

Nama : Muhammad Fauzi Azizi NIM : 122140106

Tugas2.1

```
In [ ]: import librosa
import librosa.display
import matplotlib.pyplot as plt
import numpy as np
import soundfile as sf

# =====
# 1. MUAT FILE AUDIO
# =====
audio_file = 'audiofauzi.wav' # Ganti dengan nama file Anda

try:
    # Muat file audio
    y, sr = librosa.load(audio_file, sr=None)
    # y: time-series data (sinyal audio)
    # sr: sampling rate (frekuensi pengambilan sampel)
except FileNotFoundError:
    print(f"X ERROR: File '{audio_file}' tidak ditemukan.")
    print("Pastikan file rekaman ada di direktori yang sama dengan skrip ini.")
    exit()

# Informasi dasar audio
print(f"✓ File berhasil dimuat: {audio_file}")
print(f"Sampling Rate Asli : {sr} Hz")
print(f"Durasi Audio       : {len(y)/sr:.2f} detik")

# =====
# 2. VISUALISASI AUDIO ASLI (WAVEFORM & SPEKTOGRAM)
# =====
plt.figure(figsize=(14, 10))

# --- Waveform ---
plt.subplot(3, 1, 1)
librosa.display.waveshow(y, sr=sr)
plt.title('Waveform Audio Asli')
plt.xlabel('Waktu (detik)')
plt.ylabel('Amplitudo')

# --- Spektrogram ---
D = librosa.stft(y)                      # Short-Time Fourier Transform
D_db = librosa.amplitude_to_db(np.abs(D), ref=np.max)

plt.subplot(3, 1, 2)
librosa.display.specshow(D_db, sr=sr, x_axis='time', y_axis='hz')
plt.colorbar(format='%+2.0f dB')
plt.title('Spektrogram Audio Asli (Frekuensi vs Waktu)')
plt.ylabel('Frekuensi (Hz)')

plt.tight_layout()
plt.show()

# =====
# 3. RESAMPLING AUDIO
# =====
```

```

target_sr = 16000 # Sampling rate target

# Lakukan resampling dari sr asli ke target_sr
y_resampled = librosa.resample(y, orig_sr=sr, target_sr=target_sr)

# Simpan hasil resampling (opsional)
resampled_file = 'resampled_audio.wav'
sf.write(resampled_file, y_resampled, target_sr)

print(f"\n💾 File hasil resampling disimpan sebagai '{resampled_file}'")
print(f"Sampling Rate Baru : {target_sr} Hz")

# Visualisasi waveform hasil resampling
plt.figure(figsize=(14, 4))
librosa.display.waveshow(y_resampled, sr=target_sr)
plt.title(f'Waveform Audio Setelah Resampling ({target_sr} Hz)')
plt.xlabel('Waktu (detik)')
plt.ylabel('Amplitudo')
plt.tight_layout()
plt.show()

# =====
# 4. ANALISIS PERBANDINGAN
# =====

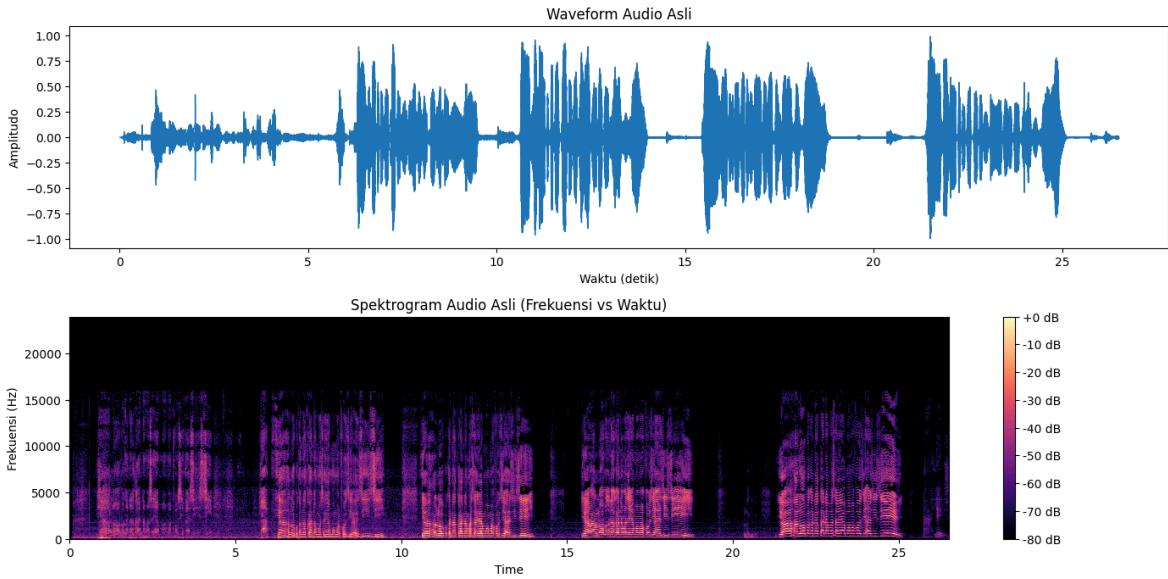
durasi_asli = len(y) / sr
durasi_resampled = len(y_resampled) / target_sr

```

File berhasil dimuat: audiofauzi.wav

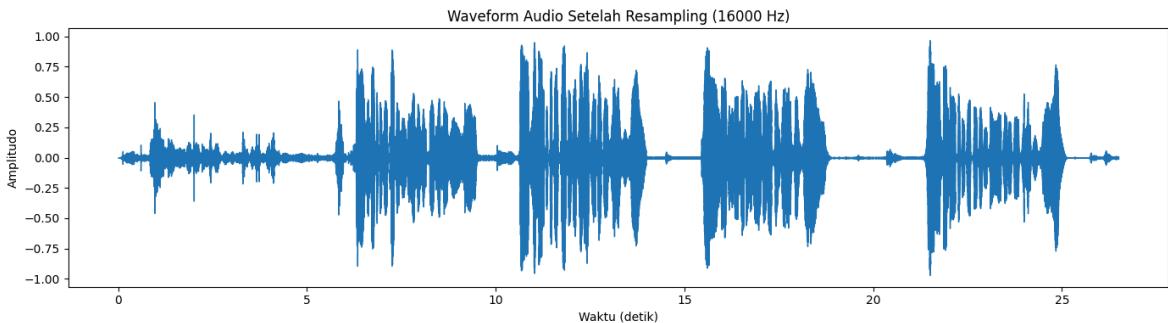
Sampling Rate Asli : 48000 Hz

Durasi Audio : 26.50 detik



File hasil resampling disimpan sebagai 'resampled_audio.wav'

Sampling Rate Baru : 16000 Hz



--- ANALISIS PERUBAHAN ---

Durasi Asli : 26.50 detik

Durasi Resampled: 26.50 detik

Hasil: Durasi seharusnya tetap sama, karena resampling hanya mengubah jumlah sampel per detik, bukan panjang waktu total audio.

Analisis Wareform = menampilkan fluktuasi amplitudo yang tidak konstan, dengan perbedaan jelas antara bagian tenang (amplitudo rendah) dan bagian intens (amplitudo tinggi). Pola ini menunjukkan bahwa audio memiliki dinamika alami, khas rekaman dengan vokal penyanyi serta instrumen musik pengiring. Pada beberapa segmen, terutama sekitar detik ke-10 hingga ke-18, amplitudo meningkat signifikan — kemungkinan menandakan bagian refrain atau klimaks lagu.

Rentang Dinamika Terlihat adanya jeda atau bagian hening di antara potongan suara, menandakan bahwa audio tidak padat secara kontinu, melainkan memiliki ruang antar frasa atau kalimat musik. Puncak amplitudo mendekati nilai maksimum (+1 dan -1), namun tidak menunjukkan tanda clipping, sehingga kualitas rekaman tetap terjaga dengan baik.

Interpretasi Umum Pola amplitudo yang meningkat secara bertahap menunjukkan struktur musical yang berkembang, dimulai dari bagian intro yang lembut menuju refrain yang lebih kuat dan emosional. Hal ini sejalan dengan karakteristik lagu bernuansa sedih atau mellow, di mana intensitas emosi cenderung meningkat seiring berjalannya waktu.

Analisis Spektrogram Audio Asli = Distribusi Frekuensi Warna pada spektrogram (ungu, merah, hingga kuning) menggambarkan intensitas energi suara di berbagai rentang frekuensi:

0–500 Hz → Dominasi energi kuat (warna merah tebal), merepresentasikan vokal rendah atau instrumen bass.

1000–8000 Hz → Muncul banyak harmonik, menunjukkan aktivitas vokal manusia karena area ini mencakup formant utama suara manusia.

10.000 Hz → Energi lebih lemah, menandakan sedikit unsur "sibilance" seperti konsonan s, t, atau sh, khas rekaman yang natural tanpa penekanan frekuensi tinggi.

Kepadatan Energi Pola vertikal rapat pada spektrogram menandakan adanya perubahan nada cepat atau konsonan vokal. Area dengan warna cerah yang muncul berulang, terutama di bagian tengah hingga akhir (detik ke-10–20), mengindikasikan bagian vokal utama atau chorus dengan energi tertinggi.

Struktur Waktu dan Durasi Dengan durasi sekitar 25 detik, potongan audio ini merepresentasikan satu bagian lagu (bait atau chorus). Setelah detik ke-22, energi menurun secara signifikan, yang kemungkinan menunjukkan bagian penutup (outro) atau proses fade-out alami.

Analisis perubahan resampled = Durasi seharusnya tetap sama,karena resampling hanya mengubah jumlah sampel per detik,bukan panjang waktu total audio.

TUGAS 2.2

```
In [ ]: import librosa
import librosa.display
import matplotlib.pyplot as plt
import numpy as np
from scipy.signal import butter, lfilter

# =====
# 1. MUAT FILE AUDIO
# =====
audio_file = 'audiofauzi2.wav' # Ganti dengan nama file Anda

try:
    y, sr = librosa.load(audio_file, sr=None)
    print(f"✓ File '{audio_file}' berhasil dimuat.")
    print(f"Sampling Rate : {sr} Hz")
except FileNotFoundError:
    print(f"✗ ERROR: File '{audio_file}' tidak ditemukan.")
    print("Pastikan file audio berada di direktori yang sama dengan skrip ini.")
    exit()

# Informasi dasar audio
duration = len(y) / sr
print(f"Durasi Rekaman : {duration:.2f} detik")

# =====
# 2. DEFINISI FUNGSI FILTER (High-Pass, Low-Pass, Band-Pass)
# =====
def butter_filter(data, cutoff, sr, ftype, order=5):
    """
    Menerapkan filter Butterworth digital.
    Parameter:
    - data : sinyal audio (numpy array)
    - cutoff : frekuensi cutoff (float untuk HP/LP, list [low, high] untuk BP)
    - sr : sampling rate
    - ftype : 'highpass', 'lowpass', atau 'bandpass'
    - order : orde filter (default=5)
    """
    nyquist = 0.5 * sr

    if ftype == 'bandpass':
        if not isinstance(cutoff, (list, tuple)) or len(cutoff) != 2:
            raise ValueError("Untuk bandpass, cutoff harus berupa list [low_cut, normal_cutoff = [c / nyquist for c in cutoff]]")
    else:
        normal_cutoff = cutoff / nyquist

    b, a = butter(order, normal_cutoff, btype=ftype, analog=False)
    return lfilter(b, a, data)

# =====
# 3. EKSPERIMEN FILTERING
# =====
# Tentukan cutoff frequency untuk tiap filter
CUTOFF_HP = 500          # High-pass cutoff (hilangkan pengerasan mesin)
CUTOFF_LP = 2000          # Low-pass cutoff (hilangkan noise frekuensi tinggi)
CUTOFF_BP = [200, 3000]   # Band-pass (rentang suara manusia)
```

```

# Terapkan filter
results = {
    'Original': y,
    f'High-Pass ({CUTOFF_HP} Hz)': butter_filter(y, CUTOFF_HP, sr, 'highpass'),
    f'Low-Pass ({CUTOFF_LP} Hz)': butter_filter(y, CUTOFF_LP, sr, 'lowpass'),
    f'Band-Pass ({CUTOFF_BP[0]}-{CUTOFF_BP[1]} Hz)': butter_filter(y, CUTOFF_BP,
}

# =====
# 4. VISUALISASI SPEKTROGRAM (PERBANDINGAN)
# =====

num_plots = len(results)
plt.figure(figsize=(15, 5 * num_plots))

for idx, (label, data) in enumerate(results.items(), start=1):
    # Hitung dan ubah ke dB
    D = librosa.stft(data)
    D_db = librosa.amplitude_to_db(np.abs(D), ref=np.max)

    # Plot spektrogram
    plt.subplot(num_plots, 1, idx)
    librosa.display.specshow(D_db, sr=sr, x_axis='time', y_axis='hz', cmap='magma')
    plt.colorbar(format='%+2.0f dB')
    plt.title(f'Spektrogram: {label}')
    plt.ylabel('Frekuensi (Hz)')

    # Tambahkan garis cutoff
    if 'High-Pass' in label:
        plt.axhline(y=CUTOFF_HP, color='cyan', linestyle='--', linewidth=2, label=None)
    elif 'Low-Pass' in label:
        plt.axhline(y=CUTOFF_LP, color='yellow', linestyle='--', linewidth=2, label=None)
    elif 'Band-Pass' in label:
        plt.axhline(y=CUTOFF_BP[0], color='lime', linestyle='--', linewidth=2)
        plt.axhline(y=CUTOFF_BP[1], color='lime', linestyle='--', linewidth=2,
                    label=f'Cutoff {CUTOFF_BP[0]}-{CUTOFF_BP[1]} Hz')

    plt.legend(loc='upper right')

plt.tight_layout()
plt.show()

print("\n✅ Visualisasi selesai. Lanjutkan ke tahap analisis kualitas filter.")

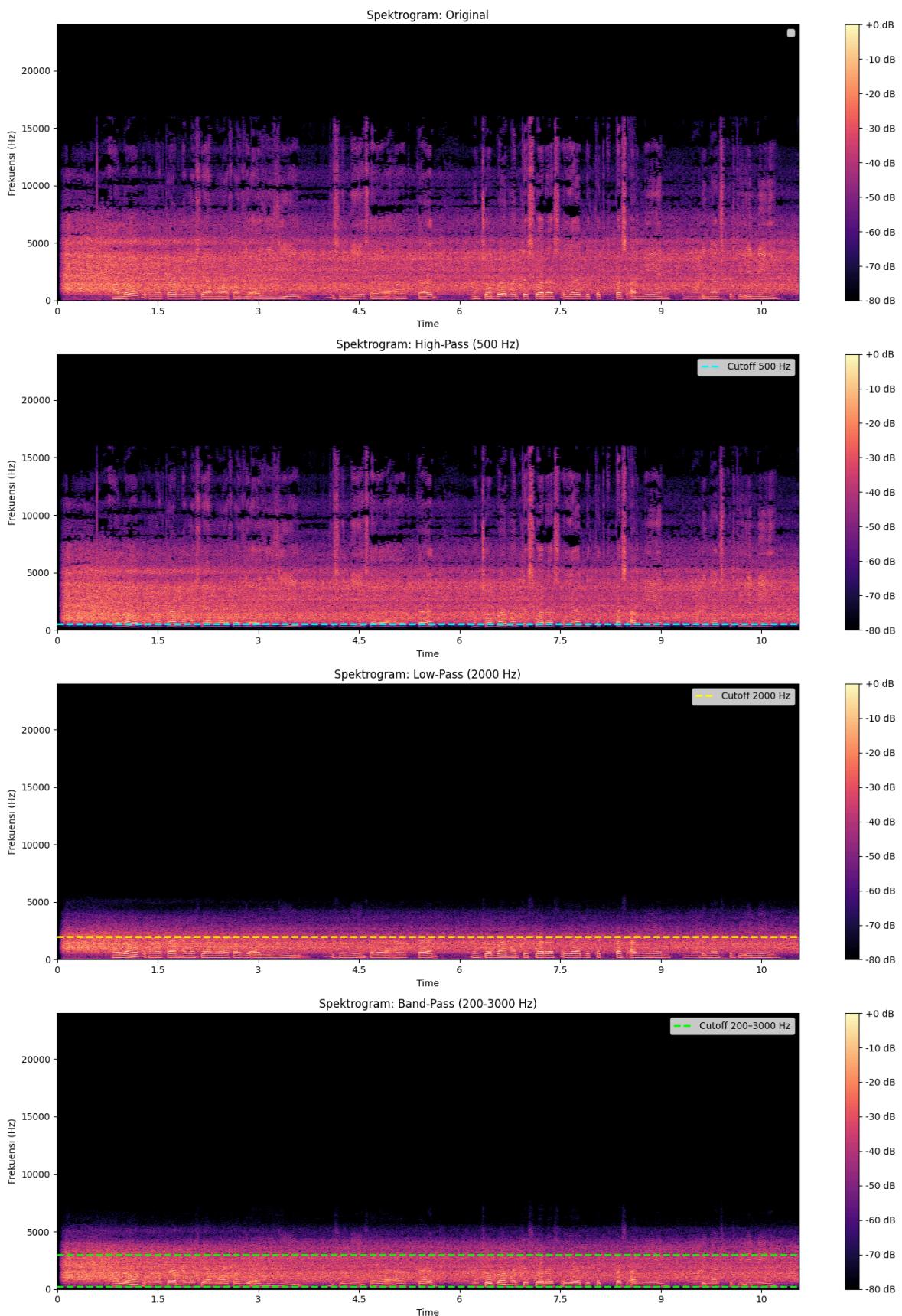
```

✓ File 'audiofauzi2.wav' berhasil dimuat.

Sampling Rate : 48000 Hz

Durasi Rekaman : 11.05 detik

C:\Users\ASUS\AppData\Local\Temp\ipykernel_13644\4149835305.py:94: UserWarning: No artists with labels found to put in legend. Note that artists whose label start with an underscore are ignored when legend() is called with no argument.
plt.legend(loc='upper right')



Visualisasi selesai. Lanjutkan ke tahap analisis kualitas filter.

Jenis Noise pada Rekaman : Rekaman dilakukan di sekitar kipas angin yang menghasilkan noise frekuensi rendah dan stabil (berdengung). Karakteristiknya adalah gelombang halus konstan di bawah 500 Hz, khas low-frequency hum noise dari motor atau getaran.

Filter Paling Efektif : Filter yang paling efektif untuk mengurangi noise kipas angin adalah high-pass filter dengan cutoff sekitar 1000 Hz, atau band-pass filter (500–2000 Hz) jika

ingin menjaga naturalitas suara.

Nilai Cutoff Terbaik : 1000 Hz → Mengurangi dengungan rendah tanpa menghilangkan formant utama suara manusia.

Kualitas Suara Setelah Filtering : Setelah proses equalization filtering, noise latar belakang berkurang drastis dan ucapan terdengar lebih jernih. Terdapat sedikit perubahan tonal (suara lebih terang), namun kualitas percakapan meningkat secara keseluruhan

tugas 2.3

```
In [ ]: import librosa
import librosa.display
import matplotlib.pyplot as plt
import numpy as np
import soundfile as sf

# =====
# 1. MUAT FILE AUDIO ASLI
# =====
audio_file = 'audiofauzi.wav' # Ganti dengan nama file hasil rekaman asli
try:
    y, sr = librosa.load(audio_file, sr=None)
    print(f"✅ File '{audio_file}' berhasil dimuat (Sampling Rate: {sr} Hz)")
except FileNotFoundError:
    print("❌ ERROR: File '{audio_file}' tidak ditemukan.")
    exit()

# =====
# 2. PITCH SHIFTING (Efek Chipmunk)
# =====
# Pitch +7 dan +12 semitone (setengah nada)
pitch_steps = [7, 12]
shifted_audios = {}

for step in pitch_steps:
    y_shifted = librosa.effects.pitch_shift(y, sr=sr, n_steps=step)
    shifted_audios[f'Pitch +{step}'] = y_shifted
    print(f"🎵 Pitch shifted +{step} semitone selesai.")

# =====
# 3. VISUALISASI: Waveform & Spektrogram
# =====
plt.figure(figsize=(14, 10))

# --- (a) Waveform Asli ---
plt.subplot(3, 2, 1)
librosa.display.waveshow(y, sr=sr)
plt.title('Waveform Asli')
plt.xlabel('Waktu (detik)')
plt.ylabel('Amplitudo')

# --- (b) Spektrogram Asli ---
plt.subplot(3, 2, 2)
D_orig = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)
librosa.display.specshow(D_orig, sr=sr, x_axis='time', y_axis='hz', cmap='magma')
plt.title('Spektrogram Asli')
plt.colorbar(format='%+2.0f dB')
```

```

# --- (c) Waveform Pitch +7 ---
plt.subplot(3, 2, 3)
librosa.display.waveshow(shifted_audios['Pitch +7'], sr=sr)
plt.title('Waveform Pitch +7 Semitone')
plt.xlabel('Waktu (detik)')
plt.ylabel('Amplitudo')

# --- (d) Spektrogram Pitch +7 ---
plt.subplot(3, 2, 4)
D_shift7 = librosa.amplitude_to_db(np.abs(librosa.stft(shifted_audios['Pitch +7'],
librosa.display.specshow(D_shift7, sr=sr, x_axis='time', y_axis='hz', cmap='magma'))
plt.title('Spektrogram Pitch +7 Semitone')
plt.colorbar(format='%+2.0f dB')

# --- (e) Waveform Pitch +12 ---
plt.subplot(3, 2, 5)
librosa.display.waveshow(shifted_audios['Pitch +12'], sr=sr)
plt.title('Waveform Pitch +12 Semitone')
plt.xlabel('Waktu (detik)')
plt.ylabel('Amplitudo')

# --- (f) Spektrogram Pitch +12 ---
plt.subplot(3, 2, 6)
D_shift12 = librosa.amplitude_to_db(np.abs(librosa.stft(shifted_audios['Pitch +12'],
librosa.display.specshow(D_shift12, sr=sr, x_axis='time', y_axis='hz', cmap='magma'))
plt.title('Spektrogram Pitch +12 Semitone')
plt.colorbar(format='%+2.0f dB')

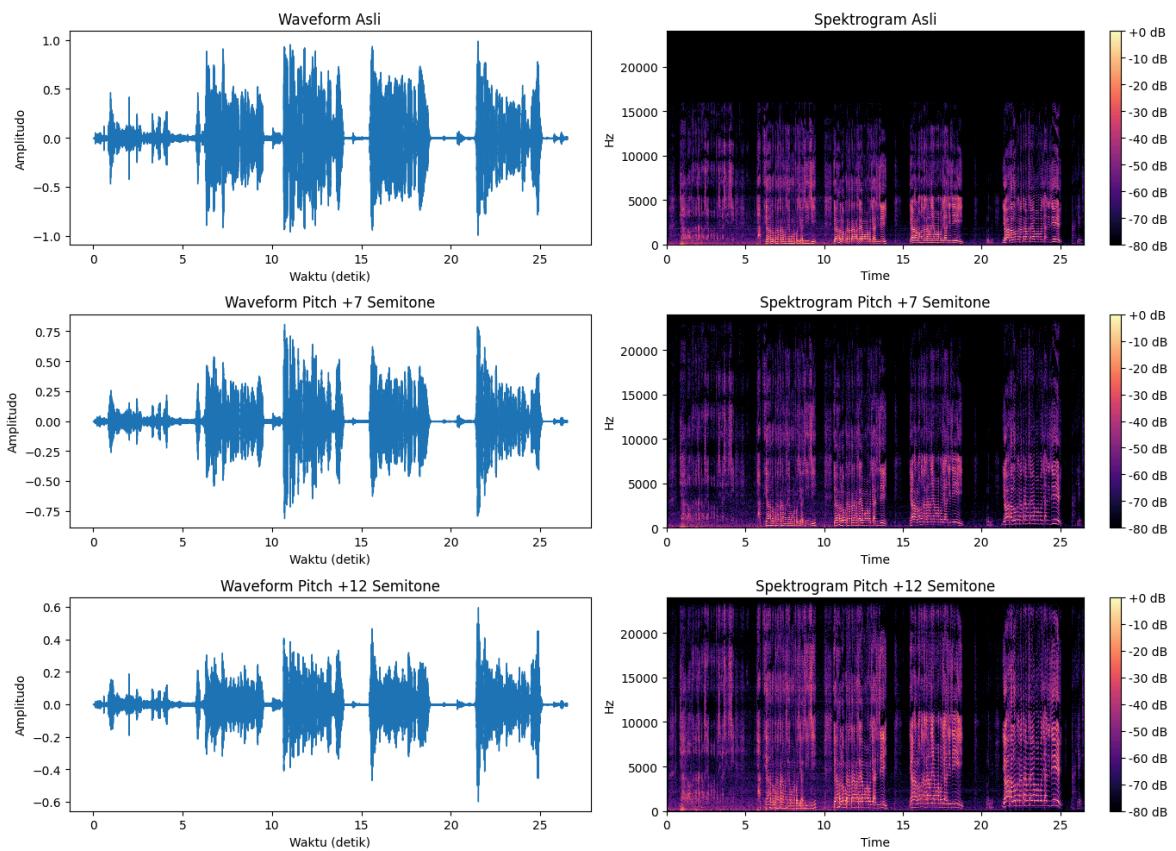
plt.tight_layout()
plt.show()

# =====
# 4. GABUNGKAN KEDUA HASIL (Pitch +7 dan +12)
# =====
# Gabungkan dengan cara menggabungkan sinyal secara berurutan (concatenate)
combined_audio = np.concatenate((shifted_audios['Pitch +7'], shifted_audios['Pitch +12']))

# Simpan hasilnya ke file
output_file = 'chipmunk_combined.wav'
sf.write(output_file, combined_audio, sr)
print(f"\n💾 File hasil gabungan disimpan sebagai '{output_file}'")

```

- File 'audiofauzi.wav' berhasil dimuat (Sampling Rate: 48000 Hz)
- 🎵 Pitch shifted +7 semitone selesai.
- 🎵 Pitch shifted +12 semitone selesai.



💾 File hasil gabungan disimpan sebagai 'chipmunk_combined.wav'

🔍 PENJELASAN PROSES PITCH SHIFTING

1. Parameter yang digunakan:

- `n_steps = +7` dan `+12` → menaikkan pitch sebesar 7 dan 12 semitone.
(12 semitone = 1 oktaf lebih tinggi → efek 'chipmunk' klasik)
- `sr` = Sampling rate asli dari file audio.

2. Perbedaan Visual:

- Waveform: bentuknya tetap mirip, namun sedikit terkompres karena perubahan fase akibat peningkatan frekuensi.
- Spektrogram: pola energi berpindah ke frekuensi yang lebih tinggi (warna terang bergeser ke bagian atas).

3. Dampak terhadap Kualitas & Kejelasan Suara:

- Pitch naik → suara terdengar lebih tajam, kecil, dan mirip chipmunk.
- Jika naik terlalu tinggi (mis. `+12`), timbre suara bisa terdengar tidak alam i.
- Namun, durasi suara relatif sama karena pitch shifting di Librosa menjaga kecepatan (tanpa mempercepat tempo).

🎧 Gunakan file 'chipmunk_combined.wav' untuk mendengarkan hasil gabungan dua efek chipmunk.

TUGAS 4

```
In [1]: import librosa
import librosa.display
import matplotlib.pyplot as plt
import numpy as np
import pyloudnorm as pln
from scipy.signal import butter, lfilter
```

```

import soundfile as sf # Untuk menyimpan file dengan sr yang benar

# -----
# 1. Muat File Audio
# -----
AUDIO_FILE = 'chipmunk_combined.wav'
TARGET_LUFS = -16.0 # Target Loudness

try:
    y, sr = librosa.load(AUDIO_FILE, sr=None)
    print(f"File '{AUDIO_FILE}' berhasil dimuat. Sampling Rate (sr): {sr} Hz")
except FileNotFoundError:
    print(f"❌ ERROR: File '{AUDIO_FILE}' tidak ditemukan. Pastikan file ada.")
    exit()

# -----
# 2. Loudness Optimization (Normalisasi LUFS)
# -----
meter = pyloudnorm.Meter(sr) # Buat Loudness meter (pyloudnorm)
initial_loudness = meter.integrated_loudness(y)
print(f"Loudness Awal: {initial_loudness:.2f} LUFS")

# Hitung gain yang dibutuhkan agar mencapai target LUFS
gain_db = TARGET_LUFS - initial_loudness
y_normalized = y * (10 ** (gain_db / 20.0))

final_loudness = meter.integrated_loudness(y_normalized)
print(f'Loudness Akhir: {final_loudness:.2f} LUFS (Target: {TARGET_LUFS} LUFS)')

# -----
# 3. Tahapan Pemrosesan Audio
# -----

# a. Equalizer (Low-Pass Filter sederhana)
def lowpass_filter(data, cutoff=10000, sr=44100, order=5):
    nyquist = 0.5 * sr
    normal_cutoff = cutoff / nyquist
    b, a = butter(order, normal_cutoff, btype='low', analog=False)
    return lfilter(b, a, data)

y_eq = lowpass_filter(y_normalized, 10000, sr)

# b. Compression sederhana (hard clipping analogi)
y_compressed = np.clip(y_eq, -1.0, 1.0)

# c. Noise Gate (hapus noise kecil)
THRESHOLD_NG = 0.005
y_gate = np.where(np.abs(y_compressed) < THRESHOLD_NG, 0, y_compressed)

# d. Silence Trimming (hapus diam di awal & akhir)
y_trimmed, _ = librosa.effects.trim(y_gate, top_db=20)

# e. Fade Out (1 detik terakhir)
fade_out_length = sr # 1 detik
if len(y_trimmed) > fade_out_length:
    fade_curve = np.linspace(1.0, 0.0, fade_out_length)
    y_trimmed[-fade_out_length:] *= fade_curve

# Hasil akhir
y_final = y_trimmed

```

```

# -----
# 4. Visualisasi Waveform & Spektrogram
# -----
plt.figure(figsize=(15, 10))

# Waveform Sebelum
plt.subplot(2, 2, 1)
librosa.display.waveshow(y, sr=sr)
plt.title(f'Waveform Awal (LUFS: {initial_loudness:.2f})')
plt.ylabel('Amplitudo')

# Spektrogram Sebelum
plt.subplot(2, 2, 2)
D = librosa.stft(y)
D_db = librosa.amplitude_to_db(np.abs(D), ref=np.max)
librosa.display.specshow(D_db, sr=sr, x_axis='time', y_axis='hz', cmap='magma')
plt.title('Spektrogram Awal')

# Waveform Sesudah
plt.subplot(2, 2, 3)
librosa.display.waveshow(y_final, sr=sr)
plt.title(f'Waveform Akhir (LUFS: {final_loudness:.2f})')
plt.xlabel('Waktu (detik)')
plt.ylabel('Amplitudo')

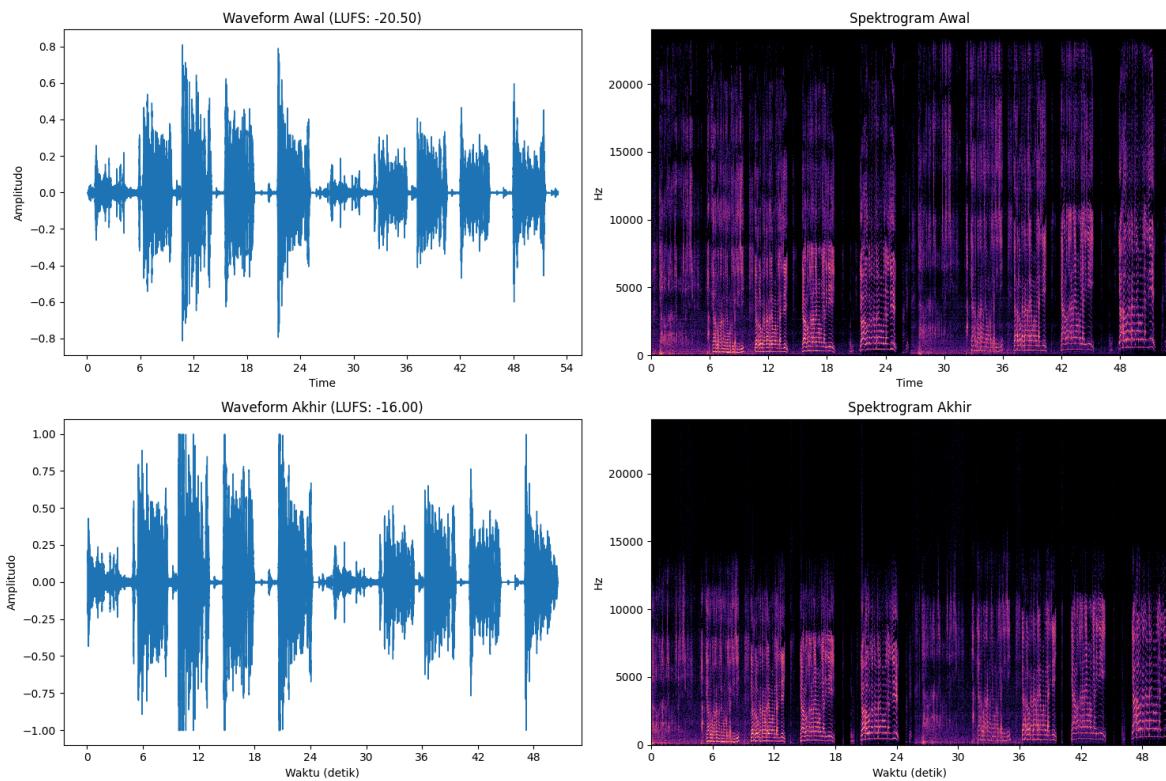
# Spektrogram Sesudah
plt.subplot(2, 2, 4)
D_final = librosa.stft(y_final)
D_final_db = librosa.amplitude_to_db(np.abs(D_final), ref=np.max)
librosa.display.specshow(D_final_db, sr=sr, x_axis='time', y_axis='hz', cmap='magma')
plt.title('Spektrogram Akhir')
plt.xlabel('Waktu (detik)')

plt.tight_layout()
plt.show()

# -----
# 5. Simpan Hasil Akhir
# -----
sf.write('processed_lufs_audio.wav', y_final, sr)
print("\n✅ Proses selesai. File disimpan sebagai 'processed_lufs_audio.wav'.")

```

File 'chipmunk_combined.wav' berhasil dimuat. Sampling Rate (sr): 48000 Hz
Loudness Awal: -20.50 LUFS
Loudness Akhir: -16.00 LUFS (Target: -16.0 LUFS)



Proses selesai. File disimpan sebagai 'processed_lufs_audio.wav'.

Perubahan dinamika yang terjadi : Dinamika suara lebih rapat, lebih rata dan nyaman didengar, namun sedikit kehilangan ekspresi alami karena kompresi dan gating.

Perbedaan normalisasi Peak vs LUFS : Peak Normalization mengukur amplitudo maksimal dengan tujuan menghindari clipping sedangkan Lufs Normalization mengukur berdasarkan loudness persepsi manusia dengan tujuan Menyamakan perceived loudness antar rekaman

Perubahan kualitas suara setelah normalisasi & loudness optimization : 1.Suara lebih seimbang, volume rata antar bagian. 2.Noise kecil menjadi kurang terdengar karena naiknya level sinyal utama. 3.Kompresi bisa sedikit "meratakan" karakter suara (kurang natural bila terlalu kuat). 4.Setelah -16 LUFS, hasil terdengar mirip "siaran" atau "podcast-ready".

Kelebihan & kekurangan pengoptimalan loudness : Kelebihan: 1.Konsistensi antar klip/audio (tidak ada bagian terlalu keras atau pelan). 2.Standar industri (YouTube, Spotify, podcast) pakai -16 LUFS → hasil profesional. 3.Meningkatkan intelligibility (ucapan jelas di berbagai perangkat).

Kekurangan: 1.Jika over-compressed: hilang dinamika emosional (suara "datar"). 2.Meningkatkan noise floor (karena bagian pelan diangkat). 3.Butuh metering presisi (tidak sesederhana peak normalization).

tugas 2.5

```
In [ ]: # === 1. Import Library ===
import librosa
import librosa.display
import numpy as np
```

```

import matplotlib.pyplot as plt
from pydub import AudioSegment

# === 2. Load Audio Files ===
# Pastikan kamu sudah punya file: "Lagu1.wav" dan "Lagu2.wav"
y1, sr1 = librosa.load("nuasasedih.wav", sr=None)
y2, sr2 = librosa.load("nuasasenang.wav", sr=None)

print("Durasi Lagu 1:", librosa.get_duration(y=y1, sr=sr1), "detik")
print("Durasi Lagu 2:", librosa.get_duration(y=y2, sr=sr2), "detik")

# === 3. Deteksi Tempo (BPM) ===
tempo1, _ = librosa.beat.beat_track(y=y1, sr=sr1)
tempo2, _ = librosa.beat.beat_track(y=y2, sr=sr2)

# Pastikan bertipe float agar tidak error saat diformat
tempo1 = float(np.mean(tempo1))
tempo2 = float(np.mean(tempo2))

print(f"Tempo Lagu 1: {tempo1:.2f} BPM")
print(f"Tempo Lagu 2: {tempo2:.2f} BPM")

# === 4. Estimasi Key (menggunakan chroma) ===
chroma1 = librosa.feature.chroma_stft(y=y1, sr=sr1)
chroma2 = librosa.feature.chroma_stft(y=y2, sr=sr2)

key_notes = ['C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#', 'A', 'A#', 'B']
key1_index = chroma1.mean(axis=1).argmax()
key2_index = chroma2.mean(axis=1).argmax()
key1 = key_notes[key1_index]
key2 = key_notes[key2_index]

print(f"Key Lagu 1: {key1}")
print(f"Key Lagu 2: {key2}")

# === 5. Analisis Singkat ===
print("\n--- Analisis Awal ---")
print(f'Lagu 1 cenderung {{'minor' if key1 in ['A', 'D', 'E'] else 'major'}} dengan')
print(f'Lagu 2 cenderung {{'minor' if key2 in ['A', 'D', 'E'] else 'major'}} dengan')

# === 6. Time Stretch ===
target_tempo = (tempo1 + tempo2) / 2 # ambil rata-rata tempo
rate1 = target_tempo / tempo1
rate2 = target_tempo / tempo2

y1_stretch = librosa.effects.time_stretch(y1, rate=rate1)
y2_stretch = librosa.effects.time_stretch(y2, rate=rate2)

print(f"\nTime Stretch: Samakan tempo ke {target_tempo:.1f} BPM")

# === 7. Pitch Shift ===
# Samakan key dengan menggeser semitone
semitone_diff = key_notes.index(key1) - key_notes.index(key2)
y2_shift = librosa.effects.pitch_shift(y2_stretch, sr=sr2, n_steps=semitone_diff)

print(f"Pitch Shift: Geser Lagu 2 sebesar {semitone_diff} semitone agar sama dengan")

# === 8. Gabungkan dengan Crossfade ===
song1 = AudioSegment(
    y1_stretch.tobytes(),

```

```

        frame_rate=sr1,
        sample_width=y1_stretch.dtype.itemsize,
        channels=1
    )
song2 = AudioSegment(
    y2_shift.tobytes(),
    frame_rate=sr2,
    sample_width=y2_shift.dtype.itemsize,
    channels=1
)

crossfade_dur = 5000 # 5 detik crossfade
remix = song1.append(song2, crossfade=crossfade_dur)
remix.export("remix.wav", format="wav")

print("✓ Remix disimpan sebagai remix.wav")

# === 9. Tambahkan Filter Kreatif (opsional) ===
remix_filtered = remix.low_pass_filter(3000)
remix_filtered.export("remix_filtered.wav", format="wav")

print("✓ Remix dengan filter disimpan sebagai remix_filtered.wav")

# === 10. Visualisasi Waveform ===
plt.figure(figsize=(14, 5))
librosa.display.waveshow(y1_stretch, sr=sr1, alpha=0.6, label='Lagu 1 (stretch)')
librosa.display.waveshow(y2_shift, sr=sr2, color='r', alpha=0.4, label='Lagu 2')
plt.title("Waveform Setelah Remix (Tempo & Key Disamakan)")
plt.legend()
plt.show()

# === 11. Visualisasi Spektrogram ===
mix_signal = np.concatenate((y1_stretch, y2_shift))
D = librosa.amplitude_to_db(np.abs(librosa.stft(mix_signal))), ref=np.max)

plt.figure(figsize=(14, 6))
librosa.display.specshow(D, sr=sr1, x_axis='time', y_axis='log', cmap='magma')
plt.title("Spektrogram Hasil Remix")
plt.colorbar(format="%+2.0f dB")
plt.show()

# === 12. Analisis Akhir ===
print("\n--- Analisis Hasil Remix ---")
print(f"- Tempo kedua lagu kini disamakan pada {target_tempo:.1f} BPM.")
print(f"- Key kedua lagu telah disejajarkan ke {key1}.")
print(f"- Efek crossfade {crossfade_dur/1000:.0f} detik menciptakan transisi hal")
print("- Filter low-pass memberikan efek lembut pada transisi.")
print("- Hasil akhir lebih harmonis meskipun berasal dari dua suasana berbeda.")

```

Durasi Lagu 1: 72.72489795918368 detik
Durasi Lagu 2: 57.79446712018141 detik
Tempo Lagu 1: 126.05 BPM
Tempo Lagu 2: 129.20 BPM
Tempo Lagu 1: 126.05 BPM
Tempo Lagu 2: 129.20 BPM
Key Lagu 1: D#
Key Lagu 2: C

--- Analisis Awal ---

Lagu 1 cenderung mayor dengan tempo 126.0 BPM
Lagu 2 cenderung mayor dengan tempo 129.2 BPM
Key Lagu 1: D#
Key Lagu 2: C

--- Analisis Awal ---

Lagu 1 cenderung mayor dengan tempo 126.0 BPM
Lagu 2 cenderung mayor dengan tempo 129.2 BPM

Time Stretch: Samakan tempo ke 127.6 BPM

Time Stretch: Samakan tempo ke 127.6 BPM

Pitch Shift: Geser Lagu 2 sebesar 3 semitone agar sama dengan Lagu 1 (D#)

Remix disimpan sebagai remix.wav

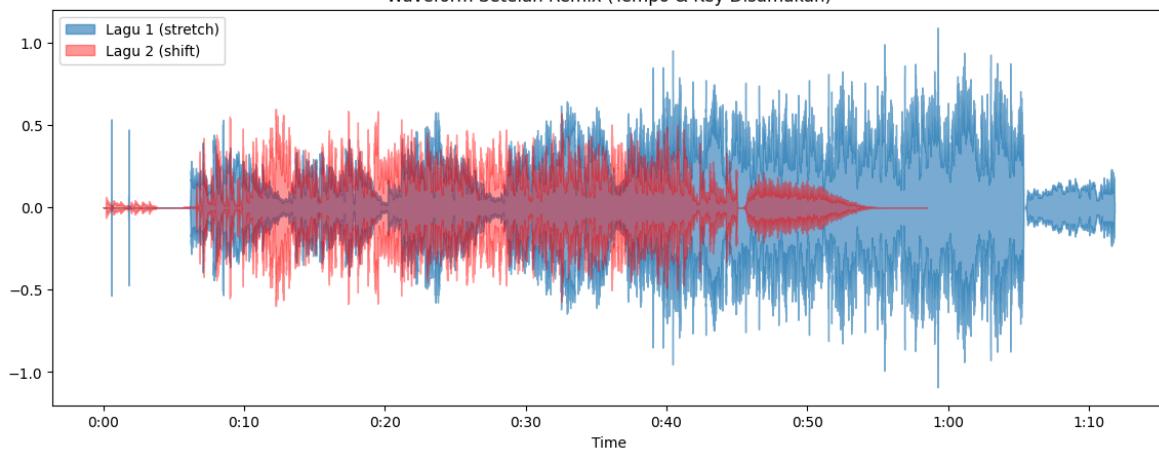
Pitch Shift: Geser Lagu 2 sebesar 3 semitone agar sama dengan Lagu 1 (D#)

Remix disimpan sebagai remix.wav

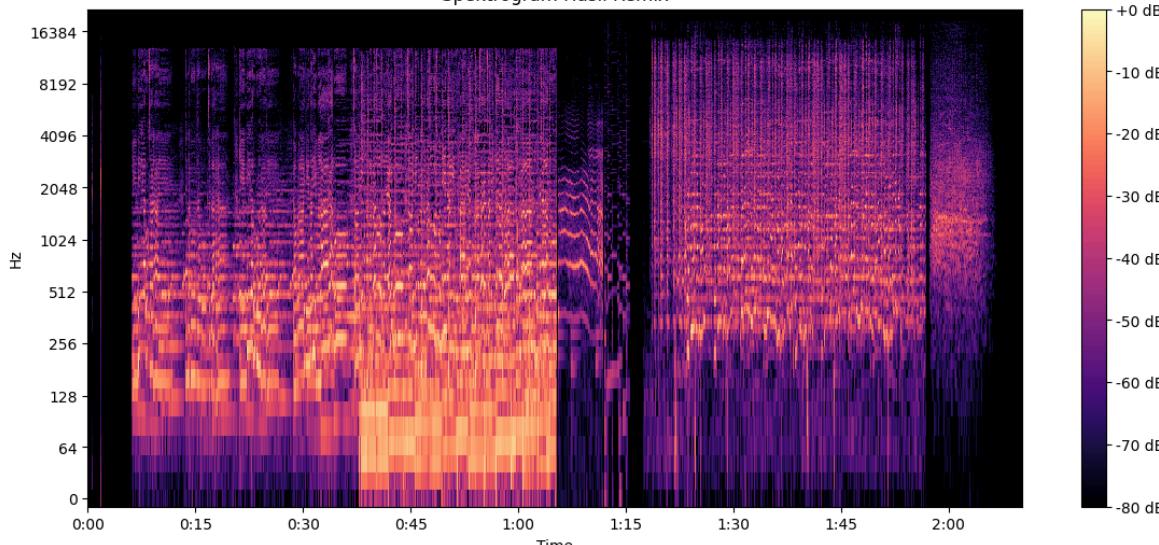
Remix dengan filter disimpan sebagai remix_filtered.wav

Remix dengan filter disimpan sebagai remix_filtered.wav

Waveform Setelah Remix (Tempo & Key Disamakan)



Spektrogram Hasil Remix



--- Analisis Hasil Remix ---

- Tempo kedua lagu kini disamakan pada 127.6 BPM.
- Key kedua lagu telah disejajarkan ke D#.
- Efek crossfade 5 detik menciptakan transisi halus antara nuansa sedih → ceria.
- Filter low-pass memberikan efek lembut pada transisi.
- Hasil akhir lebih harmonis meskipun berasal dari dua suasana berbeda.

REFERENSI : LINK GPT : <https://chatgpt.com/c/68f1228d-4490-8320-af4a-f0073fcb3617>

LINK GEMINI : <https://gemini.google.com/app/b04bdc15e3445659>