LAPORAN TUGAS BESAR II
IF2123 ALJABAR LINEAR DAN GEOMETRI

Laporan ini dibuat untuk memenuhi tugas

Mata Kuliah IF 2123 Aljabar Liniear dan Geometri

Disusun Oleh:

Kelompok "Kosu Salted Caramel"

Raden Rafly Hanggaraksa Budiarto (13522014)

Wilson Yusda (13522019)

Abdul Rafi Radityo Hutomo (13522089)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

SEMESTER I TAHUN 2022/2023

DAFTAR ISI

DAFTAR GAMBAR

BAB I
DESKRIPSI MASALAH

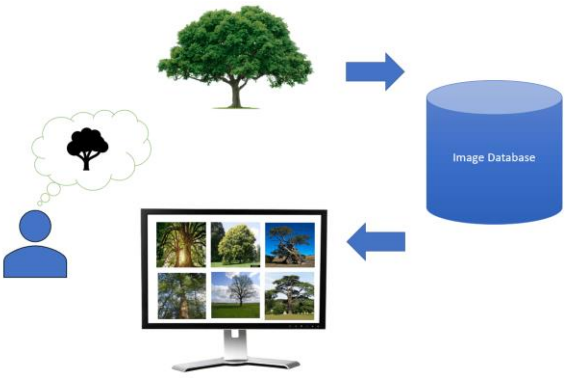
Dalam era digital, jumlah gambar yang dihasilkan dan disimpan semakin meningkat dengan pesat, baik dalam konteks pribadi maupun profesional. Peningkatan ini mencakup berbagai jenis gambar, mulai dari foto pribadi, gambar medis, ilustrasi ilmiah, hingga gambar komersial. Terlepas dari keragaman sumber 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.

Gambar 1 Contoh penerapan informational retrieval system (Google Lens)

Di dalam Tugas Besar 2 ini, Anda diminta untuk mengimplementasikan sistem temu balik gambar yang sudah dijelaskan sebelumnya dengan memanfaatkan Aljabar Vektor dalam bentuk sebuah website, dimana hal ini merupakan pendekatan yang penting dalam dunia pemrosesan data dan pencarian informasi. Dalam konteks ini, aljabar vektor digunakan untuk menggambarkan dan menganalisis data menggunakan pendekatan klasifikasi berbasis konten (Content-Based Image Retrieval atau CBIR), di mana sistem temu balik gambar bekerja dengan mengidentifikasi gambar berdasarkan konten visualnya, seperti warna dan tekstur.

BAB II
LANDASAN TEORI

TEORI DASAR



Gambar 2 Diagram Content-Based Image Retrieval

Content-Based Image Retrieval (CBIR) adalah sebuah proses yang digunakan untuk mencari dan mengambil gambar berdasarkan kontennya. Proses ini dimulai dengan ekstraksi fitur-fitur penting dari gambar, seperti warna, tekstur, dan bentuk. Setelah fitur-fitur tersebut diekstraksi, mereka diwakili dalam bentuk vektor atau deskripsi numerik yang dapat dibandingkan dengan gambar lain. Kemudian, CBIR menggunakan algoritma pencocokan untuk membandingkan vektor-fitur dari gambar yang dicari dengan vektor-fitur gambar dalam dataset. Hasil dari pencocokan ini digunakan untuk mengurutkan gambar-gambar dalam dataset dan menampilkan gambar yang paling mirip dengan gambar yang dicari. Proses CBIR membantu pengguna dalam mengakses dan mengeksplorasi koleksi gambar dengan cara yang lebih efisien, karena tidak memerlukan pencarian berdasarkan teks atau kata kunci, melainkan berdasarkan kesamaan nilai citra visual antara gambar-gambar tersebut.

CBIR dengan parameter warna

Pada CBIR kali 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

Nilai dari RGB harus dinormalisasi dengan mengubah nilai range [0, 255] menjadi [0, 1]

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

Mencari Cmax, Cmin, dan Δ

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

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

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

Selanjutnya gunakan hasil perhitungan di atas untuk mendapatkan nilai HSV

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

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

$$V = Cmaks$$

Setelah mendapatkan nilai HSV lakukanlah perbandingan antara image dari input dengan dataset dengan menggunakan cosine similarity

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

Untuk melakukan pencarian histogram, blok image dibagi menjadi n × n blok. Setiap blok akan menjadi tidak terlalu signifikan jika blok-blok tersebut terlalu besar dan akan meningkatkan waktu dalam memprosesnya jika ukuran dari blok terlalu kecil. Pada pencarian blok ini agar lebih efektif disarankan menggunakan 3 × 3 blok.

Dalam pemilihan bin histogram juga terdapat preferensi range yang berbeda-beda. Salah satu penentuan ukuran bin yang dapat dilakukan yakni:

$$H = \begin{cases} 0 & h \in [316, 360] \\ 1 & h \in [1, 25] \\ 2 & h \in [26, 40] \\ 3 & h \in [41, 120] \\ 4 & h \in [121, 190] \\ 5 & h \in [191, 270] \\ 6 & h \in [271, 295] \\ 7 & h \in [295, 315] \end{cases}$$

$$S = \begin{cases} 0 & s \in [0, 0.2) \\ 1 & s \in [0.2, 0.7) \\ 2 & s \in [0.7, 1] \end{cases}$$

$$V = \begin{cases} 0 & v \in [0, 0.2) \\ 1 & v \in [0.2, 0.7) \\ 2 & v \in [0.7, 1] \, . \end{cases}$$

Untuk setiap nilai HSV, akan meng-*increment* salah salah satu dari 72 kombinasi bin, sehingga dapat dipastikan jumlah dari seluruh histogram yaitu sebanyak jumlah pixel suatu foto.

Selain itu, dalam pengukuran block image 3x3, diperlukan adanya pembagian weight yang sesuai. Hal ini dilakukan guna meningkatkan akurasi sistem dalam menilai objek yang ada. Salah satu jenis pembobotan yang dapat dilakukan yakni:

1	3	1
1	5	1
1	3	1

Pembobotan ini telah diuji dalam beberapa skenario dalam dan diaplikasikan untuk meningkatkan ketepatan program dalam melacak objek pada suatu gambar.

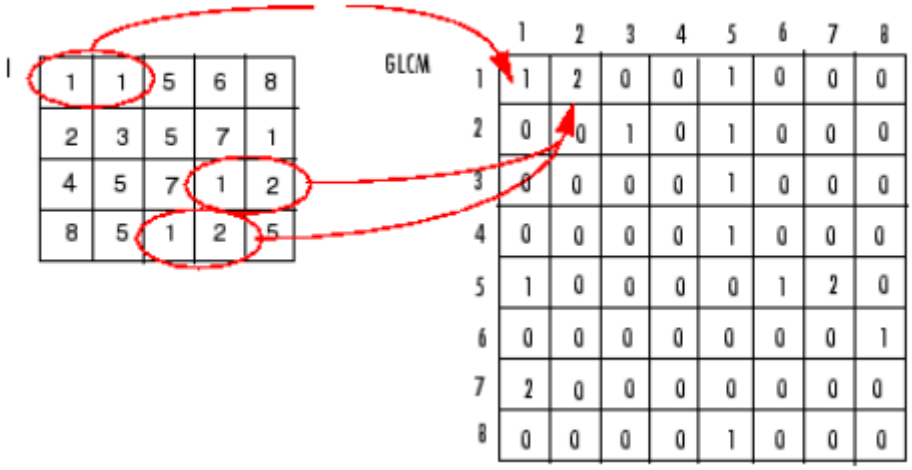
CBIR dengan parameter tekstur

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 × m piksel dan suatu parameter offset (Δx, Δy), 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 Δx dan Δy bergantung pada arah θ 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 θ adalah 0°, 45°, 90°, dan 135°. Sebagai gambaran, berikut diberikan contoh cara pembuatan co-occurrence matrix dan link cara pembuatannya (Contoh ini dapat digunakan sebagai referensi, bukan acuan sebagai acuan utama).



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 :

1. Konversi warna gambar menjadi grayscale, 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. Lakukan kuantifikasi dari nilai grayscale. Karena citra grayscale berukuran 256 piksel, maka matriks yang berkoresponden akan berukuran 256 × 256. Berdasarkan penglihatan manusia, tingkat kemiripan dari gambar dinilai berdasarkan kekasaran tekstur dari gambar tersebut. Grayscale semula dari suatu gambar akan dikompresi untuk mengurangi operasi perhitungan sebelum dibentuknya co-occurrence matrix
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.

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.

PENGEMBANGAN WEBSITE

Gambar 3 Framework Vue.js + Flask

Perencanaan dan Analisis

Sebelum memulai pembangunan sebuah website, langkah pertama dan paling kritis adalah merencanakan secara menyeluruh untuk memastikan bahwa pengerjaan berjalan secara sistematis dan efisien. Proses ini dimulai dengan memahami secara mendalam tentang fitur-fitur yang diinginkan dalam website, yang meliputi aspek-aspek seperti fungsionalitas, desain, interaktivitas pengguna, serta kompatibilitas dengan berbagai perangkat dan browser. Dalam tahap ini, sangat penting untuk mengevaluasi apakah rencana yang dibuat realistis dan dapat diimplementasikan, mengingat keterbatasan sumber daya, waktu, dan teknologi.

Perencanaan juga perlu memperhatikan pemilihan teknologi dan tools yang akan digunakan, baik untuk front-end maupun back-end. Ini termasuk memilih bahasa pemrograman, framework, database, dan hosting yang sesuai dengan kebutuhan dan skala website. Pertimbangan ini harus sejalan dengan tujuan jangka panjang website, termasuk kemudahan dalam pemeliharaan dan skalabilitas.

Terakhir, penting untuk menyusun timeline proyek secara rinci, yang mencakup milestone penting, tahap-tahap pengembangan, dan periode pengujian. Ini memungkinkan tim pengembangan untuk bekerja secara efisien dan memastikan bahwa proyek berjalan sesuai rencana.

Desain Grafis

Desain grafis dalam konteks web sering dikaitkan dengan UI/UX, di mana "UI" berarti "User Interface" dan "UX" berarti "User Experience". UI berfokus pada aspek visual dari antarmuka pengguna sedangkan UX berfokus bagaimana pengalaman pengguna ketika berinteraksi dengan antarmuka. Keduanya merupakan aspek penting dalam desain web yang bekerja bersama untuk memastikan bahwa sebuah website tidak hanya terlihat menarik secara visual, tetapi juga mudah dan menyenangkan untuk digunakan.

Berikut adalah aspek-aspek utama dari desain grafis:

1. Kekontrasan elemen
2. Keseimbangan elemen
3. Kohesi dan kesatuan elemen
4. Hierarki visual
5. Repetisi dan Ritme
6. Ruang (space)

Selain itu, elemen dalam sebuah desain mencakup:

1. Warna
2. Bentuk dan garis
3. Tekstur
4. Tipografi
5. Gambar dan ikon

Pengembangan Front-End

Pengembangan front-end merujuk pada bagian pengembangan website atau aplikasi web yang berfokus pada apa yang pengguna lihat dan interaksikan langsung dalam browser mereka. Ini meliputi desain halaman web, interaksi, animasi, dan perilaku pada input pengguna. Pengembangan front-end biasanya melibatkan kombinasi dari tiga teknologi inti: HTML, CSS, dan JavaScript, serta

berbagai frameworks dan libraries yang memperkaya dan mempermudah proses pengembangan.

Salah satu frameworks yang populer yaitu Vue.js. Vue.js adalah framework JavaScript progresif yang digunakan untuk membangun antarmuka pengguna. Vue didesain untuk menjadi mudah diintegrasikan ke dalam proyek yang ada dan untuk digunakan secara bersamaan dengan pustaka lainnya. Fitur utamanya termasuk adanya komponen yang reusable, arsitektur yang mudah diikuti, dan Ekosistem yang kaya.

Tailwind CSS adalah framework CSS yang menyediakan kelas utilitas yang bisa digunakan untuk membangun desain kustom tanpa keluar dari HTML. Berbeda dari framework seperti Bootstrap yang menyediakan komponen yang sudah jadi, Tailwind memungkinkan pengembang untuk mendesain secara lebih granular dengan menggabungkan kelas bantu. Fitur utamanya meliputi pendekatan utility first, penanganan fitur responsif, dan fitur yang *customizable*.

Gabungan antara Vue.js dan Tailwind CSS dapat menciptakan pengalaman pengembangan front-end yang sangat efisien dan fleksibel, memungkinkan pembuatan antarmuka yang responsif, ringan, dan estetis.

Pengembangan Back-End

Pengembangan Back-End merujuk pada bagian pengembangan website atau aplikasi yang berfokus pada server, aplikasi, dan database. Ini melibatkan penulisan API, script server-side, dan interaksi dengan database untuk memproses logika bisnis, menyimpan data, dan mengirimkannya kembali ke pengguna di front-end. Back-end bekerja di belakang layar dan merupakan bagian penting yang memungkinkan fungsionalitas sebuah website atau aplikasi web.

Komponen utama pengembangan Back-End yakni:

1. Server
2. Aplikasi (App)
3. Database

Flask adalah micro-framework untuk Python yang digunakan dalam pengembangan aplikasi web. Meskipun disebut sebagai "micro", Flask sangat mumpuni dan serbaguna. Berikut adalah beberapa aspek penting dari Flask:

1. Ringan dan Modular
2. Mudah dipelajari
3. Fleksibilitas
4. Routing dan View
5. Integrasi dengan Database
6. Jinja2 Templating

Flask sering digunakan untuk membangun API, aplikasi web skala kecil hingga menengah, dan sebagai alat untuk mengembangkan prototype secara cepat. Karena sifatnya yang ringan dan fleksibel, Flask memungkinkan pengembang untuk membangun back-end yang efisien dan mudah di-maintain, dengan kontrol penuh atas aspek-aspek aplikasi.

Integrasi Struktur Website

Integrasi dalam konteks pengembangan web berarti menggabungkan berbagai komponen, teknologi, dan sistem untuk membuat sebuah website berfungsi sebagai kesatuan yang koheren. Ini melibatkan beberapa aspek:

1. Integrasi Front-End dan Back-End
2. Integrasi dengan API Pihak Ketiga
3. Integrasi Database
4. Integrasi Sistem Manajemen Konten (CMS)
5. Integrasi Tools dan Frameworks

Pengujian

Pengujian merupakan tahap krusial dalam proses pengembangan website. Tujuan utama dari pengujian adalah untuk memastikan bahwa website beroperasi dengan benar di berbagai kondisi dan lingkungan, serta memenuhi semua persyaratan fungsional dan teknis yang telah ditetapkan. Berikut adalah aspek-aspek penting dalam pengujian website:

1. Pengujian Fungsional
2. Pengujian Usability
3. Pengujian Kompatibilitas
4. Pengujian Keamanan
5. Pengujian SEO
6. Validation Testing

BAB III
ANALISIS PEMECAHAN MASALAH

Langkah-Langkah Pemecahan Masalah

1. CBIR dengan parameter warna

Langkah pertama dalam mencari kemiripan suatu gambar dengan menggunakan parameter warna yakni dengan mengekstrak nilai RGB dari suatu gambar. Nilai RGB diekstrak dari setiap pixel dalam satu gambar sehingga jumlah dari R, G, dan B pada sebuah gambar akan sama dengan jumah pixel suatu gambar. Langkah selanjutnya yaitu untuk dengan mengubah nilai RGB setiap gambar menjadi sebuah nilai HSV.

Prosedur diatas dilakukan pada kedua gambar yang akan dibandingkan. Selanjutnya, diperlukan sebuah bin histogram untuk setiap nilai H, S, dan V suatu pixel. Bin yang digunakan dalam pelaksanaan tugas besar kali ini yaitu:

Untuk bin H :

[316 <= H <= 360], [1 <= H <= 25], [26 <= H <= 40], [41 <= H <= 120], [121 <= H <= 190], [191 <= H <= 270], [271 <= H <= 295], [295 <= H <= 315]

Untuk bin S:

[0 <= S < 0.2], [0.2 <= S < 0.7], [0.7 <= S <= 1]

Untuk bin V:

[0 <= S < 0.2], [0.2 <= S < 0.7], [0.7 <= S <= 1]

Dalam pengerjaan yugas besar ini, dilakukan metode block search, sehingga kami membagi gambar menjadi 9 potong (3x3) dan untuk masing masing block dilakukan pengukuran terhadap histogram.

Setiap bin histogram menandakan kombinasi HSV yang unik, sehingga akan ada 72 kombinasi. Masing masing HSV pada kedua gambar akan meng-*increment* bin histogram satu per satu sampai jumlah value histogram sama dengan jumlah pixel blok gambar yang diukur.

Setelah pengukuran selesai, akan didapat 2 histogram dari 2 blok posisi sama pada 2 gambar berbeda. Kemudian dilakukan pengukuran kemiripan dengan

menggunakan metode *cosine similarity*. Metode *cosine similarity* nantinya akan menghasilkan kemiripan antara 2 buah blok gambar dan disimpan nilainya dalam suatu tempat.

Progress ini dilakukan sampai 9 kali sesuai dnegan jumlah blok, blok 1 sumber dibandingkan dengan blok 1 target, blok 2 sumber dibandingkan dengan blok 2 target dan begitu selanjutnya sampai 9 blok selesai diukur.

Untuk setiap hasil *cosine similarity* akan memiliki weight tertentu untuk memastikan kecocokan objek pada gambar, bukan hanya latar belakang objek. Hasil *cosine similarity* antar blok ditengah dikali dengan 5 sebelum dijumlahkan dan blok dibagian atas dan bawab blok tengah bernilai 3 disebabkan pengalaman kami dalam menemukan bahwa pembobotan ini menghasilkan hasil yang lebih optimal. Perlu diingat bahwa pembobotan dapat berubah sesuai dengan tipe gambar, seperti apakah foto *landscape* atau *potrait*, serta kondisi ketika gambar tidak ditengah.

Setelah memiliki 9 nilai kemiripan, dengan adanya nilai yang dikali sesuai dengan pembobotan yang ada, akan dibagi rata sesuai dengan jumlah bobot untuk menghasilkan tingkat kemiripan antara kkedua gambar yang sebenarnya.

2. CBIR dengan parameter tekstur

Dalam implementasi Content Based Image Recognition dengan texture, dilakukan simplifikasi dari gambar terlebih dahulu dengan merubahnya dari pewarnaan RGB menjadi pewarnaan *grayscale*. Hal tersebut dapat dilakukan sebab pengaruh warna RGB maupun *grayscale* dalam sebuah gambar tidak memengaruhi tekstur dari gambar tersebut. Langkah tersebut dilakuka dengan menggunakan rumus G = 0.299*R + 0.587*G + 0.114*B, dengan G adalah nilai Grayscale dan R, G, B secara berturut-turut adalah Red, Green, dan Blue

Kemudian, dari nilai-nilai tersebut dikonstruksi sebuah *CoOccurence Matrix*, yaitu sebuah matriks yang berisi informasi mengenai posisi pixel dengan satu sama lain. Pada program kami, kami menggunakan sudut 0 derajat dan jarak 1 pixel, sehingga isi *Co-Occurence Matrix* untuk setiap index x, y adalah jumlah kemunculan nilai pixel x dan nilai pixel y pada gambar. Kemudian, *CoOccurence Matrix* dibuat simetrik dengan cara menambahkannya ke transpose diri sendirinya dan di normalisasi dengan membagi setial elemennya dengan jumlah elemennya.

Kemudian, dari *Co-Occurence Matrix* diekstraksi elemen tekstur kontras, homogenitas, dan entropi sesuai dengan rumus masing-masing. Dari ketiga nilai tersebut, dibuat sebuah *feature vector*, yaitu vektor berdimensi tiga yang mendeskripsikan tekstur dari gambar tersebut.

Kemudian, dua gambar dibandingkan dengan cara mengamati feature vectornya menggunakan rumus *cosine similarity*.

Proses Pemetaan Masalah Menjadi Elemen-Elemen Pada Aljabar Geometri

1. Pemetaan Masalah pada CBIR dengan parameter warna

Salah satu permasalahan yang ingin diselesaikan yakni dalam mencari kemiripan antara 2 buah gambar. Dalam metode CBIR dengan parameter warna, diperlukan konversi antara setiap nilai RGB pada suatu gambar yang diukur per pixel dan diubah ke HSV. Selanjutnya untuk setiap kecocokannya dengan bin yang sudah dibuat, akan meng-*increment* posisi bin tersebut. Kemudian akan dihasilkan sebuah histogram dengan 72 bin. Lantas bagaimana cara untuk menentukan kemiripan antar 2 buah histogram dengan bin 72. Solusinya yaitu kita dapat merepresentasikan histogram tersebut sebagai suatu vektor. Alhasil kita mendapat 2 vektor.

Cosine similarity adalah metode yang digunakan untuk mengukur kemiripan antara dua vektor dalam ruang berdimensi n. Ini sering digunakan dalam berbagai bidang, termasuk pemrosesan teks, pengambilan informasi, dan analisis data. Cosine similarity mengukur sudut antara dua vektor dalam ruang berdimensi n dan memberikan nilai yang berkisar dari -1 hingga 1, di mana nilai yang lebih tinggi menunjukkan tingkat kemiripan yang lebih tinggi. Karena dilakukan normalisasi nilai RGB, serta karena nilai histogram >=0 sehingga dot product dan besar vektor juga akan selalu positif, sehingga nilai yang didapat dari *cosine similarity* akan berkisar 0 hingga 1. Jika suatu gambar memiliki kemiripan sama persis, maka nilainya 1. Dari ini, kita dapat mengalikan 100% untuk hasil *cosine similarity.* Sehingga jika nilainya 0,72, berarti kemiripan 2 buah vektor, tepatnya 2 buah gambar yaitu 72%.

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

Kesimpulannya yaitu dengan memanfaatkan salah satu elemen pada aljabar geometri yakni *cosine similarity*, kita dapat menentukan kemiripan dari 2 buah histogram dengan merepresentasikannya dalam vektor. Dengan pemahaman dan penelusuran lanjut, elemen aljabar geometri akan dapat menyelesaikan berbagai permasalahan.

2. Pemetaan Masalah pada CBIR dengan parameter tekstur

Pada CBIR menggunakan parameter tekstur dapat diidentifikasi beberapa elemen aljabar geometri. Pertama, perubahan 3 value RGB menjadi 1 value *grayscale* serupa dengan perkalian *inner product* antara vektor berisi nilai RGB dan vektor c = $0.299e_1 + 0.587e_2 + 0.114e_3$.

Kemudian, pada tahap berikutnya dilbentuk *Co-Occurrence Matrix* yang juga diinterpretasikan sebagai vektor dengan 256*256 dimensi.

Dari *Co-Occurrence Matrix* tersebut dapat diekstraksi Contrast, Homogeneity, dan Entropynya.

Ekstraksi Contrast menggunakan rumus berikut :

Contrast:

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

Dapat diinterpretasikan sebagai elemen Aljabar Geometri, yaitu inner product antara vektor *Co-Occurrence Matrix* dengan dimensi 65536 dengan sebuah vektor berdimensi 65536 :

$$v = \sum_{i=1}^{65536} \left(\left\lfloor \frac{i}{256} \right\rfloor - (i \bmod 256)\right)^2 e_i$$

Terakhir, nilai contrast, homogenitas, dan entropi dibentuk menjadi sebuah vektor berdimensi 3 yang juga dihitung kemiripannya dengan cosine similarity.

Contoh Ilustrasi Kasus Dan Penyelesaiannya

1. **Ilustrasi Kasus Dan Penyelesaian CBIR Dengan Parameter Warna**

   CBIR dengan parameter warna dapat menyelesaikan berbagai persoalan. Salah satunya yaitu pengelompokan gambar dari segi warna. Fitur ini dapat digunakan berbagai pengguna, contohnya dapat juga digunakan bagi orang yang tidak mengenal warna untuk mengetahui seberapa mirip warna antara 2 objek yang ada.

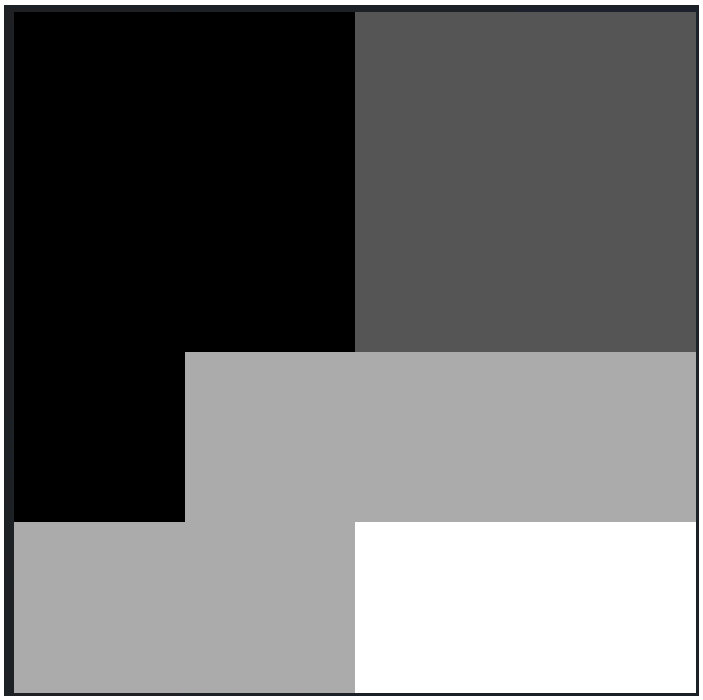
   Anggaplah terdapat 2 buah gambar dengan ukuran 3x3 pixel. Seorang pengguna akan dapat mengukur kemiripan kedua gambar dengan metode CBIR dengan parameter warna apabila dia merasa bahwa faktor warna cukup penting dalam penentuan kemiripin. Contohnya seperti ingin membandingkan binatang yang tinggal di wilayah yang sama. Seperti harimau dan singa yang memiliki pola warna yang serupa, ataupun binatang yang tinggal di wilayah bersalju sehingga memiliki warna putih. Ini terjadi karena keduanya akan memiliki polawarna yang sama sehingga histogram warna mereka akan memiliki kemiripan yang tinggi.

   Ilustrasi kasus dapat dijelaskan dengan seorang petualang yang ingin mencari tahu ada berapa banyak binatang yang memiliki warna yang sama, atau umumnya tinggal di daerah dengan musim yang sama. Ia akan dapat menggunakan metode CBIR dengan parameter warna untuk mencari hewan dengan pola warna serupa dari dataset.Asumsikan ukuran foto 3x3. Dengan mengekstrak nilai RGB dan mengubah ke nilai HSV, akan dihasilkan 18 *increment* yang dapat ditambahkan ke bin. Misalkan gambar 1 memiliki nilai HSV : [(317,0,0),(2,0,0),(317,1,1), (317,0,0),(2,0,0),(317,1,1), (317,0,0),(2,0,0),(317,1,1)] dan HSV gambar 2 bernilai : [(2,1,1),(317,1,1),(317,0.8,0.8), (2,1,1),(317,1,1),(317,0.8,0.8), (2,1,1),(317,1,1),(317,0.8,0.8)] maka akan menambahkan bin sesuai dengan format yang ada.Secara global, akan ada 1 bin dengan nilai 9 ((317,1,1) & (317,0.8,0.8)) dan 3 bin dengan nilai 3 ((317,0,0),(2,0,0),(2,1,1)) dan sisa 54 bin dengan nilai 0.

   Dalam kasus ini, karena berupa 3x3, pengisian seharusnya dilakukan secara blok sehingga untuk 1 pixel akan dibandingkan dengan pixel lainnya pada posisi serupa, contohnya (317,0,0) pada gambar pertama akan langsung

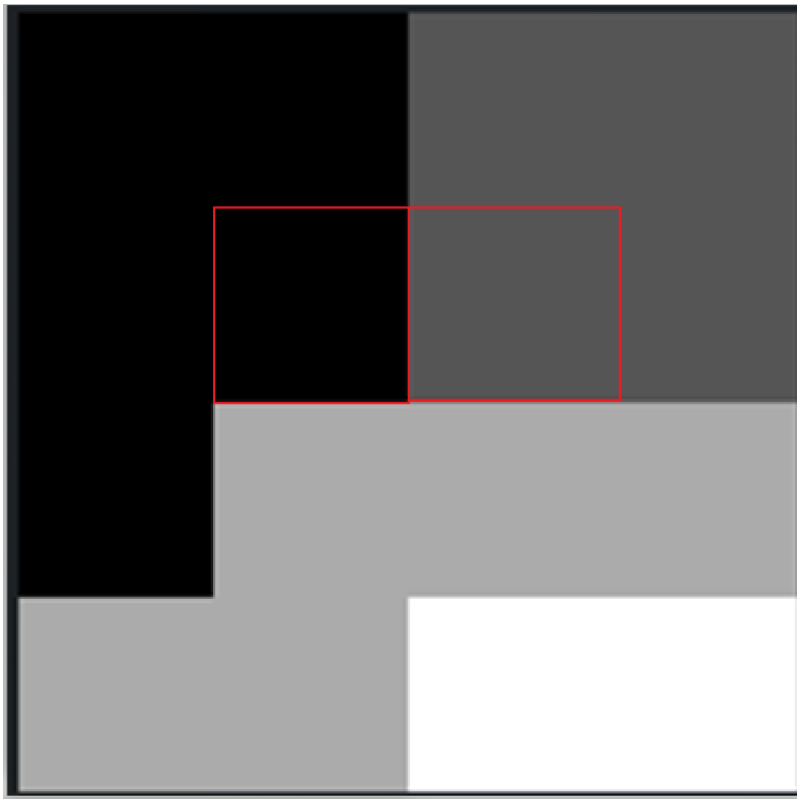
dibandingkan dengan (2,1,1) vektor kedua. Perbandingan akan dilakukan terus menerus, sampai 9 kali, dengan bagian tengah dan bagian atas dan bawah tengah memiliki bobot tertentu guna meningkatkan akurasi. Hasil dari pembagian jumlah bobotlah penentu kemiripan. Dari setiap gambar, akan diambil yang memiliki kemiripan diatas 60%. Itulah yang akan ditampilkan dan dilihat oleh sang petualang.

2. **Ilustrasi Kasus Dan Penyelesaian CBIR Dengan Parameter Tekstur**
   Contoh salah satu kasus perbandingan dengan CBIR Tekstur adalah perbandingan kedua gambar grayscale berikut :



Gambar tersebut akan diproses oleh library ekstraksi image pada bahasa C menjadi nilai RGB terlebih dahulu. Kemudian, nilai RGB yang didapat akan dikonversikan menjadi nilai Grayscale sesuai dengan rumus konversi.

Setelah itu, program C akan mengiterasi seluruh pasangan pixel yang bersebelahan secara horizontal, dan menginkremen nilai *Co-Occurence Matrix* pada indeks (x, y) dengan x adalah nilai grayscale, pixel yang di kiri dan y adalah nilai grayscale pixel yang di kanan.



Contohnya, pada gambar yang diberikan, saat program membaca nilai grayscale pada pixel yang diberi outline merah, yang memiliki nilai grayscale 0 (kiri) dan 85 (kanan), maka *Co-Occurence Matrix* pada indeks (0, 85) akan di-increment. Hal tersebut akan dilakukan untuk setiap pixel pada gambar.

Kemudian, *Co-Occurence Matrix* yang telah didapat dari proses tersebut akan ditambahkan dengan transpose dirinya sendiri agar terbentuk matriks yang simetris dan setiap elemennya akan dibagi jumlah dari semua elemennya. Proses ini disebut menormalisasi matriks tersebut.

Setelah itu, dari matriks *Co-Occurence* setiap nilai pixelnya diiterasi dan nilai Contrast, Homogeneity, dan Entropy diakumulasi sesuai dengan rumus masing-masingnya. Dari ketiga nilai tersebut dibentuk sebuah vektor berdimensi 3 yang disebut sebagai *feature vector*.

Perbandingan gambar dilakukan dengan menggunakan cosine similarity antara *feature vector* gambar pertama dengan *feature vector* gambar kedua.

BAB IV

IMPLEMENTASI DAN UJI COBA

Implementasi Program Utama

1. CBIR DENGAN PARAMETER WARNA

```
FUNCTION SEGMENT_IMAGE_INTO_3X3(IMAGE):
      WIDTH, HEIGHT <- IMAGE.SIZE
      SEG_WIDTH <- WIDTH // 3
      SEG_HEIGHT <- HEIGHT //3
      SEGMENTS <- []
      I TRAVERSAL [0..2]
            J TRAVERSAL [0..2]
                  LEFT <- I*SEG_WIDTH
                  UPPER <- J*SEG_HEIGHT
                  IF (I!=2) THEN
                        RIGHT <- (I+1)*SEG_WIDTH
                  ELSE
                        RIGHT <- WIDTH
                  IF (J!=2) THEN
                        LOWER <- (J + 1)*SEG_HEIGHT
                  ELSE:
                        LOWER <- HEIGHT
      #CROP IMAGES BY ITS (X,Y) THEN APPEND TO LIST
      SEGMENTS.APPEND(IMAGE.CROP((LEFT, UPPER, RIGHT, LOWER)))
      -> SEGMENTS


FUNCTION RGB_TO_HSV(IMAGE)
#ACCEPTS IMAGE AS AN NUMPY ARRAY OF RGB
     IMAGE <- IMAGE/255
#SPLIT ARRAY TO R,G, AND B, RESPECTIVELY
     R, G, B <- IMAGE[..., 0], IMAGE[..., 1], IMAGE[..., 2]
     CMAX <- NP.MAX(IMAGE, AXIS=-1)
     CMIN <- NP.MIN(IMAGE, AXIS=-1)
     DELTA <- CMAX - CMIN
     H <- NP.ZEROS_LIKE(CMAX)
     MASK <- DELTA > 0
```

```
     IDX_R <- (CMAX = R) & MASK

     IDX_G <- (CMAX = G) & MASK

     IDX_B <- (CMAX = B) & MASK

     H[IDX_R] <- ((G[IDX_R] - B[IDX_R]) / DELTA[IDX_R]) MOD 6

     H[IDX_G] <- ((B[IDX_G] - R[IDX_G]) / DELTA[IDX_G]) + 2

     H[IDX_B] <- ((R[IDX_B] - G[IDX_B]) / DELTA[IDX_B]) + 4

     H <- (H * 60) MOD 360

     S <- NP.WHERE(CMAX = 0, 0, DELTA / CMAX)

     V <- CMAX

#STACK ALL H,S, AND V INTO A NUMPY ARRAY

            -> NP.STACK((H, S, V), AXIS=-1)


FUNCTION VECTOR(IMAGE):

     IMAGE <- IMAGE.CONVERT('RGB')

     IMAGE <- NP.ARRAY(IMAGE)

     HSVNUMPY <- RGB_TO_HSV(IMAGE)

     HBIN <- NP.ARRAY([(316, 360), (1, 26), (26, 41), (41, 121), (121,
191), (191, 271), (271, 295),(295,316)])

     SBIN <- NP.ARRAY([(0, 0.2), (0.2, 0.7), (0.7, 1.1)])

     VBIN <- NP.ARRAY([(0, 0.2), (0.2, 0.7), (0.7, 1.1)])

     HISTOGRAM <- NP.ZEROS((LEN(HBIN), LEN(SBIN), LEN(VBIN)),
DTYPE='INT64')

     H_MASKS <- [((HBIN[I][0] <= HSVNUMPY[:, :, 0]) & (HSVNUMPY[:, :, 0]
< HBIN[I][1])) | (HSVNUMPY[:, :, 0] == 0) IF I == 0 ELSE (HBIN[I][0] <=
HSVNUMPY[:, :, 0]) & (HSVNUMPY[:, :, 0] < HBIN[I][1]) I TRAVERSAL
[0..LEN(HBIN)]]

      S_MASKS = [(SBIN[J][0] <= HSVNUMPY[:, :, 1]) & (HSVNUMPY[:, :, 1] <
SBIN[J][1]) J TRAVERSAL [0..LEN(SBIN)]]

     V_MASKS = [(VBIN[K][0] <= HSVNUMPY[:, :, 2]) & (HSVNUMPY[:, :, 2] <
VBIN[K][1]) K TRAVERSAL [0..LEN(VBIN)]]

     I TRAVERSAL [0..LEN(HBIN)]

         J TRAVERSAL [0..LEN(SBIN)]

              K TRAVERSAL [0..LEN(VBIN)]:

                  HISTOGRAM[I, J, K] <- NP.SUM(H_MASKS[I] & S_MASKS[J] &
V_MASKS[K])

      -> HISTOGRAM.FLATTEN()


FUNCTION MANUAL_DOT_PRODUCT(VEC_A, VEC_B)

      -> NP.int64(NP.SUM(VEC_A * VEC_B))
```

```
FUNCTION MANUAL_NORM(VEC):
      IF ORD IS NONE AND AXIS IS NONE:
              NORM_SQUARED <- NP.SUM(VEC ** 2)
              -> NP.SQRT(NORM_SQUARED).ASTYPE(NP.INT64)
          #INI HANYA DI RUN JIKA SUDAH MELEWATI BATAS, KARENA NILAI
      DIANGGAP INF JIKA MANUAL, SECARA KESEHARIAN MASIH GUNAKAN YANG
      MANUAL
          ELIF AXIS IS NONE:
              -> NP.LINALG.NORM(VEC, ORD=ORD).ASTYPE(NP.INT64)
          ELSE:
              -> NP.LINALG.NORM(VEC, ORD=ORD, AXIS=AXIS).ASTYPE(NP.INT64)


FUNCTION MANUAL_COSINE_SIMILARITY(VEC_A, VEC_B, NORM_ORDER=NONE,
NORM_AXIS=NONE)
    IF NP.ALL(HIST1 = 0) AND NP.ALL(HIST2 = 0):
        -> 1
    DOT_PRODUCT <- MANUAL_DOT_PRODUCT(HIST1, HIST2)
    NORM_HIST1 <- MANUAL_NORM(HIST1, ORD=NORM_ORDER, AXIS=NORM_AXIS)
    NORM_HIST2 <- MANUAL_NORM(HIST2, ORD=NORM_ORDER, AXIS=NORM_AXIS)
    IF NORM_HIST1 != 0 AND NORM_HIST2 != 0:
        -> DOT_PRODUCT / (NORM_HIST1 * NORM_HIST2)
    ELSE:
         -> 0
FUNCTION CALCULATE_WEIGHTED_COSINE_SIMILARITY(HISTOGRAMS1, HISTOGRAMS2)
     I TRAVERSAL [1..9]
         IF (I = 4) THEN
              WEIGHTS = 5
         ELIF (I IN (3,5)) THEN
              WEIGHTS = 3
         ELSE
              WEIGHTS = 1
     TOTAL_SIMILARITY <- 0
     I TRAVERSAL [1..9]
         SIMILARITY <- MANUAL_COSINE_SIMILARITY(HISTOGRAMS1[I],
HISTOGRAMS2[I])
         WEIGHTED_SIMILARITY <- SIMILARITY * WEIGHTS[I]
         TOTAL_SIMILARITY <- TOTAL_SIMILARITY + WEIGHTED_SIMILARITY
```

```
     OUTPUT(TOTAL_SIMILARITY / SUM(WEIGHTS) *100)
      -> TOTAL_SIMILARITY / SUM(WEIGHTS) *100

FUNCTION CACHING(JSON_FILE_PATH, INPUT_HISTOGRAM):
    TRY:
        OPEN JSON FILE TO READ
    EXCEPT EXCEPTION AS E:
        OUTPUT(F"ERROR READING JSON FILE: {E}")
        RETURN NONE
    RESULTS = []
    ITERATE FOR EVERY (IMAGE_PATH, IMAGE_DATA) IN DATA.ITEMS():
        IMAGE_VECTOR <- NP.ARRAY(IMAGE_DATA["IMAGE_VECTORS"])
        SIMILARITY <-
CALCULATE_WEIGHTED_COSINE_SIMILARITY(INPUT_HISTOGRAM, IMAGE_VECTOR)
        RESULTS.APPEND([IMAGE_PATH,SIMILARITY])
    -> RESULTS

FUNCTION CREATEHISTOGRAM(FOLDER, JSON_FILE):
    CLEAR_JSON(JSON_FILE)
    DATA<-{}
    ITERATE THROUGH EVERY FILE IN FOLDER:
        IMAGE_PATH <- OS.PATH.JOIN(FOLDER, FILENAME)

        IF (IMAGE PATH IS FILE AND ENDS WITH .JPG|.PNG|.JPEG) THEN
            HISTOGRAM <- CREATE_HISTOGRAMS_FOR_SEGMENTS(IMAGE_PATH)
            RESULTS <- {
                "IMAGE_PATH": IMAGE_PATH,
                "IMAGE_VECTORS": NUMPY_ARRAY_TO_LIST(HISTOGRAM)
            }
            DATA[IMAGE_PATH] <- RESULTS
    APPEND TO JSON

FUNCTION CREATE_HISTOGRAMS_FOR_SEGMENTS(IMAGE_PATH):
    OPEN IMAGE PATH AND STORE AT 'IMAGE' VARIABLE
    SEGMENTS <- SEGMENT_IMAGE_INTO_3X3(IMAGE)
    HISTOGRAMS <- [VECTOR(SEGMENT) FOR EVERY SEGMENT IN SEGMENTS]
```

```
    -> NP.ARRAY(HISTOGRAMS)
#JSON FUNCTION
FUNCTION NUMPY_ARRAY_TO_LIST(NUMPY_ARRAY):
    -> NUMPY_ARRAY.TOLIST()

FUNCTION READ_VECTORJSON(FILE_PATH, INDEX):
    OPEN JSON FILE ACCORDING TO PATH AND LOAD TO 'DATA'
    WITH OPEN(FILE_PATH, 'R') AS JSON_FILE:
    IF (INDEX < LEN(DATA)) THEN
        -> DATA[INDEX]['IMAGE2_VECTORS']
    ELSE:
        OUTPUT("INDEX OUT OF RANGE.")
        -> NONE

FUNCTION APPEND_TO_JSON(FILE_PATH, NEW_DATA)
    IF FILE PATH EXIST, OPEN THE JSON AND LOAD INSIDE "DATA"
    APPEND 'NEW_DATA' INSIDE DATA
    IF DONT EXIST, CREATE NEW 'DATA' WITH 'NEW_DATA' AS FISRT ELEMENT
    OPEN FILE PATH FOR WRITING THE NEW 'DATA' INSIDE JSON

FUNCTION CLEAR_JSON(FILE_PATH)
    OPEN THE JSON FILE AND WRITE VALUE TO [], WHICH IS EMPTY FORMAT
```

2. CBIR DENGAN PARAMETER TEKSTUR

```
FUNCTION RGBTOGRAYSCALE(INTEGER R, INTEGER G, INTEGER B) -> INTEGER
    -> 0.299*R + 0.587*G + 0.114*B


PROCEDURE CONSTRUCTCOOCCURENCEMATRIX(INPUT : IMAGE DATA[0..3*X*Y - 1],
INPUT : INTEGER X, INPUT : INTEGER Y, OUTPUT : ARRAY OF INTEGER
RESULT[0..256*256 - 1])


    INTEGER : I, J, FIRSTID, SECONDID


    I TRAVERSAL[0..255]
        J TRAVERSAL[0..255]
            RESULT[256*I + J] = 0


    I TRAVERSAL[0..Y-1]
        J TRAVERSAL[0..X-2]
            FIRSTID = RGBTOGRAYSCALE(
                DATA[3*(I*X + J) + 0],
                DATA[3*(I*X + J) + 1],
                DATA[3*(I*X + J) + 2]
            )
            SECONDID = RGBTOGRAYSCALE(
                DATA[3*(I*X + J + 1) + 0],
                DATA[3*(I*X + J + 1) + 1],
                DATA[3*(I*X + J + 1) + 2]
            )
            RESULT[256*FIRSTID + SECONDID] += 1



PROCEDURE CONSTRUCTNORMALIZEDOCCMATRIX(INPUT : IMAGE DATA[0..3*X*Y - 1],
INPUT : INTEGER X, INPUT : INTEGER Y, OUTPUT : ARRAY OF INTEGER
RESULT[0..256*256 - 1])


    ARRAY OF INTEGER OCCMATRIX[65536]
    CONSTRUCTCOOCCURENCEMATRIX(IMG, X, Y, OCCMATRIX)
```

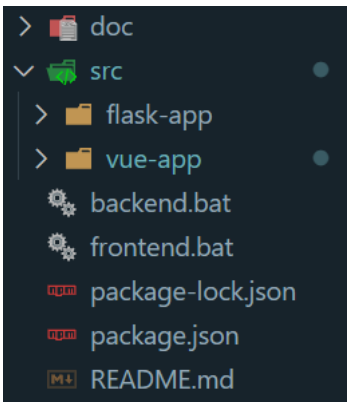
```
    REAL SUM = 2 * (X - 1) * Y

    INTEGER I, J
    I TRAVERSAL[0..255]
        J TRAVERSAL[0.255]
            RESULT[256*I + J] =(OCCMATRIX[256*I + J] + OCCMATRIX[256*J +
I])/SUM



PROCEDURE GETCHE(INPUT : IMAGE IMG, INPUT : INTEGER X, INPUT : INTEGER Y,
OUTPUT : REAL C, OUTPUT : REAL H, OUTPUT : REAL E)
    OCCMATRIX[65536]
    CONSTRUCTNORMALIZEDOCCMATRIX(IMG, X, Y, OCCMATRIX)
    INTEGER I, J
    REAL EL
    C = 0
    H = 0
    E = 0
    I TRAVERSAL[0..255]
        J TRAVERSAL[0..255]
            EL = OCCMATRIX[256*I + J]
            C += EL*(I - J)*(I - J)
            H += EL/(1 + ((I-J)*(I - J)))
            IF (EL != 0) THEN
                E += EL * -LOG2(EL)

FUNCTION COSINESIMILARITY(REAL V1, REAL V2, INT N) -> REAL
    REAL LENGTH1 = 0
    REAL LENGTH2 = 0
    REAL DOTPRODUCT = 0

    INTEGER I
    I TRAVERSAL[0..N-1]
        LENGTH1 += V1[I]*V1[I]
```

```
        LENGTH2 += V2[I]*V2[I]
        DOTPRODUCT += V1[I]*V2[I]

    LENGTH1 = AKAR(LENGTH1)
    LENGTH2 = AKAR(LENGTH2)

    -> DOTPRODUCT/(LENGTH1*LENGTH2)

FUNCTION COMPAREIMAGE(INPUT : STRING PATH1, INPUT : STRING PATH2) -> REAL
    INTEGER X1, Y1, N1
    INTEGER X2, Y2, N2
    IMAGE DATA1 = STBI_LOAD(PATH1, &X1, &Y1, &N1, 1)
    IMAGE DATA2 = STBI_LOAD(PATH2, &X2, &Y2, &N2, 1)
    ARRAY OF REAL V1[3]
    ARRAY OF REAL V2[3]
    GETCHE(DATA1, X1, Y1, V1[0], V1[1], V1[2])
    GETCHE(DATA2, X2, Y2, V2[0], V2[1], V2[2])
    STBI_IMAGE_FREE(DATA1)
    STBI_IMAGE_FREE(DATA2)
    -> COSINESIMILARITY(V1, V2, 3)

PROCEDURE INITIALIZEDATASET(STRING FOLDER_PATH)
    FILE FP
    STRING OUTPUT1
    INTEGER X1, Y1, N1
    ARRAY OF REAL V1[3]

    FP = OPEN("TXT/CHE.TXT", WRITE)

    DIRECTORY DP

    DP = OPENDIR (FOLDER_PATH)
    WHILE (NEXTFILEEXIST) DO
        STRING FILE_PATH
```

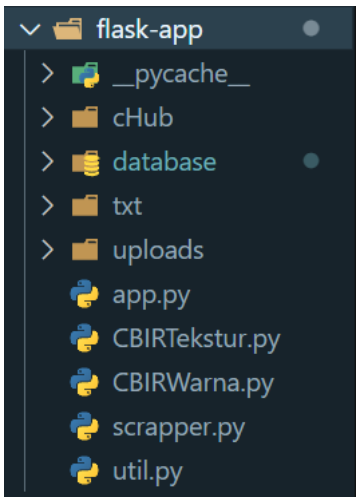
```
        IMAGE DATA1 = STBI_LOAD(FILE_PATH, X1, Y1, N1, 3)
        GETCHE(DATA1, X1, Y1, V1[0], V1[1], V1[2])
        WRITE(FP, C, H, E, FILE_PATH)
        STBI_IMAGE_FREE(DATA1)

    CLOSE(DP)


FUNCTION COMPAREWITHDATASET(IMAGE) -> ARRAY OF (REAL, FILE_PATH)
    FILE FP
    STRING OUTPUT1
    INTEGER X1, Y1, N1
    ARRAY OF REAL V1[3]

    FP = OPEN("TXT/CHE.TXT", WRITE)

    DIRECTORY DP

    DP = OPENDIR (FOLDER_PATH)
    WHILE (NEXTFILEEXIST) DO
        STRING FILE_PATH

        IMAGE DATA1 = STBI_LOAD(FILE_PATH, X1, Y1, N1, 3)
        GETCHE(DATA1, X1, Y1, V1[0], V1[1], V1[2])
        WRITE(FP, C, H, E, FILE_PATH)
        STBI_IMAGE_FREE(DATA1)

    CLOSE(DP)

    FILE FP
    STRING OUTPUT1
    INTEGER X1, Y1, N1
    ARRAY OF REAL V1[3]
```

```
    IMAGE DATA1 = STBI_LOAD(FILE_PATH, X1, Y1, N1, 3)
    GETCHE(DATA1, X1, Y1, V1[0], V1[1], V1[2])

    FP = OPEN("TXT/COMPAREDWITHDATASET.TXT", WRITE)

    FILE FDS
    STRING LINE
    INTEGER LEN = 0

    ARRAY OF REAL V2[3]
    STRING FILENAME[300]

    FDS = FOPEN("TXT/CHE.TXT", READ)

    WHILE (NEXTLINEEXIST)
        GETVALUE(REAL, REAL, REAL, STRING , V2[0], V2[1], V2[2],
FILENAME)
        REAL SIMILARITY = COSINESIMILARITY(V1, V2, 3)
        WRITE(FDS, SIMILARITY, FILENAME)
```

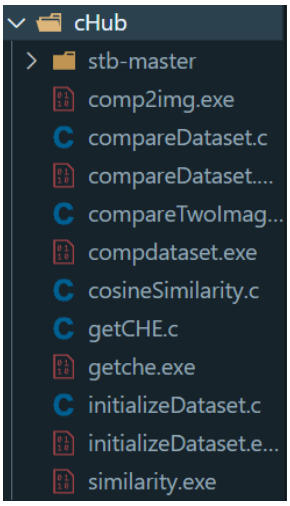
STRUKTUR PROJEK



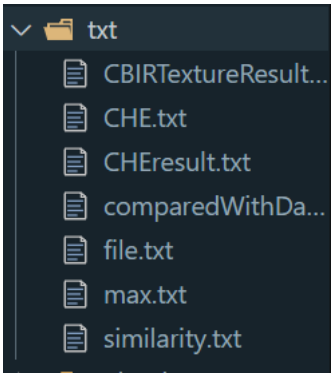
Gambar 4 Struktur utama repositori



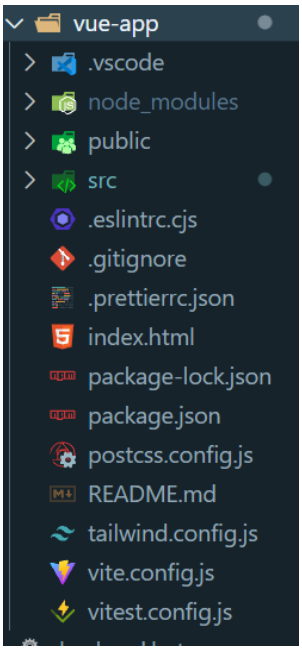
Gambar 5 Struktur flask-app



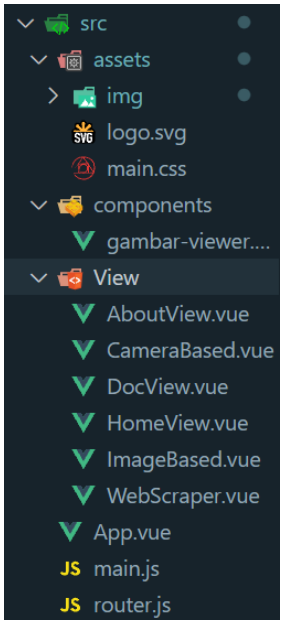
Gambar 6 Struktur cHUB dari flask



Gambar 7 Struktur folder txt dari flask



Gambar 8 Struktur vue-app



Gambar 9 Struktur src dari vue-app

Projek ini dibuat menggunakan panduan *Vue.js framework* dan *Flask API*. Seluruh source code yang kami buat telah disimpan di dalam folder "src". Didalam folder "src", terdapat dua aplikasi yaitu "flask-app" dan "vue-app". "flask-app" berisi source code yang menangani *backend* dari projek ini. "vue-app" berisi source code yang menangangi frontend dari projek ini. Projek ini dapat dijalani dengan menjalani dua file beformat *".bat"* pada terminal yang berbeda, yaitu "frontend.bat" dan "backend.bat".

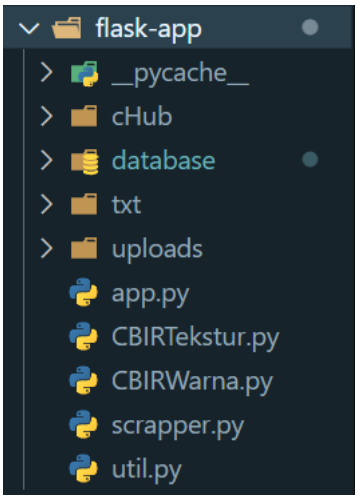
Struktur Repositori

Folder ”doc” berisi laporan dari Tugas Besar 2 Aljabar Linear dan Geometri dalam bentuk ”.pdf”. Fold ”src” berisi tentang seluruh source code projek dalam tugas ini. Bahasa yang digunakan dalam projek ini adalah Javascript dan C. File *backend.bat* dan *frontend.bat* merupakan shell script yang digunakan untuk menjalankan projek. File *README.md* merupakan markdown yang menyimpan informasi yang diperlukan atas projek ini.

Folder ”src”

Folder ini berisi dua folder utama, yaitu vue-app dan flask-app. Folder vue-app berperan untuk menyimpan seluruh source code frontend. Folder flask-app berperan untuk menyimpan seluruh source code backend.

Folder flask-app menyimpan folder *cHub, txt,* dan *uploads*. Selain itu, terdapat juga file python antara lain *app.py, CBIRTekstur.py,CBIRWarna.py,scrapper.py,tempCodeRunnerFile.py,util.py*.

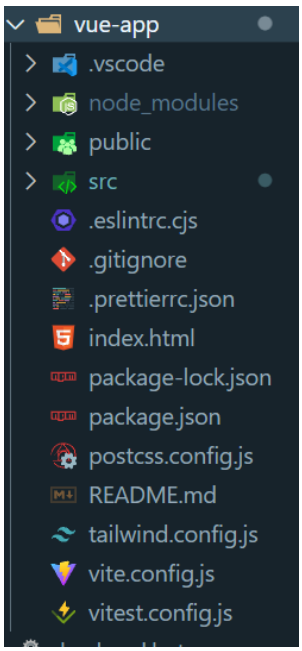


Gambar 10 Struktur flask-app

- Folder cHub merupakan folder yang berisi file dengan extension .c, library yang digunakan, beserta hasil compilenya dengan extension .exe.
- Folder txt merupakan folder yang berisi file txt yang diwrite oleh program dalam bahasa C dan di read oleh program python.
- Folder uploads merupakan gambar yang di input oleh pengguna.

- Folder database mennyimpan seluruh database dari projek ini. Folder ini hanya berisi gambar saja.
- *"app.py"* merupakan file yang menjalankan Flask.
- *"CBIRTekstur"* merupakan file yang berisikan seluruh fungsi / algoritma untuk menjalankan CBIR berbasis tekstur
- *"CBIRWarna"* merupakan file yang berisikan seluruh fungsi / algoritma untuk menjalankan CBIR berbasis warna
- *"scrapper.py"* merupakan file yang berisikan seluruh fungsi / algoritma untuk *website image scraping.*
- *"tempCodeRunner.py"* merupakan file yang muncul jika terjadi error pada flask.
- *"util.py"* merupakan file yang berisi seluruh operasi untuk menambahkan/menghapus folder atau file dalam repositori.

Folder "vue-app" menyimpan folder node_modules,public, dan src. Terdapat pula file pendukung untuk menyokong vue.js dalam melakukan kerjanya.

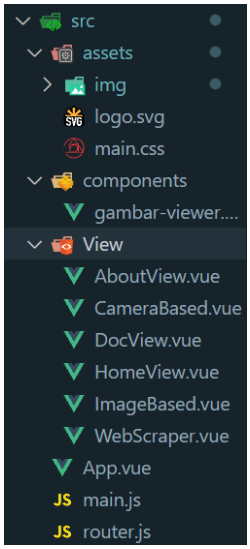


Gambar 11 Struktur vue-app

- Folder *".vscode"* dan *"node_modules"* merupakan penyokong dari vue.js. sehingga dapat diabaikan.

- Folder public digunakan untuk menyimpan asset statis yang tidak diproses oleh webpack.
- Folder src berisi seluruh source code frontend yang digunakan projek ini.
- File yang lainnya merupakan file penyokokng dari vue,js sehingga dapat diabaikan.

Konten dari folder src didalam vue-app adalah sebagai berikut



Gambar 12 Isi dari folder src vue-app

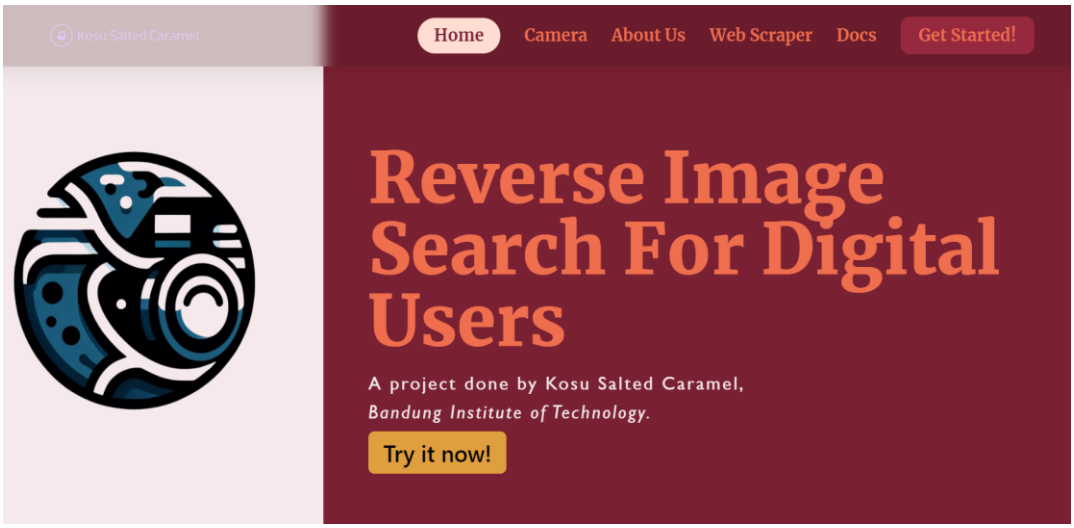
- Folder assets meruapakan seluruh asset statis, seperti gambar dan icon, yang digunakan dalam front-end.
- Folder components berisi seluruh komponen komponen yang dapat digunakan berkali-kali / *reusable* dalam layout web.
- Folder View berisi seluruh page yang ada pada website ini.
- File *"app.vue"* merupakan file utama yang menyambung component dan view pada projek.
- File *"main.js"* berguna untuk menyambungkan file dalam bentuk *".vue"* ke html.
- File *"router.js"* berguna untuk menyediakan routing pada projek ini.

Frontend

Frontend dari projek ini menggunakan framework “Vue.js”. Website yang dibuat memiliki beberapa page, yaitu:

1. Home
2. Image
3. Camera
4. About Us
5. Web Scraper
6. Docs

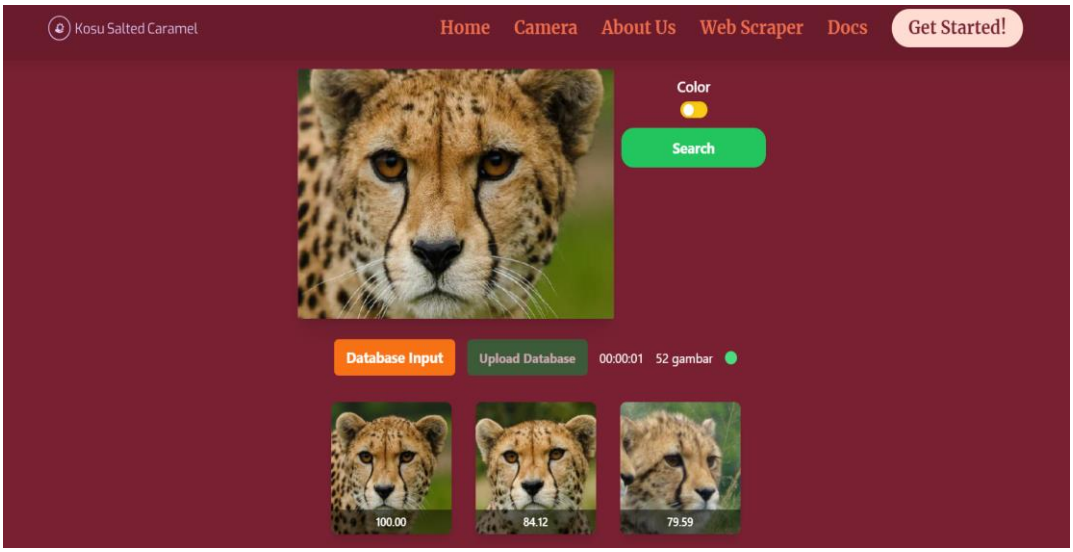
Home



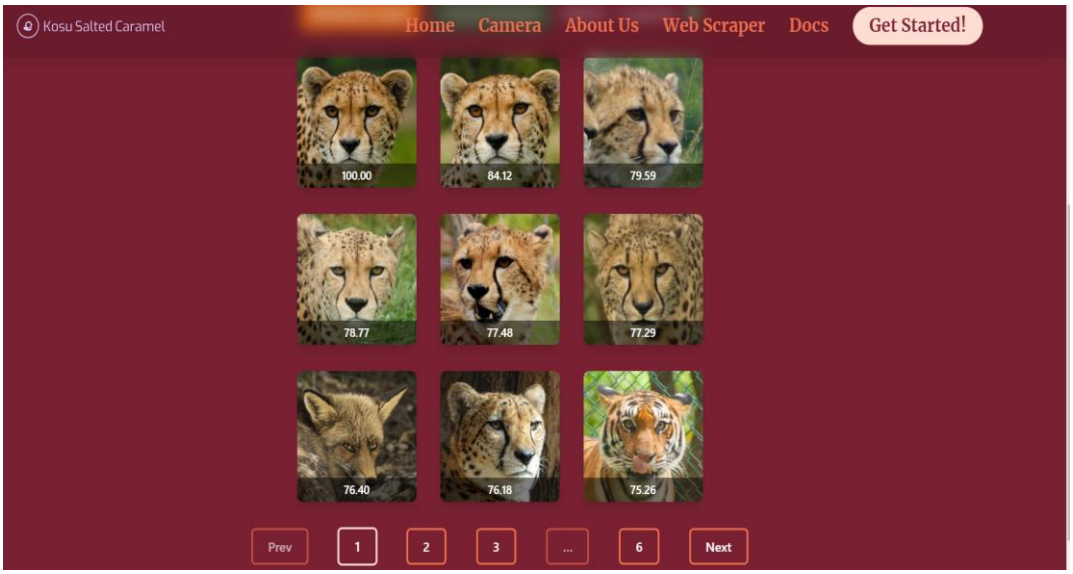
Gambar 13 Home Page

Laman “Home” merupakan tempat interaksi pengguna pertama kali terhadap website ini. Konten dari laman ini adalah penjelasan singkat dari tujuan dari website ini.

Image



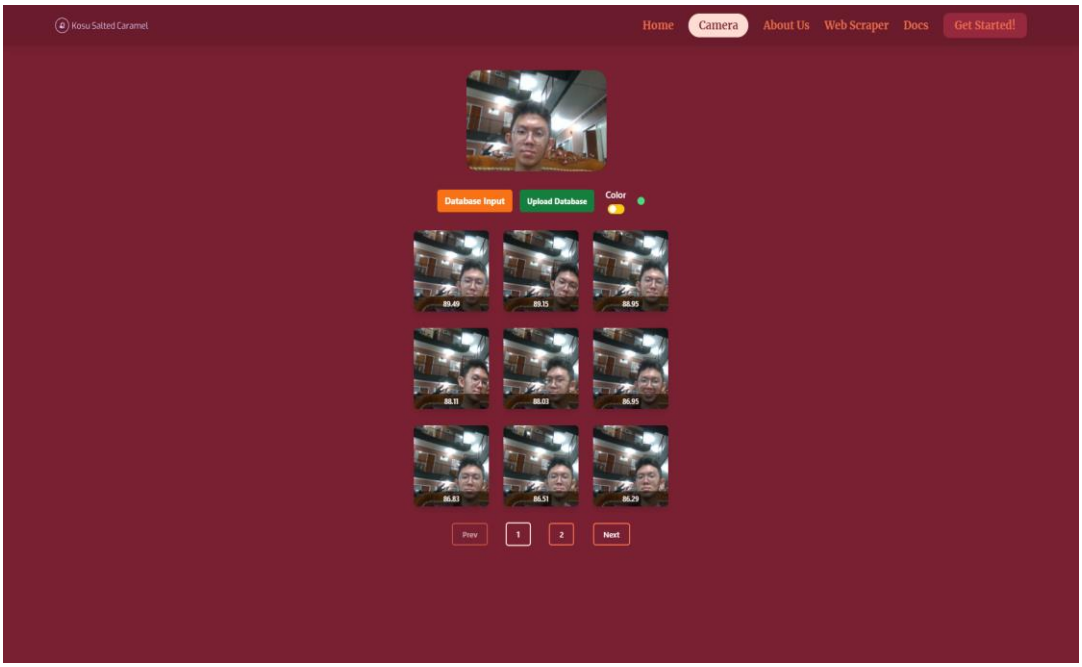
Gambar 14 User-Input Page (1)



Gambar 15 User-Input Page (2)

Laman “Image” merupakan tempat pengguna untuk memasukkan gambar dan database pilihannya untuk melakukan perbandingan CBIR berbasis warna. Terdapat *time-stamp* dan *status light* yang menyatakan lama pemrosesan dan status pemrosesan.

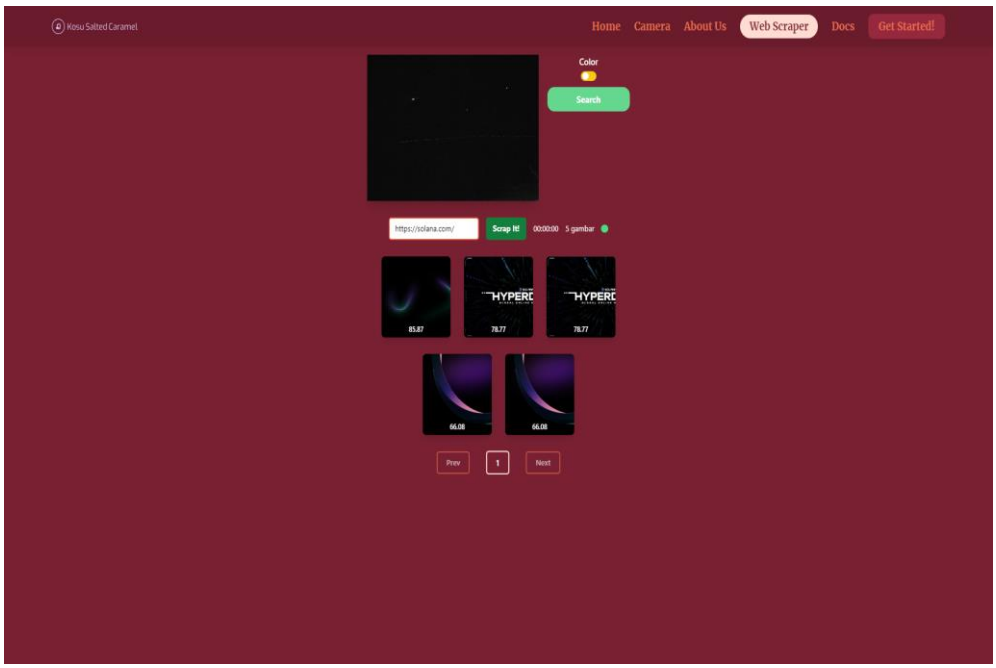
Camera



Gambar 16 Camera Page

Laman Camera serupa dengan Laman Image. Alih-alih meminta user untuk mengupload gambar input pilihannya, laman ini menggunakan kamera yang terhubung ke pengguna dan membandingkannya secara realtime terhadap database yang dimasukkan.

Web Scraper



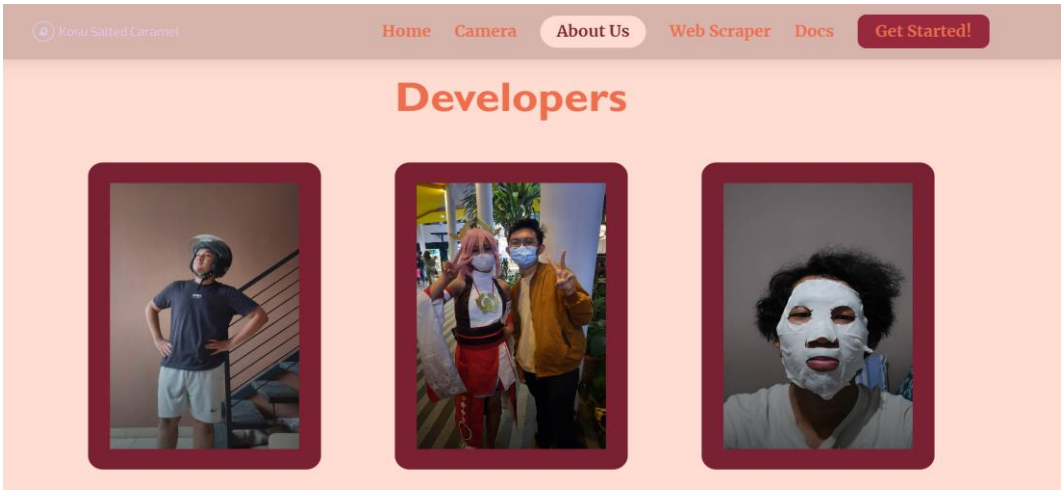
Gambar 17 Laman Web Scraper

Laman Camera serupa dengan Lama Image. Page ini mengganti tombol “Database Input” dengan input text yang berisi URL dari website yang diinginkan. Tombol “Scrap It!” mengambil seluruh gambar dari URL yang dimasukkan kedalam database.

About Us

Laman kami sebagai developer dalam mengaspirasikan serta mempromosikan diri kami sebagai seorang pembuat website. Dalam laman ini tercantum identitas kami serta tautan menuju media komunikasi kami seperti Github dan Instagram.

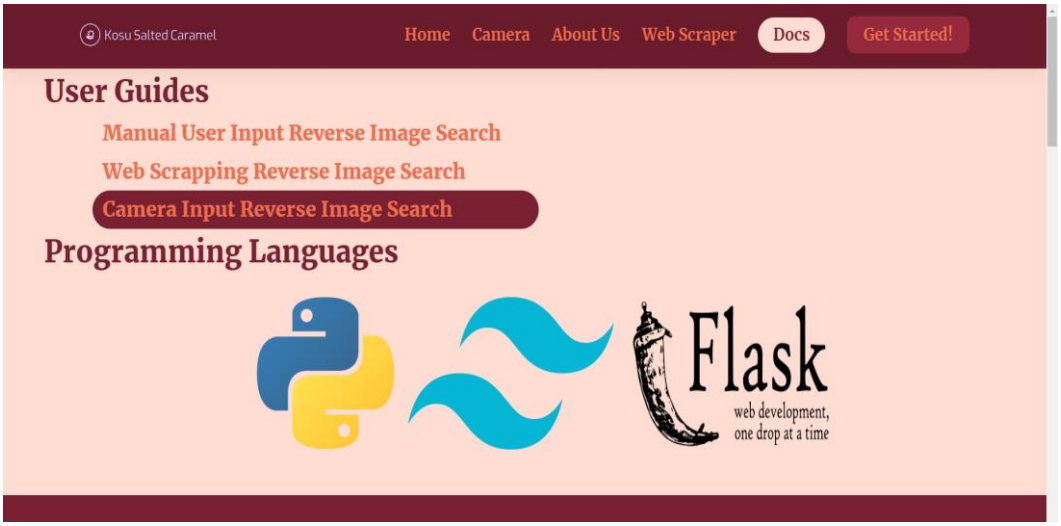
Gambar 18 About Us Page (1)



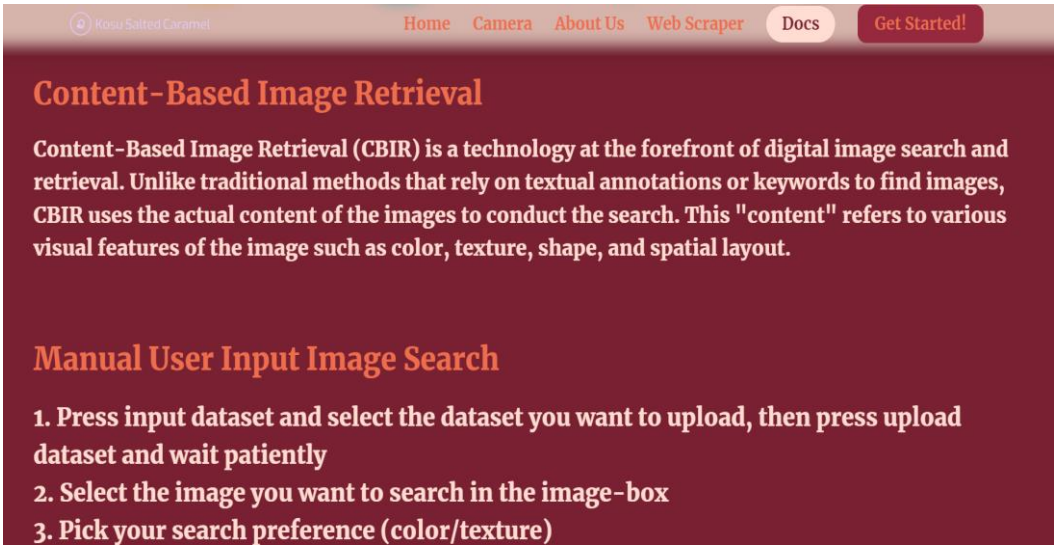
Gambar 19 About Us Page (2)

Docs

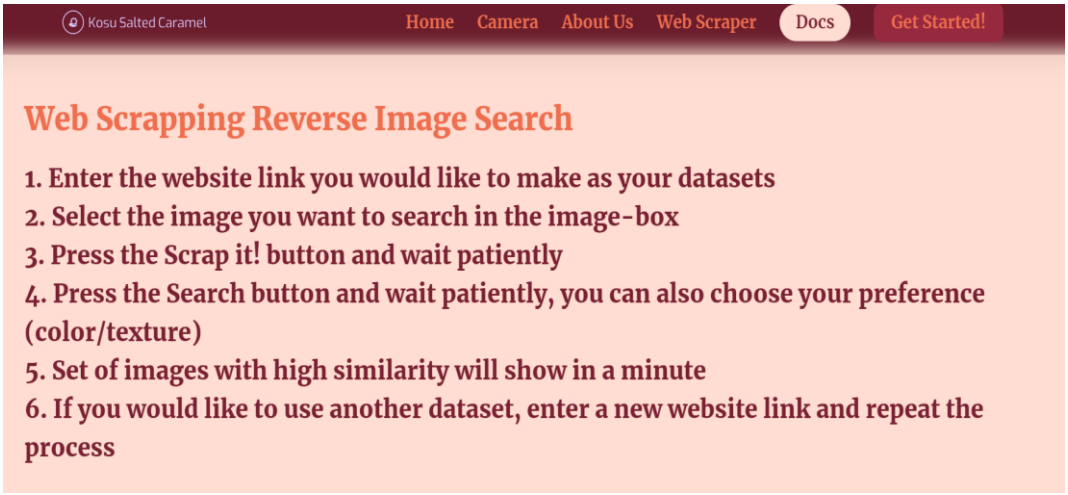
Laman ini berisi segala informasi mengenai website, mulai dari peralatan pengembangan website, sampai dengan panduan untuk tiap page pada website kami. Panduan dijelaskan dalam poin dan ditulis sejelas mungkin guna memaksimalkan pengalaman pengguna.



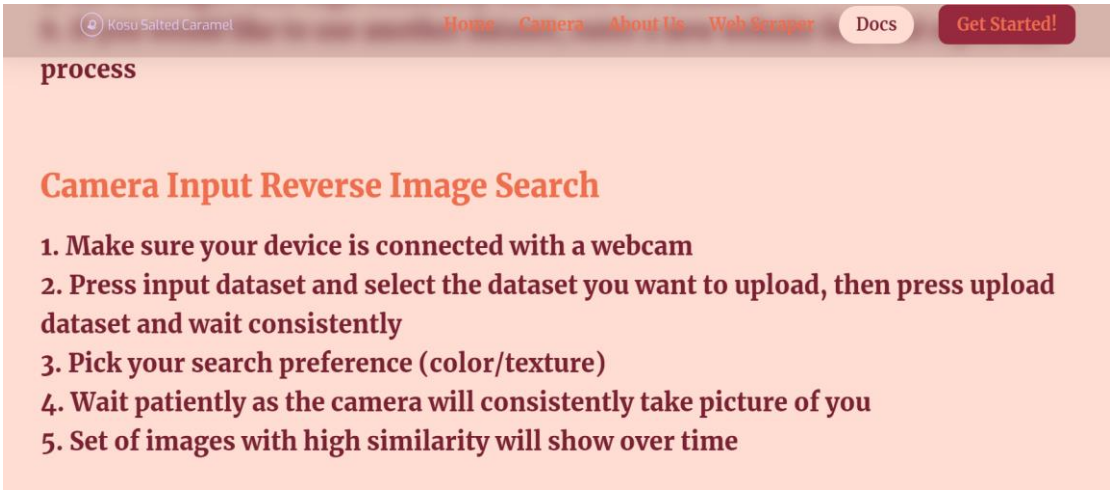
Gambar 20 Guides Page (1)



Gambar 21 Guides Page (2)



Gambar 22 Guides Page (3)



Gambar 23 Guides Page (4)

Backend

Backend dari projek ini menggunakan “flask” sehingga hampir seluruh source code untuk backend dari projek ini memiliki bahasa python. Seluruh routing yang akan dikirmkan dari frontend akan di handle dengan berbagai fungsi. Berikut merupakan seluruh library yang diimport dalam projek ini.

```
1. from flask import Flask,request,jsonify
2. from flask_cors import CORS
3. from werkzeug.utils import secure_filename
4. from io import BytesIO
5. import os
6. import base64
7. from PIL import Image
8.
9. # Import library buatan
10. from util import deleteFolderContent
```

```
11. from CBIRTekstur import compareImage
12. from CBIRWarna import *
13. from scrapper import scrape_images
```

Berikut merupakan set-up dari flask pada projek ini.

```
1. # Flask Setup
2. app = Flask(__name__)
3. CORS(app,supports_credentials=True)
4. app.config["DEBUG"] = True
5. app.config['UPLOAD_FOLDER'] = "uploads"
6. app.config['DB_FOLDER'] = "../vue-app/src/assets/img"
```

Content Based Image Retrieval Warna

Routing yang ditangani oleh flask dalam meminta request untuk melakukan CBIR Warna dibagi menjadi dua. Request yang didapati dalam bentuk “FolderStorage” atau request dalam bentuk “base64”. Request dalam bentuk “FolderStorage” didapatkan ketika pengguna melakukan CBIR dalam bentuk image. Request dalam bentuk “base64” didapatkan ketika pengguna melakukan CBIR dalam bentuk camera.

Berikut merupakan penanganan dalam bentuk ”FileStorage”

```
 1. @app.route('/uploadColor',methods=['POST','GET'])
 2. def cbir_color_list():
 3.     data = []
 4.     if 'image' in request.files:
 5.         image = request.files['image']
 6.         imageinput_result = create_histograms_for_segments(image)
 7.         if image:
 8.             filename = secure_filename(image.filename)
 9.             for fileDB in os.listdir('../vue-app/src/assets/img') :
10.                 imageDB_result =
create_histograms_for_segments("../vue-app/src/assets/img/"+fileDB)
11.                 dataObject = {
12.                     'imageTitle': fileDB,
13.                     'similarity':
calculate_weighted_cosine_similarity(imageinput_result,imageDB_result)
14.                 }
15.                 data.append(dataObject)
16.     return jsonify(data)
```

Berikut merupakan penanganan dalam bentuk base64

```
@app.route('/uploadColorCamera',methods=['POST','GET'])
def cbir_color_list_camera():
    print("tes kamera")
    data = []
    base64_string = request.json['file']
    base64_list = base64_string.split(',')
    imgdata = base64.b64decode(base64_list[1])
    img = (BytesIO(imgdata))
    imageinput_result = create_histograms_for_segments(img)
    for fileDB in os.listdir('../vue-app/src/assets/img') :
        imageDB_result = create_histograms_for_segments("../vue-app/src/assets/img/"+fileDB)
        dataObject = {
            'imageTitle': fileDB,
            'similarity': calculate_weighted_cosine_similarity(imageinput_result,imageDB_result)
        }
        data.append(dataObject)
    return jsonify(data)
```

Content Based Image Retrieval Tekstur

Routing yang ditangani oleh flask dalam meminta request untuk melakukan CBIR Tekstur dibagi menjadi dua. Request yang didapati dalam bentuk “FolderStorage” atau request dalam bentuk “base64”. Request dalam bentuk “FolderStorage” didapatkan ketika pengguna melakukan CBIR dalam bentuk image. Request dalam bentuk “base64” didapatkan ketika pengguna melakukan CBIR dalam bentuk camera.

Berikut merupakan penanganan dalam bentuk ”FileStorage”

```
@app.route('/uploadTexture',methods=['POST','GET'])
def cbir_texture_list():
    data = []
    if 'image' in request.files:
        image = request.files['image']
        if image:
            filename = secure_filename(image.filename)
            filepath = os.path.join(app.config['UPLOAD_FOLDER'],filename)
```

```
            image.save(filepath)
            for fileDB in os.listdir('../vue-app/src/assets/img') :
                dataObject = {
                    'imageTitle': fileDB,
                    'similarity': float(compareImage(filepath,'../vue-app/src/assets/img/'+fileDB)) * 100
                }
                data.append(dataObject)
    return jsonify(data)
```

Berikut merupakan penanganan dalam bentuk ”base64”

```
@app.route('/uploadTextureCamera',methods=['POST','GET'])
def cbir_texture_list_camera():
    data = []
    base64_string = request.json['file']
    base64_list = base64_string.split(',')
    imgdata = base64.b64decode(base64_list[1])
    img = Image.open(BytesIO(imgdata))
    img.save("uploads/gambarTemp.png")
    for fileDB in os.listdir('../vue-app/src/assets/img') :
        dataObject = {
            'imageTitle': fileDB,
            'similarity': float(compareImage("uploads/gambarTemp.png",'../vue-app/src/assets/img/'+fileDB)) * 100
        }
        data.append(dataObject)
    os.remove("uploads/gambarTemp.png")
    return jsonify(data)
```

Database Input

Metode pengambilan database dalam projek ini dapat dibagi menjadi dua. Pertama, user dapat mengunggah database yang diinginkannya berupa folder yang berisi seluruh file bertipe gambar. Kedua, user dapat mengunggah / mendapati database dari website yang ingin di-*scrap*.

Kode berikut menangani pengunggahan database berupa folder

```
# Router for Database Upload
@app.route('/uploadDB',methods=['POST'])
def uploadDB():
    folder = request.files.getlist('files')
    deleteFolderContent("../vue-app/src/assets/img")
    for file in folder:
```

```
        if file:
            filename = secure_filename(file.filename)
            filename = filename.replace(" ","_")
            filepath = os.path.join(app.config['DB_FOLDER'],filename)
            file.save(filepath)
    return jsonify({'status': 'Succes'})
```

Kode berikut menangani penggungahan database menggunakan metode scraping

```
# Router for Scrapper
@app.route('/uploadScrap',methods=['POST'])
def uploadScrap():
    url = request.json['string']
    print(url)
    deleteFolderContent("../vue-app/src/assets/img")
    scrape_images(url,"../vue-app/src/assets/img")
    return jsonify({'status': 'Succes'})
```

Scraper

Metode scraping dalam projek ini diawali dengan menggunakan *requests* untuk mengambil konten halaman web. Kemudian dilanjutkan dengan penggunaan BeautifulSoup untuk mengurai html. Dalam proses ini, dicari tag <img> pada html dan dicarisumber foto. Sumber foto dapat berupa ”src” maupun ”data-src” dan masih banyak lagi namun dalam projek kali ini hanya kedua tipe diatas yang ditampilkan. Kemudian, dengan menggunakan pustaka urllib.parse dilakukan parsing (parse) URL gambar (image_url) menjadi komponen-komponennya seperti skema ("scheme"), lokasi jaringan ("netloc"), jalur ("path"), dan parameter kueri ("query"). Kemudian diperiksa apakah dalam parameter kueri dari URL gambar terdapat parameter dengan nama "url".

Tujuan dari langkah ini adalah untuk memeriksa apakah URL gambar telah disarangkan (nested) dalam parameter "url" dalam URL yang lebih besar. Jika ya, maka ini adalah URL gambar yang sebenarnya yang akan diunduh. Hal ini dilakukan untuk memastikan URL yang diunduh lengkap dan valid.

```
import requests
from bs4 import BeautifulSoup
import os
import re

```

```
def sanitize_filename(filename):
    # Replace invalid filename characters with underscores
    return re.sub(r'[<>:"/\\|?*]', '_', filename)

def get_extension_from_content_type(content_type):
    if 'image/jpeg' in content_type:
        return '.jpg'
    elif 'image/png' in content_type:
        return '.png'
    elif 'image/svg+xml' in content_type:
        return '.svg'
    else:
        return ''

def download_image(url, folder, base_filename):
    try:
        response = requests.get(url)
        if response.status_code == 200:
            content_type = response.headers.get('Content-Type', '')
            print(content_type)
            extension = get_extension_from_content_type(content_type)
            if (extension == '.jpg' or extension=='.png'):
                sanitized_filename = sanitize_filename(base_filename)
+ extension
                full_path = os.path.join(folder, sanitized_filename)
                with open(full_path, 'wb') as f:
                    f.write(response.content)
    except Exception as e:
        print(f"Error downloading {url}: {e}")

def scrape_images(url, save_folder):
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')
    images = soup.find_all('img')

    if not os.path.exists(save_folder):
        os.makedirs(save_folder)

    for idx, img in enumerate(images):
        src = img.get('src')
        if src:
            image_url = src if src.startswith('http') else url + src
            base_filename = f"image_{idx}"
            download_image(image_url, save_folder, base_filename)
        src = img.get('data-src')
        if src:
            image_url = src if src.startswith('http') else url + src
            base_filename = f"image_{idx}"
            download_image(image_url, save_folder, base_filename)
```

PENJELASAN TATA CARA PENGGUNAAN PROGRAM

1. **Untuk Metode User-Input**

   Langkah pertama dari penggunaa user input yaitu memasukkan dataset. Dataset dimasukkan pada bagian ”input dataset” dan jika sudah yakin dapat menekan tombol ”upload dataset”. Tunggu beberapa saat karena sistem sedang memproses gambar yang ada.

   Setelah bola kuning berubah menjadi hijau, maka artinya dataset sudah diupload dan image sudah boleh diinput. Pilih input gambar yang ingin dicari,

dan tekan tombol ”Search”. Gambar akan langsung keluar sesuai uruan kemiripin dengan syarat kemiripan diatas 60%

2. **Untuk Metode Website-Scraping**

   Langkah pertama yakni memasukkan input web yang ingin dilakukan scraping. Perlu diingatkan bahwa tidak semua web bisa discrap sehingga terkadang scraping dapat gagal atau sangat lama. Jika scraping sudah selesai, bola kuning akan berubah menjadi hijau. Setelah itu, input gambar yang ingin dicari dari hasil scrape. Gambar akan langsung keluar sesuai uruan kemiripin dengan syarat kemiripan diatas 60%

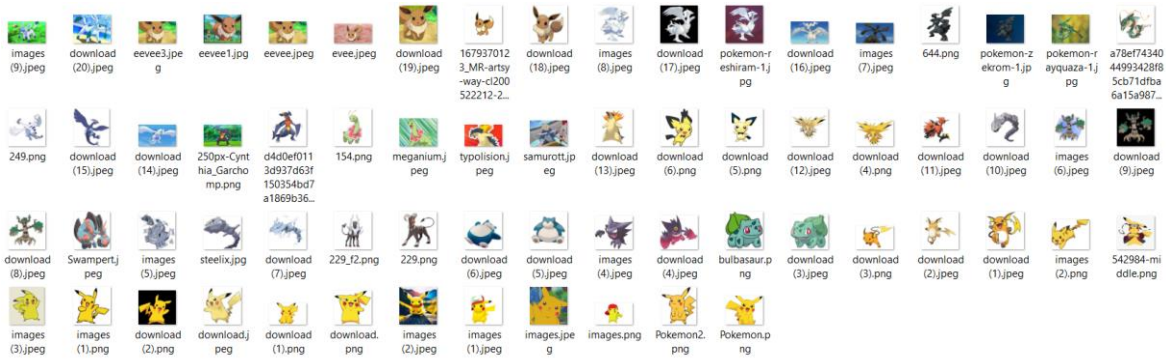
3. **Untuk Metode Kamera**

   Pastikan device terhubung dengan kamera . Input akan langsung berjalan dengan menerapkan metode *live*. Masukkan dataset yang ingin dibandingkan dan tekan upload. Dataset dimasukkan pada bagian ”input dataset” dan jika sudah yakin dapat menekan tombol ”upload dataset”. Tunggu beberapa saat karena sistem sedang memproses gambar yang ada.

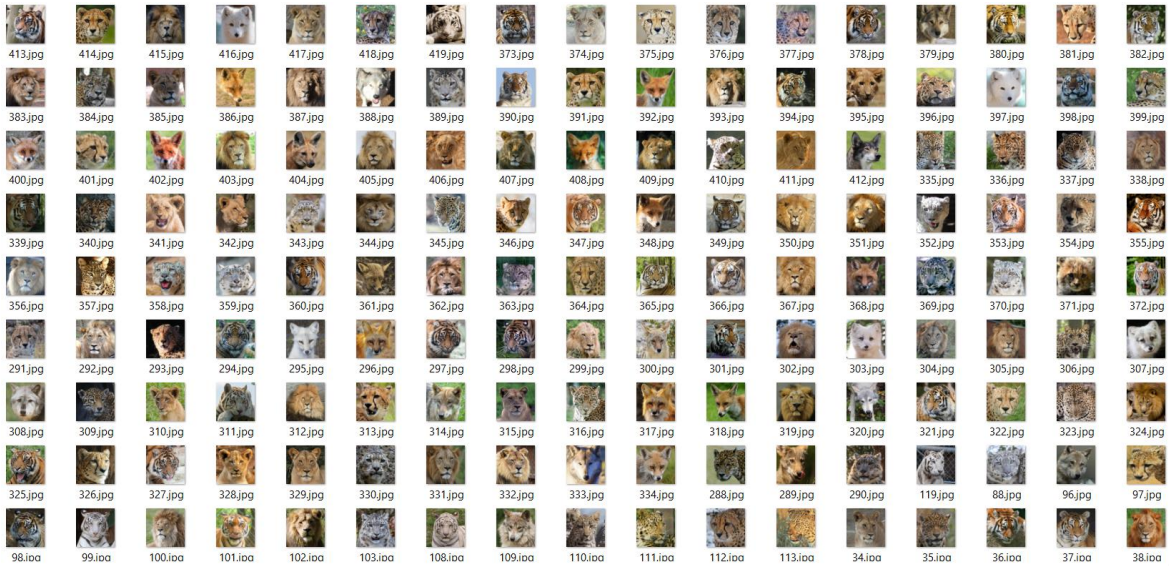
   Setelah bola kuning berubah menjadi hijau, barulah pelan pelan gambar yang mirip akan keluar. Jika tidak, maka mungkin tidak ada yang mirip.

HASIL PENGUJIAN

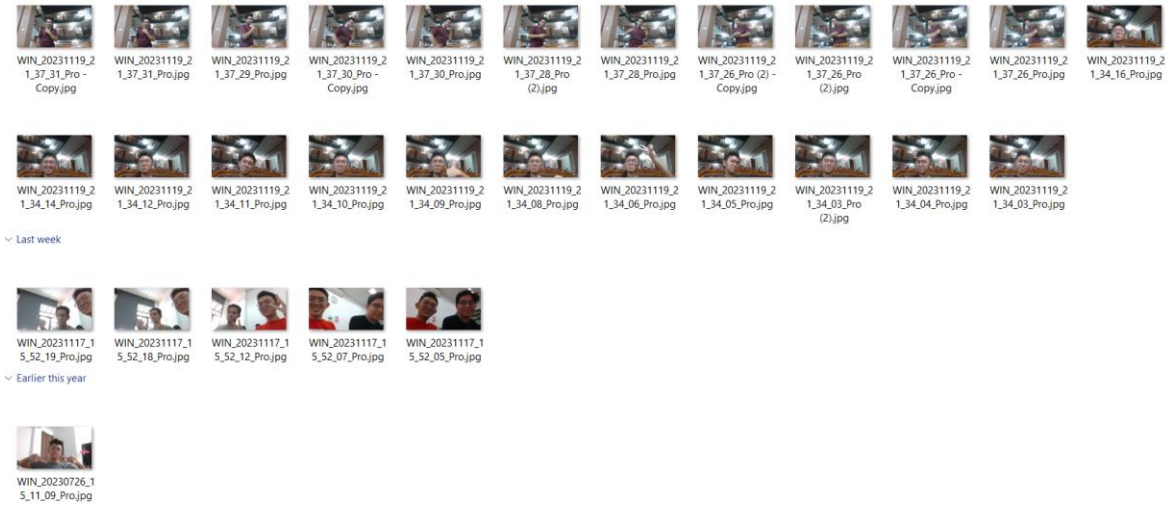
Dataset yang dipakai



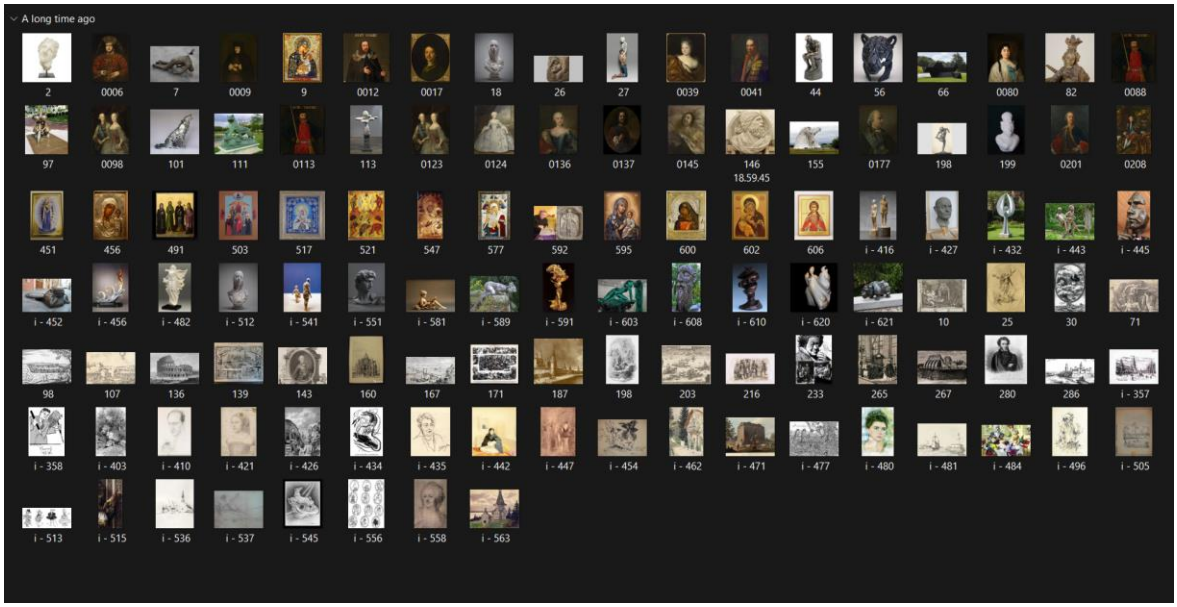
Gambar 24 Dataset 1 (Pokemon)



Gambar 25 Dataset 2 (Fauna)



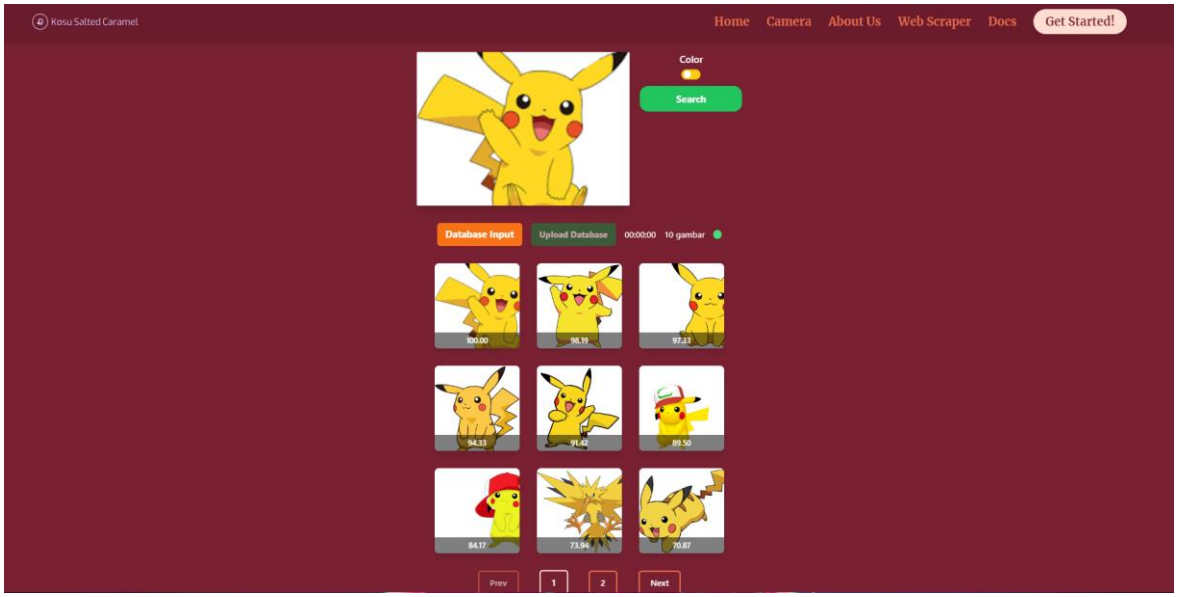
Gambar 26 Dataset 3 (Selfie)



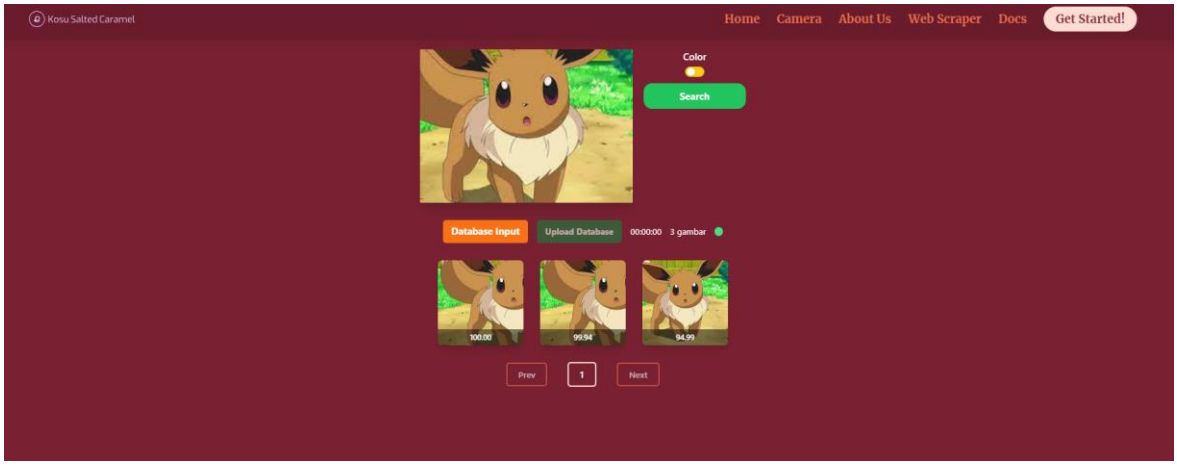
Gambar 27 Dataset 5 (Painting)

1. Hasil Metode User-Input

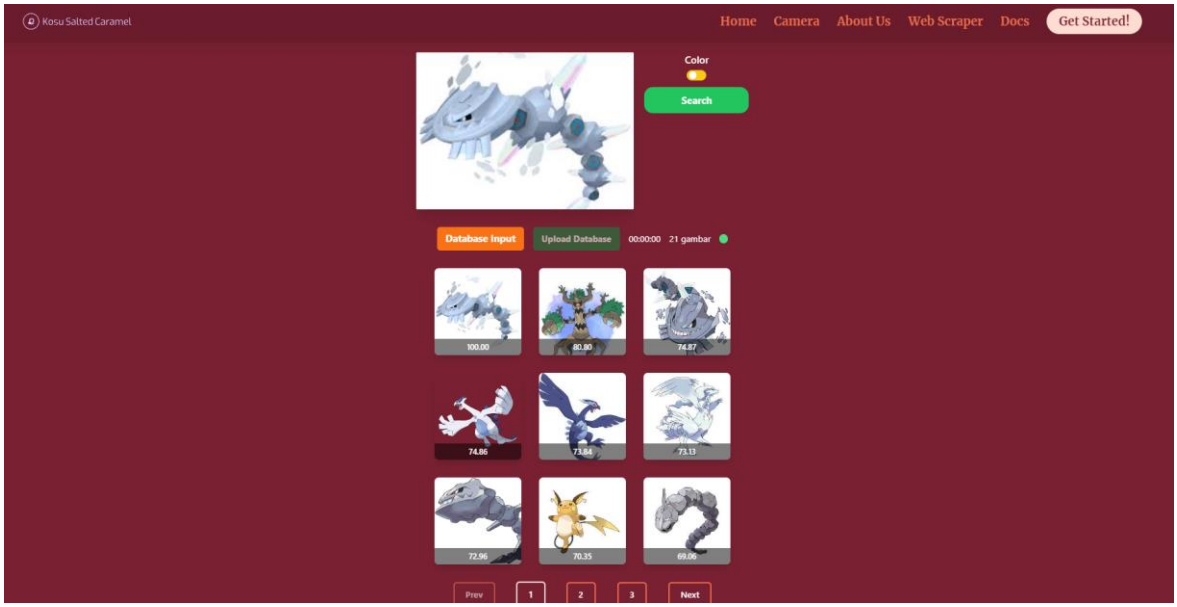
a. Parameter Warna



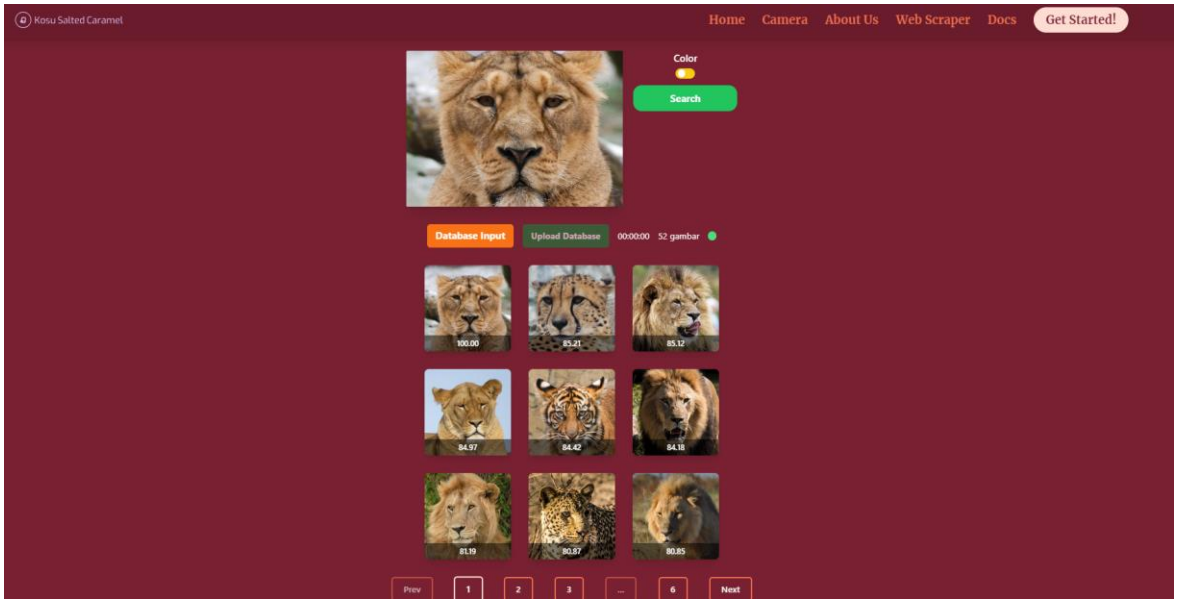
Gambar 28 User-Input Color CBIR Test 1 (Pikachu)



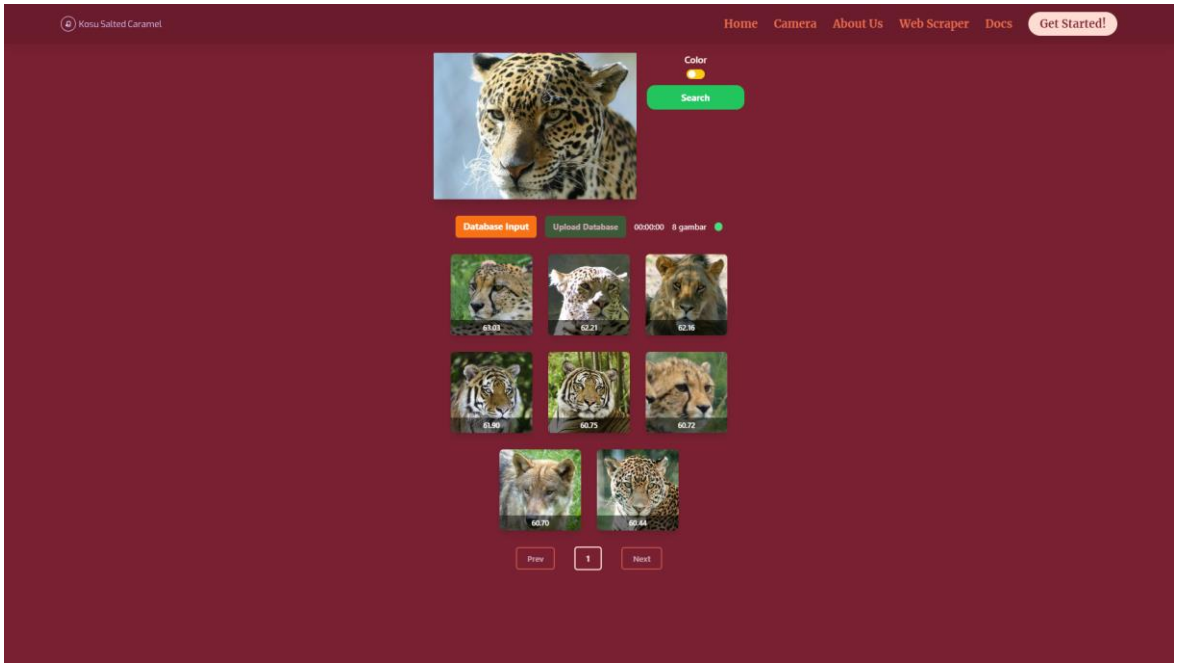
Gambar 29 User-Input Color CBIR Test 2 (Eevee)



Gambar 30 User-Input Color CBIR Test 3 (Steelix)

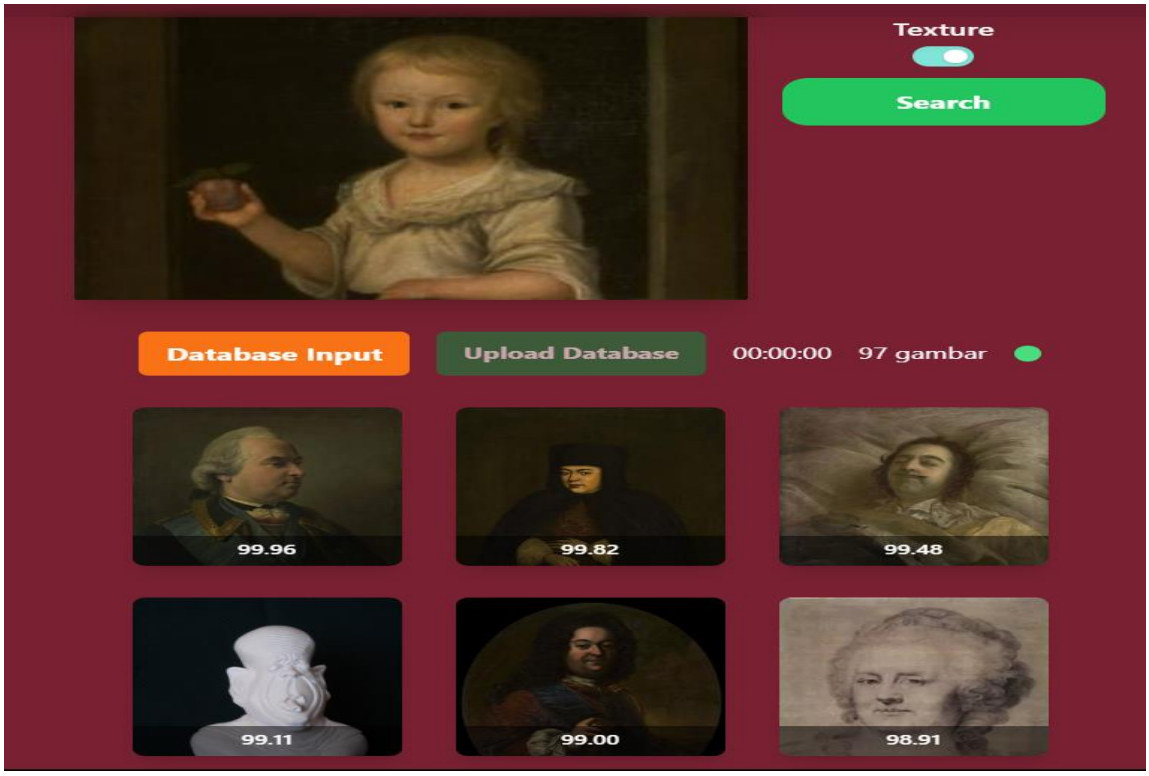


Gambar 31 User-Input Color CBIR Test 4 (Lion)

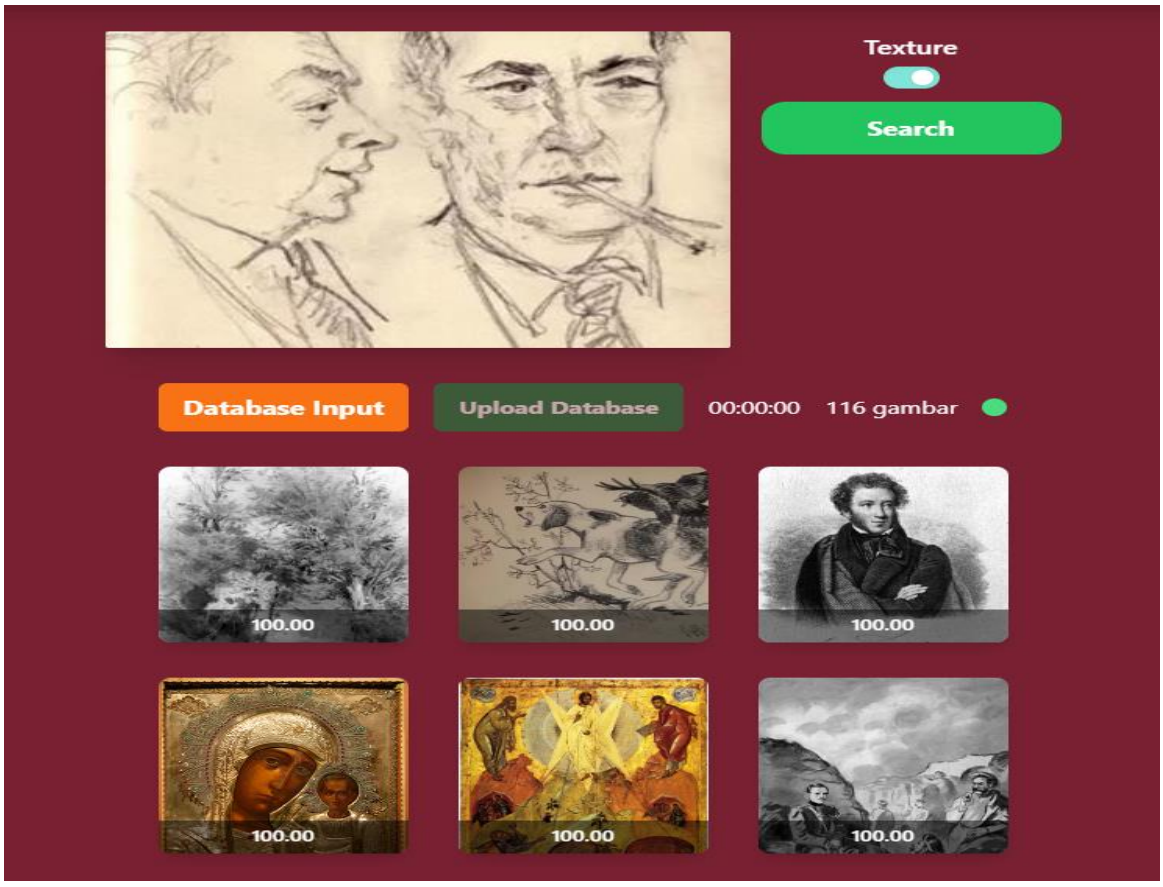


Gambar 32 User-Input Color CBIR Test 5 (Cheetah)

Parameter Tekstur



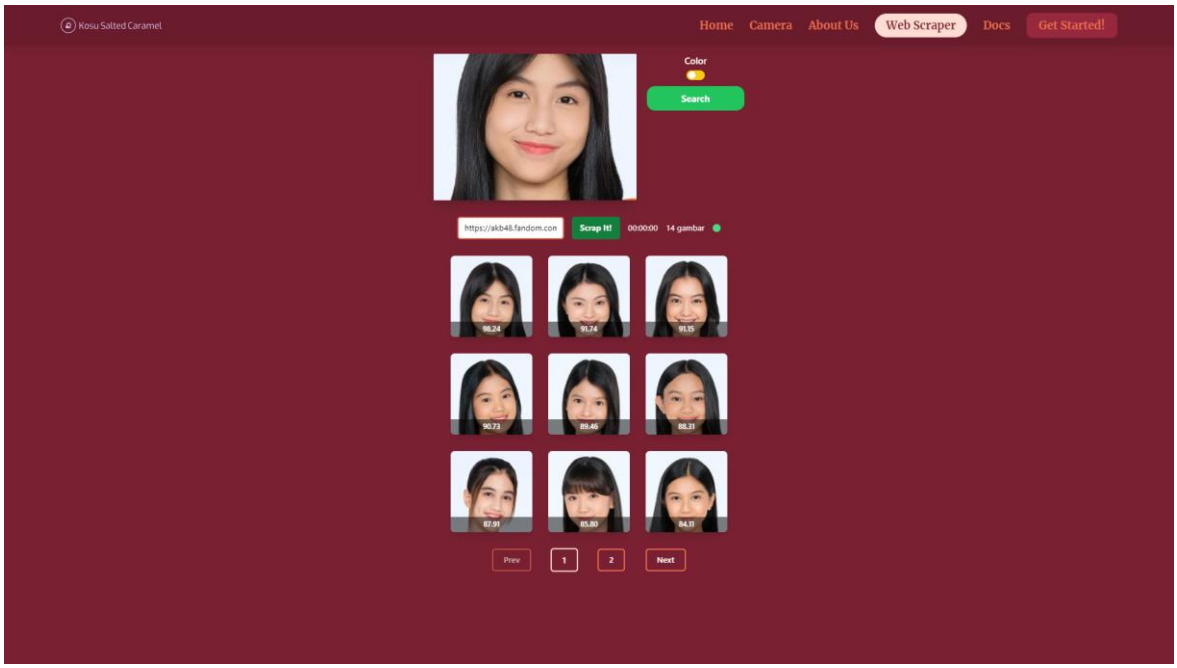
Gambar 33 User-input Texture CBIR Test 1(Painting 1)



Gambar 34 User-input Texture CBIR Test 2 (Painting 2)

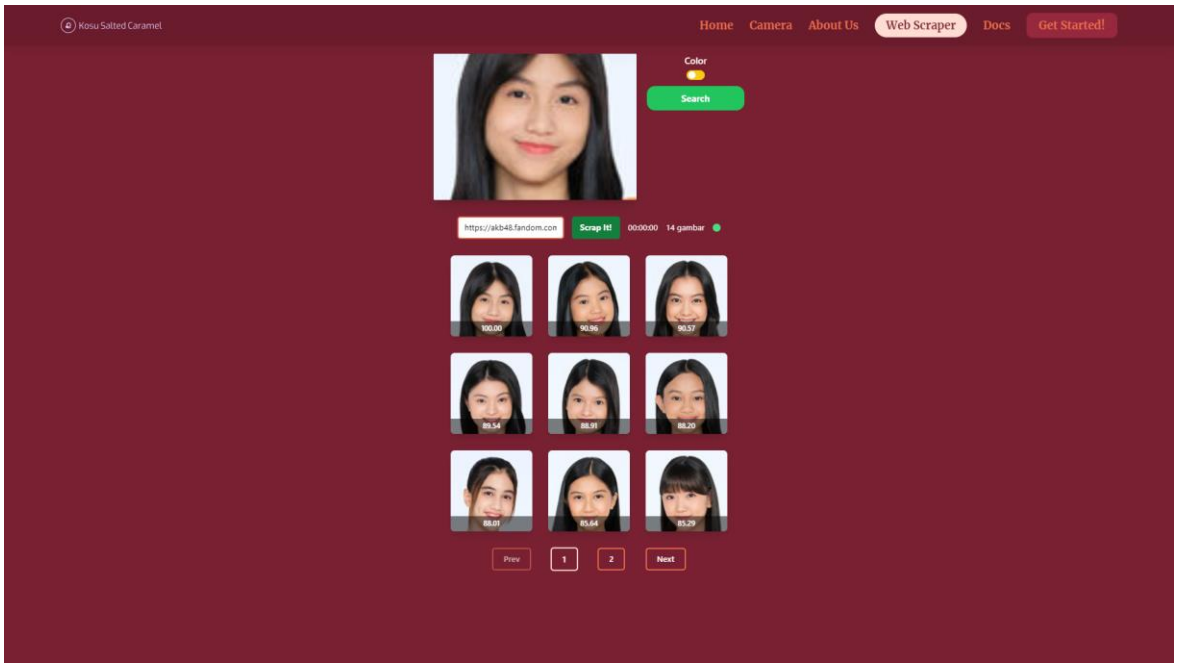
2. Hasil Metode Web-Scraping

a. Parameter Warna



Gambar 35 Web-Scraping Color CBIR Test (JKT48 Different Size)

NB : Untuk foto diatas memiliki perbedaan ukuran sehingga hasil mungkin terganggu

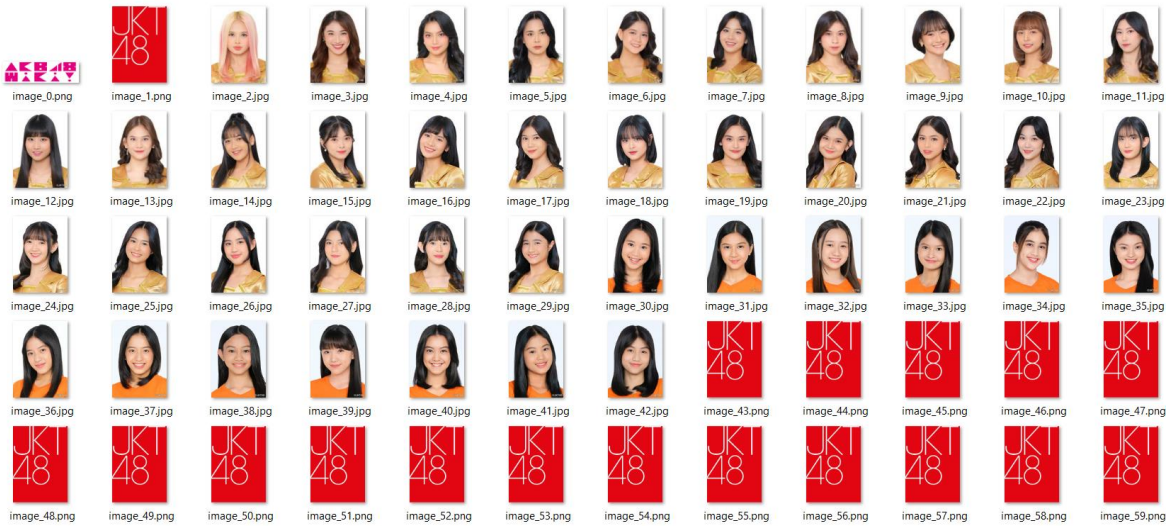


Gambar 36 Web-Scraping Color CBIR Test 2 (JKT48 original image)

NB : Ini untuk resolusi yang sama

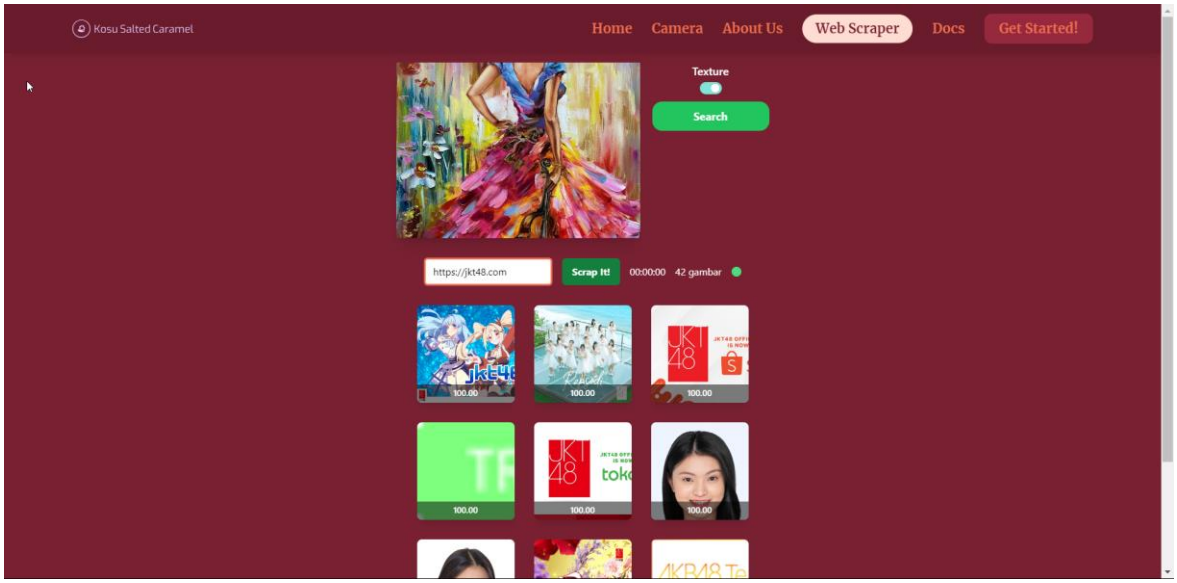
Link yang diuji: https://akb48.fandom.com/wiki/JKT48_Members

Hasil Scraping:



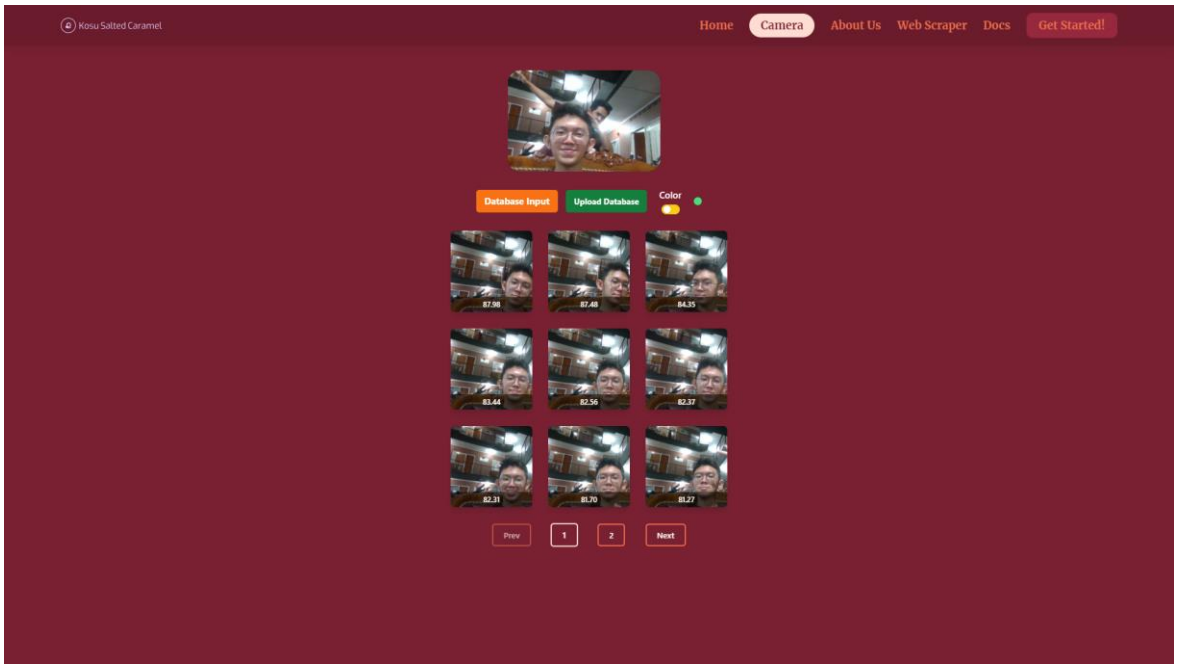
Gambar 37 Web-Scraping results from JKT48 website

b. Parameter Tekstur

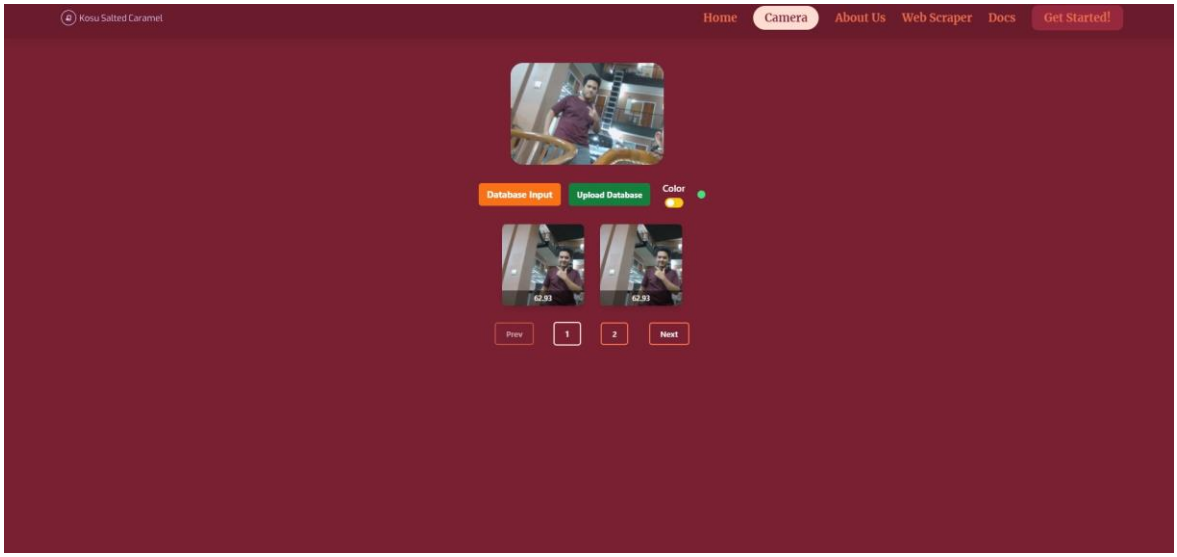


3. Hasil Metode Kamera

a. Parameter Warna

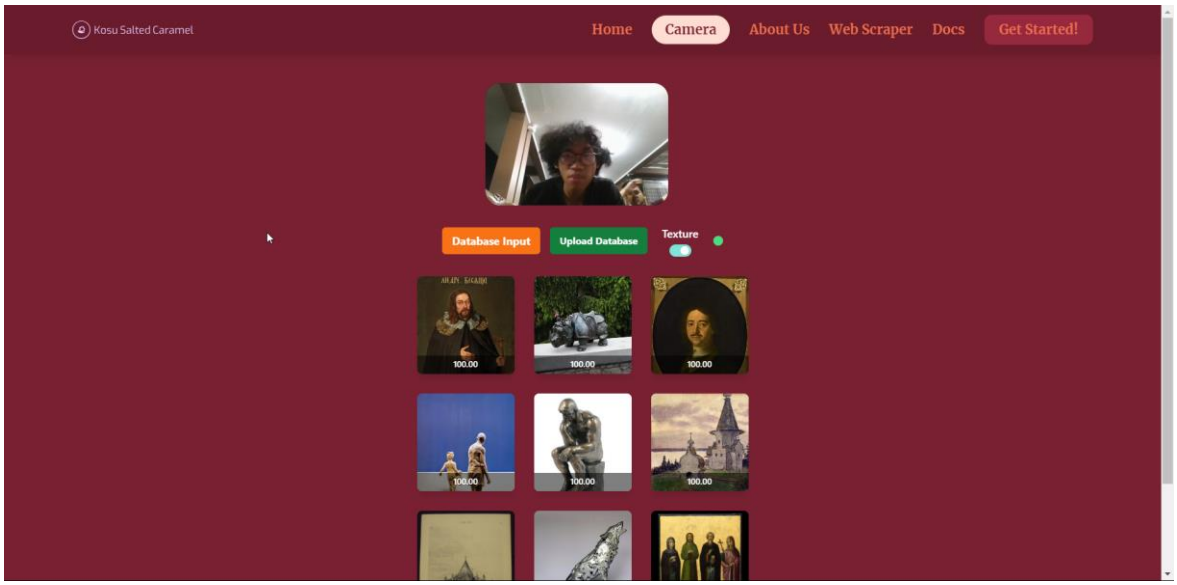


Gambar 38 Camera Color CBIR Test 1 (Wilson's Face VS Wilson and Rafly Selfies)



Gambar 39 Camera Color CBIR Test 2 (Rafly's face vs Wilson and Rafly selfie)

B. Parameter Tekstur



Gambar 40 Camera Texture CBIR Test 1 (Dito's face vs painting)

ANALISIS

Dari berbagai percobaan yang dilakukan, didapat beberapa analisis yang mungkin perlu untuk diutarakan. Salah satunya yaitu perbandingan terkait CBIR parameter warna dan tekstur. Kami mendapati bahwa keduanya unggul di situasi yang berbeda, parameter warna bekerja dengan baik dalam menghadapi gambar dengan warna yang jelas. Bahkan jika suatu gambar memiliki tampilan *blur*, selama warna terlihat, CBIR parameter warna dapat dengan jelas memberikan gambar dengan pola warna sesuai. Selain itu, untuk mengakses objek secara terperinci, kami mendapati bahwa CBIR dengan parameter warna sedikit lebih unggul. Hal ini disebabkan karena kami menggunakan metode blok 3x3 yang membagi gambar menjadi bobot bobot yang ada sehingga memberikan fokus banyak jika objek berada di tengah.

Dari sisi kecepatan dan pembangunan algoritma memiliki kesusahan yangs serupa sehingga memerlukan analisis menyeluruh. Salah satu tantangan yang ada dar CBIR parameter warna yakni diperlukannya fitur pembagian menjadi blok 3x3 sehingga perlu dilakukan iterasi sebanyak 9 kali dan hal tersebut dapat menghambat kecepatan.

Kami juga mendapati penggunaan konsep caching bermanfaat mengingat bahwa jika suatu dataset tidak diulang dalam suatu input, hasil caching juga dapat digunakan dalam *one time usage*.

Beberapa pertimbangan yang mungkin perlu diperhatikan salah satunya yaitu adanya latar. Background dengan warna monokrom (hanya 1 warna) umumnya akan memiliki kemiripan tinggi dengan gambar berbeda dengan background serupa.

Ukuran yang berbeda pada gambar yang sama tidak akan memberikan gambar 100% dimana semakin beda resolusi dan ukuran, semakin jauh pula dari angka 100%. Juga diketahui komparasi antara 2 gambar berbeda dengan ukuran yang berbeda juga akan memberikan hasil yang tidak akurat mengingat bahwa hasil pengambilan block juga akan dipengaruhi. Apalagi jika ratio panjang dan lebar berbanding terbalik (landscape dan portrait).

Parameter warna umunya memiliki keakurasian yang lebih tinggi dalam tugas besar ini baik dengan kamera, user input maupun web scraping. Hal ini disebabkan faktor blok yang menunjang akurasi.

Parameter tekstur cenderung memberikan hasil yang sangat dekat dengan 100%, hal tersebut disebabkan oleh nilai contrast yang mendominasi sebab nilainya yang besar dan mencapai nilai ratusan hingga ribuan, dan apabila dianalisis rumus cosine similarity yang digunakan untuk 3 fitur tekstur :

$$\cos\theta = \frac{C_1C_2 + H_1H_2 + E_1E_2}{\sqrt{C_1^2 + H_1^2 + E_1^2}\sqrt{C_2^2 + H_2^2 + E_2^2}}$$

Ketika C1 dan C2 jauh lebih besar daripada Homogeneity dan Entropynya didapat nilai cosine similaritynya:

$$\cos\theta = \frac{C_1C_2}{C_1\ C_2} = 1$$

CBIR Tekstur kurang cocok untuk dipakai dalam mengidentifikasi kumpulan gambar yang berasal dari media yang sama, contohnya dataset hewan yang diambil dengan sebuah kamera. Namun, CBIR Tekstur memiliki kelebihan dalam mengidentifikasi gambar yang berasal dari media yang berbeda, contohnya untuk mengidentifikasi sebuah karya lukisan dengan media cat air dan cat akrilik.

BAB V
KESIMPULAN DAN SARAN

KESIMPULAN

Metode Content Based Image Retrieval (CBIR) merupakan metode yang sangat cocok dalam pembuatan sebuah search engine, disebabkan karena metode pengumpulan data yang berbasis pada kemiripan gambar membuat hasi yang keluar juga akurat sesuai dengan input gambar pengguna. Selain itu, diperlukan juga analisis permintaan menyeluruh dari sisi pengguna untuk mengetahui parameter apa yang diinginkan. Parameter warna dan parameter tekstur merupakan 2 buah parameter yang berfokus pada hal yang berbeda, sehingga hasil pencarian dari kedua parameter tersebut juga berbeda. Dengan pemahaman yang tepat dan pemilihan parameter yang sesuai, hasil dari CBIR pasti akan sesuai dengan keinginan pengguna.

SARAN

Dalam pengerjaan sebuah proyek ataupun dalam konteks ini berupa tugas besar, akan lebih baik jika dilakukan koordinasi. Ditengah kesibukan antar mahasiswa, justru semakin memerlukan adanya sedikit komunikasi untuk memastikan kinerja para mahasiswa seperti sebuah grafik lurus yang stabil. Kurangnya komunikasi dapat membuat seseorang mungkin lalai akan tugas dan mengakibatkan peforma tugas menurun. Selain itu, diperhatikan dalam pengerjaan program utama daripada fitur tambahan gua memastikan pengumpulan dapat dilakukan secara tepat waktu

KOMENTAR ATAU TANGGAPAN

Tugas besar ini merupakan salah satu tugas besar yang paling berkesan selama perkuliahan semester 3 di jurusan Teknik Informatika. Kami sangat antusias dalam mengaplikasikan pengembangan web dengan suatu program pencarian gambar. Tidak pernah kami sangka bahwa kami akan se-tertarik ini dalam mempelajari teknik pengembangan web, mengingat bahwa kami memiliki dasar yang kurang kuat dalam pengembangan web, namun sekarang kami sudah mulai memiliki pemahaman dan memiliki pengalaman dalam mengeksplorasi teknik pengembangan website.

Komentar para developers:

1. Raden Rafly Hanggaraksa Budiarto (13522014): Senang banget bisa belajar web dari perkuliahan
2. Wilson Yusda (13522019) : Seru Tubesnya, makin tertarik belajar website

3. Abdul Rafi Radityo Hutomo (13522089) : Image Processing dalam Bahasa C seru juga ternyata

REFLEKSI

Kami mempelajari banyak hal dalam tugas besar kali ini. Dari sisi teknis seperti pembelajaran teori *Content Based Image Retrieval (CBIR)*, perencanaan program yang cepat untuk meningkatkan efektivitas program, pembuatan *front-end* dan *back-end* sebuah website dan mengintegrasi hubungan keduanya, serta memastikan pengalaman pengguna sesuai dan nyaman bagi semua user. Dari sisi non-teknis, kami mengasah kemampuan pemikiran dan perencanaan kami, mengingat deadline yang cepat, melatih kerja sama tim, manajemen waktu, serta mempererat hubungan pertemanan antar mahasiswa.

Kami menyadari terdapat kala ketika usaha yang kami lakukan tidak sesuai hasil, seperti kendala manajemen waktu buruk, miskomunikasi, sampai adanya deadline pengerjaan yang terlewat. Namun kami yakin hal ini melatih kami untuk terus bergerak ke arah yang lebih baik dan projek ini merupakan sebuah *stepping stone* bagi kami dalam menghadapi dunia kerja nantinya.

RUANG PERBAIKAN

Projek yang kami lakukan sudah memenuhi ekspetasi dari spesifikasi yang ada. Daripada ruang perbaikan, kami mendapati beberapa ruang peningkatan yang dapat dilakukan, seperti permasalahan tampilan , pengembangan tampilan sela pergantian fitur ataupun dalam proses perjalanan sebuah fitur. Dengan tampilan yang bagus, maka kami menjadi 1 langkah lebih depan dalam pembelajaran informatika dan akan meningkatkan performa dalam mengerjakan hal serupa dalam projek berikutnya.

DAFTAR PUSTAKA

https://www.sciencedirect.com/science/article/pii/S0895717710005352

Huang, W., Dai, J., & Wu, Q. (Year not provided). Image Retrieval Based on Weighted Block Color Histogram and Texton Co-occurrence Matrix.

LINK VIDEO DEMO

https://youtu.be/L54hTDGbORs?feature=shared