```python
from flask import Flask, request, jsonify
from flask_cors import CORS
import cv2
import numpy as np
import mediapipe as mp
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.preprocessing import StandardScaler
import os

app = Flask(__name__)
CORS(app)

mp_drawing = mp.solutions.drawing_utils
mp_holistic = mp.solutions.holistic

model_path = "Greeting1of20.9361.h5"
loaded_model = load_model(model_path)

signs = ["good morning", "alright", "good afternoon", "how are you", "hello"]

scaler = StandardScaler()

def extract_holistic_landmarks(frame, holistic):
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    results = holistic.process(rgb_frame)

    hand_landmarks = []
    face_landmarks = []
    pose_landmarks = []

    if results.left_hand_landmarks:
        hand_landmarks.extend([(lm.x, lm.y, lm.z) for lm in results.left_hand_landmarks.landmark])
    if results.right_hand_landmarks:
        hand_landmarks.extend([(lm.x, lm.y, lm.z) for lm in results.right_hand_landmarks.landmark])
    if results.face_landmarks:
        face_landmarks.extend([(lm.x, lm.y, lm.z) for lm in results.face_landmarks.landmark])
    if results.pose_landmarks:
        pose_landmarks.extend([(lm.x, lm.y, lm.z) for lm in results.pose_landmarks.landmark])

    return {
        'hand_landmarks': np.array(hand_landmarks) if hand_landmarks else None,
        'face_landmarks': np.array(face_landmarks) if face_landmarks else None,
        'pose_landmarks': np.array(pose_landmarks) if pose_landmarks else None,
    }

@app.route('/upload_video', methods=['POST'])
def upload_video():
    if 'file' not in request.files:
        return jsonify({"error": "No file part"}), 400

    file = request.files['file']
    if file.filename == '':
        return jsonify({"error": "No selected file"}), 400

    video_path = "uploaded_video.mp4"
    file.save(video_path)

    prediction = process_video(video_path)
    os.remove(video_path)

    return jsonify({"predicted_sign": prediction})

def process_video(video_path):
    holistic = mp_holistic.Holistic(static_image_mode=False, min_detection_confidence=0.5)
    cap = cv2.VideoCapture(video_path)
    video_landmarks = []

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break
```

```
        landmarks = extract_holistic_landmarks(frame, holistic)
        combined_landmarks = []

        if landmarks['hand_landmarks'] is not None:
            combined_landmarks.extend(landmarks['hand_landmarks'])
        if landmarks['face_landmarks'] is not None:
            combined_landmarks.extend(landmarks['face_landmarks'])
        if landmarks['pose_landmarks'] is not None:
            combined_landmarks.extend(landmarks['pose_landmarks'])

        if combined_landmarks:
            video_landmarks.append(combined_landmarks)

    cap.release()
    holistic.close()

    if video_landmarks:
        x_array = np.array(video_landmarks, dtype=object)
        max_length = 543
        x_padded = pad_sequences(x_array, maxlen=max_length, padding='post', dtype='float32')
        x_scaled = scaler.fit_transform(x_padded.reshape(-1, x_padded.shape[-1]))
        x_scaled = x_scaled.reshape(x_padded.shape)

        prediction = loaded_model.predict(x_scaled)
        predicted_classes = np.argmax(prediction, axis=1)
        predicted_sign = signs[predicted_classes[0]]

        return predicted_sign
    else:
        return "No valid landmarks found in video."

if __name__ == "__main__":
    app.run(debug=True, host='0.0.0.0', port=5000)
```