

Institut Teknologi Bandung
Sekolah Teknik Elektro dan Informatika
Program Studi Informatika Semester II 2023/2024



Laporan Tugas Besar III

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

IF2211 Strategi Algoritma

Shazya Audrea Taufik K01 - 13522063

Zahira Dina Amalia K01 - 13522085

Muhammad Neo Cicero Koda K02 - 13522108

Daftar Isi

BAB I.....	3
DESKRIPSI TUGAS.....	3
1.1 Deskripsi Tugas.....	3
1.2 Spesifikasi.....	4
BAB II.....	6
LANDASAN TEORI.....	6
2. 1 Deskripsi Singkat Algoritma.....	6
2. 1. 1 Algoritma Knuth-Morris-Pratt (KMP).....	6
Gambar 2.1.1.1 Ilustrasi Algoritma KMP.....	6
2. 1. 2 Algoritma Boyer-Moore (BM).....	8
Gambar 2.1.2.1 Ilustrasi Algoritma BM.....	8
2. 1. 3 Regex.....	10
2. 2 Penjelasan Teknik Pengukuran Persentase Kemiripan.....	11
2. 3 Penjelasan Singkat Mengenai Aplikasi Desktop yang Dibangun.....	12
BAB III.....	14
ANALISIS PEMECAHAN MASALAH.....	14
3. 1 Langkah-Langkah Pemecahan Masalah.....	14
3. 2 Proses Pemetaan Elemen dan Penyelesaian Solusi dengan Algoritma KMP dan BM.....	15
3. 2. 1 Pemetaan Elemen dan Penyelesaian Solusi dengan Algoritma KMP.....	16
3. 2. 2 Pemetaan Elemen dan Penyelesaian Solusi dengan Algoritma BM.....	17
3. 3 Fitur Fungsional dan Arsitektur Aplikasi Desktop yang Dibangun.....	18
3. 3. 1. Fitur Fungsional Aplikasi Desktop.....	18
3. 3. 2. Arsitektur Aplikasi Desktop.....	19
3. 4 Contoh Ilustrasi Kasus.....	21
BAB IV.....	25
IMPLEMENTASI DAN PENGUJIAN.....	25
4. 1 Spesifikasi Teknis Program.....	25
4. 1. 1 Struktur Data.....	25
4. 1. 2 Fungsi dan Prosedur.....	25
4. 2 Penjelasan Tata Cara Penggunaan Program.....	63
4. 3 Hasil pengujian.....	65
4. 4 Analisis Hasil Pengujian.....	77
4. 5 Implementasi Bonus.....	77
BAB V.....	80
KESIMPULAN DAN SARAN.....	80
5. 1 Kesimpulan.....	80
5. 2 Saran, Tanggapan, dan Refleksi.....	80
LAMPIRAN.....	81
DAFTAR PUSTAKA.....	82

BAB I

DESKRIPSI TUGAS

1.1 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 dalam basis data. 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, diimplementasikan 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.

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



Gambar 1.1. Proses konversi pencocokan sidik jari

1.2 Spesifikasi

Pada tugas besar ini, dibuat sebuah sistem yang dapat melakukan identifikasi individu berbasis biometrik dengan menggunakan sidik jari dan detail sebagai berikut.

- 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 penggeraan.
- 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).
- 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.
- Program memiliki keluaran yang minimal mengandung seluruh data yang terdapat pada contoh antarmuka pada bagian penggunaan program.
- Pengguna dapat memilih algoritma yang ingin digunakan antara KMP atau BM.

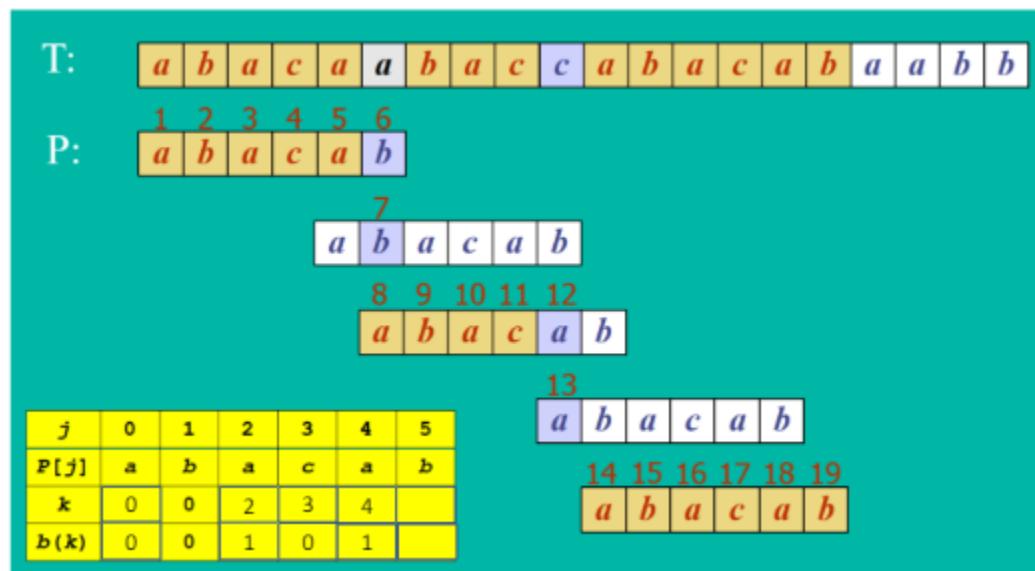
- Biodata yang ditampilkan harus biodata yang memiliki nama yang benar (gunakan Regex untuk memperbaiki nama yang rusak dan gunakan KMP atau BM untuk mencari orang yang paling sesuai).
- Program memiliki antarmuka yang user-friendly. Anda juga dapat menambahkan fitur lain untuk menunjang program yang Anda buat (unsur kreativitas).

BAB II

LANDASAN TEORI

2. 1 Deskripsi Singkat Algoritma

2. 1. 1 Algoritma Knuth-Morris-Pratt (KMP)



Gambar 2.1.1.1 Ilustrasi Algoritma KMP

Algoritma Knuth-Morris-Pratt (KMP) adalah algoritma pencocokan string yang digunakan untuk menemukan pola dalam teks. Algoritma ini menghindari perbandingan yang tidak perlu dengan cara memanfaatkan informasi dari perbandingan sebelumnya. Proses pencocokan dilakukan dari kiri ke kanan, mirip dengan algoritma brute force, tetapi dengan pergeseran yang lebih baik saat terjadi ketidaksesuaian atau *mismatch* antara teks dan pola. Algoritma KMP menggunakan fungsi pinggiran (*border function*) yang disebut juga dengan fungsi gagal (*failure function*) untuk menentukan seberapa jauh pola dapat digeser saat terjadi ketidaksesuaian. Fungsi ini memproses pola untuk menemukan kecocokan antara awalan (prefix) dan akhiran (suffix) pola itu sendiri. Berikut logika dalam pembuatan fungsi pinggiran tersebut.

```
ALGORITMA ComputeBorder(pattern)
Input: pattern - string
Output: border - array of integers
```

```

m ← panjang(pattern)
border ← array of integers dengan panjang m
border[0] ← 0
j ← 0

UNTUK i dari 1 hingga m - 1 lakukan
    SELAMA j > 0 DAN pattern[i] ≠ pattern[j] lakukan
        j ← border[j - 1]
    AKHIRI SELAMA
    JIKA pattern[i] = pattern[j] maka
        j ← j + 1
    AKHIRI JIKA
    border[i] ← j
AKHIRI UNTUK

kembalikan border
AKHIRI ComputeBorder

```

Adapun untuk pencarian dengan algoritma KMP sebagai berikut,

```

ALGORITMA KMPMatch(text, pattern)
    Input: text - string, pattern - string
    Output: index dari kemunculan pertama pola dalam teks atau -1 jika
            tidak ditemukan

    n ← panjang(text)
    m ← panjang(pattern)
    border ← ComputeBorder(pattern)
    j ← 0

    UNTUK i dari 0 hingga n - 1 lakukan
        SELAMA j > 0 DAN text[i] ≠ pattern[j] lakukan
            j ← border[j - 1]
        AKHIRI SELAMA
        JIKA text[i] = pattern[j] maka

```

```

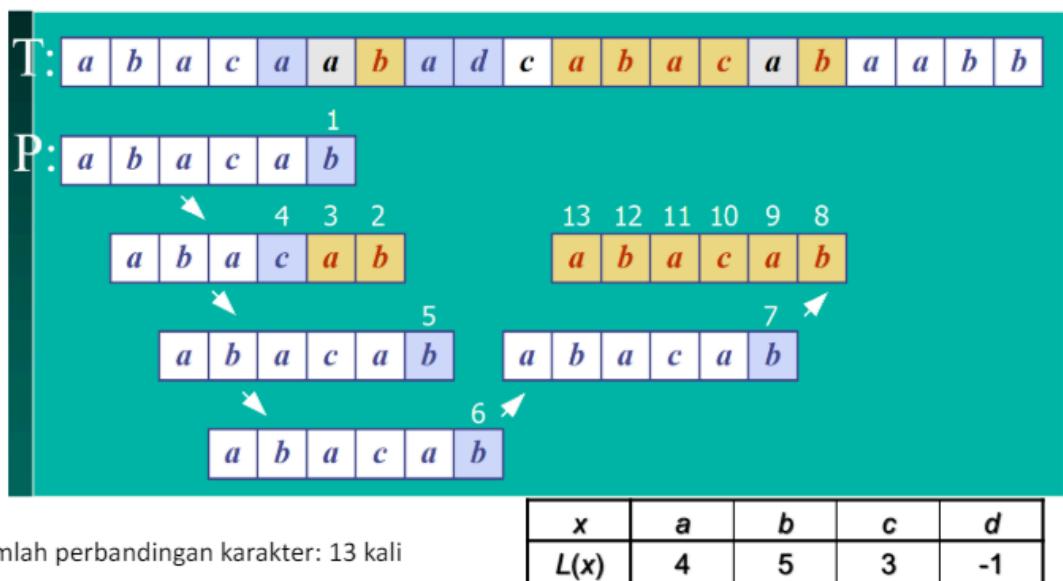
j ← j + 1
AKHIRI JIKA
JIKA j = m maka
    kembalikan i - m + 1
AKHIRI JIKA
AKHIRI UNTUK

kembalikan -1
AKHIRI KMPMatch

```

Kompleksitas waktu dari algoritma KMP adalah $O(m+n)$, di mana m adalah panjang pola dan n adalah panjang teks, membuatnya jauh lebih cepat dibandingkan brute force pada kasus rata-rata dan terburuk.

2. 1. 2 Algoritma Boyer-Moore (BM)



Gambar 2.1.2.1 Ilustrasi Algoritma BM

Algoritma Boyer-Moore (BM) adalah algoritma pencocokan string yang berjalan lebih efisien pada teks dengan alfabet besar. Algoritma ini menggunakan dua teknik utama: teknik kaca spion (*looking-glass technique*) dan teknik loncatan karakter (*character-jump technique*). Teknik kaca spion melakukan pencocokan dari belakang ke depan pada pola, dimulai dari karakter terakhir. Teknik loncatan karakter menggeser

pola ke kanan berdasarkan kemunculan terakhir karakter yang tidak cocok dalam pola. Fungsi kemunculan terakhir (*last occurrence function*) dibangun selama preprocessing untuk setiap karakter dalam alfabet, memetakan indeks kemunculan terakhir karakter tersebut dalam pola. Berikut algoritma pembentukan fungsi *last occurrence* tersebut.

```

ALGORITMA BuildLast(pattern)
    Input: pattern - string
    Output: last - array of integers

    last ← array of integers dengan panjang 128 (ASCII charset)
    UNTUK i dari 0 hingga 127 lakukan
        last[i] ← -1
    AKHIRI UNTUK

    UNTUK i dari 0 hingga panjang(pattern) - 1 lakukan
        last[pattern[i]] ← i
    AKHIRI UNTUK

    kembalikan last
AKHIRI BuildLast

```

Adapun untuk algoritma pencarian pattern dengan Boyer-Moore (BM),

```

ALGORITMA BMMatch(text, pattern)
    Input: text - string, pattern - string
    Output: index dari kemunculan pertama pola dalam teks atau -1 jika
            tidak ditemukan

    last ← BuildLast(pattern)
    n ← panjang(text)
    m ← panjang(pattern)
    i ← m - 1
    j ← m - 1

    SELAMA i < n lakukan
        JIKA pattern[j] = text[i] maka

```

```

JIKA j = 0 maka
    kembalikan i
AKHIRI JIKA
i ← i - 1
j ← j - 1
LAINNYA
lo ← last[text[i]]
i ← i + m - MIN(j, 1 + lo)
j ← m - 1
AKHIRI JIKA
AKHIRI SELAMA

kembalikan -1
AKHIRI BMMatch

```

Kompleksitas waktu terburuk dari algoritma Boyer-Moore adalah $O(nm + A)$, di mana A adalah ukuran alfabet, tetapi dalam praktiknya, algoritma ini sangat cepat untuk teks dengan alfabet besar seperti teks bahasa Inggris.

2. 1. 3 Regex

Regex (Regular Expressions) adalah teknik untuk pencocokan pola menggunakan ekspresi reguler. Ekspresi reguler adalah urutan karakter yang mendefinisikan pola pencarian. Regex digunakan secara luas dalam berbagai aplikasi seperti editor teks, mesin pencari web, dan analisis data. Regex bekerja dengan membangun mesin keadaan hingga deterministik (DFA) atau nondeterministik (NFA) yang memproses teks untuk mencari pola yang sesuai dengan ekspresi reguler. Kompleksitas waktu pencocokan pola dengan ekspresi reguler tergantung pada implementasi spesifik dari mesin keadaan dan panjang serta kerumitan ekspresi reguler. Regex sangat fleksibel dan kuat, tetapi bisa menjadi kurang efisien untuk pola yang sangat kompleks atau teks yang sangat panjang. Regex yang digunakan pada program ini adalah sebagai berikut.

```

string fixedText = Regex.Replace(text, "[143678059]", match =>
    numberSubs[match.Value[0]].ToString());

```

2.2 Penjelasan Teknik Pengukuran Persentase Kemiripan

Pengukuran persentase kemiripan antara dua string atau teks dapat dilakukan dengan beberapa metode, salah satunya adalah menggunakan algoritma pencocokan string seperti KMP, BM, atau regex. Selain itu, ada juga metode yang lebih umum digunakan untuk mengukur kemiripan, seperti Levenshtein Distance. Algoritma ini mengukur jarak edit antara dua string, yaitu jumlah operasi (penyisipan, penghapusan, atau substitusi) yang diperlukan untuk mengubah satu string menjadi string lainnya.

Metode-metode ini dapat digunakan untuk berbagai aplikasi seperti pengenalan teks, pemrosesan bahasa alami, dan deteksi plagiarisme, dengan memilih metode yang paling sesuai berdasarkan kebutuhan spesifik dan sifat data yang dianalisis. Pada permasalahan ini, kami menggunakan levenshtein distance sebagai perhitungan kesamaan antara dua teks. Adapun untuk algoritma pencarian pattern dengan Levenshtein Distance adalah sebagai berikut.

```
Fungsi LevenshteinDistance(s1, s2)
    m = panjang(s1)
    n = panjang(s2)
    Buat matriks D berukuran (m+1) x (n+1)

    Untuk i dari 0 sampai m
        D[i, 0] = i
    Akhir Untuk

    Untuk j dari 0 sampai n
        D[0, j] = j
    Akhir Untuk

    Untuk i dari 1 sampai m
        Untuk j dari 1 sampai n
            Jika s1[i-1] = s2[j-1] maka
                biaya = 0
            Tetapi
                biaya = 1
            Akhir Jika
```

```

D[i, j] = minimum(D[i-1, j] + 1, D[i, j-1] + 1,
                    D[i-1, j-1] + biaya)

Akhir Untuk
Akhir Untuk
Kembalikan D[m, n]
Akhir Fungsi

```

Penjelasan singkat dari algoritma di atas adalah sebagai berikut.

- **Inisialisasi Matriks:** Matriks D diinisialisasi dengan ukuran $(m+1) \times (n+1)$. Baris pertama dan kolom pertama diisi dengan nilai indeks masing-masing, yang mewakili jarak dari string kosong ke substring yang bersangkutan.
- **Pengisian Matriks:** Dua loop bersarang digunakan untuk mengisi matriks. Loop luar melintasi setiap karakter dalam s1, dan loop dalam melintasi setiap karakter dalam s2.
- **Menentukan Biaya:** Jika karakter yang dibandingkan sama, biayanya 0; jika berbeda, biayanya 1.
- **Menghitung Nilai Matriks:** Nilai untuk $D[i, j]$ dihitung sebagai minimum dari nilai atas, kiri, dan diagonal kiri atas ditambah biaya.
- **Nilai Akhir:** Nilai $D[m, n]$ merupakan Levenshtein Distance antara s1 dan s2.

Dengan cara ini, Levenshtein Distance dapat digunakan untuk mengukur seberapa mirip dua string dengan menghitung jumlah operasi minimum yang diperlukan untuk mengubah satu string menjadi string lainnya. Persentase kesamaan dapat dihitung sebagai kebalikan dari jarak edit relatif terhadap panjang string yang lebih panjang.

2.3 Penjelasan Singkat Mengenai Aplikasi Desktop yang Dibangun

Aplikasi desktop yang dibangun adalah **Fingerprint Checker by Finggerfingeran**, sebuah aplikasi berbasis WPF (Windows Presentation Foundation) yang dirancang untuk mencocokkan gambar sidik jari dengan basis data gambar yang ada. Pengguna dapat memilih gambar sidik jari dari komputer mereka menggunakan `OpenFileDialog`, yang kemudian akan ditampilkan dalam UI aplikasi. Aplikasi ini menawarkan dua algoritma pencocokan string, yaitu KMP (Knuth-Morris-Pratt) dan BM (Boyer-Moore), yang dapat dipilih oleh pengguna untuk mencari kecocokan gambar sidik jari dengan gambar-gambar dalam basis data. Proses pencarian

dilakukan secara paralel untuk meningkatkan kecepatan pencocokan. Hasil pencocokan terbaik, berdasarkan posisi kecocokan atau kemiripan Levenshtein, akan ditampilkan kepada pengguna. Jika kecocokan ditemukan, aplikasi akan menampilkan informasi biodata yang terkait dengan gambar sidik jari yang cocok, termasuk NIK, nama, tempat/tanggal lahir, golongan darah, jenis kelamin, alamat, agama, status perkawinan, status pekerjaan, dan kewarganegaraan.

Komponen utama aplikasi ini meliputi `MainWindow.xaml.cs`, yang mengelola logika interaksi pengguna dan pengolahan data utama, serta menangani pemilihan gambar, pemilihan algoritma, dan proses pencarian kecocokan. Layout UI untuk aplikasi, termasuk kontrol untuk pemilihan gambar, tombol pencarian, dan tampilan hasil, didefinisikan dalam `MainWindow.xaml`. Metode untuk mengkonversi gambar ke representasi biner dan ASCII serta memproses blok ASCII pusat dari gambar untuk digunakan dalam pencocokan terdapat dalam `ImageProcessor.cs`. Interaksi dengan basis data, termasuk pengambilan jalur gambar dan informasi biodata, dikelola oleh `DatabaseManager.cs`. Algoritma Levenshtein untuk menghitung jarak edit dan kesamaan antara dua string diimplementasikan dalam `Levenshtein.cs`, sementara algoritma pencocokan string KMP dan BM diimplementasikan dalam `KMP.cs` dan `BoyerMoore.cs`.

BAB III

ANALISIS PEMECAHAN MASALAH

3.1 Langkah-Langkah Pemecahan Masalah

Pemecahan masalah dilakukan dengan membagi-bagi permasalahan menjadi permasalahan-permasalahan yang lebih kecil. Pada program ini, diperlukan langkah dimana citra sidik jari diproses menjadi biner untuk kemudian dikonversi ke ASCII agar dapat lebih mudah diproses. Diperlukan juga langkah dimana ASCII kemudian diproses berdasarkan algoritma yang telah dipilih, antara algoritma KMP dan BM. Setelah diproses, akan didapatkan evaluasi dari kesamaan, antara tidak sama atau sama atau memiliki kemiripan tertentu (dengan batasan persentase yang telah ditetapkan 75%). Selain itu, diperlukan langkah untuk memperbaiki nama yang *corrupt* pada database. Secara rincinya, berikut langkah-langkah pemecahan masalah pada program ini.

1. Pengolahan Citra Sidik Jari

- **Input Citra:** Pengguna memasukkan citra sidik jari mereka melalui aplikasi. Citra ini bisa berasal dari scanner sidik jari atau file gambar yang sudah ada.
- **Pra-pemrosesan:** Citra akan di-preprocess untuk meningkatkan kualitas dan konsistensi sidik jari, termasuk normalisasi, binerisasi, dan penghapusan noise.
- **Ekstraksi Fitur:** Sidik jari diubah menjadi representasi biner, yang melibatkan konversi citra menjadi kumpulan piksel hitam dan putih untuk menggambarkan ridge dan valley dari sidik jari.

2. Konversi Biner ke ASCII

- **Pembentukan Blok Biner:** String biner dari proses sebelumnya dibagi menjadi blok-blok yang lebih kecil untuk memudahkan pemrosesan.
- **Konversi ke ASCII:** Setiap blok biner kemudian dikonversi menjadi karakter ASCII. Hal ini dilakukan dengan mengambil setiap baris 8-bit dari string biner dan mengubahnya menjadi karakter ASCII yang sesuai, memudahkan pencocokan pola karena karakter ASCII lebih mudah untuk dikelola dan membandingkan dibandingkan dengan string biner.

3. Pencocokan Sidik Jari

- **Algoritma Pencocokan:** Menggunakan dua algoritma pencocokan string, yaitu Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM):
 - **KMP:** Efektif untuk pencarian cepat berbasis prefix dalam string, dimana sidik jari yang diinput dicocokkan melawan database sidik jari menggunakan informasi tentang prefix yang terulang.
 - **BM:** Memanfaatkan aturan bad character shift untuk melompati sebagian besar tidak cocoknya karakter, membuatnya lebih cepat dalam pencocokan dengan text yang lebih panjang.
- **Evaluasi Kesamaan:** Pencocokan sidik jari dievaluasi berdasarkan tingkat kemiripan yang dihasilkan oleh algoritma, memungkinkan identifikasi meski adanya perbedaan kecil antara sampel sidik jari.

4. Koreksi Nama dengan Bahasa Alay

- **Deteksi dan Koreksi Nama Alay:** Data nama yang rusak atau disimpan dalam bentuk bahasa alay akan dikoreksi menggunakan ekspresi reguler. Nama dalam database yang mengandung angka atau simbol yang tidak sesuai akan disesuaikan kembali ke bentuk aslinya.
- **Penyesuaian dengan Levenshtein Distance:** Untuk memastikan keakuratan nama yang diperbaiki, Levenshtein Distance digunakan untuk menemukan kemiripan string terdekat antara input yang diperbaiki dan nama asli dalam database.
- **Integrasi Data:** Nama yang sudah dikoreksi dan terverifikasi kemiripannya kemudian akan dihubungkan kembali ke data individu terkait dalam database, memastikan bahwa data yang ditampilkan adalah akurat dan relevan.

Langkah-langkah ini memastikan bahwa sistem dapat secara efisien dan akurat mengidentifikasi individu berdasarkan sidik jari, dengan kemampuan tambahan untuk menangani input nama yang rusak atau tidak standar.

3.2 Proses Pemetaan Elemen dan Penyelesaian Solusi dengan Algoritma KMP dan BM

3. 2. 1 Pemetaan Elemen dan Penyelesaian Solusi dengan Algoritma KMP

Dalam algoritma KMP Algoritma Knuth-Morris-Pratt (KMP) digunakan untuk mencari keberadaan sebuah "word" dalam "text" dengan efisien. Ini bekerja dengan menghindari pengulangan pengecekan karakter yang sudah sesuai. KMP melakukan ini dengan memanfaatkan informasi yang sudah diketahui dari pencocokan sebelumnya melalui *longest prefix* yang juga merupakan suffix (LPS) untuk menghindari perbandingan yang tidak perlu saat terjadi mismatch. Berikut ini langkah-langkah algoritma KMP:

→ Pemetaan Elemen

KMP memproses pattern untuk menciptakan sebuah array yang dikenal sebagai LPS (*Longest Prefix which is also Suffix*). Array LPS digunakan untuk menentukan seberapa jauh kita harus "melompat" dalam pencocokan, sehingga menghindari perbandingan ulang karakter yang sudah diketahui tidak cocok. Berikut langkah-langkah pada pembuatan Array LPS:

1. Inisialisasi: Array LPS diinisialisasi dimana `lps[0]` selalu 0 karena satu karakter tidak memiliki prefix yang juga adalah suffix.
2. Iterasi Melalui Pattern: Mulai dari indeks kedua pattern, kita cek berulang kali untuk mencari prefix terpanjang yang juga bisa menjadi suffix.
 - Jika karakter pada indeks saat ini (`pattern[i]`) cocok dengan `pattern[length]`, maka `lps[i]` diatur menjadi `length+1` dan kita lanjutkan ke karakter berikutnya.
 - Jika tidak cocok, kita update `length` menjadi `lps[length-1]` untuk mencoba prefix yang lebih pendek. Proses ini diulangi sampai kita menemukan kecocokan atau `length` menjadi 0.

→ Penyelesaian Solusi

1. **Iterasi Melalui Teks:** Dimulai dari awal teks, kita bandingkan karakter demi karakter dengan pattern menggunakan indeks `i` untuk teks dan `j` untuk pattern.
2. **Pengecekan Kecocokan:**
 - Jika `pattern[j]` cocok dengan `text[i]`, kedua indeks diincrement untuk melanjutkan pencocokan ke karakter selanjutnya.
 - Jika seluruh karakter di pattern telah cocok (`j == m`), maka kita telah menemukan kecocokan, dan fungsi mengembalikan indeks dimulainya kecocokan di teks.

3. **Handling Mismatch:**
 - Jika terjadi mismatch (`pattern[j]` tidak sama dengan `text[i]`) dan `j` tidak 0, kita set `j` ke `lps[j-1]` yang menunjukkan indeks berikutnya dari pattern yang harus dicocokkan.
 - Jika `j` adalah 0 (tidak ada kecocokan sama sekali pada awal pattern), maka hanya `i` yang di-*increment* untuk melanjutkan pencarian di teks.
4. **Kinerja Algoritma:** Berbeda dengan brute force, KMP menghindari pencocokan ulang karakter yang telah diketahui tidak akan membawa kecocokan, sehingga mengurangi jumlah perbandingan secara signifikan.

3. 2. 2 Pemetaan Elemen dan Penyelesaian Solusi dengan Algoritma BM

Dalam hal ini dikenal BM lebih cepat dari KMP karena bergerak lebih jauh dari karakter yang tidak cocok. BM menggunakan heuristik Heuristic Bad Character: Menyatakan seberapa jauh kita dapat melompat jika ada ketidakcocokan karakter. Pencarian dimulai dari akhir pattern, membandingkan karakter dengan text dan melompat menggunakan informasi dari tabel pergeseran saat ditemukan ketidakcocokan. Berikut adalah pemetaan dan langkah-langkah algoritma BM:

→ Pemetaan Elemen

Algoritma BM menggunakan dua tabel utama untuk mengoptimalkan pencocokan: **Bad Character Shift** atau yang disebutkan pada perkuliahan yakni **Last Occurrence Table**. Dalam program ini, **Bad Character Shift** fokus pada penggunaan karakter yang tidak cocok untuk menentukan seberapa jauh pattern harus digeser. **Pembuatan Tabel Bad Character Shift** adalah sebagai berikut.

1. **Inisialisasi Tabel:** Tabel `badCharShift` diinisialisasi dengan ukuran abjad (256 untuk ASCII), di mana setiap elemen diatur dengan panjang pattern. Ini menunjukkan jarak maksimum yang mungkin digeser jika terjadi mismatch.
2. **Pengisian Tabel:** Tabel diisi dengan mengatur indeks terakhir kemunculan setiap karakter di pattern. Jika karakter pada indeks `i` dari pattern, `badCharShift` untuk karakter tersebut diatur ke `pattern.Length - 1 - i`. Ini berarti jika karakter tersebut tidak cocok, pattern akan digeser sedemikian rupa sehingga karakter mismatch di text sejajar dengan terakhir kali karakter tersebut muncul di pattern.

→ Penyelesaian Solusi

Proses pencarian atau pencocokan dilakukan sebagai berikut:

1. **Iterasi Melalui Text:** Algoritma mulai dari awal text dan mencoba membandingkan pattern dari akhir ke awal (menggunakan teknik looking-glass).
2. **Perbandingan Karakter:** Jika karakter pada `text[i+j]` sama dengan `pattern[j]`, perbandingan terus dilakukan ke arah kiri. Jika semua karakter cocok (`skip` tetap 0 setelah loop), maka indeks `i` (posisi start kecocokan di text) akan dikembalikan.
3. **Handling Mismatch:** Jika terjadi mismatch pada `text[i+j]`, maka `skip` diatur berdasarkan tabel `badCharShift`. Nilai `skip` adalah selisih antara posisi karakter di text dan nilai yang tercatat di tabel `badCharShift` untuk karakter tersebut. Ini memastikan bahwa pada iterasi berikutnya, karakter di text akan sejajar dengan terakhir kali karakter tersebut muncul di pattern.
4. **Geser Pattern:** Pattern digeser sesuai dengan nilai `skip` yang telah ditentukan, dan pencarian dilanjutkan sampai pattern tidak bisa digeser lagi atau seluruh text telah dicocokkan.

3. 3 Fitur Fungsional dan Arsitektur Aplikasi Desktop yang Dibangun

3. 3. 1. Fitur Fungsional Aplikasi Desktop

Pada aplikasi desktop yang dibangun terdapat beberapa fitur fungsional yang dapat dilakukan. Beberapa di antaranya adalah sebagai berikut.

- Fitur pemilihan dan meng-*upload* sebuah gambar yang hendak dibandingkan
Pada fitur ini, pengguna dapat melakukan pemilihan gambar dari Windows Explorer. Gambar yang dapat diterima berupa gambar-gambar dengan format “jpg”, “jpeg”, “png”, “bmp”. Gambar ini kemudian akan ditampilkan pada aplikasi sebagai gambar yang dipilih untuk dibandingkan.
- Fitur pemilihan algoritma
Fitur ini akan menentukan algoritma pencarian yang digunakan. Terdapat dua algoritma pilihan yang tersedia pada program ini, yaitu algoritma KMP

(Knuth-Morris-Pratt) dan BM (Boyer-Moore). Pengguna hanya dapat memilih satu macam algoritma yang akan digunakan untuk pencarian.

- Fitur pencarian

Ketika gambar dan algoritma sudah terpilih, barulah program dapat melakukan pencarian. Apabila gambar belum ada yang terpilih sebagai sumber untuk pencarian dan tombol “search” ditekan, maka program akan menampilkan pesan untuk memasukkan gambar terlebih dahulu. Adapun untuk pilihan algoritma secara default berupa KMP sehingga ketika pengguna tidak menggunakan fitur pemilihan algoritma dan sudah memasukkan gambar, program dapat melakukan pencarian dan menggunakan algoritma KMP.

3. 3. 2. Arsitektur Aplikasi Desktop

Aplikasi Fingerprint Checker by Fingerfingeran adalah aplikasi desktop berbasis WPF (Windows Presentation Foundation) yang dirancang untuk mencocokkan gambar sidik jari dengan basis data gambar yang ada. Aplikasi ini menggunakan berbagai algoritma pencocokan string dan teknik pengolahan citra untuk menemukan kecocokan terbaik. Berikut adalah penjelasan rinci mengenai arsitektur aplikasi ini berdasarkan kode dan struktur proyek yang diberikan.

1. DatabaseLoader:

- **MockGenerator.py** dan **OriginalMockGenerator.py**: Skrip Python yang mungkin digunakan untuk menghasilkan data mock untuk pengujian.
- **SchemaConverter.py** dan **SchemaLoader.cs**: Skrip dan kelas yang berhubungan dengan pemuatan dan konversi skema basis data.
- **DatabaseLoader.csproj**: File proyek untuk komponen pemuat basis data.

2. src:

- **Assets**: Direktori yang berisikan aset gambar yang digunakan dalam aplikasi.
- **AlayFixer.cs**: Kelas yang berfungsi untuk memperbaiki teks yang *corrupt*.
- **Biodata.cs**: Kelas yang menyimpan informasi biodata.
- **BM.cs**: Implementasi algoritma Boyer-Moore untuk pencocokan string.
- **DatabaseManager.cs**: Kelas untuk mengelola interaksi dengan basis data. Pada file ini terdapat metode-metode yang digunakan untuk mengelola

interaksi dengan basis data, termasuk pengambilan jalur gambar dan informasi biodata, juga menggunakan jalur gambar untuk mendapatkan nama dan informasi terkait dari basis data.

- **KMP.cs:** Implementasi algoritma Knuth-Morris-Pratt untuk pencocokan string.
- **Levenshtein.cs:** Implementasi algoritma Levenshtein untuk menghitung jarak kemiripan.
- **MainWindow.xaml:** Mendefinisikan layout UI untuk aplikasi, termasuk kontrol untuk pemilihan gambar, tombol pencarian, dan tampilan hasil. Selain itu, menggunakan XAML untuk mengatur elemen visual dan tata letak.
- **MainWindow.xaml.cs:** Definisi UI dan logika interaksi utama untuk jendela aplikasi. File ini mengelola logika interaksi pengguna dan pengolahan data utama. File juga menangani pemilihan gambar, pemilihan algoritma, dan proses pencarian kecocokan dan menampilkan hasil pencarian kepada pengguna.
- **Processor_Image.cs:** Kelas untuk memproses gambar sidik jari. File ini berisi metode untuk mengkonversi gambar ke representasi biner dan ASCII dan memproses blok ASCII pusat dari gambar untuk digunakan dalam pencocokan.
- **ResultData.cs:** Kelas yang menyimpan data hasil pencarian.

3. test:

- **dataset:** Direktori yang berisi dataset dari gambar sidik jari untuk pengujian.
- **testcases:** Direktori yang berisi kasus uji untuk pengujian aplikasi.

Dengan arsitektur ini, aplikasi **Fingerprint Checker by Fingerfingeran** memberikan solusi efisien untuk mencocokkan gambar sidik jari menggunakan metode pengolahan citra dan pencocokan string yang canggih, yang dapat diaplikasikan dalam berbagai situasi seperti verifikasi identitas dan keamanan.

3.4 Contoh Ilustrasi Kasus

Berikut adalah contoh ilustrasi kasus yang meliputi pemroses foto, proses pencarian foto paling mirip, beserta hasilnya.

Pertama, gambar sidik jari diubah ke bentuk binary.

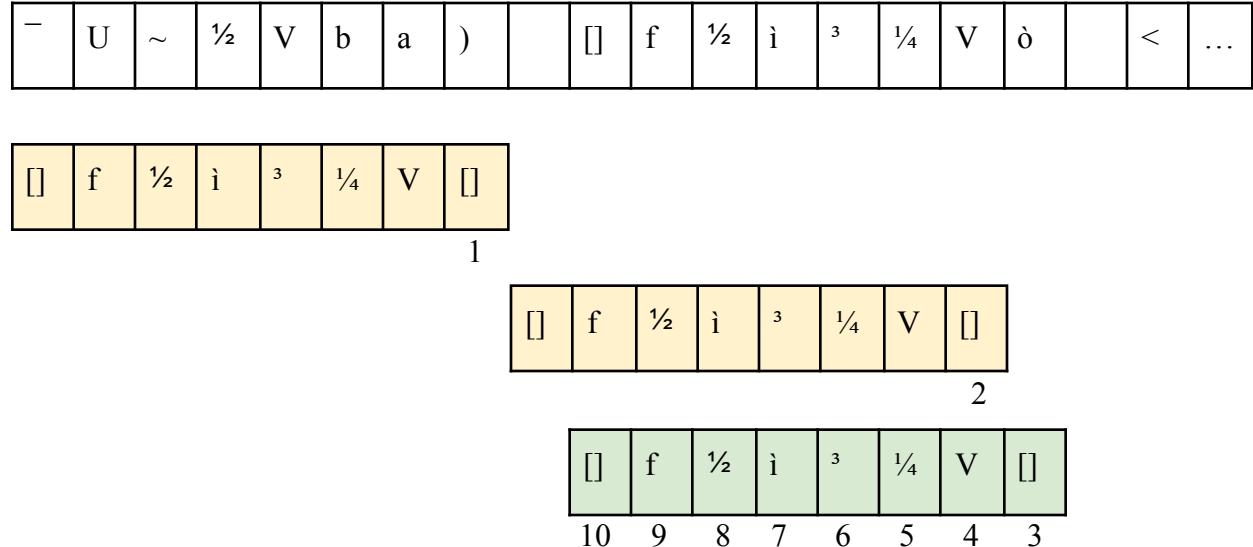

	...	0	0	1	1	1	1	0	0	...
	...	0	1	0	1	0	0	1	0	...
	...	0	1	1	1	1	1	0	0	...
	...	0	0	1	0	1	1	1	0	...
	...	1	0	1	1	0	1	0	1	...
	...	0	1	1	1	0	1	1	0	...
	...	0	1	0	0	1	0	1	0	...
	...	0	0	1	0	1	0	0	0	...

Hasil binary tersebut kemudian akan dikonversi ke bentuk ascii 8 bit dan diambil 30 pixel di tengah.

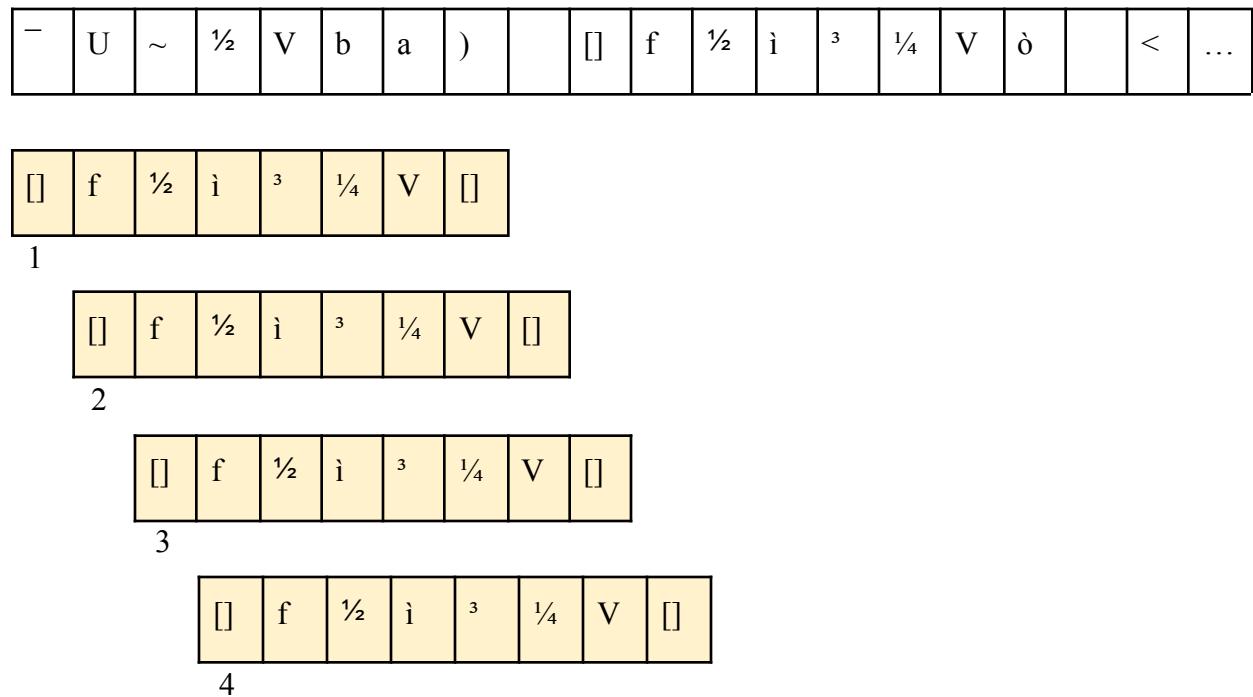

	...	[]	f	½	ì	³	¼	V	[]	...

Pattern tersebut akan dijadikan pembanding pada foto-foto sidik jari yang terdapat pada dataset. Foto-foto tersebut secara keseluruhan akan dijadikan dalam bentuk ascii dan dibandingkan sebagai teks penuhnya.

Menggunakan Boyer-Moore Algorithm, (hanya contoh menggunakan 8 character ascii sebagai pattern saja dan pada kasus adanya *exact match*)



Menggunakan Knuth Morris Pratt Algorithm, (hanya contoh menggunakan 8 character ascii sebagai pattern saja dan pada kasus adanya *exact match*)



[]	f	$\frac{1}{2}$	ì	3	$\frac{1}{4}$	V	[]
5							

[]	f	$\frac{1}{2}$	ì	3	$\frac{1}{4}$	V	[]
6							

[]	f	$\frac{1}{2}$	ì	3	$\frac{1}{4}$	V	[]
7							

[]	f	$\frac{1}{2}$	ì	3	$\frac{1}{4}$	V	[]
8							

[]	f	$\frac{1}{2}$	ì	3	$\frac{1}{4}$	V	[]
9							

[]	f	$\frac{1}{2}$	ì	3	$\frac{1}{4}$	V	[]
10	11	12	13	14	15	16	17

Jika tidak ditemukan *exact match* pada algoritma BM ataupun KMP, maka perlu digunakan Levenshtein distance untuk mencari yang paling mirip. Hal ini dilakukan untuk semua foto pada dataset untuk mencari kesamaan tertinggi.

(Contoh)

[]	f	$\frac{1}{2}$	ì	3	$\frac{1}{4}$	V	[]
----	---	---------------	---	------	---------------	---	----

[]	f	$\frac{1}{2}$	ì	3	a	V	>
----	---	---------------	---	------	---	---	---

Levenshtein distance pada kasus ini adalah 2.

Rumus yang digunakan untuk mencari persen kemiripan antara kedua itu adalah:

$$\text{Similarity} = (1.0 - \frac{\text{distance}}{\text{Max}(length 1, length 2)}) * 100$$

$$\text{Similarity} = (1.0 - (\frac{2}{8})) * 100$$

$$\text{Similarity} = (0.75) * 100 = 75\%$$

Maka karena persen kemiripan lebih besar atau sama dengan threshold yang ditetapkan, maka akan bisa ditetapkan sebagai salah satu calon sidik jari dengan kemiripan tertinggi. Jika semua sidik jari pada dataset memiliki kemiripan yang lebih kecil, maka ditetapkan bahwa tidak ada yang mirip.

Sidik jari yang ditemukan tersebut akan dicari namanya pada tabel sidik jari. Kemudian, akan dicari biodata berdasarkan nama tersebut pada tabel biodata. Nama pada biodata berkemungkinan korup (alay), maka perlu diperbaiki.

Misal, nama yang benar terdiri dari [“Muhammad Neo Cicero Koda”, “Zahira Dina Amalia”, “Shazya Audrea Taufik”] dan pemilik sidik jari tersebut adalah Muhammad Neo Cicero Koda.

Akan dicari pada tabel biodata yang hasil perbaikannya merupakan nama “Muhammad Neo Cicero Koda”. Perbaikan angka pada kata alay menggunakan regex. Perbaikan penyingkatan menggunakan levenshtein. Misal, mh4mMD n Ccr KD → mhamMd n Ccr KD → Muhammad Neo Cicero Koda.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4. 1 Spesifikasi Teknis Program

4. 1. 1 Struktur Data

Struktur Data	Penjelasan
List	Struktur data yang menyimpan daftar.
DateTime	Tanggal
Biodata	Tersusun atas NIK (string), Nama (string), TempatLahir (string), TanggalLahir (DateTime), JenisKelamin (string), GolonganDarah (string), Alamat (string), Agama (string), StatusPerkawinan (string), Pekerjaan (string), Kewarganegaraan (string)
ResultData	Tersusun atas NIK (string), Name (string), Place (string), Birthdate (DateTime), Gender (string), BloodType(string), Address (string), Religion (string), MaritalStatus (string), WorkStatus(string), Nationality (string)

4. 1. 2 Fungsi dan Prosedur

1. AlayFixer.cs

Kelas ini digunakan untuk memperbaiki kata alay yang terdapat pada tabel biodata. Fungsi pada kelas ini adalah FixAlayText yang digunakan untuk melakukan tujuan kelas tersebut.

```
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;
namespace src {
    public static class AlayFixer
    {
        public static string FixAlayText(string text, List<string>
correctNames)
        {
            Dictionary<char, char> numberSubs = new Dictionary<char,
```

```

char>
{
    {'1', 'i'}, {'4', 'a'}, {'6', 'g'}, {'0', 'o'}, {'3', 'e'},
    {'7', 't'}, {'8', 'b'}, {'5', 's'}, {'9', 'p'}, {'2', 'z'}
};

string fixedText = Regex.Replace(text, "[143678059]", match
=> numberSubs[match.Value[0]].ToString());
fixedText = fixedText.ToLower();
string ClosestMatch(string input, List<string> names)
{
    var similarities = names.Select(name => new
    {
        Name = name,
        Similarity =
Levenshtein.CalculateLevenshteinSimilarity(input, name.ToLower())
    }).ToList();
    var closest = similarities.OrderByDescending(x =>
x.Similarity).First();
    return closest.Name;
}
string correctedText = ClosestMatch(fixedText, correctNames);
return correctedText;
}
}
}

```

2. Biodata.cs

Kelas ini digunakan untuk membentuk objek biodata yang terdiri dari NIK, nama, tempat lahir, tanggal lahir, jenis kelamin, golongan darah, alamat, agama, status perkawinan, pekerjaan, dan kewarganegaraan.

```

using System;
namespace src {
    public class Biodata
    {
        public string NIK { get; set; }

```

```

        public string Nama { get; set; }
        public string TempatLahir { get; set; }
        public DateTime TanggalLahir { get; set; }
        public string JenisKelamin { get; set; }
        public string GolonganDarah { get; set; }
        public string Alamat { get; set; }
        public string Agama { get; set; }
        public string StatusPerkawinan { get; set; }
        public string Pekerjaan { get; set; }
        public string Kewarganegaraan { get; set; }
    }
}

```

3. BM.cs

Kelas ini digunakan untuk menjalankan algoritma Boyer-Moore. Kelas ini memiliki atribut `_badCharacterShift` dan `_pattern`. Fungsi `BuildBadCharacterShift` digunakan untuk menentukan metode shift pada algoritma ini. Fungsi `Search` digunakan untuk mencari *exact match* pattern pada suatu teks.

```

using System;
namespace src {
    public class BoyerMoore {
        private readonly int[] _badCharacterShift;
        private readonly string _pattern;
        public BoyerMoore(string pattern) {
            _pattern = pattern;
            _badCharacterShift = BuildBadCharacterShift(pattern);
        }
        private int[] BuildBadCharacterShift(string pattern) {
            const int alphabetSize = 256;
            int[] badCharShift = new int[alphabetSize];
            for (int i = 0; i < alphabetSize; i++)
            {

```

```

        badCharShift[i] = pattern.Length;
    }
    for (int i = 0; i < pattern.Length - 1; i++)
    {
        badCharShift[pattern[i]] = pattern.Length - 1 - i;
    }
    return badCharShift;
}
public int Search(string text)
{
    int m = _pattern.Length;
    int n = text.Length;
    int skip;
    for (int i = 0; i <= n - m; i += skip)
    {
        skip = 0;
        for (int j = m - 1; j >= 0; j--)
        {
            if (_pattern[j] != text[i + j])
            {
                skip = Math.Max(1, _badCharacterShift[text[i +
j]] - (m - 1 - j));
                break;
            }
        }
        if (skip == 0) return i;
    }
    return -1;
}
}
}

```

4. DatabaseManager.cs

Kelas ini digunakan untuk menghubungkan program dengan basis data.

```

using System;
using System.Drawing;
using System.Text;
using System.Diagnostics;
using System.Collections.Generic;
using System.Linq;
using System.Text.RegularExpressions;
using System.Reflection.Metadata.Ecma335;
using MySql.Data.MySqlClient;
using System.Collections.Generic;
using DotNetEnv;
namespace src {
    public static class DatabaseManager
    {
        private static string connectionString;
        static DatabaseManager()
        {
            Env.Load("../.env");
            connectionString =
                $"server={Environment.GetEnvironmentVariable("DB_SERVER")}; " +
                $"user={Environment.GetEnvironmentVariable("DB_USER")}; " +
                $"password={Environment.GetEnvironmentVariable("DB_PASSWORD")}; " +
                $"database=fingerprint";
        }
        public static string[] GetImagePathsFromDatabase()
        {
            string query = "SELECT berkas_citra FROM sidik_jari";
            List<string> imagePaths = new List<string>();
            using (var connection = new
            MySqlConnection(connectionString))
            {
                connection.Open();
                var command = new MySqlCommand(query, connection);
                using (var reader = command.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        string imagePath =
                            reader.GetString("berkas_citra");
                        imagePath = DecryptXOR(imagePath, 129);
                        imagePaths.Add(imagePath);
                    }
                }
            }
            return imagePaths.ToArray();
        }
    }
}

```

```

    }

    public static Biodata GetBiodataForName(string name, List<string>
correctNames)
    {
        string query = "SELECT * FROM biodata";

        // List<Biodata> biodataList = new List<Biodata>();
        using (var connection = new
MySqlConnection(connectionString))
        {
            connection.Open();
            Biodata data = new Biodata();
            var command = new MySqlCommand(query, connection);
            command.Parameters.AddWithValue("@name", name);
            using (var reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    string namaHasil =
DecryptXOR(reader.GetString("nama"), 129);
                    namaHasil = AlayFixer.FixAlayText(namaHasil,
correctNames);
                    if (namaHasil == name) {
                        data.NIK =
DecryptXOR(reader.GetString("NIK"), 129);
                        data>Nama = namaHasil;
                        data.TempatLahir =
DecryptXOR(reader.GetString("tempat_lahir"), 129);
                        data.TanggalLahir =
DecryptDate(reader.GetDateTime("tanggal_lahir"), 129);
                        data.JenisKelamin =
reader.GetString("jenis_kelamin");
                        data.GolonganDarah =
DecryptXOR(reader.GetString("golongan_darah"), 129);
                        data.Alamat =
DecryptXOR(reader.GetString("alamat"), 129);
                        data.Agama =
DecryptXOR(reader.GetString("agama"), 129);
                        data.StatusPerkawinan =
reader.GetString("status_perkawinan");
                        data.Pekerjaan =
DecryptXOR(reader.GetString("pekerjaan"), 129);
                        data.Kewarganegaraan =
DecryptXOR(reader.GetString("kewarganegaraan"), 129);
                    };
                }
            }
        }
    }
}

```

```

        return data;
    }
}
public static string GetNameFromImagePath(string imagePath)
{
    imagePath = DecryptXOR(imagePath, 129);
    string query = "SELECT nama FROM sidik_jari WHERE
berkas_citra = @ImagePath";
    string name = "";
    using (var connection = new
MySqlConnection(connectionString))
    {
        connection.Open();
        using (var command = new MySqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@ImagePath",
imagePath);
            using (var reader = command.ExecuteReader())
            {
                if (reader.Read())
                {
                    return DecryptXOR(reader.GetString("nama"),
129);
                }
            }
        }
        return name;
    }
}
public static List<string> GetCorrectNamesFromDatabase()
{
    string query = "SELECT nama FROM sidik_jari";
    List<string> correctNames = new List<string>();
    using (var connection = new
MySqlConnection(connectionString))
    {
        connection.Open();
        using (var command = new MySqlCommand(query, connection))
        {
            using (var reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    string correctName =
reader.GetString("nama");
                    correctName = DecryptXOR(correctName, 129);
                    correctNames.Add(correctName);
                }
            }
        }
    }
}
```

```

        }
    }
}
return correctNames;
}
public static List<string> GetAlayNamesFromDatabase()
{
    string query = "SELECT nama FROM biodata";
    List<string> alayNames = new List<string>();
    using (var connection = new
MySqlConnection(connectionString))
    {
        connection.Open();
        using (var command = new MySqlCommand(query, connection))
        {
            using (var reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    string alayName = reader.GetString("nama");
                    alayNames.Add(alayName);
                }
            }
        }
    }
    return alayNames;
}
public static string FindAlayName(Dictionary<string, string>
dictionary, string alayName)
{
    if (dictionary.TryGetValue(alayName, out string foundName))
    {
        return foundName;
    }
    return null;
}
private static string DecryptXOR(string encryptedText, byte key)
{
    StringBuilder decryptedText = new StringBuilder();
    foreach (char c in encryptedText)
    {
        decryptedText.Append((char)(c ^ key));
    }
    return decryptedText.ToString();
}
private static DateTime DecryptDate(DateTime encryptedDate, int

```

```

key)
{
    return encryptedDate.AddYears(-key);
}
}
}

```

5. KMP.cs

Kelas ini menyimpan fungsi-fungsi yang diperlukan pada perhitungan algoritma KMP (Knuth-Morris-Pratt).

```

namespace src {
    public class KMP
    {
        private readonly int[] _lps;
        private readonly string _pattern;

        public KMP(string pattern)
        {
            _pattern = pattern;
            _lps = BuildLPSArray(pattern);
        }

        private int[] BuildLPSArray(string pattern)
        {
            int[] lps = new int[pattern.Length];
            int length = 0;
            int i = 1;

            lps[0] = 0;

            while (i < pattern.Length)
            {
                if (pattern[i] == pattern[length])
                {
                    length++;
                    lps[i] = length;
                    i++;
                }
            }
        }
    }
}

```

```

        }
        else
        {
            if (length != 0)
            {
                length = lps[length - 1];
            }
            else
            {
                lps[i] = 0;
                i++;
            }
        }
    }

    public int Search(string text)
{
    int m = _pattern.Length;
    int n = text.Length;
    int i = 0;
    int j = 0;

    while (i < n)
    {
        if (_pattern[j] == text[i])
        {
            j++;
            i++;
        }

        if (j == m)
        {
            return i - j;
        }
    }
}

```

```

        else if (i < n && _pattern[j] != text[i])
        {
            if (j != 0)
            {
                j = _lps[j - 1];
            }
            else
            {
                i++;
            }
        }
        return -1;
    }
}
}

```

6. Levenshtein.cs

Kelas ini menyimpan fungsi-fungsi yang diperlukan pada perhitungan algoritma Levenshtein

```

using System;

namespace src {
    public class Levenshtein {
        public static double CalculateLevenshteinBlockString(string
block, string str)
        {
            int blockLength = block.Length;
            int strLength = str.Length;

            double maxPercentage = double.MinValue;

            for (int i = 0; i <= strLength - blockLength; i++)
            {
                string substring = str.Substring(i, blockLength);
            }
        }
    }
}

```

```

        double distance = CalculateLevenshteinSimilarity(block,
substring);

        if (distance > maxPercentage)
        {
            maxPercentage = distance;
        }

    }

    return maxPercentage;
}

public static double CalculateLevenshteinSimilarity(string s1,
string s2)
{
    int distance = ComputeLevenshteinDistance(s1, s2);
    int maxLen = Math.Max(s1.Length, s2.Length);
    return (1.0 - (double)distance / maxLen) * 100;
}

public static int ComputeLevenshteinDistance(string s1, string
s2)
{
    int[,] d = new int[s1.Length + 1, s2.Length + 1];

    for (int i = 0; i <= s1.Length; i++)
    {
        d[i, 0] = i;
    }
    for (int j = 0; j <= s2.Length; j++)
    {
        d[0, j] = j;
    }

    for (int i = 1; i <= s1.Length; i++)
    {

```

```

        for (int j = 1; j <= s2.Length; j++)
        {
            int cost = (s1[i - 1] == s2[j - 1]) ? 0 : 1;
            d[i, j] = Math.Min(Math.Min(d[i - 1, j] + 1, d[i, j - 1] + 1), d[i - 1, j - 1] + cost);
        }
    }

    return d[s1.Length, s2.Length];
}
}
}

```

7. MainWindow.xaml.cs

File ini berguna sebagai controller untuk UI yang sudah ada pada file MainWindow.xaml

```

using System;
using System.IO;
using System.Collections.Generic;
using System.Diagnostics;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using Microsoft.Win32;
using src;

namespace WpfApp
{
    public partial class MainWindow : Window
    {
        private string toCompareImagePath = string.Empty;
        private string selectedAlgorithm = "KMP";
        private string bestMatchImagePath = string.Empty;
    }
}

```

```

private Biodata data = new();
public MainWindow()
{
    InitializeComponent();
    SetupEventHandlers();
}

private void SetupEventHandlers()
{
    choosePictureButton.Click += ChoosePictureButton_Click;
    searchButton.Click += SearchButton_Click;
    kmpRadioButton.Checked += RadioButton_CheckedChanged;
    bmRadioButton.Checked += RadioButton_CheckedChanged;
}

private void RadioButton_CheckedChanged(object sender, RoutedEventArgs e)
{
    if (sender is RadioButton radioButton)
    {
        selectedAlgorithm = radioButton.Content.ToString();
    }
}

private async void ChoosePictureButton_Click(object sender, RoutedEventArgs e)
{
    var dialog = new OpenFileDialog
    {
        Filter = "Image Files|*.jpg;*.jpeg;*.png;*.bmp",
        Title = "Select a Fingerprint Image"
    };

    if (dialog.ShowDialog() == true)
    {
}

```

```

        toCompareImagePath = dialog.FileName;
        var fingerprintImage = FindName("fingerprintImage") as
Image;
        if (fingerprintImage != null)
        {
            BitmapImage bitmap = new BitmapImage(new
Uri(toCompareImagePath));
            fingerprintImage.Source = bitmap;
            fingerprintImage.Width = 200;
            fingerprintImage.Height = 200;
            fingerprintImage.Stretch = Stretch.Uniform;
        }
    }

    private async void SearchButton_Click(object sender,
RoutedEventArgs e)
{
    var fingerprintImage = FindName("fingerprintImage") as Image;
    var ImagePath = FindName("ImagePath") as Image;
    var dynamicMessage = FindName("dynamicMessage") as TextBlock;
    if (fingerprintImage?.Source == null)
    {
        UpdateMessage("Please upload an image.");
        return;
    }

    UpdateMessage("LOADING ...");
    string[] imagePathsFromDatabase =
DatabaseManager.GetImagePathsFromDatabase();
    List<string> correctNames =
DatabaseManager.GetCorrectNamesFromDatabase();

    string name = "";
    long execution = 0;
    double bestLevenshteinSimilarity = 0;
}

```

```

        Stopwatch stopwatch = new Stopwatch();
        stopwatch.Start();

        List<string> binaryStrings1 =
ImageProcessor.ConvertImageToBinaryString2(toCompareImagePath);
        List<string> asciiStrings1 =
ImageProcessor.ConvertBinaryToAscii2(binaryStrings1);
        string asciiBlock1 =
ImageProcessor.ExtractCentralAsciiBlock2(asciiStrings1, 30);

        int bestMatchPosition = -1;
        bestLevenshteinSimilarity = 0;

        UpdateMessage("Mulai ...");

        matchFoundMessage.Visibility = Visibility.Hidden;
        detailsGrid.Visibility = Visibility.Hidden;
        executionTimeMessage.Text = $"Execution Time: ";
        similarityPercentageMessage.Text = $"Similarity: ";
        ShowSearchResults(false, new Biodata(), null);

        await Task.Run(() =>
{
    if (!ImageProcessor.checkEmptyPattern(asciiBlock1))
    {
        try
        {
            Parallel.ForEach(imagePathsFromDatabase,
(imagePath2, state) =>
{
            List<string> binaryStrings2 =
ImageProcessor.ConvertImageToBinaryString2(imagePath2);
            List<string> asciiStrings2 =
ImageProcessor.ConvertBinaryToAscii2(binaryStrings2);

```

```

        int matchPosition = -1;
        if (selectedAlgorithm == "BM")
        {
            Dispatcher.Invoke(() =>
UpdateMessage("Pencarian dengan BM..."));
            BoyerMoore bm = new
BoyerMoore(asciiBlock1);
            foreach (string asciiString2 in
asciiStrings2)
            {
                matchPosition =
bm.Search(asciiString2);
                if (matchPosition != -1)
                {
                    break;
                }
            }
        }
        else if (selectedAlgorithm == "KMP")
        {
            Dispatcher.Invoke(() =>
UpdateMessage("Pencarian dengan KMP..."));
            KMP kmp = new KMP(asciiBlock1);
            foreach (string asciiString2 in
asciiStrings2)
            {
                matchPosition =
kmp.Search(asciiString2);
                if (matchPosition != -1)
                {
                    break;
                }
            }
        }
    }

    if (matchPosition != -1)

```

```

        {
            bestMatchPosition = matchPosition;
            bestMatchImagePath = imagePath2;
            state.Stop();
        }
        else
        {
            double levenshteinSimilarity = 0;
            foreach (string asciiString2 in
asciiStrings2)
            {
                levenshteinSimilarity =
Levenshtein.CalculateLevenshteinBlockString(asciiBlock1, asciiString2);
                if (levenshteinSimilarity >
bestLevenshteinSimilarity &!state.IsStopped)
                {
                    bestLevenshteinSimilarity =
levenshteinSimilarity;
                    bestMatchImagePath = imagePath2;
                }
            }
        });
    }

    if (bestMatchPosition != -1)
    {
        name =
DatabaseManager.GetNameFromImagePath(bestMatchImagePath);
        bestLevenshteinSimilarity = 100;
    }
    else
    {
        name =
DatabaseManager.GetNameFromImagePath(bestMatchImagePath);
    }
}

```

```

        catch (Exception ex)
        {
            Dispatcher.Invoke(() => {
                UpdateMessage("An error occurred during the
search process.");
            });
        }

        data = DatabaseManager.GetBiodataForName(name,
correctNames);
    }

    stopwatch.Stop();
    execution = stopwatch.ElapsedMilliseconds;
    Console.WriteLine($"Time taken:
{stopwatch.ElapsedMilliseconds} ms");
}

if (ImageProcessor.checkEmptyPattern(asciiBlock1) ||
bestLevenshteinSimilarity < 75)
{
    UpdateMessage("No match found.");
    dynamicMessage.Text = "No match found.";
    executionTimeMessage.Text = $"Execution Time: {execution}
ms";
}
else
{
    UpdateMessage("Search Complete!");
    dynamicMessage.Text = "Search Complete!";
    matchFoundMessage.Visibility = Visibility.Visible;
    executionTimeMessage.Text = $"Execution Time: {execution}
ms";
    similarityPercentageMessage.Text = $"Similarity:
{bestLevenshteinSimilarity:F2}%";
}

```

```

        ShowSearchResults(true, data,
Path.GetFullPath(bestMatchImagePath));
    }
}

private void UpdateMessage(string message)
{
    Dispatcher.Invoke(() =>
{
    var dynamicMessage = FindName("dynamicMessage") as
TextBlock;
    if (dynamicMessage != null)
    {
        dynamicMessage.Text = message;
    }
});
}

private void ShowSearchResults(bool show, Biodata biodata, string
ImagePathResult)
{
    Dispatcher.Invoke(() =>
{
    matchFoundMessage.Visibility = show ? Visibility.Visible
: Visibility.Hidden;
    detailsGrid.Visibility = show ? Visibility.Visible :
Visibility.Hidden;

    var resultData = new ResultData
{
        NIK = biodata.NIK,
        Name = biodata>Nama,
        Place = biodata.TempatLahir,
        Birthdate = biodata.Tanggallahir,
        BloodType = biodata.GolonganDarah,

```

```

        Gender = biodata.JenisKelamin,
        Address = biodata.Alamat,
        Religion = biodata.Agama,
        MaritalStatus = biodata.StatusPerkawinan,
        WorkStatus = biodata.Pekerjaan,
        Nationality = biodata.Kewarganegaraan,
    };

    var resultImage = FindName("resultImage") as Image;
    if (resultImage != null)
    {
        if (ImagePathResult != null)
        {
            BitmapImage bitmap = new BitmapImage(new
Uri(ImagePathResult));
            resultImage.Source = bitmap;
            resultImage.Width = 200;
            resultImage.Height = 200;
            resultImage.Stretch = Stretch.Uniform;
            resultImage.Visibility = Visibility.Visible;
        }
        else
        {
            resultImage.Visibility = Visibility.Hidden;
        }
    }

    var panel = FindName("rightPanel") as StackPanel;
    if (panel != null)
    {
        panel.DataContext = resultData;
    }
});
```

}

}

```
}
```

8. Processor_Image.cs

Kelas ini digunakan untuk menyimpan metode-metode yang akan digunakan pada pemrosesan citra.

```
using System;
using System.Drawing;
using System.Text;
using System.Collections.Generic;

namespace src {
    public static class ImageProcessor
    {
        public static List<string> ConvertImageToBinaryString2(string
imagePath)
        {
            try
            {
                using (Bitmap bitmap = new Bitmap(imagePath))
                {
                    List<string> binaryStrings = new List<string>();

                    for (int y = 0; y < bitmap.Height; y++)
                    {
                        StringBuilder rowString = new StringBuilder();

                        for (int x = 0; x < bitmap.Width; x++)
                        {
                            Color pixelColor = bitmap.GetPixel(x, y);
                            int gray = (pixelColor.R + pixelColor.G +
pixelColor.B) / 3;
                            rowString.Append(gray < 128 ? "0" : "1");
                        }
                    }
                }
            }
        }
    }
}
```

```

        binaryStrings.Add(rowString.ToString());
    }

    return binaryStrings;
}
}

catch (Exception ex)
{
    throw new ArgumentException($"Error processing image at
path {imagePath}: {ex.Message}");
}
}

public static List<string> ConvertBinaryToAscii2(List<string>
binaryStrings)
{
    int rows = binaryStrings.Count;
    int cols = binaryStrings[0].Length;
    List<string> asciiStrings = new List<string>();

    for (int y = 0; y < rows - 7; y++)
    {
        StringBuilder rowString = new StringBuilder();

        for (int x = 0; x < cols; x++)
        {
            StringBuilder binaryChunk = new StringBuilder();
            for (int i = 0; i < 8; i++)
            {
                binaryChunk.Append(binaryStrings[y + i][x]);
            }

            char asciiChar =
(char)Convert.ToInt32(binaryChunk.ToString(), 2);
            rowString.Append(asciiChar);
        }
    }
}

```

```

        }

        asciiStrings.Add(rowString.ToString());
    }

    return asciiStrings;
}

public static string ExtractCentralAsciiBlock2(List<string>
asciiStrings, int length)
{
    int middleIndex = asciiStrings.Count / 2;
    string middleString = asciiStrings[middleIndex];
    int center = middleString.Length / 2;
    int start = Math.Max(0, center - length / 2);
    return middleString.Substring(start, Math.Min(length,
middleString.Length - start));
}

public static bool checkEmptyPattern(string pattern)
{
    foreach (char c in pattern)
    {
        if ((int) c != 255)
        {
            return false;
        }
    }
    return true;
}
}

```

9. ResultData.cs

Kelas ini berguna untuk menyimpan komponen-komponen yang berupa hasil dari pencarian

```
using System;
using System.ComponentModel;
using System.IO;
using System.Windows.Media.Imaging;

namespace src{
    public class ResultData : INotifyPropertyChanged
    {
        private string nik;
        public string NIK
        {
            get => nik;
            set
            {
                nik = value;
                OnPropertyChanged(nameof(NIK));
            }
        }

        private string name;
        public string Name
        {
            get => name;
            set
            {
                name = value;
                OnPropertyChanged(nameof(Name));
            }
        }

        private string place;
        public string Place
        {
            get => place;
            set
            {
```

```
        place = value;
        OnPropertyChanged(nameof(Place));
    }
}

private DateTime birthdate;
public DateTime Birthdate
{
    get => birthdate;
    set
    {
        birthdate = value;
        OnPropertyChanged(nameof(Birthdate));
    }
}

private string bloodType;
public string BloodType
{
    get => bloodType;
    set
    {
        bloodType = value;
        OnPropertyChanged(nameof(BloodType));
    }
}

private string gender;
public string Gender
{
    get => gender;
    set
    {
        gender = value;
        OnPropertyChanged(nameof(Gender));
    }
}
```

```
        }

    }

    private string address;
    public string Address
    {
        get => address;
        set
        {
            address = value;
            OnPropertyChanged(nameof(Address));
        }
    }

    private string religion;
    public string Religion
    {
        get => religion;
        set
        {
            religion = value;
            OnPropertyChanged(nameof(Religion));
        }
    }

    private string maritalStatus;
    public string MaritalStatus
    {
        get => maritalStatus;
        set
        {
            maritalStatus = value;
            OnPropertyChanged(nameof(MaritalStatus));
        }
    }
}
```

```
private string workStatus;
public string WorkStatus
{
    get => workStatus;
    set
    {
        workStatus = value;
        OnPropertyChanged(nameof(WorkStatus));
    }
}

private string nationality;
public string Nationality
{
    get => nationality;
    set
    {
        nationality = value;
        OnPropertyChanged(nameof(Nationality));
    }
}

private string imagePath;
public string ImagePath
{
    get => imagePath;
    set
    {
        imagePath = value;
        OnPropertyChanged(nameof(ImagePath));
    }
}

public BitmapImage ImageSource
```

```

    {
        get
        {
            if (File.Exists(ImagePath))
            {
                var bitmap = new BitmapImage();
                bitmap.BeginInit();
                bitmap.UriSource = new Uri(ImagePath,
                UriKind.Absolute);
                bitmap.CacheOption = BitmapCacheOption.OnLoad;
                bitmap.EndInit();
                return bitmap;
            }
            return null;
        }
    }

    public event PropertyChangedEventHandler PropertyChanged;
    protected void OnPropertyChanged(string propertyName)
    {
        PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(propertyName));
    }
}

```

10. SchemaLoader.cs

Kelas ini berguna untuk membuat basis data.

```

using System;
using MySqlConnector;
using DotNetEnv;
class SchemaLoader
{
    static void Main()
    {
        Env.Load("../.env");
        string server = Environment.GetEnvironmentVariable("DB_SERVER")
?? "localhost";
    }
}

```

```

        string user = Environment.GetEnvironmentVariable("DB_USER") ??
"root";
        string password =
Environment.GetEnvironmentVariable("DB_PASSWORD") ?? "12345";
        string database1 = "fingerprint";
        string database2 =
Environment.GetEnvironmentVariable("DB_SOURCE") ??
"fingerprint_original";
        string connectionString = $"Server={server};User
ID={user};Password={password};" ;
        try
{
    using (var conn = new MySqlConnection(connectionString))
    {
        conn.Open();
        CreateDatabase(conn, database1);
        CreateDatabase(conn, database2);
        LoadSchema(conn, database1);
        LoadSchema(conn, database2);
        Console.WriteLine("Databases and tables created
successfully.");
    }
}
catch (Exception ex)
{
    Console.WriteLine($"An error occurred: {ex.Message}");
}
static void CreateDatabase(MySqlConnection conn, string database)
{
    using (var cmd = new MySqlCommand($"CREATE DATABASE IF NOT EXISTS
`{database}`;", conn))
    {
        cmd.ExecuteNonQuery();
    }
}
static void LoadSchema(MySqlConnection conn, string database)
{
    conn.ChangeDatabase(database);
    string sqlDump = @"
-- MySQL dump 10.13 Distrib 8.0.36, for Linux (x86_64)
--
-- Host: localhost      Database: tubes3_stima24
--
-- -----
-- Server version  8.0.36-0ubuntu0.22.04.1
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS

```

```

*/;

/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!50503 SET NAMES utf8mb4 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0
*/;

/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

-- 
-- Table structure for table `biodata`
-- 

DROP TABLE IF EXISTS `biodata`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `biodata` (
  `NIK` varchar(16) NOT NULL,
  `nama` varchar(100) DEFAULT NULL,
  `tempat_lahir` varchar(50) DEFAULT NULL,
  `tanggal_lahir` date DEFAULT NULL,
  `jenis_kelamin` enum('Laki-Laki','Perempuan') DEFAULT NULL,
  `golongan_darah` varchar(5) DEFAULT NULL,
  `alamat` varchar(255) DEFAULT NULL,
  `agama` varchar(50) DEFAULT NULL,
  `status_perkawinan` enum('Belum Menikah','Menikah','Cerai')
DEFAULT NULL,
  `pekerjaan` varchar(100) DEFAULT NULL,
  `kewarganegaraan` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`NIK`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;
/*!40101 SET character_set_client = @saved_cs_client */;

-- 
-- Dumping data for table `biodata`
-- 

LOCK TABLES `biodata` WRITE;
/*!40000 ALTER TABLE `biodata` DISABLE KEYS */;
/*!40000 ALTER TABLE `biodata` ENABLE KEYS */;
UNLOCK TABLES;

-- 
-- Table structure for table `sidik_jari`
-- 

DROP TABLE IF EXISTS `sidik_jari`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
```

```

/*!50503 SET character_set_client = utf8mb4 */;
CREATE TABLE `sidik_jari` (
`berkas_citra` text,
`nama` varchar(100) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_general_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
--
-- Dumping data for table `sidik_jari`
--
LOCK TABLES `sidik_jari` WRITE;
/*!40000 ALTER TABLE `sidik_jari` DISABLE KEYS */;
/*!40000 ALTER TABLE `sidik_jari` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;
/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
-- Dump completed on 2024-05-04 15:57:34
";
foreach (var sql in sqlDump.Split(new[] { ';' },
StringSplitOptions.RemoveEmptyEntries))
{
    if (!string.IsNullOrWhiteSpace(sql))
    {
        using (var cmd = new MySqlCommand(sql, conn))
        {
            cmd.ExecuteNonQuery();
        }
    }
}
}
}

```

11. MockGenerator.py

Kelas ini berguna untuk membuat basis data yang sudah terenkripsi.

```

from faker import Faker
import random
import os
import mysql.connector
from datetime import datetime

```

```

from dotenv import load_dotenv
def alay_upper_lower(text):
    result = ''.join(random.choice([char.lower(), char.upper()]) for char
in text)
    return result
def alay_numbers(text):
    replacement = {'a': '4', 'i': '1', 'e': '3', 'o': '0', 's': '5', 'g':
'6', 't': '7'}
    result = ''.join(replacement.get(char.lower(), char) for char in
text)
    return result
def alay_remove_vowels(text):
    vowels = "aeiouAEIOU"
    result = ''.join(char for char in text if char not in vowels or
random.random() > 0.5)
    return result
def alay_abbreviation(text):
    words = text.split()
    result = ' '.join(alay_remove_vowels(word) for word in words)
    return result
def alay_combination(text):
    if random.random() > 0.75:
        result = alay_upper_lower(alay_numbers(alay_abbreviation(text)))
        return result
    else:
        return text
def generate_bahasa_alay(text):
    print("Kata orisinal:", text)
    print("Kombinasi huruf besar-kecil:", alay_upper_lower(text))
    print("Penggunaan angka:", alay_numbers(text))
    print("Penyingkatan:", alay_abbreviation(text))
    print("Kombinasi ketiganya:", alay_combination(text))
fake = Faker()
def generate_fake_data():
    nik = fake.random_number(digits=16)
    nama = fake.name()
    tempat_lahir = fake.city()
    tanggal_lahir = fake.date_of_birth()
    jenis_kelamin = random.choice(['Laki-Laki', 'Perempuan'])
    golongan_darah = random.choice(['A', 'B', 'AB', 'O']) +
random.choice(['+', '-'])
    alamat = fake.address()
    agama = random.choice(['Islam', 'Kristen', 'Katolik', 'Hindu',
'Buddha', 'Konghucu'])
    status_perkawinan = random.choice(['Belum Menikah', 'Menikah',
'Cerai'])
    pekerjaan = fake.job()

```

```

kewarganegaraan = fake.country()
    return [nik, nama, tempat_lahir, tanggal_lahir, jenis_kelamin,
golongan_darah, alamat, agama, status_perkawinan, pekerjaan,
kewarganegaraan]
def list_files(directory):
    files = [f for f in os.listdir(directory) if
os.path.isfile(os.path.join(directory, f))]
        return files
def xor_encrypt_decrypt(text, key, length):
    encrypted = ''.join(chr(ord(char) ^ key) for char in text)
    return encrypted[:length]
def mod_encrypt_date(date_obj, key):
    year = (date_obj.year + key) % 9999
    month = (date_obj.month + key - 1) % 12 + 1
    day = (date_obj.day + key - 1) % 31 + 1
    while True:
        try:
            encrypted_date = datetime(year, month, day).date()
            break
        except ValueError:
            day -= 1
    return encrypted_date
def mod_decrypt_date(date_obj, key):
    year = (date_obj.year - key) % 9999
    month = (date_obj.month - key - 1) % 12 + 1
    day = (date_obj.day - key - 1) % 31 + 1
    while True:
        try:
            decrypted_date = datetime(year, month, day).date()
            break
        except ValueError:
            day -= 1
    return decrypted_date
load_dotenv("../.env")
directory_path = r'../test/dataset'
files = list_files(directory_path)
conn = mysql.connector.connect(
    host=os.getenv("DB_SERVER"),
    user=os.getenv("DB_USER"),
    password=os.getenv("DB_PASSWORD"),
    database="fingerprint"
)
cursor = conn.cursor()
encryption_key = int(os.getenv("ENCRYPTION_KEY"))
k = 0
num_biodata = 600
for i in range(num_biodata):

```

```

        sql_biodata = "INSERT INTO biodata (NIK, nama, tempat_lahir,
tanggal_lahir, jenis_kelamin, golongan_darah, alamat, agama,
status_perkawinan, pekerjaan, kewarganegaraan) VALUES (%s, %s, %s, %s,
%s, %s, %s, %s, %s, %s)"
        fake_biodata = generate_fake_data()
        original_name = fake_biodata[1]
        fake_biodata[0] = xor_encrypt_decrypt(str(fake_biodata[0]),
encryption_key, 16)
        fake_biodata[1] =
xor_encrypt_decrypt(alay_combination(original_name), encryption_key, 100)
        fake_biodata[2] = xor_encrypt_decrypt(fake_biodata[2],
encryption_key, 100)
        fake_biodata[5] = xor_encrypt_decrypt(fake_biodata[5],
encryption_key, 200)
        fake_biodata[6] = xor_encrypt_decrypt(fake_biodata[6],
encryption_key, 200)
        fake_biodata[7] = xor_encrypt_decrypt(fake_biodata[7],
encryption_key, 50)
        fake_biodata[9] = xor_encrypt_decrypt(fake_biodata[9],
encryption_key, 100)
        fake_biodata[10] = xor_encrypt_decrypt(fake_biodata[10],
encryption_key, 50)
        encrypted_date = mod_encrypt_date(fake_biodata[3], encryption_key)
        fake_biodata[3] = encrypted_date
        cursor.execute(sql_biodata, fake_biodata)

        for j in range(10):
            if k >= len(files):
                k = 0
            sql_sidik_jari = "INSERT INTO sidik_jari (berkas_citra, nama)
VALUES (%s, %s)"
            fake_sidik = (xor_encrypt_decrypt("../test/dataset/" + files[k],
encryption_key, 200), xor_encrypt_decrypt(original_name, encryption_key,
100)) # Encrypt name
            cursor.execute(sql_sidik_jari, fake_sidik)
            k += 1
conn.commit()
cursor.close()
conn.close()

```

12. OriginalMockGenerator.py

Kelas ini berguna untuk membuat basis data yang tidak terenkripsi.

```

from faker import Faker
import random
import os

```

```

import mysql.connector
from dotenv import load_dotenv
def alay_upper_lower(text):
    result = ''.join(random.choice([char.lower(), char.upper()]) for char
in text)
    return result
def alay_numbers(text):
    replacement = {'a': '4', 'i': '1', 'e': '3', 'o': '0', 's': '5', 'g':
'6', 't': '7'}
    result = ''.join(replacement.get(char.lower(), char) for char in
text)
    return result
def alay_remove_vowels(text):
    vowels = "aeiouAEIOU"
    result = ''.join(char for char in text if char not in vowels or
random.random() > 0.5)
    return result
def alay_abbreviation(text):
    words = text.split()
    result = ' '.join(alay_remove_vowels(word) for word in words)
    return result
def alay_combination(text):
    if random.random() > 0.75:
        result = alay_upper_lower(alay_numbers(alay_abbreviation(text)))
        return result
    else:
        return text
def generate_bahasa_alay(text):
    print("Kata orisinal:", text)
    print("Kombinasi huruf besar-kecil:", alay_upper_lower(text))
    print("Penggunaan angka:", alay_numbers(text))
    print("Penyingkatan:", alay_abbreviation(text))
    print("Kombinasi ketiganya:", alay_combination(text))
fake = Faker()
def generate_fake_data():
    nik = fake.random_number(digits=16)
    nama = fake.name()
    tempat_lahir = fake.city()
    tanggal_lahir = fake.date_of_birth()
    jenis_kelamin = random.choice(['Laki-Laki', 'Perempuan'])
    golongan_darah = random.choice(['A', 'B', 'AB', 'O']) +
random.choice(['+', '-'])
    alamat = fake.address()
    agama = random.choice(['Islam', 'Kristen', 'Katolik', 'Hindu',
'Buddha', 'Konghucu'])
    status_perkawinan = random.choice(['Belum Menikah', 'Menikah',
'Cerai'])

```

```

pekerjaan = fake.job()
kewarganegaraan = random.choice(['Indonesia', 'France',
'Netherlands', 'Italy', 'USA', 'Malaysia', 'Singapore'])
return [nik, nama, tempat_lahir, tanggal_lahir, jenis_kelamin,
golongan_darah, alamat, agama, status_perkawinan, pekerjaan,
kewarganegaraan]
def list_files(directory):
    files = [f for f in os.listdir(directory) if
os.path.isfile(os.path.join(directory, f))]
    return files
load_dotenv("../.env")
directory_path = r'../test/dataset'
files = list_files(directory_path)
conn = mysql.connector.connect(
    host=os.getenv("DB_SERVER"),
    user=os.getenv("DB_USER"),
    password=os.getenv("DB_PASSWORD"),
    database=os.getenv("DB_SOURCE")
)
cursor = conn.cursor()
k = 0
for i in range(1000):
    sql = "INSERT INTO biodata (NIK, nama, tempat_lahir, tanggal_lahir,
jenis_kelamin, golongan_darah, alamat, agama, status_perkawinan,
pekerjaan, kewarganegaraan) VALUES (%s, %s, %s, %s, %s, %s, %s, %s,
%s, %s)"
    fake_biodata = generate_fake_data()
    original_name = fake_biodata[1]
    fake_biodata[1] = alay_combination(original_name)
    cursor.execute(sql, fake_biodata)
    if i < 600:
        for j in range(10):
            sql = "INSERT INTO sidik_jari (berkas_citra, nama) VALUES
(%s, %s)"
            fake_sidik = ("../test/dataset/" + files[k], original_name)
            cursor.execute(sql, fake_sidik)
            k += 1
conn.commit()

```

13. SchemaConverter.py

Kelas ini berguna untuk mengubah basis data yang tidak terenkripsi menjadi terenkripsi.

```

import os
import mysql.connector
from datetime import datetime

```

```

from dotenv import load_dotenv
def list_files(directory):
    files = [f for f in os.listdir(directory) if
os.path.isfile(os.path.join(directory, f))]
    return files
def xor_encrypt_decrypt(text, key, length):
    encrypted = ''.join(chr(ord(char) ^ key) for char in text)
    return encrypted[:length]
def mod_encrypt_date(date_obj, key):
    year = (date_obj.year + key) % 9999
    month = (date_obj.month + key - 1) % 12 + 1
    day = (date_obj.day + key - 1) % 31 + 1
    while True:
        try:
            encrypted_date = datetime(year, month, day).date()
            break
        except ValueError:
            day -= 1
    return encrypted_date
def mod_decrypt_date(date_obj, key):
    year = (date_obj.year - key) % 9999
    month = (date_obj.month - key - 1) % 12 + 1
    day = (date_obj.day - key - 1) % 31 + 1
    while True:
        try:
            decrypted_date = datetime(year, month, day).date()
            break
        except ValueError:
            day -= 1
    return decrypted_date
load_dotenv("../.env")
directory_path = r'../test/dataset'
files = list_files(directory_path)
source_conn = mysql.connector.connect(
    host=os.getenv("DB_SERVER"),
    user=os.getenv("DB_USER"),
    password=os.getenv("DB_PASSWORD"),
    database=os.getenv("DB_SOURCE")
)
source_cursor = source_conn.cursor()
source_cursor.execute("SELECT NIK, nama, tempat_lahir, tanggal_lahir,
jenis_kelamin, golongan_darah, alamat, agama, status_perkawinan,
pekerjaan, kewarganegaraan FROM biodata")
source_data = source_cursor.fetchall()
source_cursor.execute("SELECT berkas_citra, nama FROM sidik_jari")
source_fingerprint_data = source_cursor.fetchall()
encryption_key = int(os.getenv("ENCRYPTION_KEY"))

```

```

encrypted_data = []
for row in source_data:
    encrypted_row = list(row)
    encrypted_row[0] = xor_encrypt_decrypt(str(encrypted_row[0]),
encryption_key, 16)
    encrypted_row[1] = xor_encrypt_decrypt(encrypted_row[1],
encryption_key, 100)
    encrypted_row[2] = xor_encrypt_decrypt(encrypted_row[2],
encryption_key, 100)
    encrypted_row[3] = mod_encrypt_date(encrypted_row[3], encryption_key)
    encrypted_row[5] = xor_encrypt_decrypt(encrypted_row[5],
encryption_key, 200)
    encrypted_row[6] = xor_encrypt_decrypt(encrypted_row[6],
encryption_key, 200)
    encrypted_row[7] = xor_encrypt_decrypt(encrypted_row[7],
encryption_key, 50)
    encrypted_row[9] = xor_encrypt_decrypt(encrypted_row[9],
encryption_key, 100)
    encrypted_row[10] = xor_encrypt_decrypt(encrypted_row[10],
encryption_key, 50)
    encrypted_data.append(tuple(encrypted_row))
encrypted_fingerprint_data = []
for row in source_fingerprint_data:
    encrypted_row = list(row)
    encrypted_row[0] = xor_encrypt_decrypt(row[0], encryption_key, 200)
    encrypted_row[1] = xor_encrypt_decrypt(row[1], encryption_key, 100)
    encrypted_fingerprint_data.append(tuple(encrypted_row))
dest_conn = mysql.connector.connect(
    host=os.getenv("DB_SERVER"),
    user=os.getenv("DB_USER"),
    password=os.getenv("DB_PASSWORD"),
    database="fingerprint"
)
dest_cursor = dest_conn.cursor()
sql biodata = "INSERT INTO biodata (NIK, nama, tempat_lahir,
tanggal_lahir, jenis_kelamin, golongan_darah, alamat, agama,
status_perkawinan, pekerjaan, kewarganegaraan) VALUES (%s, %s, %s, %s,
%s, %s, %s, %s, %s, %s)"
for row in encrypted_data:
    dest_cursor.execute(sql biodata, row)
sql sidik_jari = "INSERT INTO sidik_jari (berkas_citra, nama) VALUES (%s,
%s)"
for row in encrypted_fingerprint_data:
    dest_cursor.execute(sql sidik_jari, row)
dest_conn.commit()
source_cursor.close()
source_conn.close()

```

```
dest_cursor.close()  
dest_conn.close()
```

4.2 Penjelasan Tata Cara Penggunaan Program

(interface program, fitur-fitur yang disediakan program, dan sebagainya).

Program fingerprint matching ini dapat dijalankan dengan melakukan hal-hal sebagai berikut:

0. Pastikan agar semua *requirement* pada program sudah ter-*install*:
 - .NET (Menjalankan program utama)
 - Python 3.11.9 (Memuat basis data)
 - python-dotenv (Untuk mengatur *environment variables* di Python, jalankan perintah *pip install python-dotenv* untuk meng-*install*)
 - DotNetEnv (Mengatur *environment variables* di C#)
1. *Clone repository* dengan mengetik:

```
git clone https://github.com/hiirrs/Tubes3_fingger-finggeran.git
```

2. Pada root folder, tambahkan file .env yang berisi variabel berikut:

```
DB_SERVER=localhost # Ganti dengan nama server pada perangkat Anda  
DB_USER=root # Ganti dengan nama pengguna pada perangkat Anda  
DB_PASSWORD=12345 # Ganti dengan password pada perangkat Anda  
DB_SOURCE=fingerprint_original # Nama basis data default untuk  
fingerprint yang belum dienkripsi  
ENCRYPTION_KEY=129 # Kunci enkripsi/dekripsi data
```

3. Pindah ke direktori DatabaseLoader:

```
cd DatabaseLoader
```

4. Jalankan perintah berikut untuk menginisialisasi skema basis data:

```
dotnet run
```

5. Jika basis data sidik jari belum tersedia, jalankan perintah berikut untuk membuat *mock* data:

```
python OriginalMockGenerator.py
```

6. Jalankan perintah berikut untuk mengenkripsi basis data:

```
python SchemaConverter.py
```

7. Pindah ke direktori src:

```
cd ../src
```

8. Jalankan perintah berikut untuk menjalankan program:

```
dotnet run
```

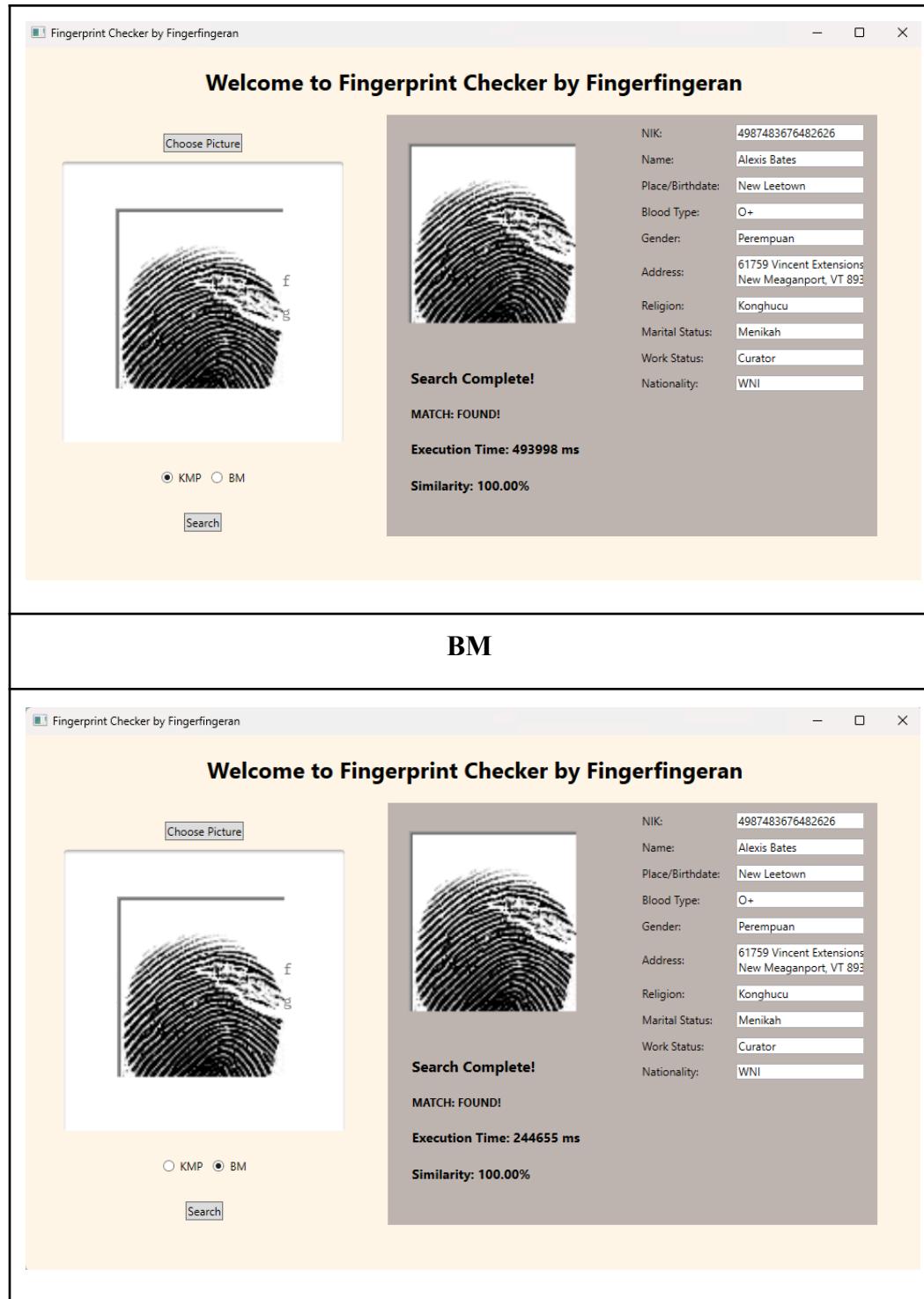
9. Tampilan utama program akan muncul. Masukkan sidik jari yang ingin dicocokkan dan algoritma yang ingin digunakan. Setelah itu, tekan tombol Search untuk mendapatkan hasil pencocokan.

4. 3 Hasil pengujian

4. 3. 1. Pengujian Biasa

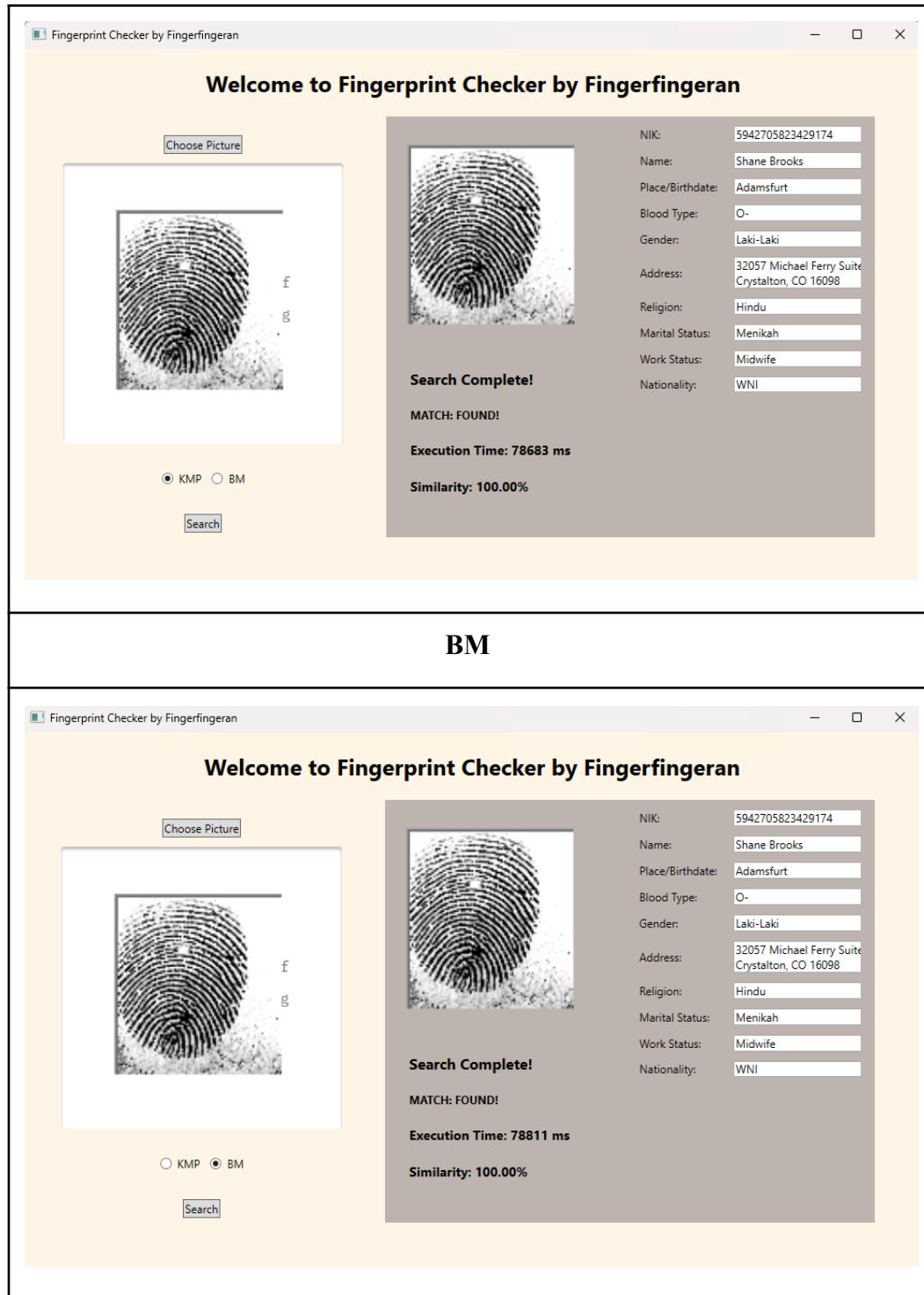
- Gambar 2_F_Left_middle_finger

KMP



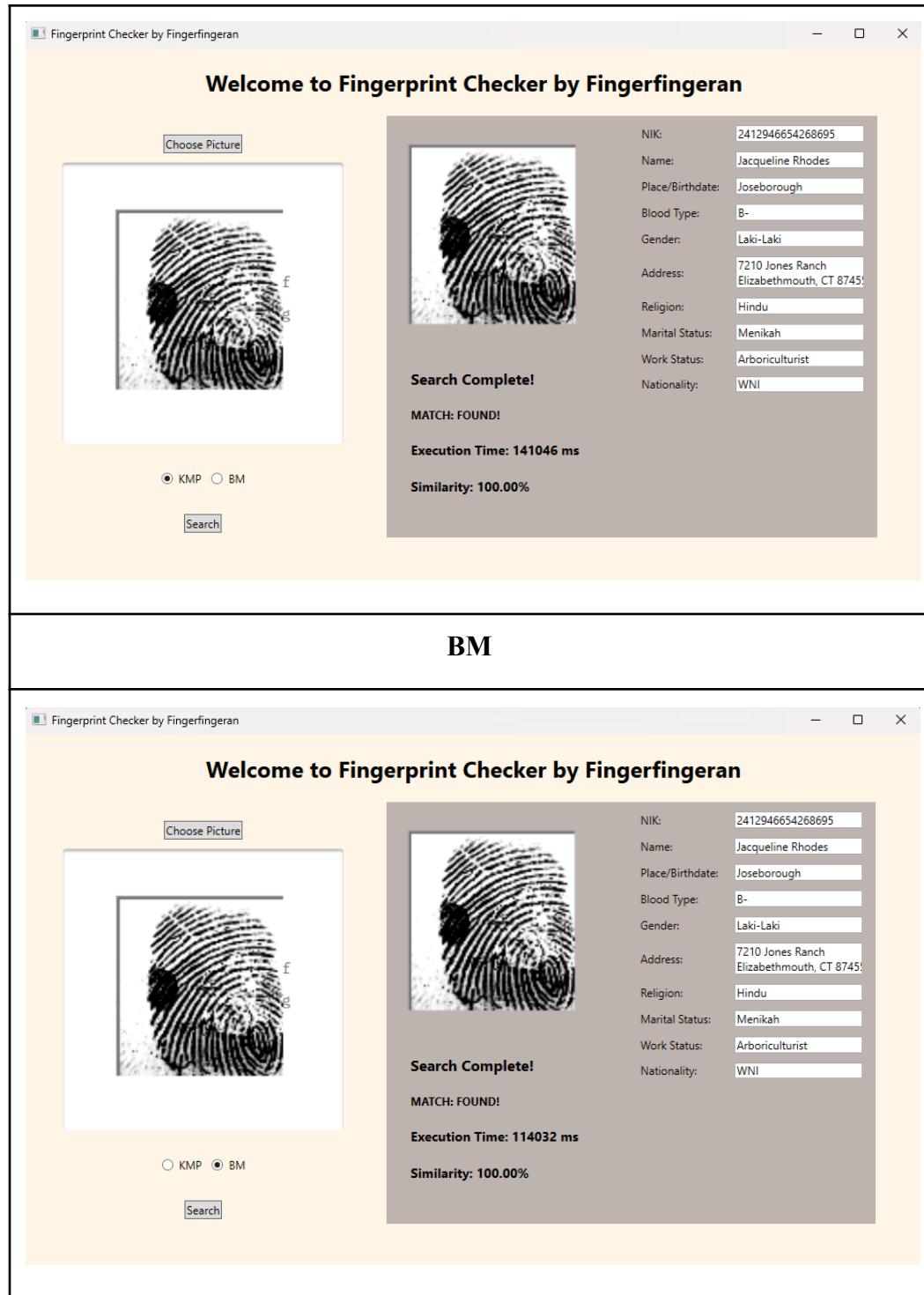
- Gambar 306_M_Left_little_finger

KMP



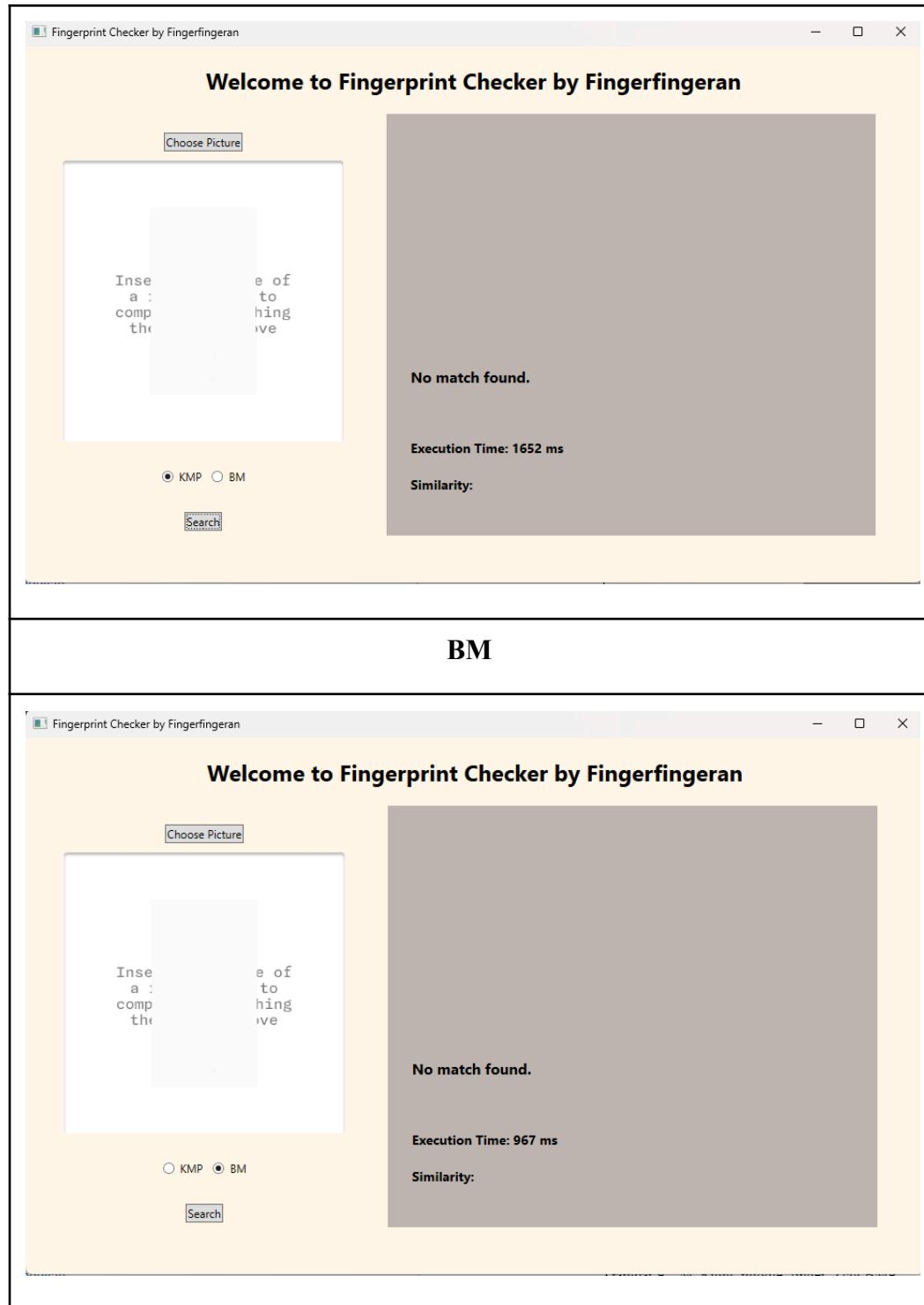
- Gambar 585_M_Left_index_finger

KMP



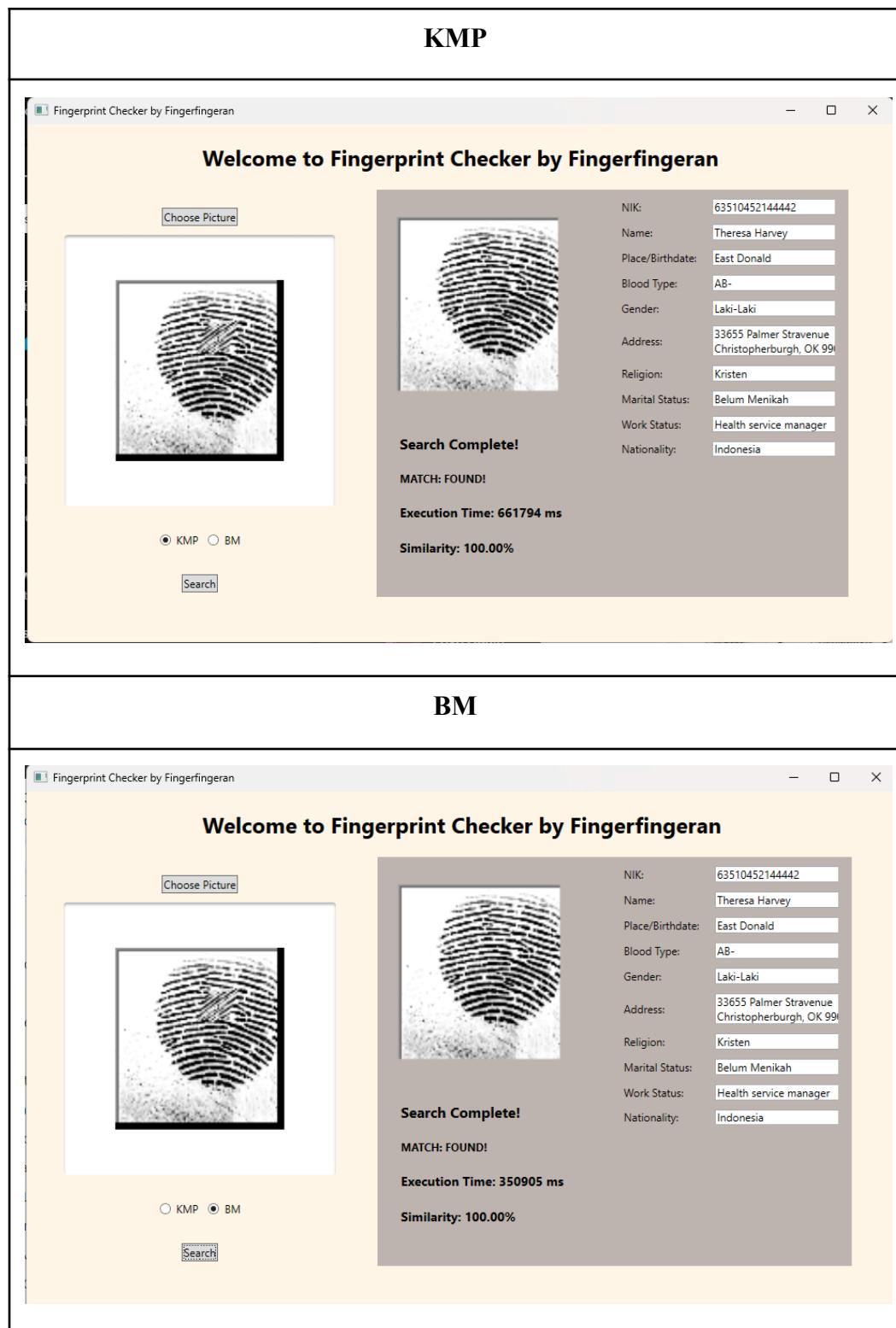
- Gambar bukan sidik jari (gambar putih kosong)

KMP

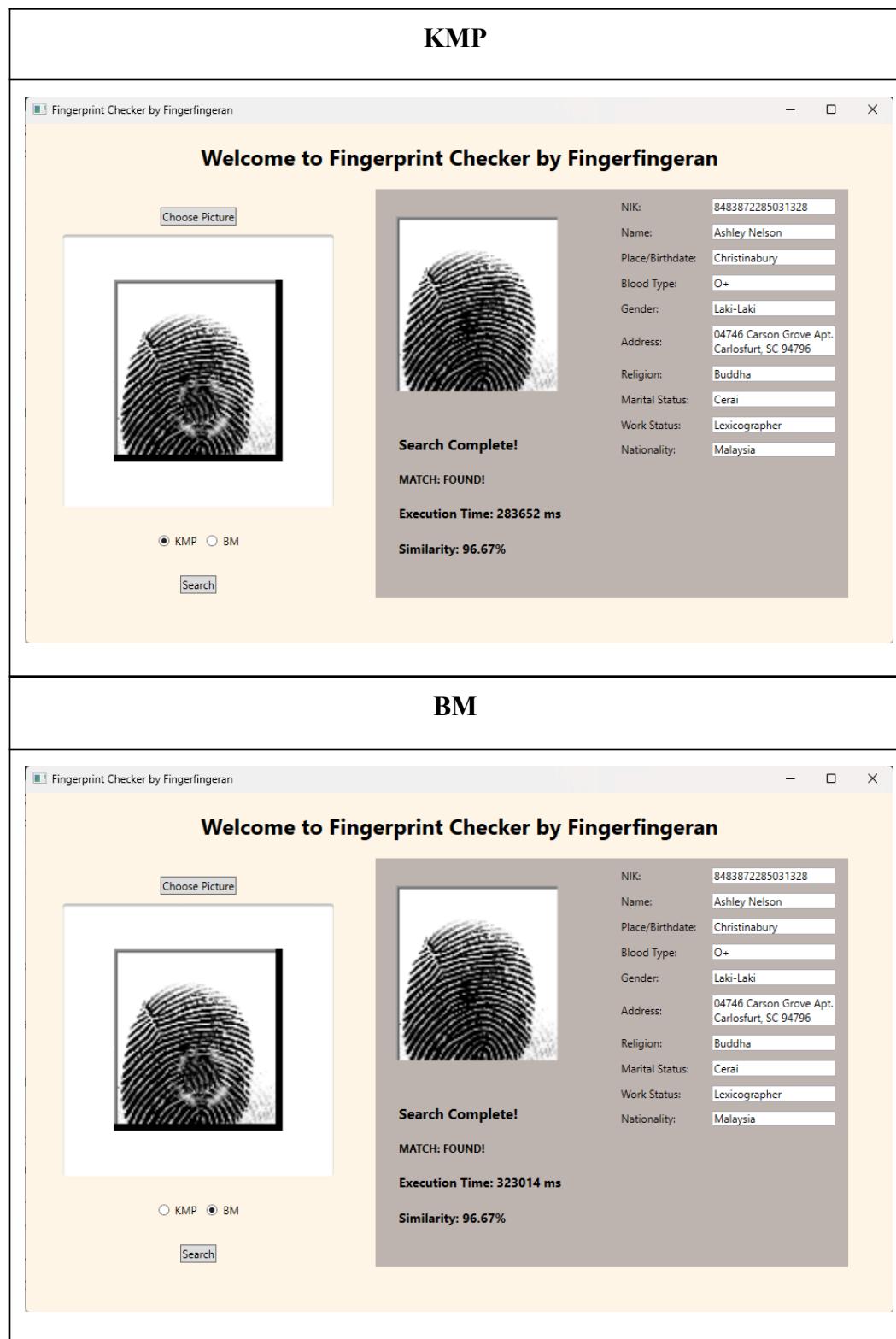


4. 3. 2. Pengujian pada Level Easy

- Gambar 9_M_Right_middle_finger_Zcut.BMP



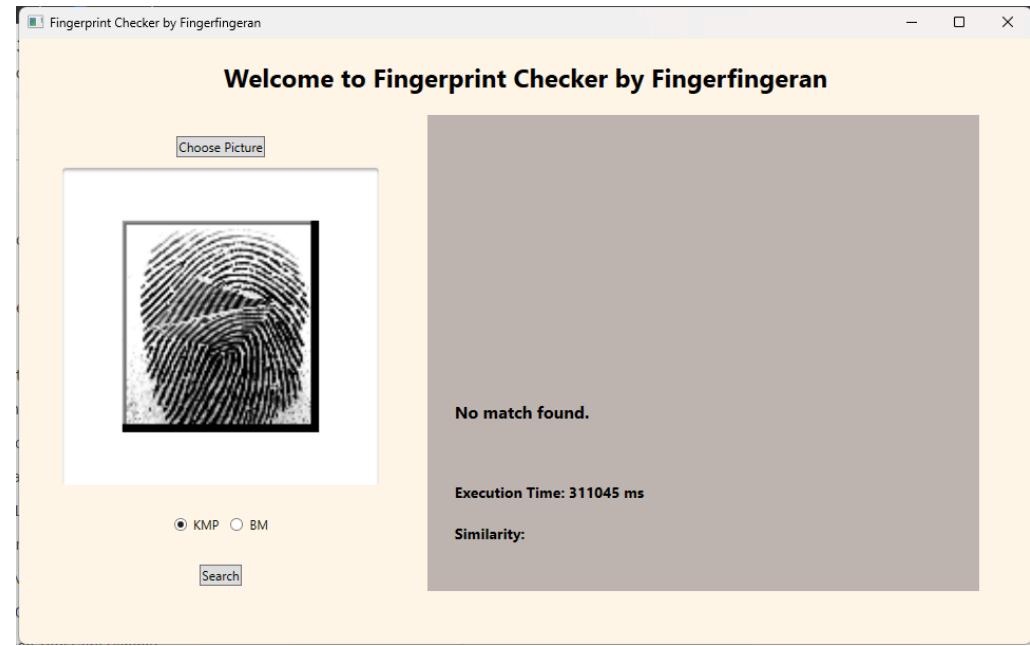
- Gambar 101__M_Left_index_finger_CR.BMP



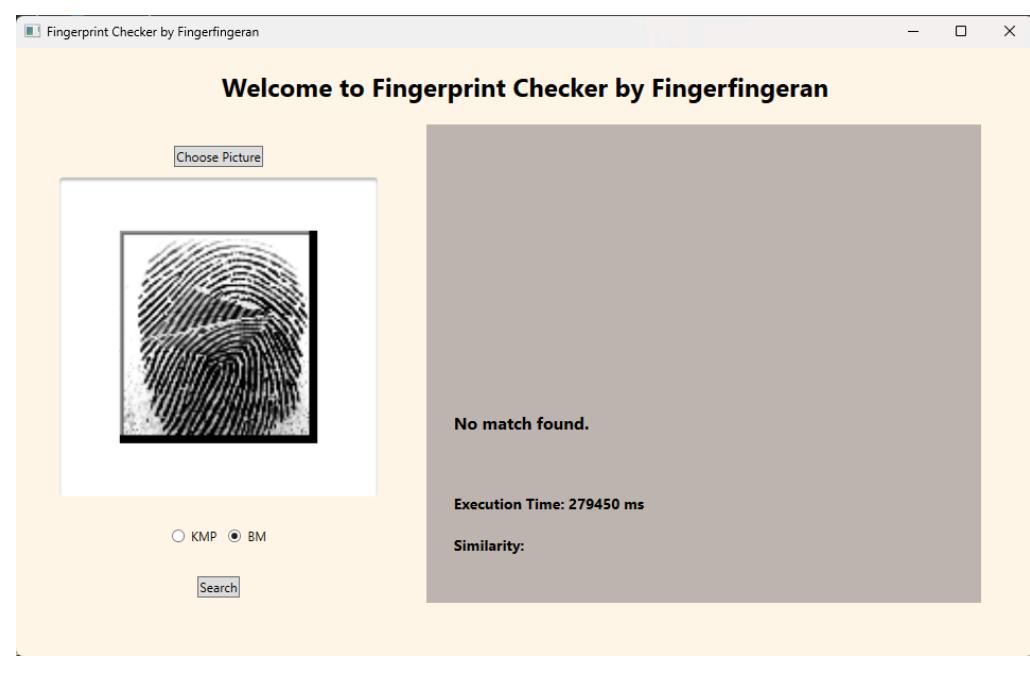
4. 3. 3. Pengujian pada Level Medium

- Gambar 100_M_Left_index_finger_Zcut

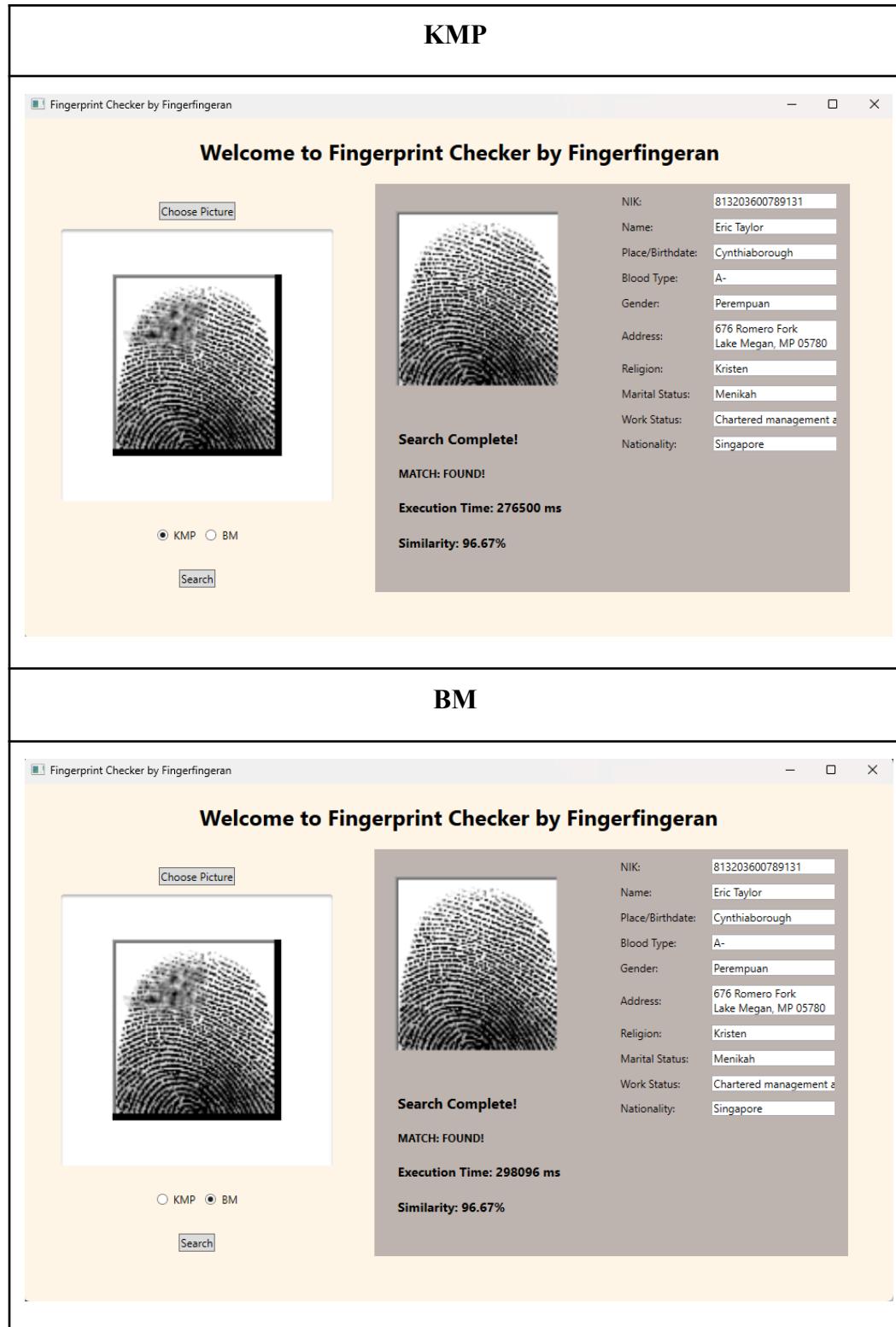
KMP



BM

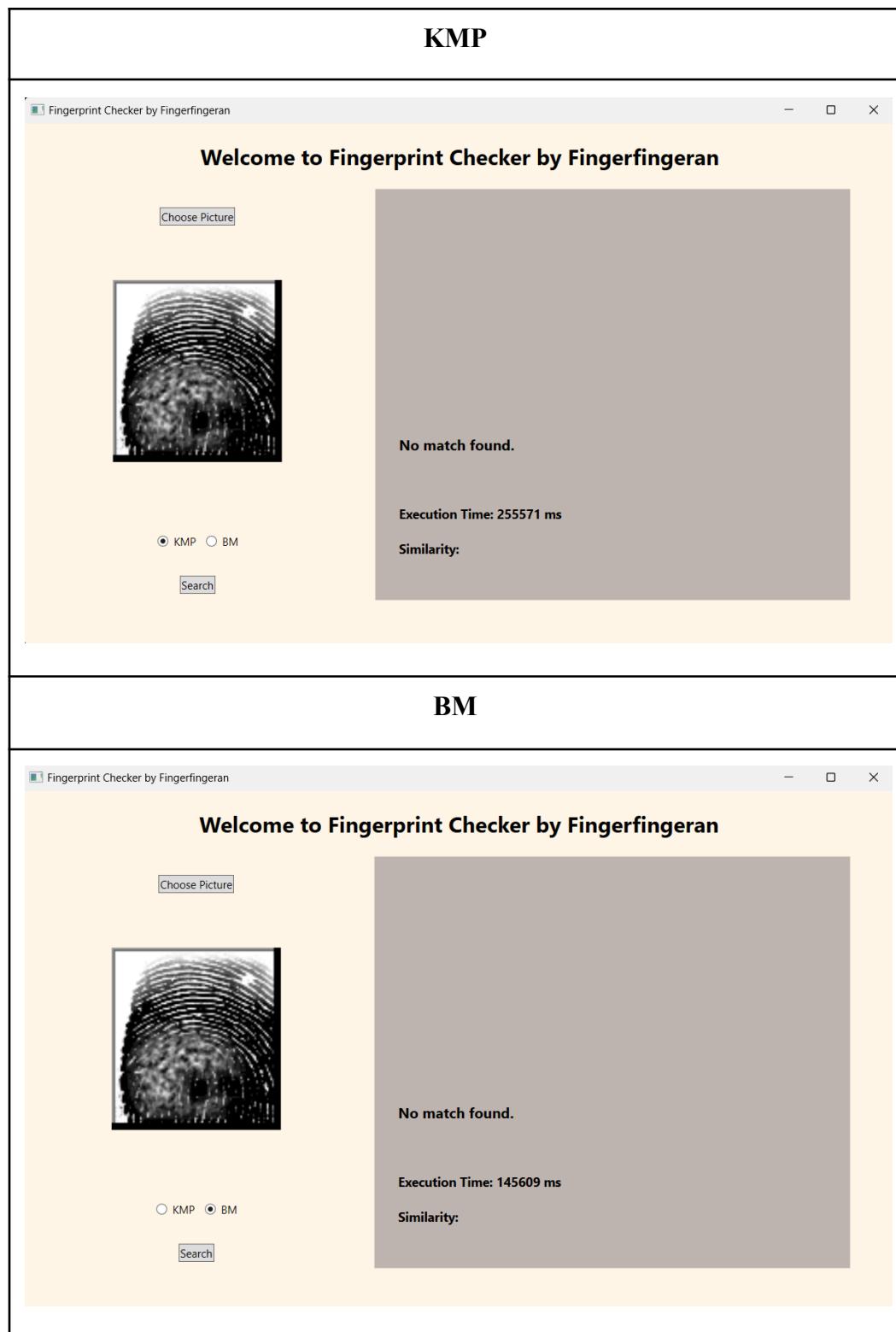


- Gambar 115_F_Right_thumb_finger_Obl

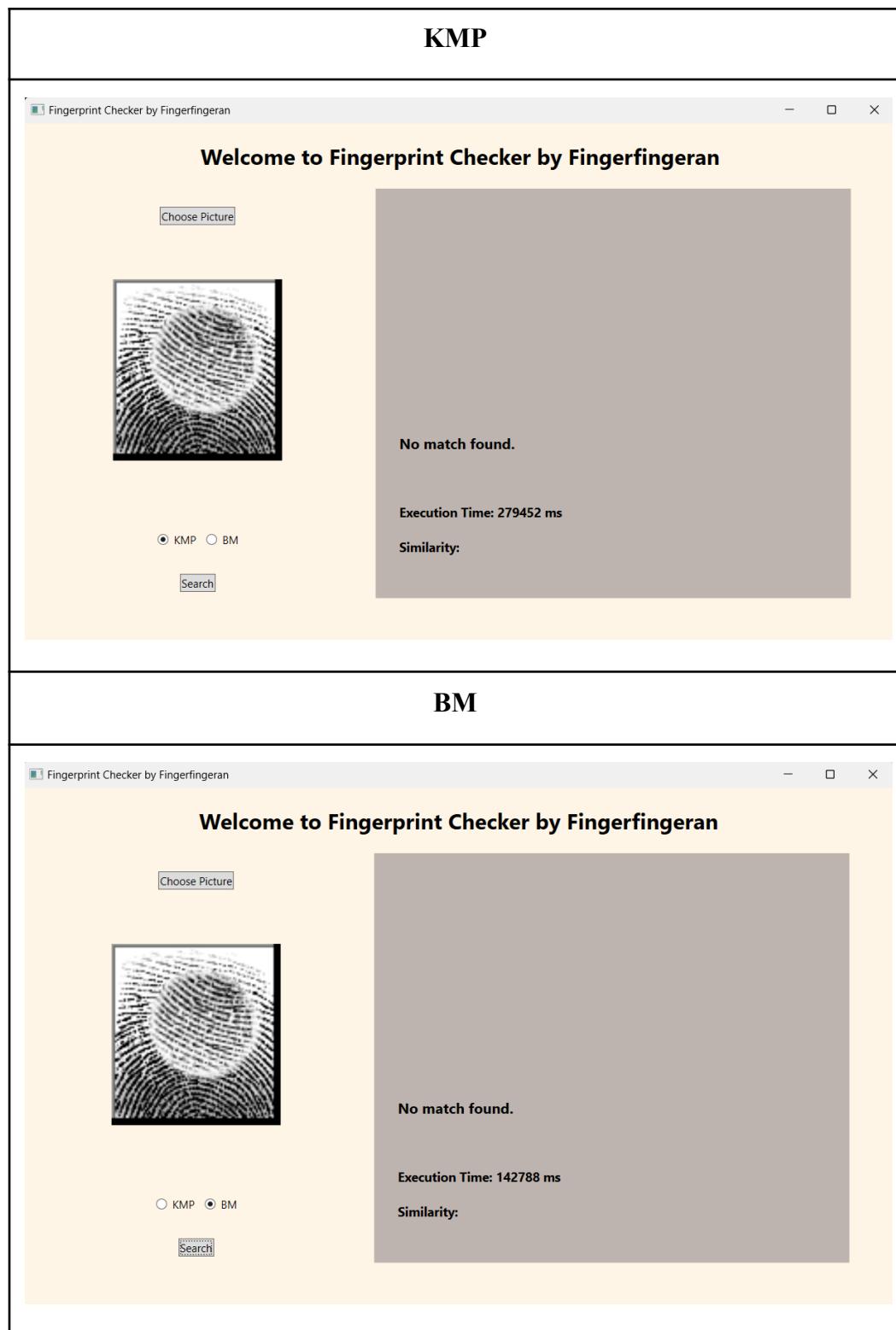


4. 3. 4. Pengujian pada Level Hard

- Gambar 3_M_Right_index_finger_Obl



- Gambar 68_M_Right_thumb_finger_CR

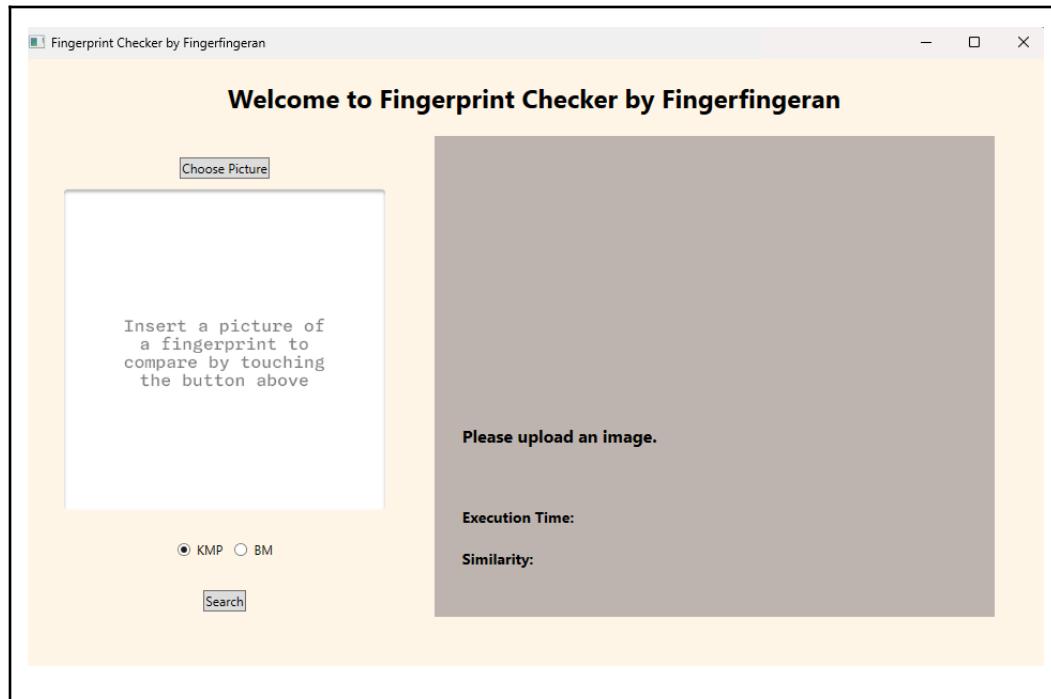


4. 3. 5. Other Cases

- Gambar tampilan awal ketika pertama kali dijalankan



- Gambar tampilan awal ketika tombol ‘search’ ditekan tetapi gambar yang ingin dibandingkan belum ada



4.4 Analisis Hasil Pengujian

Berdasarkan hasil pengujian yang telah dilakukan, didapatkan beberapa analisis sebagai berikut.

1. Hasil biodata memberikan hasil yang sesuai dengan nama yang sudah diperbaiki dari nama *corrupted*. Dapat disimpulkan bahwa regex berhasil dijalankan dengan baik.
2. Pada pengujian normal (kasus gambar tidak ter-*altered*), setiap gambar yang dimasukkan memberikan hasil yang sesuai dengan persentase kemiripan 100% yang berarti sesuai. Hal ini menunjukkan pada kasus normal, program telah berjalan dengan baik.
3. Pada pengujian *altered*, program dapat memberikan hasil yang sesuai pada level easy. Ada pun pada level *medium* tidak semua kasus memberikan hasil, tergantung pada persentase kemiripan yang dihasilkan. Program dapat memberikan hasil tidak ditemukan dikarenakan adanya *threshold* pada program ini yaitu dengan kemiripan 75%. Hal ini dikarenakan lebih baik untuk lebih sulit ditemukan sidik jari yang sesuai dibandingkan salah dalam memberikan data. Oleh karena itu, pada level hard yang diuji, tidak ada kasus yang memiliki tingkat kemiripan diatas threshold. Hal ini menunjukkan bahwa perubahan yang dilakukan pada sidik jari cukup banyak hingga menurunkan nilai kemiripan.
4. Telah diuji juga ketika pengguna mengunggah gambar putih. Program akan mengeluarkan bahwa tidak ditemukan hasil. Hal ini dikarenakan tidak adanya sidik jari pada gambar yang putih polos.

4.5 Implementasi Bonus

Enkripsi adalah proses mengubah informasi atau data menjadi bentuk kode sehingga hanya orang-orang yang memiliki kunci rahasia (disebut kunci dekripsi) dapat membaca data tersebut. Dekripsi adalah proses mengembalikan data yang telah dienkripsi menjadi format aslinya yang bisa dibaca dan dipahami. Proses ini menggunakan kunci dekripsi yang sesuai dengan metode enkripsi yang digunakan untuk mengunci data tersebut.

Pada program ini, kami mengimplementasikan enkripsi XOR. Dalam operasi XOR, ketika dua bit dibandingkan, maka hasilnya adalah 1 jika kedua bit berbeda, dan 0 jika kedua bit sama. Setiap bit dari plaintext di-XOR dengan bit yang sesuai dari kunci.

Enkripsi pada program ini diimplementasikan pada file `MockGenerator.py`. Data dienkripsi sebelum dimasukkan pada basis data.

```
def xor_encrypt_decrypt(text, key, length):
    encrypted = ''.join(chr(ord(char) ^ key) for char in text)
    return encrypted[:length]
def mod_encrypt_date(date_obj, key):
    year = (date_obj.year + key) % 9999
    month = (date_obj.month + key - 1) % 12 + 1
    day = (date_obj.day + key - 1) % 31 + 1
    while True:
        try:
            encrypted_date = datetime(year, month, day).date()
            break
        except ValueError:
            day -= 1
    return encrypted_date
def mod_decrypt_date(date_obj, key):
    year = (date_obj.year - key) % 9999
    month = (date_obj.month - key - 1) % 12 + 1
    day = (date_obj.day - key - 1) % 31 + 1
    while True:
        try:
            decrypted_date = datetime(year, month, day).date()
            break
        except ValueError:
            day -= 1
    return decrypted_date
```

Dekripsi diimplmentasikan pada file `DatabaseManager.cs` yang meliputi cuplikan kode program berikut:

```
private static string DecryptXOR(string encryptedText, byte key)
{
    StringBuilder decryptedText = new StringBuilder();
    foreach (char c in encryptedText)
    {
        decryptedText.Append((char)(c ^ key));
    }
    return decryptedText.ToString();
}
private static DateTime DecryptDate(DateTime encryptedDate, int key)
{
    return encryptedDate.AddYears(-key);
}
```


BAB V

KESIMPULAN DAN SARAN

5. 1 Kesimpulan

Melalui Tugas Besar III Strategi Algoritma ini, kami diminta untuk membuat suatu aplikasi yang dapat mencari kesamaan sidik jari. Kami membuat *user interface* dengan menggunakan Windows Presentation Foundation (WPF) serta bahasa C# untuk algoritma. Melalui pengujian yang kami lakukan, algoritma BM cenderung lebih cepat memberikan hasil karena dapat pencocokan dilakukan menggunakan ascii yang kemungkinan besar tidak memiliki kesamaan prefix dan suffix. Program beberapa kali mengeluarkan hasil tidak ditemukan ketika gambar sidik jari sudah dimodifikasi secara cukup signifikan. Apabila tidak ditemukan kesamaan yang persis, maka digunakan Levenshtein Distance. Selain, penggunaan regex digunakan untuk mengubah format sesuai aturan regex tertentu. Berdasarkan pengujian yang kami lakukan, aplikasi dapat berjalan dengan relatif baik.

5. 2 Saran, Tanggapan, dan Refleksi

Selama pengerjaan Tugas Besar III Strategi Algoritma ini, kami menemukan beberapa hal yang bisa ditingkatkan kedepannya. Hal tersebut adalah sebagai berikut:

1. Algoritma BM dan KMP secara sendiri tidak terlalu baik untuk mencari tingkat kemiripan terbaik karena hanya mencari exact match.
2. Algoritma Regular Expression efektif digunakan untuk menyesuaikan penulisan dengan aturan penulisan tertentu.
3. Aplikasi kami secara keseluruhan belum tentu bisa dikatakan sangat baik. Masih ada bagian-bagian yang dapat diperbaiki agar tentunya menghasilkan aplikasi yang lebih baik dari yang sekarang.
4. Peningkatan dalam penulisan komentar dan dokumentasi untuk mempermudah kerja sama tim dan *maintenance* kode program.

LAMPIRAN

- Tautan *repository* Github:

https://github.com/hiirrs/Tubes3_fingger-finggeran.git

- Tautan video:

<https://youtu.be/EmN14bUlzUs>

DAFTAR PUSTAKA

[1] Rinaldi, M. (n.d.). *Pencocokan string (String matching/pattern matching)* Diakses 23 Mei 2024 melalui

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

[2] Rinaldi, M. (n.d.). *String Matching dengan Regular Expression* Diakses 23 Mei 2024 melalui

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

[3] Dropbox. *Apa itu Enkripsi?* Diakses 8 Juni 2024 melalui

<https://experience.dropbox.com/id-id/resources/what-is-encryption>.

[4] Supatmi, S. (2024). *Fingerprint Identification using Bozorth and Boyer-Moore Algorithm.*

Diakses 23 Mei 2024 melalui
<https://iopscience.iop.org/article/10.1088/1757-899X/662/2/022040>