```python
import streamlit as st
import speech_recognition as sr
from deep_translator import GoogleTranslator
from gtts import gTTS
import tempfile
import os
from io import BytesIO
import base64
import time

# App title and description
st.set_page_config(page_title="TransLingua", page_icon="@", layout="wide")
st.title("TransLingua: AI-Powered Multilingual Translator")
st.markdown("#### An alternative to Google Translate with speech capabilities")

# Define available languages
LANGUAGES = {
    'auto': 'Detect Language',
    'en': 'English',
    'es': 'Spanish',
    'fr': 'French',
    'de': 'German',
    'it': 'Italian',
    'pt': 'Portuguese',
    'ru': 'Russian',
    'ja': 'Japanese',
    'zh-CN': 'Chinese (Simplified)',
    'ar': 'Arabic',
    'hi': 'Hindi',
    'ko': 'Korean',
    'tr': 'Turkish',
    'vi': 'Vietnamese',
    'nl': 'Dutch',
    'sv': 'Swedish',
}

# Function to convert text to speech
def text_to_speech(text, language):
    try:
        tts = gTTS(text=text, lang=language, slow=False)
        fp = BytesIO()
        tts.write_to_fp(fp)
        return fp
    except Exception as e:
        st.error(f"TTS Error: {e}")
        return None

# Function to create an autoplay audio element
def autoplay_audio(audio_bytes):
    b64 = base64.b64encode(audio_bytes.getvalue()).decode()
    md = f"""
        <audio autoplay>
            <source src="data:audio/mp3;base64,{b64}" type="audio/mp3">
        </audio>
    """
    st.markdown(md, unsafe_allow_html=True)

# Function to recognize speech from microphone
def recognize_speech(language="en-US"):
    r = sr.Recognizer()
    with sr.Microphone() as source:
        st.write("Listening...")
        r.adjust_for_ambient_noise(source)
        audio = r.listen(source)
        st.write("Processing speech...")

    try:
        if language == "auto" or language == "en":
            text = r.recognize_google(audio)
        else:
            text = r.recognize_google(audio, language=language)
        return text
    except sr.UnknownValueError:
        st.error("Could not understand audio")
        return None
    except sr.RequestError as e:
        st.error(f"Error with the speech recognition service: {e}")
        return None

# Sidebar for app settings
with st.sidebar:
    st.header("Settings")

    # Translation direction
    st.subheader("Translation Direction")
    source_lang = st.selectbox("Source Language", list(LANGUAGES.keys()),
                               format_func=lambda x: LANGUAGES[x], index=0)
    target_lang = st.selectbox("Target Language", list(LANGUAGES.keys())[1:],
                               format_func=lambda x: LANGUAGES[x], index=0)

    # Translation options
    st.subheader("Options")
    enable_speech_input = st.checkbox("Enable Speech Input", value=True)
    enable_speech_output = st.checkbox("Enable Speech Output", value=True)

    # About section
    st.markdown("---")
    st.markdown("## About TransLingua")
    st.info("""
    TransLingua is an open-source alternative to Google Translate.

    Features:
    - Text translation
    - Speech-to-text
    - Text-to-speech
    - Auto language detection
```

```
    Built with Streamlit, DeepTranslator, gTTS, and SpeechRecognition.
    """)

# Main app area
st.markdown("---")

# Input section
col1, col2 = st.columns(2)
with col1:
    st.subheader(f"Input ({LANGUAGES[source_lang]})")

    # Input options
    input_method = st.radio("Input Method",
                            ["Text Input", "Voice Input"] if enable_speech_input else ["Text Input"],
                            horizontal=True)

    if input_method == "Text Input":
        input_text = st.text_area("Enter text to translate", height=200)
    else:  # Voice Input
        if st.button("🎤 Start Recording"):
            with st.spinner("Listening..."):
                input_text = recognize_speech("auto" if source_lang == "auto" else source_lang)
                if input_text:
                    st.success("Speech recognized!")
                    st.write(f"Recognized text: {input_text}")
                else:
                    input_text = ""
        else:
            input_text = ""

    if st.button("Translate"):
        if not input_text:
            st.warning("Please enter some text to translate.")
        else:
            with st.spinner("Translating..."):
                try:
                    # Handle auto detection
                    if source_lang == "auto":
                        translator = GoogleTranslator(target=target_lang)
                    else:
                        translator = GoogleTranslator(source=source_lang, target=target_lang)

                    translated_text = translator.translate(input_text)

                    # Get detected language if auto was selected
                    detected_lang = "auto-detected"
                    if source_lang == "auto" and translated_text:
                        try:
                            detected_lang = GoogleTranslator().detect(input_text)
                            for code, name in LANGUAGES.items():
                                if code == detected_lang:
                                    detected_lang = name
                                    break
                        except:
                            detected_lang = "unknown"

                    # Store in session state
                    st.session_state.translated_text = translated_text
                    st.session_state.detected_lang = detected_lang if source_lang == "auto" else None

                except Exception as e:
                    st.error(f"Translation Error: {e}")
                    st.session_state.translated_text = ""
                    st.session_state.detected_lang = None

# Output section
with col2:
    st.subheader(f"Output ({LANGUAGES[target_lang]})")

    if 'translated_text' in st.session_state and st.session_state.translated_text:
        # Show detected language if auto was selected
        if st.session_state.detected_lang:
            st.info(f"Detected language: {st.session_state.detected_lang}")

        # Display translation
        st.markdown("### Translation:")
        st.markdown(f"<div style='background-color:black; padding: 15px; border-radius: 5px;'>{st.session_state.translated_text}</div>", unsafe_allow_html=True

        # Text-to-speech button
        if enable_speech_output:
            if st.button("🔊 Listen"):
                with st.spinner("Generating audio..."):
                    audio_bytes = text_to_speech(st.session_state.translated_text, target_lang)
                    if audio_bytes:
                        st.audio(audio_bytes, format='audio/mp3')
                        autoplay_audio(audio_bytes)

        # Copy button (JavaScript implementation)
        st.markdown("""
        <div style="display: flex; justify-content: flex-end; margin-top: 10px;">
            <button onclick="navigator.clipboard.writeText(`{}`); this.textContent='Copied!'; setTimeout(() => this.textContent='Copy to Clipboard', 2000);"
                    style="background-color: #4CAF50; color: white; border: none; padding: 5px 10px; text-align: center;
                           text-decoration: none; display: inline-block; font-size: 14px; border-radius: 4px; cursor: pointer;">
                Copy to Clipboard
            </button>
        </div>
        """.format(st.session_state.translated_text), unsafe_allow_html=True)

# History section
st.markdown("---")
st.subheader("Translation History")

# Initialize history in session state if it doesn't exist
if 'translation_history' not in st.session_state:
    st.session_state.translation_history = []
```

```
# Add current translation to history if it exists
if 'translated_text' in st.session_state and st.session_state.translated_text and input_text:
    # Add to history if it's a new translation
    if not st.session_state.translation_history or st.session_state.translation_history[0][0] != input_text:
        source_lang_display = st.session_state.detected_lang if source_lang == "auto" and st.session_state.detected_lang else LANGUAGES[source_lang]
        st.session_state.translation_history.insert(0,
            (input_text, st.session_state.translated_text, source_lang_display, LANGUAGES[target_lang], time.strftime("%H:%M:%S")))
        # Keep only the last 10 translations
        if len(st.session_state.translation_history) > 10:
            st.session_state.translation_history = st.session_state.translation_history[:10]

# Display history
if not st.session_state.translation_history:
    st.info("Your translation history will appear here.")
else:
    for i, (src_text, tgt_text, src_lang, tgt_lang, timestamp) in enumerate(st.session_state.translation_history):
        with st.expander(f"#{i+1} - {timestamp} - {src_lang} → {tgt_lang}"):
            st.markdown(f"**Source:** {src_text}")
            st.markdown(f"**Translation:** {tgt_text}")

# Footer
st.markdown("---")
st.markdown("Made with ❤ using Streamlit | TransLingua © 2025")
```