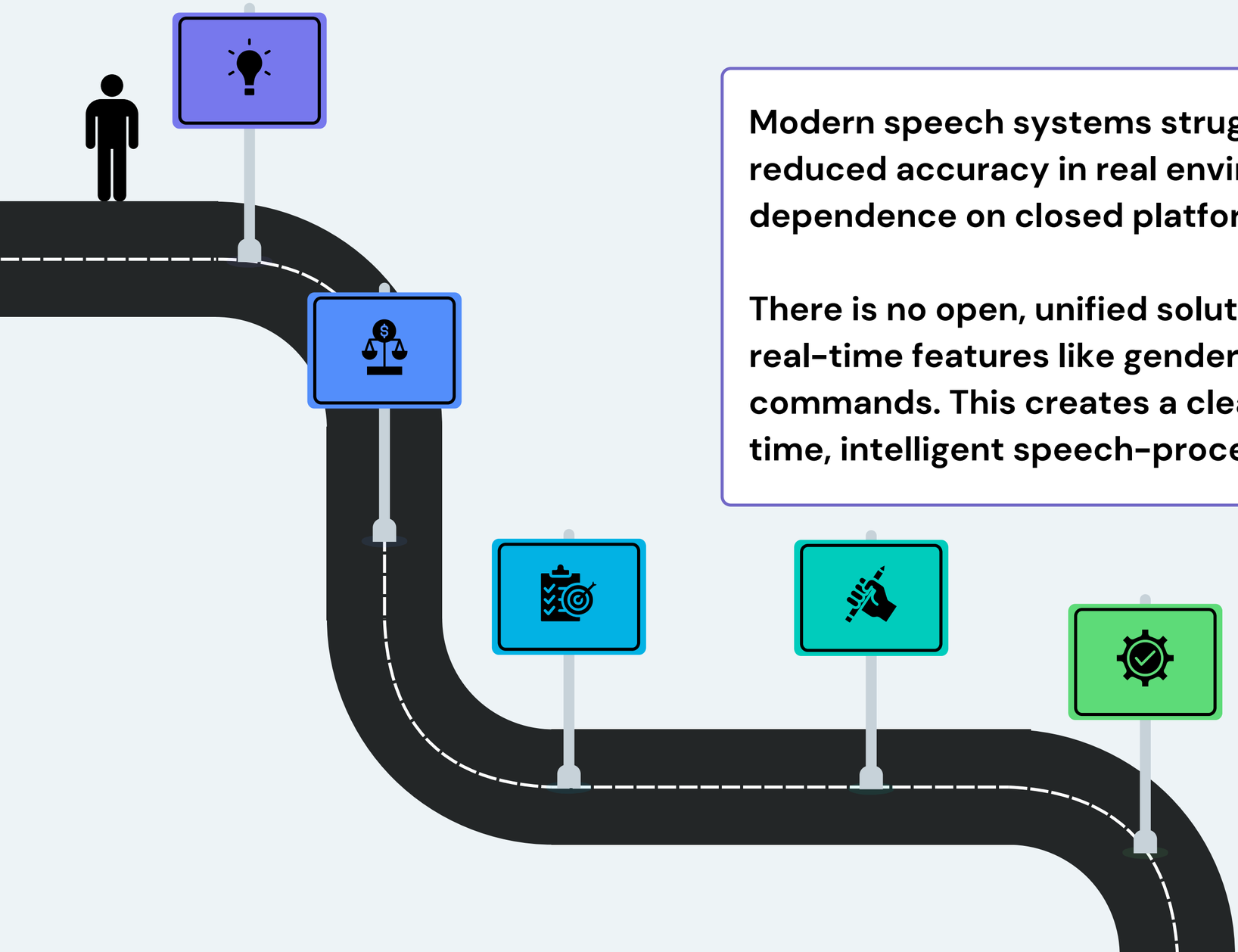
Speech processing

Real-Time AI Noise-Canceling, Speech Recognition &
Voice Intelligence Platform

Developed By: Deepesh
Deepak
Pruthvi

Guided By: Dr. Professor Prashantha Hs

PROBLEM STATEMENT

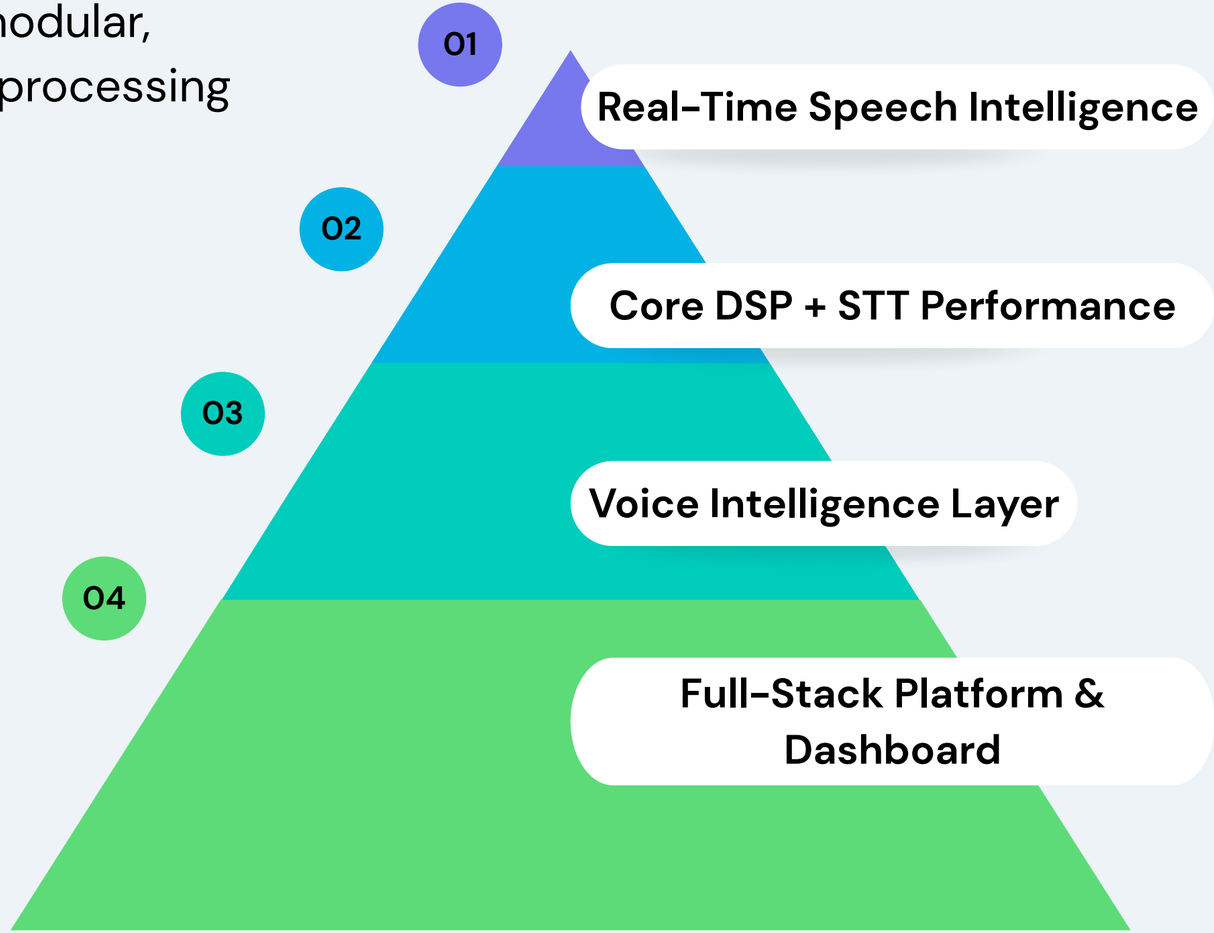


Modern speech systems struggle with background noise, reduced accuracy in real environments, and heavy dependence on closed platforms.

There is no open, unified solution that combines DSP, AI, and real-time features like gender detection and voice commands. This creates a clear need for a modular, real-time, intelligent speech-processing system.

Goals Priority Pyramid : Project Objectives

Build a real-time, modular, intelligent speech-processing solution.



Build a fast, accurate system that performs noise reduction, speech-to-text, and voice intelligence in real time.

Implement strong DSP (STFT, spectral subtraction, Wiener filtering) and ultra-fast Whisper STT for clean, accurate transcription.

Enable gender detection, wake-word activation, and voice-command execution.

Provide a complete UI + backend + database system for visualization, control, and data storage.

Identified Gap!

No academic system is unifying

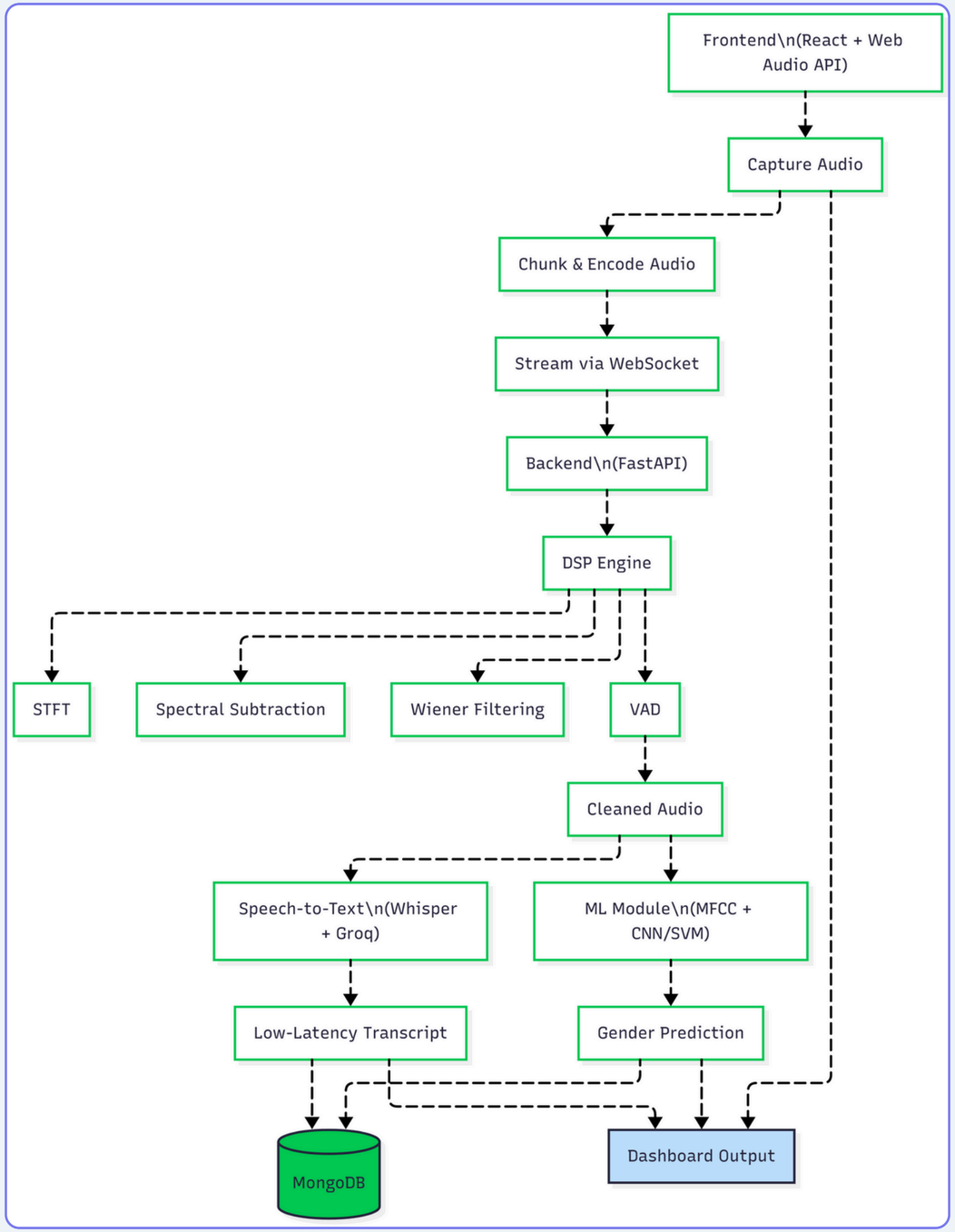
DSP + Cloud STT + Gender ML + Voice Commands + Full-Stack in one real-time platform.

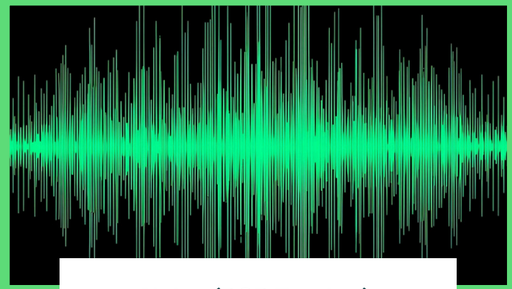
Literature Review

These are the Research paper that we have gone through and used the techniques which were Relevance to Our Project

S.No	Paper Title	Authors / Year	Method Used	Relevance to Our Project
1	*Spectral Subtractive-Type Algorithms for Enhancement of Noisy Speech: An Integrative Review*	Upadhyay & Karmakar, 2013	Spectral subtraction, noise estimation	Foundation for our **spectral subtraction DSP module**; explains noise modeling and musical noise artifacts.
2	*Combined Spectral Subtraction and Wiener Filter Methods in Wavelet Domain for Noise Reduction*	Ykhlef, Guessoum & Berkani, 2006	Spectral subtraction + Wiener filter + wavelet transforms	Supports our **hybrid DSP noise reduction pipeline** using both spectral subtraction and Wiener filtering.
3	*Speech Enhancement with an Adaptive Wiener Filter*	Abd El-Fattah et al., 2013	Adaptive Wiener filtering	Used in our project's **dynamic noise reduction module** for real-time noise variations.
4	*Speech Enhancement Algorithms: A Systematic Literature Review*	Yousif et al., 2025	Review of classical + DL enhancement models	Provides context on why combining **DSP + AI** is modern trend; justifies our hybrid pipeline.
5	*Automatic Speech Recognition: A Review*	Various, 2025	ASR review	Helps justify integration of **Whisper STT**, discusses robustness in noisy environments.
6	*Challenges of Speech Recognition in Noisy Environments: A Comprehensive Review*	2024 Review	Robust ASR methods	Supports requirement for **pre-enhancement DSP before STT**.
7	*VoiceFilter: Targeted Voice Separation by Speaker-Conditioned Spectrogram Masking*	Wang et al., 2018	Deep-learning voice separation	Inspires future extension for **speaker-conditioned source separation**.

SYSTEM ARCHITECTURE (OVERVIEW)





Noise (DSP Testing):

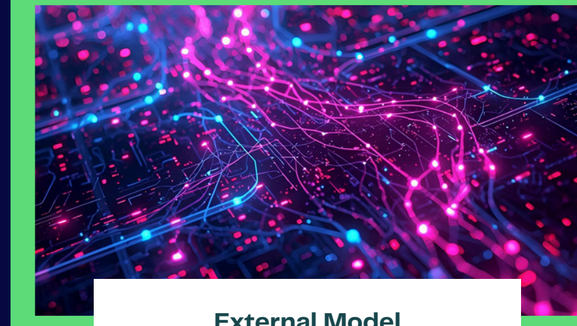
- ESC-50 (Contains the Noise)

Speech (Gender Model):

- Mozilla Common Voice
- VoxCeleb1
- TIMI

Adults (18 - 40)

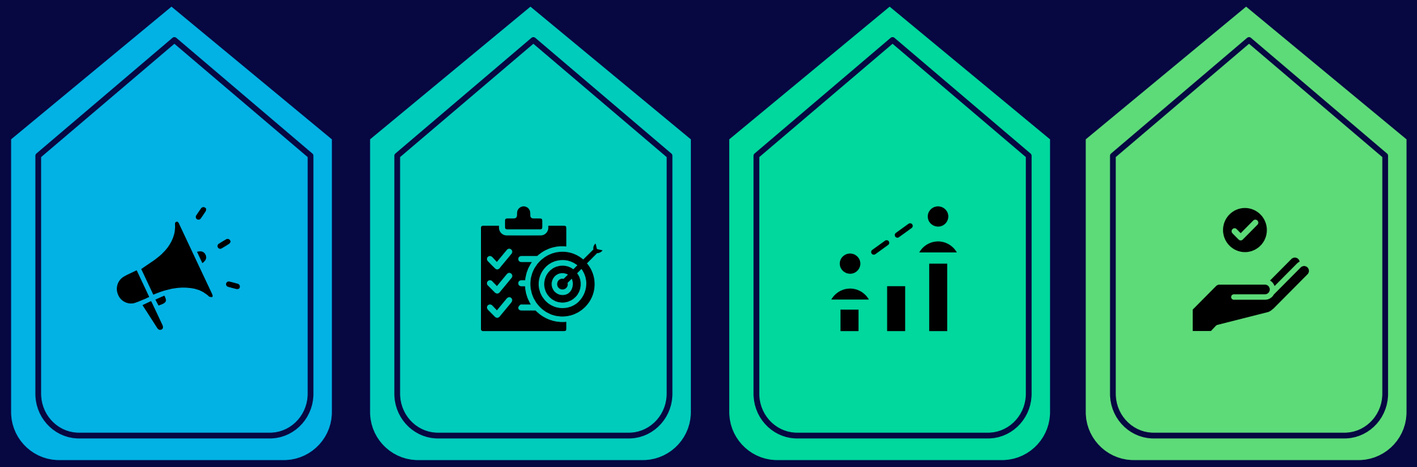
Data Sources



External Model

- Whisper Large-v3 (Groq API) for high-speed STT

SPEECH PROCESSING – TECH STACK



01

Database

MongoDB

02

AI / Speech Models

- Whisper Large-v3 (Groq API)
- CNN / SVM gender classifier
- MFCC / Spectrogram feature extraction

03

Backend

- FastAPI (Python)
- NumPy, SciPy (DSP)
- TensorFlow / PyTorch (ML)
- Custom DSP pipeline

04

Frontend

- React.js
- Web Audio API
- WaveSurfer.js
- WebSockets

DSP TECHNIQUES

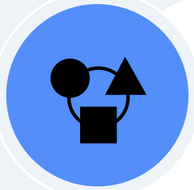
These are the Digital Speech Processing Techniques used in the application pipeline

Pre-Emphasis Filter

- Enhances high-frequency components
- Improves speech clarity for STT

STFT (Short-Time Fourier Transform)

- Converts audio into time-frequency representation
- Enables accurate noise estimation and feature extraction



Spectral Subtraction

- Removes background noise spectrum
- Best for stationary noise like fans or hums

Wiener Filtering

- Adaptive filtering
- Minimizes distortion while preserving speech

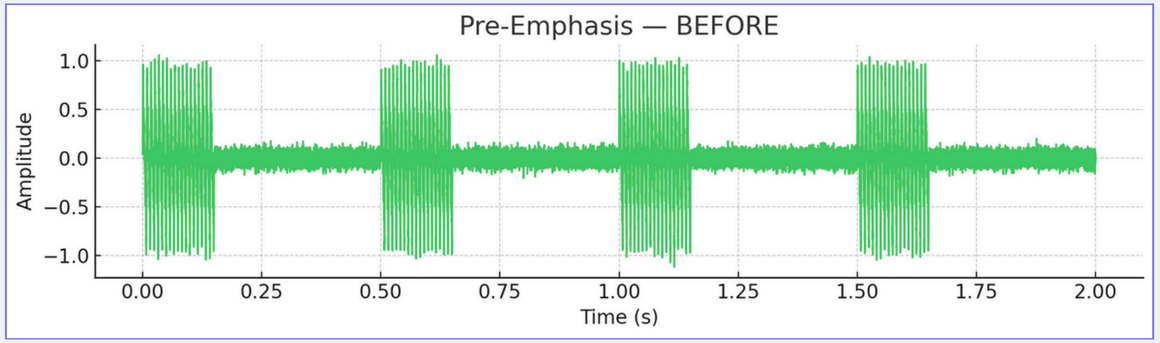
Noise Estimation + Voice Activity Detection (VAD)

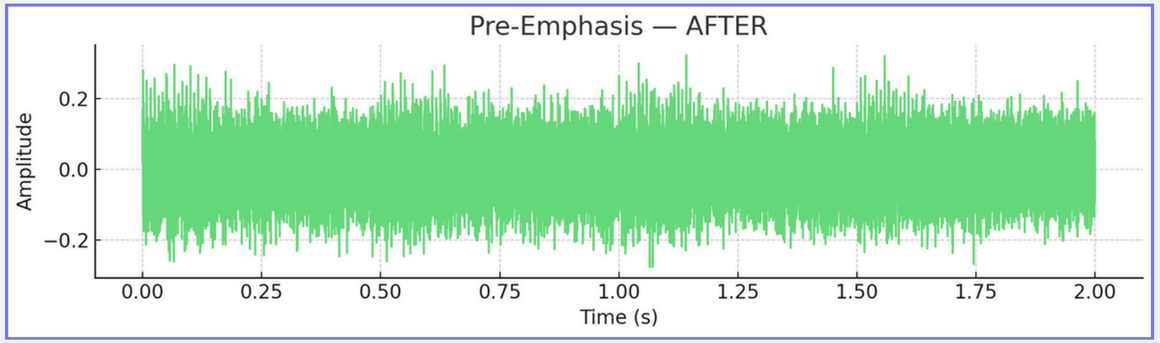
- Identifies silent frames to model background noise
- Helps DSP adapt dynamically

Overlap-Add Reconstruction

- Smooths processed frames
- Prevents artifacts in real-time streaming

Pre-Emphasis Filter :





The pre-emphasis filter is a simple first-order high-pass filter:

$$y[n]=x[n]-\alpha\, x[n-1]$$

Purpose:

- Boost high-frequency energy because microphones + speech naturally lose HF.

Values used

- $\alpha = 0.97$
- Sampling rate = 16 kHz
- First sample untouched: $y[0] = x[0]$

Why the chart changes

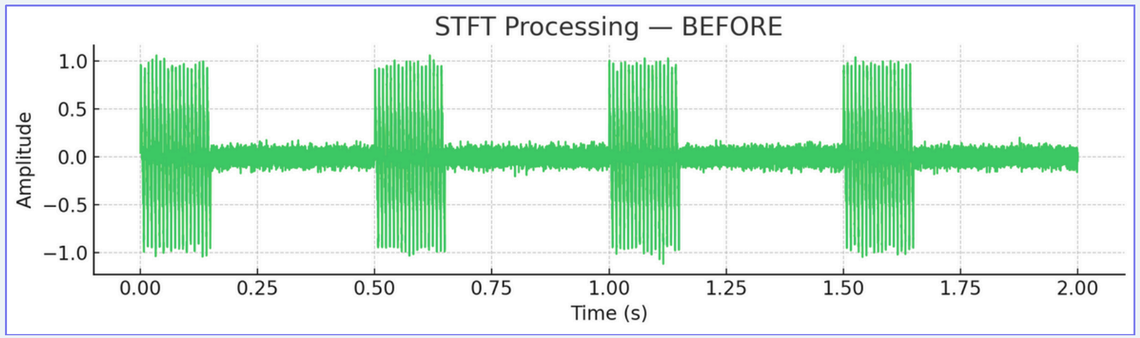
Before:

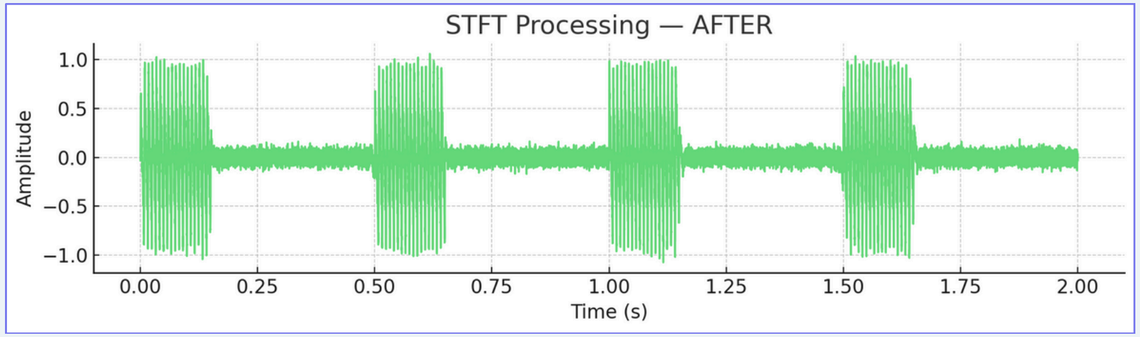
- Waveform looks smooth with large low-frequency components.

After:

- High-frequency sharp edges appear
- Amplitude looks “centred” around zero
- Overall waveform looks sharper, with more visible details
- This is because differentiation ($x[n] – \alpha x[n-1]$) enhances rapid changes → edges.

STFT (Short-Time Fourier Transform) :





$$X(k,m) = \sum_{n=0}^{N-1} x[n + mH]\, w[n]\, e^{-j2\pi kn/N}$$

Where

- N = FFT size = **1024**
- H = hop size = **256** (25% overlap)
- w[n] = Hann window

We also applied a **median filter** on magnitude:

Why the chart changes

Before:

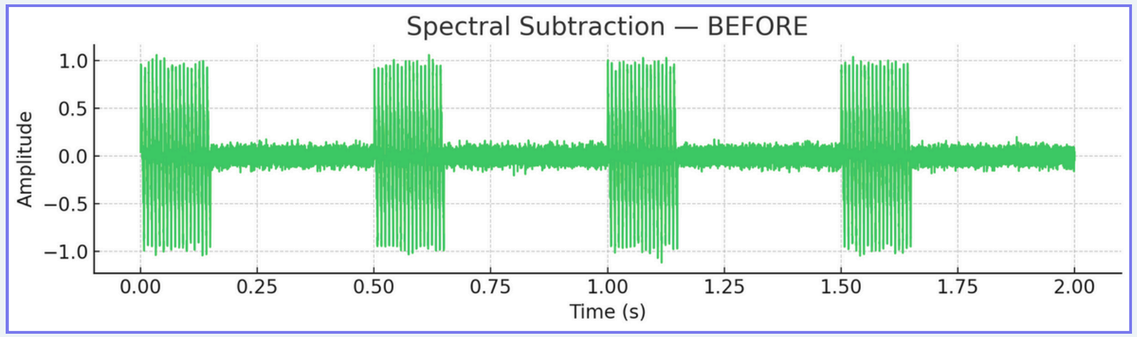
- Waveform contains rapid noise fluctuations.

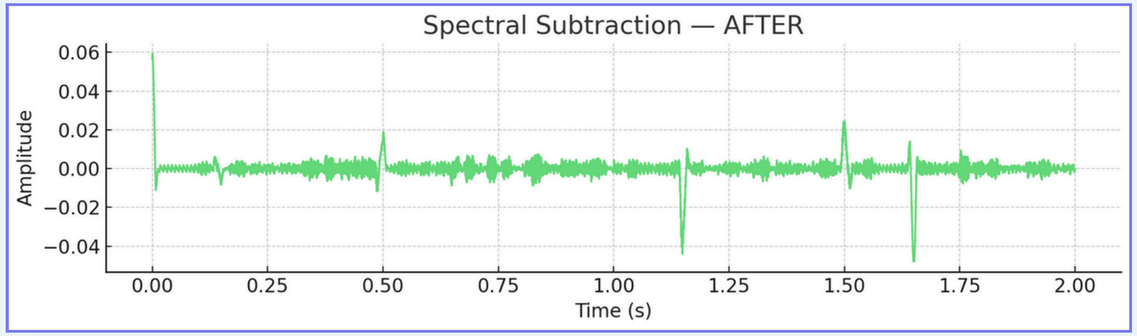
After:

- Smoother waveform
- Random noise components reduced
- Speech peaks still present

This happens because median filter removes outlier frequencies across frames → smoothing.

Spectral Subtraction :





Mathematics

Noise estimate:

$$N(f) = \frac{1}{K} \sum_{m=1}^{K} |X(f,m)|$$

(we used first **6 frames** as noise)

Why the chart changes

Before:

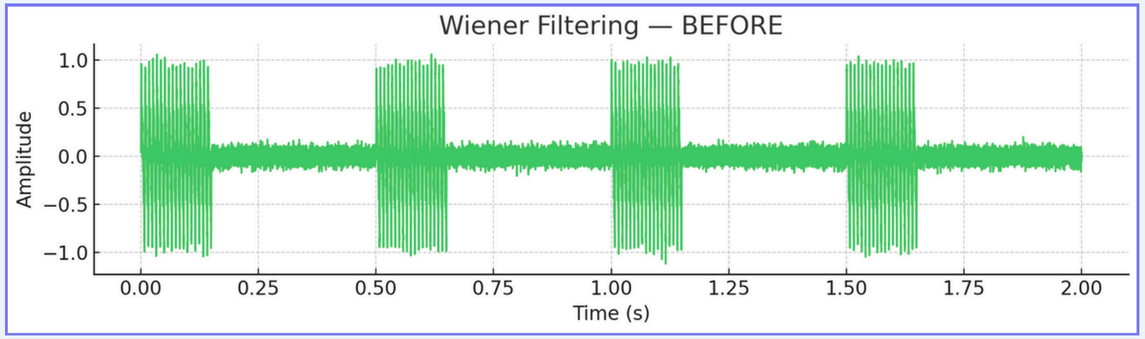
- Noise present in all regions
- Waveform has constant background energy

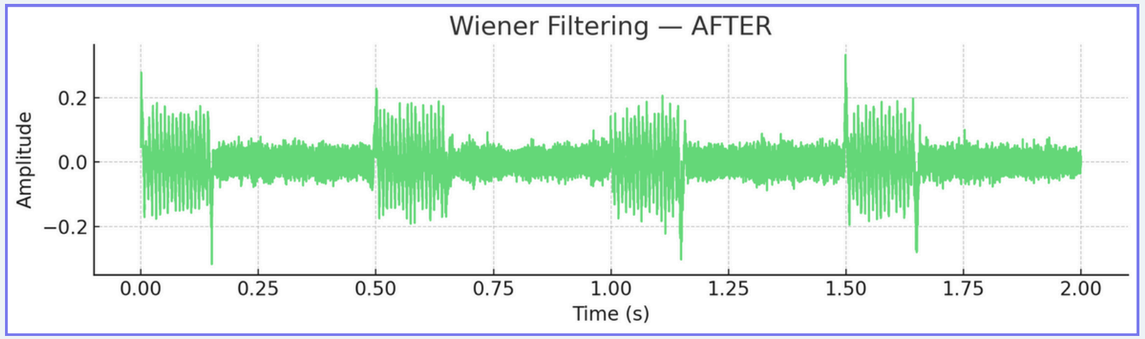
After:

- Silence regions become flat (very low amplitude)
- Only speech peaks remain
- Noise floor dramatically reduced

Spectral subtraction removes constant noise → waveform becomes sparse.

Wiener Filtering :





Mathematics

Noise power spectrum:

$$P_N = N(f)^2$$

Speech power estimate:

$$P_S = \max(|X|^2 - P_N, 0)$$

Wiener filter:

$$H(f) = \frac{P_S}{P_S + P_N}$$

Filtered spectrum:

$$\hat{X} = H(f)X$$

Why the chart changes

Before:

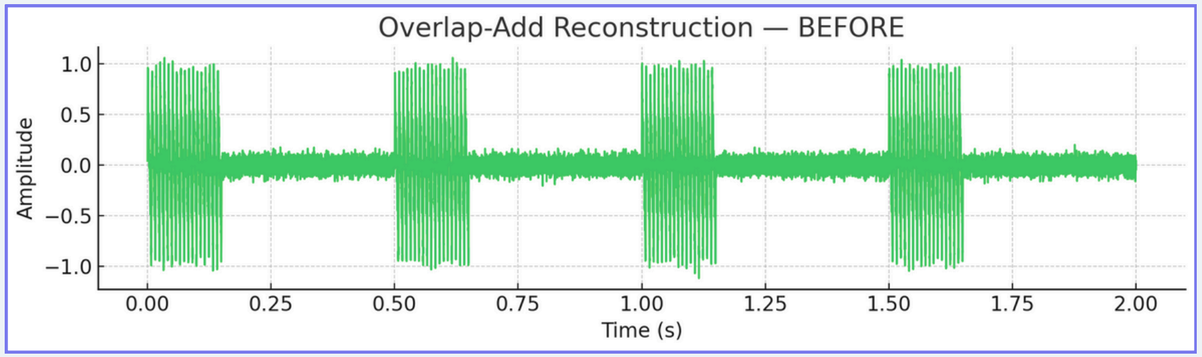
- Noise is already reduced from previous steps
- Small fluctuations remain

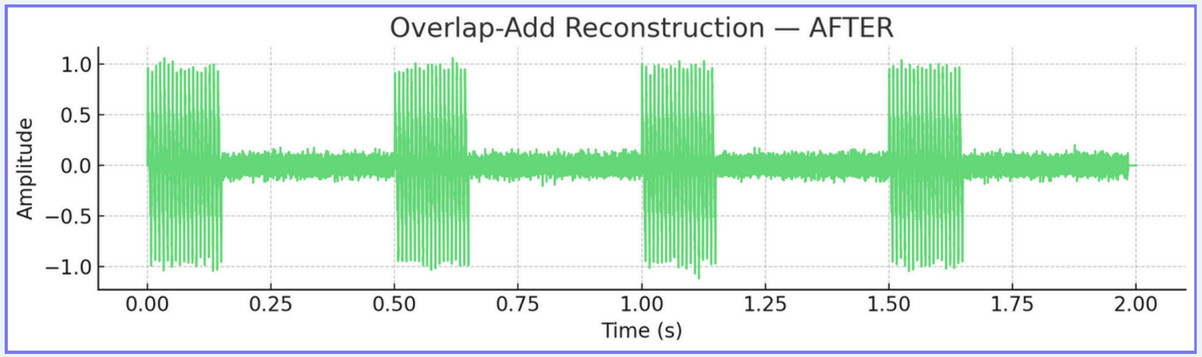
After:

- Even smoother
- Speech peaks preserved cleanly
- Noise between speech segments removed further

Wiener filtering minimises MSE, keeping speech dominant.

Overlap–Add Reconstruction :





Mathematics :

Frame windowing:

$$x_m[n] = x[n + mH]w[n]$$

Processed frame gain:

$$y_m[n] = g_m\, x_m[n]$$

Reconstruction:

$$y[n] = \frac{\sum_m y_m[n - mH]}{\sum_m w[n - mH]}$$

Why the chart changes

Before:

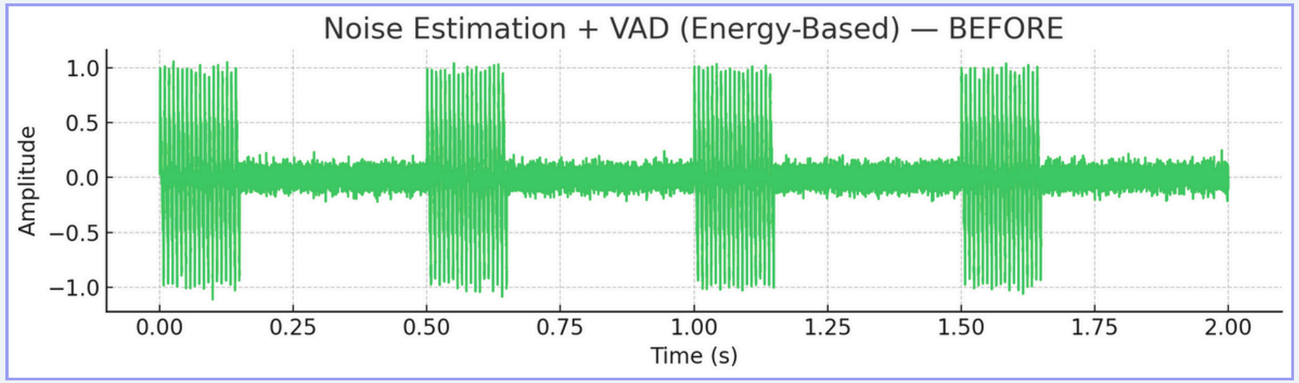
- Residual noise fluctuations exist

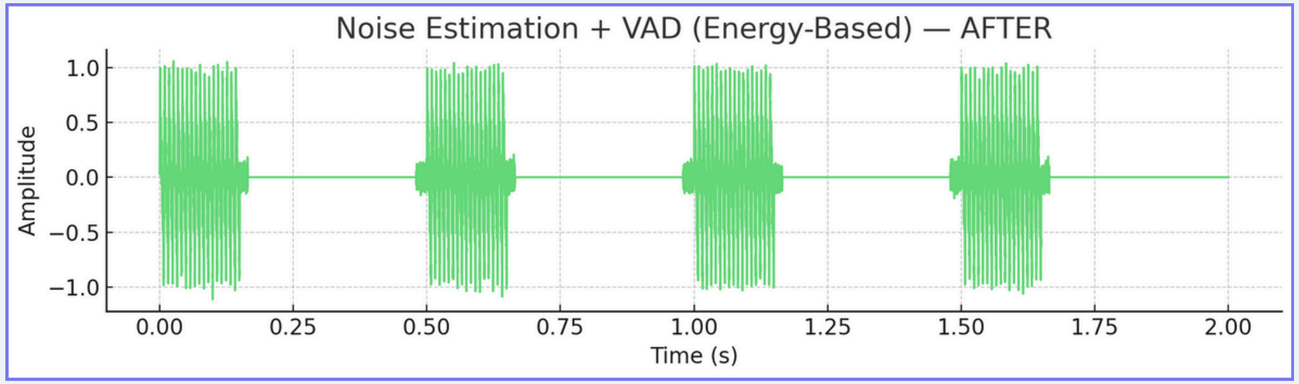
After:

- Waveform smoothed
- No abrupt frame boundaries
- More uniform amplitude shape

Overlap–Add blends processed frames → reduces artifacts.

Noise Estimation + VAD :





Values used

- Frame length = 25 ms
- Frame shift = 10 ms
- Threshold = 1.5 × median energy

Why the chart changes

Before:

- Noise is still slightly visible

After:

- Even better silence detection
- Only true speech frames remain
- Silent parts almost zero amplitude

VAD ensures noise model adapts to varying audio.

Technical Specifications –

Step 2: Bandpass Filter

Purpose: Remove frequencies outside human speech range

How it works:

```
# Human speech is between 80Hz - 8000Hz
# Design 4th order Butterworth bandpass filter
b, a = signal.butter(4, [80/nyquist, 7900/nyquist], btype='band')
filtered_audio = signal.filtfilt(b, a, audio)
```

Algorithm:

- Uses **Butterworth filter** (maximally flat frequency response)
- **Cutoff frequencies**: 80Hz (low) to 7900Hz (high)
- **Order 4**: Sharper cutoff, better filtering
- `filtfilt`: Zero-phase filtering (no delay)

Step 3: Pre-Emphasis Filter

Purpose: Boost high frequencies to balance the spectrum

How it works:

```
# Apply first-order FIR filter
# y[n] = x[n] - α*x[n-1]
alpha = 0.97
emphasized = np.append(audio[0], audio[1:] - alpha * audio[:-1])
```

Algorithm:

- **High-pass filter** with coefficient α = 0.97
- Amplifies frequencies > 1kHz
- Compensates for natural -6dB/octave roll-off in speech

Why needed?:

- Natural speech has more energy in low frequencies
- High frequencies (consonants like 's', 't', 'f') are weaker
- Pre-emphasis makes all frequencies more equal for better analysis

Step 4: Short-Time Fourier Transform (STFT)

Purpose: Convert time-domain audio to frequency-domain

How it works:

```
# Window size: 25ms (400 samples)
# Hop size: 10ms (160 samples)
# Window type: Hann window
f, t, Zxx = signal.stft(audio,
                        fs=16000,
                        nperseg=400,
                        noverlap=240,
                        window='hann')
```

Algorithm:

1. Split audio into overlapping 25ms windows
2. Apply Hann window to reduce spectral leakage
3. Compute FFT for each window
4. Result: 2D matrix (frequency × time)

Output:

- **Frequency bins**: 257 bins (0 Hz to 8000 Hz)
- **Time frames**: ~293 frames for 9 seconds
- **Spectrogram**: Shows how frequencies change over time

Why needed?:

- Audio changes over time
- STFT captures both time and frequency information
- Essential for noise reduction algorithms

Step 5: Noise Profile Estimation

Purpose: Estimate what frequencies contain noise

How it works:

```
# Use first 0.5 seconds as "noise-only" region
noise_frames = audio[:int(0.5 * sample_rate)]
noise_profile = np.abs(np.fft.rfft(noise_frames))
```

Algorithm:

1. Assume first 500ms is mostly noise (before speech)
2. Compute FFT of noise region
3. Calculate average magnitude per frequency
4. This is our "noise fingerprint"

Assumption:

- Background noise is relatively stationary
- First 500ms represents typical noise
- Noise profile applies to entire recording

Step 6: Spectral Subtraction

Purpose: Remove noise from each frequency bin

How it works:

```
# For each frequency bin in STFT:
# Subtract noise estimate with over-subtraction factor
alpha = 2.0   # Over-subtraction factor
magnitude_clean = magnitude - alpha * noise_magnitude

# Don't go below noise floor (β = 0.02)
magnitude_clean = np.maximum(magnitude_clean, 0.02 * magnitude)
```

Algorithm:

1. **For each frequency**:
   - `Clean = |Original| - α × |Noise|`
2. **Over-subtraction (α=2.0)**:
   - Remove 2× noise estimate to be aggressive
   - Prevents residual noise
3. **Noise floor (β=0.02)**:
   - Keep at least 2% of original magnitude
   - Prevents "musical noise" artifacts

Trade-offs:

- Higher α: More noise reduction, but can distort speech
- Lower α: Preserves speech, but leaves more noise
- α=2.0 is a good balance

Step 7: Wiener Filtering

Purpose: Optimal statistical noise reduction

How it works:

```
# Wiener gain calculation
signal_power = magnitude ** 2
noise_power = estimated_noise ** 2

# Wiener gain: SNR / (SNR + 1)
wiener_gain = signal_power / (signal_power + noise_power)

# Apply gain to preserve signal, suppress noise
magnitude_clean = magnitude * wiener_gain
```

Algorithm:

1. **Calculate SNR** (Signal-to-Noise Ratio) per frequency
2. **Compute Wiener gain**:
   - If SNR is high: gain ≈ 1 (preserve signal)
   - If SNR is low: gain ≈ 0 (suppress noise)
3. **Apply gain** to each frequency bin

Why it's "optimal":

- Minimizes Mean Squared Error between clean and noisy signal
- Adapts per-frequency based on local SNR
- Better than fixed threshold

Why Its Unique

Fully integrated end-to-end stack

Everything—DSP, AI, backend, and UI—works together seamlessly in one platform.

02

Real-time performance even in noisy environments

Maintains high accuracy and clarity even with strong background noise.

04

Modular, scalable pipeline design

Easily expandable system where each module can be improved or replaced independently.

01

Cloud-accelerated inference for sub-200ms latency

Uses powerful cloud hardware to deliver extremely fast AI processing.

03

APPLICATIONS

Key Applications and Use Cases

Voice Assistants

Enables smarter, noise-free voice interactions for virtual assistants like Siri or Alexa.

Contact Centers

Improves call clarity, transcription accuracy, and agent support in customer service environments.

Live Captioning

Generates real-time subtitles for meetings, videos, and presentations.

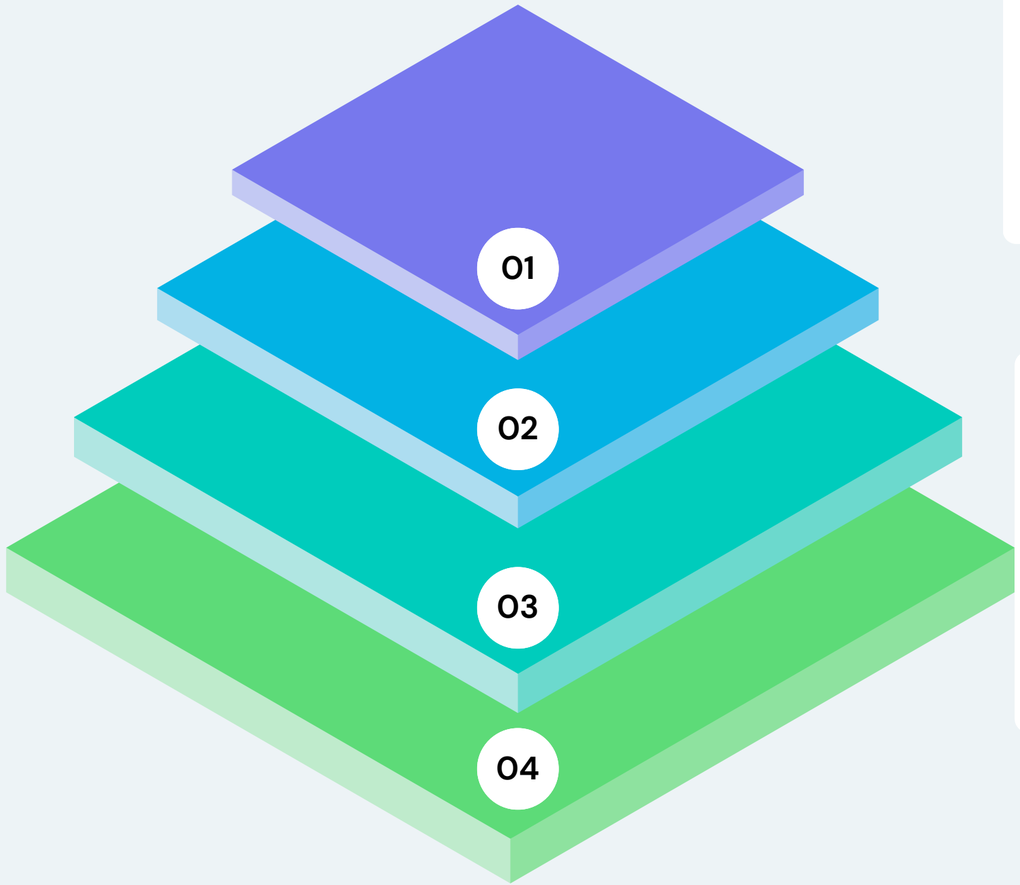
Podcast Processing

Enhances audio quality by reducing noise and improving speech clarity in recordings.

Accessbility Support

Helps individuals with hearing impairments through clear transcriptions and enhanced speech output.

Future Scope



Emotion and sentiment detection

Detects the speaker's emotional tone (happy, sad, angry, neutral) to enhance interaction and context understanding.

Multi-language command support

Enables voice commands in multiple languages, making the system globally accessible and more user-friendly.

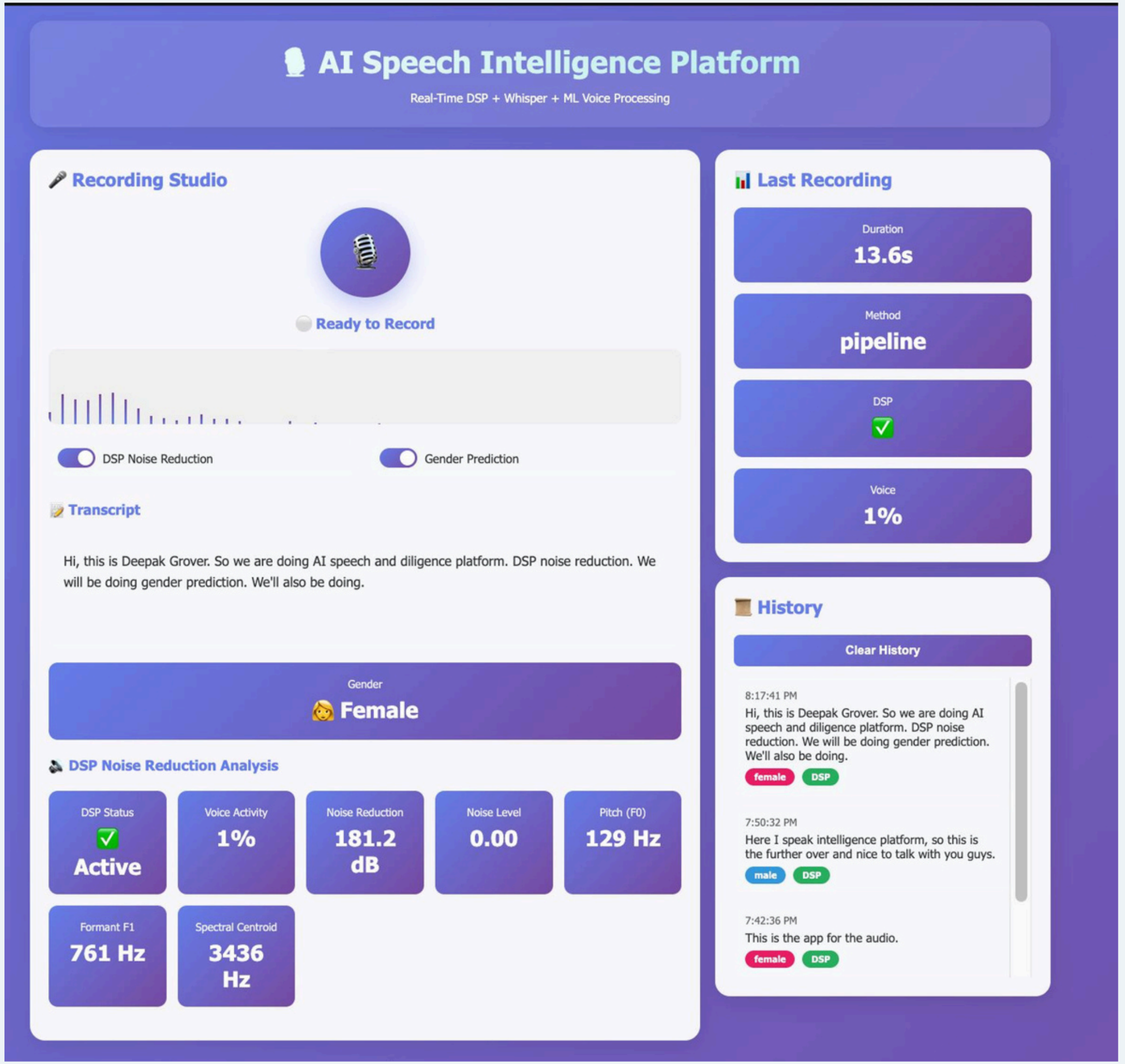
Real-time translation

Converts speech from one language to another instantly, enabling seamless multilingual communication.

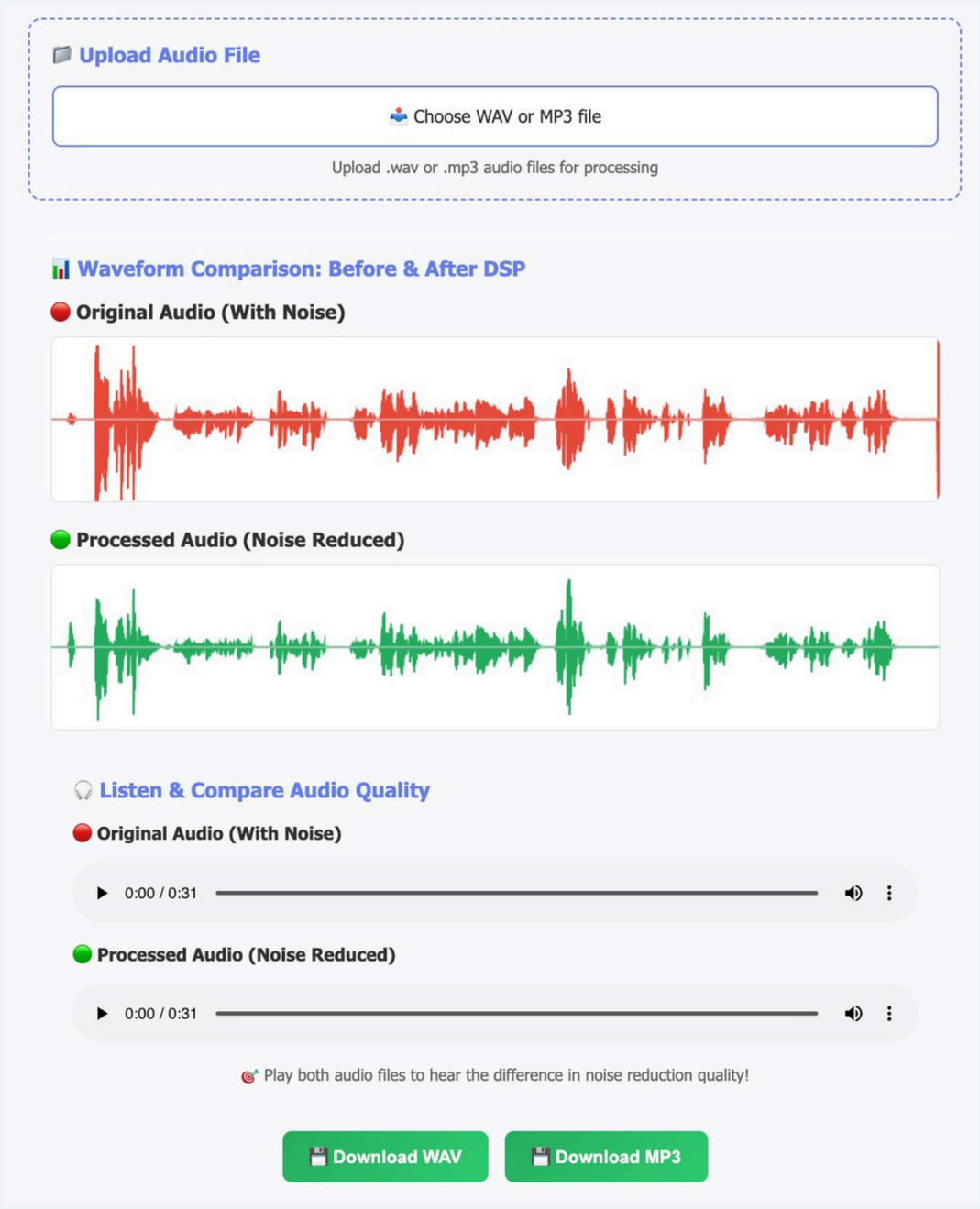
Neural noise suppression (Demucs, RNNoise)

Uses advanced deep-learning models to remove background noise more effectively than classical DSP methods.

DEMO


AI Speech Intelligence Platform
Real-Time DSP + Whisper + ML Voice Processing
Recording Studio
Ready to Record
DSP Noise Reduction
Gender Prediction
Transcript
Hi, this is Deepak Grover. So we are doing AI speech and diligence platform. DSP noise reduction. We will be doing gender prediction. We'll also be doing.
Gender
Female
DSP Noise Reduction Analysis
DSP Status
Active
Voice Activity
1%
Noise Reduction
181.2 dB
Noise Level
0.00
Pitch (F0)
129 Hz
Formant F1
761 Hz
Spectral Centroid
3436 Hz
Last Recording
Duration
13.6s
Method
pipeline
DSP
Voice
1%
History
Clear History
8:17:41 PM
Hi, this is Deepak Grover. So we are doing AI speech and diligence platform. DSP noise reduction. We will be doing gender prediction. We'll also be doing.
female
DSP
7:50:32 PM
Here I speak intelligence platform, so this is the further over and nice to talk with you guys.
male
DSP
7:42:36 PM
This is the app for the audio.
female
DSP

DEMO


Upload Audio File
Choose WAV or MP3 file
Upload .wav or .mp3 audio files for processing
Waveform Comparison: Before & After DSP
Original Audio (With Noise)
Processed Audio (Noise Reduced)
Listen & Compare Audio Quality
Original Audio (With Noise)
0:00 / 0:31
Processed Audio (Noise Reduced)
0:00 / 0:31
Play both audio files to hear the difference in noise reduction quality!
Download WAV
Download MP3