

**LAPORAN TUGAS BESAR 2**  
**IF2123 ALJABAR LINEAR DAN GEOMETRI**  
Aplikasi Aljabar Vektor dalam Sistem Temu Balik Gambar



Disusun oleh:

Benardo	(13522055)
Albert	(13522081)
Derwin Rustanly	(13522115)

**Program Studi Teknik Informatika  
Sekolah Teknik Elektro dan Informatika  
Institut Teknologi Bandung  
2023**

## Daftar Isi

<b>Daftar Isi</b>	<b>i</b>
<b>BAB I</b>	<b>1</b>
1. Latar Belakang	1
2. Tujuan	1
3. Spesifikasi Program	1
<b>BAB II</b>	<b>3</b>
2. Sistem Temu Balik Informasi (Content Based Information Retrieval System)	5
3. Kesamaan Cosinus (Cosine Similarity)	9
B. Pengembangan Website	10
<b>BAB III</b>	<b>13</b>
1. Langkah-Langkah Pemecahan Masalah	13
2. Proses Pemetaan masalah menjadi elemen-elemen pada aljabar geometri	16
3. Contoh ilustrasi kasus dan penyelesaian	17
<b>BAB IV</b>	<b>20</b>
1. Implementasi Program Utama	20
2. Penjelasan Struktur Program	23
3. Penjelasan Tata Cara Penggunaan Program	26
4. Hasil Pengujian	27
5. Analisis	42
<b>BAB V</b>	<b>44</b>
1. Kesimpulan	44
2. Saran	44
3. Komentar	44
<b>Daftar Pustaka</b>	<b>46</b>
<b>LAMPIRAN</b>	<b>47</b>

## **BAB I**

### **Deskripsi Masalah**

#### **1. Latar Belakang**

Dalam era digital, jumlah gambar yang dihasilkan dan disimpan semakin meningkat dengan pesat, baik dalam konteks pribadi maupun profesional. Peningkatan ini mencakup berbagai jenis gambar, mulai dari foto pribadi, gambar medis, ilustrasi ilmiah, hingga gambar komersial. Terlepas dari keragaman sumber dan jenis gambar ini, sistem temu balik gambar (image retrieval system) menjadi sangat relevan dan penting dalam menghadapi tantangan ini. Dengan bantuan sistem temu balik gambar, pengguna dapat dengan mudah mencari, mengakses, dan mengelola koleksi gambar mereka. Sistem ini memungkinkan pengguna untuk menjelajahi informasi visual yang tersimpan di berbagai platform, baik itu dalam bentuk pencarian gambar pribadi, analisis gambar medis untuk diagnosis, pencarian ilustrasi ilmiah, hingga pencarian produk berdasarkan gambar komersial.

#### **2. Tujuan**

Tujuan penggeraan tugas besar ini adalah:

- 2.1. Mengimplementasikan Sistem Temu Balik Gambar Berbasis Konten (*Content Based Information Retrieval; CBIR*) dengan parameter warna dan tekstur berdasarkan prinsip-prinsip aljabar vektor.
- 2.2. Menyusun program berupa aplikasi berbasis *website* yang mengimplementasikan Sistem Temu Balik Gambar sebagaimana tertera pada poin 2.1.

#### **3. Spesifikasi Program**

Adapun spesifikasi dari program yang akan diimplementasikan adalah sebagai berikut.

- 3.1. Program menerima input *folder dataset* dan sebuah citra gambar.
- 3.2. *Dataset* gambar diunduh secara mandiri melalui pranala [berikut](#). Akan tetapi, juga digunakan *dataset* lain yang telah dipersiapkan oleh kelompok secara mandiri.
- 3.3. Program menampilkan gambar citra gambar yang dipilih oleh pengguna.
- 3.4. Program dapat memberikan kebebasan pada pengguna untuk memilih parameter pencarian yang hendak digunakan (warna atau tekstur) melalui *toggle*. *Default* parameter yang digunakan di awal dibebaskan kepada masing-masing kelompok.
- 3.5. Program mulai melakukan perhitungan nilai kecocokan antara *image* masukan dengan *dataset image* berdasarkan parameter yang telah dipilih (warna atau tekstur).
- 3.6. Program dapat menampilkan nilai kecocokan antara *image* masukan dengan setiap gambar dalam dataset.

- 3.7. Program menampilkan hasil luaran dengan melakukan *descending sorting* berdasarkan nilai kecocokan tiap gambar. Program menampilkan seluruh gambar yang **memiliki tingkat kemiripan > 60%** dengan gambar masukan.
- 3.8. Program mengimplementasikan *pagination* agar jumlah gambar dapat dibatasi dengan halaman-halaman tertentu. Jumlah gambar dalam dataset yang akan digunakan oleh asisten saat penilaian mungkin berbeda dengan jumlah gambar pada dataset yang kalian gunakan, sehingga pastikan bahwa *pagination* dapat berjalan dengan baik.
- 3.9. Program dapat menampilkan **Jumlah gambar** yang memenuhi kondisi tingkat kemiripan serta **waktu eksekusi**.

## BAB II

### Teori Singkat

#### A. Dasar teori yang digunakan secara umum

##### 1. Citra

Citra merujuk pada representasi, kemiripan, atau tiruan dari suatu objek. Sebagai hasil dari sistem perekaman data, citra dapat memiliki sifat optik seperti foto, sifat analog seperti sinyal video pada monitor televisi, atau sifat digital yang dapat disimpan langsung pada media penyimpanan. Meskipun citra mengandung banyak informasi, seringkali citra yang dimiliki mengalami penurunan kualitas, contohnya berupa cacat atau kebisingan. Hal ini membuat interpretasi citra menjadi lebih sulit karena informasi yang disampaikan oleh citra tersebut menjadi terbatas. Secara umum, suatu citra dapat diartikan sebagai fungsi  $f(x, y)$  dengan ukuran M baris dan N kolom, di mana x dan y adalah koordinat (x, y) yang menunjukkan tingkat keabuan atau intensitas citra pada titik tersebut. Jika nilai x, y bersifat hingga (finite) dan bernilai diskrit, maka citra tersebut dapat dikategorikan sebagai citra digital. Beberapa fitur yang dapat diekstrak dari sebuah citra digital dapat dikategorikan menjadi 2 parameter, yakni parameter warna dan parameter tekstur.

###### 1.1. Parameter Warna

###### 1.1.1. Rona (*Hue*)

Hue, dalam konteks warna, merujuk pada dimensi atau atribut yang menentukan warna dasar atau "tint" dari suatu warna. Hue memberikan informasi tentang di mana warna tersebut terletak pada lingkup spektrum warna, seperti merah, hijau, biru, kuning, dan sebagainya. Secara umum, hue menggambarkan jenis warna tanpa memperhitungkan kecerahan (brightness) atau kejernihan (saturation). Dalam model warna dasar, seperti model warna RGB (Red, Green, Blue) atau model warna HSL/HSV (Hue, Saturation, Lightness/Value), hue diukur dalam derajat pada lingkaran warna. Pada lingkaran warna ini, merah mungkin diwakili oleh 0 derajat, hijau oleh 120 derajat, biru oleh 240 derajat, dan seterusnya.

###### 1.1.2. Saturasi (*Saturation*)

Saturasi adalah atribut dalam warna yang mengukur sejauh mana suatu warna dari suatu objek tampak penuh atau "kaya" dalam warna tersebut. Dalam konteks model warna, seperti model warna HSL (Hue, Saturation, Lightness) atau HSV (Hue, Saturation, Value), saturasi mengacu pada intensitas warna relatif terhadap warna yang paling murni pada spektrum warna. Sebuah warna dengan saturasi tinggi akan terlihat lebih cerah dan kaya,

sedangkan warna dengan saturasi rendah akan tampak lebih pudar atau abu-abu. Saturasi memberikan informasi tentang kekuatan dan kejemuhan warna tanpa memperhatikan tingkat kecerahan atau jenis warna dasarnya. Misalnya, pada tingkat saturasi maksimum, warna merah akan tampak sebagai merah murni tanpa campuran warna lain, sementara pada tingkat saturasi yang lebih rendah, merah dapat tampak lebih lembut atau berubah menjadi warna merah muda. Saturasi berguna dalam mengekspresikan variasi warna dan mempengaruhi persepsi visual terhadap kecerahan dan intensitas warna suatu objek atau gambar.

#### 1.1.3. Nilai (*Value*)

Nilai (*Value*) merujuk pada tingkat kecerahan atau intensitas cahaya dari suatu warna. Komponen nilai menentukan sejauh mana warna tersebut tampak terang atau gelap. Nilai berkisar antara 0 hingga 1, di mana 0 menunjukkan warna yang sangat gelap atau hitam, sedangkan 1 menunjukkan warna yang sangat terang atau putih. Dengan mengubah nilai, kita dapat mengontrol tingkat kecerahan atau intensitas warna tanpa mengubah hue (jenis warna) atau saturasi (intensitas warna relatif terhadap warna murni). Misalnya, pada nilai 0, warna merah akan menjadi gelap atau hitam, sementara pada nilai 1, warna merah akan menjadi sangat terang atau putih. Pada nilai-nilai di antara 0 dan 1, kita dapat mencapai berbagai tingkat kecerahan, memungkinkan untuk menciptakan gradasi warna yang bervariasi.

### 1.2. Parameter Tekstur

#### 1.2.1. Kontras

Kontras dalam tekstur merujuk pada perbedaan atau perbandingan yang jelas antara elemen-elemen tekstur dalam suatu gambar atau objek. Dalam konteks visual, kontras tekstur mencerminkan sejauh mana elemen-elemen tekstur berbeda satu sama lain dalam hal kekasaran, pola, atau detail. Peningkatan kontras tekstur dapat membuat perbedaan antara area yang halus dan kasar lebih nyata, memperkuat ciri-ciri tekstur pada suatu permukaan. Sebaliknya, rendahnya kontras tekstur dapat membuat elemen-elemen tekstur tampak lebih seragam atau sulit dibedakan satu sama lain.

#### 1.2.2. Homogenitas

Homogenitas dalam tekstur merujuk pada sejauh mana elemen-elemen tekstur dalam suatu gambar atau objek bersifat seragam atau kurang bervariasi. Dalam konteks visual, gambar

atau permukaan yang homogen dalam tekstur akan memiliki elemen-elemen yang tampak serupa satu sama lain, tanpa perbedaan yang signifikan dalam hal kekasaran, pola, atau detail. Suatu daerah yang homogen dalam tekstur cenderung memiliki karakteristik yang konsisten dan kurang mencolok. Ini bisa berarti bahwa elemen-elemen tekstur memiliki tingkat kecerahan, warna, atau pola yang serupa, membuatnya sulit dibedakan atau membedakan detail.

#### 1.2.3. Disimilaritas

Disimilaritas dalam tekstur merujuk pada tingkat perbedaan atau ketidakseragaman antara elemen-elemen tekstur dalam suatu gambar atau objek. Ketika kita berbicara tentang disimilaritas dalam konteks tekstur visual, kita mengacu pada variasi atau kontras yang signifikan antara bagian-bagian yang berbeda dalam suatu permukaan atau gambar. Gambar atau permukaan yang memiliki tingkat disimilaritas tekstur yang tinggi cenderung menunjukkan perbedaan yang jelas dalam hal kekasaran, pola, atau detail antara berbagai area. Daerah-daerah yang memiliki disimilaritas tekstur tinggi mungkin menonjol karena perbedaan signifikan dalam karakteristik visual mereka.

#### 1.2.4. Entropi

Entropi dalam tekstur merujuk pada tingkat ketidakpastian atau kekacauan dalam distribusi elemen-elemen tekstur dalam suatu gambar atau objek. Dalam konteks tekstur visual, entropi mengukur sejauh mana elemen-elemen tersebut bervariasi atau tersebar secara acak. Jika suatu daerah memiliki tingkat entropi yang rendah, itu menunjukkan bahwa elemen-elemen tekstur cenderung seragam atau memiliki pola yang teratur. Sebaliknya, tingkat entropi yang tinggi menunjukkan bahwa elemen-elemen tekstur tersebar secara acak atau memiliki variasi yang signifikan.

## 2. Sistem Temu Balik Informasi (*Content Based Information Retrieval System*)

Content-Based Image Retrieval (CBIR) atau Sistem Pencarian Berbasis Konten pada Gambar adalah suatu pendekatan dalam bidang pengolahan citra dan temu kembali informasi yang fokus pada karakteristik dan konten visual dari suatu gambar. CBIR bertujuan untuk mencari gambar yang memiliki kesamaan visual dengan gambar referensi berdasarkan fitur-fitur seperti warna, tekstur, bentuk, atau distribusi spasial. Proses pencarian dalam CBIR dilakukan dengan membandingkan representasi fitur-fitur ini dari gambar referensi dengan gambar dalam koleksi data. Pendekatan ini sering digunakan dalam berbagai aplikasi,

termasuk manajemen koleksi gambar, identifikasi objek, dan pemrosesan medis, di mana pengguna dapat mencari gambar yang relevan dengan kebutuhan mereka tanpa mengandalkan metadata atau label manual. Salah satu keunggulan CBIR adalah kemampuannya untuk memberikan hasil pencarian berdasarkan konten visual tanpa bergantung pada anotasi atau label manual pada gambar. Meskipun demikian, tantangan utama dalam CBIR melibatkan pemodelan fitur-fitur yang efektif dan menciptakan metode pencocokan yang cepat dan akurat untuk memproses jumlah besar data gambar.

### 2.1. CBIR dengan parameter warna

Pada CBIR dengan parameter warna ini, akan dibandingkan *input* dari sebuah *image* dengan *image* yang dimiliki oleh dataset, hal ini dilakukan dengan cara mengubah *image* yang berbentuk RGB menjadi sebuah metode histogram warna yang lebih umum.

Histogram warna adalah frekuensi dari berbagai warna yang ada pada ruang warna tertentu hal ini dilakukan untuk melakukan pendistribusian warna dari *image*. Histogram warna tidak bisa mendeteksi sebuah objek yang spesifik yang terdapat pada *image* dan tidak bisa mendeskripsikan posisi dari warna yang didistribusikan.

Pembentukan ruang warna perlu dilakukan dalam rangka pembagian nilai citra menjadi beberapa *range* yang lebih kecil. Hal itu dilakukan untuk membuat sebuah histogram warna yang setiap interval tiap *range* dianggap sebagai *bin*. Histogram warna dapat dihitung dengan menghitung piksel yang menyatakan nilai warna pada setiap interval. Fitur warna mencakup histogram warna global dan histogram warna blok.

Pada perhitungan histogram, warna global HSV lebih dipilih karena warna tersebut dapat digunakan pada kertas (*background* berwarna putih) yang lebih umum untuk digunakan. Maka dari itu, perlu dilakukan konversi warna dari RGB ke HSV dengan prosedur sebagai berikut.

#### 1.1.1. Nilai dari RGB dinormalisasi dengan mengubah nilai *range* [0, 255] menjadi [0, 1]

$$R' = \frac{R}{255} \quad G' = \frac{G}{255} \quad B' = \frac{B}{255}$$

#### 1.1.2. Mencari Cmax, Cmin, dan $\Delta$

$$C_{max} = \max(R', G', B')$$

$$C_{min} = \min(R', G', B')$$

$$\Delta = C_{max} - C_{min}$$

1.1.3. Selanjutnya digunakan hasil perhitungan di atas untuk mendapatkan nilai HSV

$$H = \begin{cases} 0^\circ & \Delta = 0 \\ 60^\circ \times \left( \frac{G' - B'}{\Delta} \bmod 6 \right), C' \max = R' & \\ 60^\circ \times \left( \frac{B' - R'}{\Delta} + 2 \right), C' \max = G' & \\ 60^\circ \times \left( \frac{R' - G'}{\Delta} + 4 \right), C' \max = B' & \end{cases}$$

$$S = \begin{cases} 0 & C_{\max} = 0 \\ \frac{\Delta}{C_{\max}} & C_{\max} \neq 0 \end{cases}$$

$$V = C_{\max}$$

Setelah didapatkan nilai HSV dilakukan perbandingan antara *image* dari input dengan dataset dengan menggunakan *cosine similarity*

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

Dengan  $A$  dan  $B$  adalah vektor dan  $n$  adalah jumlah dimensi dari vektor. Tingkat kemiripan dihitung dari seberapa besar hasil dari *Cosine Similarity*.

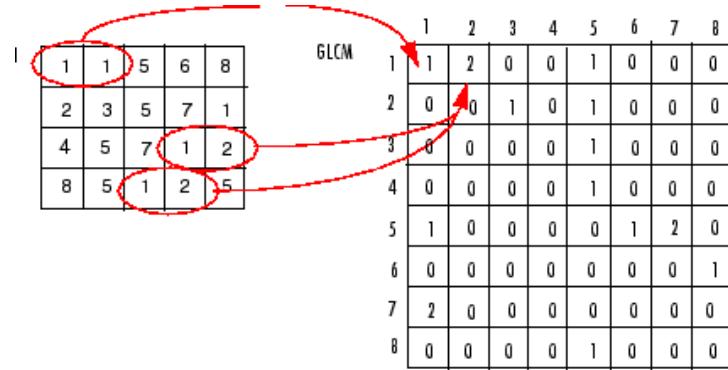
## 2.2. CBIR dengan parameter tekstur

CBIR dengan perbandingan tekstur dilakukan menggunakan suatu matriks yang dinamakan *co-occurrence matrix*. Matriks ini digunakan karena dapat melakukan pemrosesan yang lebih mudah dan cepat. Vektor yang dihasilkan juga mempunyai ukuran yang lebih kecil. Misalkan terdapat suatu gambar  $I$  dengan  $n \times m$  piksel dan suatu parameter offset ( $\Delta x, \Delta y$ ), Maka dapat dirumuskan matriksnya sebagai berikut:

Dengan menggunakan nilai  $i$  dan  $j$  sebagai nilai intensitas dari gambar dan  $p$  serta  $q$  sebagai posisi dari gambar, maka *offset*  $\Delta x$  dan  $\Delta y$  bergantung pada arah  $\theta$  dan jarak yang digunakan melalui persamaan di bawah ini.

$$C_{\Delta x, \Delta y}(i, j) = \sum_{p=1}^n \sum_{q=1}^m \begin{cases} 1, & \text{jika } I(p, q) = i \text{ dan } I(p + \Delta x, q + \Delta y) = j \\ 0, & \text{jika lainnya} \end{cases}$$

Melalui persamaan tersebut, digunakan nilai  $\theta$  adalah  $0^\circ$ .



Gambar 2. Cara Pembuatan Matrix *Occurrence*

Setelah didapat *co-occurrence matrix*, buatlah *symmetric matrix* dengan menjumlahkan *co-occurrence matrix* dengan hasil transpose-nya. Lalu cari *matrix normalization* dengan persamaan.

$$\text{MatrixNorm} = \frac{\text{MatrixOcc}}{\sum \text{MatrixOcc}}$$

Langkah-langkah dalam CBIR dengan parameter tekstur adalah sebagai berikut :

- 2.2.1. Konversi warna gambar menjadi *grayscale*, hal ini dilakukan karena warna tidaklah penting dalam penentuan tekstur. Oleh karena itu, warna RGB dapat diubah menjadi suatu warna *grayscale*  $Y$  dengan rumus:

$$Y = 0.29 \times R + 0.587 \times G + 0.114 \times B$$

- 2.2.2. Lakukan kuantifikasi dari nilai *grayscale*. Karena citra *grayscale* berukuran 256 piksel, maka matriks yang berkoresponden akan berukuran  $256 \times 256$ . Berdasarkan penglihatan manusia, tingkat kemiripan dari gambar dinilai berdasarkan kekasaran tekstur dari gambar tersebut.
- 2.2.3. Dari *co-occurrence matrix* bisa diperoleh 3 komponen ekstraksi tekstur, yaitu *contrast*, *entropy* dan *homogeneity*. Persamaan yang dapat digunakan untuk mendapatkan nilai 3 komponen tersebut antara lain :

*Contrast:*

$$\sum_{i,j=0}^{\text{dimensi}-1} P_{i,j} (i - j)^2$$

*Homogeneity :*

$$\sum_{i,j=0}^{\text{dimensi}-1} \frac{P_{i,j}}{1 + (i - j)^2}$$

*Entropy :*

$$-\left( \sum_{i,j=0}^{\text{dimensi}-1} P_{i,j} \times \log P_{i,j} \right)$$

**Keterangan :  $P$  merupakan matriks *co-occurrence***

Dari ketiga komponen tersebut, dibuatlah sebuah vektor yang akan digunakan dalam proses pengukuran tingkat kemiripan.

- 2.2.4. Ukurlah kemiripan dari kedua gambar dengan menggunakan Teorema *Cosine Similarity*, yaitu:

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

Disini  $A$  dan  $B$  adalah dua vektor dari dua gambar. Semakin besar hasil *Cosine Similarity* kedua vektor maka tingkat kemiripannya semakin tinggi.

### 3. Kesamaan Cosinus (*Cosine Similarity*)

Dalam analisis data, kesamaan cosinus (cosine similarity) merupakan salah satu parameter dalam mengukur derajat kesamaan di antara dua vektor non-nol yang didefinisikan pada ruang vektor tertentu. Kesamaan cosinus merupakan nilai cosinus di antara 2 vektor, yang didapatkan berdasarkan penurunan dari perkalian titik di antara kedua vektor tersebut. Untuk sembarang vektor non-nol berdimensi-n  $\mathbf{A}$  dan  $\mathbf{B}$ , perkalian titiknya dapat dinyatakan sebagai

$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \|\mathbf{B}\| \cos \theta$$

dengan  $\|\mathbf{A}\|$  dan  $\|\mathbf{B}\|$  berturut turut menyatakan panjang vektor  $\mathbf{A}$  dan  $\mathbf{B}$ , dan  $\cos \theta$  menyatakan cosinus sudut yang dibentuk antara kedua vektor tersebut.

Adapun, nilai kesamaan cosinus dapat di antara kedua vektor dapat dinyatakan sebagai berikut.

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

Pada umumnya, nilai dari kesamaan cosinus di antara sembarang vektor A dan B berada di dalam interval [-1,1]. Namun, dalam penentuan derajat kemiripan karakteristik dari vektor yang mewakili fitur-fitur gambar yang direpresentasikan oleh komponen Hue, Saturation, Value, Contrast, Homogeneity, dan Entropy, setiap piksel di dalam blok-blok gambar akan terkuantasisasikan terlebih dahulu dalam interval [0, 255], sehingga nilai kesamaan cosinus yang diperoleh berada di dalam interval [0,1].

## B. Pengembangan Website

Proses dan tahapan dalam pengembangan website:

### 1. Perencanaan

Proses pengembangan website dimulai dengan perencanaan yang menyeluruh. Kita menentukan tujuan dari website dan fitur-fitur apa saja yang akan dimasukkan. Ini termasuk merancang sketsa awal situs, menentukan bagaimana navigasi dan struktur halaman akan diatur. Pilihan teknologi yang akan digunakan juga diputuskan pada tahap ini, seperti bahasa pemrograman dan sistem manajemen konten.

### 2. Desain

Tahap ini berfokus pada desain visual website. Desainer UX/UI akan menciptakan desain yang tidak hanya menarik secara visual, tetapi juga mudah digunakan dan navigasi. Ini termasuk membuat wireframe (kerangka kerja dasar situs) dan mockup (rancangan lebih rinci), serta memilih skema warna dan jenis font yang akan digunakan.

### 3. Pengembangan

Setelah desain selesai, tahap pengembangan dimulai. Di sini, para pengembang frontend mengubah desain menjadi kode menggunakan HTML, CSS, dan JavaScript atau bisa juga menggunakan framework lain seperti ReactJs, VueJs , Angular, Svelte dll, sementara pengembang backend

menyiapkan server, dan menulis logika aplikasi. Integrasi API juga dilakukan pada tahap ini untuk dapat dihubungkan kepada frontend.

#### 4. Pengujian

Sebelum situs diluncurkan, penting untuk memastikan semuanya bekerja dengan baik. Pengujian ini meliputi mengecek fungsi website, memastikan situs responsif di berbagai perangkat, dan menemukan serta memperbaiki setiap bug yang ada.

#### 5. Peluncuran

Setelah semua pengujian selesai, situs siap untuk diluncurkan. Proses ini meliputi mengunggah situs ke server hosting dan mengatur domain. Optimisasi mesin pencari (SEO) juga dilakukan untuk memastikan situs mudah ditemukan di internet.

#### 6. Pemeliharaan dan Pembaruan

Pengembangan website tidak berhenti setelah peluncuran. Situs perlu diperbarui secara berkala, baik itu konten atau aspek teknisnya. Ini penting untuk menjaga situs tetap aman, up-to-date, dan relevan dengan kebutuhan pengguna.

Proses dan tahapan dalam pengembangan website ini memberikan gambaran menyeluruh tentang bagaimana sebuah website dibuat dari konseptualisasi awal hingga tahap akhir. Penting untuk memahami bahwa setiap tahapan memiliki kontribusi yang signifikan terhadap hasil akhir, memastikan bahwa website yang dihasilkan tidak hanya fungsional tetapi juga memenuhi kebutuhan pengguna dan tujuan bisnis.

Dalam proyek kami kali ini, kami telah berhasil melalui beberapa tahap penting, yaitu perencanaan, desain, pengembangan, dan pengujian. Setiap tahap ini dijalani dengan teliti dan detail, memastikan bahwa setiap aspek dari website dikembangkan dengan mempertimbangkan kualitas dan kegunaan. Dari proses perencanaan yang mendalam, desain interaktif dan intuitif, hingga pengembangan yang menggunakan teknologi terkini, kami telah mencapai tujuan untuk menciptakan produk digital yang siap diuji coba dan dievaluasi.

Namun, penting untuk dicatat bahwa dalam proyek ini, kami belum melangkah ke tahap peluncuran. Ini bukan berarti proyek ini tidak lengkap, melainkan sebuah strategi yang kami pilih untuk fokus pada inti pengembangan dan pengujian. Hal ini memberikan kami kesempatan untuk lebih mendalami dan menyempurnakan fitur-fitur utama sebelum melangkah ke tahap peluncuran. Proses ini juga memberikan kami kesempatan untuk

menerima umpan balik dan melakukan iterasi yang diperlukan, yang sangat penting dalam pengembangan produk digital.

Kesimpulannya, melalui proyek ini, kami mendapatkan pemahaman yang lebih dalam tentang proses pengembangan website, dari mulai perencanaan hingga pengujian. Kami belajar tentang pentingnya setiap tahap, dan bagaimana setiap keputusan dalam tahapan ini dapat berdampak pada hasil akhir. Pengalaman ini tidak hanya meningkatkan keahlian teknis kami, tetapi juga memberikan wawasan berharga tentang bagaimana membangun solusi digital yang efektif dan relevan dengan kebutuhan pasar dan pengguna saat ini.

## **BAB III**

### **Analisis Pemecahan Masalah**

#### **1. Langkah-Langkah Pemecahan Masalah**

Dalam mengatasi tantangan pada proyek Sistem Temu Balik Gambar, kami merinci langkah-langkah pemecahan masalah secara sistematis untuk memastikan pencapaian tujuan yang optimal. Berikut adalah langkah-langkah yang kami terapkan:

##### **1.1.Identifikasi Kebutuhan Pengguna**

Pertama-tama kami memahami kebutuhan pengguna dan fitur fitur apa saja yang dibutuhkan berdasarkan spesifikasi yang ditulis. Identifikasi kebutuhan pengguna menjadi langkah kritis dalam pengembangan sistem temu balik gambar. Analisis spesifikasi yang mendalam menjadi landasan kami, memastikan bahwa setiap fitur yang diinginkan, batasan sistem, dan kriteria kesamaan terdokumentasi dengan jelas.

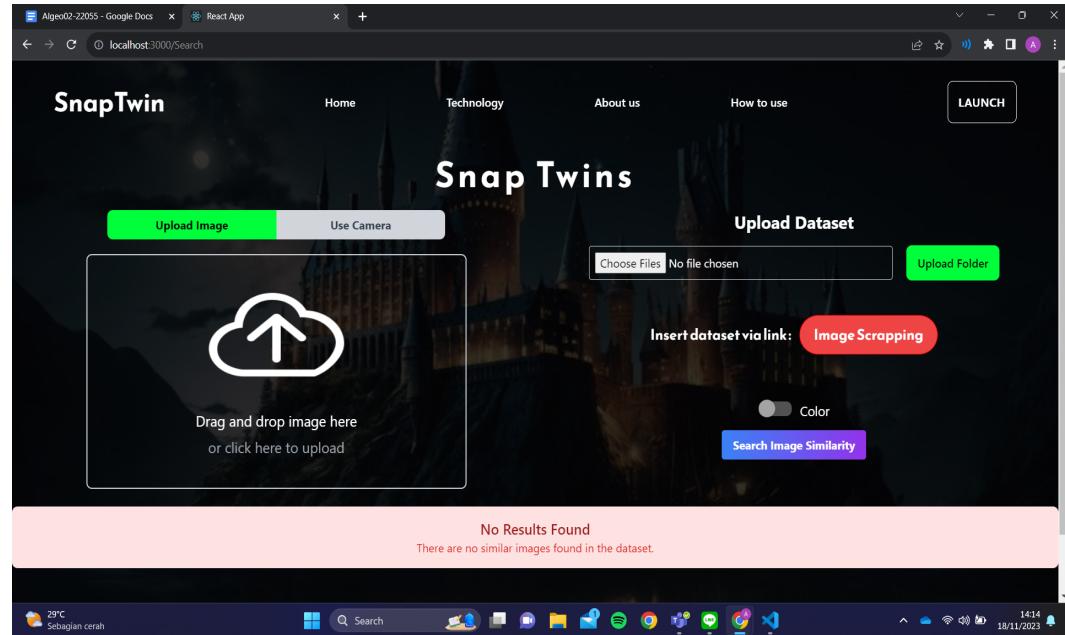
##### **1.2.Desain Antarmuka Pengguna (UI/UX)**

Setelah mendalam memahami kebutuhan pengguna dan merinci spesifikasi proyek yang sedang dijalankan, fokus selanjutnya terarah pada perancangan antarmuka pengguna yang memukau dan intuitif. Dalam proses perancangan antarmuka pengguna, kami memastikan bahwa setiap elemen desain memberikan nilai tambah fungsional dan memberikan kemudahan dalam interaksi pengguna. Antarmuka ini diatur sedemikian rupa sehingga memungkinkan pengguna dengan mudah mengunggah dataset mereka, langkah awal yang krusial dalam memulai pencarian kesamaan berbasis warna atau tekstur. Kami juga berfokus pada penyederhanaan proses pencarian itu sendiri. Desain intuitif kami memungkinkan pengguna untuk melakukan pencarian dengan cepat dan akurat, tanpa memerlukan pemahaman teknis yang mendalam. Tampilan website yang kami hasilkan didesain dengan kejelasan dan kegunaan sebagai prioritas utama, sehingga dapat diakses dan dimengerti oleh berbagai pengguna, termasuk yang mungkin tidak memiliki latar belakang teknis yang kuat.



### 1.3. Implementasi Algoritma Pencocokan

Pendekatan algoritmik diaplikasikan untuk mengatasi kompleksitas pencarian kesamaan berdasarkan warna dan tekstur. Proses implementasi ini melibatkan penerapan teknik pemrosesan gambar khusus untuk analisis warna dan tekstur, memastikan perbandingan fitur-fitur yang relevan dengan cermat. Dalam konteks pemrosesan gambar berwarna, kami mengekstrak informasi yang spesifik terkait dengan palet warna dari setiap gambar dalam dataset. Ini mencakup analisis histogram warna dan ekstraksi fitur warna, seperti hue, saturation, dan value, yang memberikan dasar yang kuat untuk representasi fitur warna yang akurat. Selanjutnya, pada aspek pencarian kesamaan berbasis tekstur, kami mengimplementasikan filter tekstur dengan parameter kontras, homogenitas, disimilaritas, dan entropi. Setelah berhasil mendapatkan representasi fitur dari setiap gambar dalam dataset, langkah selanjutnya adalah melakukan perbandingan dengan fitur-fitur gambar yang dicari. Untuk tujuan ini, kami memilih matriks perbandingan cosine similarity, yang memungkinkan evaluasi kesamaan antara fitur-fitur warna dan tekstur dari gambar yang dicari dengan dataset. Pendekatan ini memastikan bahwa hasil pencarian tidak hanya akurat tetapi juga relevan dengan kriteria warna dan tekstur yang diinginkan.



#### 1.4.Pengembangan Backend

Dalam tahap pengembangan backend, fokus kami tertuju pada pembangunan infrastruktur yang mendukung manajemen dataset yang efisien, penyimpanan data yang optimal, dan pengelolaan proses pencarian yang responsif. Mengadopsi framework Flask dengan bahasa pemrograman Python sebagai fondasi backend memberikan fleksibilitas dan kehandalan yang diperlukan untuk proyek ini. Dengan menggunakan Flask, kami telah membuat serangkaian endpoint yang dapat diakses dari frontend. Endpoint ini berfungsi sebagai titik akses yang memungkinkan frontend mengirim dan menerima data, menciptakan antarmuka yang efektif antara komponen frontend dan backend

#### 1.5.Integrasi Sistem

Integrasi sistem menjadi langkah krusial setelah pengembangan backend, desain antarmuka pengguna, dan implementasi algoritma pencocokan. Pada tahap ini, fokus utama adalah menciptakan keterhubungan yang harmonis antara frontend dan backend, sehingga seluruh sistem dapat beroperasi secara bersatu dan memberikan pengalaman pengguna yang seimbang.

Pertama-tama, dilakukan pemanggilan endpoint yang telah ditetapkan dari bagian frontend ke backend. Proses ini menciptakan jalur komunikasi yang efektif, memungkinkan frontend untuk berinteraksi dengan berbagai layanan yang disediakan oleh backend. Pemanggilan endpoint ini dirancang untuk mencakup segala aspek, mulai dari pengelolaan dataset hingga proses pencarian, sehingga seluruh fungsi sistem dapat diakses dan dikelola dengan lancar. Selanjutnya,

pengaturan penerimaan data dari backend ke frontend menjadi fokus utama dalam langkah integrasi ini. Data yang dikirim dari backend, termasuk hasil pencarian kesamaan berbasis warna atau tekstur, diintegrasikan secara rapi ke antarmuka pengguna. Kami memastikan bahwa data ini ditampilkan secara informatif dan mudah dimengerti, memberikan pengguna pemahaman yang jelas tentang hasil pencarian mereka. Melalui integrasi sistem yang cermat, setiap komponen, mulai dari antarmuka pengguna hingga algoritma pencocokan, dapat berinteraksi secara sinergis.

#### 1.6.Optimasi Sistem

Selama tahap optimasi sistem, perhatian utama tertuju pada identifikasi dan perbaikan bagian-bagian program yang belum mencapai tingkat efisiensi yang optimal. Proses ini melibatkan analisis mendalam terhadap kinerja keseluruhan sistem untuk mengidentifikasi area-area yang mungkin menjadi bottleneck atau menyebabkan penurunan kinerja. Langkah pertama adalah mencari program-program yang dapat dianggap sebagai titik lemah dalam sistem. Ini melibatkan evaluasi kritis terhadap proses-proses yang mungkin memerlukan perbaikan, baik dalam hal kecepatan pencarian maupun efisiensi pengelolaan data. Setelah identifikasi, dilakukan serangkaian tindakan pengoptimalkan. Peningkatan kecepatan pencarian menjadi prioritas, dan strategi pengoptimalan melibatkan penyesuaian algoritma pencocokan, penggunaan indeks, atau perbaikan dalam implementasi algoritma.

## 2. Proses Pemetaan masalah menjadi elemen-elemen pada aljabar geometri

### A. CBIR dengan parameter warna

Dalam pendekatan Content-Based Image Retrieval (CBIR) dengan parameter warna, digunakan konsep aljabar geometri untuk merepresentasikan dan mengukur kesamaan antara warna gambar. Pendekatan dimulai dengan merepresentasikan setiap gambar sebagai vektor dalam ruang warna HSV. Representasi ini memungkinkan kita memisahkan informasi tentang nilai Hue, Saturation, dan Value dari setiap piksel gambar. Setiap gambar diinterpretasikan sebagai vektor dalam ruang warna HSV, di mana Hue (warna), Saturation (kejemuhan), dan Value (nilai) mewakili dimensi-dimensi vektor tersebut.

Setelah pemetaan ke koordinat aljabar geometri, digunakan metode cosine similarity untuk mengukur kesamaan antara dua warna. Dengan membandingkan vektor warna dari dua gambar menggunakan cosine similarity, kami dapat

menentukan sejauh mana dua warna tersebut serupa atau berbeda. Cosine similarity menghasilkan nilai yang mencerminkan sejauh mana dua vektor warna berada dalam arah yang sama dalam ruang geometri, menggambarkan tingkat kemiripan warna secara matematis. Dengan demikian, hubungan antara elemen-elemen aljabar geometri, seperti vektor dan cosine similarity, memberikan dasar analisis yang kuat untuk mengevaluasi kemiripan warna antara gambar-gambar dalam CBIR.

#### B. CBIR dengan parameter tekstur

Dalam konteks pemetaan masalah tekstur ke elemen-elemen aljabar geometri, langkah pertama adalah mengubah gambar menjadi citra grayscale dan merepresentasikan sebagai matriks tekstur. Setiap elemen matriks ini mencerminkan intensitas piksel gambar, memungkinkan kita untuk mewakili tekstur secara matematis. Matriks tekstur ini kemudian dipetakan ke koordinat dalam ruang geometri, di mana setiap elemen matriks memiliki posisi koordinat yang sesuai.

Proses selanjutnya melibatkan perhitungan kesamaan tekstur menggunakan metode cosine similarity. Matriks tekstur dari dua gambar dibandingkan untuk mengevaluasi sejauh mana dua tekstur tersebut serupa atau berbeda. Hasil perhitungan cosine similarity juga dapat diinterpretasikan sebagai sudut antara dua vektor tekstur dalam ruang geometri. Dengan demikian, pemetaan masalah tekstur ke elemen-elemen aljabar geometri memberikan dasar matematis untuk menganalisis kesamaan tekstur dengan pendekatan yang lebih terstruktur.

Melalui penggunaan vektor, matriks, proyeksi, dan cosine similarity, kami menggambarkan hubungan matematis antara warna dan tekstur dalam konteks CBIR. Konsep-konsep ini memberikan dasar analisis yang mendalam dan objektif terhadap kemiripan visual antara gambar-gambar dalam dataset. Dengan memanfaatkan elemen-elemen aljabar geometri, kami dapat mengukur dan memahami kemiripan antara gambar secara lebih terperinci, meningkatkan efektivitas sistem CBIR untuk pencarian berbasis warna dan tekstur.

### 3. Contoh ilustrasi kasus dan penyelesaian

#### A. CBIR dengan parameter warna

Langkah-langkah yang dilakukan ketika pemecahan masalah dalam menghitung kemiripan gambar dengan parameter warna yaitu:

Gambar yang dicari :



Setelah didapatkan matriks dari gambar harimau yang dicari

Dengan membandingkan matriks dari gambar yang dicari dengan gambar lainnya menggunakan metode cosine similarity, kita dapat mengukur persentase kemiripan antara keduanya. Semakin tinggi persentase kemiripan menandakan bahwa kedua gambar yang dibandingkan semakin mirip secara visual.

**Search Results:**  
Total Images: 5  
Search Duration: 1.90 seconds

The search results display five images of big cats arranged in two rows. The top row contains three images: a tiger on the left, a cheetah in the center, and another cheetah on the right. The bottom row contains two images: a lion on the left and a leopard on the right. Each image is accompanied by its similarity score relative to the query image.

Image Category	Similarity Score
Tiger	100%
Cheetah (Top)	68.53347%
Cheetah (Bottom)	63.50576%
Lion	62.9785%
Leopard	62.75918%

### B. CBIR dengan parameter tekstur

Langkah langkah yang dilakukan ketika pemecahan masalah dalam menghitung kemiripan gambar dengan parameter warna yaitu:

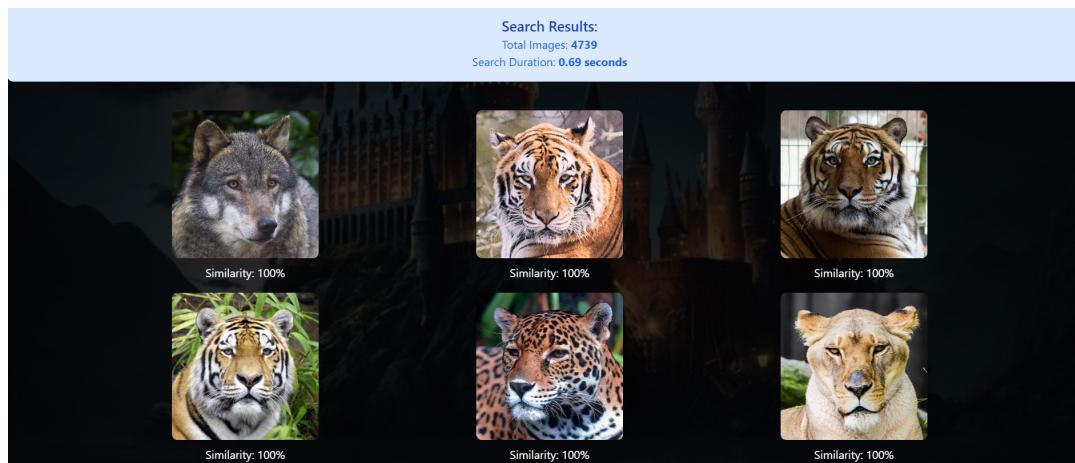
Mengubah gambar yang dicari dengan mengubah gambar menjadi grayscale :



Setelah konversi ke citra grayscale, gambar diekstraksi ke dalam bentuk matriks kejadian yang kemudian dinormalisasi. Hasilnya, terbentuk vektor yang merepresentasikan fitur-fitur kunci seperti kontras, disimilaritas, homogenitas, dan entropi dari gambar tersebut. Inilah data vektor yang dihasilkan dari proses ekstraksi fitur pada gambar.

```
"texture": [134.03272812221155, -6.820156172808774, 0.26457273131273273, 8.72391556974097]},
```

Kemudian akan dibandingkan dengan vektor-vektor dari dataset gambar yang telah diperoleh



## BAB IV

### Implementasi dan Uji Coba

#### 1. Implementasi Program Utama

##### 1.1. Implementasi Fungsi dan Prosedur CBIR *Color*

###### 1.1.1. Fungsi ImageBlockToHistogram

<pre>function ImageBlockToHistogram(image_path: string) → Array of Histogram</pre>
<b>Kamus Lokal</b> w, h, i, j: integer img.block: matrix of Image arrHist : array of Vector
<b>Algoritma</b> <pre> open(image_path, img) if (img.width &gt; 1080) then   w ← 1080   h ← (img.height/img.width * 1080)   img.resize(w,h) i traversal [0..3] j traversal [0..3]   block ← rgb_to_hsv(sliceImage((i)*img.width//4:(i+1)*img.width//4,   j*img.height//4:(j+1)*img.height//4, :))   insertLast(arrHist, makeHistogram(block)) → arrHist </pre>

###### 1.1.2. Fungsi rgb\_to\_hsv

<pre>function rgb_to_hsv(img: matrix of Image ) → matrix of Image</pre>
<b>Kamus Lokal</b> i, j: integer img: matrix of Image

<b>Algoritma</b> <pre> i traversal [0..img.width-1] j traversal [0..img.height-1]   img[i,j,0] ← img[i,j,0] / 255   img[i,j,1] ← img[i,j,1] / 255   img[i,j,2] ← img[i,j,2] / 255   prosesHSV(img)   hsv_to_hsvFeature(img) → img </pre>
---

###### 1.1.3 Prosedur prosesHSV

<pre>procedure prosesHSV(input/output pixel : array of RGB)</pre>
<b>Kamus Lokal</b> h,s,v,cmax,cmin,delta,r,g,b: integer

img: matrix of Image

```

Algoritma
    r ← pixel[0]
    g ← pixel[1]
    b ← pixel[2]
    cmax ← max(r,g,b)
    v ← cmax
    cmin ← min(r,g,b)
    delta ← cmax - cmin
    depend on (delta,cmax):
        delta = 0: h ← 0
        cmax = r: h ← 60*((g-b)/delta mod 6)
        cmax = g: h ← 60*((b-r)/delta + 2)
        cmax = b: h ← 60*((r-g)/delta + 4)
    if cmax = 0 then
        s ← 0
    else
        s ← delta/cmax
    pixel[0] ← h
    pixel[1] ← s
    pixel[2] ← v

```

#### 1.1.4. Prosedur hsv\_to\_hsvFeature

procedure hsv\_to\_hsvFeature(input/output pixel : array of RGB)

Kamus Lokal

h,s,v: integer

img: matrix of Image

Algoritma

```

    h ← pixel[0]
    s ← pixel[1]
    v ← pixel[2]
    depend on(h,s,v):
        316 ≤ h ≤ 360 or h = 0 : h ← 0
        1 ≤ h < 26 : h ← 1
        26 ≤ h < 41 : h ← 2
        41 ≤ h < 121 : h ← 3
        121 ≤ h < 191 : h ← 4
        191 ≤ h < 271 : h ← 5
        271 ≤ h < 296 : h ← 6
        295 ≤ h < 316 : h ← 7

```

```

        0 ≤ s < 0.2 : s ← 0
        0.2 ≤ s < 0.7 : s ← 1
        0.7 ≤ s ≤ 1 : s ← 2

```

```

        0 ≤ v < 0.2 : v ← 0
        0.2 ≤ v < 0.7 : v ← 1
        0.7 ≤ v ≤ 1 : v ← 2

```

```

    pixel[0] ← h
    pixel[1] ← s
    pixel[2] ← v

```

### 1.1.5 function makeHistogram

<pre>function makeHistogram(img) → Vector</pre>
<b>Kamus Lokal</b> hist : Vector i,j : integer
<b>Algoritma</b> CreateHistogram(hist) i traversal [0..img.width-1] j traversal [0..img.height-1] h ← img[i,j,0] s ← img[i,j,1] v ← img[i,j,2] hist[9*h+3*s+v] ← hist[9*h+3*s+v] + 1

## 1.2. Implementasi Fungsi dan Prosedur CBIR *Texture*

### 1.2.1 function convertImageToGrayScale

<pre>function convertImageToGrayScale(image_path) → Matrix of Image</pre>
<b>Kamus Lokal</b> img : Image w,h,R,G,B : integer img_np,grayImg : array
<b>Algoritma</b> open(image_path, img) if (img.width > 1080) then w ← 1080 h ← (img.height/img.width * 1080) img.resize(w,h)  if img.mode ≠ ‘RGB’ then img ← img.convert(‘RGB’)  i traversal [0..img.width-1] j traversal [0..img.height-1] R ← img[i,j,0] G ← img[i,j,1] B ← img[i,j,2] grayImg = 0.29*R + 0.587*G + 0.114*B → grayImg

### 1.2.2 function createOccurrenceMatrix

<pre>function createOccurrenceMatrix(grayImg) → Vector</pre>
<b>Kamus Lokal</b> occMat,normmat : Matrix sum: integer i_values, j_values,grayImg : array

```

Algoritma
    i traversal [0..img.width-1]
        j traversal [0..img.height-1]
            i_values ← grayImg[i,j]
            j_values ← grayImg[i,j]
    occMat ← occMat[i_values, j_values] +1
    normmat ← occMat.transpose + occMat
    sum ← totalElement(normmat)
    normmat ← normmat /sum
    → normmat

```

### 1.2.3 function *getTextureFeature*

```
function getTextureFeature(occMat) → array
```

#### Kamus Lokal

occMat : Matrix  
 dissimilarity, contrast, homogeneity, entropy, epsilon: float  
 i\_values, j\_values, grayImg : array

#### Algoritma

```

dissimilarity ← occMat* abs(i_values - j_values)
contrast ← occMat*((i_values - j_values)^2)
homogeneity ← occMat/ (1+ ((i_values - j_values)**2))
epsilon ← 1* 10**-10
entropy ← occMat* log(occMat + epsilon )
normmat ← normmat /sum
→ [contrast, -dissimilarity, homogeneity, -entropy]

```

## 1.3. Implementasi Fungsi Cosine Similarity

```
function cosineSimilarity(a,b : Vector)
```

#### Kamus Lokal

dot, length1, length2 : real

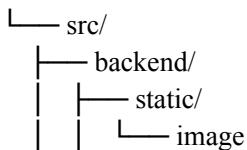
#### Algoritma

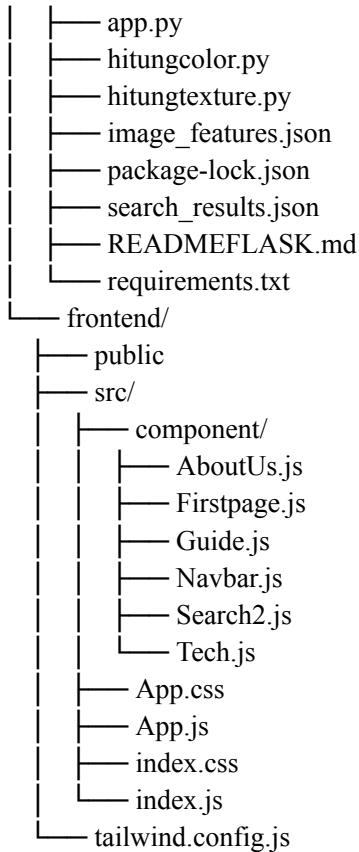
```

dot ← dotProduct(a,b)
length1 ← sqrt(dotProduct(a,a))
length2 ← sqrt(dotProduct(b,b))
return dot/(length1*length2)

```

## 2. Penjelasan Struktur Program





Struktur program dari aplikasi web ini terdiri dari dua buah folder yaitu folder frontend dan backend. Untuk frontend , kami menggunakan framework React + Tailwind CSS, dan untuk backend kami menggunakan framework Flask dan bahasa Pemrograman Python. Berikut adalah struktur dari program yang lebih terperinci,

1. **src/** : Direktori utama. Folder ini berisi dua buah Folder yaitu folder frontend dan backend.
2. **backend/** : Direktori yang menyimpan semua file program yang berkaitan dengan algoritma dan API dari website. Di dalam folder ini terdiri dari:
  - **Static / image /** : Merupakan direktori yang akan menyimpan dataset gambar yang diupload oleh user.
  - **App.py** : Merupakan file python yang memuat semua endpoint-endpoint yang akan dipanggil backend. Endpoint-endpoint ini menangani permintaan dari frontend atau pengguna dan mengembalikan respons sesuai dengan logika yang ditentukan dalam kode. File ini merupakan bagian penting dari struktur backend dalam banyak aplikasi yang berbasis Python, terutama yang menggunakan framework seperti Flask.
  - **hitungcolor.py**: Berisi algoritma yang digunakan untuk memproses suatu gambar menjadi sebuah histogram 72 elemen yang kemudian akan dikalikan dengan histogram lainnya untuk mendapatkan cosine similarity.

- **hitungtexture.py**: Berisi algoritma yang digunakan untuk memproses suatu gambar menjadi sebuah vektor yang elemen-elemen nya merupakan contrast, homogeneity, entropy dari gambar tersebut, yang kemudian akan dikalikan dengan vektor gambar lainnya untuk mendapatkan cosine similarity.
- **image\_features.json** : Merupakan suatu file yang akan menyimpan histogram dan vektor texture dari setiap gambar yang ada di dataset.
- **package-lock.json**: File ini berfungsi untuk mencatat versi spesifik dari setiap paket yang diinstal dalam proyek tersebut.
- **search\_result.json** : Menyimpan hasil dari pencarian yang dilakukan oleh user yang berisi nama file, persentase kesamaan, udan waktu pencarian.
- **READMEFLASK.md** : File ini berisi langkah-langkah yang diperlukan untuk menjalankan Flask untuk backend di website kita.
- **requirement.txt** : File ini berisi kebutuhan-kebutuhan backend yang perlu diinstall untuk menjalankan program.

**3. Frontend** : Direktori ini menyimpan semua file yang berkaitan dengan frontend , baik itu file js , maupun file css. Dalam direktori ini terdiri dari:

- **public** : Merupakan folder yang digunakan untuk menyimpan gambar-gambar yang akan digunakan dalam tampilan website.
- **src** : Merupakan direktori yang digunakan untuk menyimpan semua component-component ataupun halaman-halaman website. Dalam folder src ini terdiri dari :
  - **component** : Merupakan direktori yang menyimpan halaman-halaman dalam website kita. Terdiri dari:
    - **AboutUs.js** : Merupakan halaman tentang developer dari website ini.
    - **Firstpage.js** : Merupakan halaman yang merupakan tampilan pertama saat membuka website.
    - **Guide.js** : Merupakan halaman yang berisi panduan untuk menggunakan aplikasi web kami.
    - **Navbar.js** : Merupakan navigasi yang digunakan untuk routing ke halaman-halaman yang berbeda.
    - **Search2.js** : Merupakan halaman yang merupakan aplikasi untuk melakukan pencarian gambar yang mirip.
    - **Tech.js** : Merupakan halaman web yang berisi bahasa dan framework apa saja yang kami lakukan.
  - **App.css** : File ini digunakan untuk mendefinisikan gaya visual yang dari komponen-komponen dalam aplikasi, termasuk layout, warna, font, dan elemen desain lainnya.

- **App.js** : File ini bertindak sebagai komponen utama atau 'root component' dari aplikasi. Dalam struktur aplikasi berbasis komponen seperti React, App.js biasanya berfungsi sebagai wadah utama tempat komponen-komponen lain diimpor dan disusun. Ini termasuk layout, navigasi, dan bagian-bagian penting lain dari antarmuka pengguna.
  - **index.css** : File ini bertujuan untuk menentukan gaya visual dasar yang akan diterapkan pada keseluruhan halaman atau aplikasi web.
  - **index.js** : File ini berisi komponen utama aplikasi (seperti App dalam React) biasanya diimpor dan kemudian di render ke dalam DOM (Document Object Model) melalui library seperti ReactDOM. File ini juga mengurus router untuk pindah-pindah halaman.
4. **tailwind.config.css** : Dalam file ini, pengembang dapat mendefinisikan tema kustom, mengatur palet warna, font, ukuran, margin, padding, dan berbagai utilitas lainnya. File ini memungkinkan pengembang untuk meng extend atau meng override default Tailwind, sehingga mereka bisa menggunakan class-class utility Tailwind yang telah disesuaikan tanpa harus menulis banyak CSS tambahan.

### 3. Penjelasan Tata Cara Penggunaan Program

Untuk menjalankan program kita, user dapat mengikuti langkah-langkah di bawah ini :

- Setelah user berada di root program , user harus terlebih dahulu mengeketik ‘cd src’ di terminal, kemudian ‘cd backend’ .
- Setelah berada di path backend, user dapat membuat virtual environment bernama ‘env’ dengan menggunakan perintah “python -m venv env” di terminal.
- Setelah muncul folder env, pengguna kemudian dapat mengaktifkan virtual Environment dengan menggunakan perintah “env\Scripts\activate” di terminal untuk pengguna windows dan “env/bin/activate” untuk pengguna mac.
- Kemudian user dapat menginstall requirement yang dibutuhkan program dengan menggunakan perintah “pip install -r requirements.txt”
- Setelah berhasil diinstal, user dapat menjalankan backend dengan cara “flask run -- debug.
- Kemudian user dapat membuka terminal baru, lalu kemudian membuat perintah “cd src” , kemudian “cd frontend” .
- Setelah berhasil masuk di path frontend, user dapat menjalankan website dengan menggunakan perintah “npm run start”, setelah itu web akan terbuka dan bisa dijalankan.

Setelah website berhasil dijalankan, akan muncul tampilan homepage dari website kita, di bagian navbar terdapat beberapa pilihan page yang bisa dipilih antara lain technology page, about us page, how-to-use page , dan terakhir tombol “LAUNCH APP”. Jika anda ingin mengetahui langkah-langkah penggunaan website kami, anda dapat mengunjungi how-to-use page . Untuk melihat tech stack apa saja yang kami gunakan anda dapat mengunjungi halaman technology, lalu jika anda melihat developer yang membangun web kami, anda dapat mengunjungi page about-us. Terakhir, jika anda ingin menggunakan aplikasi kita, anda dapat mengklik tombol “LAUNCH APP”.

Setelah anda berhasil masuk ke halaman aplikasi, anda dapat terlebih dahulu, memasukkan foto yang akan anda telusuri. Setelah itu, anda perlu untuk mengunggah data set yang berisi kumpulan gambar. Untuk mengupload dataset, anda dapat memilih antara menggunakan web scraping atau dengan mengupload folder. Setelah mengupload folder , anda perlu mengklik “UPLOAD FOLDER”.Setelah muncul notifikasi berhasil di upload, anda dapat memilih anda ingin mencari kemiripannya berdasarkan apa , yaitu antara “texture” dan “color”. Setelah memilih, user dapat langsung melakukan pencarian dengan mengklik tombol “Search similarity” , setelah itu akan muncul gambar” yang memiliki similarity di atas 60%. Setelah hasil gambar semua keluar, kami menyediakan fitur download PDF , yang memuat semua hasil dari pencarian yang dilakukan, sehingga dapat mempermudah user.

Selain dengan mengupload foto, user juga dapat menggunakan fitur kamera yang dapat menangkap gambar setiap 10 detik , lalu akan menampilkan hasil yang sesuai dengan gambar tersebut.

#### 4. Hasil Pengujian

- **Dataset yang kami gunakan**

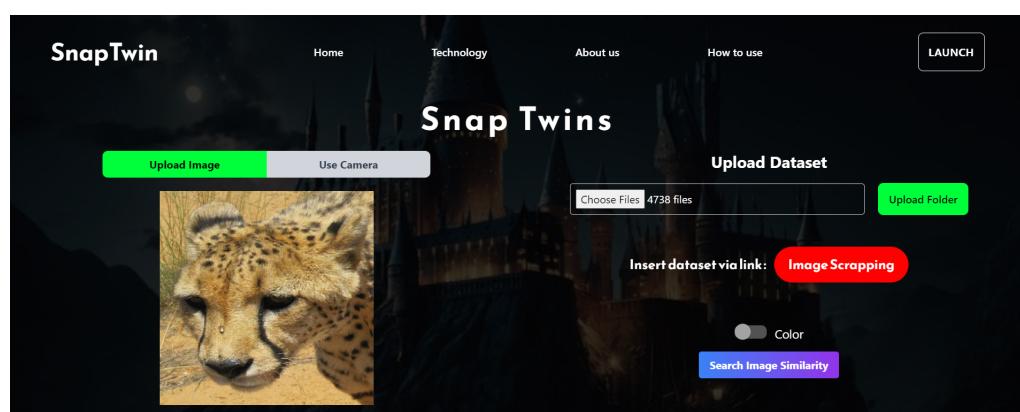
Pada pengujian kali ini, kami menggunakan dataset hewan yang diberikan dalam spesifikasi tugas besar algeo 2, yaitu sebuah dataset gambar yang berisi 4738 gambar di dalamnya yang terdiri dari berbagai jenis variasi hewan.

- **Pengujian color**

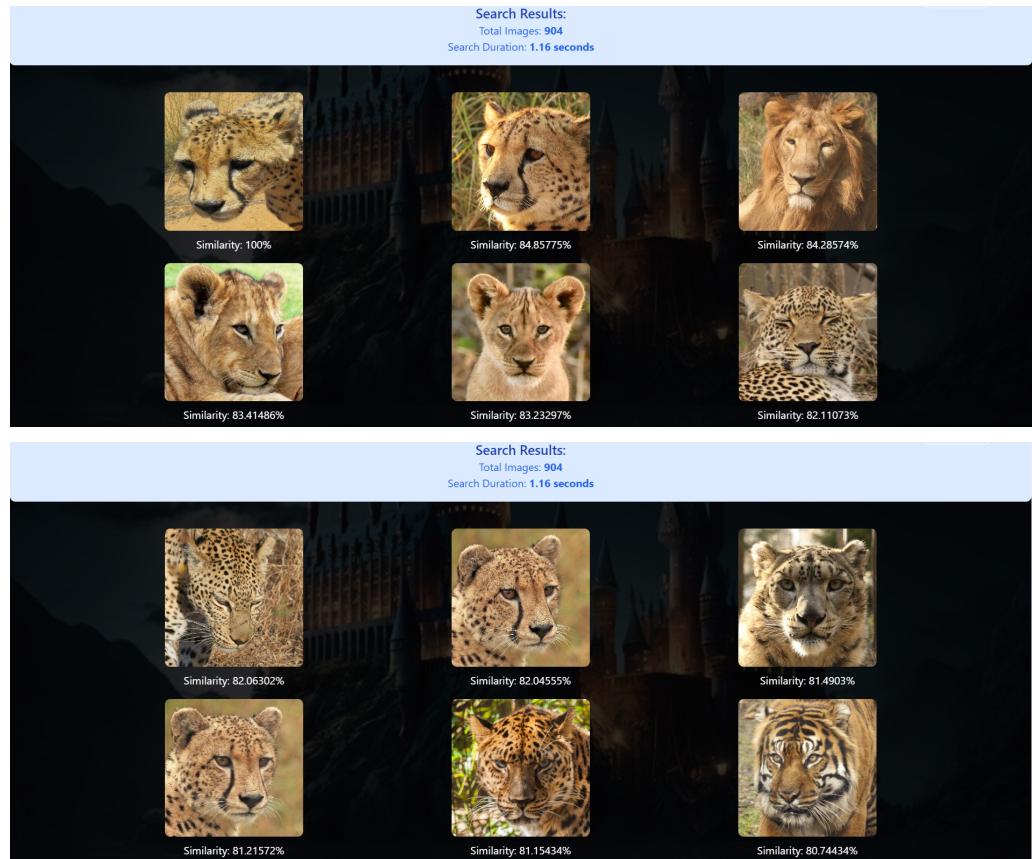
Berikut kami tampilkan 5 kali pengujian menggunakan pilihan by color:

1. Hasil Pencarian 1 (By color)

Pencarian :

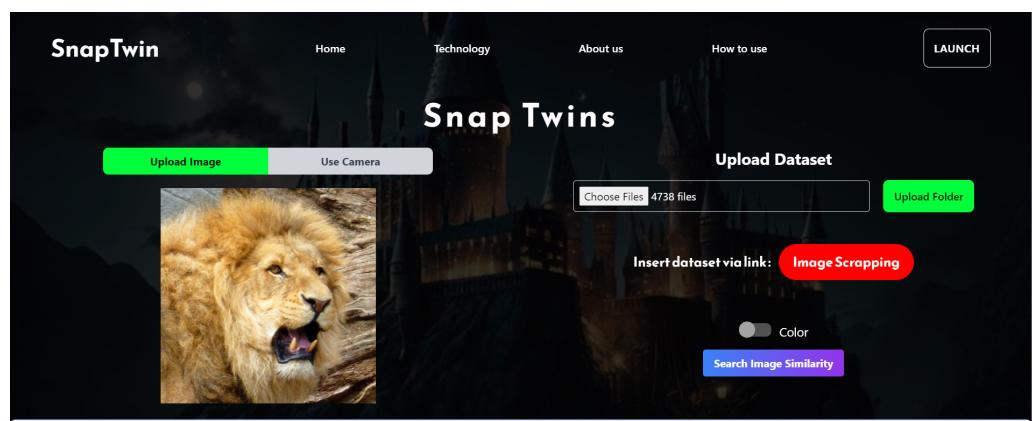


Hasil :

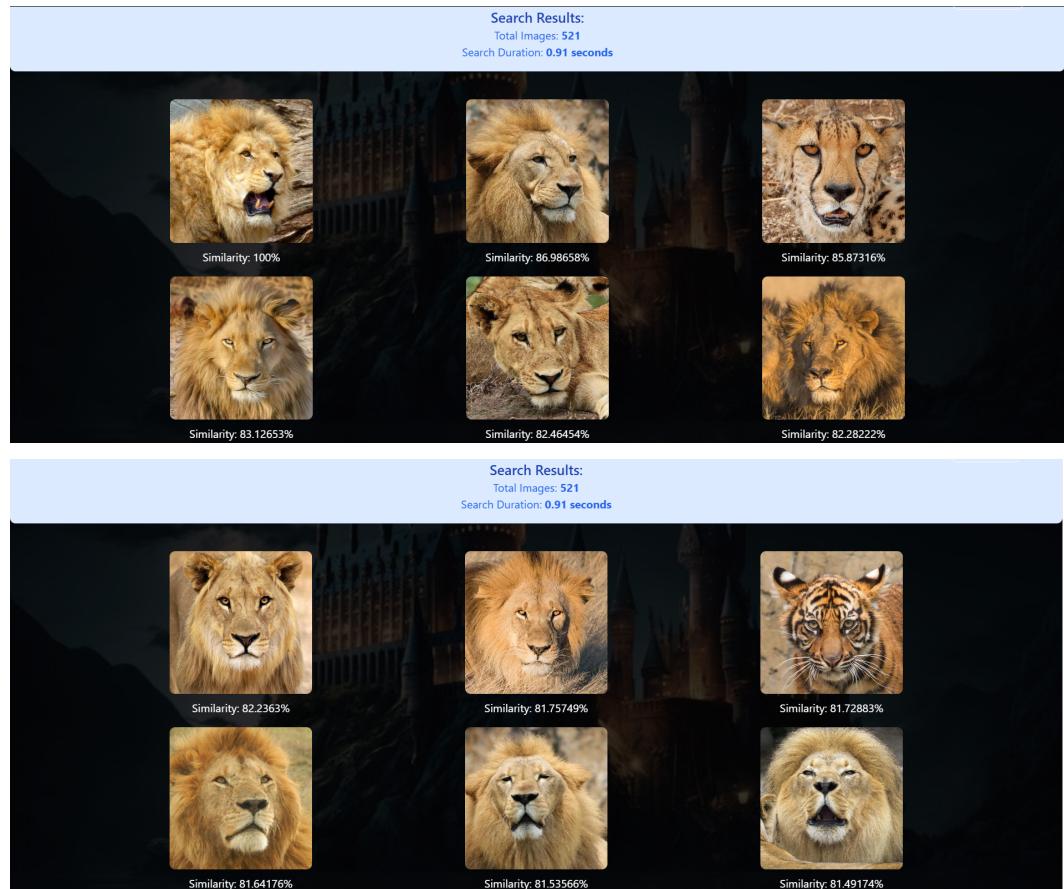


## 2. Hasil Pencarian 2 (By color)

Pencarian :

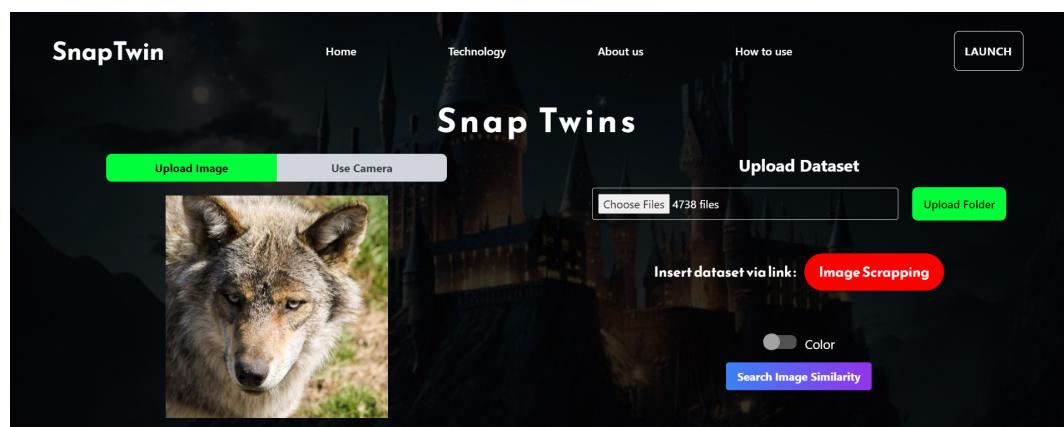


Hasil :

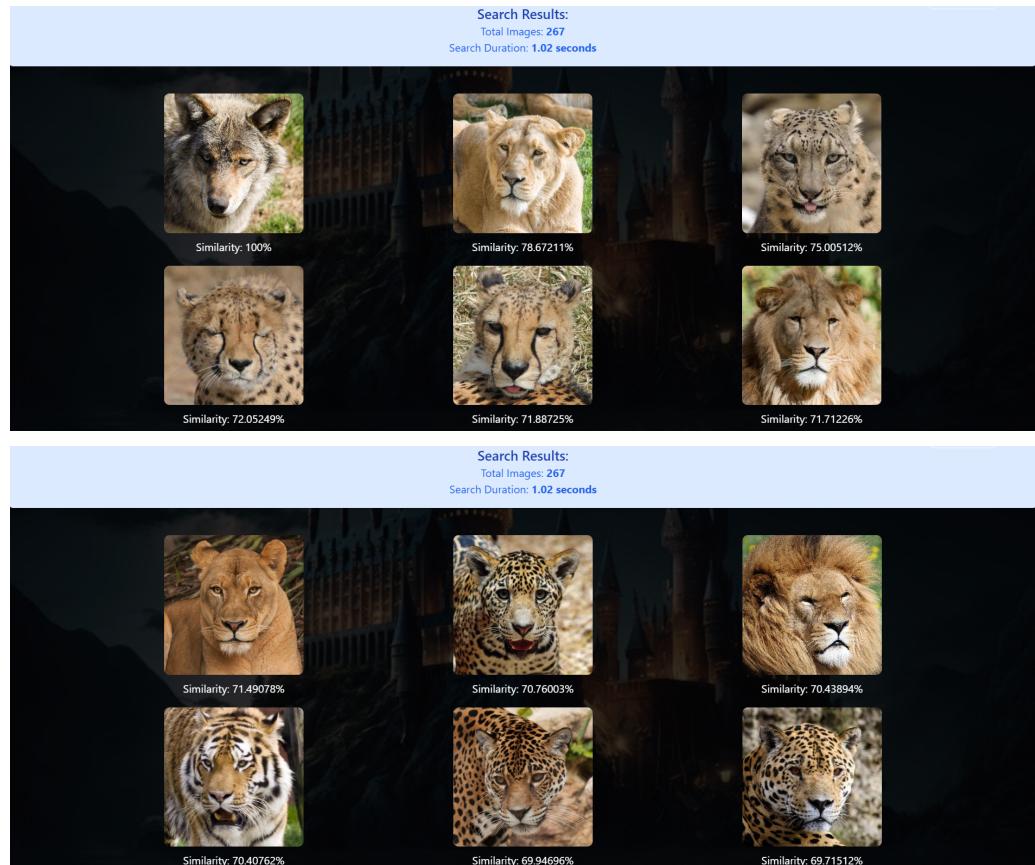


### 3. Hasil Pencarian 3 (By color)

Pencarian :

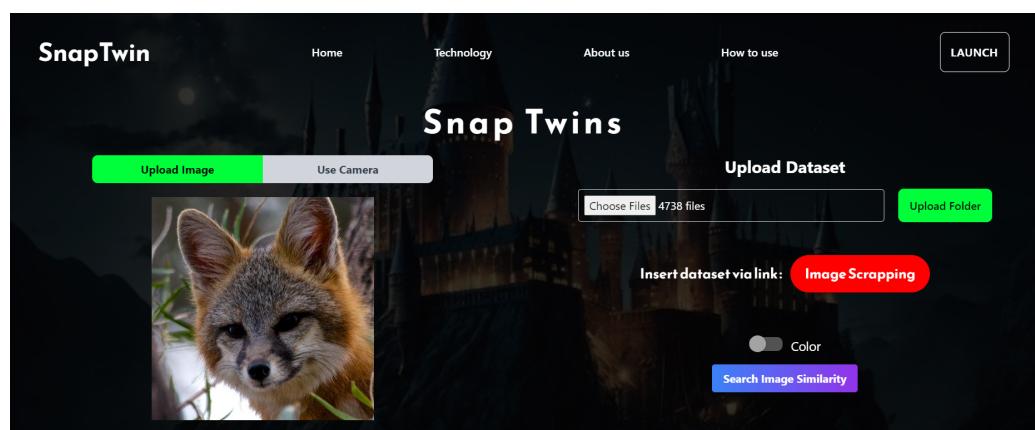


Hasil Pencarian :

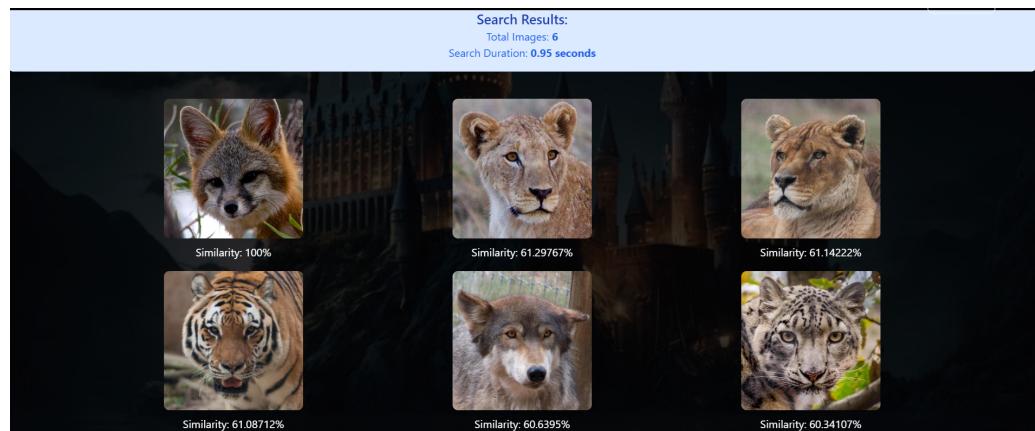


4. Hasil Pencarian 4 (By color)

Pencarian :

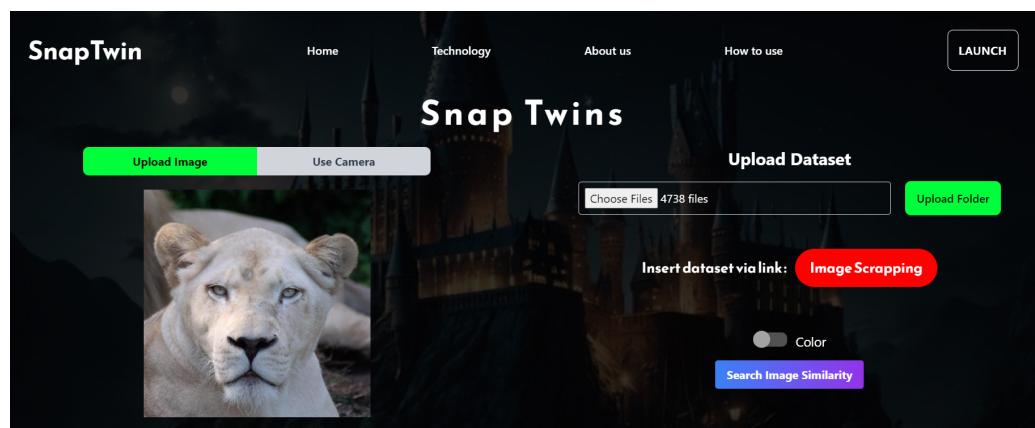


Hasil Pencarian :

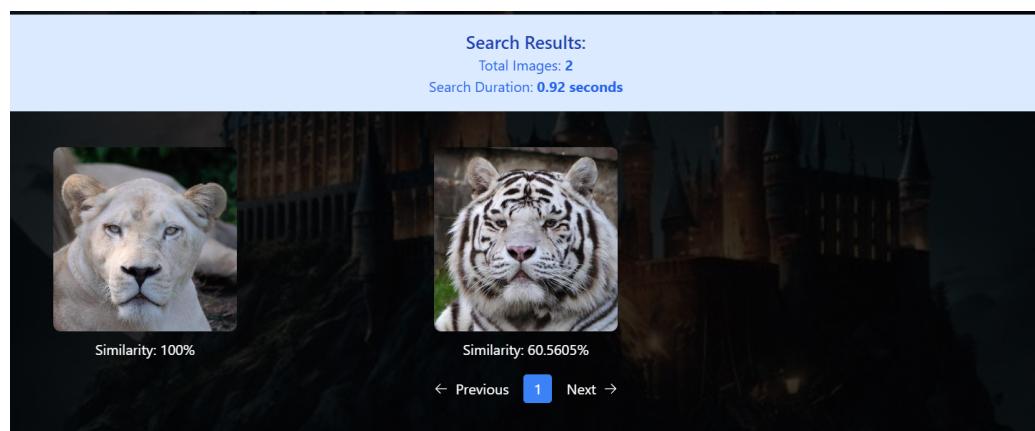


## 5. Hasil pencarian 5 (By color)

Pencarian :



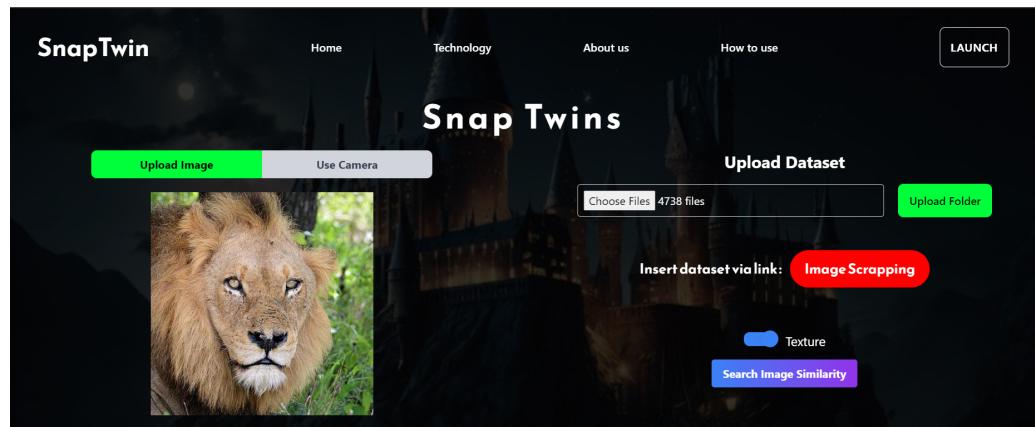
Hasil :



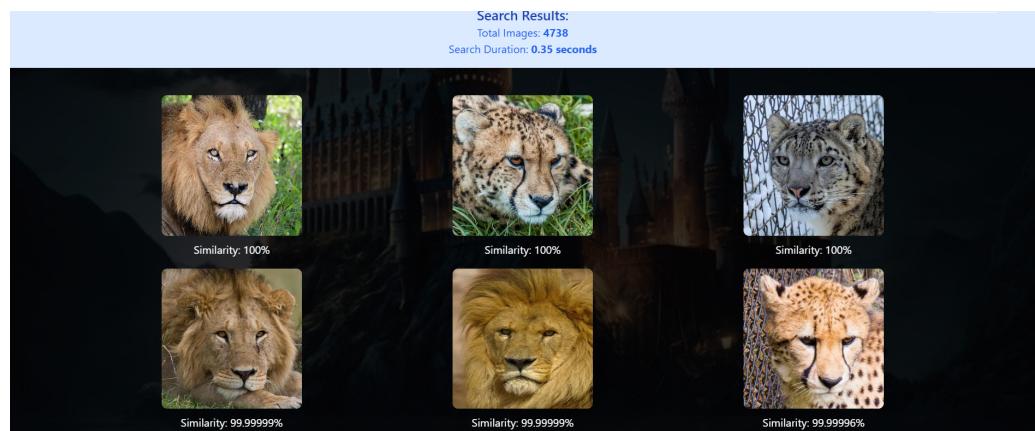
- Pengujian texture

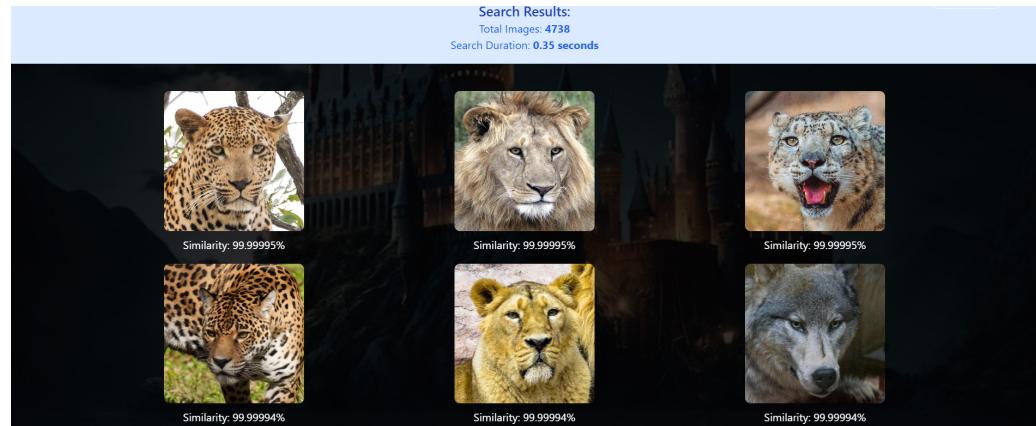
1. Hasil Pengujian 1

Pencarian :

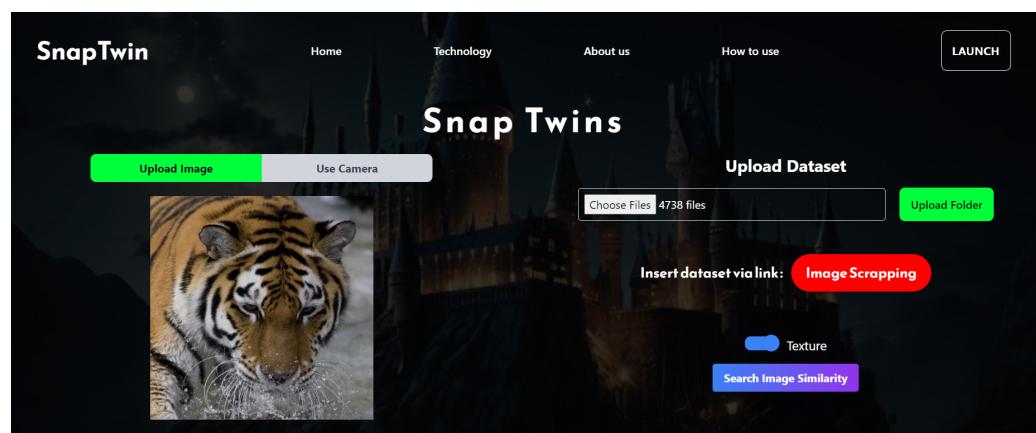


Hasil Pencarian :

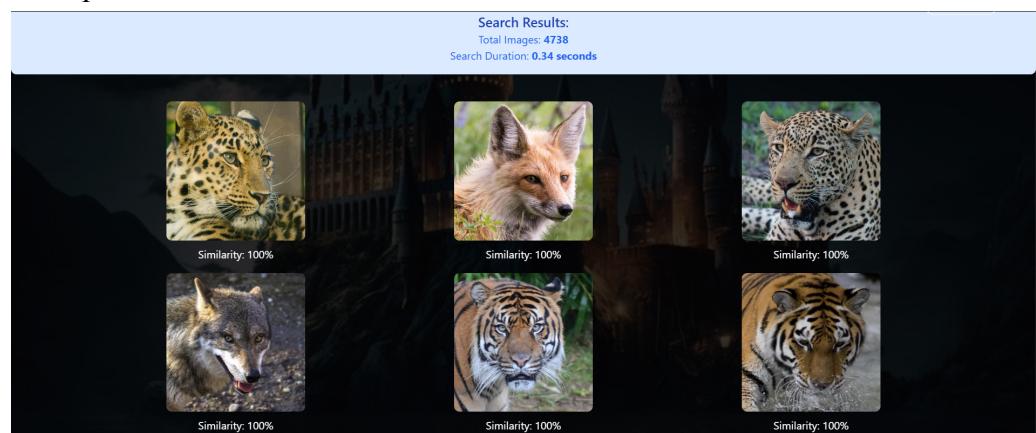


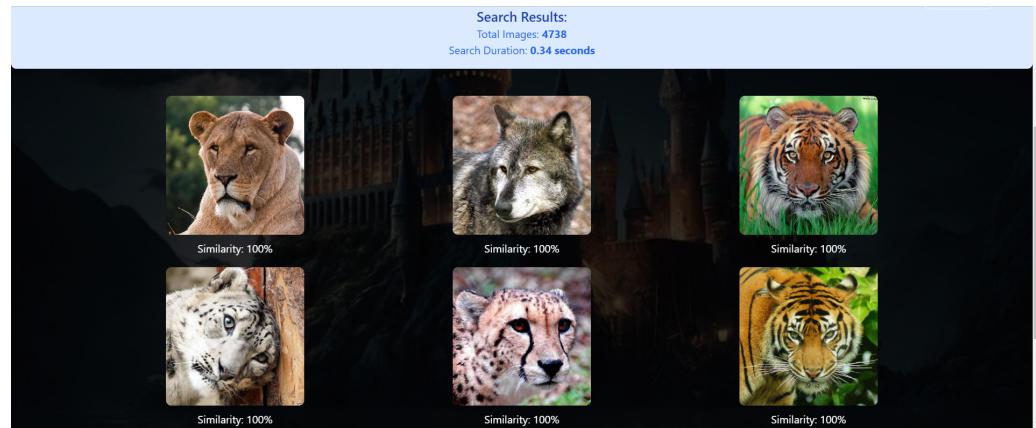


## 2. Hasil Pengujian 2 Pencarian :



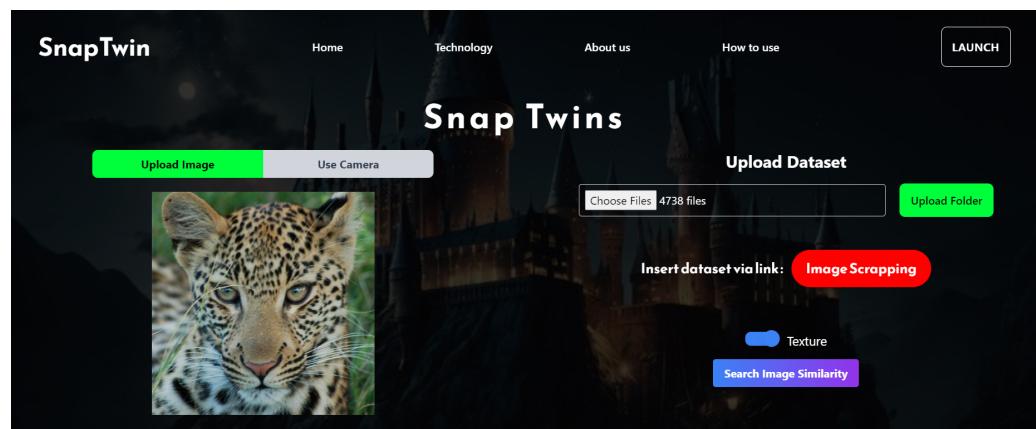
Hasil pencarian :



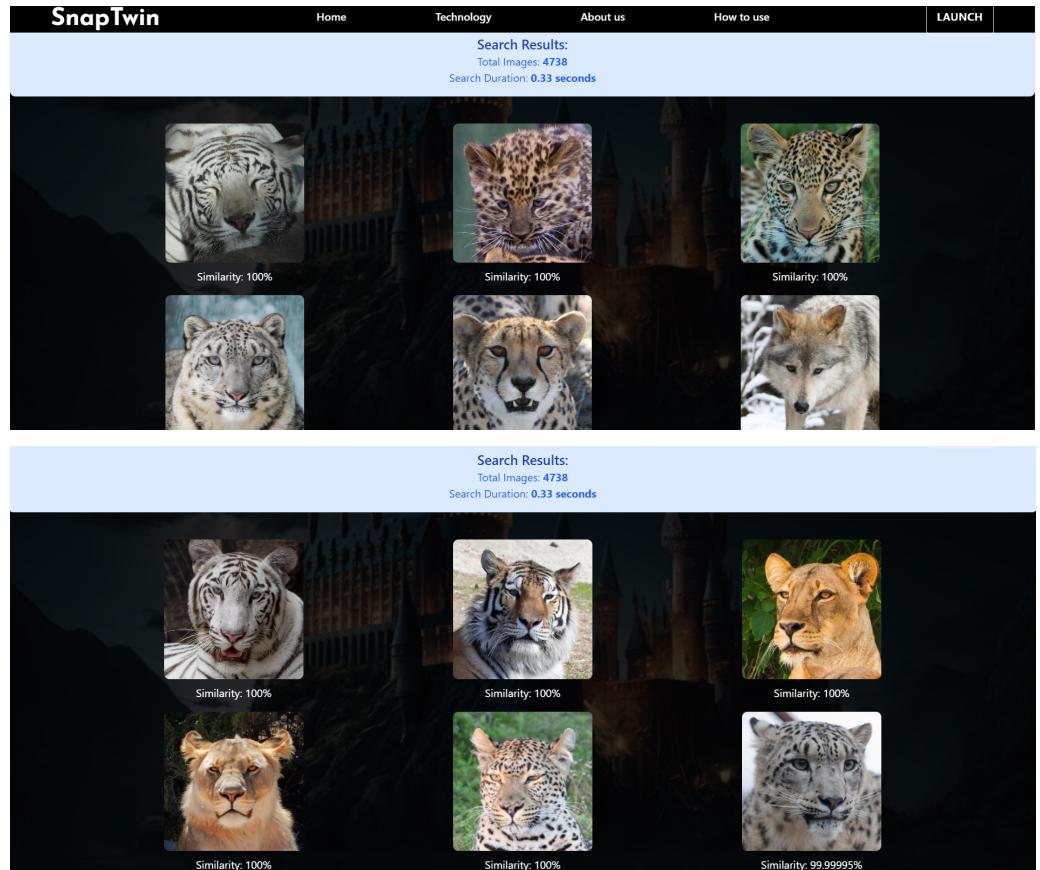


### 3. Hasil Pengujian 3

Pencarian :

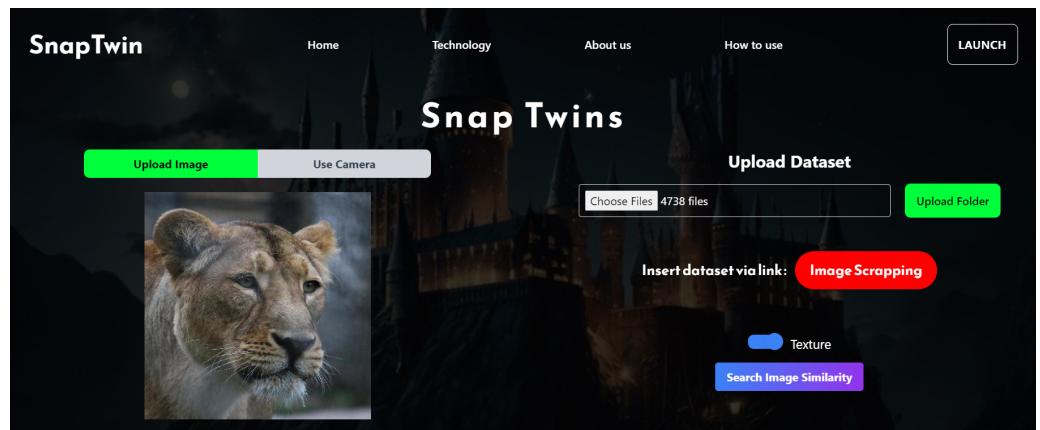


Hasil Pencarian :

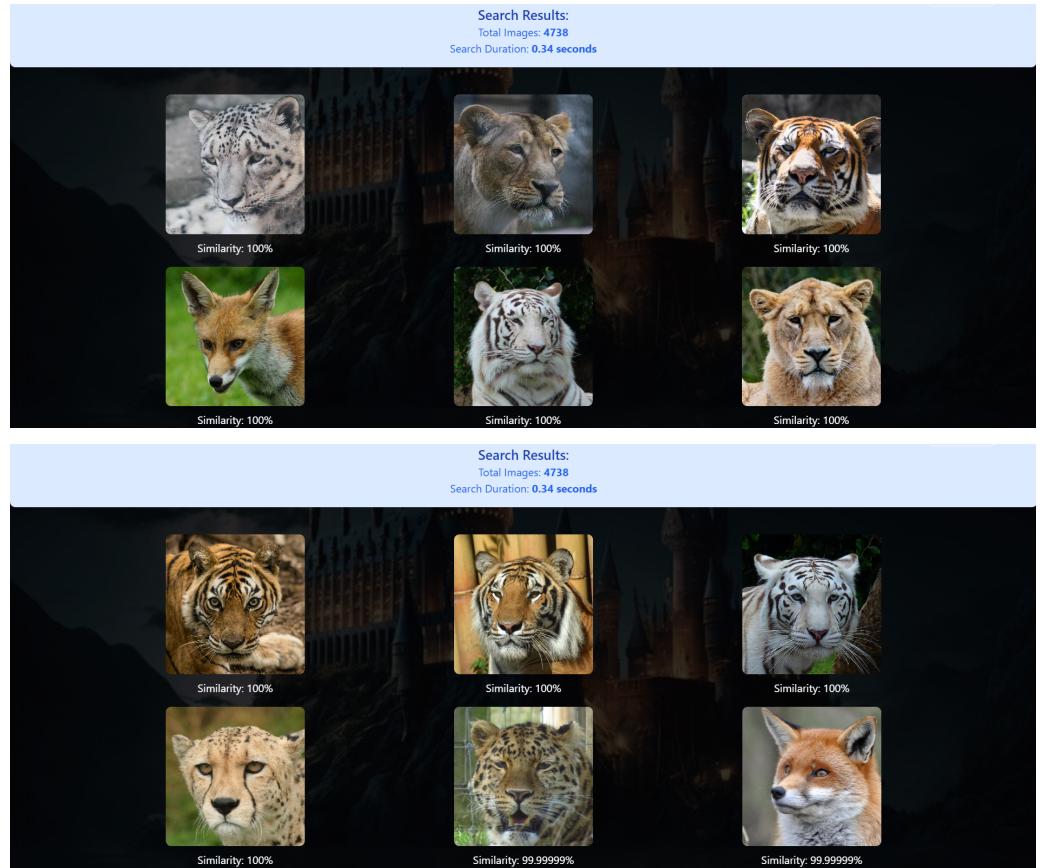


#### 4. Hasil Pengujian 4

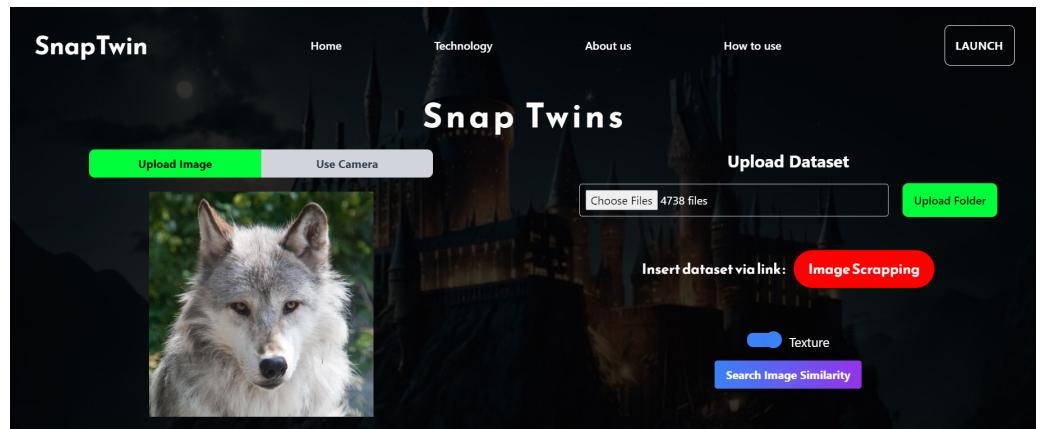
Pencarian :



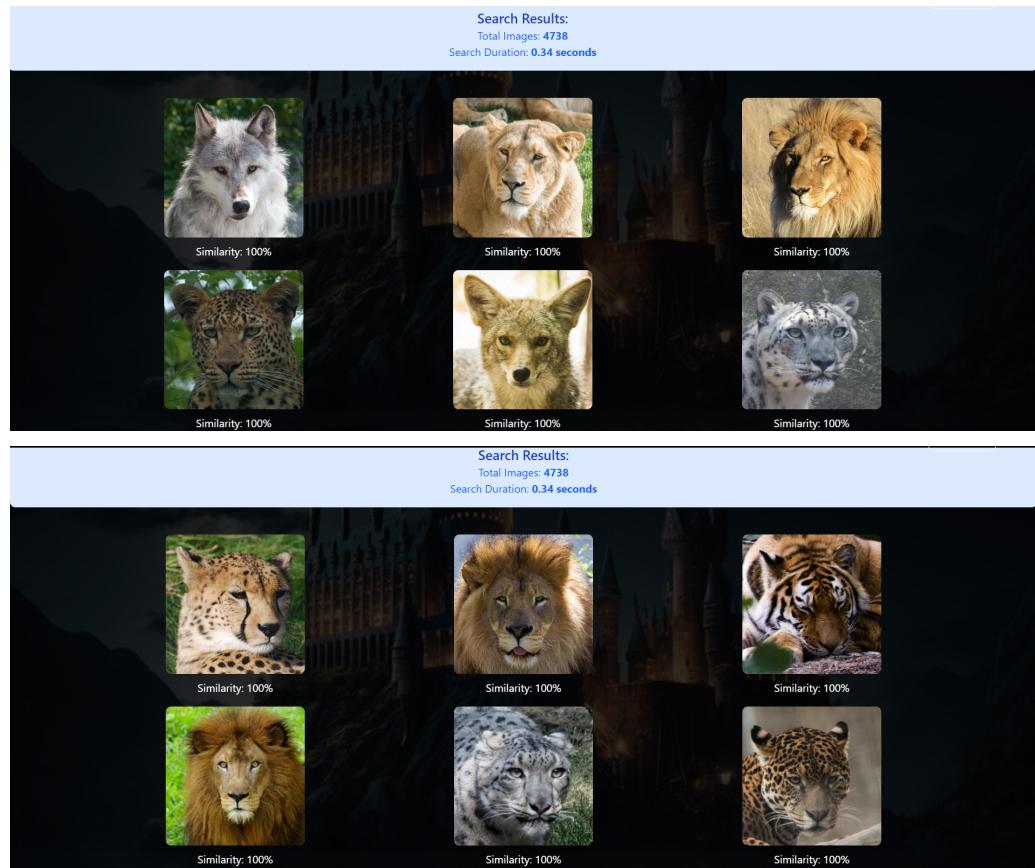
Hasil pencarian :



## 5. Hasil Pengujian 5 Pencarian :



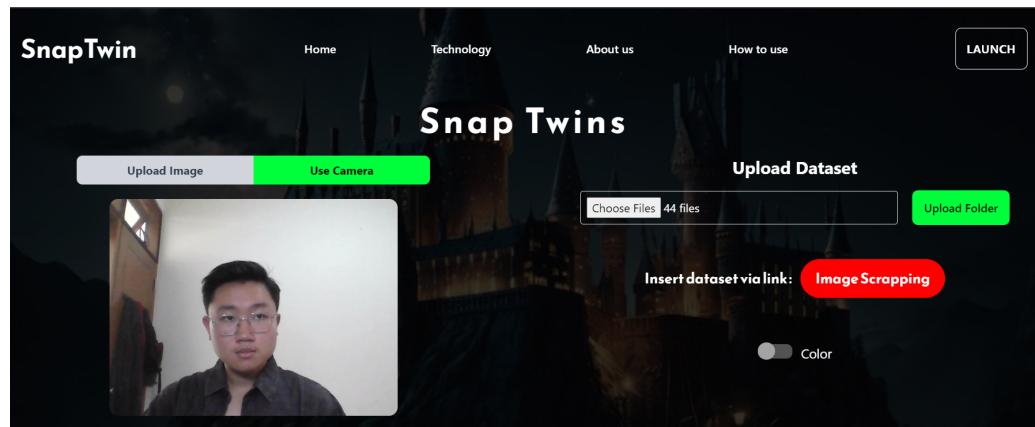
Hasil Pencarian :



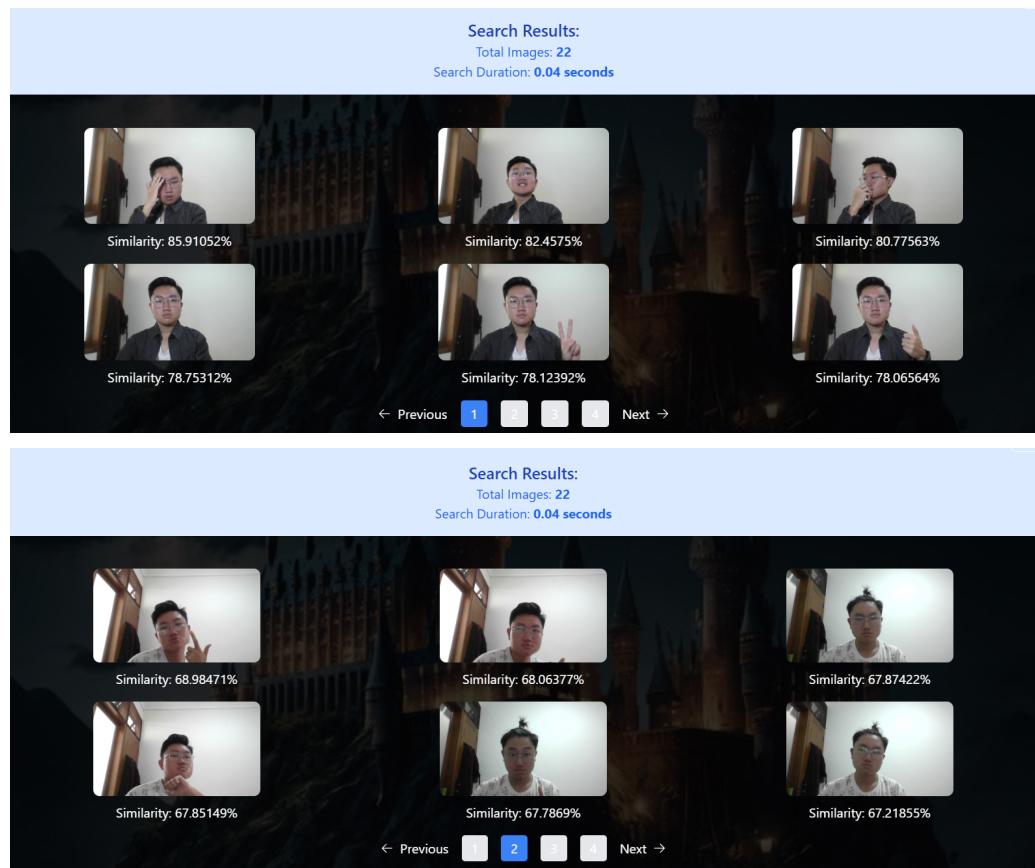
● Pengujian menggunakan camera WebCam

1. Pengujian color :

Pencarian :

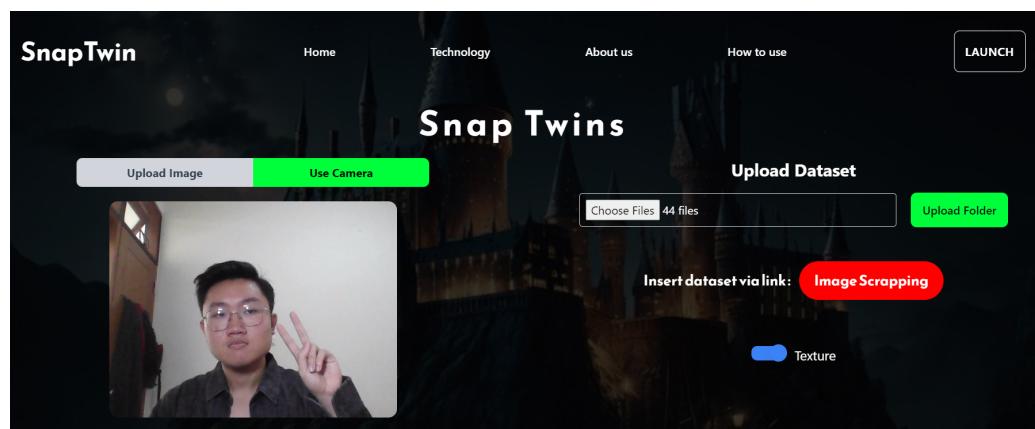


Hasil Pencarian :

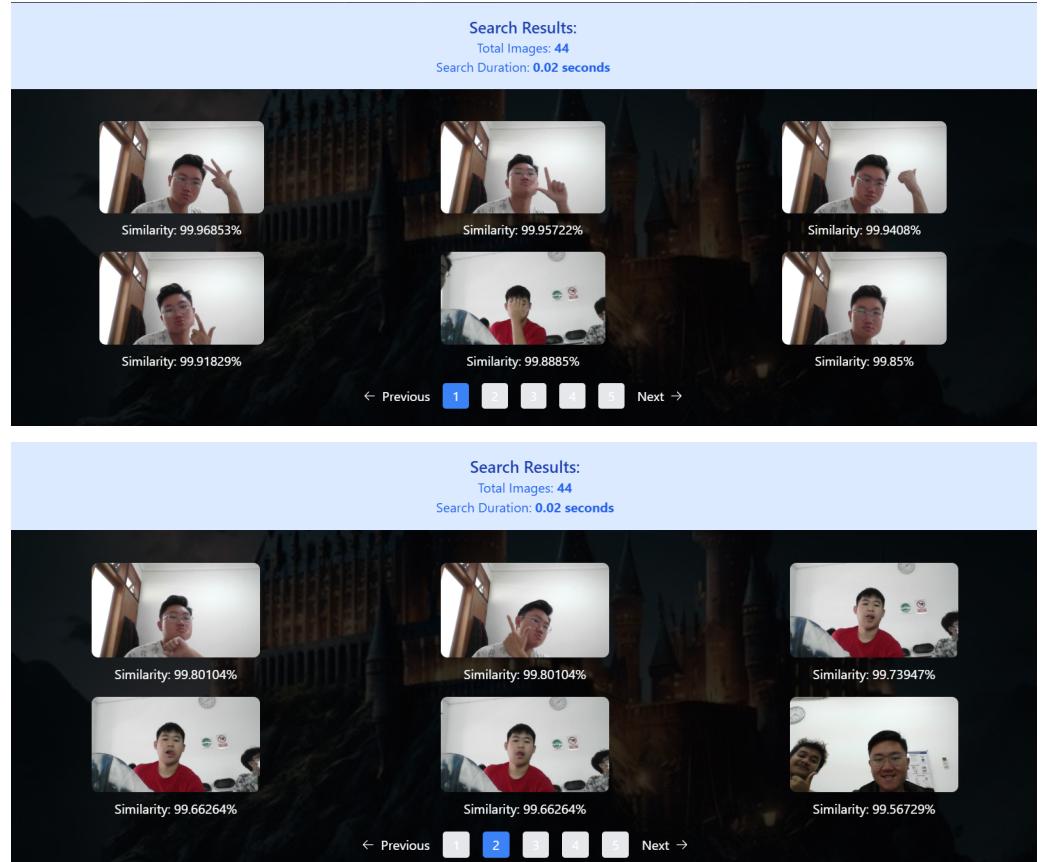


2. Pengujian texture :

Pencarian :

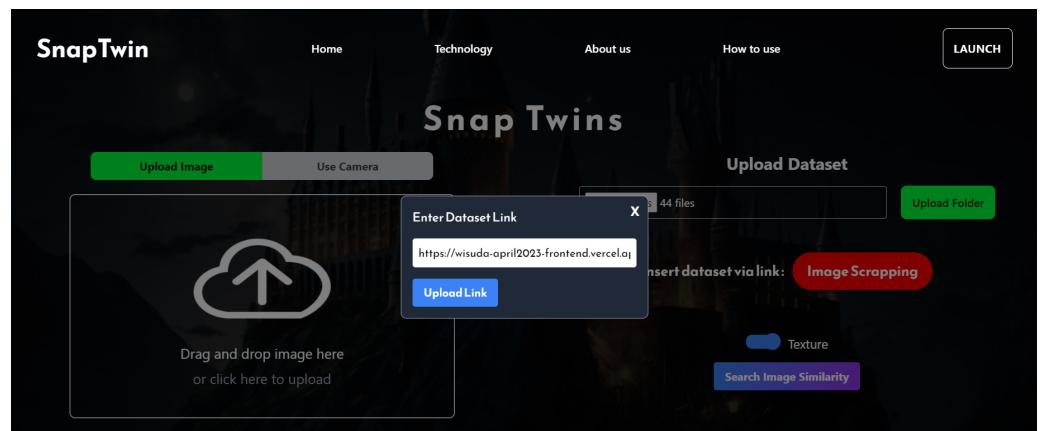


Hasil Pencarian :

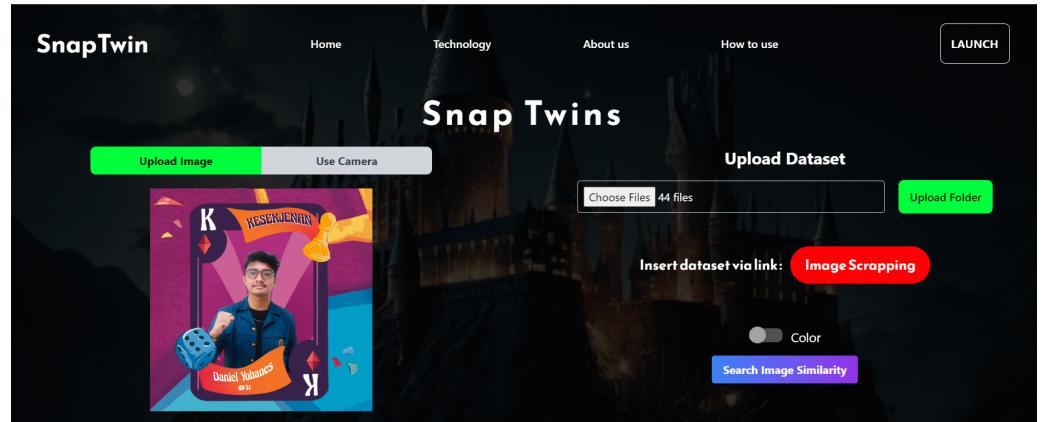


### • Pengujian Image Scraping

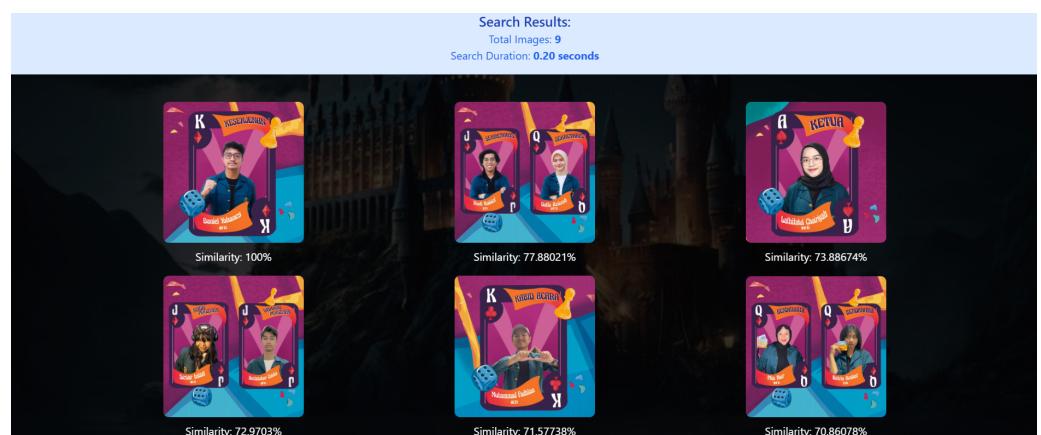
Pada image scraping kali ini kami menggunakan website wispril ITB 2023, dengan hasil sebagai berikut :



1. Pencarian by color :  
Pencarian :

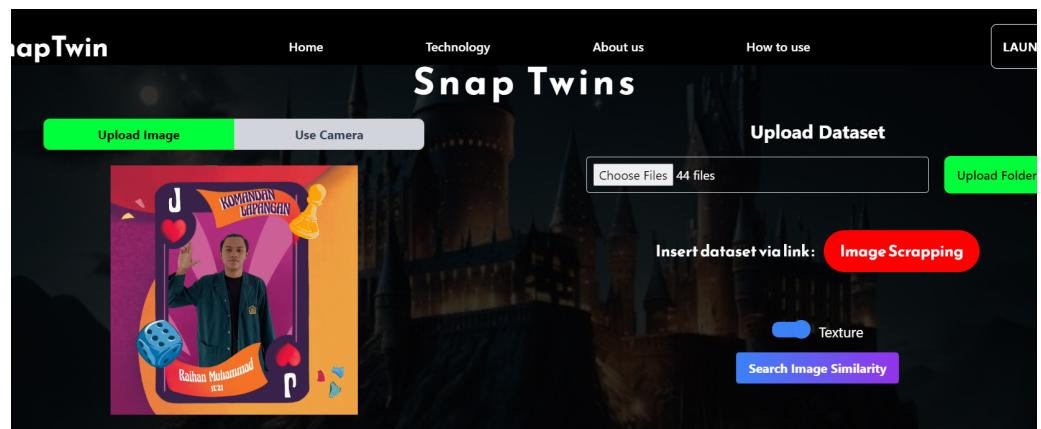


Hasil Pencarian :

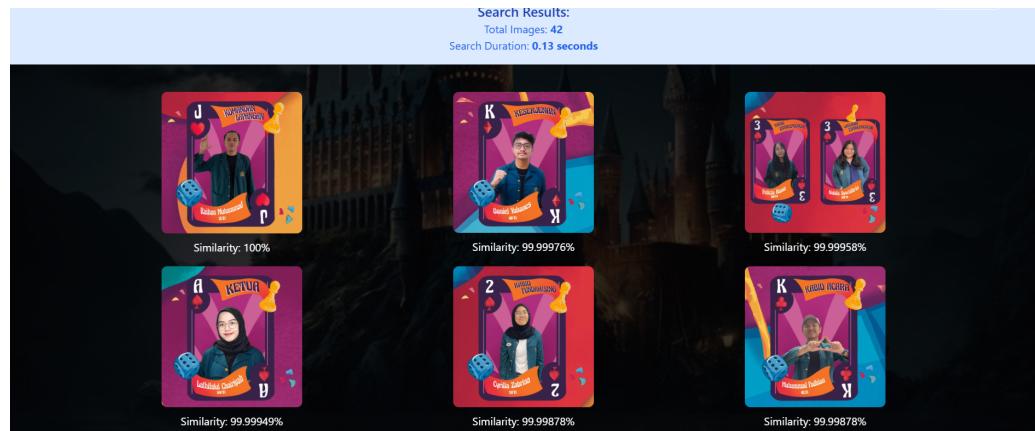


## 2. Pencarian by texture :

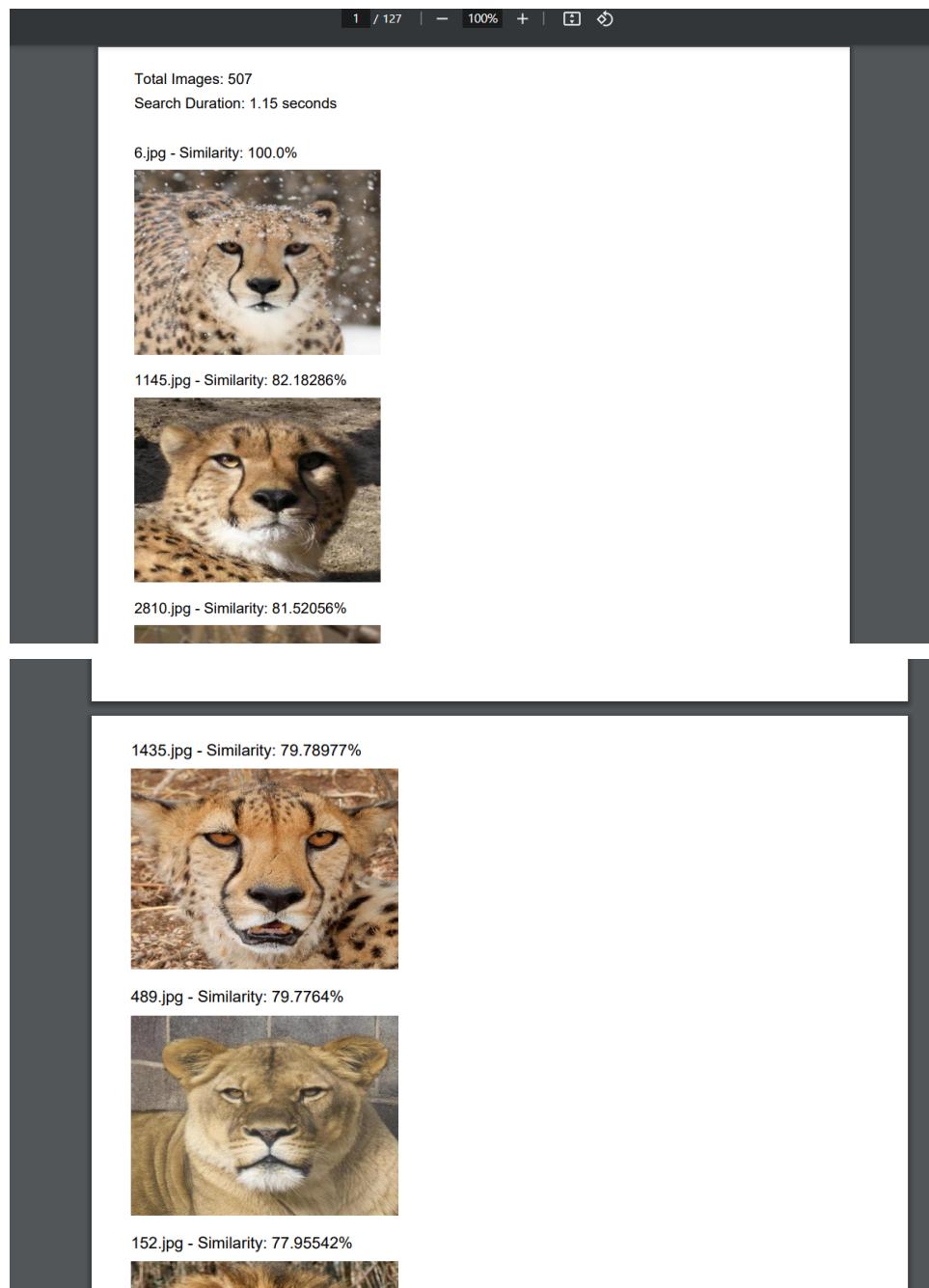
Pencarian :



Hasil pencarian :



- Pengujian fitur Download PDF



## 5. Analisis

Dari hasil penerapan Context-Based Image Retrieval (CBIR) yang mengandalkan fitur warna dan tekstur, kita memahami bahwa keefektifan metode ini sangat tergantung pada jenis dan karakteristik gambar. Untuk CBIR yang berbasis warna, metode ini efektif

dalam situasi dimana warna menjadi aspek utama, seperti dalam pencarian produk berdasarkan warna atau dalam bidang seni dan fotografi. Warna sering kali merupakan atribut yang paling mudah dikenali dan diinterpretasikan oleh pengguna, membuatnya menjadi fitur penting dalam pencarian visual. Sistem CBIR yang memfokuskan pada warna bisa menghasilkan hasil yang lebih intuitif dan sesuai dengan ekspektasi pengguna dalam skenario ini. Namun, metode ini kurang tepat untuk kasus yang memerlukan pemahaman konteks dan objek, karena sulitnya membedakan konteks dan objek dalam gambar.

Sedangkan CBIR yang berbasis tekstur lebih baik dalam mengenali gambar berdasarkan pola atau teksturnya, yang cocok untuk analisis dalam bidang medis atau studi alam. Akan tetapi, ada kendala dalam mengenali gambar dengan tekstur halus atau tidak jelas. Metode ini juga dapat menyebabkan salah identifikasi karena adanya kesamaan tekstur yang tinggi antara gambar yang berbeda. Selain itu, Gambar dengan variasi tekstur yang rendah atau lingkungan dengan pencahayaan yang dapat mengubah persepsi tekstur dapat menantang efektivitas CBIR berbasis tekstur. Dalam situasi seperti ini, fitur tekstur tidak cukup informatif untuk membuat distingsi yang signifikan antara gambar.

Berdasarkan hasil analisis desain solusi algoritma pencarian yang digunakan dalam Context-Based Image Retrieval (CBIR), penting untuk mempertimbangkan efektivitas fitur warna dibandingkan dengan tekstur dalam konteks kasus-kasus tertentu. Berdasarkan pengujian yang dilakukan, terdapat beberapa situasi di mana CBIR berbasis warna mungkin lebih unggul dibandingkan pendekatan berbasis tekstur. Hal tersebut karena dalam CBIR berbasis warna dilakukan pemotongan gambar menjadi  $4 \times 4$  sehingga menjadi 16 bagian. Hal tersebut membuat hasil dari CBIR berbasis warna menjadi lebih akurat, sedangkan untuk tekstur kami telah melakukan resizing untuk gambar terlalu besar, namun tetap saja untuk CBIR berbasis tekstur memberikan similarity yang kurang tepat karena adanya bias terhadap fitur kontras yang terlalu besar dibandingkan dengan fitur yang lain. Dengan demikian, kami menggunakan salah satu parameter tambahan pada tekstur, yaitu disimilarity. Penggunaan disimilarity didasarkan karena bentuk formula yang mirip dengan kontras yakni perkalian elemen matriks co-occurrence dengan mutlak selisih indeks kolom dan baris, dengan harapan dapat mengimbangi nilai yang didapatkan oleh fitur kontras.

## BAB V

### Penutup

#### 1. Kesimpulan

Berdasarkan implementasi dari program Sistem Temu Balik Gambar yang telah dirancang, dapat disimpulkan beberapa hal sebagai berikut.

- 1.1. Sistem Temu Balik Gambar berbasis Konten (CBIR) telah diimplementasikan dengan cara mengekstraksi parameter warna dan parameter tekstur dari suatu citra serta memanfaatkan prinsip-prinsip aljabar vektor, salah satunya penggunaan kesamaan cosinus (*cosine similarity*) untuk membandingkan kemiripan dari 2 vektor citra.
- 1.2. Program berbasis aplikasi website telah diimplementasikan dengan menggunakan *framework* React (*frontend*) dan Flask (*backend*).
- 1.3. Program menerima masukan *dataset* baik dalam bentuk *folder*, maupun dalam bentuk laman (*image scraping*). Kemudian, pengguna dapat memilih untuk memberi masukan dari suatu citra ataupun dengan mengaktifkan fitur kamera.
- 1.4. Detail implementasi serta penjelasan struktur, tata cara, serta pengujian terhadap program dapat diamati pada bab IV.

#### 2. Saran

- Pembuatan laporan harusnya dicicil awal dan diselesaikan secepatnya karena format laporan yang cukup banyak dan kompleks
- Struktur front end dan backend harusnya direncanakan dengan baik sejak awal proyek agar tugas masing-masing dapat berjalan dengan lebih bagus

#### 3. Komentar

- dari Albert : masih ada TBFO bang
- dari Derwin : speedrun TBFO lah
- dari Benardo : Ada referensi TBFO gak?

#### 4. Refleksi

Melalui perjalanan pembuatan aplikasi website dari awal hingga akhir, kami telah memperoleh berbagai wawasan berharga dan mengembangkan pemahaman yang lebih mendalam tentang teknologi dan aljabar geometri. Proyek ini membuka wawasan kami tidak hanya dalam aspek teknis pembuatan aplikasi tetapi juga dalam mengaplikasikan konsep aljabar geometri, seperti vektor dan matriks, dalam konteks nyata. Kami belajar bagaimana membangun fondasi aplikasi dengan menggunakan *framework* Flask untuk backend dan React untuk frontend. Proses integrasi antara frontend dan backend menjadi pengalaman yang sangat berharga, memberikan pemahaman praktis tentang bagaimana

komponen-komponen aplikasi bekerja secara bersamaan untuk menciptakan sistem yang koheren. Melalui tantangan ini, kami juga mendapatkan pengetahuan tentang berbagai package dan library baru, seperti library untuk ekspor PDF dan penggunaan Numpy untuk perhitungan matematis yang kompleks. Pengalaman ini mengajarkan kami bagaimana menyeimbangkan kebutuhan fungsionalitas dengan efisiensi dan skalabilitas dalam pengembangan perangkat lunak. Kami mendapatkan pemahaman praktis tentang pengembangan aplikasi, mulai dari awal hingga realisasi akhir.

## 5. Ruang Perbaikan

Dalam proses pembuatan program Sistem Temu Balik Gambar berbasis aplikasi website yang telah diimplementasikan, kami menyadari bahwa program masih dapat dioptimasi lebih lanjut. Salah satu aspek yang dapat diimplementasikan kedepannya adalah dengan memanfaatkan bahasa pemrograman lainnya yang menawarkan kompleksitas waktu dalam eksekusi program yang lebih efisien, contohnya Go. Selain itu, penggunaan *docker* juga dapat diimplementasikan sehingga pengguna dapat menjalankan *frontend* dan *backend* secara bersamaan.

## **Daftar Pustaka**

Anton, Howard. Elementary Linear Algebra with Applications Eleventh Edition. USA: Wiley, 2014.

Yue, Jun. Content-based image retrieval using color and texture fused features. Mathematical and Computer Modelling, Volume 54, Issues 3–4, 2011.

<https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2023-2024/Algeo-14-Aplikasi-dot-product-pada-IR-2023.pdf> (Diakses pada tanggal 18 November 2023)

<https://yunusmuhammad007.medium.com/feature-extraction-gray-level-co-occurrence-matrix-glcm-10c45b6d46a1> (Diakses pada tanggal 10 November 2023)

## LAMPIRAN

- Link Repository :

Link berikut merupakan link untuk menuju ke github repository dimana aplikasi dikembangkan.

<https://github.com/AlbertChoe/Algeo02-22055>

- Link Video

Link berikut merupakan link untuk menuju ke video youtube dimana aplikasi di promosikan.

<https://youtu.be/1u7IHUatbKY>