



PROJECT NAME: HUMAN SPEECH ANALYSIS

Project Report

Course Name: Signal Processing Lab

Course Title: EEE 4602

Group Members:

<u>Name</u>	<u>ID</u>
Khandaker Adeba Tabassum	200021102
Sadat Al Rashad	200021106
Md Rakibul Islam Rakib	200021107
Md. Nazmul Aman	200021132

Objective

In this project, we created a MATLAB App that will record a voice signal and perform different operations on the signal. It will also calculate certain parameters from the signal and plot/display them in our app. It also allows us to listen to our signal after certain operations.

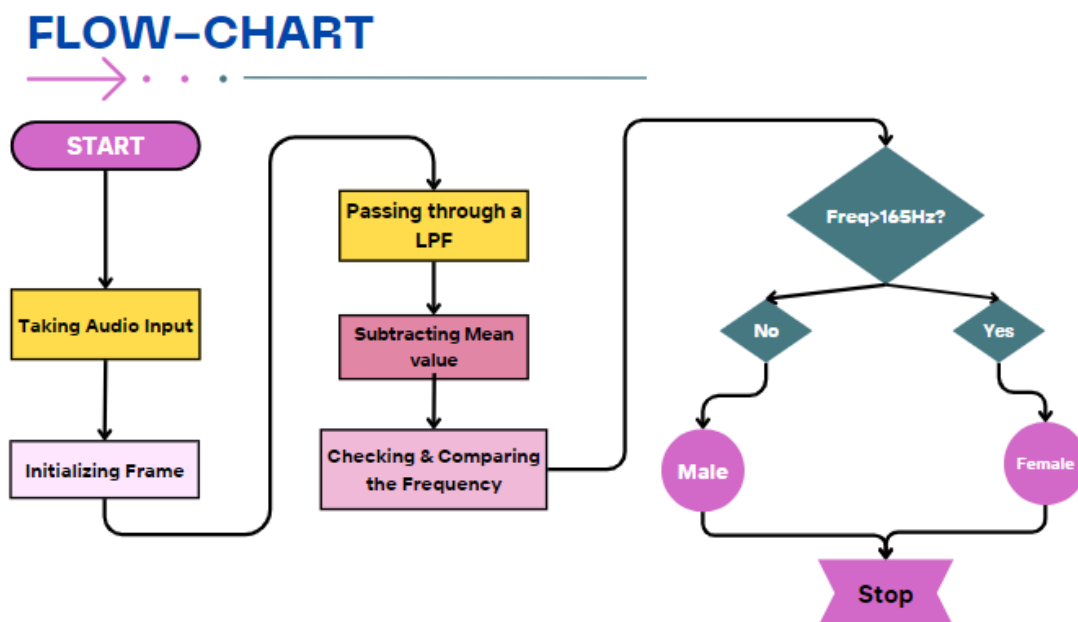
Here is a list of features that are implemented in our app:

- Noise Removal/Filtering
- Silence Removal
- Calculating and Smoothing Power Spectral Density
- Determination of Fundamental Frequency/Pitch
- Harmonic-to-noise Ratio (HNR)
- Determination of Zero-Crossing-Rate (ZCR)
- Spectral Rolloff
- Gender Prediction by Estimation of Fundamental Frequency

Description

It has the advantage of taking inputs in two ways. One is by recording voice for 5 seconds and another by uploading a pre-recorded sample. Once input is given, then it has options for plots that indicate the characteristics of the audio signal.

Gender detection:



To detect gender from audio signals, we follow a specific process. Firstly, we divide the whole signal into frames of equal size. Then, we pass each frame through a low pass filter to reduce noise. To normalize the values in each frame, we subtract the mean value. Next, we compare two adjacent values to check if they have opposite signs or not. This helps us determine the number of zero crossings in each frame. By calculating zero crossings, we can use the formula

$$f_0 = \frac{f_s \times \text{ZCR}}{2}$$

to calculate the pitch. Finally, we take the mean pitch of the entire signal to estimate the fundamental frequency. We have also determined a threshold value for distinguishing between Males and Females. It is 165 Hz. A typical adult male tends to have a fundamental frequency between 90 to 155 Hz while a typical adult female has a fundamental frequency between 165 to 255 Hz.

Code Snippet:

```
frame = 3500;
[b0, a0] = butter(5, 320/(Fs/2), 'low');
F0 = zeros(1, floor(length(y)/frame));
for i = 1:floor(length(y)/frame)
    x = y(1+(i-1)*frame:i*frame);
    xin = filter(b0, a0, abs(x));
    xin = xin - mean(xin);
    x2 = [xin(2:end); 0];
    zc = sum((xin > 0 & x2 < 0) | (xin < 0 & x2 > 0));
    F0(i) = 0.5 * Fs * zc / length(x);
end
Fx = mean(F0);
if Fx < 85
    fprintf('Not a Voice\n');
else
    if Fx > 165
        fprintf('Female Voice\n');
    else
        fprintf('Male Voice\n');
    end
end
plot(F0)
```

Silence Removal

We divide the signal into frames and then normalize the values. After that, we find the maximum value of each frame and compare it with a threshold of 0.05. If the amplitude of a frame is less than the threshold, we consider it as a 'frame of silence' and discard it. This way, we eliminate low-amplitude components and reshape the signal using the reshape function, considering only the high-amplitude components.

Code Snippet:

```
maxval=max(y(1:end,1));  
data = y(1:end,1)./maxval;  
% do framing  
f_d = 0.025;  
frame_size = round(f_d * Fs);  
n_f = floor(length(data)/frame_size); %no. of frames  
temp = 0;  
for i = 1 : n_f  
frames(i,:) = data(temp + 1 : temp + frame_size);  
temp = temp + frame_size;  
end  
% silence removal based on max amplitude  
m_amp = abs(max(frames,[],2)); % find maximum of each frame  
id = find(m_amp > 0.05); % finding ID of frames with max amp > 0.03  
fr_ws = frames(id,:); % frames without silence  
% reconstruct signal  
y1 = reshape(fr_ws',1,[]);  
y1=y1.*maxval;
```

Noise removal

We designed a low-pass filter with a cutoff frequency of 3000 Hz, applied it to the original signal, and plotted the noise-free signal. The cutoff frequency was chosen through trial and error.

Code Snippet:

```
[b, a] = butter(5, 3000/(Fs/2), 'low');  
y2 = filter(b,a,y);
```

Power Spectral Density

There are two types of Power Spectral Density (PSD): Generalized PSD and Welch PSD. Generalized PSD tends to have a lot of jitters in it, whereas Welch PSD generates a much

smoother graph. To achieve this, a window is taken and a Hann taper is used to attenuate the amplitude of the first and last samples of a window. With the help of NFFT (Number of Fast Fourier transforms), Welch PSD generates a much smoother graph than Generalized PSD.

Code Snippet:

```
h = spectrum.welch; % create welch spectrum object
d = psd(h, y, 'Fs', Fs);
d1= psd(h, y2, 'Fs', Fs);
l = length(y);
NFFT = 2^nextpow2(l);
xf = abs(fft(y, NFFT));
xf1= abs(fft(y2,2^nextpow2(length(y2))));
figure;
subplot(2,2,1)
hold on
plot(d);
plot(d1);
hold off
legend(["Before Filtering","After Filtering"])
Hpsd = dspdata.psd(xf(1:length(xf)/2), 'Fs', Fs);
Hpsd2= dspdata.psd(xf1(1:length(xf1)/2), 'Fs', Fs);
subplot(2,2,2)
plot(Hpsd);
subplot(2,2,[3,4])
plot(Hpsd2);
```

Zero Crossing Rate

The code counts the number of zero crossings (changes in sign) between consecutive samples for each frame. It does so by iterating over the samples. The total number of zero crossings is then divided by twice the length of the frame to get the zero-crossing rate, which is then plotted. To calculate the mean zero-crossing rate, the mean of all zero-crossing rates computed for individual frames is taken. The formula for calculating the zero-crossing rate is given by

$$ZCR = \frac{1}{N-1} \sum_{n=1}^{N-1} |\text{sign}(x[n]) - \text{sign}(x[n-1])|$$

Code Snippet:

```

f_d = 0.025;
frame_size = round(f_d * Fs);
n_f = floor(length(y)/frame_size); %no. of frames
temp = 0;
for i = 1 : n_f
frames(i,:) = y(temp + 1 : temp + frame_size);
temp = temp + frame_size;
end
zcr=zeros(1,size(frames,1));
for i= 1:size(frames,1)
    for j=1:size(frames,2)-1
        zcr(i)=zcr(i)+abs(sign(frames(i,j+1))-sign(frames(i,j)));
    end
    zcr(i)=zcr(i)/(2*length(frames));
end
figure;
plot(zcr)

```

The Harmonics-to-Noise Ratio (HNR)

It is a measure used in speech and audio signal processing to quantify the ratio between harmonics (periodic components) and noise (non-periodic components) in a signal. To calculate HNR, we divided the signal into segments, computed the autocorrelation of each segment to find the fundamental frequency, separated harmonic and noise components, and then calculated the HNR for each segment. Finally, the mean HNR was computed across all segments. The formula for calculating HNR is given by:

$$HNR = 10 \cdot \log_{10} \left(\frac{\sum_{f \in \text{harmonics}} P(f)}{\sum_{f \in \text{noise}} P(f)} \right)$$

Code Snippet:

```

segment_length = 500;
num_segments = floor(length(y) / segment_length);
HNR = zeros(1, num_segments);
for i = 1:num_segments
    segment = y((i-1)*segment_length+1 : i*segment_length);
    [autocorr, lags] = xcorr(segment, 'coeff');

```

```

[~,I] = max(autocorr(lags>0));
lag = lags(I);
T0 = lag / Fs;
t = (0:length(segment)-1) / Fs;
harmonic = cos(2*pi*t/T0);
E_harmonic = sum((segment .* harmonic).^2);
E_noise = sum((segment .* (1-harmonic)).^2);
HNR(i) = 10 * log10(E_harmonic / E_noise);
end
mean_hnr=mean(HNR)

```

Spectral Roll-Off

The Spectral Roll-Off is used to characterize the distribution of signal energy across different frequency bands. It signifies the frequency below which a certain amount of energy lies. We used the MATLAB function 'spectralRolloffPoint', which computes the roll-off point based on the signal's Power Spectral Density (PSD).

Code Snippet:

```

r1 = spectralRolloffPoint(y,Fs);
t1 = linspace(0,size(y,1)/Fs,size(r1,1));
figure;
plot(t1,r1)

```

Spectrum of the signal

To compute and plot the frequency component of the audio signal, we performed a fast Fourier transform on both the original signal and the noisy signal. Here NFFT was calculated to determine the number of points used in the FFT. NFFT fastens the Fourier transform algorithm.

Code Snippet:

```

l = length(y2);
NFFT = 2^nextpow2(l);
f = Fs/2*linspace(0,1,NFFT/2+1);
xf = abs(fft(y, NFFT));
xf2 = abs(fft(y2, NFFT));
plot(10^(-3)*f, xf(1:NFFT/2+1),'g', 10^(-3)*f, xf2(1:NFFT/2+1), 'r')
legend('Before Filtering','After Filtering');

```

Limitations

Accuracy: When real-time voice recording is on, accuracy can be influenced by factors like background noise, speaker accents, and emotional states. Our project has a generalized fundamental frequency threshold to distinguish between males and females, which may not be true for all cases as age, ethnicity, and medical conditions can lead to variations in pitch.

Algorithm: In this project, we have used a simple low-pass filter to reduce noise. That's why it can't entirely eliminate the noise which sometimes results in wrong estimations, especially in terms of gender classification.

Future Improvements

Better Noise-Removal: We want to implement a much more sophisticated algorithm to remove noise. It can be done by further studying different noise reduction algorithms and exploring different Matlab Functions.

Advanced Feature Extractions: We want to extract much more advanced features from the voice signal like Formant Frequencies, Mel-Frequency Cepstral Coefficients (MFCCs), Voiced and Unvoiced segments, Spectral Centroid, etc.

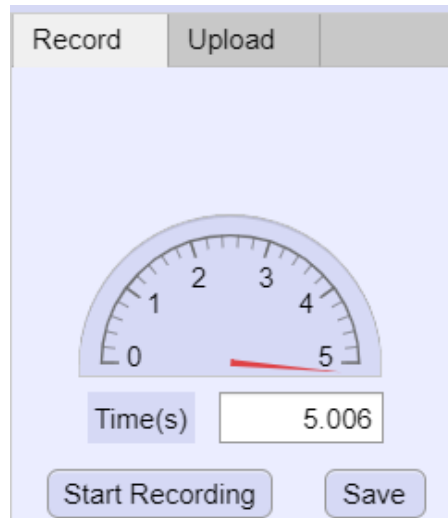
Deep Learning Integration: We want to integrate deep learning techniques to predict gender more accurately by analyzing more features. Integration of Neural Networks will also allow us to implement speaker recognition as well as predict different features such as speaker recognition, emotional state, age, different medical conditions, etc.

ILLUSTRATION OF THE PROJECT

Input:

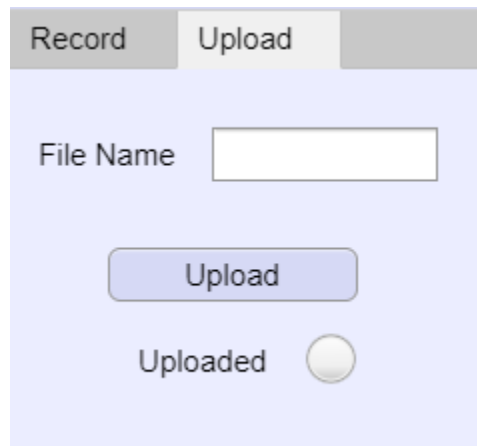
Input can be given in 2 ways:

1. Taking real-time voice records from the user for 5 seconds, which can be saved further as well.



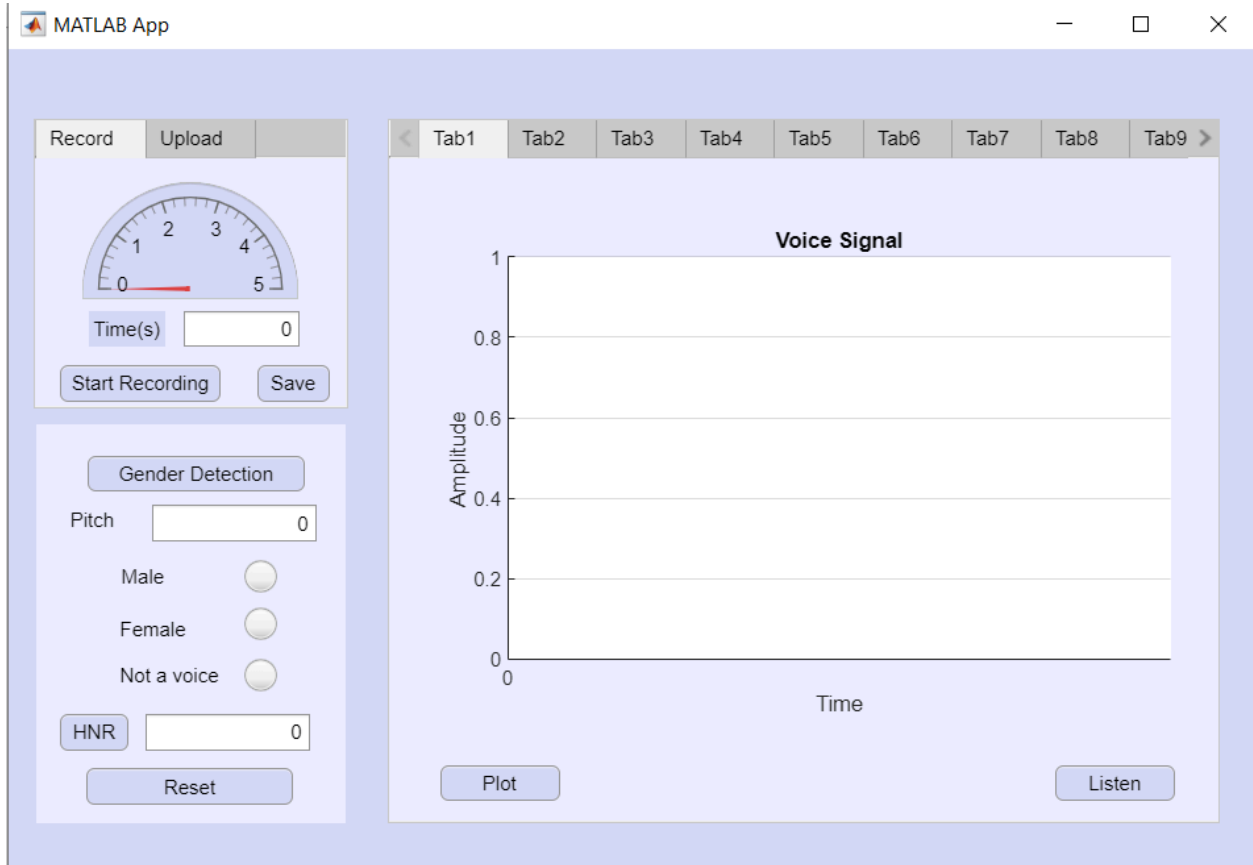
The screenshot shows the 'Record' tab of a web application. At the top, there are three tabs: 'Record' (active), 'Upload', and an unlabeled tab. Below the tabs is a large light blue area containing a semi-circular timer with a red needle pointing to 5. Below the timer is a text input field labeled 'Time(s)' containing the value '5.006'. At the bottom of the area are two buttons: 'Start Recording' and 'Save'.

2. Audio file upload



The screenshot shows the 'Upload' tab of the same web application. At the top, there are three tabs: 'Record', 'Upload' (active), and an unlabeled tab. Below the tabs is a large light blue area. It contains a text input field labeled 'File Name'. Below the input field is an 'Upload' button. At the bottom of the area is the text 'Uploaded' followed by a small circular progress indicator.

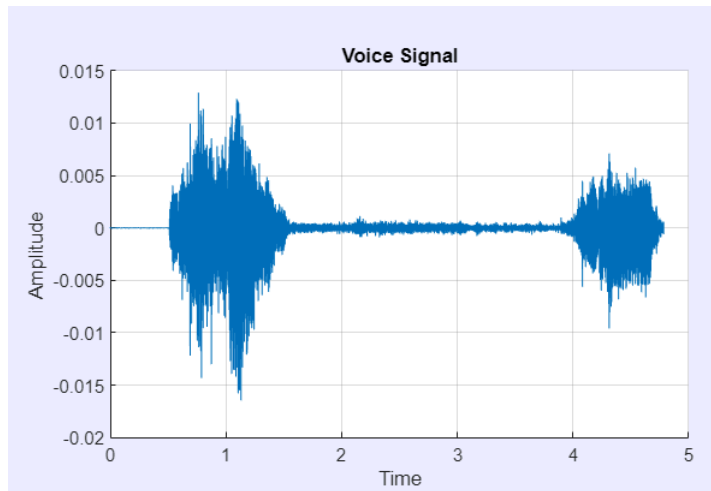
User Interface:



Results:

For Female Users:

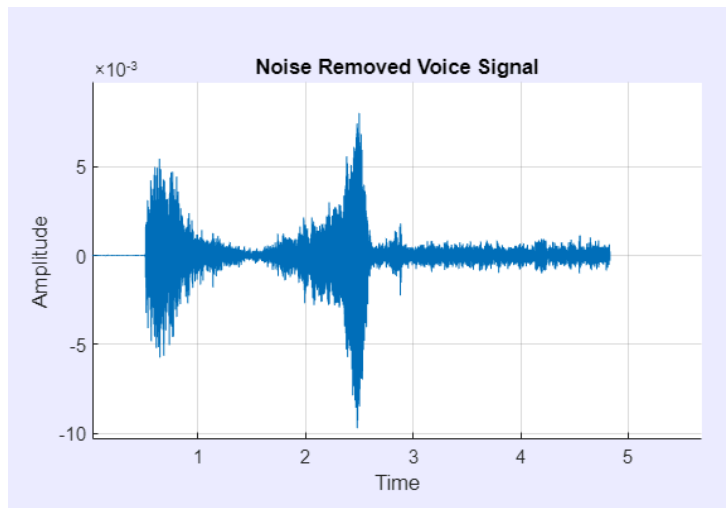
Recorded voice signal(Amplitude vs time):



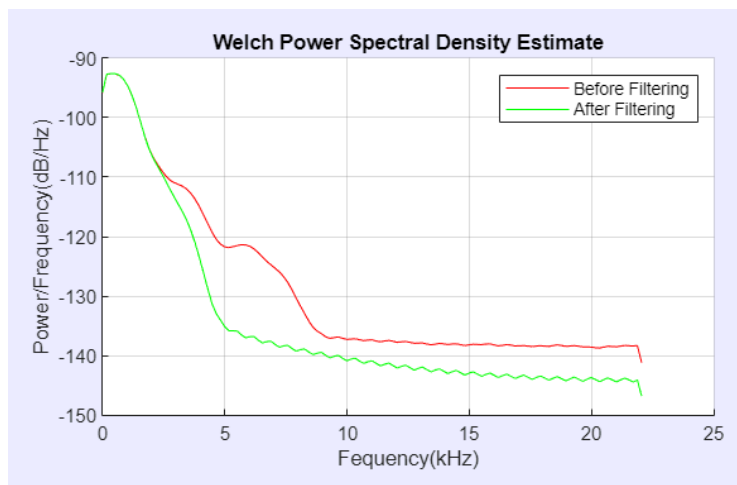
After removing silence (Amplitude vs time):



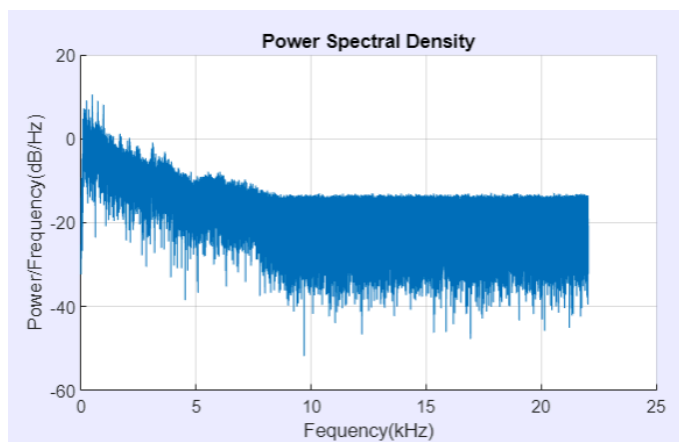
Noise removed voice signal (Amplitude vs time):



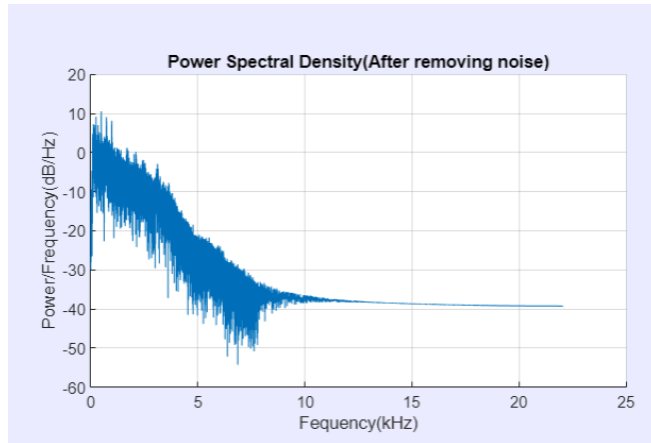
Welch Power Spectral Density Estimate:



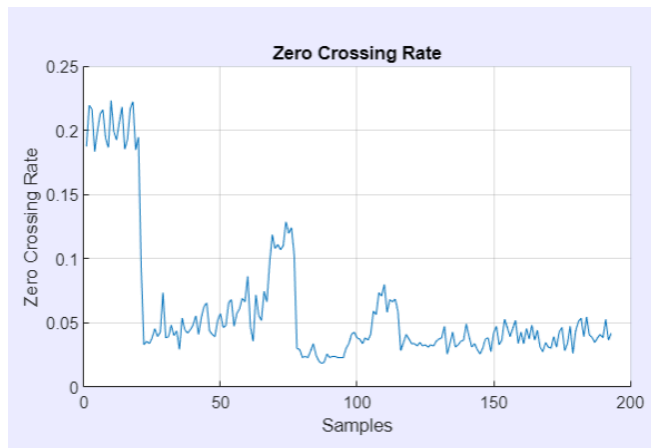
Power Spectral Density:



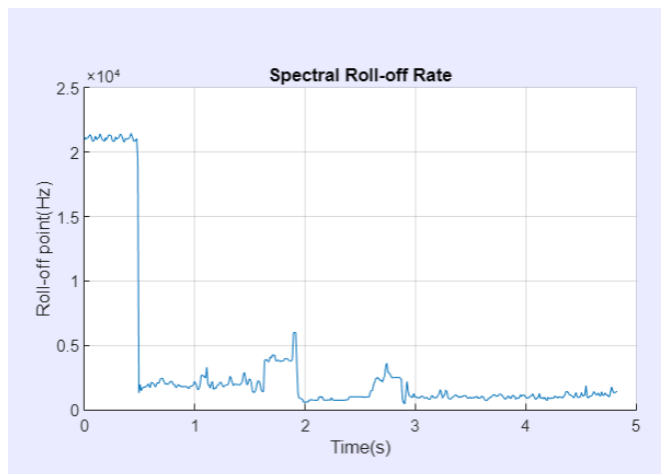
Power Spectral Density after removing noise:



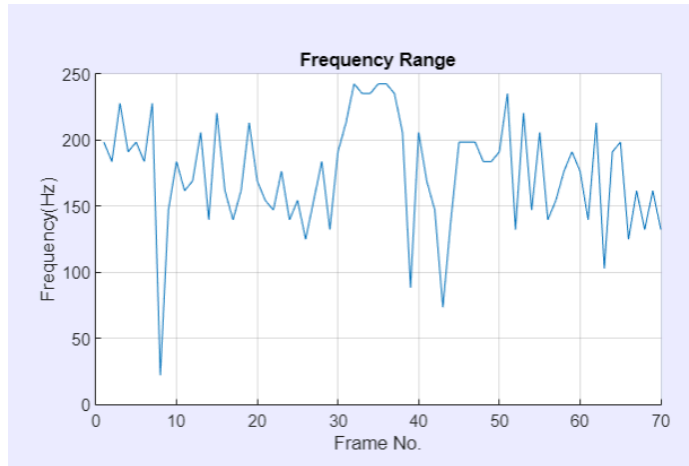
Zero Crossing Rate:



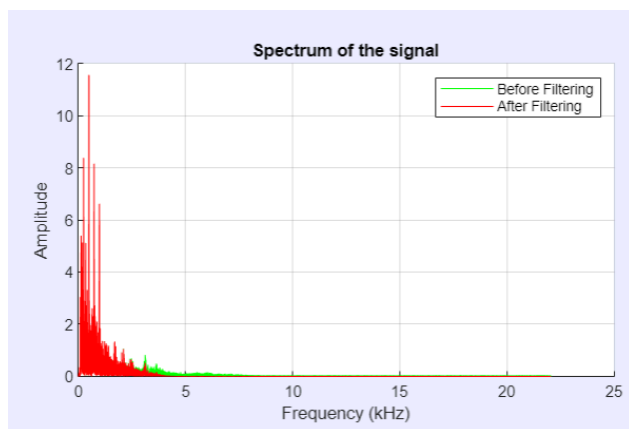
Spectral Rolloff:



Frequency Range:



The spectrum of the signal:



HNR:

HNR

-6.992

Gender Detection:

Gender Detection

Pitch

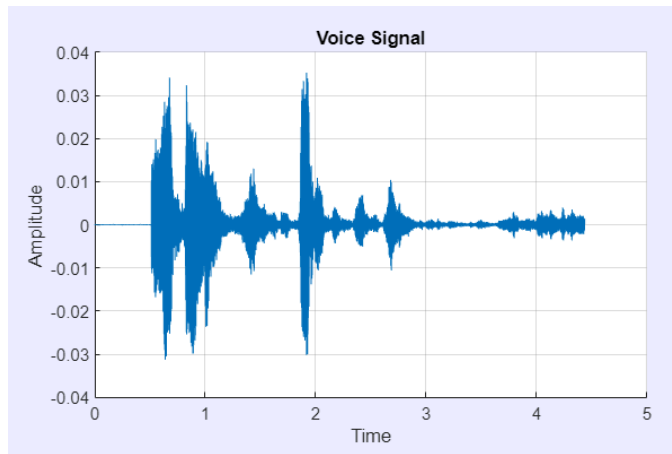
Male ☐

Female ☒

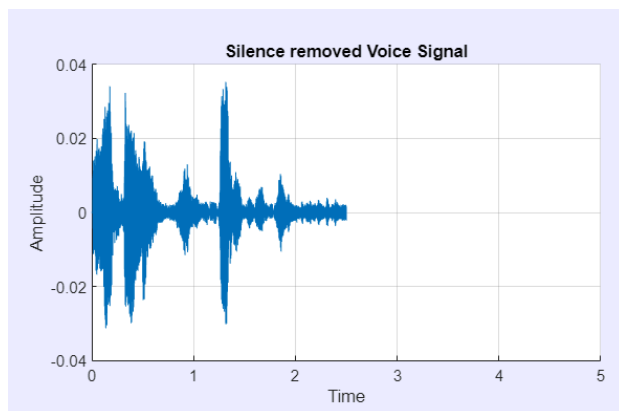
Not a voice ☐

For Male Users

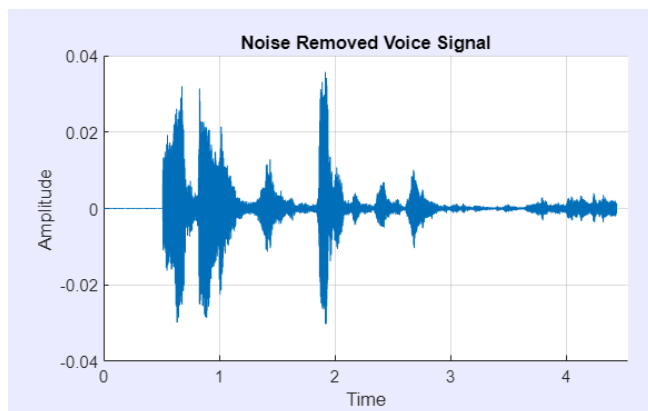
Recorded voice signal(Amplitude vs time):



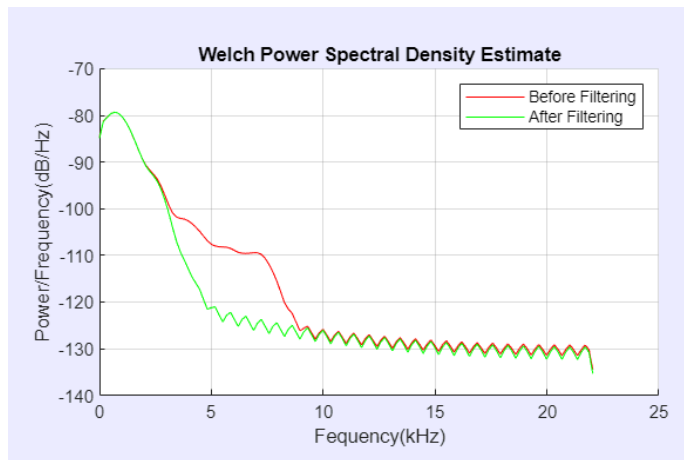
After removing silence (Amplitude vs time):



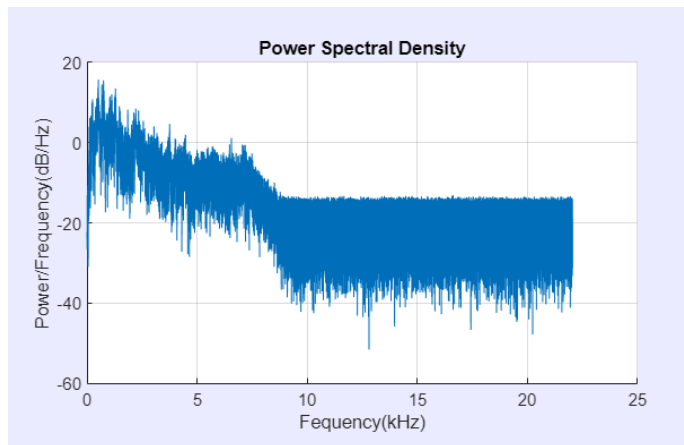
Noise removed voice signal(Amplitude vs time):



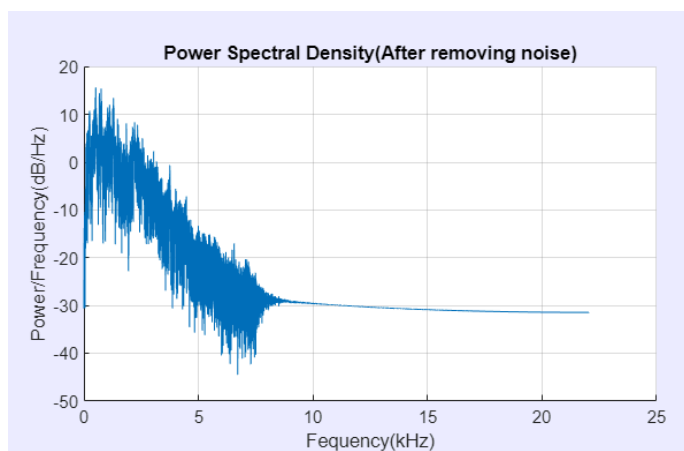
Welch Power Spectral Density Estimate:



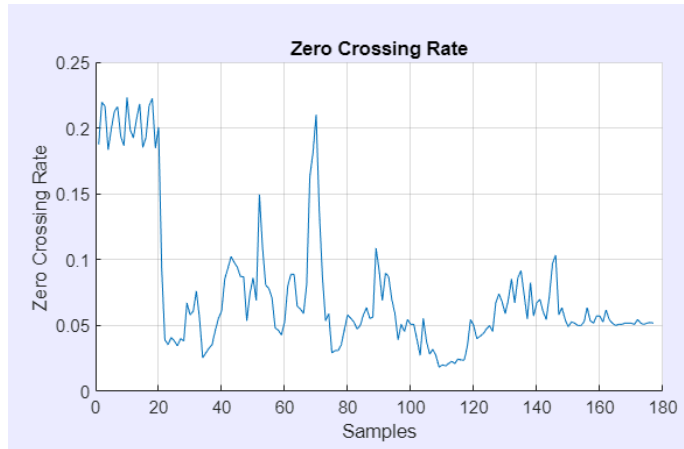
Power Spectral Density:



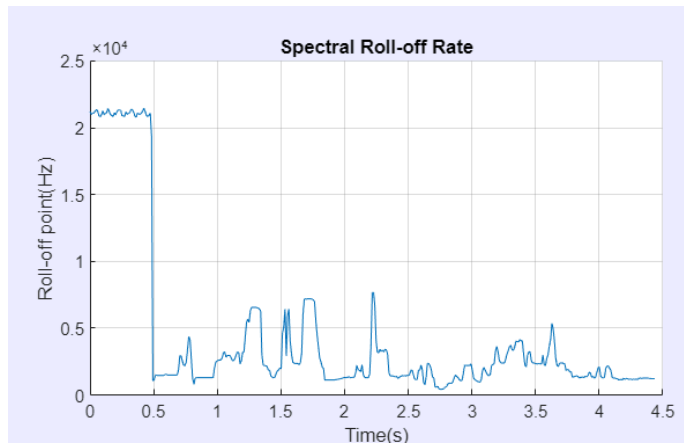
Power Spectral Density after removing noise:



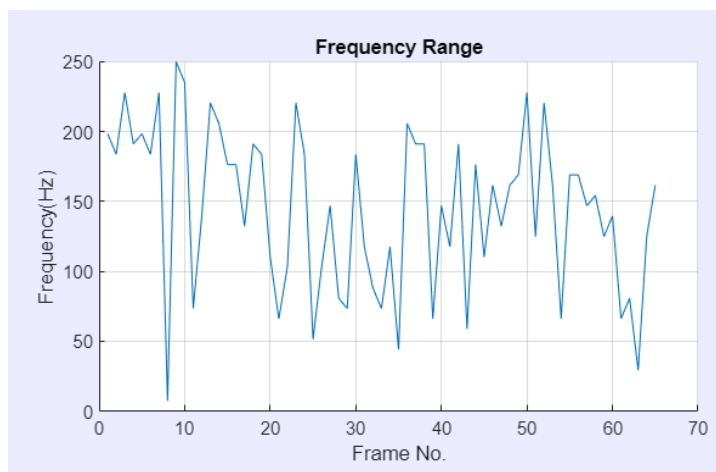
Zero Crossing Rate:



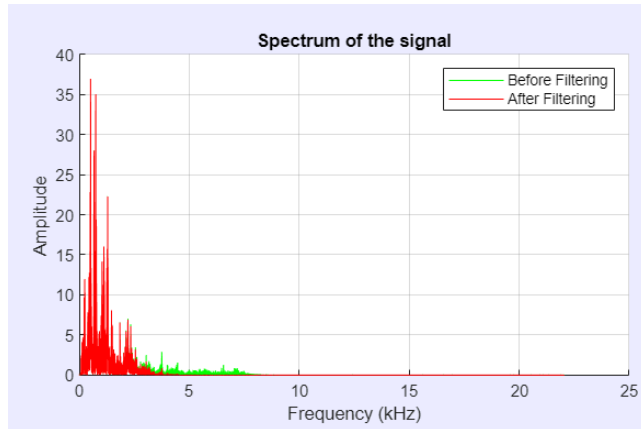
Spectral Rolloff:



Frequency Range:



The spectrum of the signal:



HNR:

Gender Detection:

Gender Detection

Pitch

Male ☒

Female ☐

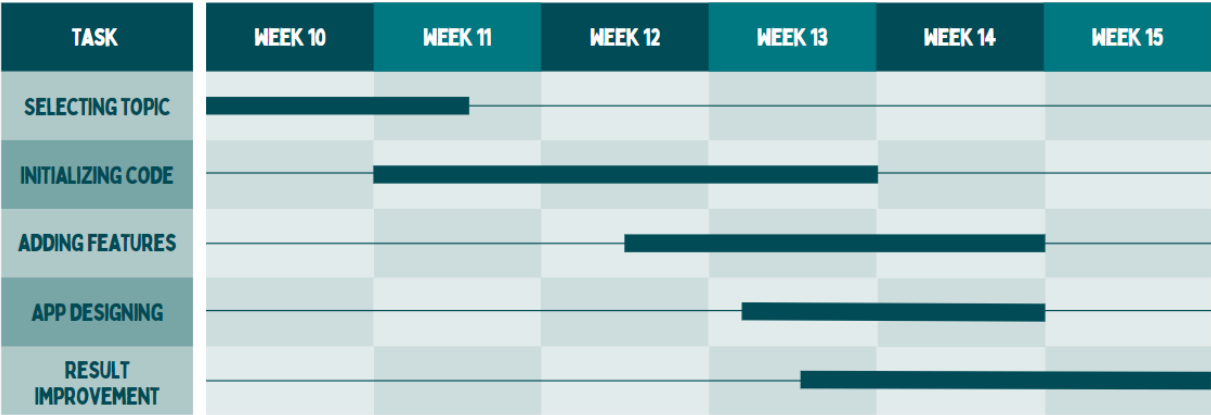
Not a voice ☐

Gender Detection Results:

Subject No.	Subject Gender	Estimated Pitch (Hz)	Predicted Gender
1	Female	117.5	Male
2	Female	168.3	Female
3	Female	166.8	Female
4	Female	125.4	Male
5	Male	144.9	Male
6	Male	151.6	Male
7	Male	148.9	Male
8	Male	109.3	Male
9	Female	174.7	Female

Gantt Chart:

GANTT CHART



Contribution:

<u>Name</u>	<u>ID</u>	Contribution
Khandaker Adeba Tabassum	200021102	Code Implementation, Report
Sadat Al Rashad	200021106	Code Implementation, Troubleshooting, Report
Md Rakibul Islam Rakib	200021107	Designing GUI, Presentation, Report
Md. Nazmul Aman	200021132	Code integration with GUI, Presentation