```
# --- Install dependencies (run only once) ---
!pip install -q gradio deep-translator langdetect gTTS SpeechRecognition pydub
!apt-get -qq update && apt-get -qq install -y ffmpeg

# --- Imports ---
import re, math, tempfile, os
from collections import Counter
import gradio as gr
from langdetect import detect, LangDetectException
from deep_translator import GoogleTranslator
from gtts import gTTS
import speech_recognition as sr
from pydub import AudioSegment

# --- Stopwords & Regex ---
STOPWORDS = {"a","an","the","and","or","but","if","while","with","to","from",
"in","on","at","for","of","by","as","is","am","are","was","were",
"be","been","being","it","its","this","that","these","those","he",
"she","they","them","his","her","their","you","your","i","me","my",
"we","our","us","do","does","did","doing","done","not","no","nor"}

_sentence_splitter = re.compile(r'(?<=[.!?])\s+(?=[A-Z0-9("\'])')
_word_re = re.compile(r"[A-Za-z0-9']+")

# --- Text Processing ---
def split_sentences(text):
    text = text.strip()
    if not text:
        return []
    text = re.sub(r'\s+', ' ', text)
    return _sentence_splitter.split(text)

def tokenize(text):
    return [w.lower() for w in _word_re.findall(text)]

def build_vocab(sentences):
    tf = Counter()
    for sentence in sentences:
        for word in tokenize(sentence):
            if word not in STOPWORDS and len(word) > 1:
                tf[word] += 1
    return tf

def compute_idf(sentences, vocab):
```

```
    N = len(sentences)
    df = Counter()
    for sentence in sentences:
        seen = set()
        for word in tokenize(sentence):
            if word in vocab:
                seen.add(word)
        for word in seen:
            df[word] += 1
    return {word: math.log((1 + N) / (1 + df[word])) + 1 for word in vocab}

def sentence_scores(sentences, tf, idf):
    scores = []
    for i, sentence in enumerate(sentences):
        words = [w for w in tokenize(sentence) if w in tf]
        if not words:
            scores.append((0.0, i))
            continue
        score = sum(tf[w] * idf[w] for w in words) / (len(words) ** 0.8)
        scores.append((score, i))
    return scores

def summarize(text, ratio=0.3, min_sentences=2, max_sentences=5):
    sentences = split_sentences(text)
    if not sentences:
        return ""
    tf = build_vocab(sentences)
    if not tf:
        return " ".join(sentences[:min_sentences])
    idf = compute_idf(sentences, tf)
    scores = sentence_scores(sentences, tf, idf)
    k = int(round(len(sentences) * ratio))
    k = max(min_sentences, k)
    k = min(max_sentences, len(sentences))
    top_indices = [i for _, i in sorted(scores, key=lambda x: x[0], reverse=True)[:k]]
    top_indices.sort()
    return " ".join(sentences[i] for i in top_indices)

# --- Speech-to-text ---
def speech_to_text(audio_path):
    r = sr.Recognizer()
    if not audio_path:
        return None
    # convert to wav if needed
```

```
    ext = os.path.splitext(audio_path)[1].lower()
    wav_path = audio_path
    if ext != ".wav":
        tmp_wav = tempfile.NamedTemporaryFile(suffix=".wav", delete=False)
        AudioSegment.from_file(audio_path).export(tmp_wav.name, format="wav")
        wav_path = tmp_wav.name
    try:
        with sr.AudioFile(wav_path) as source:
            audio_data = r.record(source)
        return r.recognize_google(audio_data)
    except sr.UnknownValueError:
        return "⚠ Could not understand audio"
    except Exception as e:
        return f"⚠ Speech recognition error: {e}"

# --- Main Summarizer Function ---
def summarize_colab(input_text, ratio, min_sentences, max_sentences, output_lang,
voice_output, audio_file):
    # If audio is uploaded, transcribe it
    if audio_file:
        stt = speech_to_text(audio_file)
        if not stt or stt.startswith("⚠"):
            return stt if isinstance(stt, str) else "⚠ Audio could not be processed", None
        input_text = stt

    if not input_text or not str(input_text).strip():
        return "⚠ Please enter text or upload audio", None

    try:
        try:
            input_lang = detect(input_text)
        except LangDetectException:
            input_lang = 'en'

        if input_lang != 'en':
            input_text = GoogleTranslator(source='auto', target='en').translate(input_text)

        summary = summarize(input_text, ratio, min_sentences, max_sentences)

        if output_lang != 'en':
            summary = GoogleTranslator(source='en', target=output_lang).translate(summary)

        audio_path = None
        if voice_output:
```

```
            tts = gTTS(text=summary, lang=output_lang)
            tmp_file = tempfile.NamedTemporaryFile(delete=False, suffix=".mp3")
            tts.save(tmp_file.name)
            audio_path = tmp_file.name

        return summary, audio_path

    except Exception as e:
        return f"⚠ Error: {e}", None

# --- Gradio Interface ---
iface = gr.Interface(
    fn=summarize_colab,
    inputs=[
        gr.Textbox(lines=10, placeholder="Type text here...", label="Input Text"),
        gr.Slider(0.1, 0.9, value=0.4, label="Summary Ratio"),
        gr.Slider(1, 5, value=2, step=1, label="Minimum Sentences"),
        gr.Slider(1, 10, value=5, step=1, label="Maximum Sentences"),
        gr.Dropdown(
            choices=[("English", "en"), ("Hindi", "hi"), ("French", "fr"),
                     ("German", "de"), ("Spanish", "es"), ("Tamil", "ta")],
            label="Output Language",
            value="en"
        ),
        gr.Checkbox(label="Enable Voice Output", value=False),
        # Gradio 4.x style: only type="filepath"
        gr.Audio(type="filepath", label="🎙 Upload Audio (optional)")
    ],
    outputs=[
        gr.Textbox(label="Summary Output"),
        gr.Audio(label="Voice Output")
    ],
    title="Multilingual Summarizer (Colab Compatible)",
    description="Paste text or upload an audio file. Summarizes and optionally outputs voice."
)

iface.launch(share=True)
```