EX NO : 10 Mini Project	Voice and Music Separation from Audio Signal using MATLAB 🎵	Batch No : 02
DATE : 24/04/2025		Roll No : 23ECR117

PROBLEM STATEMENT :

The project aims to implement a MATLAB-based system that can automatically separate **vocals** and **instrumentals** from an input audio file using digital signal processing techniques. The approach focuses on frequency-domain filtering using **IIR bandpass** and **bandstop filters**. The solution also includes visualization of time-domain waveforms, **frequency spectra (FFT)**, and **spectrograms**, as well as a **GUI** to play and interact with the audio components.

AIM :

To develop a MATLAB-based system that separates vocals and instrumentals from an audio file using digital signal processing techniques. The system should provide visual analysis and interactive playback through a GUI.

SOFTWARE REQUIRED :

MATHLAB ONLINE SOFTWARE

PROCEDURE :

1) **Audio Input :**

- Load an audio file using **uigetfile().**
- Convert to **mono** if **stereo** for simpler processing.

2) **Filter Design :**

- Design a **bandpass IIR filter** (300–3400 Hz) for **vocals.**
- Design a **bandstop IIR filter** (300–3400 Hz) to **isolate instrumentals**.

3) **Filtering :**

- Apply filters using **filtfilt()** to prevent phase distortion.
- Generate two separate audio signals : **vocals** and **instrumentals**.

4) **Visualization :**

- Plot time-domain waveforms for **original**, **vocals**, and **instrumentals**.
- Compute and plot frequency spectra using **FFT**.
- Display spectrograms using **spectrogram()** to visualize time-frequency content.
- Use **freqz()** to plot filter frequency responses.

5) **GUI Implementation :**

i) Create a **GUI** with buttons to **:**

- Play original, vocals, and instrumentals.
- Pause/resume playback.

ii) Display status messages to indicate playback state**.**

6) Save Outputs :

- Save the filtered audio signals to **vocals_extracted.wav** and **instrumental_extracted.wav**.

FLOWCHART :

PROGRAM CODE :

The complete **MATLAB** script performs audio loading, filtering, visualization, playback, and saving. It includes:

- **Filter creation** : using **designfilt()**
- **Filtering** : using **filtfilt()**
- **Plotting** : waveforms, FFTs, spectrograms
- **GUI creation** : playback control with pause/resume
- **Saving outputs** : with **audiowrite()**

Full Code :

```
clc;
clear;
global player status_txt Fs;

% ---------- Load Audio File ----------
[file, path] = uigetfile({'*.wav'}, 'Select an audio file');
if isequal(file, 0)
    disp('No file selected.');
    return;
end

[audioIn, Fs] = audioread(fullfile(path, file));
audioIn = mean(audioIn, 2); % Convert to mono if stereo
t = (0:length(audioIn)-1)/Fs;

% ---------- Design Filters ----------
vocalFilter = designfilt('bandpassiir', ...
    'FilterOrder', 6, ...
    'HalfPowerFrequency1', 300, ...
    'HalfPowerFrequency2', 3400, ...
    'SampleRate', Fs);

instrFilter = designfilt('bandstopiir', ...
    'FilterOrder', 6, ...
    'HalfPowerFrequency1', 300, ...
    'HalfPowerFrequency2', 3400, ...
```

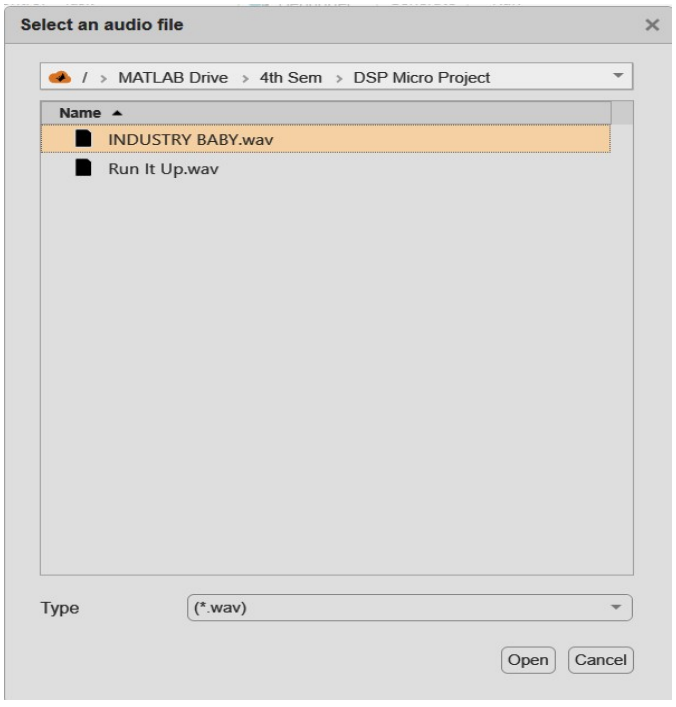
```
    'SampleRate', Fs);

%---------- Apply Filters ----------
vocalPart = filtfilt(vocalFilter, audioIn);
instrumentalPart = filtfilt(instrFilter, audioIn);

% --------- Plot Separate Waveforms with Different Colors ----------
figure('Name', 'Original Audio');
plot(t, audioIn);
title('Original Audio (Stereo)');
xlabel('Time (s)');
ylabel('Amplitude');

figure('Name', 'Vocal Part');
plot(t, vocalPart, 'r');
title('Vocal Part');
xlabel('Time (s)');
ylabel('Amplitude');

figure('Name', 'Instrumental Part');
plot(t, instrumentalPart, 'b');
title('Instrumental Part');
xlabel('Time (s)');
ylabel('Amplitude');

%--------- Plot FFTs ----------
n = length(audioIn);
f = Fs*(0:(n/2))/n;

fft_original = abs(fft(audioIn)/n);
fft_vocal = abs(fft(vocalPart)/n);
fft_instr = abs(fft(instrumentalPart)/n);
figure('Name', 'FFT Spectrum');
plot(f, fft_original(1:n/2+1), 'k');
hold on;
```

```
plot(f, fft_vocal(1:n/2+1), 'r');
plot(f, fft_instr(1:n/2+1), 'b');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
legend('Original', 'Vocals', 'Instrumentals');
title('Frequency Domain Analysis');

% -------- Plot Separate Spectrograms ----------
figure('Name', 'Spectrogram - Original Audio');
spectrogram(audioIn, 256, [], [], Fs, 'yaxis');
title('Original Audio Spectrogram');

figure('Name', 'Spectrogram - Vocal Part');
spectrogram(vocalPart, 256, [], [], Fs, 'yaxis');
title('Vocal Part Spectrogram');

figure('Name', 'Spectrogram - Instrumental Part');
spectrogram(instrumentalPart, 256, [], [], Fs, 'yaxis');
title('Instrumental Part Spectrogram');

% ---------- Plot Filter Responses ----------
figure('Name', 'Bandpass Filter Response (Vocals)');
freqz(vocalFilter, 512, Fs);
title('Vocal Bandpass Filter Response');

figure('Name', 'Bandstop Filter Response (Music)');
freqz(instrFilter, 512, Fs);
title('Instrumental Bandstop Filter Response');

% --------- GUI Playback with Status and Pause/Resume ---------
f_gui = figure('Name','Playback Controls','Position',[300 300 400 250]);
status_txt = uicontrol('Style','text','String','Status: Idle',...
    'Position',[100 200 200 20],'FontSize',10,'ForegroundColor','blue');
```

```
% ------------ Play Function -----------
function playAudio(data, label)
    global player Fs status_txt;

    try
        stop(player);
    catch
    end

    player = audioplayer(data, Fs);
    set(status_txt, 'String', ['Status: Playing - ', label]);
    play(player);
end

% ------- Toggle Pause/Resume Function -------------------
function togglePause()
    global player status_txt;
    if isempty(player)
        return;
    end

    if isplaying(player)
        pause(player);
        set(status_txt, 'String', 'Status: Paused');
    else
        resume(player);
        set(status_txt, 'String', 'Status: Playing');
    end
end
% ------------- GUI Buttons --------------------------
uicontrol('Style','pushbutton','String','Play Original',...
    'Position',[50 150 120 30],...
    'Callback', @(~,~) playAudio(audioIn, 'Original'));

uicontrol('Style','pushbutton','String','Play Vocals',...
```

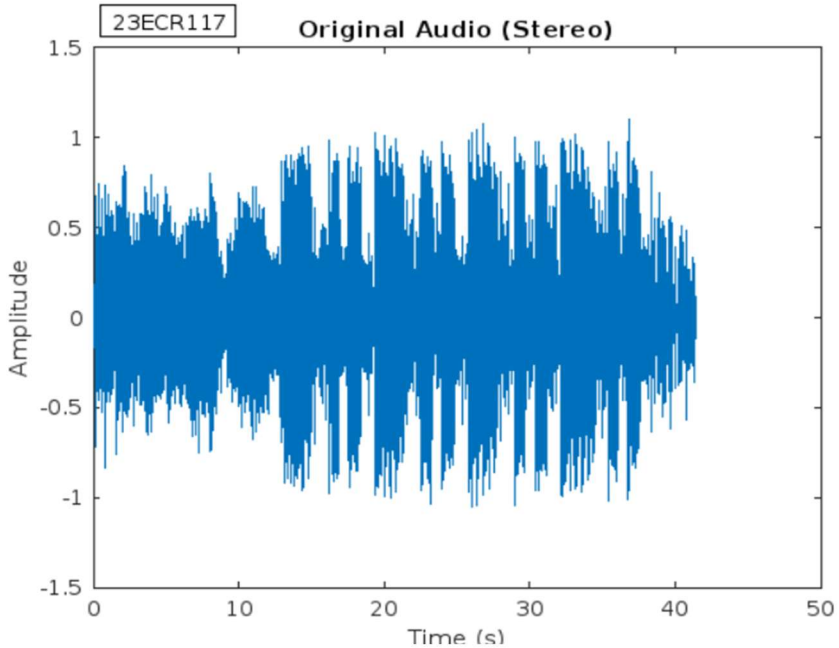
```
    'Position',[230 150 120 30],...
    'Callback', @(~,~) playAudio(vocalPart, 'Vocals'));

uicontrol('Style','pushbutton','String','Play Instrumental',...
    'Position',[50 100 120 30],...
    'Callback', @(~,~) playAudio(instrumentalPart, 'Instrumental'));

uicontrol('Style','pushbutton','String','Pause/Resume',...
    'Position',[230 100 120 30],...
    'Callback', @(~,~) togglePause());

% ------------------- Save Outputs (Optional) ----------------------
audiowrite('vocals_extracted.wav', vocalPart, Fs);
audiowrite('instrumental_extracted.wav', instrumentalPart, Fs);
disp('Audio saved as "vocals_extracted.wav" and "instrumental_extracted.wav"');
```

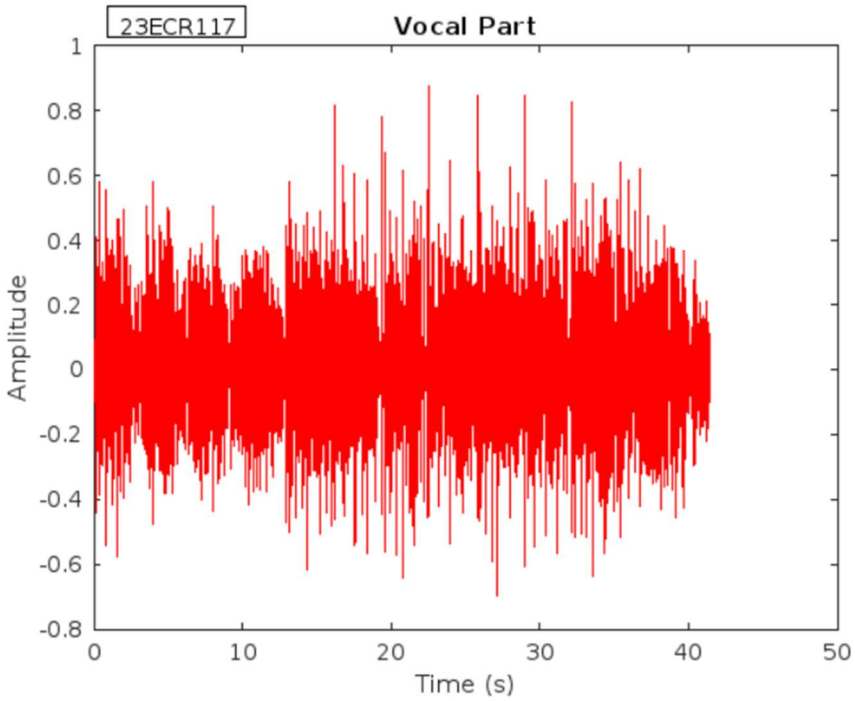
INPUT :

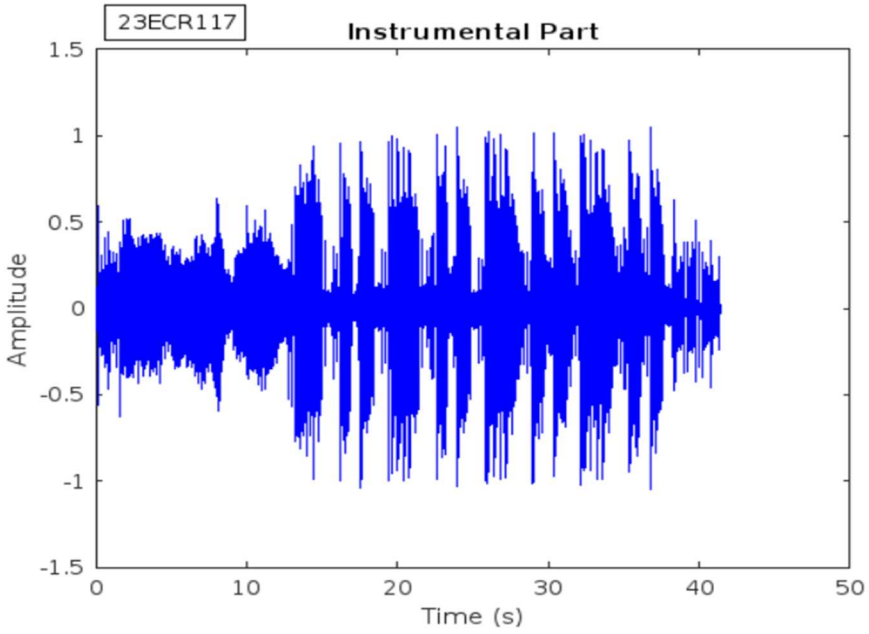


OUTPUT :

1) **Waveform Plots :**
    - Three plots: original, vocal-only (red), and instrumental-only (blue).

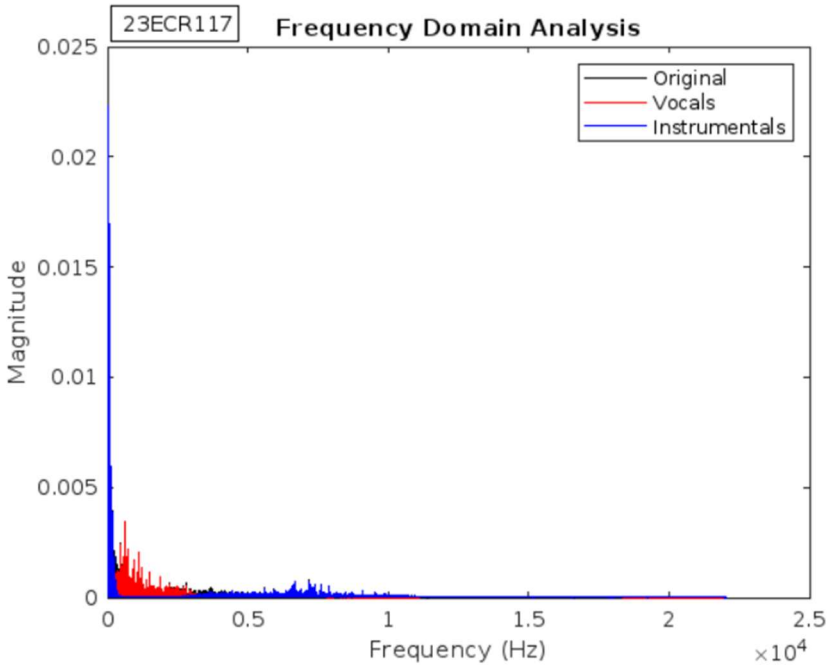

23ECR117
Original Audio (Stereo)
Amplitude
Time (s)


23ECR117
Vocal Part
Amplitude
Time (s)


23ECR117
Instrumental Part
Amplitude
Time (s)

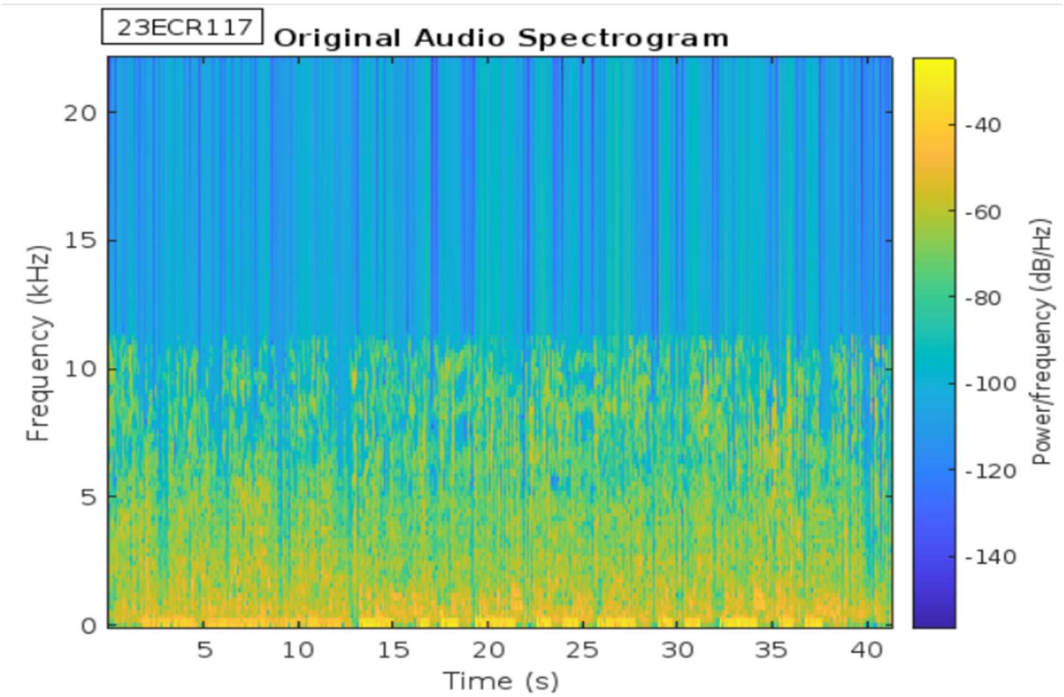
2) FFT Spectrum Plot :

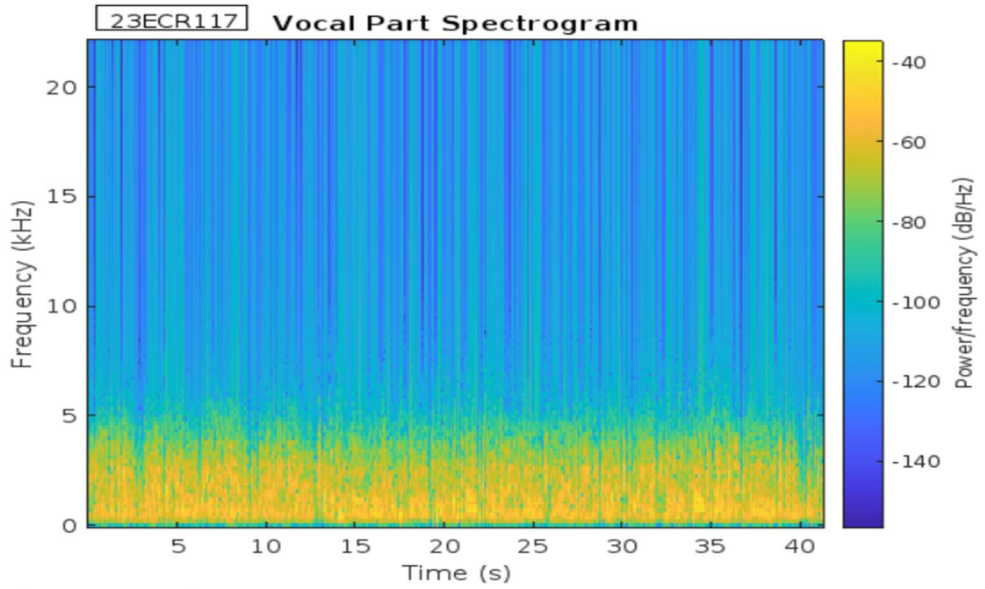
- **Black** : Original audio spectrum.
- **Red** : Vocal frequencies visible in 300–3400 Hz.
- **Blue** : Instrumentals with vocal range suppressed.

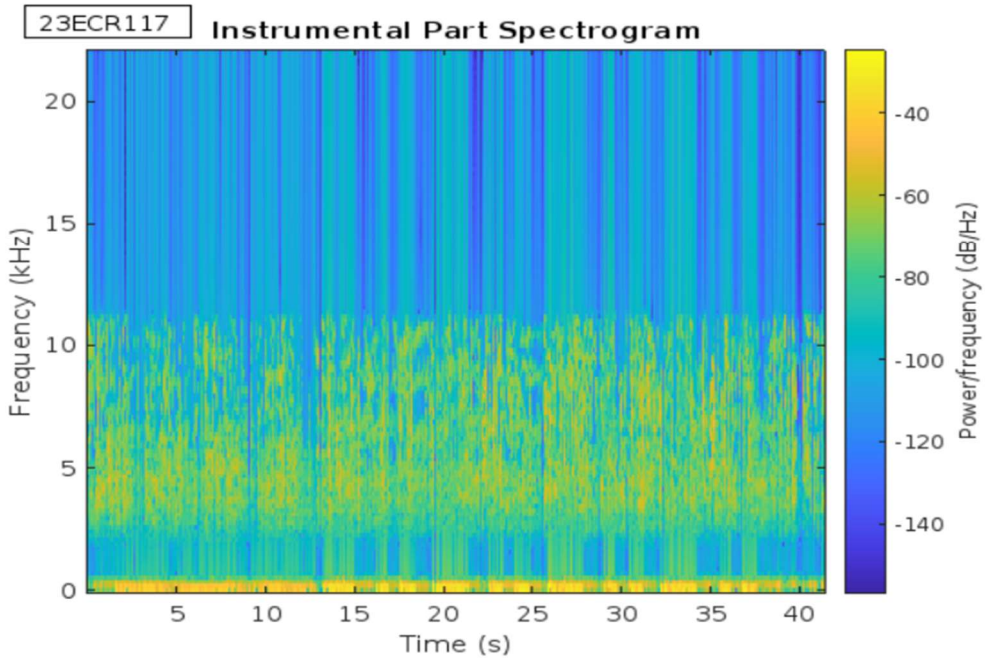


3) Spectrograms :

- **Original** : Full-band time-frequency visualization.
- **Vocals** : Dense vertical patterns around 300–3400 Hz.
- I**nstrumentals** : Broad frequency content with a suppressed vocal band.

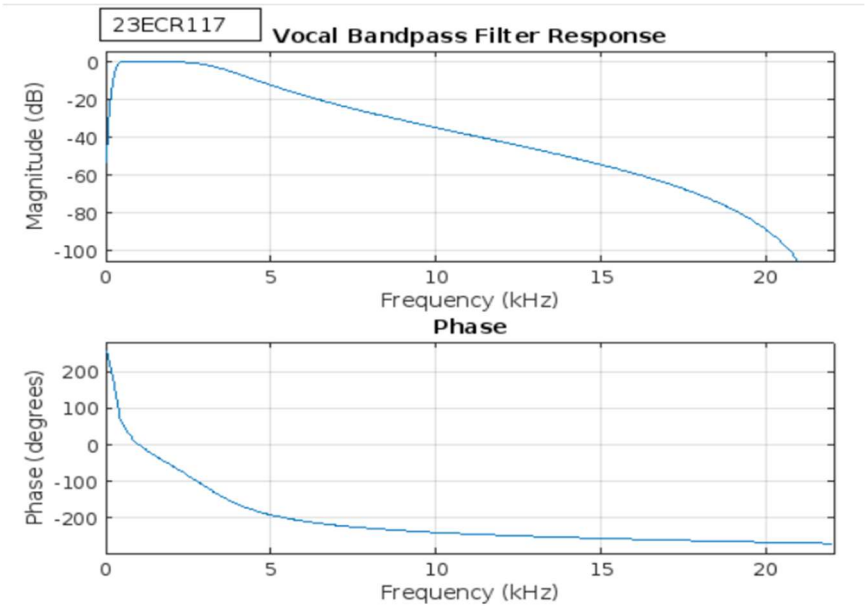


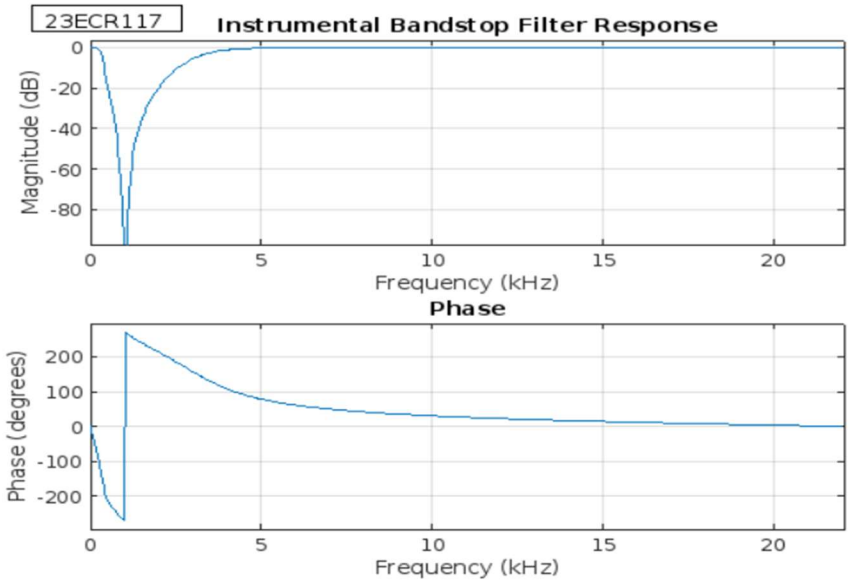




4) **Filter Responses :**

- **Bandpass** : Allows vocal range.
- **Bandstop** : Suppresses vocal range.

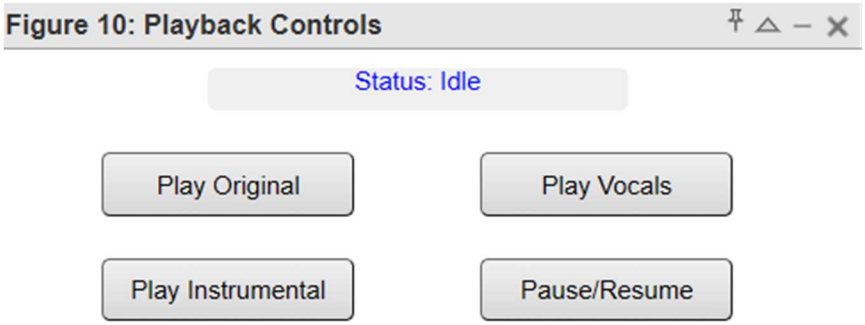




5) **GUI Panel :**

Interactive GUI with :

- Play buttons for all three audio signals.
- Pause/Resume button.
- Playback status display.



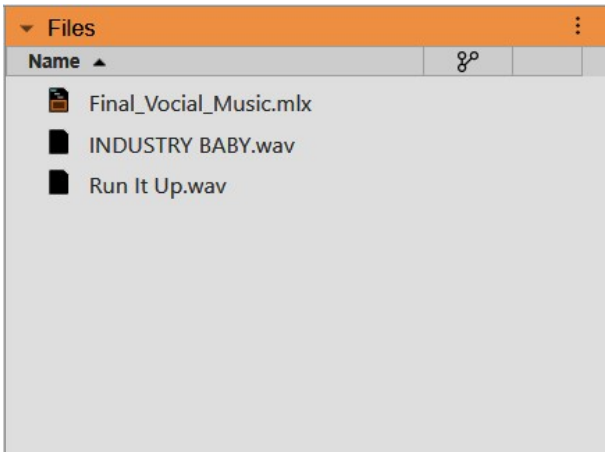
6) **Saved Audio Files:**

- vocals_extracted.wav
- instrumental_extracted.wav

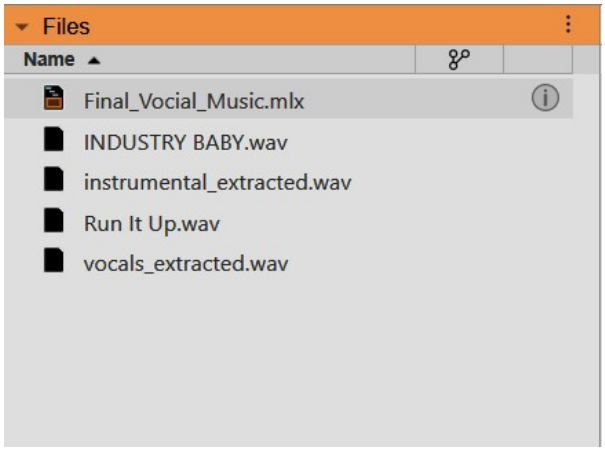
```
Audio saved as "vocals_extracted.wav" and "instrumental_extracted.wav"
```

MatLAB Drive :

Before Runing the code



After Runing the code



GRAPH DESCRIPTION :

1. **Waveforms :**
    - Time-domain signals clearly show different characteristics of **vocals (speech-like, spiky)** and **instrumentals (smooth, repetitive).**
2. **FFT Plot :**
    - **Original** has broad frequency distribution.
    - **Vocals** peak in midrange (speech band).
    - **Instrumentals** show suppressed midrange and stronger low/high frequencies.
3. **Spectrograms :**
    - **Vocals** exhibit harmonically rich vertical streaks.
    - **Instrumentals** are more frequency-diverse and diffuse.
4. **Filter Responses :**
    - **freqz()** confirms **bandpass** sharply focuses on vocal range.
    - **Bandstop** eliminates that same range, keeping the rest intact.

PRACTICAL COMPONENT	MARKS	MARKS AWARDED
Problem Identification	20	
Methodology	20	
Result	10	
Viva	10	
(Total Marks) 60		

CONCLUSION :

This project effectively demonstrates the application of basic **DSP** concepts for audio source separation using **MATLAB**. **Bandpass** and **bandstop IIR filters** successfully **isolate vocal** and **instrumental** elements from an input audio track.

The **GUI** adds interactivity and usability, and the system can be extended further with machine learning or adaptive filtering techniques for enhanced accuracy. Applications include karaoke generation, audio remixing, and preprocessing for speech recognition systems.