

**LAPORAN TUGAS BESAR 2**  
**IF2123 - ALJABAR LINIER DAN GEOMETRI**



**Kelompok 25**  
~~ git commit -m "tubes paling N\*\*\*\* yang pernah kami kerjakan" ~~

Fajar Kurniawan 13523027

Mochammad Fariz Rifqi Rizqulloh 13523069

Muhammad Adha Ridwan 13523098

**PROGRAM STUDI TEKNIK INFORMATIKA**  
**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**  
**INSTITUT TEKNOLOGI BANDUNG**

**2024**

## Daftar Isi

<b>Daftar Isi</b>	<b>1</b>
<b>BAB I : DESKRIPSI MASALAH</b>	<b>2</b>
1.1. Pemrosesan Suara dan Gambar	2
1.2. Information Retrieval	3
<b>BAB II : TEORI SINGKAT</b>	<b>4</b>
2.1 Sistem Temu Balik Suara (MIR)	4
2.2 Metode Ekstraksi Fitur Untuk Query by Humming	4
2.2.1 Pemrosesan Audio	4
2.2.2 Ekstraksi Fitur	5
2.2.3 Penghitungan Similaritas	6
2.3 Image Retrieval dengan PCA	6
2.3.1 Pemuatan dan Pemrosesan Citra	6
2.3.2 Standarisasi Data	7
2.3.3 Penggunaan Singular Value Decomposition (SVD) untuk Komputasi PCA	7
<b>BAB III : ARSITEKTUR WEBSITE</b>	<b>10</b>
3. 1. Frontend	10
3. 2. Backend	10
3.2.1. Implementasi Album Picture Finder - Principal Component Analysis	11
3.2.2. Music Information Retrieval - Query by Humming	13
<b>BAB IV : EKSPERIMEN</b>	<b>15</b>
4.1 Tata Cara Penggunaan Program	15
4.2 Uji pada Dataset dengan Gambar dan MIDI	15
4.3 Analisis dari Desain Solusi Pemrosesan Gambar dan Suara	19
<b>BAB V : KESIMPULAN</b>	
5.1 Kesimpulan	21
1. Sistem Temu Balik Gambar	21
2. Sistem Temu Balik Suara	21
3. Efisiensi Sistem	22
4. Kendala dan Potensi Peningkatan	22
5.2 Saran dan Refleksi	22
5.3 Komentar	22
<b>DAFTAR PUSTAKA</b>	<b>24</b>

## BAB I : DESKRIPSI MASALAH

### 1.1. Pemrosesan Suara dan Gambar

Suara selalu menjadi hal yang paling penting dalam kehidupan manusia. Manusia berbicara mengeluarkan suara dan mendengarkan suatu suara untuk diresap ke otak dan mencari informasi dari suara tersebut. Suara juga bisa dijadikan orang-orang di dunia ini sebuah media untuk membuat karya seni. Contohnya adalah alat mendeteksi lagu. Manusia bisa mendeteksi suara dengan menggunakan indera pendengar dan memberikan kesimpulan akan apa jenis suara tersebut melalui respon dari otak. Sama seperti manusia, teknologi juga bisa mendeteksi suara dan memberikan jawaban mereka melalui algoritma-algoritma yang beragam bahkan bisa melebihi kapabilitas manusia. Dengan menggunakan algoritma apapun, konsep dari pendekripsi dan interpretasi suara itu bisa juga disebut dengan sistem temu balik suara atau bisa disebut juga dengan audio retrieval system. Banyak aplikasi yang menggunakan konsep sistem temu balik contohnya adalah Shazam.



**Gambar 1.1** Shazam sebagai aplikasi audio retrieval system

Selain suara, manusia juga memiliki penglihatan sebagai salah satu inderanya dan bisa melihat warna dan gambar yang bermacam-macam. Teknologi komputasi juga memiliki kapabilitas yang sama dan bisa melihat gambar sama seperti kita, tetapi teknologi seperti ini juga bisa merepresentasikan gambar tersebut sebagai beragam-ragam angka yang bisa disebut juga fitur. Di dalam Tugas Besar 2 ini, kami membuat semacam aplikasi Shazam yaitu sebuah aplikasi yang meminta input lagu dan aplikasi tersebut mendekripsi apa nama dari lagu tersebut dan beberapa detail lainnya. Pada tugas besar ini, kami menggunakan aljabar vektor untuk mencari perbandingan antar satu audio dengan audio yang lain. Kami menggunakan konsep yang

bernama *Music Information Retrieval* atau MIR untuk mencari dan mengidentifikasi suara berdasarkan fitur-fitur yang dimilikinya. Tidak hanya itu, kami juga menggunakan konsep *Principal Component Analysis* (PCA) untuk mencari kumpulan audio melalui deteksi wajah berbagai orang.

## **1.2. *Information Retrieval***

*Information Retrieval* adalah konsep meminta informasi dari sebuah data dengan memasukkan data tertentu. Pada tugas besar ini, kami berikut dengan dua jenis *Information Retrieval*. *Image Retrieval* dan *Music Information Retrieval*. *Image Retrieval* adalah konsep untuk memasukkan sebuah input gambar dan berharap mendapatkan gambar yang ada di data sesuai dengan informasi dan perhitungan yang diinginkan. Sedangkan *Music Information Retrieval* (MIR) adalah konsep untuk memasukkan sebuah input audio dan berharap mendapatkan audio yang ada di data sesuai dengan informasi dan perhitungan yang diinginkan. Pada tugas besar kali ini, kalian akan mengimplementasikan *Image Retrieval* dengan menggunakan *Principal Component Analysis* dan *Music Information Retrieval* dengan menggunakan *humming*.

## BAB II : TEORI SINGKAT

### 2.1 Sistem Temu Balik Suara (MIR)

Music Information Retrieval (MIR) adalah bidang multidisiplin yang bertujuan untuk menganalisis, memproses, dan memahami data musik melalui pendekatan komputasi. MIR mencakup berbagai aspek seperti pengenalan pola, klasifikasi genre, pengenalan melodi, pencarian musik berbasis query, hingga temu balik musik berdasarkan humming atau potongan audio. Teknologi ini sangat relevan di era digital untuk membantu pengguna mencari dan mengenali musik dengan cara yang lebih intuitif, baik melalui suara, teks, maupun data digital lainnya.

Ekstraksi fitur pada file audio adalah langkah krusial dalam analisis data musik, yang bertujuan untuk mengidentifikasi karakteristik utama dari sinyal audio seperti pitch, timbre, ritme, dan dinamika. Salah satu pendekatan utama dalam ekstraksi fitur adalah penggunaan teknik *windowing*, di mana sinyal audio dibagi menjadi segmen-semen kecil (disebut jendela atau window) untuk analisis lokal dalam domain waktu atau frekuensi. Setiap jendela diproses secara individual untuk menghitung fitur spesifik. Pada pekerjaan ini, fitur spesifik yang dimaksud adalah **ATB** (Absolute Tone Based), **FTB** (Feature Tone Based), dan **RTB** (First Tone Based). Dengan memanfaatkan pitch yang diekstraksi pada setiap jendela, sistem dapat mengidentifikasi pola musik, membandingkan melodi, atau mencocokkan query berbasis humming dengan database musik. Proses ini memungkinkan representasi audio menjadi lebih terstruktur dan siap untuk berbagai aplikasi MIR, seperti temu balik musik atau analisis genre.

MIDI (Musical Instrument Digital Interface) merupakan format data musik digital yang sangat efisien dalam merepresentasikan nada (pitch) dan durasi dari notasi musik tanpa menyertakan informasi audio mentah seperti timbre atau tekstur suara. Format MIDI memungkinkan representasi nada dalam bentuk angka, yang langsung dapat diolah untuk keperluan analisis. Dalam konteks Sistem Temu Balik Suara (MIR), format MIDI akan mempermudah proses identifikasi pitch dibandingkan dengan file audio yang memerlukan teknik lebih kompleks seperti transformasi Fourier atau analisis domain waktu. Dengan MIDI, pitch dari sebuah notasi musik dapat langsung dikenali melalui nilai-nilai yang telah terdefinisi, sehingga mempercepat proses ekstraksi fitur.

### 2.2 Metode Ekstraksi Fitur Untuk Query by Humming

#### 2.2.1 Pemrosesan Audio

Pemrosesan audio dalam sistem query by humming menggunakan file MIDI

dengan fokus pada track melodi utama, umumnya di Channel 1. Setiap file MIDI diproses menggunakan metode windowing yang membagi melodi menjadi segmen 20-40 beat dengan sliding window 4-8 beat. Teknik ini memungkinkan pencocokan fleksibel dari berbagai bagian lagu yang mungkin diingat pengguna.

Proses windowing disertai normalisasi tempo dan pitch untuk mengurangi variasi humming. Setiap note event dikonversi menjadi representasi numerik yang mempertimbangkan durasi dan urutan nada, memungkinkan sistem membandingkan potongan melodi dengan database. Berikut adalah formula untuk melakukan normalisasi tempo yang dibutuhkan.

$$NP(note) = \frac{(note-\mu)}{\sigma}$$

**Gambar 2.1** Formula normalisasi tempo

### 2.2.2 Ekstraksi Fitur

#### Distribusi tone

Distribusi tone diukur berdasarkan tiga viewpoints. Fitur Absolute Tone Based (ATB) menghitung frekuensi kemunculan setiap nada berdasarkan skala MIDI (0-127). Histogram yang dihasilkan memberikan gambaran distribusi absolut nada dalam data. Hal ini penting untuk menangkap karakteristik statis melodi dalam sinyal audio. Langkah pertama adalah membuat histogram dengan 128 bin, sesuai dengan rentang nada MIDI dari 0 hingga 127. Kemudian, frekuensi kemunculan masing-masing nada MIDI dalam data dihitung. Setelah itu, histogram dinormalisasi untuk mendapatkan distribusi yang terstandarisasi.

Fitur Relative Tone Based (RTB) menganalisis perubahan antara nada yang berurutan, menghasilkan histogram dengan nilai dari -127 hingga +127. RTB berguna untuk memahami pola interval melodi, yang lebih relevan dalam mencocokkan humming dengan dataset yang tidak bergantung pada pitch absolut. Dimulai dengan membangun histogram yang memiliki 255 bin dengan rentang nilai dari -127 hingga +127. Selanjutnya, dihitung selisih antara nada-nada yang berurutan dalam data. Terakhir, dilakukan normalisasi pada histogram yang telah dibuat.

Fitur First Tone Based (FTB) fokus pada perbedaan antara setiap nada dengan nada pertama, menciptakan histogram yang mencerminkan hubungan relatif terhadap titik referensi awal. Pendekatan ini membantu menangkap struktur relatif nada yang lebih stabil terhadap variasi pitch pengguna. Histogram dibuat dengan 255 bin, juga mencakup rentang nilai dari -127 hingga +127. Kemudian, selisih antara setiap nada dalam data dengan nada pertama dihitung. Histogram yang dihasilkan kemudian dinormalisasi untuk menghasilkan distribusi yang seimbang

### Normalisasi

Normalisasi memastikan bahwa semua nilai dalam histogram berada dalam skala probabilitas. Berikut adalah formula umum dari normalisasi yang digunakan:

$$H_{norm} = \frac{H[d]}{\sum_d^{127} H[d]}$$

**Gambar 2.2** Formula umum normalisasi

Dimana H adalah Histogram dan d adalah bin dari histogram tersebut.

Bahas mengenai fitur Fitur Absolute Tone Based (ATB), Fitur Relative Tone Based (RTB), serta Fitur First Tone Based (FTB)

### 2.2.3 Penghitungan Similaritas

Setiap histogram diubah menjadi sebuah vektor dan dilakukan perhitungan kemiripannya menggunakan *cosine similarity*. Pada jurnal terkait, metode yang digunakan adalah euclidean distance, tetapi pada tugas kali ini metode perhitungan similaritas yang digunakan adalah cosine similarity. Cosine Similarity adalah ukuran untuk menentukan seberapa mirip dua vektor dalam ruang berdimensi tinggi, dengan menghitung sudut cosinus di antara keduanya. Semakin kecil sudutnya (semakin dekat ke 1 hasilnya), semakin mirip kedua vektor tersebut. Sehingga cosine similarity bisa dijadikan salah satu metode lain dalam perhitungan similaritas. Berikut adalah formula dari cosine similarity

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

**Gambar 2.3** Formula *cosine similarity*

## 2.3 Image Retrieval dengan PCA

### 2.3.1 Pemuatan dan Pemrosesan Citra

Dilakukan pemrosesan seluruh gambar yang ada pada dataset. Gambar diubah menjadi grayscale untuk mengurangi kompleksitas gambar dan membuat fokus menjadi bagian terang dan gelap gambar. Yang berarti setiap gambar direpresentasikan dalam intensitas pixel saja tanpa informasi warna.

$$I(x,y) = 0.2989 \cdot R(x,y) + 0.5870 \cdot G(x,y) + 0.1140 \cdot B(x,y)$$

Selanjutnya, gambar akan diubah besarnya sehingga ukurannya sama untuk seluruh gambar. Ukuran seluruh gambar harus konsisten untuk membuat perhitungan semakin akurat. Lalu vektor grayscale pada gambar diubah menjadi 1D untuk membuat dimensi vektor dapat dilakukan pemrosesan data. Jika gambar memiliki dimensi  $M \times N$ , maka hasilnya adalah vektor dengan panjang  $M \times N$ .

$$I = [I_1, I_2, \dots, I_{M \times N}]$$

### 2.3.2 Standarisasi Data

Pada step ini dilakukan standarisasi data di sekitar nilai 0. Dihitung rata rata dari setiap gambar untuk suatu piksel.

$$\mu_j = \frac{1}{N} \sum_{i=1}^N x_{ij}$$

di mana:

- $x_{ij}$ : nilai piksel ke- $j$  pada gambar ke- $i$ ,
- $N$ : jumlah total gambar dalam dataset.

Lalu piksel tersebut dikurangi dengan rata-rata yang sudah dihitung untuk melakukan standarisasi dengan rumus:  $x'_{ij} = x_{ij} - \mu_j$

### 2.3.3 Penggunaan *Singular Value Decomposition* (SVD) untuk Komputasi PCA

Dilakukan perhitungan SVD pada gambar-gambar yang sudah distandarisasi. Matriks kovarians dibuat dari data yang sudah distandarisasi.

$$C = \frac{1}{N} X'^T X'$$

di mana:

- $X'$ : matriks data yang sudah distandarisasi.

Lalu dilakukan dekomposisi nilai singular untuk mendapatkan komponen utama.

$$\mathbf{C} = \mathbf{U}\Sigma\mathbf{U}^T$$

- $\mathbf{U}$ : matriks eigenvector (komponen utama),
- $\Sigma$ : matriks eigenvalue (menunjukkan varian data di sepanjang komponen utama).

Lalu diambil n jumlah component utama teratas dari hasil SVD dan dilakukan proyeksi gambar ke komponen utama. Dipilih k-komponen utama teratas ( $k \ll M \cdot N$ ) dan data diproyeksikan.

$$\mathbf{Z} = \mathbf{X}'\mathbf{U}_k$$

di mana:

- $\mathbf{U}_k$ : matriks eigenvector dengan n-dimensi.

#### 2.3.4 Penghitungan Similaritas

Dilakukan pencarian kesamaan dengan mencari jarak euclidean dari tiap dataset dengan gambar query. Lalu dilakukan pengurutan kecocokan dari yang paling tinggi. Pertama-tama, direpresentasikan gambar query dalam ruang komponen utama dengan proyeksi yang sama.

$$\mathbf{q} = (\mathbf{q}' - \mu)\mathbf{U}_k$$

Dimana:

- $\mathbf{q}$  = Vektor proyeksi dari gambar query ke ruang komponen utama (PCA).
- $\mathbf{q}'$ : Gambar query dalam format vektor (setelah grayscale, resize, dan flattening).
- $\mu$ : Rata-rata piksel dari dataset (per piksel).
- $\mathbf{U}_k$ : Matriks eigenvector dengan k dimensi utama dari PCA.

Kemudian, dihitung jarak Euclidean antara gambar query dengan semua gambar dalam dataset:

$$d(\mathbf{q}, \mathbf{z}_i) = \sqrt{\sum_{j=1}^k (q_j - z_{ij})^2}$$

- $d(\mathbf{q}, \mathbf{z}_i)$  = Jarak antara gambar query  $\mathbf{q}$  dan gambar ke- $i$  dalam ruang komponen utama.
- $\mathbf{z}_i$  = Vektor proyeksi dari gambar ke- $i$  dalam dataset ke ruang komponen utama.
- $q_j$ : Elemen ke- $j$  dari vektor proyeksi query  $\mathbf{q}$ .
- $z_{ij}$ : Elemen ke- $j$  dari vektor proyeksi gambar ke- $i$ , yaitu  $\mathbf{z}_i$ .
- $k$ : Jumlah dimensi ruang komponen utama yang dipilih.

Lalu, diurutkan hasil berdasarkan jarak terkecil.

## BAB III : ARSITEKTUR WEBSITE

### 3. 1. Frontend

Vite mengoptimalkan pengalaman pengembangan dengan menyediakan hot module replacement (HMR) yang cepat untuk React, yang mempercepat siklus pengembangan. React bertanggung jawab untuk merender tampilan antarmuka pengguna (UI) secara dinamis dan interaktif. Dalam React, rendering adalah proses memperbarui UI berdasarkan perubahan pada state atau props aplikasi. Vite memastikan proses bundling yang efisien dan pemuatan aplikasi yang cepat.

Untuk meningkatkan performa, React dengan Vite meminimalkan rendering ulang yang tidak perlu. Komponen hanya dirender ulang ketika ada perubahan pada state atau props, sehingga mencegah terjadinya bottleneck performa dan meningkatkan pengalaman pengguna dengan membuat UI lebih cepat.

FastAPI digunakan untuk backend, menyediakan lapisan API yang efisien untuk menangani permintaan. FastAPI mendukung endpoint asinkron berperforma tinggi yang menangani pemrosesan data, interaksi dengan database, dan operasi backend lainnya. Endpoint ini dapat dipanggil dari frontend React. FastAPI dapat mengirimkan data gambar dan suara sehingga dapat ditampilkan dan dimainkan oleh frontend. Endpoint FastAPI dapat dipanggil oleh frontend untuk melakukan suatu proses pada data server. Terdapat endpoint untuk masing-masing keperluan antara lain, upload gambar, midi, dataset, mapper untuk hubungan gambar dan midi, melakukan *querying* baik untuk gambar maupun midi, *querying* berdasarkan masukan pada search bar, menyimpan data file upload baik gambar, midi, mapper, maupun dataset.

### 3. 2. Backend

Struktur Program Utama terbagi menjadi 3 bagian utama, yaitu Midi Processor, Image Processor, dan Main Server. Midi Processor memuat fungsi-fungsi pembantu untuk memproses dan query file MIDI, sementara Image Processor memuat fungsi-fungsi pembantu untuk memproses dan query file gambar, serta Main Server merupakan komponen penghubung Backend dengan Frontend

### 3.2.1. Implementasi Album Picture Finder - Principal Component Analysis

Procedure/Function	Kegunaan
<pre>def process_image(image_path):</pre>	Subprogram digunakan untuk melakuakn <i>pre-process</i> terhadap sebuah file image yang tersimpan dalam path <code>image_path</code> .
<pre>def process_dataset_concurrently(directory, max_workers=8):</pre>	Subprogram controller untuk menerima masukan dari file-file image dari dataset yang tersimpan di dalam directory. Program dijalankan dengan menggunakan <i>multithreading</i>
<pre>def standardize_dataset(processed_dataset):</pre>	Subprogram untuk melakukan standarisasi terhadap matrix gambar yang telah dilinearsiasi..
<pre>def perform_truncated_svd(standardized_dataset, n_components):</pre>	Subprogram untuk melakukan dekomposisi SVD terhadap dataset gambar yang telah distandarisasi, dengan mengambil mengambil <code>n_components</code> sebagai banyaknya dimensi ruang komponen yang akan dipilih
<pre>def process_query_image(image_path):</pre>	Subprogram untuk melakukan proses terhadap gambar yang akan di- <i>query</i> (gambar yang di input oleh user)
<pre>def cosine_similarity(vec1, vec2):</pre>	Subprogram untuk menghitung similarity dari 2 buah vektor dengan menggunakan metode <i>cosine similarity</i>
<pre>def query_image(query_image_path, eigenvectors, projected_dataset, mean_dataset):</pre>	Subprogram controller untuk melakukan penghitungan kemiripan dari gambar yang di- <i>query</i> dengan gambar-gambar yang ada di database.
<pre>def initialize_dataset_concurrently(directory):</pre>	Subprogram controller untuk melakukan <i>pre-processing</i> semua file (dataset) yang terdapat di dalam folder directory.

```
def get_similarities(similarities,  
sorted_indices, threshold = 0.7):
```

Subprogram untuk mengembalikan list tupel (nama\_file, similaritas) yang memenuhi syarat similaritas diatas threshold. Hasil dari `get_similarities` akan dikirim menuju `main.py` untuk ditampilkan pada website

#### ALUR KERJA PROGRAM

```
dataset_directory = <directory dimana dataset disimpan>  
vectors, projected_dataset, mean_dataset = initialize_dataset_concurrently(dataset_directory)  
  
query_image_path = <Path dimana image yang diquery oleh user disimpan>  
similarities, sorted_indices = query_image(query_image_path, eigenvectors, projected_dataset,  
mean_dataset)  
  
result = get_similarities(similarities, sorted_indices)
```

#### PENJELASAN :

Program akan dipanggil oleh [main.py], dimana main.py dapat menerima masukan directory dataset serta path image yang diquery oleh user saat user melakukan upload

### 3. 2. 2. Music Information Retrieval - Query by Humming

Procedure/Function	Kegunaan
<code>def get_midi_notes(file_path):</code>	Subprogram untuk mendapatkan list of notes dari file midi yang tersimpan di dalam <code>file_path</code> .
<code>def shrink_atb_histogram(hist):</code>	Subprogram yang digunakan untuk melakukan penyusutan ukuran histogram fitur ATB. Subprogram akan mengembalikan suatu vektor berdimensi 25
<code>def shrink_rtb_ftb_histogram(hist):</code>	Subprogram yang digunakan untuk melakukan penyusutan ukuran histogram fitur RTB atau FTB. Subprogram akan mengembalikan suatu vektor berdimensi 25
<code>def get_feature(notes, window_size=40, step=8):</code>	Subprogram akan memproses list of notes, dan mengekstrak feature ATB, RTB, dan FTB dari list tersebut
<code>def process_single_midi_file(file_path):</code>	Subprogram controller untuk melakukan pre-process dari suatu file midi yang disimpan dalam path <code>file_path</code>
<code>def cosine_similarity(vec1, vec2):</code>	Subprogram untuk menghitung similaritas antara 2 vektor dengan menggunakan metode <i>cosine similarity</i>
<code>def process_all_midi_files_concurrently(directory):</code>	Subprogram controller untuk melakukan pre-process ke semua file MIDI yang tersimpan dalam folder directory. Program dijalankan dengan menggunakan <i>multithreading</i>
<code>def compute_similarity_for_feature(feature, queries):</code>	Subprogram untuk menghitung similaritas antara features yang telah di preprocess sebelumnya, dengan feature dari file MIDI yang diquery

<code>def compare(features, queries):</code>	Subprogram controller untuk melakukan komparasi similaritas antara features yang telah di preprocess sebelumnya, dengan feature dari file MIDI yang diquery, Program dijalankan dengan menggunakan <i>multithreading</i>
<code>def get_similarities(sorted_results, threshold=0):</code>	Subprogram untuk mengembalikan list tupel ( <code>nama_file, similaritas</code> ) yang memenuhi syarat similaritas diatas threshold. Hasil dari <code>get_similarities</code> akan dikirim menuju <code>main.py</code> untuk ditampilkan pada website

## ALUR KERJA PROGRAM

```

directory_path = <directory dimana dataset disimpan>
query_path = <directory dimana file MIDI query disimpan>

preprocess_result = process_all_midi_files_concurrently(directory_path)

query_notes = get_midi_notes(query_path)
queries = get_feature(query_notes)

sorted_similarity = compare(preprocess_result, queries)
result = get_similarities(sorted_similarity)

```

## PENJELASAN :

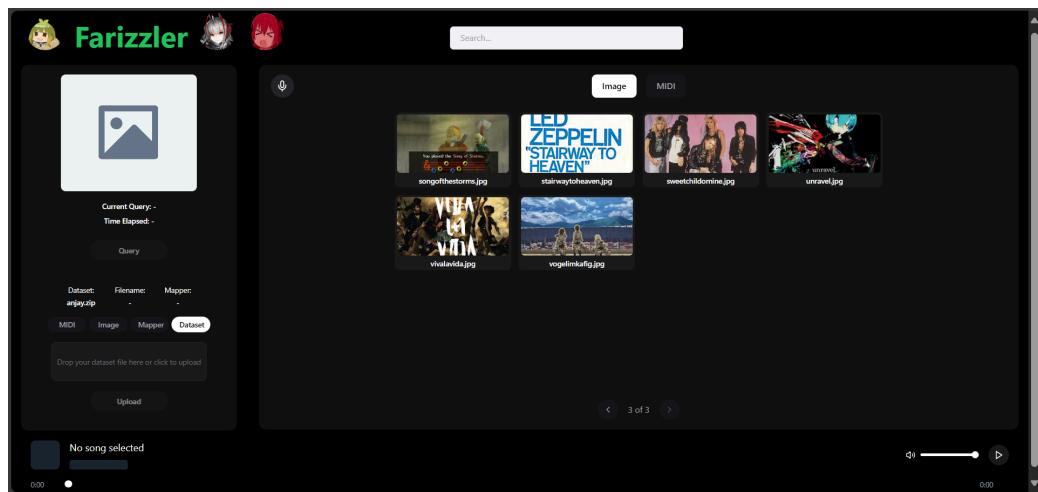
Program akan dipanggil oleh [main.py], dimana main.py dapat menerima masukan directory dataset serta path image yang diquery oleh user saat user melakukan upload

## BAB IV : EKSPERIMENT

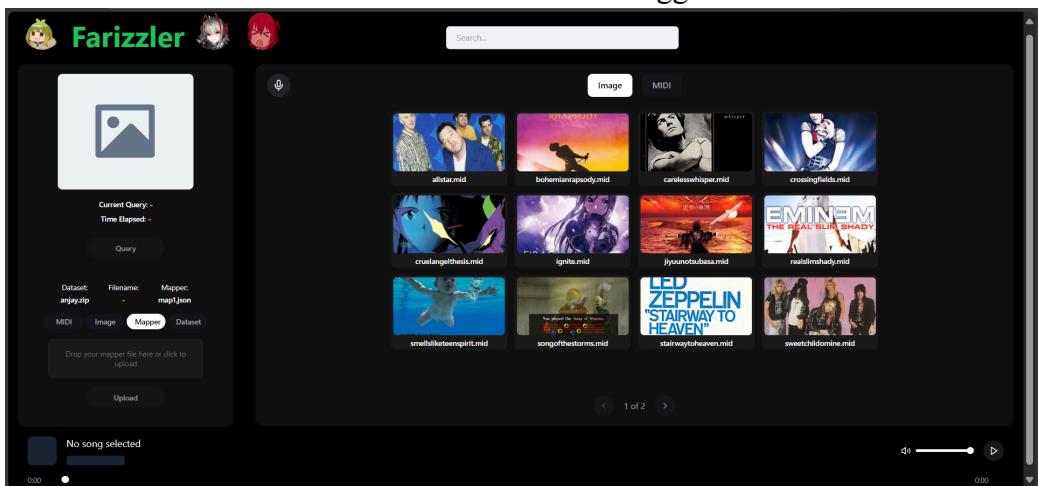
### 4.1 Tata Cara Penggunaan Program

- Query dengan upload file
  1. Unggah dataset
  2. Unggah mapper
  3. Unggah file yang ingin digunakan untuk query (audio MIDI atau gambar jpg, jpeg, dan png)
  4. Pada gallery, pilih tab Image untuk query berdasarkan gambar atau MIDI untuk query berdasarkan audio
  5. Tekan tombol query untuk melakukan query
  6. Hasil query dengan persentase kemiripan akan ditunjukkan di gallery
- Query dengan humming
  1. Unggah dataset
  2. Unggah mapper
  3. Tekan tab MIDI untuk mengaktifkan query dengan audio
  4. Tekan tombol mikrofon di pojok kiri atas gallery
  5. Mulai *humming* atau menyenandungkan lagu yang ingin dicari
  6. Hasil query dengan persentase kemiripan akan ditunjukkan di gallery
  7. Untuk memutar lagu
- Cara memutar lagu yang dipilih
  1. Pilih lagu yang ingin diputar dari galeri
  2. Klik tombol play di sudut kanan bawah untuk memutar lagu
  3. Klik tombol pause untuk menjeda lagu
  4. Jika ingin mengatur volume lagu yang diputar, sesuaikan *slider* volume yang ada di samping tombol play

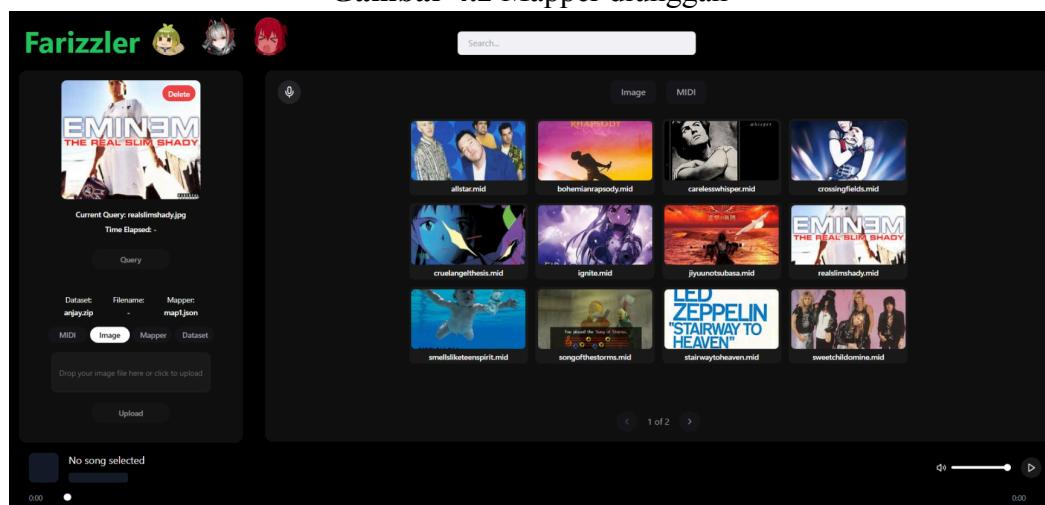
### 4.2 Uji pada Dataset dengan Gambar dan MIDI



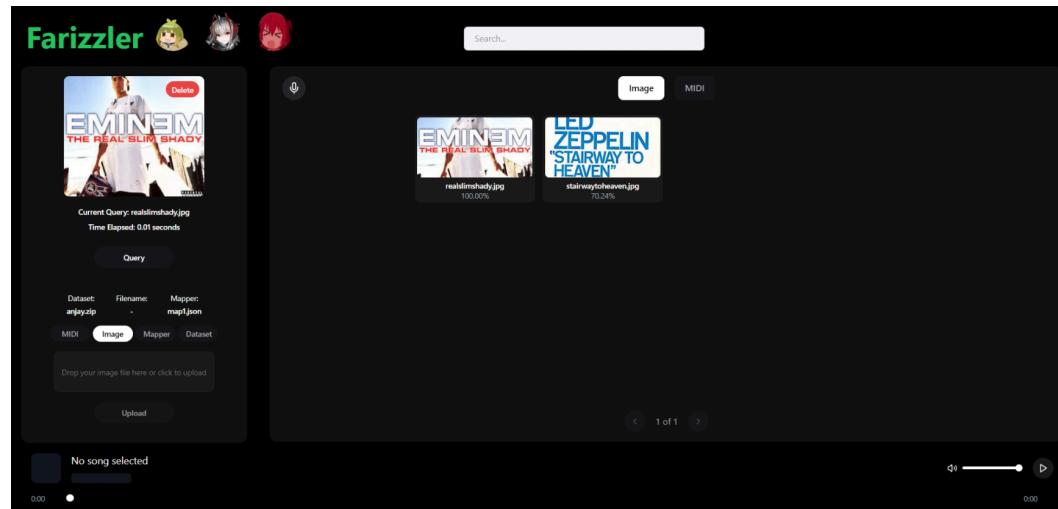
Gambar 4.1 Dataset diunggah



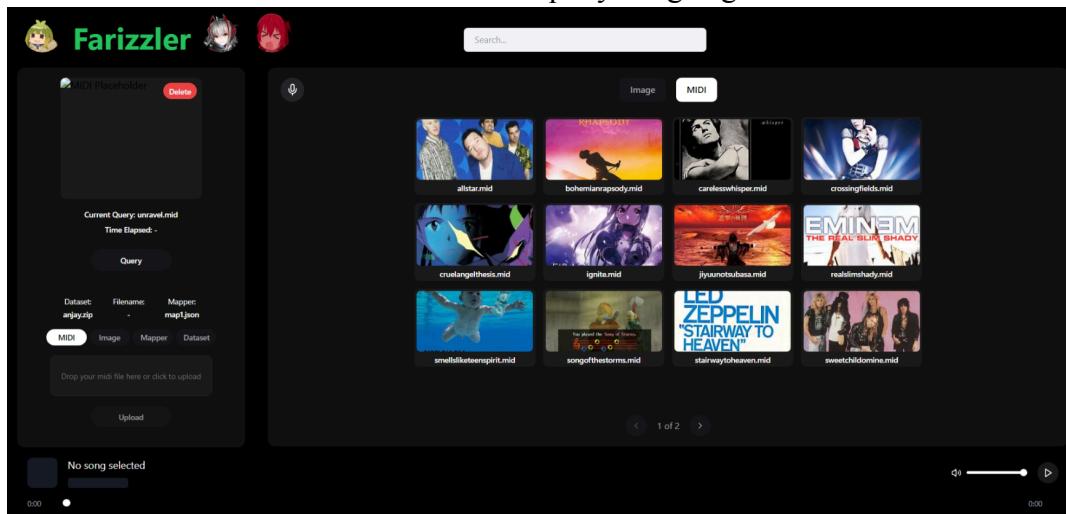
Gambar 4.2 Mapper diunggah



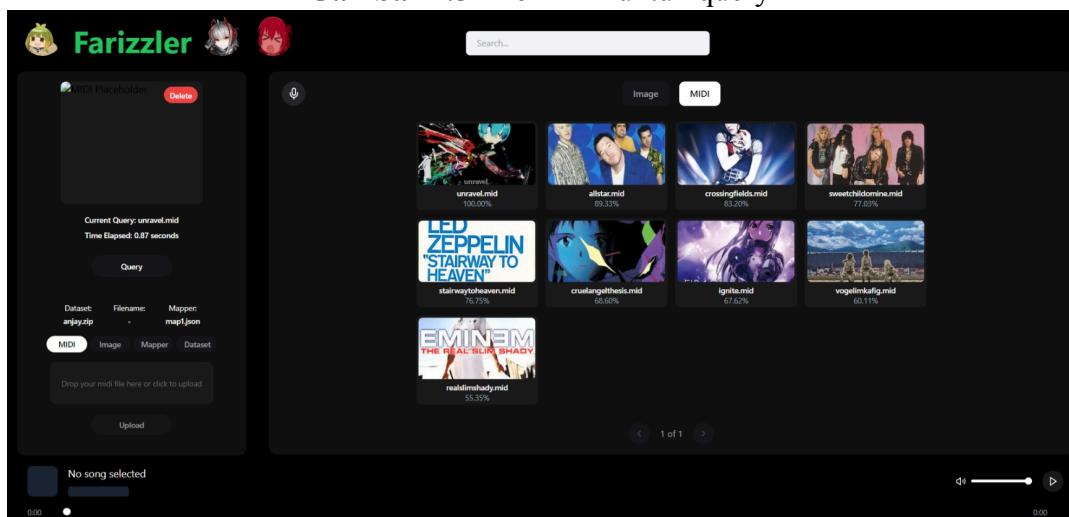
Gambar 4.3 File gambar untuk query diunggah



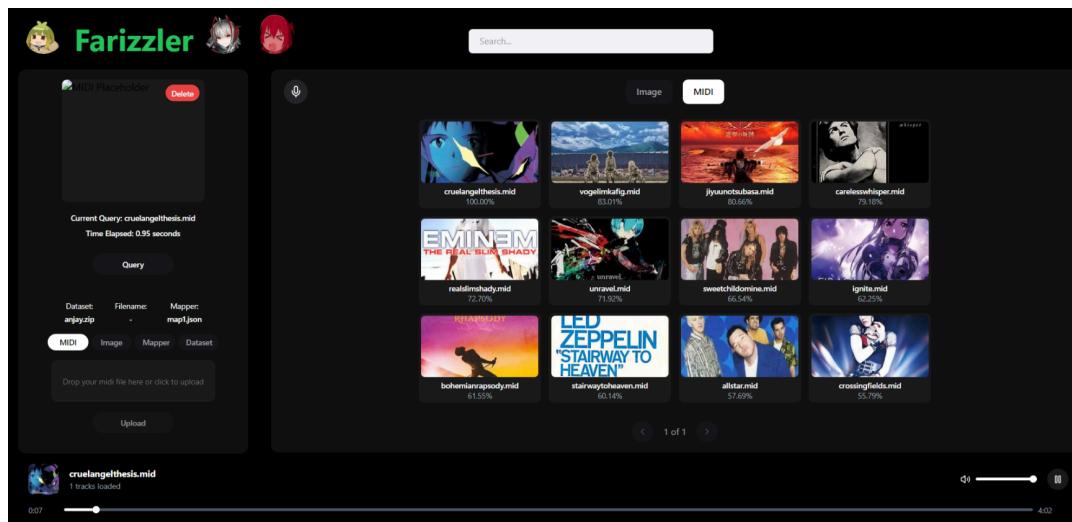
Gambar 4.4 Berhasil query dengan gambar



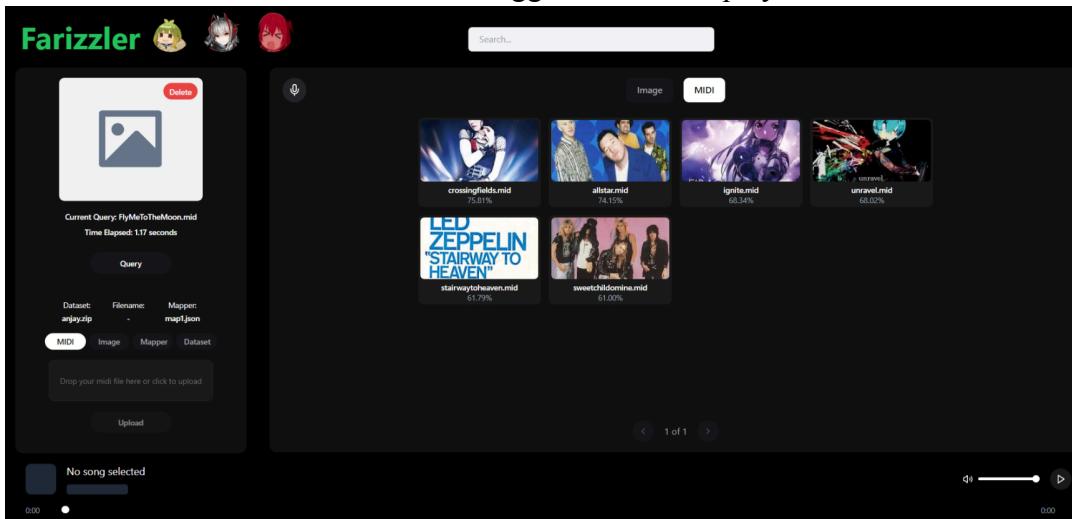
Gambar 4.5 File MIDI untuk query



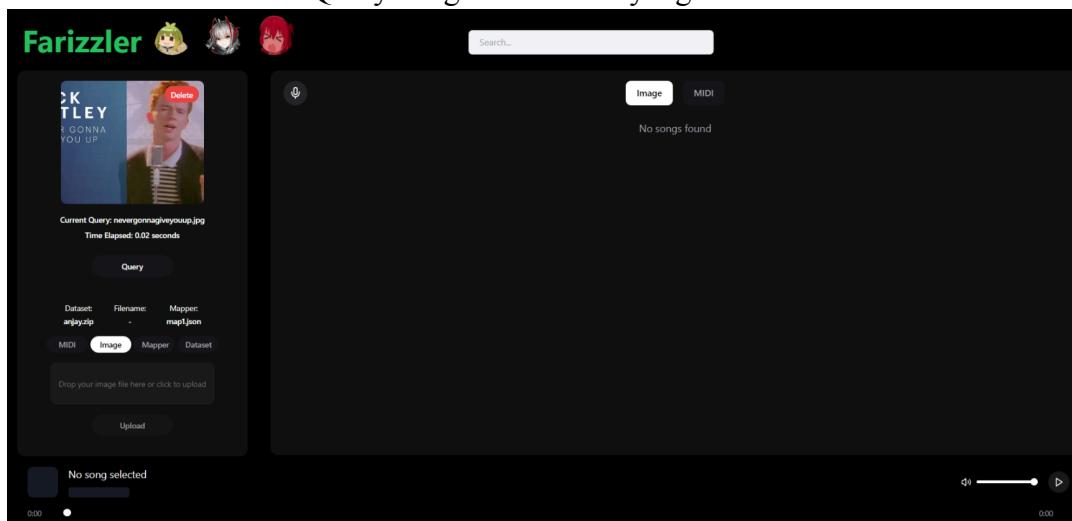
Gambar 4.6 Berhasil query dengan file MIDI



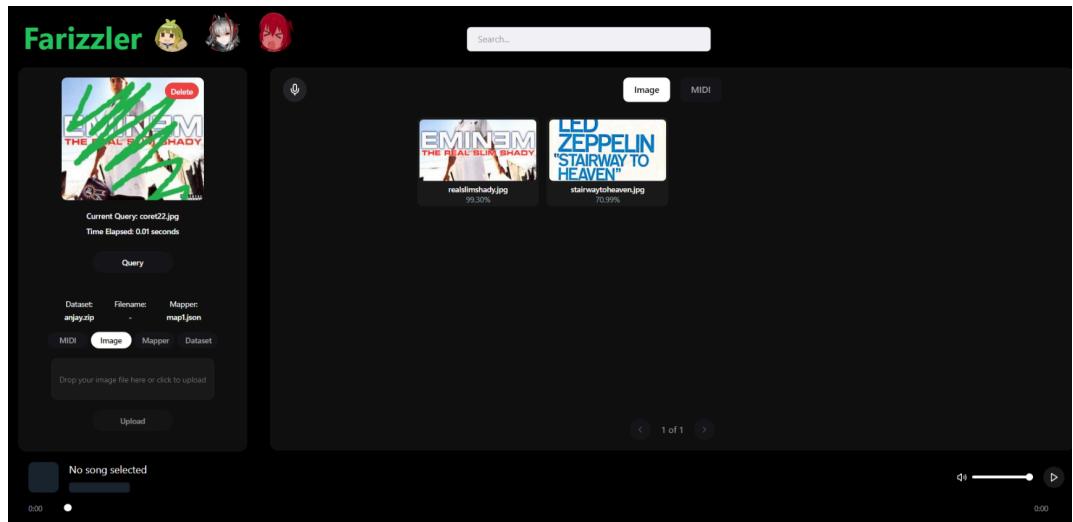
Gambar 4.7 Menggunakan audio player



Gambar 4.8 Query dengan file MIDI yang tidak ada di dataset



Gambar 4.9 Query dengan gambar yang tidak ada di dataset



**Gambar 4.10** Query dengan gambar yang ada di dataset tetapi sudah dicoret-coret

### 4.3 Analisis dari Desain Solusi Pemrosesan Gambar dan Suara

Berdasarkan percobaan kami pada sistem temu balik gambar, persentase kemiripan antara gambar-gambar yang berada di dalam dataset dengan gambar yang di query berada di sekitar angka 55% - 100%. Dikarenakan hal tersebut, kelompok kami memutuskan untuk memasang threshold kemiripan sebesar 75%. Gambar yang ditampilkan pada website merupakan gambar-gambar yang memiliki tingkat kemiripan di atas 75%. Apabila gambar yang diquery ada di dalam database, maka gambar tersebut akan memiliki tingkat kemiripan di angka 99.9% - 100%. Apabila gambar yang diquery merupakan modifikasi dari gambar yang terdapat di dataset, dimana modifikasi berupa sedikit coretan, tingkat kemiripan gambar tersebut masih berada di sekitar angka 90-95%. Apabila gambar yang diquery merupakan modifikasi dari gambar yang terdapat di dataset, dimana modifikasi berupa coretan yang banyak, dan/atau disertai dengan adanya perubahan warna, maka persentase kemiripan menurun signifikan di angka 75-85%. Apabila gambar yang di query merupakan gambar yang tidak terdapat pada dataset dan bukan merupakan modifikasi dari salah satu gambar dalam dataset, didapat bahwa gambar-gambar dengan persentase tertinggi, tidak memiliki kemiripan yang signifikan dengan gambar yang diquery, akan tetapi gambar-gambar dengan tema warna yang mirip. Hal tersebut dapat terjadi dikarenakan pada pemrosesan gambar, terdapat langkah yang melakukan konversi gambar menjadi gambar hitam-putih.

Waktu yang dibutuhkan oleh program kelompok kami dalam melakukan pre proses 5000 gambar

dengan total ukuran 110 mb adalah sekitar 5 detik. Angka tersebut didapat ketika pengujian backend local, tanpa integrasi dengan frontend. Akan tetapi, ketika dicoba pada back-end yang diintegrasikan dengan front-end, waktu yang diperlukan berada di angka sekitar 10 detik.

Berdasarkan percobaan kami pada sistem temu balik suara, persentase kemiripan antara audio-audio yang berada di dalam dataset dengan audio yang diquery berada di sekitar 30% - 100%. Berdasarkan hal tersebut, kelompok kami memutuskan untuk memasang threshold untuk file audio midi sebesar 55%, sehingga audio-audio yang ditampilkan pada website merupakan audio dengan tingkat kemiripan di atas 55%. Ketika program dijalankan untuk melakukan query audio yang terdapat di dataset, maka program tersebut akan mengembalikan audio yang sama dengan tingkat kemiripan 99.9% - 100%. Ketika program dijalankan untuk melakukan query audio yang tidak terdapat di dataset, 5 audio teratas yang dikembalikan oleh program memiliki tingkat kemiripan di angka 80-85%. Dalam dataset yang diberikan, terdapat beberapa file MIDI yang *corrupt*, apabila file corrupt tersebut dijadikan sebagai query, maka audio tidak diproses. Pada terminal backend, terdapat pesan error berupa “HTTP Exception 500”

Waktu yang dibutuhkan oleh program kelompok kami dalam melakukan pre proses sekitar 17000 file audio midi dengan total ukuran sekitar 700 mb adalah sekitar 220 detik. Angka tersebut didapat ketika pengujian backend local, tanpa integrasi dengan frontend. Waktu yang dibutuhkan untuk melakukan satu kali query adalah sekitar 30 detik.

## BAB V : KESIMPULAN

### 5.1 Kesimpulan

#### 1. Sistem Temu Balik Gambar

- Tingkat kemiripan gambar yang diquery dengan gambar dalam dataset bervariasi antara **55% hingga 100%**, dengan gambar yang identik memiliki tingkat kemiripan **99.9% - 100%**.
- Gambar yang merupakan modifikasi ringan memiliki tingkat kemiripan di kisaran **90% - 95%**, sedangkan modifikasi berat (misalnya banyak coretan dan perubahan warna) menghasilkan tingkat kemiripan lebih rendah, di kisaran **75% - 85%**.
- Gambar yang tidak terdapat dalam dataset cenderung menghasilkan output gambar dengan tema warna yang mirip, karena proses konversi gambar ke hitam-putih mempengaruhi hasil temu balik.
- Threshold kemiripan ditetapkan sebesar **75%**, sehingga hanya gambar dengan tingkat kemiripan di atas angka tersebut yang ditampilkan pada website.
- Waktu pre-prosesing **5000 gambar (110 MB)** adalah **5 detik** pada pengujian backend lokal. Namun, integrasi dengan frontend meningkatkan waktu proses menjadi **10 detik**.

#### 2. Sistem Temu Balik Suara

- Tingkat kemiripan audio yang diquery dengan audio dalam dataset bervariasi antara **30% hingga 100%**, dengan audio identik memiliki tingkat kemiripan **99.9% - 100%**.
- Query audio yang tidak terdapat dalam dataset menghasilkan 5 audio teratas dengan tingkat kemiripan di kisaran **80% - 85%**.
- Beberapa file MIDI yang corrupt tidak dapat diproses, menyebabkan error **“HTTP Exception 500”** pada backend.
- Threshold kemiripan untuk audio ditetapkan sebesar **55%**, sehingga hanya audio dengan tingkat kemiripan di atas angka tersebut yang ditampilkan pada website.
- Waktu pre-prosesing **17000 file MIDI (700 MB)** adalah **220 detik** pada backend lokal. Waktu yang dibutuhkan untuk satu kali query adalah **30 detik**.

### 3. Efisiensi Sistem

- Sistem temu balik gambar memiliki efisiensi waktu yang relatif lebih baik dibandingkan sistem temu balik suara, baik dalam proses pre-prosesing maupun query.
- Pengintegrasian backend dengan frontend berdampak signifikan terhadap waktu eksekusi pada kedua sistem.

### 4. Kendala dan Potensi Peningkatan

- Konversi gambar ke hitam-putih mengurangi akurasi temu balik untuk gambar yang memiliki warna dominan.
- Penanganan file corrupt pada dataset suara perlu ditingkatkan agar sistem tidak menghasilkan error yang memengaruhi kestabilan backend.
- Waktu query untuk sistem suara dapat dioptimalkan lebih lanjut guna meningkatkan pengalaman pengguna.

### 5.2 Saran dan Refleksi

Dalam pengerjaan tugas besar ini, diperlukan ketelitian dalam pembacaan spek sehingga tidak salah dalam implementasinya. Selain itu, perlu dieksplor lebih banyak skema atau kondisi sehingga semua kondisi dapat ditemukan penyelesaian yang tepat (*error handling*). Dalam pengerjaan tugas besar ini, kurangnya komunikasi dapat menyebabkan miskomunikasi dalam progres masing-masing anggota, sehingga kedepannya hal tersebut perlu kami perbaiki. Melalui tugas besar ini, kami memahami pentingnya kemampuan kerja sama, koordinasi, dan komunikasi yang baik.

### 5.3 Komentar

- Fariz :

وَإِمَّا تَحَافَنَ مِنْ قَوْمٍ خَيَانَةً فَأُنْبِذْ إِلَيْهِمْ عَلَى سَوَاءٍ إِنَّ اللَّهَ لَا يُحِبُّ الْخَائِنِينَ

And if you 'O Prophet' see signs of betrayal by a people, respond by openly terminating your treaty with them. Surely Allah does not like those who betray"

- Adha : semoga saya tidak rasis lagi

- Fajar : saya hampir keluar (dari jadwal penggerjaan)
- Mengingat banyaknya tugas besar yang sedang berjalan, alangkah baiknya apabila tenggat waktu dari penggerjaan tugas besar dapat diperlama.
- Ada banyak sekali ruang untuk perbaikan dari penggerjaan tugas besar kali ini. Koordinasi yang lebih baik dapat meningkatkan kinerja.

## DAFTAR PUSTAKA

Institut Teknologi Bandung. (2024). *Tugas Besar 2 IF2123 Aljabar Linier dan Geometri: Image Retrieval dan Music Information Retrieval Menggunakan PCA dan Vektor Semester I Tahun 2024/2025*. Sekolah Teknik Elektro dan Informatika, Program Studi Teknik Informatika.

Informatika.stei.itb.ac.id. (2023). Review matriks. Diakses pada 15 Desember 2024, dari <https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-01-Review-Matriks-2023.pdf>

University of Victoria (2015), Music Information Retrieval, Canada Tier II Research Chair in Computer Analysis of Audio and Music Department of Computer Science

<https://mido.readthedocs.io/en/stable/>, terakhir diakses pada 15 Desember 2024

## LAMPIRAN

Link repository: <https://github.com/Fajar2k5/Algeo02-23027>

Link video: <https://linktr.ee/algeo25>