Cleantech: transforming waste management with transfer learning

---

1. Introduction

- **Project Title:**
  cleantech: transforming waste management with transfer learning
- **Team Members:**

Name	Role
R.Narasimha rao	Team Lead / Domain Expert
M.V S S Sarma	Machine Learning Engineer
M.Likitha	Frontend Developer (React)
K. Deekshith	Backend Developer (Node.js)

---

2. Project Overview

- **Purpose:**
  To revolutionize waste management by leveraging **transfer learning techniques** to automatically classify waste into **biodegradable, recyclable, or trash**, enabling smarter and more efficient recycling and disposal practices.
- **Features:**
  - Automated Waste Classification
  - User-Friendly Web Interface.
  - Real-Time Prediction
  - Sustainable Impact
  - Accurate & Efficient

---

3. Architecture

- **Frontend (React):**
  - User Interface
  - Image Upload
  - API Integration
  - Real-Time Prediction
  - Feedback Handling
  - Responsive Design
  - Separation of Concerns
  - Custom Styling

- **Backend (Node.js & Express.js):**
    - Flask Framework
    - Transfer Learning (VGG16)
    - Image Preprocessing
    - Prediction Endpoint (/predict)
    - Cross-Origin Requests (CORS)
    - Model Loading
    - Error Handling
    - JSON Response
- **Database (MongoDB):**
    - NoSQL Storage
    - Image Metadata Logging
    - calable & Fast
    - Data Analytics Support
    - Audit & Monitoring
    - Backup & Recovery
    - Integration with Flask

---

4. Setup Instructions

- **Prerequisites:**
    - Type “pip install numpy” and click enter.
    - Type “pip install pandas” and click enter.
    - Type “pip install scikit-learn” and click enter.
    - Type ”pip install matplotlib” and click enter.
    - Type ”pip install scipy” and click enter.
    - Type ”pip install seaborn” and click enter.
    - Type ”pip install tensor flow” and click enter.
    - Type “pip install Flask” and click enter
- **Installation Steps:**
- # Clone the repository
- git clone https://github.com/your-org/CleanTech: Transforming Waste Management with Transfer Learning.git
- cd Clean Tech: Transforming Waste Management with Transfer Learning
- # Setup backend
- cd server
- npm install
- # Setup frontend
- cd ../client

- npm install
-
- # Setup Python ML model (if applicable)
- cd ../ml-model
- pip install -r requirements.txt
- **Environment Variables (.env in server folder):**
- PORT=5000
- MONGO_URI=mongodb+srv://<username>:<password>@cluster.mongodb.net/db
- JWT_SECRET=your_jwt_secret_key
- PYTHON_SCRIPT_PATH=./ml-model/predict.py

---

5. Folder Structure

- **Static :**
  - ➢ Assets
  - ➢ Forms
  - ➢ uploads
- **Templates:**
  - ➢ Blog-single.html
  - ➢ Blog.html
  - ➢ Index.html
  - ➢ Portfolio-details.html
- App.py
- Healthy_vs_rotten.h5
- Ipython.html
- Readme.txt

---

6. Running the Application

- **Frontend:**
- HTML
- CSS
- **Backend:**
- PYTHON
- HTTP

- **ML Model Server (optional if standalone Flask app):**
- cd ml-model
- python app.py

7. API DOCUMENTATION

Endpoint	Method	Description	Request Body / Params	Sample Response
/api/auth/register	POST	Register a new user	{ "name": "John", "email": "john@mail.com", "password": "123456" }	{ "token": "jwt_token", "user": { "id": 1, "name": "John" } }
/api/auth/login	POST	Login user	{ "email": "john@mail.com", "password": "123456" }	{ "token": "jwt_token", "user": { "id": 1, "name": "John" } }
/api/predict	POST	Submit waste image for prediction	Form-Data: image (file)	{ "prediction": "Recyclable" }
/api/records	GET	Get all prediction records (admin)	Header: Authorization: Bearer <JWT Token>	[{ "id": "abc123", "filename": "waste.jpg", "category": "Biodegradable", "timestamp": "2025-06-30T10:00:00Z" }]

8. Authentication

- Uses **JWT tokens** for secure user sessions.
- Tokens stored in **localStorage**.
- Protected routes with middleware validation.
- Roles: admin, user – used to control access to certain features like user management or analytics.

9. User Interface

(Add images in actual README or documentation PDF)

- **Login/Register Screens**
- **Prediction Form** – input patient data

- **Prediction Result View**
- **Admin Dashboard** – view all patient records and results
- **Visual Analytics** – charts and trends

---

10. Testing

- **Tools Used:**
  - Jest (unit testing for backend logic)
  - React Testing Library (component testing)
  - Postman (manual API testing)
  - PyTest (for Python model testing)
- **Strategy:**
  - Unit tests for validation, utility functions.
  - Integration tests for REST APIs.
  - Snapshot/UI testing for React components.

---

11. Screenshots or Demo

- 📷 *Screenshots of Key Pages:*
  - Login Page
  - Prediction Form
  - Prediction Results
  - Admin Dashboard

---

12. Known Issues

- Limited Dataset Diversity
- Misclassification of Overlapping Waste
- No Image Validation on Frontend
- Backend Error Handling
- No Feedback Loop for Learning
- Authentication Not Role-Based

---

13. Future Enhancement:

- Allow users to classify waste directly using live camera input from mobile or webcam.

- Upgrade model to detect and classify multiple waste items within a single image using object detection (e.g., YOLO, SSD).
- Add a feature where users can correct wrong predictions, allowing the system to learn continuously over time.
- Develop Android/iOS apps for wider reach and usability in field waste management scenarios.
- Store geolocation with classification results to map and monitor waste types across different area
- Introduce a point system or rewards to encourage users to actively participate in proper waste segregation.
- Create an admin dashboard showing classification trends, user activity, and prediction accuracy over time.
- Add support for multiple regional languages to increase accessibility across diverse user bases.