Full Stack Development with MERN Project Documentation format

# Introduction

* + **Project Title:** CleanTech - Transforming Waste Management with Transfer Learning
  + **Team Members:**

Veda Bhavishya Gudivaka – Project development, tesing and documentation

Satwika Saladi

Varri Charishma

Yerva Nagasri

# Project Overview

* + **Purpose:** The primary goal of CleanTech is to reduce human error, increase waste processing speed, and encourage better recycling practices by automating the waste classification process using Transfer Learning.
  + **Features:**
    - AI-powered waste classification system using transfer learning.
    - Identifies and categorizes municipal solid waste into Biodegradable, Recyclable, and Trash using pre-trained convolutional neural networks.
    - Web-based application for waste image input and classification.
    - Displays classification results.

# Architecture

* + **Frontend:** The user interface for the web-based application is built using HTML and CSS. It includes an upload page and a result page that displays the prediction label and image.
  + **Backend:** The backend architecture utilizes Flask to receive and preprocess uploaded images, and to return results to the frontend.
  + **Database:** The primary report mentions "Terminal logs showing predictions", implying local file system storage for logs rather than a dedicated database. There is no explicit mention of a separate database schema or interactions in the provided CleanTech report.

# Setup Instructions

* + **Prerequisites:** Python, TensorFlow, Keras, Flask, OpenCV, NumPy, Jupyter (for model development).
  + **Installation:** Clone the repository git clone repo\_name, pip install requiremennts.txt and then run python app.py

# Folder Structure

* + **Client:**

Templates/

-Index.html

-About.html

-Contact.html

-Predict.html

-Result.html

* + **Server:**

Static/

-styles.css

App.py

waste\_classifier\_model.h5

# Running the Application

* + Provide commands to start the frontend and backend servers locally.
    - **Frontend:** Auto-served via Flask's templating system.
    - **Backend:**

Run using:  
python app.py  
Open browser: http://127.0.0.1:5000

# API Documentation

**Endpoint:** /predict  
**Method:** POST  
**Request:** multipart/form-data with image file  
**Response:** Rendered HTML page showing prediction label and uploaded image

# Authentication

No authentication implemented. Future improvements may include login/signup and role-based access for users and admins.

# User Interface

# Styled using style.css - Responsive layout with centered buttons, flexbox containers, and image previews - Navigation bar includes Home, About, Predict, Contact - Predict page accepts image upload and displays result visually

# Testing

# Manual testing with positive and negative test cases - Verified edge cases like unsupported file types and empty uploads - UAT document created for validation and bug tracking

# Screenshots or Demo

# 

# Known Issues

- Classification may fail on low-quality images  
- No history/logs stored for predictions  
- Requires stable internet if deployed via external server

# Future Enhancements

- Add database support (MongoDB)  
- Integrate real-time camera feed classification  
- Deploy via Firebase Hosting + Google Cloud Run  
- Add user login system with activity tracking  
- Mobile app version