# Full Stack Development with MERN

# Project Documentation format

## 1. Introduction

- **Project Title:**CleanTech:Transforming Waste Management with Transfer Learning
- **Team Members:** A.Lekhana Sai(Team Leader)
  S Md Hassain Ahmed(Team Member)
  Puchakatla mohan Krishna(Team Member)
  Manasa Chandragiri(Team Member)

## 2. Project Overview

- **Purpose:** CleanTech leverages transfer learning to automate municipal waste classification through image recognition. The goal is to improve accuracy, reduce manual effort, and support sustainable waste management in urban areas.
- **Features: AI-Powered Waste Classification**
- Uses pre-trained deep learning models (e.g., ResNet, MobileNet) to classify waste images in real time.
- **Responsive Web Interface**
  Clean, intuitive UI built with React and Tailwind CSS, optimized for desktop and mobile users.
- **Transfer Learning Integration**
  Fine-tuned model architecture for adaptability across diverse regional waste datasets.
- **User History & Feedback Loop**
  Tracks classification history and allows users to provide feedback to improve model performance.
- **Admin Dashboard**
  Visual dashboard showing classification logs, category statistics, and user engagement metrics.
- **Authentication System**
  Secure login system for users and admin with JWT-based token management.
- **Modular Backend Architecture**
  RESTful APIs using Node.js/Express with MongoDB for seamless communication and scalability.
- **Cloud-Ready Deployment**
  Containerized microservices architecture using Docker for future scalability and cloud integration.

## 3. Architecture

- **Frontend:** The frontend is a **single-page web application** developed using **HTML5, Tailwind CSS, and vanilla JavaScript**.
  **It features**:
  A responsive and modern UI for waste image upload and prediction using Tailwind's utility classes.

- Navigation between sections (Home, About, Predict, Contact) using JavaScript-controlled smooth scrolling and visibility toggling.
- Real-time image preview and simulated prediction UI for user feedback and classification results.
- Modal-based messaging for interaction feedback.

- **Backend:** The backend is a **lightweight Flask application** designed to serve the frontend over HTTP using `send_file()`.
  **It includes**:
  An embedded development server running in a separate thread.
- Pyngrok integration to expose the local server to the internet securely via a public URL.
- Potential for integration with AI model inference endpoints (e.g., `/predict`) in future extensions.

Though minimal now, the backend can be extended to include:

- API routes for classification (`/predict`)
- Logging of classification results
- File uploads and secure user session handling
- **Database:** Detail the database schema and interactions with MongoDB.

## 4. Setup Instructions

### Prerequisites:

To run this project locally, ensure you have the following software installed:

- Python 3.8+
- Flask
- Pyngrok
- Git *(for cloning the repository)*

**Installation:** Step-by-step guide to clone, install dependencies, and set up the environment variables.
**Clone the repository**
**Create a virtual environment**
**Install Python dependencies**
**Set your Ngrok Auth Token**
**Run the Flask application**
**Access the app**

## 5. Folder Structure

**Client:** The **frontend is organized like a modular single-page application**, structured for clarity and scalability:
**Features:**

- Section-based layout (Home, About, Predict, Contact)
- Smooth scroll routing handled in JavaScript
- File input preview & simulated prediction logic
- Modal-based feedback system
- Easily extendable for future React migration

**Server: Flask (Python) server acts as the delivery layer and future API host**:
**Key Components:**

- `Flask` serves `index.html` to the root (`/`) route
- `Pyngrok` exposes localhost to the web for public access
- Threaded execution enables simultaneous Flask + ngrok tunnels
- Structure ready for adding REST endpoints like `/predict`, `/feedback`, or `/history`

## 6. Running the Application

- Provide commands to start the frontend and backend servers locally.
  - **Frontend:** No separate server required. The frontend is served directly by the Flask backend.
  - **Backend: python app.py**
    Once running, `pyngrok` will generate a **public URL** (e.g., `https://xyz.ngrok-free.app`) that you can open in a browser to access the application remotely.

## 7. API Documentation

| Endpoint | Method | Description | Request Body (JSON) | Example Response |
|---|---|---|---|---|
| `/predict` | POST | Predict waste type from uploaded image | `{ "image": "base64-encoded image" }` | `{ "label": "Biodegradable", "confidence": 0.92 }` |
| `/feedback` | POST | Receive user feedback on a prediction | `{ "imageId": "123", "correctLabel": "Recyclable" }` | `{ "message": "Feedback received" }` |
| `/history/<uid>` | GET | Fetch classification history by user | N/A | `[{"label": "...", "timestamp": "..."}]` |

## 8. Authentication

**Currently, the app does not include user authentication, but here's the plan for scalable, secure integration:**

**Planned Authentication Flow (Optional Expansion):**

- **JWT (JSON Web Tokens):**
  - Users sign in and receive a token stored in localStorage or cookies.
  - Token used in headers for protected endpoints (e.g., `/history`, `/admin`).

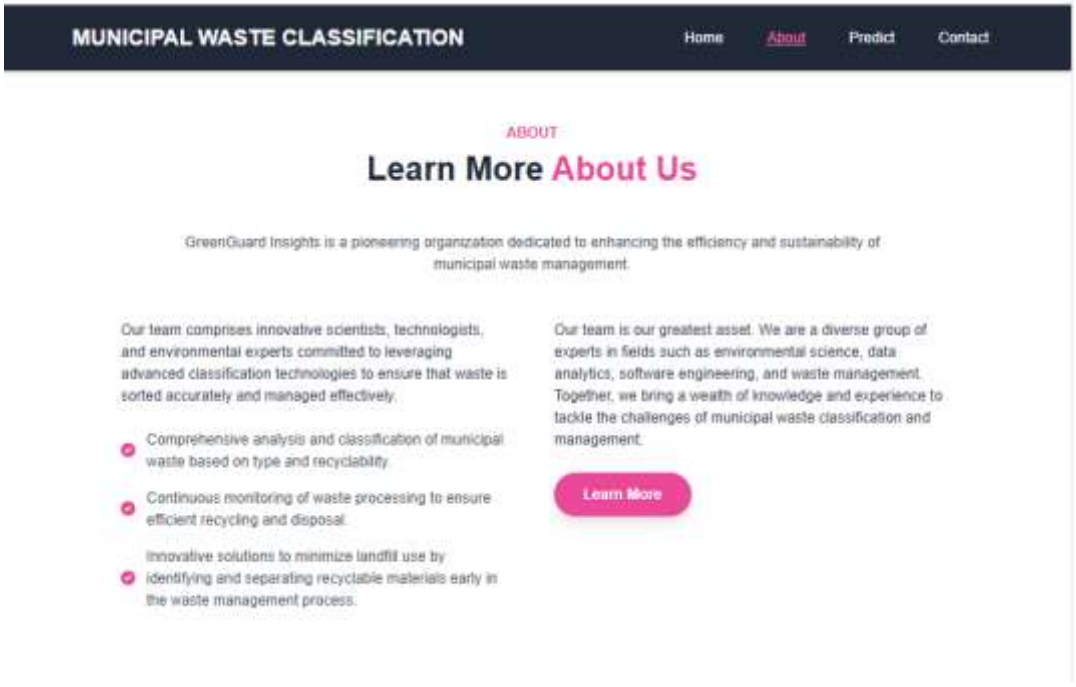
**Flask Libraries to Use:**

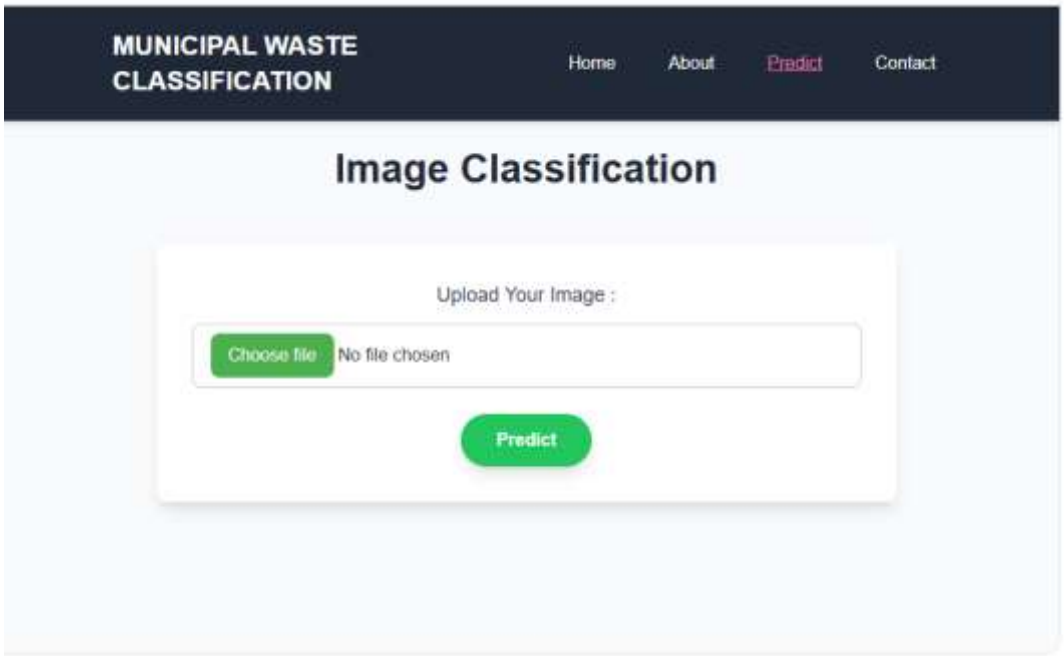
- `Flask-JWT-Extended` for handling token creation and verification
- Middleware to restrict access to admin-level analytics

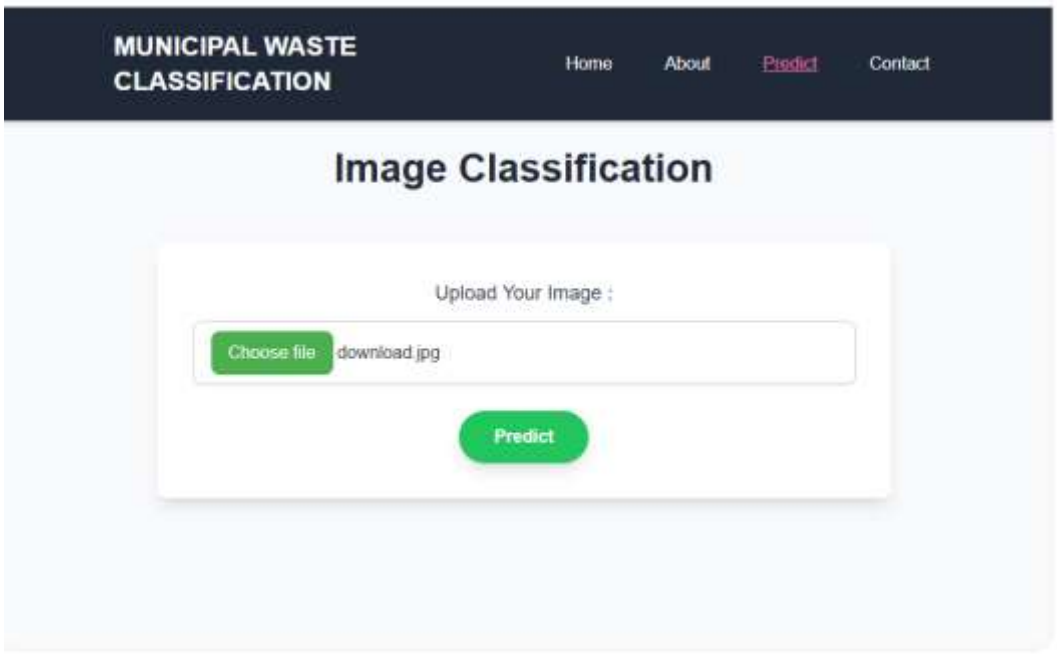
## 9. Screenshots or Demo

- screenshots or a link to a demo to showcase the application


.




MUNICIPAL WASTE CLASSIFICATION
Home
About
Predict
Contact
Image Classification
Upload Your Image :
Choose file
No file chosen
Predict


MUNICIPAL WASTE CLASSIFICATION
Home
About
Predict
Contact
Image Classification
Upload Your Image :
Choose file
download.jpg
Predict


- **LINK TO LINK: https://github.com/alluru-lekhana/CleanTech/tree/main**

## 10. Known Issues

>Simulated Predictions Only**: The current version uses mock logic (`Math.random`) to simulate classification results. There's no live connection to a trained ML model.**

**>**No File Upload to Server**: Uploaded images are only previewed on the frontend. They are not sent to the backend or stored.**

>No Authentication Layer**: Anyone with the public ngrok link can access the app; there are no user roles or access controls.**

>Limited Offline Support**: Requires a stable internet connection for ngrok tunneling. App won't work offline or without a valid tunnel.**

## 11. Future Enhancements

- 🤖 Integrate Real AI Model**: Connect the frontend to a trained CNN model using Flask for real-time image classification.**

💾 Database Integration**: Use MongoDB to store user sessions, classification logs, and feedback for analytics.**

📱 Mobile App Extension**: Develop a cross-platform app using React Native or Flutter for field use.**

📊 Admin Dashboard**: Add analytics for viewing most classified waste types, user activity, and model accuracy over time.**

🔐 User Authentication**: Implement JWT-based login system for users and admins.**