# Voice Conversion System

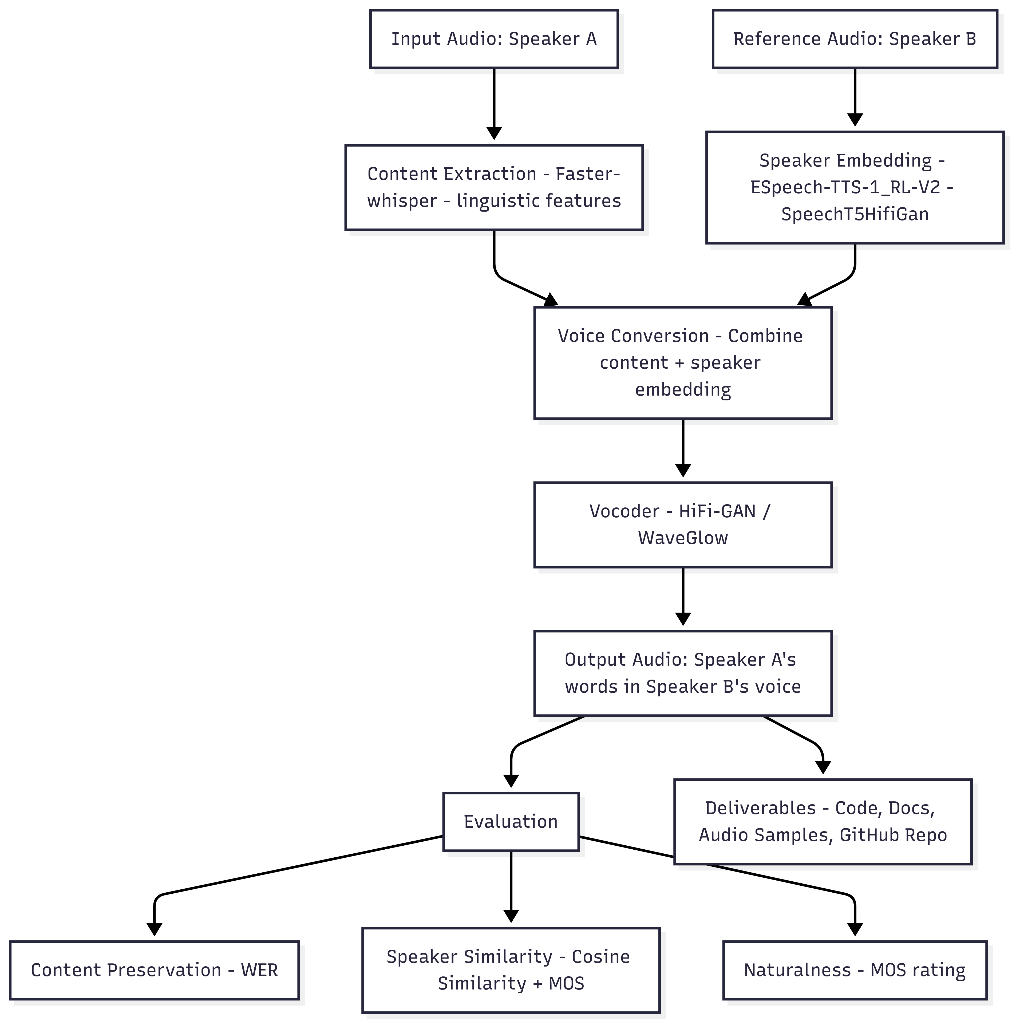
Convert Speaker As voice to Speaker Bs voice while preserving content

**System Implementation: Voice Recording, Transcription, and Speech Synthesis**

**1. Overview**

The system captures audio from the user’s microphone in a browser, transcribes it to text using **Faster-Whisper**, and optionally synthesizes speech using **SpeechT5** with voice cloning capabilities. The implementation uses Python in a **Google Colab** or Jupyter environment and relies on ffmpeg for audio conversion, torch for model inference, and soundfile for audio I/O.

**2. Workflow**

****

The system consists of three main components:

**2.1 Audio Capture**

* **Browser Recording**: Audio is recorded directly in the browser using JavaScript and the Web Audio API.
* **Data Conversion**: The recording is stored in **Base64-encoded WebM format**.
* **Python Integration**: The Base64 data is decoded in Python and converted to **WAV** format using ffmpeg.

**2.2 Transcription using Faster-Whisper**

* **Model Selection**: The medium.en Faster-Whisper model is used by default.
* **Device Selection**: The code automatically detects cuda if a GPU is available; otherwise, it falls back to cpu.

**Output**:

* Transcription text is stored in the transcribed\_text variable.
* Segment-level timestamps are printed.
* Detected language is reported.

**2.3 Speech Synthesis using SpeechT5 (Voice Cloning)**

* **Speaker Embeddings**: Synthesizing speech in a cloned voice requires a **precomputed speaker embedding**.

**Output**:

Synthesized audio is saved in recordings/ with a timestamped filename and can be played back in Colab.

**4. Dependencies**

* **Audio and I/O**: soundfile, ffmpeg-python, IPython.display
* **ASR Model**: faster-whisper
* **TTS Model**: transformers (SpeechT5 & HifiGan)
* **Core ML Framework**: torch

**5. Execution Flow**

1. **User clicks button** → Recording starts in browser.
2. **Recording stops** → Audio is sent to Python, converted to WAV.
3. **Transcription** → Faster-Whisper generates text with segment timestamps.
4. **Voice cloning (optional)** → If speaker embeddings are available, SpeechT5 synthesizes speech.
5. **Audio Output** → Original and synthesized audio are saved and played back in Colab.

**Approach, System Architecture, and Evaluation**

**1. Approach**

The overall approach for this project is an **end-to-end pipeline** that combines audio capture, speech-to-text transcription, and text-to-speech synthesis with optional voice cloning. The steps are as follows:

1. **Audio Recording**
   * Capture audio from the user via a **browser interface** using JavaScript.
   * Convert the recorded WebM audio into WAV format using **FFmpeg** for compatibility with ML models.
2. **Speech-to-Text (Transcription)**
   * Use the **Faster-Whisper** model to transcribe audio into text.
   * Detect the language automatically and generate segment-level timestamps.
   * Store the transcription in a variable (transcribed\_text) for further processing.
3. **Voice Cloning (Optional)**
   * Generate **speaker embeddings** from a reference audio of the user’s voice using **SpeechBrain**.
   * These embeddings allow the system to synthesize speech in the same voice as the reference audio.
4. **Text-to-Speech Synthesis**
   * Use **SpeechT5** along with **HifiGan vocoder** to generate natural-sounding speech from the transcribed text.
   * Apply the speaker embedding for personalized voice cloning (if available).
5. **Output and Visualization**
   * Save both original and synthesized audio in a recordings/ folder.
   * Display audio playback and waveform visualizations in the notebook.

Git repo : [https:/github.com/MohanK-17/Voice-Conversion-System/blob/main/stt\_tts.ipynb](https://github.com/MohanK-17/Voice-Conversion-System/blob/main/stt_tts.ipynb)