# Back end Code:

## app.py

from flask import Flask, render\_template, request

from keras.models import load\_model

from PIL import Image

import numpy as np

import os

app = Flask(\_\_name\_\_)

model = load\_model("poultry\_model.h5")  # replace with your actual .h5 file name

class\_labels = ['Coccidiosis', 'Healthy', 'NewCastle', 'Salmonella']

# ⬇️ Fix: Define preprocessing function

def preprocess\_image(img\_path):

    img = Image.open(img\_path).convert('RGB')

    img = img.resize((224, 224))

    img = np.array(img) / 255.0

    img = np.expand\_dims(img, axis=0)

    return img

@app.route('/')

def index():

    return render\_template('index.html')

@app.route('/upload', methods=['GET', 'POST'])

def upload():

    prediction = None

    if request.method == 'POST':

        file = request.files['file']

        if file and file.filename.lower().endswith(('.jpg', '.jpeg', '.png')):

            save\_path = os.path.join('static', 'uploads', file.filename)

            file.save(save\_path)

            img = preprocess\_image(save\_path)

            preds = model.predict(img)[0]

            predicted\_label = class\_labels[np.argmax(preds)]

            prediction = f"The infection type detected as {predicted\_label}"

        else:

            prediction = "Unsupported file format. Please upload a JPG, JPEG, or PNG."

    return render\_template('prediction.html', prediction=prediction)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(debug=True)

# Trained Model :

## Model.h5:

import os

import cv2

import numpy as np

from sklearn.preprocessing import LabelEncoder

from sklearn.model\_selection import train\_test\_split

from keras.utils import to\_categorical

from keras.applications.vgg16 import VGG16

from keras.models import Model

from keras.layers import Flatten, Dense, Dropout

from keras.optimizers import Adam

# ----- Parameters -----

IMG\_WIDTH, IMG\_HEIGHT = 224, 224

CATEGORIES = ['Salmonella', 'New Castle Disease', 'Coccidiosis', 'Healthy']

BASE\_PATH = './data/data/train'

MAX\_IMAGES = 500

# ----- Read Image Data -----

def read\_data(base\_path, categories, max\_images=500):

data = []

labels = []

for label in categories:

path = os.path.join(base\_path, label)

count = 0

for img\_file in os.listdir(path):

if count >= max\_images:

break

img\_path = os.path.join(path, img\_file)

try:

img = cv2.imread(img\_path)

img = cv2.resize(img, (IMG\_WIDTH, IMG\_HEIGHT))

data.append(img)

labels.append(label)

count += 1

except:

continue

return np.array(data), np.array(labels)

print("Loading data...")

X, y = read\_data(BASE\_PATH, CATEGORIES, MAX\_IMAGES)

X = X / 255.0

# ----- Encode labels -----

le = LabelEncoder()

y\_encoded = to\_categorical(le.fit\_transform(y))

# ----- Train-test split -----

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X, y\_encoded, test\_size=0.2, random\_state=42)

# ----- Build Model -----

print("Building model...")

base\_model = VGG16(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3))

for layer in base\_model.layers:

layer.trainable = False

x = base\_model.output

x = Flatten()(x)

x = Dense(128, activation='relu')(x)

x = Dropout(0.3)(x)

output = Dense(4, activation='softmax')(x)

model = Model(inputs=base\_model.input, outputs=output)

model.compile(optimizer=Adam(learning\_rate=1e-4), loss='categorical\_crossentropy', metrics=['accuracy'])

# ----- Train Model -----

print("Training model...")

model.fit(X\_train, y\_train, validation\_data=(X\_val, y\_val), epochs=10, batch\_size=32)

# ----- Save Model -----

print("Saving model...")

model.save("poultry\_model.h5")

print("Model saved as poultry\_model.h5")