Full Stack Development with Python

Project Documentation

1. Introduction

• **Project Title:** Smart Sorting: Transfer Learning for Identifying Rotten Fruits and Vegetables. Smart Sorting is an AI-based web application that automatically classifies fruits and vegetables as **Fresh** or **Rotten** from images. The system combines a user-friendly web interface with a deep-learning model based on transfer learning (MobileNetV2). It is designed to support farmers, vendors, warehouses, and supermarkets in performing faster and more consistent quality checks, reducing food wastage and improving customer satisfaction.

• **Team Members:** List team members and their roles.

Team ID: LTVIP2026TMIDS77295
Team Size: 4
Team Leader: Venkata Durga Parvathi Veeramalla
Team member: Divija Durga Eedupuganti
Team member: Eella Harika
Team member: Eluri Sumanth

Roles: Venkata Durga Parvathi Veeramalla- Frontend design, Flask routing, UI/UX, Machine learning model, backend integration,Dataset preparation, data splitting, documentation,Testing, error handling.

2. Project Overview

• **Purpose:**

The project aims to build an AI-powered system that automatically classifies fruits and vegetables as **Fresh** or **Rotten** from images. It reduces manual inspection effort in farms, warehouses, and supermarkets, minimizes food wastage, and supports better quality control.

Goals:

- Achieve high classification accuracy using transfer learning.
- Provide an easy-to-use web interface for non-technical users.
- Enable fast, near real-time predictions from uploaded images.

• **Features:**

- Web interface with **Home, About, Predict, Contact** pages.
- Image upload for fruits/vegetables and instant prediction (Fresh/Rotten) with confidence score.
- Display of model performance metrics (accuracy, classes, dataset size) in the About page.

- Error handling for missing/invalid files.
- Modular codebase for retraining and future extension to more fruit types.

3. Architecture

• **Frontend:**

- Implemented using **HTML5, CSS3, and vanilla JavaScript** with Jinja2 templating.
- **Pages**:
  - index.html – Landing page with project title, short description, and navigation bar.
  - about.html – Description of the model, transfer learning, and statistics (accuracy, classes, dataset size).
  - predict.html – Upload form and image preview.
  - result.html – Shows prediction output.
  - contact.html – Team ID and member details.
- CSS in static/style.css provides a modern responsive design with dark/light theme toggle and mobile navigation.

• **Backend:**

- Backend built using Flask in app.py.
- Key routes:
  - / – Home page.
  - /about – About page with metrics.
  - /predict (GET/POST) – Image upload and prediction.
  - /uploads/<filename> – Serves uploaded images from local uploads/ folder.
  - /contact – Contact page.
- For predictions, the route /predict:

1. Accepts uploaded image file.
2. Saves it into uploads/ directory.
3. Calls predict_image(filepath) from predict.py.
4. Renders result.html with predicted label, confidence score, and image path.

Machine Learning Component

- Model built using TensorFlow/Keras with MobileNetV2 base (pre-trained on ImageNet).
- Only top layers are trained initially; optional fine-tuning is supported.
- Script train.py:
  - Uses ImageDataGenerator with augmentation and train/validation split.

    - Trains the model for specified epochs and saves weights as model/healthy_vs_rotten.h5.
- Script predict.py:
    - Loads the saved model.
    - Preprocesses the input image (resize 224×224, normalization).
    - Returns prediction label and confidence.

•**Database:**

- **Dataset** stored in folders:
    - dataset/all/fresh, dataset/all/rotten.
    - Split into dataset/train and dataset/test via split_dataset.py.
- **Uploaded images** stored temporarily in uploads/.
- No relational DB currently; extension point for storing prediction history.

4. Setup Instructions

4.1 Prerequisites

- Python 3.10+
- pip / virtual environment
- TensorFlow, Keras, NumPy, Matplotlib, Seaborn, scikit-learn, Flask
- (Optional) Git for cloning the repository

4.2 Installation

1. **Clone or copy** the project folder.
2. Open terminal in project directory, create and activate a virtual env (optional).
3. Install dependencies:

bash

```
pip install -r requirements.txt
```

(or manually: pip install flask tensorflow numpy pillow matplotlib seaborn scikit-learn).

4. Ensure the dataset folders and model/healthy_vs_rotten.h5 are present.
5. Run the Flask application:

bash

```
python app.py
```

6. Open browser at http://127.0.0.1:5000/.

5. Folder Structure

```
SmartInternz_SmartSorting_Final/
│
├── app.py              # Flask application
├── predict.py          # Prediction helper
├── train.py            # Model training script
├── split_dataset.py    # Train/test split
├── model/
│   └── healthy_vs_rotten.h5
├── dataset/
│   ├── all/
│   │   ├── fresh/
│   │   └── rotten/
│   ├── train/
│   └── test/
├── uploads/            # Uploaded images at runtime
├── templates/
│   ├── base.html
│   ├── index.html
│   ├── about.html
│   ├── predict.html
│   ├── result.html
│   └── contact.html
└── static/
    ├── style.css
    └── (images/icons)
```

6. Running the Application

- In project root: python app.py
- Flask starts on http://localhost:5000/.
- Prediction flow:
    - Go to **Predict** page → choose an image → click **Upload / Predict** → view result.

7. API Documentation

Route	Method	Description	Request Parameters	Response
/	GET	Returns Home page.	–	index.html
/about	GET	Shows project and model information.	–	about.html
/predict	GET	Shows image upload form.	–	predict.html
/predict	POST	Accepts image, runs model, returns result.	form-data: file (image)	result.html with label/confidence
/uploads/<filename>	GET	Serves uploaded image from uploads/.	filename in path	Image file
/contact	GET	Shows team & contact details.	–	contact.html

8. Authentication

- Current prototype **does not implement authentication**.
- All pages and prediction endpoint are open for demonstration.
- Future work: integrate simple login for admin (model retraining, dataset management).

9. User Interface

- Clean, responsive layout with **navbar**, hero section, and cards.
- Dark/light mode toggle implemented via JavaScript and CSS.
- Predict page shows:
    - File selection button.
    - Thumbnail preview of selected image.
    - Predict button and result card.

Include screenshots of:

- Home page
- About page (with metrics)
- Predict page (before and after prediction)
- Contact page

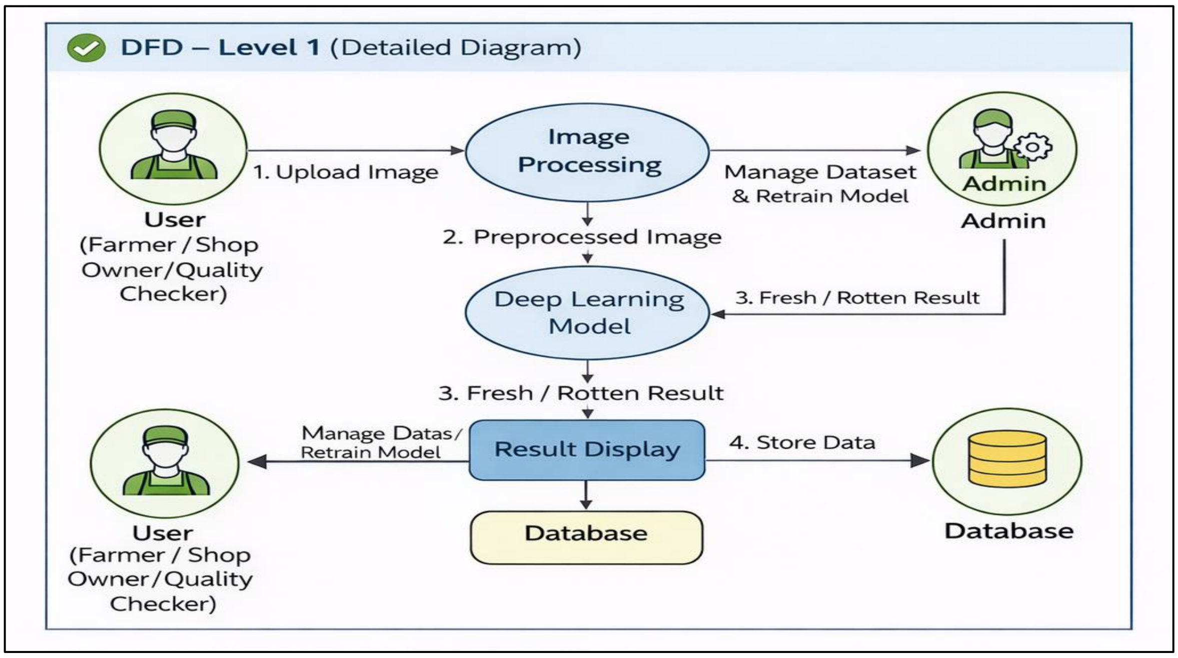
10. Testing

- **Functional testing**:
    - Verified each route (/, /about, /predict, /contact) loads correctly.
    - Tested image upload with valid/invalid files and missing file.
    - Confirmed error message “Please upload an image” appears when no file is selected.
- **Model testing**:
    - Evaluated on **test dataset** using confusion matrix and classification report.
    - Training accuracy ~94%, validation accuracy ~91% for base model; fine-tuned model up to ~95% validation accuracy.

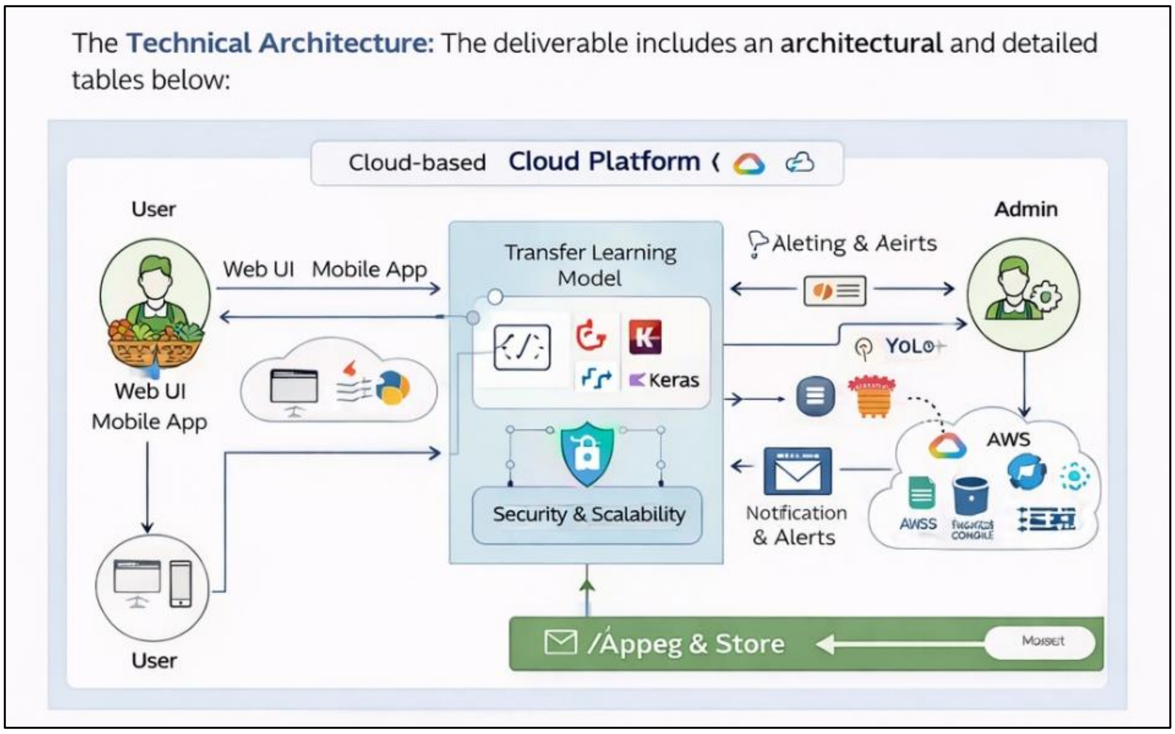
- **Cross-browser testing**:
    - Basic tests on Chrome and Edge for layout and responsiveness.

11. Screenshots or Demo

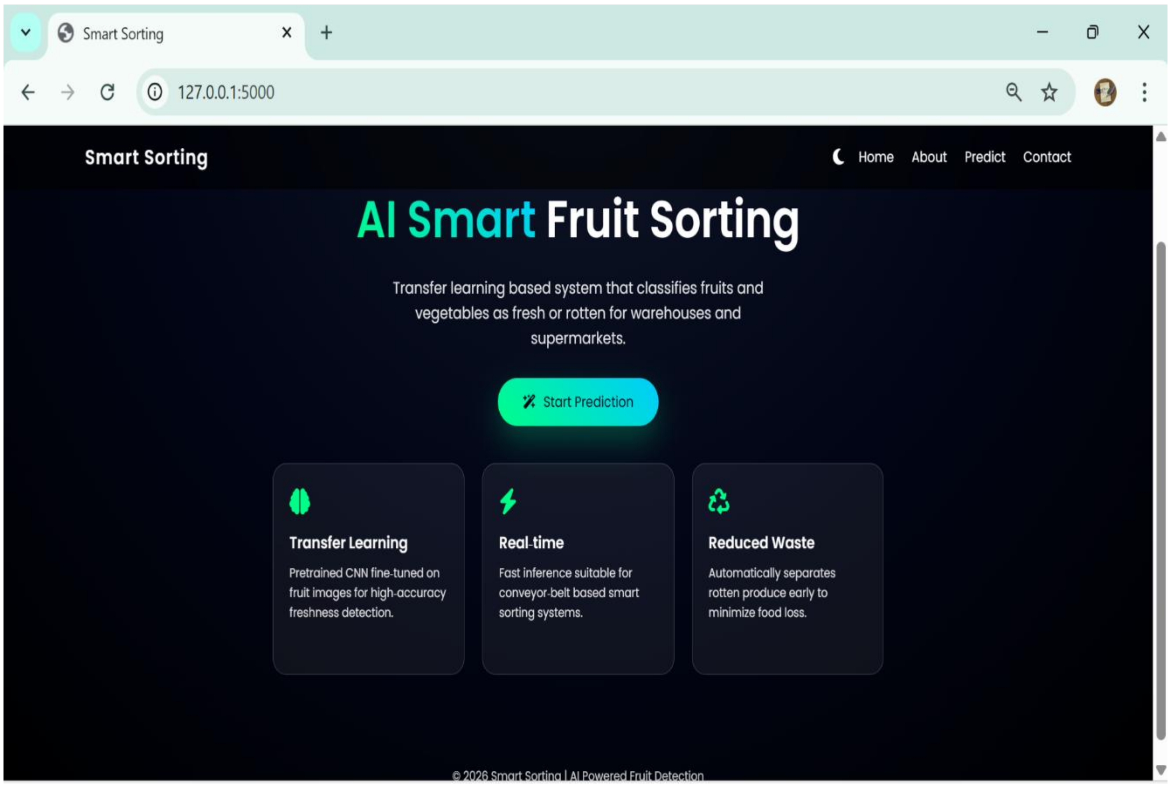
Data Flow:


DFD – Level 1 (Detailed Diagram)
User (Farmer / Shop Owner/Quality Checker)
1. Upload Image
Image Processing
Manage Dataset & Retrain Model
Admin
Admin
2. Preprocessed Image
Deep Learning Model
3. Fresh / Rotten Result
3. Fresh / Rotten Result
Manage Datas/ Retrain Model
Result Display
4. Store Data
User (Farmer / Shop Owner/Quality Checker)
Database
Database

Architecture diagram:


The Technical Architecture: The deliverable includes an architectural and detailed tables below:

UI screenshots (Home, About, Predict, Result, Contact).



Smart Sorting

AI Smart Fruit Sorting

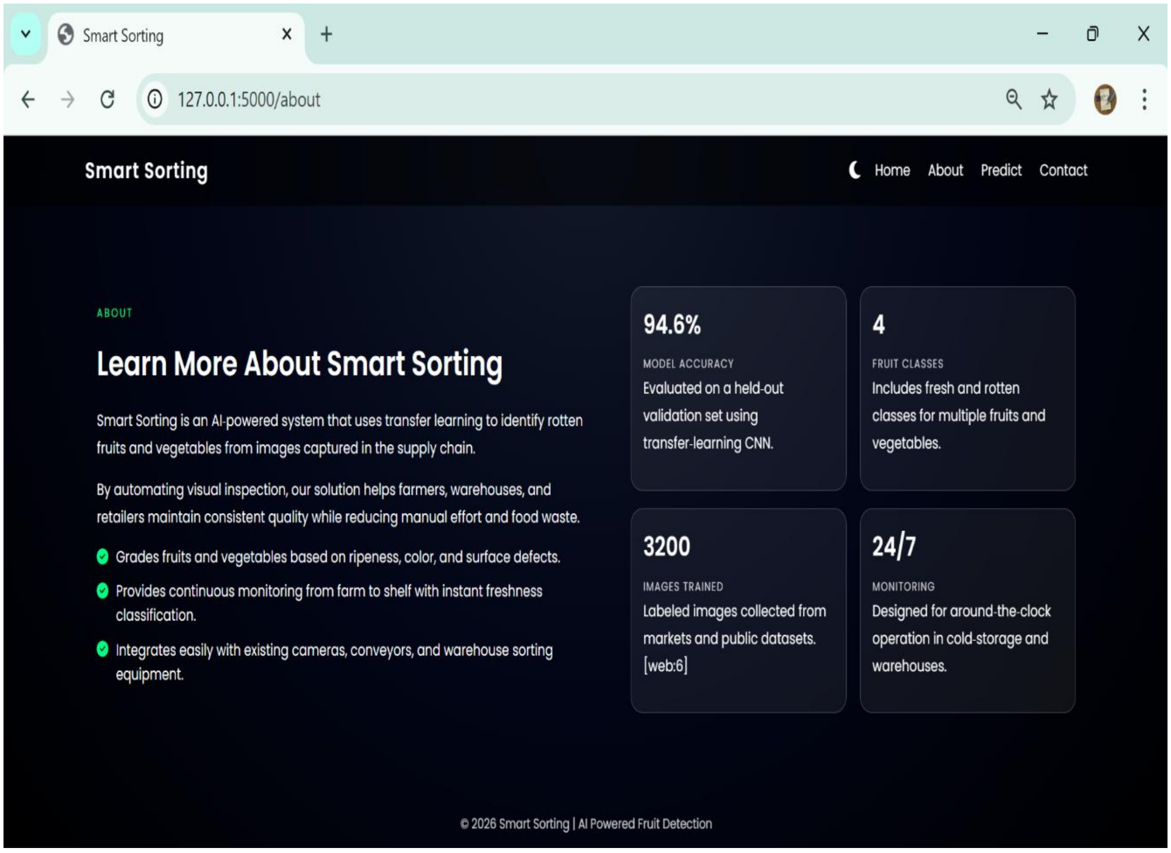
Transfer learning based system that classifies fruits and vegetables as fresh or rotten for warehouses and supermarkets.

Transfer Learning
Pretrained CNN fine-tuned on fruit images for high-accuracy freshness detection.

Real-time
Fast inference suitable for conveyor-belt based smart sorting systems.

Reduced Waste
Automatically separates rotten produce early to minimize food loss.

© 2026 Smart Sorting | AI Powered Fruit Detection



Smart Sorting

ABOUT

Learn More About Smart Sorting

Smart Sorting is an AI-powered system that uses transfer learning to identify rotten fruits and vegetables from images captured in the supply chain.

By automating visual inspection, our solution helps farmers, warehouses, and retailers maintain consistent quality while reducing manual effort and food waste.

- Grades fruits and vegetables based on ripeness, color, and surface defects.
- Provides continuous monitoring from farm to shelf with instant freshness classification.
- Integrates easily with existing cameras, conveyors, and warehouse sorting equipment.

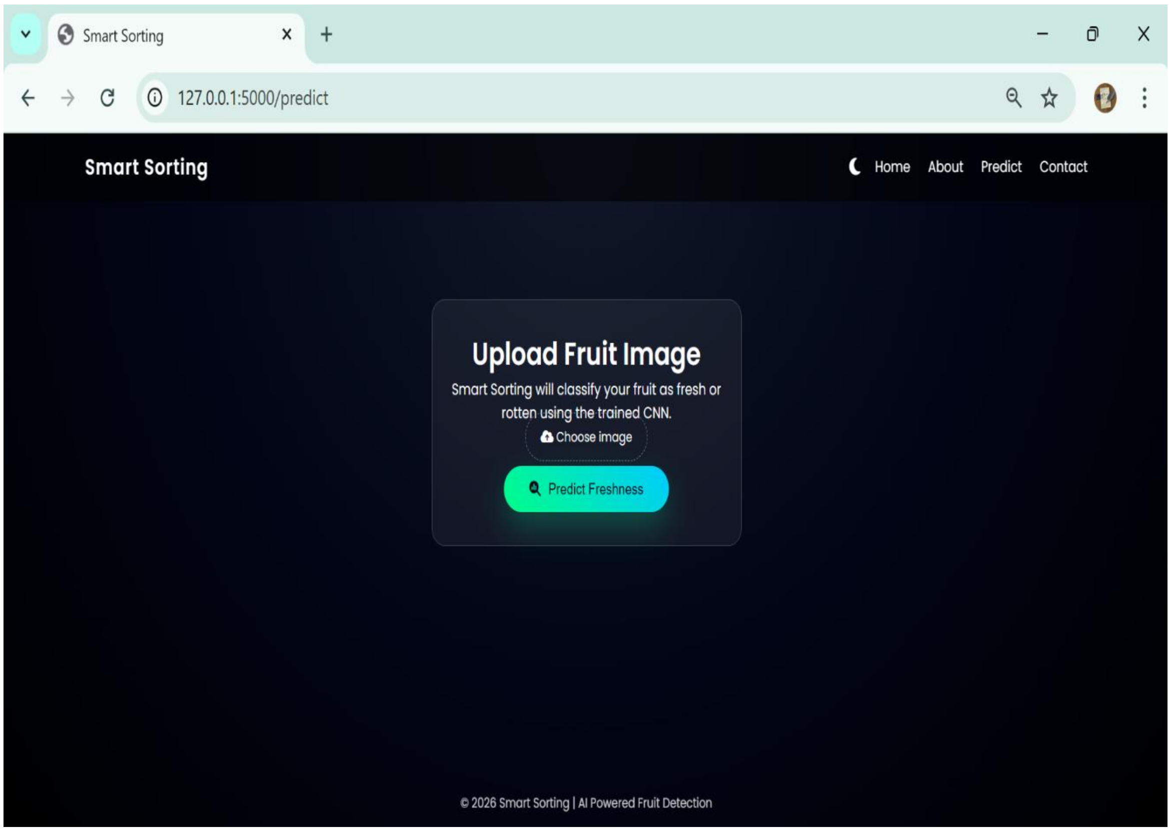
94.6%
MODEL ACCURACY
Evaluated on a held-out validation set using transfer-learning CNN.

4
FRUIT CLASSES
Includes fresh and rotten classes for multiple fruits and vegetables.

3200
IMAGES TRAINED
Labeled images collected from markets and public datasets. [web:6]

24/7
MONITORING
Designed for around-the-clock operation in cold-storage and warehouses.

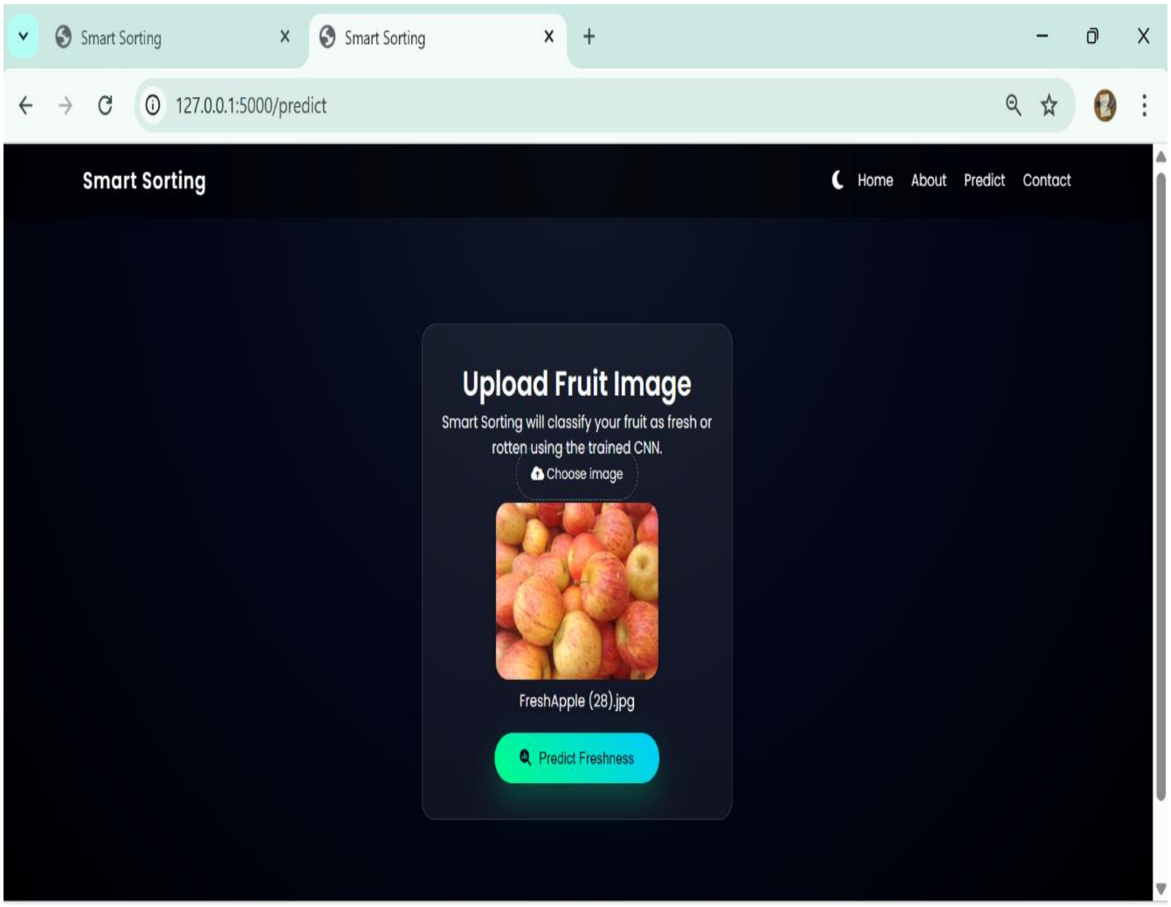
© 2026 Smart Sorting | AI Powered Fruit Detection



Upload Fruit Image

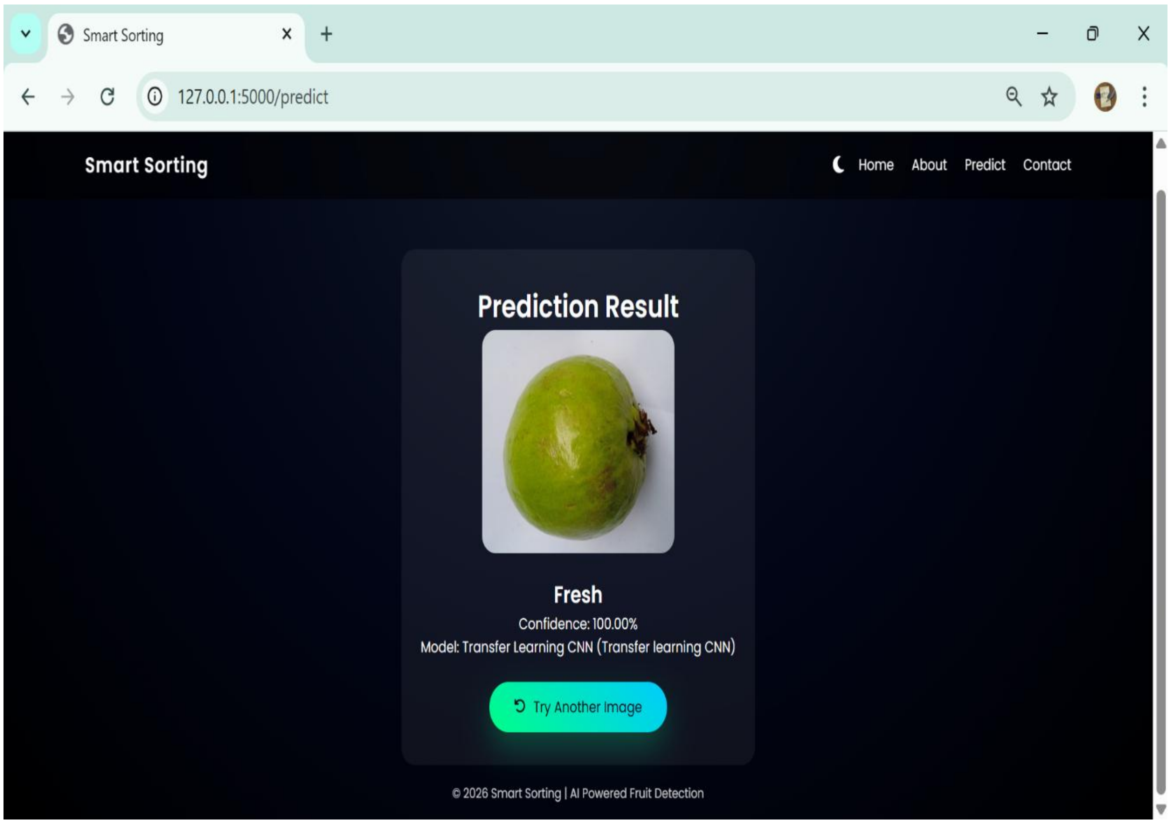
Smart Sorting will classify your fruit as fresh or rotten using the trained CNN.

© 2026 Smart Sorting | AI Powered Fruit Detection



Upload Fruit Image

Smart Sorting will classify your fruit as fresh or rotten using the trained CNN.



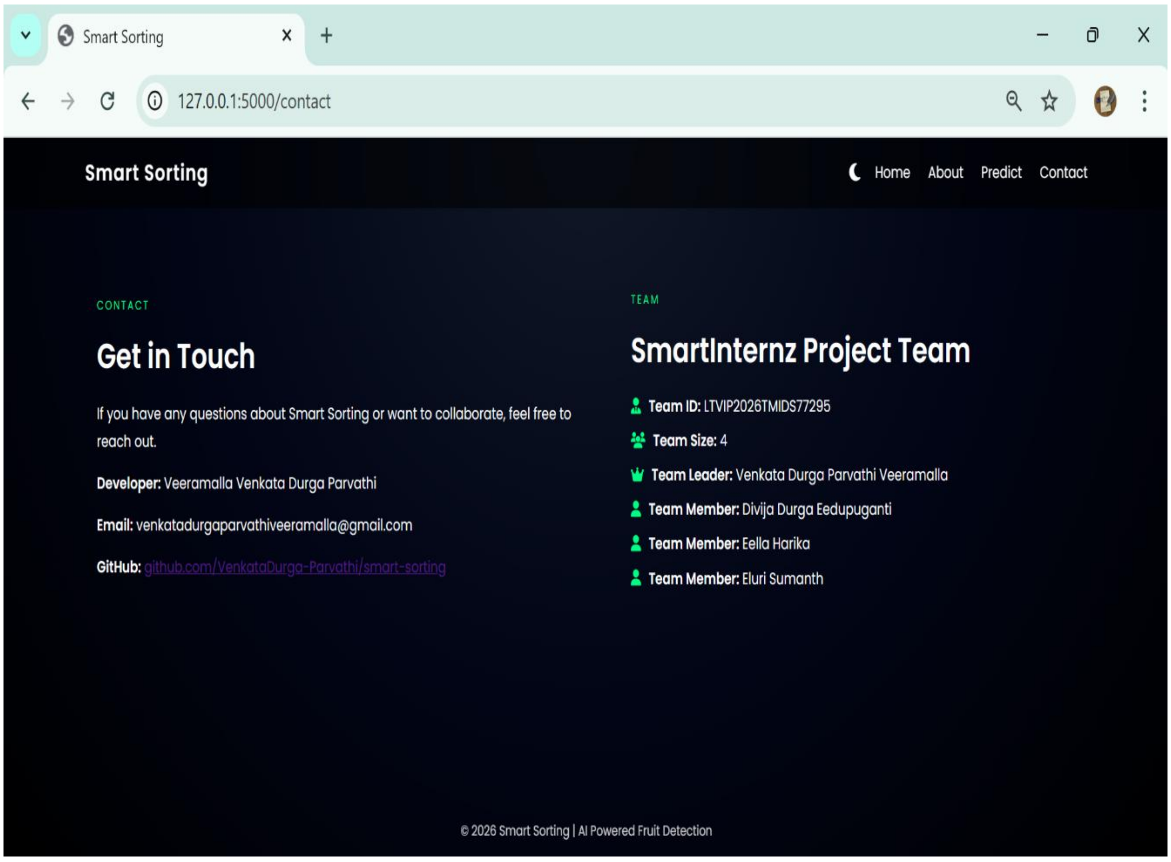
Prediction Result

Fresh

Confidence: 100.00%

Model: Transfer Learning CNN (Transfer learning CNN)

© 2026 Smart Sorting | AI Powered Fruit Detection



CONTACT

Get in Touch

If you have any questions about Smart Sorting or want to collaborate, feel free to reach out.

Developer: Veeramalla Venkata Durga Parvathi

Email: venkatadurgaparvathiveeramalla@gmail.com

GitHub: github.com/VenkataDurga-Parvathi/smart-sorting

TEAM

SmartInternz Project Team

Team ID: LTVIP2026TMIDS77295

Team Size: 4

Team Leader: Venkata Durga Parvathi Veeramalla

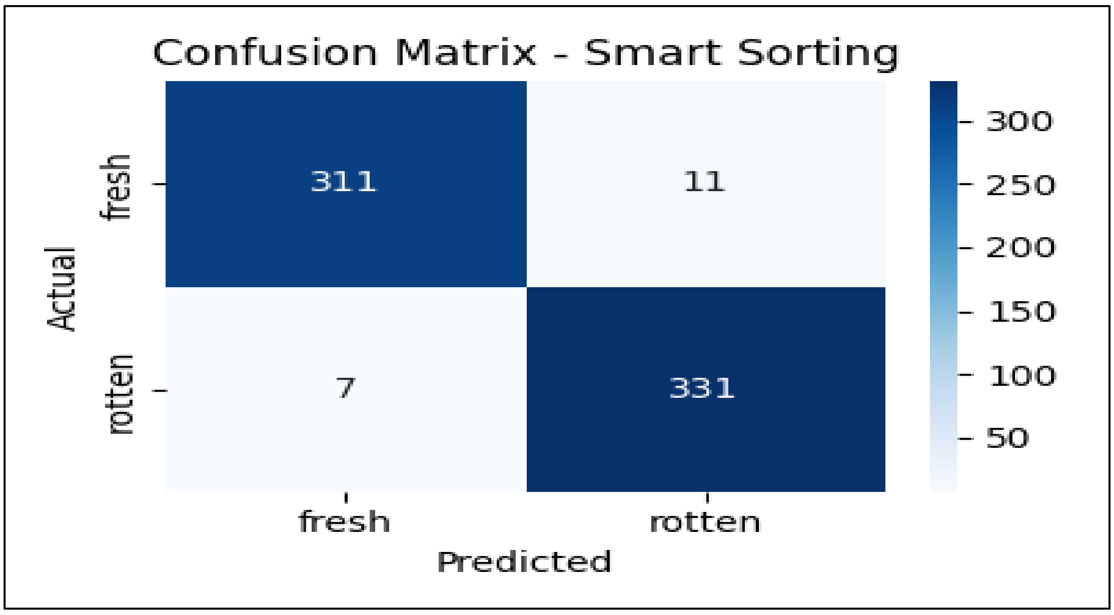
Team Member: Divija Durga Eedupuganti

Team Member: Eella Harika

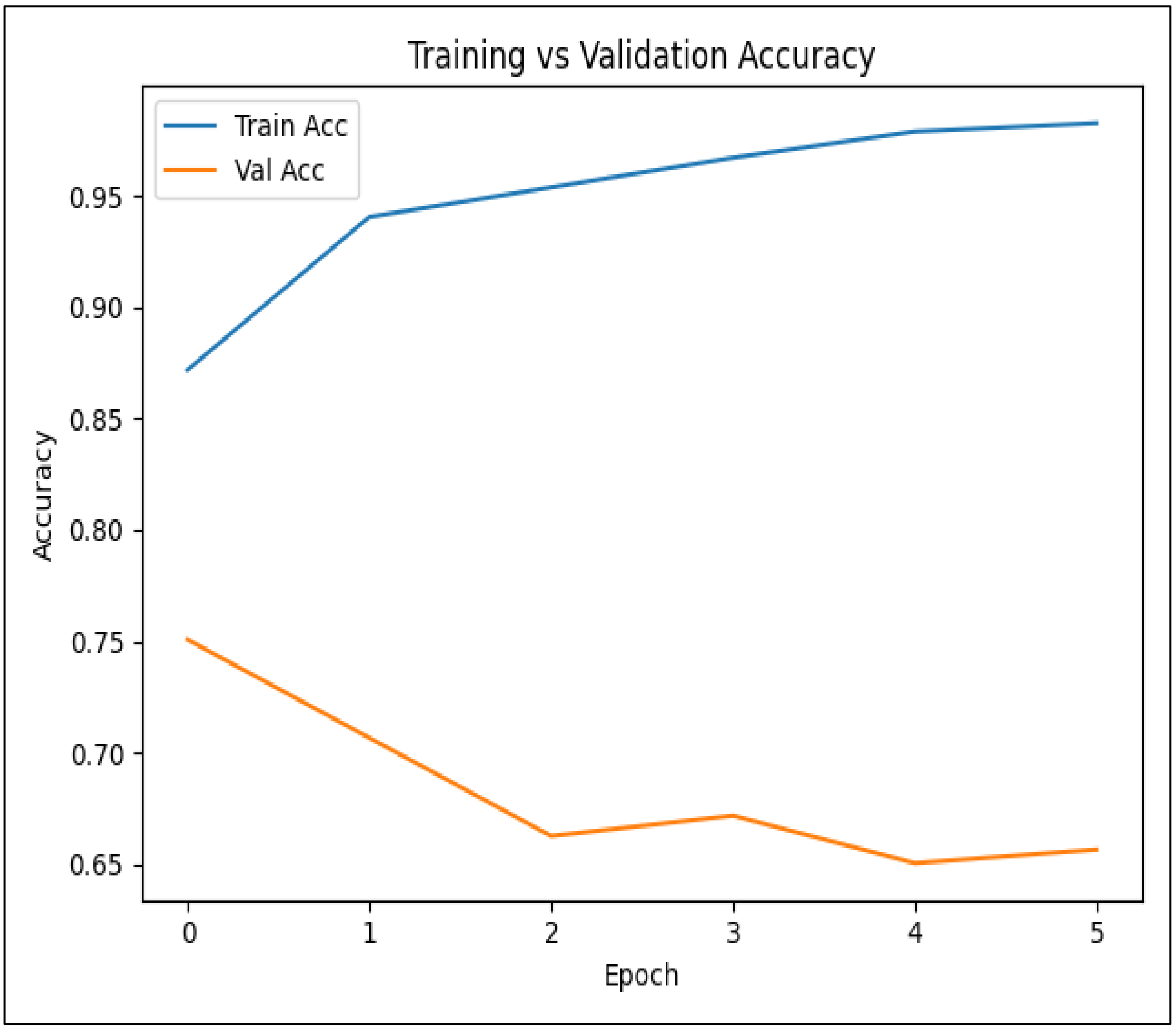
Team Member: Eluri Sumanth

© 2026 Smart Sorting | AI Powered Fruit Detection

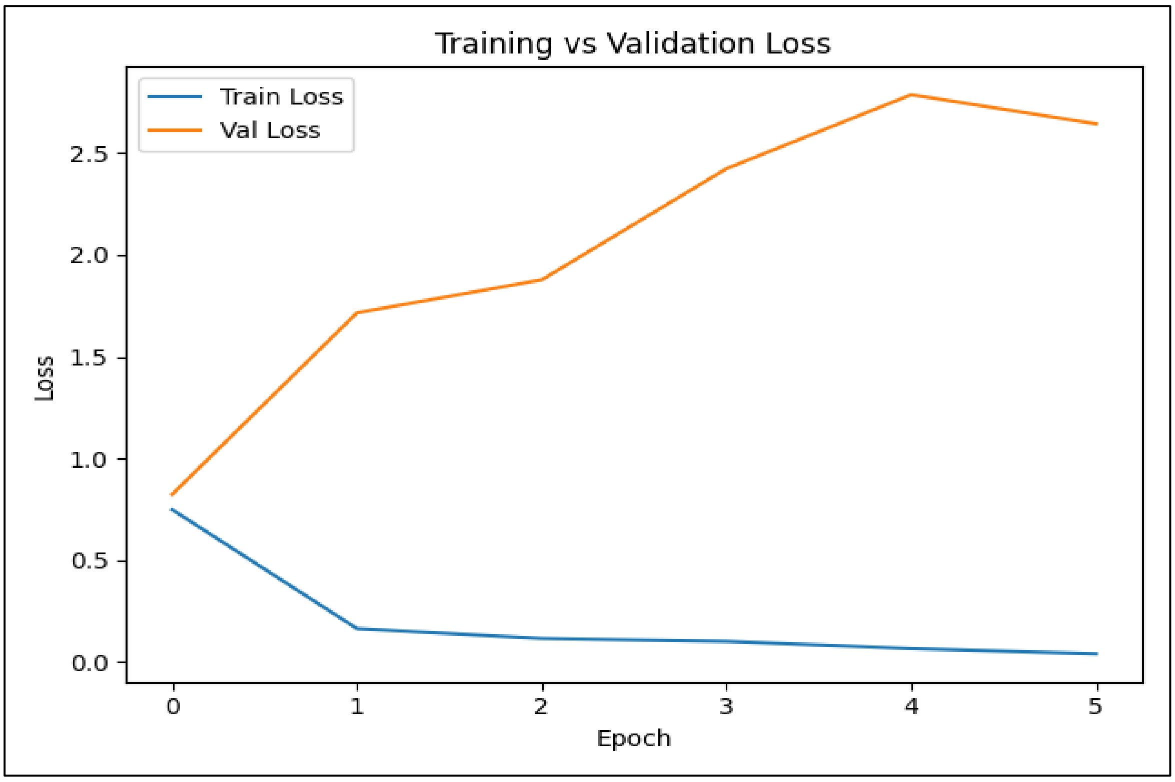
Confusion matrix and classification report:



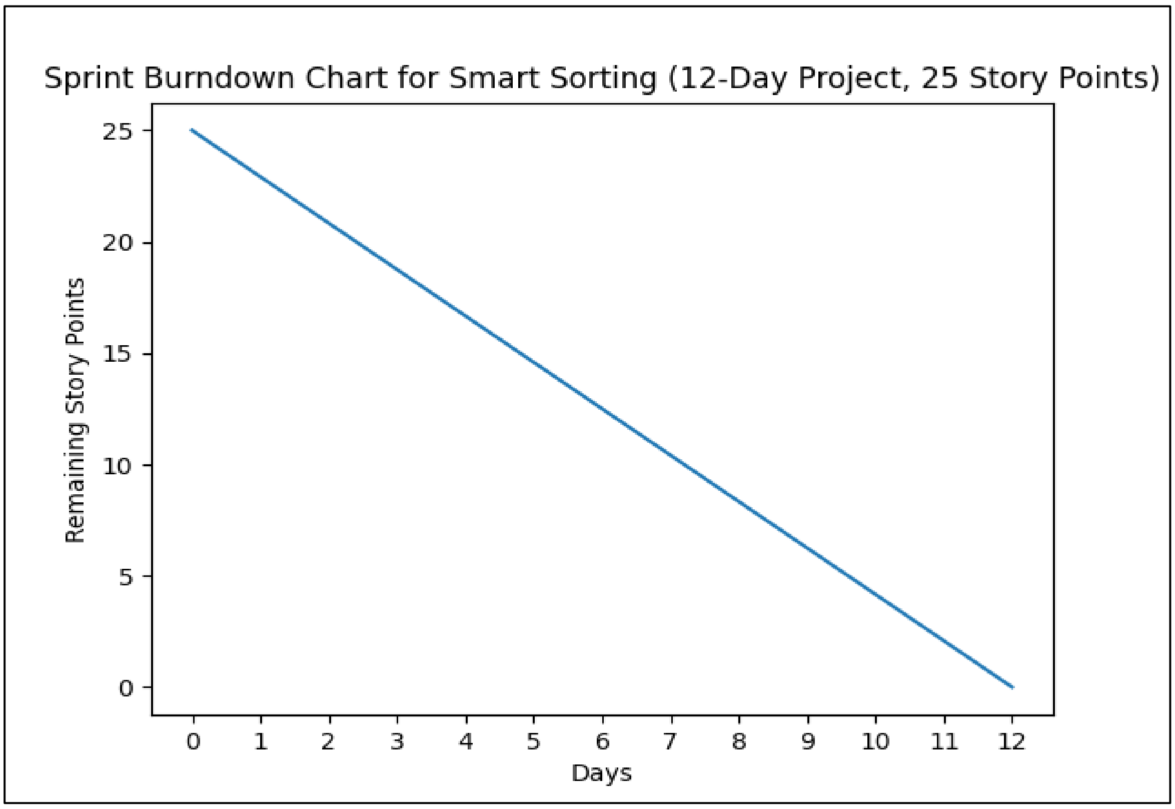
Tuned Accuracy Curve:



Tuned Loss Curve:



Burndown chart :



12. Known Issues

- No persistent database for storing prediction history yet.
- No user authentication; any user can access prediction feature.
- Current model trained mainly on specific fruits; performance may drop on unseen varieties or very noisy images.
- Application currently runs on CPU only; prediction speed may be slower on low-end systems.

13. Future Enhancements

- Add **user authentication and roles** (admin, operator).
- Implement **database (SQLite/MySQL)** for logging prediction history and user accounts.
- Extend model to multi-class classification (different fruit types and disease categories).
- Integrate with **IoT cameras or conveyor belts** for real-time sorting.
- Deploy on **cloud platform** with REST API so that external systems or mobile apps can call the model.
- Add **explainability** features (e.g., Grad-CAM heatmaps) to show which regions influenced the prediction.