Python code

```
import os
import numpy as np
import tensorflow
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import classification_report, confusion_matrix
import cv2
base_dir = 'dataset'
img_height, img_width = 224, 224
batch_size = 32
epochs = 10

train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    horizontal_flip=True,
    zoom_range=0.2,
    shear_range=0.2
)

train_generator = train_datagen.flow_from_directory(
    base_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    subset='training',
    class_mode='categorical'
)

val_generator = train_datagen.flow_from_directory(
    base_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    subset='validation',
    class_mode='categorical'
)


base_model = MobileNetV2(input_shape=(img_height, img_width, 3), include_top=False,
weights='imagenet')
base_model.trainable = False

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
predictions = Dense(train_generator.num_classes, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)

model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy',
metrics=['accuracy'])


history = model.fit(
    train_generator,
    validation_data=val_generator,
```

```
    epochs=epochs
)


os.makedirs("model", exist_ok=True)
model.save("model/poultry_model.h5")

val_generator.reset()
preds = model.predict(val_generator)
y_pred = np.argmax(preds, axis=1)
y_true = val_generator.classes

print("Classification Report:\n", classification_report(y_true, y_pred,
target_names=list(val_generator.class_indices.keys())))
print("Confusion Matrix:\n", confusion_matrix(y_true, y_pred))

model = load_model("model/poultry_model.h5")

img_path = "C:\\Users\\User\\PYTHON\\dataset"
img = cv2.imread(img_path)
img = cv2.resize(img, (img_width, img_height))
img = img / 255.0
img = np.expand_dims(img, axis=0)

prediction = model.predict(img)
predicted_class = np.argmax(prediction)


class_labels = list(train_generator.class_indices.keys())
print("Predicted Class:", class_labels[predicted_class])
```

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)
```

Front End

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Poultry Disease Detector</title>
  <style>
    body {
      margin: 0;
      font-family: 'Segoe UI', sans-serif;
      background: linear-gradient(rgba(34, 40, 49, 0.6), rgba(34, 40, 49, 0.6)),
                  url('64af17f8-7fb0-4e65-8cf4-6aec42fb59c2.png') no-repeat center center fixed;
      background-size: cover;
      display: flex;
      justify-content: center;
      align-items: center;
      min-height: 100vh;
      color: #fff;
    }

    .container {
      background: rgba(255, 255, 255, 0.95);
      padding: 30px 40px;
      border-radius: 15px;
      max-width: 500px;
      width: 90%;
      box-shadow: 0 8px 20px rgba(0, 0, 0, 0.4);
      text-align: center;
      color: #222;
    }

    h2 {
      color: #2c3e50;
      margin-bottom: 25px;
```

```
}

input[type="file"] {
  margin-top: 10px;
  padding: 8px;
  width: 100%;
}

#preview {
  display: none;
  margin-top: 15px;
  max-width: 100%;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(0,0,0,0.2);
}

button {
  background-color: #27ae60;
  color: white;
  padding: 12px 25px;
  border: none;
  margin-top: 20px;
  font-size: 16px;
  border-radius: 8px;
  cursor: pointer;
  transition: background 0.3s ease;
}

button:hover {
  background-color: #219150;
}

#loader {
  margin-top: 15px;
  color: #666;
```

```
      font-style: italic;
      display: none;
    }

    #result {
      margin-top: 20px;
      background: #ecf0f1;
      padding: 15px;
      border-left: 6px solid #2980b9;
      border-radius: 8px;
      font-size: 18px;
      color: #222;
    }
  </style>
</head>
<body>

<div class="container">
  <h2>Poultry Disease Prediction</h2>

  <form id="uploadForm" enctype="multipart/form-data" method="post" action="http://127.0.0.1:5000/predict">
    <input type="file" name="file" id="imageInput" accept="image/*" required><br>
    <img id="preview" alt="Image Preview">
    <button type="submit">Predict</button>
    <div id="loader">🔄 Analyzing image, please wait...</div>
  </form>

  <div id="result"></div>
</div>

<script>
  const form = document.getElementById('uploadForm');
  const loader = document.getElementById('loader');
  const resultDiv = document.getElementById('result');
  const imageInput = document.getElementById('imageInput');
```

```
const preview = document.getElementById('preview');

imageInput.addEventListener('change', function () {
  const file = this.files[0];
  if (file) {
    preview.src = URL.createObjectURL(file);
    preview.style.display = 'block';
  } else {
    preview.style.display = 'none';
  }
});

form.addEventListener('submit', async function (e) {
  e.preventDefault();
  loader.style.display = 'block';
  resultDiv.innerHTML = '';

  const formData = new FormData(form);

  try {
    const response = await fetch(form.action, {
      method: 'POST',
      body: formData
    });

    const data = await response.json();
    loader.style.display = 'none';

    resultDiv.innerHTML = `
      ✅ <strong>Prediction:</strong> ${data.predicted_class}<br>
      📈 <strong>Confidence:</strong> ${(data.confidence * 100).toFixed(2)}%
    `;
  } catch (err) {
    loader.style.display = 'none';
```

```
    resultDiv.innerHTML = `<span style="color:red;">❌ JS Error: ${err.message}</span>`;
  }
});
</script>

</body>
</html>
```