

FRESHNESS DETECTION OF FRUITS & VEGETABLES USING PYTHON

**A Minor Project Report Submitted in Partial fulfillment for the award of
Bachelor of Technology in Department of Artificial Intelligence & Data
Science**

Submitted to
RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA BHOPAL
(M.P)



MINOR PROJECT REPORT
Submitted by

Pranjal Srivastava [0103AD231148]
Priyanshu Sarathe [0103AD231162]
Ramansh Yadav [0103AD231165]

Under the supervision of

Dr. Sushil Kumar
Associate Professor



**Department of Artificial Intelligence & Data science Lakshmi
Narain College of Technology, Bhopal (M.P.)
Session 2025 - 2026**

LAKSHMI NARAIN COLLEGE OF TECHNOLOGY, BHOPAL

Department of Artificial Intelligence & Data science



CERTIFICATE

This is to certify that the work embodied in this project work entitled "**Freshness Detection of Fruits & Vegetables using Python**" has been satisfactorily completed by the **Pranjal Srivastava** (0103AD231148), **Ramansh Yadav** (0103AD231165) & **Priyanshu Sarathe** (0103AD231162). It is a bonafide piece of work, carried out under the guidance in **Department of AIDS, Lakshmi Narain College of Technology, Bhopal** for the partial fulfillment of the **Bachelor of Technology** during the academic year 2025- 2026.

Guided By

Dr. Sushil Kumar
Associate Professor

Approved By

Dr. Tripti Saxena
Professor & Head

LAKSHMI NARAIN COLLEGE OF TECHNOLOGY, BHOPAL

Department of Artificial Intelligence & Data science

ACKNOWLEDGEMENT

We express our deep sense of gratitude to Prof. Dr Sushil Kumar department of AIDS, L.N.C.T., Bhopal. whose kindness valuable guidance and timely help encouraged me to complete this project.

A special thank goes to Dr. Tripti Saxena (HOD) who helped me in completing this project work. He exchanged his interesting ideas & thoughts which made this project work successful.

We would also thank our institution and all the faculty members without whom this project work would have been a distant reality.

Pranjal Srivastava (0103AD231148)

Ramansh Yadav (0103AD231165)

Priyanshu Sarathe (0103AD231162)

INDEX

S.NO.	TOPICS	PAGES
1.	Problem Domain Description	1 - 7
2.	Literature Survey	8 – 11
3.	Major objective & scope of project	12– 14
4.	Problem Analysis and requirement specification	15 – 17
5.	Detailed Design (Modeling and ERD/DFD)	18 – 24
6.	Hardware/Software platform environment	25 – 26
7.	Snapshots of Input & Output	27 – 31
8.	Coding	32 – 36
9.	Project limitation and Future scope	37 – 40
10.	References	41 - 42
	TOTAL	42 pages

CHAPTER 1

PROBLEM DOMAIN DESCRIPTION

1.1 Introduction

Fruits and vegetables are naturally perishable items whose quality deteriorates over time due to physical, chemical, and microbial changes. Their freshness directly affects their nutritional value, taste, appearance, and safety for consumption. Traditionally, evaluating the freshness of produce relies on human observation—examining colour, texture, smell, and visible damage. While this manual inspection method is simple, it is often inaccurate, inconsistent, and heavily dependent on individual experience.

With increasing concerns about food safety, rising food wastage, and the need for quality assurance in retail and household settings, there is a growing demand for an automated system that can accurately detect the freshness of fruits and vegetables. Machine Learning and Computer Vision provide the capability to analyse images and identify subtle changes that indicate ripeness or spoilage. This project focuses on developing such an automated freshness detection system.

1.2 Problem Overview

The primary problem addressed in this project is the lack of an objective, fast, and reliable method for evaluating the freshness of fruits and vegetables. Consumers, retailers, and supply chains face challenges such as:

- Inability to accurately judge freshness through visual inspection alone
- Inconsistent assessments caused by human subjectivity
- Difficulty predicting how long produce will remain usable
- Increased food wastage due to premature disposal
- Challenges in maintaining quality across storage, transportation, and retail stages

Since early spoilage indicators (slight discoloration, minor bruises, texture changes) may not be easily visible to the human eye, an automated system is needed to detect freshness more accurately than manual inspection.

1.3 Problem Context and Challenges

Freshness detection of fruits and vegetables is a complex and multidimensional challenge involving biological variations, environmental influences, visual inconsistencies, and subjective human judgment. The complexity arises because produce is inherently perishable and undergoes diverse biochemical changes that differ drastically across species, varieties, and environmental conditions. In real-world settings such as households, markets, warehouses, and agricultural supply chains, assessing the freshness and usability of produce becomes even more complicated. This section elaborates on the **core challenges** that make the development of an automated freshness detection system both necessary and difficult.

1.3.1 Variability in Produce

Fruits and vegetables show **natural biological variability**, which makes it extremely difficult to generalize freshness detection using simple rules or manual inspection. Each produce item is unique due to its growth conditions, handling environment, and storage timeline.

I. Natural Color Differences

Even within the same species, fruits and vegetables may exhibit differences in colour due to:

- Genetic variations
- Soil quality
- Sunlight exposure
- Temperature conditions during growth
- Use of fertilizers and pesticides

For example, mangoes from two different regions can have distinctly different shades even when

at the same ripeness stage. Apples may vary between deep red, light red, yellowish, or striped depending on the variety.

II. Variation in Size and Shape

Size and shape variations also create challenges for image-based detection systems. The same type of fruit or vegetable may grow in:

- Different dimensions
- Different curvatures
- Slightly deformed structures
- Irregular outer appearances

This makes it difficult for traditional image processing systems to detect consistent patterns.

III. Diverse Ripening Patterns

Different produce items follow different ripening and spoilage processes.

For example:

- Bananas ripen very quickly due to ethylene production.
- Citrus fruits spoil slowly but dry out internally.
- Tomatoes soften while showing minimal external colour change initially.

Even within the same fruit type, different varieties ripen and spoil at different rates. A Cavendish banana bruises and ripens differently from a Red banana. A grape variety with thin skin spoils quickly, while a thicker-skinned variety lasts longer.

IV. Differences in Spoilage Behaviour

Spoilage signs can vary widely. Some items show early external symptoms while others degrade internally first.

For instance:

- Pomegranates remain visually fresh externally while internal seeds may have turned brown.

- Watermelons can develop internal cracks without any outer visible defect.
- Potatoes may sprout, soften, or turn green depending on storage conditions.

This variability makes manual freshness detection extremely unreliable and inconsistent. A universal automated system must be capable of adapting to all these biological differences.

1.3.2 Visual Similarity Across Freshness Levels

A major challenge arises due to the **visual overlap between fresh, ripe, semi-ripe, overripe, and spoiled produce**. Early spoilage often resembles freshness, making manual and camera-based detection very difficult.

I. Early Spoilage is Hard to Identify

At the initial stages of decomposition, the produce may appear almost identical to fresh items. Slight colour variations or subtle textural changes are often invisible to the human eye. Examples include:

- A banana that is just beginning to overripe may still appear yellow under indoor lighting.
- A tomato that has started internal decomposition may look perfectly fine externally.
- A mango may have internal fungal buildup long before visible black spots form.

These similarities lead to inaccurate assessments, especially when consumers rely only on colour or surface appearance.

II. Poor Lighting Exaggerates the Problem

Under low light or fluorescent lighting in stores and kitchens, early signs of spoilage become even harder to detect.

For example:

- Greenish shades on potatoes indicating toxicity may look normal under yellow light.
- Minor mold spots on berries may not be visible under uneven lighting.

III. Texture-Based Spoilage is Often Invisible

Some spoilage indicators are textural (softening, wrinkling, or firmness changes), which are difficult to capture using 2D images. A fruit may feel soft to touch but look perfectly fine in photos.

IV. Need for highly sensitive models

Because early-stage spoilage signs are extremely subtle, the detection model must be trained on fine-grained differences, requiring:

- High-resolution images
- Diverse datasets
- Well-structured feature extraction methods

These challenges highlight why simple vision-based systems often fail in real-world freshness detection tasks.

1.3.3 Environmental and External Factors

Freshness detection becomes more complex due to **uncontrollable environmental factors** that affect the captured image and thus influence both human and machine predictions.

I. Lighting Variations

Lighting is one of the biggest challenges in image-based detection.

Images may be captured in:

- Natural sunlight
- Dim rooms
- Flashlight environments
- Store lighting (fluorescent or LED)
- Harsh overhead lights creating shadows

Different lighting conditions drastically change the perceived colour, brightness, shadows, and

contrasts, causing model misclassification.

II. Shadows and Reflections

Shadows may hide spoilage spots, while reflections from shiny surfaces like apples or tomatoes can trick algorithms into detecting false patterns.

For example:

- Glossy apples reflect overhead lights, masking bruises.
- Wet fruit surfaces reflect white patches that resemble defects.

These visual distortions make it difficult for models to maintain consistency.

III. Background Clutter

Images captured at home or in markets often include clutter such as:

- Kitchen shelves
- Plastic bags
- Other fruits
- Packaging material
- Human hands holding the produce

Cluttered backgrounds make segmentation harder and introduce noise into the prediction.

IV. Camera Quality Differences

Users may capture produce using:

- High-end smartphone cameras
- Low-resolution old devices
- Webcams
- Grocery store CCTV cameras

Low resolutions or noise create blurry images, reducing detail and hurting accuracy.

V. Variability in Angle and Distance

Fruits captured from different angles present different features.

Examples:

- A bruise visible from one side may be invisible from another.
- A compressed or elongated perspective can distort the fruit's natural shape.
- Zoomed-in images lose contextual cues; zoomed-out images lose detail.

VI. Need for robust preprocessing

Because of these factors, the system must apply:

- Normalization
- Contrast adjustments
- Shadow reduction
- Noise filtering
- Background removal

1.3.4 Subjective Nature of Human Judgement

Freshness evaluation varies widely from person to person due to differences in personal preferences, cultural habits, and experience. What one individual considers ripe or acceptable (like a soft mango or a slightly bruised apple), another may classify as spoiled. Cultural expectations also differ—banana, tomato, or mango ripeness standards change across regions.

Even trained staff may misjudge freshness when working quickly, under poor lighting, or handling large amounts of produce. Human fatigue and inconsistency further reduce accuracy.

Because of these subjective differences, manual judgment is unreliable. An automated system provides standardized, objective, and consistent results through measurable indicators such as freshness percentage, estimated usable days, and heatmap-based explanations.

CHAPTER 2

LITERATURE SURVEY

Automatic detection of freshness and ripeness in fruits and vegetables has evolved into a major research focus due to rising concerns about food safety, increasing food waste, and the urgent need for quality assurance in global supply chains. Fruits and vegetables are highly perishable commodities whose quality deteriorates rapidly due to biochemical reactions, microbial activity, moisture loss, and improper storage conditions. Traditionally, consumers and vendors have relied on manual inspection methods such as observing colour, checking firmness, or noticing Odor to assess freshness. However, these human-centred approaches are subjective, non-standardized, and prone to inconsistencies. In response, computer vision and machine learning techniques have become powerful tools for developing automated, objective, and scalable systems for freshness detection.

Early research efforts primarily relied on classical image-processing techniques and traditional machine-learning algorithms. Researchers extracted hand-crafted features such as colour histograms, texture descriptors (e.g., GLCM, LBP), shape and morphological attributes, edge patterns, and surface irregularities. These features were then fed into machine-learning classifiers like Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), Decision Trees, Naïve Bayes, and Logistic Regression for sorting fruits and vegetables into different quality categories. Although these techniques achieved acceptable results under controlled conditions, they suffered from significant limitations when applied in uncontrolled, real-world environments. Variations in illumination, camera distance, background clutter, orientation, and natural variability within fruit varieties often resulted in inconsistent outputs. In particular, traditional feature-based methods struggled to detect subtle, early-stage spoilage indicators such as minor bruises, slight texture changes, or small fungal growth. This lack of robustness created a need for more flexible and adaptive systems.

The introduction of deep learning, especially Convolutional Neural Networks (CNNs), significantly transformed the field by enabling computers to automatically learn hierarchical visual features without manual feature engineering. CNNs excel at capturing complex patterns

such as colour gradients, texture irregularities, shine variations, and structural deformities, which are critical cues for freshness or spoilage detection. With the availability of transfer learning models such as VGG-16, VGG-19, ResNet-50, Inception-V3, MobileNet, EfficientNet, and DenseNet, researchers began leveraging pre-trained architectures trained on large-scale datasets (like ImageNet) to analyse fruit and vegetable images. These pre-trained models provided rich feature representations that greatly reduced the amount of required training data and improved generalization across different environments. As a result, deep-learning-based freshness detection systems consistently outperformed classical models in terms of accuracy, robustness, and adaptability.

A major development in recent literature is the shift from simple binary classification (fresh vs. rotten) toward more nuanced multi-class grading systems. Many researchers classify produce across multiple ripeness or spoilage stages—for example: unripe, semi-ripe, ripe, overripe, partially spoiled, and fully spoiled. Such multi-class grading is extremely valuable for post-harvest handling, inventory management, and supply-chain decision making, where knowing the precise stage of freshness can help reduce waste and optimize distribution. For instance, bananas transition through multiple colour stages, and identifying the exact ripeness level is crucial for retailers and distributors. Similarly, tomatoes, strawberries, mangoes, and apples exhibit complex patterns of degradation that require multi-stage analysis for accurate assessment. Multi-label or multi-task deep learning models have been proposed that combine fruit/vegetable identification with simultaneous evaluation of ripeness levels, allowing a single system to perform multiple quality checks at once.

Another promising direction in recent studies is the development of hybrid and multi-task deep learning systems. These models integrate CNNs with other machine learning techniques—such as Recurrent Neural Networks (RNNs), LSTMs, Transformers, or classical classifiers like SVM—to improve precision and interpretability. Hybrid approaches are beneficial in capturing temporal patterns in spoilage progression or associating multiple dependent outputs such as type detection, freshness estimation, and damage localization. Some systems incorporate object detection models such as YOLO or Faster R-CNN to locate fruit regions in cluttered scenes before applying freshness classification, making them suitable for market or warehouse environments where multiple items appear in the same frame. These innovations represent an important advancement toward deploying freshness detection systems in realistic, uncontrolled settings.

Explainable AI (XAI) has become increasingly important in this domain as well. Modern systems incorporate techniques such as Grad-CAM, Class Activation Mapping (CAM), and saliency maps to visualize the regions of an image that influence the model's decision. Heatmaps generated through these methods highlight specific portions of the fruit or vegetable, such as mouldy areas, bruises, texture inconsistencies, or colour deformations. This provides deeper insight into the model's internal reasoning, helping developers diagnose errors and allowing end users to trust the predictions. Transparency is particularly important in consumer-facing applications, where users may hesitate to rely solely on automated decisions regarding food quality and safety.

Despite the significant progress in deep-learning-based freshness analysis, several limitations persist in existing research. Many models are still trained and evaluated on datasets collected under controlled studio conditions with uniform lighting, clean backgrounds, and minimal noise. Such environments do not reflect the conditions encountered in real supermarkets, kitchens, or farms. As a result, models that perform well in laboratory settings may struggle in practical usage. Additionally, datasets used in most studies lack diversity in terms of fruit varieties, spoilage patterns, maturity stages, and environmental conditions. Many research datasets include only common fruits like apples, bananas, oranges, and tomatoes, leaving a large variety of seasonal and regional produce unrepresented.

Another major gap in current research is the absence of shelf-life prediction. While many papers focus on classifying freshness, they rarely attempt to predict **how many days** a fruit or vegetable will remain edible under different storage environments (room temperature or refrigeration). Shelf-life estimation requires modeling both visual spoilage patterns and environmental factors—a far more complex challenge. Nutritional analysis is also largely absent from existing freshness detection systems. Users not only want to know if the item is fresh, but also whether it is beneficial to consume, how many calories it contains, and what vitamins or minerals it provides. Without such information, freshness detection systems remain incomplete from a consumer perspective.

Furthermore, many existing models are computationally heavy and require powerful GPUs for inference, making them unsuitable for deployment on mobile devices or low-resource hardware. Real-world applications—such as scanning fruits in a grocery store using a smartphone—require lightweight, optimized models capable of delivering instant predictions. This challenge remains largely unaddressed in academic literature.

In contrast to these limitations, the proposed system introduces several advancements that directly address the shortcomings found in prior studies. By predicting a **continuous freshness percentage**, it offers a more intuitive and informative output than discrete class labels. The inclusion of **usable days estimation**—both with and without refrigeration—adds a practical dimension that is highly valuable for consumers, retailers, and supply-chain operators. The integration of **nutritional details and health benefits** further enhances the system’s real-world utility by supporting informed consumption decisions. Moreover, the system leverages heatmap-based explainability to justify predictions, bridging the gap between accuracy and interpretability. By combining deep-learning classification, explainable AI, nutritional intelligence, and storage-based forecasting, the system achieves a level of completeness not commonly found in prior research. The result is a highly practical, real-world-ready solution suitable for household use, retail environments, and food-quality monitoring applications.

CHAPTER 3

MAJOR OBJECTIVE & SCOPE OF PROJECT

3.1. Major Objectives of the Project

The primary objective of this project is to design and implement an intelligent system capable of detecting and analysing the freshness of fruits and vegetables using Machine Learning and Computer Vision techniques. The system aims to overcome the limitations of manual quality inspection methods by providing accurate, fast, and user-friendly predictions based on image analysis.

This project focuses on developing an application that uses captured images of fruits and vegetables to assess their current quality, suggest their remaining shelf life, provide nutritional benefits, and assist users in making informed decisions before consumption. The objectives have been formulated keeping in mind consumer convenience, food safety, and reduction of food wastage.

The major objectives include:

1. To classify fruits and vegetables based on freshness level
 - Implement deep learning-based classification for different stages of ripeness and spoilage.
 - Identify defects such as bruises, colour changes, mold, and soft spots.
2. To determine the freshness percentage
 - Provide a measurable score indicating how fresh a fruit/vegetable currently is.
3. To estimate shelf-life predictions
 - Predict the number of usable days with and without refrigeration.

- Suggest optimal storage guidelines for extended usability.
4. To provide nutritional details and calorie value
- Display calories, vitamins, minerals, and essential nutrients present.
 - Offer health benefits associated with consumption.
5. To integrate heatmap visualization for model transparency
- Show users which regions of the image influenced the prediction.
 - Build trust by making AI decisions explainable.
6. To develop a user-friendly interface for scanning
- Allow real-time scanning using smartphones/cameras.
 - Provide results instantly with a simple and clean design.
7. To reduce food wastage and enhance consumer health awareness
- Encourage smart decision-making in buying and eating fresh produce.
8. To support supermarkets, vendors, and supply chains
- Help maintain inventory quality and improve storage management.

These objectives align with sustainable development goals related to food availability, reduced wastage, and public health improvement.

3.2. Scope of the Project

The scope defines the boundaries, functionality, and potential applications of the proposed system. This project's scope is designed to ensure practical feasibility and wide usability across different audiences such as consumers, retail markets, and food suppliers.

Project Scope Includes:

- Fresh Produce Classification
 - Identification of multiple types of fruits and vegetables.
 - Categorization into various freshness levels using visual parameters.
- Image Processing and ML-based Prediction
 - CNN-based deep learning model for analysing visual features.
 - Dataset trained on different freshness stages to improve accuracy.
- Shelf-Life Analysis
 - Provides prediction under normal temperature and refrigerated conditions.
 - Helps users avoid consumption of spoiled items and reduces health risks.
- Nutritional Information Module
 - Displays calorie intake and health benefits to promote wise food choices.
- Explainable AI (XAI) Integration
 - Heatmaps showing decision-influencing features for transparency and reliability.
- Mobile and Web Integration
 - Can be developed as a mobile app or web platform for ease of access.
 - Suitable for household and commercial environments.

CHAPTER 4

PROBLEM ANALYSIS & REQUIREMENT SPECIFICATION

4.1 Problem Analysis

Freshness assessment of fruits and vegetables is a critical factor for ensuring food quality, safety, and reduced wastage. Traditionally, this task is performed manually by visually examining colour, texture, and smell. However, manual judgment is subjective, varies from person to person, and becomes highly inefficient when dealing with large quantities, such as in supermarkets or cold-storage facilities.

Fresh produce undergoes gradual biochemical changes such as colour fading, development of dark spots, loss of firmness, dehydration, and microbial spoilage. These features are visible and can be captured using images. Therefore, image-based analysis using Machine Learning models can automatically identify these patterns and determine the freshness level.

The proposed system aims to analyse the uploaded or scanned image of fruits or vegetables, detect spoilage patterns, calculate a freshness percentage, and predict how long the produce will remain usable under both room temperature and refrigeration. Additionally, the system provides nutritional information such as calories and health benefits for user awareness.

Thus, the problem can be broken down into the following key challenges:

- ➔ Identification of the fruit or vegetable type from the image
- ➔ Differentiating between fresh, ripening, and spoiled stages
- ➔ Handling variations in lighting, backgrounds, and camera quality
- ➔ Providing shelf-life prediction and nutritional data
- ➔ Offering transparency in prediction using visual explanation (heatmaps)

- ➔ This analysis reveals the necessity of a robust and intelligent automated solution capable of evaluating freshness consistently and providing meaningful decision support to end-users.

4.2 Requirement Specification

4.2.1. Functional Requirements

The system shall:

- ➔ Allow users to capture or upload an image of a fruit/vegetable.
- ➔ Detect and classify the type of fruit or vegetable present.
- ➔ Analyse visual characteristics and compute freshness percentage (0–100%).
- ➔ Predict remaining usability days:
 - ➔ Without refrigeration
 - ➔ With refrigeration
- ➔ Display nutritional details:
 - ➔ Calorie content
 - ➔ Major vitamins/minerals and health benefits
- ➔ Generate a heatmap visualization to highlight spoiled/damaged regions that influenced the prediction.
- ➔ Store previous scan results for tracking (optional based on UI).
- ➔ Provide user-friendly output including recommendations (e.g., “Consume soon”, “Fresh”, “Not safe”).

4.2.2. Non-Functional Requirements

- ➔ Performance - Image processing and prediction should execute within a few seconds.
- ➔ Accuracy - The model must ensure high prediction accuracy in real-world conditions.

- ➔ Usability - Interface should be intuitive for general users and retailers.
- ➔ Scalability - Should be able to handle multiple fruit types and database expansion.
- ➔ Security & Privacy - User-uploaded images must be processed securely without misuse.
- ➔ Portability - Deployable on web browsers or mobile devices for real-time scanning.
- ➔ Reliability - Consistent results regardless of lighting variation and noise.
- ➔ Explainability - Must visually justify predictions for user trust.

4.2.3. Hardware Requirements

- ➔ Smartphone camera / Laptop webcam for image capture
- ➔ System supporting model inference (PC or cloud server)

4.2.4. Software Requirements

- ➔ Programming Language: Python
- ➔ Frameworks: TensorFlow / PyTorch, OpenCV
- ➔ Deployment Tools: Flask / FastAPI / Streamlit
- ➔ Database: Firebase / MongoDB (if user history required)
- ➔ UI: Web or Mobile Interface

4.3 Outcome of Analysis

The analysis concludes that there is a strong need for an accurate, automated, and easily accessible system that can:

- Identify freshness visually
- Predict edible lifespan efficiently
- Provide useful dietary information
- Support households and food industry to reduce waste

The proposed Freshness Detection System fulfills these requirements through a Machine Learning-based approach integrated with nutritional intelligence and explainable outputs.

CHAPTER 5

DETAILED DESIGN (MODELING & ERD/DFD)

5.1 Detailed Design

The detailed design phase focuses on transforming the system requirements into structured models that clearly show how data flows through the system and how information is stored and organized. For the Freshness Detection System, modeling is done using:

- Data Flow Diagrams (DFD) – to represent the flow of data between processes, data stores, and external entities.
- Entity–Relationship Diagram (ERD) – to represent the logical structure of the database, including entities, attributes, and relationships.

These models help in understanding the internal working of the system before actual implementation and ensure that all functional requirements are correctly mapped to processes and data structures.

5.2 System Modeling Overview

The proposed system allows a user to capture or upload an image of a fruit or vegetable, after which the system:

1. Identifies the item type.
2. Analyses its freshness using a trained Machine Learning model.
3. Predicts remaining usable days under room and refrigerated conditions.
4. Fetches nutritional details and health benefits from a database.
5. Generates a result report containing freshness percentage, shelf-life, calories, and benefits.
6. Optionally stores the result and scan history for the user.

To represent this behaviour, the following models are used:

- Context Level DFD (Level 0) – overall view of the system as a single process.
- Level 1 DFD – decomposition into core processes such as image capture, prediction, nutrition fetch, and report generation.
- Level 2 DFD (optional) – further detailing of the prediction process.
- ERD – logical data model for users, items, scans, results, and nutrition information.

5.3 Data Flow Diagrams (DFD)

5.3.1 Context Level DFD (Level 0)

At the context level, the entire Freshness Detection System is represented as a single process that interacts with external entities. The main elements are:

➔ External Entities

- User – provides an image of the fruit/vegetable and receives the freshness report.
- Admin (optional) – manages fruit/vegetable catalogue, nutrition database, and system settings.

➔ Central Process

- Process 0: Freshness Detection System
 - Takes the input image from the user and returns:
 - Freshness percentage
 - Usable days (with and without refrigerator)
 - Nutritional information
 - Recommendation/remarks

➔ Data Flows

- From User → Process: Image, user input (optional weight, item selection)
- From Process → User: Freshness report, nutrition data, shelf-life result
- From Admin → Process: Master data updates (item types, nutrition values)

➔ Data Stores (conceptual at this level)

- D1: Item & Nutrition Database
- D2: Scan History / Results Database

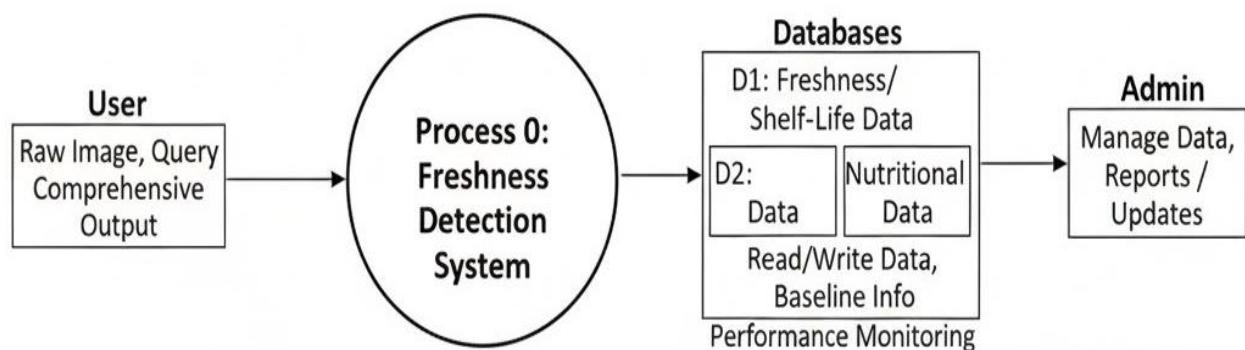


FIG 5.3.1.1 : LEVEL 0 DIAGRAM

5.3.2 Level 1 DFD

Level 1 DFD decomposes the main system into more detailed processes. The key processes are:

➔ Process 1.0 – Capture / Upload Image

- User captures or uploads an image.
- System validates format and forwards it for preprocessing.
- Data flows:
 - User → 1.0: Raw Image
 - 1.0 → 2.0: Validated Image

➔ Process 2.0 – Preprocess Image

- Resize, normalize, enhance contrast, remove noise, etc.
- Creates a standardized input for the ML model.
- Data flows:
 - 2.0 → 3.0: Preprocessed Image

➔ Process 3.0 – Predict Freshness & Shelf Life

- Invokes the trained ML model to:
 - Detect fruit/vegetable type
 - Compute freshness score / percentage
 - Estimate usable days (room + fridge) using internal logic/regression
- Data flows:
 - 3.0 → 4.0: Predicted item type, freshness %, usable days
 - 3.0 → D2: Store scan result details

➔ Process 4.0 – Fetch Nutrition & Benefits

- Uses the item type predicted in Process 3.0 to query the nutrition database.
- Retrieves calories per 100g, nutrient values, and associated health benefits.
- Data flows:
 - 4.0 ↔ D1: Item type ↔ Nutrition Data
 - 4.0 → 5.0: Nutrition + benefit details

➔ Process 5.0 – Generate Result Report

- Combines output from Process 3.0 and 4.0.
- Optionally generates a heatmap image and explanation text.
- Returns the final report to the user interface.
- Data flows:
 - 5.0 → User: Final report (freshness, shelf life, calories, benefits, heatmap)
 - 5.0 → D2: Full scan report record

➔ Data Stores in Level 1

- D1 – Item & Nutrition Database
 - Stores fruit/vegetable catalogue, calories, vitamins, benefits, default shelf-life data.
- D2 – Scan History / Results
 - Stores user scan logs including timestamp, predicted values, and optional feedback.

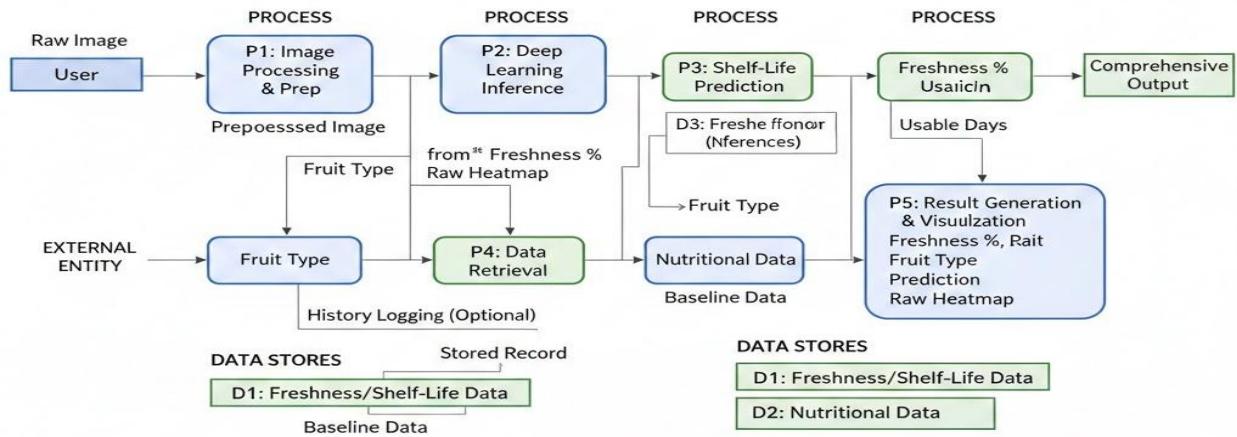


FIG 5.3.2.1 : LEVEL 1 DIAGRAM

5.3.3 Level 2 DFD – Detailed Prediction Process (Process 3.0)

Process 3.0 – Predict Freshness & Shelf Life can be further decomposed into:

- Process 3.1 – Feature Extraction
 - Extract key features from the pre-processed image (colour patterns, texture, defects).
 - Internally handled by CNN layers.
- Process 3.2 – Item Type Classification
 - Classifies which fruit/vegetable is in the image (apple, banana, tomato, etc.)
- Process 3.3 – Freshness Scoring
 - Based on visual features, computes freshness score/percentage for the detected item.
- Process 3.4 – Shelf-Life Estimation

- Uses freshness score + item base shelf-life parameters to calculate:
 - Remaining usable days at room temperature
 - Remaining usable days in refrigerator
- Process 3.5 – Heatmap Generation (Explainability)
 - Applies Grad-CAM or similar method to highlight important regions in the image.
 - Produces a heatmap overlay image.
- Data Flows (within 3.x)
 - Preprocessed image → 3.1 → 3.2 → 3.3 → 3.4
 - Predicted item type and freshness score → 3.4 & 4.0
 - Preprocessed image + model gradients → 3.5 → Heatmap image → 5.0

CHAPTER 6

HARDWARE/SOFTWARE PLATFORM ENVIRONMENT

6.1 Hardware Requirements

To efficiently run and test the proposed Machine Learning-based Freshness Detection System, the following hardware environment is recommended:

User / Development System

- **Processor:** Intel Core i5 / AMD Ryzen 5 or higher
- **RAM:** Minimum 8 GB (16 GB recommended for model training)
- **Storage:** Minimum 256 GB SSD (for dataset and model storage)
- **Graphics:**
 - Integrated GPU supported system (for lightweight training)
 - NVIDIA GPU with CUDA support recommended for faster model training
- Camera:
 - Smartphone Camera (8 MP or higher)
 - Web Camera / USB Camera for image capture in real-time scanning
- Display: 1080p or higher resolution screen
- Other Accessories: Wi-Fi connectivity, Mouse/Keyboard

6.2 Software Requirements

The proposed system integrates various software components for model building, deployment, and user interface development.

Operating System :

- Windows 10 / 11 (development environment)
- Android OS / Web Browser (end-user environment)

Programming Languages:

- **Python 3.8+** for Machine Learning Model
- JavaScript / Django / Flask (if web & app support is implemented)

Software Tools & Libraries:

- Jupyter Notebook / Google Colab – Model training & testing
- TensorFlow / Keras – Deep Learning model development
- OpenCV – Image capturing, preprocessing & feature enhancement
- NumPy, Pandas, Matplotlib – Data handling & visualization
- Grad-CAM / Heatmap libraries – Model explainability

6.3 Application Development Platforms

Depending on how the system is delivered to users, various application platforms may be utilized:

- Android Studio for developing an Android mobile app allowing users to scan fruits/vegetables directly from their phone camera.
- Flask / Django for deploying the ML model as a cloud or local API connected with a frontend interface.
- React Native or Flutter may optionally be used for cross-platform deployment supporting both Android and iOS users.
- 2.5 Version Control & Collaboration Tools GitHub Used for source code management, project documentation, issue tracking, and collaborative development among team members. It ensures version control, making the project scalable and easier to maintain or update in future enhancements.

CHAPTER 7

SNAPSHOT OF INPUT & OUTPUT

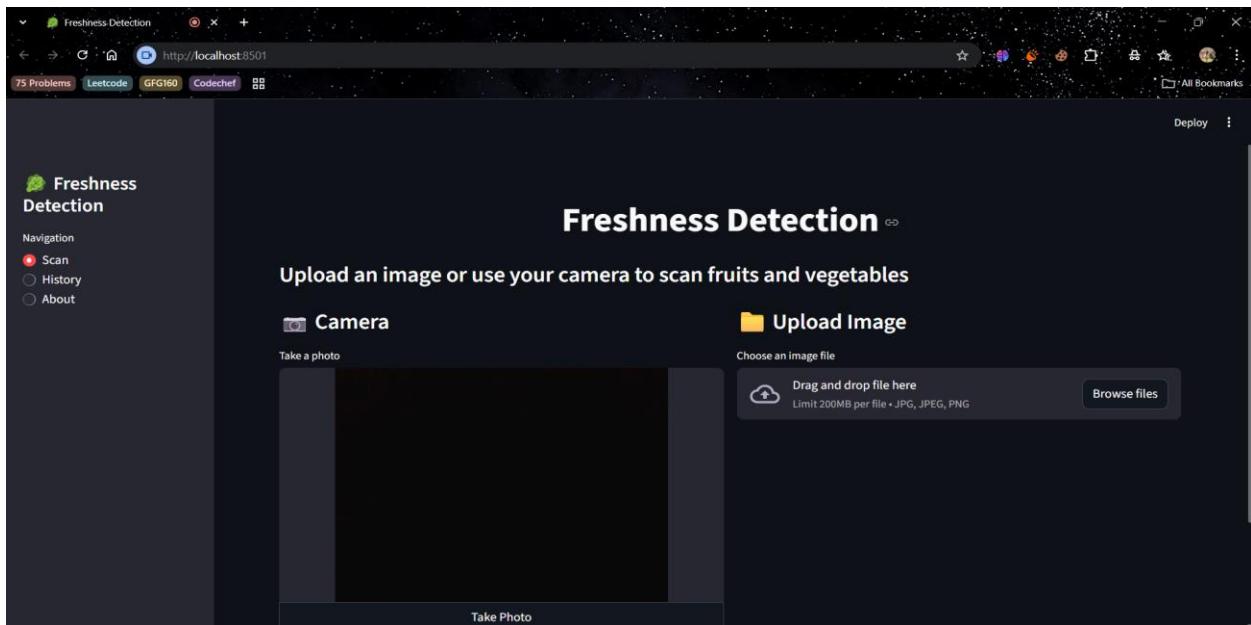


FIG 7.1 USER INTERFACE

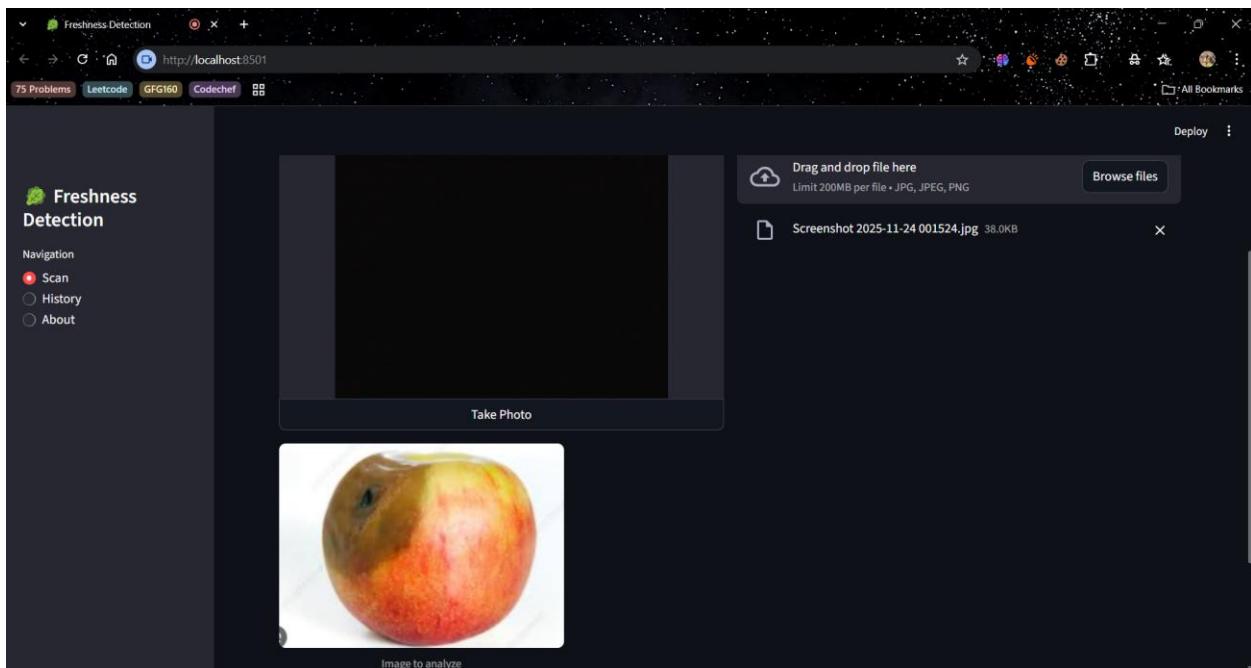


FIG 7.2 INPUT GIVEN AS A PHOTOGRAPH OF FRUIT OR VEGETABLE

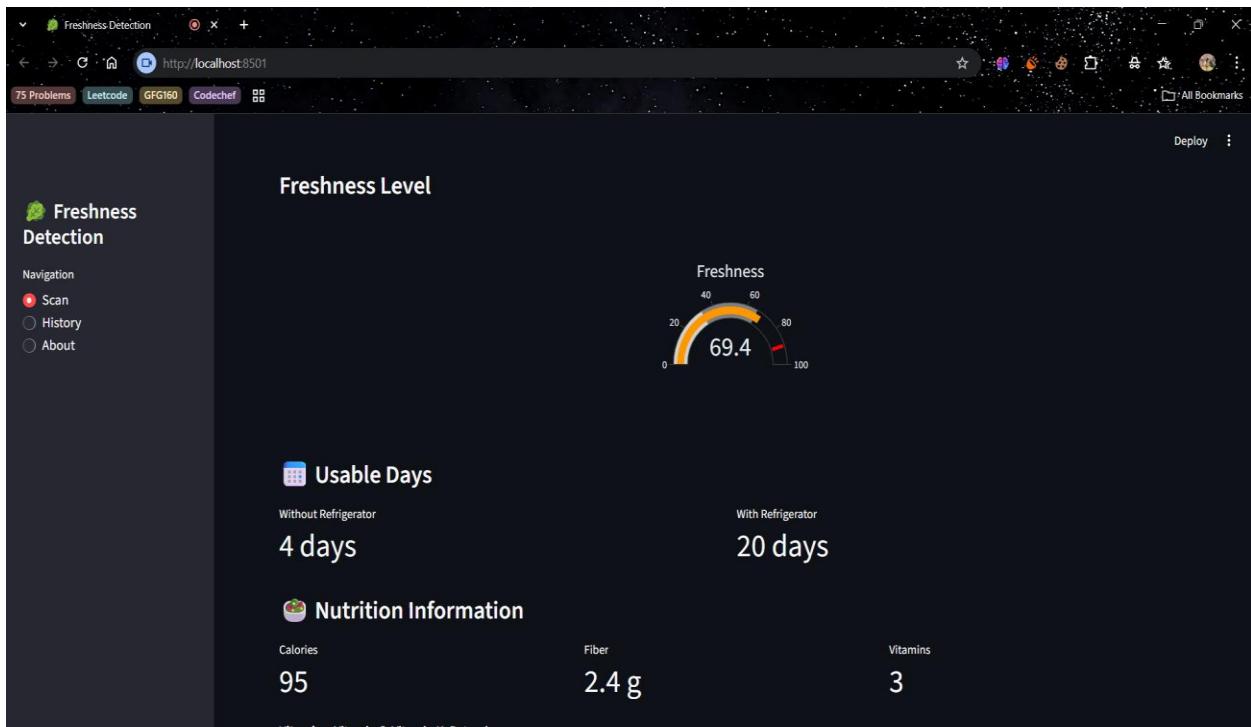
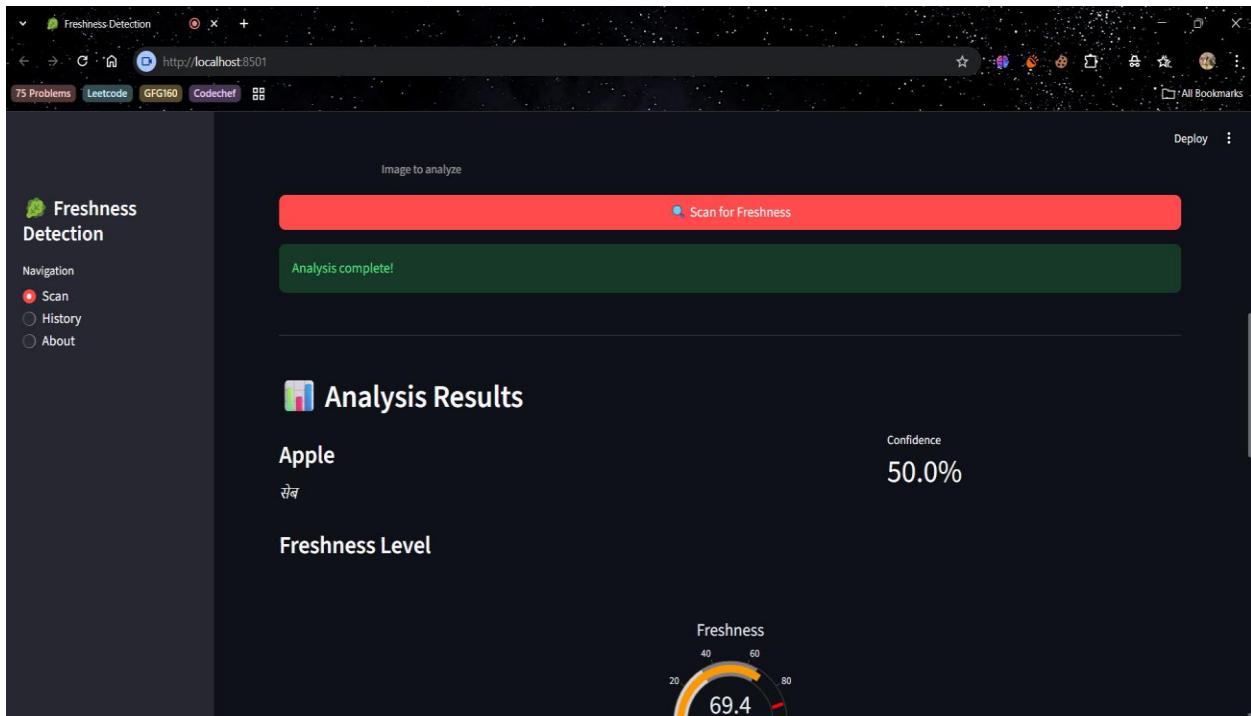


FIG 7.3 SCAN & ANALYSIS OF FRESHNESS LEVEL

The screenshot shows a dark-themed web application titled "Freshness Detection". On the left sidebar, there's a navigation menu with "Scan" selected. The main content area displays a section titled "Health Benefits" with a list of points: "Rich in antioxidants that may reduce risk of chronic diseases", "High fiber content aids digestion and promotes gut health", "May help lower cholesterol levels", "Contains quercetin which may boost brain health", and "Low calorie density helps with weight management". Below this is a section titled "Action Suggestion" with a message: "Use within 20 days - Good freshness. Store in refrigerator for longer shelf life." A link "Why this result? (View Heatmap)" is provided. At the bottom, there are four buttons: "Save to History", "View Recipes", "Share", and "Scan Another".

FIG 7.4 HEALTH BENEFITS OF SCANNED FRUIT & VEGETABLE

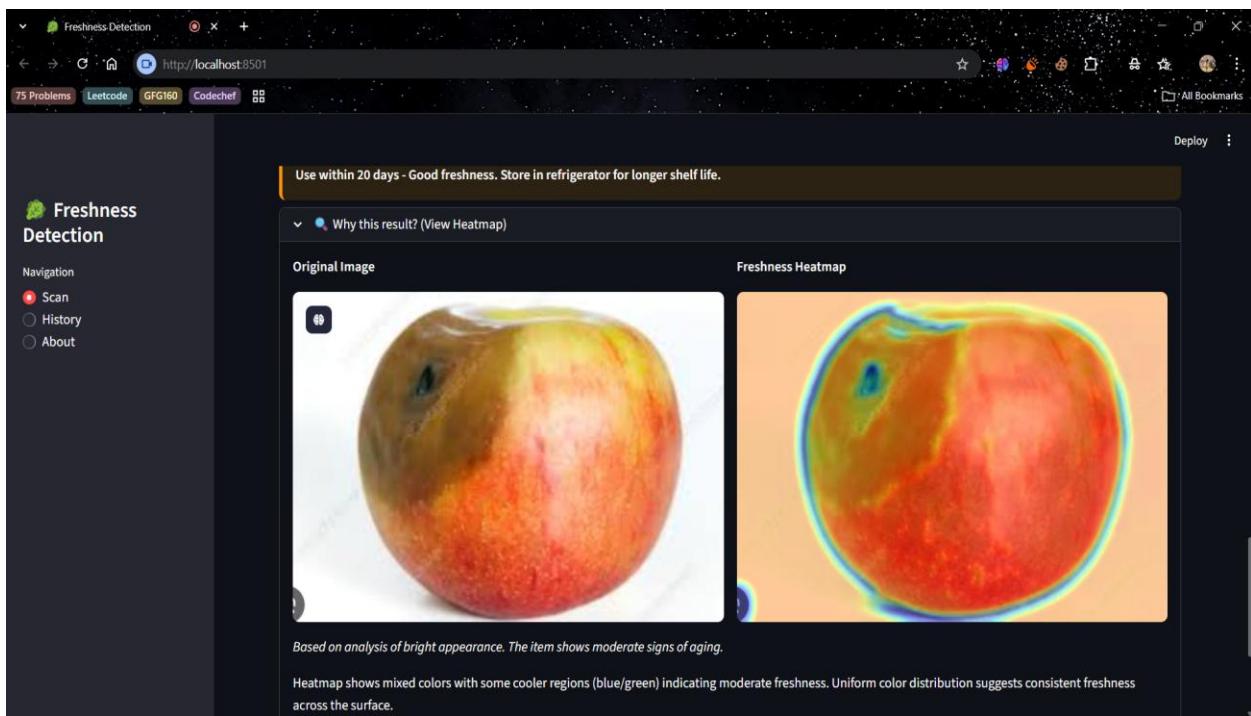


FIG 7.5 HEATMAP STUDY OF THE IMAGE

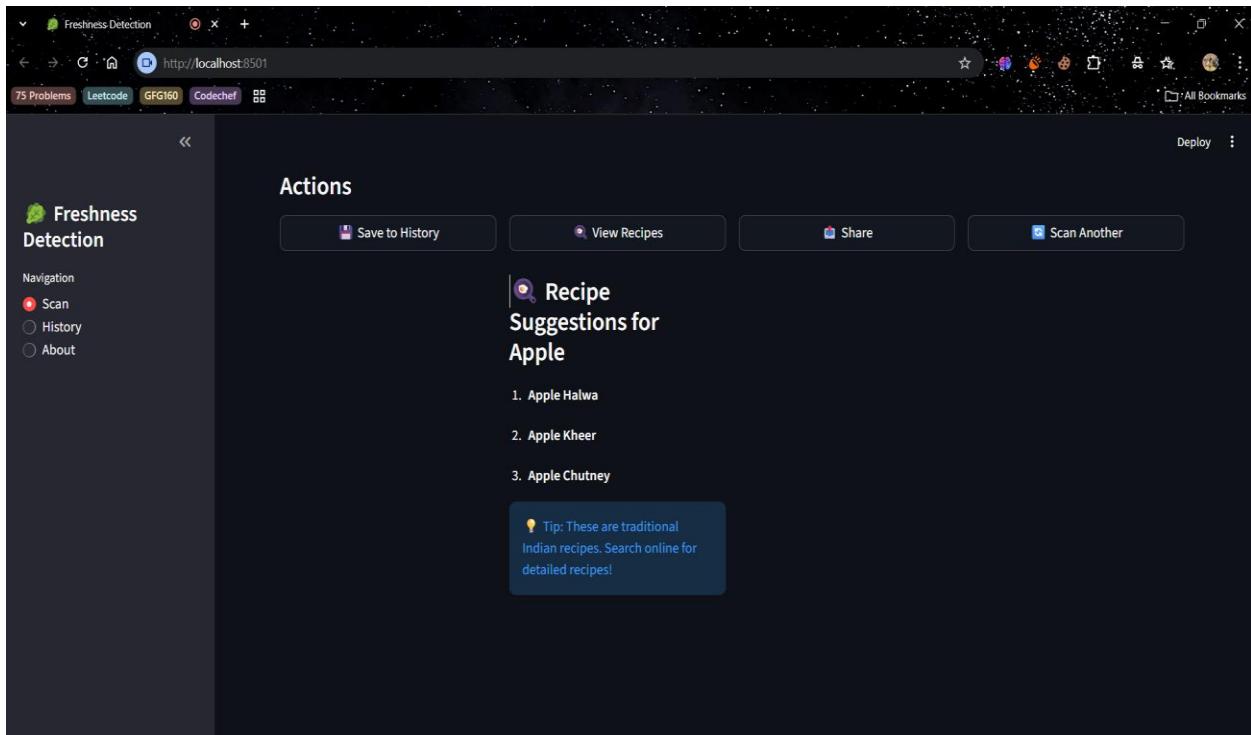


FIG 7.6 SUGGESTED RECIPES FOR THE IMAGE

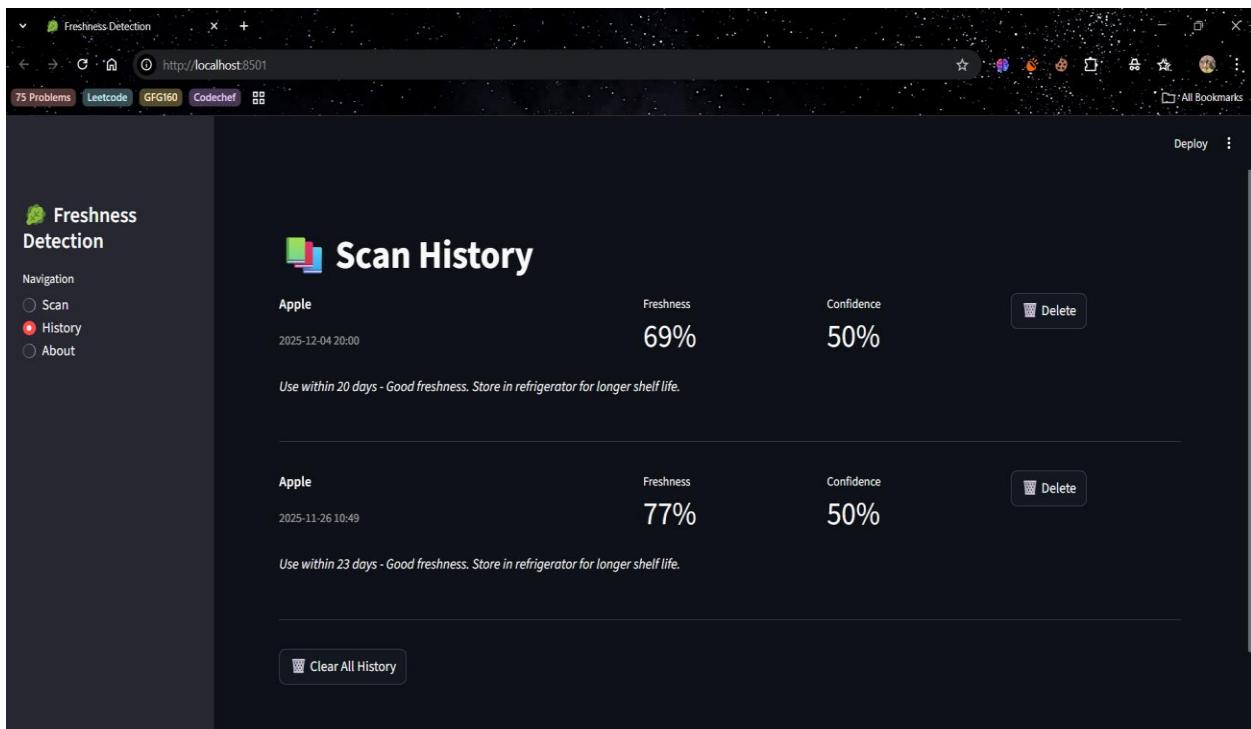


FIG 7.7 SCAN HISTORY

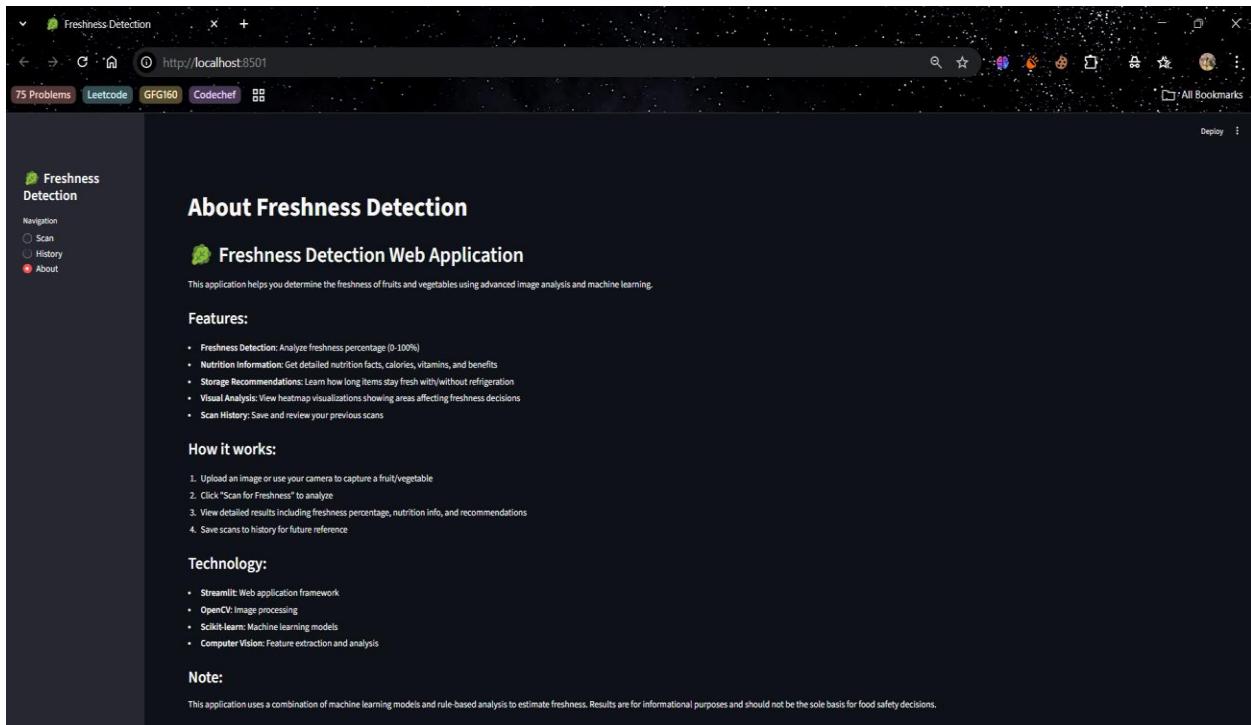
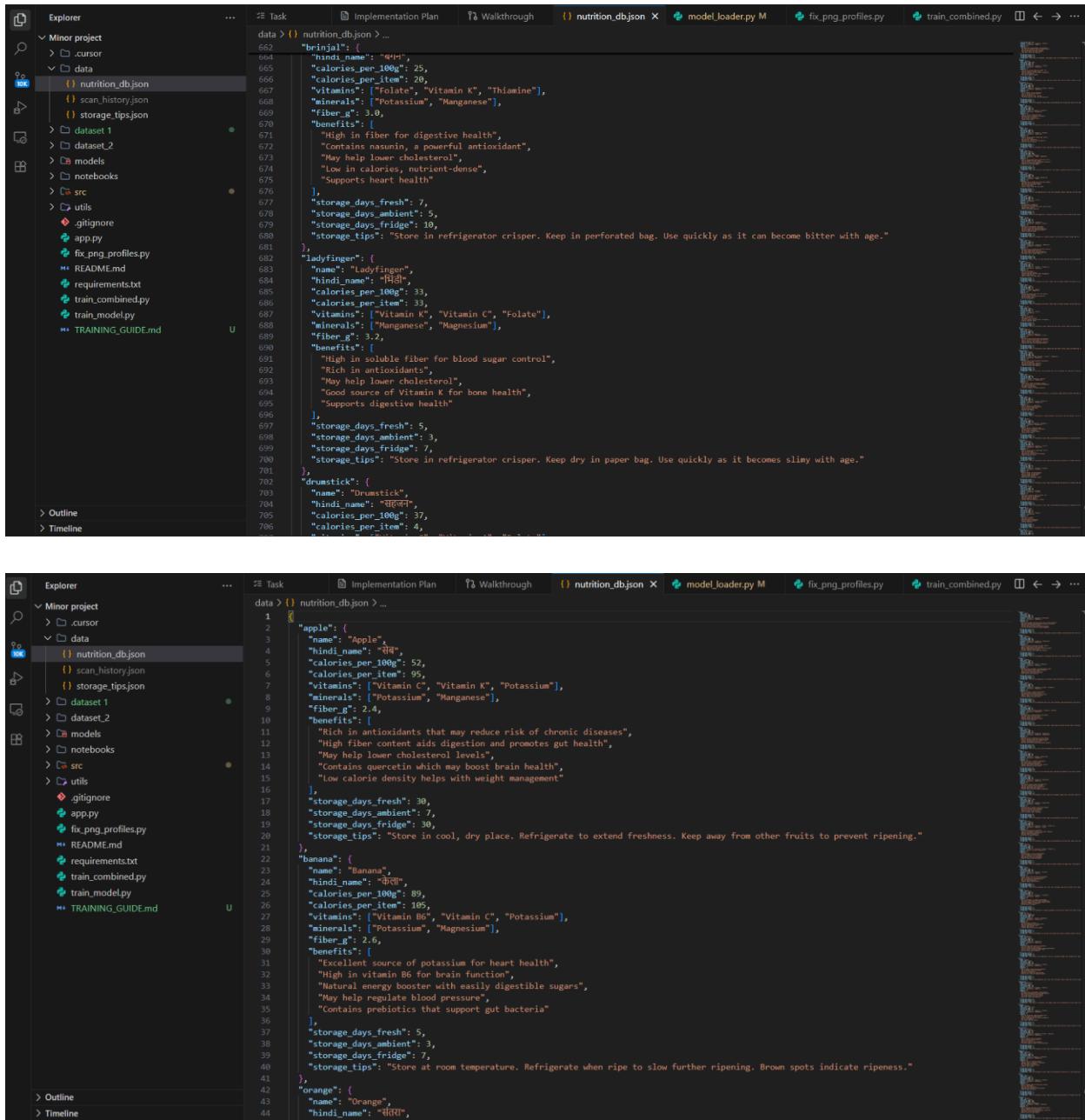


FIG 7.8 ABOUT FRESHNESS DETECTION

CHAPTER 8

CODING



The figure displays two screenshots of a code editor, likely PyCharm, showing the content of a file named `nutrition_db.json`. The left screenshot shows the initial state of the database with entries for brinjal, ladyfinger, and drumstick. The right screenshot shows the database after modifications, listing apple, banana, and orange.

```
data > [ ] nutrition_db.json > ...
662     "brinjal": {
663         "name": "Brinjal",
664         "hindu_name": "वृंदा",
665         "calories_per_100g": 25,
666         "calories_per_item": 20,
667         "vitamins": ["Folate", "Vitamin K", "Thiamine"],
668         "minerals": ["Potassium", "Manganese"],
669         "fiber_g": 3.0,
670         "benefits": [
671             "High in fiber for digestive health",
672             "Contains nasunin, a powerful antioxidant",
673             "May help lower cholesterol",
674             "Low in calories, nutrient-dense",
675             "Supports heart health"
676         ],
677         "storage_days_fresh": 7,
678         "storage_days_ambient": 5,
679         "storage_days_fridge": 10,
680         "storage_tips": "Store in refrigerator crisper. Keep in perforated bag. Use quickly as it can become bitter with age."
681     },
682     "ladyfinger": {
683         "name": "Ladyfinger",
684         "hindu_name": "डिंडी",
685         "calories_per_100g": 33,
686         "calories_per_item": 33,
687         "vitamins": ["Vitamin K", "Vitamin C", "Folate"],
688         "minerals": ["Manganese", "Magnesium"],
689         "fiber_g": 3.2,
690         "benefits": [
691             "High in soluble fiber for blood sugar control",
692             "Rich in antioxidants",
693             "May help lower cholesterol",
694             "Good source of Vitamin K for bone health",
695             "Supports digestive health"
696         ],
697         "storage_days_fresh": 5,
698         "storage_days_ambient": 3,
699         "storage_days_fridge": 7,
700         "storage_tips": "Store in refrigerator crisper. Keep dry in paper bag. Use quickly as it becomes slimy with age."
701     },
702     "drumstick": {
703         "name": "Drumstick",
704         "hindu_name": "ठिठी",
705         "calories_per_100g": 37,
706         "calories_per_item": 4,
707         "storage_tips": "Store in refrigerator crisper. Keep dry in paper bag. Use quickly as it becomes slimy with age."
708 },
709 }
```



```
data > [ ] nutrition_db.json > ...
1     [
2         "apple": {
3             "name": "Apple",
4             "hindu_name": "आम",
5             "calories_per_100g": 52,
6             "calories_per_item": 95,
7             "vitamins": ["Vitamin C", "Vitamin K", "Potassium"],
8             "minerals": ["Potassium", "Manganese"],
9             "fiber_g": 2.4,
10            "benefits": [
11                "Rich in antioxidants that may reduce risk of chronic diseases",
12                "High fiber content aids digestion and promotes gut health",
13                "May help lower cholesterol levels",
14                "Contains quercetin which may boost brain health",
15                "Low calorie density helps with weight management"
16            ],
17            "storage_days_fresh": 30,
18            "storage_days_ambient": 7,
19            "storage_days_fridge": 10,
20            "storage_tips": "Store in cool, dry place. Refrigerate to extend freshness. Keep away from other fruits to prevent ripening."
21        },
22        "banana": {
23            "name": "Banana",
24            "hindu_name": "बड़ी",
25            "calories_per_100g": 89,
26            "calories_per_item": 195,
27            "vitamins": ["Vitamin B6", "Vitamin C", "Potassium"],
28            "minerals": ["Potassium", "Magnesium"],
29            "fiber_g": 2.6,
30            "benefits": [
31                "Excellent source of potassium for heart health",
32                "High in vitamin B6 for brain function",
33                "Natural energy booster with easily digestible sugars",
34                "May help regulate blood pressure",
35                "Contains prebiotics that support gut bacteria"
36            ],
37            "storage_days_fresh": 5,
38            "storage_days_ambient": 3,
39            "storage_days_fridge": 7,
40            "storage_tips": "Store at room temperature. Refrigerate when ripe to slow further ripening. Brown spots indicate ripeness."
41        },
42        "orange": {
43            "name": "Orange",
44            "hindu_name": "संगीर",
45            "calories_per_100g": 57,
46            "calories_per_item": 114,
47            "vitamins": ["Vitamin C", "Potassium", "Folate"],
48            "minerals": ["Potassium", "Magnesium"],
49            "fiber_g": 2.3,
50            "benefits": [
51                "High in vitamin C for immune system",
52                "Rich in fiber for digestive health",
53                "May help lower cholesterol",
54                "Good source of Vitamin C for skin health"
55            ],
56            "storage_days_fresh": 7,
57            "storage_days_ambient": 5,
58            "storage_days_fridge": 10,
59            "storage_tips": "Store in refrigerator crisper. Keep in perforated bag. Use quickly as it can become bitter with age."
60        }
61    ]
62 }
```

FIG 8.1 NUTRITION_DB.json

```

data > { scan_history.json ...
1 [
2   {
3     "id": 1,
4     "timestamp": "2025-11-26T10:49:54.369729",
5     "item_name": "apple",
6     "item_display_name": "Apple",
7     "freshness_percentage": 76.73959698703734,
8     "confidence": 0.5,
9     "usable_days_ambient": 5,
10    "usable_days_fridge": 23,
11    "calories": 95,
12    "action_suggestion": "Use within 23 days - Good freshness. Store in refrigerator for longer shelf life."
13  }
14 ]

```

FIG 8.2 SCAN_HISTORY.JSON

```

data > { storage_tips.json ...
1 {
2   "storage_conditions": {
3     "refrigerator": {
4       "humidity": "optimal (85-95%)",
5       "description": "crisper drawer or vegetable compartment maintains optimal humidity"
6     },
7     "ambient": {
8       "temperature_range": "18-25°C",
9       "humidity": "moderate (50-60%)",
10      "description": "room temperature, well-ventilated area away from direct sunlight"
11    },
12    "cool_dark": {
13      "temperature_range": "10-15°C",
14      "humidity": "low (40-50%)",
15      "description": "cool, dark, well-ventilated place like pantry or cellar"
16    }
17  },
18  "general_tips": [
19    "do not store fruits/vegetables until ready to use",
20    "prevent any damaged or spoiled items to prevent spread",
21    "store ethylene-producing fruits (apples, bananas) separately",
22    "keep root vegetables in cool, dark places",
23    "use perforated bags for better air circulation",
24    "check regularly and remove any spoiled items",
25    "store cut items in airtight containers in refrigerator"
26  ],
27  "indian_recipes": {
28    "apple": ["apple halwa", "apple kheer", "apple chutney"],
29    "banana": ["banana halwa", "banana dosa", "banana lassi"],
30    "mango": ["mango halwa", "mango lassi", "mango pickle", "mango chutney"],
31    "potato": ["aloo gobhi", "aloo paratha", "aloo tikki"],
32    "onion": ["onion ki sabzi", "onion pakora", "onion raita"],
33    "cucumber": ["cucumber raita", "cucumber salad", "cucumber pachadi"],
34    "spinach": ["palak paneer", "palak dal", "palak paratha"],
35    "jeera": ["jeera rice", "jeera masala", "jeera aloo", "jeera dal"],
36    "tandoori": ["tandoori masala", "tandoori chicken", "tandoori fish"],
37    "mutton": ["mutton masala", "mutton korma", "mutton naan", "mutton biryani"],
38    "bhindi": ["bhindi masala", "bhindi fry", "bhindi do pyaza"],
39    "mango": ["mango lassi", "mango pickle", "amras"],
40    "coconut": ["coconut chutney", "coconut ladoo", "coconut rice"],
41    "ghee": ["ghee", "tilgul", "ginger", "achari aloo", "garlic naan"],
42    "radish": ["radish", "radish chutney", "radish aloo", "radish dal"],
43    "coriander": ["coriander chutney", "coriander rice", "dhaniya aloo"],
44    "mint": ["mint chutney", "mint rice", "pudina paratha"],
45    "radish": ["mooli paratha", "mooli ki sabzi", "radish salad"],
46    "beetroot": ["beetroot kachumber", "beetroot halwa", "beetroot salad"],
47    "raw_mango": ["raw mango pickle", "raw mango chutney", "raw mango shakarkand"],
48    "peach": ["peach", "peach juice", "peach lassi", "peach ki sabzi"],
49    "bottle_gourd": ["bottle gourd", "lauki ki sabzi", "lauki kefta", "lauki dal"],
50    "bitter_gourd": ["bitter gourd", "karela fry", "stuffed karela", "karela sabzi"]
51  }
52 }

```

FIG 8.3 STORAGE_TIPS.json

```

1  [{"cell_type": "markdown", "source": "# Freshness Detection Model Training with EfficientNetB0\n"}, {"cell_type": "code", "source": "import sys\nimport os\nsys.path.append(os.path.join(os.path.dirname(os.getcwd()), 'src'))\n\nimport numpy as np\nimport cv2\nimport tensorflow as tf\nfrom model_loader import ModelTrainer, FreshnessModel\n\nprint(f'TensorFlow version: {tf.__version__}')"}, {"cell_type": "metadata", "metadata": {"kernelspec": {"display_name": "Python 3", "name": "python3"}}, "nbformat": 4, "nbformat_minor": 4}], [{"cell_type": "code", "source": "def _rule_based_freshness(self, features: Dict, item_name: str) -> float:\n    """\n        Rule-based freshness estimation using color and texture features.\n\n        Args:\n            features: Dictionary of extracted features\n            item_name: Name of the fruit/vegetable\n\n        Returns:\n            Estimated freshness percentage (0-100)\n    """\n    freshness = 75.0 # Base freshness\n\n    # Color-based rules\n    brightness = features.get('V_mean', 128) / 255.0 # Normalize to 0-1\n    saturation = features.get('S_mean', 128) / 255.0\n\n    # Higher brightness and saturation generally indicate freshness\n    if brightness > 0.6:\n        freshness += 10\n    elif brightness < 0.4:\n        freshness -= 15\n\n    if saturation > 0.5:\n        freshness += 5\n    elif saturation < 0.3:\n        freshness -= 10\n\n    # Texture-based rules\n    # Lower contrast and higher homogeneity might indicate uniform texture (fresh)\n    contrast = features.get('glcm_contrast_mean', 0.5)\n    homogeneity = features.get('glcm_homogeneity_mean', 0.5)\n\n    if contrast < 0.3:\n        freshness += 5\n    elif contrast > 0.7:\n        freshness -= 10\n\n    if homogeneity > 0.7:\n        freshness -= 5\n    elif homogeneity < 0.4:\n        freshness -= 10"}]

```

FIG 8.4 MODEL_TRAINING.ipynb

```

1  [{"cell_type": "code", "source": "def _rule_based_freshness(self, features: Dict, item_name: str) -> float:\n    """\n        Rule-based freshness estimation using color and texture features.\n\n        Args:\n            features: Dictionary of extracted features\n            item_name: Name of the fruit/vegetable\n\n        Returns:\n            Estimated freshness percentage (0-100)\n    """\n    freshness = 75.0 # Base freshness\n\n    # Color-based rules\n    brightness = features.get('V_mean', 128) / 255.0 # Normalize to 0-1\n    saturation = features.get('S_mean', 128) / 255.0\n\n    # Higher brightness and saturation generally indicate freshness\n    if brightness > 0.6:\n        freshness += 10\n    elif brightness < 0.4:\n        freshness -= 15\n\n    if saturation > 0.5:\n        freshness += 5\n    elif saturation < 0.3:\n        freshness -= 10\n\n    # Texture-based rules\n    # Lower contrast and higher homogeneity might indicate uniform texture (fresh)\n    contrast = features.get('glcm_contrast_mean', 0.5)\n    homogeneity = features.get('glcm_homogeneity_mean', 0.5)\n\n    if contrast < 0.3:\n        freshness += 5\n    elif contrast > 0.7:\n        freshness -= 10\n\n    if homogeneity > 0.7:\n        freshness -= 5\n    elif homogeneity < 0.4:\n        freshness -= 10"}]

```

FIG 8.5 FRESHNESS_ANALYZER.py

```

stination.py    data_loader.py    freshness_analyzer.py    heatmap_generator.py    model_loader.py    fix_png_profiles.py    train_combined.py
src > heatmap_generator.py
13  class HeatmapGenerator:
14      def generate_heatmap(self, image: np.ndarray, features: dict,
15          h, w = image.shape[1:2]
16          heatmap = np.zeros((h, w), dtype=np.float32)
17
18          # Convert to RGB for processing
19          rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
20          hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
21          gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
22
23          # Create heatmap based on various factors
24
25          # 1. Color-based regions (brightness and saturation)
26          brightness_map = hsv[:, :, 2].astype(np.float32) / 255.0
27          saturation_map = hsv[:, :, 1].astype(np.float32) / 255.0
28
29          # Higher brightness and saturation = higher heat (fresher areas)
30          color_score = (brightness_map * 0.6 + saturation_map * 0.4)
31
32          # 2. Texture-based regions (edge detection)
33          edges = cv2.Canny(gray, 50, 150)
34          edge_map = (edges > 0).astype(np.float32)
35
36          # More edges = lower heat (less fresh areas)
37          texture_score = 1.0 - (edge_map * 0.3)
38
39          # 3. Color uniformity (variance in local regions)
40          # Calculate local color variance
41          kernel_size = 15
42          kernel = np.ones((kernel_size, kernel_size), np.float32) / (kernel_size * kernel_size)
43
44          gray_float = gray.astype(np.float32)
45          local_mean = cv2.filter2D(gray_float, -1, kernel)
46          local_sq_mean = cv2.filter2D(gray_float ** 2, -1, kernel)
47          local_variance = local_sq_mean - local_mean ** 2
48
49          # Normalize variance
50          variance_norm = 1.0 - np.clip(local_variance / 1000.0, 0, 1)
51
52          # Combine factors
53          heatmap = (color_score * 0.4 + texture_score * 0.3 + variance_norm * 0.3)
54
55          # Adjust based on overall freshness
56          if freshness_low:
57              heatmap *= 0.5
58
59          # If freshness is low, highlight non-freshly areas more
60
61      return heatmap
62
63  
```

FIG 8.6 HEATMAP_GENERATOR.py

```

stination.py    data_loader.py    freshness_analyzer.py    heatmap_generator.py    image_processor.py    model_loader.py    fix_png_profiles.py
src > image_processor.py
15  class ImageProcessor:
16      def preprocess_image(self, image: np.ndarray) -> Dict[str, np.ndarray]:
17
18          # Normalize images
19          rgb_norm = rgb.astype(np.float32) / 255.0
20          hsv_norm = hsv.astype(np.float32) / 255.0
21          h, w = image.shape[0], image.shape[1] / 100.0 # H:W:0.100 -> 0.1
22          hsv_norm[:, :, 1] = hsv_norm[:, :, 1] / 255.0 # saturation: Value: 0-255 -> 0.1
23          lab_norm = lab.astype(np.float32) / 255.0
24          gray_norm = gray.astype(np.float32) / 255.0
25
26          return {
27              "rgb": rgb_norm,
28              "hsv": hsv_norm,
29              "lab": lab_norm,
30              "gray": gray_norm,
31              "h": h,
32              "w": w
33          }
34
35      def extract_color_features(self, processed: Dict[str, np.ndarray]) -> Dict[str, float]:
36
37          """
38              Extract color features from image.
39
40              Args:
41                  processed: Dictionary of processed images
42
43              Returns:
44                  Dictionary of color features
45
46          """
47          features = []
48
49          # RGB channel statistics
50          for i, channel in enumerate(['R', 'G', 'B']):
51              channel_data = rgb[:, :, i].flatten()
52              features[f"(channel){channel}_mean"] = float(np.mean(channel_data))
53              features[f"(channel){channel}_std"] = float(np.std(channel_data))
54              features[f"(channel){channel}_kurtosis"] = float(np.kurtosis(channel_data))
55
56          # HSV channel statistics
57          for i, channel in enumerate(['H', 'S', 'V']):
58              channel_data = hsv[:, :, i].flatten()
59              features[f"(channel){channel}_mean"] = float(np.mean(channel_data))
60              features[f"(channel){channel}_std"] = float(np.std(channel_data))
61              features[f"(channel){channel}_kurtosis"] = float(np.kurtosis(channel_data))
62
63          # LAB channel statistics
64          for i, channel in enumerate(['L', 'A', 'B']):
65              channel_data = lab[:, :, i].flatten()
66              features[f"(channel){channel}_mean"] = float(np.mean(channel_data))
67              features[f"(channel){channel}_std"] = float(np.std(channel_data))
68
69          return features
70
71  
```

FIG 8.7 IMAGE_PROCESSOR.py

```

station.py   data_loader.py   freshness_analyzer.py   heatmap_generator.py   model_loader.py   app.py   fix_png_profiles.py
137     def analyzing_image(image_input):
138         with st.spinner("Analyzing image... Please wait."):
139             try:
140                 if not img_bytes or len(img_bytes) == 0:
141                     return
142
143                 # Try loading with OpenCV
144                 image = load_image_from_bytes(img_bytes)
145
146                 # If openCV fails, try using PIL to convert
147                 if image is None or image.size == 0:
148                     try:
149                         # Try loading with PIL first, then convert to OpenCV format
150                         pil_image = Image.open(io.BytesIO(img_bytes))
151                         img_array = np.array(pil_image)
152
153                         # Convert RGB to BGR for OpenCV
154                         if len(img_array.shape) == 3:
155                             if img_array.shape[2] == 4: # RGBA
156                                 img_array = cv2.cvtColor(img_array, cv2.COLOR_RGBA2BGR)
157                             else:
158                                 img_array = cv2.cvtColor(img_array, cv2.COLOR_RGB2BGR)
159                         image = img_array
160
161                     except Exception as e:
162                         st.error(f"Error loading image: {str(e)}. Please ensure the file is a valid image (JPG, PNG).")
163
164                 if image is None or image.size == 0 or len(image.shape) < 2:
165                     st.error("Error loading image. Please ensure the file is a valid image (JPG, PNG).")
166
167                 # Check image dimensions
168                 if image.shape[0] < 96 or image.shape[1] < 96:
169                     st.warning("Image is very small. Results may be less accurate. Please use a higher resolution image.")
170
171                 # Analyze
172                 result = st.session_state.analyzer.analyze(image)
173
174                 if not result:
175                     st.error("Analysis failed. Please try again.")
176
177                 # Generate heatmap
178                 try:
179                     heatmap, overlay = st.session_state.heatmap_gen.create_visualization(
180                         image, result['features'], result['freshness_percentage']
181                     )
182                     result['heatmap_overlay'] = overlay
183                     result['heatmap'] = heatmap
184                 except Exception as e:
185                     st.warning(f"Could not generate heatmap: {str(e)}")
186                     result['heatmap_overlay'] = image
187                     result['heatmap'] = None
188
189                 result['original_image'] = image
190
191                 # Store result
192                 st.session_state.current_result = result

```

FIG 8.8 APP.py

```

station.py   data_loader.py   fix_png_profiles.py   README.md   train_combined.py   train_model.py   data_general.py
111     def main():
112
113         # Create generators and return freshness labels
114         class FreshnessGenerator:
115             def __init__(self, base_generator):
116                 self.base_gen = base_generator
117             def __len__(self):
118                 return len(self.base_gen)
119             def __getitem__(self, idx):
120                 return self.base_gen.get_freshness_batch(idx)
121             def on_epoch_end(self):
122                 self.base_gen.on_epoch_end()
123
124         train_fresh_gen = FreshnessGenerator(train_gen)
125         val_fresh_gen = FreshnessGenerator(val_gen)
126         test_fresh_gen = FreshnessGenerator(test_gen) if test_gen else None
127
128         freshness_model = ModelTrainer.train_freshness_regressor(
129             img_size=IMG_SIZE,
130             batch_size=BATCH_SIZE,
131             epochs=POCKS,
132             train_generator=train_fresh_gen,
133             val_generator=val_fresh_gen,
134             steps_per_epoch=len(train_fresh_gen),
135             validation_steps=len(val_fresh_gen)
136         )
137
138         # Evaluate freshness regressor on validation and test sets
139         print("Evaluating freshness regressor on validation set...")
140         val_loss, val_mae = freshness_model.evaluate(val_fresh_gen, verbose=1)
141         print(f"\nFreshness regressor Validation MAE: {val_mae * 100:.2f}%")
142
143         if test_fresh_gen is not None:
144             print("Evaluating freshness regressor on test set...")
145             test_loss, test_mae = freshness_model.evaluate(test_fresh_gen, verbose=1)
146             print(f"\nFreshness regressor TEST MAE: {test_mae * 100:.2f}%")
147
148         # Step 5: Test predictions on TEST set
149         print("... Validation Set Predictions ...")
150         num_val_samples = min(6, len(val_paths))
151         val_sample_indices = np.random.choice(len(val_paths), num_val_samples, replace=False)
152
153         for idx, i in enumerate(val_sample_indices):
154             # Load image from path
155             img, val_path = val_paths[i]
156             image = cv2.imread(img_path)
157
158             if image is None:
159                 continue
160             image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
161             image = cv2.resize(image, (IMG_SIZE, IMG_SIZE))
162             image = image.astype(np.float32) / 255.0
163
164             true_item = item_classes[item_val[i]]
165             true_freshness = y_fresh_val[i]

```

FIG 8.9 TRAIN_MODEL.py

CHAPTER 9

PROJECT LIMITATIONS & FUTURE SCOPE

9.1 Project Limitations

Despite the significant capabilities offered by the Freshness Detection System, there are certain limitations that currently restrict its performance and usability:

1. Dataset Limitations

- The model relies on a pre-collected dataset of fruits and vegetables.
- Limited diversity in lighting conditions, angles, surface textures, and decay types may affect accuracy in real-world environments.
- Rare or region-specific fruits and vegetables may not be detected properly.

2. Dependence on Visual Features Only

- The current system evaluates freshness purely based on visual cues such as colour changes, texture, mold formation, etc.
- Internal spoilage (e.g., over-ripeness or internal browning) cannot be detected through camera images alone.

3. Environmental Variability

- Excessive shadows, reflections, or poor camera quality can reduce prediction accuracy.
- Results may vary depending on device camera resolution and scanning distance.

4. Generalized Shelf-Life Estimation

- The prediction of remaining freshness duration is based on average decay

patterns.

- It does not yet consider storage history, humidity, temperature, or handling conditions of the produce.

5. Processing Time and Computational Load

- Real-time model inference may require computational resources that low-end smartphones may struggle to support.
- On-device processing limits large and deeper model architectures.

6. No Chemical/Smell-Based Analysis

- The absence of biochemical evaluation (e.g., ethylene gas measurement) limits deeper understanding of spoilage.

7. User-Dependent Image Scanning

- Incorrect scanning techniques by users can cause inaccurate predictions.
- Small defects might be missed if not captured clearly.

8. Limited Explainability

- Although heatmaps provide focus regions, the exact reasoning behind model decisions can still appear unclear to non-technical users.

9.2 Future Scope of the Project

To enhance system performance and expand its usability in real-life applications, the following improvements can be introduced:

1. Enhanced and Expanded Dataset
 - Include a larger variety of fruits and vegetables under multiple spoilage stages.
 - Add global datasets for better adaptability in different geographic regions.
2. Multi-Sensor Integration
 - Incorporate technologies such as:
 - Gas Sensors (detect ethylene & ammonia)
 - Temperature and humidity sensors
 - Near-Infrared (NIR) imaging
 - This will improve internal freshness detection beyond the visible surface.
3. AI-Driven Shelf-Life Prediction
 - Introduce real-time tracking of storage conditions using IoT.
 - Individualized shelf-life estimation based on:
 - Storage temperature
 - Exposure to air
 - Handling frequency
4. Cross-Platform Deployment
 - Develop lightweight model versions for smooth integration in:
 - Android/iOS apps
 - Smart refrigerators

- Retail PoS systems in supermarkets

5. Integration with Smart Supply Chain

- Helps farmers, vendors, and suppliers reduce food waste by monitoring freshness during transportation and storage.

6. Personalized Health Suggestions

- Provide health-based guidance like:
 - nutritional value according to age/diet
 - consumption recommendations for patients (diabetes, obesity, etc.)

7. Voice and Multi-Language Support

- Better accessibility for users in rural areas and non-English speaking regions.

8. Real-Time Analytics & Cloud Support

- Centralized tracking of freshness trends.
- Enable batch scanning for commercial storage units.

CHAPTER 10

REFERENCES

- [1] Y. Yuan, “Vegetable and fruit freshness detection based on deep features,” Journal of Food Quality & Safety, 2024. ScienceDirect
- [2] Y. Zhang, “Fruit freshness detection based on multi-task convolutional neural network,” 2024. ScienceDirect +1
- [3] N. Sultana et al., “An extensive dataset for successful recognition of fresh and rotten fruits,” 2022. ScienceDirect +1
- [4] Y. Shu, J. Zhang, Y. Wang, Y. Wei, “Fruit Freshness Classification and Detection Based on the ResNet-101 Network and Non-Local Attention Mechanism,” Foods, vol. 14, 2025. MDPI+1
- [5] U. Amin et al., “Automatic Fruits Freshness Classification Using CNN and Transfer Learning,” Applied Sciences, 2023. MDPI
- [6] “Fruits and Vegetables Freshness Categorization Using Deep Learning,” CMC: Computers, Materials & Continua, 2022 (or 2023 depending on publication). ResearchGate+1
- [7] A. Bhargava and others, “Fruits and vegetables quality evaluation using computer vision,” 2018. ScienceDirect
- [8] “Fresh and Rotten Fruits Classification Using CNN” (apple, banana, orange) — Research article on fruit freshness classification using CNN. ResearchGate

- [9] M. Iqbal, “Canned Apple Fruit Freshness Detection Using Hybrid Image Processing and Machine Learning,” 2025. Wiley Online Library
- [10] M. Rizzo et al., “Fruit ripeness classification: A survey,” 2023. ScienceDirect
- [11] C. Wang et al., “Application of Convolutional Neural Network-Based Methods for Fruit Production Process Monitoring,” Frontiers in Plant Science, 2022. Frontiers
- [12] “Fruit Freshness Detection Using IoT and Deep Learning” — review paper discussing ML/Deep-Learning + IoT for freshness detection, 2024. IJCRT
- [13] Dataset “Fruits-360” — a publicly available image dataset of fruits, vegetables, nuts, and seeds (often used in freshness / classification tasks). kaggle.com
- [14] “Spoiled and Fresh Fruit Inspection Dataset” (includes multiple fruit types, used for freshness/spoilage classification) — research dataset source. Mendeley Data+1