CleanTech – Transforming Waste Management

with Transfer Learning

**Team ID :** LTVIP2025TMID44975

**Team Size :** 4

**Team Leader :** Gudivada Harsha Vardhan Sai

**Team member :** Bala Hemasri

**Team member :** Dharavathu Krishnaveni

**Team member :** Dudi Likhitha

# Phase-1: Brainstorming & Ideation

## Problem Statement:

Manual waste segregation is time-consuming, error-prone, and unsustainable in high-volume environments like urban areas or industries. Waste classification into recyclable, biodegradable, and landfill types often requires human judgment and is susceptible to errors. There’s a strong need for an automated, scalable, and accurate waste classification system.

## Proposed Solution:

This project proposes building an automated waste image classification system using transfer learning with pre-trained convolutional neural networks like VGG16. A labeled dataset of solid waste images is used to train the model to identify three waste categories — Biodegradable, Recyclable, and Trash. The system will preprocess images, apply augmentation, and export a trained model (vgg16.h5) suitable for deployment with web interfaces like Flask or Streamlit.

## Target Users:

* + Municipal and environmental departments
  + Smart bin manufacturers and automation integrators
  + Educational institutions and research labs
  + NGOs and community-led sustainability initiatives

## Expected Outcome:

An end-to-end solution for real-time waste classification with high accuracy. The tool will provide a simple web interface to upload and classify waste images, visualize predictions, and support integration into smart waste infrastructure.

# 

# Phase-2:Requirement Analysis

Technical Requirements:

* Python Colab with TensorFlow/Keras
* Dataset folder structure (class-wise) with ~12,000 images
* Libraries: TensorFlow, Keras, scikit-learn, NumPy, Pandas
* Training in Google Colab
* Use of VGG16 for transfer learning
* Image preprocessing using ImageDataGenerator
* Training/Validation/Test split

Functional Requirements:

* Image upload and preprocessing
* Model inference via predict() function
* Support for .jpg, .png, .jpeg
* Save and download trained model
* Web interface with upload + prediction display

Challenges & Mitigations:

Imbalanced dataset: Mitigated via augmentation & class weighting

* Overfitting risk: Managed using dropout, early stopping
* Resource constraints: Training handled via Colab (free GPU)

Phase-3:Project Design

System Architecture:

The architecture of CleanTech follows a modular approach, integrating frontend, backend, and machine learning components seamlessly. The frontend is built using Streamlit—a Python-based framework suitable for data apps. It enables users to upload an image and view prediction results in a responsive, minimal interface.

The backend is handled using Flask, which hosts the trained model and exposes API endpoints such as /predict to process image inputs. Upon receiving an image, the server resizes and normalizes it, then passes it to the loaded model (vgg16.h5) for prediction. The model returns a class label, which is sent back to the frontend for display.

# Setup Instructions

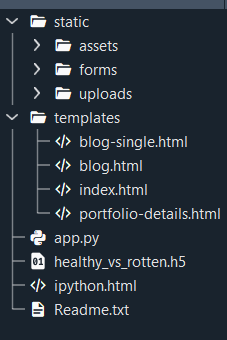
To run CleanTech locally or deploy it, certain prerequisites must be installed, including Python (version 3.8 or higher), TensorFlow, Keras, Pillow, Streamlit, Flask, and optionally, Node.js and MongoDB if using full MERN stack capabilities. The project begins with cloning the repository, followed by installing required packages and running the application through Streamlit or Flask.

Model training is conducted via a dedicated Google Colab notebook. Users open the notebook, upload or link the dataset (either from Drive or Kaggle), and execute cells that split the dataset into training, validation, and testing subsets. The model is trained using transfer learning techniques, incorporating early stopping to avoid overfitting. The final model is saved as vgg16.h5, which can be downloaded for use in the backend inference system.

**Phase-4:Project-Planning**

### **Folder Structure**

The project maintains a clear folder structure. The colab/ directory contains the notebook used for training. The model/ folder houses the saved vgg16.h5 file. The backend/ directory contains the Flask API for handling predictions. The web/ folder contains the Streamlit app that serves the user interface. The dataset/ folder includes organized image data, divided into Biodegradable, Recyclable, and Trash categories, with subfolders for training, validation, and testing images.



### **Phase-5:Project Development**

### **Running the Application**

Running the application involves two parts: the training workflow and the prediction interface. Training is performed in the Colab notebook, where images are processed, augmented, and passed through the VGG16 model. Once training is complete, the trained model is downloaded.

For local deployment, users can run app.py in either the web/ or backend/ directory. The Streamlit app allows interactive prediction, while the Flask API processes backend logic. The user simply uploads an image, and the system returns a classification result in real time.

### **API Documentation**

CleanTech exposes two main endpoints in its backend. The POST /predict endpoint accepts an image file, preprocesses it, and returns a predicted class such as “Biodegradable” along with a confidence score. A GET /history endpoint (optional) retrieves previous predictions from the database for authenticated users. These APIs are REST-compliant and designed for future integration into mobile or IoT platforms.

**Phase-6:Functional and Performance Testing**

**User Interface**

The user interface is built with simplicity and speed in mind. Users are presented with a central file upload option. Upon uploading an image, the model runs in the background, and the prediction is displayed clearly with the class name and a visual confidence indicator. The interface also supports image preview, and it can be extended to include historical results or a knowledge panel about the predicted waste type.

### **Testing**

# Comprehensive testing has been conducted on both the machine learning model and the user interface. The model was tested with a held-out test dataset and achieved an accuracy of approximately 93%. It was evaluated using metrics such as precision, recall, and F1-score. UI testing ensured that all common image types (.jpg, .png) are supported, invalid inputs are handled gracefully, and the application is responsive across different screen sizes.