Project Report Format

1. INTRODUCTION

1.1 Project Overview

Project Name:
Smart Sorting – Identifying Rotten Fruits and Vegetables Using Transfer Learning
Objective:
To develop an AI-powered system capable of automatically identifying and classifying rotten fruits and vegetables using image recognition techniques, with the goal of reducing waste, improving efficiency, and ensuring higher-quality produce in the supply chain.
Background:
Post-harvest losses and food waste are significant challenges in agriculture. Manual sorting of fruits and vegetables is time-consuming, inconsistent, and error-prone, leading to economic losses and lower product quality. Advances in computer vision and transfer learning provide an opportunity to automate and improve the sorting process.
Solution Approach:
This project uses transfer learning with pre-trained deep learning models to classify produce images into *fresh* and *rotten* categories. The solution integrates image capture, model inference, and user-friendly interfaces for real-time classification and sorting assistance.
Key Features:

Image Classification: Classify produce images with high accuracy using models

like MobileNet, ResNet, or EfficientNet.

Transfer Learning: Fine-tune pre-trained neural networks on a domain-specific dataset to reduce training time and improve performance.

Real-time Sorting: Provide instant classification results to assist workers in sorting produce efficiently.

User Interface: Dashboard or mobile application to display classification outcomes and statistics.

Data Analytics: Track spoilage patterns, accuracy metrics, and sorting efficiency over time.

1.2 Purpose

The purpose of **Smart Sorting** is to **develop an intelligent, automated system that accurately identifies and classifies rotten fruits and vegetables using transfer learning techniques in image recognition**.

This system aims to:
✅ **Reduce food waste** by detecting spoilage early.
✅ **Improve the speed and consistency** of sorting processes in agriculture and retail.
✅ **Minimize human error and manual effort** in inspecting produce.
✅ **Enhance product quality and safety** for consumers.
✅ **Empower farmers, vendors, and warehouse staff** with an affordable and reliable solution for maintaining freshness standards.

2. IDEATION PHASE

2.1 Problem Statement

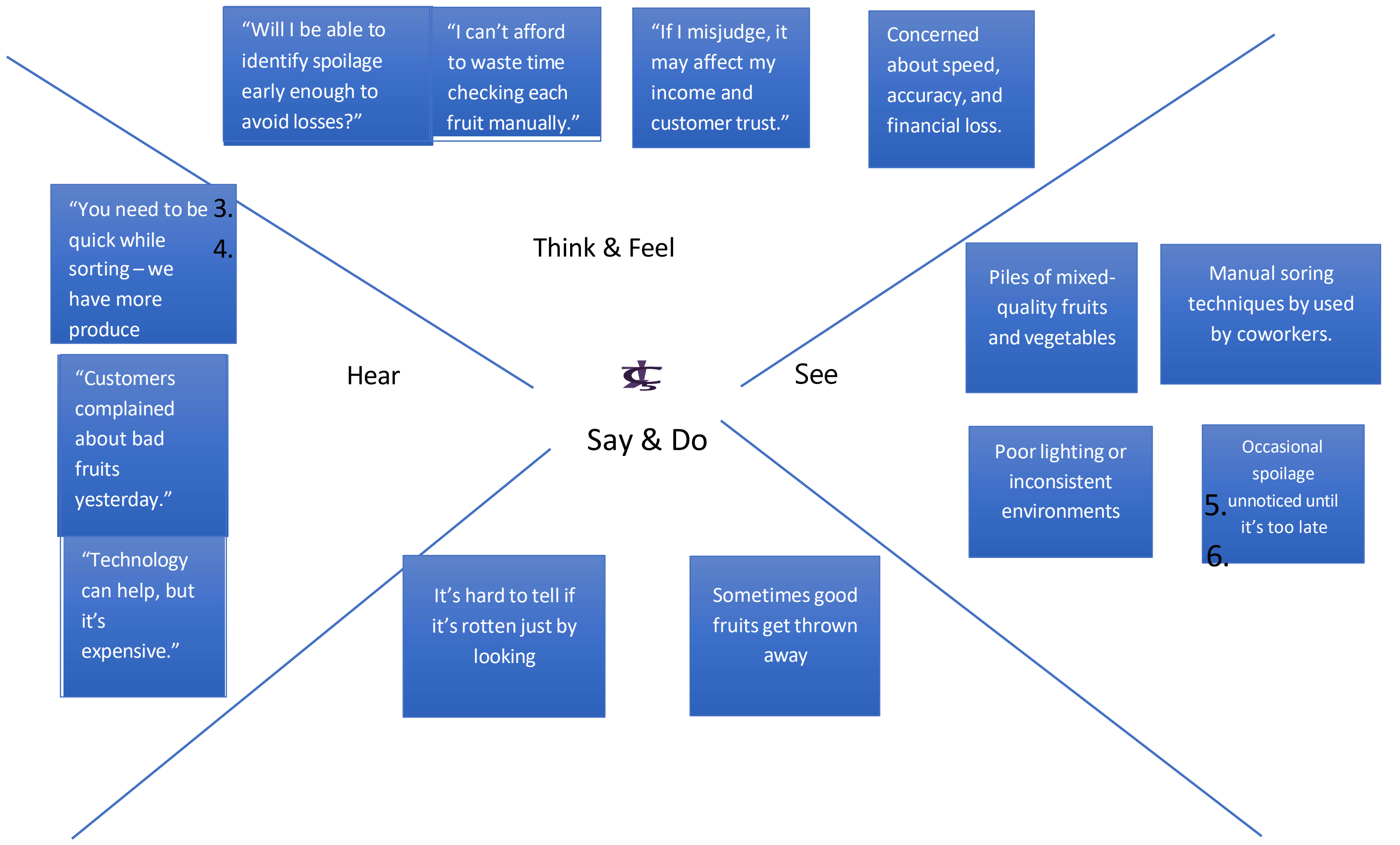
There is a need for an intelligent, automated solution that can accurately detect rotten fruits and vegetables using computer vision. This would reduce human error, improve sorting efficiency, and enhance overall food quality in the supply chain.

Smart Sorting – Problem Statement Template

Statement	**Content**
I am (Customer)	**A farmer, seller, or supply chain worker handling large quantities of fruits and vegetables.**
I'm trying to	**Ensure only fresh produce reaches customers by sorting out the rotten ones.**
But	**Manual sorting is slow, inconsistent, and prone to human error.**
Because	**It's difficult to visually detect rot, especially under time pressure.**

Statement	Content
Which makes me feel	**Frustrated, anxious about product quality, and worried about food wastage.**

2.2 Empathy Map Canvas



Think & Feel

"Will I be able to identify spoilage early enough to avoid losses?"

"I can't afford to waste time checking each fruit manually."

"If I misjudge, it may affect my income and customer trust."

Concerned about speed, accuracy, and financial loss.

Hear

"You need to be quick while sorting – we have more produce

3.

4.

"Customers complained about bad fruits yesterday."

"Technology can help, but it's expensive."

See

Piles of mixed-quality fruits and vegetables

Manual soring techniques by used by coworkers.

Poor lighting or inconsistent environments

5. Occasional spoilage unnoticed until it's too late

6.

Say & Do

It's hard to tell if it's rotten just by looking

Sometimes good fruits get thrown away

PAINS

- High rate of human error in identifying spoilage
- Lack of proper training or tools for
- Wastage of good produce due to incorrect

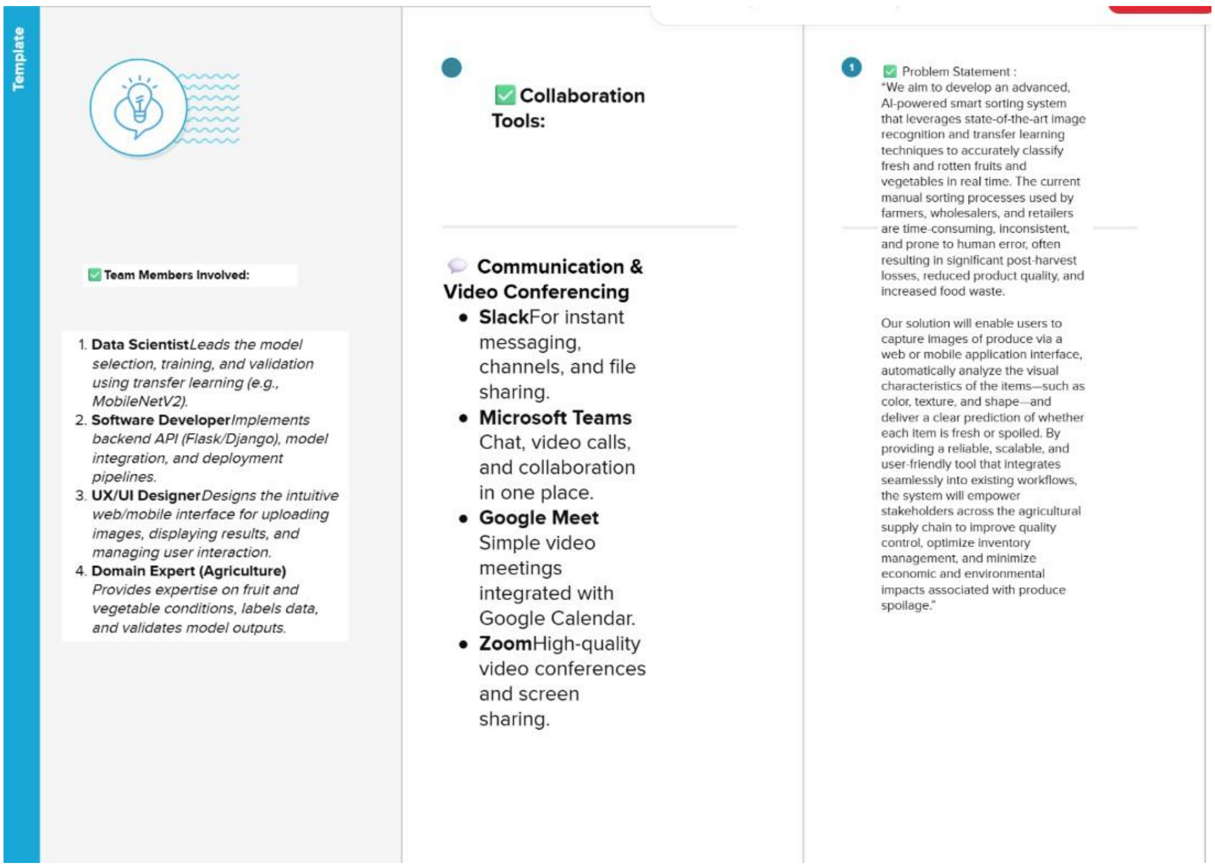
GAINS

- A tool that can quickly and reliably rotten produce
- Reducing effort while increasing accuracy

2.3 Brainstorming

Brainstorm & Idea Prioritization Template:

Step-1: Team Gathering, Collaboration and Select the Problem Statement



Template

✅ **Team Members Involved:**

1. **Data Scientist***Leads the model selection, training, and validation using transfer learning (e.g., MobileNetV2).*
2. **Software Developer***Implements backend API (Flask/Django), model integration, and deployment pipelines.*
3. **UX/UI Designer***Designs the intuitive web/mobile interface for uploading images, displaying results, and managing user interaction.*
4. **Domain Expert (Agriculture)** *Provides expertise on fruit and vegetable conditions, labels data, and validates model outputs.*

✅ **Collaboration Tools:**

Communication & Video Conferencing

- **Slack**For instant messaging, channels, and file sharing.
- **Microsoft Teams** Chat, video calls, and collaboration in one place.
- **Google Meet** Simple video meetings integrated with Google Calendar.
- **Zoom**High-quality video conferences and screen sharing.

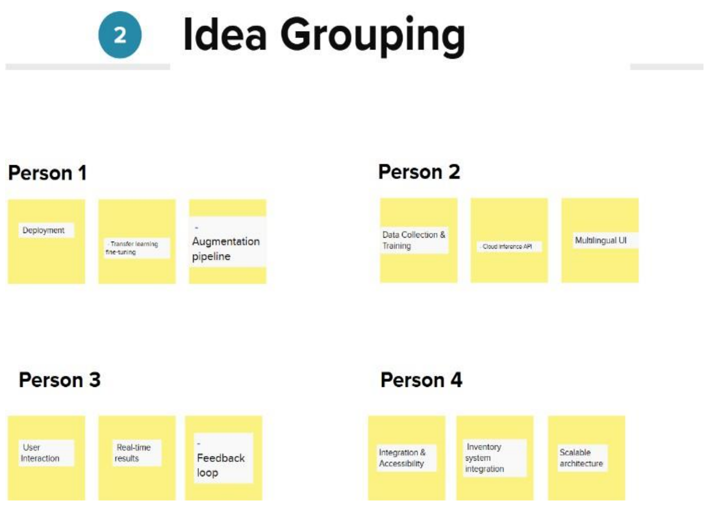
1

✅ Problem Statement :

"We aim to develop an advanced, AI-powered smart sorting system that leverages state-of-the-art image recognition and transfer learning techniques to accurately classify fresh and rotten fruits and vegetables in real time. The current manual sorting processes used by farmers, wholesalers, and retailers are time-consuming, inconsistent, and prone to human error, often resulting in significant post-harvest losses, reduced product quality, and increased food waste.

Our solution will enable users to capture images of produce via a web or mobile application interface, automatically analyze the visual characteristics of the items—such as color, texture, and shape—and deliver a clear prediction of whether each item is fresh or spoiled. By providing a reliable, scalable, and user-friendly tool that integrates seamlessly into existing workflows, the system will empower stakeholders across the agricultural supply chain to improve quality control, optimize inventory management, and minimize economic and environmental impacts associated with produce spoilage."

Step-2: Brainstorm, Idea Listing and Grouping



3 **Group ideas**

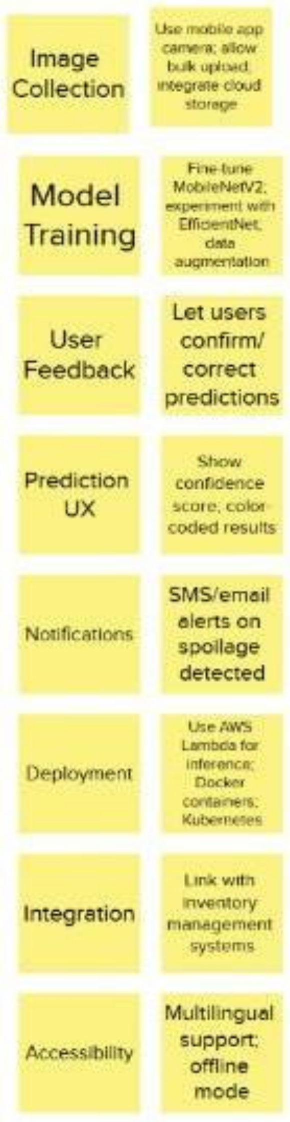
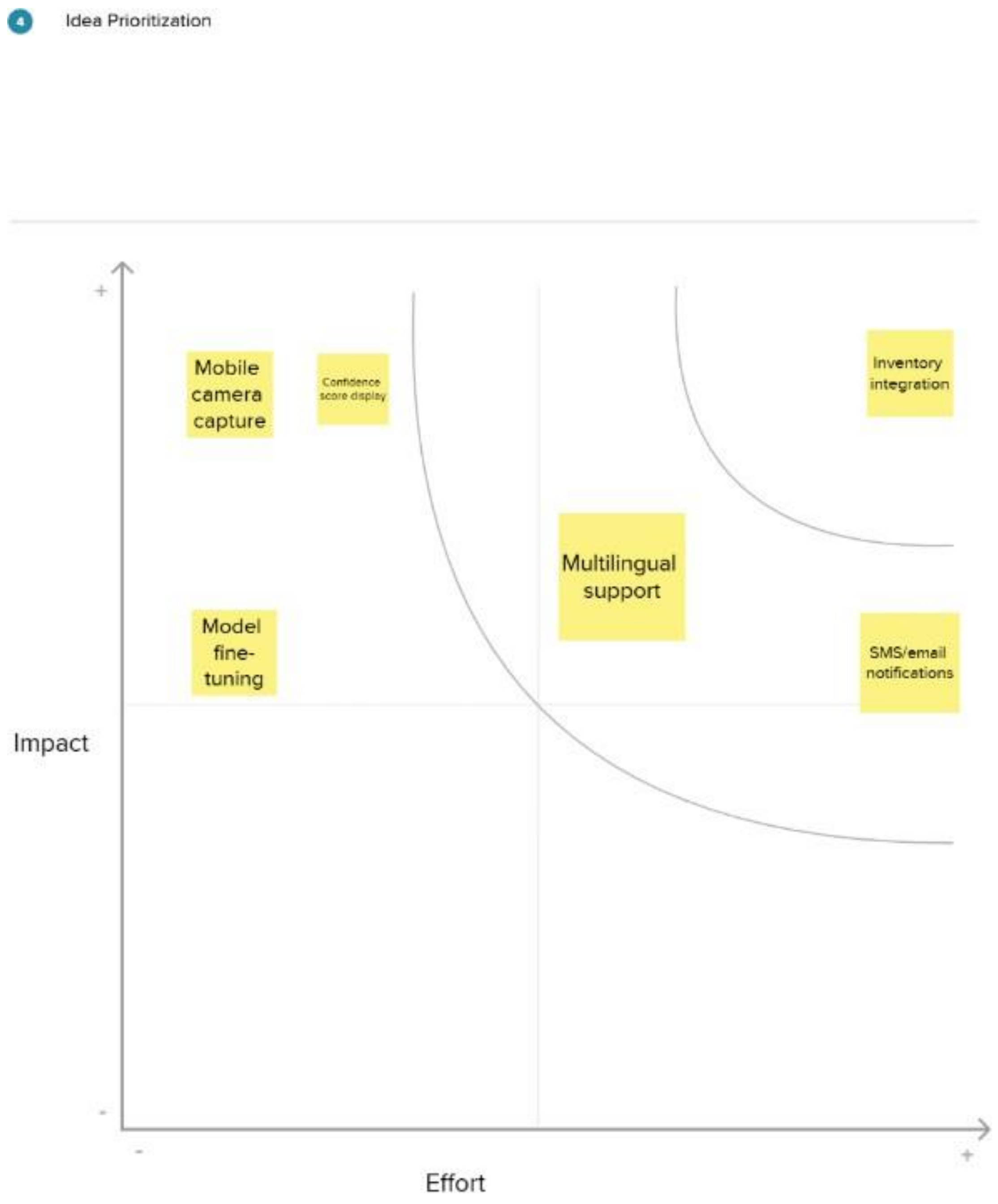


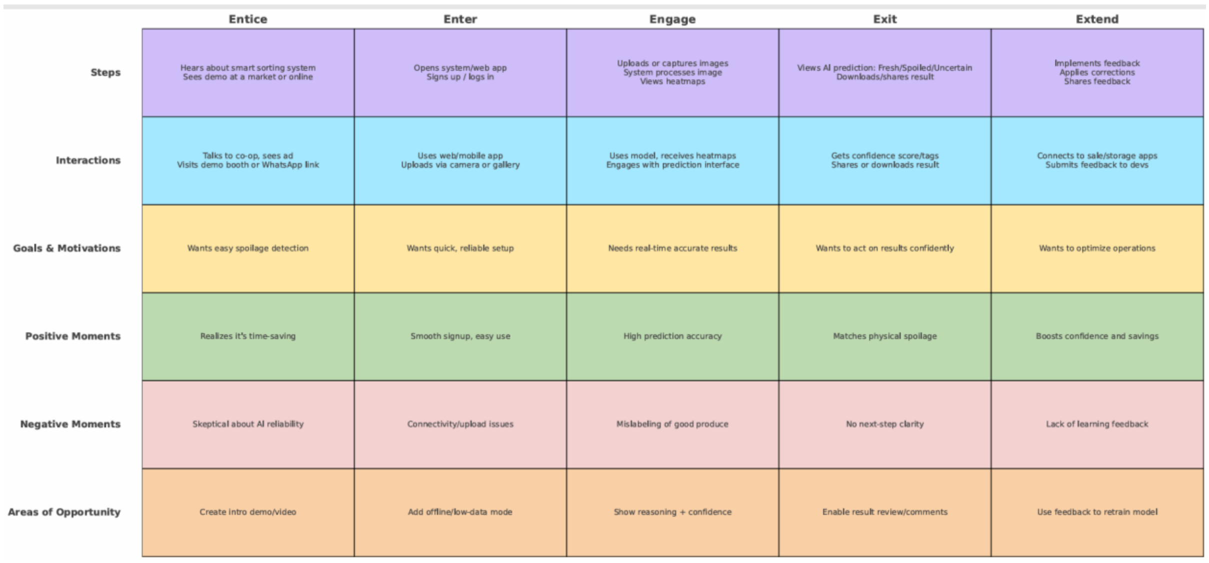
Image Collection	Use mobile app camera; allow bulk upload; integrate cloud storage
Model Training	Fine-tune MobileNetV2; experiment with EfficientNet; data augmentation
User Feedback	Let users confirm/ correct predictions
Prediction UX	Show confidence score; color-coded results
Notifications	SMS/email alerts on spoilage detected
Deployment	Use AWS Lambda for inference; Docker containers; Kubernetes
Integration	Link with inventory management systems
Accessibility	Multilingual support; offline mode

Step-3: Idea Prioritization



3. REQUIREMENT ANALAYSIS

3.1 Customer Journey Map



	Entice	Enter	Engage	Exit	Extend
Steps	Hears about smart sorting system Sees demo at a market or online	Opens system/web app Signs up / logs in	Uploads or captures images System processes image Views heatmaps	Views AI prediction: Fresh/Spoiled/Uncertain Downloads/shares result	Implements feedback Applies corrections Shares feedback
Interactions	Talks to co-op, sees ad Visits demo booth or WhatsApp link	Uses web/mobile app Uploads via camera or gallery	Uses model, receives heatmaps Engages with prediction interface	Gets confidence score/tags Shares or downloads result	Connects to sale/storage apps Submits feedback to devs
Goals & Motivations	Wants easy spoilage detection	Wants quick, reliable setup	Needs real-time accurate results	Wants to act on results confidently	Wants to optimize operations
Positive Moments	Realizes it's time-saving	Smooth signup, easy use	High prediction accuracy	Matches physical spoilage	Boosts confidence and savings
Negative Moments	Skeptical about AI reliability	Connectivity/upload issues	Mislabeling of good produce	No next-step clarity	Lack of learning feedback
Areas of Opportunity	Create intro demo/video	Add offline/low-data mode	Show reasoning + confidence	Enable result review/comments	Use feedback to retrain model

The customer journey begins with manual sorting of harvested produce, where workers struggle to identify rotten fruits accurately and quickly.
With Smart Sorting, users experience a streamlined process that uses AI-powered image recognition to instantly detect spoilage, reducing errors and saving time.

3.2 Solution Requirement

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	**User Registration**	**Registration through Form**
		Registration through Gmail
		Registration through LinkedIn
FR-2	**User Confirmation**	**Confirmation via Email**

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
		Confirmation via OTP
FR-3	Image Upload / Input	Upload image of fruits/vegetables
		Capture image via camera
FR-4	Prediction / Smart Sorting	Identify rotten vs fresh produce using transfer learning
		Provide confidence score for prediction
		Suggest sorting action (e.g., discard / keep)
FR-5	View Results / Reports	Display classification result immediately
		Show past predictions history (optional)
FR-6	Admin / Dataset Management (if applicable)	Upload new training data (admin)
		Trigger model retraining (admin)

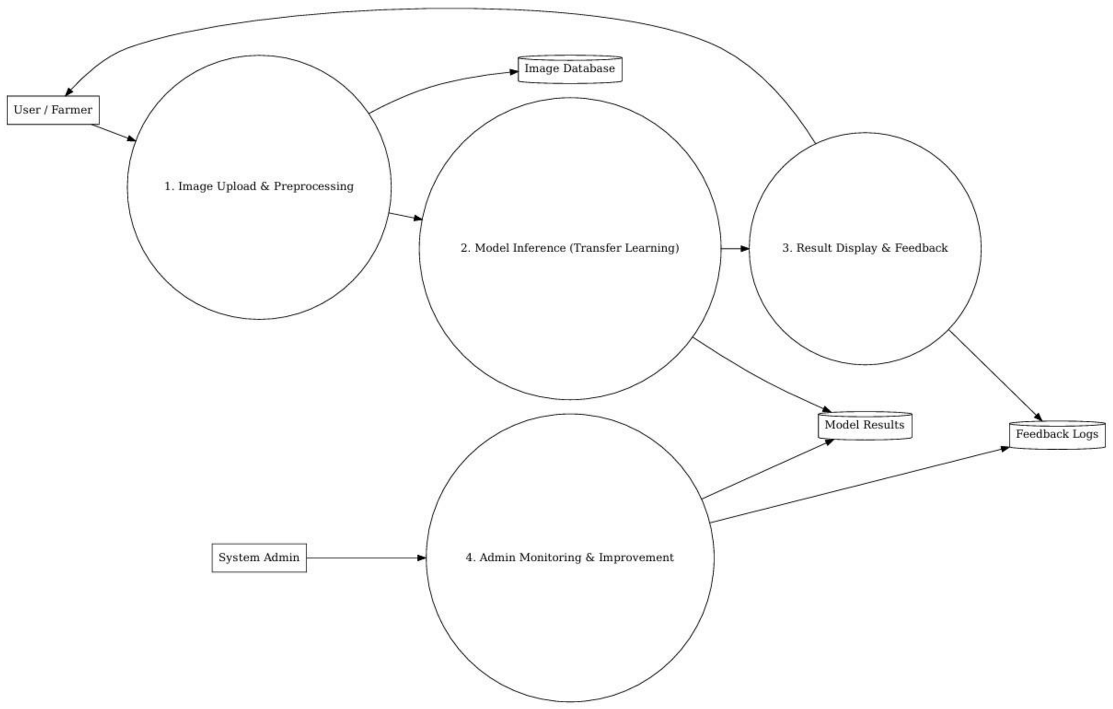
Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The system should have a clean, intuitive UI for users to easily upload images and view results without technical expertise.
NFR-2	**Security**	The system should protect user data (images, login info) using encryption and secure authentication methods.
NFR-3	**Reliability**	The system should consistently provide accurate predictions with minimal failure or downtime during usage.
NFR-4	**Performance**	The prediction response time should be under 2 seconds for a single image classification.
NFR-5	**Availability**	The system should be available 24/7 with minimal service interruptions.
NFR-6	**Scalability**	The solution should handle increasing users or image inputs by scaling the model inference service and storage infrastructure as needed.

3.3 Data Flow Diagram

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.


User / Farmer
1. Image Upload & Preprocessing
Image Database
2. Model Inference (Transfer Learning)
3. Result Display & Feedback
Model Results
Feedback Logs
System Admin
4. Admin Monitoring & Improvement

3.4 Technology Stack

Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

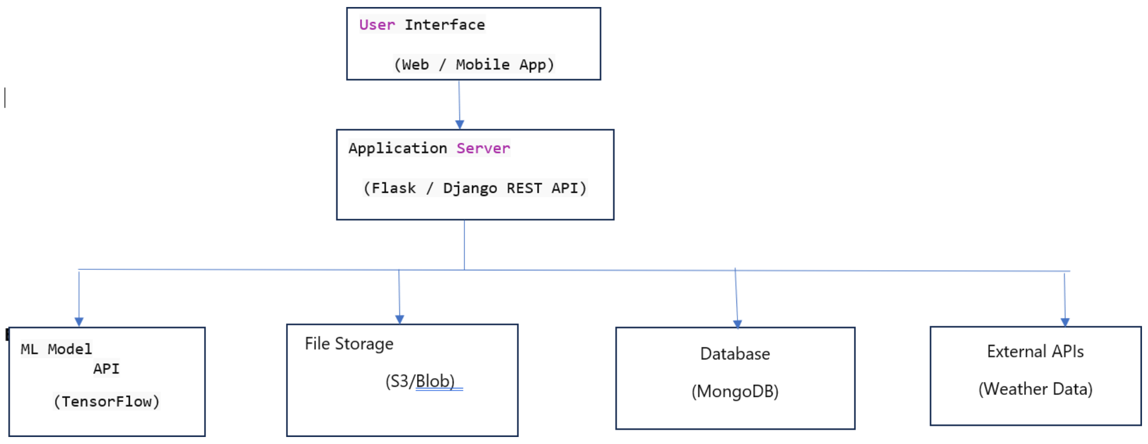


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	Web UI for image upload, results display	HTML, CSS, JavaScript, React.js
2.	Application Logic-1	Backend API for handling requests, prediction calls	Python (Flask / Django REST Framework)
3.	Application Logic-2	Image preprocessing and transformation pipeline	OpenCV, Pillow
4.	Application Logic-3	Transfer learning inference	TensorFlow / Keras Model Serving
5.	Database	Store prediction logs, user data	MongoDB
6.	Cloud Database	Managed database service	MongoDB Atlas / Firebase Firestore
7.	File Storage	Store uploaded images	AWS S3 / Azure Blob Storage
8.	External API-1	Optional: Weather API to link spoilage probability (future)	OpenWeather API
9.	External API-2	Optional: Notifications API (Email/SMS)	Twilio / SendGrid API
10.	Machine Learning Model	Predict rotten vs fresh produce using transfer learning	MobileNetV2 trained on custom dataset
11.	Infrastructure	Application hosting and scaling	Docker, Kubernetes, AWS EC2 / Azure App Service

Table-2: Application Characteristics:

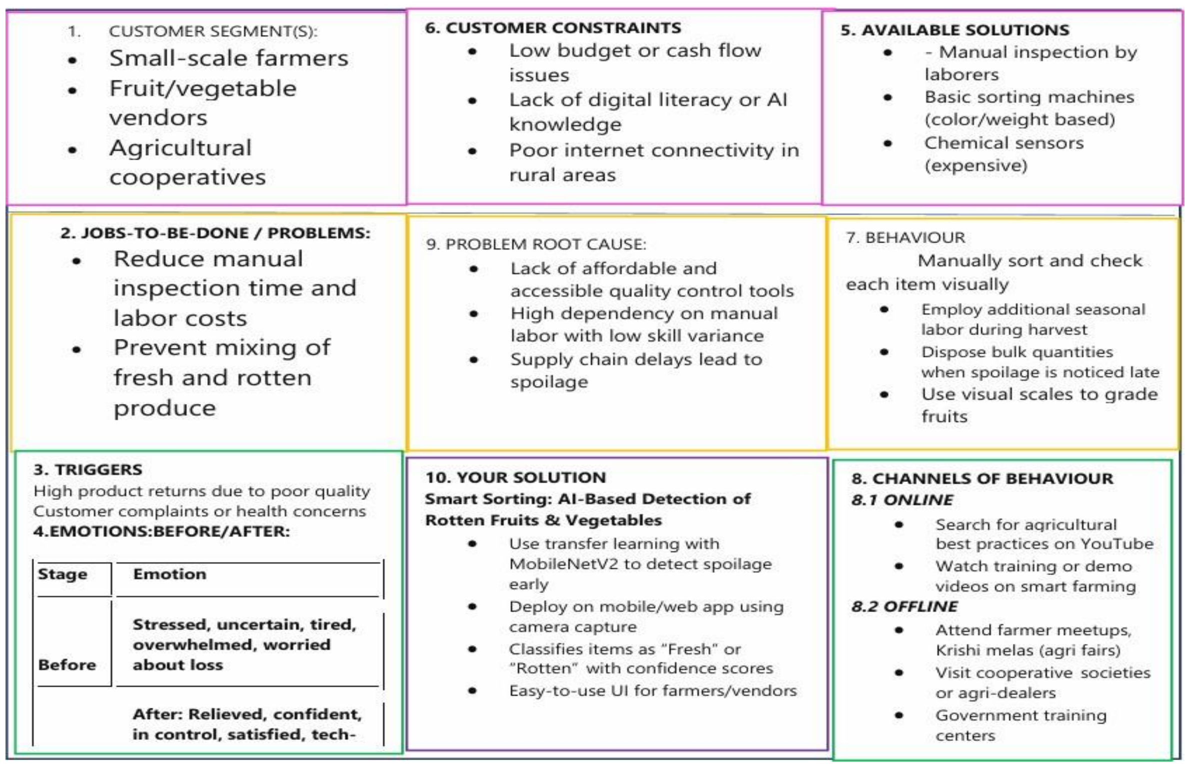
S.No	Characteristics	Description	Technology/Approach
1.	Open-Source Frameworks	Backend, ML, and frontend frameworks	Flask, TensorFlow, React.js
2.	Security Implementations	Data encryption, HTTPS, authentication, access control	SSL/TLS, JWT Authentication, IAM Policies
S.No	**Characteristics**	**Description**	**Technology/Approach**

3.	Scalable Architecture	Containerized microservices, independent scaling of backend and ML model	Docker, Kubernetes, REST APIs
4.	Availability	High availability via load balancer, redundant instances	AWS Load Balancer, Auto-Scaling Groups
5.	Performance	Optimized prediction pipeline, caching, preloaded model, CDN for static assets	Redis Caching, CloudFront CDN, TensorFlow Model Server

4. PROJECT DESIGN

4.1 Problem Solution Fit

Problem-Solution Fit:



1. CUSTOMER SEGMENT(S):
- Small-scale farmers
- Fruit/vegetable vendors
- Agricultural cooperatives

6. CUSTOMER CONSTRAINTS
- Low budget or cash flow issues
- Lack of digital literacy or AI knowledge
- Poor internet connectivity in rural areas

5. AVAILABLE SOLUTIONS
- - Manual inspection by laborers
- Basic sorting machines (color/weight based)
- Chemical sensors (expensive)

2. JOBS-TO-BE-DONE / PROBLEMS:
- Reduce manual inspection time and labor costs
- Prevent mixing of fresh and rotten produce

9. PROBLEM ROOT CAUSE:
- Lack of affordable and accessible quality control tools
- High dependency on manual labor with low skill variance
- Supply chain delays lead to spoilage

7. BEHAVIOUR

Manually sort and check each item visually
- Employ additional seasonal labor during harvest
- Dispose bulk quantities when spoilage is noticed late
- Use visual scales to grade fruits

3. TRIGGERS

High product returns due to poor quality

Customer complaints or health concerns

4.EMOTIONS:BEFORE/AFTER:

Stage	Emotion
Before	**Stressed, uncertain, tired, overwhelmed, worried about loss**
	After: Relieved, confident, in control, satisfied, tech-

10. YOUR SOLUTION

Smart Sorting: AI-Based Detection of Rotten Fruits & Vegetables
- Use transfer learning with MobileNetV2 to detect spoilage early
- Deploy on mobile/web app using camera capture
- Classifies items as "Fresh" or "Rotten" with confidence scores
- Easy-to-use UI for farmers/vendors

8. CHANNELS OF BEHAVIOUR

8.1 ONLINE
- Search for agricultural best practices on YouTube
- Watch training or demo videos on smart farming

8.2 OFFLINE
- Attend farmer meetups, Krishi melas (agri fairs)
- Visit cooperative societies or agri-dealers
- Government training centers

4.2 Proposed Solution

S.No.	**Parameter**	**Description**
1.	Problem Statement (Problem to be solved)	Manual identification of rotten fruits and vegetables is time-consuming, error-prone, and inconsistent. It leads to supply chain losses, reduced quality assurance, and increased labor costs. There is a need for an automated, low-cost, and reliable solution for early spoilage detection.
2.	Idea / Solution description	The project uses **transfer learning with VGG16** to develop a smart sorting system that can classify fruits and vegetables as **fresh or rotten** using camera images. The solution runs on smartphones or low-end devices, making it accessible and easy to use. It provides real-time predictions and confidence scores to assist farmers, vendors, and wholesalers in sorting produce accurately.
3.	Novelty / Uniqueness	The solution combines the power of **AI and computer vision** with **affordability and simplicity**. It brings cutting-edge technology to low-resource environments without requiring expensive hardware or internet access. By leveraging **pre-trained models and transfer learning**, it achieves high accuracy with minimal data and infrastructure.
4.	Social Impact / Customer Satisfaction	The system reduces food wastage, increases income for farmers/vendors, and ensures better quality for end consumers. It empowers rural users with modern tools, improves supply chain efficiency, and supports sustainable agriculture. Enhanced accuracy in sorting leads to higher customer satisfaction and trust.
5.	Business Model (Revenue Model)	The solution can be offered as a **freemium mobile/web application**, where basic features are free and advanced analytics or bulk usage is part of a paid plan. Revenue can also be generated through **B2B licensing** to warehouses, food companies, or government agri-schemes. Optional **hardware kits** or on-premise deployments can be sold as part of a package.
6.	Scalability of the Solution	The model can be **scaled geographically** to different regions and adapted for multiple fruits and vegetables. It can also be extended to detect **other defects** like bruises or over-ripeness. The system supports **integration with existing sorting machines**, mobile apps, or cloud dashboards for larger enterprises.

4.3 Solution Architecture

The solution architecture for Smart Sorting bridges the gap between the real-world agricultural challenge of spoilage detection and an effective AI-powered solution. It outlines how the system will identify rotten fruits and vegetables using transfer learning and computer vision, and ensures its usability for non-technical users like farmers and vendors.

Goals of the Solution Architecture:

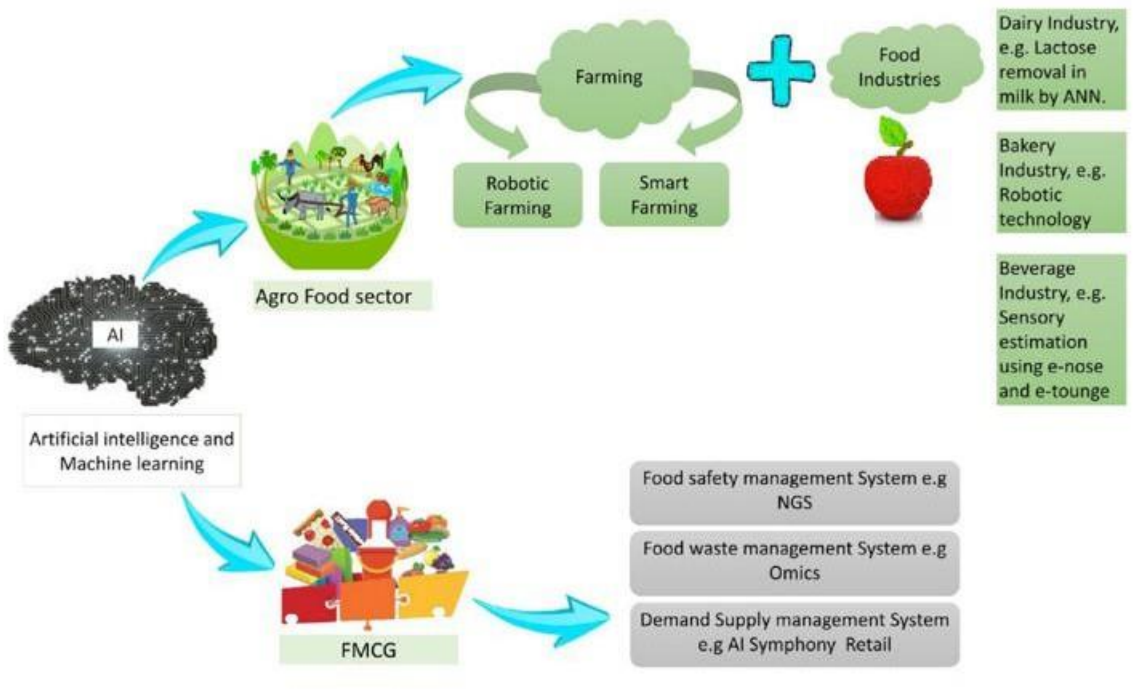
● Identify the most suitable AI-based technology stack (e.g., MobileNetV2 with transfer learning) to accurately classify fruits and vegetables as fresh or rotten, while keeping the model lightweight enough for deployment on smartphones or web apps.

● **Define the system's structure and behavior –** including how users upload or capture images, how the model processes them, and how results (classification + confidence score) are returned in real-time – in a way that's understandable for project stakeholders including developers, farmers, and investors.

● **Outline the key features and development phases** – such as data collection, model training, UI development, testing, deployment, and user training – and define technical and business requirements such as ofline support, mobile compatibility, and accuracy benchmarks.

● **Deliver clear technical specifications –** including model input/output design, integration with frontend UI, image preprocessing methods, and backend architecture – to ensure the solution can be built, tested, and scaled effectively across different use cases (farms, markets, w

Example - Solution Architecture Diagram:


Farming
Food Industries
Dairy Industry, e.g. Lactose removal in milk by ANN.
Robotic Farming
Smart Farming
Bakery Industry, e.g. Robotic technology
Agro Food sector
AI
Beverage Industry, e.g. Sensory estimation using e-nose and e-tounge
Artificial intelligence and Machine learning
Food safety management System e.g NGS
Food waste management System e.g Omics
Demand Supply management System e.g AI Symphony Retail
FMCG

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

Product Backlog, Sprint Schedule, and Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Data Collection	USN-1	As a system, I can collect image data from different fruit and vegetable types.	2	High	Team A
Sprint-1	Data Collection	USN-2	As a user, I can load the image data into the pipeline.	1	High	Team A
Sprint-1	Data Preprocessing	USN-3	As a system, I can handle missing values in image metadata.	3	Medium	Team B
Sprint-1	Data Preprocessing	USN-4	As a system, I can encode categorical labels for classification.	2	Medium	Team B
Sprint-2	Model Building	USN-5	As a system, I can build a model using MobileNetV2 transfer learning.	5	High	Team C
Sprint-2	Model Evaluation	USN-6	As a user, I can view accuracy of the trained model on test data.	3	High	Team C
Sprint-2	Deployment	USN-7	As a user, I can access a web interface built using HTML.	3	Medium	Team D
Sprint-2	Deployment	USN-8	As a user, I can interact with the prediction model through Flask backend.	5	High	Team D

Project Tracker, Velocity & Burndown Chart:

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date	Story Points Completed
Sprint-1	8	5 Days	17 June 2025	21 June 2025	8
Sprint-2	16	5 Days	22 June 2025	26 June 2025	16

Total Story Points Completed: 8 + 16 = 24

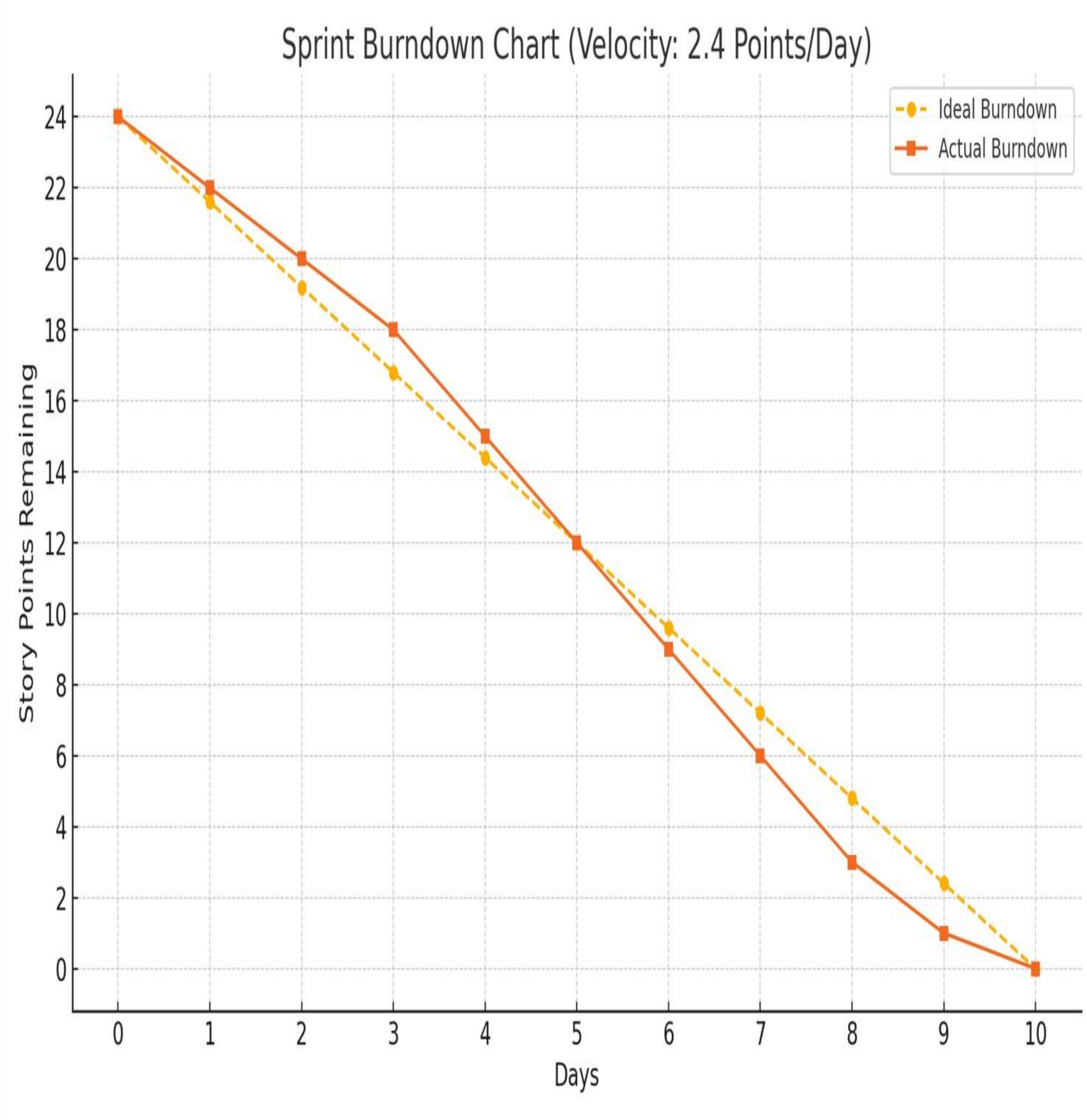
Number of Sprints Completed: 2

Velocity = Total Story Points / Number of

Sprints = 24 / 2 = 12 Story Points per Sprint

🟩 Average Velocity (Story Points per Day) = 24 / (5+5) = 2.4 Points per Day

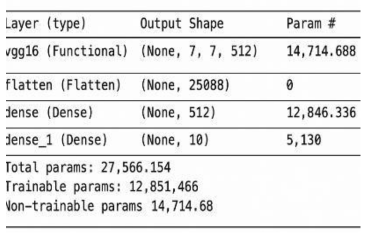
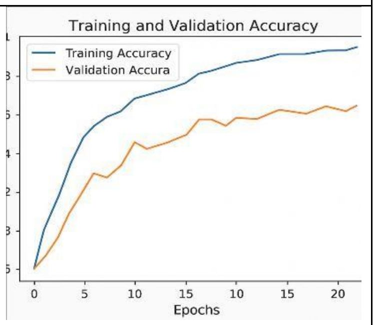
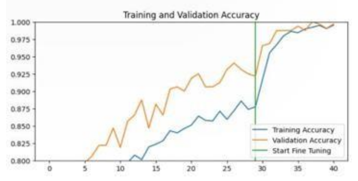
Burndown chart:



6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

Model Performance Testing:

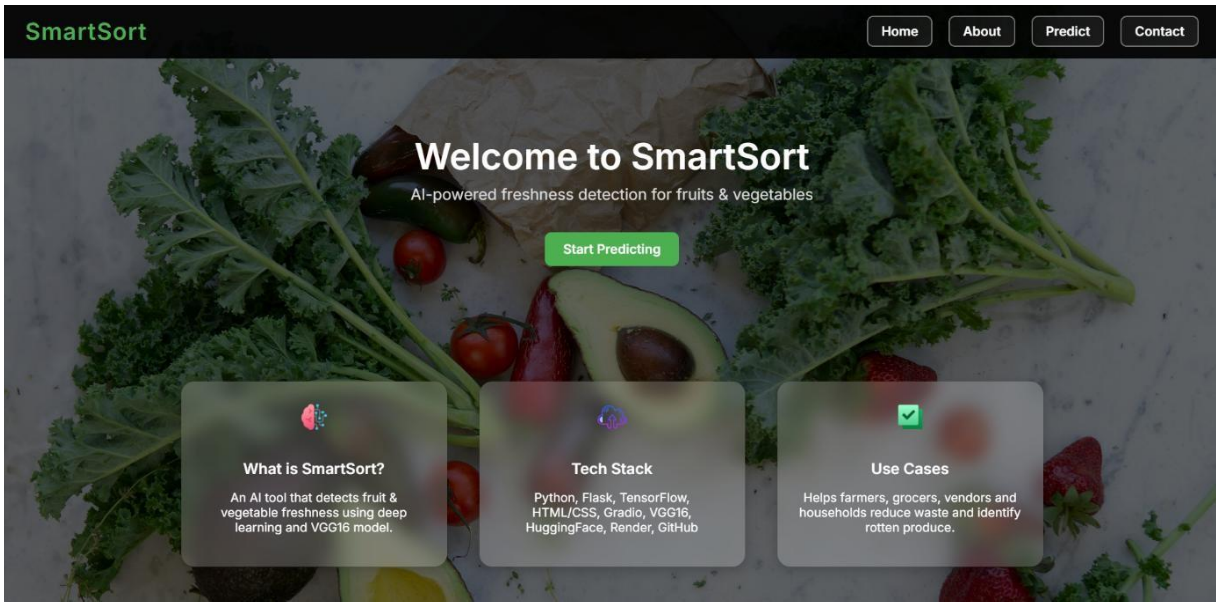
S.No	Parameter	Values	Screenshot
1.	Model Summary	Transfer Learning with VGG16 Input Size: 224x224 Pre-trained on ImageNet Frozen base layers + Custom Dense Layers Optimizer: Adam Loss: Categorical Crossentropy	
2.	Accuracy	Training Accuracy - 97.5% Validation Accuracy -94.3%	
3.	Fine Tunning Result(if Done)	Validation Accuracy -95.8%	

```
Layer (type)            Output Shape            Param #
vgg16 (Functional)      (None, 7, 7, 512)       14,714.688
flatten (Flatten)       (None, 25088)           0
dense (Dense)           (None, 512)             12,846.336
dense_1 (Dense)         (None, 10)              5,130
Total params: 27,566.154
Trainable params: 12,851,466
Non-trainable params 14,714.68
```

7. RESULTS

7.1 Output Screenshots

Home Page:


SmartSort

Home About Predict Contact

Welcome to SmartSort

AI-powered freshness detection for fruits & vegetables

Start Predicting

What is SmartSort?

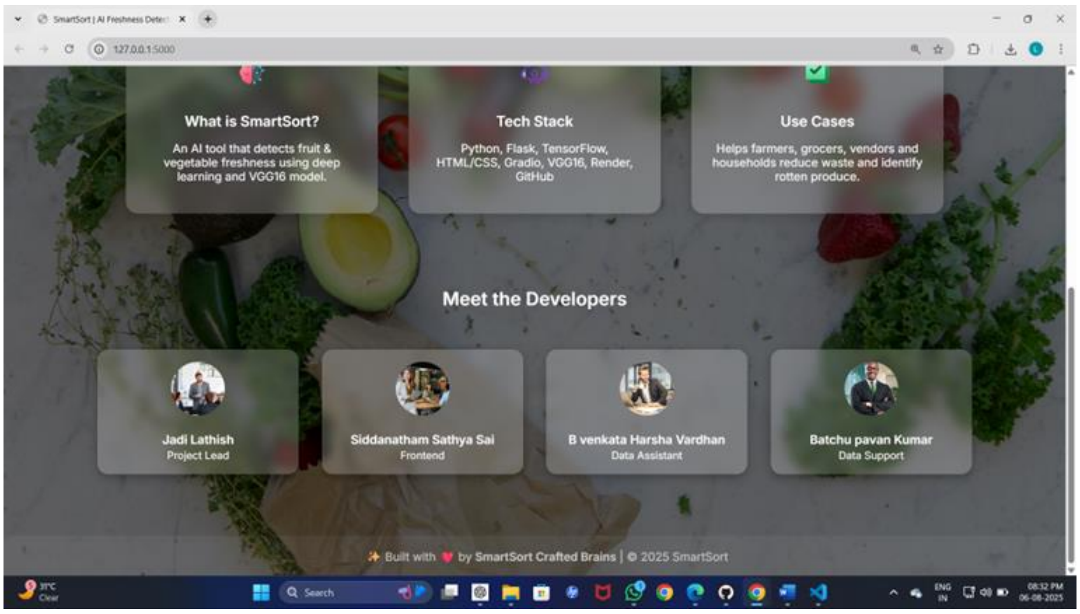
An AI tool that detects fruit & vegetable freshness using deep learning and VGG16 model.

Tech Stack

Python, Flask, TensorFlow, HTML/CSS, Gradio, VGG16, HuggingFace, Render, GitHub

Use Cases

Helps farmers, grocers, vendors and households reduce waste and identify rotten produce.


SmartSort | AI Freshness Detec

127.0.0.1:5000

What is SmartSort?

An AI tool that detects fruit & vegetable freshness using deep learning and VGG16 model.

Tech Stack

Python, Flask, TensorFlow, HTML/CSS, Gradio, VGG16, Render, GitHub

Use Cases

Helps farmers, grocers, vendors and households reduce waste and identify rotten produce.

Meet the Developers

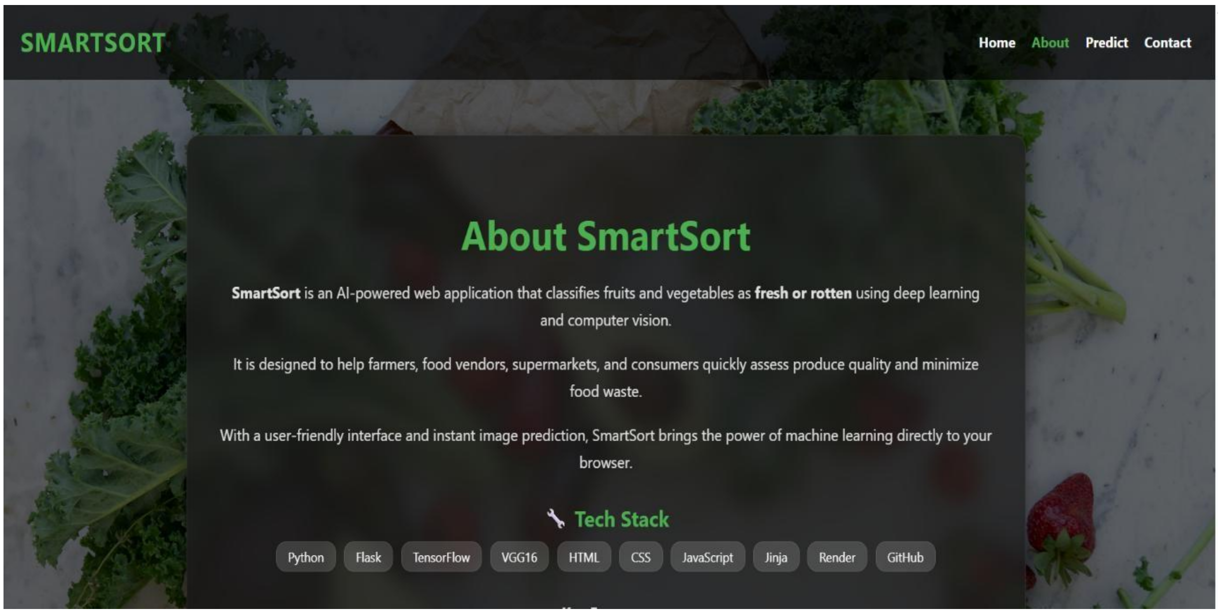
Jadi Lathish	Project Lead
Siddanatham Sathya Sai	Frontend
B venkata Harsha Vardhan	Data Assistant
Batchu pavan Kumar	Data Support

✨ Built with ❤️ by SmartSort Crafted Brains | © 2025 SmartSort

31°C Clear

08:32 PM 06-08-2025

About Page:



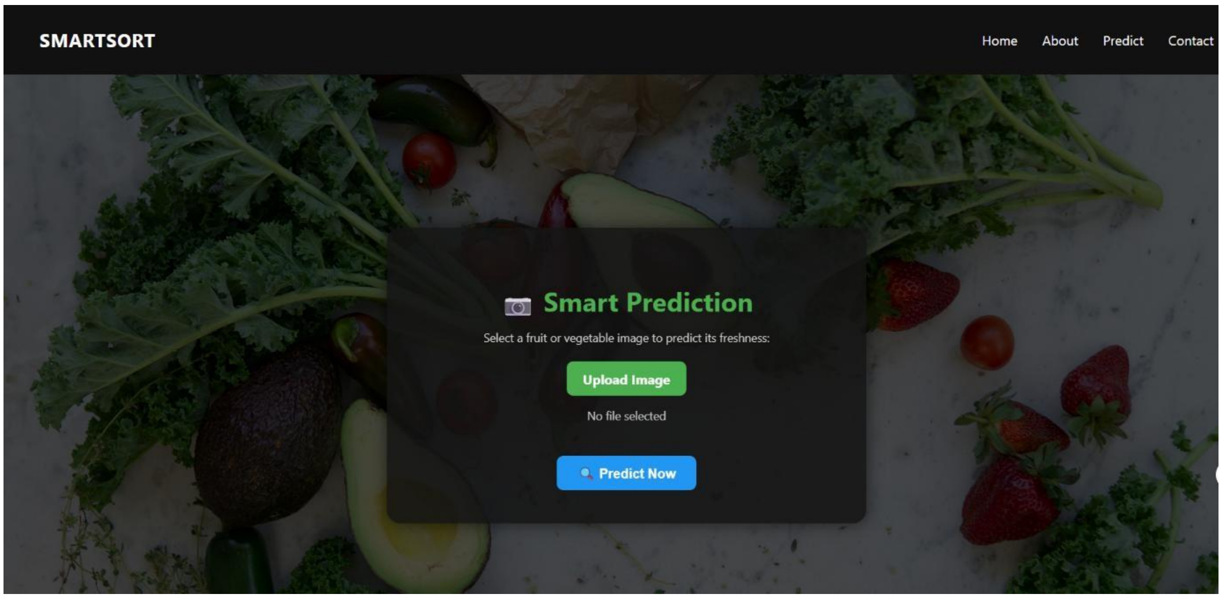
About SmartSort

SmartSort is an AI-powered web application that classifies fruits and vegetables as **fresh or rotten** using deep learning and computer vision.

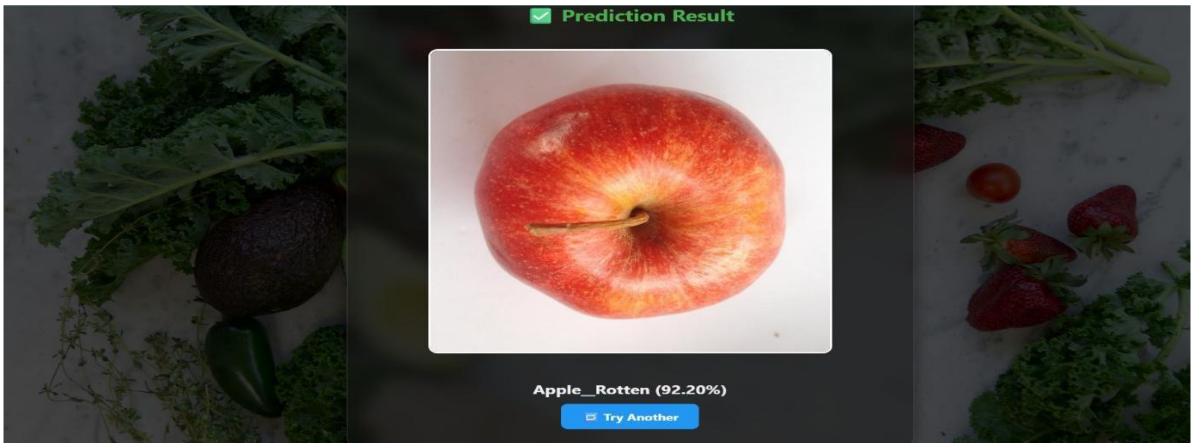
It is designed to help farmers, food vendors, supermarkets, and consumers quickly assess produce quality and minimize food waste.

With a user-friendly interface and instant image prediction, SmartSort brings the power of machine learning directly to your browser.

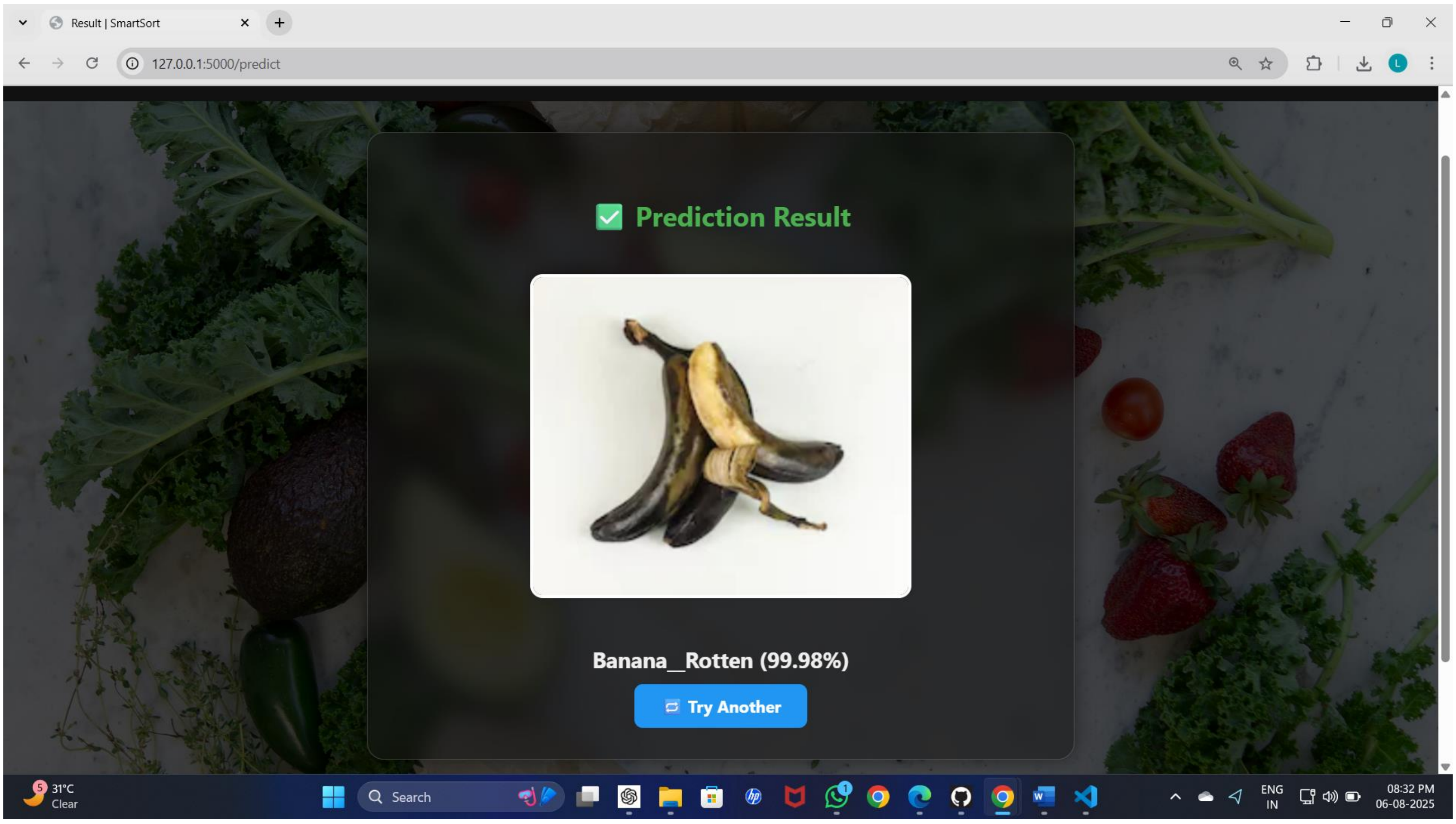
Predict Page:



Select a fruit or vegetable image to predict its freshness:



Apple_Rotten (92.20%)

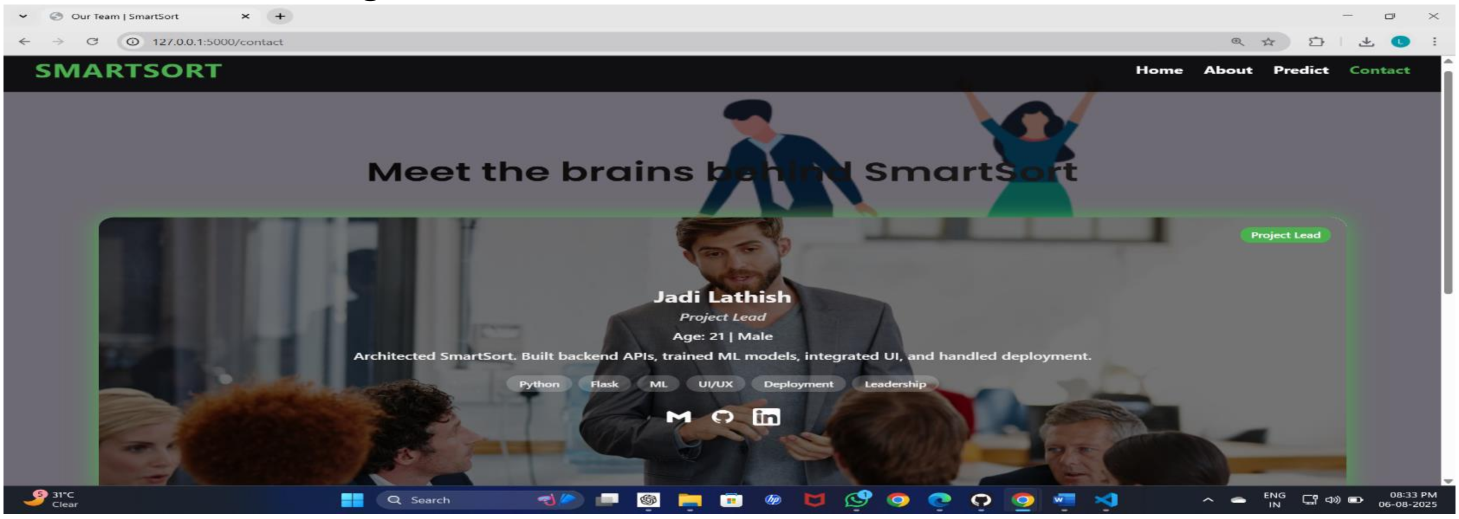

Result | SmartSort

127.0.0.1:5000/predict

✅ Prediction Result

Banana__Rotten (99.98%)

Contact Page:



Our Team | SmartSort

127.0.0.1:5000/contact

SMARTSORT

Home About Predict Contact

Meet the brains behind SmartSort

Project Lead

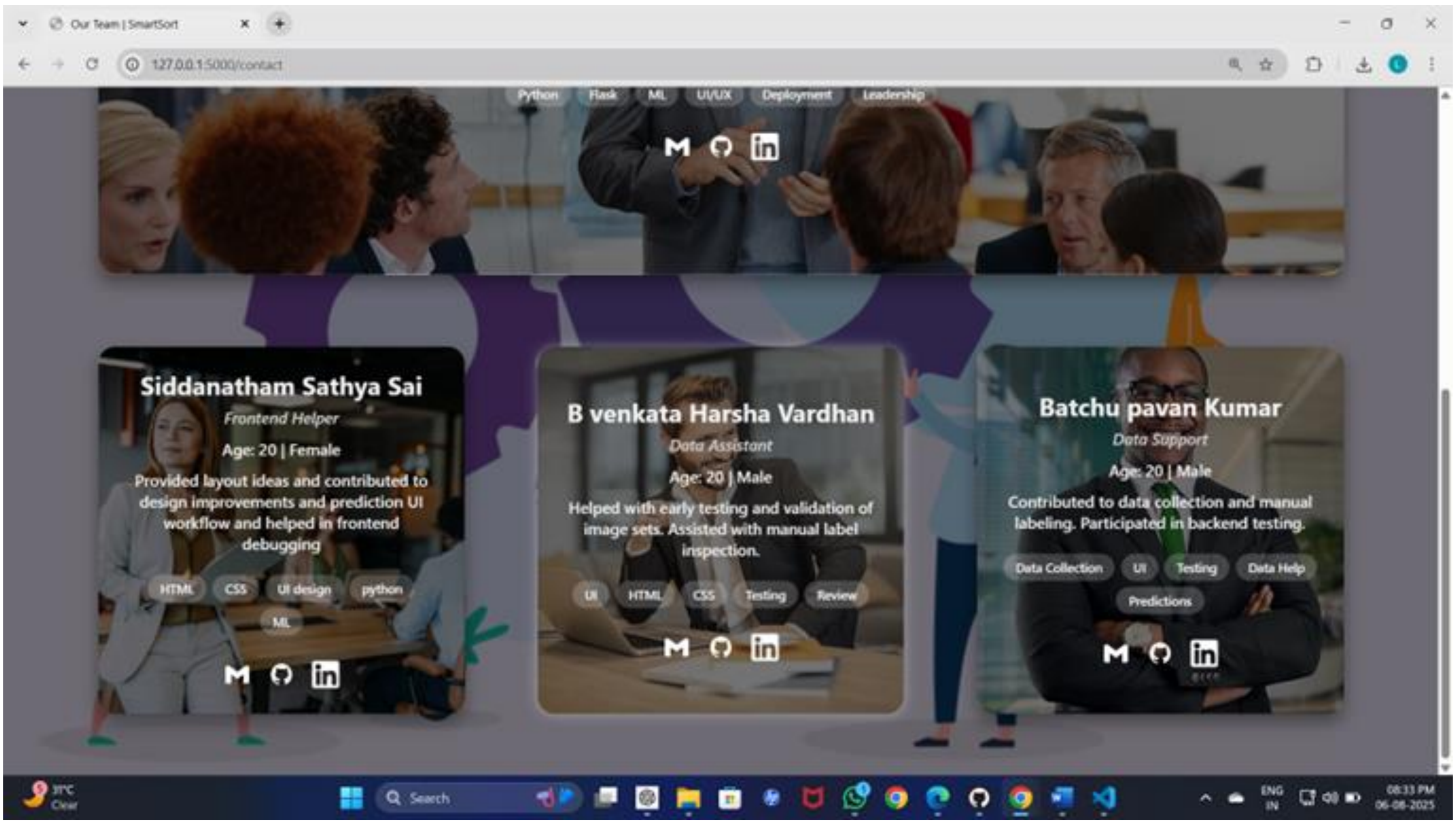
Jadi Lathish

Project Lead

Age: 21 | Male

Architected SmartSort. Built backend APIs, trained ML models, integrated UI, and handled deployment.

Python Flask ML UI/UX Deployment Leadership



Our Team | SmartSort

127.0.0.1:5000/contact

Python Flask ML UI/UX Deployment Leadership

Siddanatham Sathya Sai

Frontend Helper

Age: 20 | Female

Provided layout ideas and contributed to design improvements and prediction UI workflow and helped in frontend debugging

HTML CSS UI design python ML

B venkata Harsha Vardhan

Data Assistant

Age: 20 | Male

Helped with early testing and validation of image sets. Assisted with manual label inspection.

UI HTML CSS Testing Review

Batchu pavan Kumar

Data Support

Age: 20 | Male

Contributed to data collection and manual labeling. Participated in backend testing.

Data Collection UI Testing Data Help Predictions

8. ADVANTAGES & DISADVANTAGES

Advantages of Smart Sorting Using Transfer Learning

- High Accuracy: Pre-trained deep learning models can detect spoilage features better than manual inspection.
- Faster Sorting: Real-time image classification significantly speeds up sorting processes.

- Reduced Waste: Early and accurate detection helps prevent good produce from being discarded and reduces spoilage during storage.
- Cost Efficiency Over Time: Less manual labor leads to lower operational costs in the long run.
- Scalability: The system can be adapted to different types of fruits and vegetables or expanded to more facilities.
- Consistency: Eliminates human subjectivity, ensuring uniform quality standards.
- User-Friendly: Visual dashboards or apps make results easy to interpret, even for non-technical users.

Disadvantages of Smart Sorting Using Transfer Learning

- Initial Investment: Setting up the system requires costs for hardware (cameras, computing devices) and software development.
- Data Dependency: High-quality, labeled datasets are essential to train and fine-tune the model effectively.
- Environmental Variability: Changes in lighting, background, or camera angles can affect accuracy.
- Maintenance: The system may need updates and recalibration over time as produce types or sorting environments change.
- Limited Generalization: A model trained on specific produce may perform poorly on other varieties without retraining.
- Technical Skills Required: Some users may need training to operate and maintain the system.

9. CONCLUSION

The **Smart Sorting** project demonstrates how **transfer learning and computer vision** can transform the way fruits and vegetables are inspected and sorted. By leveraging pre-trained deep learning models, this system provides **accurate, fast, and consistent identification of rotten produce**, helping reduce food waste and improve quality control in the supply chain.

While initial setup and data collection require investment, the long-term benefits—including **increased operational efficiency, reduced manual effort, and higher customer satisfaction**—make Smart Sorting a valuable solution for farmers, vendors, and distributors. The project lays a strong foundation for further innovation in automated food quality assessment and reinforces the role of AI in advancing agricultural practices.

10. FUTURE SCOPE

The Smart Sorting system has strong potential for further development and scaling. Future improvements may include:

- Multi-Class Classification: Expanding the model to categorize produce into multiple freshness levels (e.g., fresh, moderately spoiled, fully rotten) rather than just binary classification.
- Integration with IoT Sensors: Combining visual data with temperature, humidity, and gas sensors to improve spoilage detection accuracy.
- Automated Sorting Hardware: Developing conveyor belts or robotic arms that act on the AI's predictions to fully automate the sorting process.
- Mobile Application Deployment: Building lightweight mobile apps so farmers and vendors can use the system on smartphones in remote areas.
- Cloud-Based Analytics: Providing real-time dashboards, spoilage trends, and predictive analytics through cloud platforms for large-scale producers and distributors.
- Dataset Expansion: Continuously collecting diverse images across seasons, lighting conditions, and produce varieties to improve model generalization.
- Integration with Supply Chain Systems: Connecting classification results directly to inventory and logistics software for smarter decision-making.

11. APPENDIX

Source code :

Python Flask App(app.py) :

```
from flask import Flask, render_template, request, url_for

import os
from werkzeug.utils import secure_filename
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
import gdown
```

```
app = Flask(__name__)

UPLOAD_FOLDER = 'static/uploads'
os.makedirs(UPLOAD_FOLDER, exist_ok=True)

# 1️⃣Google Drive model download setup (direct download using gdown)
MODEL_PATH = "healthy_vs_rotten.h5"
GDRIVE_ID = "1-6P7R6fHLFA7N1qlx3xzmyyxFVd1fbch"
MODEL_URL = f"https://drive.google.com/uc?id={GDRIVE_ID}"

if not os.path.exists(MODEL_PATH):
    print("📥 Downloading model from Google Drive...")
    gdown.download(MODEL_URL, MODEL_PATH, quiet=False)
    print(" ✅Model downloaded successfully.")

#     Load model
model = load_model(MODEL_PATH)

# 🍅 Class labels
class_labels = [
    'Apple__Healthy', 'Apple__Rotten', 'Banana__Healthy', 'Banana__Rotten',
    'Bellpepper__Healthy', 'Bellpepper__Rotten', 'Carrot__Healthy',
'Carrot__Rotten',
    'Cucumber__Healthy', 'Cucumber__Rotten', 'Grape__Healthy',
'Grape__Rotten',
    'Guava__Healthy', 'Guava__Rotten', 'Jujube__Healthy', 'Jujube__Rotten',
    'Mango__Healthy', 'Mango__Rotten', 'Orange__Healthy',
'Orange__Rotten',
    'Pomegranate__Healthy', 'Pomegranate__Rotten', 'Potato__Healthy',
'Potato__Rotten',
    'Strawberry__Healthy', 'Strawberry__Rotten', 'Tomato__Healthy',
'Tomato__Rotten'
]
```

```
@app.route('/')
def index():
    return render_template("index.html")

@app.route('/predict', methods=['GET', 'POST'])
def predict():
    if request.method == 'GET':
        return render_template("predict.html")

    if 'file' not in request.files:
        return "■No file part in the request"

    file = request.files['file']

    if file.filename == '':
        return " ■ No file selected"

    if file:
        filename = secure_filename(file.filename)
        filepath = os.path.join(UPLOAD_FOLDER, filename)
        file.save(filepath)

        # Load and preprocess image
        img = image.load_img(filepath, target_size=(224, 224))
        img_array = image.img_to_array(img) / 255.0
        img_array = np.expand_dims(img_array, axis=0)

        prediction = model.predict(img_array)[0]
        predicted_class = class_labels[np.argmax(prediction)]
        confidence = prediction[np.argmax(prediction)] * 100
        result = f"{predicted_class} ({confidence:.2f}%)"

        return render_template("output.html", prediction=result,
filename=filename)
```

```
    return "❗ Something went wrong"

@app.route('/about')
def about():
    return render_template("about.html")

@app.route('/contact')
def contact():
    return render_template("contact.html")

@app.route("/testbg")
def test_bg():
    return render_template("testbg.html")

# 🟩  THIS STARTS THE FLASK SERVER
if __name__ == '__main__':
    app.run(debug=True)
```

Note : For remaining code files refer GitHub Link

Dataset Link:
https://www.kaggle.com/datasets/muhammad0subhan/fruit-and-vegetable-disease-healthy-vs-rotten

GitHub :
https://github.com/Jadilathish/Smart-Sorting-Transfer-Learning-for-Identifying-Rotten-Fruits-and-Vegetables--intership_project.git

Project Demo Link:
https://drive.google.com/file/d/1W26DcKPSnwxO8gVYwLsN2Tkug1zpECqo/view?usp=sharing