waste classes can be identified
- Real-Time Coordinates of the part of the feed, detected as waste
- Multi-Lingual support
- Explanation about the type of waste classified.

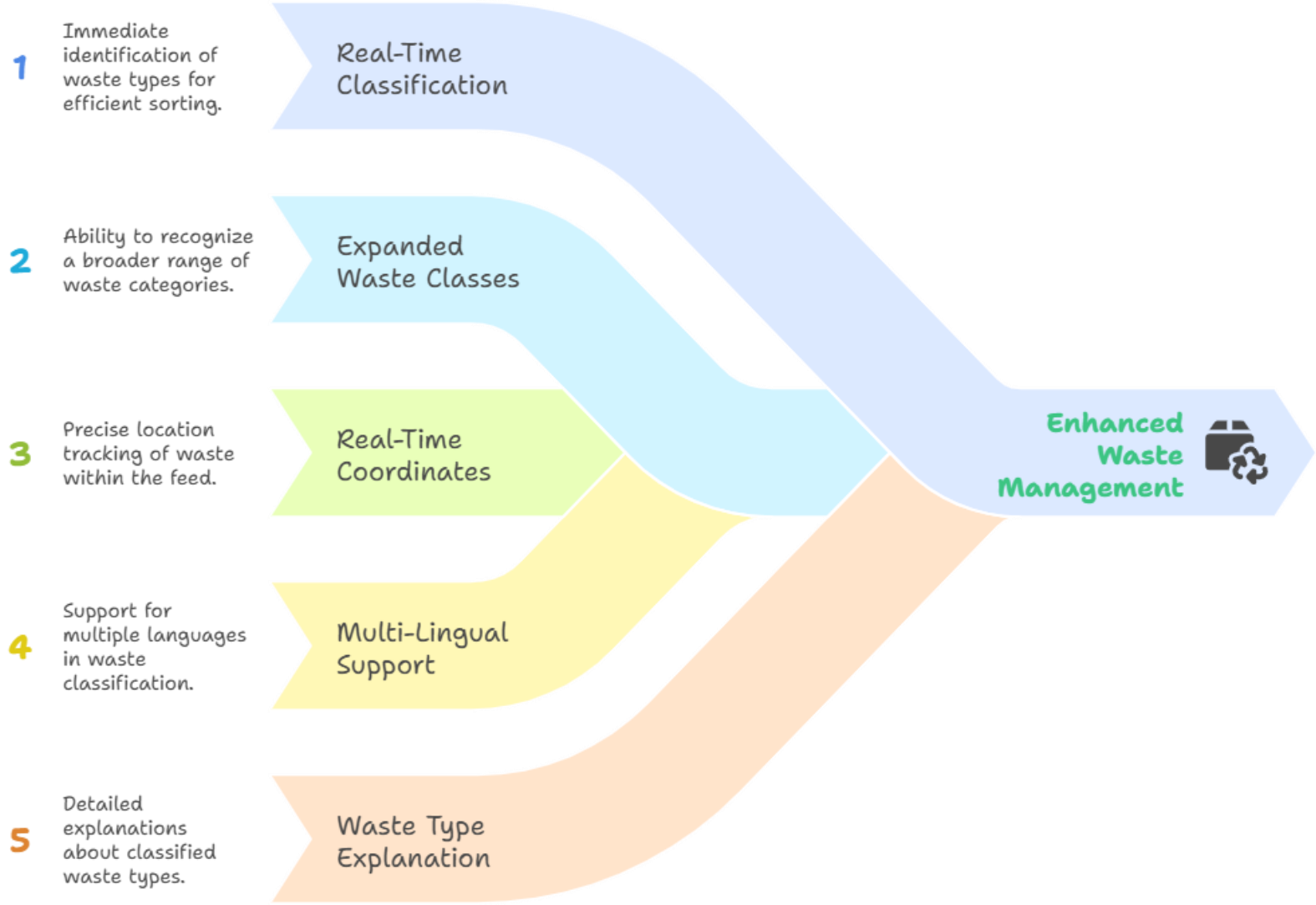
- *The Admin- Side Application:*

- Can keep a record of the users using the tool
- Can have a look at the data being fed by the users.
- Can contact the user, if there is anything suspicious/ dangerous/ hazardous waste is found, and needs treatment immedietly.

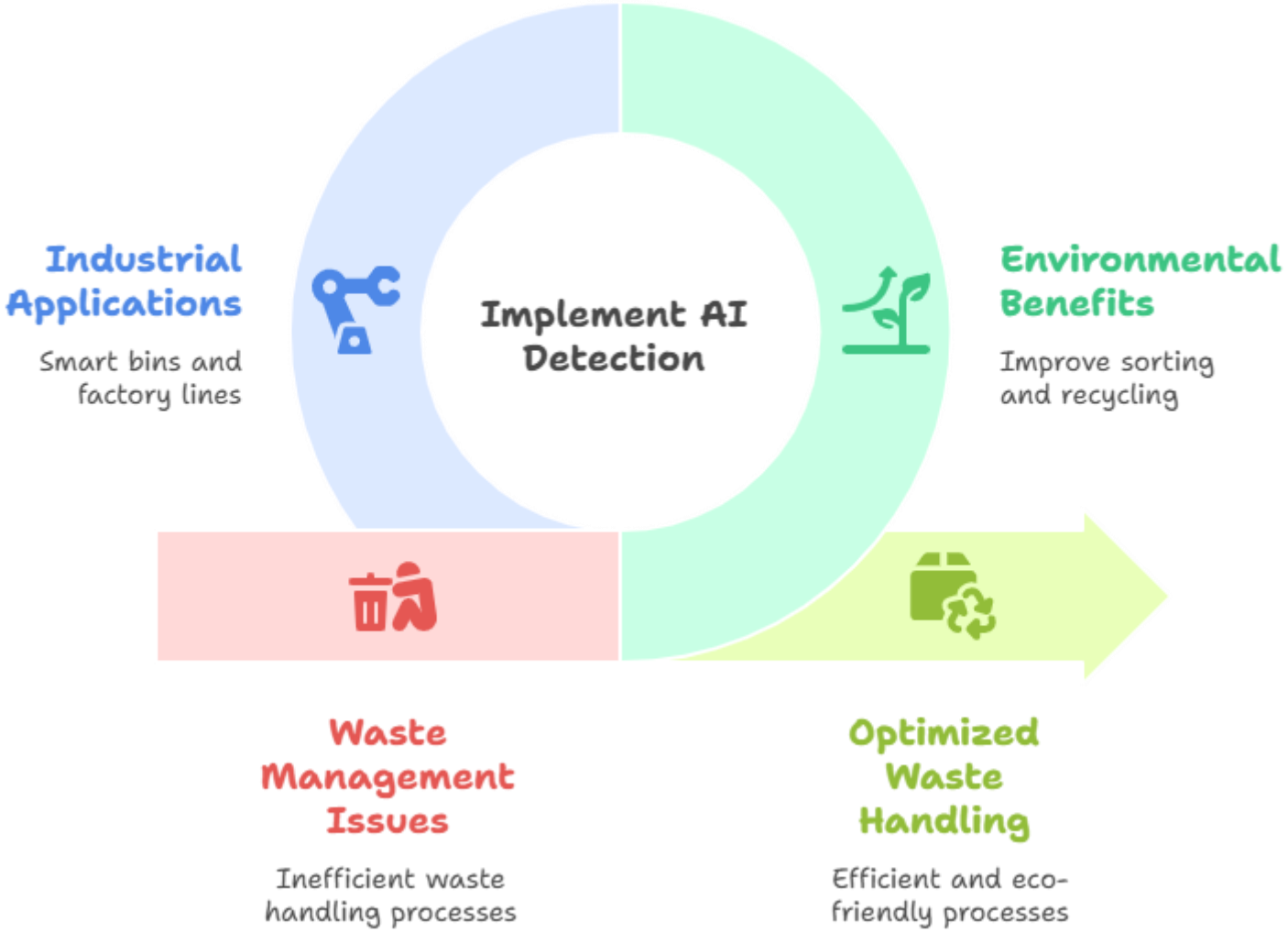
PROCESS FUNNEL:



SOLUTION FEATURES



IMPACT OF THE SOLUTION



STAND- OUT AREAS

Model Retraining Process

Initial Model
Untrained model ready for data

Data Input
Plug in new datasets for training

Model Training
Retrain the model using `train.py`

Retrained Model
Model updated with new data

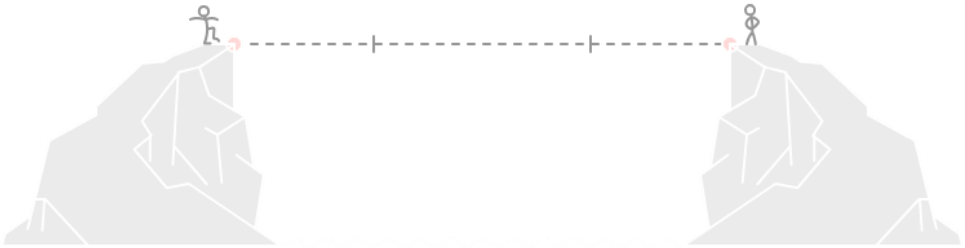
From Image Upload to Prediction Display

Image Upload
Initial image for processing

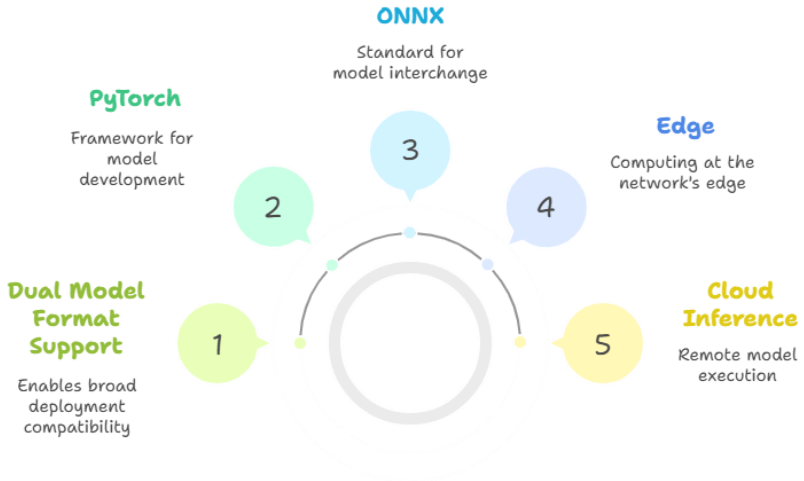
Process Image
Streamlit/Flask UI processes the image

View Prediction
Display the prediction results

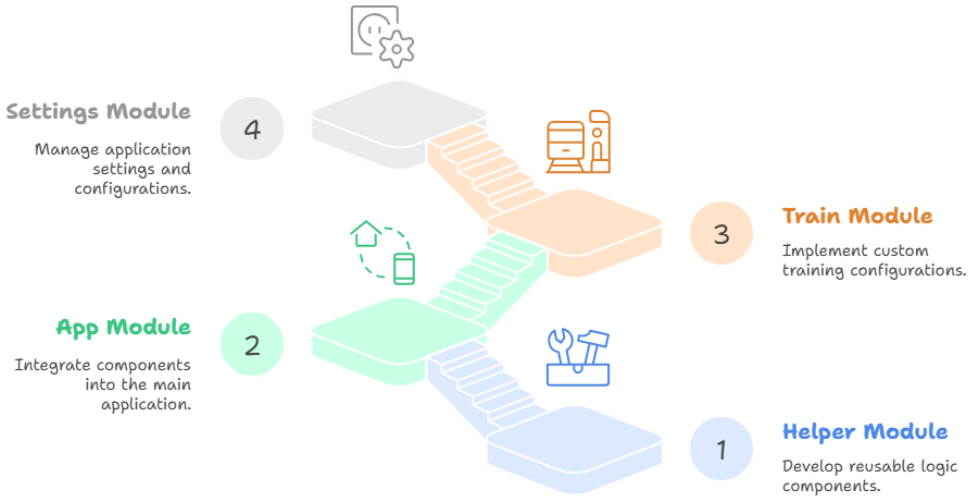
Prediction Displayed
Results shown to the user



Model Deployment Ecosystem



Achieving Modular Architecture



ARCHITECTURE DIAGRAM

01

User Interface

The user interacts with the application interface.

02

Image Input

The user provides an image for processing.

03

Preprocessing Logic

The image undergoes preprocessing steps.

04

Model Inference

The model analyzes the image to make predictions.

05

Prediction Output

The model generates a prediction based on the image.

06

Display/Save UI

The prediction is displayed or saved through the UI.

USER-SIDE INTERFACE

Start Application (UI)

User initiates the application through a Streamlit/Flask interface.

Detect Objects Button

User clicks the button to start the AI detection process.

Waste Classification Logic

The system classifies waste into recyclable, non-recyclable, or hazardous categories.

Optional Console Sidebar

User can access logs and class history in the console sidebar.

Live Webcam Feed Activated

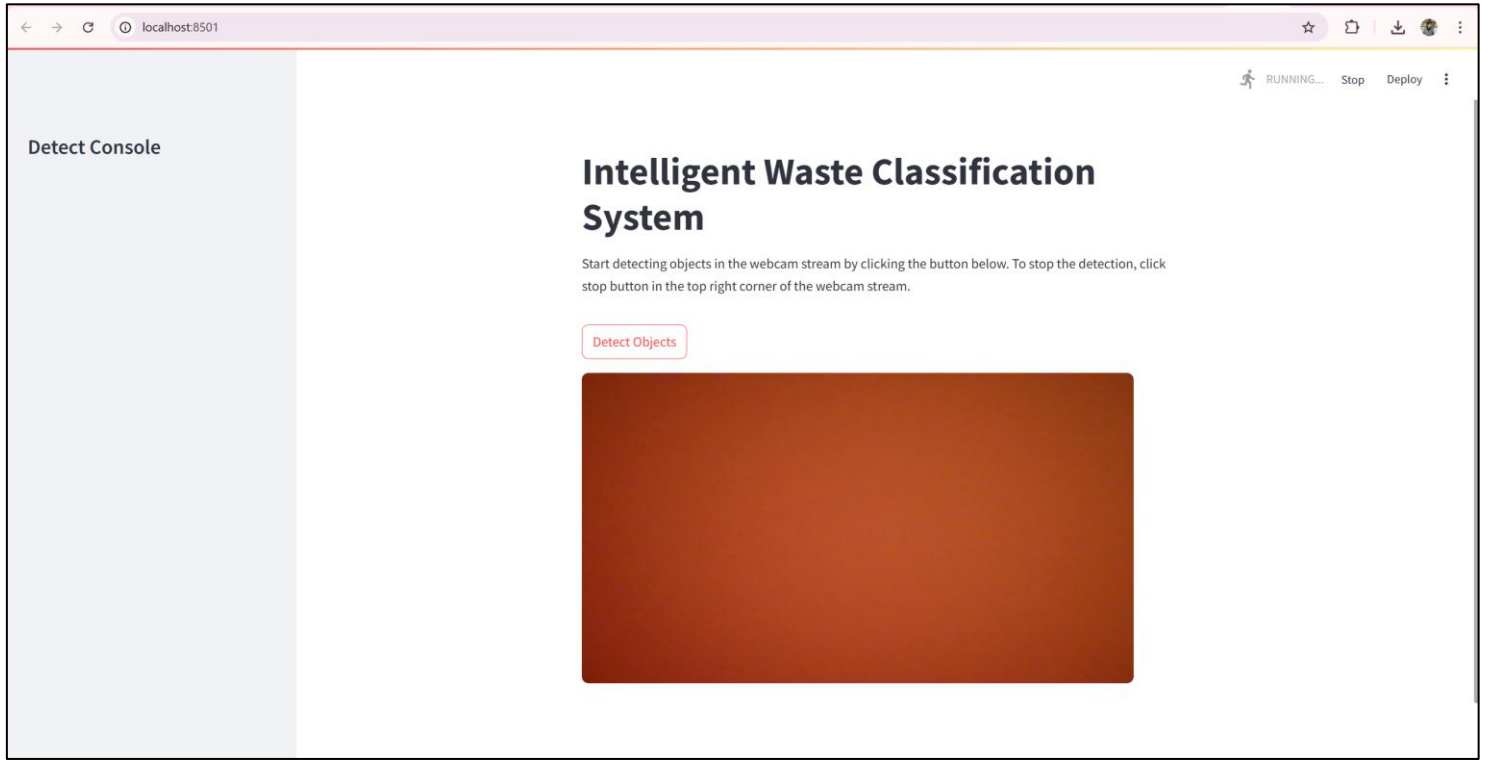
The webcam feed is activated to capture video input.

Real-Time Object Detection

The system uses an ML model to detect objects in real-time.

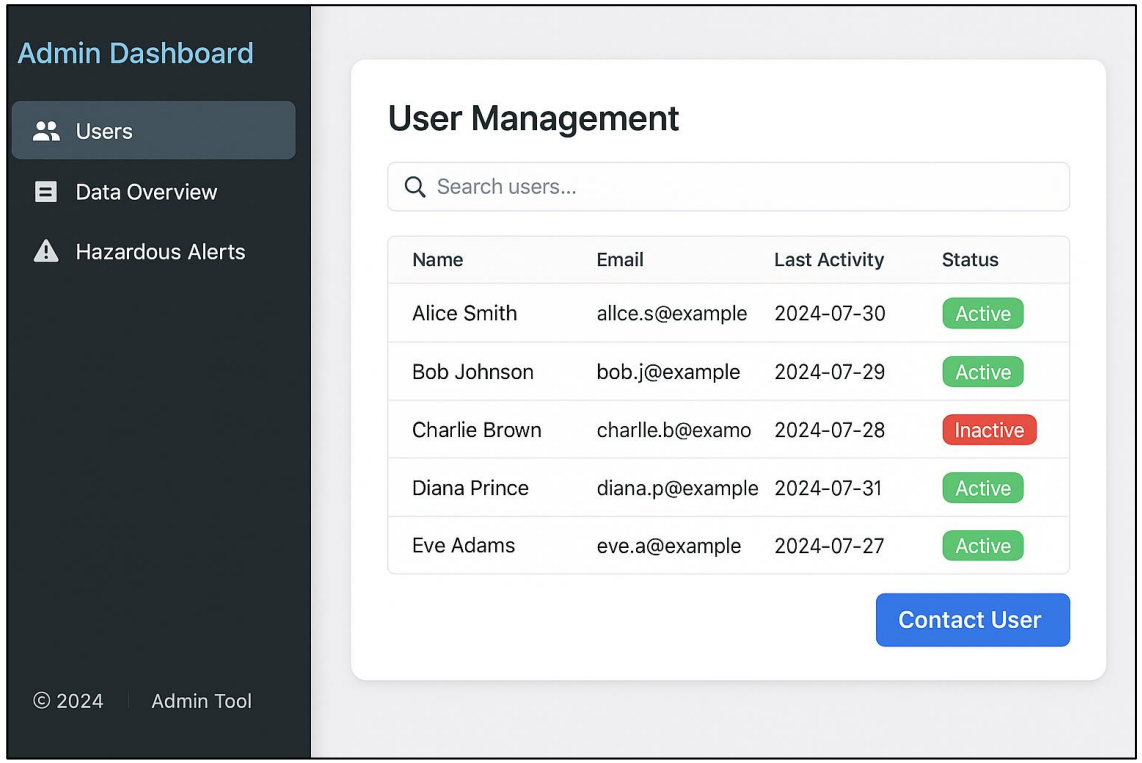
Prediction Output Displayed

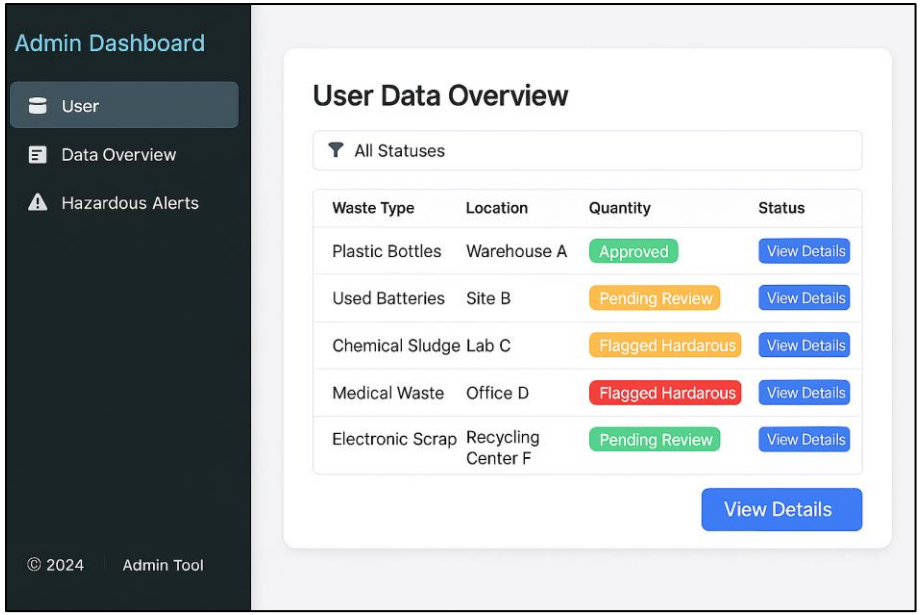
The system displays the prediction output with labels, categories, and accuracy.

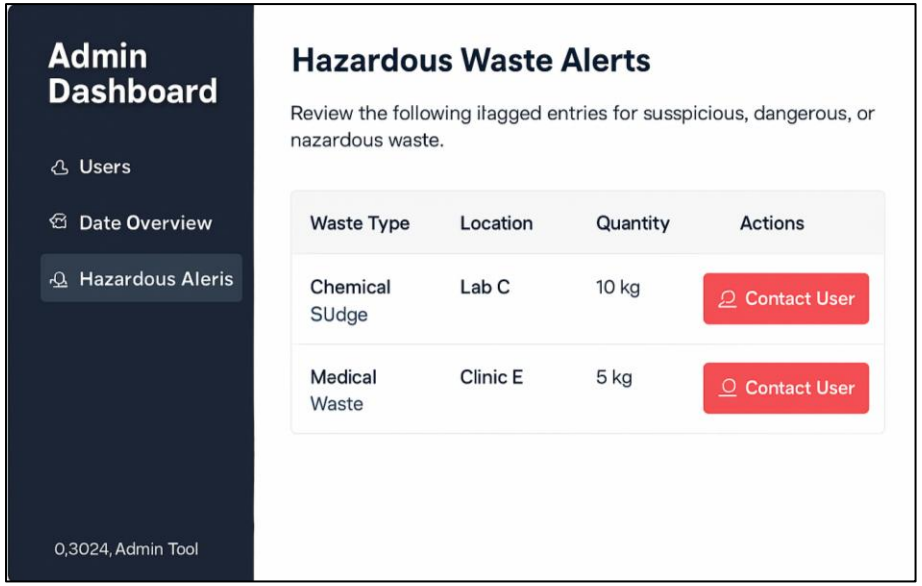


"Smart detection for a cleaner tomorrow."

ADMIN-SIDE INTERFACE







"Smart detection for a cleaner tomorrow."

FUTURE SCOPE

Real-time FPS Counter

Develop a high-precision, low-latency FPS counter fully integrated with GPU/CPU utilization metrics and live overlay visualization for enhanced performance monitoring.

Input Heatmaps / Overlay

Incorporate dynamic, real-time input heatmaps and overlays to visualize model attention regions, improving interpretability and debugging capabilities.

Performance

Implement comprehensive performance monitoring with real-time FPS, GPU/CPU usage stats, latency, and memory consumption metrics presented via dashboards.

Feedback System

Implement an interactive, real-time user feedback loop to collect input, enabling continuous improvement of model accuracy and user experience.

Model Output

Upgrade output visualization to include detailed prediction heatmaps, confidence scores, and multi-class probabilities directly overlaid on input images.

Feedback Loop

Introduce systematic user feedback collection and integration to enable iterative model fine-tuning and adaptation to evolving user needs.

Dockerized Deployment

Containerize the full application stack using Docker for reproducibility; deploy on cloud platforms like HuggingFace Spaces and Streamlit Cloud to enable scalable, seamless access.

Helper Functions

Refactor helper functions into modular, well-documented APIs designed for plug-and-play integration, enhancing code reuse and maintainability.

Deployment

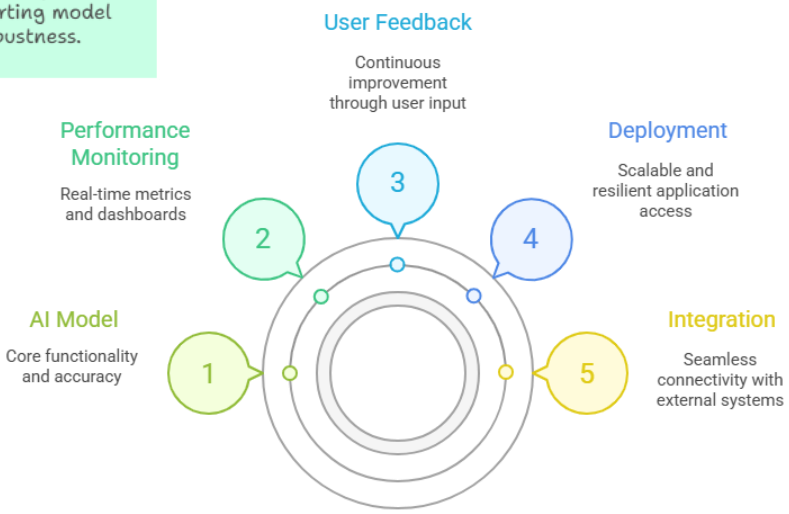
Establish containerized and orchestrated deployments using Docker and cloud-native platforms for high availability, scalability, and resilience.

Integration of Cloud

Expand cloud integration by storing detailed model outputs, logs, and analytics; leverage Streamlit Cloud and other services to enhance accessibility, persistence, and collaboration.

Dataset Handling

Automate dataset validation, augmentation pipelines, and version control to ensure high-quality, traceable data inputs supporting model robustness.




THANK
YOU