SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

CHAPTER-1
INTRODUCTION

1.1 PROJECT OVERVIEW

In the present era, the agricultural and food industries play a vital role in maintaining food quality and ensuring consumer safety. One of the major challenges faced by these industries is the accurate identification and separation of rotten fruits and vegetables during processing, packaging, and distribution stages. Traditionally, this task is carried out through manual inspection by human workers, which is time-consuming, labor-intensive, and prone to errors. The efficiency of manual sorting largely depends on the experience and continuous attention of workers, often resulting in inconsistencies and reduced productivity.

With the increasing global demand for fresh and high-quality produce, there is a growing need for automated and intelligent systems that can efficiently detect and sort rotten fruits and vegetables. Smart Sorting addresses this challenge by integrating artificial intelligence, deep learning, and computer vision technologies. The project employs transfer learning, where pre-trained deep learning models are adapted to identify various fruits and vegetables and classify them as fresh or rotten. This approach eliminates the need to develop complex models from scratch, significantly reduces training time, and delivers high accuracy even with limited datasets.

These intelligent systems can be deployed in real-time industrial environments, enabling automation of the sorting process, minimizing human error, and improving operational efficiency. The system operates by capturing images of fruits and vegetables using cameras installed on conveyor belts or storage units. The captured images are then analyzed by a trained neural network, which classifies each item based on its condition, ensuring reliable and efficient quality control

1.2 PROBLEM SPECIFICATION AND PURPOSE

The identification of rotten fruits and vegetables is traditionally carried out through manual inspection, which is slow, inconsistent, and prone to human error. As food production and distribution continue to increase, manual inspection becomes inefficient and fails to maintain consistent quality standards. Existing automated systems often rely on basic image processing techniques, which struggle under varying lighting conditions, complex surface textures, and different spoilage patterns.

Therefore, there is a need for a more accurate, intelligent, and adaptable solution to identify rotten produce. The Smart Sorting project aims to address this challenge by using deep learning and image-based analysis to classify fruits and vegetables as fresh or rotten. The goal is to develop a reliable system capable of detecting subtle visual changes such as color variation, texture degradation, and signs of spoilage, ensuring improved quality control and efficiency.

SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

DATE	20-02-2026
TEAM ID	LTVIP2026TMIDS90622
PROJECT NAME	SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES
MAXIMUM MARKS	2 MARKS

CHAPTER-2
2.1 PROBLEM STATEMENT

Food spoilage is a major issue in the agricultural supply chain, retail markets, and household storage systems. A significant percentage of fruits and vegetables are wasted due to delayed detection of spoilage, improper sorting, and human error during manual inspection. Traditional sorting methods rely heavily on visual inspection by workers, which is time-consuming, inconsistent, labor-intensive, and prone to inaccuracies—especially when dealing with large volumes.

Rotten or spoiled produce often exhibits subtle visual changes such as discoloration, mold growth, texture degradation, or surface damage. Detecting these variations manually can be difficult, particularly in early stages of spoilage.

To address this issue, there is a need for an intelligent, automated system capable of accurately classifying fresh and rotten fruits and vegetables in real time. By leveraging **transfer learning** with pre-trained deep learning models (e.g., ResNet, MobileNet, VGG16, EfficientNet), the system can utilize previously learned image features to improve classification performance even with limited dataset availability.

Therefore, the problem addressed in this project is the development of an intelligent, automated fruit and vegetable identification system using transfer learning techniques. The system should be capable of accurately classifying different types of fruits and vegetables from input images in real-time while handling variations in lighting, background, and object orientation.

The proposed system aims to:

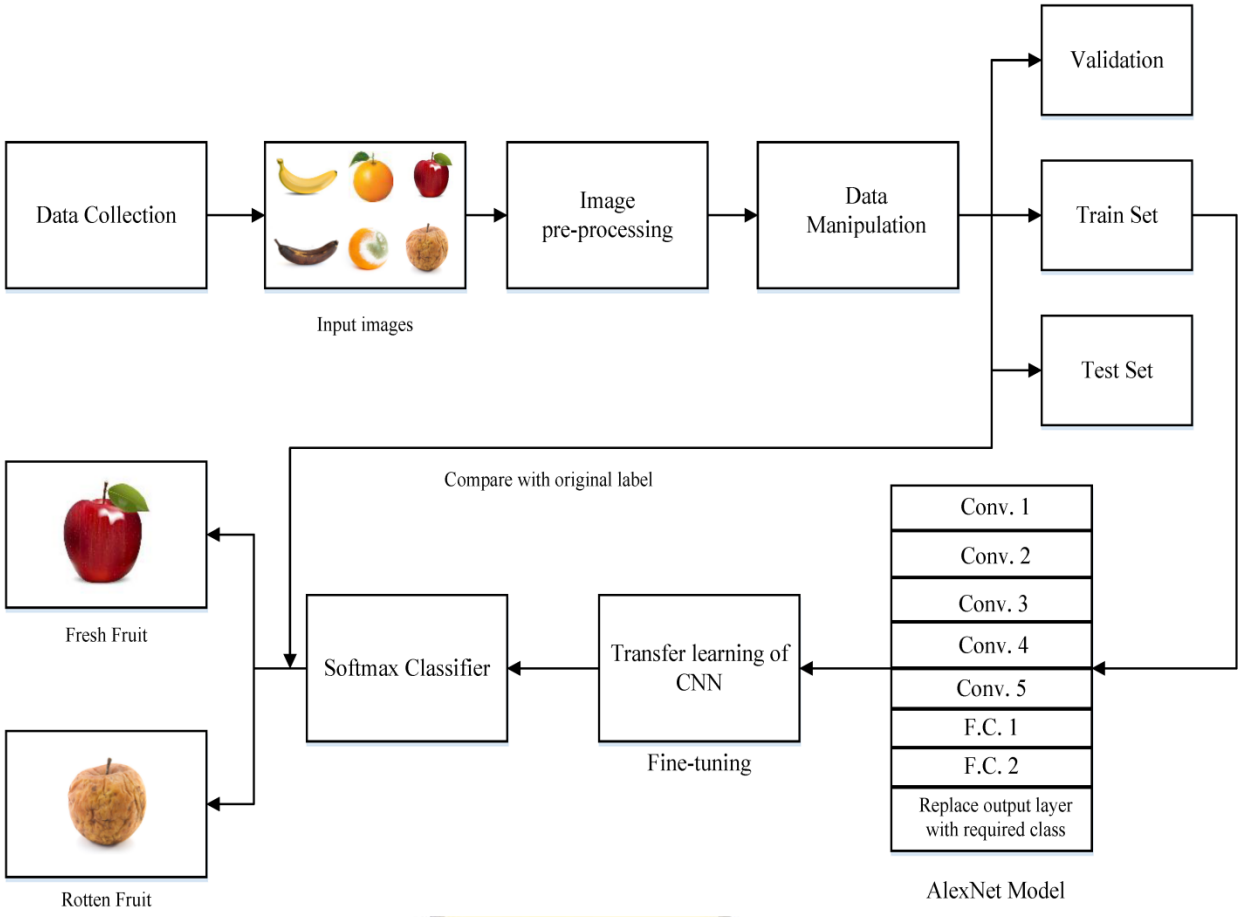
- ✓ Reduce dependency on manual labor in sorting processes.
- ✓ Improve classification accuracy and reliability.
- ✓ Minimize training time and computational requirements using transfer learning.
- ✓ Enable real-time identification suitable for industrial applications.
- ✓ Provide a scalable solution that can be extended to additional categories.

Gayathri Educational Society
ESTD-2001

SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

DATE	20-02-2026
TEAM ID	LTVIP2026TMIDS90622
PROJECT NAME	SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES
MAXIMUM MARKS	4 MARKS

2.2 EMPATHY MAP CANVAS



Gayathri Educational Society
ESTD-2001

SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

DATE	20-02-2026
TEAM ID	LTVIP2026TMIDS90622
PROJECT NAME	SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES
MAXIMUM MARKS	4 MARKS

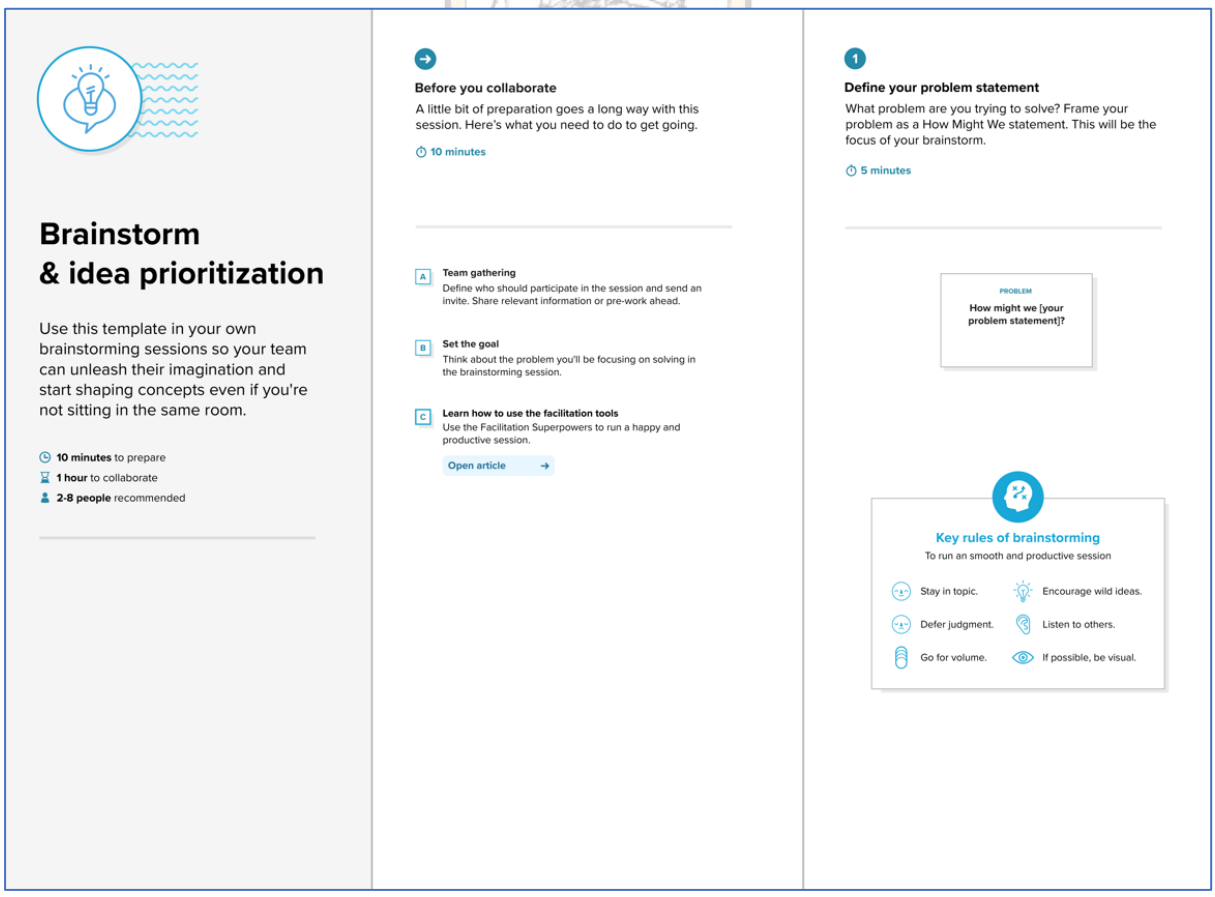
2.3 – Brainstorming

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

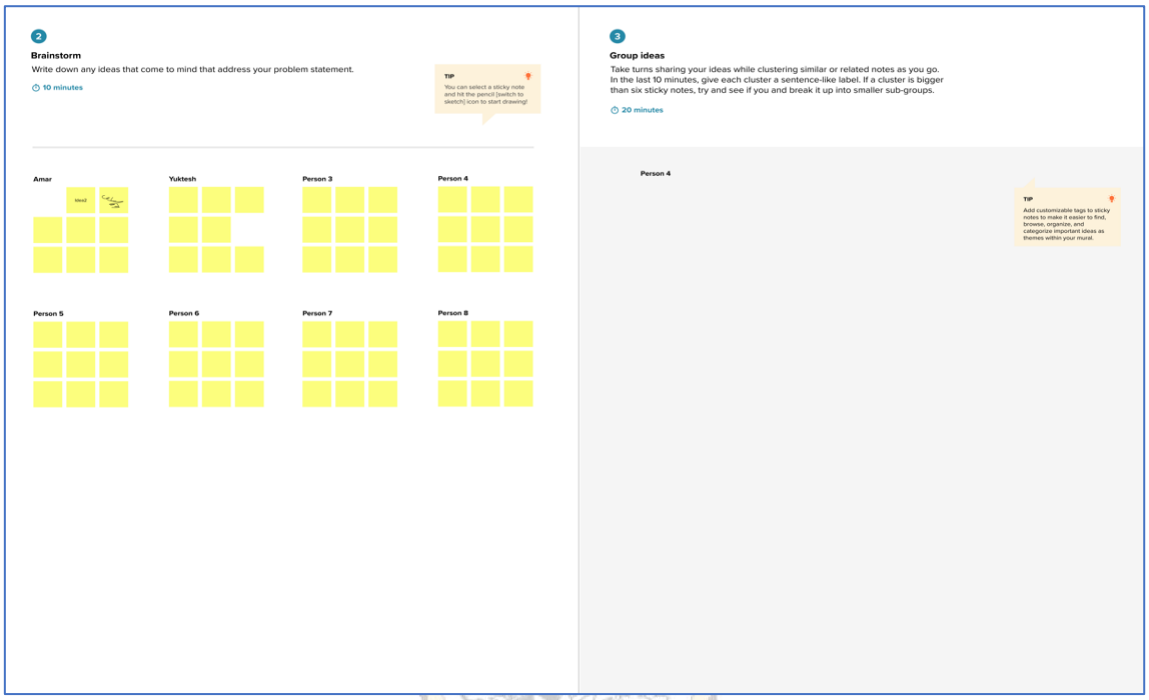
Reference: https://www.mural.co/templates/brainstorm-and-idea-prioritization

Step-1: Team Gathering, Collaboration and Select the Problem Statement

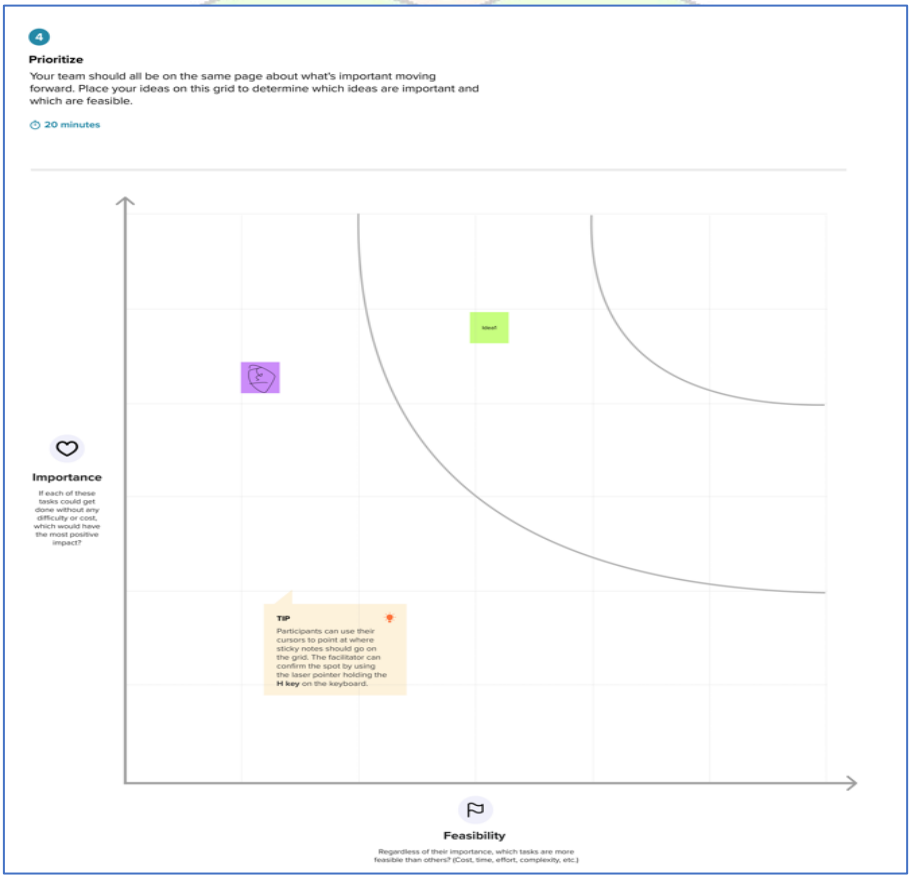


SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

Step-2: Brainstorm, Idea Listing and Grouping



Step-3: Idea Prioritization

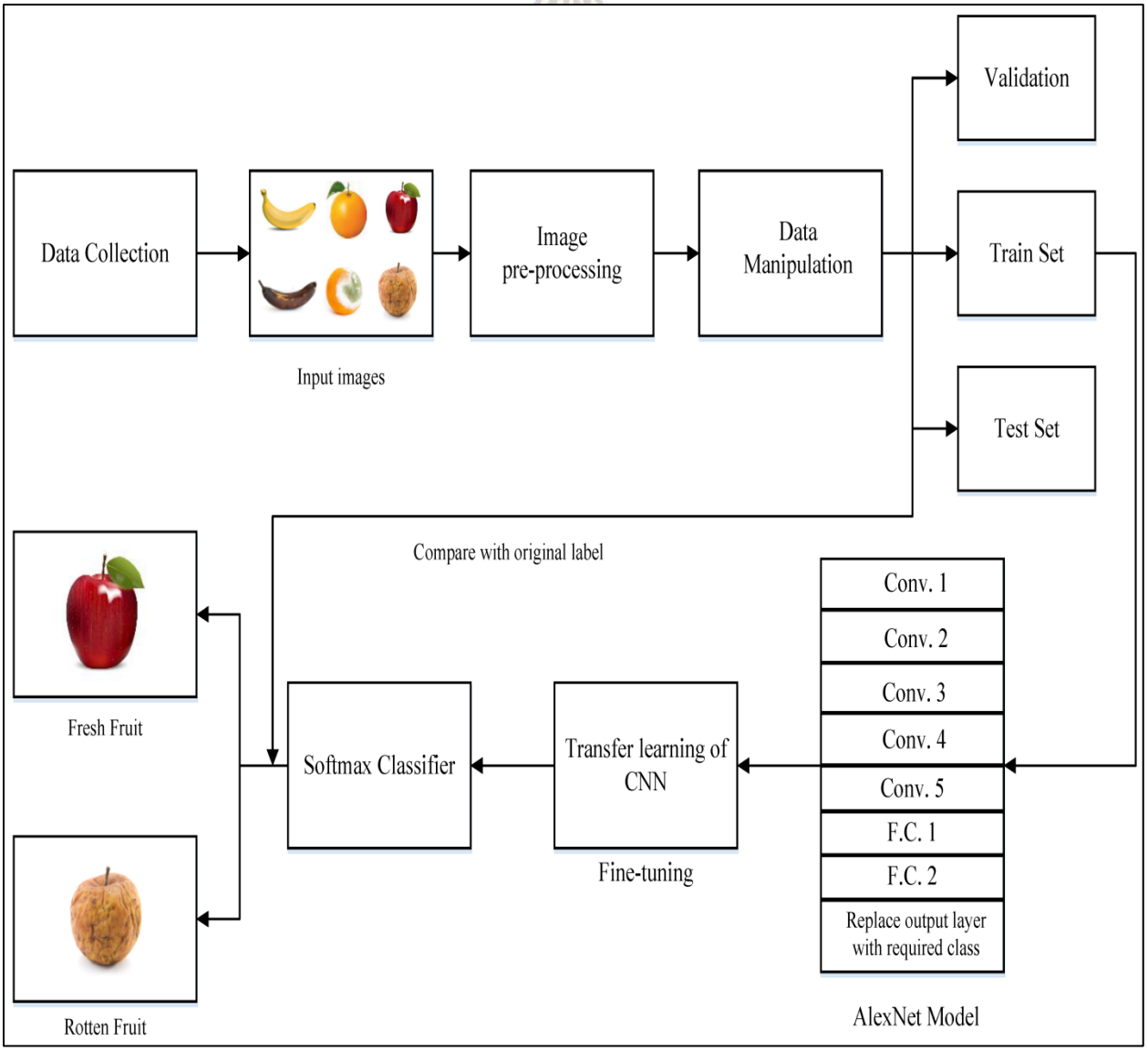


SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

DATE	20-02-2026
TEAM ID	LTVIP2026TMIDS90622
PROJECT NAME	SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES
MAXIMUM MARKS	4 MARKS

CHAPTER-3

3.1 CUSTOMER JOURNEY MAP



SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

DATE	20-02-2026
TEAM ID	LTVIP2026TMIDS90622
PROJECT NAME	SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES
MAXIMUM MARKS	2 MARKS

3.2 SOLUTION REQUIREMENT

The proposed solution requires a combination of hardware, software, and intelligent algorithms to accurately identify rotten and fresh fruits and vegetables in real time. The system must be capable of capturing high-quality images, processing them using deep learning models, and providing reliable output within a few seconds to ensure smooth user interaction.

From a hardware perspective, the system requires a high-resolution camera to capture detailed images of fruits and vegetables. Since rotten detection depends on identifying subtle discoloration, mold spots, and texture changes, image clarity is critical. A weight sensor or load cell is also required to measure the weight for price calculation. A processing unit such as a computer, laptop, or embedded device is necessary to run the trained deep learning model. For automated environments, additional components like a conveyor belt, servo motors, and a display screen are required to enable sorting and user interaction.

From a software perspective, the system must be developed using deep learning frameworks such as TensorFlow or Keras. The model will use transfer learning by fine-tuning pre-trained convolutional neural networks to classify fruits into categories such as fresh, slightly damaged, or rotten. Image preprocessing techniques such as resizing, normalization, and augmentation are required to improve accuracy under different lighting conditions.

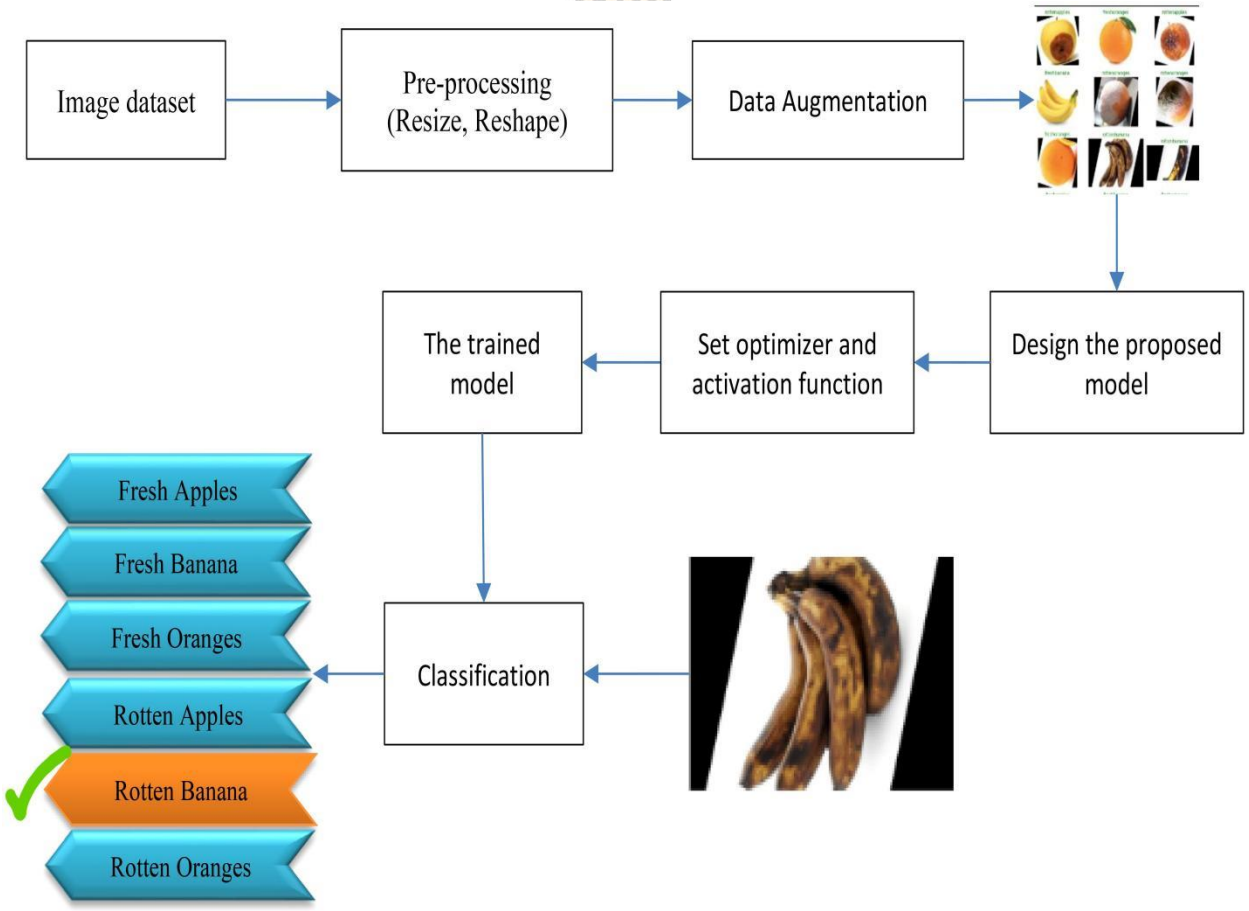
The solution also requires a well-structured dataset containing images of both fresh and rotten fruits and vegetables captured under various angles and lighting environments. Data augmentation techniques must be applied to enhance model generalization and prevent overfitting. The system should include a database to store classification results, timestamps, and quality reports for future analysis.
Performance requirements include high classification accuracy, low processing time, and real-time response capability. The system should detect rotten items within seconds and provide a confidence score to ensure transparency. It must also allow manual override in case of incorrect prediction to maintain reliability.

User interface requirements include a simple display that shows the detected item name, freshness status, weight, and price clearly. Instructions should be easy to understand so that customers of all age groups can use the system comfortably.

SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

DATE	20-02-2026
TEAM ID	LTVIP2026TMIDS90622
PROJECT NAME	SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES
MAXIMUM MARKS	3 MARKS

3.3 DATA FLOW DIAGRAM



Gayathri Educational Society
ESTD-2001

SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

DATE	20-02-2026
TEAM ID	LTVIP2026TMIDS90622
PROJECT NAME	SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES
MAXIMUM MARKS	3 MARKS

3.4 TECHNOLOGY STACK

Hardware and Software Requirements:

Since your project is based on deep learning and transfer learning (using models like TensorFlow or Keras), the system needs proper hardware and software support for image processing and model training.

1.Hardware Requirements:

1. 1Development & Training System (Minimum Requirements):

Processor: Intel i5 / Ryzen 5 (or higher)
• RAM: 8 GB (16 GB recommended for smooth training)
• Storage: 256 GB SSD minimum (512 GB recommended)
• GPU (Optional but Recommended): NVIDIA GPU with CUDA support (e.g., GTX 1650 or above)
• Camera: HD USB Camera (for real-time fruit/vegetable image capture)

These specifications are sufficient for developing and testing transfer learning models like MobileNet, VGG16, or ResNet.

1.2 Deployment Hardware (Prototype Model):

If implementing a real-time smart sorting system:

Raspberry Pi 4 (4GB/8GB RAM)
• Pi Camera Module / USB Camera
• Conveyor Belt Setup (for industrial prototype)
• Servo Motors (for sorting mechanism)
• Power Supply Unit
• Display Monitor (Optional)

This setup helps automate fruit/vegetable classification and physical sorting.

Gayathri Educational Society
ESTD-2001

Software Requirements:

2.1 Operating System:

• Windows 10/11
• Linux (Ubuntu recommended)
• macOS (optional)

2.2 Programming Language:

• Python 3.8 or above

2.3 Development Tools:

• Anaconda (for environment management)
• PyCharm / VS Code / Jupyter Notebook

2.4 Python Libraries:

• NumPy – numerical operations
• Pandas – dataset handling
• Matplotlib & Seaborn – data visualization
• OpenCV – image processing
• Scikit-learn – evaluation metrics
• TensorFlow / Keras – deep learning and transfer learning

2.5 Pre-trained Models (Transfer Learning):

• MobileNet
• VGG16
• ResNet50

These models are pre-trained on large datasets and fine-tuned for fruit/vegetable classification.

Gayathri Educational Society
ESTD-2001

3.Additional Requirements:

• Kaggle dataset (Fruits & Vegetables Fresh/Rotten dataset)
• Internet connection (for downloading datasets & pre-trained models)
• CUDA & cuDNN (if using GPU acceleration)

DATE	20-02-2026
TEAM ID	LTVIP2026TMIDS90622
PROJECT NAME	SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES
MAXIMUM MARKS	5 MARKS

CHAPTER-4

4.1 PROBLEM SOLUTION FIT

The core problem identified in retail markets and supermarkets is the difficulty in accurately detecting rotten or damaged fruits and vegetables during sorting and billing. Customers often rely only on visual inspection, which is not always reliable. Some fruits may appear fresh externally but may have internal spoilage or early-stage fungal growth. This leads to customer dissatisfaction, food wastage, financial loss, and damage to store reputation. Store managers also face challenges in monitoring quality consistently and reducing the number of spoiled products displayed for sale.

The proposed Smart Sorting system directly addresses these issues by using computer vision and transfer learning to automatically detect rotten produce in real time. Instead of manual inspection alone, the system captures images through a camera and processes them using a deep learning model trained to recognize discoloration, texture changes, mold patterns, and surface damage. Frameworks such as TensorFlow enable the use of pre-trained convolutional neural networks that can be fine-tuned specifically for freshness detection. This ensures higher accuracy and faster processing compared to traditional methods.

The solution fits the problem because it targets the exact pain points experienced by both customers and retailers. Customers gain confidence in the quality of products they purchase, reducing the risk of buying spoiled items. Retailers benefit from reduced complaints, lower wastage, improved inventory control, and enhanced brand trust. The integration of weight sensors and automatic billing further improves efficiency and minimizes human error.

Gayathri Educational Society

From a technical perspective, transfer learning makes the solution practical and scalable. Instead of requiring massive datasets and expensive infrastructure, the system leverages existing pre-trained models and adapts them to the specific use case of rotten fruit detection. This reduces development cost and implementation time, making the solution feasible for small and large retail environments.

SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

DATE	20-02-2026
TEAM ID	LTVIP2026TMIDS90622
PROJECT NAME	SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES
MAXIMUM MARKS	5 MARKS

4.2 PROPOSED SOLUTION

The proposed solution is an intelligent image-based classification system that automatically detects and separates fresh and rotten fruits and vegetables using transfer learning. The system leverages pre-trained deep learning models to accurately identify spoilage patterns such as discoloration, mold growth, texture deformation, and surface damage. This approach reduces food wastage, improves quality control, and minimizes manual inspection errors in markets, supermarkets, warehouses, and food processing units.

The core idea is to use a pre-trained Convolutional Neural Network model such as MobileNetV2, ResNet50, or VGG16 trained on large-scale datasets like ImageNet. These models already understand general image features such as edges, textures, and shapes. We fine-tune them on a custom dataset containing labeled images of fresh and rotten fruits and vegetables. By retraining only the final layers, the system adapts to the specific classification problem while requiring less data and computational power.

The solution architecture consists of four main components. First, image acquisition using a camera or mobile device captures fruit and vegetable images in real time. Second, image preprocessing techniques such as resizing, normalization, noise reduction, and augmentation improve image quality and model robustness. Third, the transfer learning model performs feature extraction and classification into categories such as fresh apple, rotten apple, fresh tomato, rotten tomato, and so on. Finally, the prediction output triggers an action, such as displaying the result on a dashboard or activating an automated sorting mechanism.

After training, the model is deployed as a web or mobile application. The front end allows users to upload or capture images, while the backend processes the image and returns the classification result. For industrial implementation, the system can be integrated with IoT-based conveyor belts where a camera continuously captures images and the model controls mechanical arms to separate rotten items automatically.

The proposed solution offers multiple advantages. It reduces dependency on manual inspection, improves accuracy and consistency, operates in real time, and is scalable for different types of produce. Since transfer learning uses pre-trained models, it significantly reduces training time and computational cost compared to building a model from scratch.

Gayathri Educational Society
ESTD-2001

DATE	20-02-2026
TEAM ID	LTVIP2026TMIDS90622
PROJECT NAME	SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES
MAXIMUM MARKS	5 MARKS

4.3 SOLUTION ARCHITECTURE

The solution architecture is designed as a modular and scalable deep learning system that integrates image acquisition, preprocessing, transfer learning-based classification, and automated sorting or result display. The architecture ensures real-time detection, high accuracy, and easy deployment in both retail and industrial environments.

1. Image Acquisition Layer

This is the input layer of the system where fruit and vegetable images are captured using cameras, mobile devices, or conveyor belt-mounted vision systems. In real-time industrial setups, a high-resolution camera continuously captures images as items move on a conveyor. In mobile or web-based applications, users can upload images directly through the interface. The captured images are forwarded to the preprocessing module.

2. Data Preprocessing Layer

The preprocessing layer prepares raw images for model input. Images are resized to match the required input dimensions of the pre-trained model (for example, 224×224 pixels). Normalization is applied to scale pixel values. Noise removal and contrast enhancement techniques improve image clarity. Data augmentation such as rotation, flipping, zooming, and brightness variation helps improve model generalization. This step ensures consistent input quality for accurate classification.

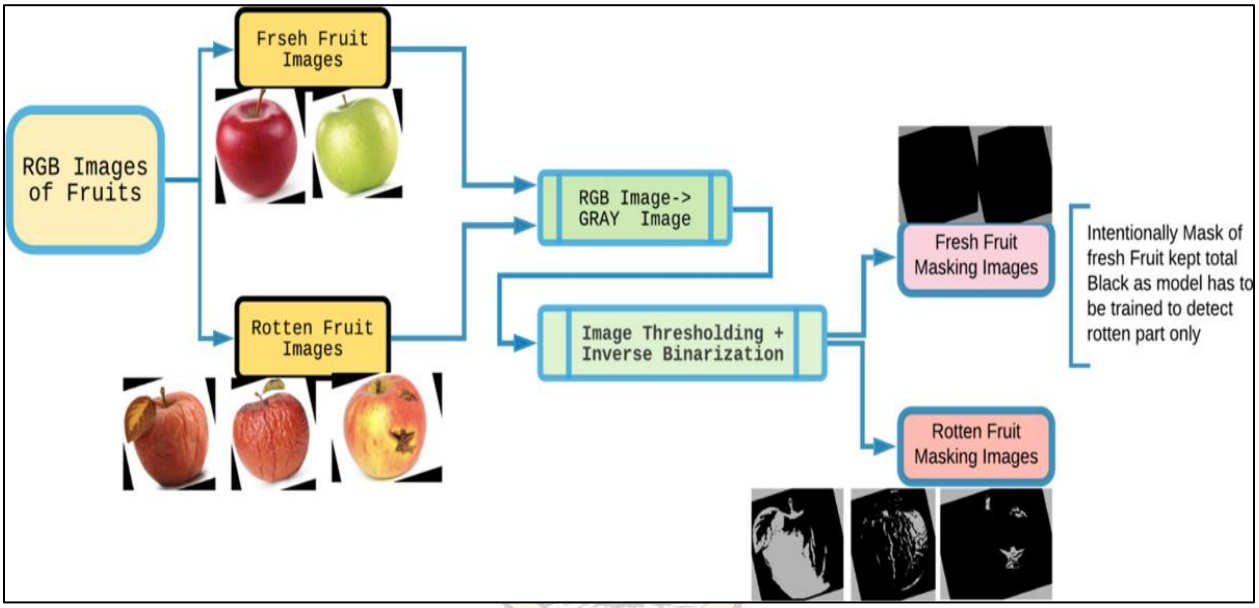
3. Feature Extraction and Transfer Learning Layer

This is the core intelligence layer of the system. A pre-trained Convolutional Neural Network such as MobileNetV2, ResNet50, or VGG16 is used as the base model. These models are originally trained on large datasets like ImageNet and can detect complex visual patterns.

4. Classification Layer

The customized output layer classifies images into predefined categories such as Fresh Apple, Rotten Apple, Fresh Tomato, Rotten Tomato, etc. A Softmax activation function is

used to produce probability scores for each class. The class with the highest probability is selected as the final prediction. Confidence scores can also be displayed for reliability analysis.



5. Decision and Control Layer

Based on the classification result, the system triggers appropriate actions. In a software-based implementation, the result is displayed on a web or mobile dashboard. In an industrial implementation, the output is connected to a microcontroller system that controls actuators or robotic arms to separate rotten items from fresh ones automatically.

6. Deployment Layer

The trained model is deployed using frameworks such as TensorFlow or PyTorch. The application can be hosted on cloud platforms for scalability or deployed on edge devices for faster real-time processing. Edge deployment ensures low latency and reduced dependency on internet connectivity.

Overall Architecture Flow

Image Capture → Image Preprocessing → Feature Extraction (Pre-trained CNN) → Fine-Tuned Classification → Prediction Output → Automated Sorting or Display

Gayathri Educational Society

SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

DATE	20-02-2026
TEAM ID	LTVIP2026TMIDS90622
PROJECT NAME	SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES
MAXIMUM MARKS	1 MARKS

Chapter – 5

Project Planning & Scheduling

5.1 - Project Milestones & Tasks

1. Data Collection

Objective: Gather high-quality, relevant data required for the project.

Tasks:

- Identify reliable data sources (databases, APIs, web scraping, surveys, etc.)
- Collect structured and unstructured data
- Ensure data relevance and completeness
- Store data in a centralized database or storage system
- Maintain data documentation for reference

Deliverables:

- Raw dataset
- Data source documentation
- Data storage setup

---

2. Data Pre-Processing

Objective: Clean and prepare the data for model training and analysis.

Tasks:

- Remove duplicates and handle missing values
- Handle outliers and inconsistent data
- Data normalization or scaling
- Feature engineering and selection
- Encode categorical variables
- Split dataset into training and testing sets

Gayathri Educational Society
ESTD 2001

Deliverables:

- Cleaned dataset
- Feature-engineered dataset
- Preprocessing scripts

3️⃣ Model Building

Objective: Develop and train a machine learning model.

Tasks:

- Select appropriate algorithm(s)
- Train model using training dataset
- Hyperparameter tuning
- Model validation and evaluation
- Compare performance metrics
- Finalize best-performing model

Deliverables:

- Trained model
- Evaluation report (Accuracy, Precision, Recall, F1-score, etc.)
- Saved model file

4️⃣ API Integration

Objective: Expose the trained model via an API for external access.

Tasks:

- Develop REST API endpoints
- Integrate model with backend framework (Flask / FastAPI / Django)
- Implement request validation
- Add error handling
- Test API responses
- Secure API (authentication if required)

Deliverables:

- Functional API

Gayathri Educational Society
ESTD-2001

- API documentation (Swagger/Postman collection)
- Deployment-ready backend

5️⃣ Web Integration

Objective: Integrate API into a web-based user interface.

Tasks:

- Design user-friendly frontend interface
- Connect frontend to backend API
- Display model predictions dynamically
- Implement input validation
- Perform end-to-end testing
- Deploy web application

Deliverables:

- Fully functional web application
- User interface design
- Live deployment link (if applicable)

Gayathri Educational Society
ESTD-2001

SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

DATE	20-02-2026
TEAM ID	LTVIP2026TMIDS90622
PROJECT NAME	SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES
MAXIMUM MARKS	2 MARKS

5.2 - Sprint Delivery Plan

🔷 Phase 1: Live Sessions (Week 1–6)

Objective: Build strong foundational knowledge and prepare interns for real-time project development.

□ Sprint 1 (Week 1–2): Fundamentals & Tools

Focus Areas:

- Introduction to Internship Program
- Programming Fundamentals (Python / Relevant Tech Stack)
- Git & GitHub
- Development Environment Setup
- Basics of Databases

Deliverables:

- Setup development environment
- GitHub repository creation
- Mini practice assignments

□ Sprint 2 (Week 3–4): Data & Backend Foundations

Focus Areas:

- Data Handling (Pandas / Data Structures)
- Data Cleaning Techniques
- Introduction to APIs
- Backend Basics (Flask / FastAPI)
- SQL & Database Integration

Gayathri Educational Society
ESTD-2001

SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

Deliverables:

- Data preprocessing assignment
- Basic API development task
- Database connectivity demo

□ Sprint 3 (Week 5–6): Machine Learning & Deployment Basics

Focus Areas:

- Machine Learning Fundamentals
- Model Training & Evaluation
- REST API Integration with Model
- Introduction to Web Integration
- Deployment Overview

Deliverables:

- Simple ML Model
- Model evaluation report
- API with working prediction endpoint

◈ Phase 2: Project Work (Week 7–15)

Objective: Apply learned skills to build a complete end-to-end project.

□ Sprint 4 (Week 7–8): Project Planning & Data Collection

Activities:

- Finalize project topic
- Define problem statement
- Collect dataset
- Perform initial data analysis (EDA)

Deliverables:

Gayathri Educational Society
ESTD-2001

- Project proposal document
- Dataset documentation
- EDA report

□ Sprint 5 (Week 9–10): Data Preprocessing & Feature Engineering

Activities:

- Clean dataset
- Handle missing values & outliers
- Feature engineering
- Data transformation
- Train-test split

Deliverables:

- Cleaned dataset
- Preprocessing pipeline
- Feature documentation

□ Sprint 6 (Week 11–12): Model Development & Optimization

Activities:

- Train multiple models
- Hyperparameter tuning
- Model comparison
- Performance evaluation

Deliverables:

- Best performing model
- Evaluation metrics report
- Saved model artifact

□ Sprint 7 (Week 13–14): API Development & Integration

Gayathri Educational Society
ESTD-2001

SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

Activities:

- Develop REST API
- Integrate trained model
- Implement validation & error handling
- Test API endpoints

Deliverables:

- Functional API
- API documentation
- Backend deployment

□ Sprint 8 (Week 15): Web Integration & Final Deployment

Activities:

- Develop frontend interface
- Connect frontend with API
- End-to-end testing
- Deployment & final presentation

Deliverables:

- Fully functional web application
- Deployment link
- Final project presentation
- Internship completion report

Final Outcome

By the end of 15 weeks, interns will have:

- Strong technical foundation
- Real-time project experience
- A complete end-to-end deployed project
- Industry-ready portfolio project

Gayathri Educational Society
ESTD-2001

SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

DATE	20-02-2026
TEAM ID	LTVIP2026TMIDS90622
PROJECT NAME	SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES
MAXIMUM MARKS	1 MARKS

5.3 - Project Progress Tracking

1. Zoho Cliq Workspace Structure

◈ Channels Setup

Create the following channels for organized communication:

📌 1. #announcements

- Official updates
- Sprint start/end notifications
- Deadlines & evaluation updates
- Meeting schedules

📌 2. #project-discussion

- Technical queries
- Implementation discussions
- Code review discussions
- Issue troubleshooting

📌 3. #daily-updates

- Daily progress reports
- Blockers
- Completed tasks

📌 4. #resources

- Shared datasets
- Documentation links
- Recorded session links
- API documentation

Gayathri Educational Society
ESTD-2001

📌 **5. #team-specific-channels (if multiple teams)**

Example:

- #team-alpha
- #team-beta

📆 **2. Sprint-Based Tracking Method**

Each sprint will follow a structured reporting cycle.

□ Daily Progress Update Format (Posted in #daily-updates)

Every intern must post:

Format:

Date:

Sprint:

Tasks Completed:

Tasks In Progress:

Blockers (if any):

Plan for Tomorrow:

🟤 **Weekly Sprint Review (Every Weekend)**

Mentor will post:

- Sprint Goals
- Completed Milestones
- Pending Tasks
- Risk Areas
- Next Week Targets

📋 **3. Task Tracking System**

Option 1: Zoho Cliq Tasks Feature

Gayathri Educational Society
ESTD-2001

Use built-in task management in Zoho Cliq:

- Assign tasks to interns
- Set deadlines
- Track completion status
- Add task priority (High / Medium / Low)

Task Status Workflow:

- **□ To Do**
- **In Progress**
- **□ Completed**
- **Blocked**

Option 2: Zoho Projects Integration (Optional)

For advanced tracking, integrate with:

- Zoho Projects

Use:

- Kanban Board
- Gantt Chart
- Milestone tracking
- Automated reminders

---

4. Sprint Progress Monitoring Dashboard

Track the following metrics weekly:

- % Tasks Completed
- API Development Progress
- Model Accuracy Improvement
- Deployment Readiness
- Bug Count
- Attendance in Live Sessions

Gayathri Educational Society
ESTD-2001

Mentor shares a weekly progress summary in:

👉 #announcements channel

☑ 5. Milestone Tracking Structure

Milestone	Week	Status	Owner	Remarks
Data Collection	Week 7–8	☐	Team	
Data Preprocessing	Week 9–10	☐	Team	
Model Building	Week 11–12	☐	Team	
API Integration	Week 13–14	☐	Team	
Web Integration	Week 15	☐	Team	

🛎 6. Escalation Process

If blocker > 24 hours:

- **Post in #project-discussion**
- **Tag mentor**
- **If unresolved → Schedule quick call via Zoho Cliq**
- **Update resolution summary in channel**

🏆 7. Performance Evaluation Criteria

Evaluation will be based on:

- Daily update consistency
- Sprint milestone completion
- Code quality
- Participation in discussions
- Final project delivery
- Timely submissions

Gayathri Educational Society
ESTD-2001

SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

Weekly Workflow Summary

Day	Activity
Monday	Sprint planning post
Tue–Thu	Development & daily updates
Friday	Progress review
Saturday	Sprint demo
Sunday	Feedback & planning

Final Outcome

Using Zoho Cliq ensures:

- Clear communication
- Accountability
- Structured sprint tracking
- Real-time problem resolution
- Professional workflow management

Gayathri Educational Society
ESTD-2001

SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

DATE	20-02-2026
TEAM ID	LTVIP2026TMIDS90622
PROJECT NAME	SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES
MAXIMUM MARKS	1 MARKS

5.4 - Team Management Tools for Agile Planning

□ What is Jira?

Jira is an Agile project management and issue-tracking tool developed by Atlassian. It supports Scrum and Kanban methodologies, helping teams plan, track, and release software efficiently.

Jira Project Structure for Internship

1□Project Creation

Create a project with:

- Project Name: Internship Capstone Project
- Template: Scrum (Recommended)
- Project Type: Software Development

Issue Types Configuration

Define standard issue types:

- □ Epic – Major project phases
- Story – Feature or functionality
- □ Task – Smaller implementation steps
- Bug – Errors or defects
- ◯ Sub-task – Breakdown of tasks

Suggested Epics (Based on Your Milestones)

SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

Epic	Description
Data Collection	Dataset gathering & validation
Data Preprocessing	Cleaning & feature engineering
Model Development	ML training & evaluation
API Integration	Backend & model API
Web Integration	Frontend & deployment

Sprint Planning Structure

Sprint Duration

- **2 Weeks per Sprint**
- **Total: 4–5 Sprints (Project Phase)**

Sprint Workflow

1. **Create Sprint in Backlog**
2. **Add Stories/Tasks to Sprint**
3. **Assign Tasks to Interns**
4. **Set Priority (High/Medium/Low)**
5. **Estimate Story Points**
6. **Start Sprint**

Workflow Status Configuration

Customize workflow:

- □ To Do
- In Progress
- □ In Review
- □ Done
- Blocked

Gayathri Educational Society
ESTD-2001

This ensures transparent tracking of task movement.

📊 Agile Boards in Jira

1️⃣ Scrum Board

Used for:

- Sprint planning
- Daily standups
- Tracking sprint progress

2️⃣ Kanban Board (Optional)

Used for:

- Continuous workflow
- API & bug tracking

☑ Agile Reports for Monitoring

Jira provides built-in reports:

- 📊 Burndown Chart – Sprint progress tracking
- ☑ Velocity Chart – Team performance
- 📋 Sprint Report – Completed vs pending work
- 🐞 Bug Report – Defect tracking

Mentors can review reports weekly to evaluate progress.

Gayathri Educational Society
ESTD-2001

👥 Role-Based Access

Define permissions:

Role	Responsibilities
Project Admin	Configure board & workflow

Scrum Master	**Sprint planning & review**
Developer (Intern)	**Task implementation**
Reviewer	**Code & feature review**

□ Daily Standup Format (Using Jira Board)

Each intern updates:

- What was completed yesterday
- What will be done today
- Any blockers

Tasks must be moved across workflow stages accordingly.

Integration Capabilities

Jira can integrate with:

- GitHub (code tracking)
- CI/CD tools
- Slack or Zoho Cliq (notifications)
- Confluence (documentation)

Example Sprint Breakdown (2 Weeks)

Sprint Goal: Complete Data Preprocessing

Planned Stories:

- Clean missing values
- Outlier detection
- Feature scaling
- Train-test split

Expected Outcome:
Clean dataset ready for model training.

Gayathri Educational Society
ESTD-2001

✅ **Benefits of Using Jira**

- Structured Agile Planning
- Clear Accountability
- Transparent Sprint Tracking
- Performance Analytics
- Real-time Status Visibility
- Industry-standard project management tool

📌 **Final Outcome**

By implementing Jira for Agile planning, the internship project will follow:

- Scrum-based sprint execution
- Organized backlog management
- Efficient task delegation
- Measurable progress tracking
- Professional software development workflow

Gayathri Educational Society
ESTD-2001

SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

DATE	20-02-2026
TEAM ID	LTVIP2026TMIDS90622
PROJECT NAME	SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES
MAXIMUM MARKS	2 MARKS

CHAPTER-6

6.1 PRE-REQUISITES

To complete this project, you must require the following software, concepts, and packages

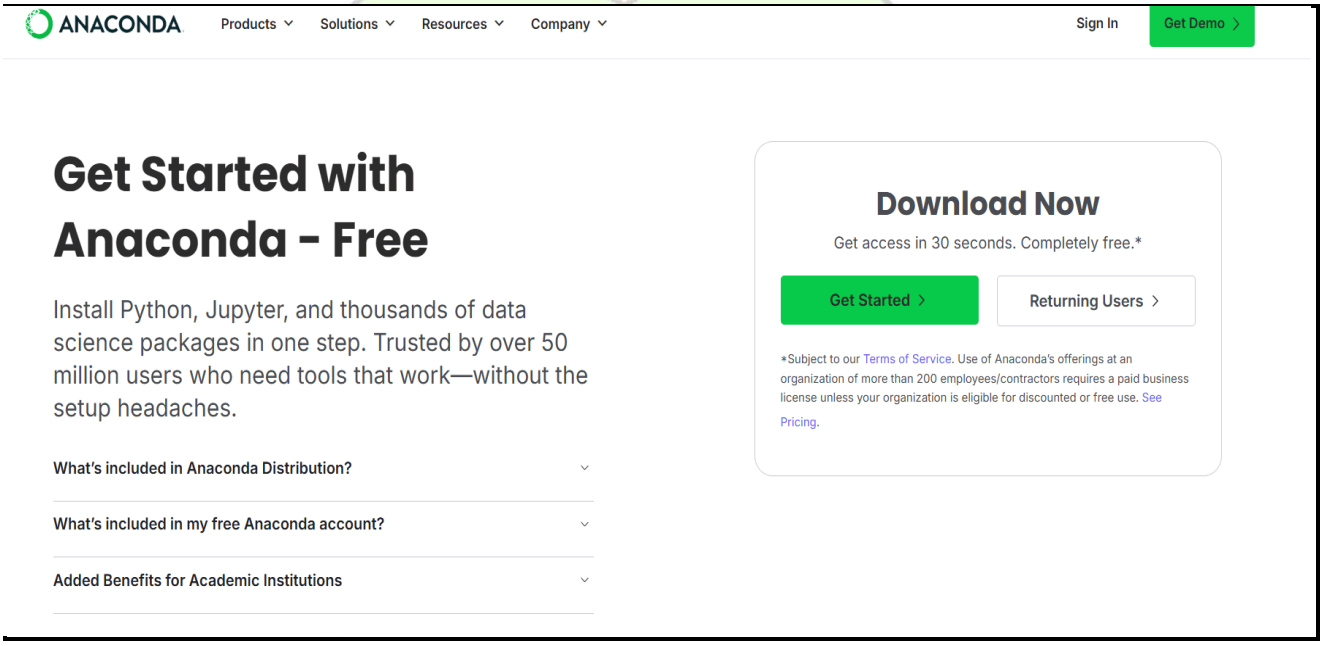
Anaconda Navigator:Refer to the link below to download Anaconda Navigator

How to Install Anaconda Navigator

Step 1: Download Anaconda

Open your web browser.

Go to: https://www.anaconda.com/download

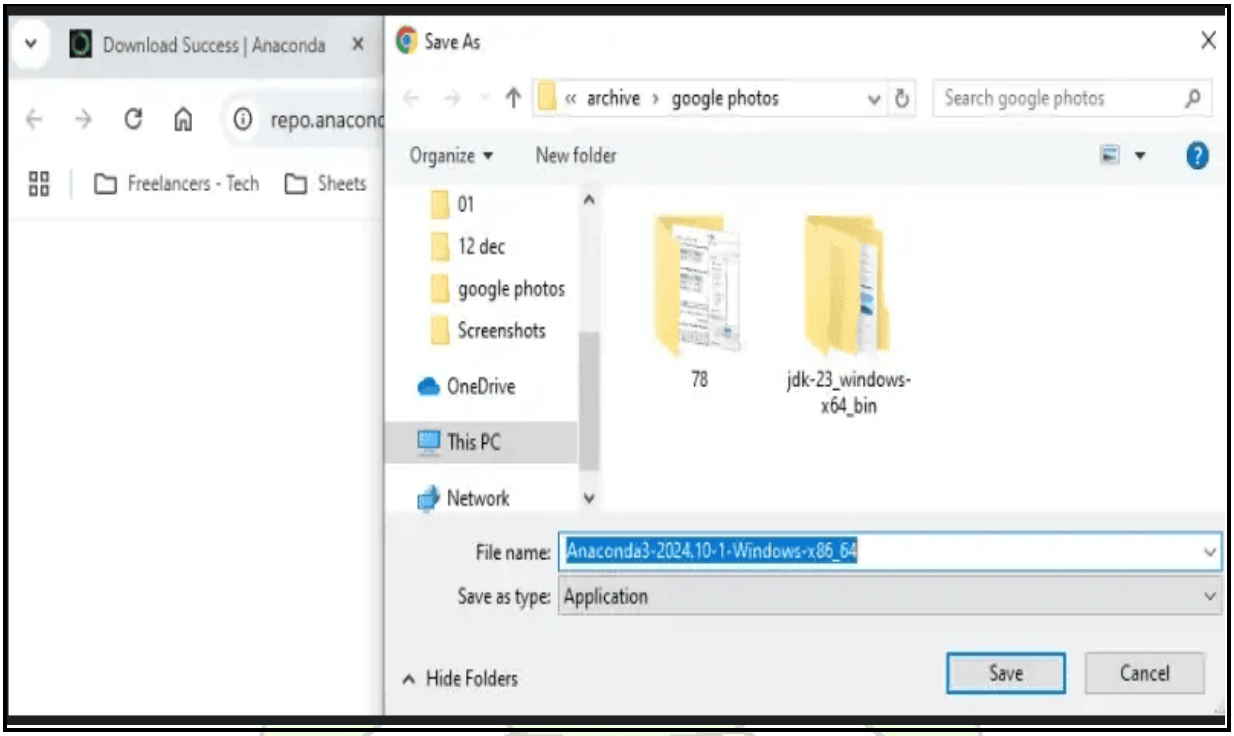


Select your Operating System:

- Windows

- MacOS
- Linux

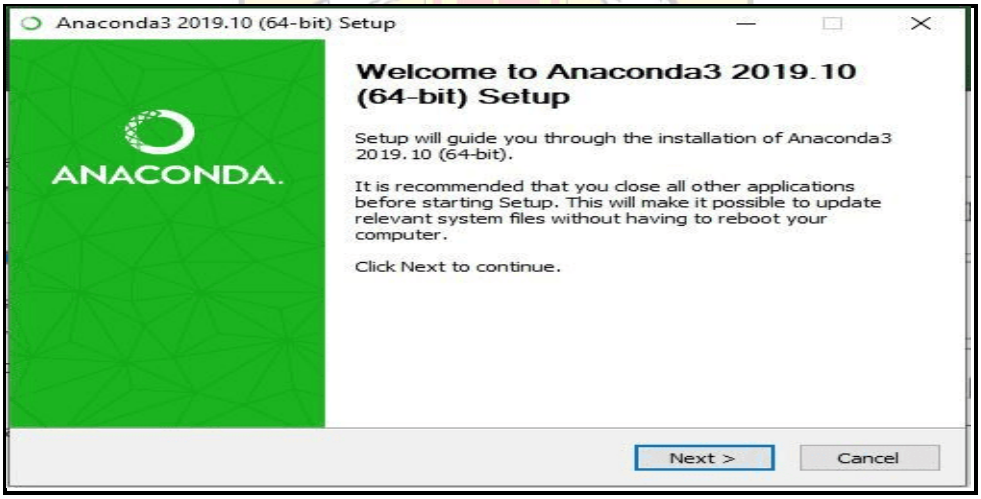
Download Anaconda Individual Edition (Free). Choose Python 3.x (64-bit) version (recommended).



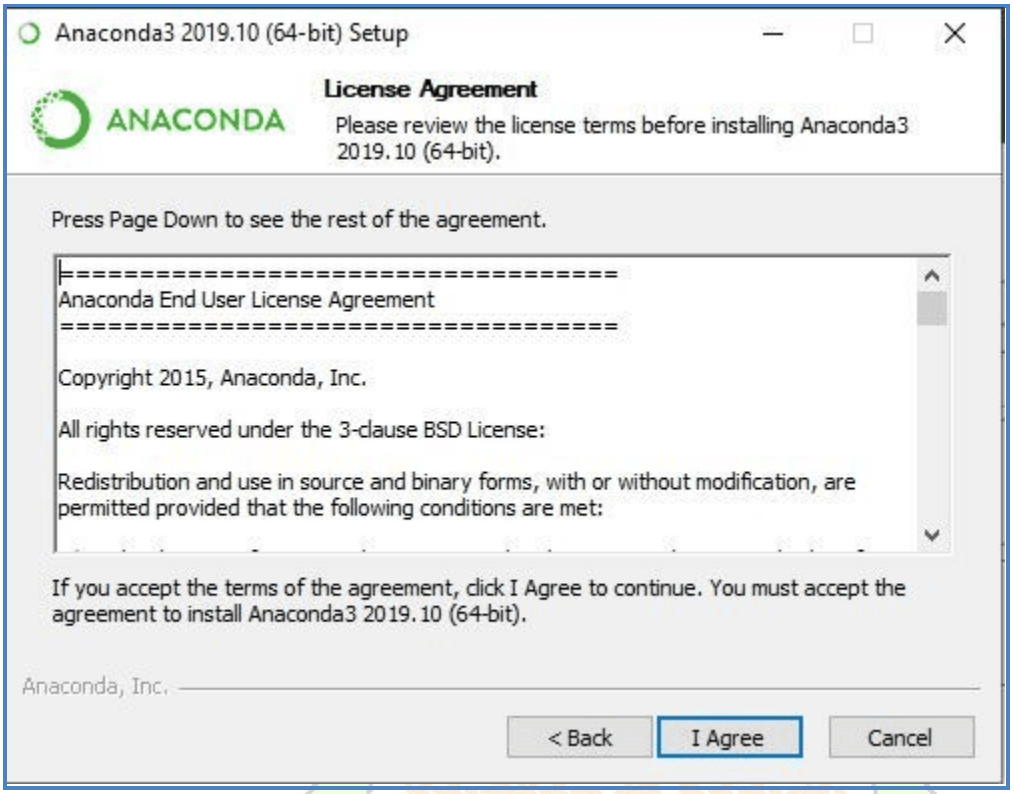
Step 2: Install Anaconda For Windows

Double-click the downloaded .exe file.

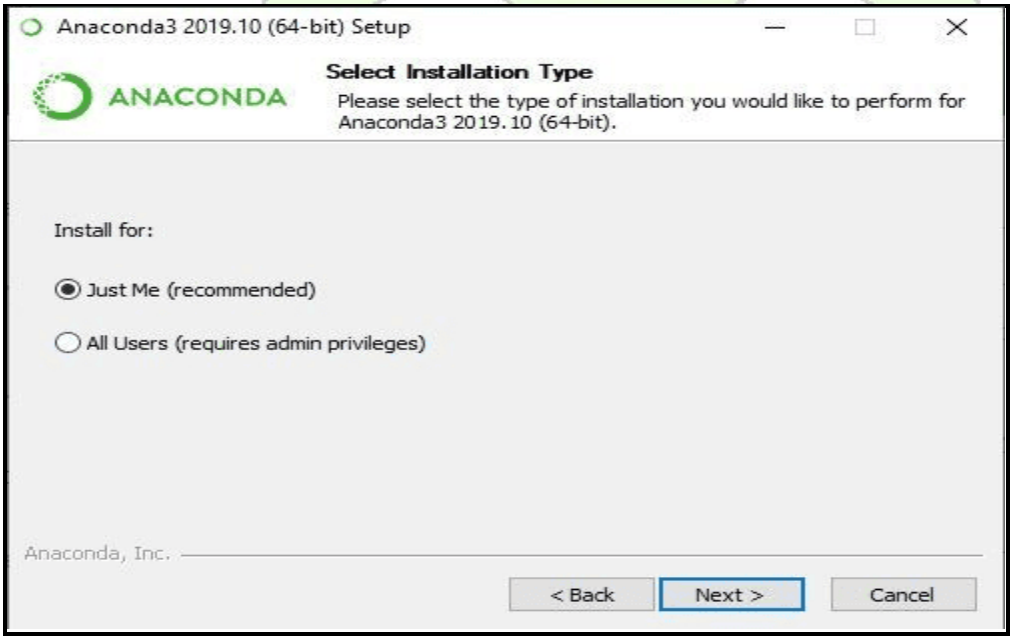
Click Next



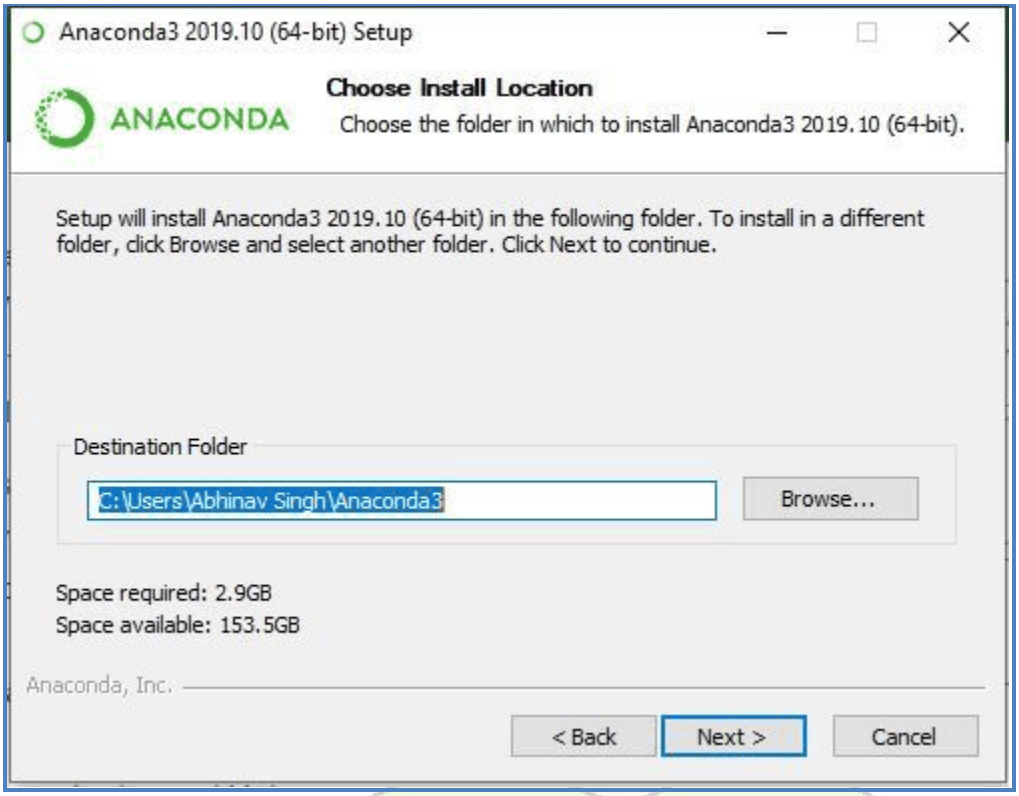
3. Click I agree to accept the license.



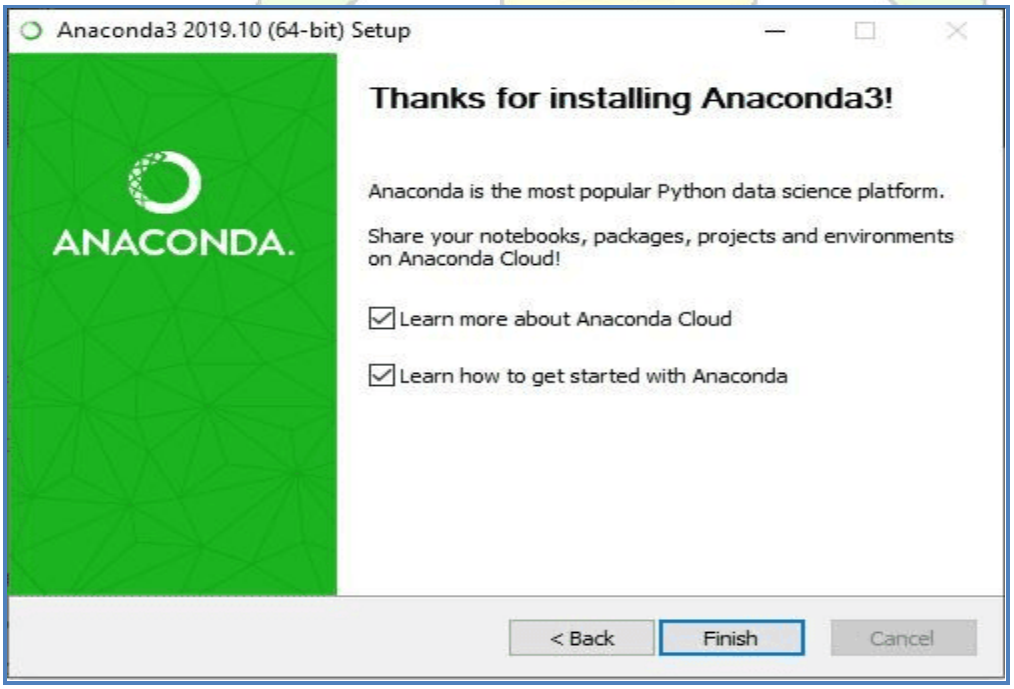
4. Select Just Me (recommended) → Click Next.



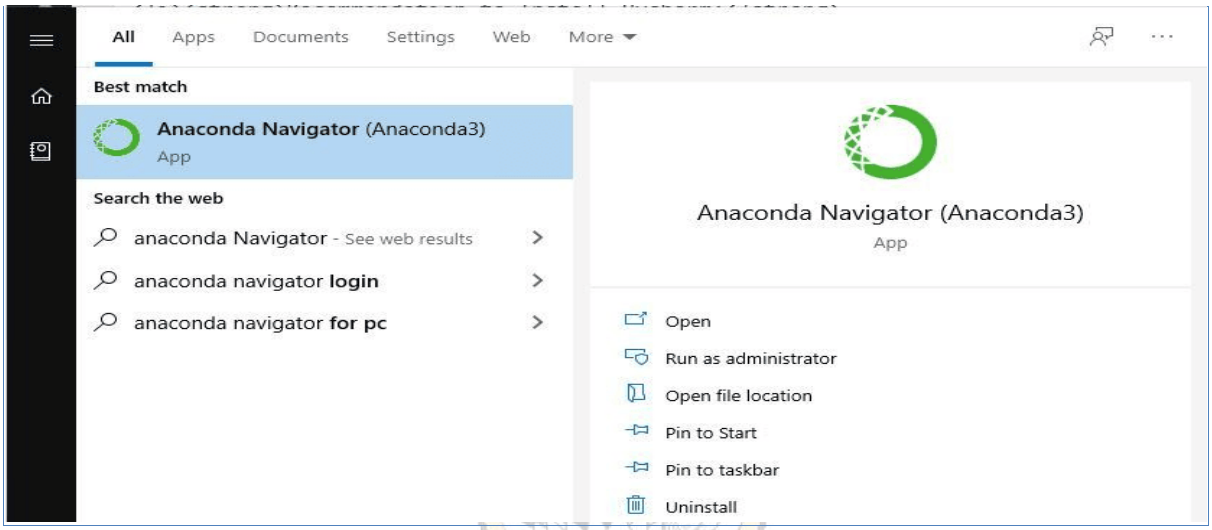
5. Choose the installation location where we need to install the anaconda software → Click Next.



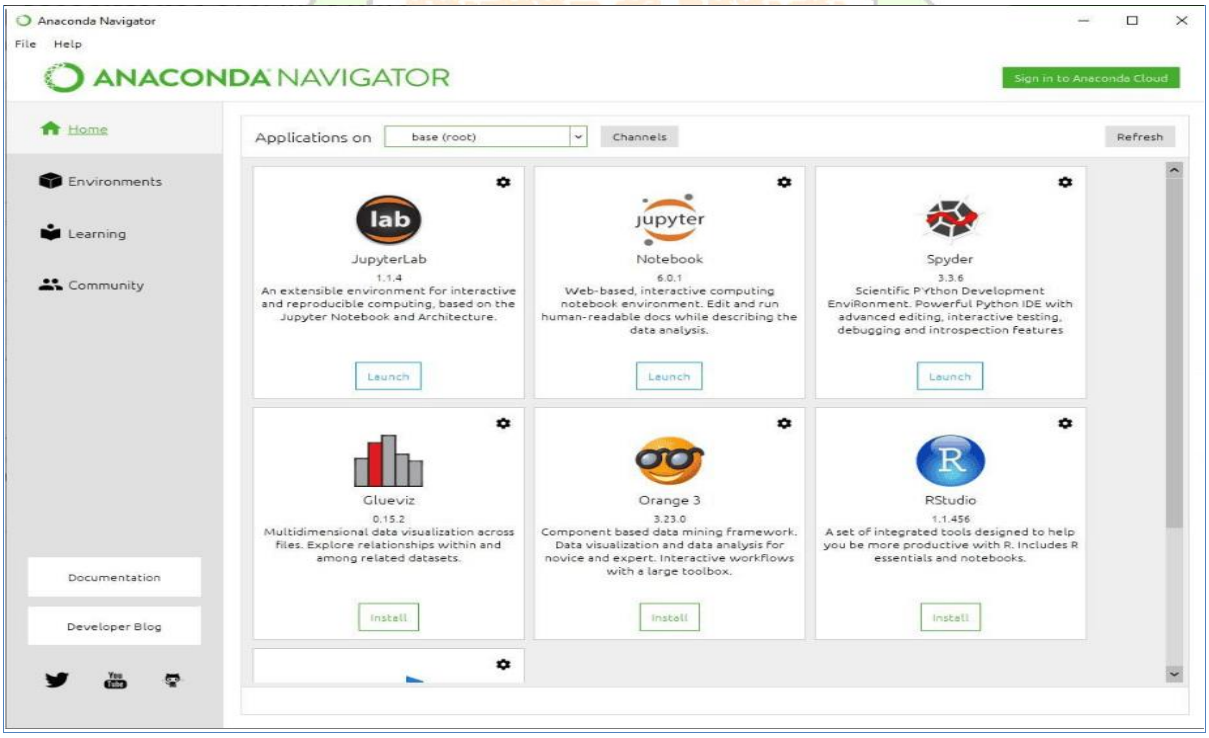
6. Check Add Anaconda to PATH (optional but useful).
7. Check Register Anaconda as default Python.
8. Click Install.



9. After installation completes, click Finish.
10. Access Anaconda Navigator
11. Once the installation process is done, Anaconda can be used to perform multiple operations. To begin using Anaconda, search for Anaconda Navigator from the Start Menu in Windows PC.



12. Explore Navigator & Features
13. You can use navigator to create new environments, install packages, and launch applications without using the command line.



How to install pycharm:

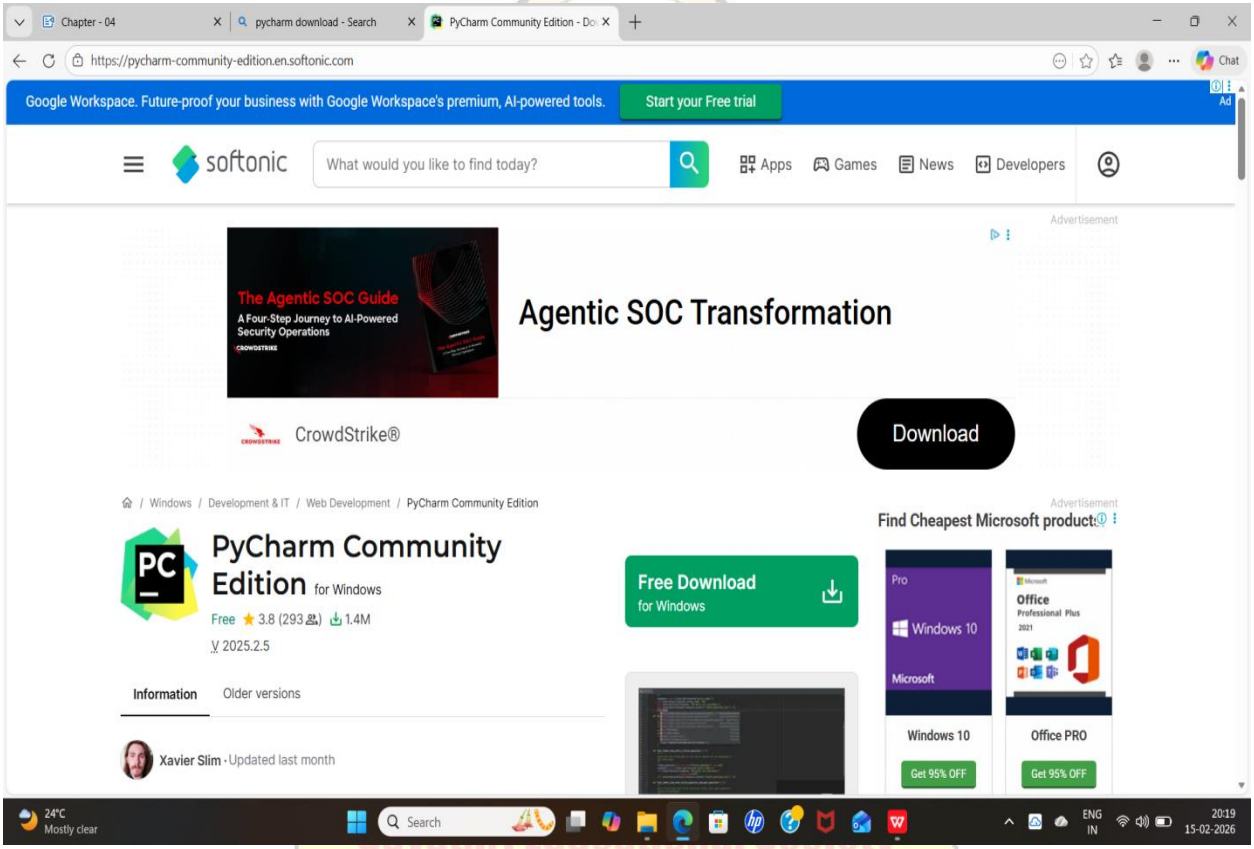
Installing PyCharm on Windows (Detailed Steps)

1. Go to the JetBrains PyCharm website

- Open any web browser (Chrome, Edge, Firefox, etc.).
- Type pycharm.jetbrains.com in the address bar and press Enter.

2. Click Download

- On the PyCharm homepage, you'll see a Download button.
- Click it to go to the download page.



3. Choose Community or Professional

You will see two versions of PyCharm:

PyCharm Community Edition (Free)

Choose this if:

- You are a beginner or student
- You are learning basic Python
- You are doing school projects, scripting, or simple programs

What it includes:

- Python editor
- Code suggestions (auto-complete)
- Debugger
- Basic project tools

4. Download the Windows (.exe) installer

- Under your chosen version, click Download for Windows.
- A file ending in .exe will start downloading.
- Wait for the download to complete.

HOW TO INSTALL THE PACKAGES :

To install the packages , open pycharm

Step-1: click on Terminal .

Step-2: Now, Run the package command one by one and they will start installing in the interpreter

Python packages:

Open anaconda prompt as administrator

Type “pip install numpy” and click enter.

Type “pip install pandas” and click enter.

Type “pip install scikit-learn” and click enter.

Type ”pip install matplotlib” and click enter.

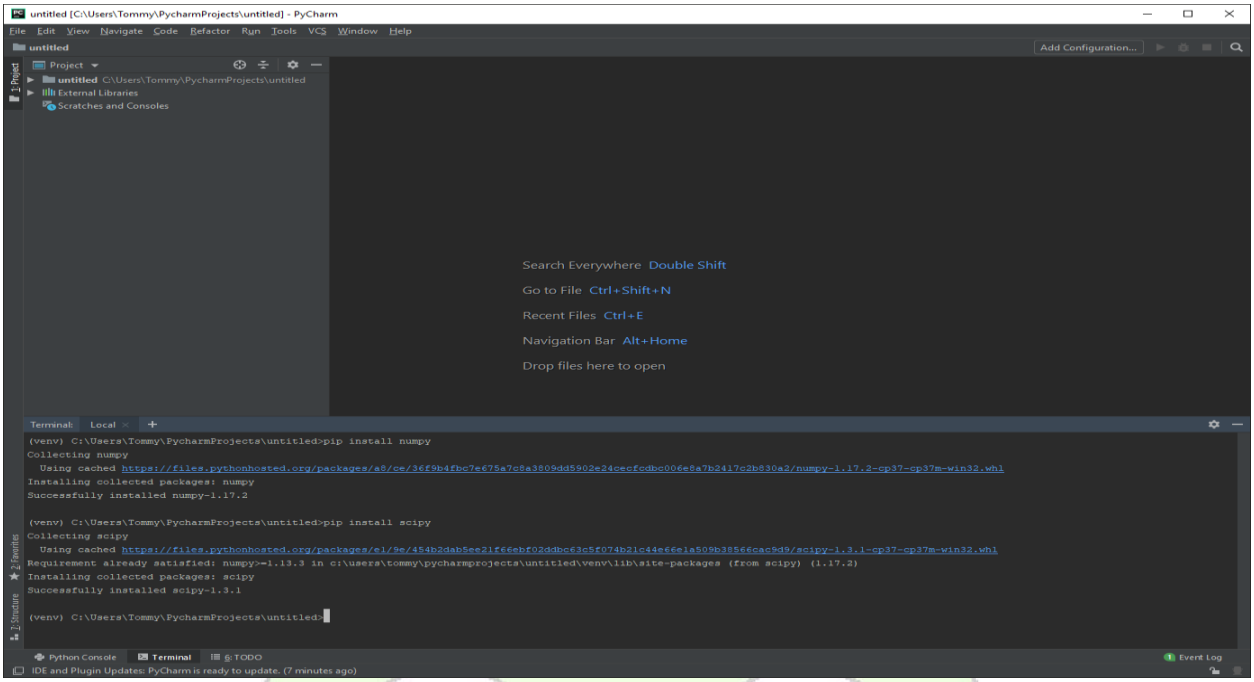
Type ”pip install scipy” and click enter.

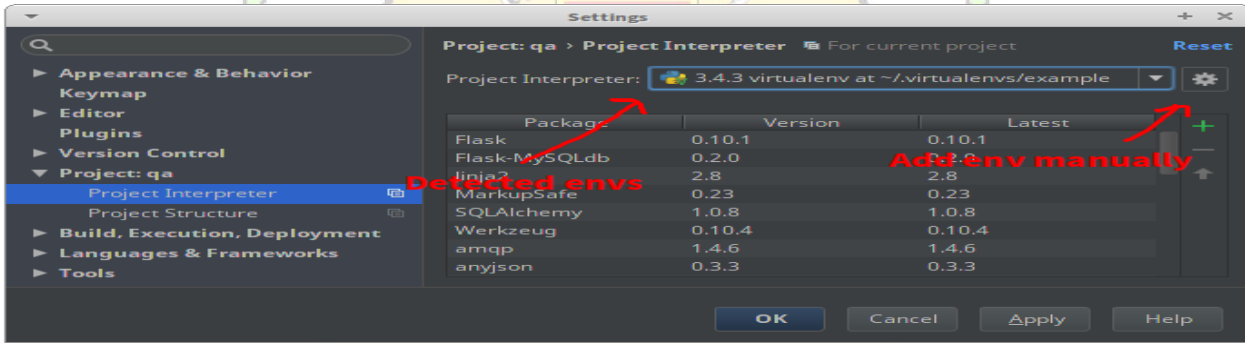
Sevathri Educational Society
ESTD-2001

Type ”pip install seaborn” and click enter.

Type ”pip install tensorflow” and click enter.

Type “pip install Flask” and click enter.





SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

DATE	20-02-2026
TEAM ID	LTVIP2026TMIDS90622
PROJECT NAME	SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES
MAXIMUM MARKS	2 MARKS

6.2 DATA COLLECTION

Collect the data-set:

It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

Activity 1: Download the dataset

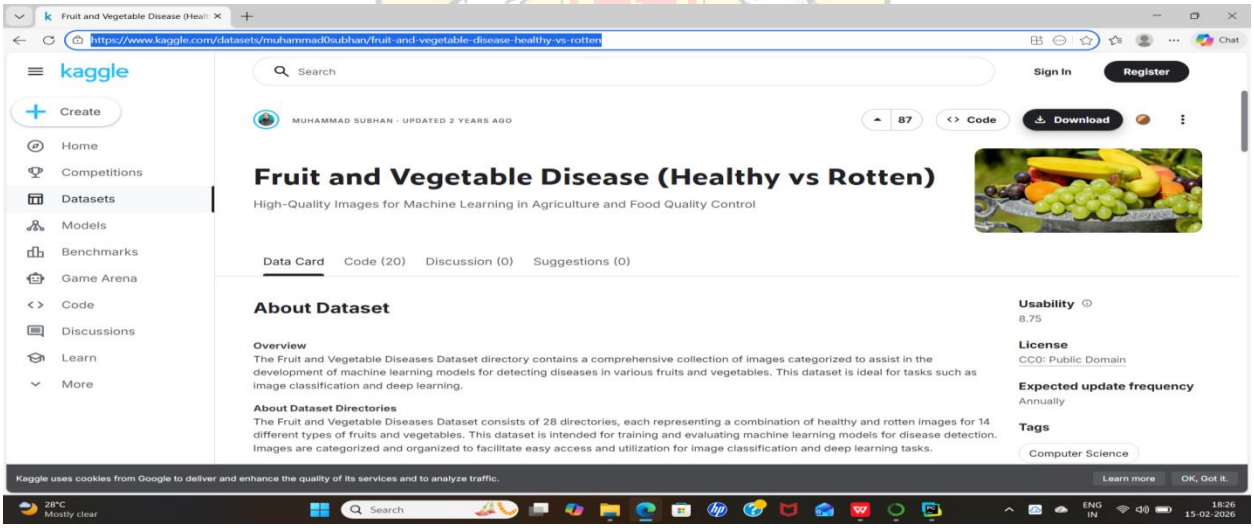
There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.In this project, we have used 28 classes of fruits and vegetables data. This data is downloaded from kaggle.com or can be connected by using API. Please refer to the link given below to download the dataset.

Link: Dataset

Fruit and Vegetable Disease (Healthy vs Rotten)

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

Note: There are several techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.



SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

IMPORT THE FOLLOWING LIBRARIES :

```
import os
import shutil
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
import shutil
from sklearn.model_selection import train_test_split
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Model
from keras.optimizers import Adam
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Model
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
from tensorflow.keras.preprocessing.image import load_img, img_to_array
```

READ THE DATASET :

Our dataset format might be in .csv, excel files, .txt, .json, or zip files, etc. We can read the dataset with the help of pandas.

At first, unzip the data and convert it into a pandas data frame.

```
!mkdir ~/.kaggle

!cp kaggle.json ~/.kaggle

!kaggle datasets download -d muhammad0subhan/fruit-and-vegetable-disease-healthy-vs-rotten
```

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /root/.kaggle/kaggle.json'
Dataset URL: https://www.kaggle.com/datasets/muhammad0subhan/fruit-and-vegetable-disease-healthy-vs-rotten
License(s): CC0-1.0
Downloading fruit-and-vegetable-disease-healthy-vs-rotten.zip to /content
100% 4.76G/4.77G [00:41<00:00, 136MB/s]
100% 4.77G/4.77G [00:41<00:00, 124MB/s]
```

```
!unzip /content/fruit-and-vegetable-disease-healthy-vs-rotten.zip
```

```python
import numpy as np
from sklearn.model_selection import train_test_split


# Set the path to the dataset
dataset_dir = '/content/Fruit And Vegetable Diseases Dataset'
classes = os.listdir(dataset_dir)

# Create directories for train, val, and test sets
output_dir = 'output_dataset'
os.makedirs(output_dir, exist_ok=True)
os.makedirs(os.path.join(output_dir, 'train'), exist_ok=True)
os.makedirs(os.path.join(output_dir, 'val'), exist_ok=True)
os.makedirs(os.path.join(output_dir, 'test'), exist_ok=True)

for cls in classes:
    os.makedirs(os.path.join(output_dir, 'train', cls), exist_ok=True)
    os.makedirs(os.path.join(output_dir, 'val', cls), exist_ok=True)
    os.makedirs(os.path.join(output_dir, 'test', cls), exist_ok=True)

    class_dir = os.path.join(dataset_dir, cls)
    images = os.listdir(class_dir)[:200]

    print(cls,len(images))

    train_and_val_images, test_images = train_test_split(images, test_size=0.2, random_state=42)
    train_images, val_images = train_test_split(train_and_val_images, test_size=0.25, random_state=42)  # 0.25 x 0.8 = 0.2

    # Copy images to respective directories
    for img in train_images:
        shutil.copy(os.path.join(class_dir, img), os.path.join(output_dir, 'train', cls, img))
    for img in val_images:
        shutil.copy(os.path.join(class_dir, img), os.path.join(output_dir, 'val', cls, img))
    for img in test_images:
        shutil.copy(os.path.join(class_dir, img), os.path.join(output_dir, 'test', cls, img))

print("Dataset split into training, validation, and test sets.")
```

Gayathri Educational Society
ESTD-2001

```python
# Define directories
dataset_dir = '/content/output_dataset'
train_dir = os.path.join(dataset_dir, 'train')
val_dir = os.path.join(dataset_dir, 'val')
test_dir = os.path.join(dataset_dir, 'test')

# Define image size expected by the pre-trained model
IMG_SIZE = (224, 224)  # Common size for many models like ResNet, VGG, MobileNet

# Create ImageDataGenerators for resizing and augmenting the images
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

val_test_datagen = ImageDataGenerator(rescale=1./255)

# Load and resize the images from directories
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=IMG_SIZE,
    batch_size=32,
    class_mode='binary'  # Assuming binary classification for healthy vs rotten
)

val_generator = val_test_datagen.flow_from_directory(
    val_dir,
    target_size=IMG_SIZE,
    batch_size=32,
    class_mode='binary'
)

test_generator = val_test_datagen.flow_from_directory(
    test_dir,
    target_size=IMG_SIZE,
    batch_size=32,
    class_mode='binary',
    shuffle=False  # Do not shuffle test data
)

# Print class indices for reference
print(train_generator.class_indices)
print(val_generator.class_indices)
print(test_generator.class_indices)
```

DATE	20-02-2026
TEAM ID	LTVIP2026TMIDS90622
PROJECT NAME	SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES
MAXIMUM MARKS	3 MARKS

6.3 DATA PREPROCESSING

Data Visualization:

The provided Python code imports necessary libraries and modules for image manipulation. It selects a random image file from a specified folder path. Then, it displays the randomly selected image using IPython's Image module. This code is useful for showcasing random images from a directory for various purposes like data exploration or testing image processing algorithms.

```
# Specify the path to your image folder
folder_path = '/content/output_dataset/train/Apple__Healthy'  # Replace with the actual path to your image folder

# List all files in the folder
image_files = [f for f in os.listdir(folder_path) if f.endswith(('.jpg', '.png', '.jpeg'))]

# Select a random image from the list
selected_image = random.choice(image_files)

# Display the randomly selected image
image_path = os.path.join(folder_path, selected_image)
display(Image(filename=image_path))
```

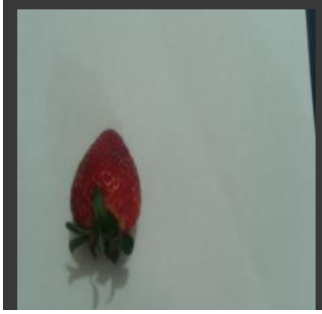
In the above code, I used class Apple_healthy 0 for prediction, This code randomly selects an image file from a specified folder (folder_path) containing JPEG, PNG, or JPEG files, and then displays the selected image using IPython's display function. It utilizes Python's OS and random modules for file manipulation and random selection, respectively. And It has predicted correctly as Apple_healthy 0.

```
# Specify the path to your image folder
folder_path = '/content/output_dataset/test/Strawberry__Healthy'  # Replace with the actual path to your image folder

# List all files in the folder
image_files = [f for f in os.listdir(folder_path) if f.endswith(('.jpg', '.png', '.jpeg'))]

# Select a random image from the list
selected_image = random.choice(image_files)

# Display the randomly selected image
image_path = os.path.join(folder_path, selected_image)
display(Image(filename=image_path))
```



In the above code, I used class strawberry_healthy for prediction, This code randomly selects an image file from a specified folder (folder_path) containing JPEG, PNG, or JPEG files, and then displays the selected image using IPython's display function. It utilizes Python's OS and random modules for file manipulation and random selection, respectively. And It has predicted correctly as strawberry_healthy .

```
# Specify the path to your image folder
folder_path = '/content/output_dataset/test/Cucumber__Rotten'  # Replace with the actual path to your image folder

# List all files in the folder
image_files = [f for f in os.listdir(folder_path) if f.endswith(('.jpg', '.png', '.jpeg'))]

# Select a random image from the list
selected_image = random.choice(image_files)

# Display the randomly selected image
image_path = os.path.join(folder_path, selected_image)
display(Image(filename=image_path))
```

In the above code, I used class cucumber_rotten for prediction, This code randomly selects an image file from a specified folder (folder_path) containing JPEG, PNG, or JPEG files, and then displays the selected image using IPython's display function. It utilizes Python's OS and

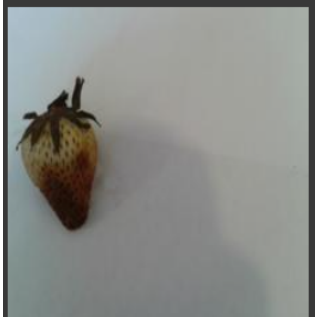
random modules for file manipulation and random selection, respectively. And It has predicted correctly as cucumber_rotten.

```
# Specify the path to your image folder
folder_path = '/content/output_dataset/test/Strawberry__Rotten'  # Replace with the actual path to your image folder

# List all files in the folder
image_files = [f for f in os.listdir(folder_path) if f.endswith(('.jpg', '.png', '.jpeg'))]

# Select a random image from the list
selected_image = random.choice(image_files)

# Display the randomly selected image
image_path = os.path.join(folder_path, selected_image)
display(Image(filename=image_path))
```



In the above code, I used class strawberry_rotten for prediction, This code randomly selects an image file from a specified folder (folder_path) containing JPEG, PNG, or JPEG files, and then displays the selected image using IPython's display function. It utilizes Python's OS and random modules for file manipulation and random selection, respectively. And It has predicted correctly as strawberry_rotten.

Gayathri Educational Society
ESTD-2001

SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

DATE	28-02-2026
TEAM ID	LTVIP2026TMIDS90622
PROJECT NAME	SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES
MAXIMUM MARKS	4 MARKS

6.4 MODEL BUILDING

Split Data and Model Building

Train-Test-Split:

In this project, we have already separated data for training and testing.

```
trainpath = "/content/output_dataset/train"
testpath= "/content/output_dataset/test"

train_datagen = ImageDataGenerator(rescale = 1./255,zoom_range= 0.2,shear_range= 0.2)
test_datagen = ImageDataGenerator(rescale = 1./255)

train = train_datagen.flow_from_directory(trainpath,target_size =(224,224),batch_size = 20)
test = test_datagen.flow_from_directory(testpath,target_size =(224,224),batch_size = 20) ,#5 ,15 , 32, 50

Found 3358 images belonging to 28 classes.
Found 1120 images belonging to 28 classes.
```

Model Building:

Vgg16 Transfer-Learning Model:

The VGG16-based neural network is created using a pre-trained VGG16 architecture with frozen weights. The model is built sequentially, incorporating the VGG16 base, a flattening layer, dropout for regularization, and a dense layer with SoftMax activation for classification into five categories. The model is compiled using the Adam optimizer and sparse categorical cross-entropy loss. During training, which spans 10 epochs, a generator is employed for the training data, and validation is conducted, incorporating call-backs such as Model Checkpoint and Early Stopping. The best-performing model is saved as "healthy_vs_rotten.h5 " for potential future use. The model summary provides an overview of the architecture, showcasing the layers and parameters involved.

```
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Model
```

```
vgg = VGG16(include_top = False,input_shape = (224,224,3))
```

```
for layer in vgg.layers:
  print(layer)
```

```
<keras.src.engine.input_layer.InputLayer object at 0x79c096fde230>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79c096fde4d0>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79c0081b7a90>
<keras.src.layers.pooling.max_pooling2d.MaxPooling2D object at 0x79bff7ef2f80>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79c094405810>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79c00834ba30>
<keras.src.layers.pooling.max_pooling2d.MaxPooling2D object at 0x79bff6dad540>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79c096fd2c20>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79c094405360>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79c094405db0>
<keras.src.layers.pooling.max_pooling2d.MaxPooling2D object at 0x79bfcc0fc490>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79bff6dae7d0>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79bff6dad4b0>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79bff6dae020>
<keras.src.layers.pooling.max_pooling2d.MaxPooling2D object at 0x79bfcc0fff10>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79bfcc0fe0b0>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79bfcc0fe770>
<keras.src.layers.convolutional.conv2d.Conv2D object at 0x79bfcc0fd300>
<keras.src.layers.pooling.max_pooling2d.MaxPooling2D object at 0x79bfcc0fe650>
```

```
len(vgg.layers)
```

```
19
```

```
for layer in vgg.layers:
  layer.trainable = False
```

```
x= Flatten()(vgg.output)
```

```
output = Dense(28, activation ='softmax')(x)
```

```
vgg16 = Model(vgg.input,output)
```

```
vgg16.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080
```

ESTD-2001

```
from keras.callbacks import EarlyStopping
from keras.optimizers import Adam
opt = Adam(lr=0.0001)

# Assuming you have defined your VGG16 model as vgg16

# Define Early Stopping callback
early_stopping = EarlyStopping(monitor='val_accuracy', patience=3, restore_best_weights=True)

# Compile the model (you may have already done this)
vgg16.compile(optimizer = "Adam" , loss='categorical_crossentropy', metrics=['accuracy'])


# # Train the model with early stopping callback
history = vgg16.fit(train, validation_data=test,
                    epochs=15,
                   steps_per_epoch=20,
                  callbacks=[early_stopping])
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
Epoch 1/15
20/20 [==============================] - 28s 1s/step - loss: 1.4851 - accuracy: 0.6100 - val_loss: 1.3616 - val_accuracy: 0.6390
Epoch 2/15
20/20 [==============================] - 25s 1s/step - loss: 1.2048 - accuracy: 0.6750 - val_loss: 1.4731 - val_accuracy: 0.6336
Epoch 3/15
20/20 [==============================] - 31s 2s/step - loss: 1.1071 - accuracy: 0.7075 - val_loss: 1.6028 - val_accuracy: 0.6023
Epoch 4/15
20/20 [==============================] - 25s 1s/step - loss: 0.8216 - accuracy: 0.7675 - val_loss: 1.1175 - val_accuracy: 0.6979
Epoch 5/15
20/20 [==============================] - 26s 1s/step - loss: 0.7029 - accuracy: 0.7875 - val_loss: 1.2511 - val_accuracy: 0.6836
Epoch 6/15
20/20 [==============================] - 26s 1s/step - loss: 0.8559 - accuracy: 0.7575 - val_loss: 1.2702 - val_accuracy: 0.6685
Epoch 7/15
20/20 [==============================] - 26s 1s/step - loss: 0.6845 - accuracy: 0.8100 - val_loss: 1.0867 - val_accuracy: 0.7265
Epoch 8/15
20/20 [==============================] - 25s 1s/step - loss: 0.5509 - accuracy: 0.8325 - val_loss: 1.2089 - val_accuracy: 0.7069
Epoch 9/15
20/20 [==============================] - 27s 1s/step - loss: 0.5796 - accuracy: 0.8400 - val_loss: 0.8013 - val_accuracy: 0.7802
Epoch 10/15
20/20 [==============================] - 25s 1s/step - loss: 0.4136 - accuracy: 0.8850 - val_loss: 0.9262 - val_accuracy: 0.7560
Epoch 11/15
20/20 [==============================] - 26s 1s/step - loss: 0.4959 - accuracy: 0.8575 - val_loss: 1.0332 - val_accuracy: 0.7292
Epoch 12/15
20/20 [==============================] - 27s 1s/step - loss: 0.5788 - accuracy: 0.8300 - val_loss: 0.8914 - val_accuracy: 0.7587
```

```
vgg16.save('healthy_vs_rotten')
```

Now, Our transfer learning model “healthy_vs_rotten.h5” is saved .

APP.PY CODE:

```
from sklearn.model_selection import train_test_split  # Fixed import
statement
import os
import shutil
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from IPython.display import Image, display
import random
from sklearn.model_selection import train_test_split
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Model
```

```
from keras.optimizers import Adam
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Model
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np
from sklearn.model_selection import train_test_split
from keras.callbacks import EarlyStopping
from keras.optimizers import Adam


# Set the path to the dataset - Update this path to the correct location of your dataset
dataset_dir = 'C:/Users/tejas/PycharmProjects/PythonProject/content/Fruit And Vegetable Diseases Dataset'  # CHANGE THIS to your actual dataset path with full absolute path

# Check if the directory exists before proceeding
if not os.path.exists(dataset_dir):
    raise FileNotFoundError(f"The dataset directory '{dataset_dir}' does not exist. Please check the path.")

# Create directories for train, val, and test sets
output_dir = 'output_dataset'
os.makedirs(output_dir, exist_ok=True)
os.makedirs(os.path.join(output_dir, 'train'), exist_ok=True)
os.makedirs(os.path.join(output_dir, 'val'), exist_ok=True)
os.makedirs(os.path.join(output_dir, 'test'), exist_ok=True)

classes = os.listdir(dataset_dir)

for cls in classes:
    os.makedirs(os.path.join(output_dir, 'train', cls), exist_ok=True)
    os.makedirs(os.path.join(output_dir, 'val', cls), exist_ok=True)
    os.makedirs(os.path.join(output_dir, 'test', cls), exist_ok=True)
```

```
    class_dir = os.path.join(dataset_dir, cls)
    images = os.listdir(class_dir)[:200]

    print(cls, len(images))

    train_and_val_images, test_images = train_test_split(
        images, test_size=0.2, random_state=42
    )
    train_images, val_images = train_test_split(
        train_and_val_images, test_size=0.25, random_state=42
    )  # 0.25 x 0.8 = 0.2

    # Copy images to respective directories
    for img in train_images:
        shutil.copy(
            os.path.join(class_dir, img),
            os.path.join(output_dir, 'train', cls, img)
        )
    for img in val_images:
        shutil.copy(
            os.path.join(class_dir, img),
            os.path.join(output_dir, 'val', cls, img)
        )
    for img in test_images:
        shutil.copy(
            os.path.join(class_dir, img),
            os.path.join(output_dir, 'test', cls, img)
        )

print("Dataset split into training, validation, and test sets.")
# Define directories
dataset_dir = 'output_dataset'
train_dir = os.path.join(dataset_dir, 'train')
val_dir = os.path.join(dataset_dir, 'val')
test_dir = os.path.join(dataset_dir, 'test')

# Define image size expected by the pre-trained model
IMG_SIZE = (224, 224)  # Common size for many models like ResNet, VGG,
```

Gayathri Educational Society ESTD-2001

MobileNet

```
# Create ImageDataGenerators for resizing and augmenting the images
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

val_test_datagen = ImageDataGenerator(rescale=1./255)

# Load and resize the images from directories
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=IMG_SIZE,
    batch_size=32,
    class_mode='binary'  # Assuming binary classification for healthy
vs rotten
)

val_generator = val_test_datagen.flow_from_directory(
    val_dir,
    target_size=IMG_SIZE,
    batch_size=32,
    class_mode='binary'
)

test_generator = val_test_datagen.flow_from_directory(
    test_dir,
    target_size=IMG_SIZE,
    batch_size=32,
    class_mode='binary',
    shuffle=False  # Do not shuffle test data
)
```

Gayathri Educational Society

```
# Print class indices for reference
print(train_generator.class_indices)
print(val_generator.class_indices)
print(test_generator.class_indices)
# Specify the path to your image folder
folder_path = 'output_dataset/train/Apple__Healthy'   # Replace with
the actual path to your image folder

# List all files in the folder
image_files   =   [f   for   f   in   os.listdir(folder_path)   if
f.endswith(('.jpg', '.png', '.jpeg'))]

# Select a random image from the list
selected_image = random.choice(image_files)

# Display the randomly selected image
image_path = os.path.join(folder_path, selected_image)
display(Image(filename=image_path))
# Specify the path to your image folder
folder_path = 'output_dataset/test/Strawberry__Healthy'   # Replace
with the actual path to your image folder

# List all files in the folder
image_files   =   [f   for   f   in   os.listdir(folder_path)   if
f.endswith(('.jpg', '.png', '.jpeg'))]

# Select a random image from the list
selected_image = random.choice(image_files)

# Display the randomly selected image
image_path = os.path.join(folder_path, selected_image)
display(Image(filename=image_path))
# Specify the path to your image folder
folder_path = 'output_dataset/test/Cucumber__Rotten'   # Replace with
the actual path to your image folder

# List all files in the folder
image_files   =   [f   for   f   in   os.listdir(folder_path)   if
```

```
f.endswith(('.jpg', '.png', '.jpeg'))]

# Select a random image from the list
selected_image = random.choice(image_files)

# Display the randomly selected image
image_path = os.path.join(folder_path, selected_image)
display(Image(filename=image_path))
# Specify the path to your image folder
folder_path = 'output_dataset/test/Strawberry__Rotten'  # Replace with
the actual path to your image folder
# List all files in the folder
image_files    =    [f    for    f    in    os.listdir(folder_path)    if
f.endswith(('.jpg', '.png', '.jpeg'))]
# Select a random image from the list
selected_image = random.choice(image_files)

# Display the randomly selected image
image_path = os.path.join(folder_path, selected_image)
display(Image(filename=image_path))
trainpath = "output_dataset/train"
testpath = "output_dataset/test"

train_datagen = ImageDataGenerator(
    rescale=1./255,
    zoom_range=0.2,
    shear_range=0.2
)
test_datagen = ImageDataGenerator(rescale=1./255)

train = train_datagen.flow_from_directory(
    trainpath,
    target_size=(224, 224),
    batch_size=20
)
test = test_datagen.flow_from_directory(
    testpath,
    target_size=(224, 224),
    batch_size=20
```

Gayathri Educational Society
ESTD-2001

```
)  # 5, 15, 32, 50
vgg = VGG16(include_top=False, input_shape=(224, 224, 3))

for layer in vgg.layers:
    print(layer)
len(vgg.layers)
for layer in vgg.layers:
    layer.trainable = False
x = Flatten()(vgg.output)
output = Dense(30 , activation='softmax')(x)
vgg16 = Model(vgg.input, output)
vgg16.summary()
opt = Adam(learning_rate=0.0001)
# Assuming you have defined your VGG16 model as vgg16
# Define Early Stopping callback
early_stopping = EarlyStopping(
    monitor='val_accuracy',
    patience=3,
    restore_best_weights=True
)
# Compile the model (you may have already done this)
vgg16.compile(
    optimizer="Adam",
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

# Train the model with early stopping callback
history = vgg16.fit(
    train,
    validation_data=test,
    epochs=15,
    steps_per_epoch=20,
    callbacks=[early_stopping]
)
vgg16.save('healthy_vs_rotten.h5')
```

Gayathri Educational Society
ESTD-2001

DATE	20-02-2026
TEAM ID	LTVIP2026TMIDS90622
PROJECT NAME	SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES
MAXIMUM MARKS	4 MARKS

6.5 APPLICATION BUILDING

In this section, we will be building a web application that is integrated into the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- ✓ Building HTML Pages
- ✓ Building server-side script

Building HTML Pages:

For this project create two HTML files namely

- ✧ index.html
- ✧ Portfolio-details.html

And save them in the templates folder.

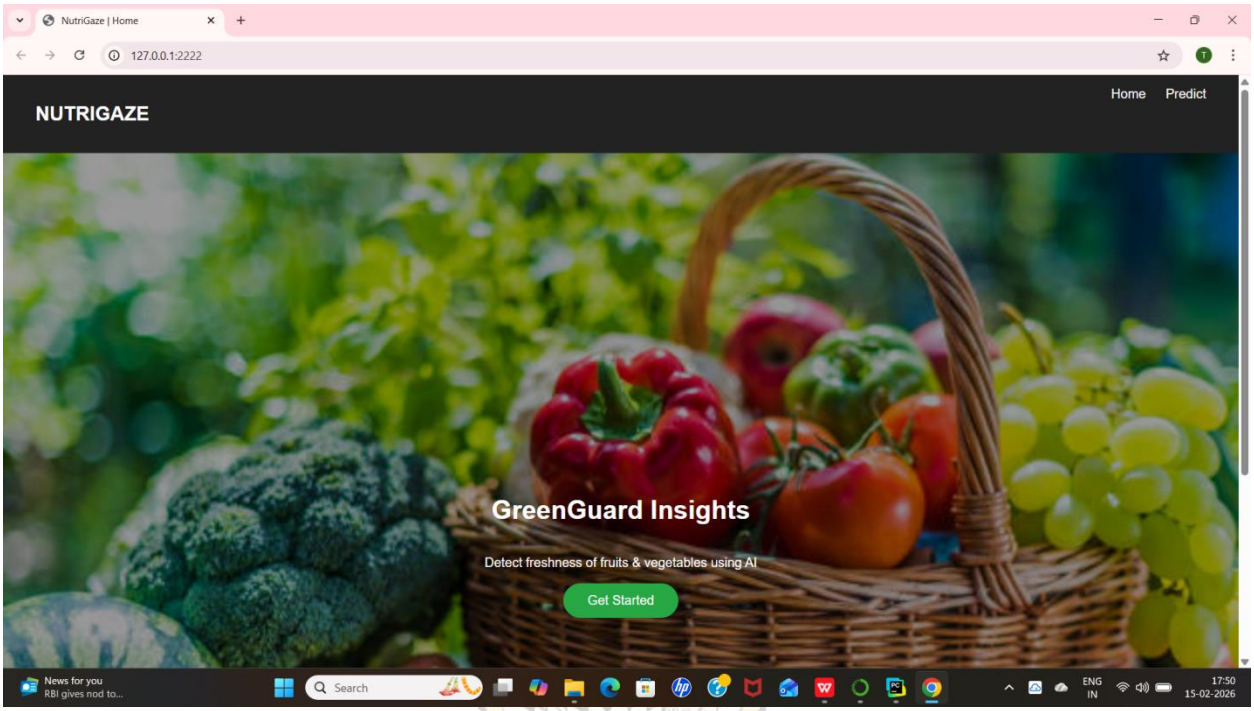
- ✧ Style.css(in static =>css => style.css)
- ✧ Image(in static => image=>img.jpg)

UI Image preview:

This html pages and css spreadsheet will help to enhance the beauty of our web page “Neutrigaze”.

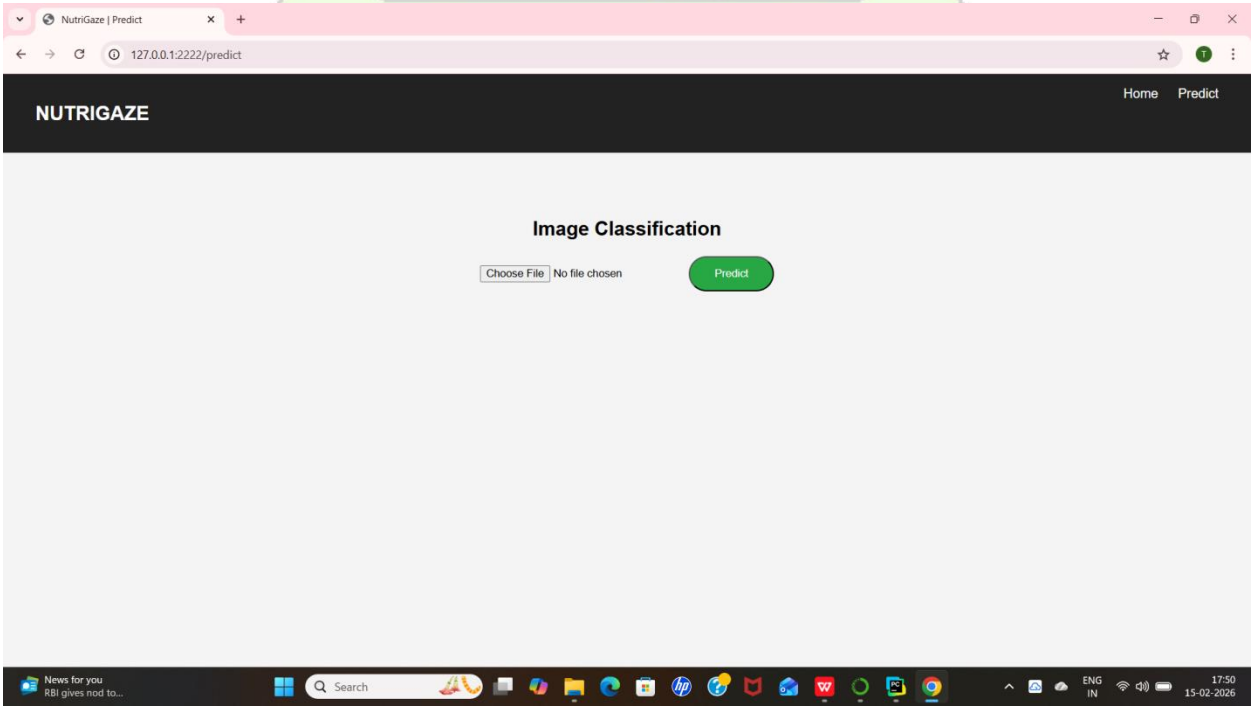
It provide home page and predict page in the preview as shown in the image given below .

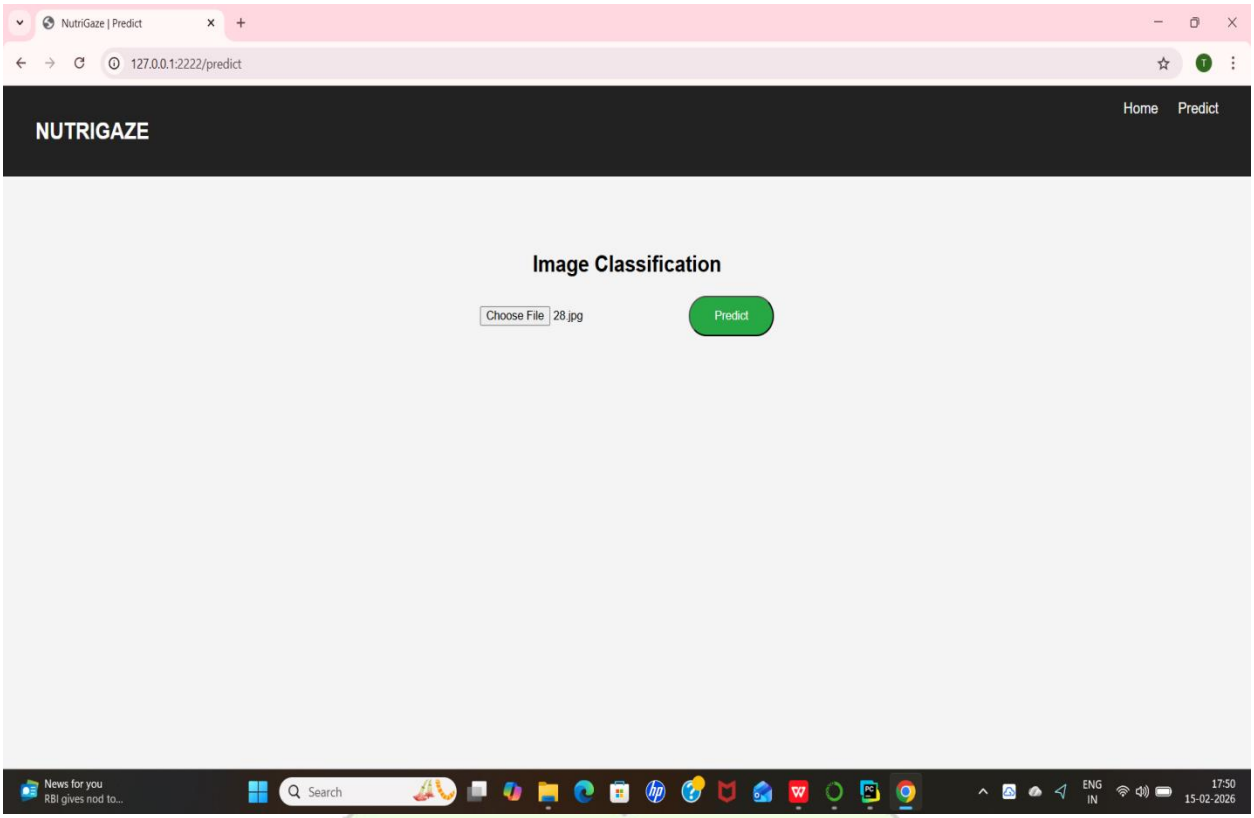
Gayathri Educational Society
ESTD-2001

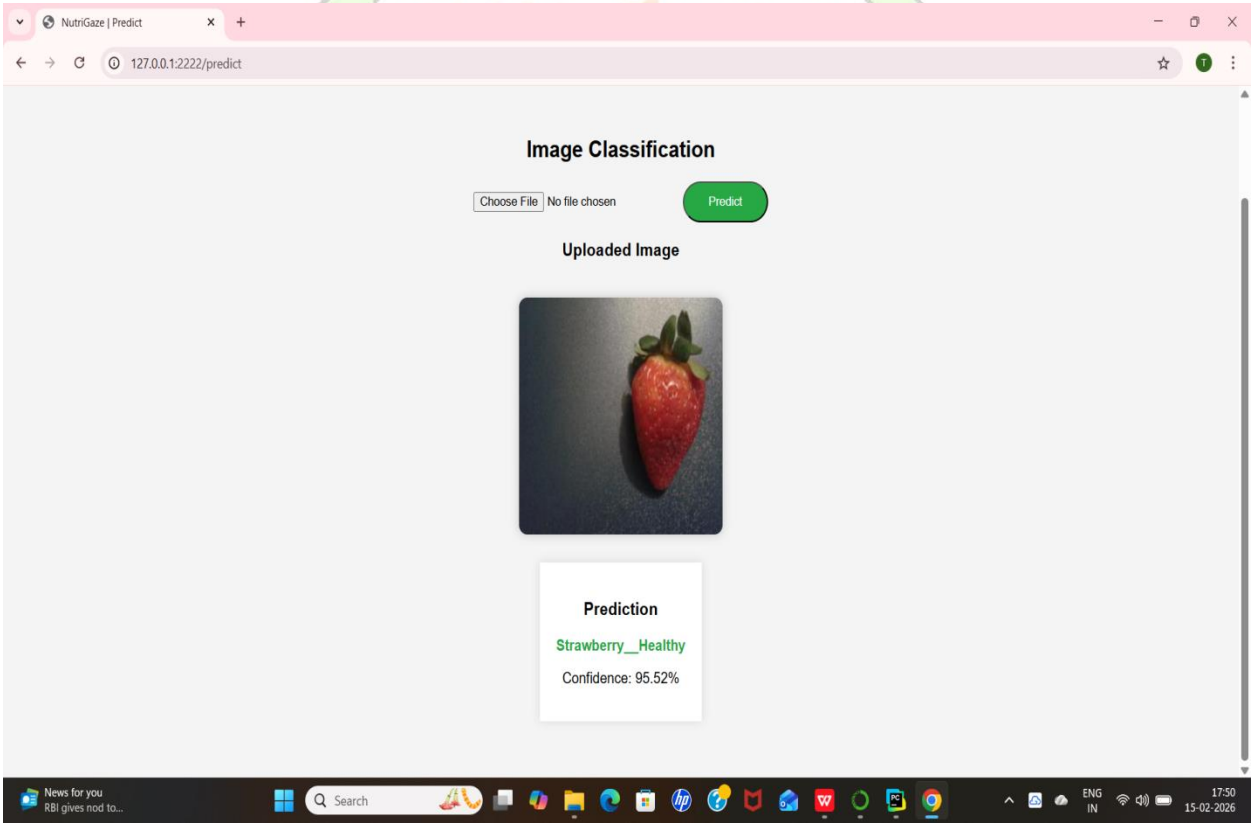


Now when you click on the predict button further in the top right corner you will get redirected to portfolio-details.html

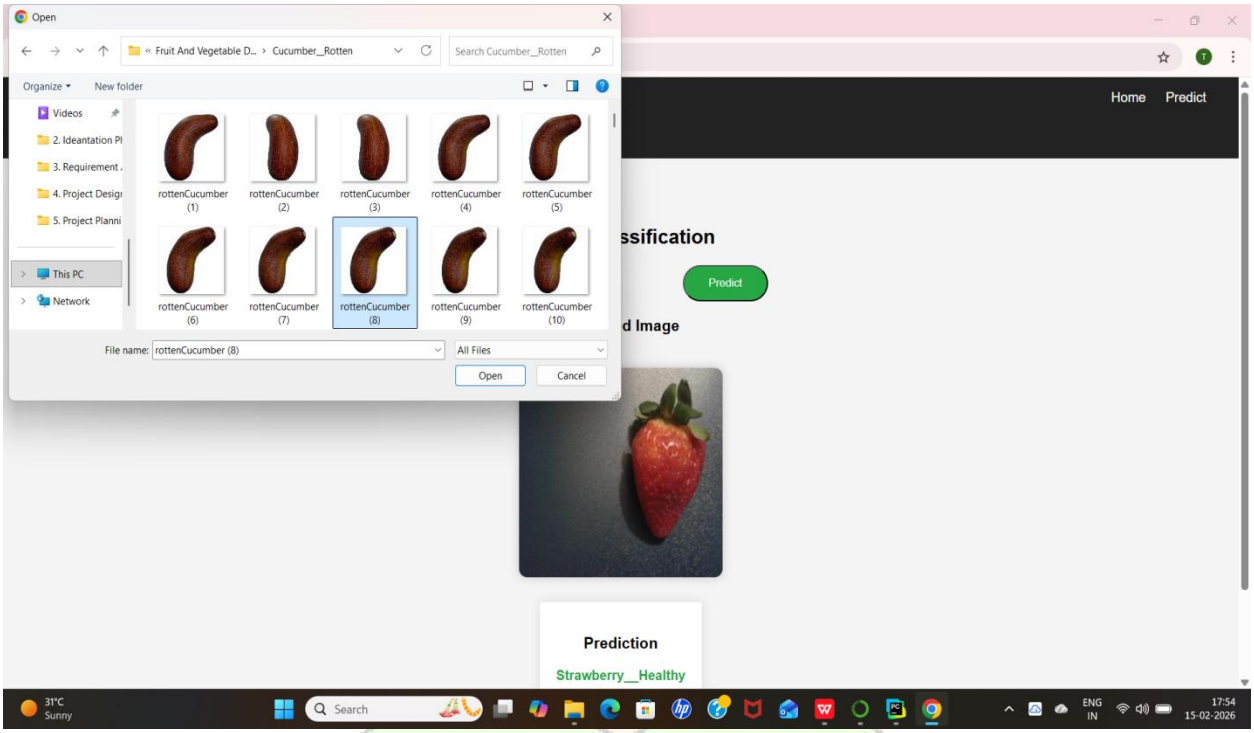
Let's look at what our inner.html file looks like and test the model:



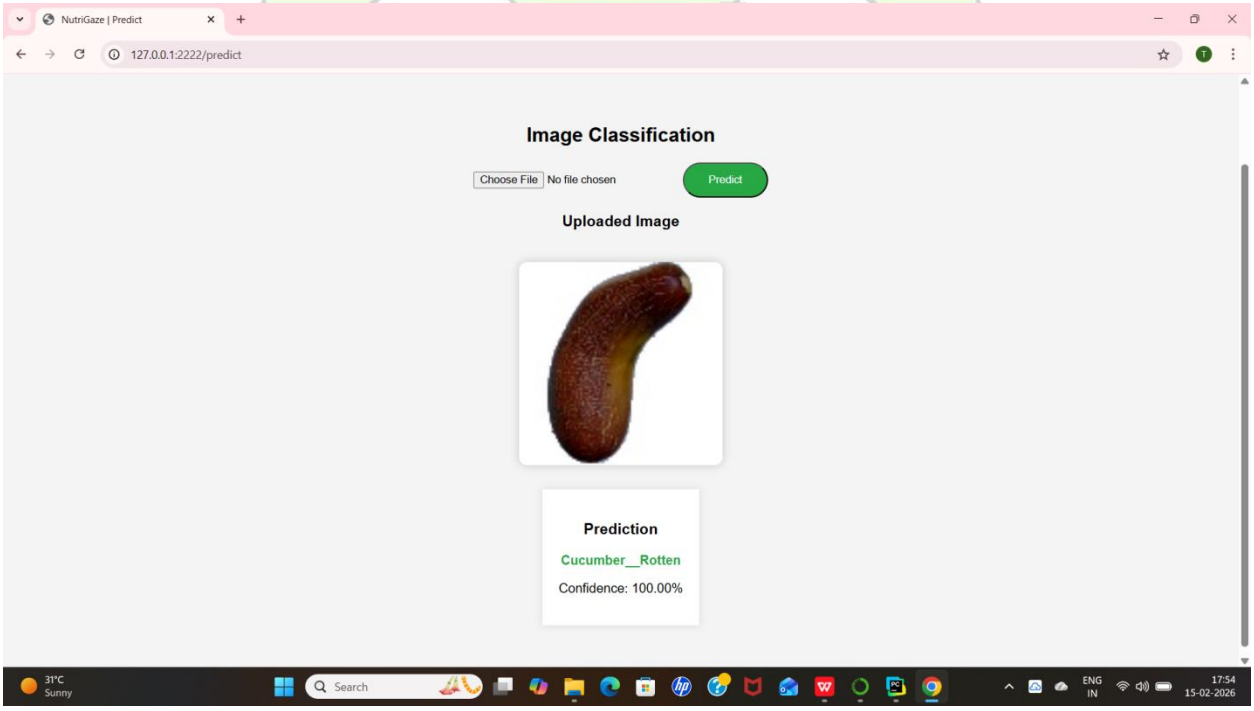

NutriGaze | Predict
127.0.0.1:2222/predict
NUTRIGAZE
Home
Predict
Image Classification
Choose File
28.jpg
Predict


NutriGaze | Predict
127.0.0.1:2222/predict
Image Classification
Choose File
No file chosen
Predict
Uploaded Image
Prediction
Strawberry__Healthy
Confidence: 95.52%

Test Class rotten cucumber (8):



Now when you click on predict button you will get output down to the image itself.Let's look how our output looks like:



Build Python code:

Import the libraries

```
from flask import Flask, render_template,request,jsonify,url_for,redirect
from tensorflow.keras.preprocessing.image import load_img,img_to_array
from PIL import Image
import numpy as np
import os
import tensorflow as tf
```

Load the saved model. Importing the Flask module in the project is mandatory. An object of the Flask class is our WSGI application. The Flask constructor takes the name of the current module (__name__) as argument.

```
app=Flask(__name__)
model = tf.keras.models.load_model('healthy_vs_rotten.h5')

@app.route('/')
def index():
    return render_template("index.html")
'''
```

Here we will be using the declared constructor to route to the HTML page which we have created earlier.

In the above example, the '/' URL is bound with the index.html function. Hence, when the index page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:

```
@app.route('/predict', methods=['GET','POST'])
def output():
    if request.method =='POST':
        f=request.files['pc_image']
        img_path = "static/uploads/" + f.filename
        f.save(img_path)
        img=load_img(img_path,target_size=(224,224))
        # Resize the image to the required size
        # Convert the image to an array and normalize it
        image_array = np.array(img)
        # Add a batch dimension
        image_array = np.expand_dims(image_array, axis=0)
        # Use the pre-trained model to make a prediction
        pred=np.argmax(model.predict(image_array),axis=1)
        index=['Apple__Healthy (0)','Apple__Rotten (1)','Banana__Healthy (2)',
               'Banana__Rotten (3)','Bellpepper__Healthy (4)','Bellpepper__Rotten (5)',
               'Carrot__Healthy (6)','Carrot__Rotten (7)', 'Cucumber__Healthy (8)','Cucumber__Rotten (9)','Grape__Healthy (10)',
               'Grape__Rotten (11)','Guava__Healthy (12)','Guava__Rotten (13)',
               'Jujube__Healthy (14)',
               'Jujube__Rotten (15)', 'Mango__Healthy (16)', 'Mango__Rotten (17)', 'Orange__Healthy (18)',
               'Orange__Rotten (19)', 'Pomegranate__Healthy (20)', 'Pomegranate__Rotten (21)', 'Potato__Healthy (22)',
               'Potato__Rotten (23)', 'Strawberry__Healthy (24)', 'Strawberry__Rotten (25)', 'Tomato__Healthy (26)', 'Tomato__Rotten (27)'
              ]
        prediction = index[int(pred)]
        print("prediction")
        #predict = prediction
        return render_template("portfolio-details.html", predict = prediction)
```

Here we are routing our app to the output() function. This function retrieves all the values from the HTML page using a Post request. That is stored in an array. This array is passed to the model. Predict () function. This function returns the prediction. This prediction value will rendered to the text that we have mentioned in the output.html page earlier.

Main Function:

```
if __name__=='__main__':
    app.run(debug = True,port = 2222)
```

Run the web application

- ✓ Open Anaconda prompt from the start menu
- ✓ Navigate to the folder where your Python script is.
- ✓ Now type the “app.py” command
- ✓ Navigate to the local host where you can view your web page.
- ✓ Click on the inspect button from the top right corner, enter the inputs, click on the predict button, and see the result/prediction on the web.

```
In [1]: runfile('C:/Users/santu/Downloads/flask2/app.py', wdir='C:/
Users/santu/Downloads/flask2')
WARNING:absl:Compiled the loaded model, but the compiled metrics have
yet to be built. `model.compile_metrics` will be empty until you train
or evaluate the model.
WARNING:absl:Error in loading the saved optimizer state. As a result,
your model is starting with a freshly initialized optimizer.
 * Serving Flask app 'app'
 * Debug mode: on
INFO:werkzeug:WARNING: This is a development server. Do not use it in
production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:2222
INFO:werkzeug:Press CTRL+C to quit
INFO:werkzeug: * Restarting with watchdog (windowsapi)
```

TEST.PY CODE:

```
from flask import Flask, render_template, request
from tensorflow.keras.preprocessing.image import load_img,
img_to_array
import numpy as np
import tensorflow as tf
import os
app = Flask(__name__)
# Load trained model
model = tf.keras.models.load_model("healthy_vs_rotten.h5")
# Threshold: If confidence is below 80%, we say "Out of Dataset"
# Adjust this (0.0 to 1.0) based on your testing!
```

Gayathri Educational Society
ESTD-2001

```
CONFIDENCE_THRESHOLD = 0.92  # Requires 92% certainty
classes = [
    'Apple__Healthy',       'Apple__Rotten',       'Banana__Healthy',
'Banana__Rotten',
    'Bellpepper__Healthy',  'Bellpepper__Rotten',  'Carrot__Healthy',
'Carrot__Rotten',
    'Cucumber__Healthy',    'Cucumber__Rotten',    'Grape__Healthy',
'Grape__Rotten',
    'Guava__Healthy',       'Guava__Rotten',       'Jujube__Healthy',
'Jujube__Rotten',
    'Mango__Healthy',       'Mango__Rotten',       'Orange__Healthy',
'Orange__Rotten',
    'Pomegranate__Healthy', 'Pomegranate__Rotten', 'Potato__Healthy',
'Potato__Rotten',
    'Strawberry__Healthy',  'Strawberry__Rotten',  'Tomato__Healthy',
'Tomato__Rotten'
]
@app.route("/")
def home():
    return render_template("index.html")
from tensorflow.keras.applications.vgg16 import preprocess_input # Add
this import
@app.route("/predict", methods=["GET", "POST"])
def predict():
    prediction = None
    confidence = None
    image_path = None
    filename = None
    if request.method == "POST":
        file = request.files.get("pc_image")
        if file:
            filename = file.filename
            image_path = os.path.join("static/uploads", filename)
            os.makedirs("static/uploads", exist_ok=True)
            file.save(image_path)

            # ✅ 1. Standard VGG16 Preprocessing
            img = load_img(image_path, target_size=(224, 224))
```

```
            img_array = np.array(img)
            # If you used rescale=1/255 in training, keep the next
line.
            #    If    not,    comment    it    out    and    use
preprocess_input(img_array)
            img_array = img_array / 255.0
            img_array = np.expand_dims(img_array, axis=0)
            # ✅ 2. Get All Probabilities
            probs = model.predict(img_array)[0]
            class_index = np.argmax(probs)
            max_prob = probs[class_index]
            # ✅ 3. Calculate "Confusion" (Gap between top 2 guesses)
            sorted_probs = np.sort(probs)
            top_choice = sorted_probs[-1]
            second_choice = sorted_probs[-2]
            gap = top_choice - second_choice
            # ✅ 4. STRICT FILTERING
            # A tiger shouldn't have a high gap, and its confidence
shouldn't be massive.
            # We require at least 85% confidence AND a 40% gap over
the next best guess.
            if top_choice < 0.85 or gap < 0.40:
                prediction = "Out of Dataset"
                confidence = top_choice * 100
            else:
                prediction = classes[class_index]
                confidence = top_choice * 100

    return                render_template("portfolio-details.html",
predict=prediction,
                          confidence=confidence,
image_path=filename)
if __name__ == "__main__":
    app.run(debug=True, port=2222)
```

Gayathri Educational Society
ESTD-2001

DATE	20-02-2026
TEAM ID	LTVIP2026TMIDS90622
PROJECT NAME	SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES
MAXIMUM MARKS	5 MARKS

CHAPTER-7

7.1 FUNCTIONAL AND PERFORMANCE TESTING

SMART SORTING: Transfer Learning for Identifying Rotten Fruits and Vegetables:

Testing plays a crucial role in ensuring that the SMART SORTING system functions correctly, delivers accurate predictions, and performs efficiently under real-world conditions. The testing process is divided into Functional Testing and Performance Testing to validate both system behavior and operational efficiency.

❖ **Functional Testing:**

Functional testing verifies whether each component of the system works according to the specified requirements. It ensures that the image classification pipeline, from input to output, operates correctly.

✓ **Input Validation Testing:**

This test ensures that the system correctly accepts valid image formats such as JPG and PNG and rejects unsupported or corrupted files. It verifies that the application handles incorrect inputs gracefully without crashing.

✓ **Image Preprocessing Testing:**

The preprocessing module is tested to confirm that images are properly resized, normalized, and augmented before being passed to the model. The system is checked to ensure that preprocessing maintains image integrity and consistency.

Gayathri Educational Society

✓ **Model Classification Testing:**

The trained transfer learning model using frameworks such as TensorFlow or PyTorch is tested with known labeled images. The predicted output is compared with actual labels to verify classification correctness for categories such as fresh and rotten produce.

✓ **Output Verification Testing:**

This test ensures that the system displays correct predictions along with confidence scores. In industrial setups, it verifies that the classification output correctly triggers sorting mechanisms such as actuators or robotic arms.

✓ **Integration Testing:**

Integration testing confirms that all modules—image acquisition, preprocessing, model inference, and output display—work together seamlessly. It ensures smooth communication between the front-end interface and backend prediction API.

✓ **Error Handling Testing:**

The system is tested under abnormal conditions such as poor lighting images, blurred images, or partial object visibility to verify that it handles errors without failure and provides meaningful feedback.

❖ **Performance Testing:**

Performance testing evaluates the efficiency, speed, scalability, and reliability of the system under different operating conditions.

✓ **Accuracy Testing:**

Model performance is evaluated using metrics such as accuracy, precision, recall, F1-score, and confusion matrix analysis. Pre-trained models like MobileNetV2 or ResNet50 are analyzed to ensure they meet the target accuracy threshold.

Gayathri Educational Society
ESTD-2001

✓ **Response Time Testing:**

The system is tested to measure how quickly it processes an image and generates a prediction. For real-time applications, inference time must remain within acceptable limits to support continuous sorting operations.

✓ **Load Testing:**

Load testing determines how the system performs when multiple images are processed simultaneously. This is especially important for web-based deployments or large-scale industrial applications.

✓ **Stress Testing:**

Stress testing evaluates system stability under extreme conditions such as high image input rates or limited hardware resources. It identifies breaking points and ensures system reliability.

✓ **Scalability Testing:**

This testing verifies whether the system can handle an increased number of fruit and vegetable categories without significant degradation in performance.

✓ **Deployment Environment Testing:**

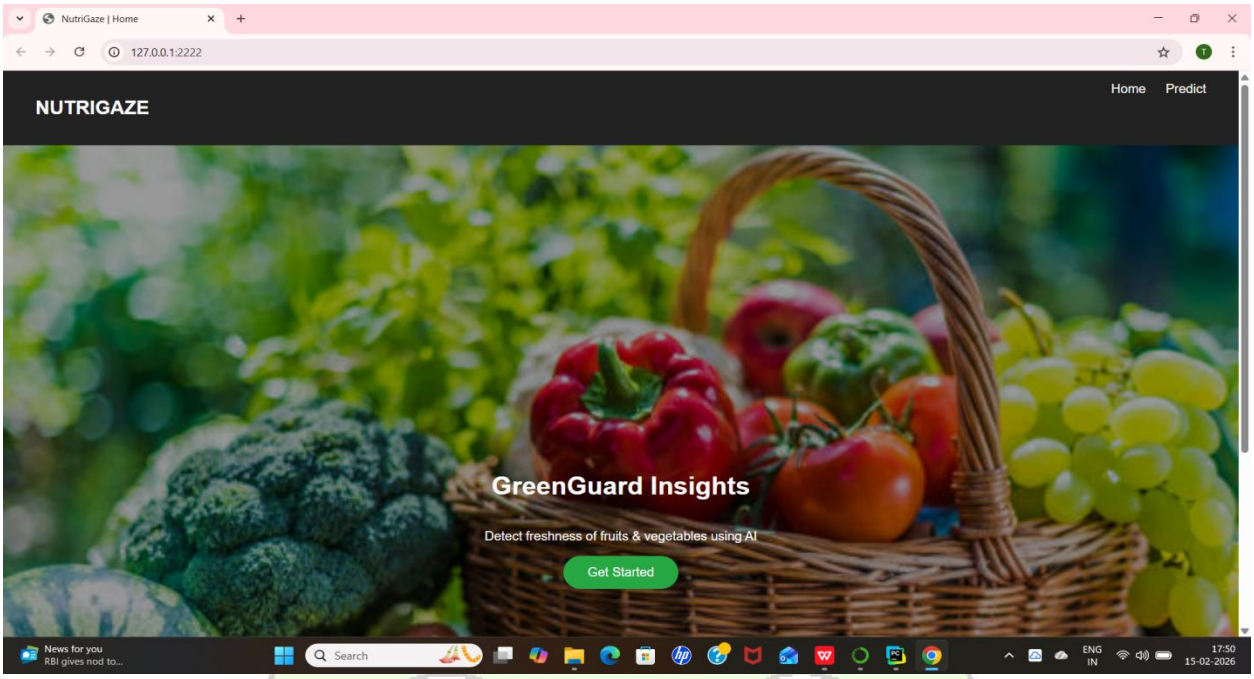
The system is tested in different environments, including local machines, cloud platforms, and edge devices, to ensure consistent performance across deployment platforms.

In conclusion, functional testing ensures that the SMART SORTING system performs its intended tasks accurately, while performance testing guarantees efficiency, reliability, and scalability. Together, these testing strategies validate that the system is robust, accurate, and ready for real-world implementation in agricultural and retail environments.

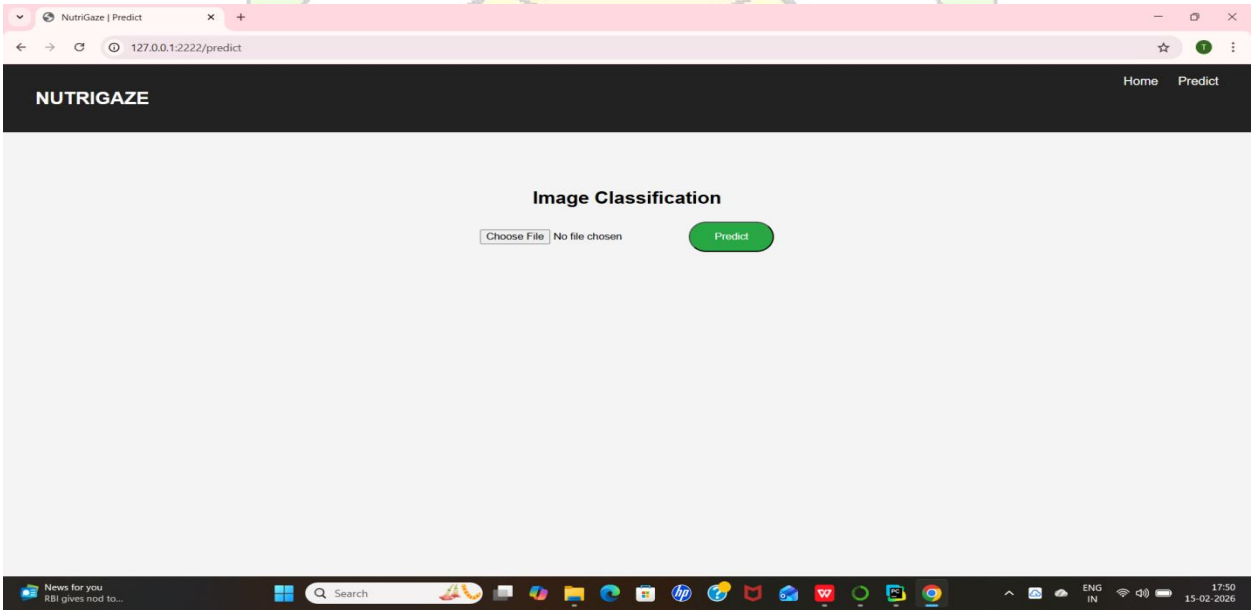
Chapter-8

8.1 OUTPUT SCREENS

HOME PAGE :

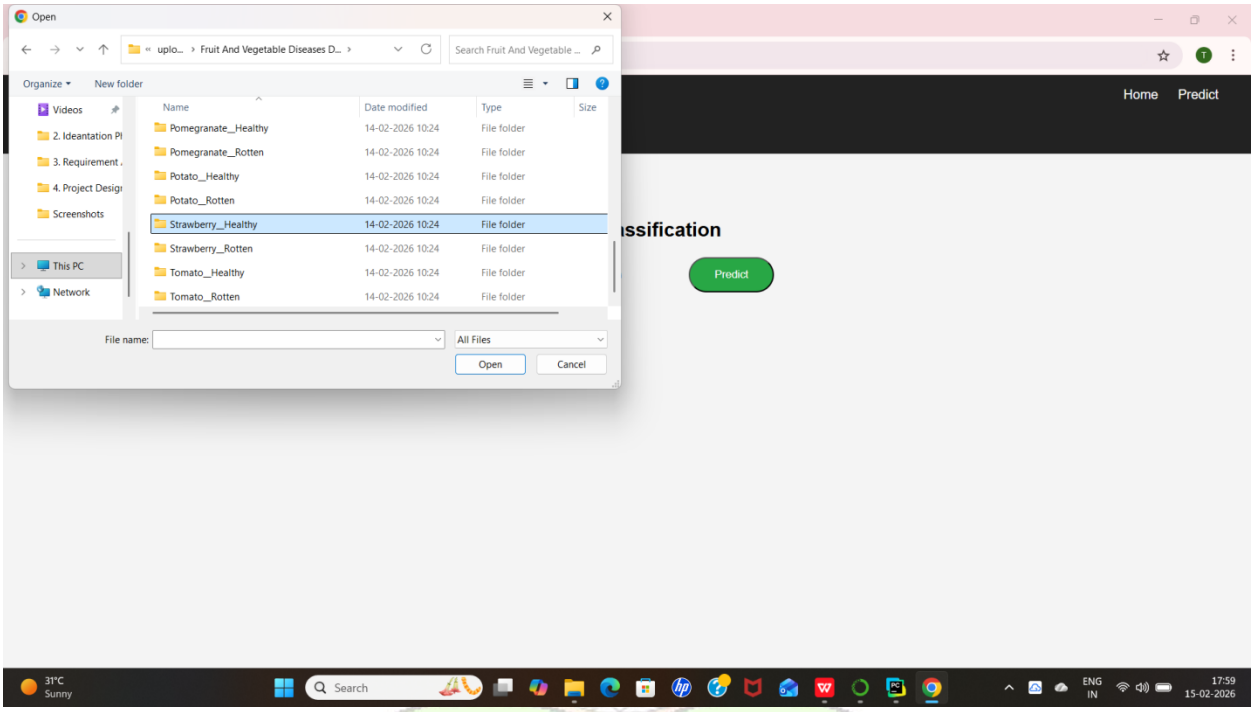


PREDICT PAGE :

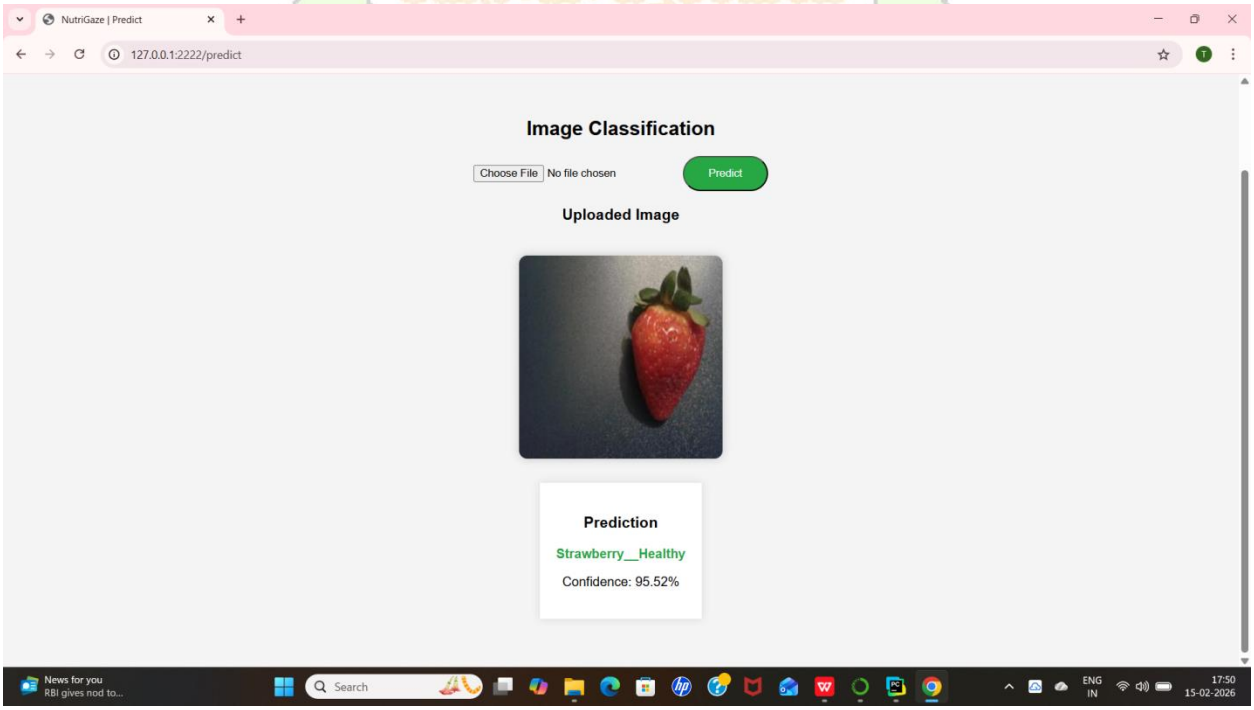


SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

CHOOSING AN IMAGE:

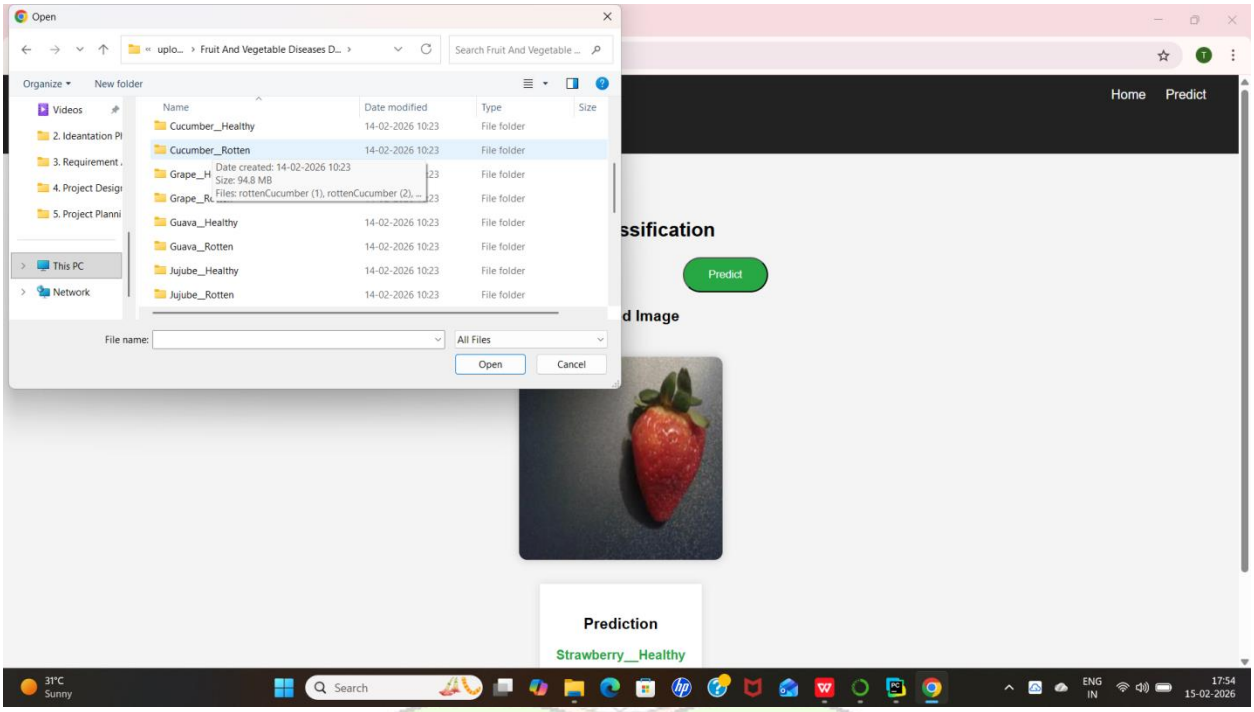


OUTPUT:

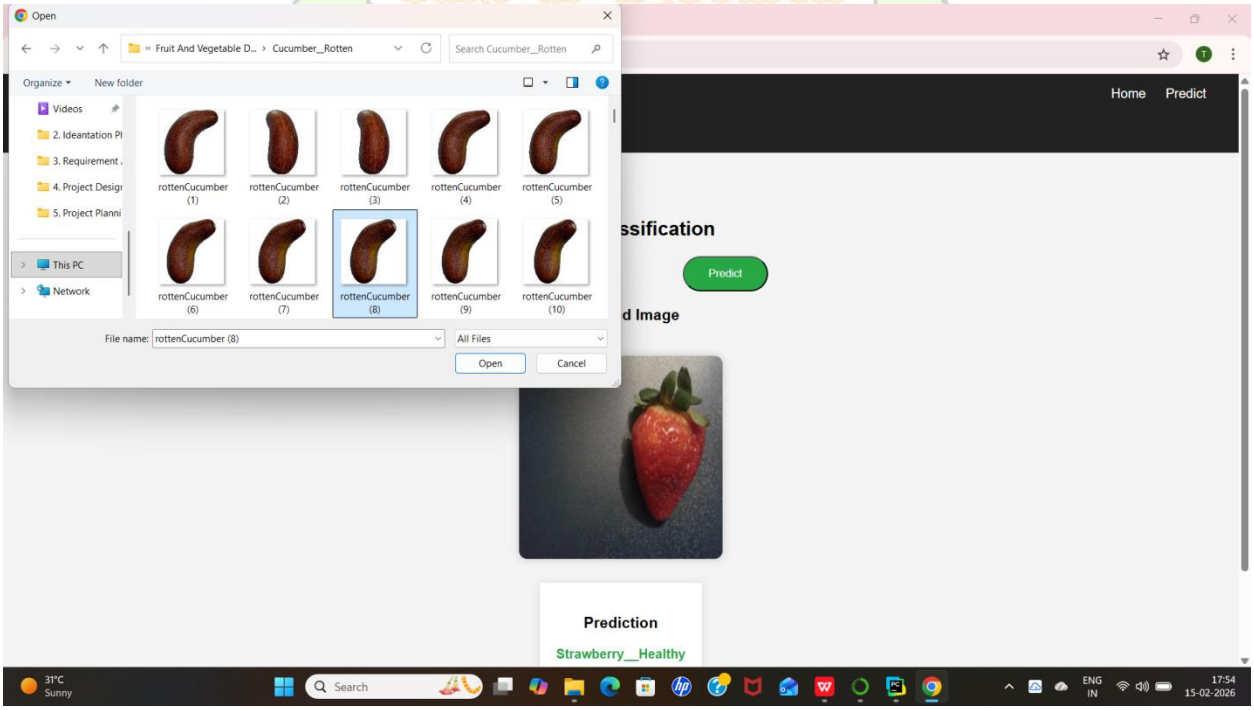


SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

TESTING ANOTHER IMAGE FROM DATASET:

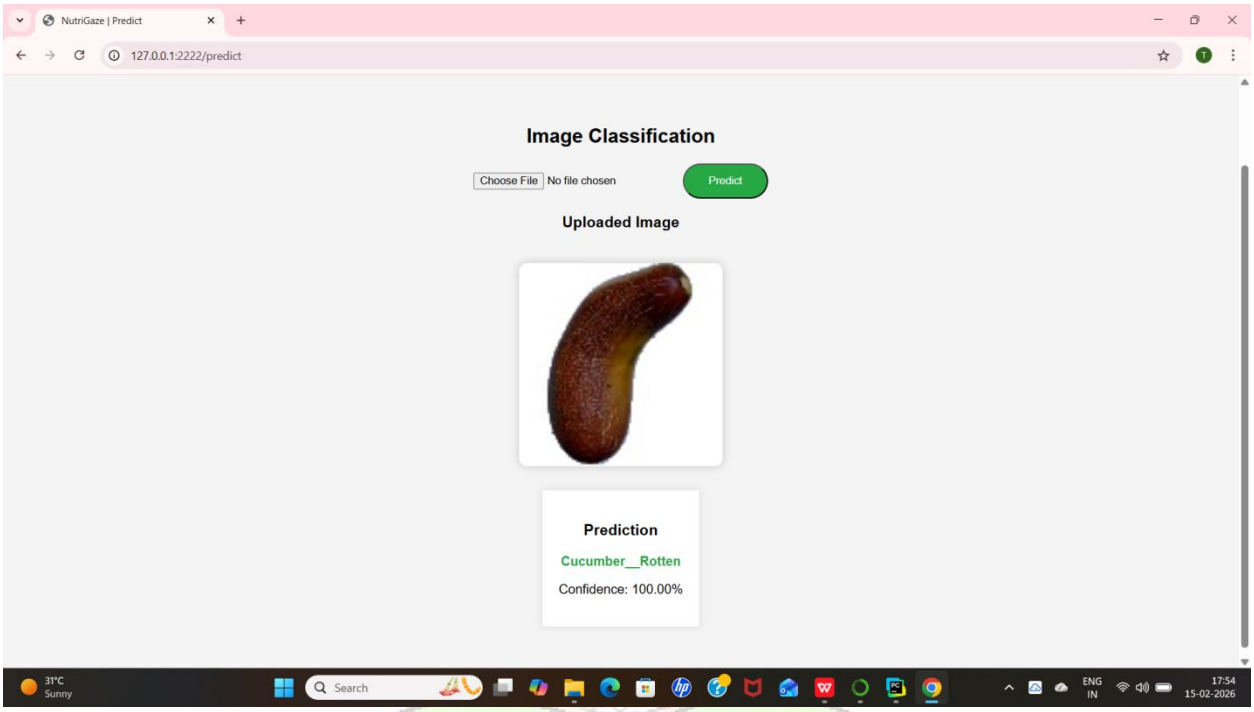


SELECTING ROTTEN CUCUMBER :

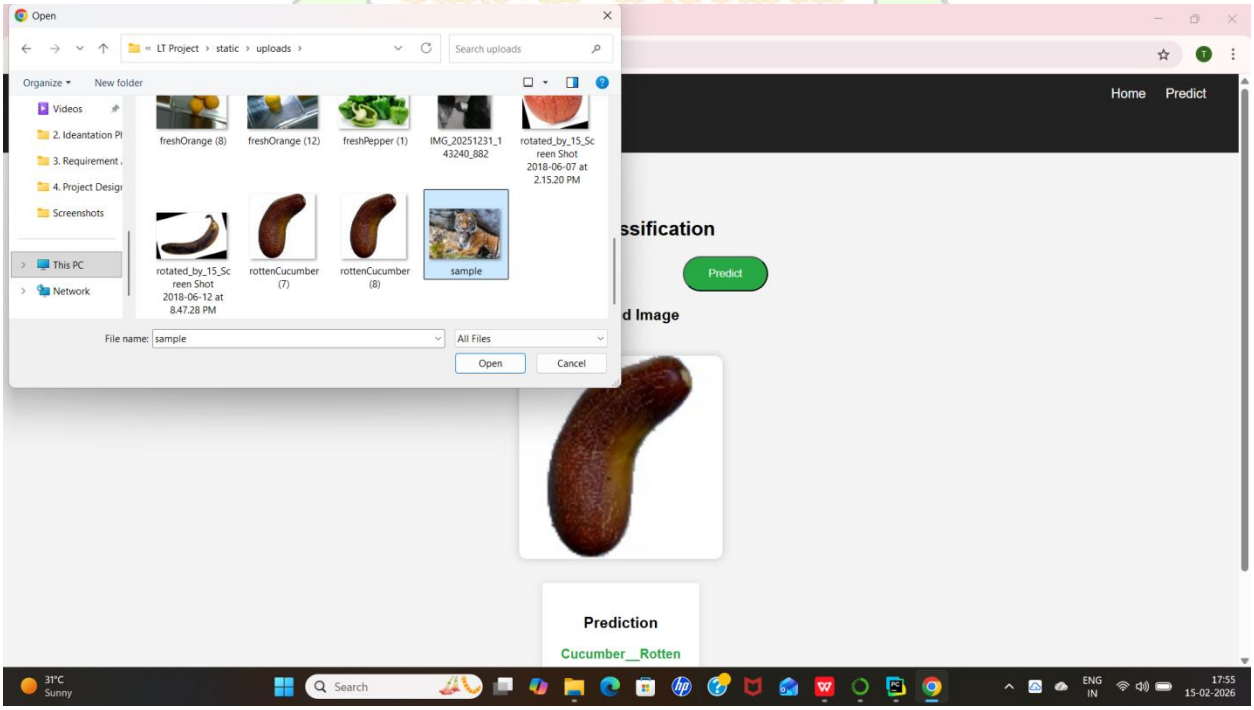


SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

OUTPUT:

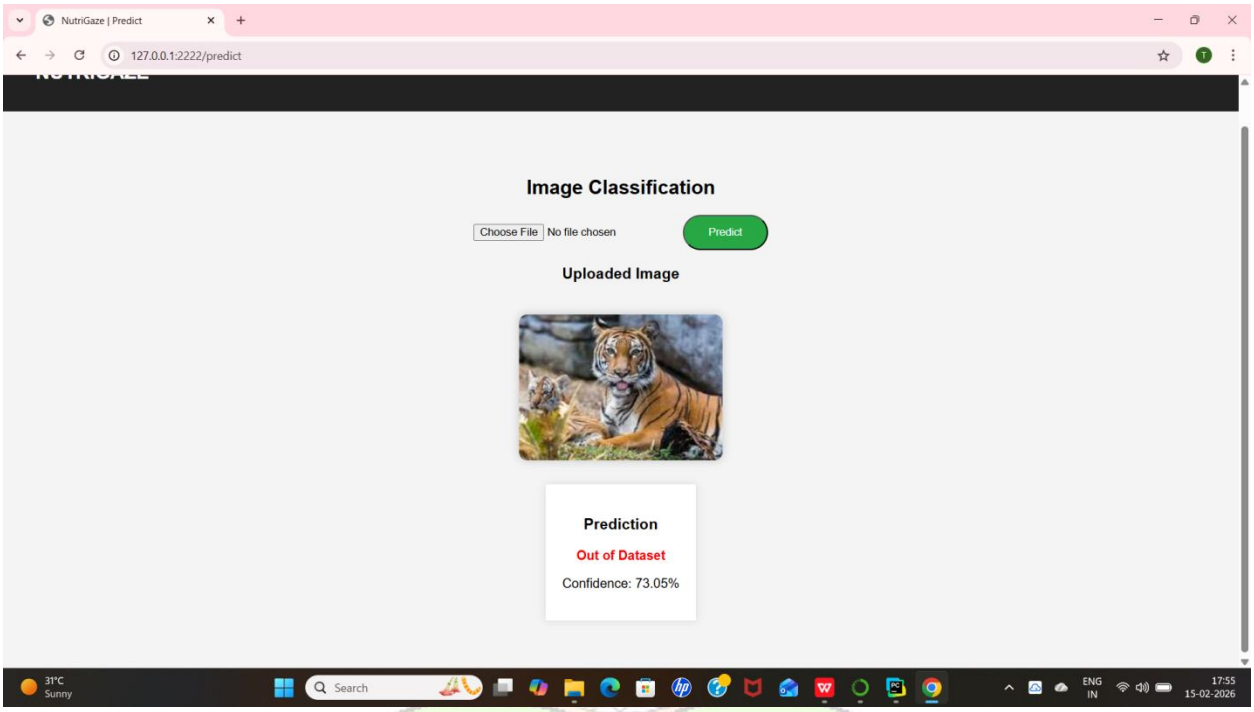


TESTING THE MODEL WITH UNKNOWN IMAGES:

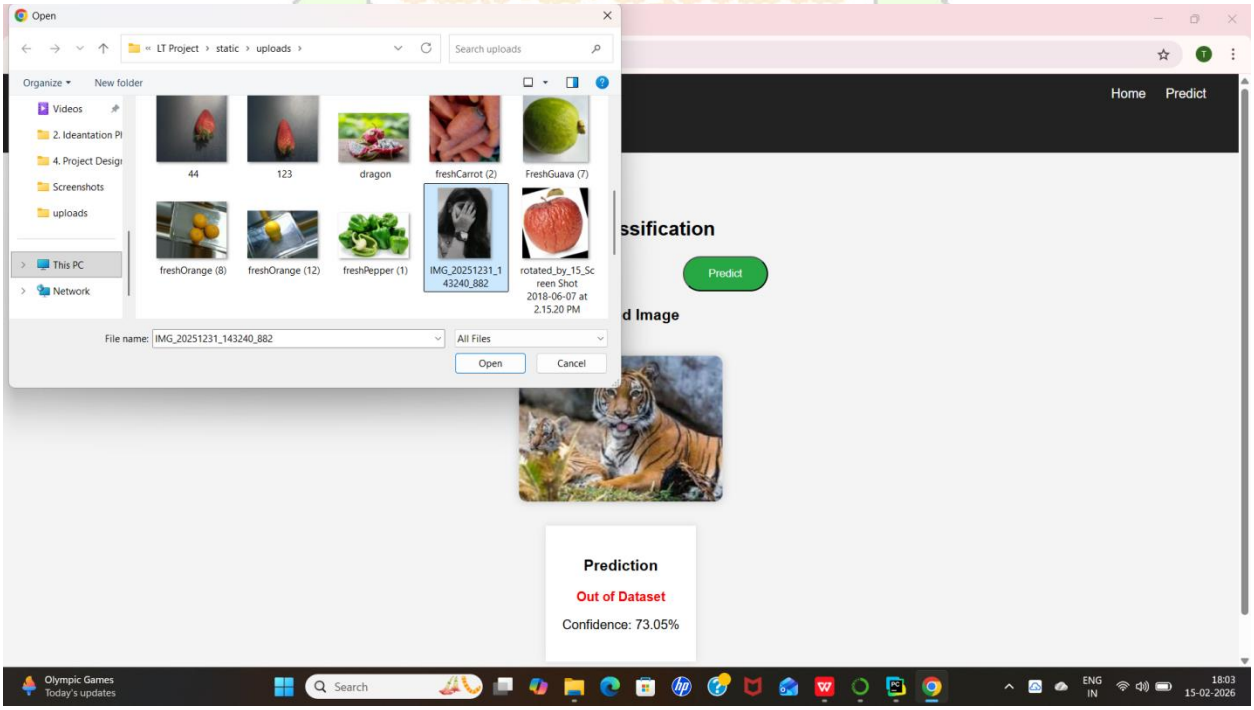


SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

OUTPUT:

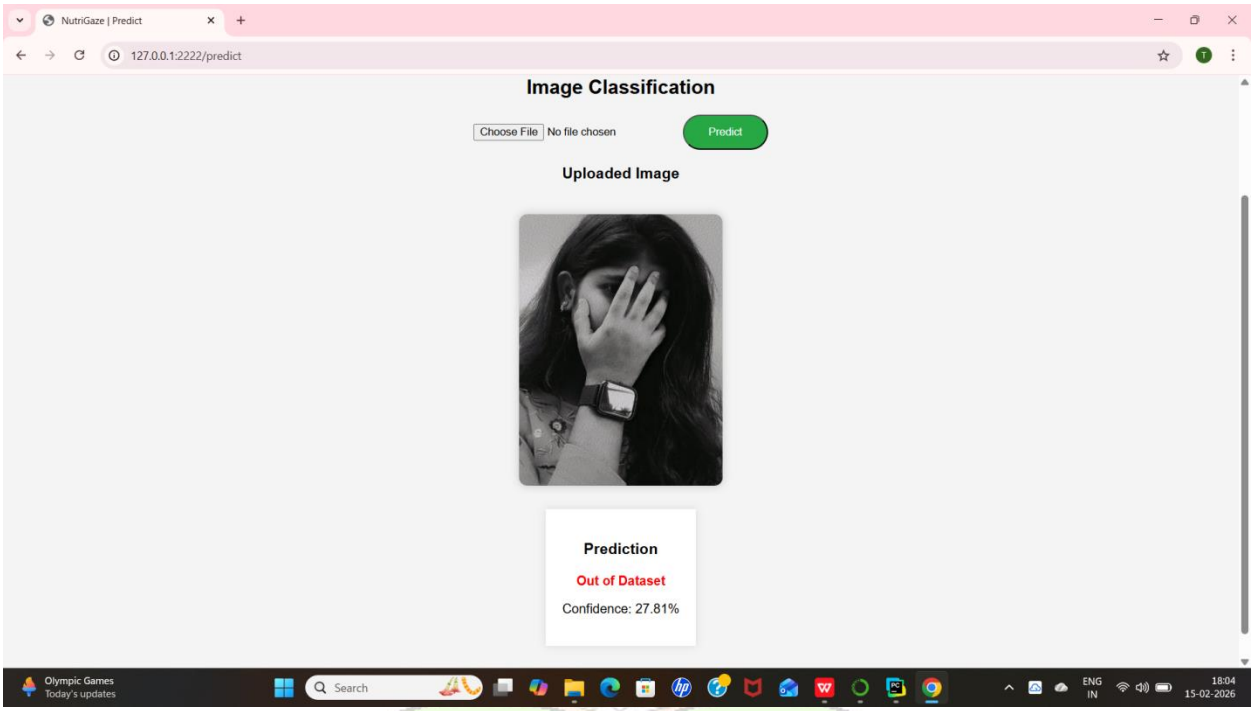


TESTING WITH HUMAN IMAGES:



SMART SORTING : TRANSFER LEARNING FOR IDENTIFYING ROTTEN FRUITS AND VEGETABLES

OUTPUT:



Gayathri Educational Society
ESTD-2001

Chapter-9

9.1 ADVANTAGES

- **High Accuracy in Detection:**

The system uses advanced transfer learning models such as MobileNetV2, ResNet50, and VGG16 to accurately detect rotten fruits and vegetables. Since these models are pre-trained on large datasets like ImageNet, they can recognize complex visual features such as discoloration, mold patches, bruises, and texture changes, resulting in high classification accuracy.

- **Reduced Manual Effort:**

Traditional sorting methods rely heavily on human inspection, which can be inconsistent and time-consuming. The Smart Sorting system automates the detection process, reducing dependency on manual labor and minimizing human errors caused by fatigue or subjective judgment.

- **Real-Time Processing:** Once deployed, the trained model can classify images within seconds. In industrial setups, the system can be integrated with conveyor belts and camera modules to perform real-time detection and automatic separation of rotten items, significantly improving operational speed.

- **Cost-Effective Implementation:**

By applying transfer learning, the system reuses knowledge from pre-trained models instead of building a deep learning model from scratch. This reduces training time, computational cost, and the need for extremely large datasets, making the solution economically feasible.

- **Scalability and Flexibility:** The architecture is scalable and can be expanded to include additional fruit and vegetable categories without redesigning the entire system. It can also be deployed as a web application, mobile application, or integrated into IoT-based industrial systems depending on the use case.

- **Improved Quality Control:**

The automated classification system ensures consistent and standardized quality assessment. This improves food safety, enhances customer satisfaction, and reduces the chances of spoiled produce reaching consumers.

Gayathri Educational Society
ESTD-2001

9.2 DISADVANTAGES

- **Dependency on Image Quality:**

The system heavily depends on clear and well-lit images. Variations in lighting conditions, shadows, background noise, or low-resolution cameras can reduce classification accuracy. Poor image capture may lead to incorrect predictions.

- **Requirement of Large and Balanced Dataset:**

Although transfer learning reduces data requirements compared to training from scratch, the system still needs a sufficiently large and well-labeled dataset. If the dataset is imbalanced or lacks variety in spoilage patterns, the model may not generalize well to real-world conditions.

- **Limited Detection to Visible Defects:**

The model primarily identifies external visual defects such as discoloration, mold, and surface damage. It may not detect internal spoilage that is not visible externally, which can limit overall quality assessment reliability.

- **Initial Development Cost:**

While operational costs are reduced in the long term, the initial setup cost can be significant. Expenses may include camera systems, hardware devices, computational resources, and development time for training models using frameworks like TensorFlow or PyTorch.

- **Need for Regular Model Updates:**

As new fruit varieties or different spoilage patterns emerge, the model may require retraining or fine-tuning. Continuous dataset updates and performance monitoring are necessary to maintain high accuracy.

- **Hardware and Infrastructure Dependency:**

For industrial deployment, integration with conveyor belts, sensors, and IoT systems requires reliable hardware infrastructure. Any hardware malfunction can disrupt the automated sorting process.

- **Risk of Overfitting:** If not properly optimized, transfer learning models such as MobileNetV2 or ResNet50 may overfit the training dataset, resulting in reduced performance on unseen data. Proper regularization and validation techniques are required to prevent this issue.

Chapter-10

10.1 CONCLUSION

The SMART SORTING system presents an innovative and practical solution to the long-standing challenge of manual fruit and vegetable quality inspection. Traditional sorting methods rely heavily on human observation, which can be inconsistent, time-consuming, and prone to errors due to fatigue or subjective judgment. By integrating deep learning and transfer learning techniques, this project introduces an automated, intelligent, and scalable system capable of accurately identifying rotten and fresh produce in real time.

The core strength of the proposed system lies in the use of pre-trained Convolutional Neural Network models such as MobileNetV2, ResNet50, and VGG16. These models, originally trained on large datasets like ImageNet, possess strong feature extraction capabilities. Through transfer learning, the system adapts these models to the specific task of classifying fresh and rotten fruits and vegetables with reduced training time and computational cost. This approach ensures high accuracy while maintaining efficiency.Throughout the project lifecycle, systematic steps were followed including problem analysis, dataset collection and preprocessing, model selection and fine-tuning, performance optimization, application integration, testing, and deployment. Each stage contributed to building a robust and reliable classification system capable of handling real-world variations such as lighting conditions, texture differences, and multiple produce categories.

The implementation of this system provides several impactful benefits. It enhances quality control standards in agricultural markets, supermarkets, and food processing industries. It reduces dependency on manual labor, increases operational speed, and ensures consistent sorting decisions. Moreover, by identifying spoiled items at an early stage, the system contributes to reducing food wastage and improving supply chain efficiency, which is crucial for economic sustainability and food safety.Despite certain limitations such as dependency on image quality and inability to detect internal spoilage, the overall effectiveness of the system demonstrates the practical potential of AI-driven solutions in agriculture and retail sectors. With further enhancements such as integration with IoT devices, real-time conveyor systems, and cloud-based monitoring dashboards, the SMART SORTING system can evolve into a fully automated smart quality inspection platform.

In conclusion, the SMART SORTING project successfully demonstrates how transfer learning and deep learning technologies can be applied to solve real-world agricultural problems. It combines accuracy, efficiency, and scalability to create a smart, reliable, and future-ready solution for identifying rotten fruits and vegetables, thereby supporting modern smart farming and food management practices.

Chapter 11

11.1 FUTURE SCOPE

SMART SORTING: Transfer Learning for Identifying Rotten Fruits and Vegetables

The SMART SORTING system establishes a strong foundation for intelligent food quality assessment, and its future scope is extensive across technological, industrial, and research domains.

- **Expansion to More Crop Varieties:**

Currently, the system focuses on selected fruits and vegetables. In the future, it can be extended to cover a wider range of produce including leafy vegetables, exotic fruits, and region-specific crops. By continuously updating the dataset and retraining models such as MobileNetV2 and ResNet50, the system can support multi-class classification for diverse agricultural products.

- **Detection of Internal Spoilage:**

The present system identifies only visible surface defects. Future enhancements may include integration with advanced imaging technologies such as hyperspectral imaging, X-ray imaging, or near-infrared sensors to detect internal decay and hidden contamination. This would significantly improve reliability in quality inspection.

- **Integration with IoT and Smart Agriculture:**

The system can be integrated with IoT-enabled conveyor belts, robotic arms, and automated packaging units for fully autonomous sorting in large-scale food processing industries. Real-time monitoring dashboards can be developed for remote supervision of sorting operations in warehouses and cold storage facilities.

- **Edge and Mobile Deployment:**

Future development can focus on lightweight model optimization for deployment on edge devices such as embedded systems or smartphones. This

would allow farmers and vendors to use the system directly in fields or marketplaces without requiring high-end computing infrastructure.

- **Cloud-Based Analytics and Data Insights:**

By integrating with cloud platforms, the system can store classification results and generate analytical reports. Data analytics can help identify spoilage patterns, seasonal quality variations, and supply chain inefficiencies. Such insights can support decision-making in inventory management and logistics.

- **Real-Time Video Processing:**

Instead of single image classification, the system can be upgraded to process real-time video streams. This enhancement would allow continuous monitoring and automatic sorting of produce moving on conveyor belts in high-speed industrial environments.

- **AI Model Improvement and Hybrid Techniques:**

Future versions may combine transfer learning with other AI approaches such as ensemble learning or attention-based deep learning models to improve classification accuracy. Advanced architectures beyond models like VGG16 could be explored to achieve better performance with reduced computational cost.

- **Integration with Supply Chain Management Systems:**

The system can be connected with enterprise resource planning and inventory systems to automatically update stock quality records. This integration would streamline operations from farm to retail distribution.

In summary, the SMART SORTING system has strong potential for technological expansion and industrial adoption. With advancements in AI, IoT, and smart agriculture technologies, it can evolve into a comprehensive automated quality inspection and food management solution for the future.

Chapter-12

12.1 SOURCE CODE

APP.PY CODE :

```
from sklearn.model_selection import train_test_split  # Fixed import
statement
import os
import shutil
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from IPython.display import Image, display
import random
from sklearn.model_selection import train_test_split
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Model
from keras.optimizers import Adam
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Model
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
from    tensorflow.keras.preprocessing.image    import    load_img,
img_to_array
import numpy as np
from sklearn.model_selection import train_test_split
from keras.callbacks import EarlyStopping
from keras.optimizers import Adam
# Set the path to the dataset - Update this path to the correct location
of your dataset
dataset_dir                                                          =
'C:/Users/tejas/PycharmProjects/PythonProject/content/Fruit       And
Vegetable Diseases Dataset'  # CHANGE THIS to your actual dataset path
with full absolute path
# Check if the directory exists before proceeding
if not os.path.exists(dataset_dir):
    raise  FileNotFoundError(f"The  dataset  directory  '{dataset_dir}'
does not exist. Please check the path.")
```

```
# Create directories for train, val, and test sets
output_dir = 'output_dataset'
os.makedirs(output_dir, exist_ok=True)
os.makedirs(os.path.join(output_dir, 'train'), exist_ok=True)
os.makedirs(os.path.join(output_dir, 'val'), exist_ok=True)
os.makedirs(os.path.join(output_dir, 'test'), exist_ok=True)
classes = os.listdir(dataset_dir)
for cls in classes:
    os.makedirs(os.path.join(output_dir, 'train', cls), exist_ok=True)
    os.makedirs(os.path.join(output_dir, 'val', cls), exist_ok=True)
    os.makedirs(os.path.join(output_dir, 'test', cls), exist_ok=True)
    class_dir = os.path.join(dataset_dir, cls)
    images = os.listdir(class_dir)[:200]
   print(cls, len(images))
    train_and_val_images, test_images = train_test_split(
        images, test_size=0.2, random_state=42
    )
    train_images, val_images = train_test_split(
        train_and_val_images, test_size=0.25, random_state=42
    )  # 0.25 x 0.8 = 0.2
    # Copy images to respective directories
    for img in train_images:
        shutil.copy(
            os.path.join(class_dir, img),
            os.path.join(output_dir, 'train', cls, img)
        )
    for img in val_images:
        shutil.copy(
            os.path.join(class_dir, img),
            os.path.join(output_dir, 'val', cls, img)
        )
    for img in test_images:
        shutil.copy(
            os.path.join(class_dir, img),
            os.path.join(output_dir, 'test', cls, img)
        )
print("Dataset split into training, validation, and test sets.")
# Define directories
```

Gayathri Educational Society
ESTD-2001

```
dataset_dir = 'output_dataset'
train_dir = os.path.join(dataset_dir, 'train')
val_dir = os.path.join(dataset_dir, 'val')
test_dir = os.path.join(dataset_dir, 'test')
# Define image size expected by the pre-trained model
IMG_SIZE = (224, 224)  # Common size for many models like ResNet, VGG,
MobileNet
# Create ImageDataGenerators for resizing and augmenting the images
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
val_test_datagen = ImageDataGenerator(rescale=1./255)
# Load and resize the images from directories
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=IMG_SIZE,
    batch_size=32,
    class_mode='binary'  # Assuming binary classification for healthy
vs rotten
)
val_generator = val_test_datagen.flow_from_directory(
    val_dir,
    target_size=IMG_SIZE,
    batch_size=32,
    class_mode='binary'
)
test_generator = val_test_datagen.flow_from_directory(
    test_dir,
    target_size=IMG_SIZE,
    batch_size=32,
    class_mode='binary',
    shuffle=False  # Do not shuffle test data
```

Gayathri Educational Society
ESTD-2001

```
)
# Print class indices for reference
print(train_generator.class_indices)
print(val_generator.class_indices)
print(test_generator.class_indices)
# Specify the path to your image folder
folder_path = 'output_dataset/train/Apple__Healthy'  # Replace with
the actual path to your image folder
# List all files in the folder
image_files  =  [f  for  f  in  os.listdir(folder_path)  if
f.endswith(('.jpg', '.png', '.jpeg'))]
# Select a random image from the list
selected_image = random.choice(image_files)
# Display the randomly selected image
image_path = os.path.join(folder_path, selected_image)
display(Image(filename=image_path))
# Specify the path to your image folder
folder_path = 'output_dataset/test/Strawberry__Healthy'  # Replace
with the actual path to your image folder
# List all files in the folder
image_files  =  [f  for  f  in  os.listdir(folder_path)  if
f.endswith(('.jpg', '.png', '.jpeg'))]
# Select a random image from the list
selected_image = random.choice(image_files)
# Display the randomly selected image
image_path = os.path.join(folder_path, selected_image)
display(Image(filename=image_path))
# Specify the path to your image folder
folder_path = 'output_dataset/test/Cucumber__Rotten'  # Replace with
the actual path to your image folder
# List all files in the folder
image_files  =  [f  for  f  in  os.listdir(folder_path)  if
f.endswith(('.jpg', '.png', '.jpeg'))]
# Select a random image from the list
selected_image = random.choice(image_files)
# Display the randomly selected image
image_path = os.path.join(folder_path, selected_image)
display(Image(filename=image_path))
# Specify the path to your image folder
```

```
folder_path = 'output_dataset/test/Strawberry__Rotten'  # Replace with
the actual path to your image folder
# List all files in the folder
image_files    =    [f    for    f    in    os.listdir(folder_path)    if
f.endswith(('.jpg', '.png', '.jpeg'))]
# Select a random image from the list
selected_image = random.choice(image_files)
# Display the randomly selected image
image_path = os.path.join(folder_path, selected_image)
display(Image(filename=image_path))
trainpath = "output_dataset/train"
testpath = "output_dataset/test"
train_datagen = ImageDataGenerator(
    rescale=1./255,
    zoom_range=0.2,
    shear_range=0.2
)
test_datagen = ImageDataGenerator(rescale=1./255)
train = train_datagen.flow_from_directory(
    trainpath,
    target_size=(224, 224),
    batch_size=20
)
test = test_datagen.flow_from_directory(
    testpath,
    target_size=(224, 224),
    batch_size=20
)  # 5, 15, 32, 50
vgg = VGG16(include_top=False, input_shape=(224, 224, 3))
for layer in vgg.layers:
    print(layer)
len(vgg.layers)
for layer in vgg.layers:
    layer.trainable = False
x = Flatten()(vgg.output)
output = Dense(30 , activation='softmax')(x)
vgg16 = Model(vgg.input, output)
vgg16.summary()
opt = Adam(learning_rate=0.0001)
```

Gayathri Educational Society
ESTD-2001

```
# Assuming you have defined your VGG16 model as vgg16
# Define Early Stopping callback
early_stopping = EarlyStopping(
    monitor='val_accuracy',
    patience=3,
    restore_best_weights=True
)
# Compile the model (you may have already done this)
vgg16.compile(
    optimizer="Adam",
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
# Train the model with early stopping callback
history = vgg16.fit(
    train,
    validation_data=test,
    epochs=15,
    steps_per_epoch=20,
    callbacks=[early_stopping]
)
vgg16.save('healthy_vs_rotten.h5')
```

TEST.PY CODE:

```
from flask import Flask, render_template, request
from tensorflow.keras.preprocessing.image import load_img,
img_to_array
import numpy as np
import tensorflow as tf
import os
app = Flask(__name__)
# Load trained model
model = tf.keras.models.load_model("healthy_vs_rotten.h5")
# Threshold: If confidence is below 80%, we say "Out of Dataset"
# Adjust this (0.0 to 1.0) based on your testing!
```

Gayathri Educational Society
ESTD-2001

```
CONFIDENCE_THRESHOLD = 0.92  # Requires 92% certainty
classes = [
    'Apple__Healthy', 'Apple__Rotten', 'Banana__Healthy',
'Banana__Rotten',
    'Bellpepper__Healthy', 'Bellpepper__Rotten', 'Carrot__Healthy',
'Carrot__Rotten',
    'Cucumber__Healthy', 'Cucumber__Rotten', 'Grape__Healthy',
'Grape__Rotten',
    'Guava__Healthy', 'Guava__Rotten', 'Jujube__Healthy',
'Jujube__Rotten',
    'Mango__Healthy', 'Mango__Rotten', 'Orange__Healthy',
'Orange__Rotten',
    'Pomegranate__Healthy', 'Pomegranate__Rotten', 'Potato__Healthy',
'Potato__Rotten',
    'Strawberry__Healthy', 'Strawberry__Rotten', 'Tomato__Healthy',
'Tomato__Rotten'
]

@app.route("/")
def home():
    return render_template("index.html")
from tensorflow.keras.applications.vgg16 import preprocess_input #
Add this import
@app.route("/predict", methods=["GET", "POST"])
def predict():
    prediction = None
    confidence = None
    image_path = None
    filename = None

    if request.method == "POST":
        file = request.files.get("pc_image")
        if file:
            filename = file.filename
            image_path = os.path.join("static/uploads", filename)
            os.makedirs("static/uploads", exist_ok=True)
            file.save(image_path)

            # □ 1. Standard VGG16 Preprocessing
```

```
            img = load_img(image_path, target_size=(224, 224))
            img_array = np.array(img)
            # If you used rescale=1/255 in training, keep the next
line.
            # If not, comment it out and use
preprocess_input(img_array)
            img_array = img_array / 255.0
            img_array = np.expand_dims(img_array, axis=0)
            # □ 2. Get All Probabilities
            probs = model.predict(img_array)[0]
            class_index = np.argmax(probs)
            max_prob = probs[class_index
            # □ 3. Calculate "Confusion" (Gap between top 2 guesses)
            sorted_probs = np.sort(probs)
            top_choice = sorted_probs[-1]
            second_choice = sorted_probs[-2]
            gap = top_choice - second_choice
            # □ 4. STRICT FILTERING
            # A tiger shouldn't have a high gap, and its confidence
shouldn't be massive.
            # We require at least 85% confidence AND a 40% gap over
the next best guess.
            if top_choice < 0.85 or gap < 0.40:
                prediction = "Out of Dataset"
                confidence = top_choice * 100
            else:
                prediction = classes[class_index]
                confidence = top_choice * 100

    return render_template("portfolio-details.html",
predict=prediction,
                           confidence=confidence,
image_path=filename)

if __name__ == "__main__":
    app.run(debug=True, port=2222)
```

INDEX.HTML CODE :

```
<!DOCTYPE html>
<html>
<head>
    <title>NutriGaze | Home</title>
    <link      rel="stylesheet"      href="{{      url_for('static',
filename='css/style.css') }}">
</head>
<body>
<header class="navbar">
    <h2>NUTRIGAZE</h2>
    <nav>
        <a href="{{ url_for('home') }}">Home</a>
        <a href="{{ url_for('predict') }}">Predict</a>
    </nav>
</header>
<section class="hero">
    <h1>GreenGuard Insights</h1>
    <p>Detect freshness of fruits & vegetables using AI</p>
    <a href="{{ url_for('predict') }}" class="btn">Get Started</a>
</section>
</body>
</html>
```

PORTFOLIO-DETAILS.HTML CODE:

```
<!DOCTYPE html>
<html>
<head>
    <title>NutriGaze | Predict</title>
    <link      rel="stylesheet"      href="{{      url_for('static',
filename='css/style.css') }}">
</head>
<body>
<header class="navbar">
    <h2>NUTRIGAZE</h2>
```

Gayathri Educational Society

```
    <nav>
        <a href="{{ url_for('home') }}">Home</a>
        <a class="active" href="{{ url_for('predict') }}">Predict</a>
    </nav>
</header>
<section class="predict-section">
    <h2>Image Classification</h2>
    <form method="POST" enctype="multipart/form-data">
        <input type="file" name="pc_image" required>
        <button class="btn green" type="submit">Predict</button>
    </form>
  {% if image_path %}
<div class="preview">
    <h3>Uploaded Image</h3>
    <img src="{{ url_for('static', filename='uploads/' + image_path)
}}" alt="Preview">
</div>
{% endif %}
{% if predict %}
<div class="result-box">
    <h3>Prediction</h3>
    <p><strong style="color: {{ 'red' if predict == 'Out of Dataset'
else '#28a745' }}">
        {{ predict }}
    </strong></p>
    <p>Confidence: {{ "%.2f"|format(confidence) }}%</p>
</div>
{% endif %}</section>
</body></html>
```

STYLE.CSS CODE:

```
body {
    font-family: Arial, sans-serif;
    background: #f4f4f4;
    margin: 0;
}
.navbar {
```

```
    background: #222;
    padding: 15px 40px;
    color: white;
    display: flex;
    justify-content: space-between;
}
.navbar a {
    color: white;
    margin-left: 20px;
    text-decoration: none;
}
.hero {
    padding: 120px;
    text-align: center;
}
.btn {
    background: #28a745;
    color: white;
    padding: 12px 30px;
    border-radius: 25px;
    text-decoration: none;
}
.predict-section {
    padding: 60px;
    text-align: center;
}
.preview img {
    width: 250px;
    margin-top: 20px;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0,0,0,.2);
}
.result-box {
    margin-top: 25px;
    background: white;
    padding: 20px;
    display: inline-block;
    box-shadow: 0 0 10px rgba(0,0,0,.1);
}
```

Gayathri Educational Society
ESTD-2001

```
.hero {
    padding: 120px;
    text-align: center;
    color: white;              /* ensures text is readable */
    background: url('../images/bg-home.jpg') no-repeat center center;
    background-size: cover;     /* makes it fill the section */
    position: relative;
}
/* Optional: add overlay for better contrast */
.hero::before {
    content: "";
    position: absolute;
    top: 0; left: 0; right: 0; bottom: 0;
    background: rgba(0,0,0,0.4);  /* dark overlay */
    z-index: 0;
}
.hero h1,
.hero p,
.hero .btn {
    position: relative;  /* keep text above overlay */
    z-index: 1;
}
.hero {
    height: 100vh;   /* full viewport height */
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;}
```

BG-HOME.JPG IMAGE :

Gayathri Educational Society

12.2 DATASET LINK

DATA COLLECTION AND PREPARATION MILESTONE :

From this milestone section , Go to “COLLECT THE DATASET “ and click on the dataset link provided in the guided project section as shown below .

Click on the link , it will navigate into “KAGGLE” website where we can find our dataset .

DATASET LINK: Fruit and Vegetable Disease (Healthy vs Rotten)

https://www.kaggle.com/datasets/muhammad0subhan/fruit-and-vegetable-disease-healthy-vs-rotten

Steps to download the dataset :

- ✧ Open the dataset page.
- ✧ Click the Download button (top right corner).
- ✧ The dataset will download as a ZIP file.
- ✧ Extract the ZIP file on your system.

DATASET QR:

12.3 GITHUB LINK :

https://github.com/chitti4569/LTVIP2026TMID90622-KONDURU-SUKANYA

GITHUB QR CODE:

PROJECT DEMO LINK :

https://drive.google.com/file/d/1h3LduVYypdaTeCsFzxACevRvxXjBTNRs/view?usp=drive_link

PROJECT DEMO QR CODE: