```
from flask import Flask, render_template, request
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
import os

app = Flask(__name__)
model = load_model("model/poultry_model.h5")

# Match the class labels used during training
class_labels = ['Coccidiosis', 'New Castle', 'Healthy', 'salmonella']  #
Replace with your real class folder names

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    if 'file' not in request.files:
        return "No file part"

    file = request.files['file']
    if file.filename == '':
        return "No selected file"

    if file:
        file_path = os.path.join('static/uploaded', file.filename)
        file.save(file_path)

        # Preprocess the image
        img = image.load_img(file_path, target_size=(224, 224))
        img_array = image.img_to_array(img) / 255.0
        img_array = np.expand_dims(img_array, axis=0)

        # Predict
        prediction = model.predict(img_array)
        predicted_class = class_labels[np.argmax(prediction)]

        return render_template('result.html', prediction=predicted_class,
image_path=file_path)

if __name__ == '__main__':
    app.run(debug=True)
```

```
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 6,
   "id": "1d3a70de-e8eb-4ae6-affa-a8dcba4b3f51",
   "metadata": {},
   "outputs": [],
   "source": [
    "import os\n",
    "import numpy as np\n",
    "import tensorflow as tf\n",
    "from tensorflow.keras.preprocessing.image import
ImageDataGenerator\n",
    "from tensorflow.keras.applications import MobileNetV2\n",
    "from tensorflow.keras.models import Model, load_model\n",
    "from tensorflow.keras.layers import Dense, GlobalAveragePooling2D\n",
    "from tensorflow.keras.optimizers import Adam\n",
    "from sklearn.metrics import classification_report,
confusion_matrix\n",
    "import cv2\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 8,
   "id": "6f9d0f15-998f-4c2f-a1bd-23a43a8685a9",
   "metadata": {},
   "outputs": [],
   "source": [
    "base_dir = 'dataset'  # Keep your dataset here\n",
    "img_height, img_width = 224, 224\n",
    "batch_size = 32\n",
    "epochs = 10\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 10,
   "id": "0849340d-f246-4463-9927-68c678510099",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Found 5451 images belonging to 4 classes.\n",
      "Found 1361 images belonging to 4 classes.\n"
     ]
    }
   ],
   "source": [
    "train_datagen = ImageDataGenerator(\n",
    "    rescale=1./255,\n",
    "    validation_split=0.2,         # 80% train, 20% validation\n",
    "    horizontal_flip=True,\n",
    "    zoom_range=0.2,\n",
    "    shear_range=0.2\n",
    ")\n",
    "\n",
    "train_generator = train_datagen.flow_from_directory(\n",
    "    base_dir,\n",
    "    target_size=(img_height, img_width),\n",
    "    batch_size=batch_size,\n",
    "    subset='training',\n",
    "    class_mode='categorical'\n",
    ")\n",
    "\n",
    "val_generator = train_datagen.flow_from_directory(\n",
    "    base_dir,\n",
    "    target_size=(img_height, img_width),\n",
    "    batch_size=batch_size,\n",
    "    subset='validation',\n",
    "    class_mode='categorical'\n",
    ")\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 12,
   "id": "11137dd1-71ae-4ead-89b1-625c6041f9ad",
   "metadata": {},
   "outputs": [],
   "source": [
    "base_model = MobileNetV2(input_shape=(img_height, img_width, 3), 
include_top=False, weights='imagenet')\n",
    "base_model.trainable = False  # Freeze pre-trained layers\n",
    "\n",
    "x = base_model.output\n",
    "x = GlobalAveragePooling2D()(x)\n",
    "x = Dense(128, activation='relu')(x)\n",
    "predictions = Dense(train_generator.num_classes, 
activation='softmax')(x)\n",
    "\n",
    "model = Model(inputs=base_model.input, outputs=predictions)\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 14,
   "id": "c811196d-0249-4334-938d-364607e60c08",
   "metadata": {},
   "outputs": [],
   "source": [
    "model.compile(optimizer=Adam(learning_rate=0.0001), \n",
    "              loss='categorical_crossentropy', \n",
    "              metrics=['accuracy'])\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 16,
   "id": "6990d98f-26fd-447f-989c-4446d31eebb0",
   "metadata": {},
   "outputs": [
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "C:\\Users\\91939\\anaconda3\\Lib\\site-
packages\\keras\\src\\trainers\\data_adapters\\py_dataset_adapter.py:121:
UserWarning: Your `PyDataset` class should call
`super().__init__(**kwargs)` in its constructor. `**kwargs` can include
`workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these
arguments to `fit()`, as they will be ignored.\n",
      "  self._warn_if_super_not_called()\n"
     ]
    },
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Epoch 1/10\n",
      "\u001b[1m171/171\u001b[0m \u001b[32m━━━━━━━━━━━━━━━━━
━━━\u001b[0m\u001b[37m\u001b[0m \u001b[1m1743s\u001b[0m 10s/step -
accuracy: 0.6491 - loss: 0.9036 - val_accuracy: 0.8861 - val_loss:
0.3376\n",
      "Epoch 2/10\n",
      "\u001b[1m171/171\u001b[0m \u001b[32m━━━━━━━━━━━━━━━━━
━━━\u001b[0m\u001b[37m\u001b[0m \u001b[1m1250s\u001b[0m 7s/step -
accuracy: 0.8932 - loss: 0.3079 - val_accuracy: 0.9052 - val_loss:
0.2685\n",
      "Epoch 3/10\n",
      "\u001b[1m171/171\u001b[0m \u001b[32m━━━━━━━━━━━━━━━━━
━━━\u001b[0m\u001b[37m\u001b[0m \u001b[1m1225s\u001b[0m 7s/step -
accuracy: 0.9227 - loss: 0.2287 - val_accuracy: 0.9236 - val_loss:
0.2256\n",
      "Epoch 4/10\n",
      "\u001b[1m171/171\u001b[0m \u001b[32m━━━━━━━━━━━━━━━━━
━━━\u001b[0m\u001b[37m\u001b[0m \u001b[1m1152s\u001b[0m 7s/step -
accuracy: 0.9237 - loss: 0.2191 - val_accuracy: 0.9236 - val_loss:
0.2194\n",
      "Epoch 5/10\n",
      "\u001b[1m171/171\u001b[0m \u001b[32m━━━━━━━━━━━━━━━━━
━━━\u001b[0m\u001b[37m\u001b[0m \u001b[1m1154s\u001b[0m 7s/step -
accuracy: 0.9375 - loss: 0.1783 - val_accuracy: 0.9339 - val_loss:
0.1999\n",
      "Epoch 6/10\n",
      "\u001b[1m171/171\u001b[0m \u001b[32m━━━━━━━━━━━━━━━━━
━━━\u001b[0m\u001b[37m\u001b[0m \u001b[1m1219s\u001b[0m 7s/step -
accuracy: 0.9472 - loss: 0.1585 - val_accuracy: 0.9309 - val_loss:
0.2054\n",
      "Epoch 7/10\n",
      "\u001b[1m171/171\u001b[0m \u001b[32m━━━━━━━━━━━━━━━━━
━━━\u001b[0m\u001b[37m\u001b[0m \u001b[1m1159s\u001b[0m 7s/step -
```

```
accuracy: 0.9434 - loss: 0.1664 - val_accuracy: 0.9317 - val_loss:
0.2041\n",
      "Epoch 8/10\n",
      "\u001b[1m171/171\u001b[0m \u001b[32m━━━━━━━━━━━━━━━━━━
━━\u001b[0m\u001b[37m\u001b[0m \u001b[1m1207s\u001b[0m 7s/step -
accuracy: 0.9447 - loss: 0.1471 - val_accuracy: 0.9412 - val_loss:
0.1683\n",
      "Epoch 9/10\n",
      "\u001b[1m171/171\u001b[0m \u001b[32m━━━━━━━━━━━━━━━━━━
━━\u001b[0m\u001b[37m\u001b[0m \u001b[1m1243s\u001b[0m 7s/step -
accuracy: 0.9522 - loss: 0.1379 - val_accuracy: 0.9390 - val_loss:
0.1741\n",
      "Epoch 10/10\n",
      "\u001b[1m171/171\u001b[0m \u001b[32m━━━━━━━━━━━━━━━━━━
━━\u001b[0m\u001b[37m\u001b[0m \u001b[1m1134s\u001b[0m 7s/step -
accuracy: 0.9558 - loss: 0.1311 - val_accuracy: 0.9493 - val_loss:
0.1546\n"
     ]
    }
   ],
   "source": [
    "history = model.fit(\n",
    "    train_generator,\n",
    "    validation_data=val_generator,\n",
    "    epochs=epochs\n",
    ")\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 18,
   "id": "03d79986-b07c-4509-aca6-3f2a03bb031b",
   "metadata": {},
   "outputs": [
    {
     "name": "stderr",
     "output_type": "stream",
     "text": [
      "WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format is
considered legacy. We recommend using instead the native Keras format,
e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model,
'my_model.keras')`. \n"
     ]
    }
   ],
   "source": [
    "os.makedirs(\"model\", exist_ok=True)\n",
    "model.save(\"model/poultry_model.h5\")\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "a0922ecf-7fee-47e9-9904-e619f1cceb54",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "\u001b[1m 4/43\u001b[0m \u001b[32m━\u001b[0m\u001b[37m━━━━━━━
━━━━━━━━━━━━\u001b[0m \u001b[1m3:39\u001b[0m 6s/step"
     ]
    }
   ],
   "source": [
    "val_generator.reset()\n",
    "preds = model.predict(val_generator)\n",
    "y_pred = np.argmax(preds, axis=1)\n",
    "y_true = val_generator.classes\n",
    "\n",
    "print(\"Classification Report:\\n\", classification_report(y_true,
y_pred, target_names=list(val_generator.class_indices.keys())))\n",
    "print(\"Confusion Matrix:\\n\", confusion_matrix(y_true, y_pred))\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "5bfc268c-ddb3-434a-ace4-0b198835d26f",
   "metadata": {},
   "outputs": [],
   "source": [
    "model = load_model(\"model/poultry_model.h5\")\n",
    "\n",
    "img_path = \"static/uploaded/image1.jpg\"  # Replace with actual test
image path\n",
    "img = cv2.imread(img_path)\n",
    "img = cv2.resize(img, (img_width, img_height))\n",
    "img = img / 255.0\n",
    "img = np.expand_dims(img, axis=0)\n",
    "\n",
    "prediction = model.predict(img)\n",
    "predicted_class = np.argmax(prediction)\n",
    "\n",
    "class_labels = list(train_generator.class_indices.keys())\n",
    "print(\"Predicted Class:\", class_labels[predicted_class])\n"
   ]
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3 (ipykernel)",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.12.7"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 5
}
```