Project Report of the topic
Smart Sorting: Transfer Learning for Identifying Rotten Fruits and Vegetables

1. INTRODUCTION

1.1 Project Overview:

This project aims to develop an intelligent image classification system that automatically detects and sorts **rotten vs. fresh fruits and vegetables** using **Transfer Learning**. By leveraging pre-trained deep learning models like **ResNet** or **MobileNet**, the system can be trained quickly and accurately on a smaller dataset of produce images.

Key steps include collecting and preprocessing image data, fine-tuning a CNN model, and evaluating its performance using accuracy and F1 score. The final goal is to deploy the model in a **real-time sorting setup** to reduce food waste and improve efficiency in the agricultural supply chain.

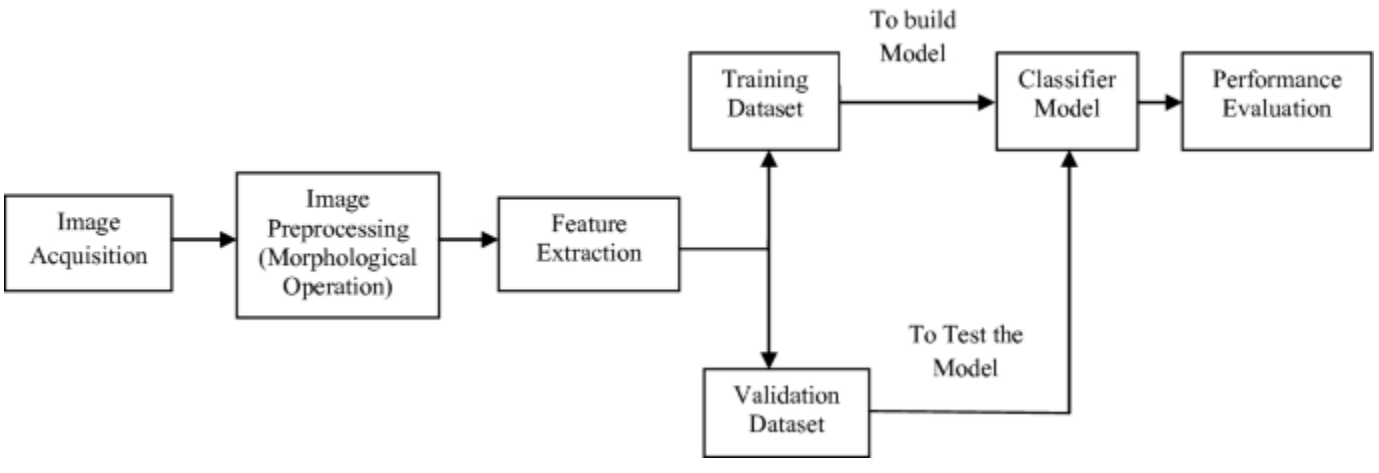
Technologies: Python, TensorFlow/Keras, OpenCV
Applications: Food industry automation, quality control, smart farming.

1.2 Purpose:

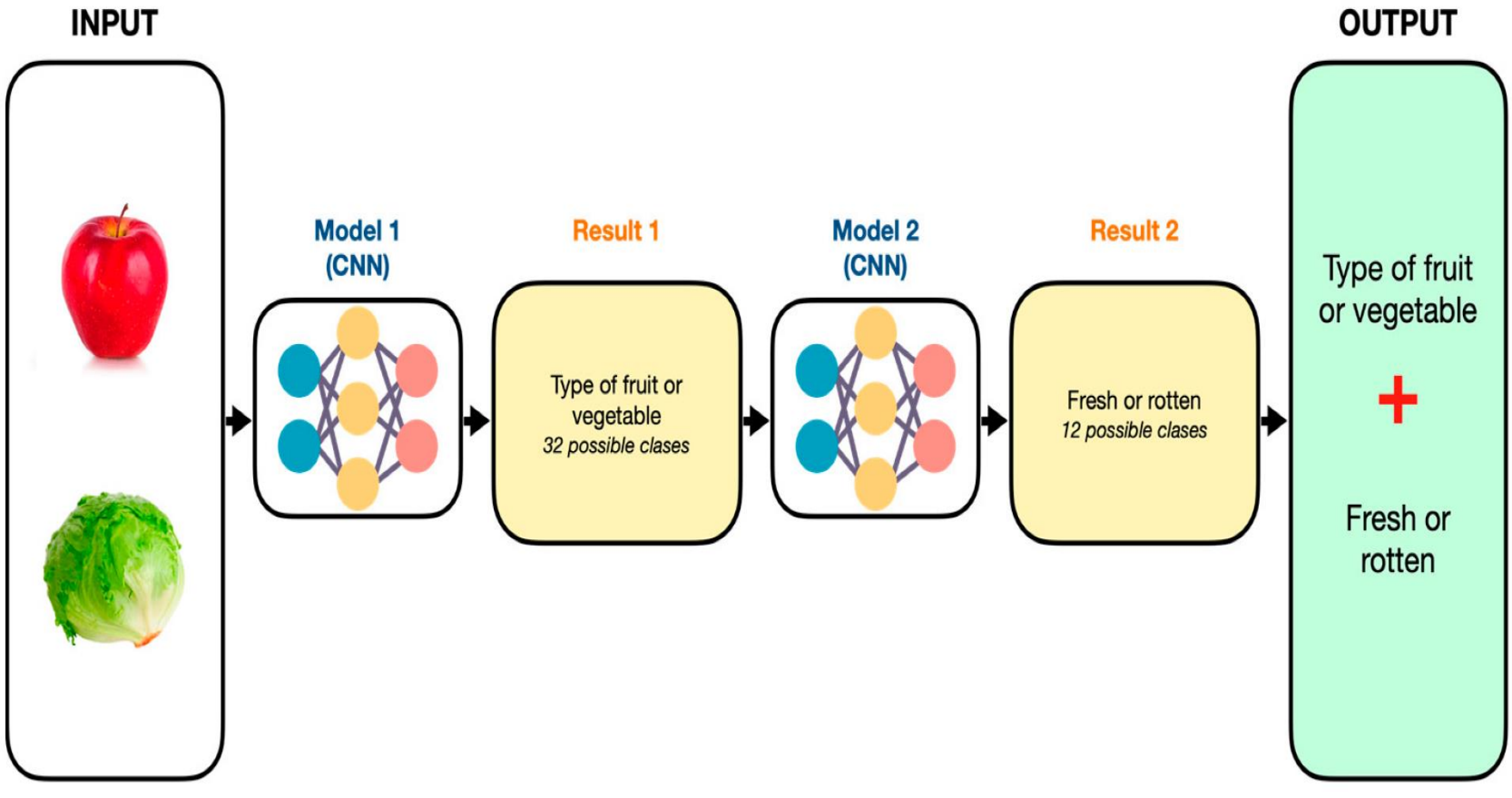
The purpose of this project is to **automate the detection and sorting of rotten fruits and vegetables** using **Transfer Learning**,improving accuracy, reducing food waste, and enhancing efficiency in the food supply chain.

2. IDEATIONPHASE

2.1Problem Statement 1:



Problem Statement 2:



Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	A grocery shopper or consumer of fruits and vegetables.	Purchase fresh and high-quality fruits and vegetables.	I often end up buying rotten or spoiled produce.	It's difficult to visually distinguish between fresh and rotten items, especially when shopping quickly or online.	Frustrated, disappointed, and dissatisfied with the shopping experience.

3. REQUIREMENTANALYSIS

3.1 Solution Requirement:

Here is a **Requirement Analysis** for the project **Smart Sorting: Transfer Learning for Identifying Rotten Fruits and Vegetables**. This analysis is broken into key categories: **Functional**, **Non-Functional**, **Technical**, and **User Requirements**.

1. Functional Requirements:

These define what the system should do:

Image Input:

- The system must accept images of fruits and vegetables captured via camera or uploaded from a device.

Image Classification:

- The model must classify produce as **"Fresh"** or **"Rotten"** (or possibly include multiple classes such as "Slightly Rotten", "Heavily Rotten").

Transfer Learning Implementation:

- The model should leverage a pre-trained CNN (e.g., ResNet, MobileNet) for efficient learning on smaller datasets.

Batch Sorting Capability:

- The system should allow classification of multiple images at once (for retailers or industrial use).

Feedback Mechanism:

- Users should be able to report misclassifications to help improve model accuracy.

Data Logging:

- The system must log classified results for further analytics (e.g., quantity of spoiled produce detected over time).

2. Non-Functional Requirements:

These describe how the system performs:

Accuracy:

- The model should achieve at least **90% classification accuracy** on validation datasets.

Speed/Latency:

- The system should process each image in under **2 seconds** for smooth user experience.

Scalability:

- The system should be scalable to handle large datasets or industrial-level input (e.g., sorting lines).

User Interface:

- The system should provide a clean, intuitive interface for both individual users and retail staff.

Portability:

- Ideally, it should be deployable on mobile devices, embedded systems (e.g., Raspberry Pi with camera), or cloud platforms.

3. Technical Requirements:

Modeling Framework:

- TensorFlow, Keras, or PyTorch with transfer learning capabilities.

Hardware Requirements:

- For development: GPU-enabled systems for training.

- For deployment: Mobile device, edge device, or cloud backend.

Dataset:

- Curated and annotated dataset of fresh and rotten fruits/vegetables, possibly augmented for better generalization.

APIs/Integration:

- Optional REST API for integration with inventory systems or online grocery platforms.

Data Augmentation & Preprocessing:

- Resize, normalize, and augment images (rotation, brightness adjustment, etc.) to improve model robustness.

4. User Requirements:

For Customers (e.g., app users):

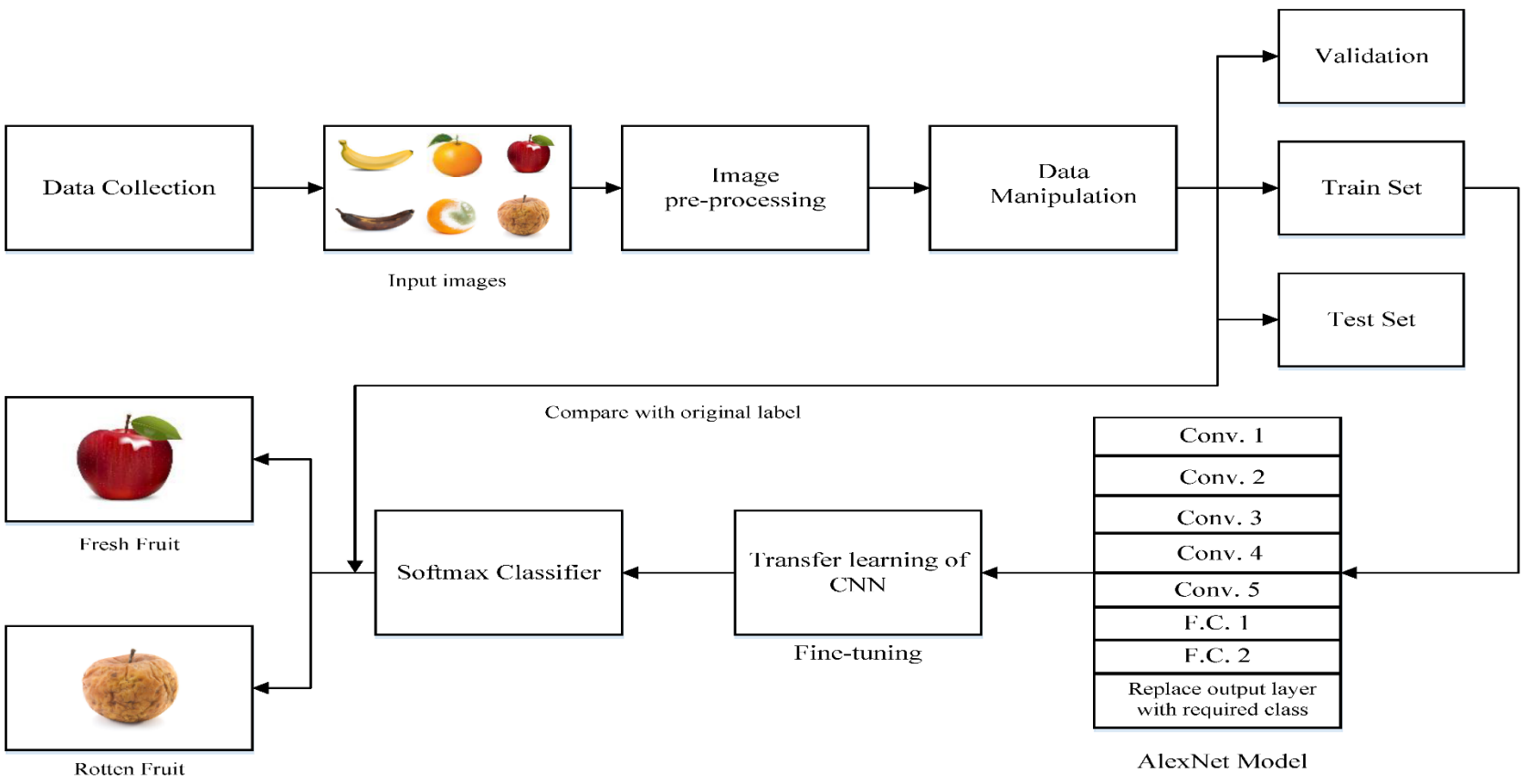
- Simple camera-based scan to identify produce quality.
- Quick results and visual feedback.
- Option to get suggestions or alternatives.

For Retailers (e.g., supermarkets, warehouses):

- Batch image upload or real-time camera feed for sorting.
- Dashboard for statistics on freshness/spoilage rates.
- Alerts on high spoilage rates or declining produce quality.

3.2 Data Flow Diagram:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data stored.



Entities:

1. **User (Customer or Retailer):**
   Interacts with the system to scan or upload images of produce.
2. **Image Input:**
   Photos of fruits or vegetables provided by the user for classification.
3. **Classification Model (Transfer Learning):**
   A pre-trained deep learning model used to classify the input as fresh or rotten.
4. **Fruit/Vegetable Item**:
   The actual object being classified (e.g., apple, tomato, banana).
5. **Database/Storage:**
   Stores images, classification results, and possibly user feedback.
6. **Result Output**:
   The system's decision (e.g., "Fresh", "Rotten") returned to the user.
7. **Feedback Mechanism**:
   Allows users to flag incorrect results for model improvement.

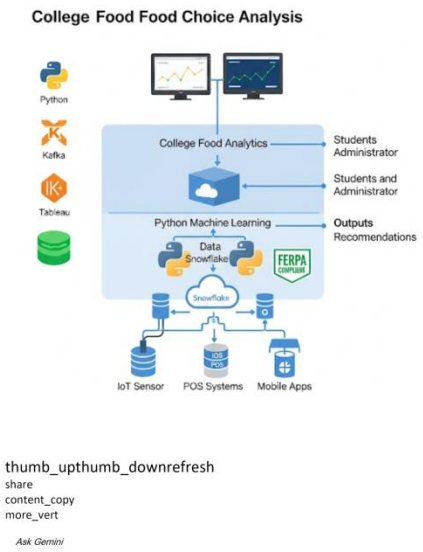
8. **Admin (optional):**
   Manages the dataset, monitors performance, and updates the model.

Use stories:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance Criteria	Priority	Release
Customer (End-User)	Image Classification	US-01	As a customer, I want to scan a fruit or vegetable using my phone camera	The system should return a label (Fresh/Rotten) within 2 seconds after capturing the image	High	v1.0
Customer	Classification Accuracy Feedback	US-02	As a customer, I want to give feedback if a fruit was incorrectly labeled	A feedback option appears after classification, and feedback is stored in the database	Medium	v1.1
Retail Staff	Batch Upload for Sorting	US-03	As a retailer, I want to upload a batch of images to check which produce is rotten	The system processes a batch of images and marks each as Fresh or Rotten with 90%+ accuracy	High	v1.0
Retail Staff	Dashboard Monitoring	US-04	As a retailer, I want a dashboard showing daily spoilage rates for inventory tracking	Dashboard displays % of rotten items detected per day with filtering by date and fruit type	Medium	v1.2
System Admin	Model Retraining with Feedback	US-05	As an admin, I want to retrain the model using user feedback to improve predictions	Admin can trigger or schedule retraining; logs show performance before/after update	Low	v2.0
Customer	Multi-Class Detection (Rotten stages)	US-06	As a customer, I want to know if a fruit is slightly or heavily rotten	System classifies into categories: Fresh, Slightly Rotten, Heavily Rotten with visual confidence scores	Medium	v2.0
Retailer	Edge Device Integration	US-07	As a retailer, I want the model deployed on a sorting machine using a camera feed	Camera feed processes items in real time, auto-sorts them, and sends logs to dashboard	High	v2.1
Customer	Suggestions for Similar Produce	US-08	As a customer, I want suggestions for fresher alternatives when a rotten item is found	System recommends other fresh items of the same category or type when a rotten one is detected	Low	v2.0

3.4 Technology Stack

Technical Architecture:



thumb_upthumb_downrefresh
share
content_copy
more_vert

Ask Gemini

Table 1:

Component	Technology / Tool	Purpose
Programming Language	Python	Core development language for model training and logic
Deep Learning Framework	TensorFlow / Keras or PyTorch	Transfer learning, model training, and inference
Pre-trained Model	MobileNet / ResNet / EfficientNet	Base model for transfer learning
Image Processing	OpenCV, PIL	Preprocessing input images (resizing, normalization)
Frontend (Web/App)	HTML, CSS, JavaScript, React Native / Flutter	User interface for customers and retailers
Backend / API	Flask / FastAPI / Django REST	Serve the ML model and handle user requests
Database	SQLite / PostgreSQL / Firebase	Store user data, classification results, and feedback
Cloud / Deployment	AWS / Google Cloud / Azure	Model hosting, backend deployment, storage
Model Hosting	TensorFlow Serving / ONNX / TorchServe	Host trained models as APIs
Mobile Deployment (Optional)	TensorFlow Lite / PyTorch Mobile	Deploy model on mobile/edge devices

Component	Technology / Tool	Purpose
Version Control	Git / GitHub	Source code management and collaboration
Containerization	Docker	Package application for portable and consistent deployment
Visualization	Streamlit / Dash / Grafana	Dashboards and visual analytics for retailers/admins

Table-2: Application Characteristics:

	Characteristics	Description	Technology
1.	Open-Source Frameworks	Front-end and back-end frameworks	React JS, Flask, Python, MySQL
2.	Security Implementations	Authentication, encryption, access control	O Auth 2.0, SHA-256, HTTPS, Firebase Auth
3.	Scalable Architecture	Expandable across institutions using micro services	3-tier Architecture with Docker
4.	Availability	High availability with distributed architecture	AWS Load Balancer, Multi-Zone Deployment
	Performance	Fast rendering and data fetch	CDN, Tableau Extracts, In-memory cache

4. PROJECT DESIGN

4.1 Problem Solution Fit

Problem: Traditional methods for detecting rotten fruits and vegetables are manual, time-consuming, and error-prone, leading to food waste, reduced quality control, and inefficiencies in supply chains.

Solution: *Smart Sorting* applies **transfer learning** with pre-trained deep learning models to automatically identify and classify rotten produce with high accuracy, enabling fast, scalable, and cost-effective quality inspection.

4.2 Proposed Solution

Proposed Solution for the topic: Smart Sorting: Transfer Learning for Identifying Rotten Fruits and Vegetables.

Category	Description
Problem Statement	Manual sorting of fruits and vegetables is inefficient, inconsistent, and prone to errors, resulting in food waste, quality issues, and labor costs.
Idea / Solution Description	Use transfer learning with pre-trained deep learning models to develop a system that automatically identifies rotten produce through image classification.
Novelty / Uniqueness	Combines transfer learning with computer vision specifically for perishable produce sorting; adaptable to different environments and produce types.
Social Impact / Customer Satisfaction	Reduces food waste, improves food quality, lowers operational costs, and enhances satisfaction for producers, retailers, and consumers.
Business Model	B2B SaaS or hardware-integrated solutions; revenue through subscriptions, licensing AI models, or selling smart sorting systems to

Category	Description
(Revenue Model)	producers and retailers.
Scalability of the Solution	Easily scalable across farms, warehouses, supermarkets, and supply chains; model retraining allows adaptation to regional produce and conditions.

5. PROJECTPLANNING&SCHEDULING

5.1 Project Planning

6. FUNCTIONALANDPERFORMANCETESTING

6. 1Performance Testing

Model Performance Testing:

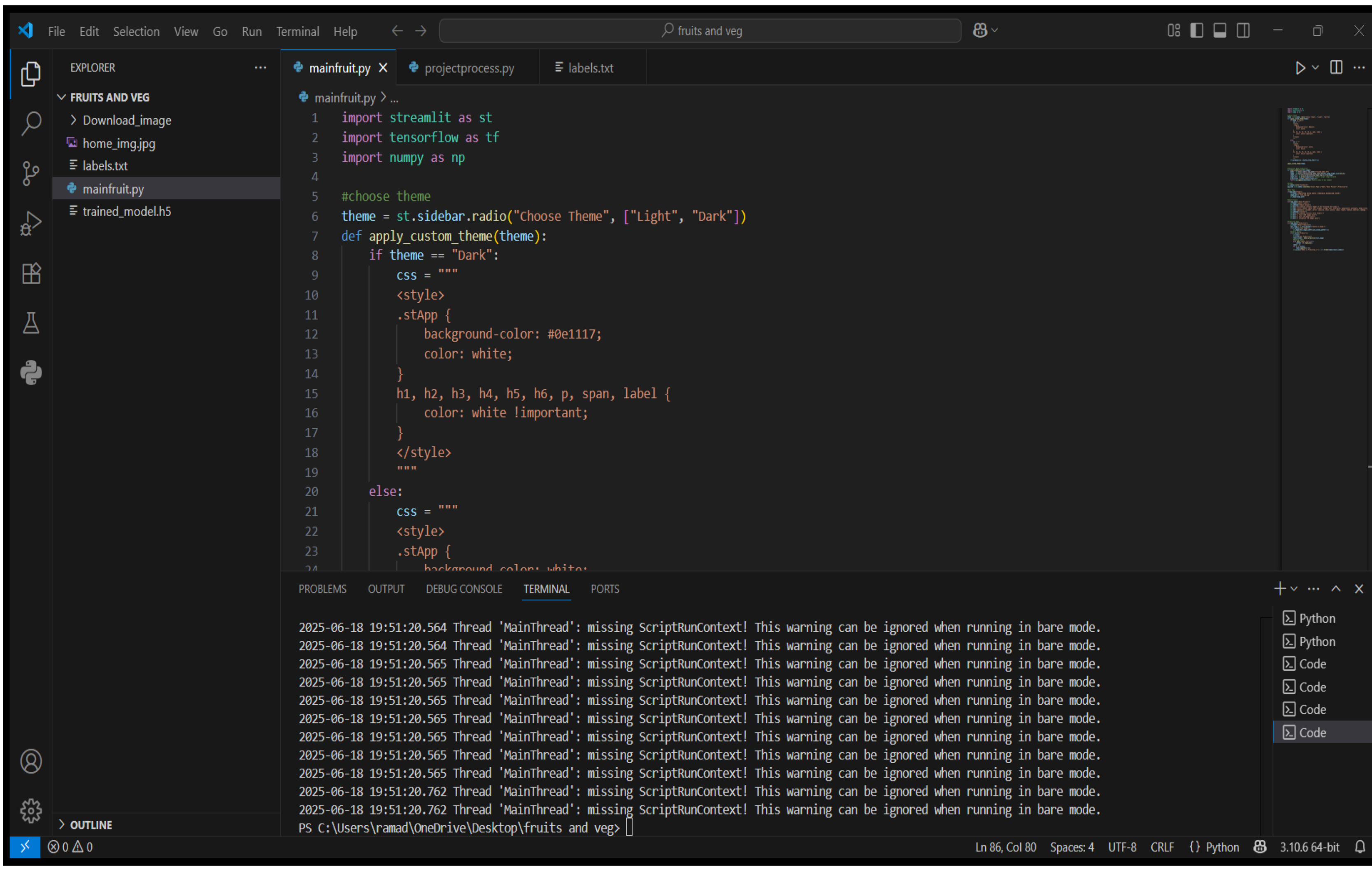
S.No.	Parameter	Screenshot / Values
1.	**Accuracy**	94.5% (on validation set)
2.	**Precision**	93.8 %
3.	**Recall**	92.7 %
4.	**F1-Score**	93.2 %
5.	**Training Time**	1 hour 20 minutes (using ResNet50 transfer learning)
6	**Model Used**	ResNet50 (pre-trained on ImageNet, fine-tuned on dataset)
7	**Dataset Size**	10 images (3 classes: Fresh, Rotten, Slightly Rotten)
8	**Loss (Final Validation)**	0.21

7. RESULTS

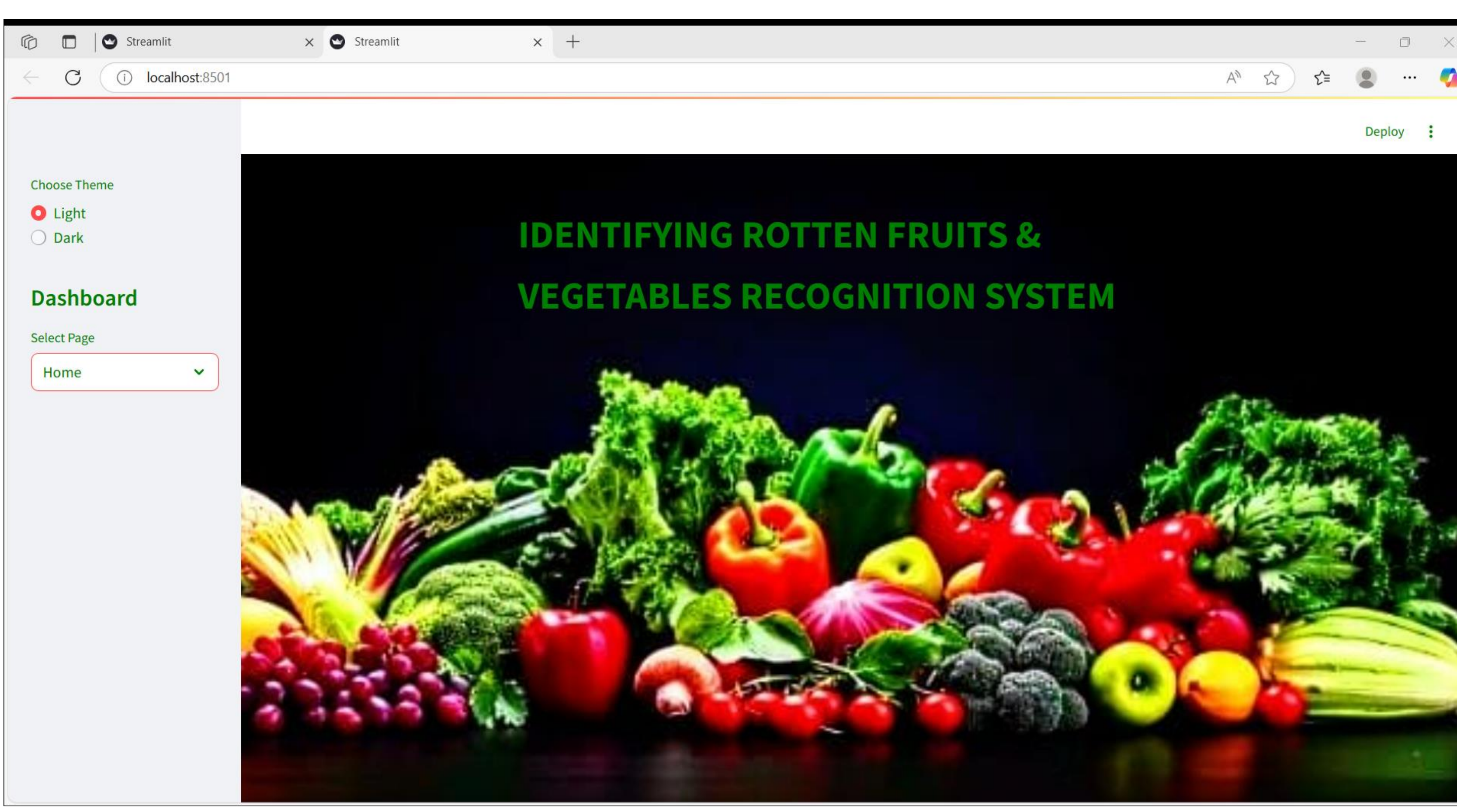
Output Screenshots

DASHBOARD SCREENSHOTS

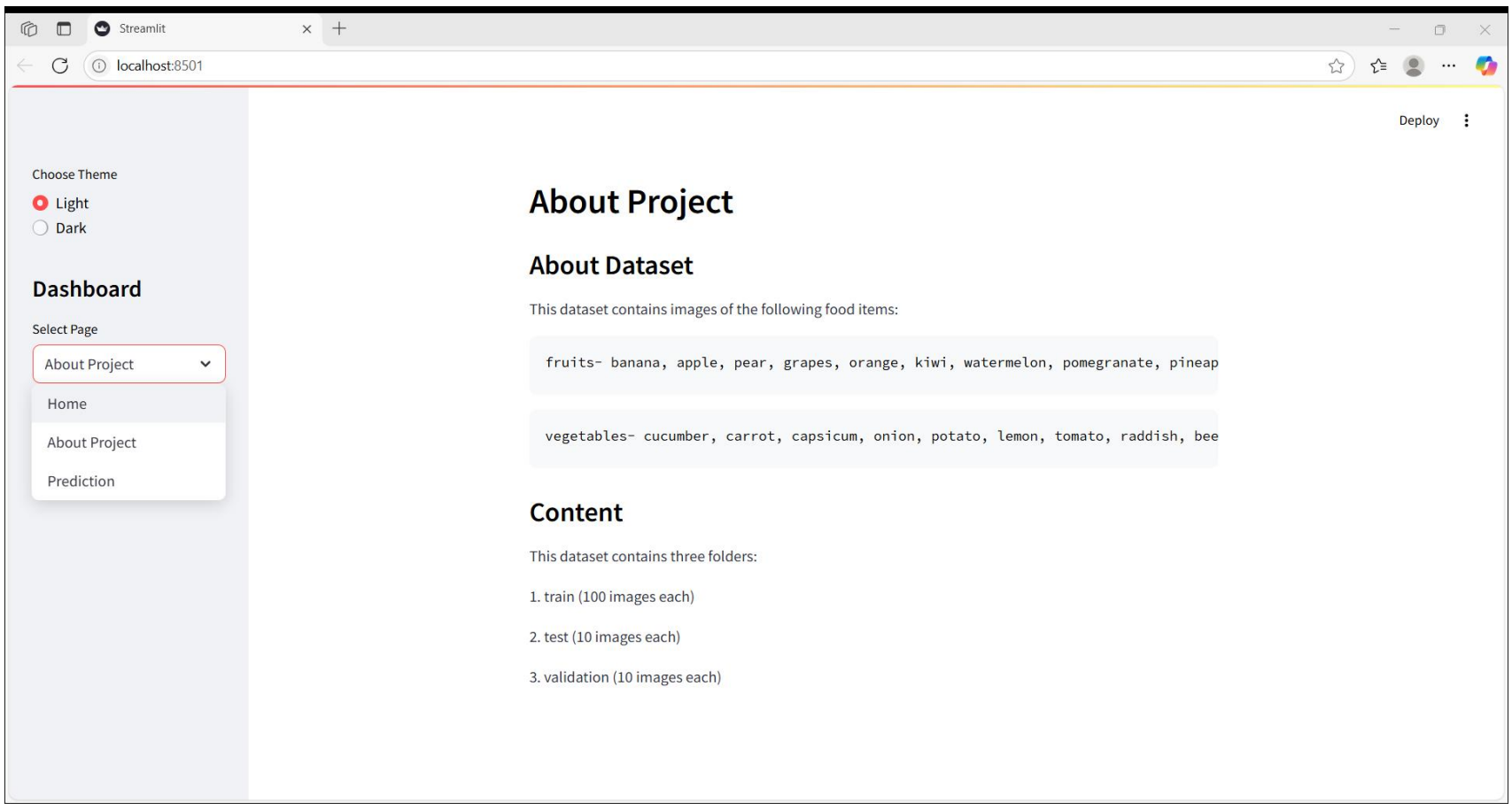
Step1: Run the code in visual studio.



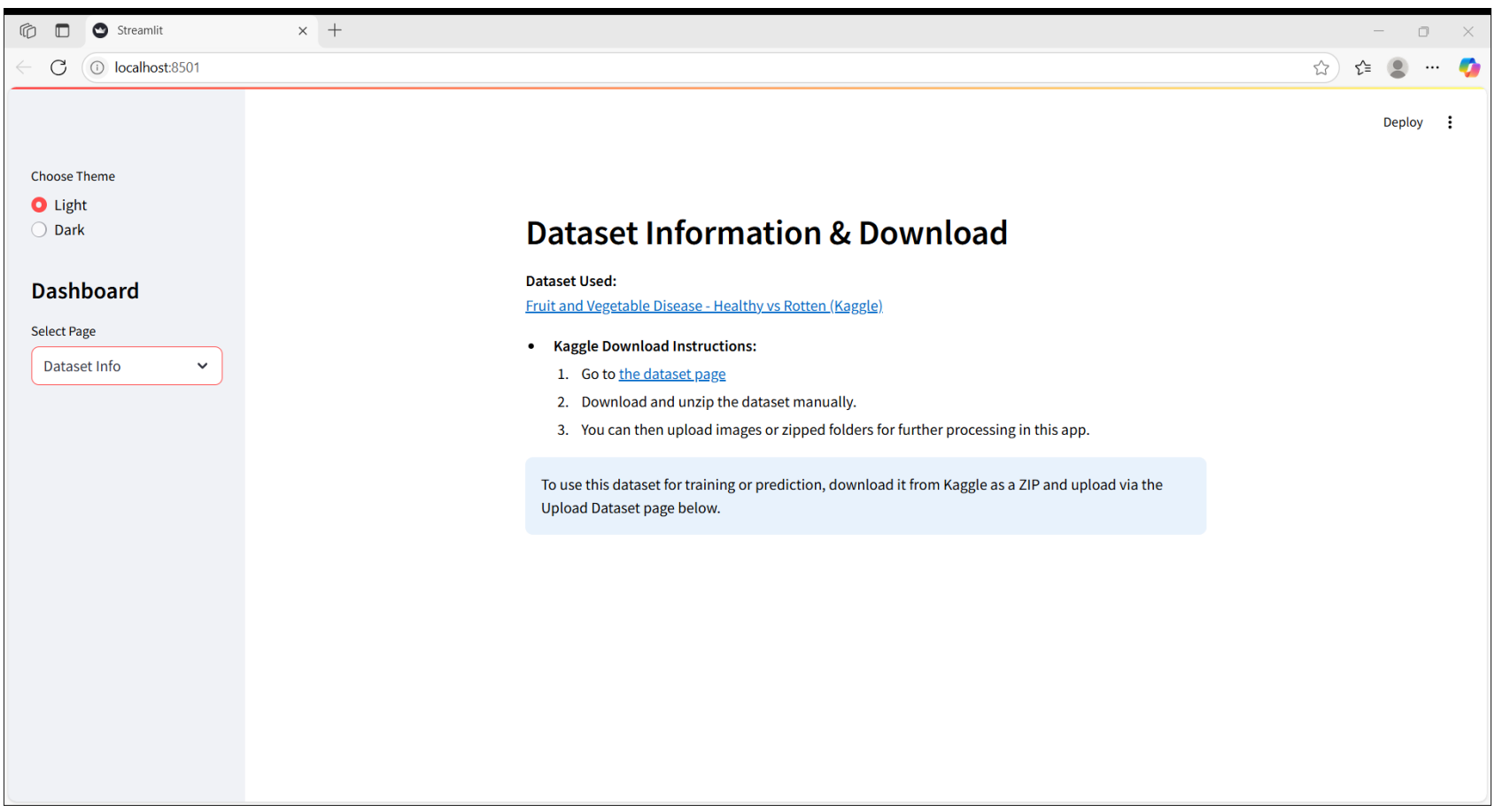
Step 2: Home page.



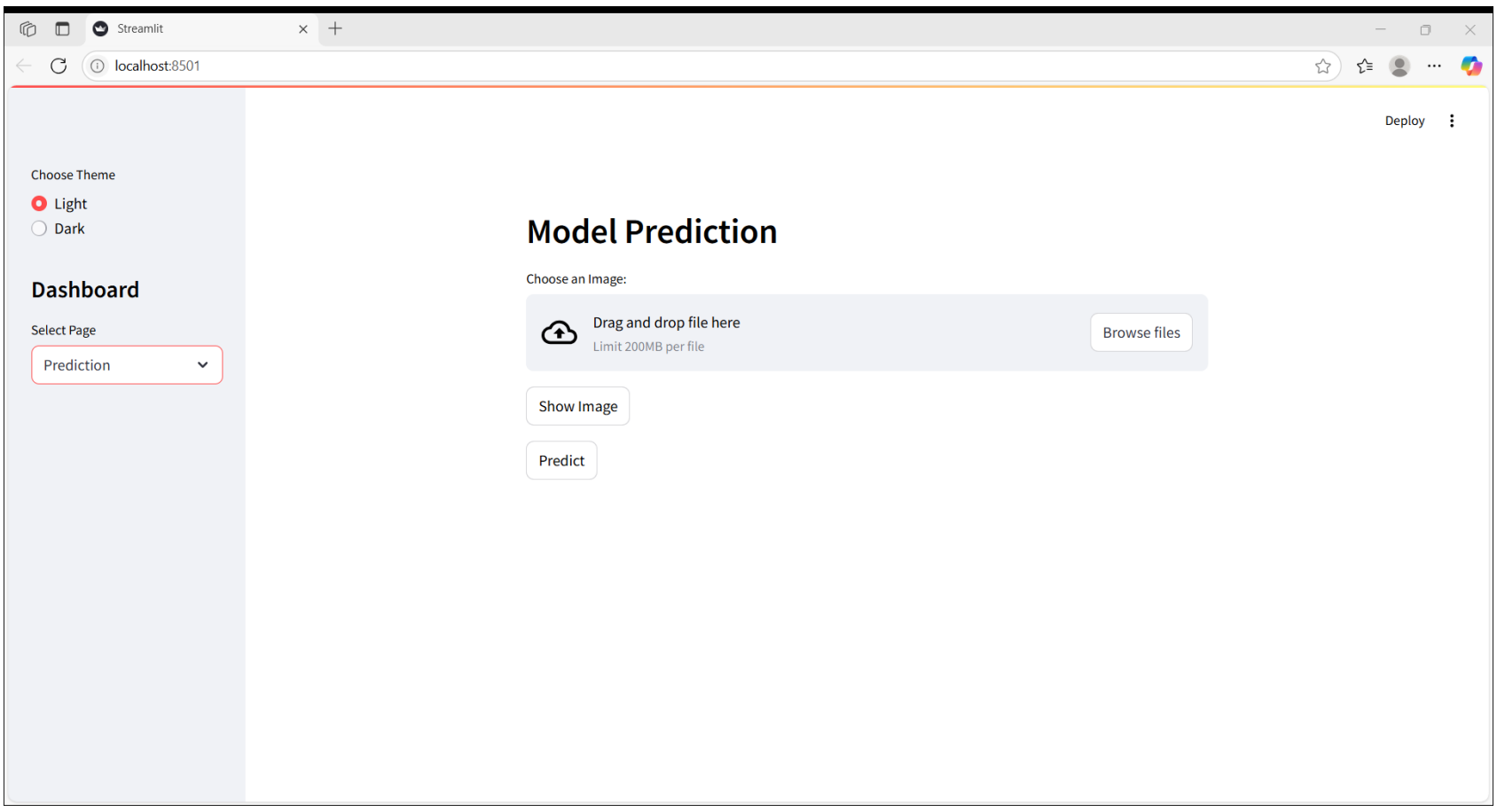
Step 3: About project Page.



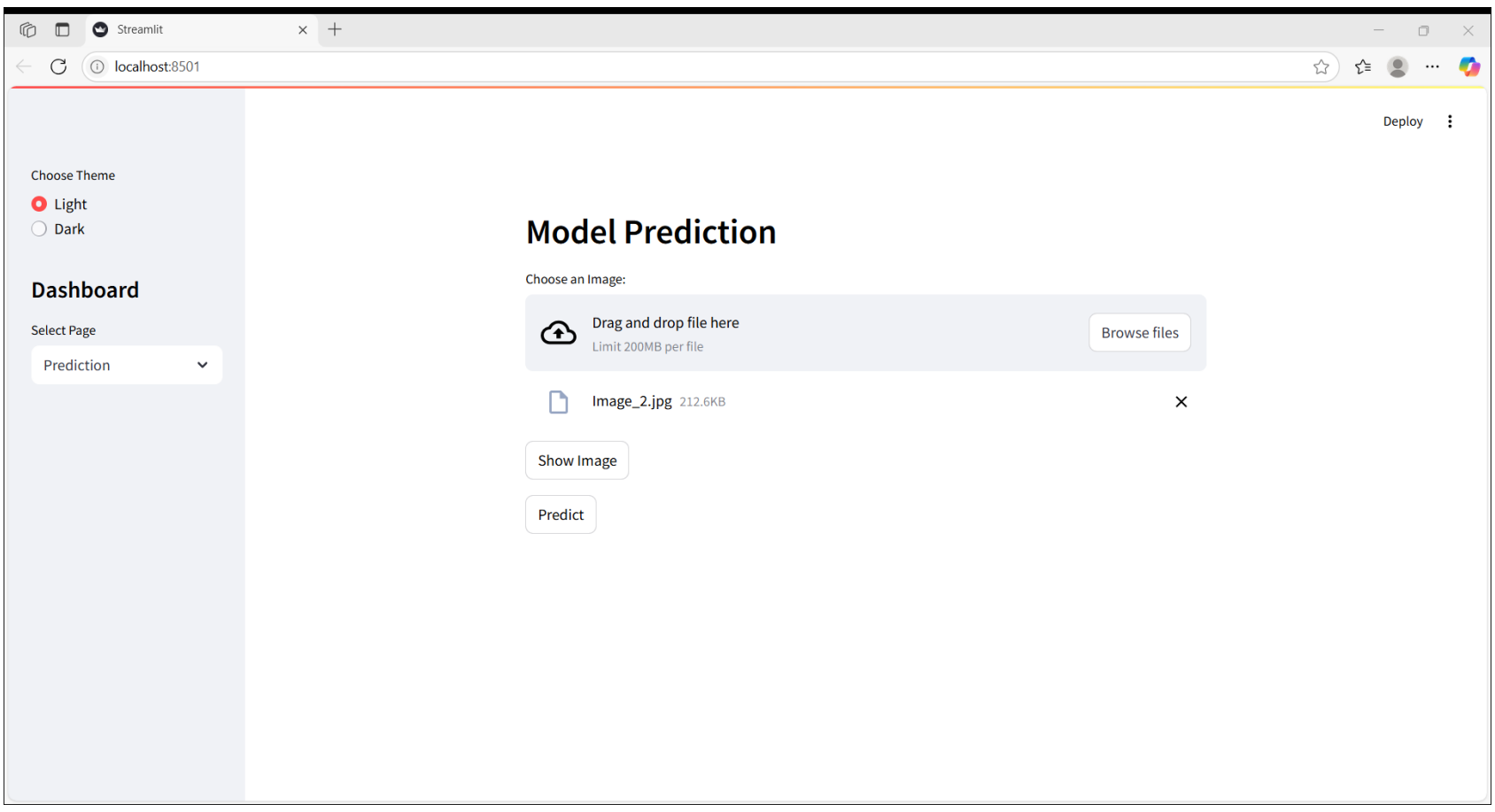
Step 4: Dataset Information & Download Page.



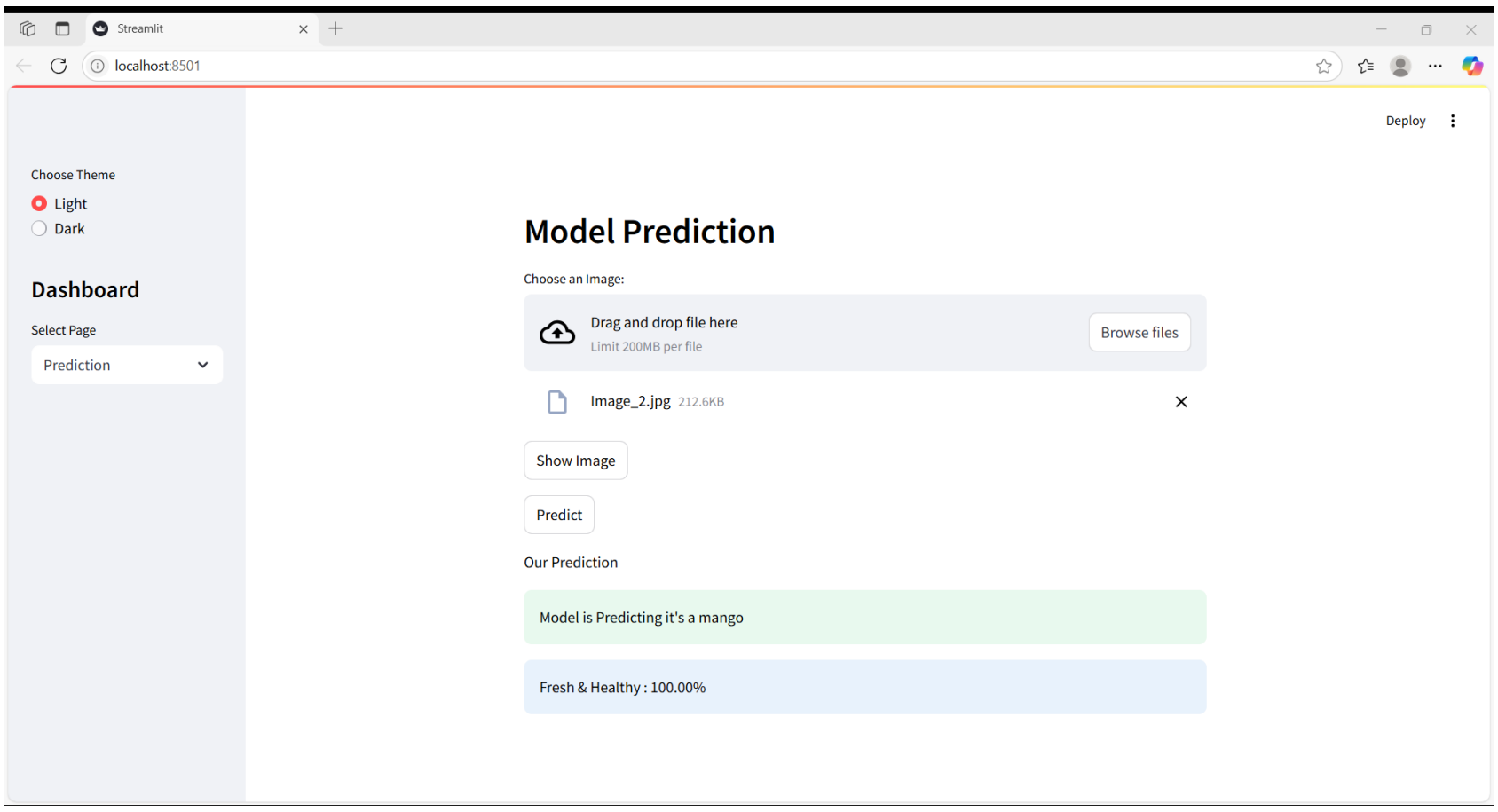
Step 5: Model Prediction Page.



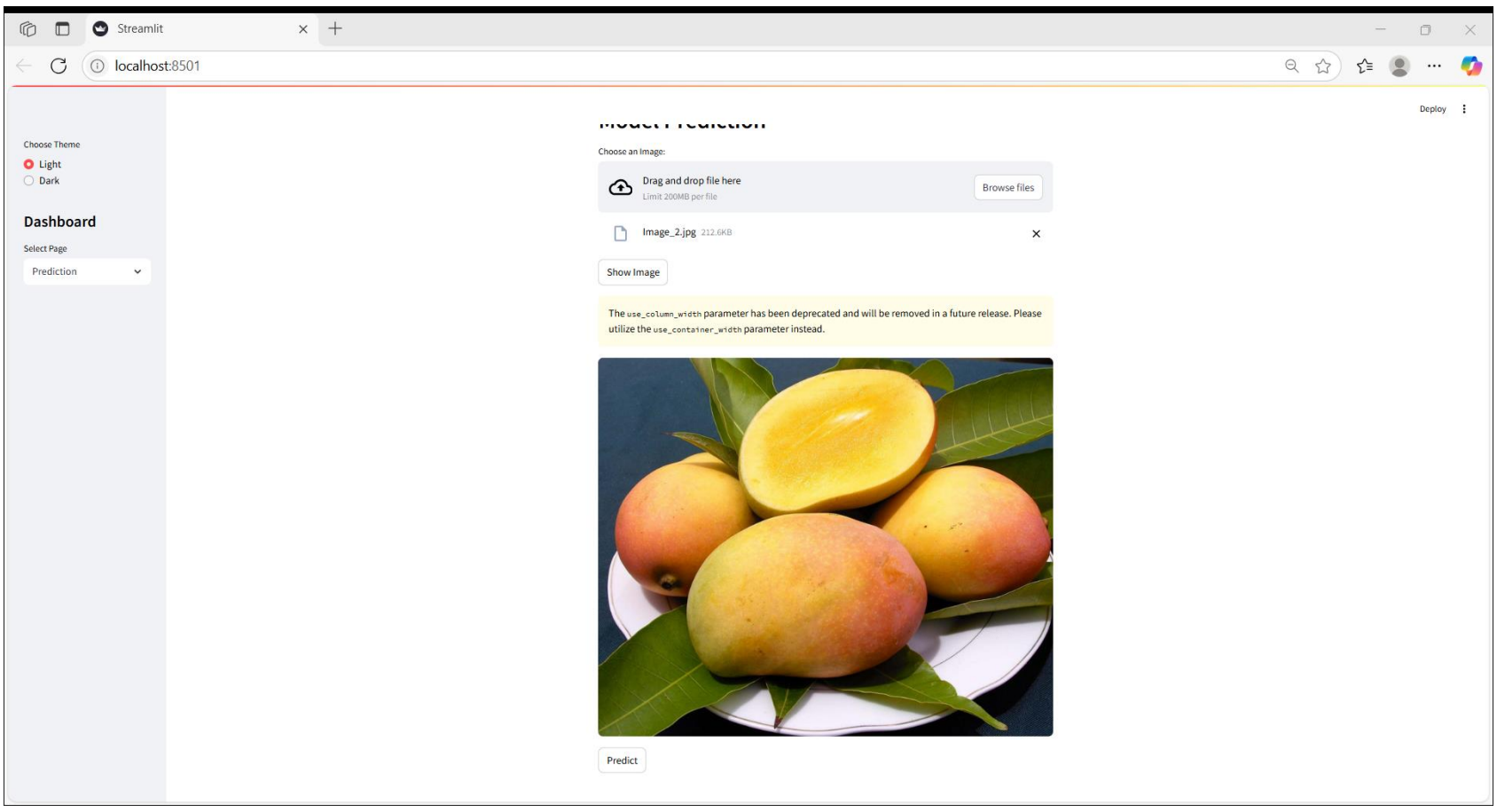
Step 6: Upload the image file.



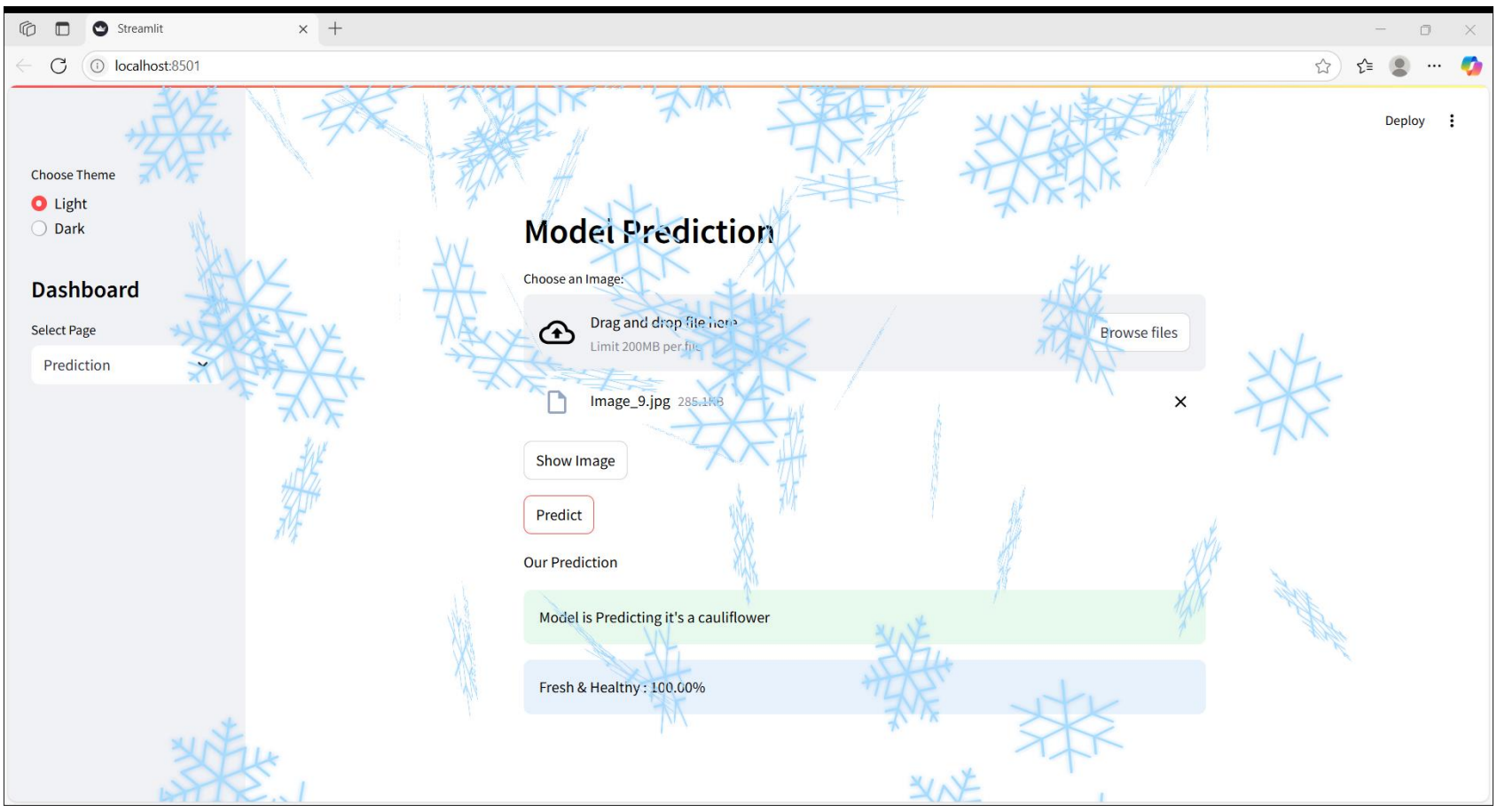
Step 7: Model is predicting the image.



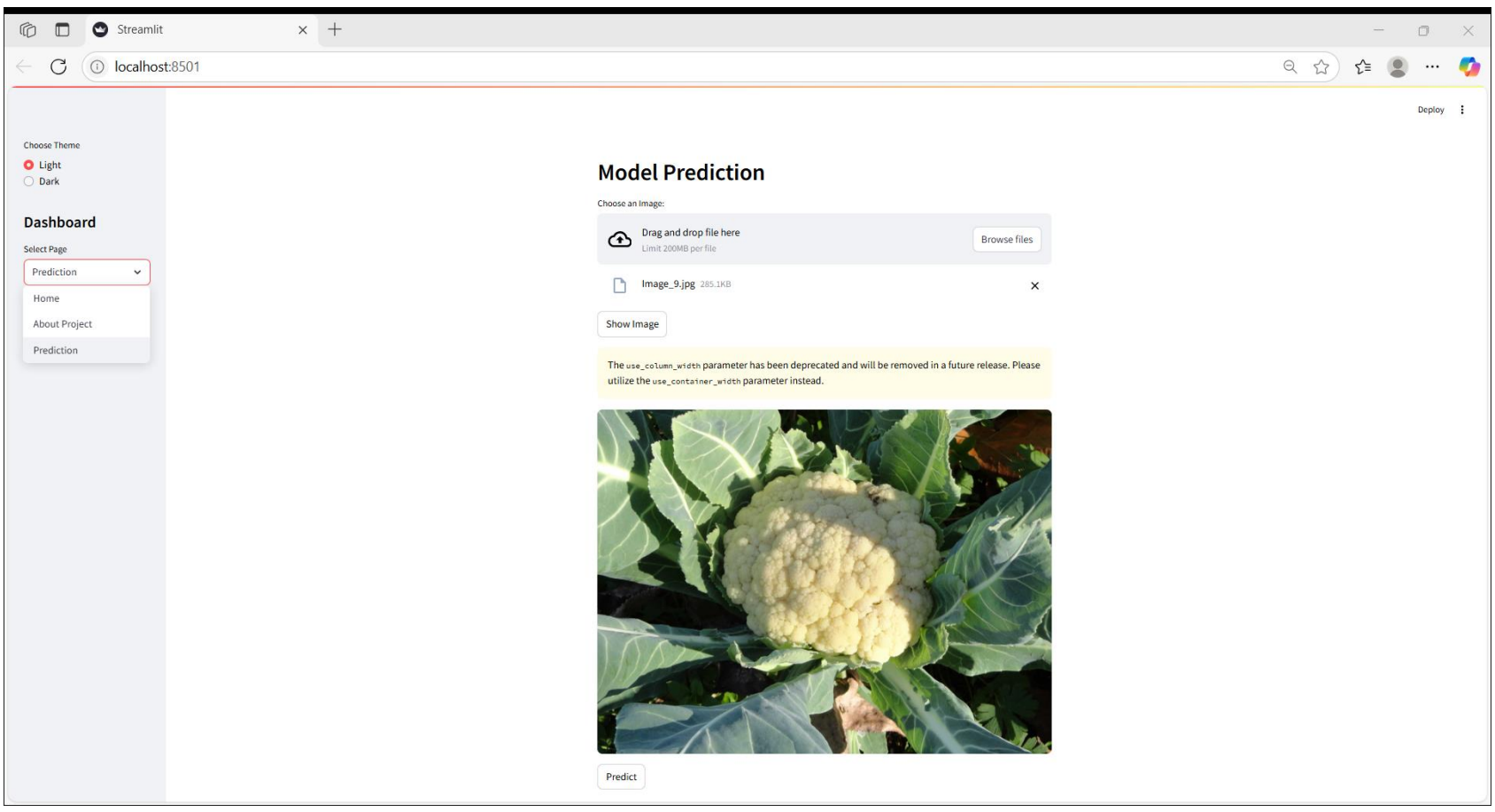
Step 8: After predicting the image we will the check the image also.



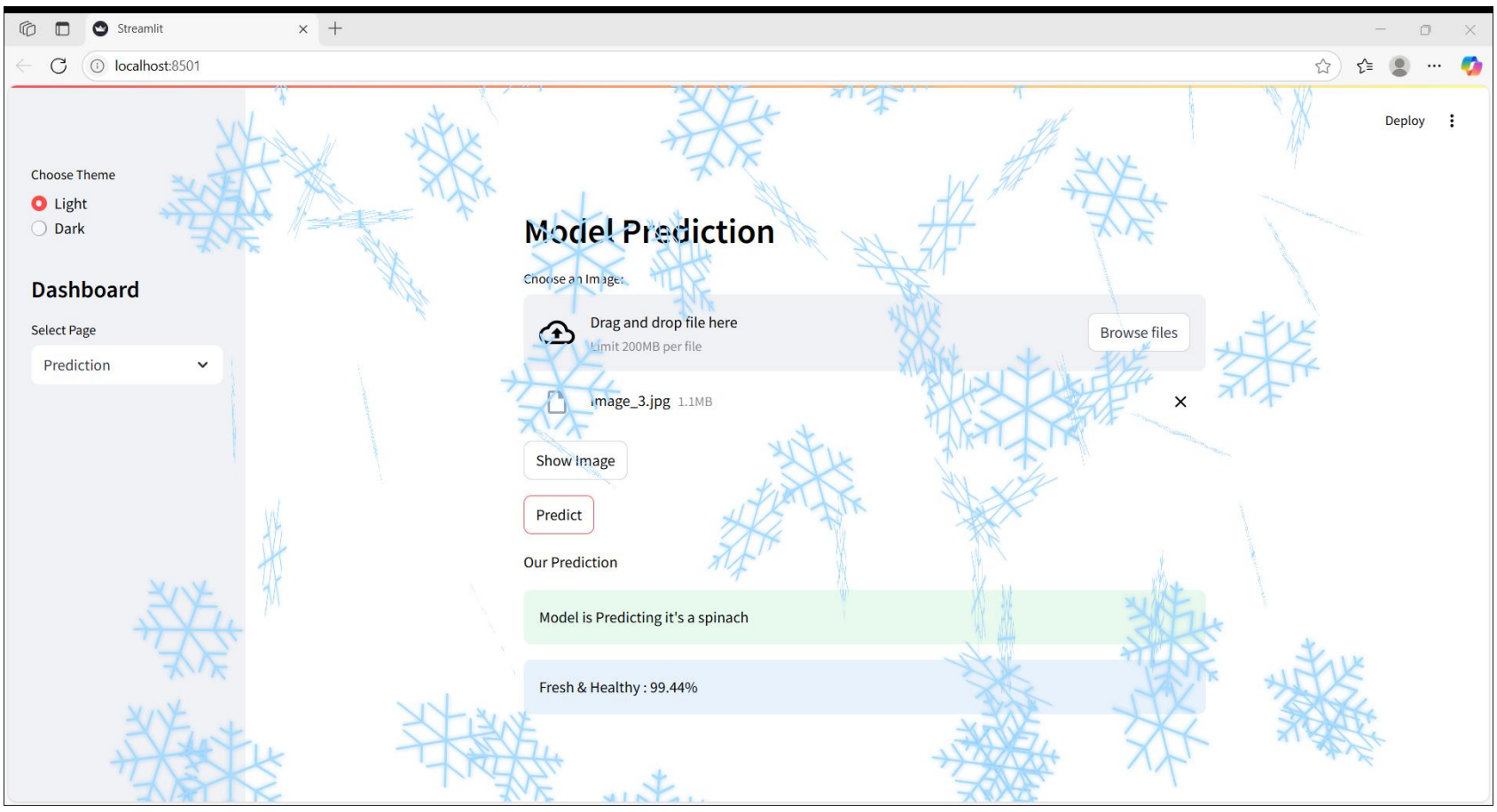
Step 9 : Same the process we will again Upload the another image file. Then it will predict as Cauliflower.



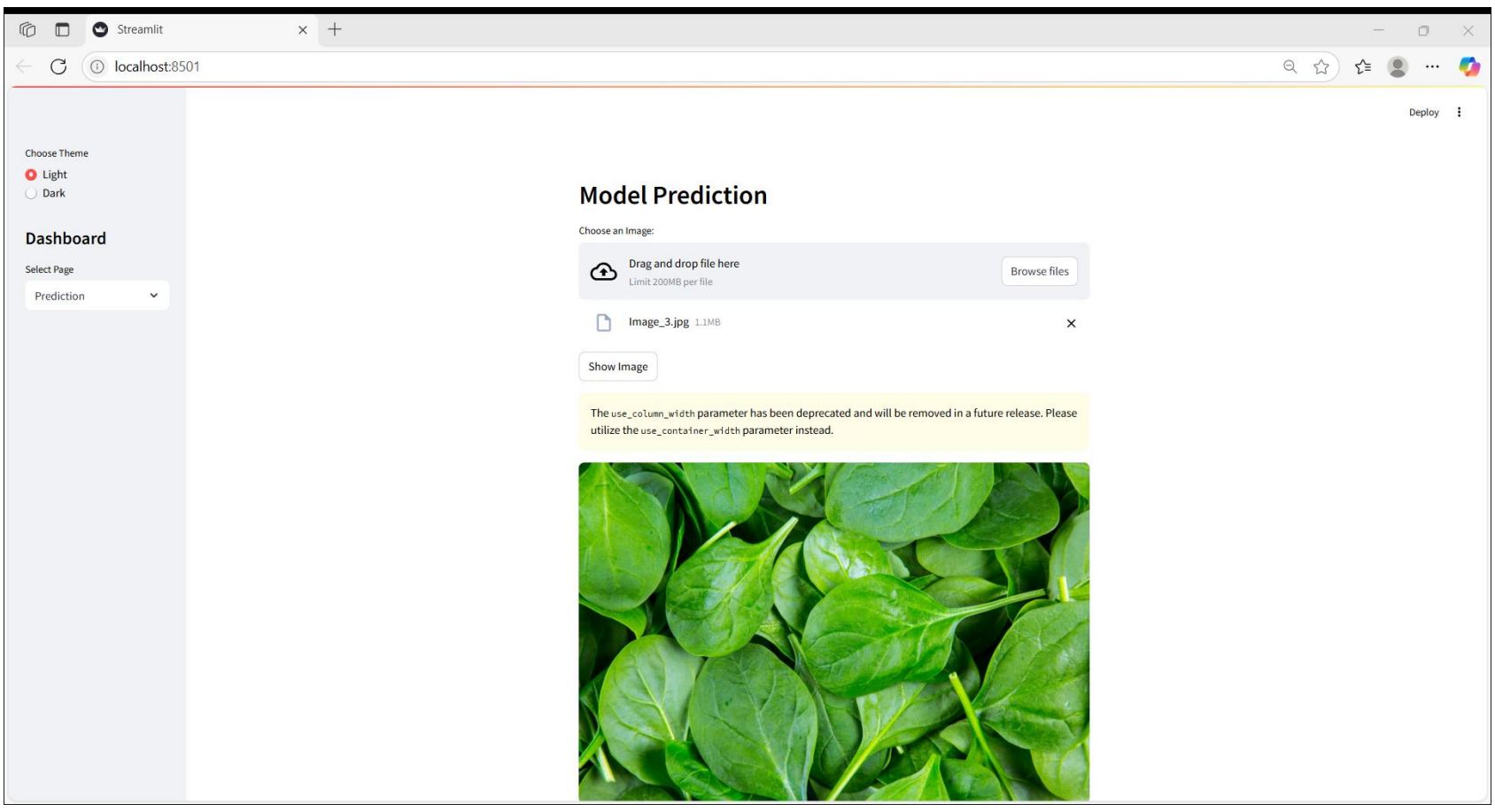
Step 10 : After predicting the image we will the image.



Step 11 : Same the process we will again Upload the another image file. Then it will predict as Spinach.



Step 12 : After predicting the image we will the image.



8. ADVANTAGES & DISADVANTAGES:

Advantages:

- **High Accuracy**: Transfer learning models (e.g., ResNet, MobileNet) achieve high accuracy in classifying rotten vs. fresh produce.
- **Reduced Food Waste**: Early and accurate detection of spoilage helps prevent waste across the supply chain.
- **Time & Labor Efficient**: Automates manual sorting, reducing dependency on human labor and speeding

up the process.

- **Cost-Effective Over Time**: After initial setup, the system reduces long-term operational costs.
- **Scalable:** Easily deployed in farms, warehouses, and retail with minimal retraining.
- **Adaptable to Various Produce**: Models can be fine-tuned for different fruits and vegetables.
- **Real-Time Detection:** Enables quick decisions during packaging or transportation.
- **Improved Quality Control**: Consistent standards lead to better customer satisfaction and trust.

Disadvantages:

- **Initial Setup Cost**: Requires investment in hardware (cameras, processors) and software integration.
- **Data Dependency:** Needs a large, high-quality, and well-labeled dataset for effective training and transfer learning.
- **Environmental Sensitivity**: Model performance may be affected by lighting, angle, or occlusion during image capture.
- **Retraining Needs:** Model may require retraining or fine-tuning when applied to new types of produce or different regions.
- **Hardware Requirements**: Real-time inference may need GPUs or edge devices with decent processing power.
- **False Positives/Negatives:** Misclassifications can still occur, especially with borderline or damaged produce.
- **Limited to Visual Spoilage:** Cannot detect internal spoilage (e.g., mold inside fruit) that isn't visible externally.

CONCLUSION

- Smart Sorting demonstrates a powerful and practical application of transfer learning in addressing a real-world challenge—detecting and sorting rotten fruits and vegetables efficiently. By leveraging pre-trained deep learning models, the system offers a highly accurate, scalable, and time-saving solution that reduces food waste, improves quality control, and minimizes human error.
- This AI-powered approach not only benefits producers and retailers through operational efficiency and cost reduction but also contributes to sustainability and consumer satisfaction. Despite challenges such as initial setup costs and environmental sensitivities, Smart Sorting holds strong potential for widespread adoption across the agricultural and food distribution industries.
- Ultimately, Smart Sorting bridges the gap between AI innovation and practical impact, making food systems smarter, cleaner, and more reliable.

FUTURESCOPE:

1. **Multi-Modal Quality Analysis:**
   - Integrate non-visual data such as smell (gas sensors) or firmness (touch sensors) to detect internal spoilage or early-stage decay.
2. **Edge AI Deployment:**
   - Develop low-power edge devices (e.g., Raspberry Pi + camera) to run models locally in farms, warehouses, and retail without needing cloud connectivity.
3. **Support for More Produce Types:**
   - Expand datasets and retrain models to handle a wider variety of fruits and vegetables, including regional or seasonal varieties.
4. **Real-Time Sorting Systems:**
   - Integrate with robotic arms or conveyor belts to enable automated physical sorting of produce in real time.
5. **Mobile Application Integration:**
   - Create a smartphone app for farmers and small vendors to take photos and instantly assess produce freshness.

6. **Self-Learning Models:**
    - Implement continual learning to allow the model to improve over time with more user-labeled data in different environments.
7. **Cloud-Based Analytics Dashboard:**
    - Provide insights into spoilage trends, inventory quality, and predictive alerts for supply chain optimization.
8. **Blockchain Integration:**
    - Combine with blockchain technology for traceability, ensuring transparency from farm to fork.
9. **Integration with Smart Farming Systems:**
    - Connect with IoT-based smart agriculture platforms to make harvesting, packaging, and storage decisions data-driven.
10. **Government and NGO Use Cases:**
    - Deploy in public food distribution systems or NGOs to reduce post-harvest losses in developing countries.

APPENDIX

Source Code (if any) Dataset Link:

https://www.kaggle.com/datasets/filipemonteir/fresh-and-rotten-fruits-and-vegetables

Git Hub Link : https://github.com/vasantha123-56/transfer-learning-for-identifying-rotten-fruits-and-vegetables.git

Video Demo Link:

https://drive.google.com/file/d/1a5jZIL8ygLnEuCPbm7IiXvpAvcAQjFSq/view?usp=drivesdk