Smart Sorting: Identifying Rotten Fruits and Vegetables Using Transfer Learning

1. INTRODUCTION

1.1 Project Overview
This project presents a smart AI-based classification system for identifying rotten fruits and vegetables using transfer learning. A web-based tool was developed that enables users to upload images of produce and receive immediate classification results indicating whether the item is fresh or rotten. This application promotes food quality control, waste reduction, and consumer awareness.

1.2 Purpose
To build a fast, reliable, and accessible AI-driven decision support system that classifies fruits and vegetables as fresh or rotten via an intuitive web interface, thereby contributing to improved food safety and reduced waste.

2. IDEATION

2.1 Problem Statement
Manually identifying spoiled produce is error-prone and time-consuming, especially in markets, storage centers, and homes. This project addresses the need for a scalable, quick, and non-expert-friendly solution to distinguish fresh from rotten produce using advanced computer vision models.

2.2 Empathy Map Canvas
Target users include vendors, consumers, warehouse personnel, and quality controllers. - **Says**: "I need to know if this is still good to eat." - **Thinks**: "I want to reduce waste and ensure food safety." - **Does**: Examines produce by sight and touch. - **Feels**: Concerned, unsure, responsible, and sometimes frustrated by misjudgments.

2.3 Brainstorming
Explored solutions: - Manual inspection checklist - Mobile scanner with sensors - AI-powered web application using image classification

Selected the AI-powered image classifier for its speed, reliability, and ease of deployment.

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map
- User accesses the smart sorting web app. - Uploads a photo of a fruit or vegetable. - System classifies the image as fresh or rotten. - Displays result with confidence score; user can save or share it.

3.2 Solution Requirement - **Dataset**: Labeled images of fresh and rotten produce. - **Modeling**: Transfer learning using pre-trained CNN (e.g., MobileNetV2, ResNet50). - **Frontend**: HTML/CSS interface for user interaction. - **Backend**: Flask app handling preprocessing, model inference, and output rendering. - **Version Control**: Codebase maintained on GitHub.

3.3 Data Flow Diagram

Image Dataset → Preprocessing & Augmentation → Transfer Learning Model Training → Flask App → User Upload → Model Inference → Output Display

3.4 Technology Stack

- **Language & Libraries**: Python, TensorFlow/Keras, OpenCV, Pillow - **Framework**: Flask - **Frontend**: HTML, CSS - **Deployment**: Localhost/Cloud - **Repository**: Git & GitHub

4. PROJECT DESIGN

4.1 Problem Solution Fit

This system reduces food quality uncertainty by providing instant, accessible classification of produce, even in low-resource environments. It is beneficial for everyday users and professionals alike.

4.2 Proposed Solution

A Flask-based web application where users upload images (JPG, PNG) and receive classification (fresh/rotten) with confidence scores. The system uses a fine-tuned pre-trained CNN for inference.

4.3 Solution Architecture

- **Frontend**: Upload interface and prediction display - **Backend**: Flask server with preprocessing and ML inference logic - **ML Model**: Transfer-learned CNN model (e.g., MobileNetV2) - **Output**: Result rendered via Flask templates

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

2-week timeline organized as: - Days 1–2: Requirement gathering, dataset collection - Days 3–6: Model training using transfer learning - Days 7–9: Flask integration and frontend development - Days 10–11: Testing and validation - Days 12–14: Documentation and final deployment

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing - **Model Accuracy**: Achieved ~88% accuracy on validation set - **Evaluation**: Precision, recall, F1-score, confusion matrix - **Input Handling**: Validates correct image formats, rejects corrupted files - **Speed**: Classifications returned within 2 seconds - **Browser Testing**: Chrome, Firefox, Edge compatibility confirmed

7. RESULTS

7.1 Output Screenshots

The application allows users to: - Upload JPG/PNG images via a drag-and-drop interface - View real-time predictions (Fresh or Rotten) with confidence levels - Download or save results for later reference

8. ADVANTAGES & DISADVANTAGES

Advantages - **Efficient**: Quick identification reduces manual inspection time - **Accessible**: Useful for anyone with internet access - **Cost-Saving**: Prevents unnecessary wastage or spoilage

Disadvantages - **Model Sensitivity**: Dependent on quality/diversity of training data - **Binary Output**: Only classifies as Fresh/Rotten, not levels of spoilage - **No Expert Override**: AI-only prediction lacks expert validation

9. CONCLUSION This project demonstrates how transfer learning and web technology can simplify real-world quality control challenges. The smart sorting web app empowers users to assess produce freshness rapidly and reliably, encouraging better food handling practices.

The system provides a technically robust yet user-friendly interface, allowing non-experts to benefit from deep learning models. It is a practical tool for food safety, logistics, and sustainability efforts.

10. FUTURE SCOPE - **Model Enhancement**: Larger datasets with varied lighting and angles - **Prediction Detail**: Add multi-stage spoilage detection - **Mobile Deployment**: Launch as a mobile app for field use - **User Feedback Loop**: Integrate active learning based on user corrections - **Partnerships**: Collaborate with food safety authorities and researchers

11. APPENDIX - **A. Dataset Source**: Public datasets (e.g., Fruits-360, Kaggle produce sets) - **B. Tools & Libraries**: Python, Keras, TensorFlow, Flask, HTML/CSS - **C. GitHub Repository**: [https://github.com/YourUsername/Smart_Sorting_Classifier]