Project Design Phase-II
Technology Stack (Architecture & Stack)

Date	30 June 2025
Team ID	LTVIP2025TMID44727
Project Name	**CleanTech: Transforming Waste Management with Transfer Learning**
Maximum Marks	4 Marks

Technical Architecture:

Technical Architecture Overview:

The application aims to classify waste into **Biodegradable**, **Recyclable**, and **Trash** using a **Transfer Learning model (VGG16)**. The architecture is modular and scalable, suitable for cloud deployment using **Flask** as backend, HTML/CSS for frontend, and integrated ML inference pipeline for prediction.

✨ Example Use Flow:

1. User uploads an image through web UI
2. Image is sent to backend Flask server
3. Pre-trained ML model (VGG16-based) processes and classifies the image
4. Result is returned to the UI and optionally stored in a cloud database

---

📘 Table 1: Components & Technologies

S.No	Component	Description	Technology
1	User Interface	Web interface for image upload and viewing results	HTML, CSS, JavaScript
2	Application Logic-1	Backend logic for handling requests and responses	Python, Flask
3	Application Logic-2	Image preprocessing and resizing before model inference	OpenCV, NumPy
4	Application Logic-3	Model prediction pipeline using Transfer Learning	TensorFlow, Keras (VGG16)
5	Database	Store classification results (optional)	SQLite (for local), MySQL (for cloud)
6	Cloud Database	Cloud storage of predictions	Firebase / AWS RDS (optional)
7	File Storage	Store uploaded images temporarily	Local File System or AWS S3 (optional)
8	External API-1	To fetch waste category guidelines from government datasets	Swachh Bharat API (optional)
9	External API-2	To fetch geolocation data based on IP	IPinfo API (optional)
10	Machine Learning Model	Image classification using VGG16 model	Transfer Learning – VGG16 (Keras/TensorFlow)
11	Infrastructure (Server)	Hosting the application on local server or cloud	Localhost / AWS EC2 / Render / Heroku

Table 2: Application Characteristics

S.No	Characteristics	Description	Technology
1	Open-Source Frameworks	Frameworks and libraries used in the project	Flask, TensorFlow, Keras, OpenCV
2	Security Implementations	Password-protected UI (if extended), input sanitization	SHA-256 Hashing (for login), HTTPS

S.No	Characteristics	Description	Technology
3	Scalable Architecture	Modular microservice-style design with separate frontend, backend, and model	Flask (3-tier), Docker-ready
4	Availability	Easy deployment on cloud, can use load balancers for horizontal scaling	AWS/GCP load balancing, Heroku Dynos
5	Performance	Image resizing before prediction, model cached in memory for fast inference	Flask, TensorFlow serving, CDN for assets

Sample Architecture Diagram