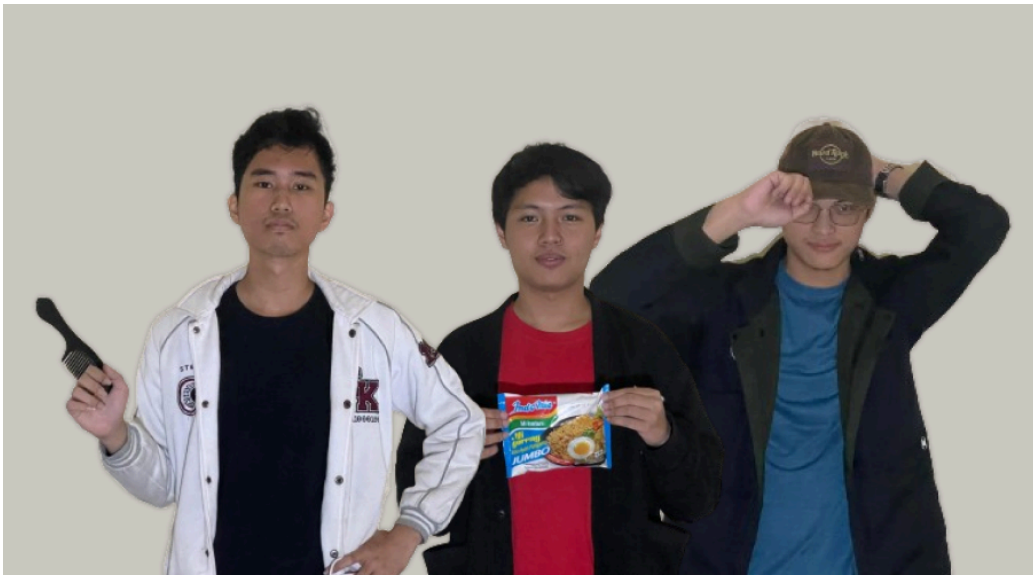
IF2211 Strategi Algoritma

Pemanfaatan Pattern Matching dalam Membangun Sistem Deteksi Individu Berbasis Biometrik Melalui Citra Sidik Jari

Laporan Tugas Besar 3

Disusun untuk memenuhi tugas mata kuliah Strategi Algoritma
pada Semester 2 (dua) Tahun Akademik 2023/2024



Oleh

Muhammad Fauzan Azhim	**13522153**
Muhammad Davis Adhipramana	**13522157**
Valentino Chryslie Triadi	**13522164**

Kelompok BesokMinggu

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

2024

Daftar Isi

Daftar Isi 2
BAB 1 3
DESKRIPSI TUGAS 3
Gambar 1.1. Ilustrasi fingerprint recognition pada deteksi berbasis biometrik. 3
Gambar 1.2. Skema implementasi konversi citra sidik jari. 4
Tabel 1. Hasil konversi citra sidik jari menjadi binary dan konversi ke ASCII 8-bits. 5
Gambar 1.3. Skema basis data yang digunakan 6
Tabel 1.2. Kombinasi ketiga variansi bahasa alay Indonesia. 7
Gambar 1.4. Referensi tampilan UI dekstop-app 8
BAB 2 10
LANDASAN TEORI 10
Gambar 2.1 Ilustrasi algoritma KMP 10
Gambar 2.2 Ilustrasi KMP Border Function 11
Tabel 2.1 Tabel Notasi Umum Regular Expression 15
Gambar 2.8 Tampilan Awal Program 17
BAB 3 19
ANALISIS PEMECAHAN MASALAH 19
3.1 Tahapan Pemecahan Masalah 19
3.2 Proses Penyelesaian Solusi dengan Algoritma KMP dan BM 20
3.3 Proses Pencarian Biodata dengan REGEX 21
3.4 Fitur Fungsionalitas dan Arsitektur Aplikasi 22
3.5 Ilustrasi Kasus 23
BAB 4 25
IMPLEMENTASI DAN PENGUJIAN 25
4.1 Spesifikasi Teknis Program 25
4.2 Tata Cara Penggunaan 31
4.3 Hasil Pengujian 32
4.4 Analisis Hasil Pengujian 41
BAB 5 42
KESIMPULAN 42
5.1 Kesimpulan 42
5.2 Saran 42
5.3 Tanggapan 42
5.4 Refleksi 42
LAMPIRAN 43
DAFTAR PUSTAKA 44

BAB 1

DESKRIPSI TUGAS

Gambar 1.1. Ilustrasi fingerprint recognition pada deteksi berbasis biometrik.

Di era digital ini, keamanan data dan akses menjadi semakin penting. Perkembangan teknologi membuka peluang untuk berbagai metode identifikasi yang canggih dan praktis. Beberapa metode umum yang sering digunakan seperti kata sandi atau pin, namun memiliki kelemahan seperti mudah terlupakan atau dicuri. Oleh karena itu, biometrik menjadi alternatif metode akses keamanan yang semakin populer. Salah satu teknologi biometrik yang banyak digunakan adalah identifikasi sidik jari. Sidik jari setiap orang memiliki pola yang unik dan tidak dapat ditiru, sehingga cocok untuk digunakan sebagai identitas individu.

Pattern matching merupakan teknik penting dalam sistem identifikasi sidik jari. Teknik ini digunakan untuk mencocokkan pola sidik jari yang ditangkap dengan pola sidik jari yang terdaftar di database. Algoritma pattern matching yang umum digunakan adalah Bozorth dan Boyer-Moore. Algoritma ini memungkinkan sistem untuk mengenali sidik jari dengan cepat dan akurat, bahkan jika sidik jari yang ditangkap tidak sempurna.

Dengan menggabungkan teknologi identifikasi sidik jari dan pattern matching, dimungkinkan untuk membangun sistem identifikasi biometrik yang aman, handal, dan mudah digunakan. Sistem ini dapat diaplikasikan di berbagai bidang, seperti kontrol akses, absensi karyawan, dan verifikasi identitas dalam transaksi keuangan.

Di dalam Tugas Besar 3 ini, Anda diminta untuk mengimplementasikan sistem yang dapat melakukan identifikasi individu berbasis biometrik dengan menggunakan sidik jari. Metode yang akan digunakan untuk melakukan deteksi sidik jari adalah Boyer-Moore dan Knuth-Morris-Pratt. Selain itu, sistem ini akan dihubungkan dengan identitas sebuah individu melalui basis data sehingga harapannya terbentuk sebuah sistem yang dapat mengenali identitas seseorang secara lengkap hanya dengan menggunakan sidik jari.

Pada tugas ini, Anda diminta untuk mengimplementasikan sebuah program yang dapat melakukan identifikasi biometrik berbasis sidik jari. Proses implementasi dilakukan dengan menggunakan algoritma Boyer-Moore dan Knuth-Morris-Pratt, sesuai dengan yang diajarkan pada materi dan salindia kuliah.

Secara sekilas, penggunaan algoritma pattern matching dalam mencocokkan sidik jari terdiri atas tiga tahapan utama dengan skema sebagai berikut.

Gambar 1.2. Skema implementasi konversi citra sidik jari.

Gambar yang digunakan pada proses pattern matching kedua algoritma tersebut adalah gambar sidik jari penuh berukuran m x n pixel yang diambil sebesar 30 pixel setiap kali proses pencocokan data. Untuk tugas ini, Anda dibebaskan untuk mengambil jumlah pixel asalkan didasarkan pada alasan yang masuk akal (dijelaskan pada laporan) dan penanganan kasus ujung yang baik (misal jika ternyata ukuran citra sidik jari tidak habis dibagi dengan ukuran pixel yang dipilih). Selanjutnya, data pixel tersebut akan dikonversi menjadi binary. Seperti yang mungkin Anda ketahui sesuai materi kuliah (ya kalau masuk kelas), karena binary hanya memiliki variasi karakter satu atau nol, maka proses pattern matching akan membuat proses pencocokan karakter menjadi lambat karena harus sering mengulangi proses pencocokan pattern. Cara yang dapat dilakukan untuk mengatasi hal tersebut adalah dengan mengelompokkan setiap baris kode biner per 8 bit sehingga membentuk karakter ASCII. Karakter ASCII 8-bit ini yang akan mewakili proses pencocokan dengan string data.

Untuk memberikan gambaran yang lebih jelas, berikut adalah contoh kasus citra sidik jari dan proses serta sampel hasil yang didapatkan. Dataset yang digunakan adalah dataset citra sidik jari yang terdapat pada bagian referensi.

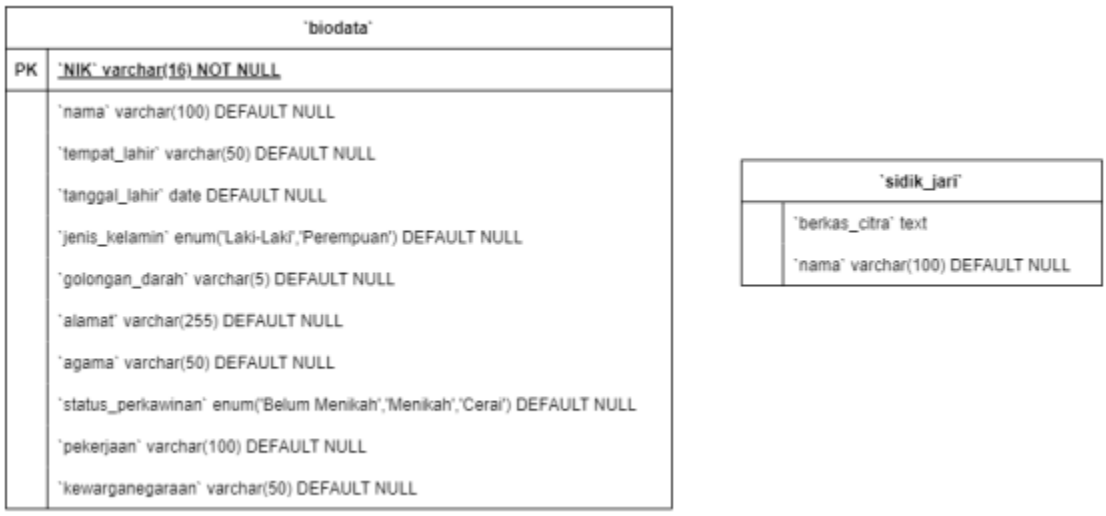
Tabel 1. Hasil konversi citra sidik jari menjadi binary dan konversi ke ASCII 8-bits.

Citra Sidik Jari	Potongan Nilai Binary	Potongan ASCII 8-bits
	1010000010100000101000 0101000001010000010100 0010100000101000001010 0001010000010100000101 0000101000001010000010 0000010100000101000001 1000001010000010100000 0100000101000001010000 1010000010100000101000 0101000001010	iÿÿÿÿÿÿÿÿÿÿÌÿÿû:Ãÿÿ<Ìÿÿu w ûÿkÿý,Pþÿ¿/ÿÿ’ »ÿÿÐ¾?ÿ&¼¡¤Úª†È/Zíÿÿq ¼ÿúTý~Rûût`íÿÿÿÿÿÿ iÿÿÿÿÿÿÿÿÿÿöÿÿÿ"býÿýÁüþÿÊ ’ þ /ÿœ2ÿúYÿÿ\ ^ÿÿp fÿÿà !›ÿÿÿþÿÿÿÿÿÿýš <ðÿÕ ²ÿÿQûÿ Nºÿášÿÿÿÿÿÿ iÿÿÿÿÿÿÿÿÿÿÿõrÛõÿÿÇ½ÿÿ‡å ÿ ‹ >ÿÙþþ6Gÿÿ;Úÿÿÿfäÿý

Pada tahap implementasi di titik ini, telah dihasilkan serangkaian karakter ASCII 8-bit yang merepresentasikan sebuah sidik jari. Hasil ini yang akan dijadikan dasar pencarian sidik jari yang sama dengan daftar sidik jari yang terdapat pada basis data. Pencarian sidik jari yang paling mirip dengan sidik jari yang menjadi masukan pengguna dilakukan dengan algoritma pencocokan string Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM). Jika tidak ada satupun sidik jari pada basis data yang exact match dengan sidik jari yang menjadi masukan melalui algoritma KMP ataupun BM, maka gunakan sidik jari paling mirip dengan kesamaan diatas nilai tertentu. Anda diberikan kebebasan untuk menentukan nilai batas persentase kemiripan ini, silakan melakukan pengujian untuk menentukan nilai tuning yang tepat dan jelaskan pada laporan. Metode perhitungan tingkat kemiripan juga dibebaskan kepada Anda asalkan dijelaskan di laporan. Akan tetapi, asisten sangat menyarankan untuk menggunakan salah satu dari algoritma Hamming Distance, Levenshtein Distance, ataupun Longest Common Subsequence (LCS).

Fungsi dari deteksi biometrik adalah mengidentifikasi seseorang, oleh sebab itu, pada proses implementasi program ini, sebuah citra sidik jari akan dicocokkan dengan biodata seseorang. Biodata yang dicocokkan terdiri atas data-data yang terdapat pada KTP, antara lain : NIK, nama, tempat/tanggal lahir, jenis kelamin, golongan darah, alamat, agama, status

perkawinan, pekerjaan, dan kewarganegaraan. Relasi ini dibuat dalam sebuah basis data dengan skema detail seperti yang tertera pada bagian di bawah ini. Sebagai tambahan, struktur relasi basis data telah disediakan, silakan gunakan dump sql berikut.



Gambar 1.3. Skema basis data yang digunakan

Seorang pribadi dapat memiliki lebih dari satu berkas citra sidik jari (relasi one-to-many). Akan tetapi, seperti yang dapat dilihat pada skema relasional di atas, keduanya tidak terhubung dengan sebuah relasi. Hal ini disebabkan karena pada kasus dunia nyata, data yang disimpan bisa saja mengalami korup. Dengan membuat atribut kolom yang mungkin korup adalah atribut nama pada tabel biodata, maka atribut nama pada tabel sidik_jari tidak dapat memiliki foreign-key yang mereferensi ke tabel biodata (silakan review kembali materi IF2240 bagian basis data relasional). Pada tugas besar kali ini, kita akan coba melakukan simulasi implementasi data korup yang hanya mungkin terjadi pada atribut nama di tabel biodata (asumsikan kolom lain pada setiap tabel tidak mengalami korup). Akan tetapi, karena tujuan utama program adalah mengenali identitas seseorang secara lengkap hanya dengan menggunakan sidik jari, maka harus dilakukan sebuah skema untuk menangani data korup tersebut.

Sebuah data yang korup dapat memiliki berbagai macam bentuk. Pada tugas ini, jenis data korup adalah bahasa alay Indonesia. Terdengar lucu, tetapi para pendahulu kita sudah membuat bahasa ini untuk berkomunikasi kepada sesamanya. Dengan mengutip dari berbagai sumber, Anda akan diminta untuk menangani kombinasi dari tiga buah variasi bahasa alay, yaitu

kombinasi huruf besar-kecil, penggunaan angka, dan penyingkatan. Contoh kasus bahasa alay dijelaskan dengan detail sebagai berikut.

Tabel 1.2. Kombinasi ketiga variansi bahasa alay Indonesia.

Variasi	Hasil
Kata orisinil	Bintang Dwi Marthen
Kombinasi huruf besar-kecil	bintanG DwI mArthen
Penggunaan angka	B1nt4n6 Dw1 M4rthen
Penyingkatan	Bntng Dw Mrthen
Kombinasi ketiganya	b1ntN6 Dw mrthn

Cara yang dapat digunakan untuk menangani ini adalah dengan menggunakan Regular Expression (Regex). Lakukan konversi pola karakter alay hingga dapat dikembalikan ke bentuk alfabetik yang bersesuaian. Setelah menggunakan Regex, Anda akan diminta kembali untuk melakukan pattern matching antara nama yang bersesuaian dengan algoritma KMP dan BM dengan ketentuan yang sama seperti saat pencocokan sidik jari. Sebagai referensi bahasa alay, Anda dapat menggunakan website alay generator yang terdapat pada bagian referensi.

Setelah menemukan nama yang bersesuaian, Anda dapat menggunakan basis data untuk mengembalikan detail biodata lengkap individu. Jika seluruh prosedur diatas diimplementasikan dengan baik, maka sebuah sistem deteksi individu berbasis biometrik dengan sidik jari telah berhasil untuk diimplementasikan.

Pada Tugas Besar kali ini, sistem yang dibangun akan diimplementasikan dengan basis desktop-app menggunakan bahasa C#. Masukan yang akan diberikan oleh pengguna saat menggunakan aplikasi adalah sebuah citra sidik jari. Selain itu program perlu untuk memiliki basis data SQL dengan struktur relasional seperti yang telah dijelaskan sebelumnya. Tampilan layout dari aplikasi yang akan dibangun adalah sebagai berikut.

Gambar 1.4. Referensi tampilan UI *dekstop-app*

Anda dapat menambahkan menu lainnya seperti gambar, logo, dan sebagainya. Tampilan front-end dari desktop-app tidak harus sama persis dengan layout yang diberikan di atas, tetapi dibuat semenarik mungkin dan wajib mencakup komponen-komponen berikut:

- Judul aplikasi
- Tombol Insert citra sidik jari, beserta display citra sidik jari yang ingin dicari
- Toggle Button untuk memilih algoritma yang ingin digunakan (KMP atau BM)
- Tombol Search untuk memulai pencarian
- Display sidik jari yang paling mirip dari basis data
- Informasi mengenai waktu eksekusi
- Informasi mengenai tingkat kemiripan sidik jari dengan gambar yang ingin dicari, dalam persentase (%)
- List biodata hasil pencarian dari basis data. Keluarkan semua nilai atribut dari individu yang dirasa paling mirip. Perlu diperhatikan pendefinisian batas kemiripan dapat memunculkan kemungkinan tidak ditemukan list biodata yang memiliki sidik jari paling mirip.

Secara umum, berikut adalah cara umum penggunaan program:

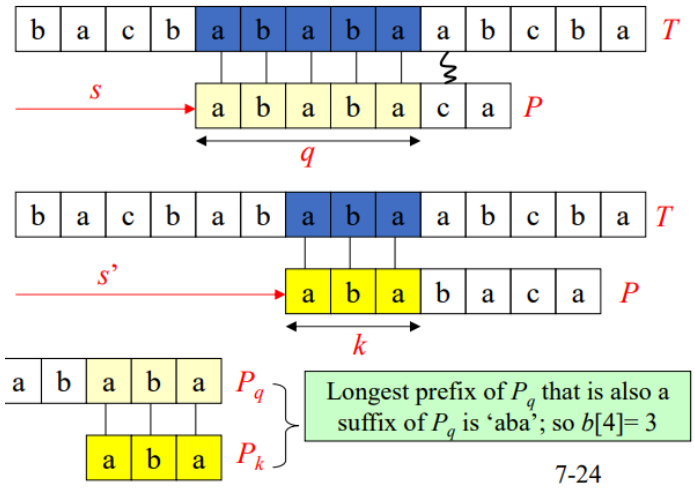
- Pengguna terlebih dahulu memasukkan citra sidik jari yang ingin dicari biodatanya.
- Setelah citra dimasukkan, pilih opsi pencarian, ingin melakukan pencarian apakah akan menggunakan algoritma KMP atau BM.
- Tekan tombol search, program kemudian akan memproses, mencari citra sidik jari dari basis data yang memiliki kemiripan dengan citra sidik jari yang menjadi masukan.
- Program akan menampilkan list biodata jika ditemukan citra sidik jari yang memiliki kemiripan dengan batas persentase tertentu. Program juga dapat mengeluarkan informasi tidak ada sidik jari yang mirip jika semua citra dalam basis data tidak memiliki kemiripan dengan masukan.
- Terdapat informasi terkait waktu eksekusi program dan persentase kemiripan citra.

BAB 2

LANDASAN TEORI

Pattern Matching adalah salah satu hal yang sangat krusial dalam perkembangan teknologi. Penerapan dari pattern matching ini mampu menciptakan teknologi-teknologi baru seperti, face recognition, fingerprint recognition, dan masih banyak lagi. Dalam melakukan pattern matching, ada banyak sekali algoritma yang dapat digunakan, salah satunya adalah dengan menggunakan metode string matching Knuth-Morris-Pratt (KMP), Boyer-Moore (BM), dan Regular Expression (REGEX).

Algoritma Knuth-Morris-Pratt (KMP) merupakan salah satu algoritma pencocokan string yang cukup efisien. Algoritma ini akan melakukan pencocokan string dari kiri ke kanan seperti pada umumnya, namun yang membedakan adalah algoritma ini mampu melakukan shifting atau pergeseran pencocokan yang lebih efisien dibanding algoritma brute force. Pergeseran atau shifting yang dilakukan oleh algoritma ini bergantung pada kesamaan suffix dan prefix dari sebuah string yang kita cocokan. String yang dicocokan akan digeser agar prefix dari sebuah string bersesuaian dengan suffix dari string tersebut. Setelah dilakukan pergeseran, algoritma ini akan melanjutkan pencocokan huruf atau karakter setelah prefix atau suffix yang telah digeser tersebut. Sebagai contoh dapat dilihat pada ilustrasi berikut



Gambar 2.1 Ilustrasi algoritma KMP

Ilustrasi diatas menggambarkan jika karakter di string p tidak sesuai dengan karakter di string T pada posisi yang sama, maka algoritma ini akan melakukan shifting atau pergeseran dengan memperhatikan prefix dan suffix terbesar dari string p yang sudah bersesuaian dengan string T. Setelah pergeseran untuk menyesuaikan suffix dengan prefix dijalankan, maka algoritma akan memulai pengecekan langsung dari karakter setelah prefix tersebut.

Untuk melakukan pergeseran atau shifting agar prefix dan suffix terbesar bersesuaian, dapat menggunakan KMP Border Function. KMP Border Function adalah salah satu cara untuk mencari prefix dan suffix terbesar yang bersesuaian. Untuk lebih jelasnya dapat dilihat ilustrasi berikut:

(k = j-1)

j	0	1	2	3	4	5	6	7	8	9
P [j]	**a**	**b**	**a**	**b**	**a**	**b**	**a**	**b**	**c**	**a**
k	0	1	2	3	4	5	6	7	8	
b[k]	0	0	1	2	3	4	5	6	0	

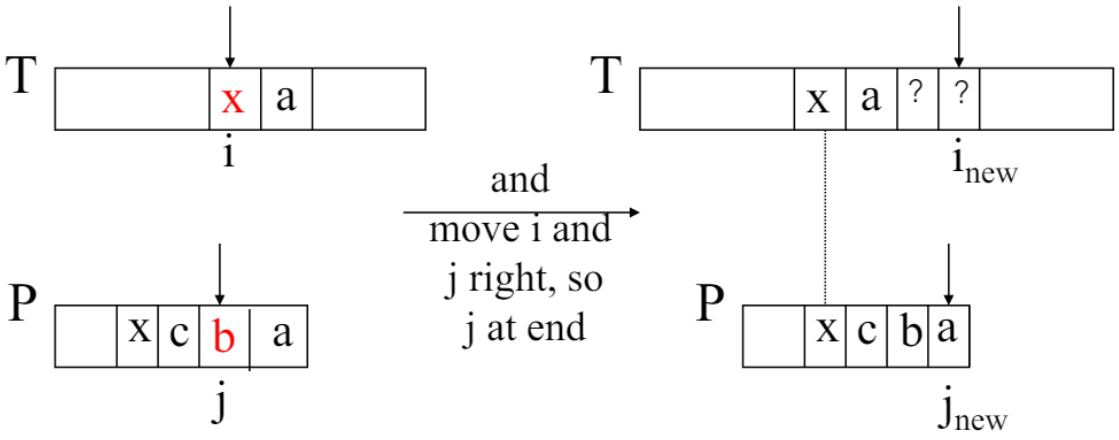
Gambar 2.2 Ilustrasi KMP Border Function

Pada ilustrasi di atas, langkah pertama yang harus dilakukan adalah menentukan nilai j, P[j], dan k dengan j adalah index dari string, P[j] adalah list of karakter, dan k adalah j-1. Kemudian tentukan b[k] yang berarti panjang prefix dan suffix terbesar untuk string yang dimulai dari index 0 hingga index k dengan ketentuan nilai maksimal b[k] adalah k-1.

Algoritma Boyer-Moore (BM) adalah salah satu algoritma pencocokan string yang cukup efisien. Algoritma ini menggunakan dua teknik utama yaitu *looking-glass technique* dan *character-jump technique*. *Looking-glass technique* adalah sebuah teknik yang membaca pattern yang akan dicari dari akhir ke awal. Selanjutnya, *character-jump technique*, sebuah teknik yang

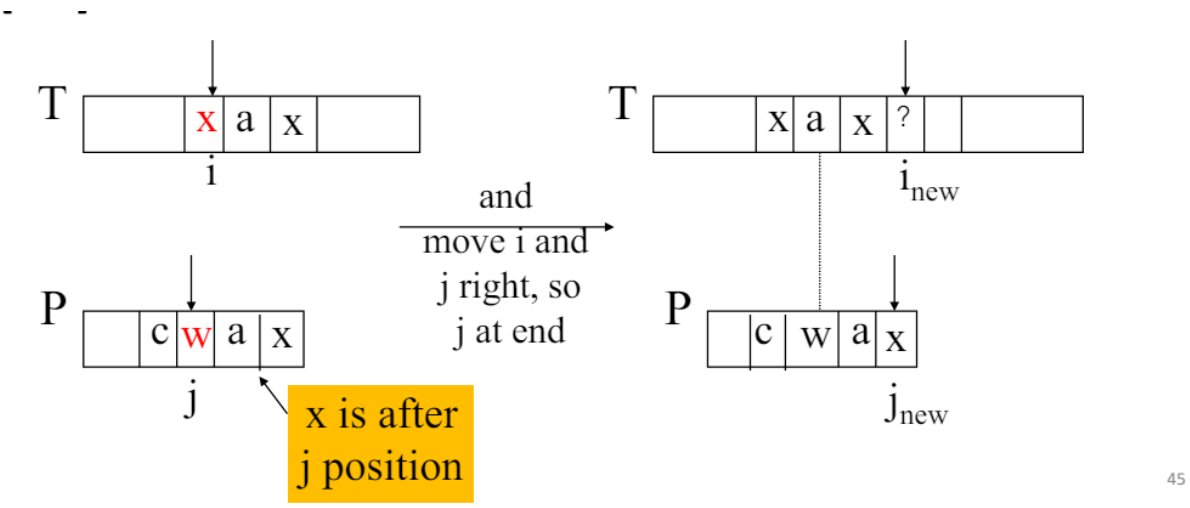
dijalankan ketika ditemukan *mismatch* pada saat pencocokan. Terdapat 3 kemungkinan dalam *character-jump technique,* diantaranya : (ket : i adalah index saat ini pada Teks T)

Case 1 (Terjadi ketika Pattern P mengandung x (*mismatch)* dimana character x ditemukan pada index yang berada < j (lokasi index character yang *mismatch* pada Pattern P) atau shift character menuju last occurrence x pada Pattern P memungkinkan) Pada kasus ini, i akan di shift menuju character x yang berada pada pattern P.



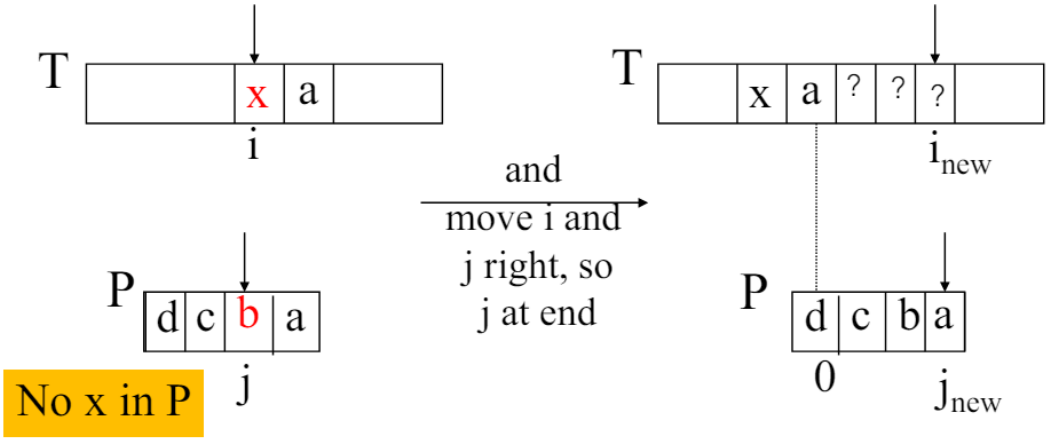
Gambar 2.3 Ilustrasi Case 1 Boyer-Moore Character Jump

Case 2 (Terjadi ketika Pattern P mengandung x (*mismatch)* dimana character x ditemukan index yang berada > j (lokasi index character yang mismatch pada Pattern P) atau shift character menuju last occurrence x pada Pattern P tidak memungkinkan) Pada kasus ini, i akan di shift sejumlah 1 character ke kanan.



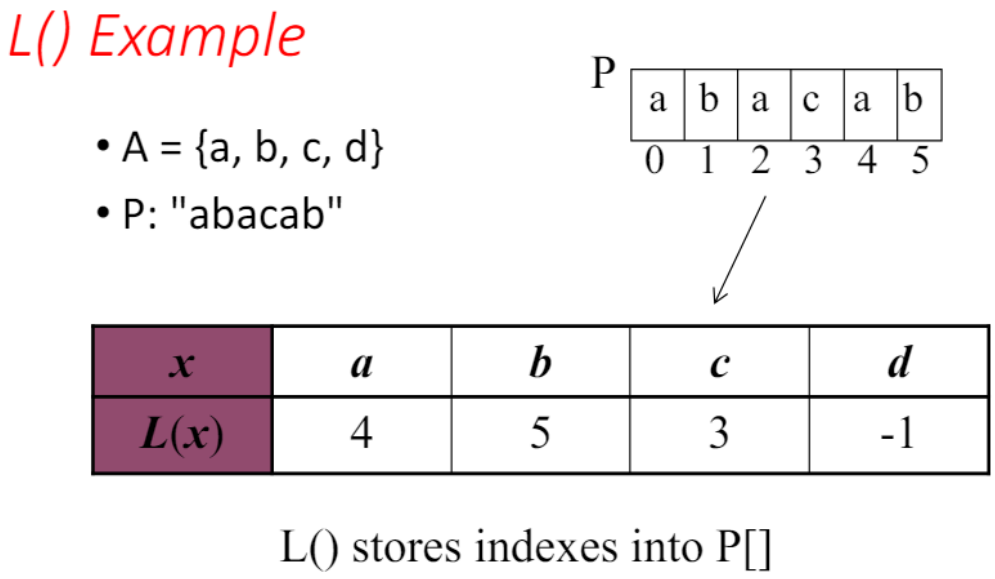
Gambar 2.4 Ilustrasi Case 2 Boyer-Moore Character Jump

Case 3 (Terjadi ketika Pattern P tidak mengandung x (mismatch)) Pada kasus ini, i akan di shift sejumlah panjang pattern



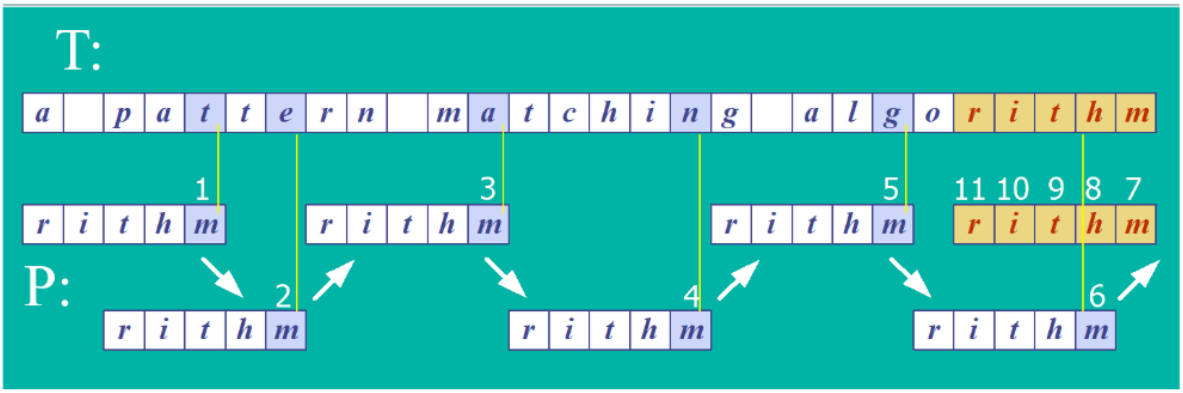
Gambar 2.5 Ilustrasi Case 3 Boyer-Moore Character Jump

Dalam melakukan pencarian dengan algoritma Boyer-Moore, diperlukan bantuan tabel *Last Occurrence* yang berisi informasi index terakhir dari masing-masing karakter di pattern. Berikut adalah contoh dari tabel *Last Occurrence*:



Gambar 2.6 Ilustrasi pembuatan Last Occurrence table|

Algoritma ini selalu melakukan pencocokan dari bagian belakang pattern dan apabila terjadi *mismatch* maka akan dilihat dari kasus 1 atau kasus 2 atau kasus 3. Penanganan setiap kasus sudah dibahas di bagian atas. Berikut adalah contoh cara kerja algoritma Boyer-Moore:



Gambar 2.7 Ilustrasi Pencocokan String dengan Algoritma Boyer-Moore

Regular Expression (REGEX) adalah salah satu metode pencocokan string dengan bantuan pattern. Pattern yang dibentuk merupakan gabungan dari beberapa notasi umum. Notasi umum Regular Expression dapat dilihat pada tabel berikut:

Tabel 2.1 Tabel Notasi Umum Regular Expression

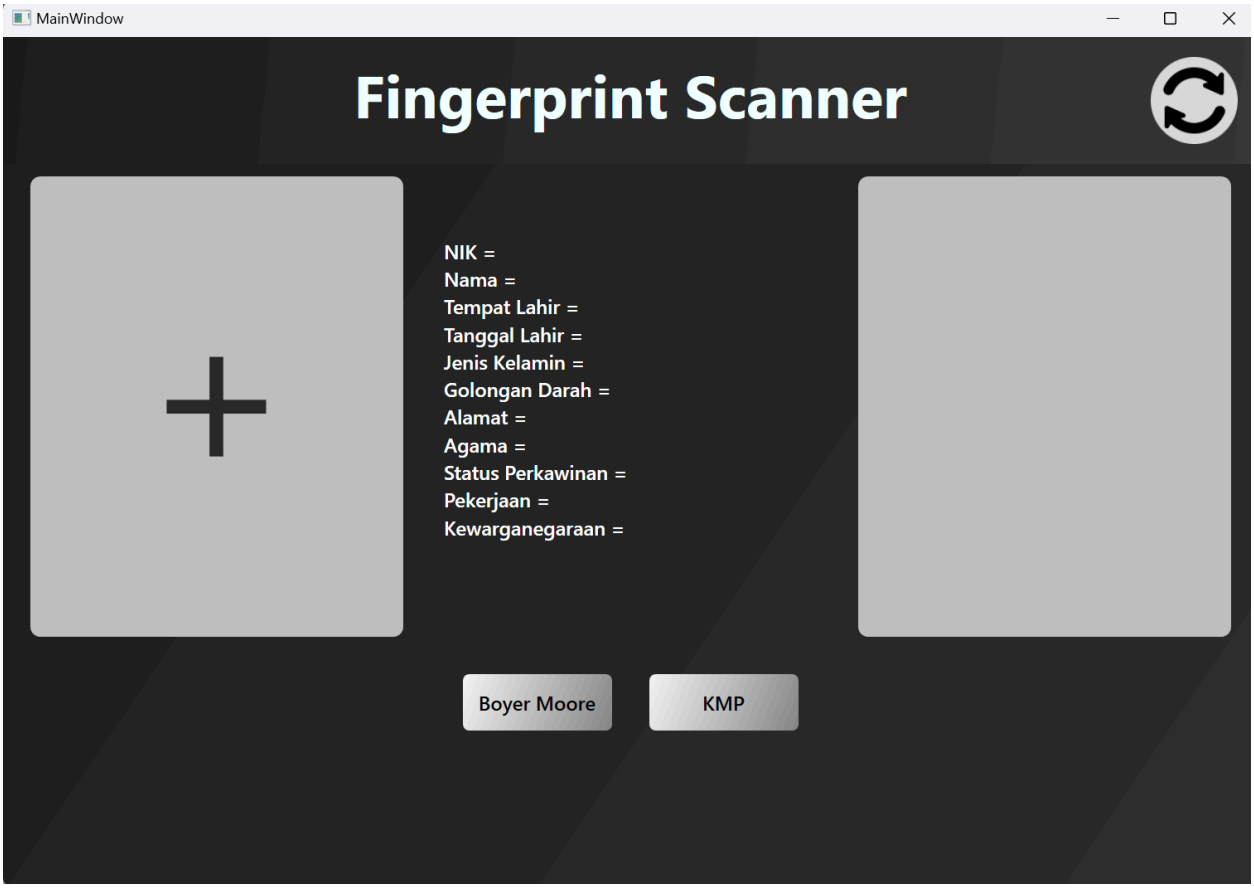
Pattern	Keterangan
`.`	Karakter apapun selain new line
`\.`	Karakter simbol harus diawali dengan \
`^`	Awal string
`$`	Akhir string
`\d`	digit
`\w`	Karakter huruf atau angka atau _
`\s`	spasi
`\D`	Karakter selain digit
`\W`	Karakter selain huruf atau angka atau _
`\S`	Karakter selain spasi
`[abc]`	Karakter a atau b atau c
`[a-z]`	Karakter a sampai z
`[^abc]`	Karakter selain a, b, dan c
`aa|bb`	String aa atau bb
`?`	Jumlah karakter 0 atau 1
`*`	Jumlah karakter 0 atau banyak
`+`	Jumlah karakter 1 atau banyak
`{n}`	Jumlah karakter sebanyak n
`{n,}`	Jumlah karakter minimal sebanyak n
`{m,n}`	Jumlah karakter antara m hingga n
`??, *?, +?, {n}?`	Jumlah karakter sesuai dengan aturan diatas namun seminimal mungkin
`(expr)`	Sebuah expresi
`(?:expr)`	String yang diikuti oleh expr namun expr dianggap sebagai bagian dari string tersebut

(?=expr)	String yang diikuti oleh expr
(?:expr)	String yang tidak diikuti oleh expr

Penjelasan teknik pengukuran persentase kemiripan adalah dengan menggunakan Levenshtein Distance. Levenshtein Distance adalah ukuran kemiripan antara dua string dengan memperhitungkan jumlah operasi penyisipan, penghapusan, dan substitusi yang diperlukan untuk mengubah satu string ke string lainnya. Operasi penyisipan adalah menambahkan karakter ke sebuah string. Operasi penghapusan adalah menghapus karakter dari string. Operasi penggantian adalah mengganti satu karakter dalam string.

Pencarian jarak levenshtein dilakukan dengan membandingkan dua buah string yang panjangnya belum tentu sama. Kemudian mencari jumlah perbedaan karakter untuk setiap karakter pada kedua string hingga salah satu atau keduanya habis atau tidak ada karakter yang bisa dibandingkan lagi. Jika masih ada sisa karakter di salah satu string maka jarak levenshtein ditambah dengan banyaknya karakter sisa.

Aplikasi dekstop yang dibangun adalah sebuah aplikasi yang dapat melakukan pencocokan sidik jari dari sebuah Input dan mampu menampilkan biodata dari seseorang yang sidik jarinya sudah terdaftar pada database. Berikut merupakan tampilan awal dari program yang dibuat :



Gambar 2.8 Tampilan Awal Program

Dalam menjaga kerahasiaan dan keamanan data di basis data, kami melakukan enkripsi data dengan menggunakan algoritma RSA. Untuk melakukan enkripsi dan dekripsi dengan algoritma RSA, diperlukan beberapa tahapan. Tahapan pertama adalah menentukan kunci publik dan kunci privat. Kunci publik yang diperlukan adalah kunci N dan e. Sedangkan kunci privat yang diperlukan adalah d. Dalam menentukan kunci publik dan kunci privat diperlukan beberapa variabel tambahan seperti P dan Q. P dan Q merupakan bilangan prima bebas non-negatif. Untuk melakukan pencarian kunci N, bisa dilakukan dengan mengalikan P dengan Q. Untuk melakukan pencarian kunci e, bisa dilakukan dengan mencari bilangan yang merupakan bilangan relatif prima dari θ(N). θ(N) dapat dicari dengan mengalikan (P-1) dengan (Q-1). Kemudian, untuk mencari kunci d bisa dilakukan dengan menggunakan persamaan

$$ed \equiv 1 \ (mod \ (\theta(N)))$$

Setelah melakukan pencarian kunci, text yang ingin di enkripsi bisa dipecah terlebih dahulu menjadi karakter-karakter yang kemudian akan dienkripsi melalui persaman berikut:

$$c = m^e \, mod \, n$$

Setelah didapatkan cipher teks hasil enkripsi, untuk mendapat teks aslinya bisa dilakukan dekripsi melalui persamaan berikut:

$$m = c^d \, mod \, n$$

BAB 3
ANALISIS PEMECAHAN MASALAH

3.1 Tahapan Pemecahan Masalah

1. Input User

   Pada tahap ini, hanya meminta input dari user berupa input gambar image serta algoritma pattern matching yang ingin digunakan

2. Process Gambar

   Pada awalnya, input gambar yang didapatkan akan diolah menjadi bentuk hitam dan putih dimana pada awalnya image diubah kebentuk grey scaled dan dinormalisasi. Selanjutnya, image akan di cek intensitasnya dengan threshold 127 dimana jika intensitasnya diatas 127 pixel tersebut akan di set ke hitam (biner 1) dan jika sebaliknya, akan di set ke putih (biner 0). Ini berguna untuk menghasilkan image sidik jari yang lebih jelas. Threshold 127 ini adalah titik tengah dari nilai maksimal grayscale yang dapat dihasilkan.

   Setelah itu, akan dihasilkan full text ASCII untuk pencocokan levenshtein distance nantinya dan pattern yang digunakan untuk string matching KMP dan Boyer Moore. Pada awalnya, image akan diubah menjadi bentuk array of bytes yang selanjutnya akan diubah menjadi bentuk ascii-nya. Pada perubahan menuju Ascii, array of bytes yang dihasilkan akan diolah sehingga untuk 1 character ascii akan berisikan 8 bit character dari image. Untuk Pattern sendiri, akan dilakukan croping pada pertengahan image, yaitu tepat pada center of image dengan ukuran 8 karakter ASCII atau 64 bit pada gambar. Pemilihan ukuran tersebut didasarkan atas keakuratan hasil yang didapat dan efisiensi pencocokan. Jika ukuran yang diambil terlalu kecil, maka tingkat keakuratan akan semakin rendah. Hal tersebut dapat terjadi karena semakin kecil ukuran yang diambil maka semakin besar kemungkinan ada pattern yang matching dengan data sidik jari

lainnya. Jika ukuran yang diambil terlalu besar, maka efisiensi pencocokan semakin rendah. Hal tersebut dapat terjadi karena semakin besar ukuran yang diambil, maka pencocokan string akan semakin lama.

3. Pattern Matching

   Setelah image sudah di process akan dilakukan pencocokan string dengan algoritma yang tersedia. Untuk seluruh image dari database, akan dilakukan string matching untuk algoritma yang dipilih. Jika tidak match, maka akan dilakukan perhitungan Similarity yaitu dengan menggunakan Levenshtein Distance.

4. Penentuan Hasil

   Jika hingga akhir tidak ada satupun image pada database yang cocok dengan input yang diberikan, akan ditampilkan biodata dari pengguna dengan similarity terbesar. Namun, jika hasilnya match, maka akan ditampilkan biodata dari data yang match tersebut.

5. Dekripsi Hasil

   Biodata yang ada di basis data merupakan basis data yang merupakan hasil enkripsi RSA. Oleh karena itu, sebelum menampilkan biodata ke aplikasi, maka perlu untuk melakukan dekripsi RSA terhadap biodata tersebut terlebih dahulu.

3.2 Proses Penyelesaian Solusi dengan Algoritma KMP dan BM

Sebelumnya, Pemrosesan gambar harus dilakukan terlebih dahulu, pada process ini akan menghasilkan sebuah pattern yang merupakan ascii dari 64 bit dari input image yang di crop pada titik tengahnya.

Pada Algoritma KMP yang digunakan, akan dikomputasikan terlebih dahulu Border Function-nya. Selanjutnya, untuk tiap image pada dataset yang sudah tertera pada database, akan dilakukan string matching KMP pada tiap imagenya dengan pattern yang ada. Jika match, maka iterasi dihentikan dan hasil akan ditampilkan. Namun jika berbeda, akan dilanjutkan dengan process Levenshtein Distance untuk menghasilkan similarity nya. Jika

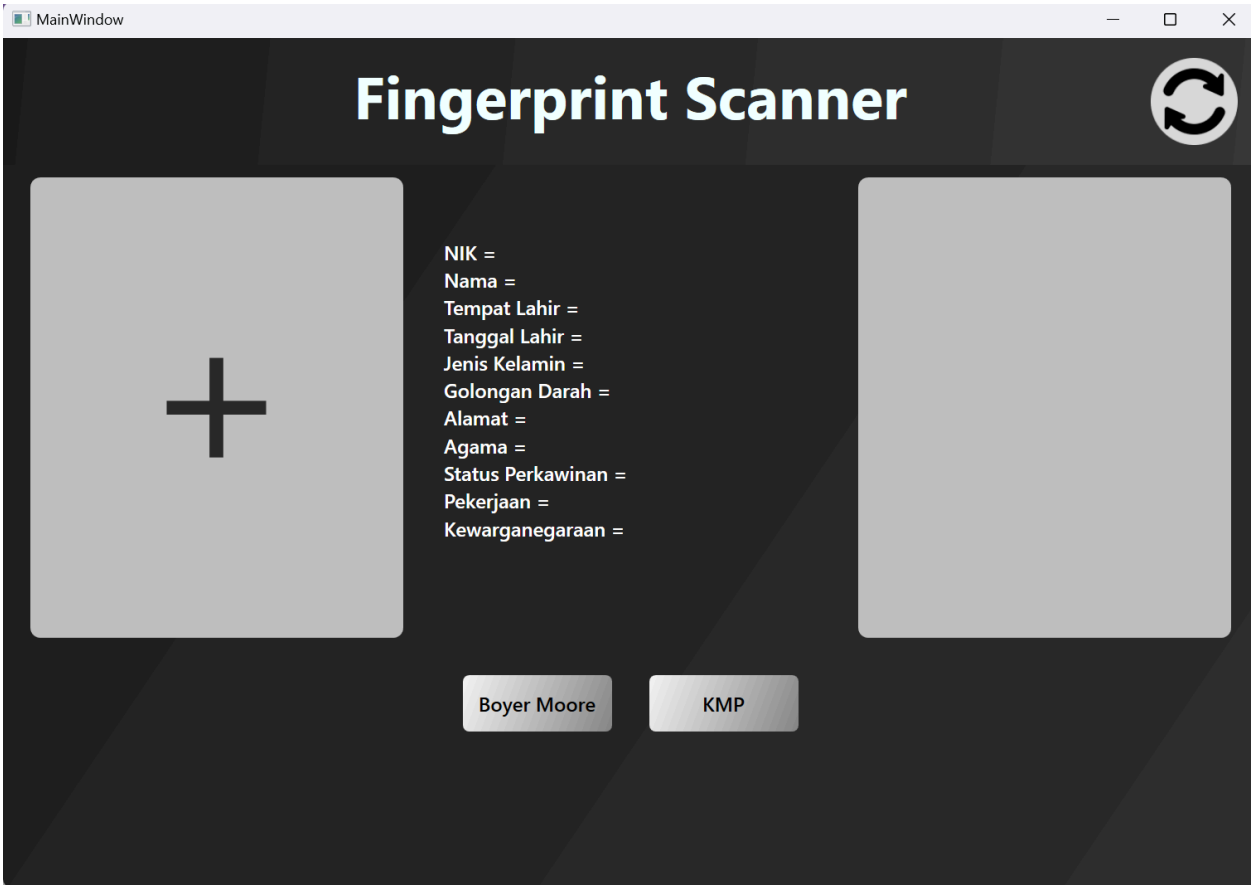
sampai akhir tidak ditemukan yang match, akan digunakan Similarity yang paling besar untuk penampilan biodatanya.

Pada Algoritma Boyer-Moore yang digunakan, akan dikomputasikan terlebih dahulu Last Occurrence Tabelnya. Selanjutnya, untuk tiap image pada dataset yang sudah tertera pada database, akan dilakukan string matching Boyer-Moore pada tiap imagenya dengan pattern yang ada. Jika match, maka iterasi dihentikan dan hasil akan ditampilkan. Namun jika berbeda, akan dilanjutkan dengan process Levenshtein Distance untuk menghasilkan similarity nya. Jika sampai akhir tidak ditemukan yang match, akan digunakan Similarity yang paling besar untuk penampilan biodatanya.

3.3 Proses Pencarian Biodata dengan REGEX

Setelah mendapat kan similarity tertinggi, didapat nama dari pemilik sidik jari nya. Lalu, nama tersebut akan dicocokan dengan pattern REGEX yang didapat dari nama_alay pada tabel Biodata. Pembuatan pattern REGEX dilakukan dengan cara melakukan pembuatan pattern bahasa singkat, dilanjutkan dengan membuat pattern bahasa besar kecil dan diakhiri dengan pembuatan pattern bahasa angka. Pembuatan pattern bahasa singkat dilakukan dengan memasukan semua kemungkinan huruf vokal (a*i*u*e*o*) ke antara 2 huruf non vokal atau angka. Pembuatan pattern bahasa besar kecil adalah dengan membuat semua huruf menjadi bisa huruf besar atau huruf kecil ([aA]). Pembuatan pattern bahasa angka dilakukan dengan mengubah angka menjadi pilihan huruf kecil dan huruf besar, berikut adalah panduan perubahan angka ke huruf 4 → [aA], 1 → [iI], 3 → [eE], 0 → [oO] , 2 → [zZ], 5 → [sS], 6 → [gG].

3.4 Fitur Fungsionalitas dan Arsitektur Aplikasi



Gambar 3.1 Ilustrasi Pencocokan String dengan Algoritma Boyer-Moore

Terdapat beberapa fitur yang disediakan pada Aplikasi ini, yaitu Input Image yang dimaksudkan untuk input sidik jari oleh user. Selanjutnya, terdapat 2 pilihan Algoritma yang dapat digunakan oleh user dalam String Matching. Aplikasi juga dapat menampilkan Output Image dari dataset yang paling cocok dengan Input yang diberikan serta tingkat Similaritynya. Terakhir, Aplikasi juga dapat menampilkan biodata dari user dengan tingkat similarity tertingginya. Pada Aplikasi ini, juga terdapat penggunaan bahasa Alay yang digunakan untuk seeding databasenya. Aplikasi juga memiliki sebuah tombol Refresh yang berguna untuk seeding ulang database. Setelah melakukan pencarian string matching dengan salah satu Algoritma, akan ditampilkan seluruh Bio data user serta persentase similarity image input dengan output tertinggi.

Arsitektur dari Aplikasi yang dibuat adalah sebagai berikut, terdapat sebuah folder bernama Tubes3_BesokMinggu yang merupakan folder berisikan source code dari aplikasi kami. Di Dalam folder tersebut berisikan seluruh source code yang dibutuhkan dalam menjalankan fungsionalitas dari Aplikasi ini, terdapat juga sebuah folder Dataset yang diperuntukkan untuk dataset yang akan digunakan nantinya.

Aplikasi ini menggunakan beberapa tech stack dalam pembuatannya yaitu C# untuk bahasa pemrograman utamanya dengan bantuan Framework WPF untuk UI dan Entity Framework untuk penyambungan databasenya. Database yang digunakan untuk aplikasi ini adalah SQLite.

3.5 Ilustrasi Kasus

Terdapat beberapa kasus yang mungkin untuk dilakukan dengan aplikasi ini :

1. Pencocokan untuk Sidik Jari yang normal

   Pada Kasus ini, kemiripan yang didapatkan akan selalu 100% karena patternya akan sama saat foto yang digunakan tidak dimanipulasi

2. Pencocokan untuk Sidik Jari yang di altered

   Pada kasus ini, sidik jari yang digunakan dapat digeser. Hal ini membuat sidik jari memiliki full text yang berbeda dengan yang normal. Pada kasus ini dapat terjadi similarity yang kurang dari 100 % hal ini karena pattern yang dideteksi bisa saja salah yang mengakibatkan ketidakcocokan

3. Pencocokan untuk Sidik Jari yang di Crop

   Pada kasus ini, sidik jari yang digunakan dapat crop sebagian. Hal ini membuat sidik jari memiliki full text yang berbeda dengan yang normal. Pada kasus ini dapat terjadi similarity yang kurang dari 100 % hal ini karena pattern yang dideteksi bisa saja salah yang mengakibatkan ketidakcocokan

4. Pencocokan untuk Sidik Jari yang di Rotate

Pada kasus ini, sidik jari yang digunakan dapat Rotasi dengan sumbu berapapun. Hal ini membuat sidik jari memiliki full text yang berbeda dengan yang normal. Pada kasus ini dapat terjadi similarity yang kurang dari 100 % hal ini karena pattern yang dideteksi bisa saja salah yang mengakibatkan ketidakcocokan

BAB 4

IMPLEMENTASI DAN PENGUJIAN

4.1 Spesifikasi Teknis Program

Nama Fungsi	Kegunaan
Class : StringMatching	
String[] added	Array yang menyimpan data seluruh text yang sudah diubah menjadi bahasa alay
toBahasaAlay(string) : string	Menghasilkan text dengan bahasa alay secara random yang unique
toBahasaBesarKecil(string) : string	Menghasilkan text yang memiliki ukuran huruf yang bisa besar ataupun kecil
toBahasaAngka(string) : string	Menghasilkan text dengan huruf yang diubah menjadi angka yang ditentukan secara random i.e: (a = 4, s = 5, etc)
toBahasaSingkat(string) : string	Menghasilkan text dengan hasil text yang huruf vokalnya beberapa hilang secara random i.e (makan = mkan)
getBahasaAlayPattern(string) : string	Menghasilkan pattern regex dari bahasa alay yang digunakan
getBahasaBesarKecilPattern(string) : string	Menghasilkan pattern regex dari peubah text ke Besar Kecil
getBahasaAngkaPattern(string) : string	Menghasilkan pattern regex dari peubah text ke angka
getBahasaSingkatPattern(string) : string	Menghasilkan pattern regex dari peubah text ke bahasa dengan penyingkatan huruf vokal
isMatch(string, string) : bool	Mengecek jika text yang dicek cocok dengan Regex pattern yang diberikan
Class : BioData	
String _NIK	Private variable untuk menyimpan data NIK

String NIK	Public property untuk getter setter bagi variable _NIK
String _nama	Private variable untuk menyimpan data nama
String nama	Public property untuk getter setter bagi variable _nama
String _tempat_lahir	Private variable untuk menyimpan data tempat lahir
String tempat_lahir	Public property untuk getter setter bagi variable _tempat_lahir
String _jenis_kelamin	Private variable untuk menyimpan data jenis kelamin
String jenis_kelamin	Public property untuk getter setter bagi variable _jenis_kelamin
String _golongan_darah	Private variable untuk menyimpan data golongan darah
String golongan_darah	Public property untuk getter setter bagi variable _golongan_darah
String _alamat	Private variable untuk menyimpan data alamat
String alamat	Public property untuk getter setter bagi variable _alamat
String _agama	Private variable untuk menyimpan data agama
String agama	Public property untuk getter setter bagi variable _agama
String _status_perkawinan	Private variable untuk menyimpan data status perkawinan
String status_perkawinan	Public property untuk getter setter bagi variable _status_perkawinan
String _pekerjaan	Private variable untuk menyimpan data pekerjaan
String pekerjaan	Public property untuk getter setter bagi

	variable _pekerjaan
String _kewarganegaraan	Private variable untuk menyimpan data kewarganegaraan
String kewarganegaraan	Public property untuk getter setter bagi variable _kewarganegaraan
PropertyChangedEventHandler PropertyChanged;	Inisialisasi event bahwa ada perubahan value attribut.
OnPropertyChanged(string propertyName)	Untuk melakukan pemanggilan bahwa ada yang berubah.

Class : Database	
DbSet<Biodata> ResultData	Variabel yang menjadi jembatan pengatur dari tabel biodata pada database.
string DBPath	Path dari database lokal.
DbSet<sidik_jari> sidik_jari	Variabel yang menjadi jembatan pengatur dari tabel sidik_jari pada database.
OnModelCreating(ModelBuilder modelBuilder) : Void	Fungsi yang dijalankan saat class ini baru dibuat, untuk mapping entitas yang ada pada tabel ke Entity Framework.
OnConfiguring(DbContextOptionsBuilder optionsBuilder) : Void	Fungsi yang dijalankan untuk menghubungkan database dari Entity Framework ke Database lokal.
seedSidikJari(string folderPath) : Void	Untuk mengisi data sidik jari jika sidik jari masih kosong.
seedBiodata() : void	Untuk mengisi data biodata sesuai jumlah sidik jari yang ada
SaveToTextProcessedSidikJari(string path) : Void	Untuk menyimpan hasil proses dari sidik jari ke text untuk caching.
getAllName() : List<string>	Untuk mendapatkan semua nama dari biodata.

Class : ResultData	
PropertyChangedEventHandler PropertyChanged;	Inisialisasi event bahwa ada perubahan value attribut.
OnPropertyChanged(string propertyName)	Untuk melakukan pemanggilan bahwa ada yang berubah.
Biodata _bio	Private variable untuk menyimpan Biodata dari hasil yang didapatkan pada perhitungan
Biodata Bio	Public property untuk _bio
sidik_jari _sidik	Private variable untuk menyimpan sidikjadi dari hasil yang didapatkan pada perhitungan
sidik_jari sidik	Public property untuk _sidik
int _lamaEksekusi	Private variable untuk menyimpan time execution program dari hasil yang didapatkan pada perhitungan
int lamaEksekusi	Public property _lamaEksekusi
Double _kecocokan	Private variable untuk menyimpan Similarity program dari hasil yang didapatkan pada perhitungan
Double kecocokan	Public property untuk _kecocokan

Class : sidik_jari	
String _berkas_citra	Private variable menyimpan image path dari suatu sidik jari
String berkas_citra	Public property untuk _berkas_citra
String _nama	Private variable menyimpan nama dari pemilik sidik jari
String nama	Public property untuk _nama
PropertyChangedEventHandler PropertyChanged;	Inisialisasi event bahwa ada perubahan value attribut.
OnPropertyChanged(string propertyName)	Untuk melakukan pemanggilan bahwa ada

	yang berubah.

Class : LoadingSpinner	
Duration Duration	Durasi dari loading bar tiap berputar
DependencyProperty DurationProperty	Set property Durasi menjadi default
Brush SpinnerColor	Variable untuk custom color modification

Class : Algoritma	
KMPSearch(string, string) : int	Kalkulasi apakah pattern yang di cek match dengan text menggunakan algoritma KMP.
ComputeBorder: int[]	Menghasilkan Last Border Function untuk algoritma KMP
BoyerMoore(string, string) : int	Kalkulasi apakah pattern yang di cek match dengan text menggunakan algoritma Boyer Moore.
BuildLast(string) : int[]	Menghasilkan Last Occurrence list untuk algoritma Boyer Moore.
LevenshteinDistance(string, string, int) : int	Menghasilkan distance antara 2 string. Digunakan nanti untuk kalkulasi similarity.

Class : MainWindow	
Database db	Menyimpan database yang digunakan pada apps
String _path	Variable untuk menyimpan data input path dari input image
ResultData ResultData	Variable untuk menyimpan hasil perhitungan
ImageButton_Click(object,	Method yang dipanggil ketika Input button

RoutedEventArgs) : void	di click. Digunakan untuk menyimpan data input user dan menampilkan pilihan gambar dari user
KMPClick(object, RoutedEventArgs) : void	Method yang dipanggil ketika KMP button di click. Digunakan untuk melakukan pattern matching pada input dengan algoritma KMP
BoyerMooreClick(object, RoutedEventArgs) : void	Method yang dipanggil ketika Boyer Moore button di click. Digunakan untuk melakukan pattern matching pada input dengan algoritma Boyer Moore
HandleButtonReColor(bool, object) : void	Method untuk recolor button KMP atau Boyer moore sehingga dapat terlihat button yang sudah di klik yang mana
RefreshClick(object, RoutedEventArgs) : void	Method untuk refresh page, digunakan untuk refresh data database dsb.
ConvertHexStringToColor(string) : Color	Merubah hex ke Color i.e (“#FFFFFF” menjadi Color.White)
HandleResultData(double) : void	Handling Result data ketika algoritma berhasil dijalankan
HandleSimilarityNumber(double) : void	Handling warna dari text Similarity untuk nilai similarity tertentu.

Class : RSA	
P, Q, D	Variable private key
N, E	Variable public key
getN() : long	Method untuk meng-generate public key N dan private P, Q
getE() : int	Method untuk meng-generate public key E
phiN() : long	Method untuk menghitung

isFactorOf(int, long) : bool	Method untuk melakukan pengecekan faktor dari sebuah bilangan
GCD(long, long) : long	Method untuk mencari faktor persekutuan terbesar dari 2 bilangan
isPrime(int) : bool	Method untuk melakukan pengecekan bilangan prima
getD() : long	Method untuk mencari private key D
encoder(string) : string	Method untuk melakukan encoding RSA teks ke cipherteks
encrypt(int) : long	Method untuk melakukan enkripsi RSA byte
decoder(string) : string	Method untuk melakukan decoding RSA cipherteks ke teks asli
decrypt(long) : long	Method untuk melakukan dekripsi RSA

4.2 Tata Cara Penggunaan

4.2.1 Download Executable

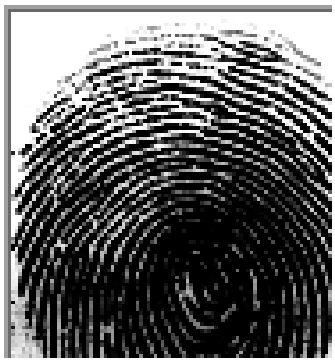
Jika Executable yang di unduh, maka buka aplikasi folder yang di unduh. Jalankan Aplikasi executable yang ada. Masukkan Dataset pada folder Dataset, lalu pencet tombol refresh untuk membuat database sidik jari. Setelah loading selesai, pencet tombol (+) yang ada di kiri tengah. Masukkan sidik jari yang ingin dicari, lalu pilih algoritma yang ingin digunakan untuk mencari.

4.2.2 Source code

Jika Source code yang di unduh, maka buka project sln menggunakan IDE yang Visual Studio atau Rider. Lakukan build / compile. Masukkan Dataset pada folder Dataset, lalu pencet tombol refresh untuk membuat database sidik jari. Setelah loading selesai, pencet tombol (+) yang ada di kiri tengah. Masukkan sidik jari yang ingin dicari, lalu pilih algoritma yang ingin digunakan untuk mencari.

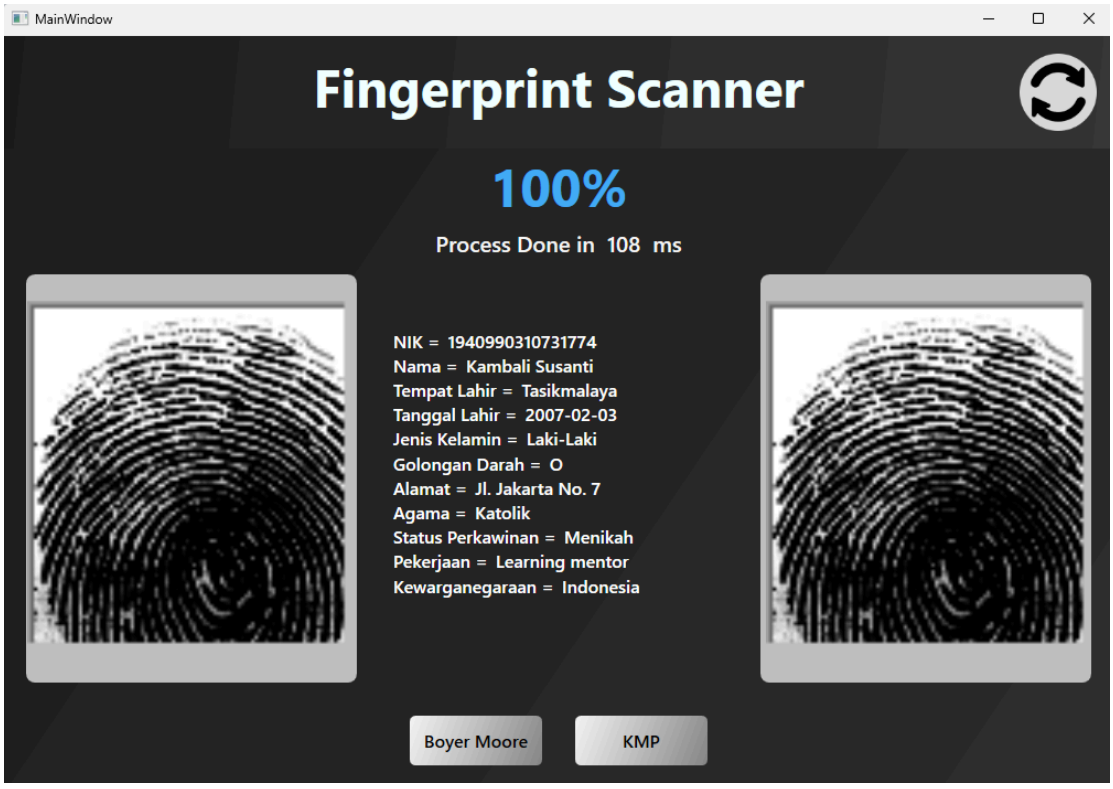
4.3 Hasil Pengujian

1. Test 1 : Input Image :

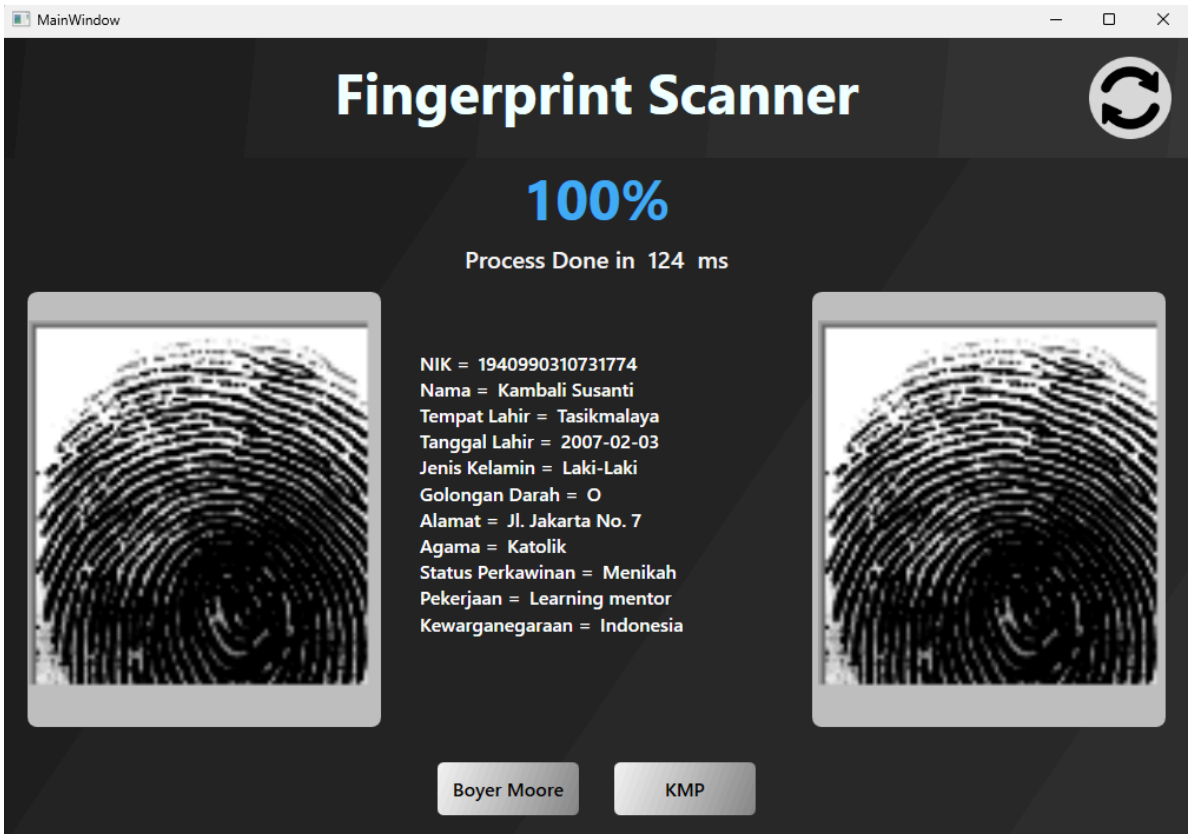


Output:

Boyer Moore :



Kmp :

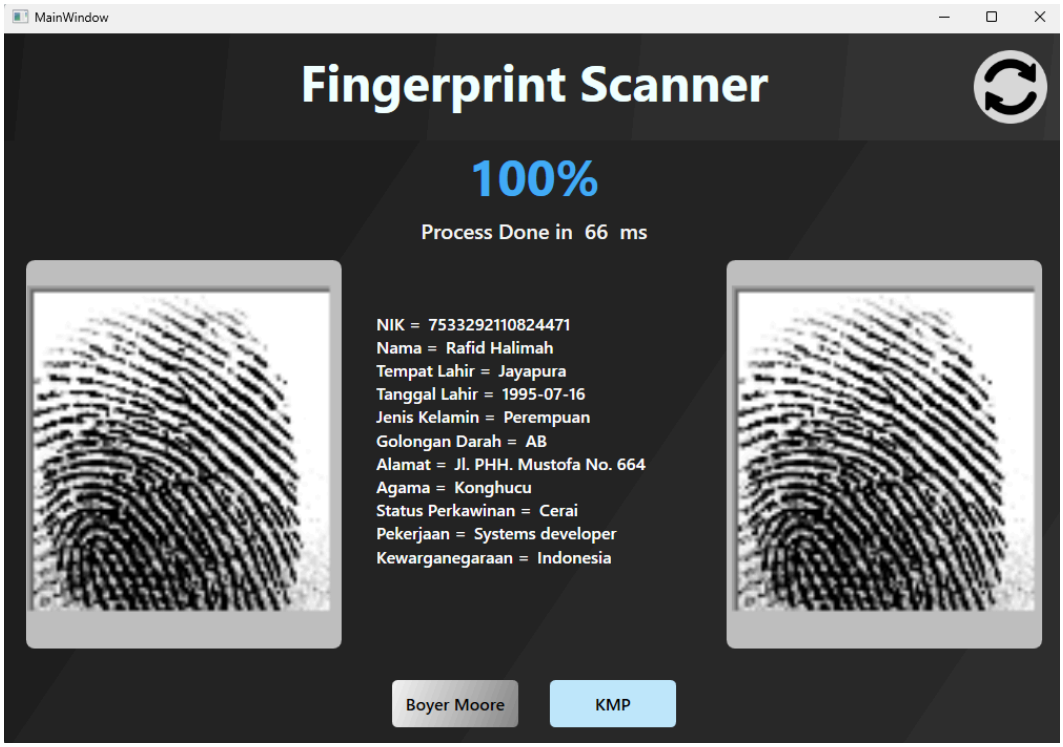


2. Test 2 : Input Image :

Output:

Boyer Moore :

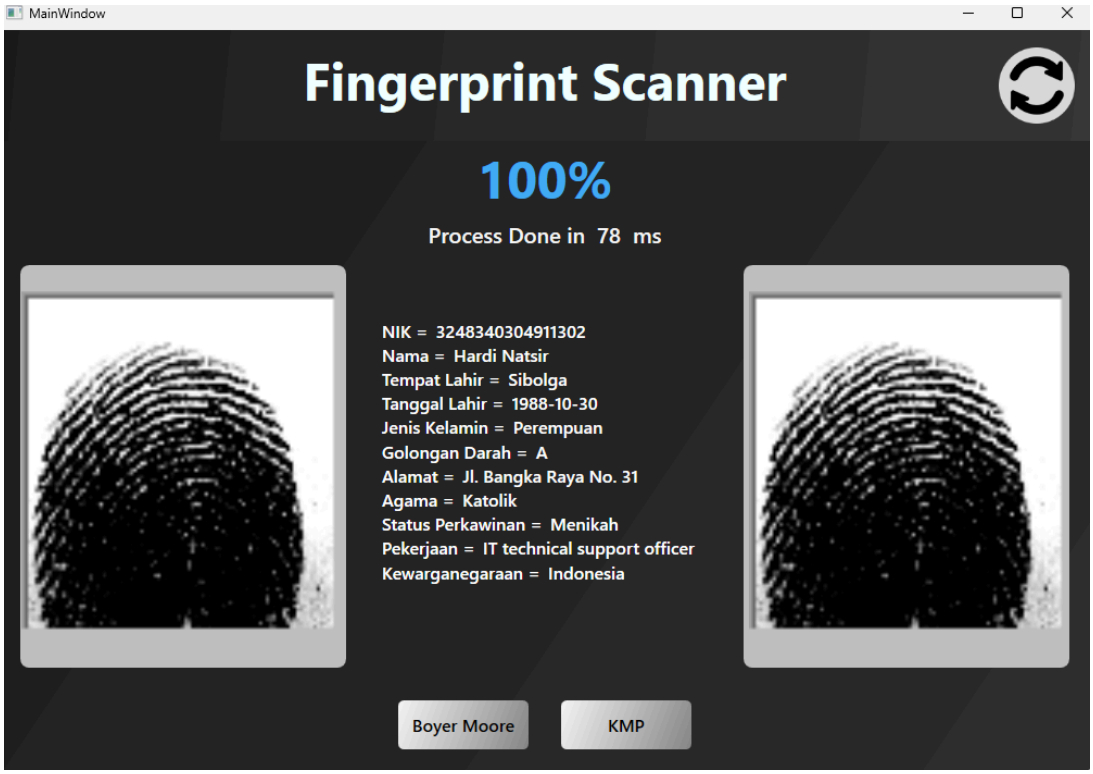
Kmp :



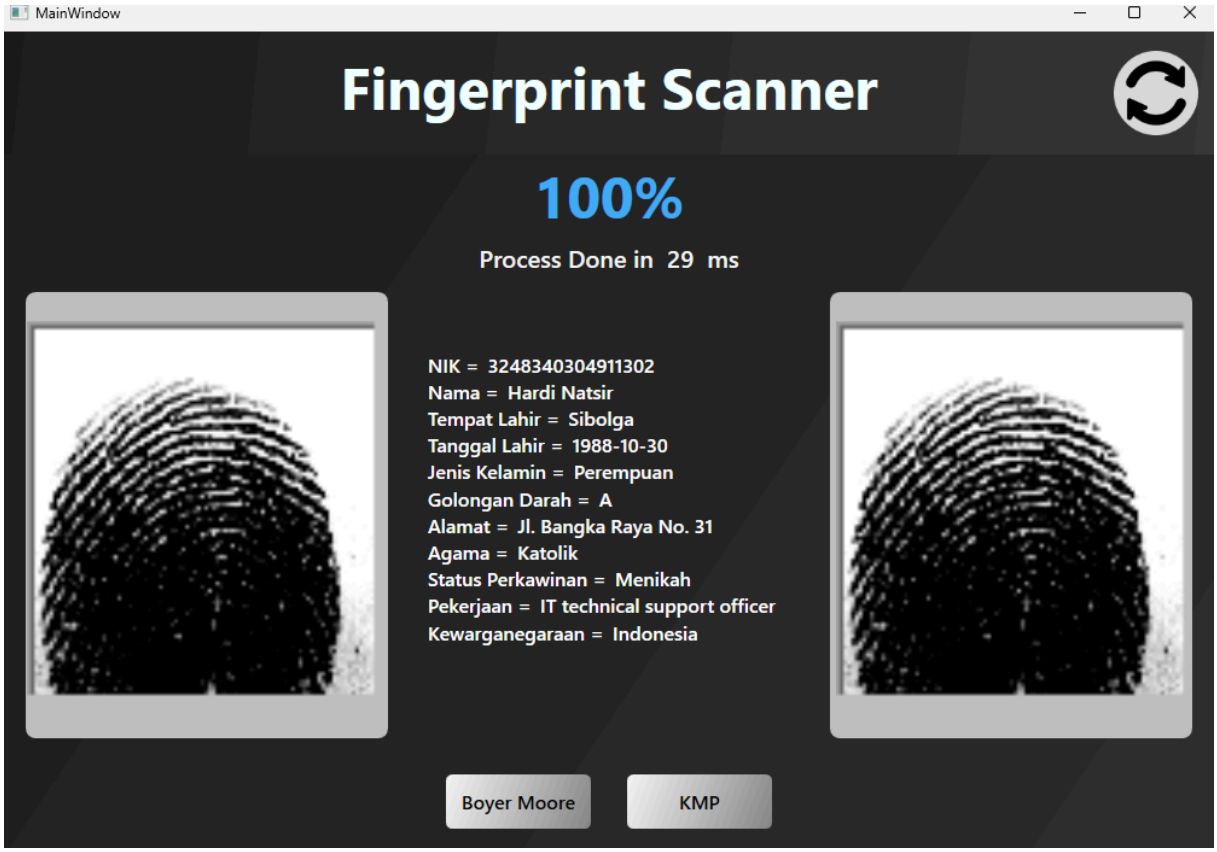
3. Test 3 : Input Image :

Output:

Boyer Moore :



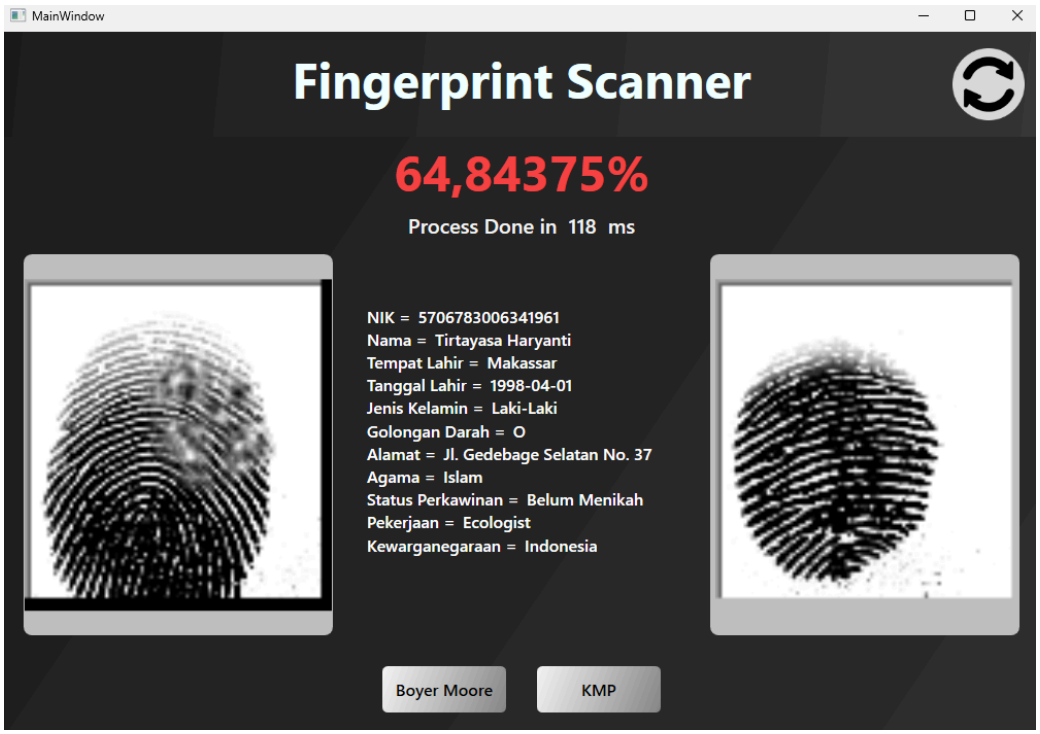
Kmp :



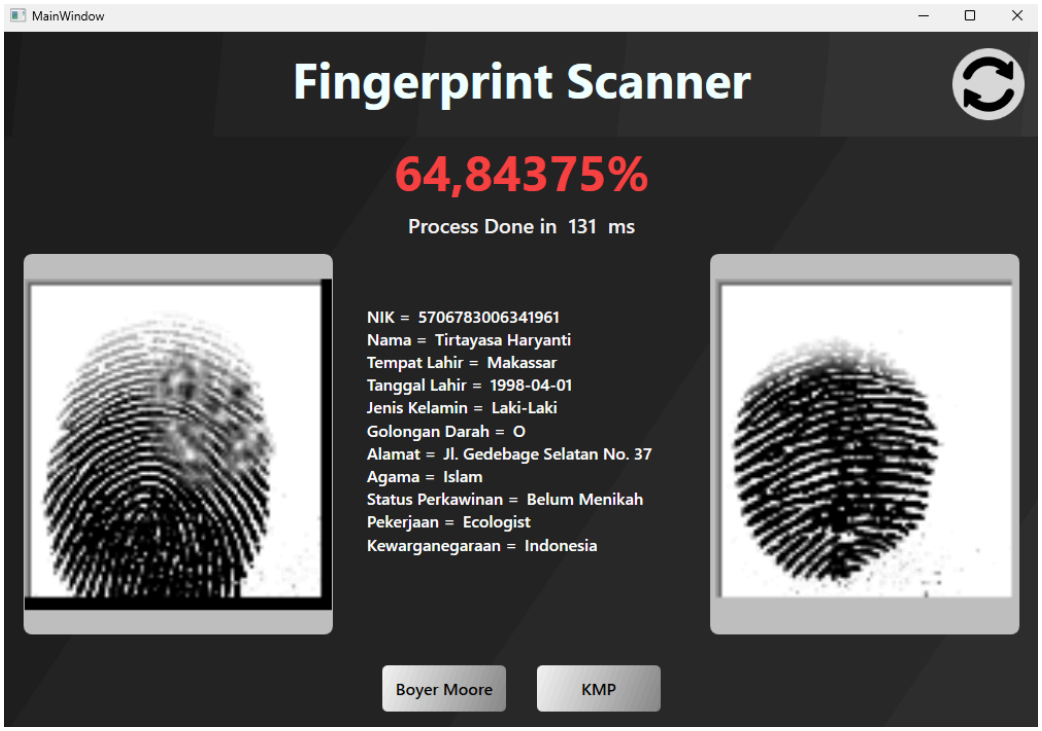
4. Test 4 : Input Image :

Output:

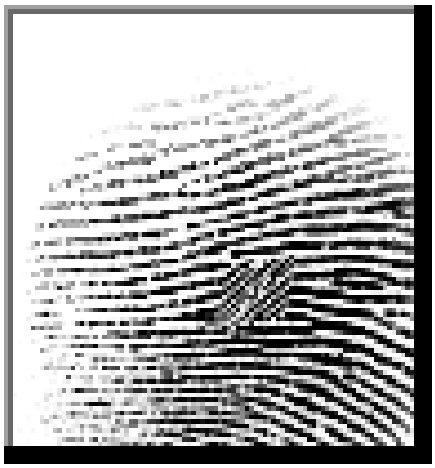
Boyer-Moore:



KMP:

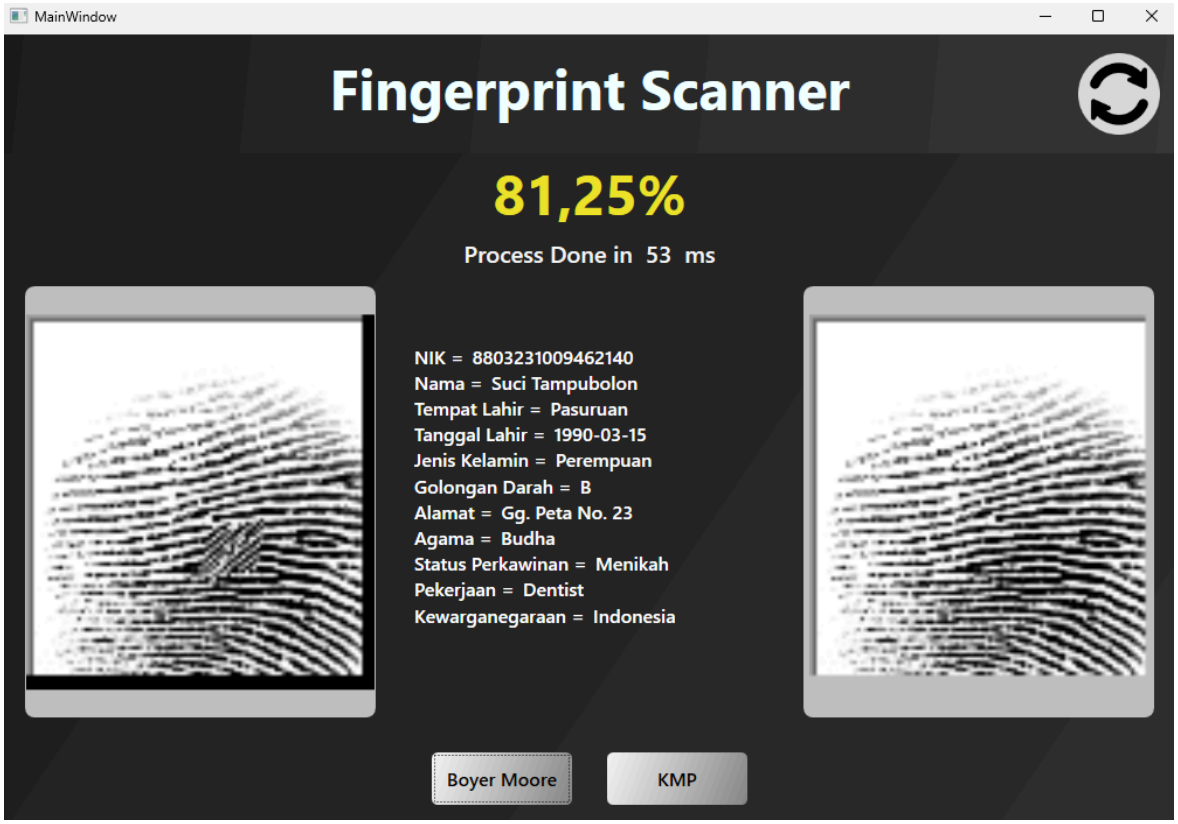


5. Test 5 : Input Image :

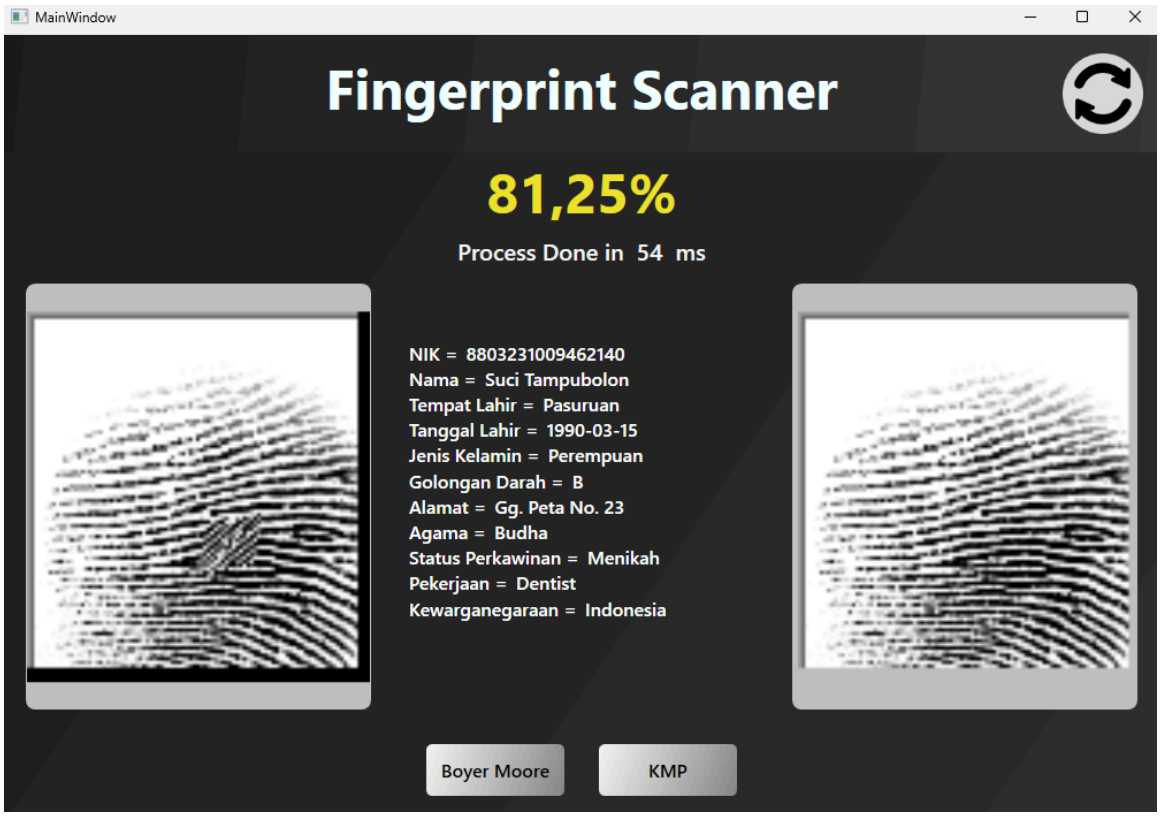


Output:

Boyer-Moore:



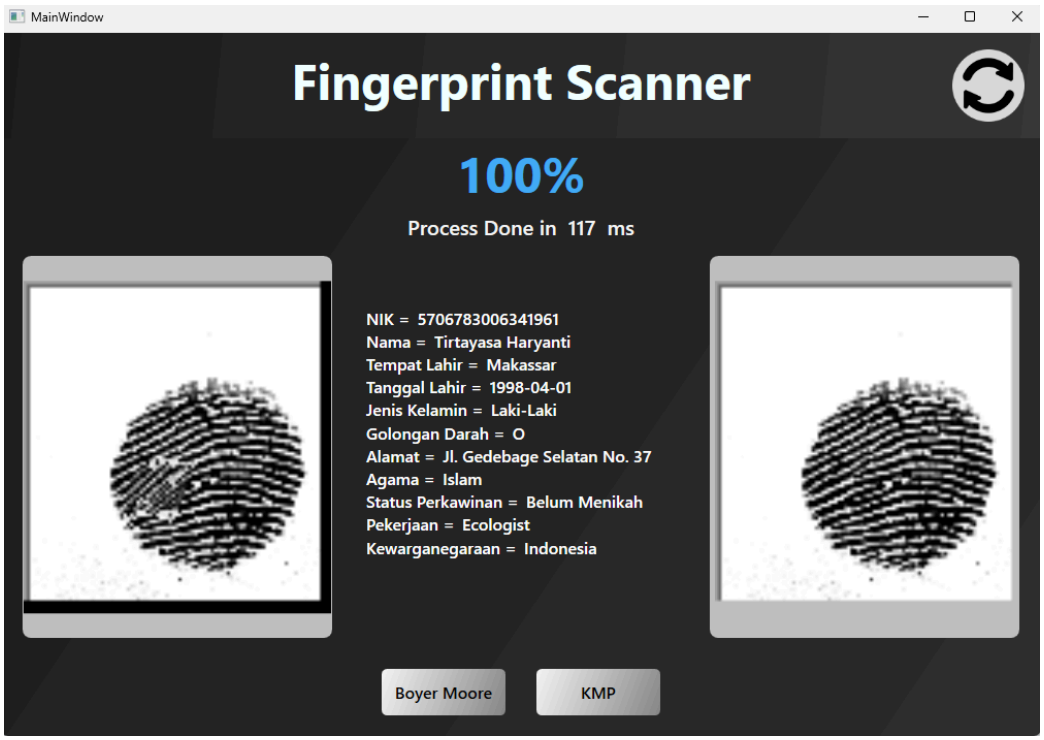
KMP:



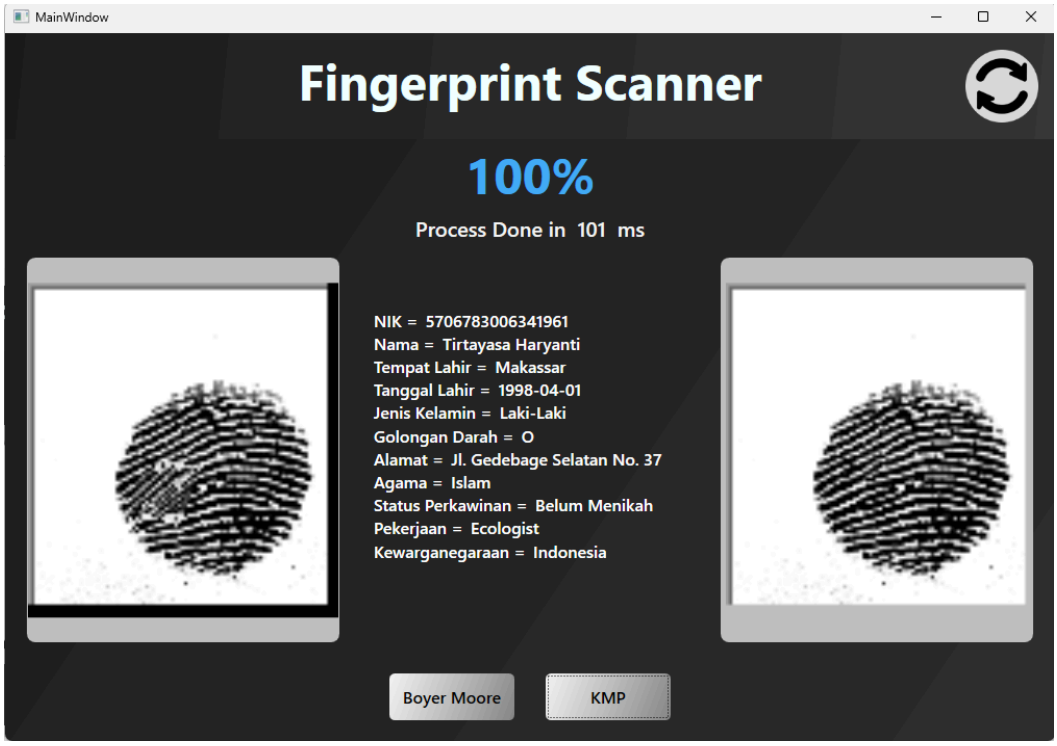
6. Test 6 : Input Image :

Output:

Boyer-Moore:



KMP:



4.4 Analisis Hasil Pengujian

Program bekerja dengan baik sesuai dengan yang diinginkan. Program dapat melakukan prosesing menggunakan Database, Regex, dan String Matching.

Kedua algoritma berhasil menemukan sidik jari yang sesuai untuk sidik jari yang alterisasi nya hanya terpotong, namun gagal untuk sidik jari yang dirotasi.

Kedua algoritma bekerja dengan sangat cepat. Di bawah 150 milidetik. Dengar rata-rata 87,7 milidetik.

BAB 5
KESIMPULAN

5.1 Kesimpulan

Algoritma Knuth Morris Pratt dan algoritma Boyer Moore dapat menjadi algoritma untuk Fingerprint Recognition dengan cukup baik. Walau masih belum bisa menangani beberapa kasus untuk sidik jari yang dirotasi, sudah cukup cepat dalam membandingkan banyak sidik jari.

5.2 Saran

Untuk mendapatkan hasil yang lebih akurat kita bisa menggunakan algoritma yang mencari titik tengah dari sidik jari yang dipindai. Dengan begitu yang yang dibandingkan berkemungkinan lebih mirip dibandingkan hanya mengambil titik tengah dari gambar.

Untuk UI bisa lebih diperindah untuk membuat pengguna lebih nyaman dalam menggunakan program. Dan ditambahkan beberapa fitur tambahan seperti animasi pencarian untuk membuat aplikasi lebih keren.

5.3 Tanggapan

Algoritma KMP dan Boyer Moore cukup unggul dalam pengimplementation FingerPrint Recognition. Meskipun demikian, masih banyak hal yang masih dapat diperbaiki seperti pengimplementasian untuk image yang diputar.

5.4 Refleksi

Eksplorasi database lokal dan constraintnya kurang menyeluruh, sehingga mendapatkan masalah yang tidak diketahui saat masa development. Kurangnya pemahaman atas Processing Image juga membuat penulis harus mengeksplorasi banyak jurnal artikel untuk membuat perbandingan lebih akurat.

LAMPIRAN

Tautan Repository: fauzanazz/Tubes3_BesokMinggu: Tugas Besar 3 Strategi Algoritma (github.com)

Tautan Video: https://youtu.be/o6ak3o0Ti9k

DAFTAR PUSTAKA

Munir, Rinaldi (2023). Algoritma RSA. informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2020-2021/Algoritma-RSA-2020.pdf (diakses pada 9 Juni 2024)

Munir, Rinaldi (2023). Pencocokan String (String/Pattern Matching). informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf (diakses pada 9 Juni 2024)

Khodra, Masayu Leylia (2023). String Matching dengan Regular Expression. informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2022-2023/String-Matching-dengan-Regex-2019.pdf (diakses pada 9 Juni 2024)