CleanTech: Transforming Waste Management with Transfer Learning

# Team Members:

Name 1: Thota Hari Prasad (Team Lead)  
Name 2: T Srivas Aditya Chowdary  
Name 3: T Ajay  
Name 4: T Yugandhar

# Phase 1: Brainstorming & Ideation

Objective:  
Identify the inefficiencies in current waste management classification methods. Understand how AI, particularly transfer learning, can automate and improve the waste segregation process using images. Study environmental impact and sustainability goals.  
  
Key Points:  
1. Problem Statement:  
Manual segregation of municipal solid waste is time-consuming, error-prone, and hazardous. Visual distinctions between biodegradable, recyclable, and non-recyclable materials are often subtle and not scalable for humans.  
  
2. Proposed Solution:  
“CleanTech” is a deep learning-based solution that leverages pre-trained CNNs like VGG16 or MobileNet to classify waste images into Biodegradable, Recyclable, or Trash. Using transfer learning ensures high performance even on relatively small datasets.  
  
3. Target Users:  
- Municipal corporations and sanitation departments  
- Waste management companies  
- Smart city planners  
- Environmental researchers  
- Educational and awareness platforms  
  
4. Expected Outcome:  
An AI-based application capable of classifying waste images instantly with high accuracy, helping automate smart bins, improve recycling rates, and support sustainable waste disposal.

# Phase 2: Requirement Analysis

Objective:  
Identify all software, hardware, and functional requirements. Understand limitations related to image quality and classification accuracy. Acknowledge data-related ethics and automation concerns.  
  
Key Points:  
1. Technical Requirements:  
- Languages: Python 3.10+  
- Frameworks: TensorFlow, Keras  
- Tools: Google Colab, Jupyter Notebook, VS Code  
- Hardware: GPU (NVIDIA recommended), 16 GB RAM  
  
2. Functional Requirements:  
- Upload waste image  
- Classify into: Biodegradable / Recyclable / Trash  
- Show prediction confidence  
- Display image and predicted label  
- Download report (Optional)  
  
3. Constraints & Challenges:  
- Imbalanced class distribution  
- Lighting variation in images  
- Similar-looking waste categories  
- Explainability for non-technical users

# Phase 3: Project Design

Objective:  
Design a modular and scalable system. Ensure usability and reliability through intuitive flow and visual output.  
  
Key Points:  
1. System Architecture:  
- Input Module → Image Preprocessing  
- Classification Module → Transfer Learning (VGG16/MobileNet)  
- Output Module → Display prediction & confidence  
  
2. User Flow:  
User uploads image → Model preprocesses → Classifies into 3 categories → Confidence displayed → (Optional: Report exported)  
  
3. UI/UX Considerations:  
- Clean, simple interface  
- Color-coded prediction output  
- Accessible on web/mobile  
- Clear error messages for invalid inputs

# Phase 4: Project Planning (Agile)

Objective:  
Plan iterative development using Agile (Scrum). Enable team collaboration and testing within short sprints.  
  
Key Points:  
1. Sprint Planning:  
- Sprint 0: Research & setup  
- Sprint 1: Dataset prep & image cleaning  
- Sprint 2: Model training with VGG16  
- Sprint 3: UI development  
- Sprint 4: Backend integration  
- Sprint 5: Testing & refinement  
  
2. Task Allocation:  
- ML Engineer: Model training, augmentation  
- Data Engineer: Data split, preprocessing  
- UI Developer: Design layout, buttons  
- Backend Developer: API, integration  
- QA Engineer: Testing edge cases & metrics  
  
3. Timeline & Milestones:  
- Week 1–2: Dataset split, augmentation  
- Week 3–4: Model trained & validated  
- Week 5: Interface & backend ready  
- Week 6: Final testing + model deployment

# Phase 5: Implementation

Objective:  
Convert the blueprint into an integrated, working application. Handle bugs and model optimization for smooth deployment.  
  
Key Points:  
1. Technology Stack:  
- Frontend: HTML, CSS, Bootstrap (or Streamlit for demo)  
- Backend: Flask API  
- Model: Keras with TensorFlow  
- Deployment: Google Colab / Heroku (Optional)  
  
2. Implementation Steps:  
1. Data collection from Kaggle  
2. Image preprocessing & augmentation  
3. Load VGG16 + Custom layers  
4. Train and validate  
5. Save and load .h5 model  
6. Create prediction pipeline  
7. Display results  
  
3. Challenges & Fixes:  
- Overfitting: Solved with augmentation  
- Class imbalance: Addressed with class\_weight  
- Slow performance: Considered MobileNet for lighter version

# Phase 6: Functional & Performance Testing

Objective:  
Ensure model works accurately across scenarios, is responsive, and provides meaningful results on real-world waste images.  
  
Key Points:  
1. Tests Performed:  
- Prediction accuracy across categories  
- Batch image testing  
- UI usability testing  
- Edge cases: transparent/blurred waste  
- Resource usage on different systems  
  
2. Results & Fixes:  
- Accuracy reached ~92–94%  
- UI freeze bug fixed  
- Added fallback if category confidence <50%  
  
3. Final Validation:  
Model outperformed baseline CNN. Suitable for smart bin integration and civic demo use. Ready for academic showcase or field pilot.  
  
4. Deployment Options:  
- Google Colab for demo  
- Flask + Streamlit + Heroku (for scalable web use)  
- Docker container (for offline or kiosk deployment)