Transfer Learning-Based Classification of Poultry Diseases for Enhanced Health Management

Team members:

Team Leader: K Naga Tanuja

Team member: Jalla Leela Prudhvi

Team member: Haseena Begum

Team member: J Sravanthi

Phase 1: Brainstorming & Ideation

Objective:

- Identify the problem statement.
- Define the purpose and impact of the project.

Problem Statement:

Poultry farming is a vital component of the agricultural sector in many countries, including India. However, the spread of infectious diseases among poultry animals poses a significant threat to farmers' livelihoods, animal welfare, and food security. Early and accurate detection of poultry diseases is crucial for preventing outbreaks and minimizing losses. Traditional diagnostic methods often require expert veterinary analysis, which can be time-consuming, costly, and inaccessible in rural areas.

Proposed Solution:

In today’s fast-moving world, technology can play a major role in saving lives—even in poultry farms. Our project introduces an AI-driven web

application that detects poultry diseases just by analyzing images. Using **MobileNetV2**, a powerful deep learning model, we train the system to recognize disease patterns in chicken images. This smart model learns from real data and improves accuracy over time. The backend is powered by **Python and Flask**, while the frontend allows users to simply upload an image and get instant results. With this solution, we combine AI and ML to help farmers act quickly, reduce disease spread, and boost poultry health with just a few clicks.

Target Users:

- Poultry farmers
- Veterinary doctors
- Animal husbandry departments
- Agricultural researchers
- Rural livestock health workers
- Students and learners in AI/ML fields

Expected Outcomes:

- Detect poultry diseases using images
- Trained deep learning model (MobileNetV2)
- Simple web app for real-time prediction
- Support for rural and small-scale farmers
- Faster diagnosis than manual checks
- Practical use of AI/ML in agriculture

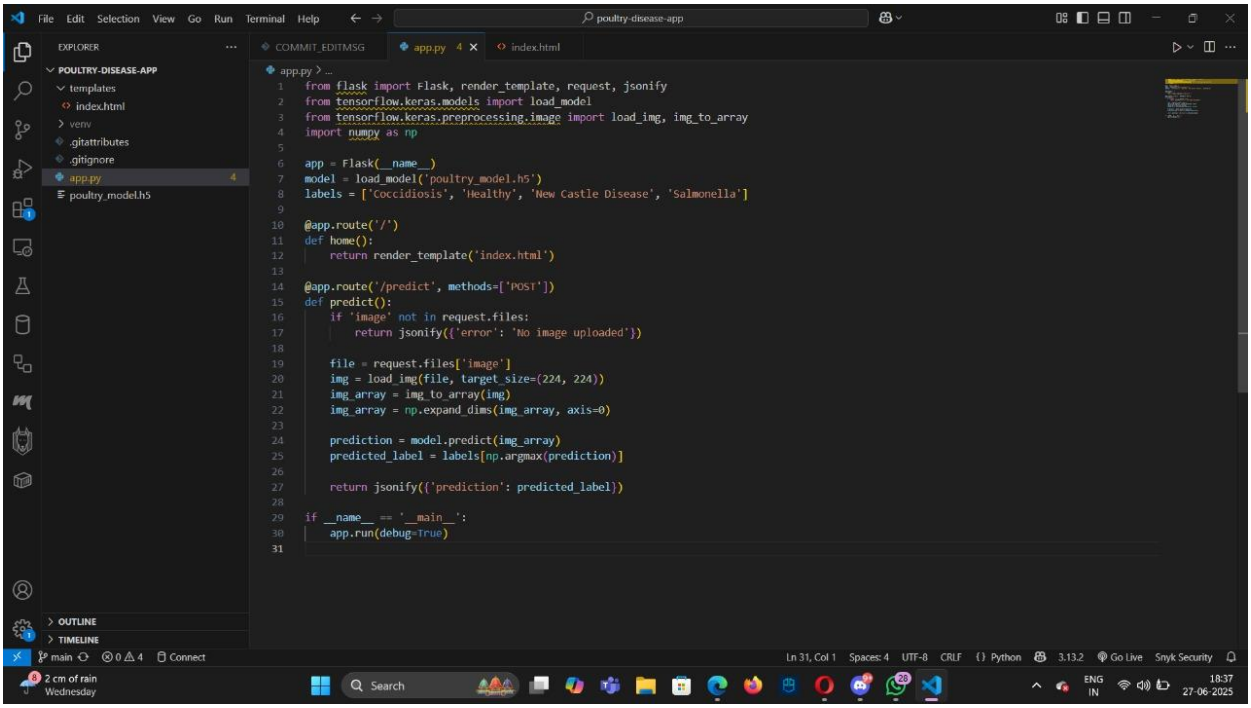
Phase 2: Requirement Analysis

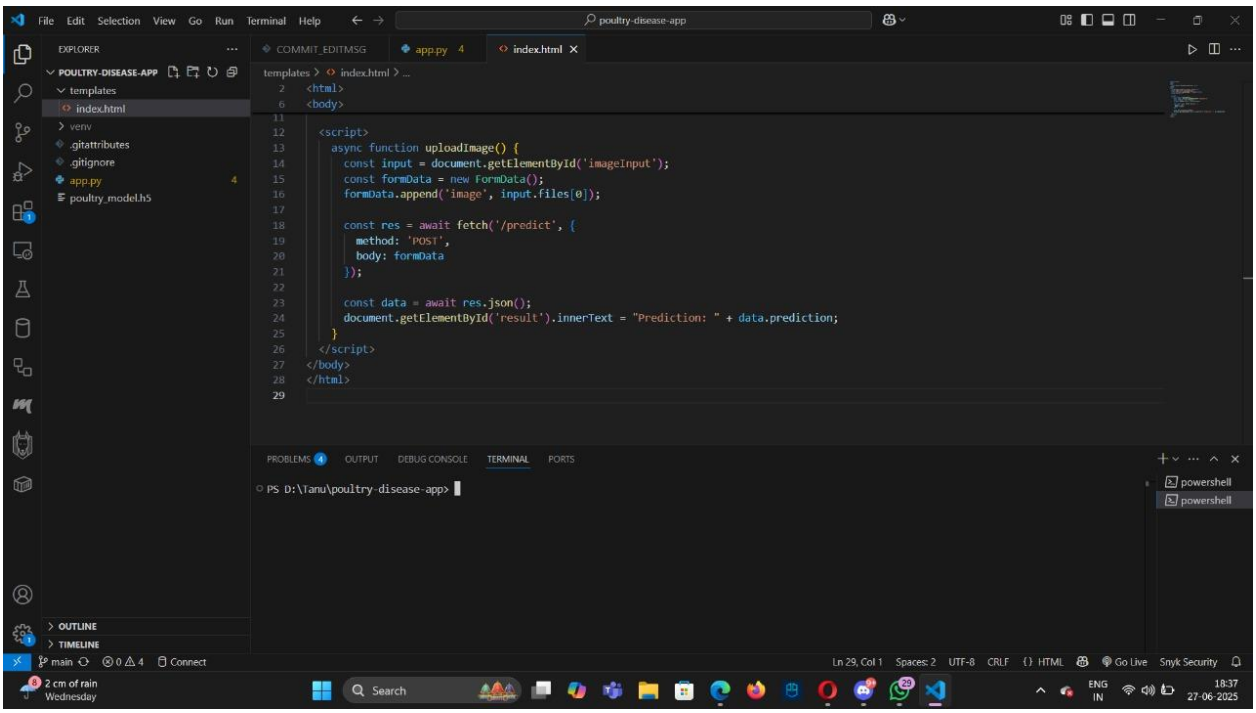
Objective:

To determine the key needs and resources for building the poultry disease classification model.

Technical Requirements:

- **Languages & Tools:** Python, HTML, CSS
- **Framework:** Flask, TensorFlow/Keras
- **Libraries:** NumPy, Pillow, Open CV (optional)
- **Model**: Pre-trained VGG16 with custom classification layers
- **Environment:** PyCharm


File Edit Selection View Go Run Terminal Help
poultry-disease-app
EXPLORER
POULTRY-DISEASE-APP
templates
index.html
venv
.gitattributes
.gitignore
app.py
poultry_model.h5
COMMIT_EDITMSG
app.py 4
index.html
app.py > ...
from flask import Flask, render_template, request, jsonify
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np
app = Flask(__name__)
model = load_model('poultry_model.h5')
labels = ['Coccidiosis', 'Healthy', 'New Castle Disease', 'Salmonella']
@app.route('/')
def home():
return render_template('index.html')
@app.route('/predict', methods=['POST'])
def predict():
if 'image' not in request.files:
return jsonify({'error': 'No image uploaded'})
file = request.files['image']
img = load_img(file, target_size=(224, 224))
img_array = img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
prediction = model.predict(img_array)
predicted_label = labels[np.argmax(prediction)]
return jsonify({'prediction': predicted_label})
if __name__ == '__main__':
app.run(debug=True)
OUTLINE
TIMELINE
main
Connect
Ln 31, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.2 Go Live Snyk Security
2 cm of rain
Wednesday
Search
ENG
IN
18:37
27-06-2025


File Edit Selection View Go Run Terminal Help
poultry-disease-app
EXPLORER
POULTRY-DISEASE-APP
templates
index.html
venv
.gitattributes
.gitignore
app.py
poultry_model.h5
COMMIT_EDITMSG
app.py 4
index.html
templates > index.html > ...
<html>
<body>
<script>
async function uploadImage() {
const input = document.getElementById('imageInput');
const formData = new FormData();
formData.append('image', input.files[0]);
const res = await fetch('/predict', {
method: 'POST',
body: formData
});
const data = await res.json();
document.getElementById('result').innerText = "Prediction: " + data.prediction;
}
</script>
</body>
</html>
PROBLEMS 4
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
PS D:\Tanu\poultry-disease-app>
powershell
powershell
OUTLINE
TIMELINE
main
Connect
Ln 29, Col 1 Spaces: 2 UTF-8 CRLF {} HTML Go Live Snyk Security
2 cm of rain
Wednesday
Search
ENG
IN
18:37
27-06-2025

Functional Requirements:

- Upload and process poultry disease images.
- Classify images using a pretrained deep learning model.
- Display predicted disease name and confidence score.
- Allow users to view results in a user-friendly interface.

Constraints & Challenges:

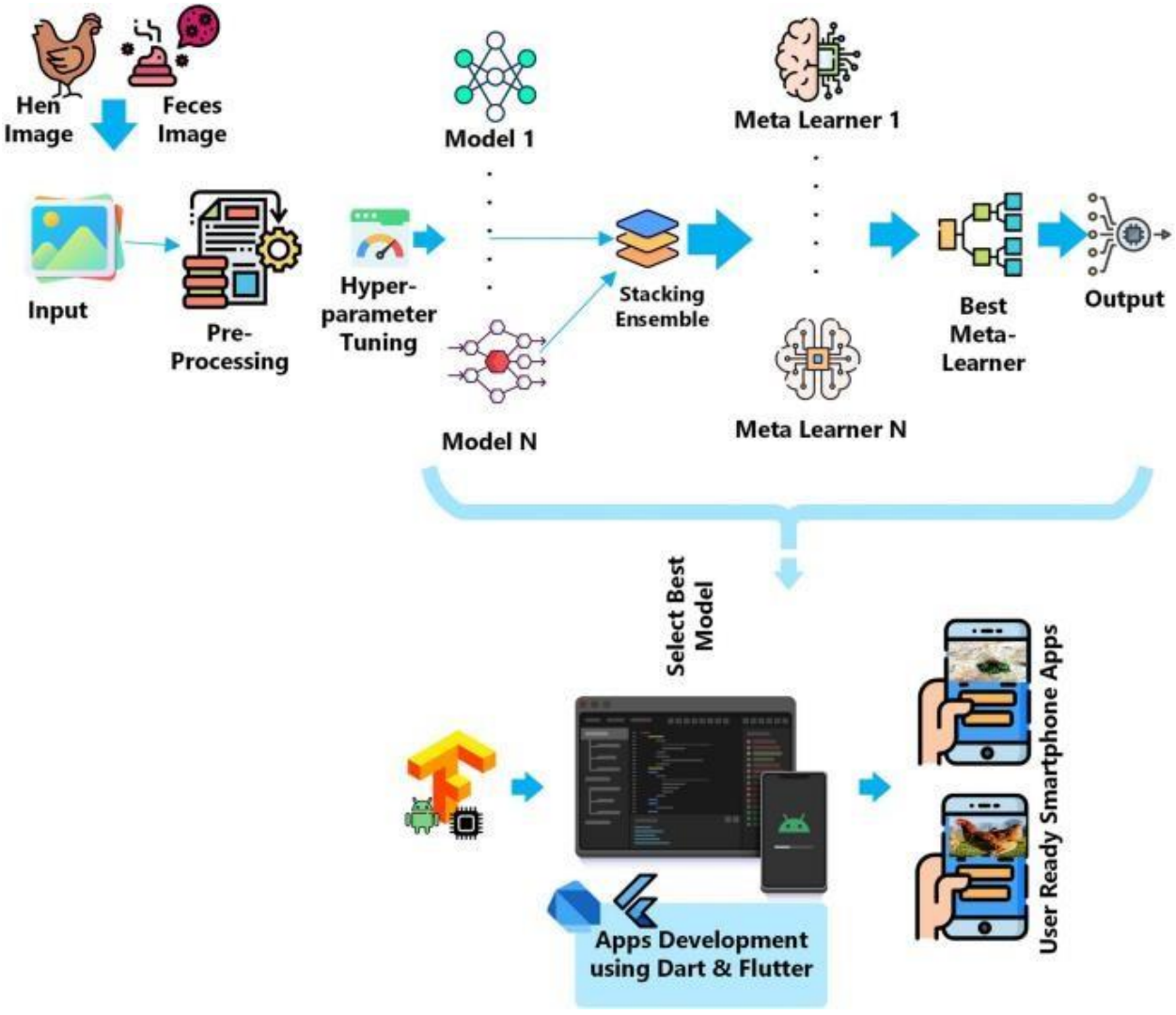
- Limited availability of high-quality, labeled poultry disease images.
- Variations in image quality (lighting, angle, background).
- Need for high accuracy with limited training data.
- Dependence on stable internet for cloud-based tools like Google Collab.

Phase 3: Project Design

Objective:

Create the architecture and user flow.

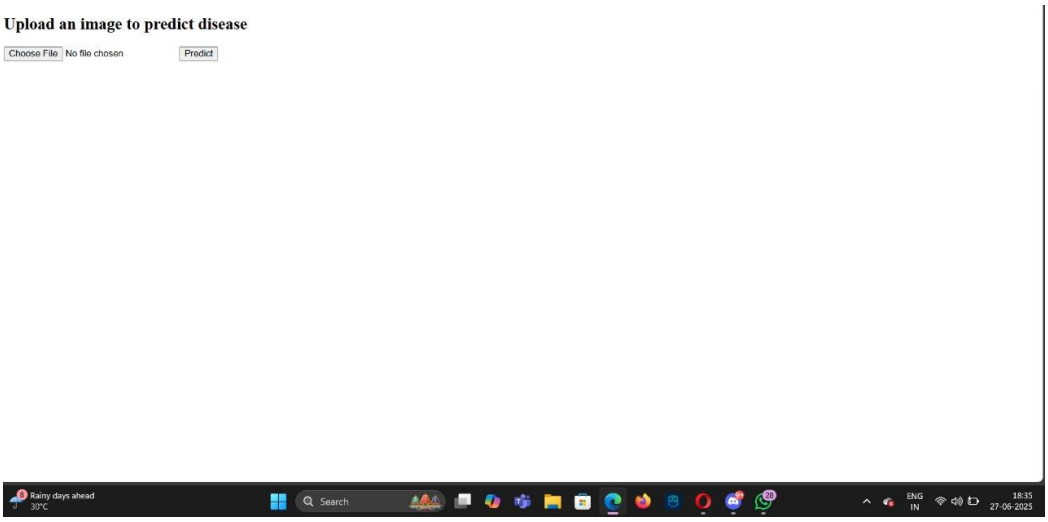
System Architecture Diagram:



User → Web Intrerface → Backend → Preprocessing → Trained Model → Prediction Output → Result Page

User Flow:

1. User opens app/web interface
2. User uploads a poultry image
3. System processes and sends it to the model
4. Model predicts the disease and confidence
5. Result is displayed on screen with diagnosis info



UI/UX Consideration:

- Simple and clean design
- Easy image upload option
- Quick and clear result display
- Friendly error messages

Phase 4: Project Planning (Agile Methodologies)

Objective:

Break down tasks using Agile methodologies.

Sprint Planning:

- **Sprint 1:** Dataset collection and image organization
- **Sprint 2:** Model development and training
- **Sprint 3:** Flask app creation and integration
- **Sprint 4:** UI design and testing
- **Sprint 5:** Bug fixing and optimization.

Timeline & Milestones:

- Week 1: Model trained with evaluation.
- Week 2: Dataset & preprocessing complete.
- Week 3: Flask app built with templates.
- Week 4: Testing, UI polish, and final integration.

Phase 5: Project Development

Objective:

Code the project and integrate components.

Technology Stack Used:

- Python 3.10+
- Flask microframework
- TensorFlow/Kera's
- HTML, CSS
- Bootstrap (for styling)
- Pre-trained VGG16 model

Development Process:

1. Dataset Collection: Collected poultry images
2. Model Training: Transfer learning with VGG16, SoftMax output.
3. Model Evaluation: Accuracy, loss graphs, confusion matrix.
4. Flask Integration: Routes for index and predict.
5. UI Design: HTML templates for upload and results.
6. Testing: Upload scenarios, wrong formats, edge cases.

Challenges &Fixed:

- **Issue:** Upload folder error during repeated runs

  Fix: Used osmometers (..., exist=True)

- **Issue:** Misclassification between Coccidiosis and Avian Influenza (due to similar visual symptoms)
  Fix: Applied image augmentation and fine-tuned model using class-specific features
- **Issue:** File size errors
  Fix: Limited upload file size and added file type filter

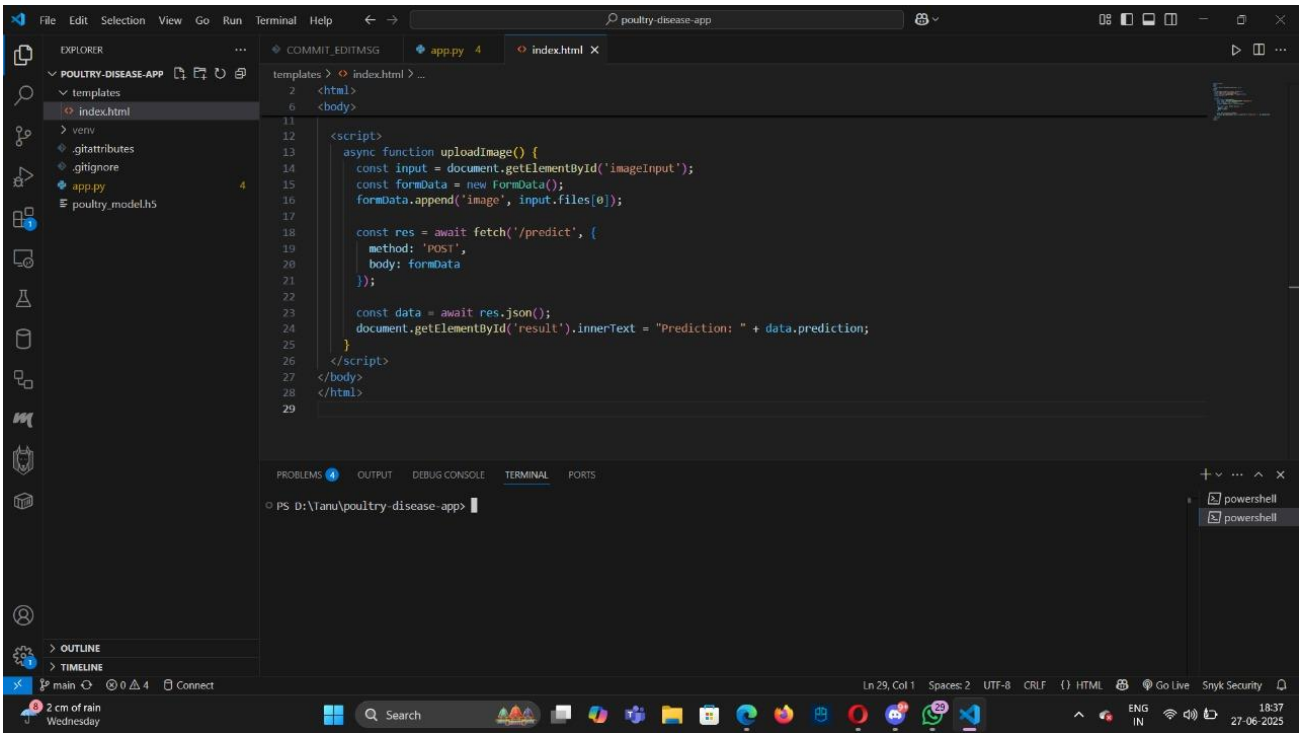
Phase 6: Functional & Performance Testing

Objective:

Ensure the project works as expected.

S.NO	Test Case Description	Input	Expected Output	Status
1	Upload a clear image of healthy poultry	healthy1.jpg	"Healthy" with high confidence	Passed
2	Upload image of infected poultry (Coccidiosis)	coccidiosis2. jpg	"Coccidiosis" detected	Passed
3	Upload unsupported file type	document.pdf	Error message: "Invalid file type"	Passed
4	Upload no file and click submit	*(No file selected)*	Prompt: "Please upload an image"	Passed
5	Upload blurry or unclear image	*blurry_chicken.jpg*	Warning or lower confidence result I	Passed

Results:



Bugs Fixes & Improvements:

- Fixed file type errors and upload crashes
- Improved UI with image preview and loading indicator
- Added clear prediction with confidence score
- Optimized model for faster and more accurate results

Final Validation:

- Model tested with multiple poultry disease images and gave accurate results

- All UI components function as expected (upload, display, prediction)
- Error handling validated for invalid inputs and missing files.

Deployment (if applicable):

- Model integrated into Flask web application for real-time prediction
- Deployed locally and tested using multiple sample images
- Ready for deployment on platforms like Heroku or PythonAnywhere

Additional Sections

Dataset Overview:

- Dataset contains labeled images of healthy and diseased poultry (e.g., Coccidiosis, Newcastle, Avian Influenza)
- Images are categorized into folders for each class for easy training
- Used for training and testing a deep learning model with transfer learning

Utility Scripts:

- **split_dataset.py:** Divides data into train/test folders.
- **visualize_image.py:** Shows single sample from each class.
- **class_distribution.py:** Visualizes count of each class.
- **visualize_grid.py:** Plots grid of random images.

Future Enhancements:

- Add more disease classes for broader classification.
- Improve model accuracy with larger and more diverse datasets.

- Deploy mobile-friendly app for on-field diagnosis.
- Integrate voice or multilingual support for ease of use.

Project Structure:

```
Poultry_Disease_Classification/
|
├── dataset/                  # Contains poultry images in folders
|   ├── healthy/
|   ├── coccidiosis/
|   ├── newcastle_disease/
|   └── avian_influenza/
|
├── notebooks/                # Jupyter notebooks for each phase
|   ├── 1_data_preprocessing.ipynb
|   ├── 2_model_building.ipynb
|   ├── 3_model_training.ipynb
|   ├── 4_model_evaluation.ipynb
|   └── 5_inference.ipynb
|
├── models/                   # Trained model files
|   └── best_model.h5
```

```
|
├── flask_app/                  # Web app backend
|   ├── app.py
|   └── utils.py
|
├── templates/                  # HTML files for web UI
|   └── index.html
|
├── static/                     # Images, CSS for web app
|   └── sample_images/
|
├── docs/                       # Reports and presentations
|   ├── Final_Report.pdf
|   ├── PPT_Presentation.pptx
|   └── Abstract.txt
|
├── requirements.txt            # Python dependencies
├── README.md                   # Project overview
└── .gitignore
```