# Written Statement on Agentic AI and IBM watsonx Usage

# This document describes how IBM watsonx and agentic AI are implemented in our intelligent waste management app. The focus is on how the AI agent is deployed, how IBM watsonx aids the deployment, and what technical stack underlies the solution.

# 1. Agentic AI Overview

At the core of the solution is an AI-powered waste classifying agent. The user takes a photo of an object and feeds it to the app, and the agent categorizes it as one of the six types of waste (plastic, paper, glass, metal, cardboard, or general waste) in real-time.

Unlike a static model, the agent autonomously handles the entire workflow:  
- preprocessing the uploaded image,  
- securely authenticating with IBM Cloud,  
- sending the request to Watson Machine Learning (WML),  
- parsing the response, and  
- returning a clear classification with confidence scores.

This functionality empowers residents to properly separate waste, increases recycling efficiency, and contributes to more sustainable urban environments.

# 2. Technology Stack

Frontend: HTML, CSS, JavaScript, Leaflet (for interactive maps)  
Backend: Python, Flask  
AI Model: TensorFlow / Keras (image classification model)  
Cloud Platform: IBM Cloud  
Model Deployment: IBM Watson Machine Learning (WML) via watsonx  
Storage: IBM Cloud Spaces (for models, data, and deployment versions)  
Communication: REST API between Flask server and IBM WML endpoint

# 3. IBM Cloud / Watsonx Workflow

**Space Creation**  
A dedicated project space is created in IBM Cloud. It stores datasets, trained models, and deployment versions, ensuring centralized management.  
 **Model Upload**  
The AI model is trained locally with TensorFlow/Keras. After training, it is saved in .h5 format and uploaded to IBM Cloud Spaces for versioning and deployment.  
  
**Deployment**  
The model is deployed through IBM Watson Machine Learning (WML) as an online service. WML provides a secure REST API endpoint along with API keys for authentication.  
 **Application Integration**  
- The Flask backend receives an uploaded image from the frontend.  
- The image is preprocessed (resized, normalized, converted to RGB, and serialized).  
- Flask requests an IAM access token from IBM Cloud using the API key.  
- The encoded image is sent to the WML deployment endpoint via REST API.  
- WML returns a JSON response with prediction values and confidence scores.

**Frontend Display**

The program examines the response and shows the expected waste class and confidence to the user. Future extensions can include automatically showing the nearest recycling center on the interactive map.

# 4. Detailed AI Agent Workflow

The AI agent follows these autonomous steps:  
  
**Image Preprocessing**  
- Convert uploaded images to RGB.  
- Resize to 224×224 pixels.  
- Normalize pixel values to [0,1].  
- Transform into a NumPy array for model compatibility.  
 **Encoding & Transmission**  
The preprocessed array is serialized into Base64 for safe JSON transport.  
 **Authentication**  
Before prediction, the agent requests a Bearer token from IBM IAM using the stored API key. This ensures secure, enterprise-grade communication with WML.  
  
**Prediction Request**The Flask server sends the encoded image to the WML endpoint using a POST request. WML processes the request with the deployed TensorFlow/Keras model.  
  
**Response Handling**  
WML offers probabilities for all the six types of waste: cardboard, glass, metal, paper, plastic, trash. The agent selects the most confident class. The result is returned in the form of a JSON response.

**User Feedback**

The frontend displays the outcome right away, assisting citizens to comprehend the correct disposal category.

This operation reflects the agential nature of the AI: it orchestrates preprocessing, secure authentication, communication, prediction, and feedback without any direct human intervention.

# 5. Implementation Steps

Prepare & Train the AI Model  
- Use TensorFlow/Keras to build and train the waste classification model.  
- Save the trained model in .h5 format.  
  
**Upload & Deploy the Model**- Upload to IBM Cloud Space.  
- Deploy via Watson Machine Learning.  
- Obtain API key and REST endpoint.  
  
**Backend (Flask)**- Accept images from the frontend.  
- Preprocess and encode images.  
- Authenticate with IBM IAM using API key.  
- Send the request to WML and parse the JSON response.  
  
**Frontend (Web App)**- File upload form for waste images.  
- JavaScript sends the image to Flask via fetch().  
- Display classification results and confidence score.  
- Future improvement: highlight nearest recycling point on the interactive map.