# Smart Sorting: Transfer Learning for Identifying Rotten Fruits and Vegetables

## Full Stack Development Documentation

## 1. Introduction

**Project Title:** Smart Sorting: Transfer Learning for Identifying Rotten Fruits and Vegetables

**Team ID:** LTVIP2026TMIDS65633

- **Team Leader :** Thammigani Gopi
- **Team member :** Umar Ansari
- **Team member :** Vaddireddy Pavan Kumar Reddy
- **Team member :** R Nandini

## 2. Project Overview

**Purpose**

Smart Sorting is an AI-based system that leverages transfer learning to automatically detect rotten fruits and vegetables from fresh ones using image classification. The project addresses inefficiencies in manual sorting within food supply chains, especially in supermarkets, food processing units, and storage facilities.

**Key Features**

- **Real-time Image Classification:** Automatic detection of fresh vs. rotten produce using deep learning
- **Transfer Learning Implementation:** Utilizes pre-trained CNN models (ResNet, VGG, MobileNet) for high accuracy with minimal training time
- **Image Upload Interface:** Web-based platform for uploading and processing fruit/vegetable images
- **Visual Dashboard:** Interactive dashboard displaying classification results and statistics
- **Database Logging:** Comprehensive logging of classification results and system performance
- **RESTful API:** Backend API for image processing and classification services
- **Responsive Design:** Mobile-friendly interface for various device compatibility

## 3. Architecture

**Frontend (HTML Templates)**

- **Template-based Architecture**: HTML templates stored in templates/ directory
- **Responsive Design:** Mobile-friendly interface for image upload and results display
- **Interactive Elements:** JavaScript for dynamic content and user interactions
- **Image Upload Interface:** Form-based image upload with preview functionality
- **Results Display:** Real-time classification results with confidence scores

**Backend (Python Flask/Django)**

- **Flask/Django Framework**: Python web framework for handling HTTP requests
- **Model Integration:** Direct integration with TensorFlow/Keras model (healthy_vs_rotten.h5)
- **File Upload Handling:** Python-based file upload and processing
- **Image Processing:** PIL/OpenCV for image preprocessing before classification
- **API Endpoints:** RESTful endpoints for image classification services

**Data Storage**

- **File-based Storage:** Images stored in uploads/ directory
- **Model Storage:** Trained model saved as healthy_vs_rotten.h5
- **Dataset Management:** Organized dataset in Fruit And Vegetable Diseases Data/
- **Results Logging:** Classification results and metadata storage

**ML Model Architecture**

- **Transfer Learning:** Fine-tuned pre-trained CNN models (ResNet/VGG/MobileNet)
- **Image Preprocessing:** Standardized image resizing, normalization, and augmentation
- **Classification Engine:** Binary classification (Fresh vs. Rotten) with confidence scores
- **Model Serving:** Integration with backend API for real-time inference

## 4. Setup Instructions

**Prerequisites**

- Python 3.8+
- TensorFlow/Keras 2.x
- Flask or Django framework
- OpenCV or PIL for image processing
- Jupyter Notebook (for model development)
- Git

**Installation**

1. **Clone the Repository**

bash
git clone https://github.com/prasanna3001200/Smart-Sorting-Transfer-Learning-for-Identifying-Rotten-Fruits-and-Vegetables.git
cd SmartBridgeProject

2. **Set up Python Environment**

bash
*# Create virtual environment*
python -m venv venv

# Activate virtual environment
# Windows:
venv\Scripts\activate
# Linux/Mac:
source venv/bin/activate

3. **Install Dependencies**

bash

```
pip install tensorflow keras flask opencv-python pillow numpy matplotlib
# Or if using requirements.txt:
pip install -r requirements.txt
```

4. **Verify Model File**
    - Ensure healthy_vs_rotten.h5 is present in the root directory
    - If model needs retraining, run Fruit Classification.ipynb
5. **Set up Directories**

bash

```
# Create necessary directories if not present
mkdir -p uploads output_dataset
```

6. Environment Configuration Create .env file or configure directly in app:
7. UPLOAD_FOLDER=uploads
8. MODEL_PATH=healthy_vs_rotten.h5

## 5. Folder Structure

**Project Structure (Based on Current Implementation)**

```
SmartBridgeProject/
├── Fruit And Vegetable Diseases Data/    # Dataset for training/testing
├── output_dataset/                       # Processed dataset outputs
├── templates/                            # HTML templates for web interface
├── uploads/                              # User uploaded images storage
├── app                                   # Main application file (Python Flask/Django)
├── Fruit Classification.ipynb            # Jupyter notebook for ML model development
├── healthy_vs_rotten.h5                  # Trained ML model file
└── [Additional files]
```

**Detailed Structure Analysis**

**Machine Learning Components:**

- Fruit Classification.ipynb - Core ML development notebook containing:
    - Data preprocessing and augmentation
    - Transfer learning model implementation
    - Model training and validation
    - Performance evaluation and visualization

**Data Management:**

- Fruit And Vegetable Diseases Data/ - Raw dataset containing:
    - Fresh fruit and vegetable images
    - Rotten/diseased produce images
    - Organized by categories and conditions
- output_dataset/ - Processed and augmented dataset ready for training

**Web Application:**

- app - Main application server (likely Python-based)
- templates/ - Frontend HTML templates for web interface

- uploads/ - Directory for storing user-uploaded images for classification

**Model Artifacts:**

- healthy_vs_rotten.h5 - Trained deep learning model (Keras/TensorFlow format)
    - Contains weights and architecture
    - Ready for inference and deployment

## 6. Running the Application

**Development Mode**

**Start the Application:**

bash

```
# Activate virtual environment
source venv/bin/activate  # Linux/Mac
# or
venv\Scripts\activate     # Windows

# Run the main application
python app
# Application runs on http://localhost:5000 (or configured port)
```

**Model Development/Training:**

bash

```
# Open Jupyter Notebook for model development
jupyter notebook
# Open Fruit Classification.ipynb for model training and evaluation
```

**Production Mode**

bash

```
# Set production environment variables
export FLASK_ENV=production

# Run with production server (e.g., Gunicorn)
pip install gunicorn
gunicorn -w 4 -b 0.0.0.0:5000 app:app
```

## 7. API Documentation

**Base URL**

http://localhost:5000

## 8. Authentication

**Implementation Details**

- **JWT Token-based Authentication:** Secure token generation and validation
- **Password Hashing**: bcrypt for secure password storage
- **Protected Routes:** Middleware-based route protection
- **Token Expiration:** Configurable token lifecycle management
- **Refresh Token:** Optional refresh token implementation for extended sessions

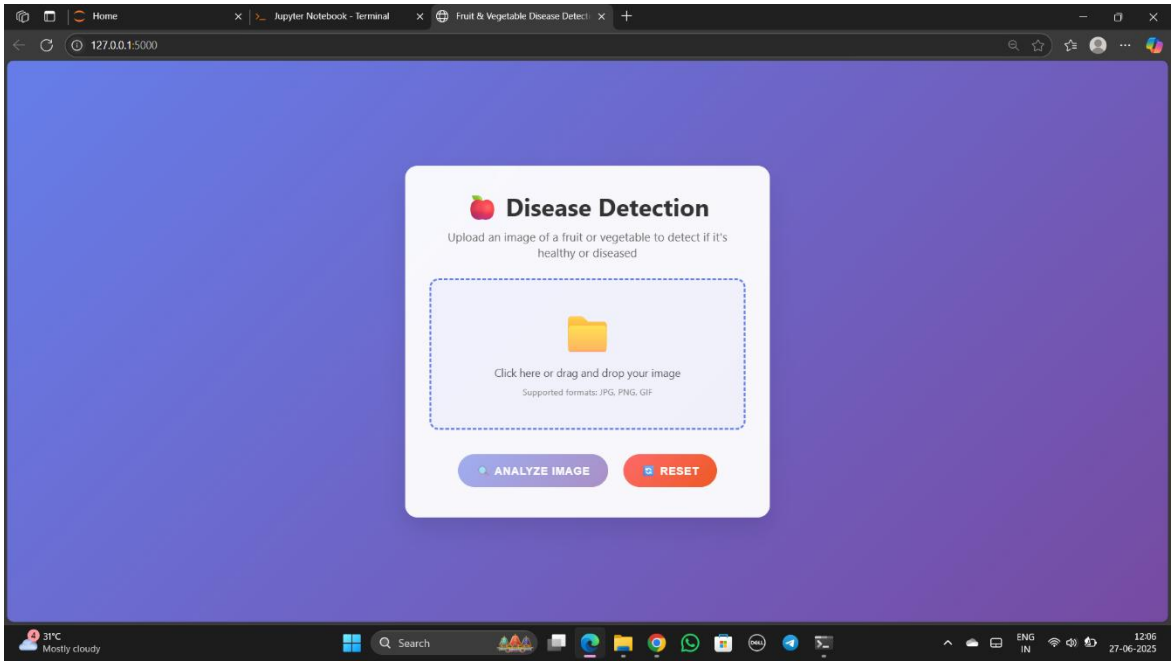
### Security Features

- Input validation and sanitization
- Rate limiting for API endpoints
- CORS configuration for cross-origin requests
- Secure file upload validation (file type, size limits)

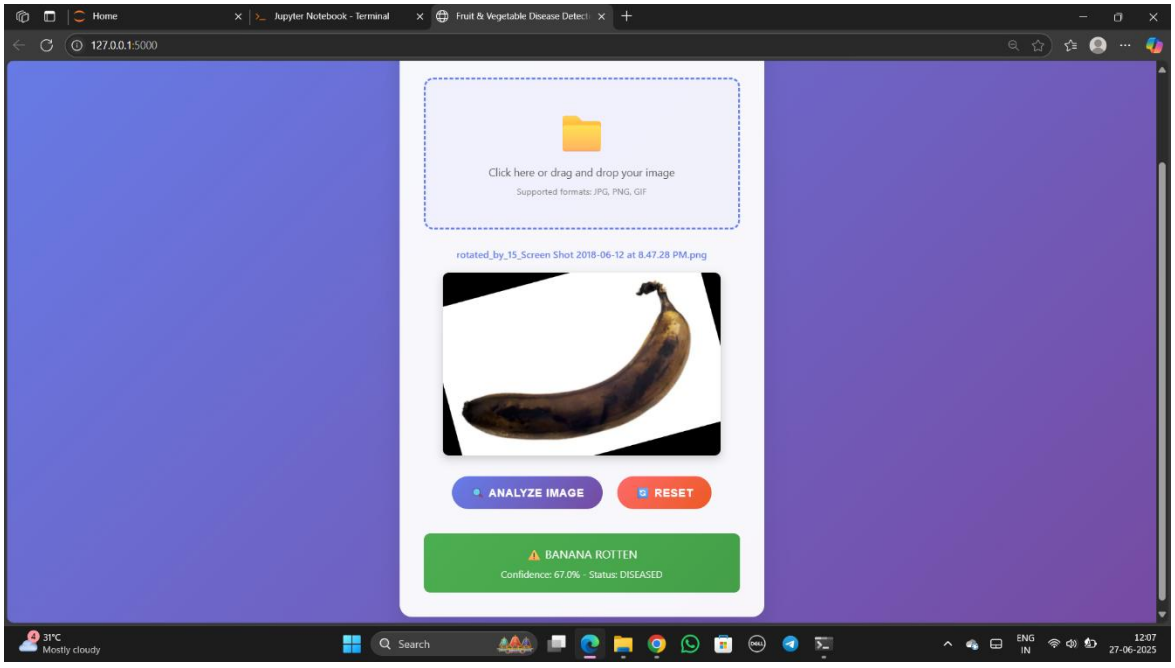
## 9. User Interface

### Main Features

### Image Upload Interface




### Classification Results Display




- Intuitive drag-and-drop image upload
- Real-time classification feedback

- Interactive charts and visualizations
- Clean, modern interface design
- Loading states and progress indicators

## 10. Testing

### Testing Strategy

- **Unit Testing:** Jest for individual component and function testing
- **Integration Testing:** API endpoint testing with Supertest
- **Model Testing:** ML model accuracy and performance validation
- **End-to-End Testing:** Cypress for complete user workflow testing

### Testing Tools

- **Frontend:** Jest, React Testing Library
- **Backend:** Jest, Supertest, MongoDB Memory Server
- **ML Model:** Python unittest, model evaluation metrics
- **E2E:** Cypress

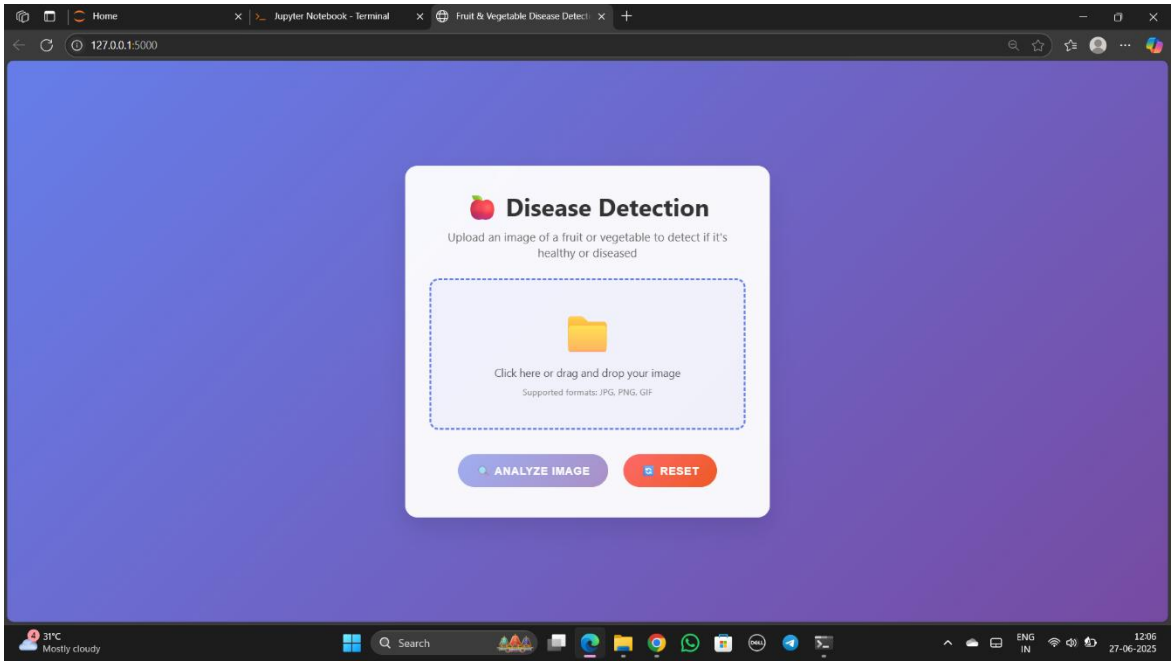
### Test Coverage

- Minimum 80% code coverage requirement
- Automated testing pipeline with CI/CD integration

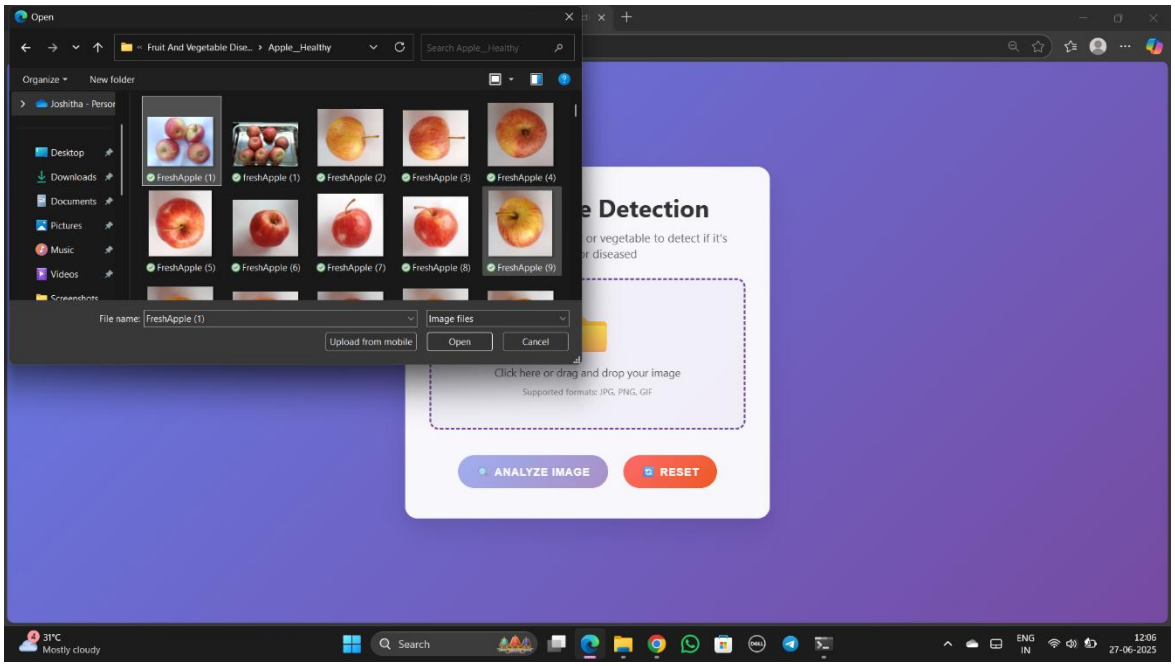
## 11. Screenshots or Demo

### Application Screenshots

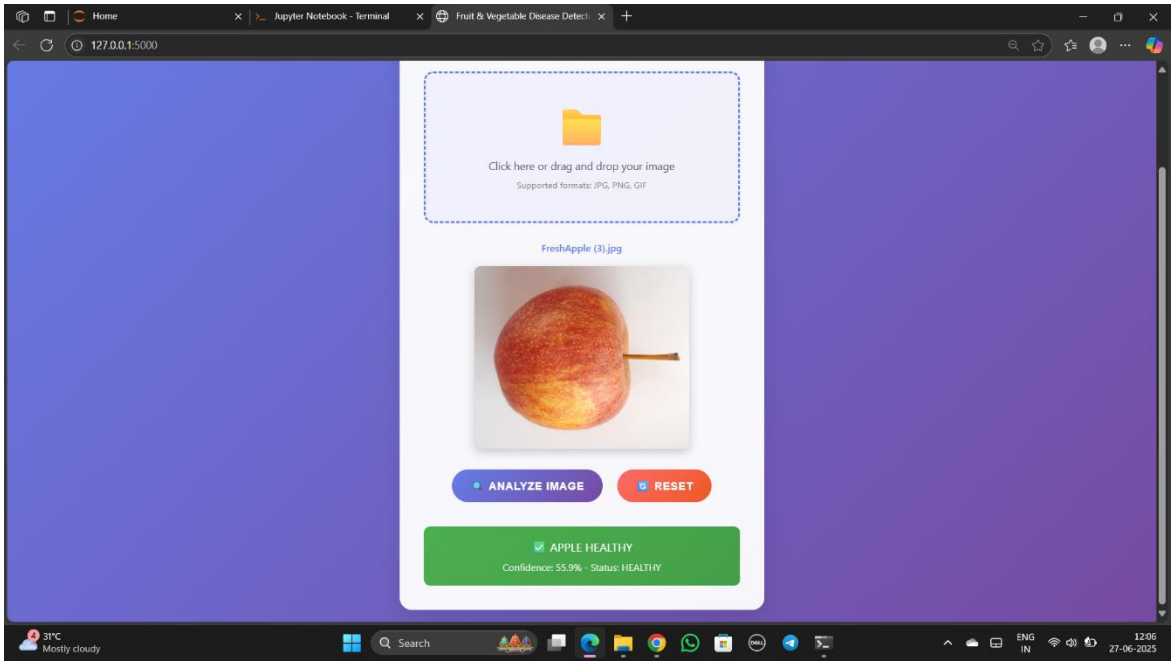
### Home Page




### Upload Process




**Results Visualization**




## 12. Known Issues

### Current Limitations

- **Model Accuracy:** Classification accuracy varies with image quality and lighting conditions
- **File Size Limits:** Maximum upload size limited to 10MB per image
- **Processing Time:** Classification may take 2-3 seconds for high-resolution images
- **Browser Compatibility:** Optimal performance on modern browsers (Chrome, Firefox, Safari)

### Planned Fixes

- Implement image preprocessing improvements for better accuracy
- Add progress indicators for longer processing times

- Optimize model inference for faster response times

## 13. Future Enhancements

**Technical Improvements**

- **Real-time Processing:** WebSocket integration for live camera feed classification
- **Multi-class Classification:** Expand to classify specific types of fruits and vegetables
- **Mobile App:** Native mobile application development
- **Edge Computing:** Deploy models on edge devices for offline processing

**Business Features**

- **Batch Processing:** Support for multiple image uploads and bulk classification
- **API Integration:** Third-party API for integration with existing systems
- **Reporting:** Advanced reporting and export functionality
- **User Management:** Admin panel for user management and system monitoring

**Machine Learning Enhancements**

- **Model Ensemble:** Combine multiple models for improved accuracy
- **Continuous Learning:** Implement feedback loop for model improvement
- **Custom Training:** Allow users to train models with their specific datasets
- **Explainable AI:** Add model interpretability features

## Social Impact & Business Model

**Social Impact**

- **Reduction in Food Waste:** Early detection prevents unnecessary disposal of good produce
- **Improved Public Health:** Removes spoiled items before reaching consumers
- **Environmental Benefits:** Reduces organic waste in landfills and greenhouse gas emissions

**Business Model**

- **B2B Services:** Enterprise solutions for food processing facilities and supermarkets
- **B2C Applications:** Consumer-facing mobile apps for household use
- **API Licensing:** Revenue through API usage and integration services
- **Subscription Model:** Tiered pricing for different user categories

**Project Team:** LTVIP2025TMID59623
**Documentation Date:** Feb 19, 2026