Poultry Disease Classification Project - Complete Requirements Document

---

1. Project Overview

This project aims to build a deep learning-based web application for classifying poultry diseases using image data. The system uses a VGG16 pre-trained model integrated into a Flask-based web interface. The user uploads an image of a chicken, and the application predicts the disease class.

---

2. Required Tools & Technologies

Programming Languages:

- Python 3.9+

Libraries & Frameworks:

- TensorFlow (Keras)
- NumPy
- Pandas
- Matplotlib (for visualization if needed)
- Flask (for backend web application)
- Jupyter Notebook (for model development)
- Pillow (for image handling)
- OpenCV (optional for preprocessing)

Web Technologies:

- HTML (for the frontend)
- CSS (optional for styling)

Environment:

- Google Colab / Jupyter Notebook (Model training)
- Localhost / Render / Replit (Deployment)

Data Source:

- Kaggle Dataset: https://www.kaggle.com/datasets/chandrashekarnatesh/poultry-diseases
- License: MIT

---

3. Folder Structure (GitHub Repository)

```
poultry-disease-classification/
│
├── app.py                          # Flask server script
├── templates/
│   └── index.html                 # UI template for web interface
├── static/uploads/               # Folder to store uploaded images
├── model/
│   ├── poultry_model.ipynb       # Jupyter Notebook containing trained model
│   └── predict_model.py          # Prediction logic from saved model/notebook
├── documents/                    # Design templates, reports
│   ├── Empathy_Map.pdf
│   ├── Project_Backlog.xlsx
│   └── Other uploaded templates...
├── data/                         # Sample images for testing
├── requirements.txt              # List of required libraries
├── README.md                     # Project documentation
└── .gitignore
```

---

4. System Requirements

Functional Requirements:

- User uploads an image through the UI.
- Server receives the image and sends it to the model.
- Model predicts the class (disease).
- Prediction result is shown on the UI.

Non-functional Requirements:

- Usability: Simple UI with image upload feature
- Performance: Should return prediction within 3–5 seconds
- Scalability: Can be deployed to cloud
- Availability: Hosted through Replit/Render for public access

---

5. App.py - Flask Server Script Requirements

- Upload image using POST method
- Store file in `static/uploads/`
- Call the `predict_image()` method from `predict_model.py`
- Render prediction result to HTML

---

6. Model Requirements (Inside .ipynb)

- Use of VGG16 as base model with frozen layers
- Add Flatten, Dense (256), Dropout, and Dense(3 or 4 classes) with softmax
- Data augmentation using `ImageDataGenerator`
- Use categorical crossentropy loss and Adam optimizer
- Achieve training & validation accuracy above 85%

---

7. Prediction Script (predict_model.py)

Stub (for now):

```
# Sample structure

def predict_image(image_path):
    # Custom logic OR manual prediction OR mocked result
    return "Coccidiosis"
```

Later (real logic):

- Load model
- Preprocess image (resize to 224x224, normalize)
- Predict and return class label

---

8. HTML Template Requirements (index.html)

- Use `<form>` to upload image
- Show preview or result
- Render prediction dynamically

---

9. Installation Requirements (requirements.txt)

```
Flask==2.2.3
tensorflow==2.13.0
numpy
pillow
```

---

10. Deployment Options

- Localhost: `python app.py`

- Render/Replit: Create account, link GitHub, deploy Flask app

---

11. Additional Files

- `README.md` with sections: Introduction, How to Run, Results, Credits
- `documents/`: Planning phase files like Project Plan, Empathy Map, DFD, Sprint Chart

---

12. Future Improvements

- Save model as `.h5` and load directly
- Replace stub `predict_model.py` with actual model logic
- Add image preview before uploading
- Deploy to public hosting
- Add classification confidence score

✅Conclusion

This document outlines the full requirements and flow to successfully build, deploy, and present your Poultry Disease Classification project using Keras, Kaggle, and Flask. All that remains is linking the files and pushing to GitHub!