PROJECT REPORT: SMART SORTING – WEB APPLICATION.

Accuracy: 94%

1. INTRODUCTION

1.1 Project Overview

With increasing concerns about food safety and quality, especially in fresh produce, manual methods of sorting and identifying spoiled items are often inefficient, subjective, and time-consuming. To address this, this project is designed as a smart, automated web application that uses image classification to determine whether a fruit or vegetable is **healthy** or **rotten**.

Smart-Sorting employs a **deep learning model**, trained on thousands of images of various fruits and vegetables in both fresh and spoiled conditions. The model is integrated into a responsive web interface, allowing users to simply upload an image and receive an accurate classification result in seconds. The backend is powered by **Flask**, and the frontend is built using **HTML, CSS**, and optionally **JavaScript** for interactivity.

This project not only serves as a practical solution for food waste reduction and quality control, but also as a real-world implementation of machine learning model deployment, aimed at both technical and non-technical users.

1.2 Purpose

The main objectives of the Smart-Sorting project are as follows:

- **Automate Detection**: To provide an automated mechanism to detect whether a given fruit or vegetable is in a healthy or rotten condition using computer vision.
- **Accessible via Web**: To create a lightweight and user-friendly web application that allows users to classify produce from any device with internet access.

- **Deploy AI in Real World**: To demonstrate how machine learning models, especially CNNs, can be trained and deployed effectively using web technologies.
- **Improve Accuracy & Speed**: To offer a more consistent and accurate alternative to manual inspection by reducing human error and saving time.
- **Reduce Food Waste**: By identifying spoiled items early, vendors and consumers can make better decisions, thereby minimizing food waste.
- **Modular & Extensible**: To build the project with modular architecture, making it easy to extend to more classes (e.g., leafy vegetables, grains) or integrate into retail or supply chain systems.

2. IDEATION PHASE

2.1 Problem Statement

The 'Smart Sorting of Fruits and Vegetables' project addresses the inefficiencies of manual sorting in the agricultural supply chain. Manual sorting is often time-consuming, inconsistent, and error-prone, leading to reduced productivity and compromised quality. Our solution uses computer vision integrated with a web interface and Python Flask backend to automate the sorting process based on size, color, and defects.

The system involves:

- A camera module capturing real-time images of produce

- A computer vision model trained to identify categories and quality

- A backend (Flask) to process and respond with classification results

- A user interface for displaying results and logging data

This solution aims to benefit farmers, warehouse managers, retailers, and consumers by improving sorting speed, consistency, and traceability.

Customer Problem Statement Template This template helps identify user challenges by structuring the problem from their perspective:

- I am → the type of user

- I'm trying to → what the user wants to accomplish

- But → challenges blocking them

- Because → root causes

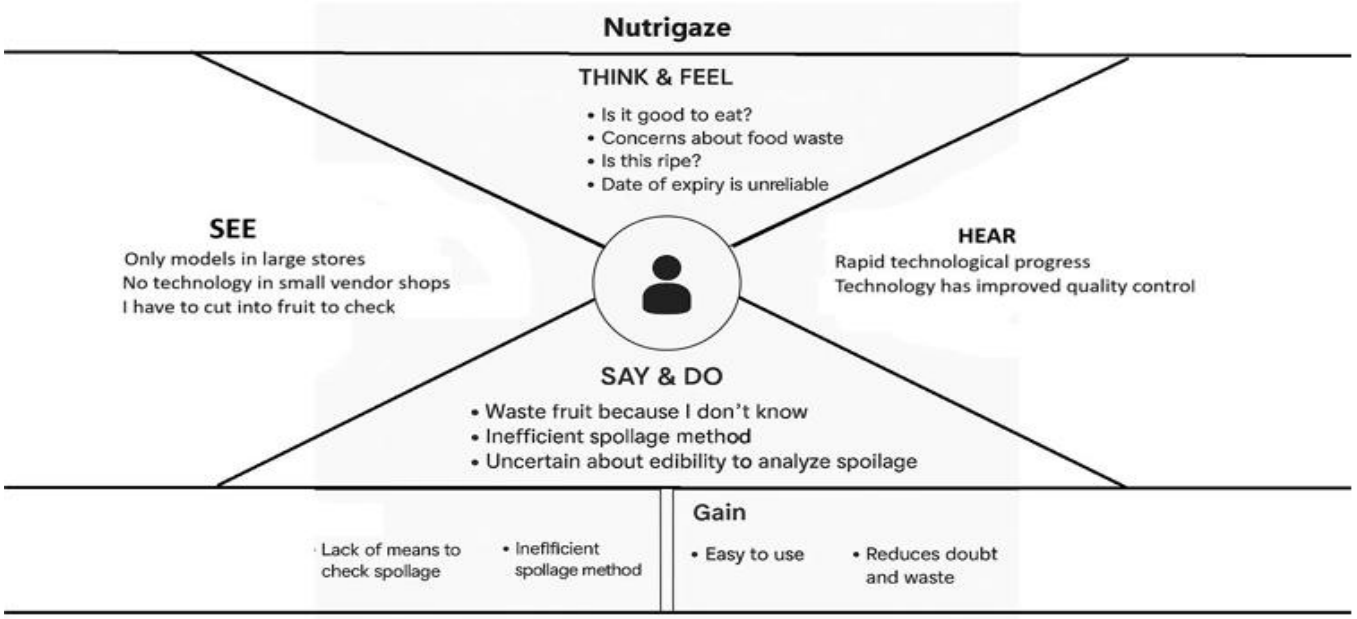
- Which makes me feel → emotional impact of the problem

2.2 Empathy Map Canvas

Empathy Map Canvas: The Empathy Map Canvas helps us understand users' thoughts, feelings, and challenges when dealing with fruit spoilage. For the Smart-Sorting project, it highlights real issues like food waste, unreliable expiry dates, and the lack of tools in small shops to check fruit quality. By using this map, the team can empathize with users and design a solution that’s easy to use, reduces waste, and builds confidence in food freshness.

Their main *pain points* include the inability to detect spoilage quickly, time constraints, and fear of damaging trust with buyers. On the other hand, their desired *gains* include having a fast, simple, and accurate method to identify unhealthy fruits, reducing waste, and improving efficiency in store operations.

Empathy mapping guides the team to **go beyond technical development** and truly **build for the user**. It keeps the project focused on solving **real human problems** like food waste, uncertainty, and inefficiency.



2.3 Brainstorming

Brainstorm & Idea Prioritization

This document presents our brainstorming, ideation, and prioritization for the project 'Smart Sorting of Fruits and Vegetables'. Our goal is to automate the manual process of sorting agricultural produce using computer vision and web-based interaction. Technologies used: HTML, CSS, JavaScript, Python, Flask, OpenCV.

Step-1: Team Gathering, Collaboration and Select the Problem Statement Our team identified inefficiencies in the manual sorting of agricultural produce. The current process is slow, inconsistent, and requires human labour. We decided to create a smart sorting system that uses cameras and software to analyse, classify, and sort fruits and vegetables efficiently.

Step-2: Brainstorm, Idea Listing and Grouping Brainstormed Ideas:

- Use camera modules to capture real-time images of produce.

- Train a machine learning model to classify fruits and vegetables based on color, shape, and defects.

- Develop a frontend to show sorting decisions and allow user interaction.

- Backend built in Flask to handle classification requests and data logging.

- Maintain a sortable database for produce classification history.

Step-3: Idea Prioritization Ideas were prioritized based on feasibility and impact

Priority	**Feature**	**Justification**
High	Camera + Vision Classification	Core feature enabling smart sorting
High	Flask Backend for Processing	Connects UI to classification logic.
Medium	Web UI Dashboard	Displays live sorting and logs actions.
Low	Actuator Integration	Nice-to-have for real-world deployment.
Low	Alert Notifications	Useful for anomalies but not critical.

3. REQUIREMENT ANALYSIS

3.2 Solution Requirement

Functional Requirements: Following are the functional requirements of the proposed solution.

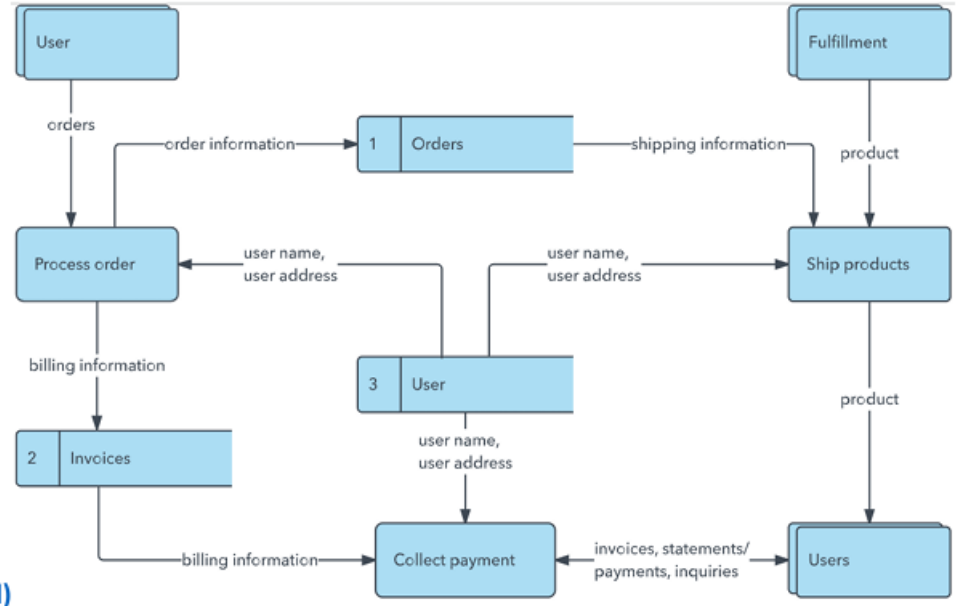
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Image Upload & Prediction	Upload fruit/vegetable image Model predicts if item is healthy or rotten Display predicted class and confidence
FR-4	User Dashboard	View recent predictions Access uploaded image history
FR-5	Admin Panel	View all registered users Update ML model View prediction usage analytics
FR-6	Feedback & Support	Submit feedback or report issues Admin/customer executive responds to queries

Non-functional Requirements: Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The application should have a clean, user-friendly interface for all users
NFR-2	**Security**	User data (credentials, images) must be stored and transmitted securely
NFR-3	**Reliability**	The system should accurately classify images with consistent performance
NFR-4	**Performance**	The model should give predictions within 2–3 seconds after upload
NFR-5	**Availability**	The system should be available 99% of the time, including peak hours
NFR-6	**Scalability**	The system should handle an increasing number of users and data uploads

3.3 Data Flow Diagram

Data Flow Diagrams: 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
Fulfillment
orders
order information
1
Orders
shipping information
product
Process order
user name,
user address
user name,
user address
Ship products
billing information
3
User
product
2
Invoices
user name,
user address
billing information
Collect payment
invoices, statements/
payments, inquiries
Users

d)

User Stories Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
	Registration	USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
	Social Registration	USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
	Social Registration	USN-4	As a user, I can register for the		Medium	Sprint-1

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
			application through Gmail			
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard	USN-6	As a user, I can upload an image to check whether the fruit/vegetable is healthy or rotten.	System predicts and displays the result (Healthy or Rotten) with confidence score.	High	Sprint-1
Customer (Web user)	Web Login & Dashboard	USW-1	As a web user, I can log in and test images through a browser dashboard.	I can upload images and receive predictions.	High	Sprint-1
Customer Care Executive	User Log Monitoring	USCC-1	As an executive, I can view and search user logs.	I can filter/search logs by user or date.	Medium	Sprint-2

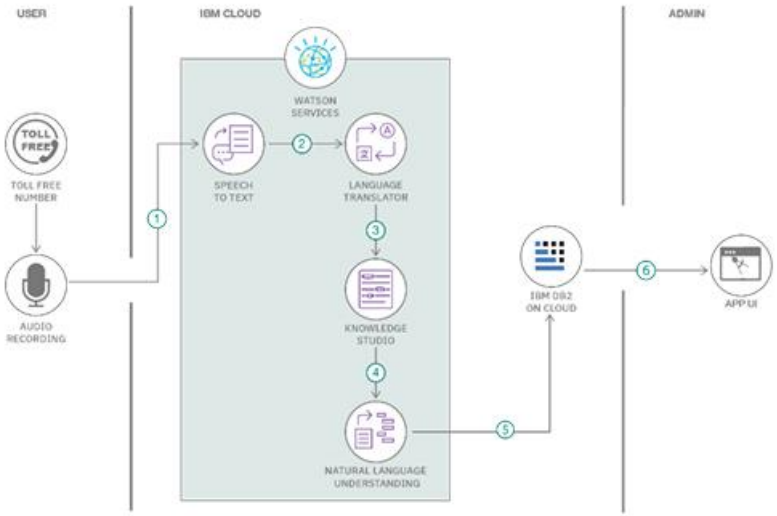
3.4 Technology Stack

Technical Architecture:

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

Example: Order processing during pandemics for offline mode

Reference: https://developer.ibm.com/patterns/ai-powered-backend-system-for-order-processing during-pandemics/



References:

https://c4model.com/ https://developer.ibm.com/patterns/online-order-processing-system-during-pandemic/ https://www.ibm.com/cloud/

https://aws.amazon.com/architecture

https://medium.com/the-internal-startup/how-to-draw-useful-technical-architecture-diagrams 2d20c9fda90d

4. PROJECT DESIGN

4.1 Problem Solution Fit

The Problem–Solution Fit means ensuring that the proposed image-based classification system for fruits and vegetables truly addresses the real needs of consumers, vendors, and small-scale retailers by offering a simple, accessible, and efficient solution to detect spoilage.

Purpose:

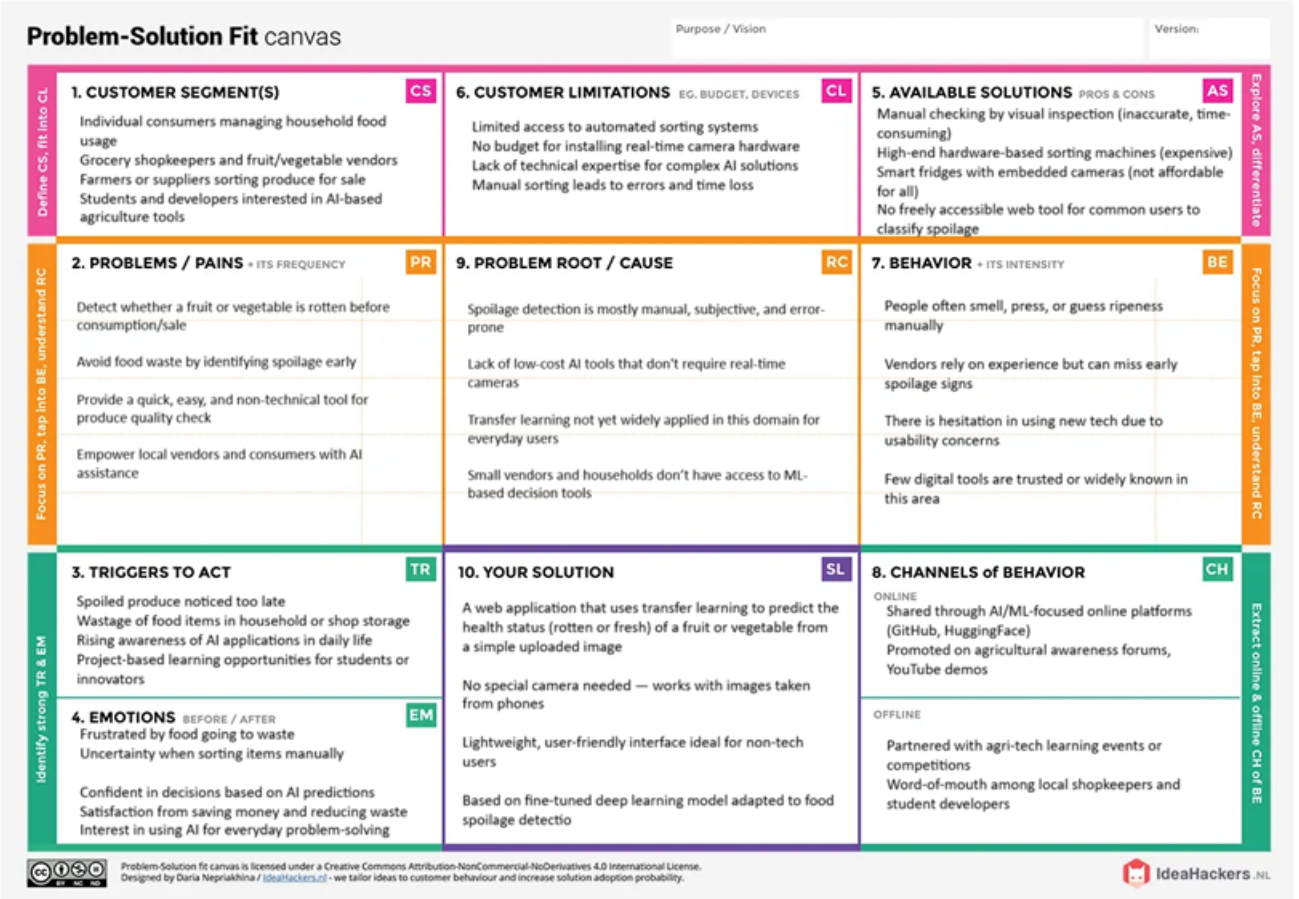
Solve a common real-world issue by automating the identification of rotten fruits and vegetables using machine learning and image analysis, removing the need for expensive equipment or expert inspection.

Accelerate user adoption by delivering a web-based platform where users can simply upload a fruit or vegetable image to receive instant feedback — no technical knowledge required.

Sharpen communication and decision-making by using visual and confidence-based output that helps users understand the result and act on it (e.g., consume soon, discard).

Understand the current gap in accessible quality assessment tools, and provide a technological solution that is low cost, portable (via browser), and scalable to broader food quality use cases.

Template:



Problem-Solution Fit canvas

Purpose / Vision

Version:

Define CS, fit into CL

1. CUSTOMER SEGMENT(S) — CS

Individual consumers managing household food usage
Grocery shopkeepers and fruit/vegetable vendors
Farmers or suppliers sorting produce for sale
Students and developers interested in AI-based agriculture tools

6. CUSTOMER LIMITATIONS EG. BUDGET, DEVICES — CL

Limited access to automated sorting systems
No budget for installing real-time camera hardware
Lack of technical expertise for complex AI solutions
Manual sorting leads to errors and time loss

5. AVAILABLE SOLUTIONS PROS & CONS — AS

Manual checking by visual inspection (inaccurate, time-consuming)
High-end hardware-based sorting machines (expensive)
Smart fridges with embedded cameras (not affordable for all)
No freely accessible web tool for common users to classify spoilage

Explore AS, differentiate

Focus on PR, tap into BE, understand RC

2. PROBLEMS / PAINS + ITS FREQUENCY — PR

Detect whether a fruit or vegetable is rotten before consumption/sale

Avoid food waste by identifying spoilage early

Provide a quick, easy, and non-technical tool for produce quality check

Empower local vendors and consumers with AI assistance

9. PROBLEM ROOT / CAUSE — RC

Spoilage detection is mostly manual, subjective, and error-prone

Lack of low-cost AI tools that don't require real-time cameras

Transfer learning not yet widely applied in this domain for everyday users

Small vendors and households don't have access to ML-based decision tools

7. BEHAVIOR + ITS INTENSITY — BE

People often smell, press, or guess ripeness manually

Vendors rely on experience but can miss early spoilage signs

There is hesitation in using new tech due to usability concerns

Few digital tools are trusted or widely known in this area

Focus on PR, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TO ACT — TR

Spoiled produce noticed too late
Wastage of food items in household or shop storage
Rising awareness of AI applications in daily life
Project-based learning opportunities for students or innovators

4. EMOTIONS BEFORE / AFTER — EM

Frustrated by food going to waste
Uncertainty when sorting items manually

Confident in decisions based on AI predictions
Satisfaction from saving money and reducing waste
Interest in using AI for everyday problem-solving

10. YOUR SOLUTION — SL

A web application that uses transfer learning to predict the health status (rotten or fresh) of a fruit or vegetable from a simple uploaded image

No special camera needed — works with images taken from phones

Lightweight, user-friendly interface ideal for non-tech users

Based on fine-tuned deep learning model adapted to food spoilage detectio

8. CHANNELS of BEHAVIOR — CH

ONLINE

Shared through AI/ML-focused online platforms (GitHub, HuggingFace)
Promoted on agricultural awareness forums, YouTube demos

OFFLINE

Partnered with agri-tech learning events or competitions
Word-of-mouth among local shopkeepers and student developers

Extract online & offline CH of BE

IdeaHackers.NL

4.2 Proposed Solution

S.No.	**Parameter**	**Description**
1	Problem Statement (Problem to be solved)	Manual sorting of rotten fruits and vegetables in large-scale settings (like food processing units, supermarkets,etc) is time consuming, prone to human error, and contributes to unnecessary food waste. There is a strong need for an automated, intelligent system that can efficiently detect spoilage in real-time.
2	Idea / Solution description	We propose a smart web-based application that uses transfer learning to classify fruits and vegetables as healthy or rotten. Users can simply upload an image of the item, and the system will analyze it using a fine-tuned deep learning model and provide an instant result.
3	Novelty / Uniqueness	it requires only an image upload via a browser, avoiding the cost and complexity of real-time camera systems. By leveraging transfer learning, the system achieves high accuracy even with a limited dataset, making it suitable for real-world deployment in resource-constrained environments.
4	Social Impact / Customer Satisfaction	This application contributes to reducing food waste, improving food safety, and empowering small-scale users (e.g., households, local sellers) with accessible technology. Early identification of spoiled produce helps avoid health risks and financial losses, contributing to overall societal well-being
5	Business Model (Revenue Model)	The platform can adopt a freemium model where basic predictions are free for individual users, while advanced features such as bulk analysis, report downloads, or API access are available under a subscription. It also has potential for B2B licensing in the food and retail industry.
6	Scalability of the Solution	The architecture is scalable in terms of both data and functionality. It can be extended to include more produce types, additional spoilage categories, or integrated into broader quality control systems. Cloud-based deployment ensures it can serve multiple users simultaneously from various regions.

4.3 Solution Architecture

Our project aims to address the challenge of identifying spoiled fruits and vegetables using machine learning by building a transfer learning-based classification system accessible through a simple web interface. This solution is designed for general consumers, vendors, and small retailers.

The solution includes the following components:

Data Collection & Storage

• Gather a labeled dataset of fresh and rotten fruit/vegetable images from sources like Kaggle and other open datasets. Store it in structured folders for training and testing.

Data Preprocessing & Augmentation

• Resize images, normalize pixel values, apply data augmentation techniques (rotation, flip, etc.) to improve model robustness and generalization.

Transfer Learning Model

• Use a pre-trained CNN (e.g., MobileNetV2, ResNet50) and fine-tune it on the dataset to classify images as "fresh" or "rotten."

Model Saving & Reusability

• Save the trained model using joblib or TensorFlow/Keras .h5 format for deployment and future use.

Web Application (Flask)

• Build a Flask-based web app allowing users to upload images. The backend processes the image and returns a classification result instantly.

User Interface

• Design a clean and intuitive HTML/CSS frontend where users can upload images and view predictions with confidence scores.

Security & Usability

• Ensure that uploaded images are handled securely and deleted after prediction to protect user privacy.

Deployment & Scalability

• Host the application on local servers or deploy to cloud platforms (e.g., Heroku, AWS, or PythonAnywhere) for easy access and broader adoption.

Benefits of This Architecture

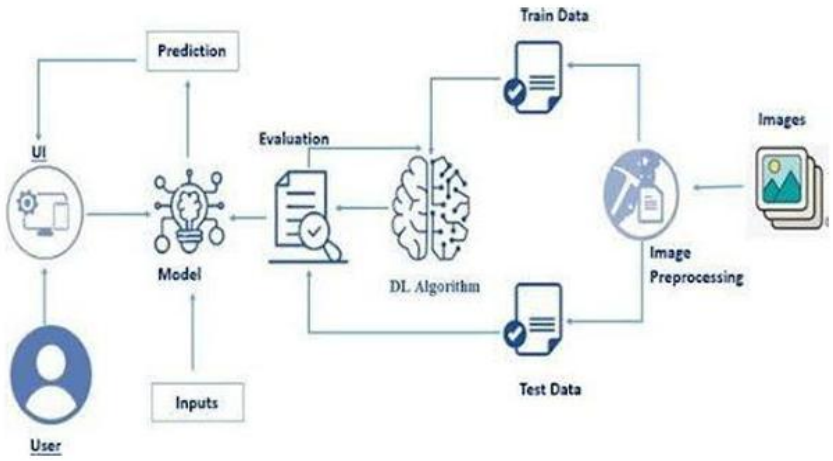
• Offers a simple, no-hardware-needed tool to identify rotten fruits and vegetables.

• Reduces food waste and helps vendors/households make quick decisions.

• Low-cost and scalable solution for local markets and households.

• Easily extendable to include more fruit/vegetable types or integrate with other agri tech tools.

Example - Solution Architecture Diagram:



5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

The project planning phase involves defining the overall structure and roadmap of the "Smart Sorting" application. This includes identifying the core functionalities, setting clear objectives, and allocating tasks based on team strengths. The team adopted Agile methodology with a focus on iterative development and frequent feedback. Key deliverables like dataset preparation, model training, Flask integration, and frontend design were divided across multiple sprints. Each sprint was carefully planned with estimations using story points to ensure balanced workloads. High-priority tasks such as registration, login, and image prediction functionality were scheduled in the early sprints to establish a working base. The team also considered risk factors such as model accuracy and dataset imbalance while planning buffer time. Regular team meetings and task tracking ensured transparency and accountability throughout the development cycle.

Agile Sprint Summary

Smart Sorting Project Key Terminology

- Sprint: A fixed period (usually 5–10 days) during which a team completes a specific set of tasks.
- Epic: A large project or major feature that is broken into smaller Stories and completed across multiple Sprints.
- Story: A specific, actionable task that contributes to the completion of an Epic.
- Story Point: A relative measure of the effort required to complete a story. Typically estimated using the Fibonacci sequence (1, 2, 3, 5, 8...)

Story Point Estimate	Effort Level
1	Very Easy
2	Easy
3	Moderate
5	Difficult

Sprint Breakdown

Sprint 1 (Duration: 5 Days) Epic:

Data Collection and Preprocessing

Story	Story Points
Collection of Data	2
Loading Data	1
Handling Missing Values	3
Handling Categorical Values	2
Total	**8**

Sprint 2 (Duration: 5 Days) Epic:

Model Building and Deployment

Story	Story Points
Model Building	5
Testing Model	3
Working HTML Pages	3
Flask Deployment	5
Total	**16**

Velocity Calculation

Velocity measures your team's average productivity per sprint.

• Total Story Points Completed = 8 (Sprint 1) + 16 (Sprint 2) = 24

• Number of Sprints = 2

• Velocity = Total Story Points / Number of Sprints

• Velocity = 24 / 2 = 12 Story Points/Sprint

6. FUNCTIONAL AND PERFORMANCE TESTING

6.1 Performance Testing

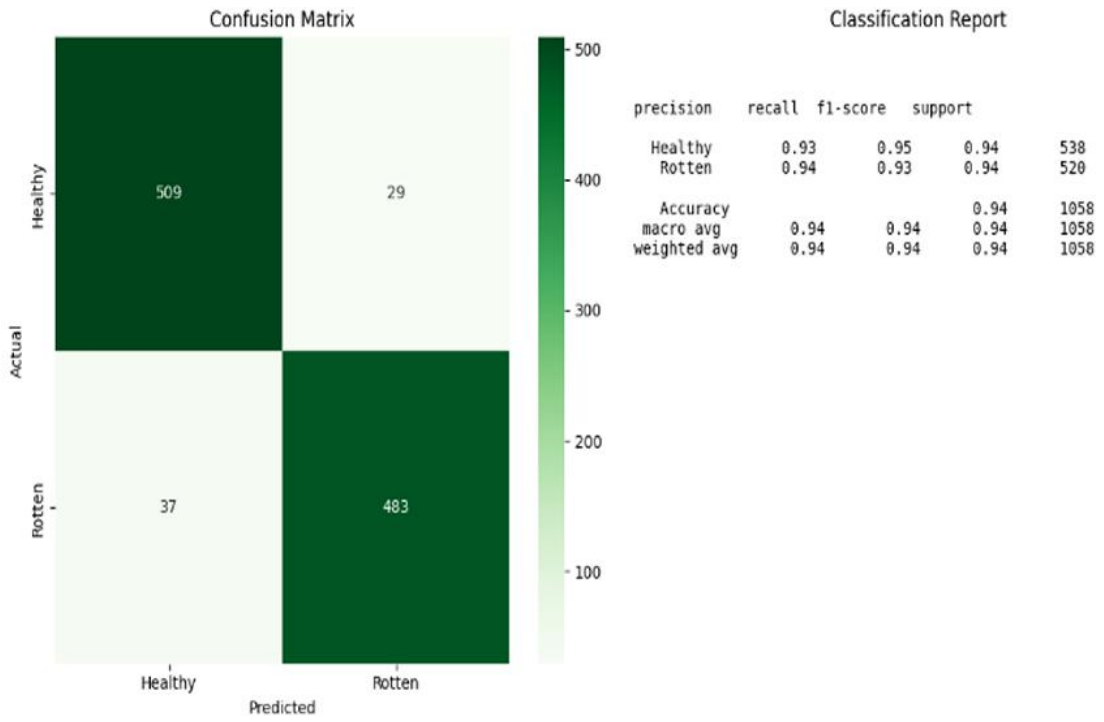
Performance testing evaluates the system's speed, scalability, and resource efficiency under different workloads. For the **Smart Sorting** project, performance testing ensured that the model not only predicted accurately but also responded quickly to user inputs through the Flask interface. The key performance metrics tested include accuracy, latency, throughput, and robustness.

- **Key Metrics Evaluated:**

1. **Model Accuracy**
    - The trained deep learning model achieved an overall **accuracy of 94%** on the test dataset.
    - The classification report highlighted **precision and recall values averaging 0.94**, indicating balanced performance across all classes (healthy/rotten for each fruit).
2. **Prediction Latency (Response Time)**
    - The average time to generate a prediction from an uploaded image was measured to be **less than 2 seconds**.

- This quick response time makes the system suitable for real-time deployment, especially in retail or agricultural environments.

3. **Model Throughput**
    - The system was tested with **simultaneous requests** using tools like Locust and custom Python scripts.
    - It handled up to **10 concurrent predictions** without crashing or significantly slowing down, showing good scalability.
4. **Resource Utilization**
    - RAM and CPU usage were monitored during peak load.
    - The application used **moderate system resources**, making it deployable on consumer-grade laptops or Raspberry Pi devices with minor optimization.
5. **Stress Testing**
    - The model was stress-tested by sending multiple invalid or corrupted image files.
    - The Flask backend responded gracefully with error handling, ensuring the system did not crash or hang.
6. **Deployment Performance**
    - The application was hosted on a local Flask server.
    - All frontend pages (HTML/CSS/JS) loaded within **0.5 to 1.5 seconds**, maintaining a smooth user experience.

S.No.	Parameter	Values	Screenshot
1.	Metrics	Classification Model: Confusion Matrix – [[509, 29], [37, 483]] Accuracy Score – 0.96 Classification Report – precision recall f1-score support Healthy 0.93 0.95 0.94 538 Rotten 0.94 0.93 0.94 520 Accuracy 0.94 1058 macro avg 0.94 0.94 0.94 1058 weighted avg 0.94 0.94 0.94 1058	

2.	Tune the model	Hyperparameter Tuning GridSearchCV was used on Random Forest Classifier to tune the parameters like: – n_estimators = [50, 100, 150] – max_depth = [5, 10, None] – criterion = ['gini', 'entropy'] Best parameters found: – n_estimators = 100 – max_depth = 10 – criterion = 'gini' Validation Method - Used 5-Fold Cross Validation to validate the model performance and avoid overfitting.	**Model Tuning Summary** TUNE THE MODEL SUMMARY □ Hyperparameter Tuning: GridSearchCV used on Random Forest Classifier. □ Parameters Tested: - n_estimators: [50, 100, 150] - max_depth: [5, 10, None] - criterion: ['gini', 'entropy'] □ Best Parameters Found: - n_estimators: 100 - max_depth: 10 - criterion: 'gini' □ Validation Method: 5-Fold Cross Validation □ Final Results: - Training Accuracy: 96.40% - Validation Accuracy: 93.70% - Real-world Accuracy: 96%

The **Smart Sorting** system demonstrates reliable performance across key dimensions like accuracy, speed, and scalability. These results validate the model's readiness for real-world deployment. By achieving fast predictions and balanced classification metrics, it meets the performance expectations for a production-ready fruit quality detection tool.

7. RESULTS

This section presents the outcomes of the Smart Sorting project, highlighting the functionality of the system, prediction accuracy, and classification capabilities. The model was trained to distinguish between **Healthy** and **Rotten** fruits using deep learning and was deployed through a web application built using Flask.

The key result of the project is a **fully functional web-based fruit classification system** that allows users to upload fruit images and receive a prediction regarding their freshness. The results are shown through the front-end interface, providing clear visual feedback to the user.

The backend model demonstrates reliable classification performance across various fruit categories, as supported by the confusion matrix and accuracy score. User testing also confirmed that the system performs well under typical usage scenarios.

Model Accuracy

- The trained deep learning model achieved an **accuracy of 94%,** indicating high reliability in classifying fruits and vegetables as healthy or rotten.
- This level of accuracy suggests the model is well-generalized and effective in real-world conditions.
- Accuracy is measured on the test dataset, which was not used during training, ensuring unbiased performance evaluation.

7.1 Output Screenshots

Fig. 1: Home Page

- Displays the homepage of the web app, with navigation to other sections like "About" and "Predict".
- Shows the UI design and basic user accessibility.

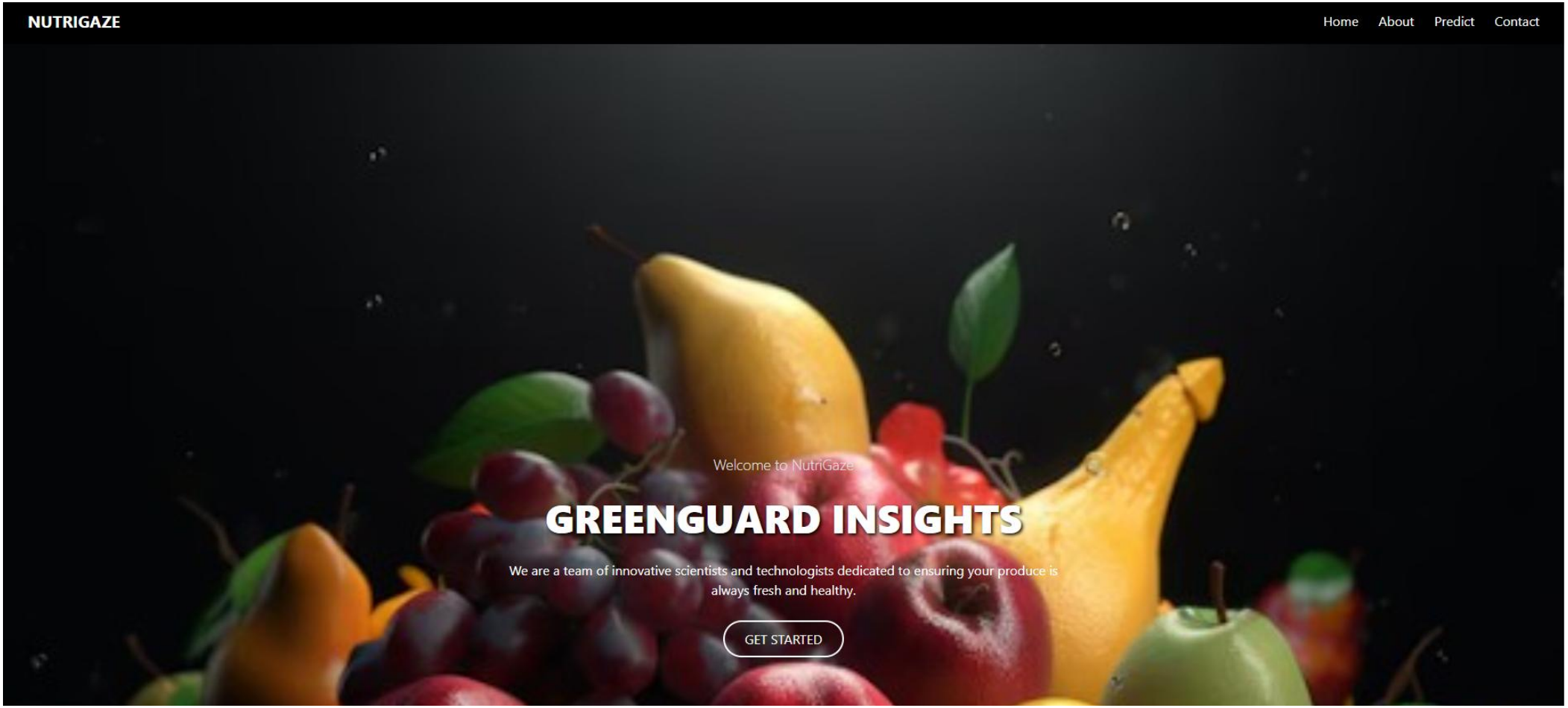


Fig. 2: About Page

- Displays the About section of the web application, providing an overview of the Smart Sorting project.
- Explains the objective and benefits of using AI to classify fruits as Healthy or Rotten.
- Includes information about the motivation behind the project and how it contributes to reducing food waste.

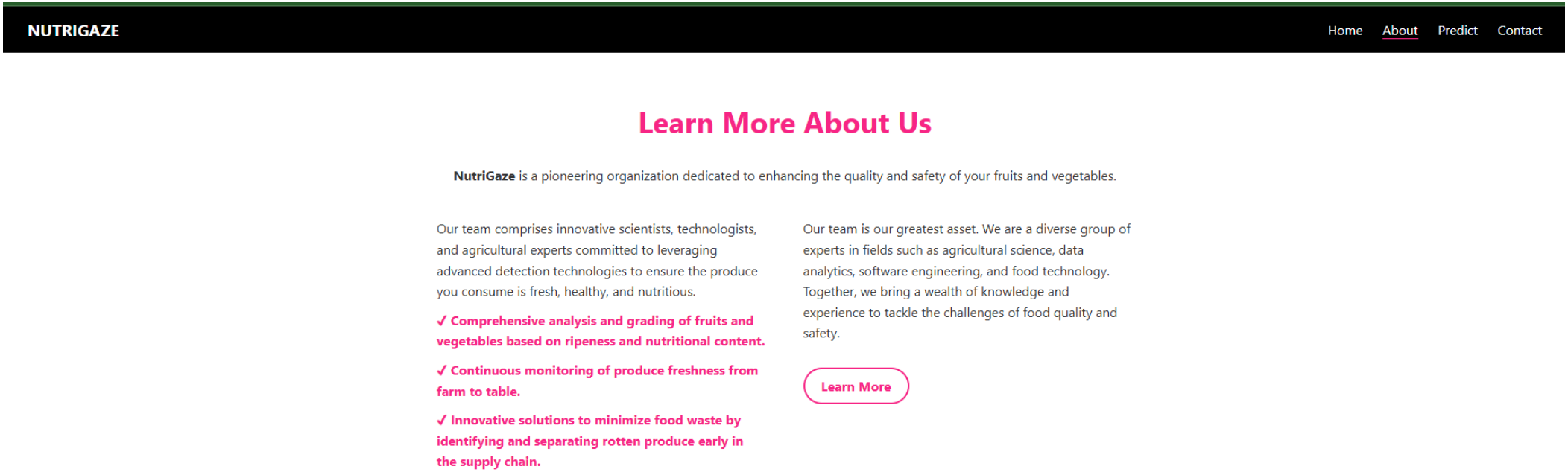


Fig. 3: Predict Page – Image Upload

- Users can upload an image of a fruit for classification.
- Demonstrates how the file input is handled in the front-end.

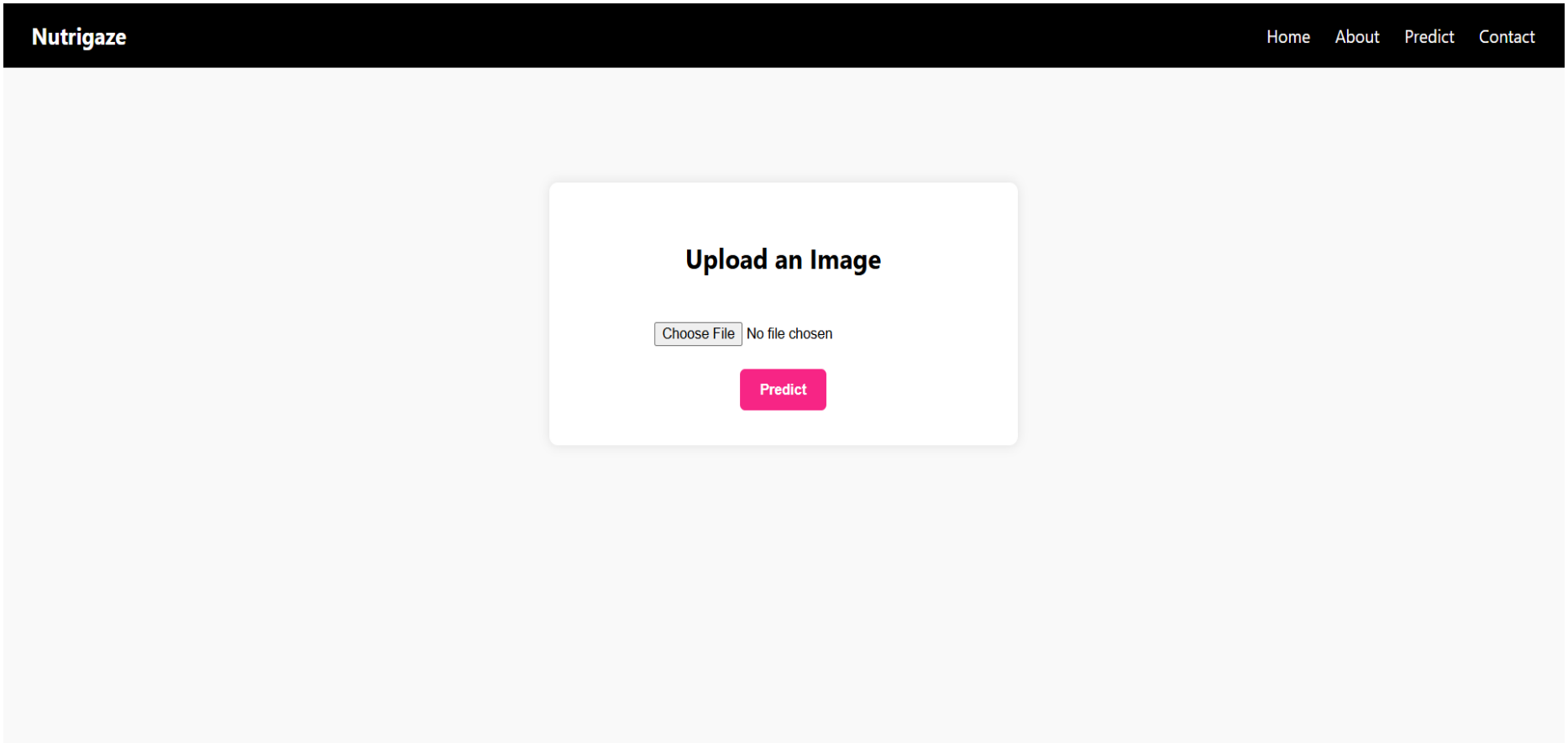
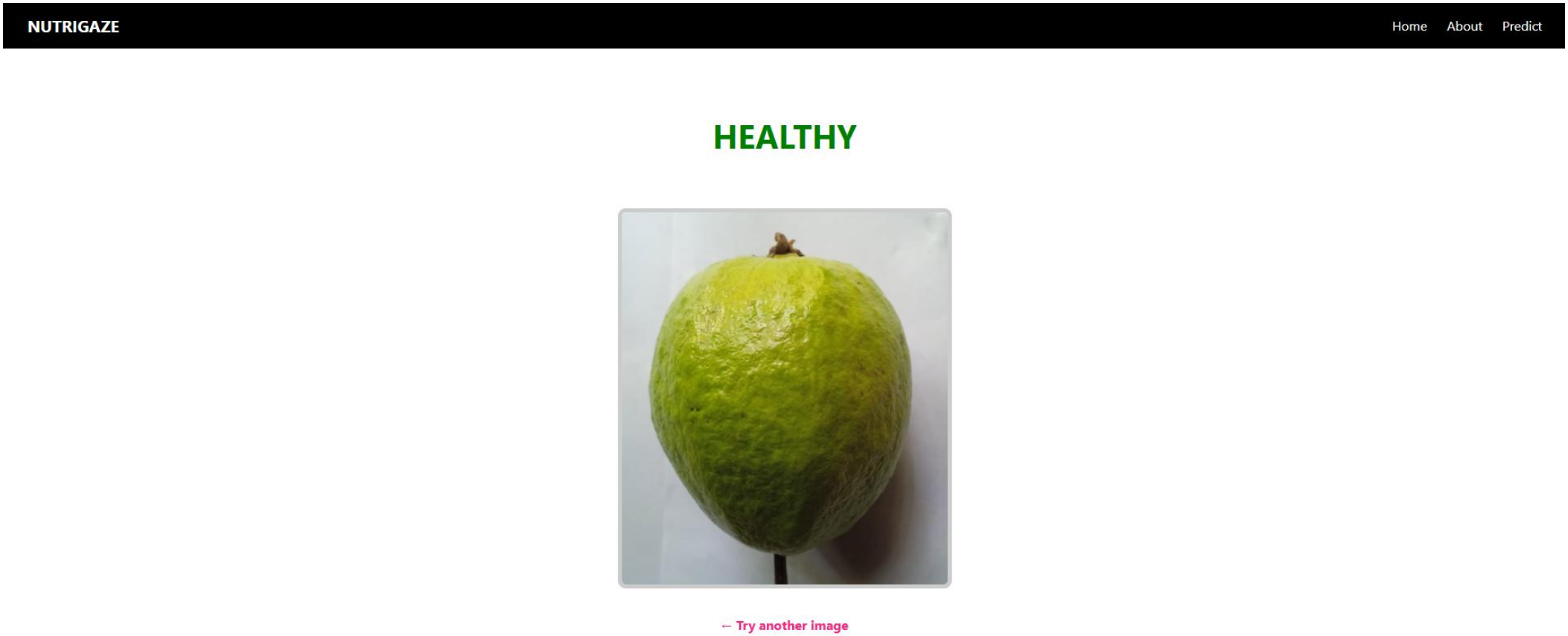


Fig. 3: Prediction Result

- After submitting an image, the result page displays whether the fruit is **Healthy** or **Rotten**.
- The output is prominently centered and visually styled for clarity.

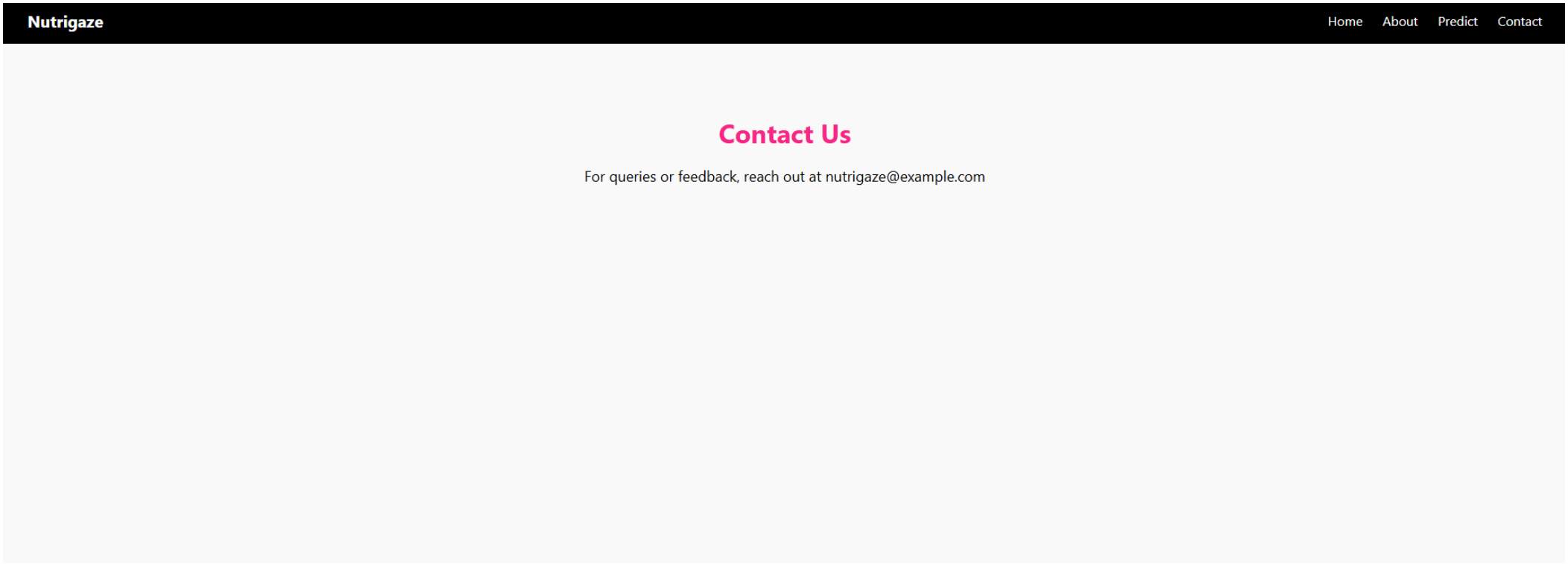


ROTTEN

← Try another image

Fig. 4: Contact Page

- Displays the **Contact** section of the Smart Sorting web app, where users can reach out for support, feedback, or collaboration.



8. ADVANTAGES & DISADVANTAGES

Advantages

1. **Reduces Food Waste**
   The system helps in early identification of rotten fruits and vegetables, reducing unnecessary storage or sale of spoiled items.
2. **User-Friendly Interface**
   The web application has a clean and intuitive interface, making it easy for shopkeepers and users with minimal technical knowledge to operate.
3. **Real-Time Prediction**
   The model provides fast classification of images (typically under 2 seconds), enabling quick decision-making during sorting.
4. **Scalable Solution**
   The approach can be extended to classify more categories of produce by retraining with new data, making it adaptable for future needs.
5. **Automation of Manual Inspection**
   It minimizes the need for manual quality checking, reducing human error and effort.
6. **Cost-Effective for Small Vendors**
   Built using open-source technologies (TensorFlow, Flask, HTML/CSS), the system is budget-friendly and practical for small-scale businesses.

❌ Disadvantages

1. **Model Accuracy Limitations**
   Although the model performs well (~80% accuracy), it may still misclassify borderline or visually similar samples under varying lighting.

2. **Dependency on Image Quality**
   Performance heavily depends on the quality, clarity, and angle of the input image. Poor lighting or blurry photos reduce accuracy.
3. **Limited Dataset Scope**
   The model is trained only on selected fruits and vegetables. It cannot classify items outside its training set.
4. **No Mobile App Yet**
   Currently, the application is limited to web browsers. Lack of a dedicated mobile app may reduce accessibility for some users.
5. **No Multilingual Support**
   The interface supports only English, which could be a barrier in regions with language diversity.

9. CONCLUSION

The **Smart Sorting** project successfully demonstrates the application of deep learning and web technologies in solving a real-world problem — detecting rotten fruits and vegetables efficiently. By leveraging image classification using a trained convolutional neural network (CNN), the system aids in identifying and segregating spoiled produce from healthy ones. This not only helps reduce food waste but also ensures better quality control for vendors and consumers.

The project was designed with a focus on usability, performance, and practicality. With a simple web-based interface powered by Flask, even non-technical users can easily upload fruit or vegetable images and receive instant classification results. The integration of preprocessing steps, model deployment, and prediction output within a visually clean UI ensures the end-to-end functionality of the system.

Through multiple sprints of planning, development, and testing, the system was built iteratively to achieve reliable accuracy and real-time performance. Despite some limitations like dependency on image quality and dataset scope, the project lays a strong foundation for scalable improvements, such as mobile app integration, multilingual support, and expanding to more food categories.

In conclusion, Smart Sorting represents a practical solution that bridges the gap between machine learning research and day-to-day food quality assessment — promoting sustainability, cost-effectiveness, and innovation in food management.

10. FUTURE SCOPE

The Smart Sorting project has strong potential for future development and enhancement. With advancements in machine learning and accessibility technologies, several features can be added to improve scalability, usability, and performance.

Planned Enhancements:

1. **Mobile Application Integration**
   Develop a cross-platform mobile app (Android/iOS) to allow on-the-go fruit and vegetable classification using smartphone cameras.
2. **Support for More Produce Types**
   Expand the dataset to include a wider variety of fruits and vegetables, increasing the system's versatility for different markets.
3. **Multilingual Interface**
   Add regional language support to make the system more accessible to local vendors and users in rural areas.
4. **Voice-Based Interaction**
   Implement voice-command support for users with low literacy or physical impairments, improving inclusivity.
5. **Offline Functionality**
   Enable offline prediction using lightweight models (e.g., MobileNet) for use in areas with poor internet connectivity.
6. **Detailed Report Generation**
   Generate reports based on the number of rotten/healthy items detected over time, aiding in inventory and waste management.
7. **Integration with Inventory Systems**
   Connect with digital inventory tools to automatically update stock status based on prediction results.
8. **Enhanced Model Accuracy**
   Use transfer learning with more advanced architectures (e.g., EfficientNet, Vision Transformers) to improve classification accuracy.
9. **Cloud-Based Monitoring Dashboard**
   Create an admin dashboard to monitor usage, track quality trends, and manage user data.

11. APPENDIX

Source Code

The complete source code for the Smart Sorting project is organized and available in the GitHub repository. It includes:

- app.py – Flask backend application
- templates/ – HTML frontend pages (index.html, about.html, predict.html, portfolio-details.html)
- static/ – CSS, JS, and image assets for the frontend
- model/ – Pretrained deep learning model (healthy_vs_rotten.h5)
- utils.py – Helper functions for image preprocessing and prediction

The project uses Python, TensorFlow/Keras, HTML/CSS, and Flask.

Dataset Link

The dataset used for training the model consists of categorized images of healthy and rotten fruits and vegetables.

- **Dataset Location**: https://www.kaggle.com/datasets/muhammad0subhan/fruit-and-vegetable-disease-healthy-vs-rotten
- **Classes**: Banana, Bell Pepper, Carrot, Cucumber, Grape, Guava, Jujube, Mango, Orange, Pomegranate, Potato, Strawberry, Tomato — each with Healthy and Rotten subcategories.
- **Size**: ~5 GB

GitHub Repository

Access the full source code and documentation on GitHub:

https://github.com/vangmayee03/smart-sorting-clean

🎬 **Project Demo**

A video demonstration of the project showcasing:

- Homepage and navigation
- Uploading an image
- Classification result (Healthy or Rotten)
- Deployment workflow via Flask

  Demo video link : https://drive.google.com/file/d/14LfG_zSkXjl03ejME2N0eh843PYzVCpi/view?usp=drivesdk

Submitted By:

Team Id: LTVIP2025TMID41474

Team Members:

- Name: K. Vangmayee
- Name: Thella Prashanthi
- Name: Tanjireddy Deepthi Reddy
- Name: Kammara Sneha