# Project Title:

**Smart Sorting: Transfer Learning for Identifying Rotten Fruits and Vegetables Team Members:**

Team Id: LTVIP2025TMID34781

Team leader: Bavireddy Lahari

Team member: Badugu David Raju

Team member: Abdul Javeed

Team member: Abdul Gafoor

**Phase 1: Brainstorming & Ideation Objective:**

* Identify the problem statement.
* Define the purpose and impact of the project.

# Problem Statement:

Sorting fruits and vegetables for quality assurance is a time-consuming and labor-intensive process. Manual sorting often results in human errors and misjudgment, leading to waste, health risks, and decreased profitability in the supply chain. There is a growing need for an automated solution that can distinguish between healthy and rotten produce.

# Proposed Solution:

Develop a machine learning-powered web application that can classify fruits and vegetables as "Healthy" or "Rotten" using a pre-trained convolutional neural network. The application accepts an image of produce, processes it, and returns a classification.

Smart Sorting is an innovative project focused on enhancing the precision and efficiency of detecting rotten fruits and vegetables using cutting-edge transfer learning techniques. By leveraging pre-trained deep learning models and adapting them to specific datasets of fruits

and vegetables, this project aims to revolutionize the process of sorting and quality control in the agricultural and food industry.

# Target Users:

* Farmers
* Wholesale and retail vendors
* Supermarkets
* Food processing companies
* Supply chain handlers



# Expected Outcome:

* Automated quality sorting of produce.
* Reduced food wastage.
* Increased productivity and accuracy.
* Useful insights into produce health conditions.

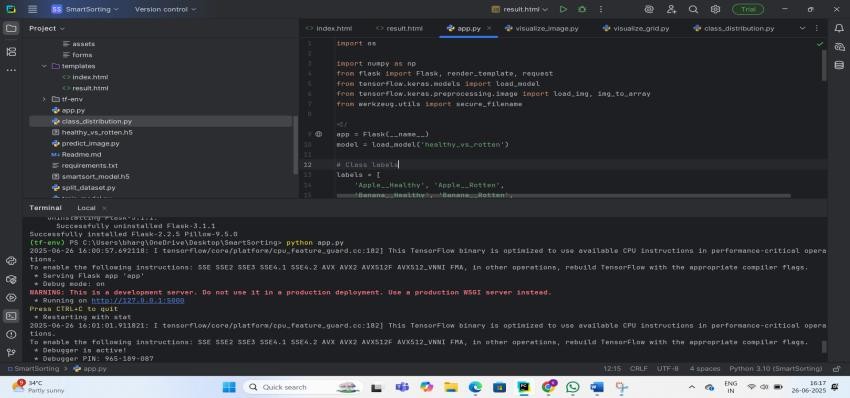


# Phase 2: Requirement Analysis Objective:

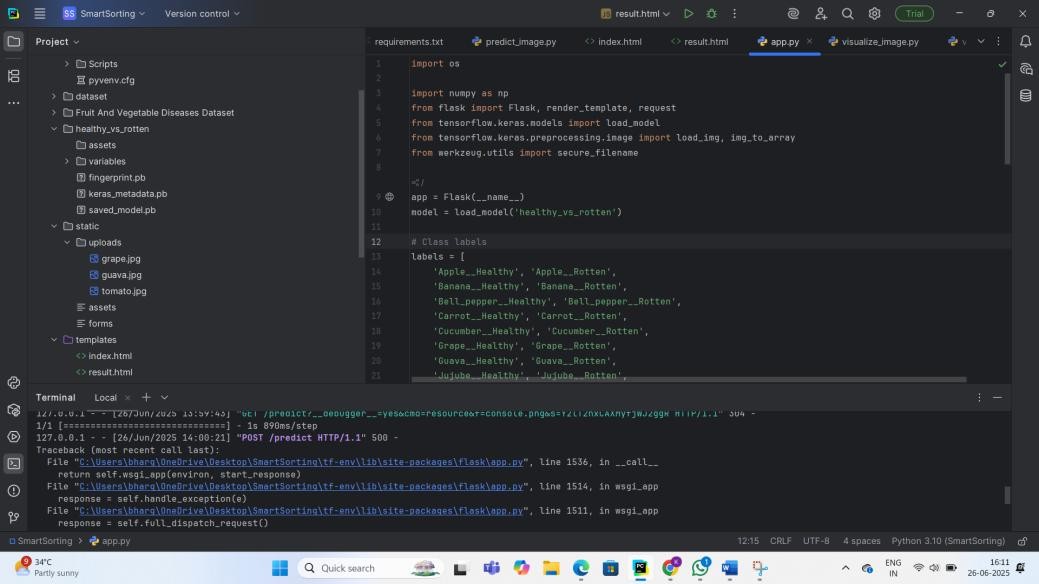
Define technical and functional requirements.

# Technical Requirements:

* **Languages & Tools:** Python, HTML, CSS
* **Frameworks:** Flask, TensorFlow/Keras
* **Libraries:** NumPy, Pillow, OpenCV (optional)
* **Model:** Pre-trained VGG16 with custom classification layers
* **Environment:** PyCharm



# Environment of PyCharm:

****

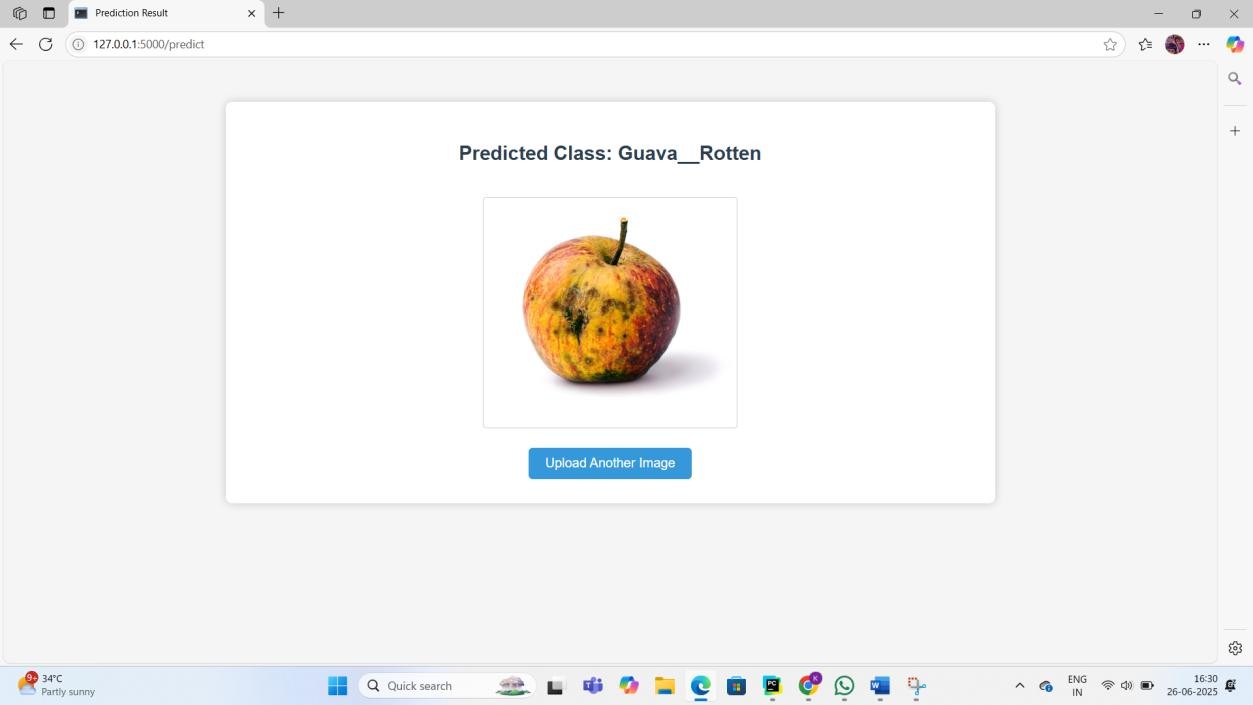
**Functional Requirements:**

* Upload interface for users to submit images.
* Backend model integration to process and predict.
* Display of prediction results with image preview.
* UI flow for easy navigation and repeated uploads.

# Constraints & Challenges:

* Limited dataset per class may affect accuracy.
* Some fruits and vegetables look similar in colour/shape (e.g., guava and apple).
* File upload issues (e.g., large image sizes or unsupported formats).
* Deployment environment setup.

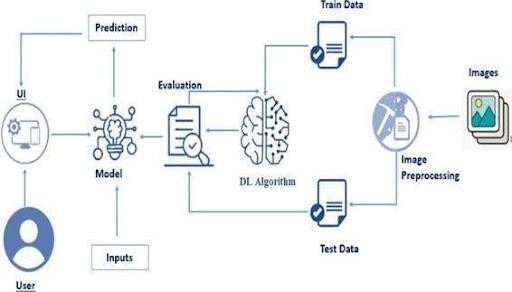
**Constraint:** In the below you can observe that it gave wrong prediction because, apple and guava shapes are almost similar.



# Phase 3: Project Design Objective:

Create the architecture and user flow.

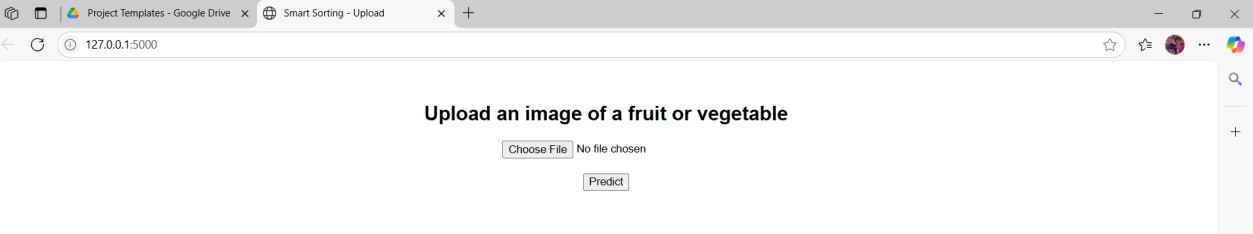
**System Architecture Diagram:**

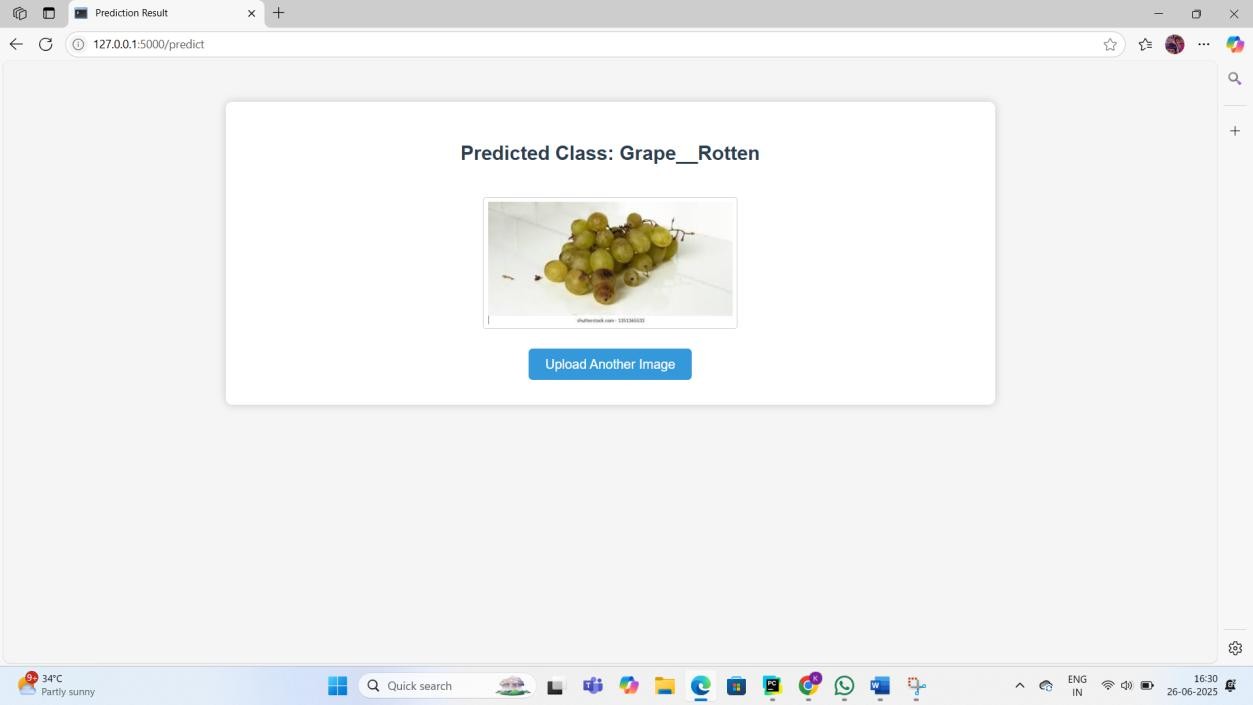
****

**User** → **Web Interface** → **Flask Backend** → **Preprocessing** → **Trained Model** →

**Prediction Output** → **Result Page User Flow:**

1. User visits home page.
2. Clicks upload and selects an image.
3. Submits form and waits for prediction.
4. Result page shows classification and image preview.
5. User can upload another image.





# UI/UX Considerations:

* + Clean layout with banner and navigation bar.
  + Upload button clearly visible.
  + Image displayed on result page.
  + Prediction text styled and highlighted.

# Phase 4: Project Planning (Agile Methodologies) Objective:

Break down tasks using Agile methodologies.

# Sprint Planning:

* + **Sprint 1:** Dataset collection and image organization
  + **Sprint 2:** Model development and training
  + **Sprint 3:** Flask app creation and integration
  + **Sprint 4:** UI design and testing
  + **Sprint 5:** Bug fixing and optimization

# Timeline & Milestones:

* + Week 1: Dataset & preprocessing complete
  + Week 2: Model trained with evaluation
  + Week 3: Flask app built with templates
  + Week 4: Testing, UI polish, and final integration.

# Phase 5: Project Development Objective:

Code the project and integrate components.

# Technology Stack Used:

* Python 3.10+
* Flask microframework
* TensorFlow/Keras
* HTML, CSS
* Bootstrap (for styling)
* Pre-trained VGG16 model.

# Development Process:

1. Dataset Collection: Labeled images of healthy and rotten produce.
2. Data Preprocessing: Resizing, normalization, train/test split.
3. Model Training: Transfer learning with VGG16, softmax output.
4. Model Evaluation: Accuracy, loss graphs, confusion matrix.
5. Flask Integration: Routes for index and predict.
6. UI Design: HTML templates for upload and results.
7. Testing: Upload scenarios, wrong formats, edge cases.

# Challenges & Fixes:

* + **Issue:** Upload folder error during repeated runs

**Fix:** Used os.makedirs (..., exist\_ok=True)

* + **Issue:** Wrong prediction for similar fruits

**Fix:** Increased training data and improved augmentation

* + **Issue:** File size errors

**Fix:** Limited upload file size and added file type filter.

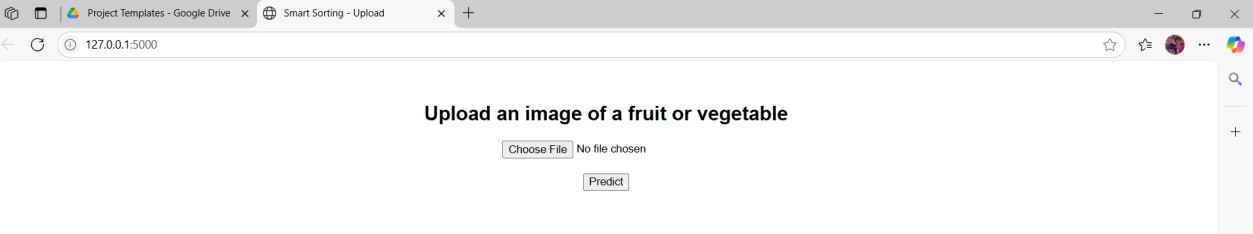
# Phase 6: Functional & Performance Testing Objective:

Ensure the project works as expected.

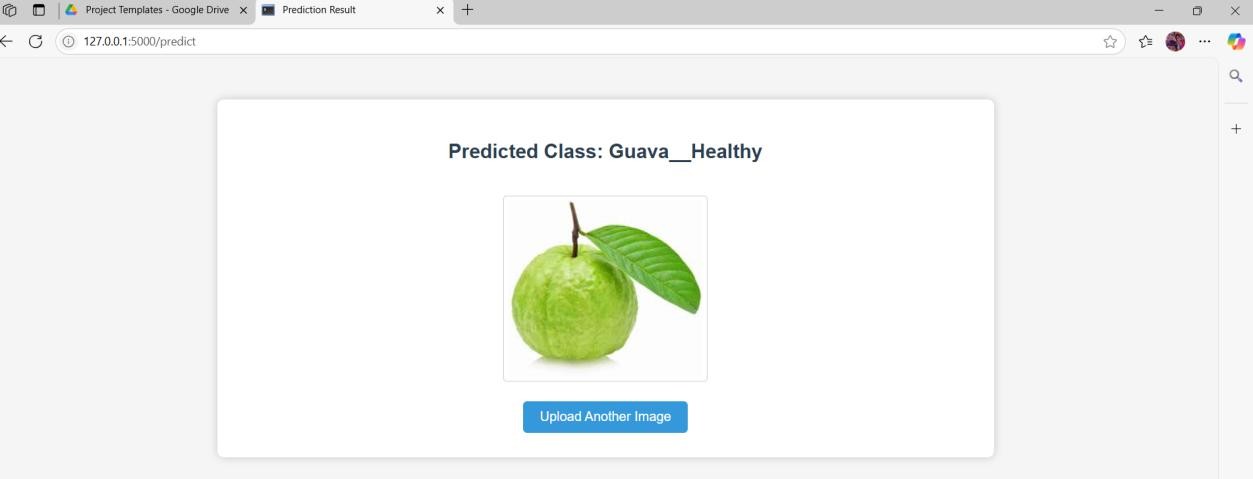
**Test Cases Executed:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SNO | Test Scenario | Input | Expected\_Output | Result |
| 1. | Upload healthy apple | Apple.jpg | Apple\_Healthy | pass |
| 2. | Upload rotten banana | Banana.jpg | Banana\_Rotten | pass |
| 3. | Upload Invalid Image | Random.txt | Error Message | pass |
| 4. | Upload healthy guava | Guava.jpg | Guava\_Healthy | pass |

**Results of app.py: Image Upload Form:**



**Prediction Result Page:**

****

**Bug Fixes & Improvements:**

* + - Improved model weights for better accuracy
    - Cleaned up frontend with custom styling
    - Added auto-clean script for uploaded images

# Final Validation:

* + - Verified accuracy against unseen test set.
    - Tested end-to-end flow.
    - Verified UI responsiveness.

# Deployment (if applicable):

* + - Hosted on local server.

\* Running on [http://127.0.0.1:5000](http://127.0.0.1:5000/)

* + - Future deployment on Render or Heroku.

# Additional Sections Dataset Overview:

* + - 28 classes: 14 fruits/vegetables × 2 (Healthy/Rotten)
    - Balanced split for training/testing
    - ~200-300 images per class

# Utility Scripts:

* + - **split\_dataset.py:** Divides data into train/test folders
    - **visualize\_image.py:** Shows single sample from each class
    - **class\_distribution.py:** Visualizes count of each class
    - **visualize\_grid.py:** Plots grid of random images

# Future Enhancements:

* + - Expand to include more fruits and vegetables.
    - Mobile app version.
    - Real-time camera input for live sorting.
    - Deploy on cloud with user authentication and tracking.

# Project Structure:

SmartSorting/

│

├── app.py # Main Flask app

├── healthy\_vs\_rotten.h5 # Trained Keras model

├── requirements.txt # Required Python packages

├── README.txt # Project documentation

│

├── static/ # Static files

│ └── uploads/ # Uploaded images go here

│

├── templates/ # HTML templates

│ ├── index.html # Upload page

│ └── result.html # Prediction result page

│

├── dataset/ # (Optional) Dataset used for training

│ ├── Apple Healthy/

│ ├── Apple Rotten/

│ └── ... (other classes)

│

├── utils/ # Utility scripts (optional)

|  |  |  |
| --- | --- | --- |
| │ | ├── split\_dataset.py | # Script to split data into train/test |
| │ | ├── visualize\_image.py | # Script to visualize single image |
| │ | ├── visualize\_grid.py | # Script to display grid of images |
| │ | └── class\_distribution.py |  |