```
# Import necessary libraries
import os  # Operating System library for file and folder operations
from flask import Flask, render_template, request, redirect, url_for  # Flask
for web application
from summarizer import Summarizer  # Summarizer library for text summarization
import subprocess  # Subprocess module for running external commands
import youtube_dl  # YouTube downloader library
import moviepy.editor as mp  # MoviePy library for video editing
import speech_recognition as sr  # SpeechRecognition library for audio
processing
from pytube import YouTube  # pytube library for downloading YouTube videos
import pyaudio  # PyAudio library for audio input/output
import wave  # Wave module for working with WAV files
from pocketsphinx import LiveSpeech  # PocketSphinx library for speech
recognition
from pydub import AudioSegment  # pydub library for audio processing
from pydub.playback import play  # Playback module for playing audio
import librosa  # Librosa library for audio analysis
import noisereduce as nr  # Noise reduction library
import numpy as np  # NumPy library for numerical operations
import time  # Time module for time-related functionality
app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = 'without\\uploads'  # Define the folder for file
uploads

# Ensure the "uploads" folder exists
if not os.path.exists(app.config['UPLOAD_FOLDER']):
    os.makedirs(app.config['UPLOAD_FOLDER'])

# Define the functions for downloading, extracting audio, transcribing, and
summarizing

def download_video_audio(video_url, video_directory):
    try:
        # Download the YouTube video using pytube
        yt = YouTube(video_url)
        yt.streams.filter(only_audio=True).first().download(output_path=video_
directory, filename="vedio.mp4")
        return os.path.join(video_directory, "vedio.mp4")
    except Exception as download_error:
        print("Error while downloading the video:", download_error)
        return None

def convert_audio_to_wav(audio_file, video_directory):
    try:
        output_audio_file = os.path.join(video_directory, "audio.wav")
        clip = mp.AudioFileClip(audio_file)
        clip.write_audiofile(output_audio_file)
```

```
        return output_audio_file
    except Exception as conversion_error:
        print("Error while converting audio:", conversion_error)
        return None

def extract_audio(video_file, output_path):
    try:
        # Check if the video file exists
        if not os.path.exists(video_file):
            raise FileNotFoundError(f"Video file '{video_file}' not found.")

        # Open the video file
        video = mp.VideoFileClip(video_file)

        # Check if the video has a valid fps
        fps = getattr(video, 'fps', None)
        if fps is None:
            raise ValueError("Invalid video fps")

        # Define the output audio file path
        audio_file = os.path.join(output_path, "audio.wav")

        # Extract and write the audio to a WAV file
        video.audio.write_audiofile(audio_file)

        return audio_file
    except (FileNotFoundError, ValueError) as e:
        print(f"Error: {str(e)}")
    except Exception as e:
        print(f"Error extracting audio: {str(e)}")
    finally:
        # Close the video object in the finally block to ensure it is closed
        if 'video' in locals():
            video.close()

    return None

def extract_audio_ffmeg(video_file, output_path):
    try:
        # set the name of the output audio file
        audio_name = f"audio.wav"

        # set the full path of the output audio file
        audio_path = os.path.join(output_path, audio_name)

        # extract the audio from the video by using ffmpeg
        command = ['ffmpeg', '-i', video_file, '-vn', '-ar', '44100', '-ac',
'2', '-sample_fmt', 's16', audio_path]
```

```
        subprocess.run(command, stdout=subprocess.PIPE,
stderr=subprocess.PIPE, text=True)

        return audio_path
    except Exception as e:
        print(f"Error extracting audio: {str(e)}")

    return None

def convert_audio_format(input_audio_file, output_audio_file):
    if not os.path.exists(output_audio_file):
        try:
            video = mp.VideoFileClip(input_audio_file)
            video.audio.write_audiofile(output_audio_file)  # Use 'pcm_s16le'
codec for WAV format
            video.close()
            return output_audio_file
        except Exception as e:
            print(f"Error converting audio: {str(e)}")
            return None
    else:
        return output_audio_file

def transcribe_audio(audio_file):
    if not os.path.exists(audio_file):
        print(f"Audio file '{audio_file}' does not exist.")
        return None

    # Check the format of the audio file and convert it to WAV if needed
    audio_format = audio_file.split('.')[-1]
    if audio_format != 'wav':
        try:
            # Convert the audio file to WAV format
            output_audio_file = audio_file.replace(audio_format, 'wav')
            audio = mp.AudioFileClip(audio_file)
            audio.write_audiofile(output_audio_file)  # Use 'pcm_s16le' codec
for WAV format
            audio.close()
            audio_file = output_audio_file
        except Exception as e:
            print(f"Error converting audio to WAV format: {str(e)}")
            return None

    try:
        recognizer = sr.Recognizer()
        with sr.AudioFile(audio_file) as source:
            audio = recognizer.record(source)
        return recognizer.recognize_google(audio, language="en-US")
```

```
    except Exception as e:
        print(f"Error transcribing audio: {str(e)}")
        return None

def transcribe_audio_uncheck(audio_file):
    try:
        recognizer = sr.Recognizer()
        with sr.AudioFile(audio_file) as source:
            audio = recognizer.record(source)
            return recognizer.recognize_google(audio, language="en-US",
show_all=True)['alternative'][0]['transcript'].strip().lower()
    except Exception as e:
        print(f"Error transcribing audio: {str(e)}")
        return None

def summarize_text(text):
    try:
        model = Summarizer()
        summary = model(text)
        return summary
    except Exception as e:
        print(f"Error summarizing text: {str(e)}")
        return None

def save_transcribed_text(transcribed_text):
    text_file_path = os.path.join(app.config['UPLOAD_FOLDER'],
'transcribed_text.txt')
    with open(text_file_path, 'w') as text_file:
        text_file.write(transcribed_text)

# Summarize microphone audio
def summarize_microphone_audio():
    try:
        # Record audio from the microphone
        recognizer = sr.Recognizer()
        with sr.Microphone() as source:
            print("Say something...")
            audio = recognizer.listen(source)

        # Transcribe the recorded audio
        audio_text = recognizer.recognize_google(audio)
        if audio_text:
            summary = summarize_text(audio_text)
            if summary:
                return summary
            else:
                return "Summary generation failed."
        else:
```

```
            return "No speech detected."
    except sr.RequestError as e:
        return f"Could not request results: {str(e)}"
    except sr.UnknownValueError:
        return "No speech detected."

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

@app.route('/summarize_video_audio', methods=['POST'])
def summarize_video_audio():
    # Return the summary as a string
    data_folder = "without\\data\\download_for_url"
    if not os.path.exists(data_folder):
        os.makedirs(data_folder)

    youtube_url = request.form['youtube_url']
    video_file_path = download_video_audio(youtube_url, data_folder)

    if video_file_path:

        extracted_audio_file = extract_audio_ffmeg(video_file_path,
data_folder)

        if extracted_audio_file:
            transcribed_text = transcribe_audio_uncheck(extracted_audio_file)

            if transcribed_text:
                summary = summarize_text(transcribed_text)

                if summary:
                    return summary
                else:
                    return "Summary generation failed."
            else:
                return "Transcription failed. Check the audio file for
issues."
        else:
            return "Audio conversion failed."
    else:
        return "Video download failed. Check the video URL and your internet
connection."

def remove_noise(audio_file):
    try:
        # Read the audio file
        audio_data, sr = librosa.load(audio_file, sr=None)
```

```
        # Apply noise reduction
        reduced_noise = nr.reduce_noise(audio_clip=audio_data,
noise_clip=audio_data)

        # Save the noise-reduced audio
        librosa.output.write_wav(audio_file, reduced_noise, sr)

        print("Noise reduction complete")
    except Exception as e:
        print(f"Error removing noise: {str(e)}")

@app.route('/summarize_microphone_audio', methods=['POST'])
def summarize_microphone_audio_route():
    # Function to record audio from the microphone
    def record_microphone_audio(filename):
        FORMAT = pyaudio.paInt16
        CHANNELS = 1
        RATE = 16000
        CHUNK = 1024
        RECORD_SECONDS = 10  # Adjust the recording duration as needed

        audio = pyaudio.PyAudio()

        stream = audio.open(format=FORMAT, channels=CHANNELS, rate=RATE,
input=True, frames_per_buffer=CHUNK)

        print("Recording...")

        frames = []

        for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
            data = stream.read(CHUNK)
            frames.append(data)

        print("Finished recording")

        stream.stop_stream()
        stream.close()
        audio.terminate()

        with wave.open(filename, 'wb') as wf:
            wf.setnchannels(CHANNELS)
            wf.setsampwidth(audio.get_sample_size(FORMAT))
            wf.setframerate(RATE)
            wf.writeframes(b''.join(frames))
    data_folder = f"without\\data\\download_from_microphone"
    # WAVE_OUTPUT_FILENAME = f'{data_folder}\\temp_{time}.wav'
```

```
    if not os.path.exists(data_folder):
        os.makedirs(data_folder)
    # Record audio from the microphone and save it to a file
    AUDIO_FILENAME = f"recorded_audio.wav"
    FILENAME = os.path.join(data_folder, AUDIO_FILENAME )
    record_microphone_audio(FILENAME)

    # Remove background noise from the recorded audio
    remove_noise(FILENAME)

    # Summarize the transcribed audio
    transcribed_text = transcribe_audio(FILENAME)
    if transcribed_text:
        summary = summarize_text(transcribed_text)
        if summary:
            return summary
        else:
            return "Summary generation failed."
    else:
        return "Transcription failed. Check the audio file for issues."

@app.route('/summarize_file_audio', methods=['POST'])
def summarize_file_audio():
    uploaded_file = request.files['video_upload']  # Get the uploaded file

    if uploaded_file:
        # Save the uploaded file to the "uploads" folder
        video_file_path = os.path.join(app.config['UPLOAD_FOLDER'],
uploaded_file.filename) # type: ignore
        uploaded_file.save(video_file_path)

        # Process the uploaded video file
        if video_file_path:
            data_folder = "without\\uploads"

            if not os.path.exists(data_folder):
                os.makedirs(data_folder)

            extracted_audio_file = extract_audio(video_file_path, data_folder)

            if extracted_audio_file:
                transcribed_text = transcribe_audio(extracted_audio_file)

                if transcribed_text:
                    # Save the transcribed text
                    save_transcribed_text(transcribed_text)
                    summary = summarize_text(transcribed_text)
```

```
                    if summary:
                        return summary
                    else:
                        return "Summary generation failed."
                else:
                    return "Transcription failed. Check the audio file for
issues."
            else:
                return "Audio extraction failed."
        else:
            return "Video processing failed. Check the uploaded file."
    else:
        return "No file uploaded."

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