***Artificial Intelligence & Machine Learning*** Project Documentation

**1.Introduction:**

* *Project Title*:

**Smart Sorting: Transfer Learning for Identifying Rotten Fruits and Vegetables**

Team *Members: -*

*Team ID :* *LTVIP2025TMID34244*

Team Leader: Gayathri Katakamsetti

Team member: Alla Adi Keshava Reddy

Team member: Grandhi Anjana Priya

Team member: Chintapalli Aarya

**2.Project Overview:**

***Purpose:***

The **primary purpose** of the Smart Sorting system using **transfer learning** is to **automate the detection and separation of rotten fruits and vegetables** from fresh ones with high accuracy and efficiency. This is achieved by leveraging pre-trained deep learning models that can quickly learn to identify visual cues of spoilage.

**Goals of Smart Sorting System:**

1. **Accurate Classification of Produce** : Identify and classify fruits and vegetables as *fresh* or *rotten* using deep learning-based image analysis with high precision and recall.
2. **Develop a Transfer Learning-Based Model** : Utilize pre-trained models (e.g., ResNet, MobileNet, EfficientNet) and fine-tune them on a domain-specific dataset of fruits and vegetables to save time and computational resources.
3. **Real-Time Detection and Sorting** : Achieve real-time or near real-time inference speeds to enable deployment on sorting lines or conveyor belts without delays.
4. **Minimize Manual Inspection** :Reduce human error and labor dependency by automating the sorting process with consistent, repeatable performance.
5. **Scalability and Deployment Readiness** :

Design a system that can be scaled and adapted across different environments — from small farms to large warehouses and distribution centers.

1. **Improve Food Supply Chain Efficiency** o Prevent rotten produce from contaminating fresh stock, thus improving shelf life and reducing supply chain losses.
2. **Support for Multiple Fruit and Vegetable Types** o Extend the system's capability to support classification across various types of fruits and vegetables with minimal retraining.
3. **User-Friendly Interface and Reporting** o Provide interfaces for operators to monitor classification results, flag errors, and generate reports for quality assurance.

***Features:***

1. **Transfer Learning-Based Classification**

Utilizes powerful pre-trained deep learning models (e.g., MobileNet, ResNet, EfficientNet) fine-tuned for identifying signs of spoilage in fruits and vegetables.

1. **High Accuracy & Robust Detection**

Capable of detecting various degrees of spoilage (e.g., bruising, mold, discoloration) under different lighting and background conditions.

1. **Multi-Class and Binary Classification Support**

* **Binary classification**: Fresh vs. Rotten
* **Multi-class classification**: Fresh, Slightly Spoiled, Severely Rotten

1. **Real-Time Image Processing**

Processes input from cameras in real-time, enabling on-the-fly sorting with minimal latency.

1. **Edge and Cloud Compatibility**

Can be deployed on edge devices (e.g., Raspberry Pi, NVIDIA Jetson) or integrated with cloud platforms for centralized processing.

**Core Functionalities:-**

**Image Capture & Preprocessing** :

Captures images of fruits/vegetables and preprocesses them (resizing, normalization) for model input.

**Spoilage Detection Using Deep Learning:**

Predicts freshness/spoilage based on visual patterns using trained CNN models.

**Sorting Decision Engine** :

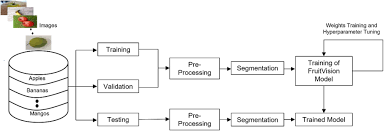
Decides the routing action (e.g., keep, discard, alert) based on prediction probabilities.

**Model Training & Fine-Tuning Module** :

Provides tools to retrain and improve model accuracy using new or additional datasets.

**3.Architecture:**

The architecture is designed to automatically detect and sort rotten fruits and vegetables using machine vision and deep learning, particularly through transfer learning, to improve accuracy and reduce development time.



**4.Setup Instructions:**

Prerequisites:

Hardware:

* Camera (USB, PiCam, or webcam)
* Raspberry Pi / Jetson Nano / Laptop with GPU
* Actuator or Servo Motor (for sorting)
* Breadboard, jumper wires (if using microcontrollers)
* Power supply

Software:

* Python 3.8+
* TensorFlow / Keras
* OpenCV
* Numpy, Matplotlib
* scikit-learn
* RPi.GPIO (for Raspberry Pi sorting)
* Jupyter Notebook or any IDE (VSCode, Thonny)

**5.Folder Structure:**

├── public/ # Static files

├── src/

│ ├── components/ # Reusable UI components

│ ├── pages/ # App pages (e.g. Home, Login)

│ ├── App.js # Main app component

│ └── index.js # Entry point

├── package.json

├── controllers/ # Request handlers

├── routes/ # API routes

├── models/ # DB models

├── app.js # Express app setup

├── server.js # Server entry point

├── .env # Environment config

├── package.json

**6.Running the Application:**

* Dataset is organized in dataset/fresh/ and dataset/rotten/
* Model is trained and saved as models/fruit\_sorter\_model.h5  Camera and (if needed) servo hardware are properly connected  All dependencies installed:
* bash
* pip install -r requirements.txt

**7**.**API Documentation:**

**This RESTful API enables:**

* Uploading fruit/vegetable images
* Classifying the image (Fresh or Rotten)
* Retrieving model prediction results
* Optional: triggering a hardware sorting action

**8.Testing:**

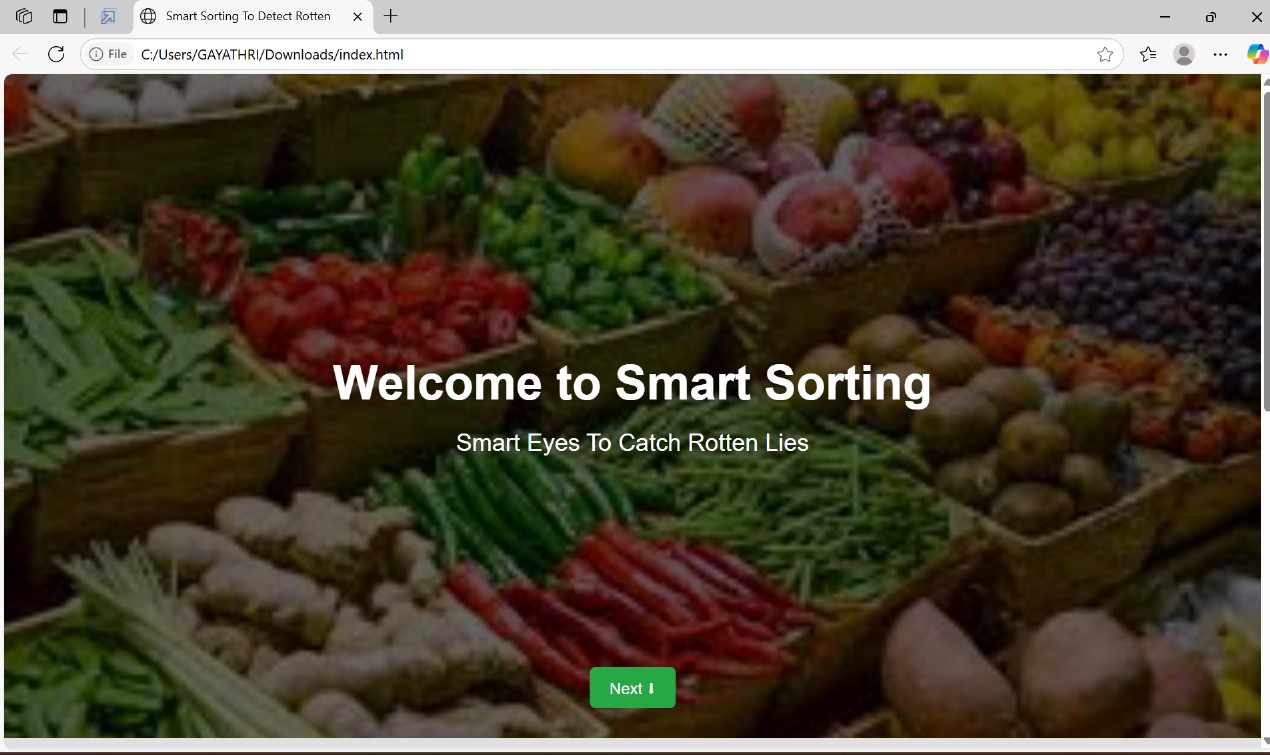
**Smart Sorting System – API Testing Guide Tools You Can Use:**

* Postman (GUI-based testing)
* cURL (Command-line)
* Python requests module (automated test scripts)
* Unit testing in Flask with pytest or unittest

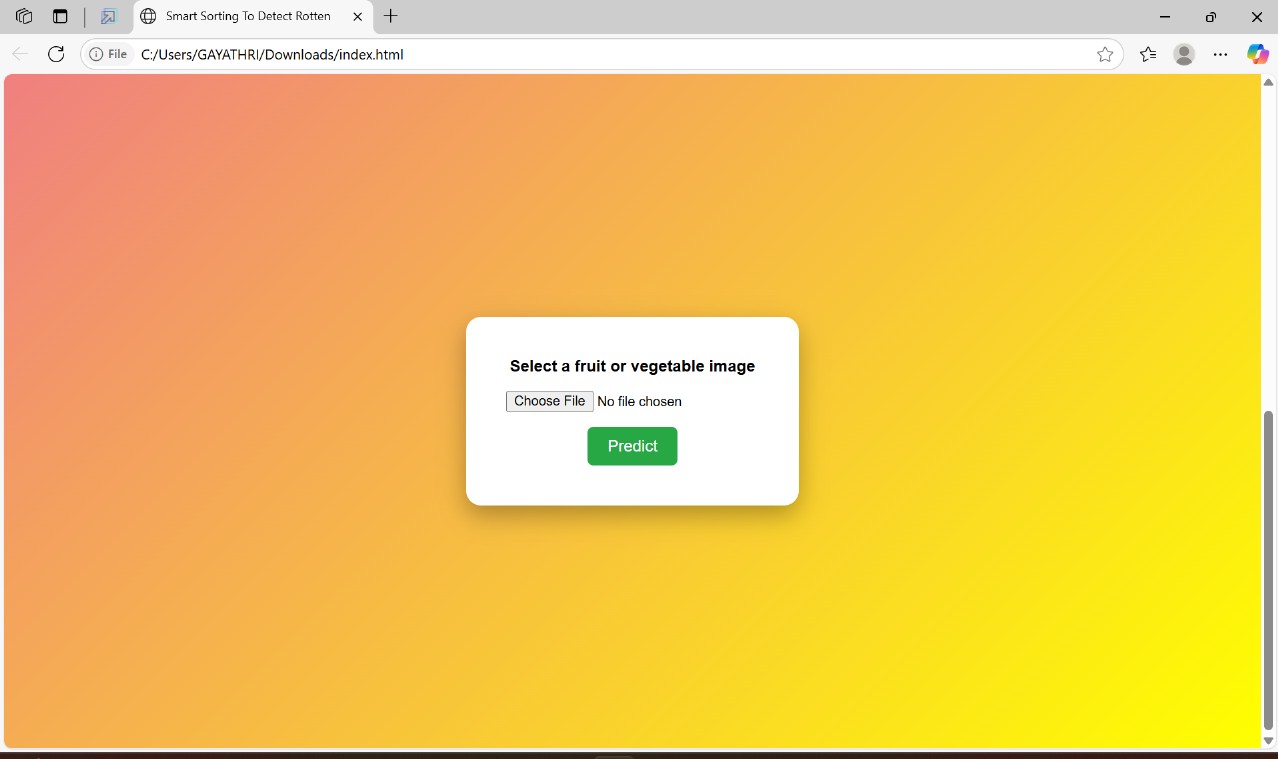
**9.Screenshots:**

**Templates screenshots:**

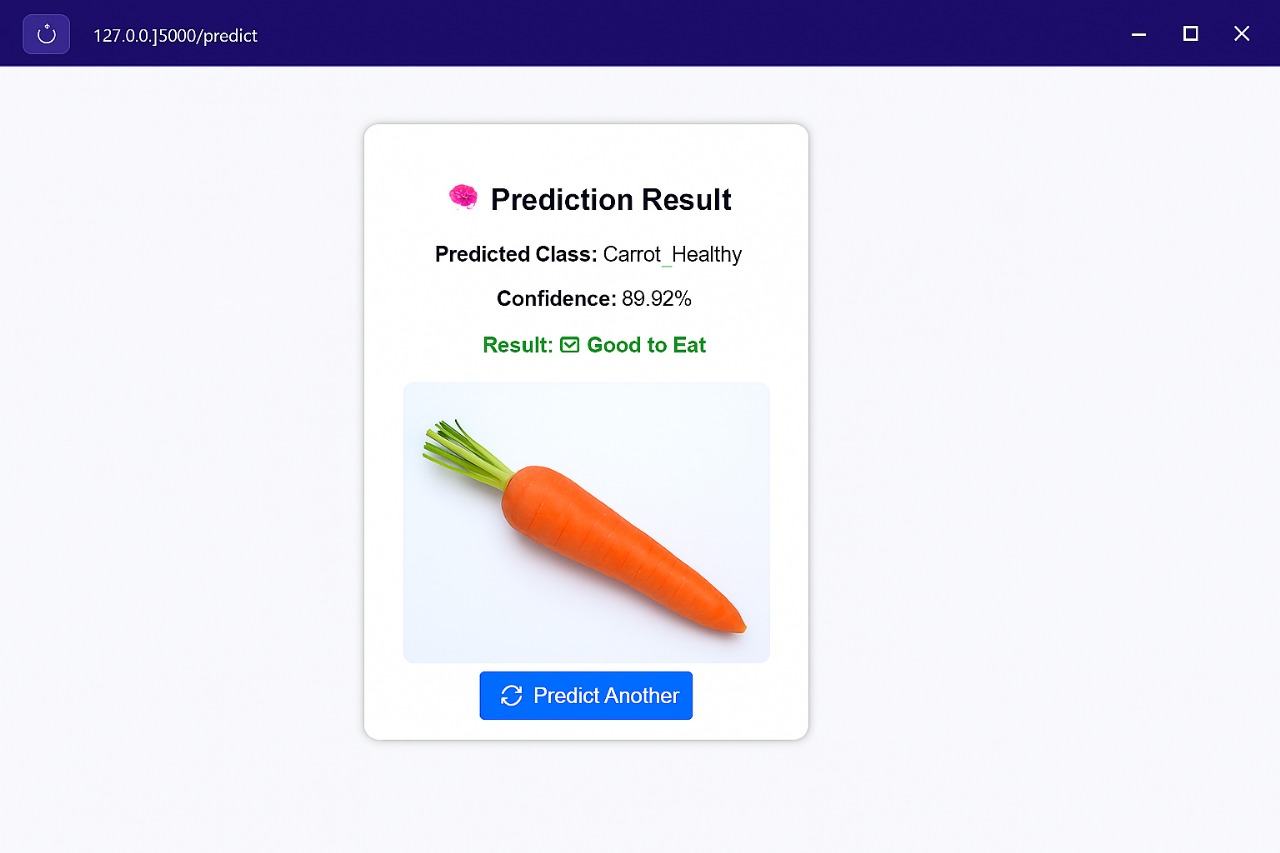
1. **Home Page:**

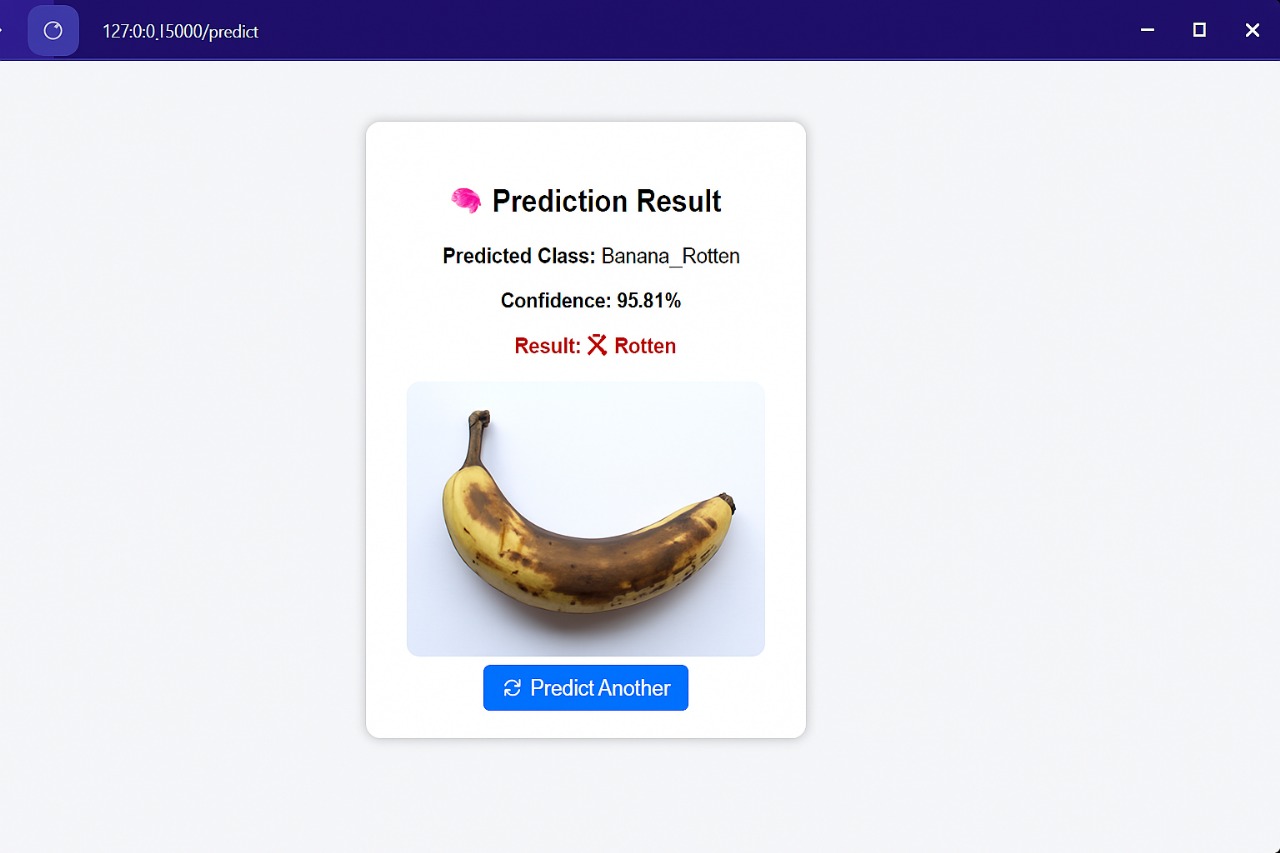


2.**About Page:**



3.**Predict Page:**





**10.Known Issues:**

**1. Dataset Limitations**

* **Insufficient Rotten Samples**: The dataset may have an imbalance, with fewer examples of rotten fruits/vegetables compared to fresh ones, affecting model accuracy.
* **Lack of Diversity**: Limited variation in fruit/vegetable types, lighting conditions, and backgrounds reduces model generalization.

**2. Misclassification Errors**

* **False Positives**: Fresh produce sometimes gets classified as rotten, leading to unnecessary waste.
* **False Negatives**: Rotten items might be identified as fresh, compromising quality control.

**3. Image Quality Dependency**

* **Blurry or Low-Resolution Images**: Poor camera quality or motion blur can degrade performance.
* **Lighting Sensitivity**: Model accuracy may drop significantly under low or inconsistent lighting.

**4. Hardware Constraints**

* **Processing Delays**: Real-time sorting may be delayed due to limited edge processing power.
* **Camera Calibration**: Improper calibration affects feature extraction and classification accuracy.

**5. Transfer Learning Overfitting**

* **Overfitting on Source Domain**: Pretrained models may not adapt well to the new dataset if fine-tuning isn't sufficient.
* **Domain Shift**: Variability between source (pretrained) and target (fruit images) domains affects accuracy.

**11.Future Enhancements:**

**1. Dataset Expansion and Improvement**

* Crowdsourced Image Collection: Engage farmers or supply chain workers to contribute labeled images, enriching dataset diversity.
* Synthetic Data Generation: Use GANs or data augmentation to create more examples of rare or subtle rot conditions.

**2. Advanced Model Architectures**

* Use of Vision Transformers (ViT): Explore transformer-based models for improved visual recognition in complex scenes.
* Multimodal Models: Combine visual features with sensor data (e.g., smell, firmness) to enhance accuracy.

**3. Real-Time Performance Optimization**

* Edge Deployment: Optimize models for deployment on edge devices using quantization, pruning, or lightweight architectures like MobileNet.
* Faster Inference Pipelines: Implement GPU acceleration or hardware-specific optimizations for real-time sorting.

**4. Adaptive Learning Systems**

* Online Learning: Allow the model to adapt continuously to new rot patterns or fruit types during deployment.
* Active Learning: Incorporate human-in-the-loop systems to review uncertain predictions and improve training data.

**5. Integration with Supply Chain Systems**

* Blockchain Traceability: Log inspection results on the blockchain for traceability and quality assurance.
* Automated Grading and Pricing: Link classification output to automated grading and dynamic pricing systems.