Sri Venkateswara College Of Engineering(SVCE)

Karakambadi Road, 517507, Mangalam, Tirupati, Andhra Pradesh 517507

(Affiliated to JNTUA)

PROJECT TITLE

SMART SORTING: Transfer Learning for Identifying Rotten Fruits and Vegetables

Artificial Intelligence & Machine Learning

TEAM ID: LTVIP2025TMID40987

Submitted by

T M Rupesh – SBAP0040987(Team Leader)

Abdul Kalam G –SBAP0041471(Team Member)

Tejaswini G– SBAP0041047(Team Member)

Chandana K – SBAP0040898(Team Member)

Hari Prasad M– SBAP0041549(Team Member)

Table of Contents

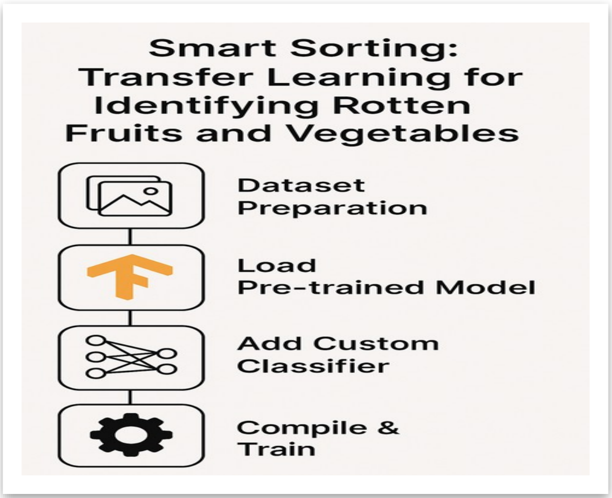
Introduction

“SMART SORTING: Transfer Learning For Identifying Rotten Fruits And Vegetables"! This project is designed to tackle a common problem in households and industries: efficiently identifying whether a fruit or vegetable is fresh or rotten.

At its core, this application leverages the power of Deep Learning, specifically Transfer
Learning, to achieve highly accurate predictions. We utilize the pre-trained VGG16 Convolutional Neural Network, a robust and widely used model, and fine-tune it on a specialized dataset containing images of both healthy and rotten produce. This approach allows us to benefit from the extensive knowledge VGG16 gained from millions of images, adapting it quickly to our specific task.

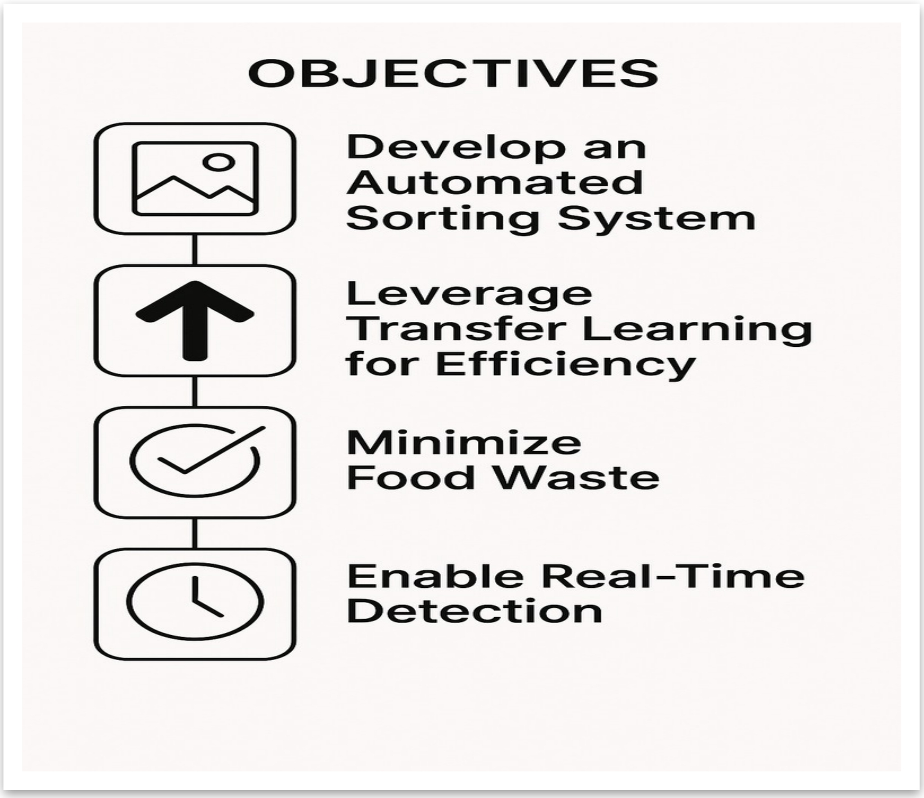
The system is deployed as a user-friendly Flask web application. This means you can interact with it through a simple web interface in your browser. Users can easily upload an image of a fruit or vegetable, and the backend, powered by our trained deep learning model, will process the image and provide an instant prediction—for example, **"Apple_healthy"** or **"Strawberry_rotten,"** along with a confidence score.

This project aims not only to provide a practical sorting solution but also to serve as a comprehensive learning experience, covering aspects from data handling and model training to full-stack web deployment.



Objectives:

The primary goal of this project is to create an automated system that can classify fruits and vegetables as fresh or rotten using deep learning. By applying transfer learning on pre-trained CNN models such as MobileNetV2, the project seeks to significantly reduce development time and resource usage while maintaining high accuracy. The system is designed to help minimize food waste by identifying spoiled produce early in the supply chain. Additional objectives include enabling real-time classification for potential integration with cameras or robotic arms and ensuring scalability across different types of produce.


OBJECTIVES
Develop an Automated Sorting System
Leverage Transfer Learning for Efficiency
Minimize Food Waste
Enable Real-Time Detection

Problem Statement:

In agricultural and food supply chains, a significant portion of produce is discarded due to spoilage, resulting in financial loss and unnecessary waste. Manual inspection of fruits and vegetables is time-consuming, inconsistent, and often unreliable, especially at large scales. There is a pressing need for an automated, efficient, and accurate system that can assist in classifying produce based on freshness. Leveraging advancements in computer vision and deep learning, this project aims to address the challenge of identifying rotten fruits and vegetables through image classification, offering a practical solution to improve food safety, reduce wastage, and streamline post-harvest sorting operations

Solution:

To address the challenge of identifying rotten fruits and vegetables efficiently, this project proposes a deep learning-based image classification system built with transfer learning. The key solutions include:

Automated Visual Inspection using pre-trained CNN models like MobileNetV2 to detect spoilage.

Transfer Learning Integration to reduce the need for extensive data and training resources.

Preprocessing Pipelines (resizing, normalization, augmentation) for consistent image inputs.

Real-Time Deployment Options with cameras or robotic sorting arms for on-the-fly classification.

Adaptability Across Produce by fine-tuning the model to support various fruit and vegetable types.

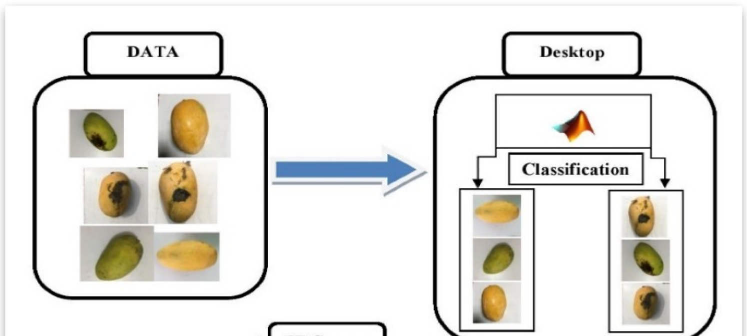
Requirements:

To implement an effective image-based sorting system for fruits and vegetables, both hardware and software components are essential. A computer with a GPU is recommended for model training, while a camera module and optional edge devices like a Raspberry Pi or Jetson Nano support real-time deployment. Consistent lighting is also important to ensure high-quality image capture. On the software side, the system requires Python, machine learning libraries like TensorFlow or PyTorch, OpenCV for image processing, and a suitable IDE such as Jupyter Notebook or VS Code. For web deployment, Flask can be used. The model relies on a well-structured dataset of labeled images, evenly distributed across fresh and rotten categories, resized to a standard resolution (e.g., 224x224 pixels). The model should use a pre-trained CNN like MobileNetV2, with an added classification layer. Functional expectations include image preprocessing, accurate classification, and display of results with confidence scores. These requirements ensure the system is reliable, adaptable, and capable of helping reduce food waste through automation.

Project Overview

Its core mission is to intelligently identify and classify fruits and vegetables as either healthy or rotten using advanced image recognition capabilities powered by machine learning. This project addresses a critical challenge in the supply chain:The efficient and accurate sorting of produce to minimize waste, enhance product quality, and ensure that only fresh goods reach consumers.

The agricultural sector frequently grapples with inefficiencies in manual sorting, which can be time-consuming, prone to human error, and inconsistent. This often leads to significant post-harvest losses due to undetected spoilage, impacting profitability and contributing to food waste. The Smart Sorting aims to mitigate these issues by providing an automated, precise, and rapid solution for quality assessment.



System Requirements:

➢ Hardware Requirements

Computer with GPU support (for training deep learning models)

Camera module (for capturing real-time images)

Raspberry Pi / Jetson Nano (optional, for edge deployment)

Lighting setup (to ensure consistent image quality)

➢ Software Requirements

Python 3.8+

TensorFlow or PyTorch (for model development)

OpenCV (for image preprocessing)

Jupyter Notebook / VS Code (for development)

Flask (optional, for web-based deployment)

➢ Dataset Requirements

Labeled images of fresh and rotten produce

Balanced dataset across categories

Image resolution: 224x224 pixels

Split into training, validation, and test sets

➢ Model Requirements

Pre-trained CNN (e.g., MobileNetV2, ResNet50)

Custom classification head

Accuracy target: ≥90% on validation set

User Requirements:

Image Upload Capability

Users should be able to upload images of fruits or vegetables through a simple interface (web or desktop).

Accurate Classification

The system must classify the uploaded image as *fresh* or *rotten* with a high degree of accuracy.

Confidence Display

Users should see the prediction result along with a confidence score (e.g., “Fresh – 94% confidence”).

Real-Time Feedback (Optional)

If used with a camera, the system should provide real-time classification as produce passes through.

Multi-Produce Support

Users should be able to classify different types of fruits and vegetables without needing to retrain the model.

User-Friendly Interface

The interface should be intuitive, with clear instructions and minimal technical jargon.

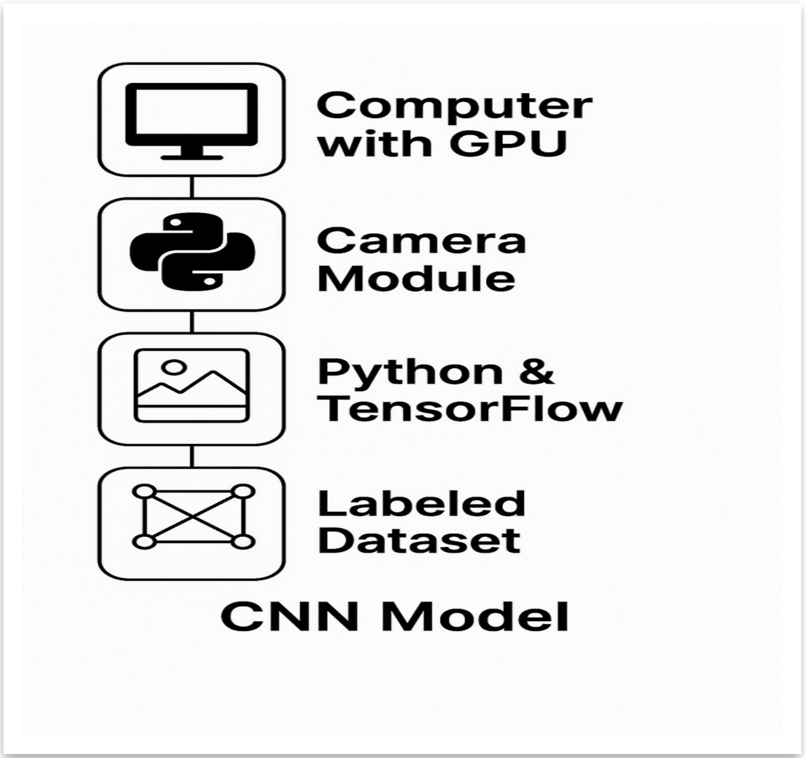
Feedback Option (Optional)

Users may provide feedback on incorrect predictions to help improve the system over time.

Accessibility

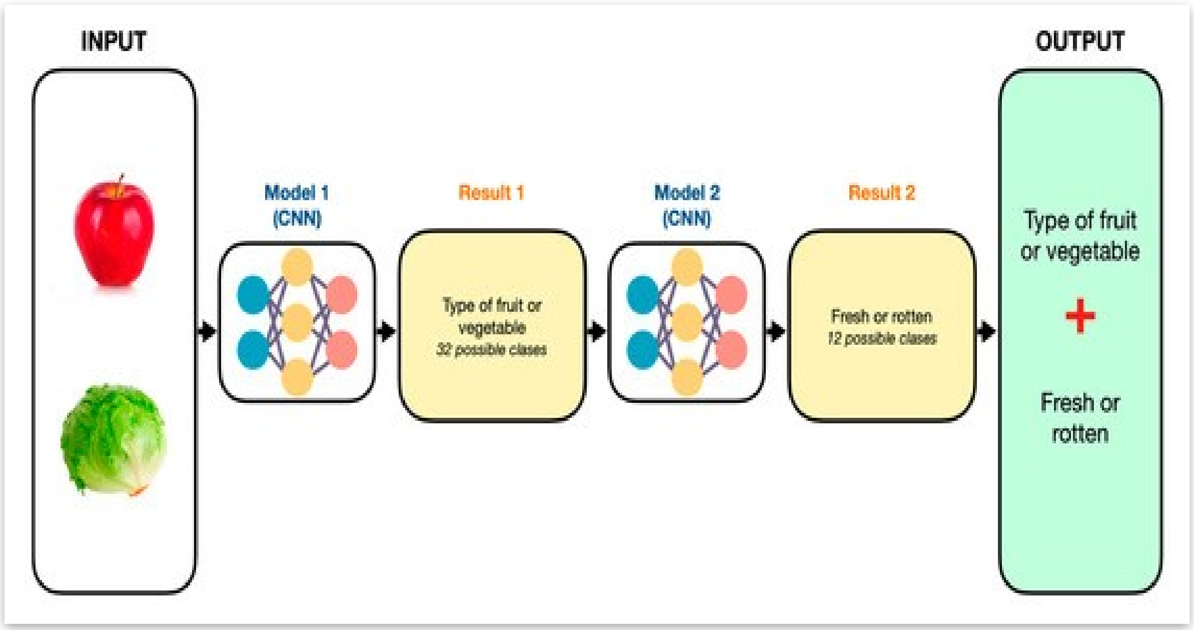
The system should be usable on common devices (laptops, tablets) and accessible to users with basic digital literacy.

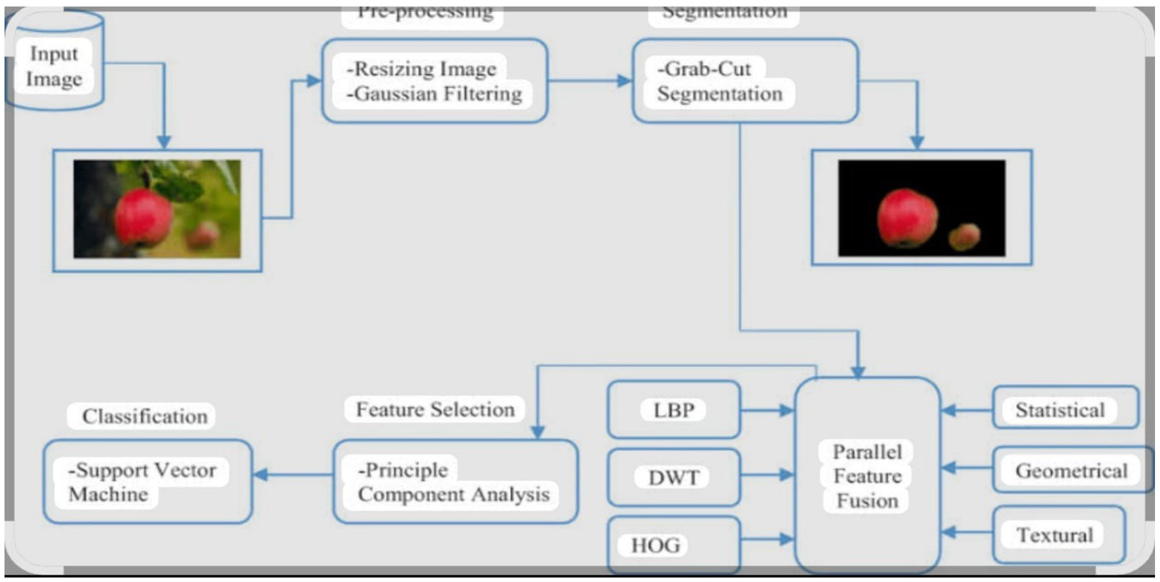
Technical Specifications:

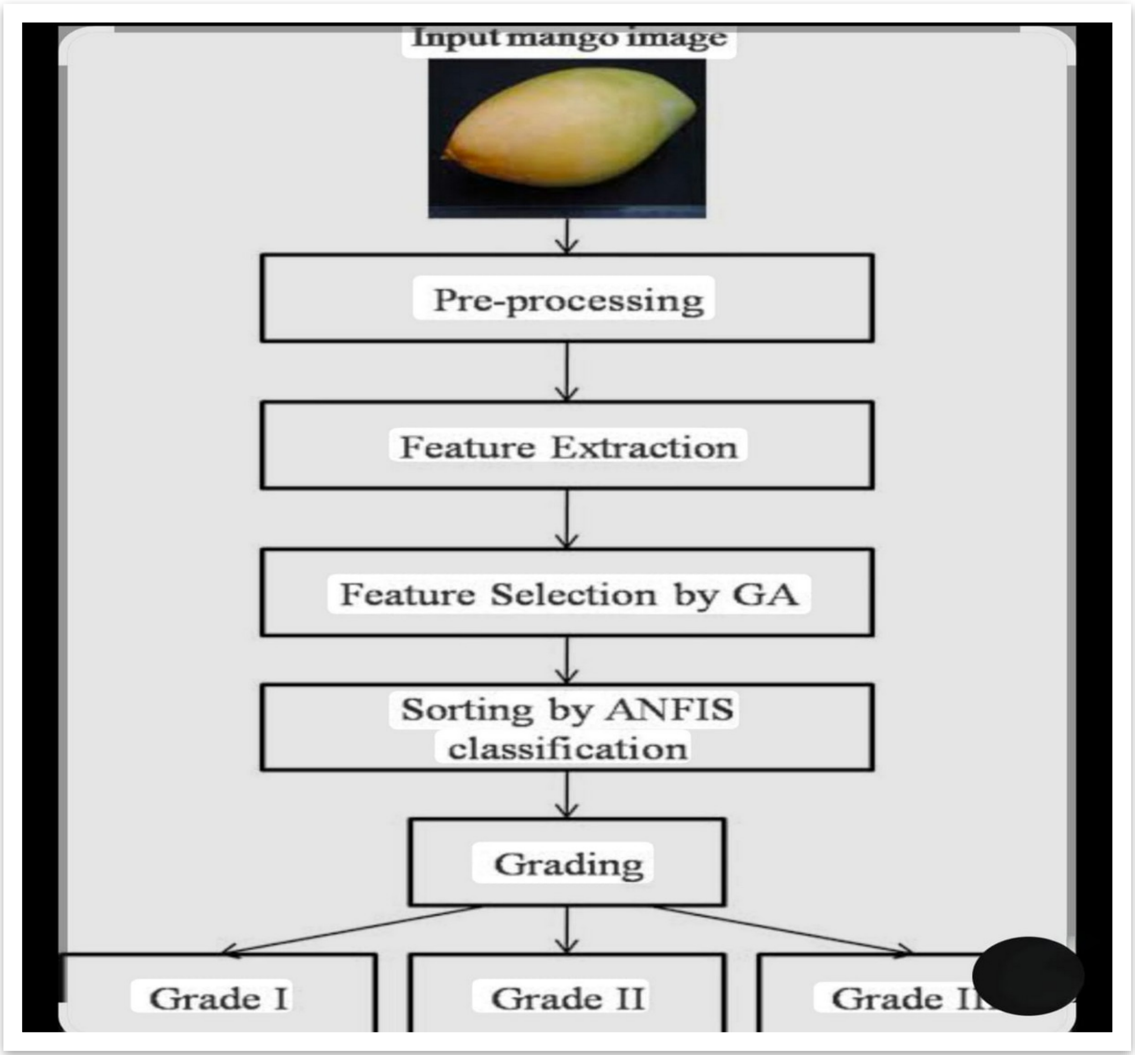


1. **Computer with GPU** – Used for training the model efficiently, especially when working with large image datasets and deep learning algorithms.
2. **Camera Module** – Captures real-time images of produce, which are then classified by the system as fresh or rotten.
3. **Python & TensorFlow** – Core tools used to develop the image classification model. Python is the programming language, and TensorFlow is the deep learning framework.
4. **Labeled Dataset** – A collection of images where each one is tagged as either fresh or rotten. This dataset is crucial for training and evaluating the model.
5. **CNN Model** – The core of the system: a Convolutional Neural Network that learns visual patterns and classifies the input image accordingly.

Technical Architecture:


INPUT
Model 1 (CNN)
Result 1
Type of fruit or vegetable
32 possible clases
Model 2 (CNN)
Result 2
Fresh or rotten
12 possible clases
OUTPUT
Type of fruit or vegetable
+
Fresh or rotten


Input Image
Pre-processing
-Resizing Image
-Gaussian Filtering
Segmentation
-Grab-Cut Segmentation
LBP
DWT
HOG
Parallel Feature Fusion
Statistical
Geometrical
Textural
Feature Selection
-Principle Component Analysis
Classification
-Support Vector Machine



PURPOSE:

The primary purpose of this project is to develop and demonstrate a practical application of deep learning and web development for a real-world problem. It serves as a proof-ofconcept for how machine vision can be integrated into everyday processes to deliver tangible benefits. Specifically, the application aims to:

Automate Quality Assessment: Provide a tool for rapid, consistent, and objective evaluation of fruit and vegetable health.

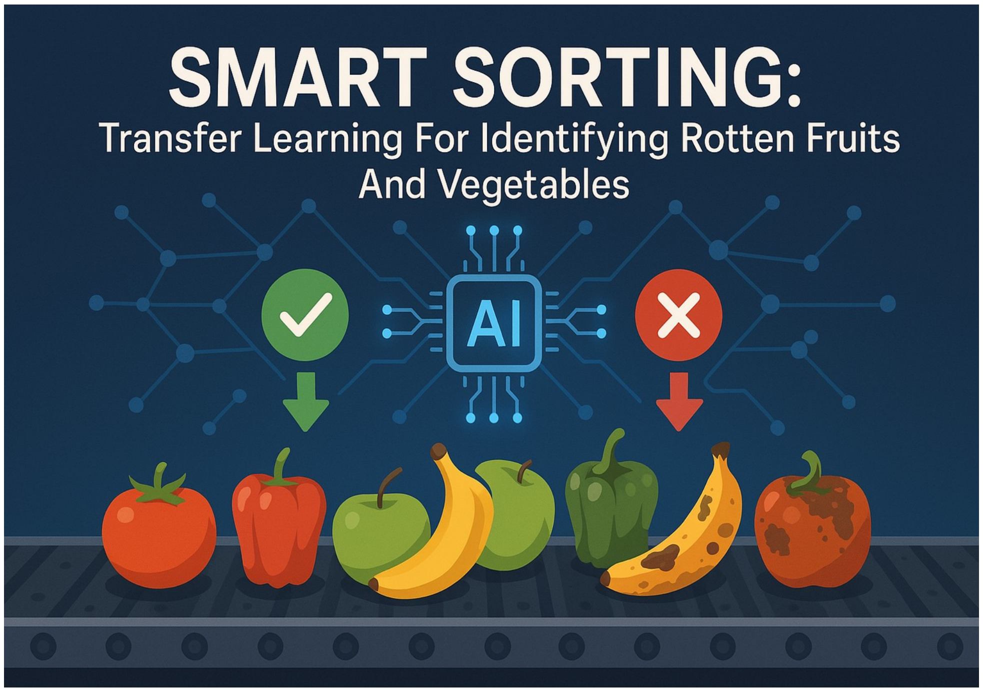
Reduce Food Waste: By accurately identifying rotten produce early, it helps in preventing the spoilage of entire batches and optimizes inventory management.

Improve Efficiency: Streamline the sorting process, reducing the need for extensive manual inspection.

FEATURES:

The "Smart Sorting " is built with user experience and robust functionality in mind, offering several key features:

- **User-Friendly Web Interface**: An intuitive and clean web-based interface allows users to easily upload images of various fruits and vegetables directly through their browser.
- **Real-time Prediction:** The application provides near real-time analysis of uploaded images, classifying the produce as "healthy" or "rotten" almost instantly.
- **Deep Learning Powered Core:** At its heart, the application utilizes a deep learning model, specifically leveraging the power of Transfer Learning. This approach allows the model to achieve high accuracy in classification by adapting knowledge from large, pre-trained neural networks to the specific task of identifying rotten produce. The trained model, healthy_vs_rotten.h5, is seamlessly integrated into the backend.
- **Prediction Confidence Levels:** Alongside the classification, the application displays a confidence score, giving users an indication of the model's certainty in its prediction.

- **Scalable Architecture:** Built with a Python Flask backend and a lightweight HTML/CSS/JavaScript frontend, the application has a modular design that facilitates future enhancements and potential scaling.
-
- 
SMART SORTING:
Transfer Learning For Identifying Rotten Fruits And Vegetables
AI

This project showcases a comprehensive solution, from data processing and model training to web application deployment, illustrating the power of integrating machine learning into practical full-stack development. It stands as a testament to leveraging AI for more sustainable and efficient practices in the agricultural domain.

Architecture:

- **Frontend**: The frontend is built using standard web technologies: HTML for structure, CSS for styling, and JavaScript for interactive elements. It consists of static templates rendered by the backend to provide the user interface for image uploads and displaying prediction results.
- **Backend**: The backend is developed with Python using the Flask web framework. It handles incoming image uploads, preprocesses the images, performs inference using the loaded machine learning model (healthy_vs_rotten.h5), and sends the prediction results back to the frontend.
- **Database**: This project does not utilize a persistent database (like MongoDB). All image processing and predictions are handled in-memory by the Flask application, and results are displayed directly to the user.

Setup Instructions:

Prerequisites:

- Python 3.x
- Pip (Python package installer)
- Required Python libraries (e.g., Flask, TensorFlow/Keras, OpenCV, Pillow, NumPy)
- Google Colab (recommended for running due to GPU requirements for ML models)
- Ngrok (for public access to the Colab-hosted application)

<u>Installation</u>: Clone

repository:

git clone

https://github.com/chandana-kota/smart-sorting-transfer-learning-for-identifying-rotten-fruits-and-vegetables

Navigate into the project directory:

cd smart-sorting-transfer-learning-for-identifying-rotten-fruits-and-vegetables

Install Python dependencies:

pip install Flask tensorflow opencv-python Pillow numpy (or list specific versions if needed in a requirements.txt file)

<u>Download the dataset and pre-trained model:</u>

The healthy_vs_rotten.h5 model is included via Git LFS. Ensure Git LFS is installed and configured (git lfs pull).

The dataset (if needed for re-training or local setup) should be downloaded separately (e.g., from Kaggle, as mentioned in my demo video).

Folder Structure:

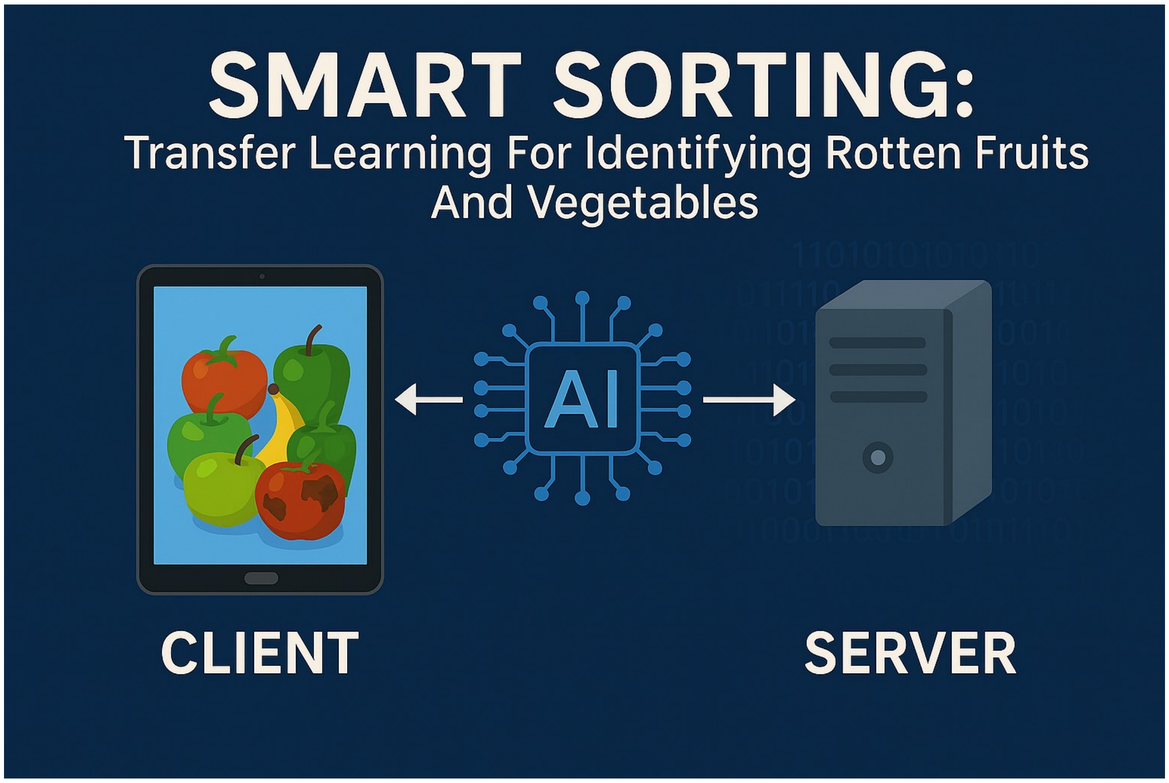
Client (Frontend):

templates/: Contains HTML files that define the user interface (e.g., index.html).

static/: (Expected if present) Would contain static assets like css/ for stylesheets, js/ for client-side JavaScript, and uploads/ for temporary uploaded images.

Server (Backend):

app.py: The main Flask application file, handling routes, model loading, image processing, and predictions. **train_model.py:** Script for training and evaluating the machine learning model. **data_prep_and_viz.py:** Script for data preprocessin g and visualization


SMART SORTING:
Transfer Learning For Identifying Rotten Fruits And Vegetables
AI
CLIENT
SERVER

healthy_vs_rotten.h5: The pre-trained Keras/TensorFlow model file used for predictions.

Running the Application :

Local Setup (e.g., in a terminal after cloning):

1. Ensure all prerequisites are met and dependencies are installed.
2. Start the Flask backend: python app.py
3. The application will typically run on http://127.0.0.1:5000.

Google Colab Setup (as shown in the demo video i.e., GITHUB repositary):

1. Mount Google Drive.
2. Unzip your project into /content/SmartSortingProject.
3. Change the current working directory to /content/SmartSortingProject.
4. Install required libraries within the Colab notebook.
5. Run app.py in the background (e.g., !nohup python app.py > app.log 2>&1 &).
6. Start ngrok to expose the Flask port (e.g., !ngrok http 5000). Access the application via the provided ngrok public URL.

API Documentation :

The application provides a web interface rather than a public API for direct consumption. However, the backend exposes the following key routes handled by **app.py:**

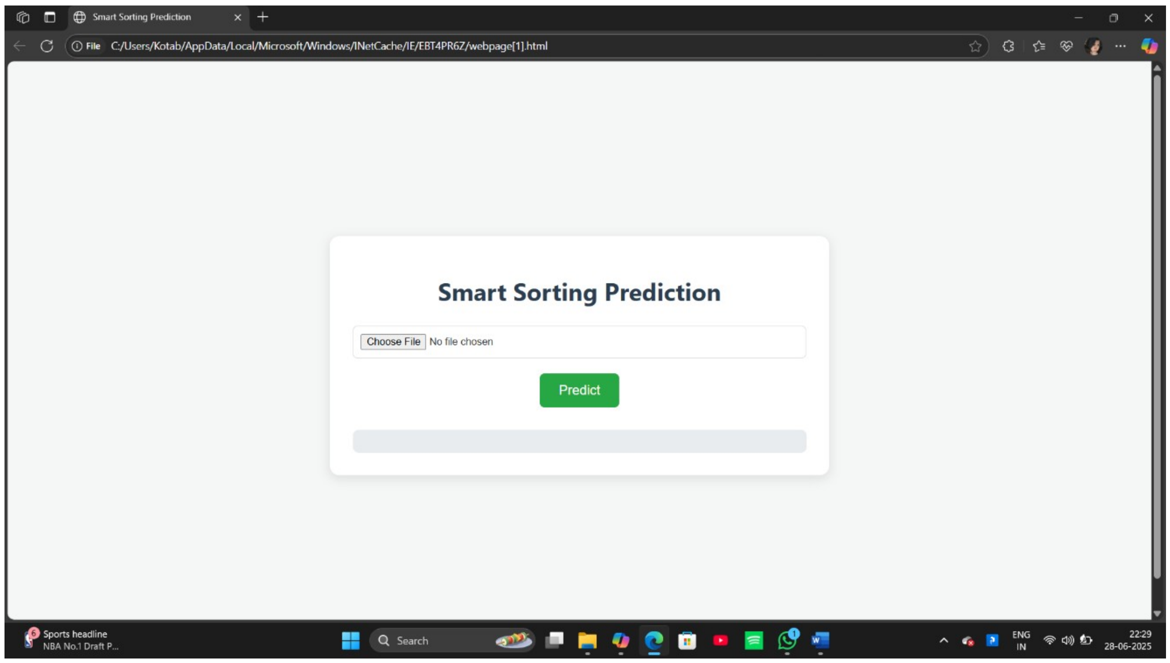
/ (GET): Serves the main index.html page, which is the entry point for the application.

/predict (POST): Accepts an image file (typically via a form submission). It processes the image, runs it through the healthy_vs_rotten.h5 model, and returns the prediction result (e.g., "Healthy" or "Rotten" along with confidence scores).

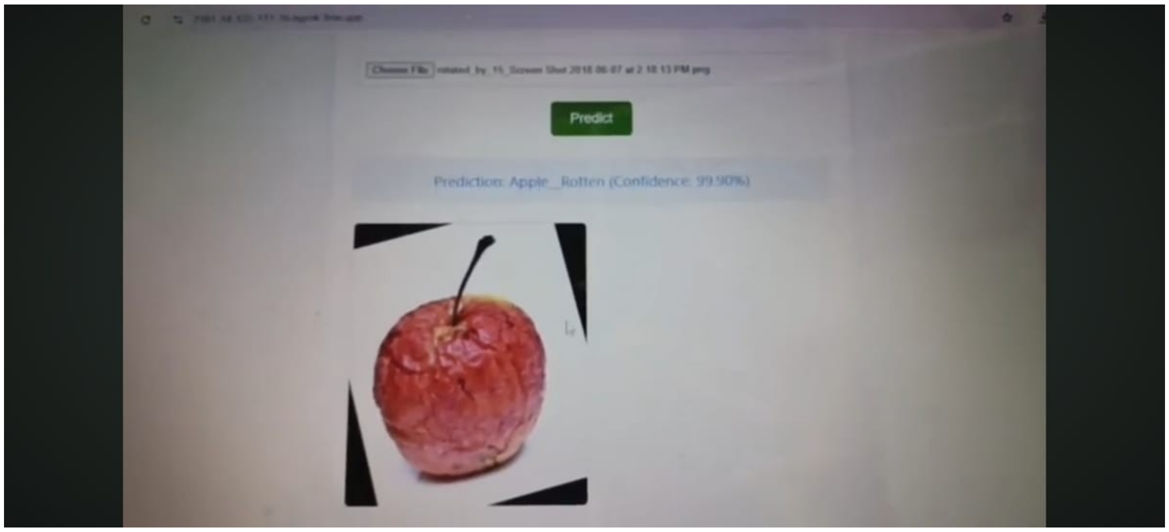
User Interface (Screenshots Section)

The UI is a simple web page allowing users to upload an image file. After processing, it displays the prediction result clearly.

THE WEB PAGE ALLOWING USERS TO UPLOAD AN IMAGE FILE:

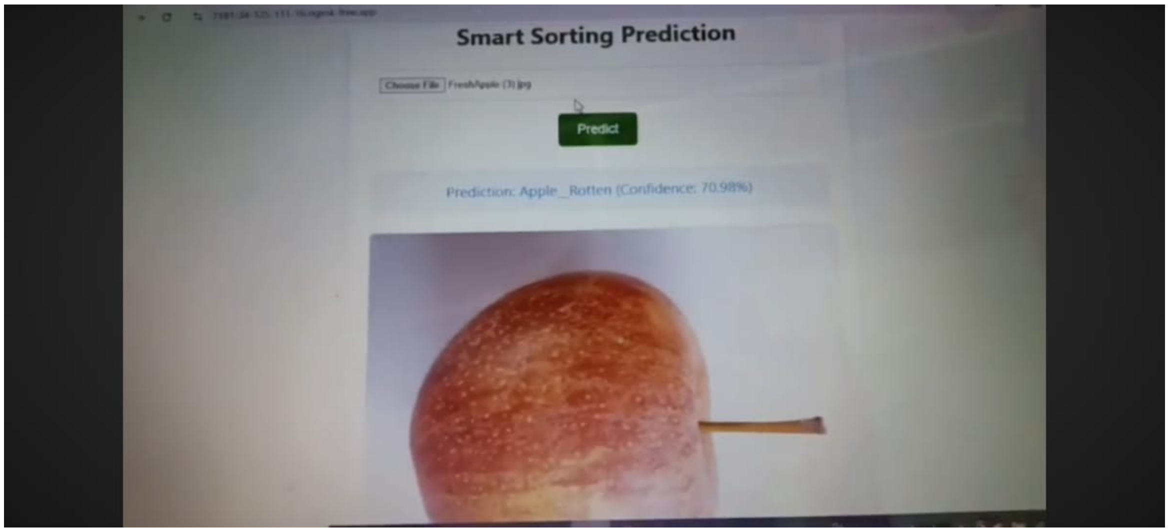

Smart Sorting Prediction
File C:/Users/Kotab/AppData/Local/Microsoft/Windows/INetCache/IE/EBT4PR6Z/webpage[1].html
Smart Sorting Prediction
Choose File
No file chosen
Predict

PREDICTION RESULTS:_



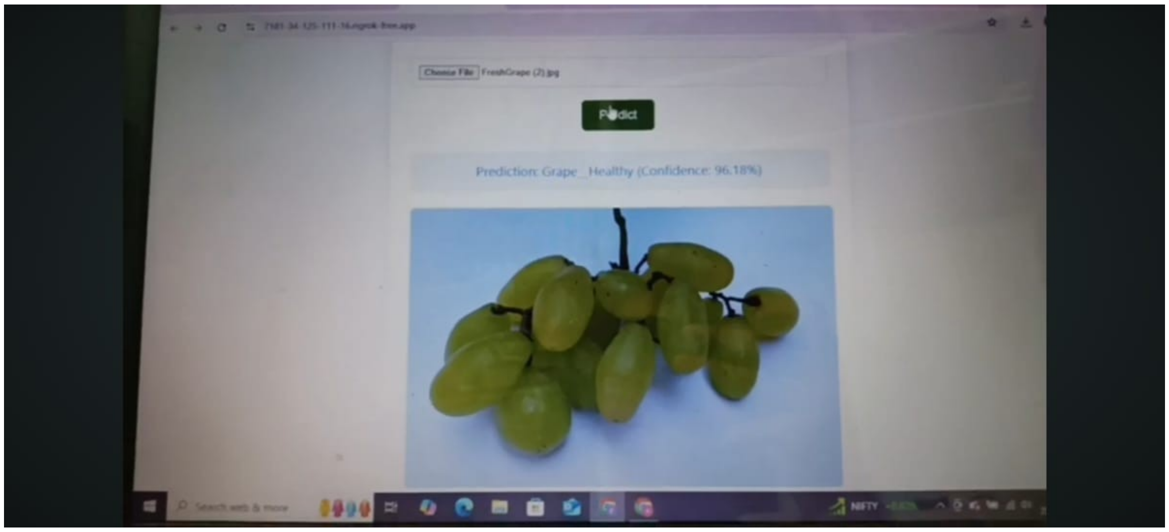
rotated_by_15_Screen Shot 2018 06 07 at 2 18 13 PM png

Prediction: Apple__Rotten (Confidence: 99.90%)

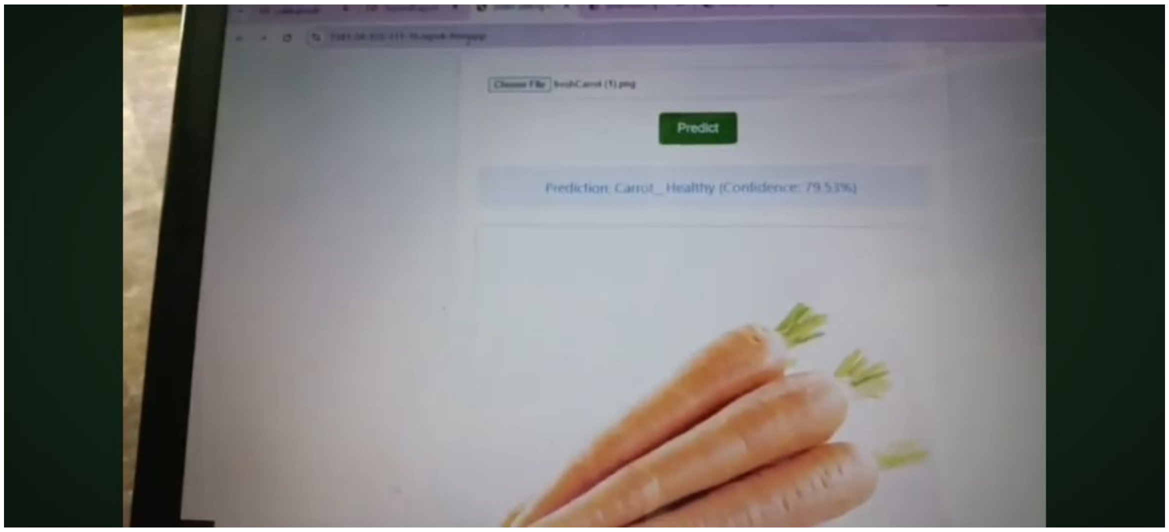


FreshApple (3).jpg

Prediction: Apple__Rotten (Confidence: 70.98%)



Prediction: Grape__Healthy (Confidence: 96.18%)

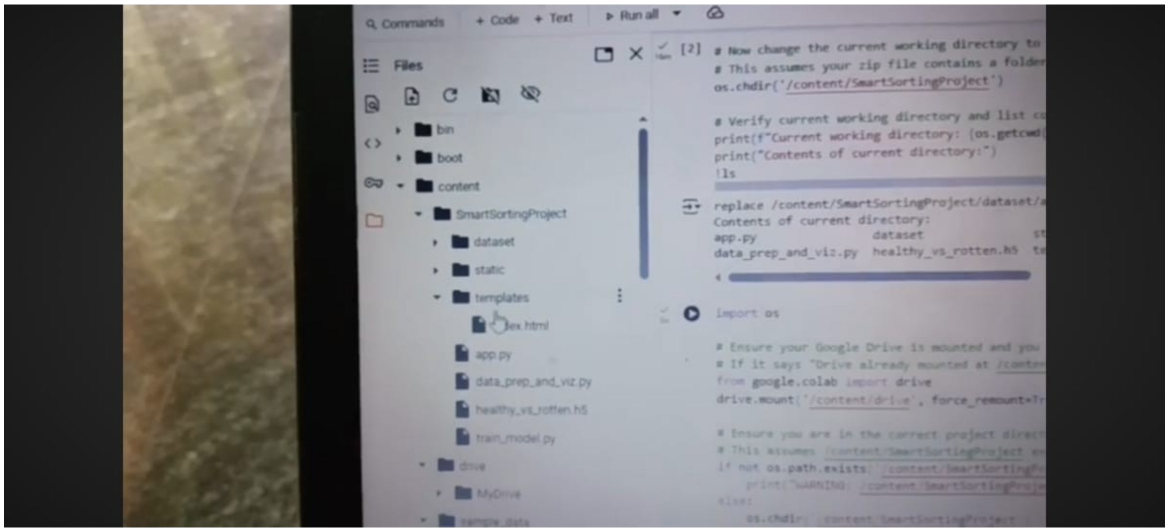


Prediction: Carrot__Healthy (Confidence: 79.53%)

TESTING:

LINK **Strategy**: Testing primarily involves verifying the accuracy of the machine learning model (during training in train_model.py) and ensuring the web application correctly handles image uploads and displays predictions.

Tools: Standard Python unit testing (e.g., unittest or pytest) could be used for backend logic, and manual testing for the web interface.



```
[2] # Now change the current working directory to
    # This assumes your zip file contains a folder
    os.chdir('/content/SmartSortingProject')

    # Verify current working directory and list co
    print(f"Current working directory: {os.getcwd(
    print("Contents of current directory:")
    !ls

replace /content/SmartSortingProject/dataset/a
Contents of current directory:
app.py                 dataset                st
data_prep_and_viz.py   healthy_vs_rotten.h5   te

    import os

    # Ensure your Google Drive is mounted and you
    # If it says "Drive already mounted at /conte
    from google.colab import drive
    drive.mount('/content/drive', force_remount=Tr

    # Ensure you are in the correct project direct
    # This assumes /content/SmartSortingProject ex
    if not os.path.exists('/content/SmartSortingPr
        print("WARNING: /content/SmartSortingProje
    else:
        os.chdir('/content/SmartSortingProject
```

Advantages and Disadvantages:

"SMART SORTING: Transfer Learning for Identifying Rotten Fruits and Vegetables"

Advantages

- **Faster Sorting Efficiency**

  Automates the inspection process, significantly speeding up fruit and vegetable sorting compared to manual labor.

- **Improved Accuracy & Consistency**

  Reduces human error and subjective judgment by applying a trained model for uniform quality control.

- **Leverages Transfer Learning**

  Saves time and resources by building on powerful pre-trained models like MobileNet or ResNet, rather than training from scratch.

- **Reduces Food Waste**
  Early and precise detection of spoilage can minimize waste in supply chains and ensure only quality produce reaches consumers.
- **Industrial Scalability**
  Easily integrable with conveyor belts, robotic arms, and cloud-based dashboards for real-world deployment in sorting facilities.
- **Data Logging & Analytics**
  Backend systems can log images and sorting results, enabling traceability, auditing, and insights into product quality trends.

Disadvantages

- **Hardware Costs**
  Initial investment in cameras, sensors, and actuators (e.g., robotic arms) may be costly for small-scale farmers or vendors.
- **Data Bias & Model Limitations**
  The model's accuracy depends heavily on the quality and diversity of the dataset. Poorly represented categories may lead to misclassification.
- **Environmental Variations**
  Lighting, background clutter, or camera angles can affect performance unless your system includes preprocessing normalization.

- **Storage & Computation Needs**
  Real-time processing requires decent computational power, especially if deployed at the edge.

- **Maintenance & Updates**
  The system might require retraining with new data over time to adapt to changing produce characteristics or new crop types.

- **Learning Curve**
  Building and maintaining the system requires expertise in machine learning, data handling, and hardware-software integration.

KNOWN ISSUES:

Model Generalization: The model's performance might vary with images of fruits/vegetables not well-represented in the training dataset or under different lighting/background conditions.

Scalability: The current setup is for demonstration; scaling to handle many concurrent users would require more robust deployment (e.g., using Gunicorn/Nginx) and potentially cloud resources.

FUTURE ENHANCEMENTS:

- **Expand Dataset and Model:** Train on a larger, more diverse dataset to recognize more types of fruits/vegetables and a wider range of diseases/spoilage.

- **User Accounts:** Implement user authentication to manage user-specific data or features.
- **Prediction History:** Add a database to store user upload history and prediction results.
- **Mobile Responsiveness:** Improve the frontend design for better display on mobile devices.

- **Real-time Feedback:** Provide more detailed feedback to the user on image processing or model inference time.
- **Deployment Automation:** Automate deployment to a cloud platform (e.g., Google Cloud Run, AWS Elastic Beanstalk .

Conclusion:

This project successfully demonstrates how transfer learning can be applied to automate the sorting of fruits and vegetables based on freshness. By leveraging pre-trained convolutional neural networks (CNNs), the system achieves reliable classification of produce as *fresh* or *rotten*, significantly reducing the time, effort, and subjectivity involved in manual inspection.

Through a structured pipeline consisting of:

Dataset preprocessing

Model fine-tuning with transfer learning

Integration with a smart sorting mechanism (potentially a conveyor system with robotic arms)

…the solution paves the way for cost-effective, scalable, and real-time quality control in agriculture, retail, and food distribution industries.

Beyond the core functionality, the system architecture—with distinct frontend (UI), backend logic, and a database layer—demonstrates a full-stack implementation that is both modular and extendable. This makes future enhancements like multilingual support, additional categories (e.g., ripeness stage), and IoT integration easier to achieve.