```python
import streamlit as st
import cv2
import numpy as np
from tensorflow.keras.models import load_model
import base64
import os
from PIL import Image

# CONFIG
MODEL_PATH = "cnnmodel.h5"
CLASSES = ['German Shepherd', 'Labrador', 'Pug','Siberian Husky']
CONFIDENCE_THRESHOLD = 0.60
INPUT_SIZE = (150, 150)


# -------------------- UI Background --------------------
def set_background(image_path):
   with open(image_path, "rb") as img_file:
       encoded = base64.b64encode(img_file.read()).decode()

   st.markdown(
       f"""
       <style>
       .stApp {{
           background-image: url(data:image/png;base64,{encoded});
           background-size: cover;
           background-position: center;
       }}
       [data-testid="stHeader"] {{
           background-color: rgba(0,0,0,0);
       }}
       </style>
       """,
       unsafe_allow_html=True
   )


# -------------------- Model Loading --------------------
@st.cache_resource
def load_dog_breed_model(path):
   if not os.path.exists(path):
       st.error("Model file not found.")
       st.stop()
   model = load_model(path)
   return model


# -------------------- Image Preprocessing --------------------
def preprocess_image(image):
   resized = cv2.resize(image, INPUT_SIZE)
   gray = cv2.cvtColor(resized, cv2.COLOR_BGR2GRAY)
   normalized = gray / 255.0
   return normalized.reshape(1, INPUT_SIZE[1], INPUT_SIZE[0], 1)


# -------------------- Prediction --------------------
def predict_breed(model, image):
   preprocessed = preprocess_image(image)
   prediction = model.predict(preprocessed, verbose=0)
   max_prob = np.max(prediction)
   class_idx = np.argmax(prediction)

   if max_prob >= CONFIDENCE_THRESHOLD:
       return CLASSES[class_idx], max_prob
   else:
       return "Unknown", max_prob


# -------------------- Streamlit App --------------------
def main():
   set_background("dogs.jpg")  # Optional: Path to background image

   st.sidebar.title("📌 Navigation")
   page = st.sidebar.radio("Go to", ["🏠 Home", "🐶 Run", "📷 Upload Image"])

   if page == "🏠 Home":
       st.title("Dog Breed Classifier 🐾")
       st.markdown(
           """
           <div style='font-size:16px'>
           <span style='color:#2196F3; font-weight:bold;'>
               This app uses a Convolutional Neural Network (CNN)
               to detect dog breeds in real-time using your webcam feed.
<br><br>
               📸 Upload an image frame or head over to the
               'Run' tab to test it live!
           </span>
           </div>
           """,
           unsafe_allow_html=True
       )

   elif page == "🐶 Run":
       st.title("Live Dog Breed Detection")

       model = load_dog_breed_model(MODEL_PATH)
       frame_placeholder = st.empty()
```

```
        if "run" not in st.session_state:
            st.session_state.run = False

        start = st.button("Start Webcam")
        stop = st.button("Stop")

        if start:
            st.session_state.run = True
        if stop:
            st.session_state.run = False

        cap = cv2.VideoCapture(0)
        while st.session_state.run:
            ret, frame = cap.read()
            if not ret:
                st.warning("Failed to capture frame.")
                break

            breed, confidence = predict_breed(model, frame)
            color = (0, 255, 0) if breed != "Unknown" else (0, 0, 255)
            cv2.putText(frame, f"{breed} ({confidence*100:.1f}%)", (20, 40),
                        cv2.FONT_HERSHEY_SIMPLEX, 1, color, 2)

            img_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            frame_placeholder.image(img_rgb, channels="RGB")

        cap.release()
        # Removed cv2.destroyAllWindows() to avoid errors in headless
environments

    elif page == "📷 Upload Image":
        st.title("About")
        st.title("Upload a Dog Image 🐶")

        uploaded_file = st.file_uploader("Upload an image", type=["jpg", "jpeg",
"png"])
        if uploaded_file is not None:
            image = Image.open(uploaded_file).convert("RGB")
            st.image(image, caption="Uploaded Image", use_container_width=True)

            # Convert to OpenCV format
            image_cv = np.array(image)
            image_cv = cv2.cvtColor(image_cv, cv2.COLOR_RGB2BGR)

            model = load_dog_breed_model(MODEL_PATH)
            breed, confidence = predict_breed(model, image_cv)

            st.markdown(f"### 🐾 Predicted Breed: `{breed}`")
            st.markdown(f"**Confidence:** `{confidence * 100:.2f}%`")
```

```python
if __name__ == "__main__":
    main()
```