

**LAPORAN TUGAS BESAR III IF2211 STRATEGI ALGORITMA
SEMESTER II 2023/2024**

**PEMANFAATAN PATTERN MATCHING DALAM MEMBANGUN
SISTEM DETEKSI INDIVIDU BERBASIS BIOMETRIK MELALUI
CITRA SIDIK JARI**

Kelompok 17 / let-me-seedik

Ibrahim Ihsan Rasyid	13522018
Panji Sri Kuncara Wisma	13522028
Imam Hanif Mulyarahman	13522030



**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG 2024**

DAFTAR ISI

LAPORAN TUGAS BESAR III IF2211 STRATEGI ALGORITMA	1
DAFTAR ISI	2
BAB I	3
DESKRIPSI MASALAH	3
BAB II	4
LANDASAN TEORI	4
A. Dasar-Dasar Pattern Matching	4
B. Algoritma Knuth-Morris-Pratt	4
C. Algoritma Boyer-Moore	4
D. Regular Expression	5
E. Teknik Pengukuran Persentase Kemiripan	6
F. Pengembangan Aplikasi Desktop dalam Bahasa C# Menggunakan .NET Framework	6
1. Integrated Development Environment (IDE)	7
2. Framework	7
BAB III	8
ANALISIS PEMECAHAN MASALAH	8
A. Langkah Pemecahan Masalah	8
B. Proses Pencarian Solusi dengan Algoritma KMP dan BM	9
C. Proses Pencocokan Nama pada Biodata Menggunakan Regex	11
D. Fitur Fungsional dan Arsitektur Aplikasi Desktop yang Dibangun	11
E. Contoh Ilustrasi Kasus	12
BAB IV	13
IMPLEMENTASI DAN PENGUJIAN	13
A. Spesifikasi Teknis Program	13
B. Tata Cara Penggunaan Program	23
C. Hasil Pengujian	23
a. Pengujian Menggunakan KMP	23
b. Pengujian Menggunakan BM	25
c. Pengujian Menggunakan Hamming Distance	27
D. Analisis Hasil Pengujian	29
BAB V	31
KESIMPULAN DAN SARAN	31
A. Kesimpulan	31
B. Saran	31
C. Tanggapan dan Refleksi	31
LAMPIRAN	32

BAB I

DESKRIPSI MASALAH



Gambar 1.1 Visualisasi Persoalan pada Tugas

Sumber:

<https://www.harianhaluan.com/news/1012054257/inilah-dokslidalam-video-viral-walikota-makassar-bisa-dipakai-di-berbagai-acara-peresmian?page=2>

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 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, kami 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.

BAB II

LANDASAN TEORI

A. Dasar-Dasar *Pattern Matching*

Pattern Matching merupakan pencocokan sebuah *pattern* berupa *string* dengan panjang m karakter pada sebuah teks berupa *string* dengan panjang n karakter untuk mencari lokasi pertama pattern tersebut ditemukan di dalam teks (diasumsikan $m << n$). Pattern matching diaplikasikan di banyak bidang, seperti search engine hingga bioinformatika

Dalam pattern matching, misalkan terdapat sebuah string $s = x_1x_2x_3\dots x_n$ yang memiliki panjang n dengan x_i adalah karakter pada indeks i dalam string tersebut. Prefix pada string s merupakan substring $s[1..k]$, dan suffix pada string s merupakan substring $s[k..n]$ dengan k adalah indeks antara 1 sampai n

B. Algoritma Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt (KMP) merupakan salah satu algoritma pattern matching. Pencocokan yang dilakukan oleh algoritma ini adalah pencocokan eksak sehingga algoritma ini tidak dapat melakukan pencarian untuk menemukan kemiripan. Sesuai namanya, algoritma KMP ditemukan oleh Donald Knuth, James H. Morris dan Vaughan Pratt pada tahun 1977.

Pada dasarnya, algoritma KMP menyerupai algoritma brute force dalam melakukan pencarian. Namun, algoritma KMP melakukan pergeseran pattern dengan lebih cerdas dibandingkan dengan brute force. Apabila ketidakcocokan antara teks dan pattern P ditemukan pada $P[j]$, algoritma KMP akan menggeser pattern hingga prefix terpanjang dari $P[0..j-1]$ yang merupakan suffix dari $P[1..j-1]$. Pergeseran tersebut ditentukan oleh *border function* $b(k)$ dengan k adalah indeks pada pattern sebelum terjadi ketidakcocokan.

Kompleksitas algoritma KMP ditentukan dari penjumlahan kompleksitas border function dan kompleksitas pencarian pada teks. Kompleksitas border function adalah $O(m)$, sedangkan kompleksitas pencarian pada teks adalah $O(n)$. Jadi, kompleksitas algoritma KMP adalah $O(m+n)$.

Keuntungan dari KMP adalah algoritmanya tidak perlu bergerak mundur dalam pencariannya. Di sisi lain, kekurangan dari KMP adalah ketika alfabet sangat besar, sehingga kemungkinan terjadi ketidakcocokan di awal teks semakin besar.

C. Algoritma Boyer-Moore

Algoritma Boyer-Moore (BM) adalah salah satu algoritma pattern matching selain KMP. Algoritma ini berjalan berdasarkan dua teknik : *looking-glass technique* yang artinya mencari pattern P secara mundur (dari indeks paling besar pada pattern) pada teks

T dan *character-jump technique* yaitu metode penggeseran pattern dengan pola atau urutan tertentu.

Secara detail, *character-jump technique* dilakukan dengan urutan kemungkinan berikut. Misalkan ketidakcocokan antara *pattern* P dan teks T terjadi pada $P[j]$ dan $T[i]$, dengan $T[i]$ adalah x . Pertama, jika terdapat x pada P , maka P digeser supaya *last occurrence* (akan dibahas lebih lanjut) dari x pada P sejajar dengan $T[i]$. Kedua, jika terdapat x pada P pada suatu indeks tertentu tetapi pergeseran ke *last occurrence* dari x tidak dapat dilakukan, maka P digeser satu karakter. Ketiga, apabila kedua kondisi sebelumnya tidak dicapai, maka P digeser supaya $P[0]$ sejajar dengan $T[i+1]$.

Algoritma BM melakukan preprocess terhadap pattern P dan alfabet A untuk membuat *last occurrence function* $L(a)$. Last occurrence function memetakan setiap karakter pada A menjadi integer dengan $L(a)$ adalah indeks terbesar i sehingga $P[i] = a$, atau -1 apabila a tidak ada dalam P dengan a adalah anggota dari A . Umumnya, hasil dari *last occurrence function* masing-masing karakter disimpan dalam sebuah tabel.

D. Regular Expression

Regular Expression (Regex) adalah serangkaian karakter yang mendefinisikan pola pencarian teks. Dalam pemrograman dan pengolahan teks, regex digunakan untuk menemukan, mengganti, atau memanipulasi substring dalam teks berdasarkan pola yang ditentukan. Pola ini dapat mencakup karakter literal, karakter khusus, dan operator, yang bersama-sama memungkinkan pencocokan kompleks dan fleksibel terhadap teks.

Berikut merupakan notasi dasar dari regex

Notasi	Arti	Contoh pattern beserta teks yang sesuai
[]	Disjungsi	/[aiveo]s/ → <u>astaga</u>
[s-s]	Disjungsi terhadap range	/[A-Z]/ → bapak <u>Panji</u>
.	Suatu karakter apapun	/si.it/ → <u>sigit</u> , <u>sikit</u>
^	Negasi	/[^Aa]/ → AAaaaaargh
?	Zero or one character of the preceding	/cl?an/ → <u>can</u> , <u>clan</u>
*	Zero or more character of the preceding	/bus*/ → <u>bu</u> , <u>buss</u> , <u>bus</u>
+	One or more character of the preceding	/she+p/ → <u>shep</u> , <u>sheeeeeep</u> , <u>shp</u>

Tabel 2.1 Notasi Dasar Regex

Dalam beberapa bahasa pemrograman, tersedia library untuk mengaplikasikan regex dengan menerima sebuah pola regex dan teks yang ingin dicocokkan, dan selanjutnya mengembalikan hasilnya apakah cocok atau tidak.

E. Teknik Pengukuran Persentase Kemiripan

Terdapat beberapa algoritma yang bisa digunakan untuk mencari kemiripan dari dua buah string, diantaranya ada algoritma Hamming Distance, algoritma Levenshtein Distance, dan algoritma Longest Common Subsequence (LCS). Pada tugas kali ini dipilih algoritma Hamming Distance untuk membandingkan kedua gambar tersebut. Algoritma tersebut dipilih karena sama sederhana dan mudah diimplementasikan.

Hamming distance adalah sebuah metode yang digunakan untuk mengukur perbedaan antara dua string atau vektor yang memiliki panjang yang sama. Pengukuran ini dilakukan dengan menghitung jumlah posisi di mana simbol-simbol yang bersesuaian berbeda antara kedua string atau vektor tersebut. Jumlah perbedaan posisi yang dihitung digunakan untuk menghitung persentase kemiripan dari kedua string atau vektor tersebut.

Examples:

```
Input : str1[] = "geeksforgeeks", str2[] = "geeksandgeeks"  
Output : 3  
Explanation : The corresponding character mismatch are  
highlighted.  
"geeksforgeeks" and "geeksandgeeks"
```

Sumber : <https://www.geeksforgeeks.org/hamming-distance-two-strings/>

Pada contoh diatas dapat dilihat bahwa nilai dari hamming distance kedua string tersebut adalah 3 yang merepresentasikan perbedaan karakter dari kedua gambar tersebut pada sisi yang bersesuaian.

F. Pengembangan Aplikasi Desktop dalam Bahasa C# Menggunakan .NET Framework

Pada Tugas Besar ini digunakan Bahasa C# untuk mengimplementasikan program yang kami buat. C# (dibaca "C sharp") adalah bahasa pemrograman sederhana dan serbaguna yang dapat digunakan untuk berbagai keperluan, seperti pemrograman

server-side pada website, pengembangan aplikasi desktop dan mobile, serta pemrograman game. C# adalah bahasa pemrograman berorientasi objek yang mengadopsi konsep seperti inheritance, class, polymorphism, dan encapsulation.

C# sangat bergantung pada .NET Framework, sebuah framework yang digunakan untuk mengkompilasi dan menjalankan kode C#. Dikembangkan oleh Microsoft dengan merekrut Anders Hejlsberg, C# dirancang sebagai bahasa pemrograman utama dalam lingkungan .NET Framework.

1. Integrated Development Environment (IDE)

Dalam pembuatan tugas besar ini, IDE yang digunakan adalah Visual Studio. Visual Studio merupakan IDE yang ideal untuk pengembangan perangkat lunak dengan berbagai bahasa pemrograman C#. Kami memilih Visual Studio karena Visual Studio lebih unggul dalam hal fitur lengkap, integrasi yang kuat, dan alat-alat yang diperlukan untuk pengembangan aplikasi C# yang kompleks dan skala besar. Visual Studio juga menawarkan fitur yang memudahkan dalam membuat antarmuka pengguna.

2. Framework

Aplikasi Desktop yang dibuat untuk tugas besar ini memanfaatkan framework Windows Form dalam membuat Antarmuka Pengguna. Windows Forms, juga dikenal sebagai WinForms, adalah sebuah platform antarmuka pengguna grafis (GUI) yang disediakan oleh Microsoft untuk pengembangan aplikasi desktop pada sistem operasi Windows. WinForms adalah bagian dari .NET Framework, tetapi juga dapat digunakan dengan .NET Core dan .NET 5 atau yang lebih baru.

BAB III

ANALISIS PEMECAHAN MASALAH

A. Langkah Pemecahan Masalah

Berikut adalah langkah-langkah pemecahan masalah dalam sistem identifikasi individu berbasis biometrik melalui citra sidik jari:

1. Desain umum sistem

- Frontend

Frontend dari aplikasi ini menggunakan windows form untuk menampilkan antarmuka pengguna. Antarmuka akan menunggu input dari pengguna untuk melakukan operasi. Pada awalnya pengguna akan memilih gambar sidik jari yang mau dibandingkan dengan cara menekan tombol pilih citra. Lalu ada tombol KMP dan BM yang menunjukkan algoritma mana yang sedang digunakan untuk pencarian pola. Lalu pengguna akan menekan tombol search yang akan menjalankan fungsi perbandingan sesuai algoritma yang dipilih. Setelah itu gambar sidik jari yang cocok dan biodata pemiliknya akan ditampilkan pada antarmuka pengguna.

- Backend

Backend dari aplikasi ini menggunakan SQL sebagai basis data dan bahasa pemrograman C#. Basis data menyimpan dua buah tabel yaitu tabel biodata dan tabel sidik_jari. Kedua tabel tersebut tidak terhubung oleh relasi apapun. Tabel sidik_jari akan menyimpan *path* menuju gambar-gambar yang ada di dalam folder tes. Gambar-gambar tersebut berformat .bmp. Nama yang ada di tabel biodata adalah kata-kata yang bisa saja *corrupt* yang dalam tugas besar kali ini adalah kata-kata alay. Operasi-operasi yang berkaitan dengan basis data akan ditangani oleh satu kelas khusus.

2. *Preprocessing* gambar dataset

Gambar sidik jari yang dikumpulkan dari dataset dikonversi menjadi binary dan kemudian dikonversi menjadi ASCII. Oleh karena gambar-gambar sidik jari tersebut berformat *grayscale* (1 pixel = 8 bit), maka setiap pixel dari gambarlah yang akan diubah menjadi karakter ASCII yang berada pada rentang 0 sampai 255. Keputusan tersebut dilakukan dengan tujuan untuk menjaga akurasi dan detail dari setiap gambar.

3. *Preprocessing* gambar uji

Gambar uji adalah gambar sidik jari yang akan dicari tahu kepemilikannya dengan melakukan pencocokan pada gambar-gambar di dataset. Gambar uji ini juga di *preprocess* dengan cara yang sama dengan gambar dataset. Setelah berubah menjadi deretan karakter ASCII, akan diambil 30 karakter tengah sebagai

pattern untuk proses pencocokan. Oleh karena 1 karakter ASCII sama dengan 1 pixel pada kasus ini, maka sama saja seperti dengan mengambil 30 pixel sebagai *pattern*. Pengambilan *pattern* di tengah adalah untuk menghindari masalah gambar yang terpotong dan tidak mencerminkan informasi sidik jari.

4. Pencocokan

Pattern yang didapat pada langkah *preprocessing* gambar uji akan dikenakan algoritma pencocokan string (KMP atau BM) untuk melakukan pengecekan apakah *pattern* tersebut ada pada dataset yang juga sudah diubah menjadi deretan karakter ASCII pada langkah *preprocessing* gambar dataset. Untuk detail cara kerja algoritma KMP dan BM akan dijelaskan pada sub bab berikutnya.

5. Pengecekan kemiripan

Apabila *pattern* gambar uji tidak ditemukan di dataset, maka akan dilakukan pengecekan kemiripan dengan algoritma *Hamming Distance*. Algoritma Hamming Distance akan membandingkan setiap karakter pada gambar satu dengan karakter yang bersesuaian pada gambar dua. Lalu akan didapatkan jumlah perbedaan karakter pada kedua gambar tersebut. Selanjutnya dicari persentase kemiripannya untuk ditampilkan pada antarmuka.

6. Pengambilan informasi biodata

Setelah menemukan gambar di dataset yang diinginkan setelah langkah pencocokan dan pengecekan kemiripan, akan didapatkan juga nama pemilik gambar sidik jari tersebut dari tabel sidik_jari. Informasi ini digunakan untuk mendapatkan biodata dari orang tersebut berdasarkan tabel biodata. Oleh karena nama pada tabel biodata bisa saja *corrupt* maka regex akan dimanfaatkan. Detail cara kerja regex akan dijelaskan di sub bab berikutnya.

7. Penampilan informasi

Informasi yang didapat akan ditampilkan pada antarmuka pengguna.

B. Proses Pencarian Solusi dengan Algoritma KMP dan BM

• KMP

Pertama adalah menghitung "border array" dari pola yang diberikan. Border array digunakan untuk menentukan seberapa banyak kita bisa menggeser pola tanpa harus membandingkan ulang semua karakter yang sudah cocok.

Langkah-langkahnya:

1. Inisialisasi panjang pola m dan array b untuk menyimpan border values.
2. Set nilai awal j (indeks pola) ke 0 dan i (indeks teks) ke 1.

3. Looping melalui pola:
 - a. Jika karakter pada posisi j dan i sama, increment kedua indeks dan simpan nilai j pada b[i].
 - b. Jika tidak sama dan j lebih besar dari 0, set j ke nilai border sebelumnya.
 - c. Jika tidak sama dan j sama dengan 0, set b[i] ke 0 dan increment i.
4. Mengembalikan array b yang berisi border values.

Berikutnya adalah proses pencarian dengan bantuan "border array" untuk mencari pola dalam teks menggunakan border array yang dihitung sebelumnya.

Langkah-langkahnya:

1. Inisialisasi panjang teks n, panjang pola m, dan array border b dari hasil langkah sebelumnya.
2. Set nilai awal i (indeks teks) dan j (indeks pola) ke 0.
3. Looping melalui teks:
 - a. Jika karakter pada posisi j dari pola dan i dari teks sama, increment kedua indeks. Jika j mencapai akhir pola, kembalikan indeks awal dari kecocokan.
 - b. Jika tidak sama dan j lebih besar dari 0, set j ke nilai border sebelumnya.
 - c. Jika tidak sama dan j sama dengan 0, increment i.
4. Jika tidak ada kecocokan, mengembalikan -1.

• BM

Sebelum memulai pencarian pola menggunakan algoritma BM, perlu dibuat terlebih dahulu tabel "Last Occurrence" yang berisi indeks terakhir suatu karakter. Tabel ini berguna untuk menentukan seberapa jauh kita harus menggeser pola apabila tidak ditemukan kecocokan pada teks. Untuk mencari nilai tabel "Last Occurrence" diperlukan langkah-langkah sebagai berikut :

1. Inisialisasi array sepanjang 256 untuk menampung semua karakter ascii yang indeksnya merupakan representasi dari nilai ascii.
2. Ini nilai array tersebut dengan indeks terakhir dari karakter yang ada pada pola, isi karakter yang tidak ada dengan nilai -1.
3. Kembalikan array yang berisi tabel "Last Occurrence"

Selanjutnya pencarian pola dilakukan menggunakan algoritma BM. Adapun Langkah-langkahnya sebagai berikut :

1. Inisialisasi panjang teks n, panjang pola m, dan array "Last Occurrence" 1 dari hasil langkah sebelumnya.

2. Set nilai awal i (indeks teks) dan j (indeks pola) ke indeks terakhir pola.
3. Misalkan ketidakcocokan antara *pattern* P dan teks T terjadi pada $P[j]$ dan $T[i]$, dengan $T[i]$ adalah x . Pertama, jika terdapat x pada P , maka P digeser supaya *last occurrence* dari x pada P sejajar dengan $T[i]$. Kedua, jika terdapat x pada P pada suatu indeks tertentu tetapi pergeseran ke *last occurrence* dari x tidak dapat dilakukan, maka P digeser satu karakter. Ketiga, apabila kedua kondisi sebelumnya tidak dicapai, maka P digeser supaya $P[0]$ sejajar dengan $T[i+1]$. Hal ini dilakukan sampai ditemukan kecocokan atau pencocokan sudah mencapai indeks akhir dari Teks.
4. Jika tidak ada kecocokan, mengembalikan -1.

C. Proses Pencocokan Nama pada Biodata Menggunakan Regex

Untuk mengambil informasi biodata dari pemilik sidik jari yang sesuai, perlu digunakan regex. Pasalnya, informasi nama yang terdapat pada tabel *biodata* tidak selalu nama yang tepat, sehingga perlu dilakukan proses pencocokan dari nama yang benar pada tabel *sidik_jari* dengan nama yang rusak pada tabel *biodata*. Jenis kerusakan nama pada tabel biodata bahasa alay Indonesia.

Terdapat tiga contoh kasus bahasa alay: kombinasi huruf besar-kecil, penggunaan angka, penyingkatan, atau gabungan ketiganya. Misal terdapat nama asli yaitu Raini Wirohadikusumo, nama yang rusak bisa saja menjadi raiNI WirOHadikuSumO, R41n1 W1rohad1ku5umo, Rain Wrhdkusm dan sebagainya. Supaya nama-nama tersebut bisa cocok dengan nama aslinya, program akan membangun regex berdasarkan nama yang ditemukan dalam tabel *sidik_jari* dan kemudian mencocokkan regex tersebut dengan nama pada tabel *biodata*

Berikut merupakan langkah-langkah yang dilakukan dalam mencocokkan nama pada tabel *sidik_jari* (selanjutnya disebut nama asli) dengan nama pada tabel *biodata* (selanjutnya disebut nama yang rusak) menggunakan regex

1. Program mendapatkan nama asli setelah melakukan pencocokan sidik jari
2. Regex dibangun berdasarkan nama asli. Setiap huruf pada nama asli dijadikan satu buah disjunction berupa huruf kecil, huruf besar, dan variasi angkanya apabila ada. Selain itu, apabila huruf tersebut merupakan huruf vokal, regex ditambah dengan simbol "?". Misal huruf 'k' menjadi [kK], huruf 'a' menjadi [aA4]?, huruf 's' menjadi [sS5], dan lain-lain.
3. Membandingkan regex yang telah dibangun dengan masing-masing data nama pada tabel *biodata*. Apabila tidak ditemukan nama yang sama, maka tidak ditemukan biodata pemilik sidik jari yang bersesuaian

D. Fitur Fungsional dan Arsitektur Aplikasi Desktop yang Dibangun

1. Fitur-Fitur Fungsional

Fitur-fitur fungsional yang terdapat dalam Aplikasi Desktop kami yaitu sebagai berikut :

- a. *Label*, digunakan untuk menampilkan output berbentuk tulisan pada antarmuka seperti isi biodata, nilai kemiripan, nilai waktu, beserta komponen teks lainnya.
- b. *Button*, pengguna dapat memilih jenis algoritma untuk dijalankan antara KMP dan BM. Selain itu button juga digunakan untuk memilih gambar yang mau dibandingkan serta menjalankan algoritma
- c. *PictureBox*, digunakan untuk menampilkan output berbentuk gambar seperti gambar sidik jari masukan dan sidik jari dari database.
- d. *BackgroundWorker*, digunakan untuk menjalankan algoritma pattern matching di latar belakang.

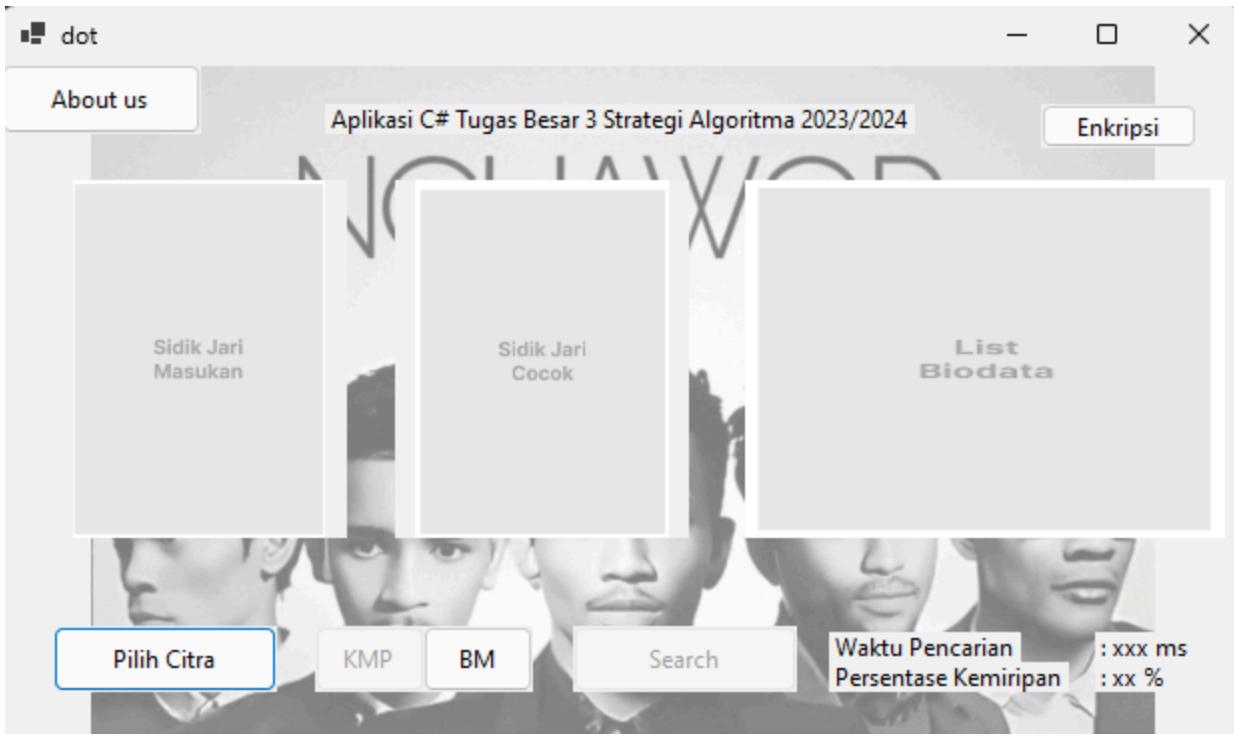
2. Arsitektur Aplikasi Desktop

Main Program pada aplikasi ini berada pada file Program.cs. Program.cs akan menjalankan file Tubes.cs yang menggunakan Tubes.Designer.cs untuk menampilkan antarmuka pengguna. Tubes.cs akan menangani masukan gambar dan pemilihan algoritma untuk pattern matching. Apabila tombol search ditekan maka class Backend akan bekerja. Backend akan melakukan algoritma pattern matching yang dipilih dan apabila tidak menemukan kecocokan, maka dilakukan algoritma Hamming Distance untuk mencari persen kemiripan antara kedua gambar. Backend lalu mengirimkan data pada antarmuka untuk ditampilkan pada aplikasi.

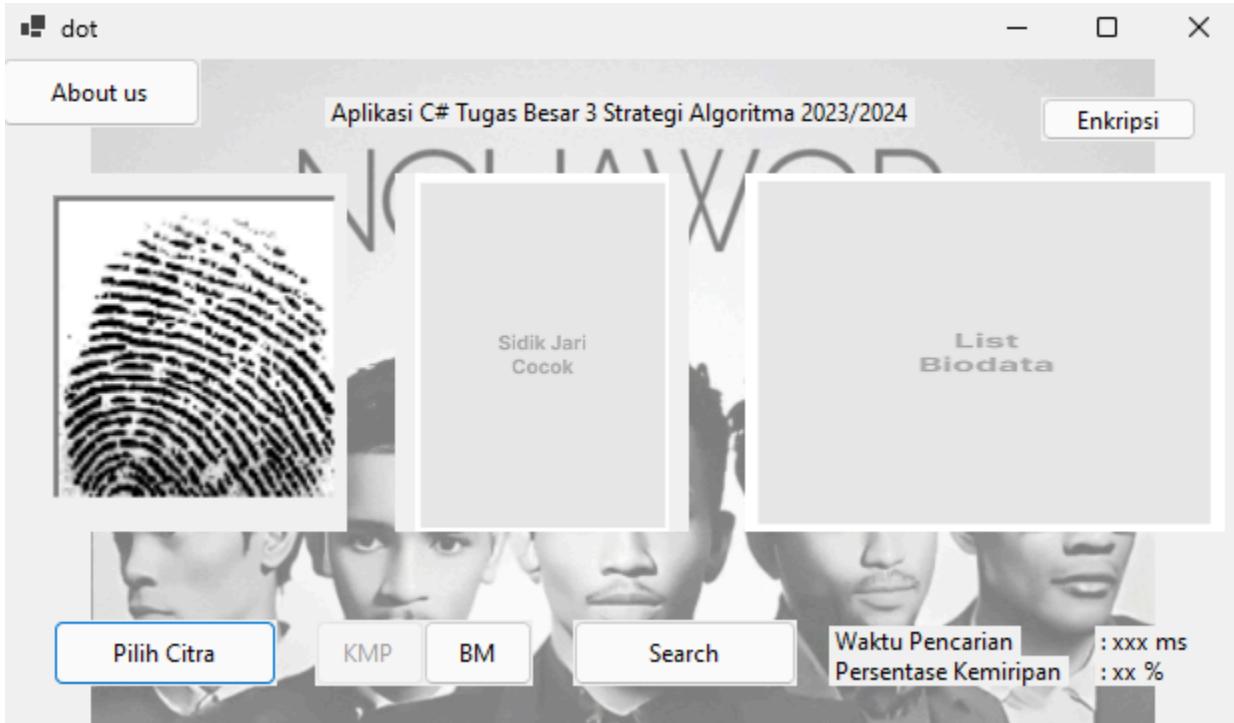
E. Contoh Ilustrasi Kasus

Berikut adalah beberapa contoh ilustrasi kasus dari program kami

1. Tampilan awal antarmuka

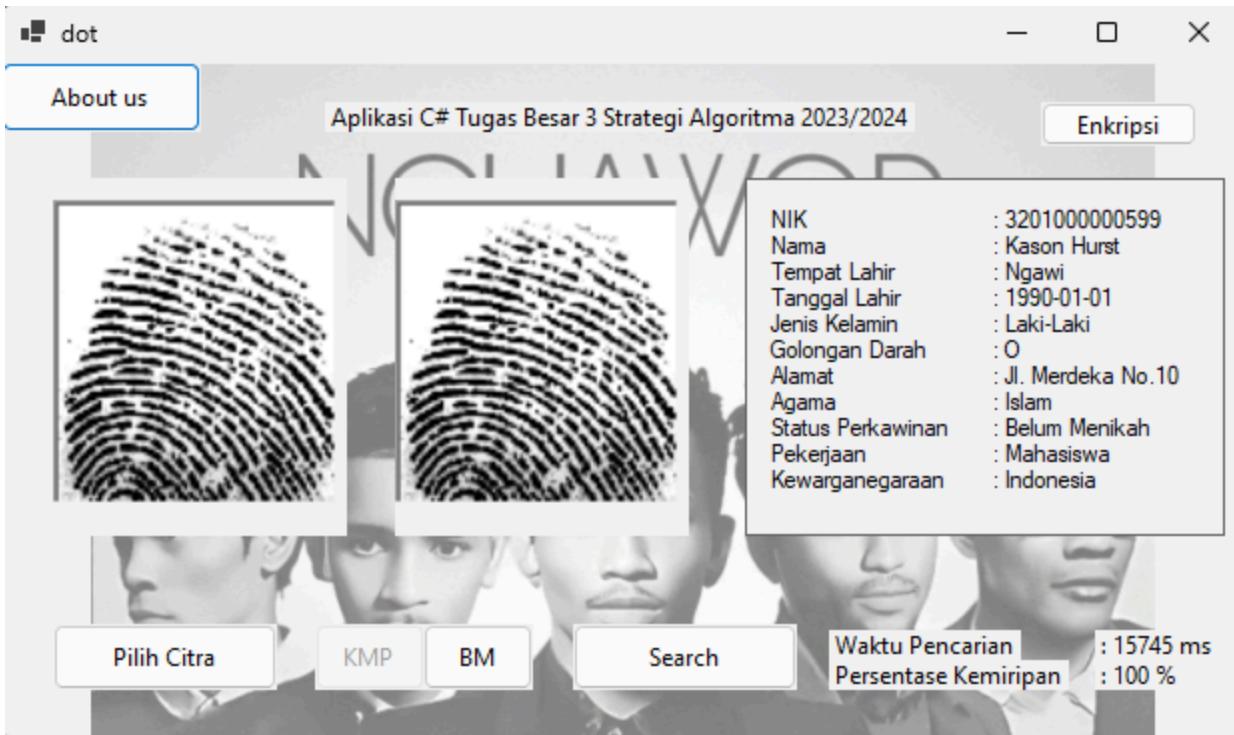


2. Tombol “Pilih Citra” bisa ditekan untuk memilih gambar yang akan dicari

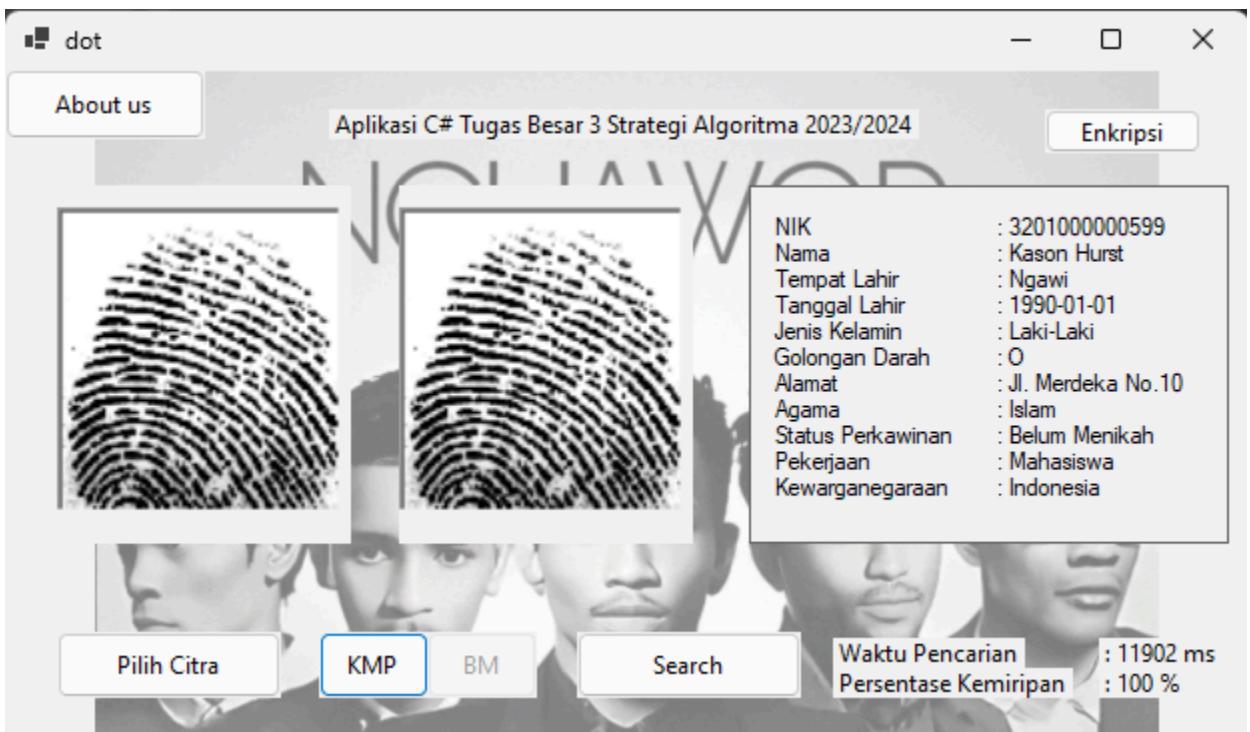


3. Tombol “KMP” atau “BM” bisa ditekan sebagai jenis algoritma pencocokan string yang dipilih. Setelah itu tombol “Search” bisa dipilih untuk memulai pencarian

- Pilihan algoritma KMP

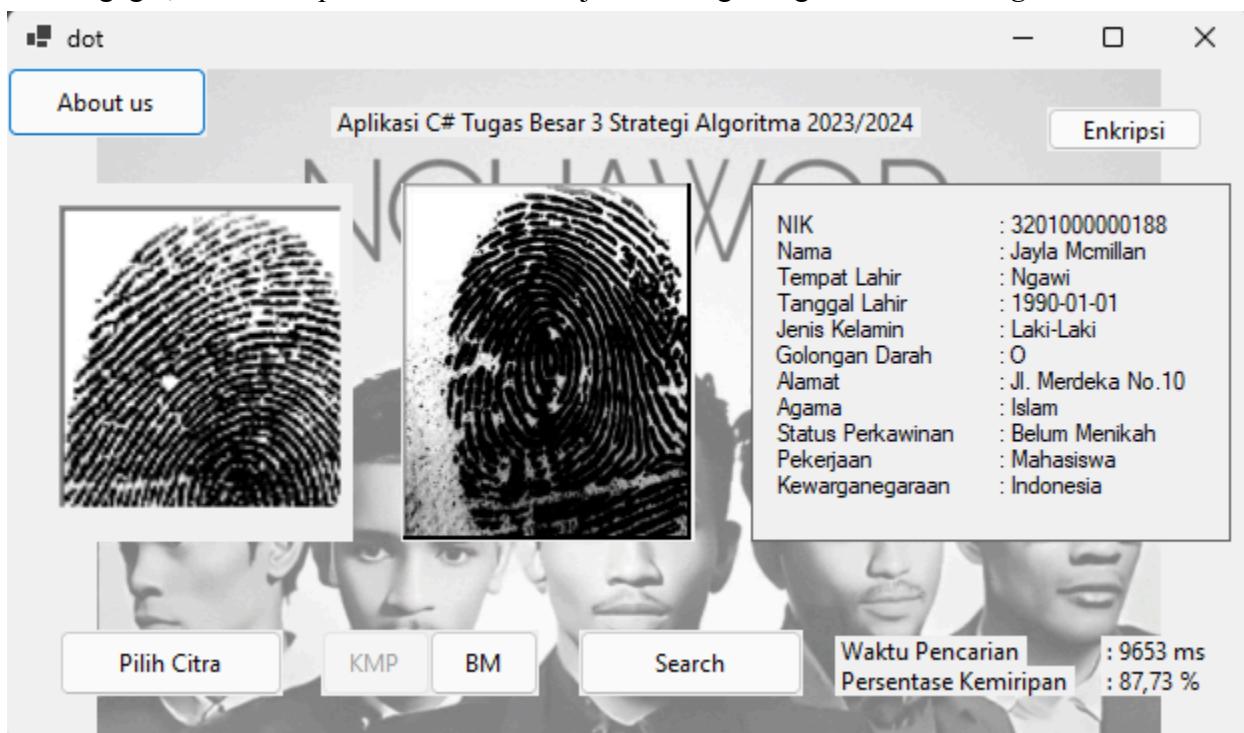


- Pilihan algoritma BM



4. Informasi terkait biodata, waktu pencarian, dan persentase kemiripan dapat dilihat pada tampilan antarmuka.

5. Persentase kemiripan 100% menandakan hasil berhasil ditemukan dengan algoritma BM atau KMP
6. Persentase kemiripan kurang dari 100% menandakan algoritma KMP dan BM gagal, kemudian pencarian akan dilanjutkan dengan algoritma *Hamming Distance*



BAB IV

IMPLEMENTASI DAN PENGUJIAN

A. Spesifikasi Teknis Program

Program berada dalam satu namespace yang sama yaitu Tubes3_let_me_seedik. Terdapat beberapa kelas yang berada pada program ini.

1. Algoritma KMP

Algoritma KMP diimplementasikan pada file KMP.cs. Algoritma ini digunakan untuk melakukan pencocokan pola pada program. Terdapat 2 fungsi yaitu fungsi ComputeBorder yang menerima string pattern dan mengembalikan array border seperti yang telah dijelaskan sebelumnya, dan fungsi KMPmatch yang melakukan algoritma pencocokan pola.

```
● ● ●

1  namespace Tubes3_let_me_seedik;
2  using System;
3  public class KMP
4  {
5      public static int[] ComputeBorder(string pattern)
6      {
7          int m = pattern.Length;
8          int[] b = new int[m];
9          int j = 0;
10         int i = 1;
11         b[0] = 0;
12         while (i < m){
13             if (pattern[j] == pattern[i]) {
14                 j++;
15                 b[i] = j;
16                 i++;
17             } else if (j > 0){
18                 j = b[j - 1];
19             } else {
20                 b[i] = 0;
21                 i++;
22             }
23         }
24     }
25     return b;
26 }
```

```

27     public static int KmpMatch(string text, string pattern)
28     {
29         int n = text.Length;
30         int m = pattern.Length;
31         int[] b = ComputeBorder(pattern);
32         int i = 0;
33         int j = 0;
34         while (i < n) {
35             if (pattern[j] == text[i]) {
36                 if (j == m - 1){
37                     return i - m + 1; // Ketemu
38                 }
39                 i++;
40                 j++;
41             }
42             else if (j > 0) {
43                 j = b[j - 1];
44             }
45             else{
46                 i++;
47             }
48         }
49         return -1; // Gak Ketemu
50     }
51 }
52

```

2. Algoritma BM

Algoritma BM diimplementasikan pada file BM.cs. Algoritma ini digunakan untuk melakukan pencocokan pola pada program. Terdapat 2 fungsi yaitu fungsi ListLastOccurrence yang menerima string pattern dan mengembalikan array last occurrence seperti yang telah dijelaskan sebelumnya, dan fungsi BMmatch yang melakukan algoritma pencocokan pola.

```

● ● ●

1  namespace Tubes3_let_me_seedik;
2  using System;
3  public class BM
4  {
5      public static int[] ListLastOccurance(string pattern)
6      {
7          int[] listLO = new int[256];
8          for (int i = 0; i < listLO.Length; i++) {
9              listLO[i] = -1;
10         }
11         for (int i = 0; i < pattern.Length; i++) {
12             listLO[(int) pattern[i]] = i;
13         }
14         return listLO;
15     }

```

```

16     public static int BmMatch(string text, string pattern)
17     {
18         int[] listLO = ListLastOccurance(pattern);
19         int lenText = text.Length;
20         int lenPattern = pattern.Length;
21         int i = lenPattern - 1;
22         if (i > lenText - 1) {
23             return -1;
24         }
25         int j = lenPattern - 1;
26         while(i < lenText - 1) {
27             if (pattern[j] == text[i]) {
28                 if (j == 0) {
29                     return i;
30                 } else {
31                     i--;
32                     j--;
33                 }
34             }
35             else {
36                 int lastOccurrence = listLO[(int)text[i]];
37                 i = i + lenPattern - Math.Min(lastOccurrence+1, j);
38                 j = lenPattern - 1;
39             }
40         }
41         return -1;
42     }
43 }
```

3. Algoritma Hamming Distance

Algoritma Hamming Distance diimplementasikan pada file Hamming.cs. Algoritma ini digunakan untuk mencari kemiripan dari dua string. Algoritma ini menerima string pattern dan string text dan mengembalikan persentase kemiripannya.



```

1  namespace Tubes3_let_me_seedik;
2  using System;
3  public class Hamming
4  {
5      public static float HammingDistance(string text, string pattern)
6      {
7          if(text.Length != pattern.Length){
8              Console.WriteLine("Panjang string harus sama");
9              return -1;
10         }
11         int i = 0;
12         int distance = 0;
13         while (i < pattern.Length) {
14             if (text[i] != pattern[i]) {
15                 distance++;
16             }
17             i++;
18         }
19         return 100-((float)distance *100/pattern.Length);
20     }
21 }
```

4. Algoritma Regex

Algoritma Regex diimplementasikan pada file Regex.cs. File ini digunakan untuk membuat regex dari nama asli untuk mencocokkan dengan nama alay pada biodata.

```
● ● ●

1  namespace Tubes3_let_me_seedik;
2  using System.Text.RegularExpressions;
3  using System.Text;
4  public class Regexp
5  {
6      public bool isMatch(string text, string pattern)
7      {
8          Match match = Regex.Match(text, pattern);
9          return match.Success;
10     }
11
12     public string createRegex(string text)
13     {
14         StringBuilder pattern = new StringBuilder();
15         foreach (char c in text)
16         {
17             pattern.Append(charToPattern(c));
18         }
19         return pattern.ToString();
20     }
21 }
```

```
22     private string charToPattern(char c)
23     {
24         switch (char.ToLower(c))
25         {
26             case 'a':
27                 return "[aA4]?";
28             case 'b':
29                 return "[bB8]";
30             case 'c':
31                 return "[cC]";
32             case 'd':
33                 return "[dD]";
34             case 'e':
35                 return "[eE3]?";
36             case 'f':
37                 return "[fF]";
38             case 'g':
39                 return "[gG69]";
40             case 'h':
41                 return "[hH]";
42             case 'i':
43                 return "[iI1]?";
44             case 'j':
45                 return "[jJ]";
46             case 'k':
47                 return "[kK]";
48             case 'l':
49                 return "[lL]";
50             case 'm':
51                 return "[mM]";
52             case 'n':
53                 return "[nN]";
```

```
54         case 'o':
55             return "[oO0]?";
56         case 'p':
57             return "[pP]";
58         case 'q':
59             return "[qQ]";
60         case 'r':
61             return "[rR]";
62         case 's':
63             return "[sS5]";
64         case 't':
65             return "[tT7]";
66         case 'u':
67             return "[uU]?";
68         case 'v':
69             return "[vV]";
70         case 'w':
71             return "[wW]";
72         case 'x':
73             return "[xX]";
74         case 'y':
75             return "[yY]";
76         case 'z':
77             return "[zZ2]";
78         case ' ':
79             return "\\s+";
80     }
81     return c.ToString();
82 }
83 }
```

5. Database Manager

Database Manager diimplementasikan pada file DatabaseManager.cs. File ini akan menangani seluruh hal yang berkaitan dengan manajemen database.

```
1  namespace Tubes3_let_me_seedik;
2
3  using MySql.Data.MySqlClient;
4  using System;
5
6  public class DatabaseManager
7  {
8      private MySqlConnection connection;
9      public DatabaseManager(string connectionString)
10     {
11         connection = new MySqlConnection(connectionString);
12     }
13
14     public void OpenConnection()
15     {
16         try
17         {
18             if (connection.State == System.Data.ConnectionState.Closed)
19                 connection.Open();
20         }
21         catch (MySqlException ex)
22         {
23             Console.WriteLine($"MySQL error: {ex.Message}");
24             throw;
25         }
26         catch (Exception ex)
27         {
28             Console.WriteLine($"General error: {ex.Message}");
29             throw;
30         }
31     }
32     public void CloseConnection()
33     {
34         if (connection.State == System.Data.ConnectionState.Open)
35             connection.Close();
36     }
}
```

Dan beberapa fungsi lain yang digunakan untuk membuat seed untuk database.

6. File Manager

File Manager diimplementasikan pada file FileManager.cs. File ini akan menangani penelusuran file.

```
1  namespace Tubes3_let_me_seedik;
2  using System.Collections.Generic;
3  using System.IO;
4  public class FileManager
5  {
6      public static List<string> GetFilePaths(string folderPath)
7      {
8          List<string> files = new List<string>();
9          folderPath = Path.Combine(folderPath);
10         var filePaths = Directory.GetFiles(folderPath, "*.BMP");
11         foreach (var filePath in filePaths) {
12             files.Add(filePath.Replace("\\\\", "/"));
13         }
14         return files;
15     }
16
17     public static List<string> GetRightThumbFingerFiles(string folderPath)
18     {
19         List<string> rightThumbFiles = new List<string>();
20         folderPath = Path.Combine(folderPath);
21         var filePaths = Directory.GetFiles(folderPath, "*Right_thumb_finger.BMP");
22         foreach (var filePath in filePaths) {
23             rightThumbFiles.Add(filePath.Replace("\\\\", "/"));
24         }
25         return rightThumbFiles;
26     }
27
28     public static List<string> GetThousandRightThumbFingerFiles(string folderPath)
29     {
30         List<string> rightThumbFiles = GetRightThumbFingerFiles(folderPath);
31         if (rightThumbFiles.Count > 1000) {
32             rightThumbFiles = rightThumbFiles.GetRange(0, 1000);
33         }
34         return rightThumbFiles;
35     }
36     public static string[] ReadFileToArray(string filePath)
37     {
38         if (File.Exists(filePath)) {
39             return File.ReadAllLines(filePath);
40         } else {
41             throw new FileNotFoundException($"File tidak ditemukan: {filePath}");
42         }
43     }
44 }
```

7. Converter Gambar

Converter gambar diimplementasikan pada file GrayscaleToASCIIConverter.cs. File ini akan mengubah gambar dari ekstensi bitmap menjadi list dari karakter ascii.

```
1  namespace Tubes3_let_me_seedik;
2  using System;
3  using System.Drawing;
4  using System.Text;
5  public class GrayscaleToASCIIConverter
6  {
7      public string ConvertToASCII(string imagePath)
8      {
9          Bitmap bitmap = new Bitmap(imagePath);
10         StringBuilder asciiImage = new StringBuilder();
11         for (int y = 0; y < bitmap.Height; y++) {
12             for (int x = 0; x < bitmap.Width; x++) {
13                 Color pixelColor = bitmap.GetPixel(x, y);
14                 int grayscaleValue = pixelColor.R;
15                 char asciiChar = (char)grayscaleValue;
16                 asciiImage.Append(asciiChar);
17             }
18         }
19         bitmap.Dispose();
20         return asciiImage.ToString();
21     }
22     public string GetBestSegment(string imagePath)
23     {
24         string asciiImage = ConvertToASCII(imagePath);
25         //panjang ASCII gambar < 30
26         if (asciiImage.Length < 30) {
27             return asciiImage;
28         }
29         int middleIndex = asciiImage.Length / 2;
30         int startIndex = middleIndex - 15;
31         if (startIndex < 0) {
32             startIndex = 0;
33         }
34         int countSegment = 30;
35         //ini gak begitu ngaruh sih
36         if (startIndex + 30 > asciiImage.Length) {
37             countSegment = asciiImage.Length - startIndex;
38         }
39         string bestSegment = asciiImage.Substring(startIndex, countSegment);
40         return bestSegment;
41     }
}
```

```

42     public string GetUpperMiddleSegment(string imagePath)
43     {
44         string asciiImage = ConvertToASCII(imagePath);
45         if (asciiImage.Length < 30) {
46             return asciiImage;
47         }
48         int middleIndex = asciiImage.Length / 2;
49         int upperMiddleIndex = middleIndex * 7/8;
50         int startIndex = upperMiddleIndex - 15;
51         if (startIndex < 0) {
52             startIndex = 0;
53         }
54         int countSegment = 30;
55         if (startIndex + 30 > asciiImage.Length) {
56             countSegment = asciiImage.Length - startIndex;
57         }
58         string upperMiddleSegment = asciiImage.Substring(startIndex, countSegment);
59         return upperMiddleSegment;
60     }
61
62     public string GetLowerMiddleSegment(string imagePath)
63     {
64         string asciiImage = ConvertToASCII(imagePath);
65         if (asciiImage.Length < 30) {
66             return asciiImage;
67         }
68         int middleIndex = asciiImage.Length / 2;
69         int lowerMiddleIndex = middleIndex + (middleIndex / 8);
70         int startIndex = lowerMiddleIndex - 15;
71         if (startIndex < 0) {
72             startIndex = 0;
73         }
74         int countSegment = 30;
75         if (startIndex + 30 > asciiImage.Length) {
76             countSegment = asciiImage.Length - startIndex;
77         }
78         string lowerMiddleSegment = asciiImage.Substring(startIndex, countSegment);
79         return lowerMiddleSegment;
80     }
81 }

```

8. Antarmuka Pengguna

Algoritma yang menangani antarmuka Pengguna berada pada file Tubes.cs dan Tubes.Designer.cs.

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using DotNetEnv;
using Org.BouncyCastle.Crypto.Agreement.Srp;

namespace Tubes3_let_me_seedik
{
    public partial class Tubes : Form
    {

```

```

        private List<ControlInfo> controlInfos = new
List<ControlInfo>();
        private Size initialFormSize;
        private bool isKMP = true;
        private bool isEnkripsi = false;
        private readonly string targetFolderPath =
System.IO.Path.GetFullPath("src/citra");

        public Tubes()
{
    Console.WriteLine(targetFolderPath);
    InitializeComponent();
    labelFormat.Font = new Font("Microsoft Sans
Serif", 8, FontStyle.Regular);
    labelData.Font = new Font("Microsoft Sans Serif",
8, FontStyle.Regular);
}

private void Tubes_Load(object sender, EventArgs e)
{
    initialFormSize = this.ClientSize;
    foreach (Control control in this.Controls)
    {
        controlInfos.Add(new ControlInfo(control,
control.Location, control.Size, control.Font));
    }
}

private void Tubes_Resize(object sender, EventArgs e)
{
    if (initialFormSize.Width == 0 || initialFormSize.Height == 0) return;

    // Hitung rasio perubahan ukuran form
    float widthRatio = (float)this.ClientSize.Width /
initialFormSize.Width;
    float heightRatio = (float)this.ClientSize.Height /
initialFormSize.Height;
    float fontRatio = Math.Min(widthRatio,
heightRatio);
}

```

```

        // Sesuaikan ukuran dan posisi kontrol berdasarkan
ratio perubahan ukuran form
        foreach (ControlInfo controlInfo in controlInfos)
        {
            controlInfo.Control.Location = new
Point((int)(controlInfo.OriginalLocation.X * widthRatio),
(int)(controlInfo.OriginalLocation.Y * heightRatio));
            controlInfo.Control.Size = new
Size((int)(controlInfo.OriginalSize.Width * widthRatio),
(int)(controlInfo.OriginalSize.Height * heightRatio));

            // Sesuaikan ukuran font
            if (controlInfo.Control is Button ||
controlInfo.Control is Label)
            {
                float newSize =
controlInfo.OriginalFont.Size * fontRatio;
                if (newFontSize > 0)
                {
                    controlInfo.Control.Font = new
Font(controlInfo.OriginalFont.FontFamily, newSize,
controlInfo.OriginalFont.Style);
                }
            }
        }

        private void buttonCitra_Click(object sender,
EventArgs e)
{
    // Create an instance of OpenFileDialog
    OpenFileDialog openFileDialog = new
OpenFileDialog();

    // Set filter options and filter index
    openFileDialog.Filter = "Image
Files|*.jpg;*.jpeg;*.png;*.bmp;*.gif";
    openFileDialog.FilterIndex = 1;

    // Set the title of the dialog
}

```

```

        openFileDialog.Title = "Select an Image";

            // Show the dialog and check if the user selected
            a file
            if (openFileDialog.ShowDialog() ==
DialogResult.OK)
            {
                // Load the selected image into the PictureBox
                pictureBoxInput.Image =
Image.FromFile(openFileDialog.FileName);
                // Optional: SetSizeMode to Zoom so the image
                scales properly
                pictureBoxInput.SizeMode =
PictureBoxSizeMode.Zoom;
                try
                {
                    // Load the selected image into the
                    PictureBox
                    pictureBoxInput.Image =
Image.FromFile(openFileDialog.FileName);
                    // Optional: SetSizeMode to Zoom so the
                    image scales properly
                    pictureBoxInput.SizeMode =
PictureBoxSizeMode.Zoom;

                    // Get the file name and create the target
                    file path
                    /*string fileName =
Path.GetFileName(openFileDialog.FileName);*/
                    string targetFilePath =
Path.Combine(targetFolderPath, "Data.bmp");

                    // Copy the selected file to the target
                    folder
                    File.Copy(openFileDialog.FileName,
targetFilePath, overwrite: true);

                    /* MessageBox.Show("Image copied
successfully      to      "      +      targetFilePath,      "Success",
MessageBoxButtons.OK, MessageBoxIcon.Information);*/

```

```

        }

        catch (Exception ex)
        {
            MessageBox.Show("An error occurred while
copying the image: " + ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    buttonSearch.Enabled = true;
}

private void buttonBM_Click(object sender, EventArgs e)
{
    /*buttonBM.BackColor = 
System.Drawing.Color.LightBlue;
buttonKMP.BackColor = 
System.Drawing.SystemColors.Control;*/
    buttonBM.Enabled = false;
    buttonKMP.Enabled = true;
    isKMP = false;
}

private void buttonKMP_Click(object sender, EventArgs e)
{
    /*buttonKMP.BackColor = 
System.Drawing.Color.LightBlue;
buttonBM.BackColor = 
System.Drawing.SystemColors.Control;*/
    buttonKMP.Enabled = false;
    buttonBM.Enabled = true;
    isKMP = true;
}

private void buttonSearch_Click(object sender,
EventArgs e)
{
    buttonSearch.Enabled = false;
    labelData.Visible = false;
}

```

```

        labelFormat.Visible = false;
                    pictureBoxOutput.Image     =
src.Properties.Resources.loading_icon_wait_vector_260nw_172256
8561;
                    pictureBoxbiodata.Image    =
src.Properties.Resources.loading_icon_wait_vector_260nw_172256
8561;
                    backgroundWorkerSearch.RunWorkerAsync();
// Thread.Sleep(3000);
}

private void backgroundWorkerSearch_DoWork(object
sender, DoWorkEventArgs e)
{
    Stopwatch timer = Stopwatch.StartNew();
    if (!isEnkripsi)
    {
        BackEnd backEnd = new BackEnd(isKMP);
    }
    else
    {
        BackEndBonus backEnd = new
BackEndBonus(isKMP);
    }
    timer.Stop();
    long timeSpan = timer.ElapsedMilliseconds;
    int waktu = (int)timeSpan;
    labelNilaiWaktu.Text = ":" + waktu.ToString() + "ms";
    buttonSearch.Enabled = true;
}

private void
backgroundWorkerSearch_RunWorkerCompleted(object
sender, RunWorkerCompletedEventArgs e)
{
    if(isEnkripsi) {
        if (BackEndBonus.persentase >= 80)
        {
            float kemiripan = BackEndBonus.persentase;
}

```

```

                labelNilaiKemiripan.Text = ":" + 
Math.Round(kemiripan, 2).ToString() + " %";
                pictureBoxbiodata.Image = null;
                string nik = BackEndBonus.biodata[0];
                string nama = BackEndBonus.biodata[1];
                string tempatlahir =
BackEndBonus.biodata[2];
                string tanggallahir =
BackEndBonus.biodata[3];
                string gender = BackEndBonus.biodata[4];
                string goldar = BackEndBonus.biodata[5];
                string alamat = BackEndBonus.biodata[6];
                string agama = BackEndBonus.biodata[7];
                string status = BackEndBonus.biodata[8];
                string pekerjaan =
BackEndBonus.biodata[9];
                string kwn = BackEndBonus.biodata[10];
                labelFormat.Text = "NIK\nNama\nTempat
Lahir\nTanggal Lahir\nJenis Kelamin\nGolongan
Darah\nAlamat\nAgama\nStatus
Perkawinan\nPekerjaan\nKewarganegaraan";
                labelFormat.Visible = true;
                labelData.Visible = true;
                labelData.Text = $"": {nik}\n: {nama}\n:
{tempatlahir}\n: {tanggallahir}\n: {gender}\n: {goldar}\n:
{alamat}\n: {agama}\n: {status}\n: {pekerjaan}\n: {kwn}";
                pictureBoxbiodata.BorderStyle =
BorderStyle.FixedSingle;
                pictureBoxOutput.Image = new
Bitmap(BackEndBonus.pathGambar);
            }
            else
            {
                pictureBoxOutput.Image =
src.Properties.Resources.download;
                pictureBoxbiodata.Image =
src.Properties.Resources.download;
                labelData.Visible = false;
                labelFormat.Visible = false;
                labelNilaiKemiripan.Text = ":" -";
            }
        }
    }
}

```

```

        }
    }
else {

    if (BackEnd.persentase >= 80)
    {
        float kemiripan = BackEnd.persentase;
        labelNilaiKemiripan.Text = ":" + +
Math.Round(kemiripan, 2).ToString() + "%";
        pictureBoxbiodata.Image = null;
        string nik = BackEnd.biodata[0];
        string nama = BackEnd.biodata[1];
        string tempatlahir = BackEnd.biodata[2];
        string tanggallahir = BackEnd.biodata[3];
        string gender = BackEnd.biodata[4];
        string goldar = BackEnd.biodata[5];
        string alamat = BackEnd.biodata[6];
        string agama = BackEnd.biodata[7];
        string status = BackEnd.biodata[8];
        string pekerjaan = BackEnd.biodata[9];
        string kwn = BackEnd.biodata[10];
        labelFormat.Text = "NIK\nNama\nTempat
Lahir\nTanggal           Lahir\nJenis           Kelamin\nGolongan
Darah\nAlamat\nAgama\nStatus
Perkawinan\nPekerjaan\nKewarganegaraan";
        labelFormat.Visible = true;
        labelData.Visible = true;
        labelData.Text = $"": {nik}\n: {nama}\n:
{tempatlahir}\n: {tanggallahir}\n: {gender}\n: {goldar}\n:
{alamat}\n: {agama}\n: {status}\n: {pekerjaan}\n: {kwn}";
        pictureBoxbiodata.BorderStyle =
BorderStyle.FixedSingle;
        pictureBoxOutput.Image = new
Bitmap(BackEnd.pathGambar);
    }
    else
    {
        pictureBoxOutput.Image =
src.Properties.Resources.download;
    }
}
}

```

```

                pictureBoxbiodata.Image =
src.Properties.Resources.download;
                labelData.Visible = false;
                labelFormat.Visible = false;
                labelNilaiKemiripan.Text = ":-";
            }
        }
    }

private void Bonus_Click(object sender, EventArgs e)
{
    string teks;
    if (!isEnkripsi)
    {
        teks = "Apa kamu yakin ingin mengenkripsi
database?";
    }
    else
    {
        teks = "Apa kamu yakin ingin mendekripsi
database?";
    }
    DialogResult dr = MessageBox.Show(teks, "Confirm",
MessageBoxButtons.YesNo, MessageBoxIcon.Question);

    if (dr == DialogResult.Yes)
    {
        Env.Load();

        string server = Env.GetString("SERVER");
        string port = Env.GetString("PORT");
        string user = Env.GetString("USER");
        string password = Env.GetString("PASSWORD");
        string database = Env.GetString("DATABASE");
        string key = Env.GetString("KEY");

        string connectionString;

        int intKey;
        bool success = int.TryParse(key, out intKey);
    }
}

```

```

        if (!success)
        {
            Console.WriteLine("Format Key Tidak
Valid");
        }

        if (string.IsNullOrEmpty(port))
        {
            connectionString = $"Server={server};User
ID={user};Password={password};Database={database};";
        }
        else
        {
            connectionString =
$"Server={server};Port={port};User
ID={user};Password={password};Database={database};";
        }

        DatabaseManager dbManager = new
DatabaseManager(connectionString);

        if (!isEnkripsi)
        {
            dbManager.EncryptAndUpdateBiodata(intKey);
            Bonus.Text = "Dekripsi";
            isEnkripsi = true;
        }
        else
        {
            dbManager.DecryptAndUpdateBiodata(intKey);
            Bonus.Text = "Enkripsi";
            isEnkripsi = false;
        }

    }

}

```

```

class ControlInfo
{
    public Control Control { get; private set; }
    public Point OriginalLocation { get; private set; }
    public Size OriginalSize { get; private set; }
    public Font OriginalFont { get; private set; }

    public ControlInfo(Control control, Point
originalLocation, Size originalSize, Font originalFont)
    {
        this.Control = control;
        this.OriginalLocation = originalLocation;
        this.OriginalSize = originalSize;
        this.OriginalFont = originalFont;
    }
}

```

File Tubes.Designer.cs adalah file yang otomatis dibuat oleh c#.

9. Backend

Algoritma Backend diimplementasikan pada file Backend.cs. File ini akan menangani dalam melakukan pencarian backend.

```

1 using System;
2 using System.Collections.Generic;
3 using System.IO;
4 using DotNetEnv;
5
6 namespace Tubes3_let_me_seudik
7 {
8     class BackEnd
9     {
10         public static float persentase { get; private set; }
11         public static string[] biodata { get; private set; }
12         public static string pathGambar { get; private set; }
13         public BackEnd(bool algoritma)
14         {
15             pathGambar = "";
16             string targetPath = "src/citra";
17             List<string> targetPaths = FileManager.GetFilesPaths(targetPath);
18             string input = targetPaths[0];
19
20             Env.Load();
21
22             string server = Env.GetString("SERVER");
23             string port = Env.GetString("PORT");
24             string user = Env.GetString("USER");
25             string password = Env.GetString("PASSWORD");
26             string database = Env.GetString("DATABASE");
27
28             string connectionString;
29             if (string.IsNullOrEmpty(port))
30             {
31                 connectionString = $"Server={server};User ID={user};Password={password};Database={database};";
32             }
33             else
34             {
35                 connectionString = $"Server={server};Port={port};User ID={user};Password={password};Database={database};";
36             }
37
38             DatabaseManager dbManager = new DatabaseManager(connectionString);
39             string[] allPath = dbManager.GetAllPath();
40             int position = 1;
41             float maxPerson = 0;
42             string real_name = "";
43             GrayscaleToASCIIConverter asciiConverter = new GrayscaleToASCIIConverter();

```

```

45         foreach(var path in allPath){
46             string wholeImage = asciiConverter.ConvertToASCII(path);
47             string bestSegment = asciiConverter.GetBestSegment(input);
48             string upperMiddleSegment = asciiConverter.GetUpperMiddleSegment(input);
49             string lowerMiddleSegment = asciiConverter.GetLowerMiddleSegment(input);
50             string wholeImageInput = asciiConverter.ConvertToASCII(input);
51
52             for(int i = 0; i < 3; i++){
53                 if(algoritma){
54                     {
55                         if(i == 0){
56                             position = KMP.KmpMatch(wholeImage, bestSegment);
57                         }
58                         else if( i == 1){
59                             position = KMP.KmpMatch(wholeImage, upperMiddleSegment);
60                         }
61                         else {
62                             position = KMP.KmpMatch(wholeImage, lowerMiddleSegment);
63                         }
64                     }
65                 else
66                 {
67                     if(i == 0){
68                         position = BM.BmMatch(wholeImage, bestSegment);
69                     }
70                     else if( i == 1){
71                         position = BM.BmMatch(wholeImage, upperMiddleSegment);
72                     }
73                     else {
74                         position = BM.BmMatch(wholeImage, lowerMiddleSegment);
75                     }
76                 }
77
78                 if(position != -1){
79                     break;
80                 }
81             }
82
83             if (position != -1)
84             {
85                 //PATH ITU ISI NYA PATH KE GAMBAR NYA
86                 real_name = dbManager.GetName(path);
87                 pathGambar = path;
88                 maxPerson = 100;
89                 break;
90             }
91             else
92             {
93                 int distance = Hamming.HammingDistance(wholeImage, wholeImageInput);
94
95                 float persent = Hamming.PersentaseKemiripan(distance, wholeImage, wholeImageInput);
96
97                 if (maxPerson < persent)
98                 {
99                     real_name = dbManager.GetName(path);
100                    pathGambar = path;
101                    maxPerson = persent;
102                }
103            }
104        }
105
106        persentase = maxPerson;
107        string[] allName = dbManager.GetAllName();
108
109        Regex r = new Regex();
110        string name_regex = r.CreateRegex(real_name);
111        string alay_name = "";
112        foreach (var name in allName) {
113            if (r.IsMatch(name, name_regex)) {
114                alay_name = name;
115                break;
116            }
117            if (alay_name == "") {
118                Console.WriteLine("nama tidak ada");
119            }
120            //INI REAL_BIODATA BUAT BIODATANYA
121            biodata = dbManager.GetBiodata(alay_name);
122            biodata[1] = real_name;
123            Console.WriteLine(biodata[1]);
124        }
125    }
126 }

```

10. Enkripsi dan Deskripsi

Algoritma yang menangani enkripsi dan deskripsi berada pada file CaesarChiper.cs.

```

1  namespace Tubes3_let_me_seedik;
2  using System;
3  using System.Text;
4  public class CaesarCipher{
5      private const string allowedChars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
6      public string Encrypt(string text, int shift)
7      {
8          StringBuilder result = new StringBuilder();
9          for (int i = 0; i < text.Length; i++) {
10              char currentChar = text[i];
11              if (allowedChars.Contains(currentChar)) {
12                  int index = allowedChars.IndexOf(currentChar);
13                  int newIndex = (index + shift) % allowedChars.Length;
14                  result.Append(allowedChars[newIndex]);
15              }
16              else {
17                  result.Append(currentChar);
18              }
19          }
20          return result.ToString();
21      }
22      public string Decrypt(string text, int shift) {
23          StringBuilder result = new StringBuilder();
24          for (int i = 0; i < text.Length; i++) {
25              char currentChar = text[i];
26              if (allowedChars.Contains(currentChar)) {
27                  int index = allowedChars.IndexOf(currentChar);
28                  int newIndex = (index - shift + allowedChars.Length) % allowedChars.Length;
29                  result.Append(allowedChars[newIndex]);
30              }
31              else {
32                  result.Append(currentChar);
33              }
34          }
35          return result.ToString();
36      }
37  }

```

B. Tata Cara Penggunaan Program

1. Buka terminal pada folder program dan jalankan command `dotnet run`
2. Pada tampilan antarmuka, tekan tombol “Pilih Citra” dan masukkan gambar sidik jari yang diinginkan
3. Pilih algoritma yang diinginkan (default adalah KMP)
4. Tekan Tombol Search untuk melakukan algoritma pencarian
5. Hasil akan ditampilkan di antarmuka pengguna.

C. Hasil Pengujian

Pengujian kami lakukan menggunakan dataset sidik jari dari Sokoto Coventry Fingerprint Dataset (SOCOFing) (<https://www.kaggle.com/datasets/ruizgara/socofing>). Kami menggunakan 600 data sidik jari pada dataset tersebut berupa sidik jari jempol kanan (right thumb) yang dimasukkan ke dalam tabel database sidik_jari. Dump SQL dari database dapat diakses pada laman ([berikut](#))

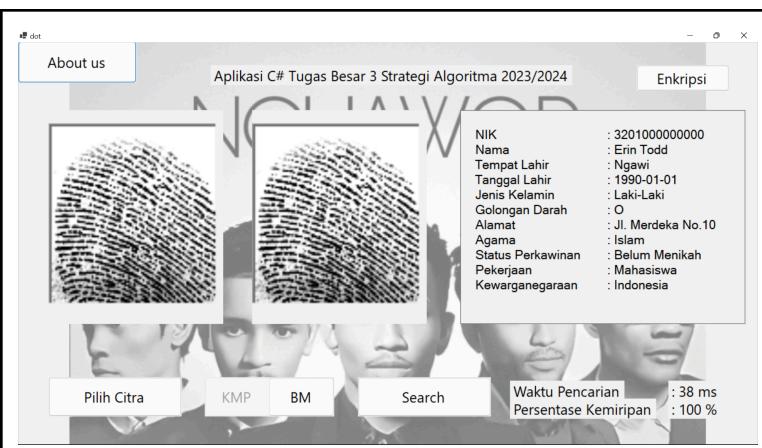
a. Pengujian Menggunakan KMP

Berikut merupakan hasil pengujian pencarian menggunakan algoritma KMP. Pencarian pada kategori ini akan menemukan hasil yang eksak

Tabel 4.1 Hasil Pengujian KMP

Kasus uji/citra yang ingin dicari	Tangkapan layar hasil pengujian
-----------------------------------	---------------------------------

100__M_Right_thumb_finger.BM
P

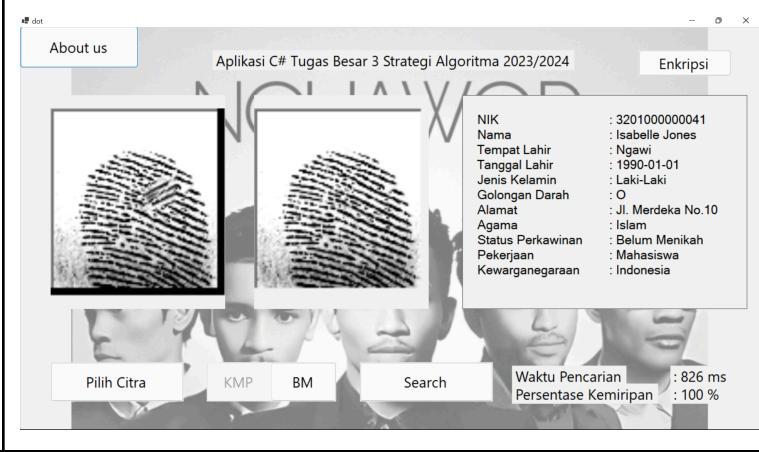
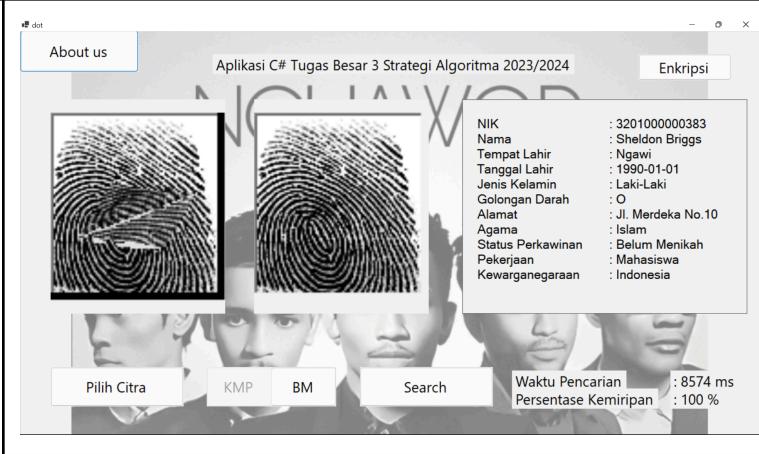


371__M_Right_thumb_finger.BM
P



9__M_Right_thumb_finger.BMP

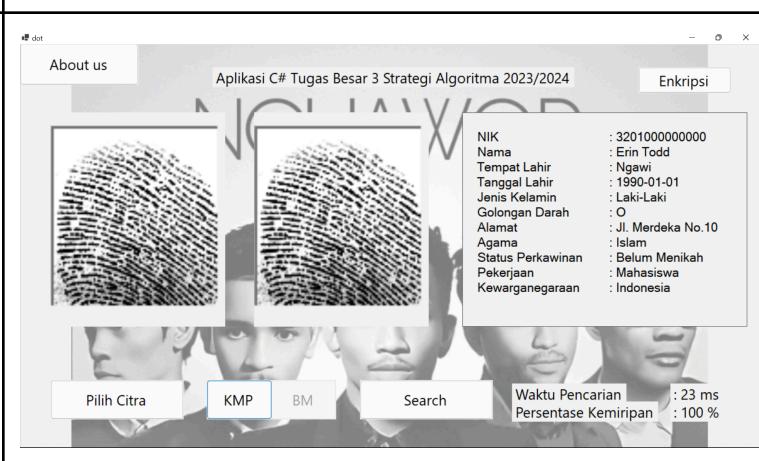


138_M_Right_thumb_finger_Zcut (Alter easy)	 <p>A screenshot of a Windows application window titled "Aplikasi C# Tugas Besar 3 Strategi Algoritma 2023/2024". The window contains two grayscale fingerprint images. Below the images is a table of search results:</p> <table border="1"> <tbody> <tr> <td>NIK</td> <td>: 320100000041</td> </tr> <tr> <td>Nama</td> <td>: Isabelle Jones</td> </tr> <tr> <td>Tempat Lahir</td> <td>: Ngawi</td> </tr> <tr> <td>Tanggal Lahir</td> <td>: 1990-01-01</td> </tr> <tr> <td>Jenis Kelamin</td> <td>: Laki-Laki</td> </tr> <tr> <td>Golongan Darah</td> <td>: O</td> </tr> <tr> <td>Alamat</td> <td>: Jl. Merdeka No.10</td> </tr> <tr> <td>Agama</td> <td>: Islam</td> </tr> <tr> <td>Status Perkawinan</td> <td>: Belum Menikah</td> </tr> <tr> <td>Pekerjaan</td> <td>: Mahasiswa</td> </tr> <tr> <td>Kewarganegaraan</td> <td>: Indonesia</td> </tr> </tbody> </table> <p>At the bottom of the window, there are buttons labeled "Pilih Citra", "KMP", "BM", "Search", and search statistics: "Waktu Pencarian : 826 ms" and "Persentase Kemiripan : 100 %".</p>	NIK	: 320100000041	Nama	: Isabelle Jones	Tempat Lahir	: Ngawi	Tanggal Lahir	: 1990-01-01	Jenis Kelamin	: Laki-Laki	Golongan Darah	: O	Alamat	: Jl. Merdeka No.10	Agama	: Islam	Status Perkawinan	: Belum Menikah	Pekerjaan	: Mahasiswa	Kewarganegaraan	: Indonesia
NIK	: 320100000041																						
Nama	: Isabelle Jones																						
Tempat Lahir	: Ngawi																						
Tanggal Lahir	: 1990-01-01																						
Jenis Kelamin	: Laki-Laki																						
Golongan Darah	: O																						
Alamat	: Jl. Merdeka No.10																						
Agama	: Islam																						
Status Perkawinan	: Belum Menikah																						
Pekerjaan	: Mahasiswa																						
Kewarganegaraan	: Indonesia																						
446_M_Right_thumb_finger_Zcut (Alter hard)	 <p>A screenshot of a Windows application window titled "Aplikasi C# Tugas Besar 3 Strategi Algoritma 2023/2024". The window contains two grayscale fingerprint images. Below the images is a table of search results:</p> <table border="1"> <tbody> <tr> <td>NIK</td> <td>: 3201000000383</td> </tr> <tr> <td>Nama</td> <td>: Sheldon Briggs</td> </tr> <tr> <td>Tempat Lahir</td> <td>: Ngawi</td> </tr> <tr> <td>Tanggal Lahir</td> <td>: 1990-01-01</td> </tr> <tr> <td>Jenis Kelamin</td> <td>: Laki-Laki</td> </tr> <tr> <td>Golongan Darah</td> <td>: O</td> </tr> <tr> <td>Alamat</td> <td>: Jl. Merdeka No.10</td> </tr> <tr> <td>Agama</td> <td>: Islam</td> </tr> <tr> <td>Status Perkawinan</td> <td>: Belum Menikah</td> </tr> <tr> <td>Pekerjaan</td> <td>: Mahasiswa</td> </tr> <tr> <td>Kewarganegaraan</td> <td>: Indonesia</td> </tr> </tbody> </table> <p>At the bottom of the window, there are buttons labeled "Pilih Citra", "KMP", "BM", "Search", and search statistics: "Waktu Pencarian : 8574 ms" and "Persentase Kemiripan : 100 %".</p>	NIK	: 3201000000383	Nama	: Sheldon Briggs	Tempat Lahir	: Ngawi	Tanggal Lahir	: 1990-01-01	Jenis Kelamin	: Laki-Laki	Golongan Darah	: O	Alamat	: Jl. Merdeka No.10	Agama	: Islam	Status Perkawinan	: Belum Menikah	Pekerjaan	: Mahasiswa	Kewarganegaraan	: Indonesia
NIK	: 3201000000383																						
Nama	: Sheldon Briggs																						
Tempat Lahir	: Ngawi																						
Tanggal Lahir	: 1990-01-01																						
Jenis Kelamin	: Laki-Laki																						
Golongan Darah	: O																						
Alamat	: Jl. Merdeka No.10																						
Agama	: Islam																						
Status Perkawinan	: Belum Menikah																						
Pekerjaan	: Mahasiswa																						
Kewarganegaraan	: Indonesia																						

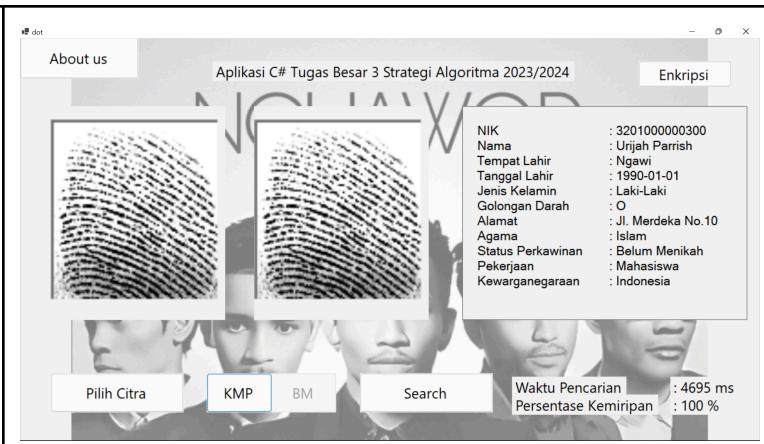
b. Pengujian Menggunakan BM

Berikut merupakan hasil pengujian pencarian menggunakan algoritma BM. Pencarian pada kategori ini akan menemukan hasil yang eksak

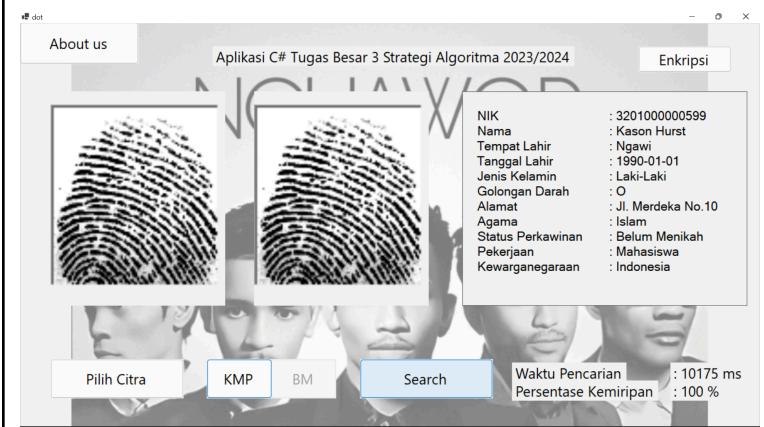
Tabel 4.2 Hasil Pengujian BM

Kasus uji/citra yang ingin dicari	Tangkapan layar hasil pengujian																						
100_M_Right_thumb_finger.BM P	 <p>A screenshot of a Windows application window titled "Aplikasi C# Tugas Besar 3 Strategi Algoritma 2023/2024". The window contains two grayscale fingerprint images. Below the images is a table of search results:</p> <table border="1"> <tbody> <tr> <td>NIK</td> <td>: 3201000000000</td> </tr> <tr> <td>Nama</td> <td>: Erin Todd</td> </tr> <tr> <td>Tempat Lahir</td> <td>: Ngawi</td> </tr> <tr> <td>Tanggal Lahir</td> <td>: 1990-01-01</td> </tr> <tr> <td>Jenis Kelamin</td> <td>: Laki-Laki</td> </tr> <tr> <td>Golongan Darah</td> <td>: O</td> </tr> <tr> <td>Alamat</td> <td>: Jl. Merdeka No.10</td> </tr> <tr> <td>Agama</td> <td>: Islam</td> </tr> <tr> <td>Status Perkawinan</td> <td>: Belum Menikah</td> </tr> <tr> <td>Pekerjaan</td> <td>: Mahasiswa</td> </tr> <tr> <td>Kewarganegaraan</td> <td>: Indonesia</td> </tr> </tbody> </table> <p>At the bottom of the window, there are buttons labeled "Pilih Citra", "KMP", "BM", "Search", and search statistics: "Waktu Pencarian : 23 ms" and "Persentase Kemiripan : 100 %".</p>	NIK	: 3201000000000	Nama	: Erin Todd	Tempat Lahir	: Ngawi	Tanggal Lahir	: 1990-01-01	Jenis Kelamin	: Laki-Laki	Golongan Darah	: O	Alamat	: Jl. Merdeka No.10	Agama	: Islam	Status Perkawinan	: Belum Menikah	Pekerjaan	: Mahasiswa	Kewarganegaraan	: Indonesia
NIK	: 3201000000000																						
Nama	: Erin Todd																						
Tempat Lahir	: Ngawi																						
Tanggal Lahir	: 1990-01-01																						
Jenis Kelamin	: Laki-Laki																						
Golongan Darah	: O																						
Alamat	: Jl. Merdeka No.10																						
Agama	: Islam																						
Status Perkawinan	: Belum Menikah																						
Pekerjaan	: Mahasiswa																						
Kewarganegaraan	: Indonesia																						

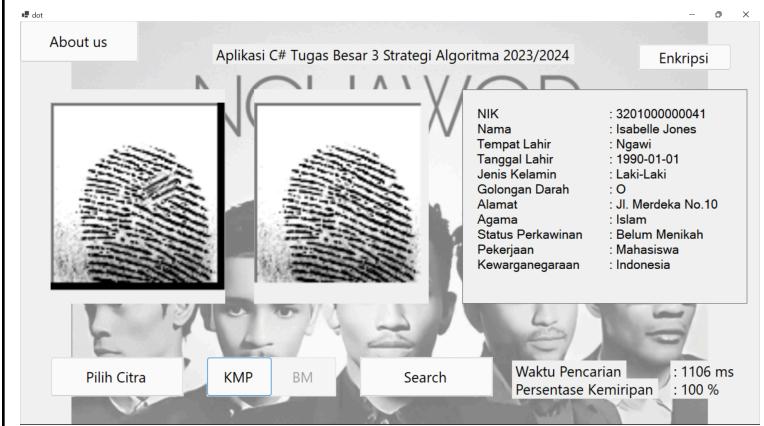
371__M_Right_thumb_finger.BM
P

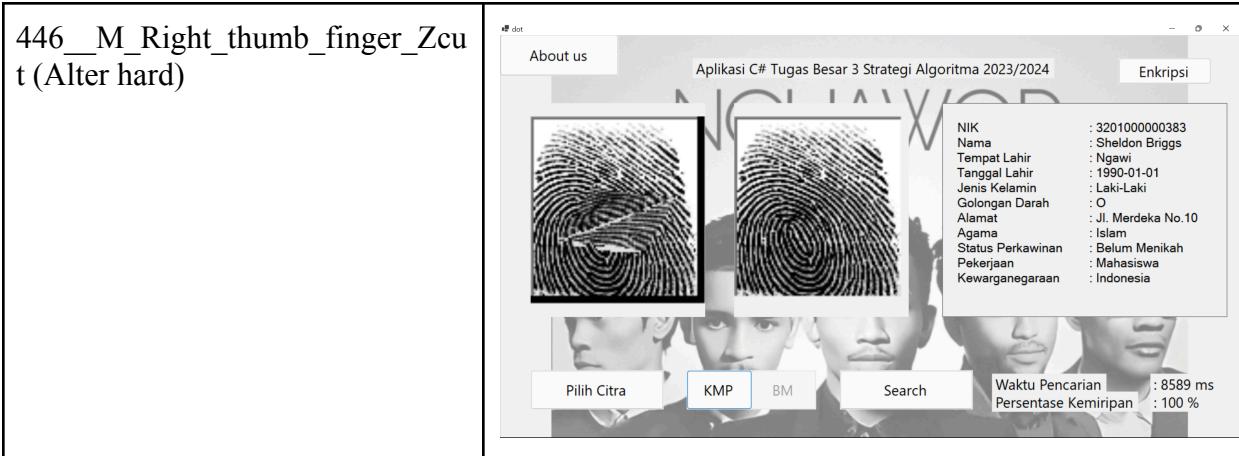


9__M_Right_thumb_finger.BMP



138__M_Right_thumb_finger_Zcut (Alter easy)





c. Pengujian Menggunakan Hamming Distance

Berikut merupakan hasil pengujian pencarian menggunakan algoritma Hamming Distance. Kategori ini dapat dibagi menjadi dua, yaitu pengujian gambar biasa (dengan gambar sidik jari dari jari selain jempol kanan) dan gambar yang berubah

- Pengujian gambar biasa

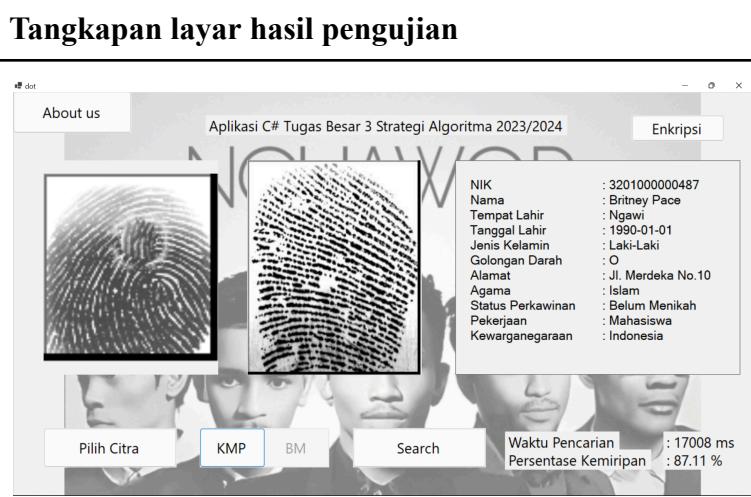
Tabel 4.3 Hasil Pengujian Hamming Distansce dengan Gambar Biasa

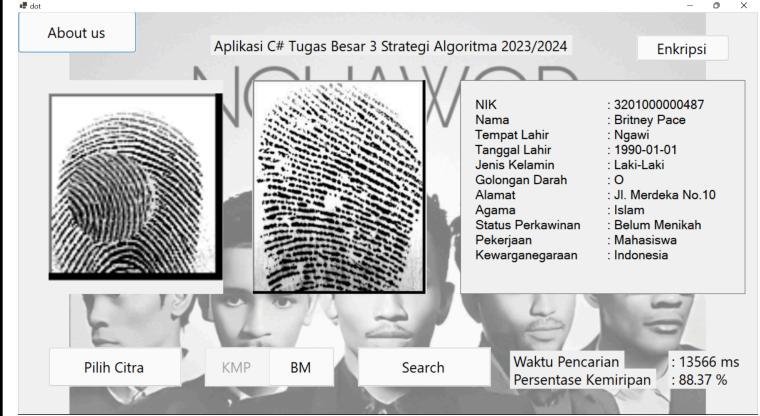
Kasus uji/citra yang ingin dicari	Tangkapan layar hasil pengujian																						
264_M_Right_index_finger	<p>A screenshot of a Windows application window titled "Aplikasi C# Tugas Besar 3 Strategi Algoritma 2023/2024". The window displays two fingerprint images side-by-side. To the right of the images is a table of search results:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr><td>NIK</td><td>: 3201000000487</td></tr> <tr><td>Nama</td><td>: Britney Pace</td></tr> <tr><td>Tempat Lahir</td><td>: Ngawi</td></tr> <tr><td>Tanggal Lahir</td><td>: 1990-01-01</td></tr> <tr><td>Jenis Kelamin</td><td>: Laki-Laki</td></tr> <tr><td>Golongan Darah</td><td>: O</td></tr> <tr><td>Alamat</td><td>: Jl. Merdeka No.10</td></tr> <tr><td>Agama</td><td>: Islam</td></tr> <tr><td>Status Perkawinan</td><td>: Belum Menikah</td></tr> <tr><td>Pekerjaan</td><td>: Mahasiswa</td></tr> <tr><td>Kewarganegaraan</td><td>: Indonesia</td></tr> </table> <p>Below the results, there are four buttons: "Pilih Citra", "KMP", "BM", and "Search". To the right of these buttons are performance metrics: "Waktu Pencarian : 10083 ms" and "Persentase Kemiripan : 88.32 %".</p>	NIK	: 3201000000487	Nama	: Britney Pace	Tempat Lahir	: Ngawi	Tanggal Lahir	: 1990-01-01	Jenis Kelamin	: Laki-Laki	Golongan Darah	: O	Alamat	: Jl. Merdeka No.10	Agama	: Islam	Status Perkawinan	: Belum Menikah	Pekerjaan	: Mahasiswa	Kewarganegaraan	: Indonesia
NIK	: 3201000000487																						
Nama	: Britney Pace																						
Tempat Lahir	: Ngawi																						
Tanggal Lahir	: 1990-01-01																						
Jenis Kelamin	: Laki-Laki																						
Golongan Darah	: O																						
Alamat	: Jl. Merdeka No.10																						
Agama	: Islam																						
Status Perkawinan	: Belum Menikah																						
Pekerjaan	: Mahasiswa																						
Kewarganegaraan	: Indonesia																						

164__M_Left_thumb_finger	
566__M_Right_index_finger	

- Pengujian gambar yang berubah

Tabel 4.4 Hasil Pengujian Hamming Distansce dengan Gambar yang Berubah

Kasus uji/citra yang ingin dicari	Tangkapan layar hasil pengujian
228__M_Left_index_finger_CR (Alter easy)	

172_M_Left_ring_finger_CR (Alter Medium)	 <p>A screenshot of a C# application window titled "Aplikasi C# Tugas Besar 3 Strategi Algoritma 2023/2024". It displays two fingerprint images side-by-side. Below the images are four buttons: "Pilih Citra", "KMP", "BM", and "Search". To the right of the buttons is a table of search results:</p> <table border="1"> <tbody> <tr><td>NIK</td><td>: 3201000000487</td></tr> <tr><td>Nama</td><td>: Britney Pace</td></tr> <tr><td>Tempat Lahir</td><td>: Ngawi</td></tr> <tr><td>Tanggal Lahir</td><td>: 1990-01-01</td></tr> <tr><td>Jenis Kelamin</td><td>: Laki-Laki</td></tr> <tr><td>Golongan Darah</td><td>: O</td></tr> <tr><td>Alamat</td><td>: Jl. Merdeka No.10</td></tr> <tr><td>Agama</td><td>: Islam</td></tr> <tr><td>Status Perkawinan</td><td>: Belum Menikah</td></tr> <tr><td>Pekerjaan</td><td>: Mahasiswa</td></tr> <tr><td>Kewarganegaraan</td><td>: Indonesia</td></tr> </tbody> </table> <p>At the bottom right, it shows "Waktu Pencarian : 13566 ms" and "Persentase Kemiripan : 88.37 %".</p>	NIK	: 3201000000487	Nama	: Britney Pace	Tempat Lahir	: Ngawi	Tanggal Lahir	: 1990-01-01	Jenis Kelamin	: Laki-Laki	Golongan Darah	: O	Alamat	: Jl. Merdeka No.10	Agama	: Islam	Status Perkawinan	: Belum Menikah	Pekerjaan	: Mahasiswa	Kewarganegaraan	: Indonesia
NIK	: 3201000000487																						
Nama	: Britney Pace																						
Tempat Lahir	: Ngawi																						
Tanggal Lahir	: 1990-01-01																						
Jenis Kelamin	: Laki-Laki																						
Golongan Darah	: O																						
Alamat	: Jl. Merdeka No.10																						
Agama	: Islam																						
Status Perkawinan	: Belum Menikah																						
Pekerjaan	: Mahasiswa																						
Kewarganegaraan	: Indonesia																						
5_M_Left_ring_finger_Obl (Alter Hard)	 <p>A screenshot of a C# application window titled "Aplikasi C# Tugas Besar 3 Strategi Algoritma 2023/2024". It displays two fingerprint images side-by-side. Below the images are four buttons: "Pilih Citra", "KMP", "BM", and "Search". To the right of the buttons is a table of search results:</p> <table border="1"> <tbody> <tr><td>NIK</td><td>: 3201000000487</td></tr> <tr><td>Nama</td><td>: Britney Pace</td></tr> <tr><td>Tempat Lahir</td><td>: Ngawi</td></tr> <tr><td>Tanggal Lahir</td><td>: 1990-01-01</td></tr> <tr><td>Jenis Kelamin</td><td>: Laki-Laki</td></tr> <tr><td>Golongan Darah</td><td>: O</td></tr> <tr><td>Alamat</td><td>: Jl. Merdeka No.10</td></tr> <tr><td>Agama</td><td>: Islam</td></tr> <tr><td>Status Perkawinan</td><td>: Belum Menikah</td></tr> <tr><td>Pekerjaan</td><td>: Mahasiswa</td></tr> <tr><td>Kewarganegaraan</td><td>: Indonesia</td></tr> </tbody> </table> <p>At the bottom right, it shows "Waktu Pencarian : 17423 ms" and "Persentase Kemiripan : 88.03 %".</p>	NIK	: 3201000000487	Nama	: Britney Pace	Tempat Lahir	: Ngawi	Tanggal Lahir	: 1990-01-01	Jenis Kelamin	: Laki-Laki	Golongan Darah	: O	Alamat	: Jl. Merdeka No.10	Agama	: Islam	Status Perkawinan	: Belum Menikah	Pekerjaan	: Mahasiswa	Kewarganegaraan	: Indonesia
NIK	: 3201000000487																						
Nama	: Britney Pace																						
Tempat Lahir	: Ngawi																						
Tanggal Lahir	: 1990-01-01																						
Jenis Kelamin	: Laki-Laki																						
Golongan Darah	: O																						
Alamat	: Jl. Merdeka No.10																						
Agama	: Islam																						
Status Perkawinan	: Belum Menikah																						
Pekerjaan	: Mahasiswa																						
Kewarganegaraan	: Indonesia																						

D. Analisis Hasil Pengujian

Berikut kami sajikan tabel perbandingan performa dari masing-masing algoritma

Tabel 4.5 Tabel Perbandingan Hasil Pengujian Algoritma KMP dengan BM

Kasus Uji	Waktu Eksekusi (ms)	
	KMP	BM
100_M_Right_thumb_finger.BMP	38	23
371_M_Right_thumb_finger.BMP	4037	4695
9_M_Right_thumb_finger.BMP	7474	10175
138_M_Right_thumb_finger_Zcut (Alter easy)	826	1106

446_M_Right_thumb _finger_Zcut (Alter hard)	8574	8589
---	------	------

Berdasarkan tabel diatas, dapat diketahui bahwa algoritma KMP sedikit lebih cepat dibandingkan algoritma BM pada kasus pencocokan sidik jari ini. Hal ini disebabkan algoritma KMP memiliki kompleksitas waktu $O(m+n)$ sedangkan algoritma BM memiliki kompleksitas waktu $O(mn)$. Selain itu, Waktu yang dibutuhkan untuk melakukan pattern matching berbanding lurus dengan letak data dalam database. Semakin dalam letak data maka semakin lama pula waktu yang dibutuhkan untuk mencari sidik jari yang cocok.

Apabila tidak menemukan pola yang sesuai dengan algoritma KMP atau BM, maka akan dicari persentase kemiripan menggunakan algoritma hamming distance dengan minimal kemiripan diatas 80%. Berdasarkan tabel 4.3, dapat dilihat bahwa masukan yang diberikan tidak ada pada database sehingga diberikan keluaran gambar sidik jari yang paling mirip. Dapat dilihat bahwa gambar sidik jari yang ditampilkan berbeda dengan sidik jari yang dimasukan, tetapi tingkat kemiripannya diatas 80% sehingga tetap ditampilkan. Apabila di dalam database tidak ada sidik jari yang memiliki tingkat kemiripan diatas 80%, maka tidak ada data yang akan ditampilkan.

BAB V

KESIMPULAN DAN SARAN

A. Kesimpulan

Algoritma KMP dan BM merupakan salah dua algoritma pattern matching. Salah satu implementasi dari algoritma ini adalah penggunaannya dalam membangun sistem deteksi individu berbasis biometrik melalui citra sidik jari, dan pada tugas besar ini, telah ditunjukkan bahwa algoritma dapat menyelesaikan persoalan tersebut. Beberapa kekurangan mungkin terdapat pada program yang kami buat. Namun, meskipun demikian, kami yakin program kami dapat bekerja sesuai dengan kasus uji yang diberikan. Secara teoretis, algoritma KMP dapat memproses pola sidik jari lebih cepat daripada BM, dan berdasarkan hasil pengujian yang kami lakukan, hal tersebut benar.

B. Saran

Terdapat beberapa saran dalam penggerjaan tugas besar selanjutnya:

- Dalam mengerjakan bonus enkripsi, dapat digunakan algoritma yang lebih aman daripada Caesar Cipher

C. Tanggapan dan Refleksi

Segala puji bagi Allah Subhanahu wa Ta'ala, karena atas rahmat dan karunia-Nya, kami dapat menyelesaikan Tugas Besar 3 IF2211 Strategi Algoritma dengan judul Pemanfaatan Pattern Matching dalam Membangun Sistem Deteksi Individu Berbasis Biometrik Melalui Citra Sidik Jari. Menurut kami, tugas besar ini sangat membantu dalam memahami algoritma pattern matching serta meningkatkan pengetahuan terkait implementasinya.

Berikut adalah tanggapan dari masing-masing anggota dalam kelompok ini:

- Ihsan: I love Visual Studio <3
- Panji: Visual Studio or Visual Studio Code ? Vim : Nah, I'd win
- Imam: Visual Studio Code Supremacy

LAMPIRAN

- Tautan repository Github : https://github.com/PanjiSri/Tubes3_let-me-seedik
- Tautan video : <https://youtu.be/VgNPApkUac>

DAFTAR PUSTAKA

Levitin, Anany. *Introduction to the Design & Analysis of Algorithms*. Addison-Wesley, 2003.

Munir, Rinaldi. *Pencocokan String 2021*. Diakses pada 9 Juni 2024.

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

Munir, Rinaldi. *String Matching dengan Regular Expression 2019*. Diakses pada 9 Juni 2024.

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2022-2023/String-Matching-dengan-Regex-2019.pdf>