VIT BHOPAL

REPORT FILE

on

Plant Leaf Disease Detection Model

Name: Deepaansh Kapoor

Registration Number: 25BCE10487

Faculty Coordinator: Dr. Monika Vyas

Course: Vityarthi Python Course

SUMMARY

This paper describes the development and application of an AI based system for plant leaf disease detection through deep learning- transfer learning. A 96% accuracy had been gained for plant diseases that classified in 39 labels from Plant Village data set. The app is built on pretrained EfficientNetV4 model, fine-tuned for crop disease detection and deployed using Flask web application running real time predictions.

1.INTRODUCTION:

(a) Background-

There have been many instance of plant diseases and their threat on agriculture production all over the world. Conventional disease detection depends on visual screening by the agriculture experts which is time-consuming, subjective and not suitable for megafarms. Early and accurate identification of a disease allows farmers to treat them , cutting down on crop loss and pesticide usage.

(b) Deep Learning in Agriculture-

Recent time studies has enabled automated disease detection systems. Convolutional Neural Networks (CNNs) can analyze leaf images and identify disease patterns with high accuracy. Pre-trained models like EfficientNet and MobileNet—reduces training time and data requirements while maintaining excellent performance

(c) Significance-

The system bridges the gap between advanced machine learning technology and practical agricultural applications:

- Real-time disease detection for farmers and agricultural professionals

- Resource optimization by early disease identification

- Treatment guidance through cause-remedy associations

- Data-driven agriculture leveraging deep learning capabilities

2. PROBLEM STATEMENT:

(a) Challenges in Current Agricultural Practices-

-Drawbacks of manual inspection: High expertise is needed to ensure proper Diagnosis of diseases Thus, it does not scale well to other diagnoses.

-Variability of Diagnosis: There may be pathological inconsistency in identifying the disease, with varying levels of experience driven according to expertise.

-Time-Intensive Process : A lot of time needs to invested into analysis and field visits

-Geographic Restrictions: Agroecological areas in remotecannot access expert advice.

-Cost implication: It is costly to involve professional disease diagnosis service for small-holder farmers

(b) Proposed Solution-

An automated AI-based system that can:
- Analyze leaf images in real-time
- Identify plant diseases across 39 categories with 96% accuracy
- Provide disease causes and treatment recommendations
- Be accessible via web interface on any device

3. FUNCTIONAL REQUIREMENTS:

(a) Core Functionality -

Requirement	Description
Image Upload	System will receive leaf images in standard formats (JPG, PNG)
Disease Classification	Model would need to identify the input images as one of 39 diseases
Accuracy Threshold	At least 90% classification accuracy on the test data set
Prediction Output	Show predicted disease class with confidence score
Disease Information	List cause and remedy for identified ailment
Real-time Processing	Predictions as of 2 seconds after image upload
Error Handling	And the system should cover invalide images responsing error message

(b) User Interactions -

-Upload leaf image via web interface

-View prediction results with disease name, probability, causes, and treatment

-Retry with different images

-Access system from web browser

(c) Backend Processing –

-Load pre-trained EfficientNetV4 model

-Preprocess input images to model specifications

-Execute inference

-Return prediction with confidence scores

-Query disease knowledge base for causes and cures

4. NON-FUNCTIONAL REQUIREMENTS:

(a) Performance Requirements –

-Response Time: Prediction within 1-3 seconds

-Model Accuracy: ≥96% on test dataset

-Scalability: Handle 10+ simultaneous users

(b) Reliability and Availability –

-Uptime: System available 99% of operational time

-Fault Tolerance: Error handling for invalid inputs

-Data Integrity: Uploaded images processed without corruption

(c) Usability Requirements –

- Web interface that does not require technical knowledge
- Clear presentation of results
- Mobile-responsive design
- Accessible error messages

(d) Security and Compliance –

- Input validation to prevent harmful image uploads
- HTTPS support for production deployment
- Privacy: No storage of uploaded images after analysis
- Compliance with agricultural data standards

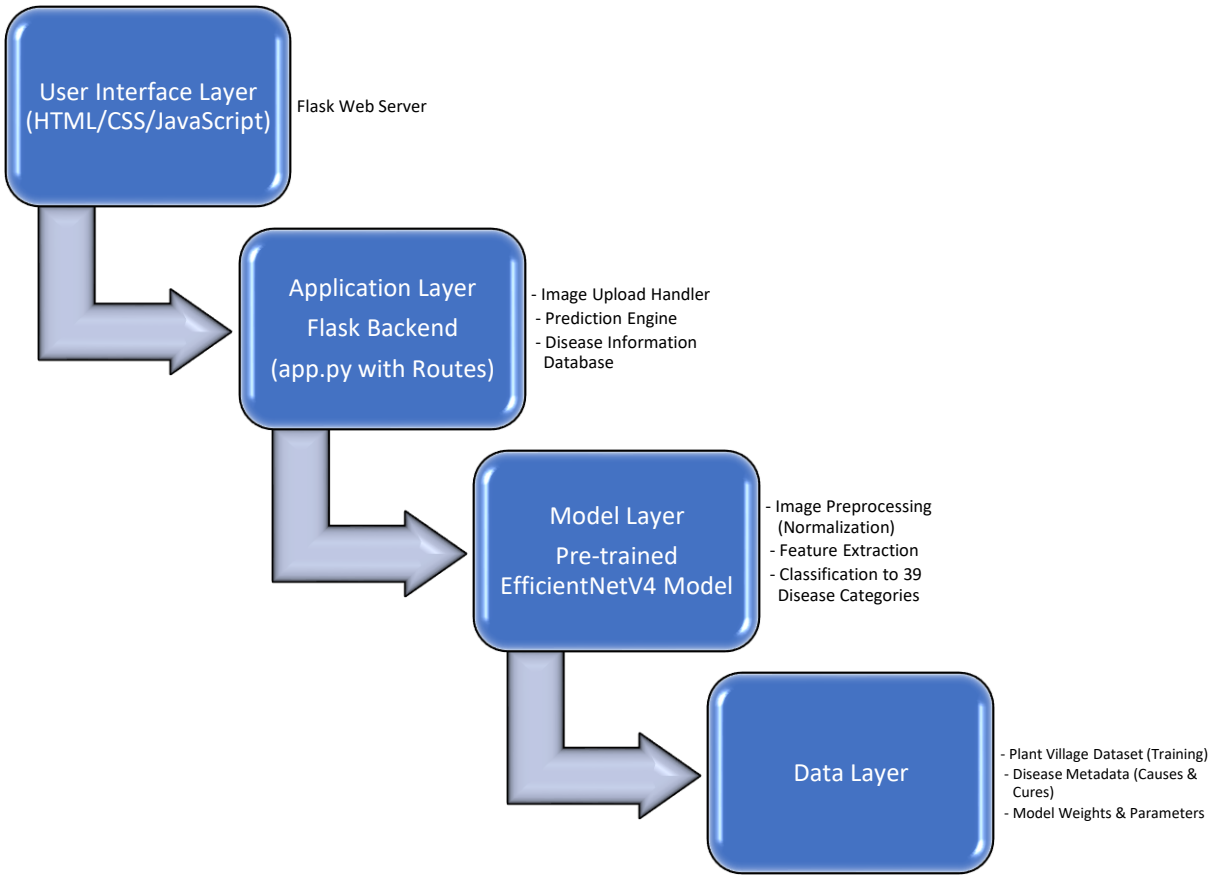
(e) Maintainability –

- Clean, well-documented code
- Modular design for easy updates

- Version control using GitHub
- Model versioning for future improvements

5. SYSTEM ARCHITECTURE:

(a) High-Level Architecture –



(b) Technology Stack–

- **Frontend:** HTML5, CSS3, JavaScript
- **Backend:** Flask (Python Web Framework)
- **Deep Learning:** TensorFlow, Keras
- **Pre-trained Model:** EfficientNetV4
- **Image Processing:** NumPy, PIL
- **Dataset:** Plant Village (TensorFlow Datasets)
- **Development Environment:** Google Colab, VS Code
- **Version Control:** GitHub

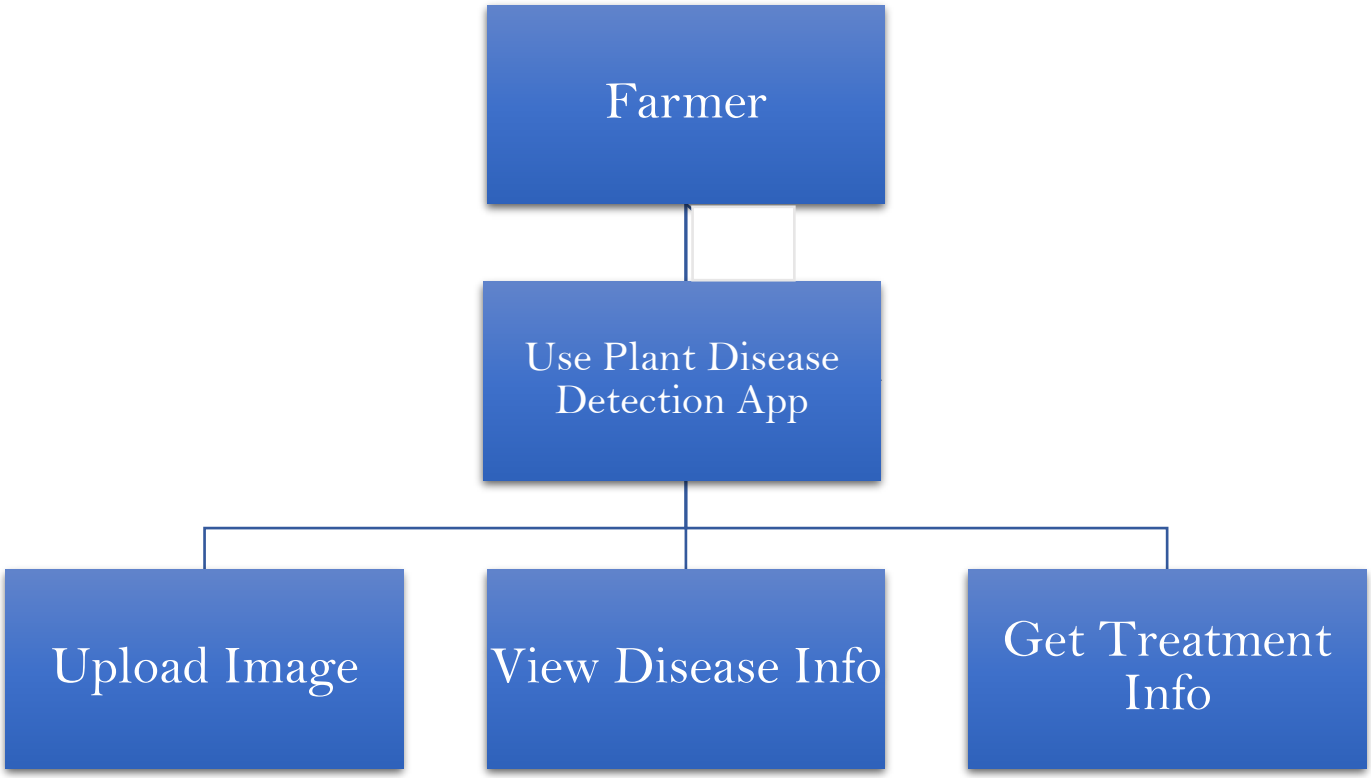
(c) Data Flow–

- User uploads leaf image via web interface
- Image sent to Flask server (backend)

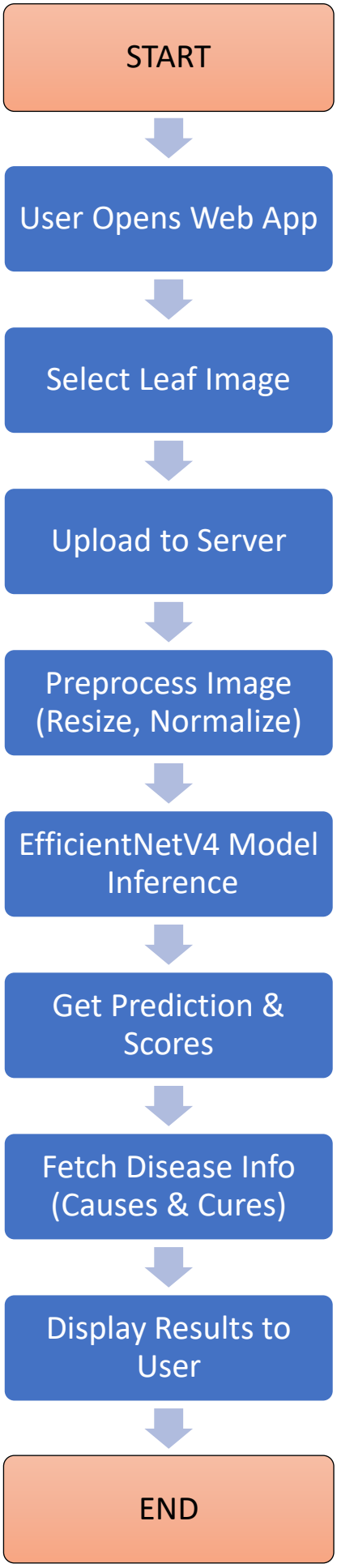
- Image pre processed (resized, normalized)
- EfficientNetV4 model performs inference
- Model outputs prediction with confidence scores
- Backend retrieves disease information (causes, cures)
- Results displayed to user with visualizations

6. DESIGN DIAGRAMS:

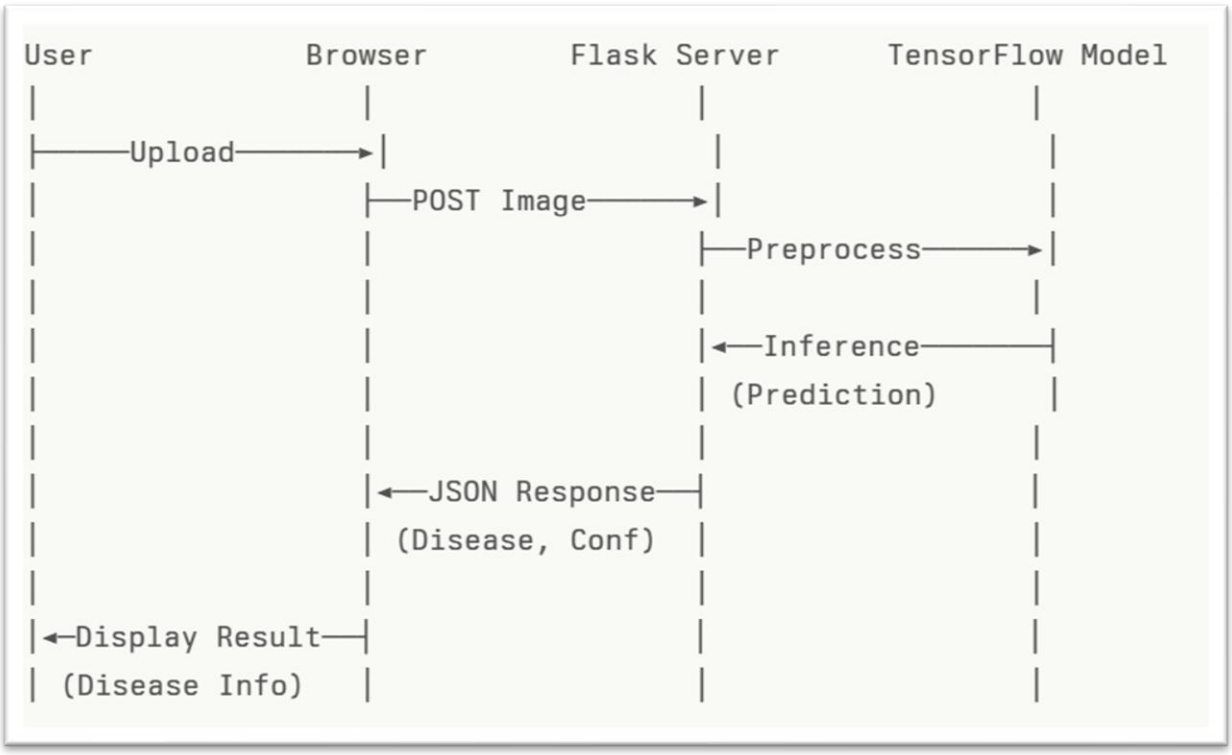
(a) **Use Case Diagram** –



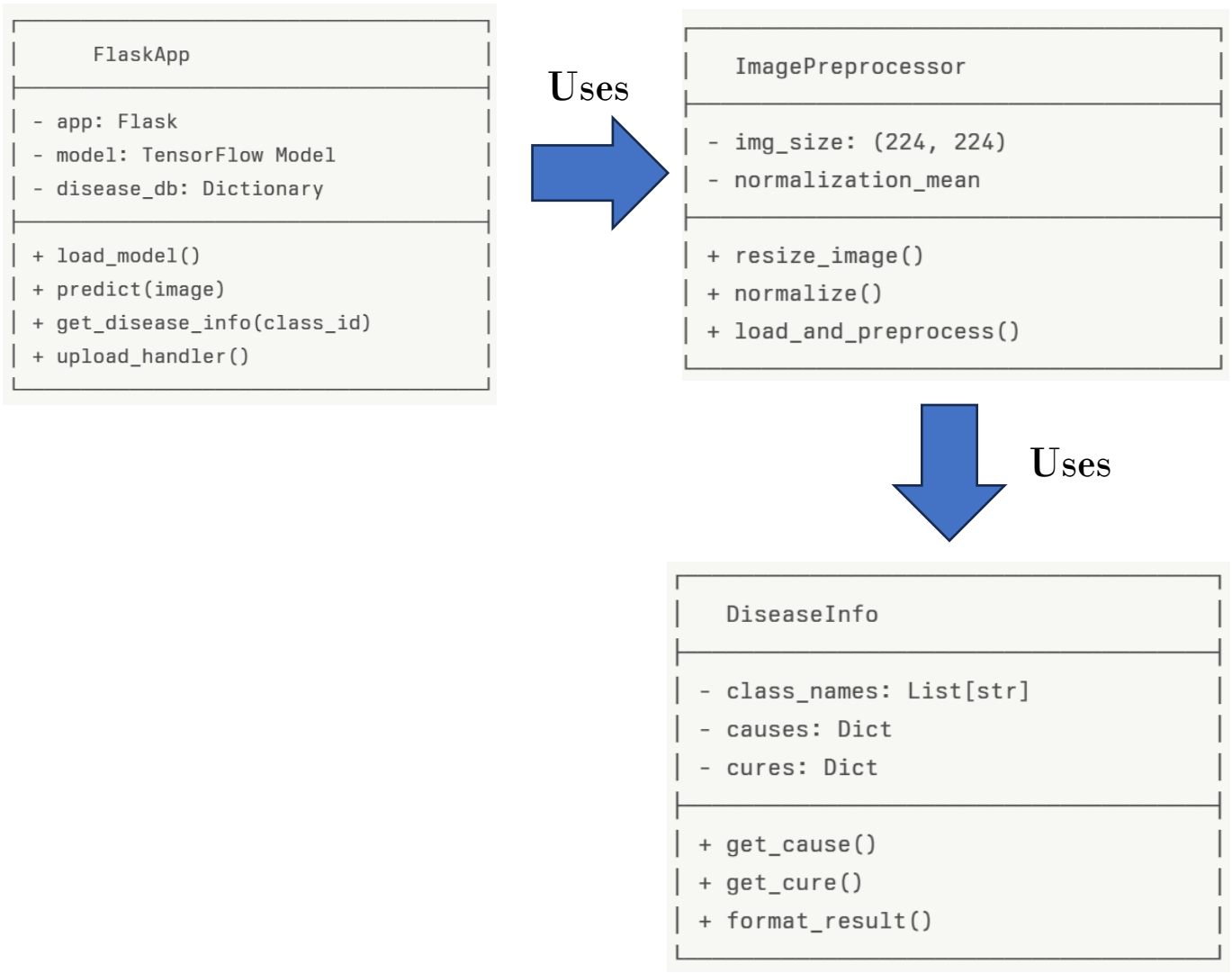
(b) Workflow Diagram –



(c) Sequence Diagram –



(d) Class/Component Diagram –



7. DESIGN DECISIONS AND RATIONALE:

(a) Model Selection: EfficientNetV4–

Decision: Use EfficientNetV4 instead of MobileNetV2

Rationale:

• Balanced Performance: EfficientNetV4 offers better accuracy than MobileNetV2 and remains computationally efficient.

• Transfer Learning: It is pre-trained on ImageNet, which helps speed up convergence.

• Model Efficiency: Its scalable design works well with different hardware limitations.

• Production-Ready: It is well-documented and widely supported in the TensorFlow ecosystem.

(b) Transfer Learning Approach–

Decision: Implement two-phase training (frozen base model, then fine-tuning)

Rationale:

•Phase 1 (Frozen Base): Leverages ImageNet-learned features, reduces overfitting on smaller agricultural dataset

•Phase 2 (Fine-tuning): Unfreezes last 100 layers of EfficientNetV4 to adapt to plant disease specifics

•Benefit: Achieves 96% accuracy with limited computational resources and training time

(c) Data Augmentation–

Decision: We should apply data augmentation techniques (rotation, flipping, brightness adjustment)

Rationale:

• Robustness: Improves model generalization to slightly different image conditions

• Overfitting Prevention: Reduces overfitting by presenting diverse variations of training images

(d) Data Splitting Strategy–

Decision: The data here was divided into 80% training, 10% validation, and 10% testing

Reasoning:

• Adequate test set: 10% gives approximately 5,400 images for a thorough performance evaluation

• Validation control: 10% validation set makes it possible to stop the training early

(e) No Database for Image Storage –

Decision: Keep all image processing in memory and do not save files to disk

Rationale:

• Privacy: Images that are sent to the server are not kept after the result is generated

• Security: Minimizes the potential for malicious activities and data security issues

• Performance: Processing can be done at a higher speed as there is no need to perform disk.

8. IMPLEMENTATION DETAILS:

(a) Data Preparation–

Step 1: Dataset Download

- Source: TensorFlow Datasets (Plant Village)
- Size: ~54,300 images
- Categories: 39 disease classes

Step 2: Data Splitting

- Used `split-folders` library to divide dataset into test
- Training: 80%
- Validation: 10%
- Testing: 10%

Step 3: Image Preprocessing

- Resized all images to EfficientNetV4 input size
- Normalized pixel values using EfficientNetV4's preprocessing method

(b) Model Evaluation–

Test Dataset Performance:

• Accuracy: 96%

• Precision Most classes showed very high precision

• Recall: Disease detection was very strong

• Confusion Matrix: Very few examples of cross-class confusion

Training Metrics:

• Training Loss: Went down from around 2.5 to around 0.15

• Training Accuracy: 98-99%

• Validation Accuracy: 96-97%

(c) Frontend Application–

HTML Structure:

•File upload input

•Submit button

•Display area for results

•Disease information panel with causes and cures

JavaScript Functionality:

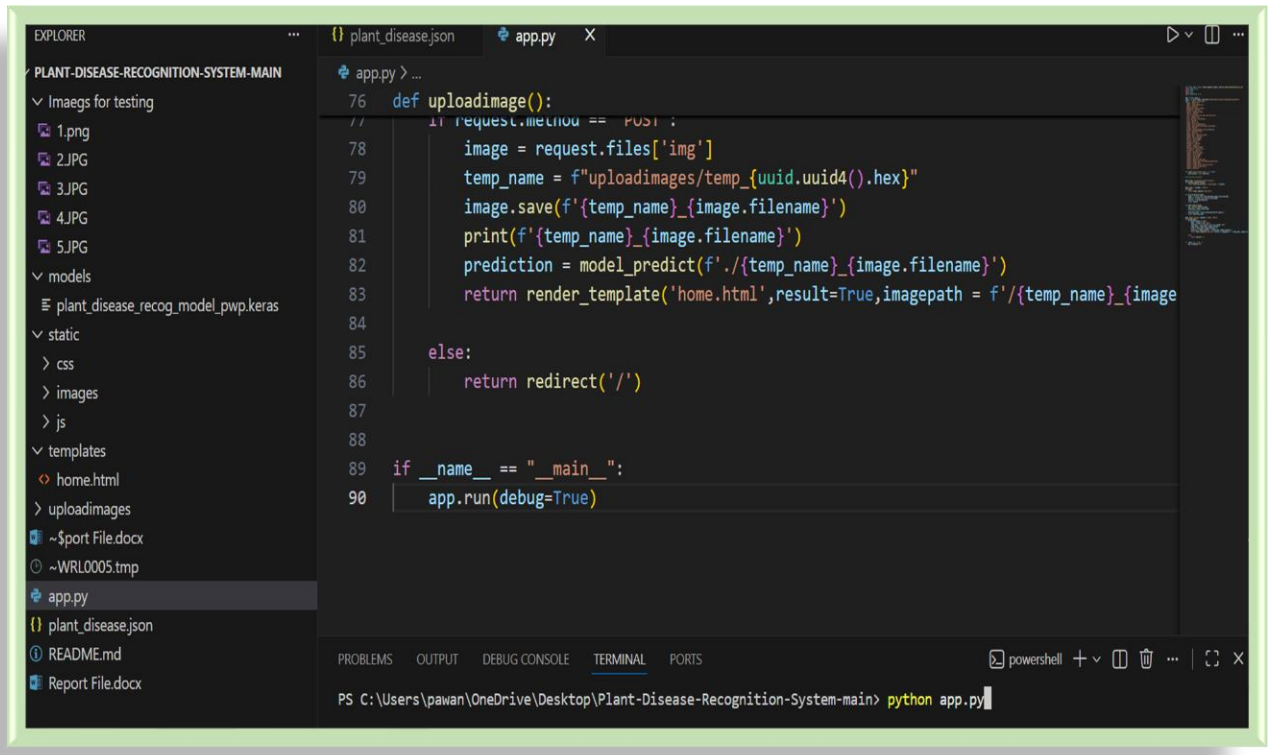
•Image preview before upload

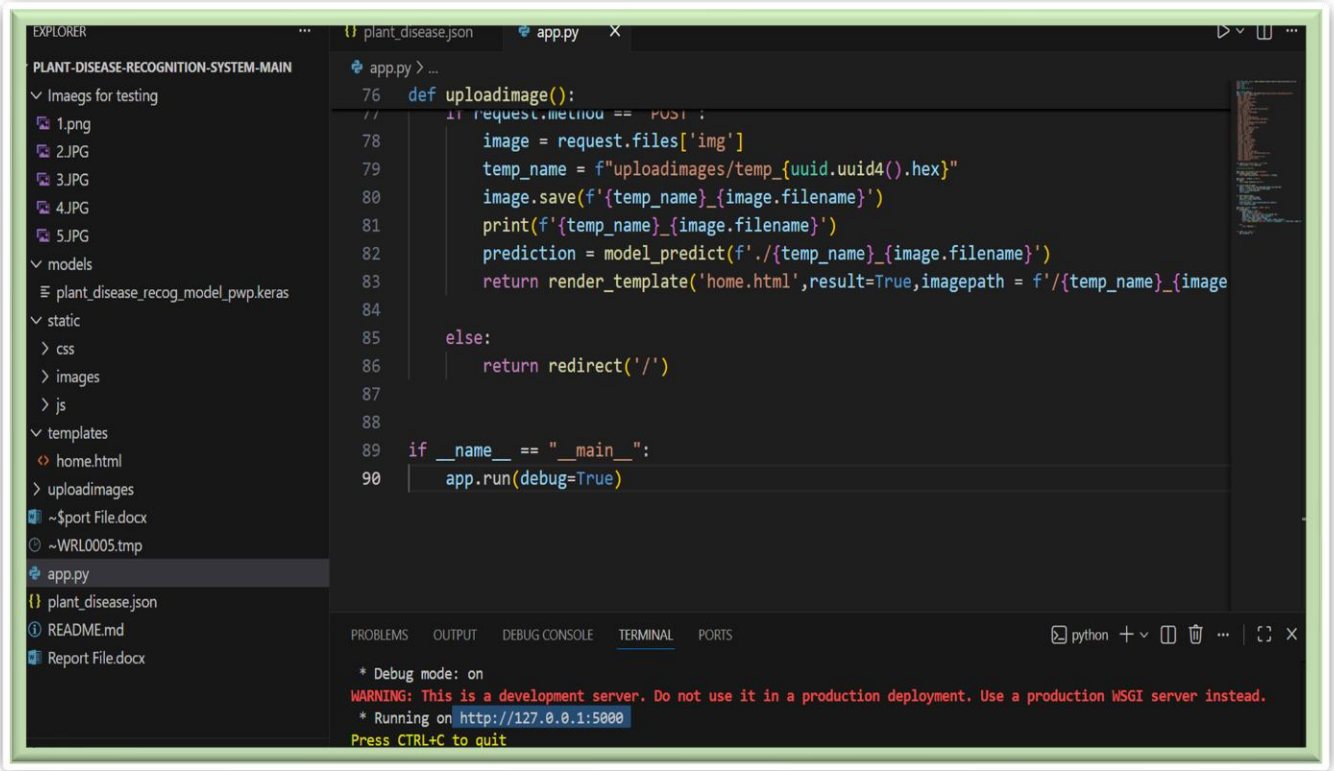
•AJAX request to Flask backend

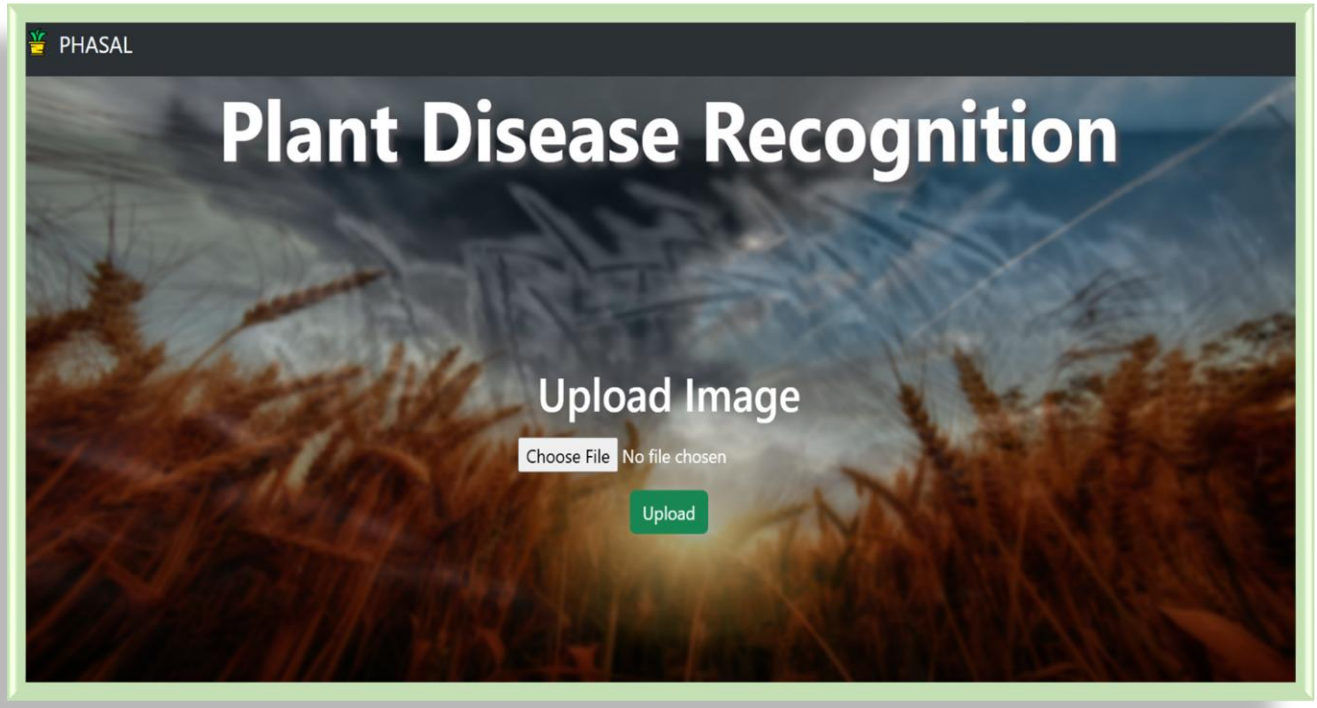
•Dynamic result rendering

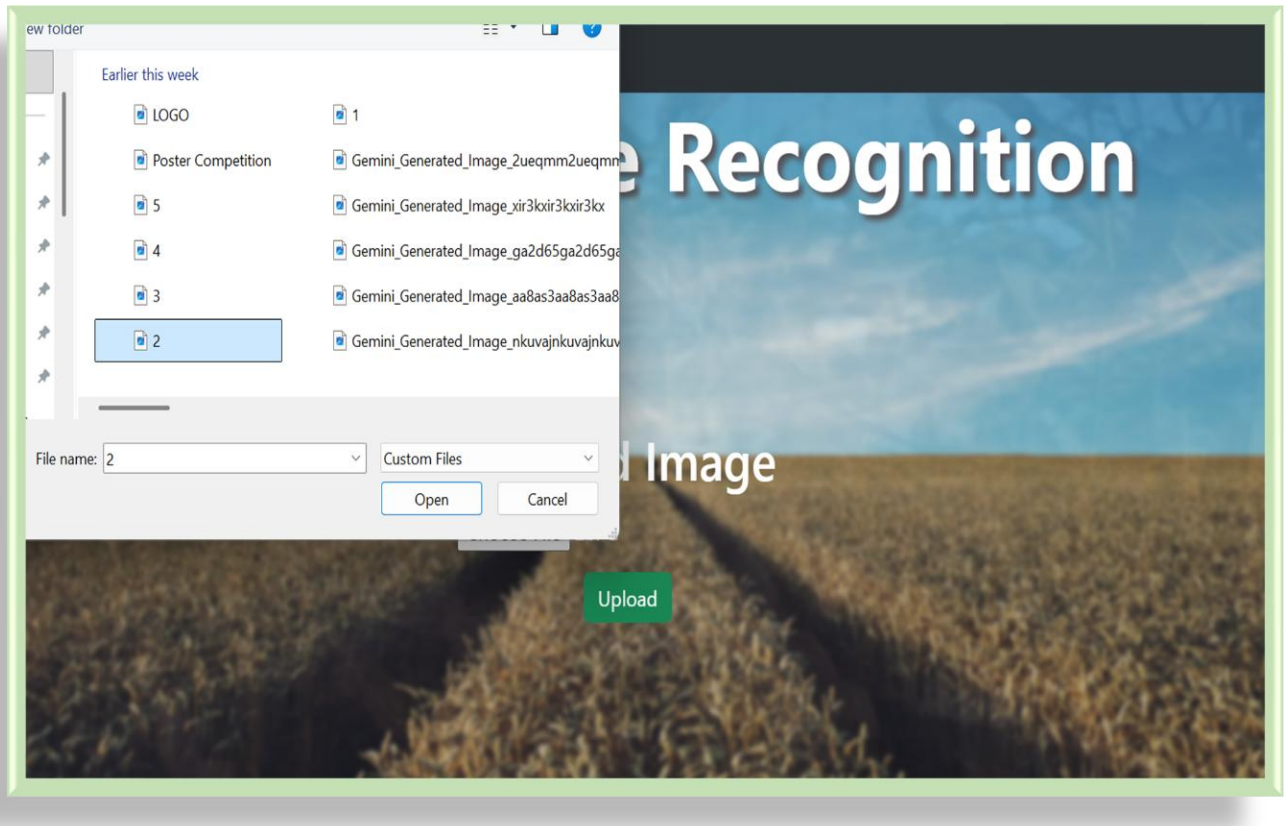
•Error message display

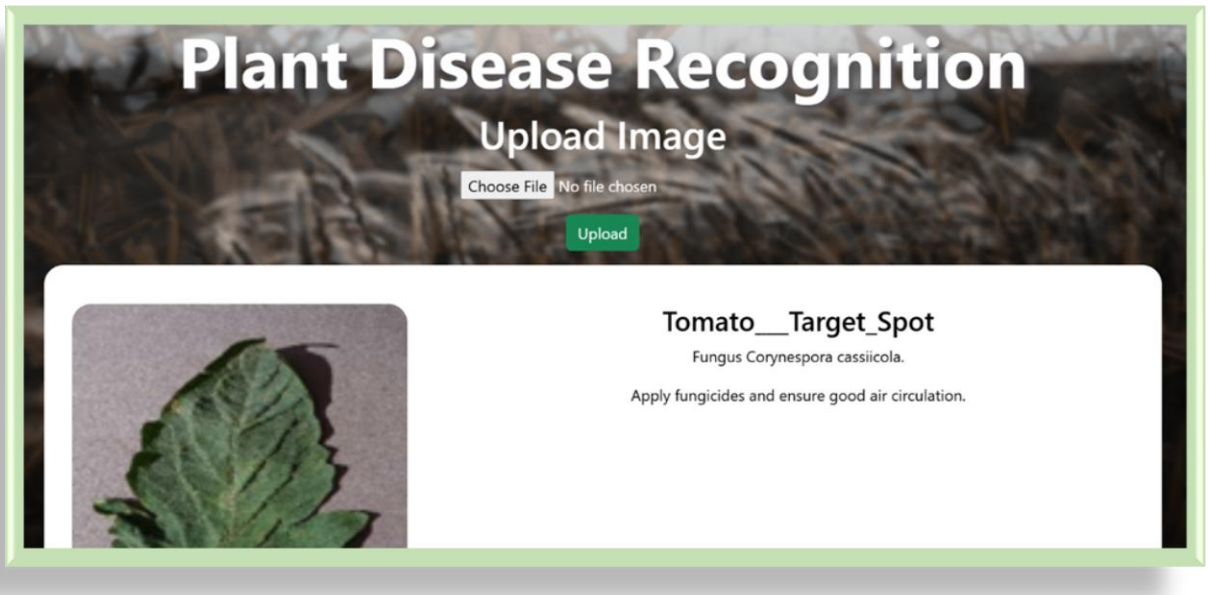
9. SCREENSHOTS AND RESLUTS:

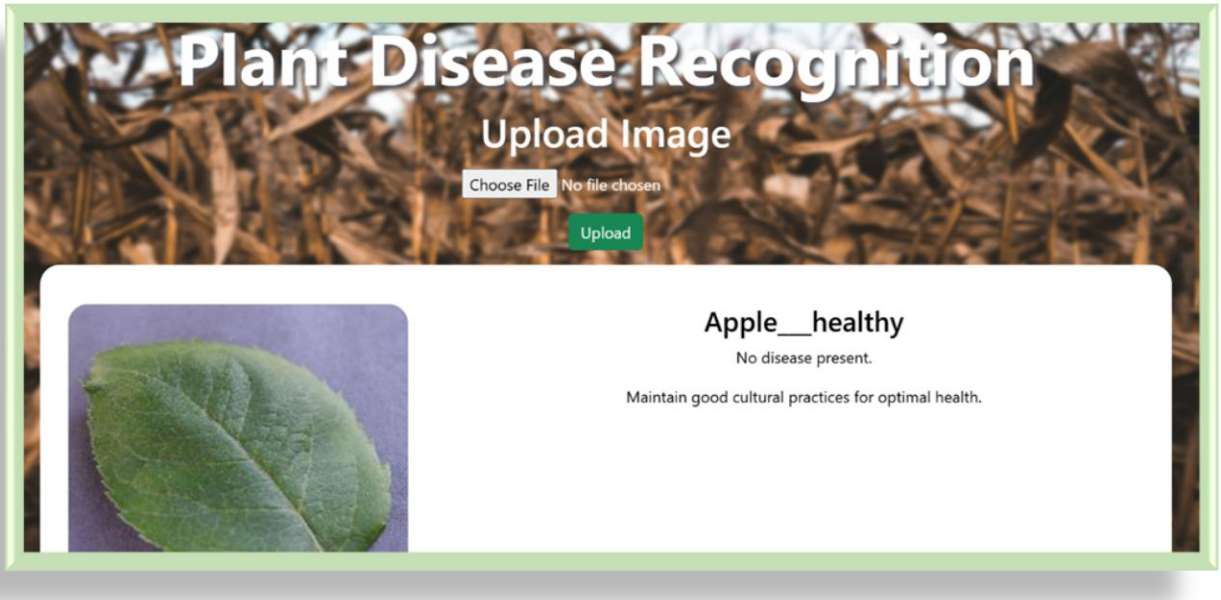


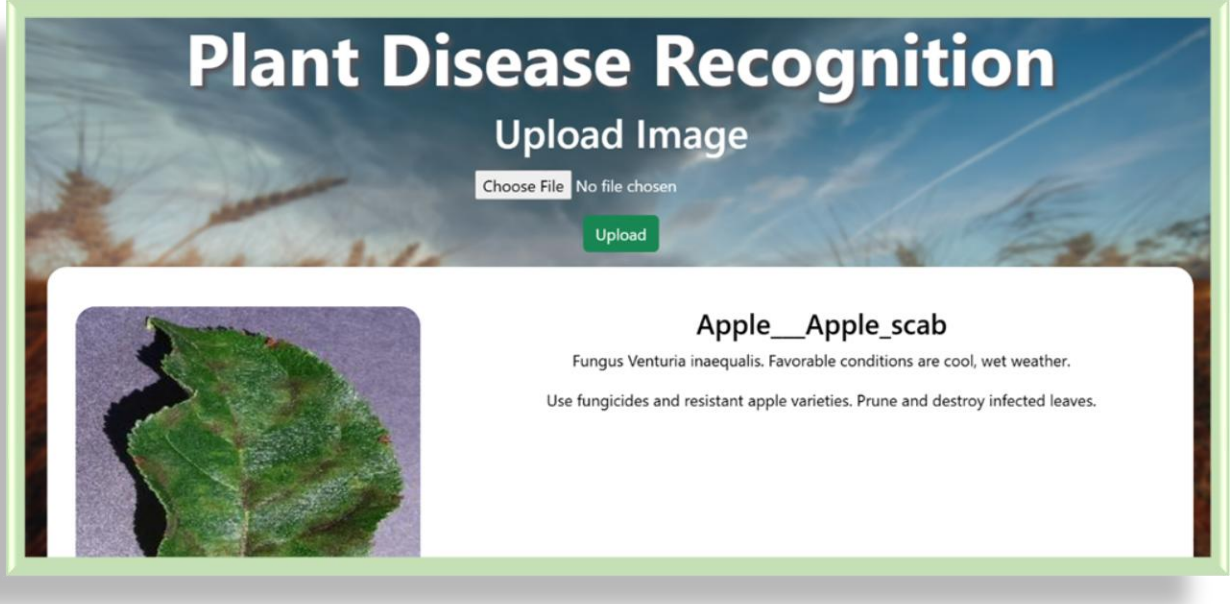



PHASAL
Plant Disease Recognition
Upload Image
Choose File
No file chosen
Upload


ew folder
Earlier this week
LOGO
1
Poster Competition
Gemini_Generated_Image_2ueqmm2ueqmn
5
Gemini_Generated_Image_xir3kxir3kxir3kx
4
Gemini_Generated_Image_ga2d65ga2d65ga
3
Gemini_Generated_Image_aa8as3aa8as3aa8
2
Gemini_Generated_Image_nkuvajnkuvajnkuv
File name: 2
Custom Files
Open
Cancel
Recognition
Image
Upload


Plant Disease Recognition
Upload Image
Choose File
No file chosen
Upload
Tomato___Target_Spot
Fungus Corynespora cassiicola.
Apply fungicides and ensure good air circulation.


Plant Disease Recognition
Upload Image
Choose File
No file chosen
Upload
Apple___healthy
No disease present.
Maintain good cultural practices for optimal health.


Plant Disease Recognition
Upload Image
Choose File
No file chosen
Upload
Apple___Apple_scab
Fungus Venturia inaequalis. Favorable conditions are cool, wet weather.
Use fungicides and resistant apple varieties. Prune and destroy infected leaves.

10. TESTING APPROACH:

(a) Unit Testing–

• Model Inference: Test model with known images to verify output shape and ranges

• Image Preprocessing: Validate image resizing and normalization

• Disease Lookup: Verify disease information retrieval

(b) Integration Testing–

• End-to-End Flow: User uploads an image → The server processes the image → The results are displayed to the user

• Error Scenarios: images that are too large

• Several users trying to upload at the same time

(c) Performance Testing–

- **Response Time:** Measure time under various conditions
- **Model Accuracy:** 96% accuracy
- **Memory Usage:** Monitor memory consumption

11. CHALLENGES FACED:

(a) Dataset Limitations–

Challenge: The Plant Village dataset features controlled images with single leaves on plain backgrounds, which are not representative of real-world field conditions.

Impact: The model can work very well with the images of the dataset but it may have difficulties with overlapping leaves, complex field background, or bad lighting.

Solution: The limitation was recognized; the suggestion for the next time: gather real-world field pictures for retraining.

(b) Model size and Deployment–

Challenge: EfficientNetV4 model weights (~203 MB) are large for some cases.

Solution: Model loads efficiently on most servers; can be used by quantization if needed.

(c) Image Upload Validation–

Challenge: Ensuring only valid image files are uploaded and processed.

Solution: By size limits, and error handling in Flask backend.

12. LEARNING AND KEY TAKEAWAYS:

(a) Transfer Learning Effectiveness–

Transfer learning can cut down the training time and the amount of data needed by a large factor while still being very accurate. Some of the features learned by the pre-trained ImageNet models are very general and can be easily transferred to the agricultural domain.

(b) Two -Phase Training Strategy–

Freezing the base model in Phase 1 prevents overfitting and unstable training. Fine-tuning unfrozen layers in Phase 2 achieves optimal performance while maintaining stability.

(c) Importance of Model Documentation–

Clear documentation of causes and cures enhances practical utility. A model is useful only if we understand and trust its predictions.

13. FUTURE ENHANCEMENTS:

(a) Real-World Field Dataset –

Initiative: Take pictures of plants in the natural environment of the field, reflecting changes in illumination, angles, leaves that are covering each other, and complex backgrounds.

Benefit: Greatly enhances the product's usability in the real world and the willingness of farmers to use it.

(b) Multi-Plant Recognition –

Initiative: recognize multiple crop types (tomato, apple, wheat, etc.)

Benefit: Makes us to reach different agricultural sectors .

(c) Farmer Feeback Loop –

Initiative**:** Collect user feedback on predictions to continuously improve model accuracy.

Benefit**:** Creates virtuous cycle of improvement through real-world validation.

14. REFERENCES:

[1] Plant Disease Recognition Model Using Deep Learning [Video]. https://www.youtube.com/watch?v=0BL42NXimF4

[2] TensorFlow Datasets. https://www.tensorflow.org/datasets/catalog/plant_village

[3] Pallab Pratim Ray. (2017). Deep Convolutional Neural Network for Plant Disease Detection. Journal of Computational Science, 21, 386-396.

[4] **VITYARTHI** ,VIT BHOPAL