AI&ML Project Documentation

**Project Title:**

**CleanTech: Transforming Waste Management with Transfer Learning**

**Team ID** **:** LTVIP2025TMID59562

**Team Members :**

* **Vadlamudi Enosh** – Data Collection and Cleaning , Model Design and Training , Evaluation and Optimization , Deployment and Integration
* **Sriramadasu Shivani**
* **Tamarana Jnanesh**
* **Ullamgunta Ambika Krishna**

**Faculty Mentor(s) Name** **: Dr. M. Ganesh**

**Project Overview**

**Purpose:**

* The purpose of this project is to build a deep learning-based image classification model to automatically identify different types of municipal waste from images. By leveraging transfer learning with VGG16, this project addresses real-world challenges in waste segregation and environmental sustainability. It aims to provide an AI-powered solution that classifies waste into categories like biodegradable, recyclable, and trash, helping streamline waste management and promote eco-friendly practices.

**Goals:**

* Automate the classification of municipal waste using image-based AI.
* Improve the efficiency of waste sorting through smart technology.
* Raise awareness and assist organizations in managing waste responsibly.
* Deliver a lightweight, deployable model that can be used in real-world scenarios such as recycling centers, smart bins, and educational platforms.

.

**Key Features:**

* Accurate classification of waste images into three categories: biodegradable, recyclable, and trash.
* Transfer learning using a pre-trained VGG16 model for high performance on a limited dataset.
* Simple web interface for uploading and predicting waste categories.
* Flask-powered backend integrated with the trained deep learning model.
* Easy-to-use, responsive design requiring no technical expertise from users.

**Prerequisites:**To complete this project, the following tools, libraries, and knowledge are required:

**Software Requirements:**

To build and run the CleanTech: Waste Classification project, the following tools and environments were used:

* **Google Colab**

Used for model training, data preprocessing, and running deep learning code in a GPUaccelerated environment.

https://colab.research.google.com

* **Anaconda Navigator**

For managing local Python environments. [**https://www.anaconda.com/products/distribution**](https://www.anaconda.com/products/distribution)

* **Python 3.x *(preferably 3.8 or above)***
* **Visual Studio Code (VS Code)** – A lightweight, powerful code editor for writing and running Python, Flask, and HTML code.
* **Python Extension for VS Code** – Enables Python development, Flask debugging, and environment management within VS Code.

https://code.visualstudio.com

* **Flask**

Used to develop the web application interface.

**Python Libraries Used:**

Install these using Anaconda Prompt or inside your Colab notebook: pip install numpy pip install pandas pip install scikit-learn pip install matplotlib pip install seaborn pip install tensorflow pip install flask

pip install pillow

**Prior Knowledge Required**

To effectively contribute to or understand this project, the following foundational topics and resources are essential:

* **Deep Learning Concepts**

Overview of deep learning and how neural networks operate.  [https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-typesof-neural-networks-in-deep-learning/](https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/)

* **Deep Learning Frameworks**

Detailed comparison between PyTorch and TensorFlow frameworks. <https://www.knowledgehut.com/blog/data-science/pytorch-vs-tensorflow>

* **Transfer Learning**

Learn how to use pre-trained models like VGG for new tasks.  [https://towardsdatascience.com/a-demonstration-of-transfer-learning-of-vggconvolutional-neural-network-pre-trained-model-with-c9f5b8b1ab0a](https://towardsdatascience.com/a-demonstration-of-transfer-learning-of-vgg-convolutional-neural-network-pre-trained-model-with-c9f5b8b1ab0a)

* **VGG16 Architecture**

Explanation of VGG16 architecture used in this project. <https://www.geeksforgeeks.org/vgg-16-cnn-model/>

* **Overfitting & Regularization**

Learn how to avoid overfitting in deep learning models using regularization.  [https://www.analyticsvidhya.com/blog/2021/07/prevent-overfitting-using-regularizationtechniques/](https://www.analyticsvidhya.com/blog/2021/07/prevent-overfitting-using-regularization-techniques/)

* **Deep Learning Optimizers**

Overview of optimizers like Adam, SGD, and RMSprop. [https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deeplearning-optimizers/](https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/)

* **Flask Basics (for Deployment)**

Beginner-friendly tutorial to create web apps with Flask. <https://www.youtube.com/watch?v=lj4I_CvBnt0>

**Project Objectives:**

By the end of this project, you will:

* Know fundamental concepts and techniques used for Deep Learning.
* Gain a broad understanding of data.
* Have knowledge of pre-processing the data/transformation techniques on outliers and some visualization concepts.

**Project Flow :**

1. **User uploads an image** through the web interface.
2. The image is **processed by a trained deep learning model** integrated with a Flask backend. 3. The **prediction result is displayed** on the UI.

**Activities Overview:**

1. **Data Preparation:**

Dataset collection, preprocessing, augmentation, and train-test splitting**.**

1. **Model Building:**

Import libraries, initialize and train the model, evaluate performance, and save the trained model.

1. **Application Development:** Create the HTML UI and build the Flask-based Python backend for prediction**.**

**Project Structure:**

w\_flask/

├── dataset/

├── biodegradable/

├── recyclable/

├── trash/

├── static/

│ ├── assets/

│ │ ├── bgavif

│ │ └── bgjpg.jpg

│ ├── form/

│ ├── uploads/

│ └── templates/

│ ├── blog-single.html

│ ├── blog.html

│ ├── index.html

│ └── portfolio-details.html

├── tf\_env/

├── .gitattributes

├── .renderyaml.txt

├── app.py

├── cnn\_training\_script.py

├── demo video.mp4

├── healthy\_vs\_rotten.h5

├── ipython.html

├── readme.txt

└── requirements.txt

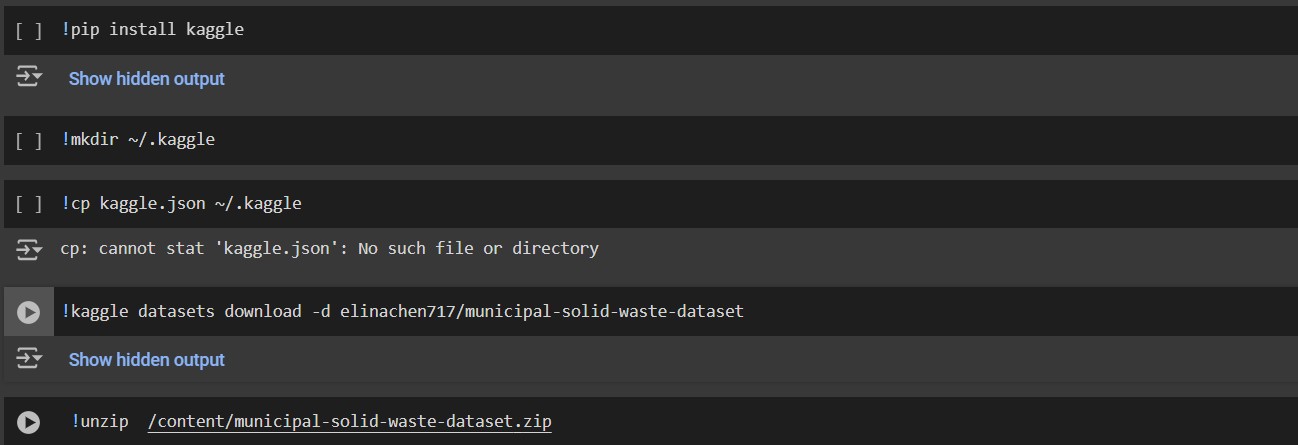
**Data Collection and Preparation:**

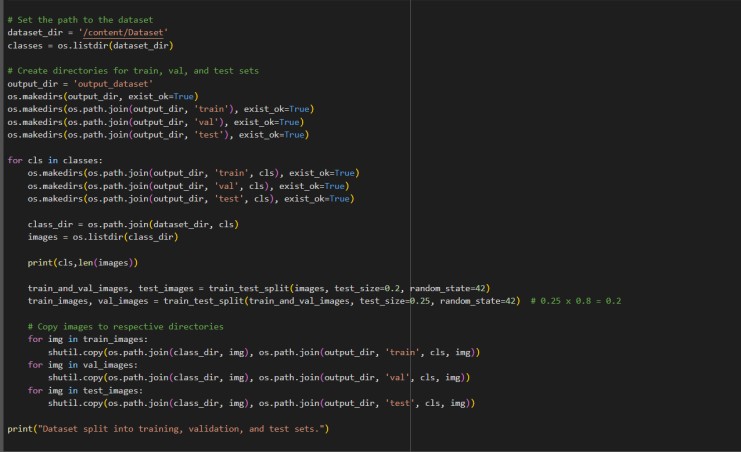
ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

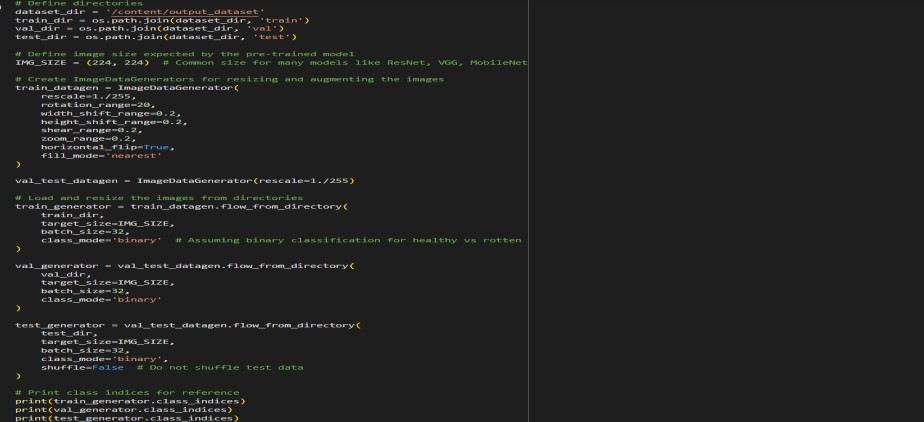
**Municipal Solid Waste Dataset – Kaggle**

<https://www.kaggle.com/datasets/elinachen717/municipal-solid-waste-dataset>

This dataset includes labeled images under categories such as **Recyclable**, **Biodegradable**, and **Trash**, making it ideal for training a waste classification model. The collected data was organized into separate folders by class and reviewed for quality before preprocessing.







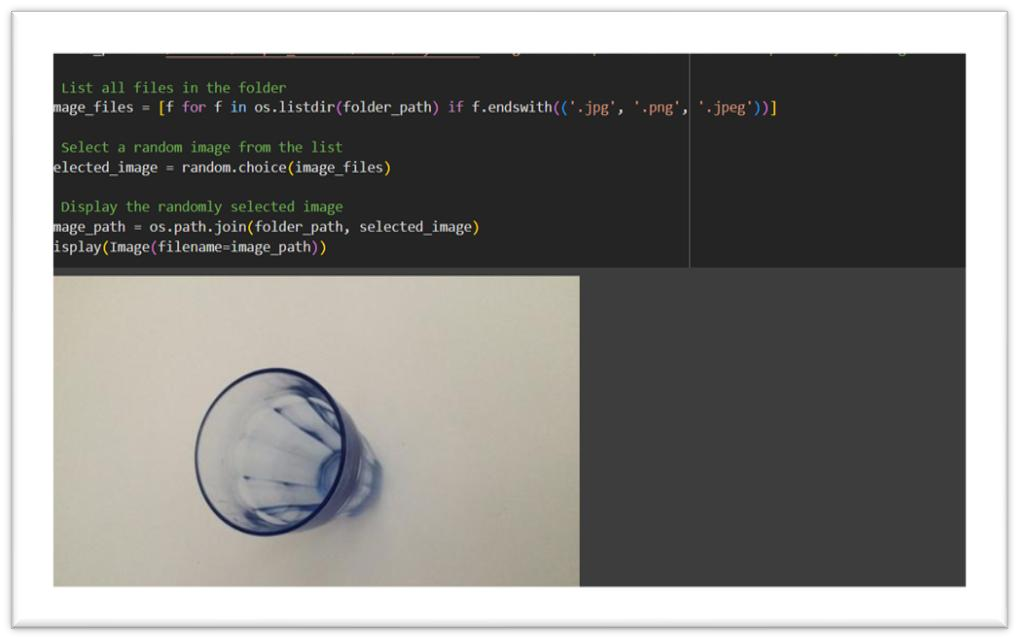
**Data Visualization:**

The dataset was visually explored using Python. A script was used to randomly select and display images from each category using the IPython.display module. This helped ensure data consistency and quality before training.

The code imports necessary libraries, picks a random image from a given folder

(e.g., Trash, Biodegradable, or Recyclable), and displays it for visual inspection. This step supports better understanding of image distribution and content during early development.





**Data Augmentation:**

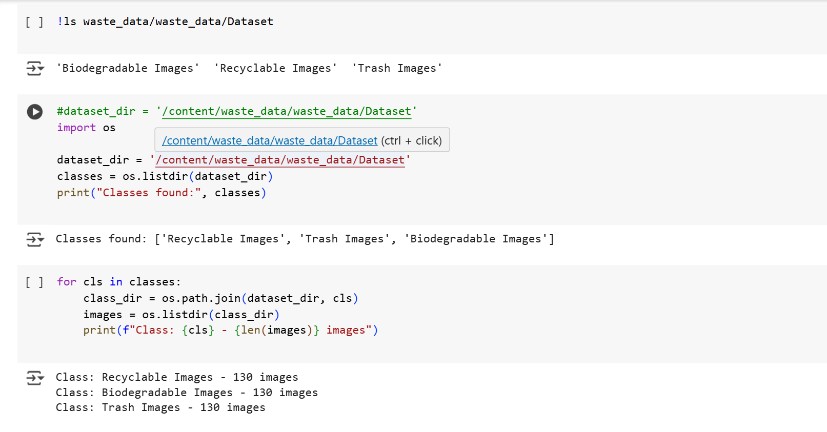
Data augmentation is a common technique used to increase dataset size and diversity by applying transformations like rotation, flipping, scaling, and brightness adjustment to images. It helps improve model generalization, especially when labeled data is limited.

In this project, augmentation was considered but ultimately skipped, as the dataset was already preprocessed and cropped. While skipping augmentation slightly increased training time, it did not significantly affect model accuracy.

**Split Data and Model Building:**

**Train-Test-Split:**

In this project, we have already separated data for training and testing.



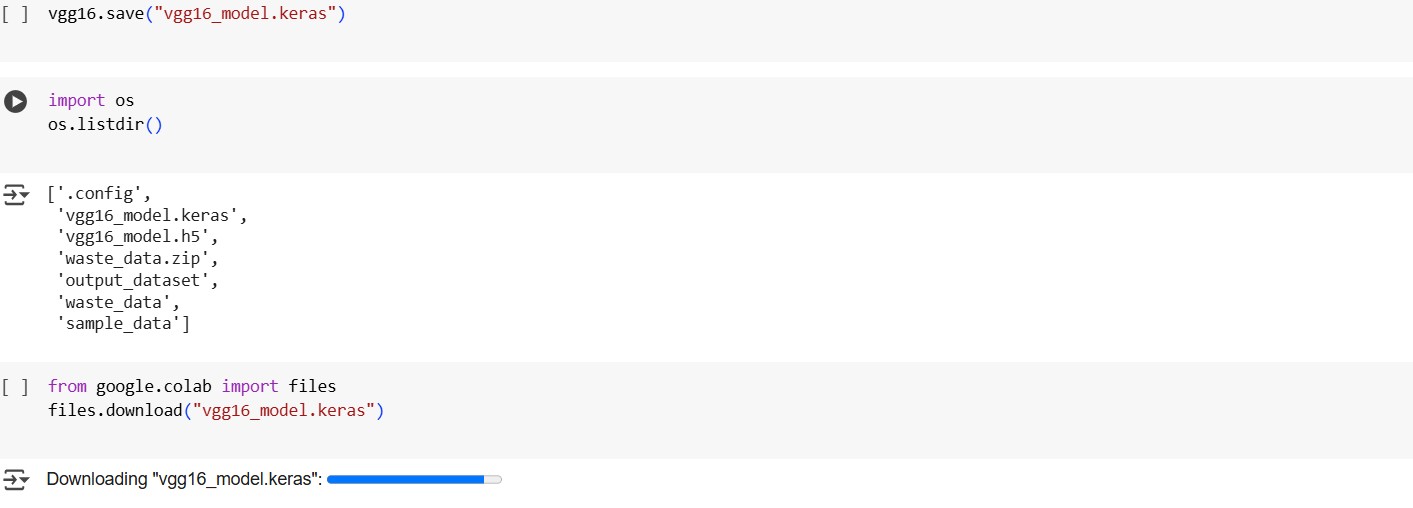
**Testing Model & Data Prediction**

Here we have tested with the Vgg16 Model With the help of the predict () function.

**Saving the model:**

Finally, we have chosen the best model now saving that model





**Application Building:**

In this section, we will be building a web application that is integrated into the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

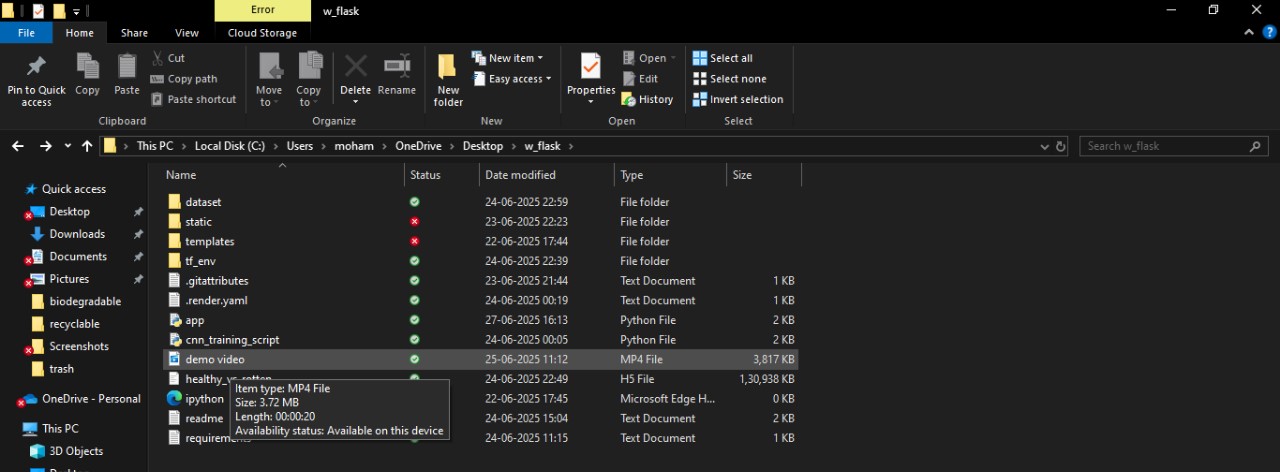
This section has the following tasks

Building HTML Pages Building server-side script

**Building HTML Pages:**

For this project create three HTML files namely

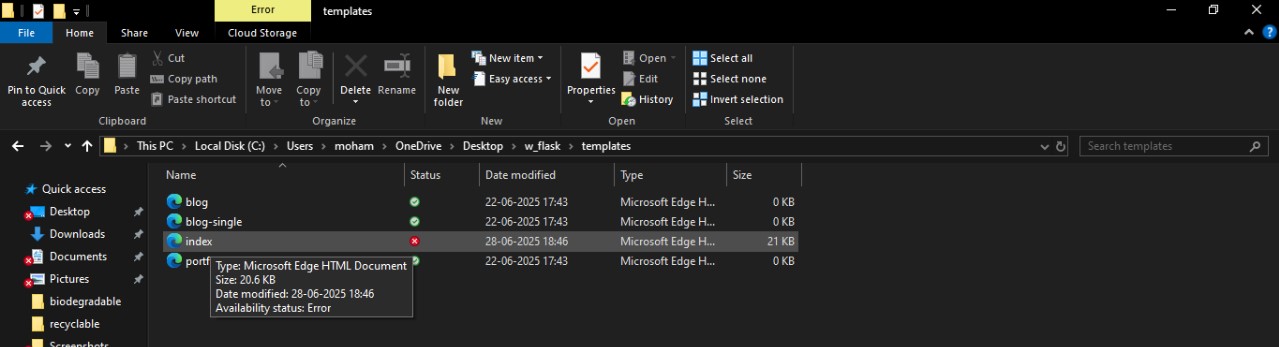
• home.html • result.html Building HTML Pages:



For this project create three HTML files namely

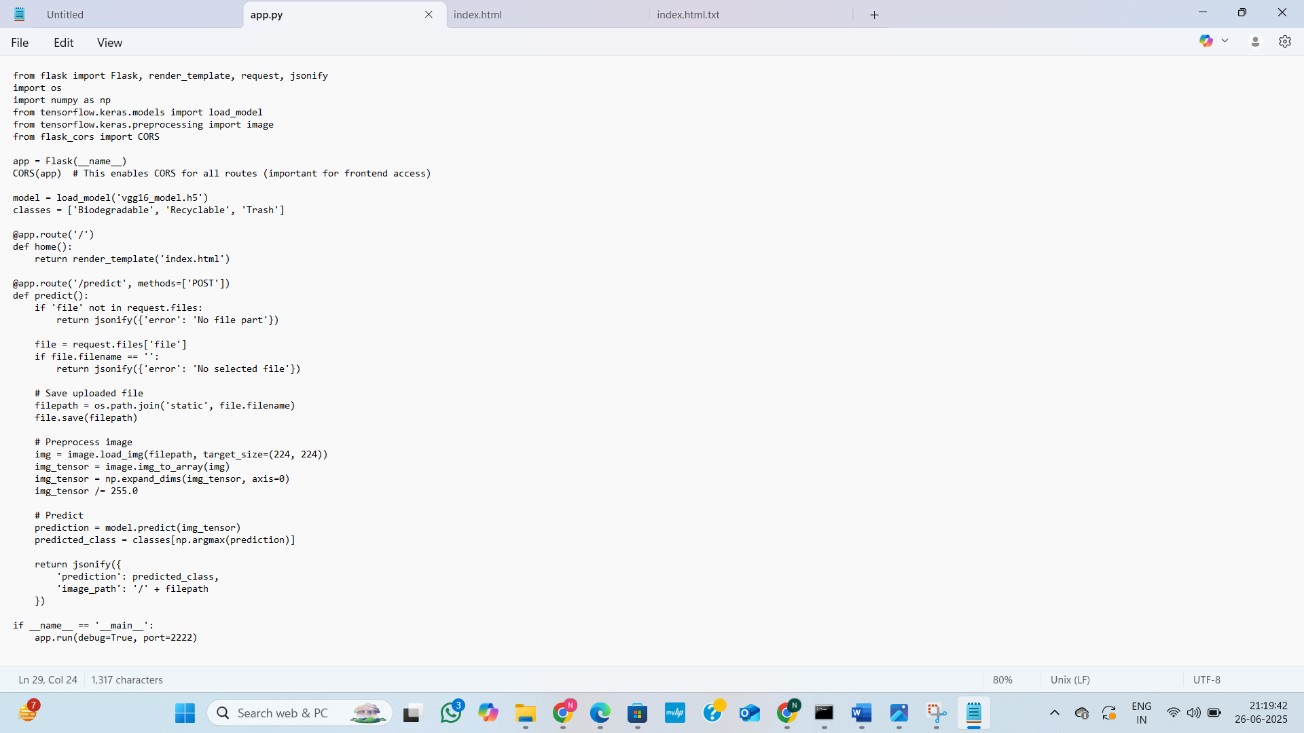
▪ index.html ▪ portofolio\_details.html

And save them in the templates folder.



**Build Python code:**

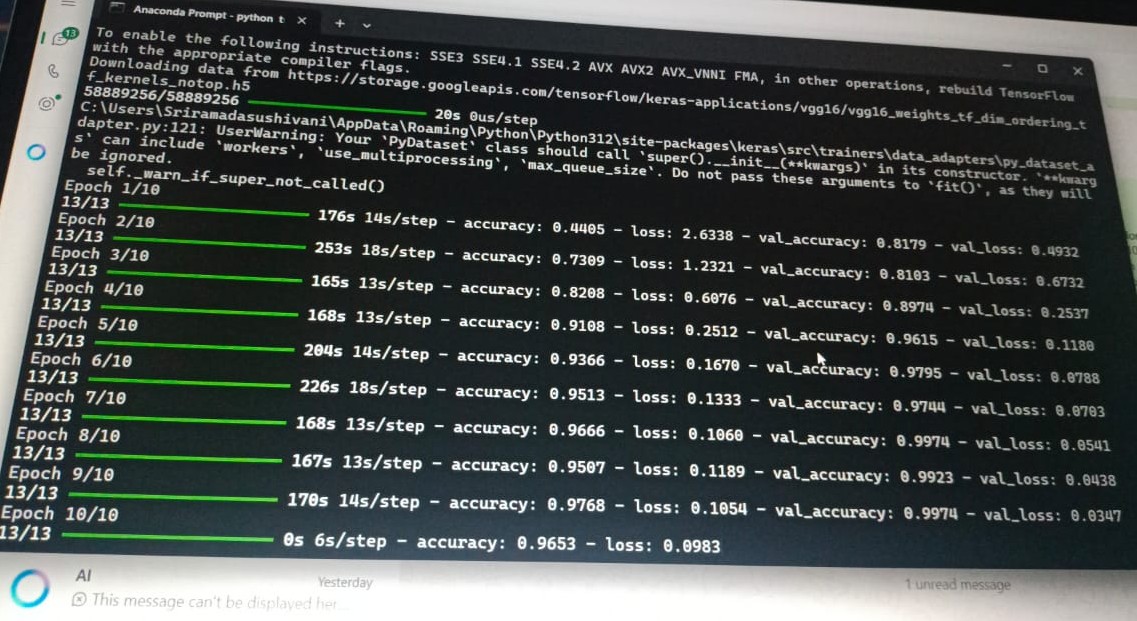
The development of python code is in the form **app.py**

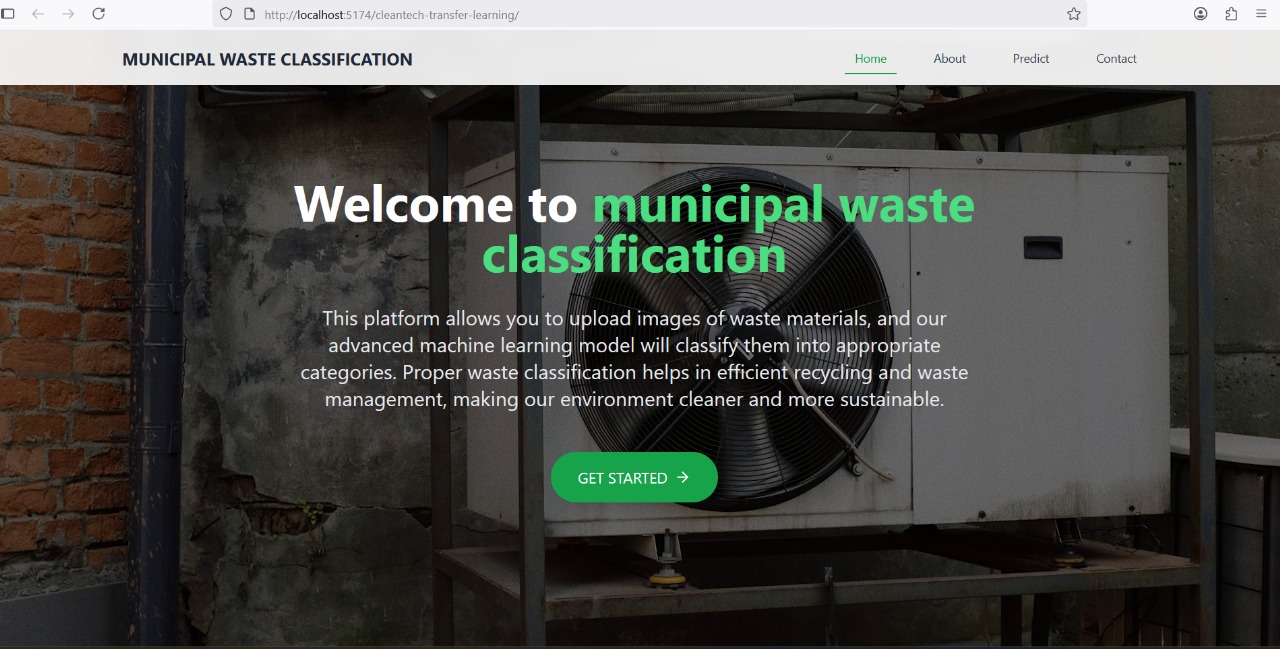


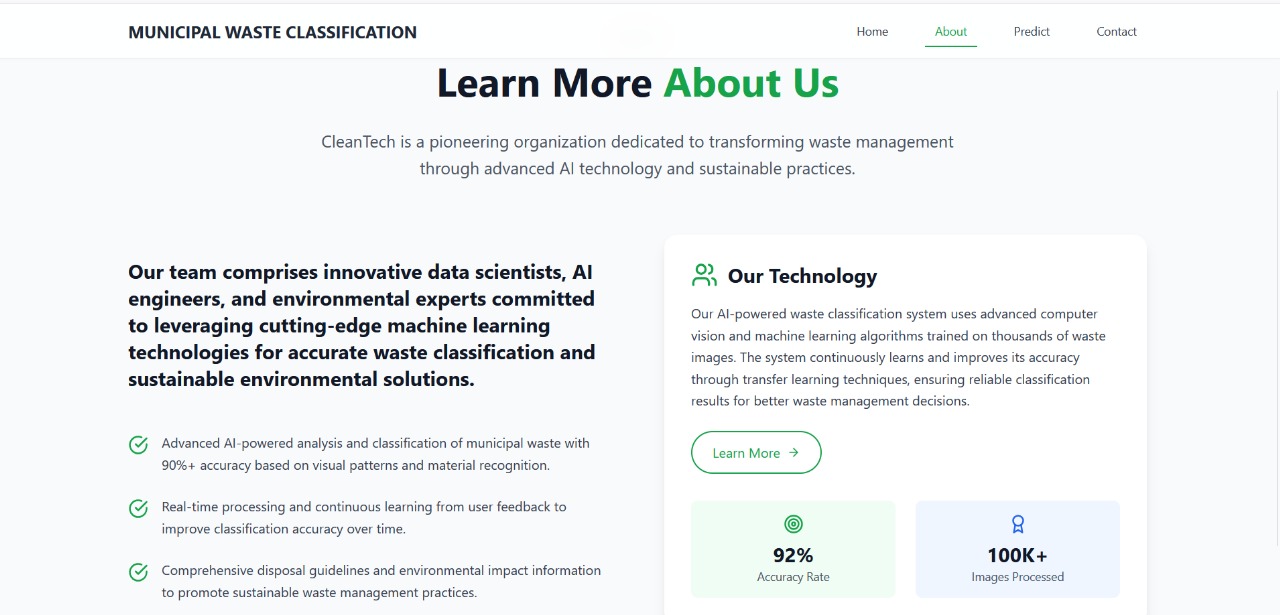
**Run the web application:**

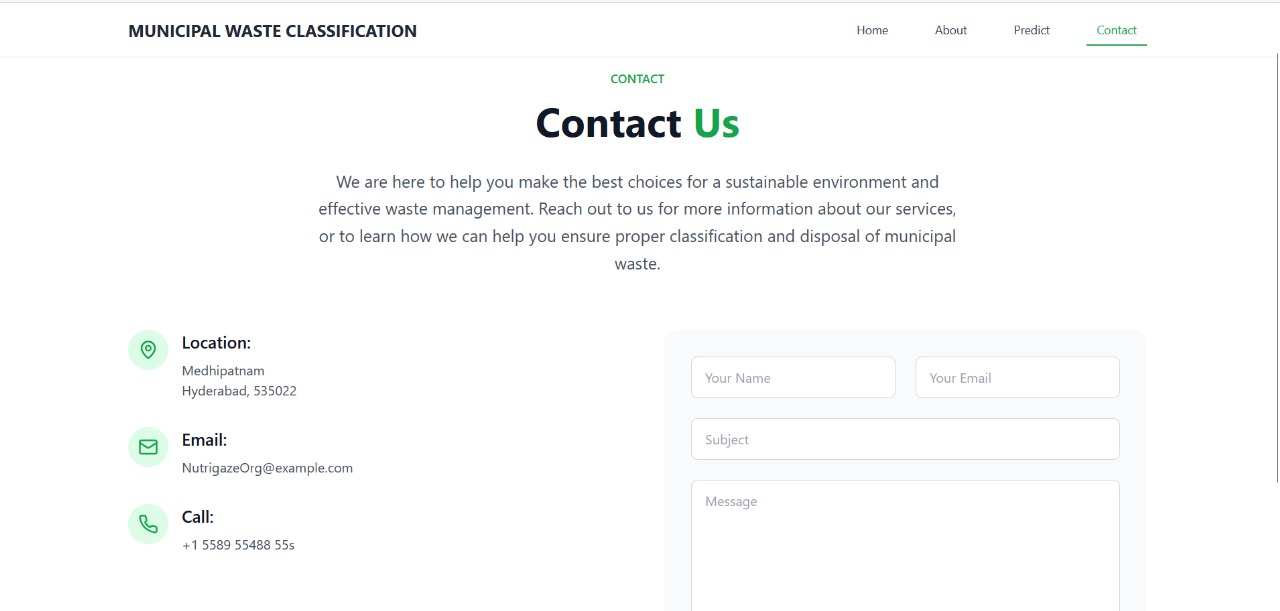
UI Image preview:

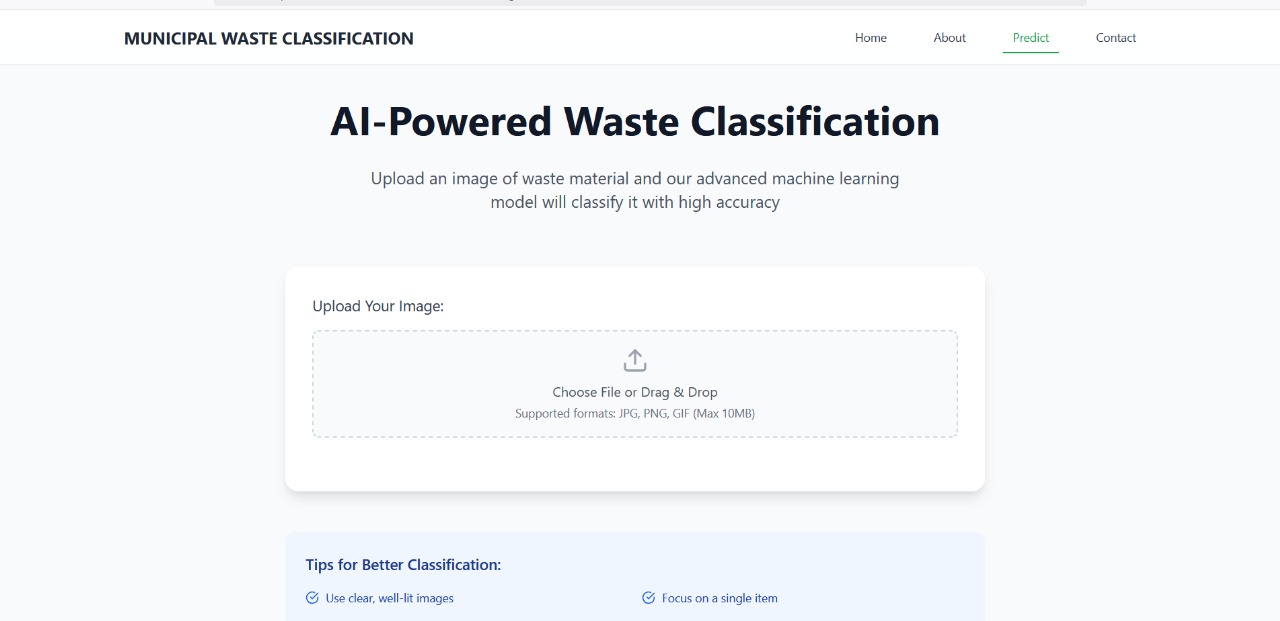
Let’s see what our index.html page looks like:

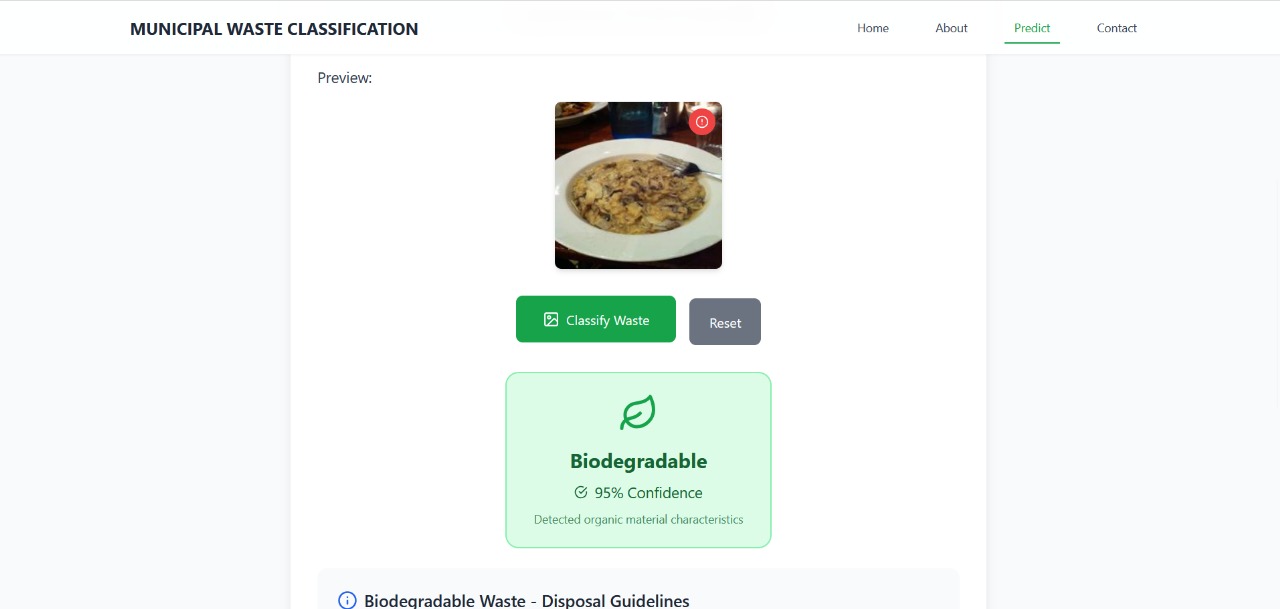


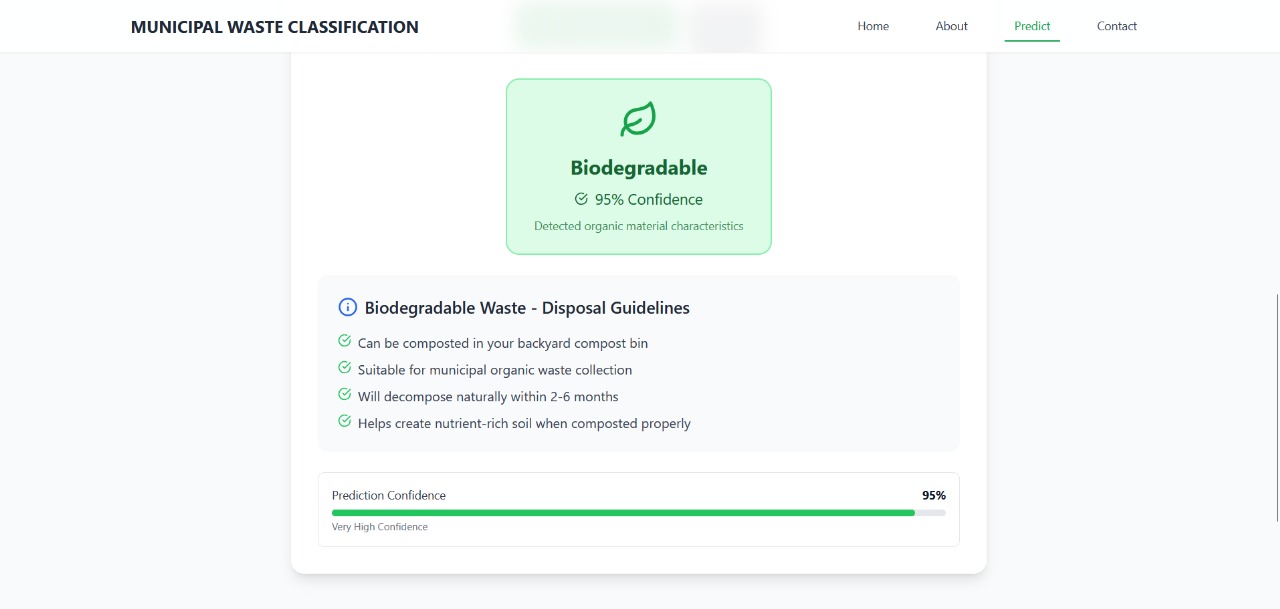


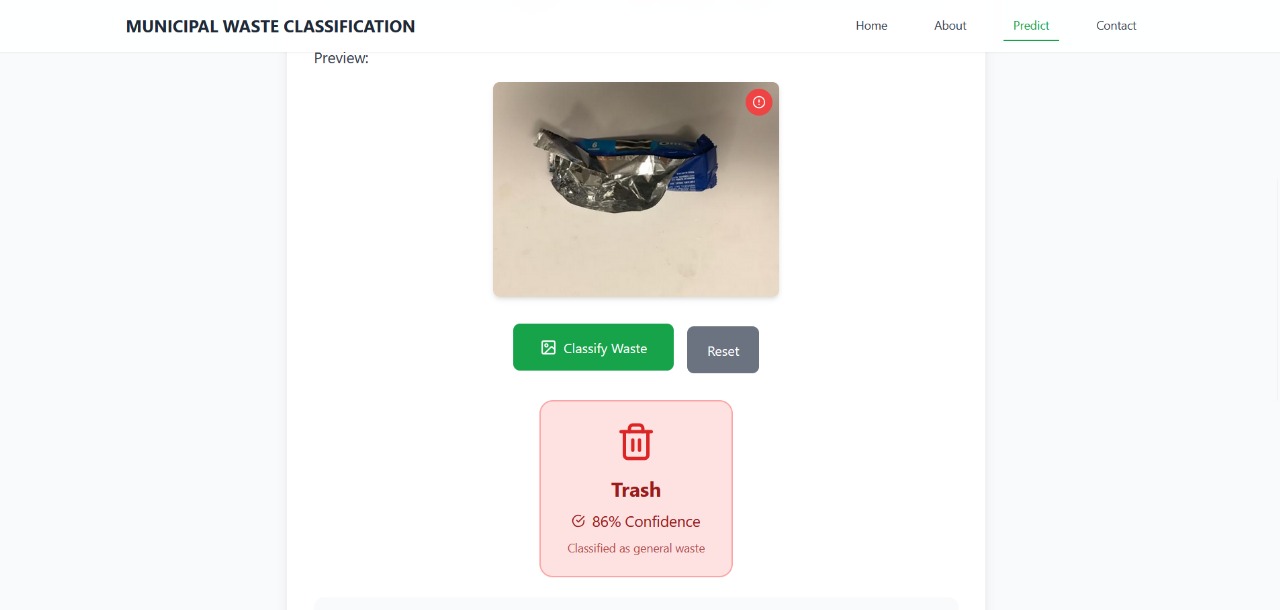












**Run Your cleantech-transfer-learning All-in-One CMD Guide**

* Cd C:\Users\Sriramadasushivani\cleantech\project
* pip install numpy pandas scikit-learn matplotlib seaborn tensorflow flask pillow
* python app.py

Then open your browser and go to:

http://localhost:5174

-------------\*\*\* THANKYOU \*\*\*--------------