

LAPORAN TUGAS BESAR 3

IF2211 STRATEGI ALGORITMA



Disusun oleh:

Yusuf Ardian Sandi	13522015
Tazkia Nizami	13522032
Muhamad Rifki Virziadeili Harisman	13522097

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2024

DAFTAR ISI

DAFTAR ISI.....	1
BAB I.....	3
DESKRIPSI TUGAS.....	3
BAB II.....	5
1. Penjelasan String Matching.....	5
2. Algoritma KMP.....	5
4. Algoritma BM.....	6
5. Regex.....	7
6. Pengukuran Similaritas.....	8
7. AES 128.....	9
8. Aplikasi Desktop KacaprukReborn.....	9
BAB III.....	10
1. Langkah-langkah Pemecahan Masalah.....	10
2. Proses Penyelesaian Masalah dengan algoritma KMP dan BM.....	10
3. Fitur fungsional dan arsitektur aplikasi web yang dibangun.....	12
3.1. Button Upload Gambar.....	12
3.2. Select Dropdown.....	13
3.3. Tampilan Informasi Waktu Eksekusi, Persentase Kecocokan, dan Biodata.....	14
4. Contoh Ilustrasi Kasus.....	14
BAB IV.....	15
1. Spesifikasi Teknis Program.....	15
1.1. Struktur Data.....	15
2. Tata Cara Penggunaan Program.....	21
2.1. Menjalankan Program.....	21
2.2. Menggunakan Program.....	21
3. Hasil Pengujian.....	27
3.1. KMP.....	27
3.2. BM.....	31
4. Analisis Hasil Pengujian.....	35
BAB V.....	37
Kesimpulan.....	37
Saran.....	37
Refleksi.....	37
LAMPIRAN.....	38
DAFTAR PUSTAKA.....	39

BAB I

DESKRIPSI TUGAS

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.

Buatlah sebuah sistem yang dapat melakukan identifikasi individu berbasis biometrik dengan menggunakan sidik jari dengan detail sebagai berikut.

1. Sistem dibangun dalam bahasa C# dengan kakas Visual Studio .NET yang mengimplementasikan algoritma KMP, BM, dan Regular Expression dalam mencocokkan sidik jari dengan biodata yang berpotensi rusak. Pelajarilah C# desktop development, disarankan untuk menggunakan Visual Studio untuk mempermudah penggerjaan.
2. Program dapat memiliki basis data SQL yang telah mencocokkan berkas citra sidik jari yang telah ada dengan seorang pribadi. Basis data yang digunakan dibebaskan asalkan bukan No-SQL (sebagai contoh, MySQL, PostgreSQL, SQLite).

3. Program dapat menerima masukan sebuah citra sidik jari yang ingin dicocokkan. Apabila citra tersebut memiliki kecocokan di atas batas tertentu (silakan lakukan tuning nilai yang tepat) dengan citra yang sudah ada, maka tunjukkan biodata orang tersebut. Apabila di bawah nilai yang telah ditentukan tersebut, memunculkan pesan bahwa sidik jari tidak dikenali.
4. Program memiliki keluaran yang minimal mengandung seluruh data yang terdapat pada contoh antarmuka pada bagian penggunaan program.
5. Pengguna dapat memilih algoritma yang ingin digunakan antara KMP atau BM.
6. Biodata yang ditampilkan harus biodata yang memiliki nama yang benar (gunakan Regex untuk memperbaiki nama yang rusak dan gunakan KMP atau BM untuk mencari orang yang paling sesuai).
7. Program memiliki antarmuka yang user-friendly. Anda juga dapat menambahkan fitur lain untuk menunjang program yang Anda buat (unsur kreativitas).

BAB II

LANDASAN TEORI

1. Penjelasan String Matching

String matching adalah proses menemukan kemunculan suatu pola (pattern) dalam teks yang lebih besar. Dalam konteks pengenalan sidik jari, string matching dapat digunakan untuk membandingkan representasi ASCII 8-bit dari sidik jari yang ingin dicari dengan representasi ASCII 8-bit dari setiap gambar pada basis data sidik jari yang sudah ada. Proses ini penting untuk memastikan sidik jari yang dimasukkan memang ada pada basis data.

Untuk mengimplementasikan string matching dalam program identifikasi sidik jari, kita menggunakan dua algoritma utama: Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM). Kedua algoritma ini memiliki pendekatan yang berbeda dalam mencari pola dalam teks, masing-masing dengan kelebihan dan kekurangannya sendiri.

2. Algoritma KMP

Algoritma KMP adalah salah satu algoritma string matching yang efisien karena dapat melakukan pencarian dengan kompleksitas waktu $O(n + m)$, di mana n adalah panjang teks dan m adalah panjang pola. Algoritma ini bekerja dengan menggunakan informasi yang diperoleh dari pola untuk menghindari pencocokan karakter yang tidak perlu. Secara umum, terdapat dua tahapan saat melakukan pemrosesan dengan algoritma KMP, yaitu preprocessing dan matching

- Preprocessing:
 1. Membuat tabel "longest proper prefix which is also suffix" (LPS) untuk pola. Tabel ini membantu menentukan seberapa banyak pergeseran yang diperlukan ketika terjadi ketidakcocokan.
 2. Tabel LPS dibuat dengan menganalisis pattern itu sendiri dan mencatat panjang dari prefiks terpanjang yang juga merupakan sufiks untuk setiap posisi dalam pattern.

- Matching:
 1. Mulai pencocokan dari awal teks dan pola.
 2. Jika terjadi kecocokan karakter antara teks dan pola, teruskan ke karakter berikutnya.
 3. Jika terjadi ketidakcocokan, gunakan tabel LPS untuk menentukan pergeseran yang diperlukan tanpa harus membandingkan ulang karakter yang sudah diketahui cocok.

4. Algoritma BM

Algoritma Boyer-Moore (BM) adalah salah satu algoritma string matching yang efisien dan sering digunakan karena dapat melakukan pencarian dengan kompleksitas waktu rata-rata yang sangat baik. Algoritma ini memanfaatkan informasi yang diperoleh dari pola dan teks untuk menghindari pencocokan karakter yang tidak perlu, sehingga mempercepat proses pencarian. Secara umum, terdapat dua tahapan utama dalam pemrosesan dengan algoritma BM, yaitu preprocessing dan matching.

- Preprocessing:

Pada tahap preprocessing, algoritma Boyer-Moore membangun dua tabel penting:

Tabel Bad Character (Karakter Buruk): Tabel ini digunakan untuk menentukan seberapa jauh pola dapat digeser ketika terjadi ketidakcocokan karakter. Untuk setiap karakter dalam alfabet, tabel ini mencatat posisi kemunculan terakhir karakter tersebut dalam pola. Jika karakter yang tidak cocok ditemukan dalam teks, pola dapat digeser ke kanan sejauh mungkin sehingga karakter yang tidak cocok tersebut berada di bawah kemunculan terakhirnya dalam pola.

- Matching:

Mulai Pencocokan dari Akhir Pola: Algoritma Boyer-Moore memulai pencocokan dari akhir pola dan teks, bukan dari awal seperti pada kebanyakan algoritma pencocokan lainnya. Ini memungkinkan penggunaan informasi dari tabel Bad Character dan Good Suffix secara lebih efektif.

Jika Terjadi Kecocokan Karakter: Jika karakter dari pola dan teks cocok, lanjutkan pencocokan ke karakter sebelumnya (bergerak ke kiri dalam pola dan teks).

Jika Terjadi Ketidakcocokan Karakter: Ketika karakter tidak cocok ditemukan, gunakan tabel Bad Character untuk menentukan seberapa jauh pola dapat digeser. Jika tabel Bad Character tidak memberikan pergeseran yang cukup besar, gunakan tabel Good Suffix untuk menentukan pergeseran yang lebih baik.

5. Regex

Penggunaan regex di sini bertujuan untuk menangani kolom nama yang korup pada tabel biodata. Adapun pendefinisian bahasa alay di sini adalah sebagai berikut

Contoh kasus bahasa alay dijelaskan dengan detail sebagai berikut.

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

Dari contoh kasus tersebut kita mendapatkan beberapa kategori kata alay yang harus ditangani

a. Kombinasi huruf besar-kecil

Untuk menangani kasus ini, kita dapat membuat semua huruf yang terdapat pada kata alay dibuat menjadi huruf kecil.

b. Penggunaan angka

Untuk menangani kasus penggunaan angka di sini saya mendefinisikan apa saja sekiranya angka yang memiliki makna sama dengan suatu huruf. Kami mendaftarkan sekumpulan angka yang memiliki makna sama dengan suatu huruf dengan cara mengetestnya di [bahasa alay generator](#). Berikut adalah sekumpulan pasangan angka-huruf yang kami dapatkan

pasangan['4'] = 'a';

pasangan['3'] = 'e';

pasangan['6'] = 'g';

pasangan['1'] = 'i';

pasangan['0'] = 'o';

pasangan['5'] = 's';

pasangan['2'] = 'z';

c. Penyingkatan

Sementara untuk menangani kasus penyingkatan, pengecekan dilakukan dengan mengiterasi seluruh karakter pada kata alay dan membandingkannya dengan karakter pada kata asli. Jika karakter pada indeks tertentu sama, maka pindah ke karakter berikutnya. Jika beda, maka cek apakah karakter pada kata asli itu huruf vokal dan karakter pada kata alay adalah konsonan, maka pindah ke karakter berikutnya di kata asli untuk mengabaikan huruf vokal pada kata asli. Jika setelah diabaikan, konsonan pada karakter asli tidak sama dengan karakter di kata alay, maka kata tersebut tidak memiliki makna yang sama dengan kata alay yang kita inginkan. Pengecekan tersebut terus dilakukan hingga pengecekan di index terakhir kata alay bernilai benar, maka dapat dinyatakan bahwa kata alay tersebut memiliki makna yang sama dengan kata ori yang kita punya.

d. Kombinasi

Untuk menangani kasus ini berarti kita akan melakukan konversi untuk seluruh kasus. Justru ini yang akan kita lakukan. Pada saat pengecekan kita akan mengonversi seluruh kata pada kata alay menjadi huruf kecil semua, kemudian mengganti seluruh angka yang terdapat pada kata alay menjadi huruf yang memiliki makna serupa. Setelah itu, kita akan melakukan pengecekan apakah kata alay tersebut memiliki makna sama dengan kata yang kita punya dengan cara mengabaikan huruf vokal yang terdapat pada kata asli.

6. Pengukuran Similaritas

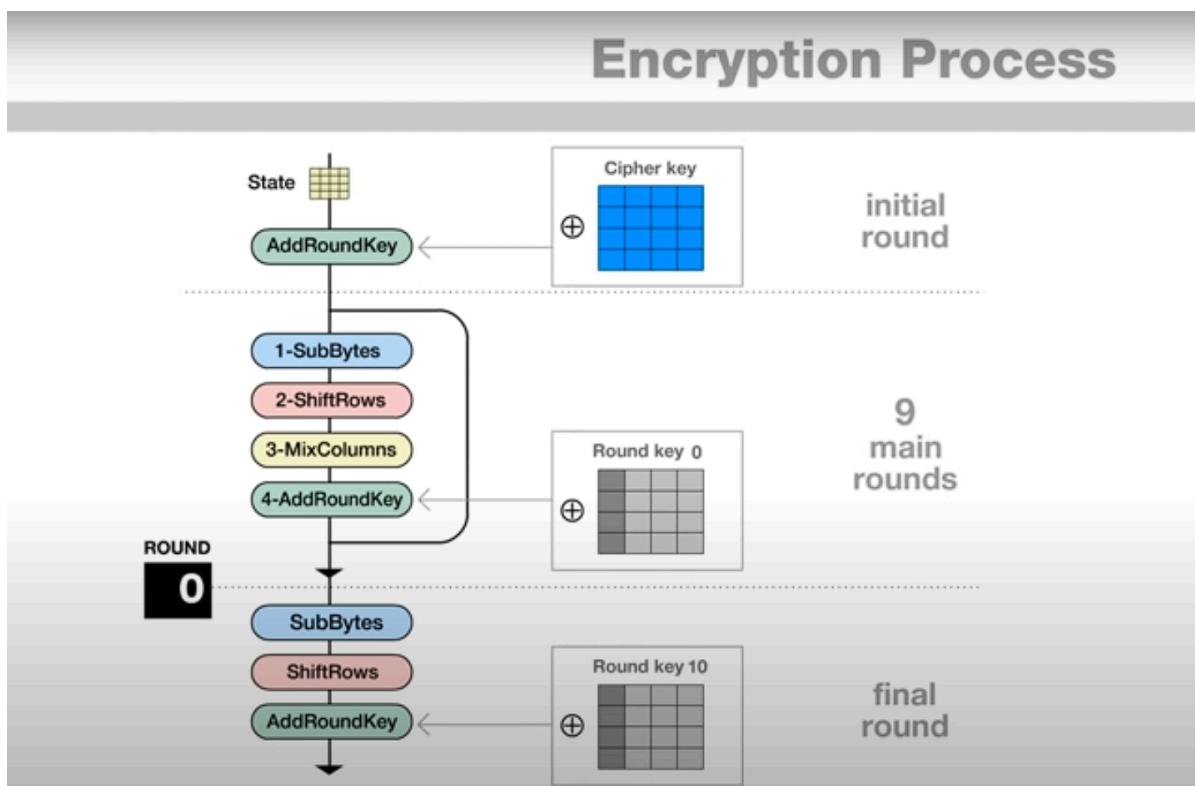
Pengukuran Similaritas diperlukan ketika pattern tidak ditemukan pada semua representasi ASCII 8-bit dari gambar di basis data. Pengukuran similaritas akan mencari gambar pada basis data dengan menggunakan algoritma Longest Common Subsequence (LCS). Algoritma LCS adalah algoritma yang bertujuan untuk mencari substring terpanjang yang muncul di kedua teks.

Setelah mendapatkan substring terpanjang, maka persentase kemiripannya kemudian dihitung dengan membandingkan panjang dari substring tersebut dengan panjang dari keseluruhan text. Perhitungan similaritas ini dilakukan untuk setiap gambar pada basis data, kemudian gambar dengan similaritas tertinggi dan melewati threshold 70 persen akan ditampilkan.

7. AES 128

Untuk mengamankan data pada tabel biodata, kami juga mengimplementasikan enkripsi-dekripsi dengan algoritma AES 128. AES adalah sebuah algoritma enkripsi simetris yang menggunakan kunci yang sama untuk enkripsi dan dekripsi. Algoritma ini bekerja dengan membagi data menjadi blok-blok yang sama besar dan kemudian melakukan serangkaian operasi enkripsi pada setiap blok.

AES memiliki tiga ukuran kunci yang mungkin: 128-bit, 192-bit, dan 256-bit. Setiap ukuran kunci mengharuskan jumlah putaran yang berbeda dalam proses enkripsi. Pada program ini kami mengimplementasikan AES dengan key berukuran 128-bit.



8. Aplikasi Desktop KacaprukReborn

Aplikasi desktop yang dibangun bernama KacaprukReborn ini dikembangkan dengan Avalonia menggunakan bahasa pemrograman C#. Avalonia adalah framework UI lintas platform yang memungkinkan pengembangan antarmuka pengguna untuk Windows dan Linux. Aplikasi ini bertujuan untuk melakukan identifikasi individu berbasis biometrik dengan menggunakan sidik jari. Mula-mula pengguna memilih gambar dengan extensi .BMP kemudian memilih algoritma string matching yang ingin digunakan. Setelah pengguna menekan tombol search, biodata dari hasil identifikasi akan ditampilkan jika ditemukan kecocokan diatas 70 persen.

BAB III

ANALISIS PEMECAHAN MASALAH

1. Langkah-langkah Pemecahan Masalah

- Memroses semua gambar yang ada pada database untuk disimpan teks hasil konversi dari gambar ke ASCII pada suatu struktur data map
- Mendapat input gambar dari pengguna lalu menyimpan tkes hasil konversi dari gambar ke ASCII pada suatu variabel string
- Melakukan iterasi pada map dan melakukan pencocokan string dengan KMP/BM pada setiap iterasinya
- Jika ditemukan ASCII yang cocok pada map, maka akan ditampilkan gambar sidik jari dan biodata yang sesuai
- Jika tidak ditemukan, maka akan dilakukan pencocokan ulang, tetapi dengan metode similaritas. Similaritas ini dihitung dengan metode LCS. Algoritma ini akan menghitung persentase panjang substring terpanjang yang terdapat pada representasi ASCII dari gambar yang diupload dengan representasi ASCII pada gambar di basis data.
- Namun, jika similaritas yang didapat berada dibawah 70 persen, maka program tidak akan menampilkan apapun karena dianggap tidak ada yang cocok ataupun mirip.

2. Proses Penyelesaian Masalah dengan algoritma KMP dan BM

2.1. Algoritma KMP

Pra-pemrosesan Pola:

- Buat tabel Longest Prefix Suffix (LPS) yang menunjukkan panjang prefiks yang sama dengan sufiks terpanjang untuk setiap posisi dalam pola.
- Ini memungkinkan pola untuk digeser dengan efisien saat terjadi ketidakcocokan tanpa perlu membandingkan ulang karakter yang sudah cocok.

Pencarian dalam Teks:

- Mulai dari awal teks dan pola.
- Bandingkan karakter dalam teks dengan pola dari kiri ke kanan.
- Jika karakter cocok, lanjutkan ke karakter berikutnya.
- Jika ada ketidakcocokan:
 - Gunakan tabel LPS untuk menentukan posisi baru dalam pola tanpa mengulang perbandingan karakter yang sudah cocok.
 - Jika seluruh pola cocok dengan bagian dari teks, catat posisi tersebut sebagai tempat kecocokan ditemukan.

- Geser pola sesuai dengan tabel LPS dan lanjutkan pencarian sampai akhir teks.

2.2. Algoritma BM

Pra-pemrosesan Pola:

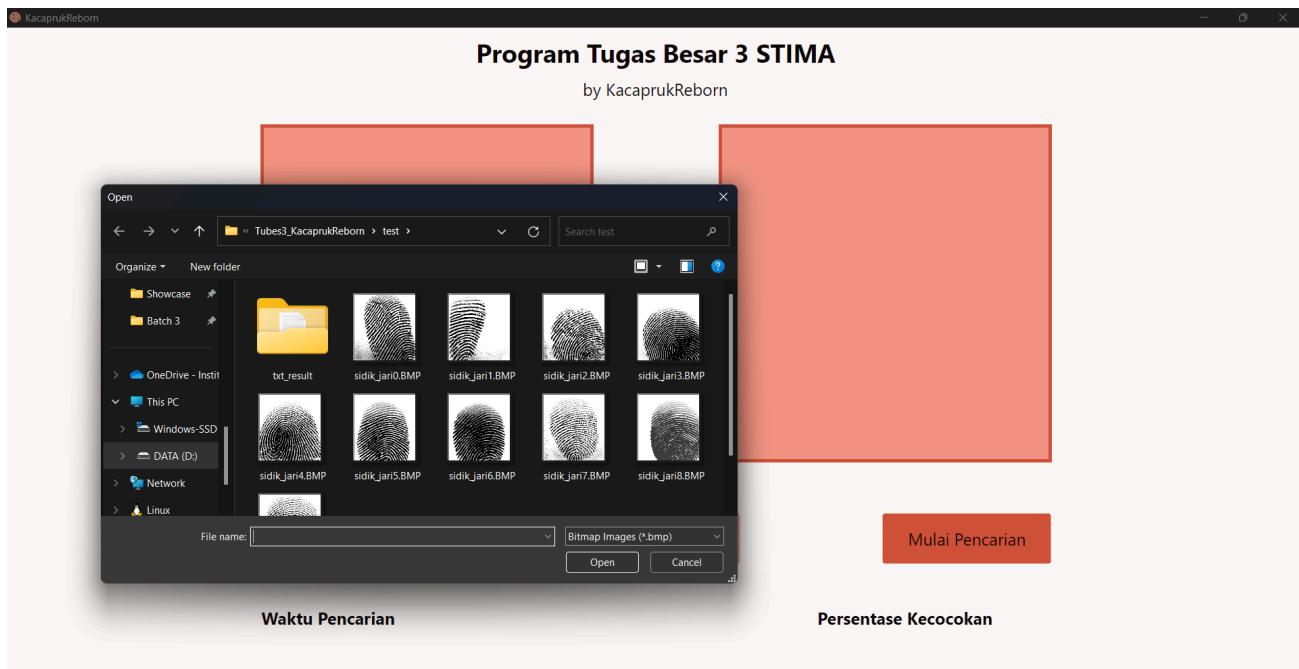
- Buat tabel untuk aturan karakter buruk (bad character table) yang menyimpan posisi terakhir kemunculan setiap karakter dalam pola.
- Jika terjadi ketidakcocokan karakter saat pencarian, tabel ini digunakan untuk menentukan seberapa jauh pola dapat digeser.

Pencarian dalam Teks:

- Mulai dengan menyelaraskan pola ke awal teks.
- Bandingkan pola dengan teks dari kanan ke kiri.
- Jika ada ketidakcocokan, gunakan tabel bad character dan good suffix untuk menentukan seberapa jauh pola harus digeser.
- Ulangi langkah ini sampai pola cocok dengan bagian dari teks atau seluruh teks telah diproses.
- Jika pola cocok sepenuhnya, catat posisi tersebut sebagai tempat kecocokan ditemukan.
- Geser pola sesuai dengan aturan yang ditentukan dan lanjutkan pencarian sampai akhir teks.

3. Fitur fungsional dan arsitektur aplikasi web yang dibangun

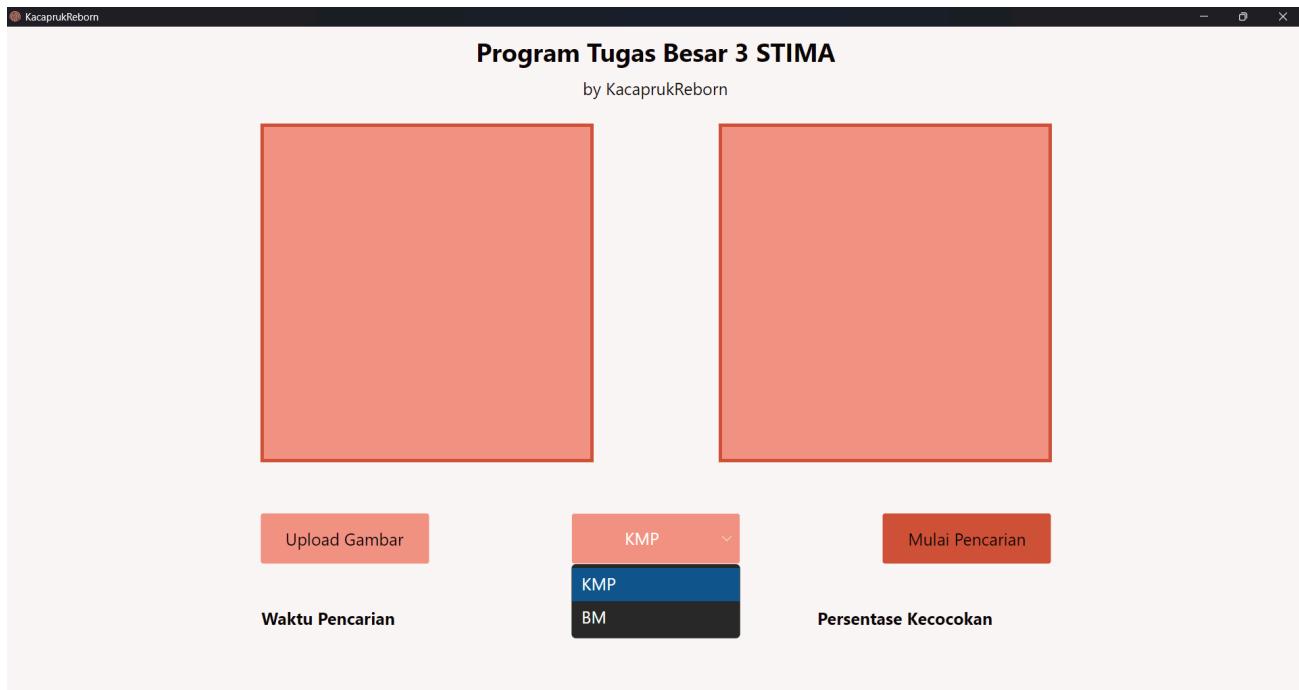
3.1. Button Upload Gambar



Gambar 3.1 Button Upload Gambar

Kami menyediakan button “Upload Gambar” agar pengguna dapat memilih gambar yang ingin dicocokkan. Setelah pengguna menekan tombol ini, akan dikeluarkan *pop up window* baru yang berfungsi khusus dalam pemilihan gambar.

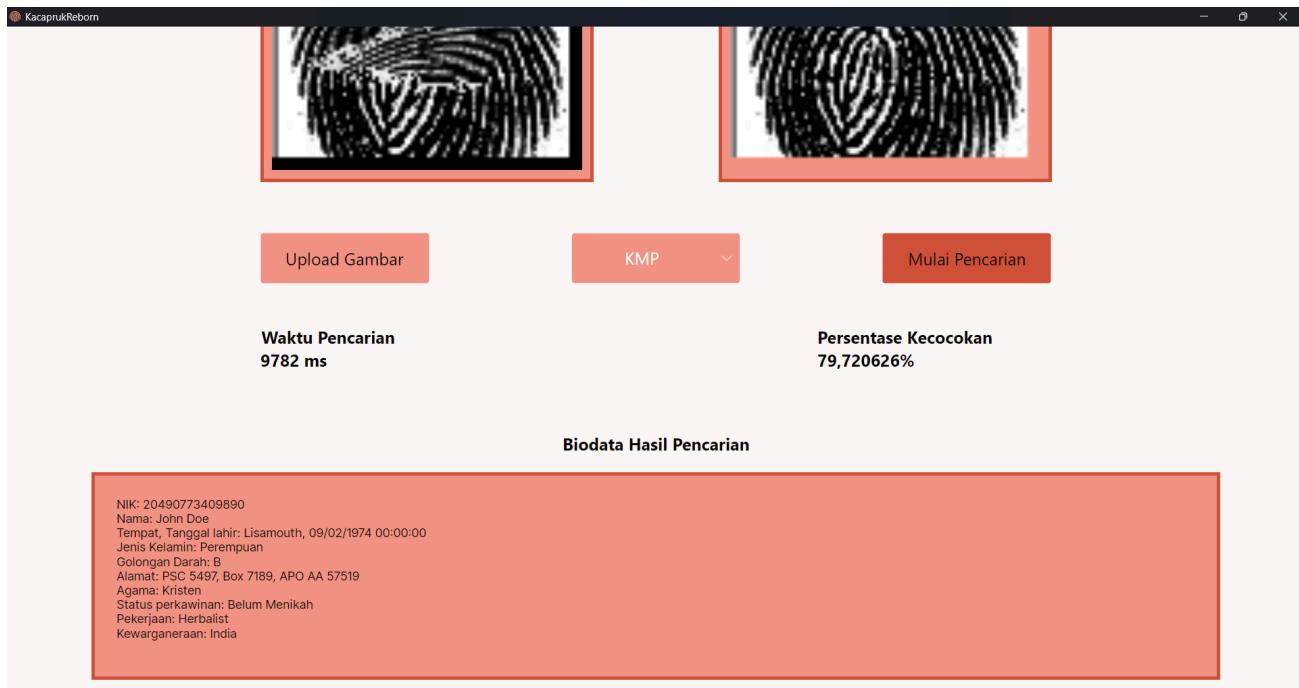
3.2. Select Dropdown



Gambar 3.2 Fitur Select Dropdown

Kami menyediakan fitur pemilihan algoritma berbasis *dropdown* untuk memudahkan pengguna.

3.3. Tampilan Informasi Waktu Eksekusi, Persentase Kecocokan, dan Biodata



Gambar 3.3 Tampilan Waktu Eksekusi, Persentase Kecocokan, dan Biodata

Kami menyediakan fitur untuk menampilkan informasi waktu eksekusi, persentase kecocokan, dan hasil pencarian biodata.

4. Contoh Ilustrasi Kasus

Untuk

BAB IV

IMPLEMENTASI DAN PENGUJIAN

1. Spesifikasi Teknis Program

Pada pengimplementasian program, terdapat beberapa bagian inti penting yang menjadi dasar dari aplikasi pencarian sidik jari. Komponen ini bisa dibagi menjadi struktur data yang mendukung kerja aplikasi dan kelas-kelas yang melakukan eksekusi program. Pada bagian ini, akan dibahas implementasi mendetail terkait aplikasi KacaprukReborn.

1.1. Struktur Data

1.1.1. Dictionary (Map)

Struktur data map/dictionary digunakan ketika kita ingin menyimpan data berupa pasangan hash dari suatu key ke value. Penggunaan struktur data ini terdapat pada dua kasus. Pertama sebagai penampung pasangan angka-huruf pasangan bahasa alay. Kedua, untuk menyimpan ASCII dari gambar pada database. Kumpulan karakter ASCII ini akan dipasangkan dengan pasangan nama dan berkas citra dari gambar.

```
Dictionary<char, char> pasangan = new Dictionary<char, char>();
```

```
var imageAsciiMap = new Dictionary<string?, (string, string?)>();
```

1.1.2. Array

Struktur data array digunakan ketika program membutuhkan penyimpanan kontigu yang linear. Penggunaan array pada program ini sangat mendominasi karena cukup banyak digunakan. Beberapa contoh penggunaan array di program ini adalah sebagai berikut. Digunakan untuk menyimpan bad char, dan compute LPS

```
int[] lps = ComputeLPSArray(pattern);
```

```
int[] badchar = new int[NO_OF_CHARS];
```

Array juga bisa direpresentasikan dengan tipe list seperti,

```
List<string> alays = new List<string>();
```

```
List<Biodata> biodatas = dh.GetBiodatas();
```

1.1.3. Matriks

Matriks digunakan ketika program membutuhkan penyimpanan yang berdimensi lebih dari satu. Penggunaan struktur data ini terdapat pada beberapa kasus seperti berikut

```
int[,] dp = new int[m + 1, n + 1];
```

```
6 references
private byte[][] subkeys = new byte[ROUND_NUM + 1][];
```

1.2. Daftar Kelas

Nama Kelas	Attribut	Method	Guna Method
KMP	-	ComputeLPSArray	Fungsi untuk membuat array LPS (Longest Proper Prefix which is also Suffix)
		KMPSearch	Menghasilkan true jika terdapat kecocokan pattern
		FindPatternInTexts	Menghasilkan nama asli orang yang memiliki kecocokan ASCII
BM	NO_OF_CHARS	Max	Menghasilkan nilai maksimum dari dua integer.
		BadCharHeuristic	Menyimpan data bad char (last occurrences)
		BMSearch	Menghasilkan true jika terdapat kecocokan pattern
		FindPatternInTexts	Menghasilkan nama asli orang yang memiliki kecocokan ASCII

ImageProcessor	-	ConvertBitmapToAscii	Mengonversi sebuah gambar menjadi teks ascii dan menghasilkan teks ascii sebagai sebuah string
		BinaryToAscii	Mengonversi binary menjadi teks ascii (string)
		ToGrayscale	Mengubah bitmap menjadi grayscale dan mengembalikan nilai intensitas
		GetPatternFromAscii	Mengambil pattern ASCII di baris tengah
		ProcessImagesToAscii	Process all images in the database to ASCII
DatabaseHelper	connectionString, connection	GetBiodatas	Get all biodata from the database
		GetSidikJaris	Get all fingerprint data from the database
		GetBiodataFromAlay	Mendapatkan sebuah biodata yang namanya telah dicocokan dengan nama yang asli
		GetNamaFromAlay	Mendapatkan seluruh nama alay dari tabel biodata
		varcharToText	Mengubah tipedata varchar ke text
		UpdateBiodata	Melakukan update tabel biodata ke server sql
		UpdateSidikJari	Melakukan update tabel sidik jari ke server sql

Biodata	Nik, Nama, Tempat_lahir, Tanggal_lahir, Jenis_kelamin, Golongan_darah, Alamat, Agama, Status_perkawinan, Pekerjaan, Kearganegaraan	showInfo	Menampilkan semua value biodata
SidikJari	Berkas_citra, Nama	-	-
Converter	Algorithm	EncryptBiodata	proses mengenkripsi biodata
		DecryptBiodata	proses dekripsi biodata
		EncryptDatabase	proses dekripsi database
AES128	BLOCK_SIZE, KEY_SIZE, ROUND_NUM, sBox, inverseSBox, roundCoefficient, key, subkeys, Key, Subkeys, BYTE_POLY_REDUCTION, POLY_REDUCTION	keywordTransform	Proses transformasi blok teks sesuai dengan S-BOX byte substitution table
		keyTransform	Memperbarui kunci utama (key) untuk setiap putaran enkripsi atau dekripsi.
		shiftRows	Menggeser baris-baris dalam blok teks 4x4 (16 byte) sesuai dengan aturan operasi ShiftRows dalam AES.
		inverseShiftRows	Kebalikan dari proses shiftRows.
		galoisMult2	Melakukan perkalian dalam Galois Field GF(2^8) dengan nilai

			2.
		galoisDefaultMult	Fungsi ini melakukan perkalian dalam Galois Field $GF(2^8)$ untuk berbagai nilai mult.
		mixColumn	Melakukan tahapan mix column dengan mengalikan tiap column dengan matriks rijndael's galois field
		inverseMixColumn	Melakukan invers dari proses mix column
		encryptBlock	Proses mengenkripsi text menjadi ciphertext pada level blok byte
		decryptBlock	Proses dekripsi ciphertext menjadi text pada level blok byte
		Encrypt	Proses antara untuk memvalidasi segala kondisi untuk melakukan enkripsi pada level blok byte
		Decrypt	Proses antara untuk memvalidasi segala kondisi untuk melakukan dekripsi pada level blok byte
EncryptionHelper	machine	PrintByteArray	untuk print array byte (buat debug)
		TrimSpace	untuk menghapus spasi di belakang karakter paling akhir

	AddSpace	menambah spasi setelah karakter paling akhir bila diperlukan
	encryption	proses yang menjembatani proses enkripsi
	decryption	Proses yang menjembatani proses dekripsi
ConvertAlay	parseAngka	Proses untuk mengkonversi angka ke huruf pada bahasa alay
	validasi	Proses untuk melakukan validasi apakah kata alay memiliki makna sama dengan kata asli yang kita punya dengan cara menangani semua kombinasi bahasa alay.
	findAlayMatch	Proses mencari kata asli dari kata alay yang kita punya
Similarity	LcsLength	Fungsi untuk menghitung panjang LCS (Longest Common Subsequence)
	SimilarityScore	Fungsi untuk menghitung skor similaritas antara dua string
	FindBestMatch	Fungsi untuk menemukan pasangan dengan similaritas tertinggi, candidates isinya ascii

			representation dan valuanya path dan namanya
--	--	--	--

2. Tata Cara Penggunaan Program

2.1. Menjalankan Program

Setelah menginstall .NET SDK dan berhasil menjalankan server sql, kompilasi program dapat dilakukan dengan mengikuti langkah-langkah berikut: (Untuk lebih lengkapnya, silahkan lihat README [disini](#))

- Clone repository lalu masuk ke dalam directory nya
- Lakukan konfigurasi database terlebih dahulu
- Pastikan file database(.sql) yang anda inginkan telah berada di direktori "src/Database/"
- Masuk ke server sql anda
- Buat database baru di server sql anda lalu keluar dari server sql
- Lakukan restore file sql ke dalam server sql anda
- Database sudah tertanam di server sql Anda!
- Pastikan pada connection di file DatabaseHelper sudah sesuai dengan konfigurasi akun sql Anda
- Jika Anda ingin meningkatkan keamanan data biodata pada database, Anda dapat mengaktifkannya pada file program.cs.
- Program siap dijalankan dengan perintah berikut

```
cd "src"
```

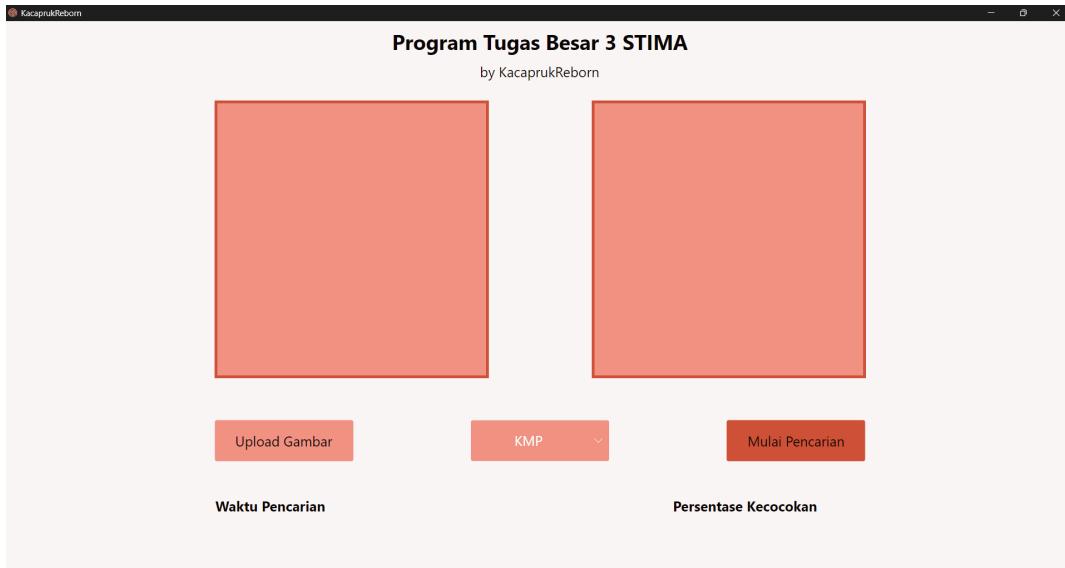
```
dotnet run
```

2.2. Menggunakan Program

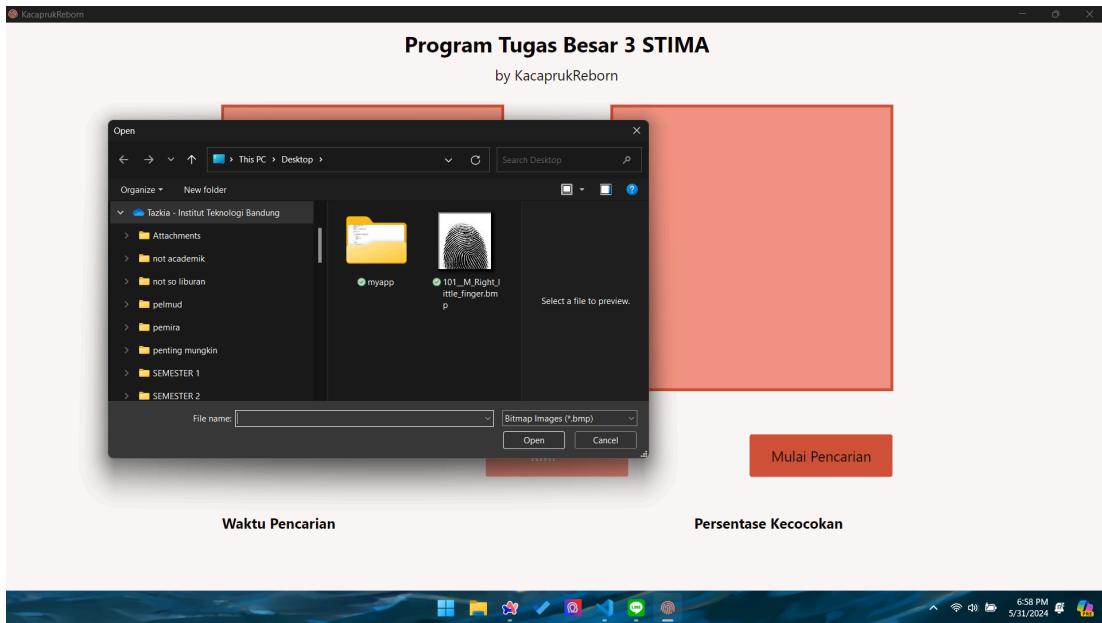
Secara garis besar, ada dua tahapan dalam penggunaan program, yaitu saat pengguna memasukkan input untuk pencarian, dan saat program menampilkan hasil.

- Tahap memberikan Input

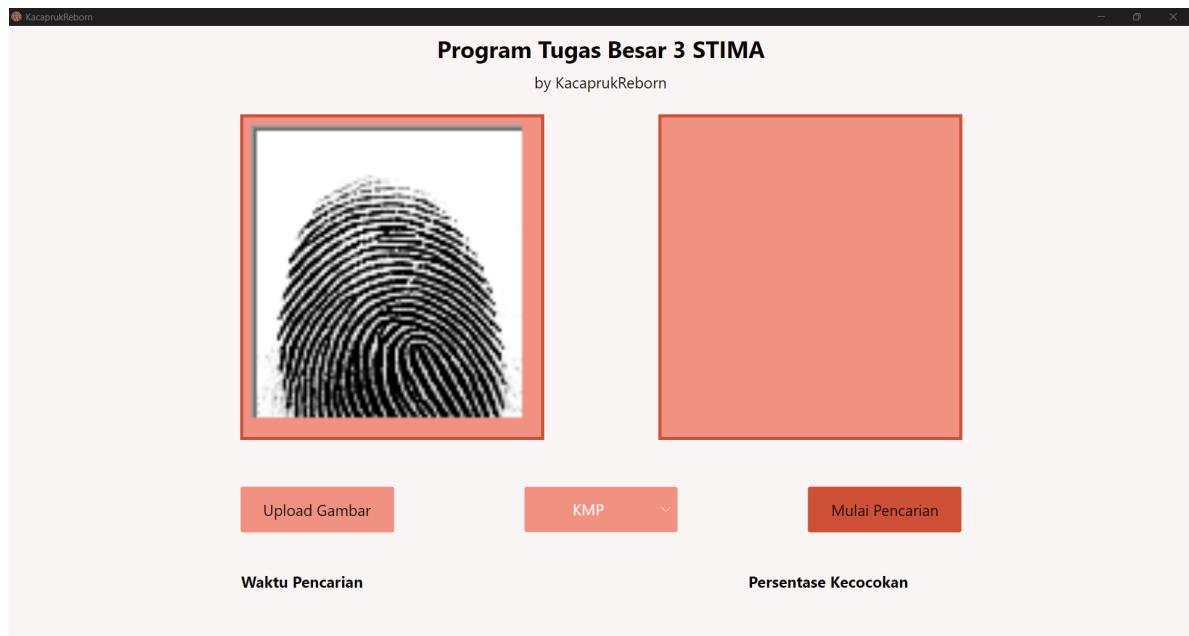
Setelah program dijalankan, maka program akan ditampilkan pada layar. Berikut adalah tangkapan layar ketika program baru dijalankan.



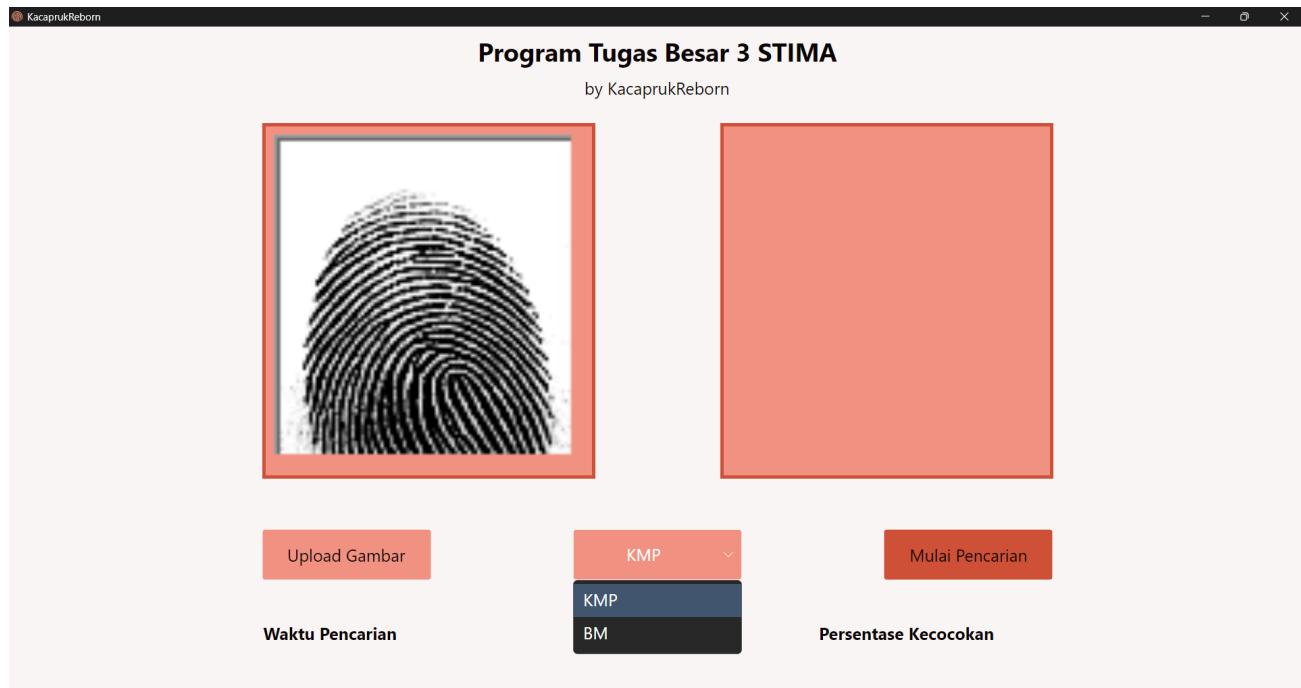
Pada bagian tengah kiri, terdapat tombol dengan tulisan “Upload Gambar”, tekan tombol tersebut untuk membuka overlay untuk melakukan penguploadan gambar. Berikut adalah tampilan dari overlay ketika akan mengupload gambar.



Pilih gambar sidik jari yang akan diupload ke program KacaprukReborn. Overlay akan membatasi gambar yang diupload hanya gambar dengan ekstensi .bmp saja. Setelah gambar terupload, maka gambar akan ditampilkan pada panel sebelah kiri, berikut adalah contohnya.



Sebelum melanjutkan ke tahap pencarian, pengguna dapat memilih algoritma yang akan digunakan. Pemilihan dilakukan dengan membuka ComboBox yang terdapat di tengah tampilan program. Secara default, algoritma KMP akan terpilih. Berikut adalah tampilan ComboBox saat menampilkan pilihan algoritma, antara algoritma KMP dan algoritma BM.

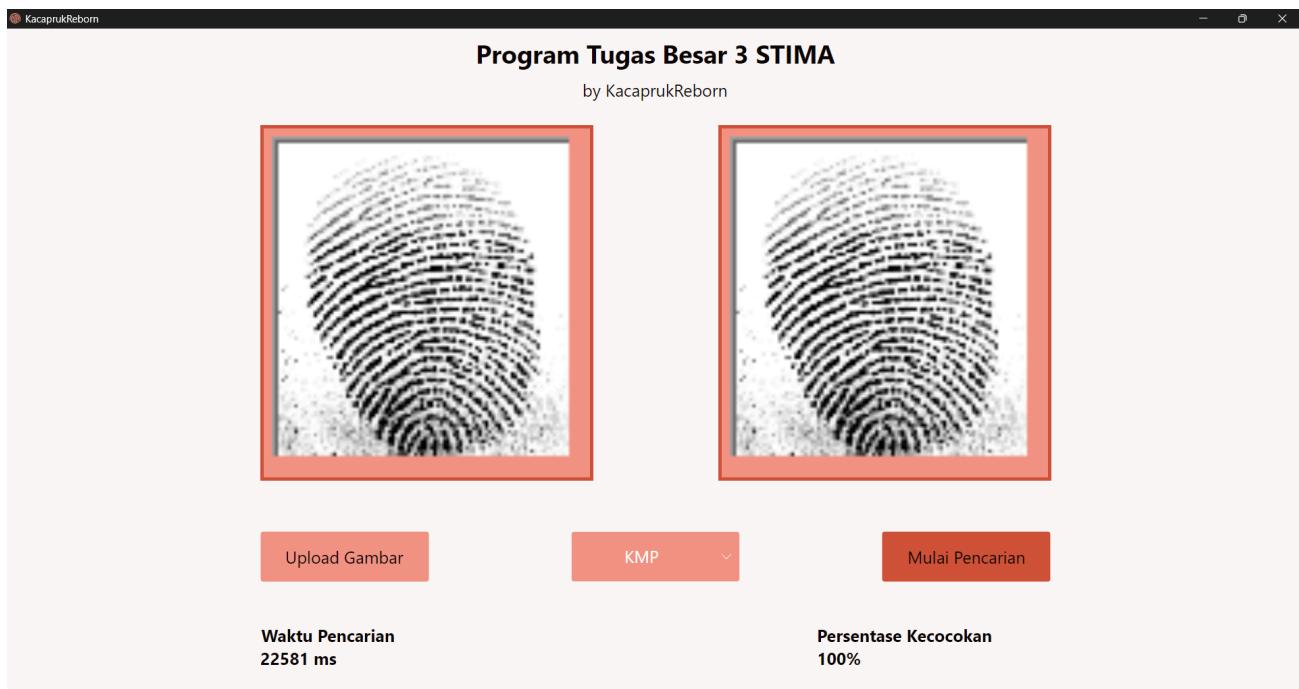


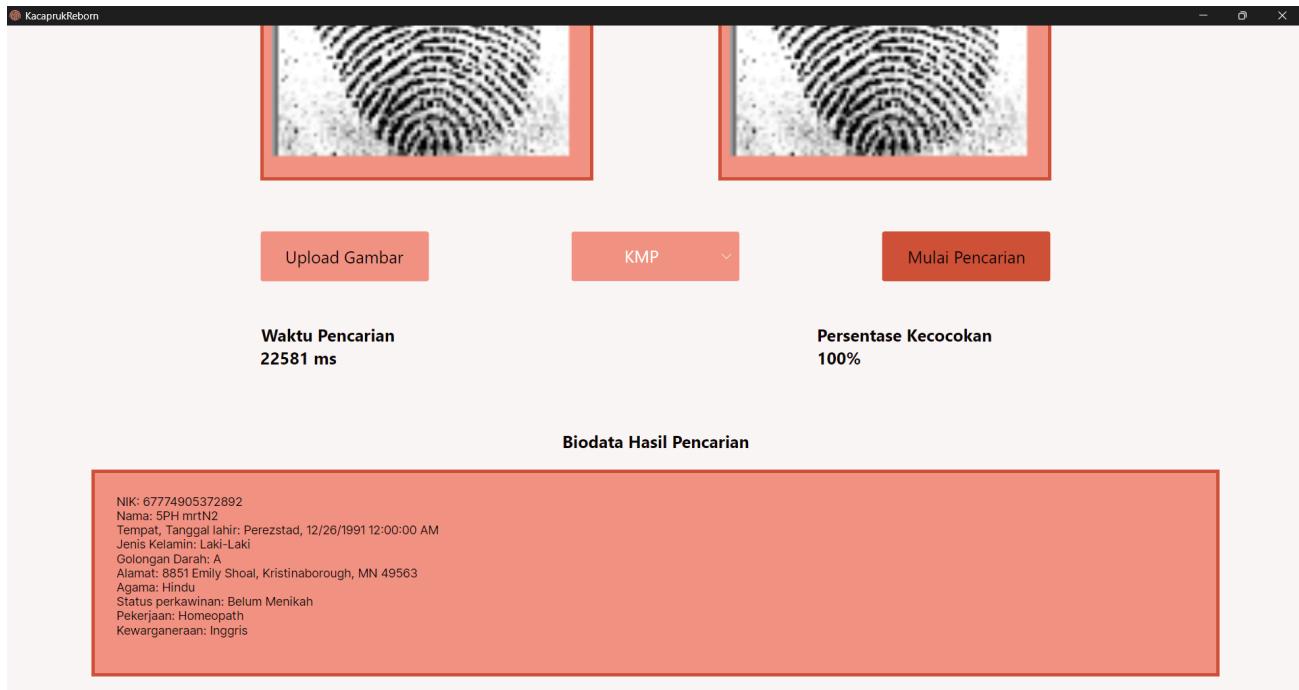
Setelah memilih algoritma, maka pencarian dapat dimulai dengan cara menekan tombol di kanan yang bertuliskan “Mulai Pencarian”.

- Tahap menampilkan Output

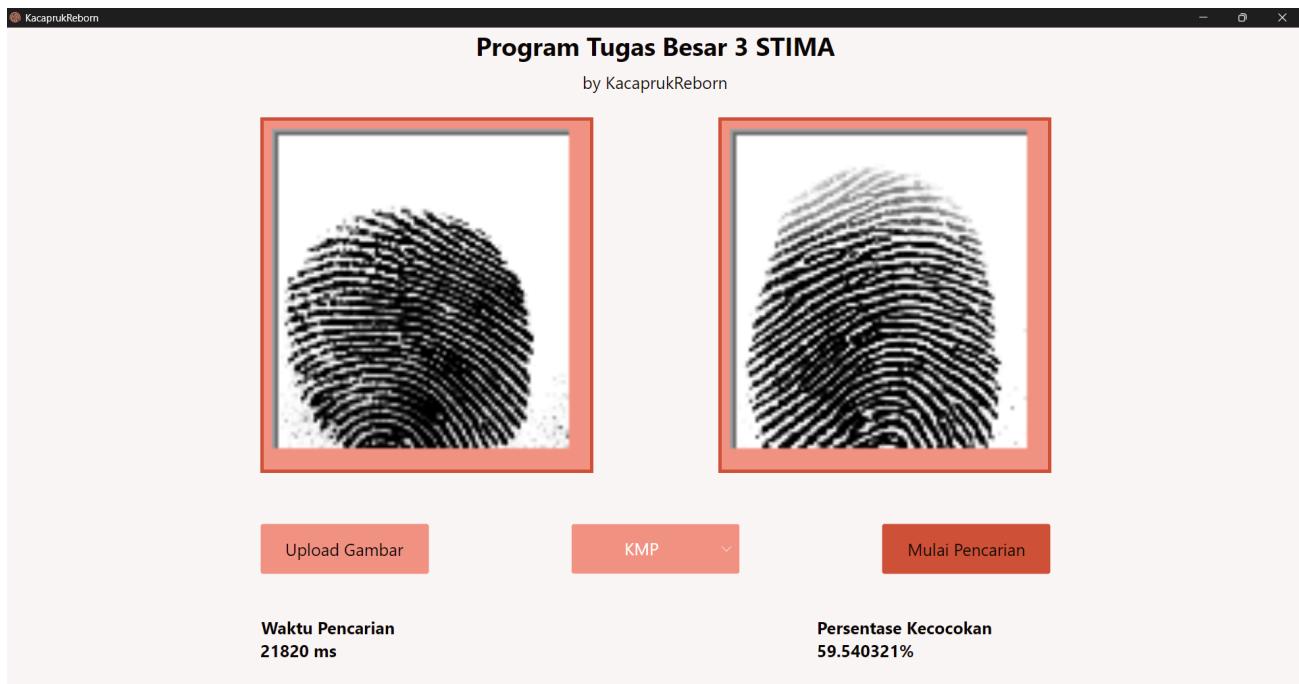
Setelah pencarian selesai, maka program akan menampilkan sidik jari yang dari database dan menampilkan biodata dari pemilik sidik jari tersebut. Terdapat 3 kemungkinan hasil yang akan ditampilkan, yaitu:

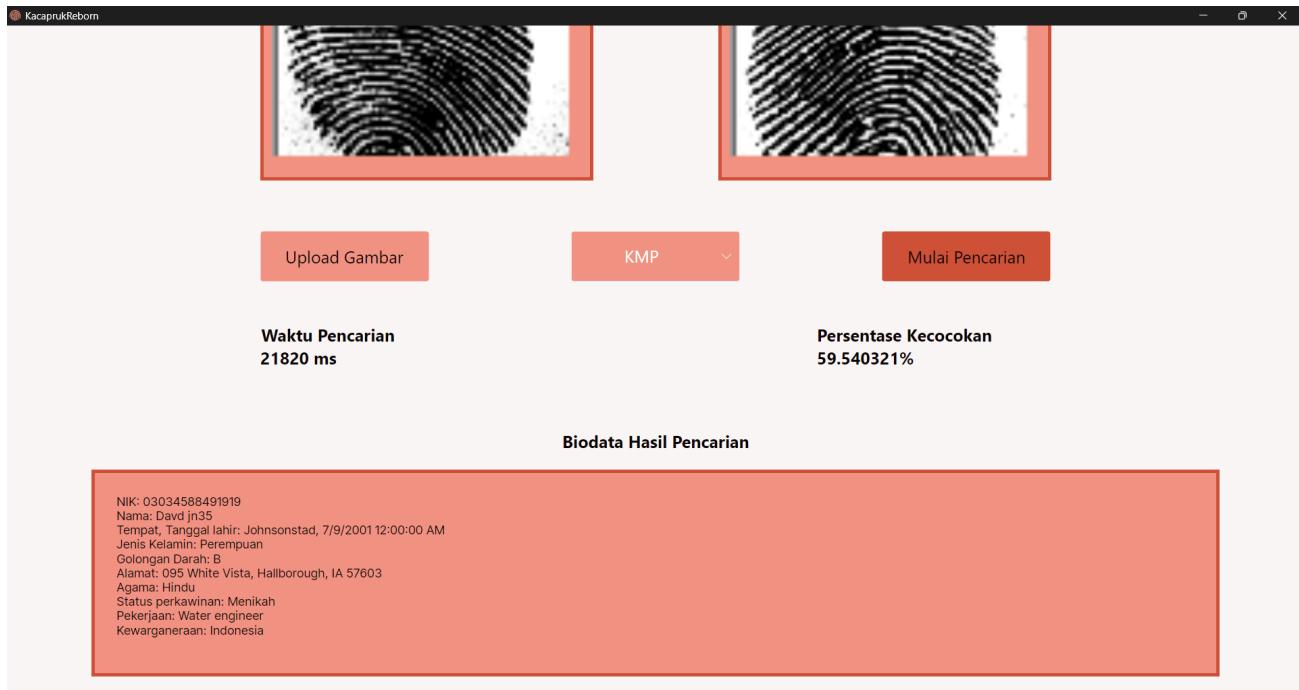
1. Jika gambar sidik jari ditemukan pada basis data dengan algoritma KMP atau BM, maka gambar tersebut akan ditampilkan pada layar dengan persentase kecocokan 100%.



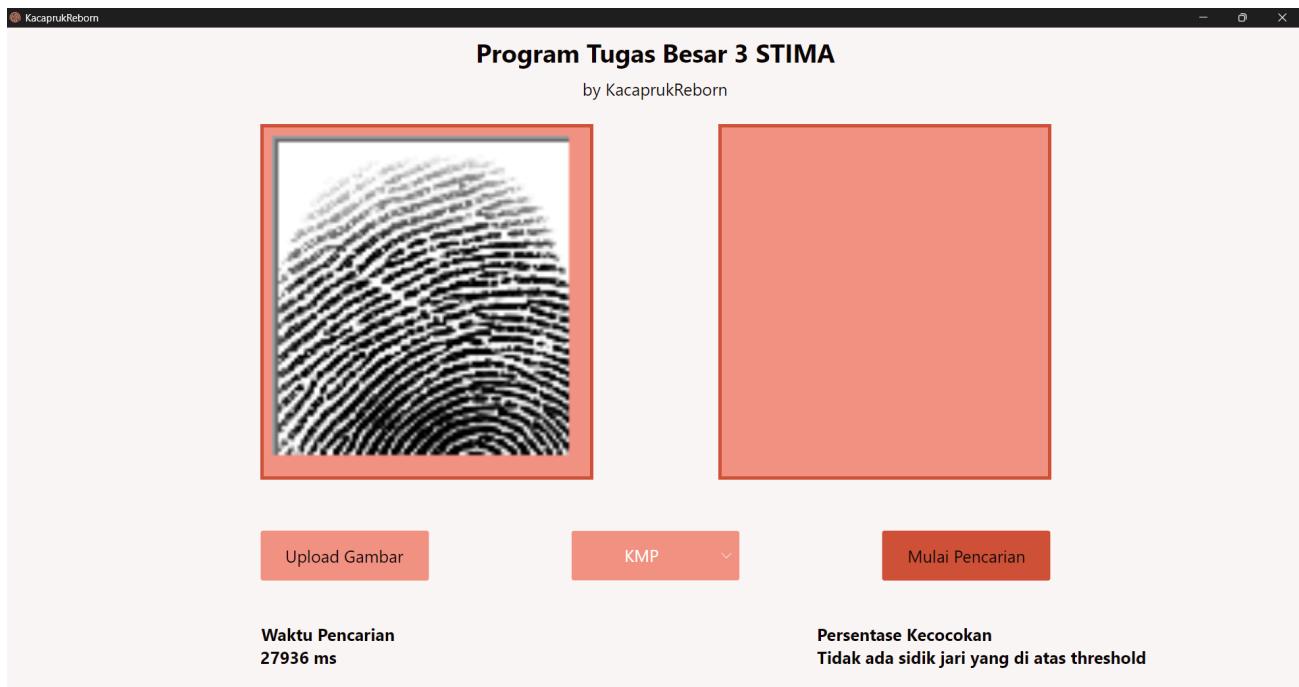


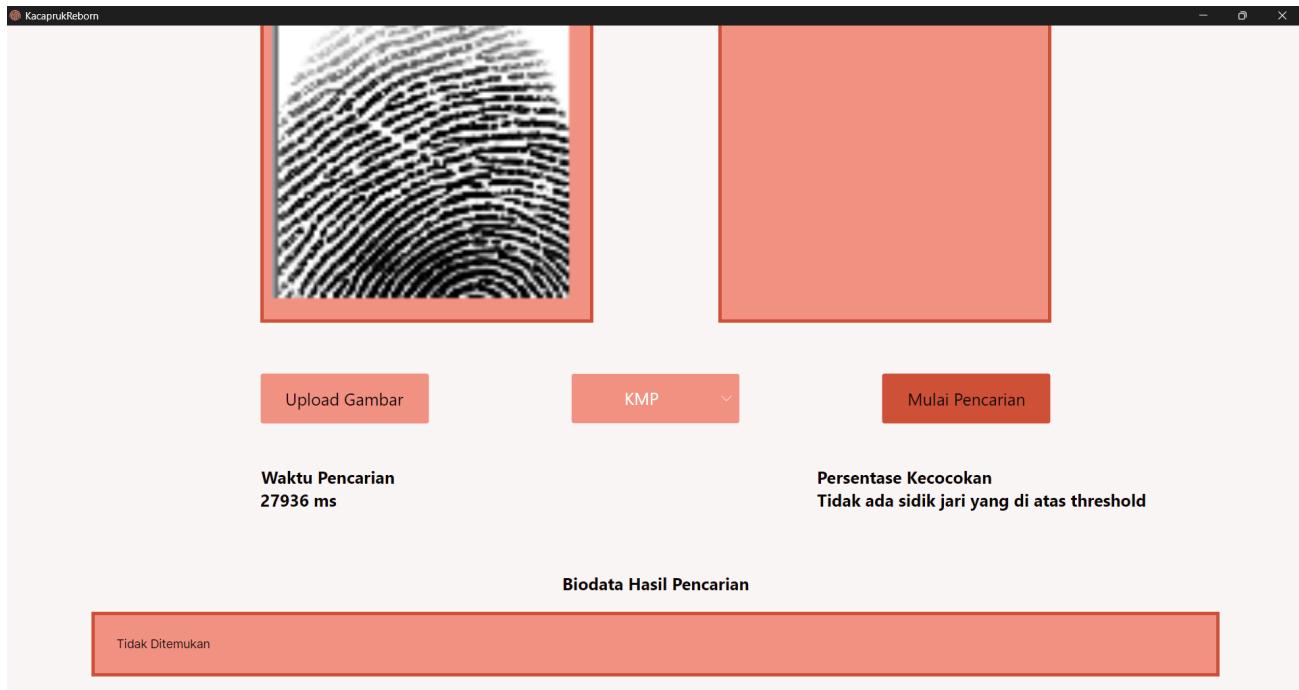
2. Jika gambar sidik jari tidak ada, maka yang akan ditampilkan adalah gambar dengan tingkat similaritas tertinggi yang melewati suatu batasan 70 persen





3. Jika tidak ada gambar yang memiliki tingkat similaritas yang melewati batasan, maka akan ditampilkan pesan bahwa tidak ada sidik jari yang cocok.



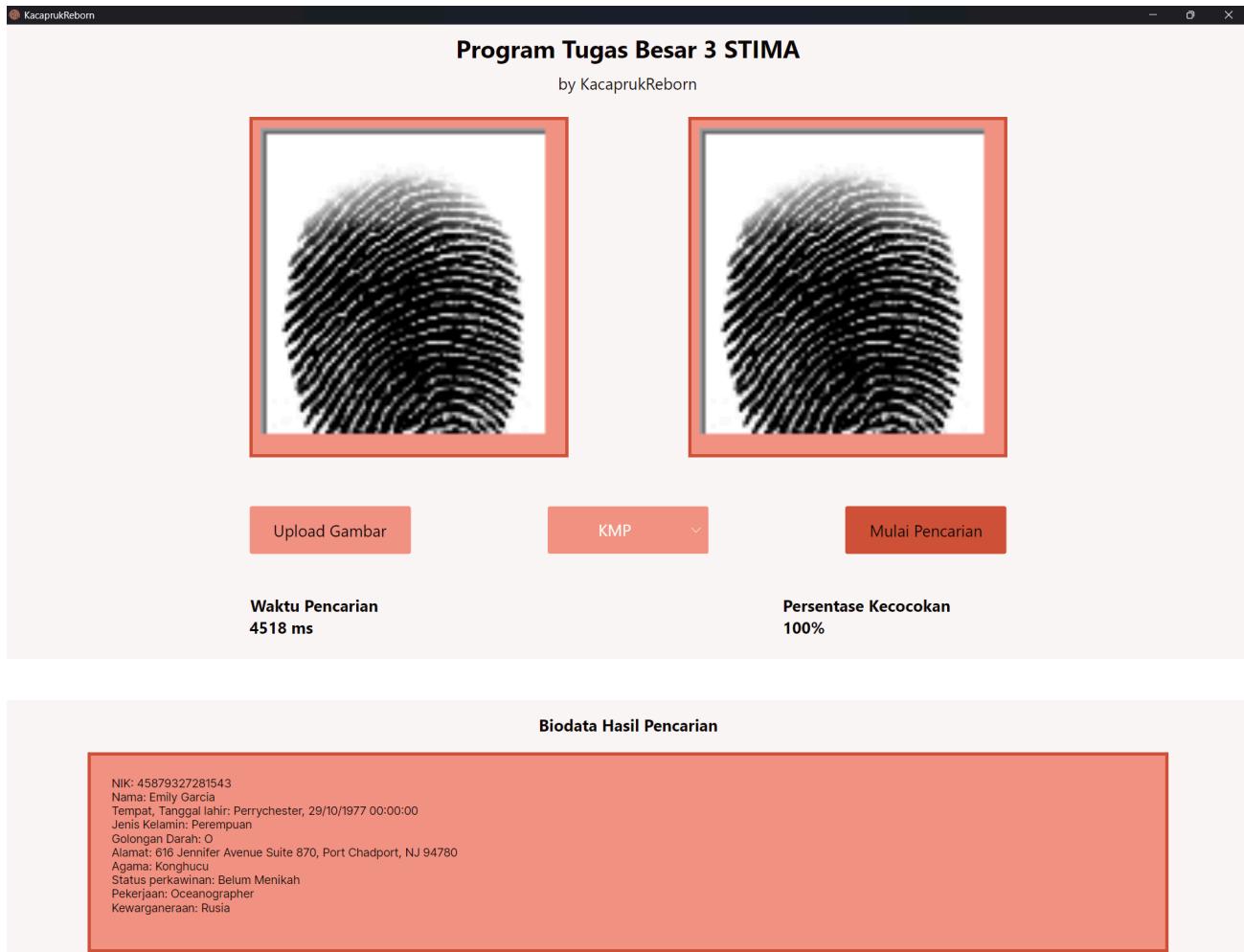


3. Hasil Pengujian

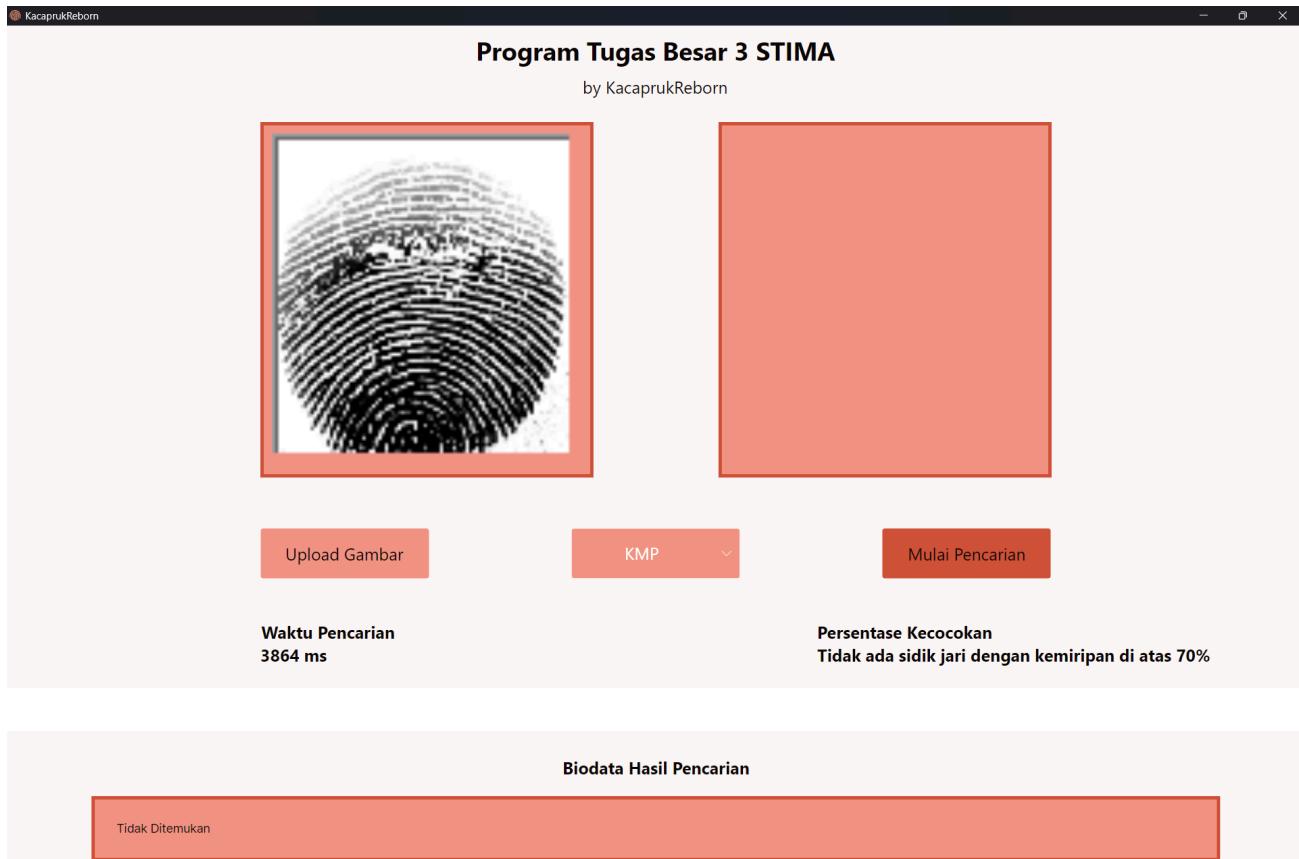
Pengujian akan mencakup pengujian beberapa masukan untuk memastikan bahwa program berjalan dengan baik, dan pengujian untuk membandingkan antara KMP dan BM.

3.1. KMP

1. Pengujian KMP dengan gambar yang terdapat pada Database.
 - File Pattern : sidik_jari6.bmp (di folder test repository)
 - Database : Beberapa sidik jari yang diambil dari Kaggle Asisten
 - Algoritma : KMP



2. Pengujian KMP dengan gambar yang tidak terdapat pada Database.
 - File Pattern : 1__M_Left_ring_finger.bmp (dari Kaggle Asisten)
 - Database : Beberapa sidik jari yang diambil dari Kaggle Asisten
 - Algoritma : KMP



3. Pengujian KMP dengan gambar yang Altered pada Database.
- File Pattern : 1__M_Left_index_finger_CR.bmp
 - Database : Beberapa sidik jari yang diambil dari Kaggle Asisten
 - Algoritma : KMP

KacaprukReborn

Program Tugas Besar 3 STIMA

by KacaprukReborn



Upload Gambar

KMP

Mulai Pencarian

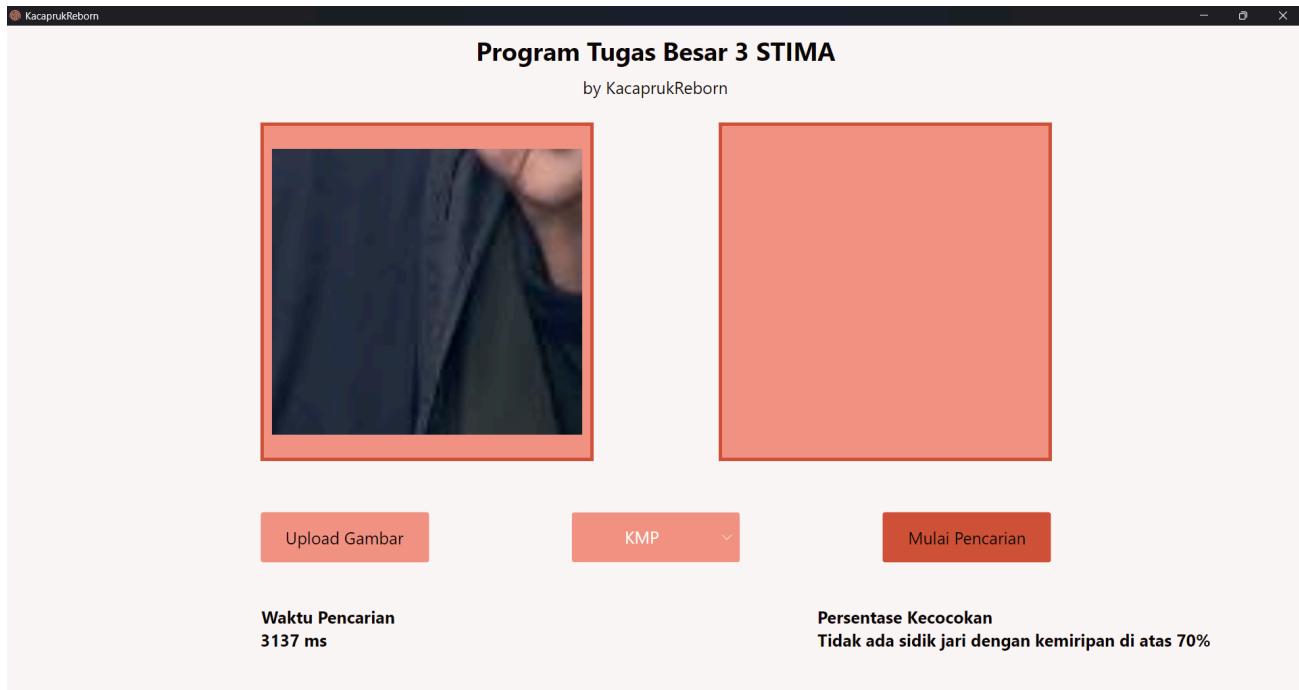
Waktu Pencarian
4747 ms

Persentase Kecocokan
72,231028%

Biodata Hasil Pencarian

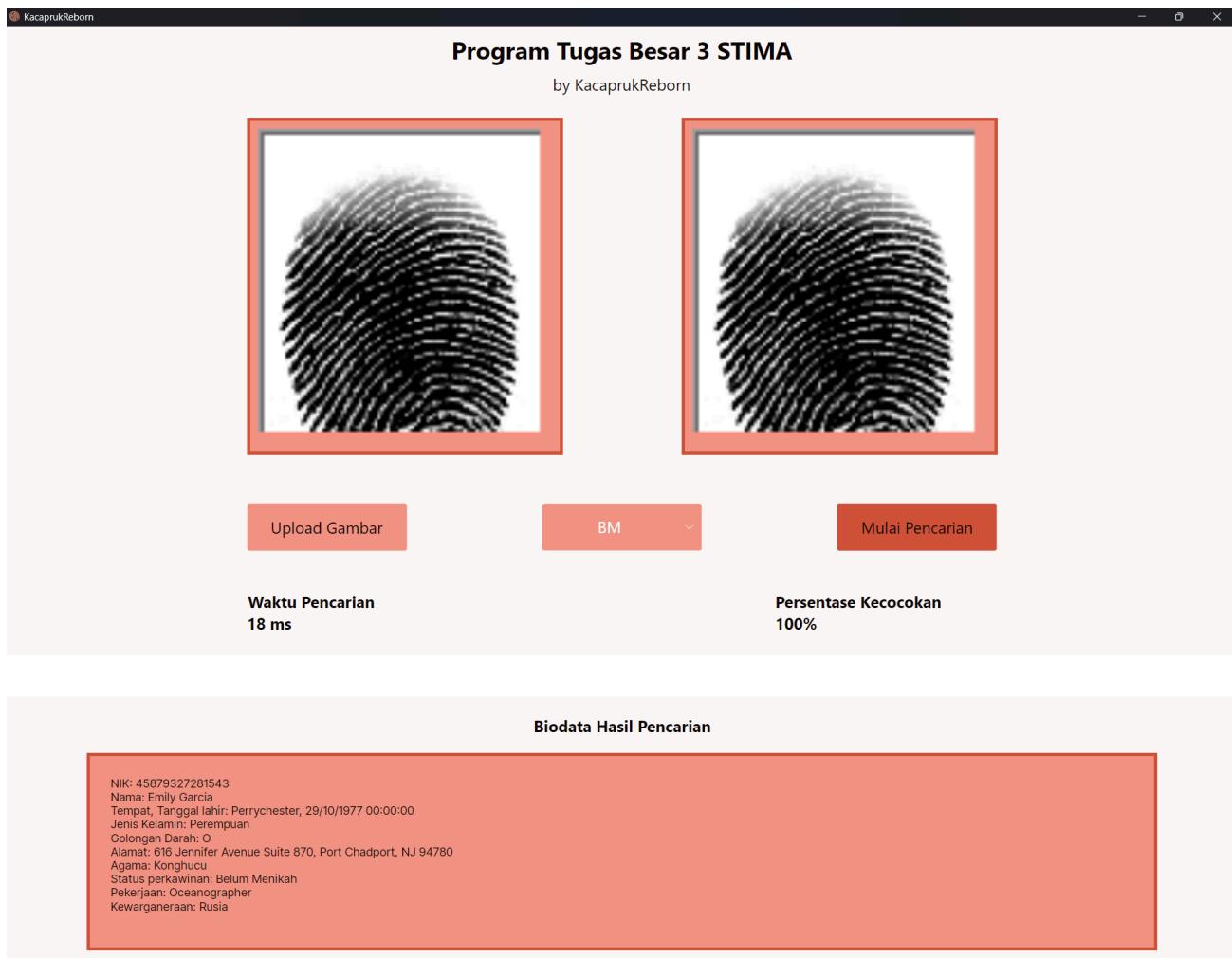
NIK: 20490773409890
Nama: John Doe
Tempat, Tanggal lahir: Lisamouth, 09/02/1974 00:00:00
Jenis Kelamin: Perempuan
Golongan Darah: B
Alamat: PSC 5497, Box 7189, APO AA 57519
Agama: Kristen
Status perkawinan: Belum Menikah
Pekerjaan: Herbalist
Kewarganegaraan: India

4. Pengujian KMP dengan gambar yang terdapat pada Database.
 - File Pattern : warna_cropped.bmp
 - Database : Beberapa sidik jari yang diambil dari Kaggle Asisten
 - Algoritma : KMP

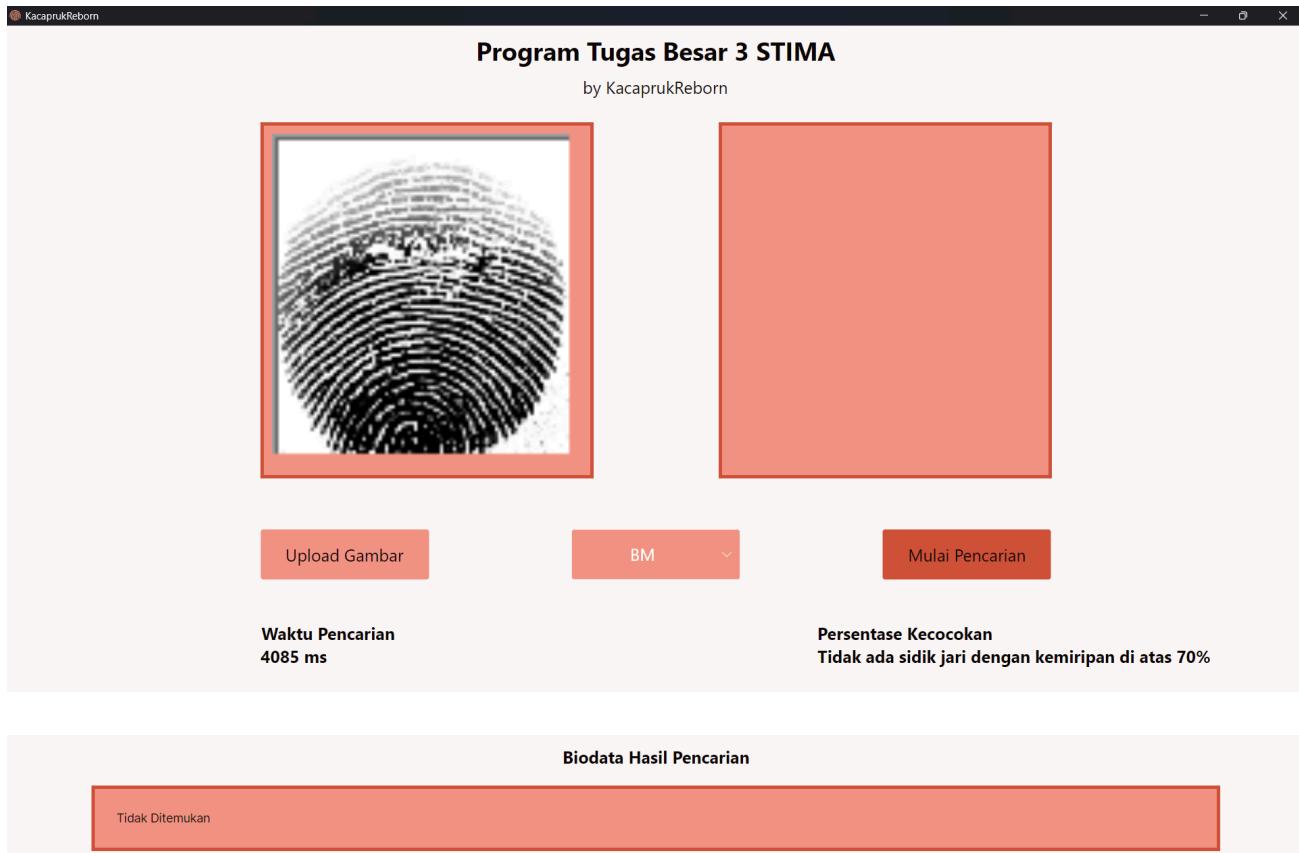


3.2. BM

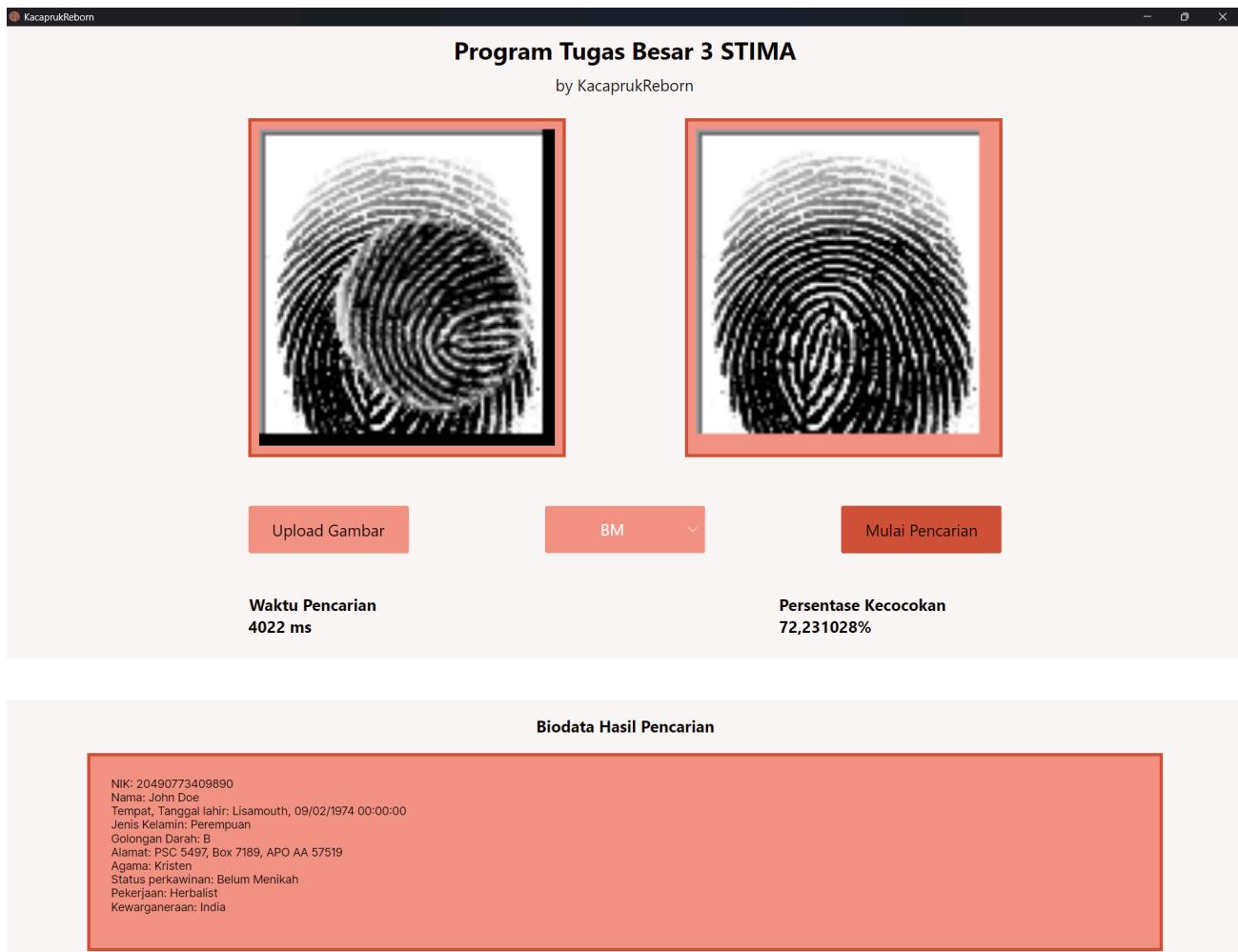
1. Pengujian BM dengan gambar yang terdapat pada Database.
 - File Pattern : sidik_jari6.bmp (di folder test repository)
 - Database : Beberapa sidik jari yang diambil dari Kaggle Asisten
 - Algoritma : BM



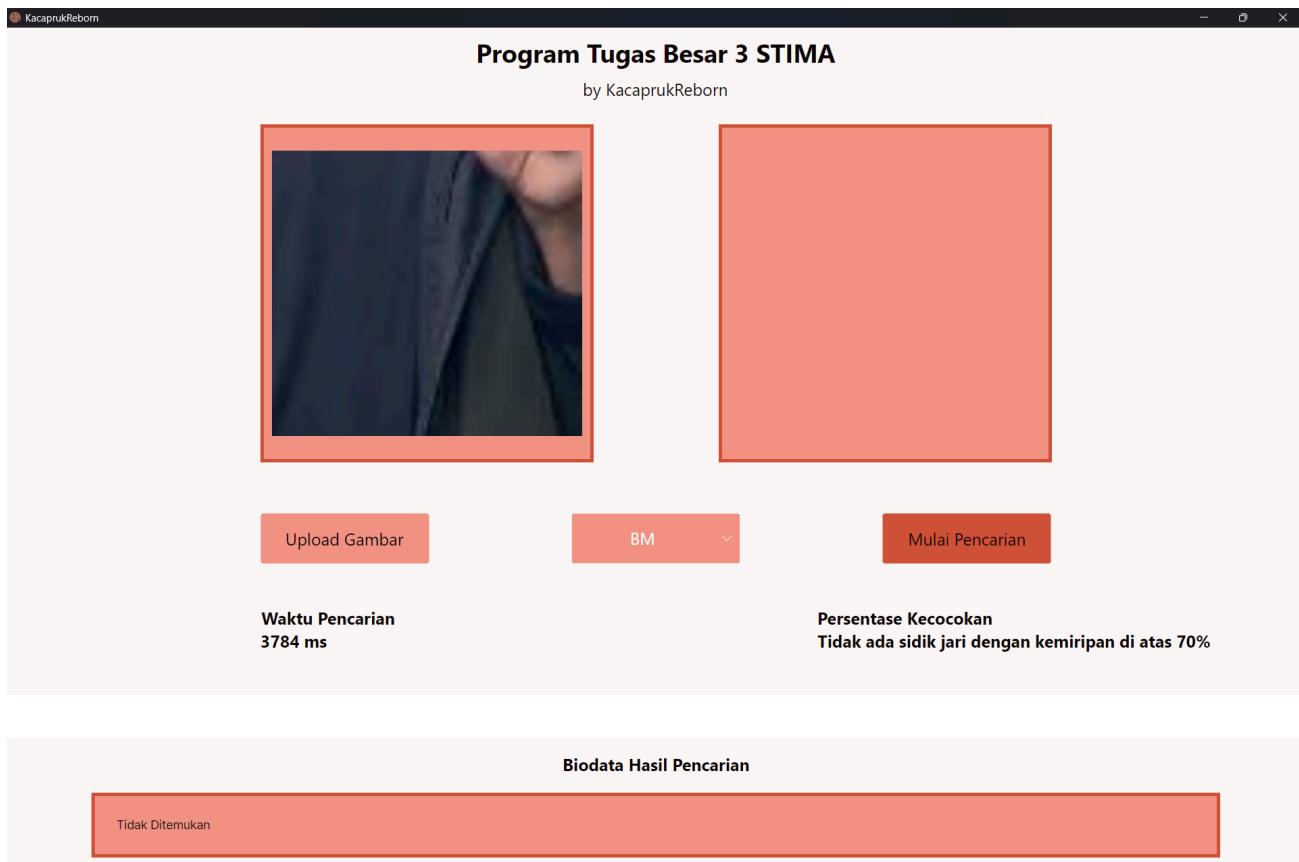
2. Pengujian BM dengan gambar yang terdapat pada Database.
 - File Pattern : 1__M_Left_ring_finger.bmp (dari Kaggle Asisten)
 - Database : Beberapa sidik jari yang diambil dari Kaggle Asisten
 - Algoritma : BM



3. Pengujian BM dengan gambar yang terdapat pada Database.
 - File Pattern : 1__M_Left_index_finger_CR.bmp
 - Database : Beberapa sidik jari yang diambil dari Kaggle Asisten
 - Algoritma : BM



4. Pengujian BM dengan gambar yang terdapat pada Database.
 - File Pattern : warna_cropped.bmp
 - Database : Beberapa sidik jari yang diambil dari Kaggle Asisten
 - Algoritma : BM



4. Analisis Hasil Pengujian

Dari hasil pengujian pada bagian 3, dapat terlihat bahwa algoritma BM memiliki rata-rata waktu eksekusi yang lebih cepat daripada algoritma KMP. Secara teori, algoritma BM memang memiliki keuntungan dan dapat bekerja lebih cepat ketika terdapat cukup banyak alfabet pada teks, dalam hal ini adalah representasi sidik jari dalam bentuk ASCII 8 bit. Pada program ini, sidik jari yang diubah menjadi representasinya menjadi ASCII akan memiliki kemungkinan alfabet yang muncul sangat banyak, perlu diingat bahwa dari 8 bit saja sudah ada 2^8 atau 256 kemungkinan karakter ASCII 8 bit yang muncul.

Algoritma Boyer-Moore (BM) memang sebagai salah satu algoritma pencarian teks yang paling efisien dalam praktik karena dua strategi utama: bad character heuristic dan good suffix heuristic. Bad character heuristic memungkinkan algoritma untuk melompati sejumlah karakter saat

pencocokan karakter tidak sesuai, dengan demikian mengurangi jumlah perbandingan yang diperlukan. Good suffix heuristic memungkinkan lompatan yang lebih panjang berdasarkan pola karakter yang cocok sebagian. Keunggulan ini terutama terlihat dalam teks dengan panjang yang besar dan set karakter yang luas, seperti pada representasi ASCII dari sidik jari.

Sementara itu, algoritma Knuth-Morris-Pratt (KMP) memiliki kelebihan dalam pencarian pola yang lebih deterministik, tanpa perlu kembali pada karakter sebelumnya dalam teks. KMP menggunakan tabel lompatan LPS (Longest Prefix Suffix) yang mengidentifikasi posisi potensial dari pencocokan berikutnya tanpa mengulang perbandingan karakter yang telah dibandingkan sebelumnya. Namun, meskipun efisien dalam kasus pola dan teks dengan set karakter yang terbatas, KMP seringkali tidak dapat mengimbangi kecepatan lompatan besar yang dimungkinkan oleh heuristik BM dalam kasus teks dengan alfabet yang luas.

Dalam konteks aplikasi pencocokan sidik jari, setiap sidik jari direpresentasikan sebagai serangkaian karakter ASCII yang unik. Variabilitas yang tinggi dari karakter-karakter ini membuat algoritma BM lebih efisien dibandingkan KMP. Selain itu, karakteristik sidik jari yang umumnya memiliki pola yang kompleks dan tidak berulang secara jelas juga menguntungkan BM yang dapat melewatkannya lebih banyak karakter selama pencocokan.

Namun demikian, perlu juga dicatat bahwa performa aktual dari kedua algoritma ini sangat bergantung pada implementasi spesifik dan kondisi data. Misalnya, dalam kasus dimana pola memiliki banyak pengulangan, KMP mungkin mendekati atau bahkan melampaui BM dalam beberapa skenario. Oleh karena itu, dalam praktik nyata, terdapat kemungkinan bahwa algoritma KMP memiliki waktu eksekusi yang lebih cepat daripada BM, karena itu penggunaan algoritma KMP juga layak untuk dicoba.

Secara keseluruhan, hasil pengujian menunjukkan bahwa BM memiliki keunggulan yang konsisten dalam konteks pencocokan sidik jari berbasis ASCII 8 bit dalam aplikasi ini, memberikan waktu eksekusi yang lebih cepat dan efisiensi yang lebih tinggi dibandingkan KMP. Keuntungan ini memperkuat pilihan algoritma BM sebagai solusi yang lebih cocok untuk masalah pencocokan pola dalam domain sidik jari yang menghasilkan representasi ASCII 8 bit yang kompleks dan bervariasi.

BAB V

PENUTUP

Kesimpulan

Dari program ini, dapat disimpulkan bahwa pencocokan sidik jari dapat dilakukan dengan menerapkan konsep string matching dengan beberapa kemungkinan algoritma, yaitu algoritma KMP dan BM. Pada pengujinya, didapatkan bahwa algoritma BM lebih cocok untuk digunakan dalam proses pencocokan sidik jari yang diubah representasinya menjadi ASCII 8 bit daripada algoritma KMP. Hal ini terjadi karena variasi dari karakter yang muncul setelah sidik jari diubah representasinya menjadi ASCII sangat banyak (256 kemungkinan karakter).

Saran

Sebaiknya untuk pengembangan aplikasi dengan bahasa C# menggunakan visual studio agar lebih memudahkan di awal. Kemudian ditambahkan batasan yaitu pengembangan aplikasi desktop dengan C# ini menggunakan Visual Studio menjadi kudu mesti harus wajib!

Refleksi

Pada tugas besar ini kami mendapatkan pengalaman dalam membuat desktop app menggunakan bahasa C# dengan avalonia. Hal ini membuat kami sadar bahwa C# tidak terlalu bagus untuk digunakan dalam desktop app development karena dokumentasi avalonia yang terbatas membuat kami kesulitan dalam mengelola style dari GUI. Selain itu, tidak adanya error message yang ditampilkan di terminal membuat kami kesulitan dalam mencari bug, apalagi kami juga tidak bisa melakukan print pada line tertentu. Oleh karena itu, kami melakukan debug manual dengan cara mengganti print dengan aksi berupa penulisan pada suatu file txt.

LAMPIRAN

Link repository : https://github.com/rifchzsckki/Tubes3_KacaprukReborn

Link Video : <https://youtu.be/w1oih1EhRMc>

DAFTAR PUSTAKA

5 Common Encryption Algorithms and the Unbreakables of the Future, Arcserve, diakses pada tanggal 1 Juni 2024 pada laman

<https://www.arcserve.com/blog/5-common-encryption-algorithms-and-unbreakables-future>

AES Rijndael Cipher explained as a Flash animation, AppliedGo, diakses pada tanggal 1 Juni 2024 pada laman <https://www.youtube.com/watch?v=gP4PqVGudtg&t=114s>

AES Explained (Advanced Encryption Standard), Computerphile, diakses pada tanggal 1 Juni 2024 pada laman <https://www.youtube.com/watch?v=O4xNJsjtN6E>

Booyer Moore Algorithm for Pattern Searching, geeksforgeeks.org, diakses pada tanggal 29 Mei 2024 pada laman

<https://www.geeksforgeeks.org/boyer-moore-algorithm-for-pattern-searching/>

Booyer Moore Algorithm for Pattern Searching, geeksforgeeks.org, diakses pada tanggal 29 Mei 2024 pada laman

<https://www.geeksforgeeks.org/kmp-algorithm-for-pattern-searching/>

Longest Common Subsequence (LCS), geeksforgeeks.org, diakses pada tanggal 8 Juni 2024 pada laman <https://www.geeksforgeeks.org/longest-common-subsequence-dp-4/>

Munir, Rinaldi 2021, Pencocokan string 2021, diakses pada tanggal 29 Mei 2024 pada laman

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmlk/2020-2021/Pencocokan-string-2021.pdf>