## Project Documentation

## 1.Introduction

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

**Team Members:**

## **Team Leader :** Dharani Nagarapu

## **Team member :** Kancharla Manikanta

## **Team member :** Bhaskar Sai Gudipati

## **Team member :** Komiripalepu Tirumala Ranganadh

## 2. Project Overview

**Purpose:** NutriGaze helps users detect the freshness of fruits and vegetables using AI-powered

image classification to reduce waste and improve food safety. Users can upload images of fruits

or vegetables, and the system predicts whether they are healthy or rotten using a VGG16 deep

learning model. It provides a clean, responsive interface built with Flask and displays the

predicted class label alongside the uploaded image for clear confirmation. This system offers a

quick, reliable, and accessible method for consumers, vendors, and supermarkets to verify

produce freshness, thereby reducing manual inspection errors and supporting proactive quality

control. NutriGaze also demonstrates practical application of deep learning in daily life while

ensuring a seamless user experience.

**Features:**

* Upload fruit/vegetable images for freshness detection.
* Predicts healthy or rotten using a VGG16 deep learning model.
* Clean, responsive UI with Flask backend.
* Output with class label and uploaded image for confirmation.

## 3. Architecture

**Frontend:**

* HTML5, CSS3, Bootstrap for styling.
* Jinja templating for dynamic data rendering.

**Backend:**

* Python Flask for routing, model inference, and file handling.
* TensorFlow/Keras for running the pre-trained VGG16 model.

**Database:**

* No persistent database required for inference.
* Uses static/uploads/ for temporary file storage during prediction.

## 4. Setup Instructions

**Prerequisites:**

* Python 3.x
* TensorFlow, Keras
* Flask
* Anaconda (optional for environment management)

**Installation:**

1. Clone the repository.
2. Create a virtual environment: conda create -n nutrigaze-env python=3.10
3. Activate environment: conda activate nutrigaze-env
4. Install dependencies:

* pip install flask tensorflow keras numpy pillow

1. Place healthy\_vs\_rotten.h5 in the project root.
2. Run the app:

* python app.py

1. Open http://127.0.0.1:5000 in your browser.

## 

## 5. Folder Structure

• **app.py** – Main Flask application handling routing, image upload, prediction using the

trained model, and rendering web pages.

• **static/**

* **uploads/** – Stores user-uploaded images temporarily during prediction.
* **assets/** – Stores images, CSS files, and other static assets for styling and layout.

• **templates/**

* **index.html** – Home page with project overview and navigation.
* **predict.html** – Page for uploading images and viewing predictions.
* **about.html**, **portfolio-details.html** – Additional informational pages about the project.

• **healthy\_vs\_rotten.h5** – Trained VGG16 deep learning model file used for classifying fruit

and vegetable images as healthy or rotten.

## 6. Running the Application

python app.py

Visit http://127.0.0.1:5000 to access NutriGaze.

The application will launch the Flask development server locally, allowing you to test

predictions in your browser. Ensure your environment is activated and the

healthy\_vs\_rotten.h5 model file is in the project root before running.

## 7. API Documentation

**Route:** /predict

* **Method:** POST
* **Description:** Accepts an image upload and returns prediction.
* **Response:** Rendered HTML page with predicted label and uploaded image.
* The route internally preprocesses the image and uses the VGG16-based model for

classification.

* Handles invalid uploads gracefully, displaying clear error messages to the user.

## 

## 8. Authentication

Currently, NutriGaze does not implement authentication as it is a lightweight offline testing

tool. Future versions can integrate JWT-based authentication if user tracking and

dashboards are added.

## 9. User Interface

* Simple upload interface with drag-and-drop or file selection.
* Clear display of prediction and image preview.
* Mobile responsive UI.
* Navigation bar with clear links to Home, About, and Predict pages for smooth navigation.
* Attractive hero section and clean layout ensuring ease of use for all user levels.

## 10. Testing

**Tools Used:** Manual testing with Jupyter and Flask interface.

**Strategy:**

* Tested with various quality images and lighting conditions.
* Verified output correctness against known data.
* Tested for invalid file handling and error messages.

## 11. Screenshots or Demo

## 

## 

## 

## 12. Known Issues

* Model accuracy may reduce with poor lighting images.
* Does not detect spoilage not visible externally.
* No persistent storage of history.
* Requires stable internet for dependency installation during initial setup.

## 13. Future Enhancements

* Add mobile app with camera-based live prediction.
* Extend dataset with more fruit and vegetable categories.
* Add backend analytics and report generation for vendors.
* Integrate SMS/email alerts for bulk scanning environments.
* Add user authentication and history tracking dashboards.