

# **LAPORAN TUGAS BESAR 3**

## **IF2211 STRATEGI ALGORITMA**



Disusun oleh:

Denise Felicia Tiowanni	(13522013)
Zaki Yudhistira Candra	(13522031)
Muhammad Naufal Aulia	(13522074)

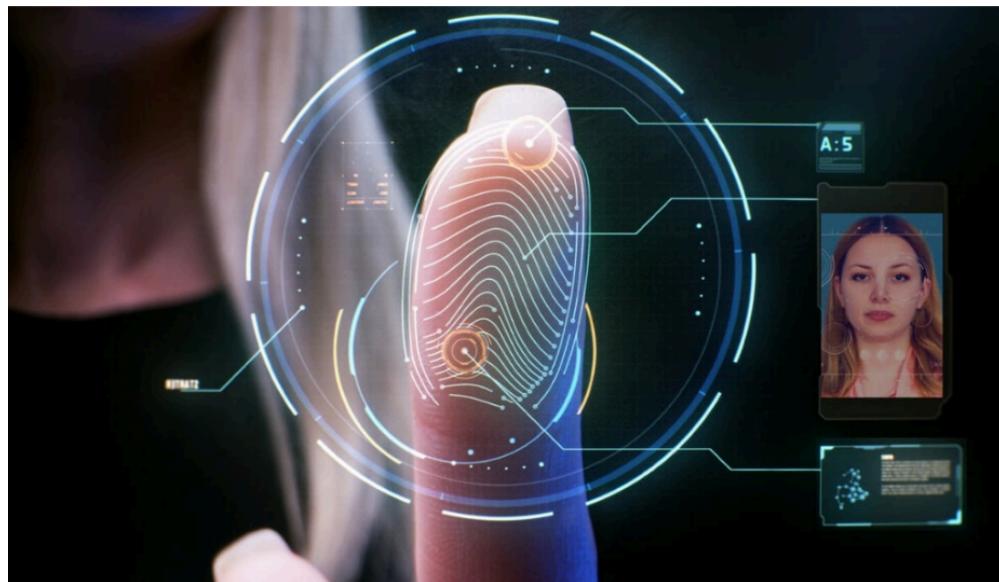
**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
BANDUNG  
2024  
DAFTAR ISI**

<b>DAFTAR ISI.....</b>	<b>0</b>
<b>BAB I.....</b>	<b>1</b>
<b>DESKRIPSI TUGAS.....</b>	<b>1</b>
1.1 Deskripsi Tugas.....	2
<b>BAB II.....</b>	<b>4</b>
2.1 Algoritma KMP (Knuth-Morris-Pratt).....	4
2.2 Algoritma BM (Boyer-Moore).....	4
2.3 Regular Expression.....	5
2.4 Teknik Pengukuran Persentase Kemiripan.....	6
2.4 Aplikasi Desktop Printuyul.....	7
<b>BAB III.....</b>	<b>8</b>
3.1 Langkah-langkah Pemecahan Masalah.....	8
3.2 Proses Penyelesaian Solusi.....	8
3.3 Tentang Aplikasi.....	10
3.4 Contoh Ilustrasi Kasus.....	10
3.4.1 Algoritma BM.....	10
3.4.2 Algoritma KMP.....	12
<b>BAB IV.....</b>	<b>14</b>
4.1 Struktur Program.....	14
4.2 Backend.....	15
4.3 Frontend.....	36
4.4 Tata Cara Penggunaan Program.....	48
4.5 Hasil Pengujian.....	49
4.5.1 Antarmuka Aplikasi.....	49
4.5.2 Field Input.....	50
4.5.3 Uji Coba Kasus.....	55
4.6 Analisis Hasil Pengujian.....	60
<b>BAB V.....</b>	<b>62</b>
5.1 Kesimpulan.....	62
5.2 Saran.....	62
5.3 Refleksi.....	63
5.4 Tanggapan.....	63
<b>DAFTAR PUSTAKA.....</b>	<b>65</b>

## **BAB I**

### **DESKRIPSI TUGAS**

#### **1.1 Deskripsi Tugas**



(Sumber: [https://miro.medium.com/v2/resize:fit:1400/1\\*jxmEbVn2FFWybZslicJCWQ.png](https://miro.medium.com/v2/resize:fit:1400/1*jxmEbVn2FFWybZslicJCWQ.png))

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

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

Dengan menggabungkan teknologi identifikasi sidik jari dan pattern matching, dimungkinkan untuk membangun sistem identifikasi biometrik yang aman, handal, dan mudah digunakan. Sistem

ini dapat diaplikasikan di berbagai bidang, seperti kontrol akses, absensi karyawan, dan verifikasi identitas dalam transaksi keuangan.

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

## **BAB II**

### **LANDASAN TEORI**

#### **2.1 Algoritma KMP (Knuth-Morris-Pratt)**

Algoritma Knuth-Morris-Pratt (KMP) adalah algoritma pencocokan string yang menghindari pencarian ulang yang tidak perlu dengan menggunakan informasi dari pola itu sendiri. Langkah pertama dalam algoritma KMP adalah membangun tabel border atau tabel fail, yang membantu dalam menentukan seberapa jauh pola dapat digeser jika terjadi ketidakcocokan. Setelah tabel border dibuat, algoritma KMP menggunakan tabel ini untuk mencari pola dalam teks.

#### **2.2 Algoritma BM (Boyer-Moore)**

Algoritma Boyer-Moore (BM) adalah algoritma pencocokan string yang menggunakan dua heuristik utama untuk mempercepat pencocokan: Bad Character Heuristic atau Character-Jump dan Good Suffix Heuristic atau Looking-Glass. Kedua heuristik ini membantu mengurangi jumlah perbandingan yang diperlukan dengan memungkinkan pola untuk digeser lebih jauh saat terjadi ketidakcocokan, sehingga mempercepat proses pencocokan secara signifikan.

##### a. Bad Character Heuristic (Character-Jump)

Bad Character Heuristic membantu menentukan berapa banyak pola bisa digeser ketika terjadi ketidakcocokan karakter. Saat terjadi ketidakcocokan karakter antara pola dan teks, heuristik ini menggunakan tabel yang menyimpan posisi terakhir dari setiap karakter dalam pola. Jadi, jika karakter teks yang sedang dibandingkan tidak cocok dengan karakter pola, kita hanya perlu melihat tabel untuk mengetahui seberapa jauh pola dapat digeser sehingga karakter teks dapat sejajar dengan kemunculan terakhir karakter tersebut dalam pola atau melompati karakter yang tidak muncul sama sekali. Berikut merupakan implementasi pembuatan Bad Character Table kami dalam algoritma BM.

##### b. Good Suffix Heuristic (Looking-Glass)

Good Suffix Heuristic disebut sebagai Looking Glass Heuristic karena berfokus pada pencocokan bagian belakang pola (suffix) untuk mengoptimalkan pergeseran pola saat terjadi ketidakcocokan. Ketika ketidakcocokan terjadi setelah beberapa karakter akhir dari pola cocok dengan teks, heuristik ini menentukan pergeseran pola berdasarkan bagian dari pola yang cocok. Heuristik ini menggunakan tabel yang mencatat berapa jauh

pola bisa digeser sehingga bagian yang sudah cocok tetap cocok di pergeseran berikutnya. Jadi, jika sebuah suffix dari pola sudah cocok, maka heuristik ini mencari pergeseran terbesar yang menjaga kecocokan suffix tersebut. Berikut merupakan implementasi pembuatan Good Suffix Table kami dalam algoritma BM.

### 2.3 Regular Expression

Regular Expression (Regex) dalam permasalahan string matching dapat didefinisikan sebagai urutan karakter yang menentukan pola pencarian. Regex biasanya digunakan untuk menemukan atau mengganti teks dalam string dengan mengidentifikasi pola tertentu yang spesifik.

Regex terdiri dari beberapa komponen, diantaranya:

- a. Karakter Literal: Mencocokkan karakter itu sendiri. Misalnya, a akan mencocokkan karakter 'a' dalam teks.
- b. Metakarakter: ., \*, +, ?, [ ], {, }, \, |, ^, dan \$.
- c. Karakter Kelas: Didefinisikan dengan tanda kurung siku [ ]. Contohnya, [abc] akan mencocokkan 'a', 'b', atau 'c'. Rentang juga dapat digunakan, misalnya [a-z] untuk mencocokkan semua huruf kecil dari a hingga z.
- d. Anchor: Digunakan untuk mencocokkan posisi dalam string, contohnya adalah ^ untuk awal string dan \$ untuk akhir string.
- e. Quantifiers: Menentukan berapa banyak kemunculan karakter atau grup yang diinginkan.
  - \*: 0 atau lebih
  - +: 1 atau lebih
  - ?: 0 atau 1
  - {n}: Tepat n kali
  - {n,}: Setidaknya n kali
  - {n,m}: Antara n dan m kali
- f. Grup dan Alternasi
  - Kurung ( ) digunakan untuk mengelompokkan ekspresi.
  - | (pipe) digunakan untuk menunjukkan salah satu pola alternatif. Misalnya, (a|b) akan mencocokkan 'a' atau 'b'.

Pada tugas besar ini, regex bertujuan untuk mencocokkan nama asli dengan nama-nama yang telah diubah dengan bahasa alay Indonesia (misalnya, menggunakan angka atau huruf besar/kecil serta singkatan) yang terdapat dalam database biodata.

#### 2.4 Teknik Pengukuran Persentase Kemiripan.

Teknik pengukuran persentase kemiripan gambar sidik jari dilakukan melalui beberapa tahap dan menggunakan beberapa algoritma, yaitu algoritma utama berupa Knuth-Morris-Pratt (KMP) atau Boyer-Moore (BM), serta Hamming Distance. Algoritma Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM) digunakan untuk melakukan pencarian pola secara tepat (*exact matching*). Jika algoritma ini menemukan pola yang cocok secara tepat dalam data referensi, maka gambar sidik jari dianggap identik dengan referensi tersebut. Dengan begitu, program akan langsung menganggap bahwa gambar sidik jari cocok dengan referensi dengan persentase kemiripan 100%.

Namun dalam kondisi tidak ditemukan kecocokan yang tepat menggunakan algoritma KMP atau BM, program akan menggunakan pendekatan Hamming Distance untuk mengukur kemiripan. Dengan distance yang dimaksud adalah perhitungan jumlah posisi di mana dua string dengan panjang yang sama berbeda. Persentase kemiripan kemudian dihitung berdasarkan jarak Hamming dengan rumus:

$$\text{similarity} = 1.0 - (\text{distance}/\text{pattern.Length})$$

Proses ini dilakukan untuk setiap gambar referensi terpotong (*cropped reference images*). Hasilnya adalah gambar dengan persentase kemiripan tertinggi yang akan diambil sebagai gambar yang paling mirip. Setelah menghitung persentase kemiripan, program kemudian membandingkan hasil tersebut dengan *threshold* yang telah ditentukan: jika persentase kemiripan di atas 50%, gambar sidik jari tersebut dikategorikan sebagai "mirip", jika persentase kemiripan di bawah 50%, gambar sidik jari tersebut dianggap "tidak ditemukan". Pemilihan angka 50% ini didasarkan atas intuisi dan percobaan berulang pencarian sidik jari dengan gambar sidik jari masukkan yang telah "dirusak". Dengan begitu kami memutuskan menggunakan angka tersebut sebagai batas yang cukup baik untuk program kami.

## **2.4 Aplikasi Desktop Printuyul**

Printuyul merupakan sebuah aplikasi yang memungkinkan pengguna untuk melakukan pencarian fingerprint dari foto masukan fingerprint serta mengetahui biodata dari pemilik fingerprint tersebut. Printuyul akan meminta pengguna memilih algoritma pencarian, apakah dengan Boyer-Moore ataupun Knuth-Morris-Pratt. Namun, apabila dengan kedua algoritma tersebut tidak ditemukan pencocokan yang *exact*, maka aplikasi akan tetap melakukan pencarian dengan menggunakan Hemming Distance. Aplikasi ini dibangun menggunakan bahasa pemrograman C# untuk logika pemrosesan data, backend, maupun frontend. Untuk pembangunan frontend, digunakan .NET Avalonia.

## **BAB III**

### **ANALISIS PEMECAHAN MASALAH**

#### **3.1 Langkah-langkah Pemecahan Masalah**

Persoalan utama dalam masalah yang dideskripsikan pada Bab 1 adalah melakukan identifikasi biometrik dari citra sidik jari untuk mendapatkan identitas pemilik sidik jari yang sesuai. Permasalahan ini dapat dipecahkan dengan menggunakan pendekatan manipulasi citra menjadi karakter ASCII untuk dilakukan *pattern matching* menggunakan dua algoritma, yakni Boyer-Moore dan Knuth Morris Pratt.

Untuk mencapai hal tersebut, gambar sidik jari akan terlebih dahulu diolah dengan mengubahnya ke dalam bentuk biner, kemudian dikelompokkan setiap 8 bit kode biner untuk diubah lagi menjadi bentuk karakter ASCII 8-bit, karakter inilah yang menjadi perwakilan proses *pattern matching*. Dalam permasalahan ini, string *pattern* yang akan dicocokan dengan string data berasal dari gambar sidik jari masukan yang telah di-*crop* hingga ukurannya menjadi 1x64 pixel, sementara string data berasal dari gambar sidik jari secara penuh agar dapat dilakukan pergeseran pencocokan string *pattern* dengan string tersebut. Jika tidak ditemukan *exact match* dari kedua algoritma tadi, maka akan dilanjutkan dengan menggunakan algoritma Hamming Distance yang mengharuskan gambar data untuk dilakukan proses *crop* yang sama agar kedua string dengan panjang sama dapat dibandingkan.

Setelah *path* gambar yang mirip dengan gambar yang diunggah pengguna ditemukan, program akan mengambil nama pemilik pengguna dengan *path* gambar tersebut. Nama tersebut kemudian akan diubah menjadi sebuah pola *regular expression* yang akan digunakan untuk mencocokkan nama alay pada database biodata. Apabila nama alay cocok dengan ekspresi *regular expression* yang diberikan, maka program akan mengembalikan biodata KTP orang tersebut, seperti tanggal lahir, alamat, agama, golongan darah, dst. Namun, apabila tidak ditemukan, maka program akan mengembalikan pesan bahwa tidak ditemukan orang dengan sidik jari yang diunggah.

Ukuran gambar yang diambil dalam *pattern matching* ini, yaitu 1x64 pixel, ditetapkan atas dasar kemudahan string dalam proses pencocokannya pada suatu baris di teks (ASCII gambar referensi). Ukuran 64 pixel digunakan setelah mencoba berbagai ukuran

lain, dan didapat bahwa yang paling efektif namun masih efisien dalam menentukan kemiripan dua gambar ialah ukuran ini. Hal ini karena 64 pixel akan menjadi string ASCII dengan panjang 8 karakter yang tidak terlalu pendek untuk dibandingkan dengan gambar lain dan tidak terlalu panjang sehingga masih mendapat eksekusi dengan waktu yang masuk akal. Namun dengan begitu, didapat kekurangan yakni panjang gambar yang dapat diproses tidak boleh berukuran lebih kecil dari 64 pixel.

### 3.2 Proses Penyelesaian Solusi

#### 3.2.1 Solusi dengan Algoritma KMP

Algoritma Knuth-Morris-Pratt (KMP) adalah algoritma pencarian string yang menggunakan informasi dari pola untuk menghindari perbandingan yang tidak perlu dan dapat melakukan "lompatan" berdasarkan *border function*, membuatnya lebih efisien dalam banyak kasus. Proses penyelesaian solusi menggunakan algoritma KMP pada masalah identifikasi sidik jari melibatkan langkah-langkah berikut:

1. Mempersiapkan border: dengan menggunakan *border function*, akan didapatkan nilai border yang sesuai untuk tiap karakter pada pola (*pattern*), yakni prefiks terbesar dari pola[0 .. j-1] yang juga merupakan sufiks dari pola[1 .. j-1].
2. Pencarian: Mulai pencocokan pola dengan teks dari kiri ke kanan.
3. Perbandingan: Bandingkan karakter dari pola dengan karakter teks. Jika karakter cocok, lanjutkan ke karakter berikutnya pada pola dan teks. Jika karakter tidak cocok, gunakan nilai dari *border function* untuk menentukan posisi baru pada pola tanpa mengulangi perbandingan yang sudah dilakukan.
4. Pergerakan Pola: Geser pola sesuai dengan nilai dari hasil *border function*. Jika perbedaan antara pola dengan teks terjadi di awal perbandingan, pergeseran pola dilakukan menyamai algoritma brute force, yakni per karakter.
5. Ulangi: Ulangi langkah 3 dan 4 sampai pola ditemukan atau teks (dalam hal ini ASCII dari *reference image*) habis.

#### 3.2.1 Solusi dengan Algoritma BM

Algoritma Boyer-Moore (BM) adalah algoritma pencarian string yang memanfaatkan informasi pada pola untuk melakukan lompatan pencarian yang lebih jauh, mengurangi

jumlah perbandingan yang diperlukan. Proses penyelesaian solusi menggunakan algoritma BM pada masalah identifikasi sidik jari melibatkan langkah-langkah berikut:

1. Mempersiapkan Tabel Heuristik: Buat tabel Bad Character dan Good Suffix untuk pola (*pattern*) dari sidik jari yang diunggah pengguna.
2. Pencarian: Mulai pencocokan pola dengan teks dari kanan ke kiri.
3. Perbandingan: Bandingkan karakter dari ujung kanan pola dengan karakter teks. Jika terjadi ketidakcocokan, gunakan tabel Bad Character dan Good Suffix untuk menentukan seberapa jauh pola bisa digeser.
4. Pergerakan Pola: Geser pola sesuai dengan nilai terbesar dari kedua heuristik.
5. Ulangi: Ulangi langkah 3 dan 4 sampai pola ditemukan atau teks (dalam hal ini ASCII dari *reference image*) habis.

### 3.3 Tentang Aplikasi

Fitur fungsional pada aplikasi *desktop* terdiri dari:

1. Pencarian Fingerprint: Pengguna dapat mengunggah foto fingerprint untuk dilakukan pencarian di database.
2. Pemilihan Algoritma Pencarian: Pengguna dapat memilih algoritma pencarian (Boyer-Moore atau Knuth-Morris-Pratt).
3. *Fallback* Pencarian: Jika pencocokan *exact* tidak ditemukan, aplikasi akan menggunakan Hemming Distance untuk mencari fingerprint yang paling mendekati.
4. Informasi Biodata: Aplikasi menampilkan biodata pemilik fingerprint yang ditemukan.

Arsitektur aplikasi yang dibangun terdiri atas dua bagian utama, yaitu backend dan frontend. Backend dibangun menggunakan bahasa pemrograman C# dan bertugas untuk melakukan pencarian fingerprint di database. Backend menerima data input berupa foto fingerprint dan algoritma yang ingin digunakan dari frontend, kemudian melakukan pencarian fingerprint dan mengirimkan data output ke frontend. Sementara itu, frontend dibangun menggunakan .NET Avalonia dan bertugas untuk menampilkan antarmuka aplikasi yang memungkinkan pengguna untuk mengunggah fingerprint, memilih algoritma pencarian, dan menampilkan hasil pencarian berupa biodata pemilik fingerprint.

### 3.4 Contoh Ilustrasi Kasus

Misalkan seorang polisi menemukan sebuah sidik jari pada suatu *crime scene*. Polisi tersebut ingin mengetahui pemilik sidik jari tersebut, sehingga ia mengambil foto dari sidik jari tersebut, mengunggahnya pada program untuk mencocokannya pada database. Untuk proses ilustrasi pencocokan string, kita misalkan *pattern* yang didapat dari foto yang diunggah adalah “asksdjfg” dan *text reference image* yang dibandingkan adalah “ggjfaksasksdjfghhj”.

#### 3.4.1 Algoritma BM

Dari contoh kasus diatas, didapat bahwa:

- Pattern (P): asksdjfg
- Text (T): ggjfaksasksdjfghhj

Selanjutnya, akan dibuat Bad Character Table (*last occurrence table*) untuk *pattern* “asksdjfg” yaitu sebagai berikut.

Huruf	a	s	k	d	j	f	g
LO	0	3	2	4	5	6	7

Dengan tabel tersebut, berikut adalah perbandingan dan *shift* yang dilakukan ketika pencocokan string. Untuk karakter yang *last occurrence* nya lebih besar dari posisi *pattern* (indeks j) saat ini, maka akan digunakan rumus  $i_{new} = i_{old} + m (\text{ukuran pattern}) - j$

Sementara itu, apabila karakter mempunyai *last occurrence* yang lebih kecil dari posisi *pattern* saat ini, maka akan digunakan rumus  $i_{new} = i_{old} + m (\text{ukuran pattern}) - (LO + 1)$ . Kemungkinan terakhir adalah ketika karakter pada *text* tidak ditemukan pada *pattern*, atau ketika  $LO = -1$ , maka akan digunakan rumus  $i_{new} = i_{old} + m (\text{ukuran pattern})$ .

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
text	g	g	j	f	a	k	s	a	s	k	s	d	j	f	g	h	h	j
j	a	s	k	s	d	j	f	g										

Algoritma BM dilakukan dari belakang pattern. Dapat dilihat bahwa ‘a’ dan ‘g’ tidak cocok, sehingga kita cek last occurrence ‘a’ di Bad Character Table, yaitu 0. Sementara itu indeks j dari ‘g’ adalah 7. Karena  $2 < 7$ , maka  $i_{new} = 7 + 8 - (0 + 1) = 14$ . Dengan demikian, kita akan shift text sehingga ujung pattern akan sama dengan indeks i yang baru.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
text	g	g	j	f	a	k	s	a	s	k	s	d	j	f	g	h	h	j
j								a	s	k	s	d	j	f	g			

Hasil akhirnya, kita menemukan pola “asksdjfg” ditemukan pada teks “ggjfaksasksdjfghhj” (yaitu dari posisi 8).

### 3.4.2 Algoritma KMP

Dari contoh kasus diatas, didapat bahwa:

- Pattern (P): asksdjfg
- Text (T): ggjfaksasksdjfghhj

Selanjutnya, akan digunakan *border function* sehingga menghasilkan tabel border untuk pattern “asksdjfg” sebagai berikut:

Huruf	a	s	k	s	d	j	f	g
L0	0	0	0	1	0	0	0	0

Dengan tabel tersebut, berikut adalah perbandingan dan *shift* yang dilakukan ketika pencocokan string.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
text	g	g	j	f	a	k	s	a	s	k	s	d	j	f	g	h	h	j
j	a	s	k	s	d	j	f	g										

Karena pada indeks pertama sudah terjadi *mismatch*, maka dilakukan pergeseran per satu karakter layaknya pada algoritma brute force. Berdasarkan tabel perbandingan di atas, maka pergeseran seperti ini terjadi hingga teks dengan indeks i ke-4:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
text	g	g	j	f	a	k	s	a	s	k	s	d	j	f	g	h	h	j
j					a	s	k	s	d	j	f	g						

Dari progress di atas, *mismatch* terjadi pada teks indeks i ke-5 dengan huruf pada polanya adalah huruf s. Mengacu pada tabel border, s mempunyai nilai 0 sehingga pergeseran pola dimulai dari indeks j ke-0 pada i ke-5. Begitu seterusnya dilakukan dengan mengacu pada nilai di tabel border hingga akhirnya ditemukan pola “asksdjfg” pada teks “ggjfaksasksdjfghhj” (yaitu dari posisi 8).

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
text	g	g	j	f	a	k	s	a	s	k	s	d	j	f	g	h	h	j
j								a	s	k	s	d	j	f	g			

Dengan demikian, kita mengetahui bahwa *text reference image* yang cocok dengan *pattern* sidik jari yang diambil posisi adalah “ggjfaksasksdjfghhj”. Selanjutnya, kita akan mengambil nama dari pemilik citra sidik jari tersebut dari database. Misalkan namanya adalah “Densu”, maka kita akan mengubah “Densu” menjadi pola *regular expression*, yaitu [dD][eE3]?[nN][sS5][uU]?. Pola tersebut nantinya akan dipakai untuk mengambil biodata orang yang sesuai pada *database* pola.

## **BAB IV**

### **IMPLEMENTASI DAN PENGUJIAN**

#### **4.1 Struktur Program**

```
Tubes3_Printuyul /  
|   └── doc/  
|       └── Printuyul  
|   └── src/  
|       ├── Encryption/  
|       |   ├── encrypt_csv.cs  
|       |   ├── EncryptionHelper.cs  
|       |   ├── InsertBiodataFromCSV.cs  
|       |   ├── InsertSidikJariFromCSV.cs  
|       |   └── Program.cs  
|       ├── FingerprintApi/  
|       |   ├── BooyerMoore.cs  
|       |   ├── DataController.cs  
|       |   ├── EncryptionHelper.cs  
|       |   ├── FingerData.cs  
|       |   ├── FingerprintApi.csproj  
|       |   ├── FingerprintApi.http  
|       |   ├── FingerprintController.cs  
|       |   ├── FingerprintMatcher.cs  
|       |   ├── ImageConverter.cs  
|       |   ├── KnuthMorrisPratt.cs  
|       |   ├── KTPData.cs  
|       |   ├── LCS.cs  
|       |   ├── Program.cs  
|       |   ├── Regex.cx  
|       |   ├── EncryptedData.db  
|       |   └── MainData.db  
|   └── newjeans_alvalonia/
```

```

|   |
|   |   └── assets # Folder gambar
|   |
|   |   └── App.axaml.cs
|   |
|   |   └── AppSatate.cs
|   |
|   |   └── KTPData.cs
|   |
|   |   └── MainWindow.axaml.cs
|   |
|   |   └── MessageBox.axaml.cs
|   |
|   |   └── Program.cs
|   |
|   |   └── SeconWindow.axaml.cs
|   |
|   └── ThirdWindow.axaml.cs
|
└── test/
    |
    |   └── CsvFiles # Berisi file-file CSV
    |
    └── DataBackup # Berisi backup basis data
|
└── run.bat
|
└── setup.bat
|
└── readme.md

```

Program ini dibagi menjadi dua bagian utama yaitu frontend dan backend. Backend program ini berfungsi untuk menerima data dari frontend, melakukan pemrosesan data, dan mengirimkan hasil pemrosesan data kembali ke frontend. Pada folder lib di backend ini berisi logika utama untuk algoritma BFS, IDS, utilitas untuk scraping dan membentuk sebuah path. Sementara frontend program ini berfungsi untuk mengirimkan data ke backend, menerima hasil pemrosesan data dari backend, dan menampilkan hasil pemrosesan data kepada pengguna. Untuk memudahkan, digunakan Docker yang dapat langsung mengintegrasikan keduanya.

## 4.2 Backend

Fungsi yang dibangun guna menunjang keberjalanan algoritma dan program terbagi dalam beberapa *file*, di antaranya adalah:

### 1. BoyerMoore.cs

Melakukan pencocokan string dengan algoritma Boyer-Moore.

<b>BuildBadCharacterTable</b>
private int[] BuildBadCharacterTable(string pattern)

```

{
    int[] badCharTable = new int[256];
    int patternLength = pattern.Length;

    for (int i = 0; i < 256; i++)
    {
        badCharTable[i] = -1;
    }

    for (int i = 0; i < patternLength; i++)
    {
        badCharTable[(int)pattern[i]] = i;
    }

    return badCharTable;
}

```

#### **BuildGoodSuffixTable**

```

private int[] BuildGoodSuffixTable(string pattern)
{
    int m = pattern.Length;
    int[] goodSuffixTable = new int[m];
    int[] suffixes = new int[m];

    for (int i = 0; i < m; i++)
    {
        suffixes[i] = -1;
        goodSuffixTable[i] = m;
    }

    int f = 0;
    int g = m - 1;

    for (int i = m - 2; i >= 0; --i)
    {
        if (i > g && suffixes[i + m - 1 - f] < i - g)
        {
            suffixes[i] = suffixes[i + m - 1 - f];
        }
        else
        {
            if (i < g)
            {
                g = i;
            }
            f = i;
            while (g >= 0 && pattern[g] == pattern[g + m - 1 - f])
            {
                --g;
            }
            suffixes[i] = f - g;
        }
    }

    for (int i = 0; i < m - 1; ++i)
    {

```

```

        goodSuffixTable[m - 1 - suffixes[i]] = m - 1 - i;
    }

    for (int i = 0; i <= m - 2; ++i)
    {
        goodSuffixTable[m - 1 - suffixes[i]] = m - 1 - i;
    }

    return goodSuffixTable;
}

```

### BM

```

public List<int> BM(string text, string pattern)
{
    List<int> matches = new List<int>();
    int[] badCharTable = BuildBadCharacterTable(pattern);
    int[] goodSuffixTable = BuildGoodSuffixTable(pattern);
    int textLength = text.Length;
    int patternLength = pattern.Length;
    int s = 0;

    while (s <= (textLength - patternLength))
    {
        int j = patternLength - 1;

        while (j >= 0 && pattern[j] == text[s + j])
        {
            j--;
        }

        if (j < 0)
        {
            matches.Add(s);
            s += (s + patternLength < textLength) ? patternLength - badCharTable[text[s + patternLength]] : 1;
        }
        else
        {
            s += Math.Max(goodSuffixTable[j], j - badCharTable[text[s + j]]);
        }
    }

    return matches;
}

```

## 2. KnuthMorrisPratt.cs

Melakukan pencocokan string dengan algoritma Knuth-Morris-Pratt.

### computeBorder

```

private static int[] computeBorder(string pattern)
{
    int j = 0;
    int i = 1;

```

```

int[] fail = new int[pattern.Length];
fail[0] = 0;

while (i < pattern.Length)
{
    if (pattern[i] == pattern[j])
    {
        j++;
        fail[i] = j;
        i++;
    }
    else
    {
        if (j != 0)
        {
            j = fail[j - 1];
        }
        else
        {
            fail[i] = 0;
            i++;
        }
    }
}
return fail;
}

```

### KMP

```

public List<int> KMP(string text, string pattern)
{
    int M = pattern.Length;
    int N = text.Length;
    List<int> matches = new List<int>();

    int[] fail = computeBorder(pattern);

    int i = 0;
    int j = 0;
    while (i < N)
    {
        if (pattern[j] == text[i])
        {
            j++;
            i++;
        }

        if (j == M)
        {
            matches.Add(i - j);
            j = fail[j - 1];
        }
        else if (i < N && pattern[j] != text[i])
        {
            if (j != 0)
            {
                j = fail[j - 1];
            }
        }
    }
}

```

```

        else
        {
            i++;
        }
    }

    return matches;
}

```

### 3. DataController.cs

Digunakan untuk memasukkan data pada DB diawal program.

#### getFingerData

```

public List<FingerprintData> getFingerData(){
    List<FingerprintData> fingers = new List<FingerprintData>();

    string query = @"
        SELECT * FROM sidik_jari;
    ";

    using (var command = new SqlCommand(query, sql_conn))
    {
        using (var reader = command.ExecuteReader())
        {
            while (reader.Read())
            {
                string name = reader["name"].ToString();
                string path = reader["path"].ToString();
                fingers.Add(new FingerprintData(name, path));
            }
        }
    }

    return fingers;
}

```

#### insertFingerPrint

```

public void insertFingerPrint(FingerprintData finger){
    string query = $"INSERT INTO sidik_jari (name, path) VALUES (@name, @path);";

    using (var command = new SqlCommand(query, sql_conn))
    {
        command.Parameters.AddWithValue("@name", finger.getName());
        command.Parameters.AddWithValue("@path", finger.getPath());
        command.ExecuteNonQuery();
    }
}

```

### **insertFingerList**

```
public void insertFingerList(List<FingerprintData> finger_list){  
    foreach (var finger in finger_list)  
    {  
        insertFingerPrint(finger);  
    }  
}
```

### **insertKTP**

```
public void insertKTP(KTPData ktp){  
    string query = @"INSERT INTO biodata (NIK, name, birth_place, birth_date, gender,  
blood_type, address, religion, marriage_status, job, citizenhip)  
VALUES (@NIK, @name, @birth_place, @birth_date, @gender, @blood_type,  
@address, @religion, @marriage_status, @job, @citizenhip);";  
  
    using (var command = new SqlCommand(query, sql_conn))  
    {  
        command.Parameters.AddWithValue("@NIK", ktp.NIK);  
        command.Parameters.AddWithValue("@name", ktp.name);  
        command.Parameters.AddWithValue("@birth_place", ktp.birth_place);  
        command.Parameters.AddWithValue("@birth_date", ktp.birth_date);  
        command.Parameters.AddWithValue("@gender", ktp.gender);  
        command.Parameters.AddWithValue("@blood_type", ktp.blood_type);  
        command.Parameters.AddWithValue("@address", ktp.address);  
        command.Parameters.AddWithValue("@religion", ktp.religion);  
        command.Parameters.AddWithValue("@marriage_status", ktp.marriage_status);  
        command.Parameters.AddWithValue("@job", ktp.job);  
        command.Parameters.AddWithValue("@citizenhip", ktp.citizenhip);  
        command.ExecuteNonQuery();  
    }  
}
```

### **insertKTPList**

```
public void insertKTPList(List<KTPData> ktp_list){  
    foreach (var ktp in ktp_list)  
    {  
        insertKTP(ktp);  
    }  
}
```

### **TraverseSidikJadi**

```
public List<FingerprintData> TraverseSidikJari()  
{  
    List<FingerprintData> fingerDataList = new List<FingerprintData>();  
    string query = "SELECT berkas_citra, nama FROM sidik_jari;";  
    using (var command = new SqlCommand(query, sql_conn))  
    {  
        using (var reader = command.ExecuteReader())
```

```

    {
        while (reader.Read())
        {
            string berkasCitra = reader["berkas_citra"].ToString();
            string nama = reader["nama"].ToString();
            fingerDataList.Add(new FingerprintData(nama, berkasCitra));
        }
    }
    return fingerDataList;
}

```

#### TraverseBiodata

```

public void TraverseBiodata()
{
    string query = @"SELECT NIK, nama, tempat_lahir, tanggal_lahir, jenis_kelamin,
golongan_darah, alamat, agama, status_perkawinan, pekerjaan, kewarganegaraan FROM biodata;";
    using (var command = new SqliteCommand(query, sql_conn))
    {
        using (var reader = command.ExecuteReader())
        {
            while (reader.Read())
            {
                string nik = reader["NIK"].ToString();
                string biodataNama = reader["nama"].ToString();
                string tempatLahir = reader["tempat_lahir"].ToString();
                Console.WriteLine($"NIK: {nik}, Nama: {biodataNama}, Tempat Lahir:
{tempatLahir}");
            }
        }
    }
}

```

#### 4. FingerprintController.cs

Merupakan *file controller* yang dipanggil pada frontend. Melakukan *fetch* dan *pass* dari frontend dan ke frontend.

#### FingerprintController()

```

public FingerprintController()
{
    // Controller data = new Controller("MainData.db");
    Controller data = new Controller("EncryptedData.db");
    var fingerDataList = data.TraverseSidikJari();

    foreach (var fingerData in fingerDataList)
    {
        string filePath = fingerData.getPath();

        using (Image<Rgba32> image = Image.Load<Rgba32>(filePath))
        {
            int[,] binaryArray = ImageConverter.ConvertToBinary(image);
        }
    }
}

```

```

        string asciiString =
ImageConverter.ConvertBinaryArrayToAsciiString(binaryArray);
referenceImagesMap[filePath] = asciiString;

        using (Image<Rgba32> croppedImage = ImageConverter.CropImageTo1x64(image))
{
    int[,] croppedBinaryArray =
ImageConverter.ConvertToBinary(croppedImage);
    string croppedAsciiString =
ImageConverter.ConvertBinaryArrayToAsciiString(croppedBinaryArray);
    croppedReferenceImagesMap[filePath] = croppedAsciiString;
}
}
}
}
}

```

### ProcessImage

```

public async Task<IActionResult> ProcessImage([FromForm] IFormFile image, [FromForm] string
algorithm)
{
    if (image == null || image.Length == 0)
        return BadRequest("No image uploaded.");

    string tempFilePath = Path.GetTempFileName();
    var stopwatch = new System.Diagnostics.Stopwatch();

    try
    {
        using (var stream = System.IO.File.Create(tempFilePath))
        {
            await image.CopyToAsync(stream);
        }

        string pattern;
        stopwatch.Start();

        using (Image<Rgba32> uploadedImage = Image.Load<Rgba32>(tempFilePath))
        {
            int[,] patternBinaryArray = ImageConverter.ConvertToBinary(uploadedImage);

            using (Image<Rgba32> croppedPatternImage =
ImageConverter.CropImageTo1x64(uploadedImage))
            {
                int[,] croppedPatternBinaryArray =
ImageConverter.ConvertToBinary(croppedPatternImage);
                pattern =
ImageConverter.ConvertBinaryArrayToAsciiString(croppedPatternBinaryArray);
            }
        }

        FingerprintMatcher matcher = new FingerprintMatcher(algorithm);
        var result = matcher.FindMostSimilarFingerprint(pattern, referenceImagesMap,
croppedReferenceImagesMap);
        string similarImage = result.mostSimilarImage;
        double percentage = result.maxSimilarity;
        bool exactMatchFound = result.exactMatchFound;
    }
}

```

```

        stopwatch.Stop();

        var executionTime = stopwatch.ElapsedMilliseconds;

        return Ok(new { similarImage = Path.GetFileName(similarImage), percentage,
executionTime, exactMatchFound });

    }
    catch (Exception ex)
    {
        return StatusCode(500, $"Internal server error: {ex.Message}");
    }
    finally
    {
        if (System.IO.File.Exists(tempFilePath))
        {
            System.IO.File.Delete(tempFilePath);
        }
    }
}

```

#### GetImage

```

public IActionResult GetImage(string filename)
{
    string folderPath = Path.Combine(Directory.GetCurrentDirectory(), "../test");
    string filePath = Path.Combine(folderPath, filename);

    if (!System.IO.File.Exists(filePath))
    {
        return NotFound();
    }

    byte[] imageBytes = System.IO.File.ReadAllBytes(filePath);
    return File(imageBytes, "image/bmp");
}

```

#### GetBiodata

```

public IActionResult GetBiodata(string filename)
{
    try
    {
        Console.WriteLine("masuk");
        string imagePath = ("../test/" + filename);
        Console.WriteLine("Image path: " + imagePath);

        // Controller dataController = new Controller("MainData.db");
        Controller dataController = new Controller("EncryptedData.db");
        string query = "SELECT nama FROM sidik_jari WHERE berkas_citra = @imagePath;";
        string ownerName = "";

        using (var command = new SqlCommand(query, dataController.sql_conn))
        {
            command.Parameters.AddWithValue("@imagePath", imagePath);
        }
    }
}

```

```

        using (var reader = command.ExecuteReader())
    {
        if (reader.Read())
        {
            if (!reader.IsDBNull(reader.GetOrdinal("nama")))
            {
                ownerName = reader["nama"].ToString();
                Console.WriteLine("Owner name: " + ownerName);
            }
            else
            {
                Console.WriteLine("Nama column is null for the current row.");
            }
        }
        else
        {
            Console.WriteLine("No owner found for image path: " + imagePath);
        }
    }

    if (string.IsNullOrEmpty(ownerName))
    {
        return NotFound("Owner not found.");
    }

    // regex pattern utk nama owner
    string namePattern = Regex.CreateWeirdNameRegex(ownerName);
    Console.WriteLine("Regex pattern: " + namePattern);

    query = "SELECT * FROM biodata;";
    List<KTPData> biodataList = new List<KTPData>();

    using (var command = new SqlCommand(query, dataController.sql_conn))
    {
        using (var reader = command.ExecuteReader())
        {
            while (reader.Read())
            {
                string biodataName =
EncryptionHelper.DecryptString(reader["nama"].ToString());
                biodataName = biodataName.Trim('\'');
                // Console.WriteLine("Trimmed biodata name: " + biodataName);

                if (System.Text.RegularExpressions.Regex.IsMatch(biodataName,
namePattern, System.Text.RegularExpressions.RegexOptions.IgnoreCase))
                {
                    Console.WriteLine("Match found: " + biodataName);
                    var ktpData = new KTPData(
trimDoubleQuote(EncryptionHelper.DecryptString(reader["NIK"].ToString())),
// reader["nama"].ToString(),
ownerName,

trimDoubleQuote(EncryptionHelper.DecryptString(reader["tempat_lahir"].ToString())),
trimDoubleQuote(EncryptionHelper.DecryptString(reader["tanggal_lahir"].ToString())),
trimDoubleQuote(EncryptionHelper.DecryptString(reader["jenis_kelamin"].ToString())));

```

```

trimDoubleQuote(EncryptionHelper.DecryptString(reader["golongan_darah"].ToString())),
trimDoubleQuote(EncryptionHelper.DecryptString(reader["alamat"].ToString())),
trimDoubleQuote(EncryptionHelper.DecryptString(reader["agama"].ToString())),
trimDoubleQuote(EncryptionHelper.DecryptString(reader["status_perkawinan"].ToString())),
trimDoubleQuote(EncryptionHelper.DecryptString(reader["pekerjaan"].ToString())),
trimDoubleQuote(EncryptionHelper.DecryptString(reader["kewarganegaraan"].ToString()))
);
biodataList.Add(ktpData);
}
}
}
}

if (biodataList.Count == 0)
{
    Console.WriteLine("No matching biodata found for owner name: " +
ownerName);
    return NotFound("No matching biodata found.");
}

// return yang pertama
return Ok(biodataList.First());
}
catch (Exception ex)
{
    Console.WriteLine("Exception occurred: " + ex.Message);
    return StatusCode(500, $"Internal server error: {ex.Message}");
}
}

```

#### trimDoubleQuote

```

public string trimDoubleQuote(string str)
{
    str = str.Trim('\'');
    return str;
}

```

### 5. FingerprintMatcher.cs

Melakukan proses *searching* baik dengan Hamming Distance ataupun dengan BM atau KMP.

#### HammingDistance

```

private int HammingDistance(string s1, string s2)
{
    s1 = s1.Normalize(NormalizationForm.FormC);
    s2 = s2.Normalize(NormalizationForm.FormC);
}

```

```

int distance = 0;
int len = Math.Min(s1.Length, s2.Length);

for (int i = 0; i < len; i++)
{
    if (s1[i] != s2[i])
    {
        distance++;
    }
}

return distance;
}

```

### Search

```

public (List<int> matches, Dictionary<string, double> similarityPercentages) Search(
    string pattern,
    Dictionary<string, string> referenceImagesMap,
    Dictionary<string, string> croppedReferenceImagesMap)
{
    List<int> matches = new List<int>();
    Dictionary<string, double> similarityPercentages = new Dictionary<string, double>();

    // Iterate through the reference images map
    foreach (var kvp in referenceImagesMap)
    {
        string imagePath = kvp.Key;
        string referenceText = kvp.Value;

        // Perform exact matching using the chosen algorithm
        List<int> algorithmMatches = algorithm == "KMP" ? kmp.KMP(referenceText, pattern) :
bm.BM(referenceText, pattern);
        if (algorithmMatches.Count > 0)
        {
            matches.AddRange(algorithmMatches);
            similarityPercentages[imagePath] = 1.0;
            continue;
        }
    }

    // If no exact matches are found, use Hamming Distance on the cropped images map
    if (matches.Count == 0)
    {
        Console.WriteLine("No exact matches found. Using Hamming Distance on cropped
images.");
    }

    // Iterate through the cropped reference images map
    foreach (var kvp in croppedReferenceImagesMap)
    {
        string imagePath = kvp.Key;
        string croppedReferenceText = kvp.Value;

        // Perform Hamming Distance calculation
        int distance = HammingDistance(pattern, croppedReferenceText);
        Console.WriteLine($"distance: {distance}");
        double similarity = 1.0 - (double)distance / pattern.Length;
        Console.WriteLine($"similarity: {similarity}");
    }
}

```

```

        similarityPercentages[imagePath] = similarity;
    }
}

```

### FindMostSimilarFingerprint

```

public (string mostSimilarImage, double maxSimilarity, bool exactMatchFound)
FindMostSimilarFingerprint(
    string pattern,
    Dictionary<string, string> referenceImagesMap,
    Dictionary<string, string> croppedReferenceImagesMap)
{
    var result = Search(pattern, referenceImagesMap, croppedReferenceImagesMap);
    List<int> matches = result.matches;
    Dictionary<string, double> similarityPercentages = result.similarityPercentages;

    if (matches.Count > 0)
    {
        Console.WriteLine("Exact matches found:");

        foreach (int match in matches)
        {
            Console.WriteLine($"Pattern found at index: {match}");
        }
        string exactMatchImage = null;
        foreach (var kvp in similarityPercentages)
        {
            if (kvp.Value == 1.0)
            {
                exactMatchImage = kvp.Key;
                break;
            }
        }

        double maxSimilarity = 100;
        return (exactMatchImage ?? string.Empty, maxSimilarity, true);
    }
    else
    {
        double maxSimilarity = 0;
        string mostSimilarImage = null;

        foreach (var kvp in similarityPercentages)
        {
            if (kvp.Value >= maxSimilarity)
            {
                maxSimilarity = kvp.Value;
                mostSimilarImage = kvp.Key;
            }
        }

        Console.WriteLine($"maxSimilarity: {maxSimilarity}");

        maxSimilarity *= 100;
        if (maxSimilarity >= 50)
        {
            Console.WriteLine($"Most similar fingerprint found in image:
{Path.GetFileName(mostSimilarImage)} with similarity {maxSimilarity}%");
        }
    }
}

```

```

        return (mostSimilarImage ?? string.Empty, maxSimilarity, false);
    }
    else
    {
        Console.WriteLine("No similar fingerprint found.");
        return (null, 0, false);
    }
}
}

```

## 6. ImageConverter.cs

Mengubah gambar yang diunggah ataupun gambar *reference* menjadi *binary* dan kemudian menjadi ASCII.

### ConvertToBinary

```

public static int[,] ConvertToBinary(Image<Rgba32> original, byte threshold = 128)
{
    int width = original.Width;
    int height = original.Height;
    int[,] binaryArray = new int[width, height];

    for (int y = 0; y < height; y++)
    {
        for (int x = 0; x < width; x++)
        {
            // ambil pixel color
            Rgba32 pixelColor = original[x, y];
            // convert pixelnya ke grayscale
            byte grayValue = (byte)(0.3 * pixelColor.R + 0.59 * pixelColor.G + 0.11 *
pixelColor.B);
            // convert ke binary berdasarkan threshold
            binaryArray[x, y] = grayValue < threshold ? 0 : 1;
        }
    }

    return binaryArray;
}

```

### ConvertBinaryArrayToAsciiString

```

public static string ConvertBinaryArrayToAsciiString(int[,] binaryArray)
{
    int rows = binaryArray.GetLength(0);
    int columns = binaryArray.GetLength(1);

    StringBuilder asciiString = new StringBuilder();

    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < columns; j++)
        {
            asciiString.Append(binaryArray[i, j]);
        }
    }
}

```

```

    }

    string binary = asciiString.ToString();
    // Ensure the binary string has a length that is a multiple of 8
    int remainder = binary.Length % 8;
    if (remainder != 0)
    {
        binary = binary.PadRight(binary.Length + (8 - remainder), '0');
    }

    StringBuilder finalAsciiString = new StringBuilder();
    for (int k = 0; k < binary.Length; k += 8)
    {
        string byteString = binary.Substring(k, 8);
        int asciiCode = Convert.ToInt32(byteString, 2);
        char character = (char)asciiCode;
        finalAsciiString.Append(character);
    }

    return finalAsciiString.ToString();
}

```

#### CropImageTo1x64

```

public static Image<Rgba32> CropImageTo1x64(Image<Rgba32> image)
{
    // Calculate the middle pixel position, ensuring it starts at a multiple of 8
    int startX = (image.Width / 2) / 8 * 8;
    int startY = (image.Height / 2) - 32;

    // Ensure startY is a multiple of 8
    startY = (startY / 8) * 8;

    // Crop the image, ensuring dimensions are within bounds
    if (startY + 64 > image.Height)
    {
        startY = image.Height - 64;
    }

    return image.Clone(ctx => ctx.Crop(new Rectangle(startX, startY, 1, 64)));
}

```

#### PrintBinaryArray

```

public static void PrintBinaryArray(int[,] binaryArray)
{
    int rows = binaryArray.GetLength(0);
    int columns = binaryArray.GetLength(1);

    for (int i = 0; i < rows; i++)
    {
        for (int j = 0; j < columns; j++)
        {
            Console.Write(binaryArray[i, j]);
        }
        Console.WriteLine();
    }
}

```

```
    }  
}
```

## 7. Encrypt\_csv.cs

Melakukan enkripsi terhadap *file* CSV.

### Main

```
static void Main()  
{  
    string key = "printuyulGgXXXXXXXXX6969";  
    string inputFile = "biodata.csv";  
    string outputFile = "encrypted_biodata.csv";  
  
    List<string[]> encryptedRows = new List<string[]>();  
  
    using (StreamReader reader = new StreamReader(inputFile))  
    {  
        string line;  
        while ((line = reader.ReadLine()) != null)  
        {  
            string[] row = line.Split(',');  
            string[] encryptedRow = Array.ConvertAll(row, cell => Encrypt(cell, key));  
            encryptedRows.Add(encryptedRow);  
        }  
    }  
  
    using (StreamWriter writer = new StreamWriter(outputFile))  
    {  
        foreach (string[] row in encryptedRows)  
        {  
            writer.WriteLine(string.Join(", ", row));  
        }  
    }  
  
    Console.WriteLine("Encryption complete.'");  
}
```

### Encrypt

```
static string Encrypt(string plainText, string key)  
{  
    byte[] iv = new byte[16];  
    using (Aes aesAlg = Aes.Create())  
    {  
        aesAlg.Key = Encoding.UTF8.GetBytes(key);  
        aesAlg.IV = iv;  
  
        ICryptoTransform encryptor = aesAlg.CreateEncryptor(aesAlg.Key, aesAlg.IV);  
  
        using (MemoryStream msEncrypt = new MemoryStream())  
        {  
            using (CryptoStream csEncrypt = new CryptoStream(msEncrypt, encryptor,  
CryptoStreamMode.Write))  
            {
```

```

        using (StreamWriter swEncrypt = new StreamWriter(csEncrypt))
    {
        swEncrypt.Write(plainText);
    }
}
byte[] encryptedBytes = msEncrypt.ToArray();
byte[] combined = iv.Concat(encryptedBytes).ToArray();
return Convert.ToBase64String(combined);
}
}
}

```

## 8. KTPData.cs

Konstruktor untuk data-data yang terdapat pada KTP.

<b>KTPData</b>
<pre> public KTPData(string NIK, string name, string birth_place, string birth_date, string gender, string blood_type, string address, string religion, string marriage_status, string job, string citizenship) {     this.NIK = NIK;     this.name = name;     this.birth_place = birth_place;     this.birth_date = birth_date;     this.gender = gender;     this.blood_type = blood_type;     this.address = address;     this.religion = religion;     this.marriage_status = marriage_status;     this.job = job;     this.citizenship = citizenship; } </pre>

## 9. FingerData.cs

Mengandung *setter* dan *getter* untuk data-data pada sidik jari.

<b>getName</b>
<pre> public string getName(){     return this.name; } </pre>

<b>getPath</b>
<pre> public string getPath(){     return this.path; } </pre>

<b>setName</b>
----------------

```
public void setName(string name) {
    this.name = name;
}
```

#### setPath

```
public void setPath(string path) {
    this.path = path;
}
```

### 10. InsertBiodataFromCSV.cs

Memasukkan data biodata dari CSV ke DB.

#### Main

```
static void Main(string[] args)
{
    string csvFilePath = "encrypted_biodata.csv";
    string dbPath = "EncryptedData.db";

    List<string[]> biodataEntries = ReadCsv(csvFilePath);
    InsertIntoDatabase(biodataEntries, dbPath);

    Console.WriteLine("Data inserted successfully.");
}
```

#### ReadCsv

```
static List<string[]> ReadCsv(string filePath)
{
    List<string[]> rows = new List<string[]>();
    using (var reader = new StreamReader(filePath))
    {
        reader.ReadLine(); // Skip the header
        while (!reader.EndOfStream)
        {
            string[] row = reader.ReadLine().Split(',');
            rows.Add(row);
        }
    }
    return rows;
}
```

#### InsertIntoDatabase

```
static void InsertIntoDatabase(List<string[]> biodataEntries, string dbPath)
{
    using (var conn = new SQLiteConnection($"Data Source={dbPath};Version=3;"))
    {
        conn.Open();
```

```

using (var cmd = conn.CreateCommand())
{
    cmd.CommandText = "DROP TABLE IF EXISTS biodata";
    cmd.ExecuteNonQuery();

    // Create the biodata table if it does not exist
    cmd.CommandText = @"CREATE TABLE IF NOT EXISTS biodata (
        NIK TEXT PRIMARY KEY,
        nama TEXT NOT NULL,
        tempat_lahir TEXT NOT NULL,
        tanggal_lahir TEXT NOT NULL,
        jenis_kelamin TEXT NOT NULL,
        golongan_darah TEXT NOT NULL,
        alamat TEXT NOT NULL,
        agama TEXT NOT NULL,
        status_perkawinan TEXT NOT NULL,
        pekerjaan TEXT NOT NULL,
        kewarganegaraan TEXT NOT NULL)";
    cmd.ExecuteNonQuery();

    // Insert data into the biodata table
    cmd.CommandText = @"INSERT INTO biodata (NIK, nama, tempat_lahir,
    tanggal_lahir, jenis_kelamin, golongan_darah, alamat, agama, status_perkawinan, pekerjaan,
    kewarganegaraan)
    VALUES (@NIK, @nama, @tempat_lahir, @tanggal_lahir, @jenis_kelamin,
    @golongan_darah, @alamat, @agama, @status_perkawinan, @pekerjaan, @kewarganegaraan)";

    foreach (var entry in biodataEntries)
    {
        // Call this function before the executemany statement
        CheckForExtraValues(entry);
        cmd.Parameters.Clear();
        cmd.Parameters.AddWithValue("@NIK", entry[0]);
        cmd.Parameters.AddWithValue("@nama", entry[1]);
        cmd.Parameters.AddWithValue("@tempat_lahir", entry[2]);
        cmd.Parameters.AddWithValue("@tanggal_lahir", entry[3]);
        cmd.Parameters.AddWithValue("@jenis_kelamin", entry[4]);
        cmd.Parameters.AddWithValue("@golongan_darah", entry[5]);
        cmd.Parameters.AddWithValue("@alamat", entry[6]);
        cmd.Parameters.AddWithValue("@agama", entry[7]);
        cmd.Parameters.AddWithValue("@status_perkawinan", entry[8]);
        cmd.Parameters.AddWithValue("@pekerjaan", entry[9]);
        cmd.Parameters.AddWithValue("@kewarganegaraan", entry[10]);
        cmd.ExecuteNonQuery();
    }
}
}

```

## 11. InsertSidikJariFromCSV.cs

Memasukkan data sidik jari dari CSV ke DB.

Main
<pre> static void Main(string[] args) {     string csvFilePath = "encrypted_sidik_jari.csv"; </pre>

```

        string dbPath = "EncryptedData.db";

        List<string[]> sidikJariEntries = ReadCsv(csvFilePath);
        InsertIntoDatabase(sidikJariEntries, dbPath);

        Console.WriteLine("Data inserted successfully.");
    }
}

```

### ReadCsv

```

static List<string[]> ReadCsv(string filePath)
{
    List<string[]> rows = new List<string[]>();
    using (var reader = new StreamReader(filePath))
    {
        reader.ReadLine(); // Skip the header
        while (!reader.EndOfStream)
        {
            string[] row = reader.ReadLine().Split(',');
            rows.Add(row);
        }
    }
    return rows;
}

```

### InsertIntoDatabase

```

static void InsertIntoDatabase(List<string[]> sidikJariEntries, string dbPath)
{
    using (var conn = new SQLiteConnection($"Data Source={dbPath};Version=3;"))
    {
        conn.Open();
        using (var cmd = conn.CreateCommand())
        {
            cmd.CommandText = "DROP TABLE IF EXISTS sidik_jari";
            cmd.ExecuteNonQuery();

            // Create the sidik_jari table if it does not exist
            cmd.CommandText = @"CREATE TABLE IF NOT EXISTS sidik_jari (
                berkas_citra TEXT NOT NULL,
                nama TEXT NOT NULL)";
            cmd.ExecuteNonQuery();

            // Insert data into the sidik_jari table
            cmd.CommandText = @"INSERT INTO sidik_jari (berkas_citra, nama)
                VALUES (@berkas_citra, @nama)";

            foreach (var entry in sidikJariEntries)
            {
                // Call this function before the executemany statement
                CheckForExtraValues(entry);
                cmd.Parameters.Clear();
                cmd.Parameters.AddWithValue("@berkas_citra", entry[0]);
                cmd.Parameters.AddWithValue("@nama", entry[1]);
                cmd.ExecuteNonQuery();
            }
        }
    }
}

```

```
        }
    }
}
```

## 12. Regex.cs

Membuat ekspresi atau *pattern regular expression* dari suatu string.

### CreateWeirdNameRegex

```
public static string CreateWeirdNameRegex(string realName)
{
    // create regex pattern
    string pattern = "";
    foreach (char c in realName)
    {
        pattern += GetCharPattern(c);
    }

    // tutup dengan ^ dan $ supaya dia harus match seluruh string (bukan partial)
    return $"^{pattern}$";
}
```

### GetCharPattern

```
private static string GetCharPattern(char c)
{
    return c switch
    {
        'a' => "[aA4]?", 
        'e' => "[eE3]?", 
        'i' => "[iI1]?", 
        'o' => "[oO0]?", 
        'u' => "[uU]?", 
        'b' => "[bB8]", 
        'c' => "[cC]", 
        'd' => "[dD]", 
        'f' => "[fF]", 
        'g' => "[gG6]", 
        'h' => "[hH]", 
        'j' => "[jJ]", 
        'k' => "[kK]", 
        'l' => "[lL]", 
        'm' => "[mM]", 
        'n' => "[nN]", 
        'p' => "[pP]", 
        'q' => "[qQ]", 
        'r' => "[rR]", 
        's' => "[sS5]", 
        't' => "[tT7]", 
        'v' => "[vV]", 
        'w' => "[wW]", 
        'x' => "[xX]", 
        'y' => "[yY]", 
        'z' => "[zZ]", 
    };
}
```

```

        'A' => "[aA4]?",  

        'E' => "[eE3]?",  

        'I' => "[iI1]?",  

        'O' => "[oO0]?",  

        'U' => "[uU]?",  

        'B' => "[bB8]",  

        'C' => "[cC]",  

        'D' => "[dD]",  

        'F' => "[fF]",  

        'G' => "[gG6]",  

        'H' => "[hH]",  

        'J' => "[jJ]",  

        'K' => "[kK]",  

        'L' => "[lL]",  

        'M' => "[mM]",  

        'N' => "[nN]",  

        'P' => "[pP]",  

        'Q' => "[qQ]",  

        'R' => "[rR]",  

        'S' => "[sS5]",  

        'T' => "[tT7]",  

        'V' => "[vV]",  

        'W' => "[wW]",  

        'X' => "[xX]",  

        'Y' => "[yY]",  

        'Z' => "[zZ]",  

        _ => c.ToString()
    };
}

```

### 4.3 Frontend

Fungsi dan *styling* yang dibangun guna menunjang keberjalan algoritma dan program terbagi dalam beberapa *file*, di antaranya adalah:

#### 1. KTPData.cs

KTPData.cs hanya berisi kelas KTPData yang sama dengan backend. *File* ini juga diletakkan di frontend karena akan dipanggil pada frontend.

#### 2. MainWindow.xaml

Halaman pertama atau *home page* dari aplikasi.

```

<Window xmlns="https://github.com/avaloniaui"  

    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  

    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  

    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  

    mc:Ignorable="d"  

    d:DesignWidth="990"  

    d:DesignHeight="646"  

    x:Class="newjeans_avalonia.MainWindow"  

    Title="Printuyul"  

    CanResize="False"  

    Width="990"  

    Height="646"  

    WindowStartupLocation="CenterOwner">

```

```

<Window.Background>
    <ImageBrush Source="avares://newjeans_avalonia/assets/home.png"/>
</Window.Background>

<Window.Styles>
    <!-- Custom Button Style -->
    <Style Selector="Button">
        <Setter Property="Background" Value="Transparent"/>
        <Setter Property="BorderBrush" Value="Transparent"/>
        <Setter Property="BorderThickness" Value="0"/>
        <Setter Property="Template">
            <ControlTemplate>
                <Border Background="{TemplateBinding Background}"
                    BorderBrush="{TemplateBinding BorderBrush}"
                    BorderThickness="{TemplateBinding BorderThickness}">
                    <ContentPresenter Content="{TemplateBinding Content}"
                        HorizontalAlignment="{TemplateBinding
HorizontalContentAlignment}"
                        VerticalAlignment="{TemplateBinding
VerticalContentAlignment}"/>
                </Border>
            </ControlTemplate>
        </Setter>
    </Style>
</Window.Styles>

<!-- Layout to hold the image button -->
<Panel>
    <!-- Button with an Image inside it -->
    <Button x:Name="NavigateButton" Width="550" Height="150" Margin="-70,-260,0,0"
        Background="Transparent" BorderBrush="Transparent" BorderThickness="0">
        <Image Source="avares://newjeans_avalonia/assets/start.png"/>
    </Button>
</Panel>
</Window>

```

### 3. MainWindow.xaml.cs

*Styling* untuk halaman pertama dari aplikasi.

```

// MainWindow.xaml.cs
using Avalonia.Controls;
using Avalonia.Interactivity;

namespace newjeans_avalonia
{
    public partial class MainWindow : Window
    {
        private AppState _appState;

        public MainWindow()
        {
            InitializeComponent();
            _appState = new AppState();
            this.FindControl<Button>("NavigateButton")!.Click += OnNavigateButtonClick;
        }

        private void OnNavigateButtonClick(object? sender, RoutedEventArgs e)

```

```

    {
        SecondWindow secondWindow = new SecondWindow(_appState);
        secondWindow.Show();
        this.Close();
    }
}

```

#### 4. SecondWindow.xaml

Halaman kedua dari aplikasi, mengatur kemampuan mengunggah gambar, memilih algoritma, *searching*, dan melihat hasil biodata.

```

<Window xmlns="https://github.com/avaloniaui"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    x:Class="newjeans_avalonia.SecondWindow"
    Title="Printuyul"
    Width="990" Height="646" CanResize="False"
    WindowStartupLocation="CenterOwner">
    <Window.Background>
        <ImageBrush Source="avares://newjeans_avalonia/assets/home2.png"/>
    </Window.Background>

    <Window.Styles>
        <!-- Custom Button Style -->
        <Style Selector="Button">
            <Setter Property="Background" Value="Transparent"/>
            <Setter Property="BorderBrush" Value="Transparent"/>
            <Setter Property="BorderThickness" Value="0"/>
            <Setter Property="Template">
                <ControlTemplate>
                    <Border Background="{TemplateBinding Background}"
                        BorderBrush="{TemplateBinding BorderBrush}"
                        BorderThickness="{TemplateBinding BorderThickness}">
                        <ContentPresenter Content="{TemplateBinding Content}"
                            HorizontalAlignment="{TemplateBinding
                            HorizontalContentAlignment}"
                            VerticalAlignment="{TemplateBinding
                            VerticalContentAlignment}"/>
                    </Border>
                </ControlTemplate>
            </Setter>
        </Style>
    </Window.Styles>

    <Panel>
        <Button x:Name="InsertButton" Width="430" Height="110" Margin="160,-420,0,0"
            Background="Transparent" BorderBrush="Transparent" BorderThickness="0">
            <Image Source="avares://newjeans_avalonia/assets/insert.png"/>
        </Button>

        <Image x:Name="DisplayImage" Height="130" Width="300" HorizontalAlignment="Left"
            Margin="120,-70,0,0"/>
    </Panel>

```

```

<ComboBox x:Name="MethodDropdown" Width="260" Height="40" Margin="25,570,0,0"
    VerticalAlignment="Top" BorderBrush="#3533BD" BorderThickness="3">
    <ComboBoxItem Content="Boyer Moore"/>
    <ComboBoxItem Content="Knuth Morris Pratt"/>
</ComboBox>

<Button x:Name="SearchButton" Width="300" Height="90" Margin="240,520,0,0"
    Background="Transparent" BorderBrush="Transparent" BorderThickness="0">
    <Image Source="avares://newjeans_avalonia/assets/search.png"/>
</Button>

<Button x:Name="NavigateButton2" Width="113" Height="31" Margin="683,581,0,0"
    Background="Transparent" BorderBrush="Transparent" BorderThickness="0">
    <Image Source="avares://newjeans_avalonia/assets/details.png"/>
</Button>

<Image x:Name="ResultsImage" Height="130" Width="300" HorizontalAlignment="Left"
Margin="690,-200,0,0" />

<TextBlock x:Name="SimilarityTextBlock" Margin="1030,457,0,0" Width="300" Height="40"
VerticalAlignment="Top" Foreground="#3533BD" FontSize="16"
FontFamily="{StaticResource FairfaxFont}"/>
    <TextBlock x:Name="ExecutionTimeTextBlock" Margin="1030,487,0,0" Width="300"
Height="40"
        VerticalAlignment="Top" Foreground="#3533BD" FontSize="16"
FontFamily="{StaticResource FairfaxFont}"/>

<Image x:Name="LoadingImage" Source="avares://newjeans_avalonia/assets/loading.png"
Width="990" Height="646" HorizontalAlignment="Center" VerticalAlignment="Center"
IsVisible="false"/>
</Panel>

</Window>

```

## 5. SecondWindow.xaml.cs

*Styling* untuk halaman kedua.

```

using Avalonia.Controls;
using Avalonia.Media.Imaging;
using Avalonia.Threading;
using System.Linq;
using System.Collections.Generic;
using System.Net.Http;
using System.Net.Http.Headers;
using System.Threading.Tasks;
using System.IO;
using System;

namespace newjeans_avalonia
{
    public partial class SecondWindow : Window
    {
        private AppState _appState;
        private static readonly HttpClient client = new HttpClient();

        public SecondWindow(AppState appState)
        {

```

```

InitializeComponent();
AppState = AppState;

// Reset UI elements
DisplayImage.Source = null;
MethodDropdown.SelectedItem = null;
SimilarityTextBlock.Text = string.Empty;
ExecutionTimeTextBlock.Text = string.Empty;
ResultsImage.Source = null;

InsertButton.Click += InsertButton_Click;
SearchButton.Click += OnSearchButtonClick;
this.FindControl<Button>("NavigateButton2")!.Click += OnNavigateButtonClick2;

if (_AppState.CurrentImage != null)
{
    DisplayImage.Source = _AppState.CurrentImage;
}

if (!string.IsNullOrEmpty(_AppState.SelectedAlgorithm))
{
    MethodDropdown.SelectedItem = MethodDropdown.Items.OfType<ComboBoxItem>()
        .FirstOrDefault(item => item.Content!.ToString() ==
_AppState.SelectedAlgorithm);
}

SimilarityTextBlock.Text = _AppState.Similarity;
ExecutionTimeTextBlock.Text = _AppState.ExecutionTime;

if (_AppState.ResultImage != null)
{
    ResultsImage.Source = _AppState.ResultImage;
}
}

private async void InsertButton_Click(object? sender,
Avalonia.Interactivity.RoutedEventArgs e)
{
    var dialog = new OpenFileDialog
    {
        Title = "Select an Image",
        Filters = new List<FileDialogFilter>
        {
            new FileDialogFilter { Name = "Image Files", Extensions = new List<string>
{ "bmp" } }
        },
        AllowMultiple = false
    };

    var result = await dialog.ShowAsync(this);
    if (result != null && result.Any())
    {
        var bitmap = new Bitmap(result.First());
        DisplayImage.Source = bitmap;
        _AppState.CurrentImage = bitmap; // Save the image in the state
    }
}

private async void OnNavigateButtonClick2(object? sender,
Avalonia.Interactivity.RoutedEventArgs e)

```

```

    {
        if (_appState.ResultImage == null)
        {
            await ShowMessageAsync("Please perform a search before seeing details.");
            return;
        }

        ThirdWindow thirdWindow = new ThirdWindow(_appState);
        thirdWindow.Show();
        this.Close(); // Optionally close the current window
    }

    private async void OnSearchButtonClick(object? sender,
Avalonia.Interactivity.RoutedEventArgs e)
{
    var selectedAlgorithm = (MethodDropdown.SelectedItem as
ComboBoxItem)?.Content.ToString();
    _appState.SelectedAlgorithm = selectedAlgorithm;

    if (selectedAlgorithm == "Boyer Moore")
    {
        selectedAlgorithm = "BM";
    }
    else if (selectedAlgorithm == "Knuth Morris Pratt")
    {
        selectedAlgorithm = "KMP";
    }

    if (DisplayImage.Source is Bitmap currentBitmap &&
!string.IsNullOrEmpty(selectedAlgorithm))
    {
        // stopwatch execution time
        var stopwatch = new System.Diagnostics.Stopwatch();
        stopwatch.Start();

        LoadingImage.IsVisible = true;
        await ProcessImageAsync(currentBitmap, selectedAlgorithm);
        LoadingImage.IsVisible = false;

        // stop stopwatch
        stopwatch.Stop();
        // update _appState kemudian tampilkan di ExecutionTimeTextBlock
        _appState.ExecutionTime = $"{stopwatch.ElapsedMilliseconds} ms";
        ExecutionTimeTextBlock.Text = _appState.ExecutionTime;
    }
    else
    {
        await ShowMessageAsync("Please select an image and an algorithm.");
    }
}

private async Task ProcessImageAsync(Bitmap image, string algorithm)
{
    try
    {
        using (var client = new HttpClient())
        {
            client.BaseAddress = new Uri("http://localhost:5141/");
            var content = new MultipartFormDataContent();

```

```

        var memoryStream = new MemoryStream();
        image.Save(memoryStream);
        memoryStream.Position = 0;

        var imageContent = new StreamContent(memoryStream);
        imageContent.Headers.ContentType = new MediaTypeHeaderValue("image/png");
        content.Add(imageContent, "image", "uploadedImage.png");
        content.Add(new StringContent(algorithm), "algorithm");

        var response = await client.PostAsync("api/fingerprint/process", content);
        if (response.IsSuccessStatusCode)
        {
            var result = await response.Content.ReadAsAsync<dynamic>();

            // jika result.similarImage null, maka ShowMessageAsync("No matching
            biodata found.")
            if (result.similarImage == null)
            {
                await ShowMessageAsync("No matching biodata found.");
                return;
            }

            string similarImage = result.similarImage ?? "N/A";
            double percentage = result.percentage ?? 0;
            long executionTime = result.executionTime ?? 0; // Execution time in
            milliseconds
            bool exactMatchFound = result.exactMatchFound ?? false; // Handle null
            case

            string imageUrl =
            $"http://localhost:5141/api/fingerprint/image/{similarImage}";
            Console.WriteLine(imageUrl);
            await FetchAndDisplayImageAsync(imageUrl);

            _AppState.ResultImageFilename = similarImage;
            _AppState.Similarity = $"{percentage} %";
            _AppState.ExecutionTime = $"{executionTime} ms";
            SimilarityTextBlock.Text = _AppState.Similarity;
            // ExecutionTimeTextBlock.Text = _AppState.ExecutionTime;

            string apiUrl =
            $"http://localhost:5141/api/fingerprint/biodata/{similarImage}";

            var biodataResponse = await client.GetAsync(apiUrl);
            if (biodataResponse.IsSuccessStatusCode)
            {
                var biodata = await biodataResponse.Content.ReadAsAsync<KTPData>();

                _AppState.ktpData.name = biodata.name;
                _AppState.ktpData.NIK = biodata.NIK;
                _AppState.ktpData.birth_place = biodata.birth_place;
                _AppState.ktpData.birth_date = biodata.birth_date;
                _AppState.ktpData.gender = biodata.gender;
                _AppState.ktpData.blood_type = biodata.blood_type;
                _AppState.ktpData.address = biodata.address;
                _AppState.ktpData.religion = biodata.religion;
                _AppState.ktpData.marriage_status = biodata.marriage_status;
                _AppState.ktpData.job = biodata.job;
                _AppState.ktpData.citizenship = biodata.citizenship;
                // _AppState.ktpData.executionTimeKTP = biodata.executionTimeKTP;

```

```

        // Console.WriteLine("Masuk seocond");
        // ubah executionTime dan executionTimeKTP menjadi long, lalu
jumlahkan waktu executionTime dengan executionTimeKTP
        // long totalExecutionTime = executionTime +
long.Parse(_appState.ktpData.executionTimeKTP);
        // // test print
        // Console.WriteLine($"Execution TOTAL: {totalExecutionTime} ms");

        // // ubah totalExecutionTime menjadi string
        // _appState.ExecutionTime = $"{totalExecutionTime} ms";
        // // ExecutionTimeTextBlock.Text = _appState.ExecutionTime;
    }
else
{
    await ShowMessageAsync("No matching biodata found.");
}

if (!exactMatchFound)
{
    if (algorithm == "BM")
    {
        await ShowMessageAsync("No exact match found using BM. The
search is done using Hamming Distance.");
    }
    else if (algorithm == "KMP")
    {
        await ShowMessageAsync("No exact match found using KMP. The
search is done using Hamming Distance.");
    }
}

// ExecutionTimeTextBlock.Text = _appState.ExecutionTime; // catcher
overall
}
else
{
    await ShowMessageAsync("Error processing image.");
}
}
catch (Exception ex)
{
    await ShowMessageAsync($"Exception occurred: {ex.Message}");
}

private async Task ShowMessageAsync(string message)
{
    var messageBox = new MessageBox { Message = message };
    await messageBox.ShowDialog(this);
}

private async Task FetchAndDisplayImageAsync(string imageUrl)
{
    try
    {
        using (var client = new HttpClient())
        {

```

```

        var response = await client.GetAsync(imageUrl).ConfigureAwait(false);

        if (response.IsSuccessStatusCode)
        {
            var imageBytes = await
response.Content.ReadAsByteArrayAsync().ConfigureAwait(false);

            await Task.Run(async () =>
{
                using (var stream = new MemoryStream(imageBytes))
{
                    var bitmap = new Bitmap(stream);

                    await Dispatcher.UIThread.InvokeAsync(() =>
{
                        ResultsImage.Source = bitmap;
                        _appState.ResultImage = bitmap;
                    });
                }
            );
        }
        else
        {
            await ShowMessageAsync("Error fetching image from the server.");
        }
    }
    catch (Exception ex)
{
    await ShowMessageAsync($"Exception occurred while fetching the image:
{ex.Message}");
}
}
}
}

```

## 6. ThirdWindow.xaml

Halaman ketiga dari aplikasi, menampilkan biodata.

```

<Window xmlns="https://github.com/avaloniaui"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    x:Class="newjeans_avalonia.ThirdWindow"
    Title="Printuyul"
    Width="990" Height="646" CanResize="False"
    WindowStartupLocation="CenterOwner">

<Window.Background>
    <ImageBrush Source="avares://newjeans_avalonia/assets/home3.png"/>
</Window.Background>

<Window.Styles>
    <!-- Custom Button Style -->
    <Style Selector="Button">
        <Setter Property="Background" Value="Transparent"/>
        <Setter Property="BorderBrush" Value="Transparent"/>
    
```

```

        <Setter Property="BorderThickness" Value="0"/>
        <Setter Property="Template">
            <ControlTemplate>
                <Border Background="{TemplateBinding Background}"
                    BorderBrush="{TemplateBinding BorderBrush}"
                    BorderThickness="{TemplateBinding BorderThickness}">
                    <ContentPresenter Content="{TemplateBinding Content}"
                        HorizontalAlignment="{TemplateBinding
HorizontalContentAlignment}""
                        VerticalAlignment="{TemplateBinding
VerticalContentAlignment}"/>
                </Border>
            </ControlTemplate>
        </Setter>
    </Style>
</Window.Resources>

<Panel>
    <Button x:Name="BackButton" Width="107" Height="51" Margin="215,440,0,0"
        Background="Transparent" BorderBrush="Transparent" BorderThickness="0">
        <Image Source="avares://newjeans_avalonia/assets/back.png"/>
    </Button>

    <Button x:Name="RetryButton" Width="107" Height="51" Margin="668,440,0,0"
        Background="Transparent" BorderBrush="Transparent" BorderThickness="0">
        <Image Source="avares://newjeans_avalonia/assets/retry.png"/>
    </Button>

    <!-- Text placeholders -->
    <TextBlock x:Name="NamaText" FontSize="24" FontFamily="{StaticResource FairfaxFont}"
Margin="436,126,0,0" Foreground="#3533BD"/>
    <TextBlock x:Name="NikText" FontSize="24" FontFamily="{StaticResource FairfaxFont}"
Margin="436,198,0,0" Foreground="#3533BD"/>
    <TextBlock x:Name="TempatLahirText" FontFamily="{StaticResource FairfaxFont}"
FontSize="24" Margin="436,222,0,0" Foreground="#3533BD"/>
    <TextBlock x:Name="TanggalLahirText" FontFamily="{StaticResource FairfaxFont}"
FontSize="24" Margin="436,246,0,0" Foreground="#3533BD"/>
    <TextBlock x:Name="JenisKelaminText" FontFamily="{StaticResource FairfaxFont}"
FontSize="24" Margin="436,270,0,0" Foreground="#3533BD"/>
    <TextBlock x:Name="GolonganDarahText" FontFamily="{StaticResource FairfaxFont}"
FontSize="24" Margin="436,294,0,0" Foreground="#3533BD"/>
    <TextBlock x:Name="AlamatText" FontFamily="{StaticResource FairfaxFont}" FontSize="24"
Margin="436,318,0,0" Foreground="#3533BD"/>
    <TextBlock x:Name="AgamaText" FontFamily="{StaticResource FairfaxFont}" FontSize="24"
Margin="436,342,0,0" Foreground="#3533BD"/>
    <TextBlock x:Name="StatusPerkawinanText" FontFamily="{StaticResource FairfaxFont}"
FontSize="24" Margin="436,366,0,0" Foreground="#3533BD"/>
    <TextBlock x:Name="PekerjaanText" FontFamily="{StaticResource FairfaxFont}"
FontSize="24" Margin="436,390,0,0" Foreground="#3533BD"/>
    <TextBlock x:Name="KewarganegaraanText" FontFamily="{StaticResource FairfaxFont}"
FontSize="24" Margin="436,414,0,0" Foreground="#3533BD"/>

    <Image x:Name="LoadingImage" Source="avares://newjeans_avalonia/assets/loading.png"
        Width="990" Height="646" HorizontalAlignment="Center" VerticalAlignment="Center"
        IsVisible="false"/>
</Panel>

</Window>

```

## 7. ThirdWindow.axaml.cs

*Styling* untuk halaman ketiga dari aplikasi.

```
using Avalonia.Controls;
using Avalonia.Media.Imaging;
using System;
using System.IO;
using System.Net.Http;
using System.Threading.Tasks;
using Avalonia.Threading;

namespace newjeans_avalonia
{
    public partial class ThirdWindow : Window
    {
        private AppState _appState;

        public ThirdWindow(AppState appState)
        {
            InitializeComponent();
            _appState = appState;

            FetchAndDisplayBiodata();

            this.FindControl<Button>("BackButton")!.Click += OnBackButtonClick;
            this.FindControl<Button>("RetryButton")!.Click += OnRetryButtonClick;
        }

        private async void FetchAndDisplayBiodata()
        {
            if (_appState.ResultImage != null)
            {
                NamaText.Text = _appState.ktpData.name;
                NikText.Text = _appState.ktpData.NIK;
                TempatLahirText.Text = _appState.ktpData.birth_place;
                TanggalLahirText.Text = _appState.ktpData.birth_date;
                JenisKelaminText.Text = _appState.ktpData.gender;
                GolonganDarahText.Text = _appState.ktpData.blood_type;
                AlamatText.Text = _appState.ktpData.address;
                AgamaText.Text = _appState.ktpData.religion;
                StatusPerkawinanText.Text = _appState.ktpData.marriage_status;
                PekerjaanText.Text = _appState.ktpData.job;
                KewarganegaraanText.Text = _appState.ktpData.citizenship;
            }
        }

        private async Task ShowMessageAsync(string message)
        {
            var messageBox = new MessageBox { Message = message };
            await messageBox.ShowDialog(this);
        }

        private void OnBackButtonClick(object? sender, Avalonia.Interactivity.RoutedEventArgs e)
        {
            SecondWindow secondWindow = new SecondWindow(_appState);
            secondWindow.Show();
            this.Close();
        }
    }
}
```

```

        }

    private void OnRetryButtonClick(object? sender, Avalonia.Interactivity.RoutedEventArgs e)
    {
        ResetAppState();
        SecondWindow secondWindow = new SecondWindow(_appState);
        secondWindow.Show();
        this.Close();
    }

    private void ResetAppState()
    {
        _appState.CurrentImage = null;
        _appState.ResultImage = null;
        _appState.SelectedAlgorithm = null;
        _appState.Similarity = string.Empty;
        _appState.ExecutionTime = string.Empty;
        _appState.ResultImageFilename = null;
        _appState.ktpData = new KTPData("default_NIK", "default_name",
"default_birth_place", "default_birth_date",
"default_address", "default_religion",
"default_citizenship");
    }
}
}

```

## 8. MessageBox.axaml

Digunakan untuk mengatur *message box*.

```

<Window xmlns="https://github.com/avaloniaui"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    x:Class="newjeans_avalonia.MessageBox"
    Title="Notification"
    Width="300" Height="148.7" WindowStartupLocation="CenterOwner">

    <Window.Background>
        <ImageBrush Source="avares://newjeans_avalonia/assets/notification_box.png"/>
    </Window.Background>

    <Window.Styles>
        <!-- Custom Button Style -->
        <Style Selector="Button">
            <Setter Property="Background" Value="Transparent"/>
            <Setter Property="BorderBrush" Value="Transparent"/>
            <Setter Property="BorderThickness" Value="0"/>
            <Setter Property="Template">
                <ControlTemplate>
                    <Border Background="{TemplateBinding Background}"
                        BorderBrush="{TemplateBinding BorderBrush}"
                        BorderThickness="{TemplateBinding BorderThickness}">
                        <ContentPresenter Content="{TemplateBinding Content}"
                            HorizontalAlignment="{TemplateBinding
HorizontalContentAlignment}"
                            VerticalAlignment="{TemplateBinding
VerticalContentAlignment}"/>
                    </Border>
                </ControlTemplate>
            </Setter>
        </Style>
    </Window.Styles>

```

```

VerticalContentAlignment}"/>
    </Border>
    </ControlTemplate>
</Setter>
</Style>
</Window.Styles>

<StackPanel VerticalAlignment="Center" HorizontalAlignment="Center">
    <TextBlock x:Name="MessageTextBlock" TextWrapping="Wrap" Text="Default Message"
HorizontalAlignment="Center"
    Margin="10"
    Foreground="#FFFFFF" FontSize="16" FontFamily="{StaticResource FairfaxFont}" />
    <Button x:Name="OkButton" Width="80" Height="30" Margin="10"
HorizontalAlignment="Center">
        <Image Source="avares://newjeans_avalonia/assets/ok.png"/>
    </Button>
</StackPanel>
</Window>

```

## 9. MessageBox.axaml.cs

*Styling* untuk pesan kesalahan dari message box.

```

using Avalonia.Controls;
using Avalonia.Interactivity;

namespace newjeans_avalonia
{
    public partial class MessageBox : Window
    {
        public MessageBox()
        {
            InitializeComponent();
            OkButton.Click += OkButton_Click;
        }

        public string Message
        {
            get => MessageTextBlock.Text!;
            set => MessageTextBlock.Text = value;
        }

        private void OkButton_Click(object? sender, RoutedEventArgs e)
        {
            this.Close();
        }
    }
}

```

## 4.4 Tata Cara Penggunaan Program

Program dapat dijalankan dengan cara membuka terminal dari root directory, lalu jalankan perintah `run.bat` pada folder (bisa dilakukan dengan double click pada file `run.bat`). Program akan

secara otomatis menjalankan backend dan front end hingga terbuka pop up dan menampilkan tampilan awal aplikasi. Setelah itu, user dapat langsung menggunakan program dengan:

1. Klik tombol "Start Here" untuk melanjutkan ke halaman berikutnya
2. Klik tombol "Insert" untuk mengupload gambar sidik jari yang hendak dicari. File gambar yang diperbolehkan hanyalah yang berekstensi .bmp
3. Pilih algoritma yang ingin digunakan pada dropdown di bagian bawah halaman. Pada bagian ini terdapat dua algoritma yang dapat digunakan, yaitu algoritma Knut Morris Pratt dan algoritma Boyer Moore.
4. Klik tombol "Search" untuk memulai pencarian sidik jari yang telah diupload. Tampilan Loading akan muncul selama proses pencarian berlangsung.
5. Setelah proses pencarian selesai, hasil pencarian akan muncul pada bagian kanan halaman. Hasil ini mencakup gambar sidik jari yang paling mirip, persentase kemiripan kedua gambar tersebut, dan waktu eksekusinya (termasuk waktu overhead fetch data dari database).
6. Pengguna dapat menekan tombol "See Details" untuk melihat detail dari hasil pencarian. Detail ini mencakup biodata dari pemilik sidik jari yang ditemukan.
7. Pengguna dapat menekan tombol "Back" untuk kembali ke halaman sebelumnya.
8. Pengguna juga dapat menekan tombol "Retry" untuk melakukan pencarian sidik jari lainnya.

## 4.5 Hasil Pengujian

### 4.5.1 Antarmuka Aplikasi

Tampilan Awal Halaman Home

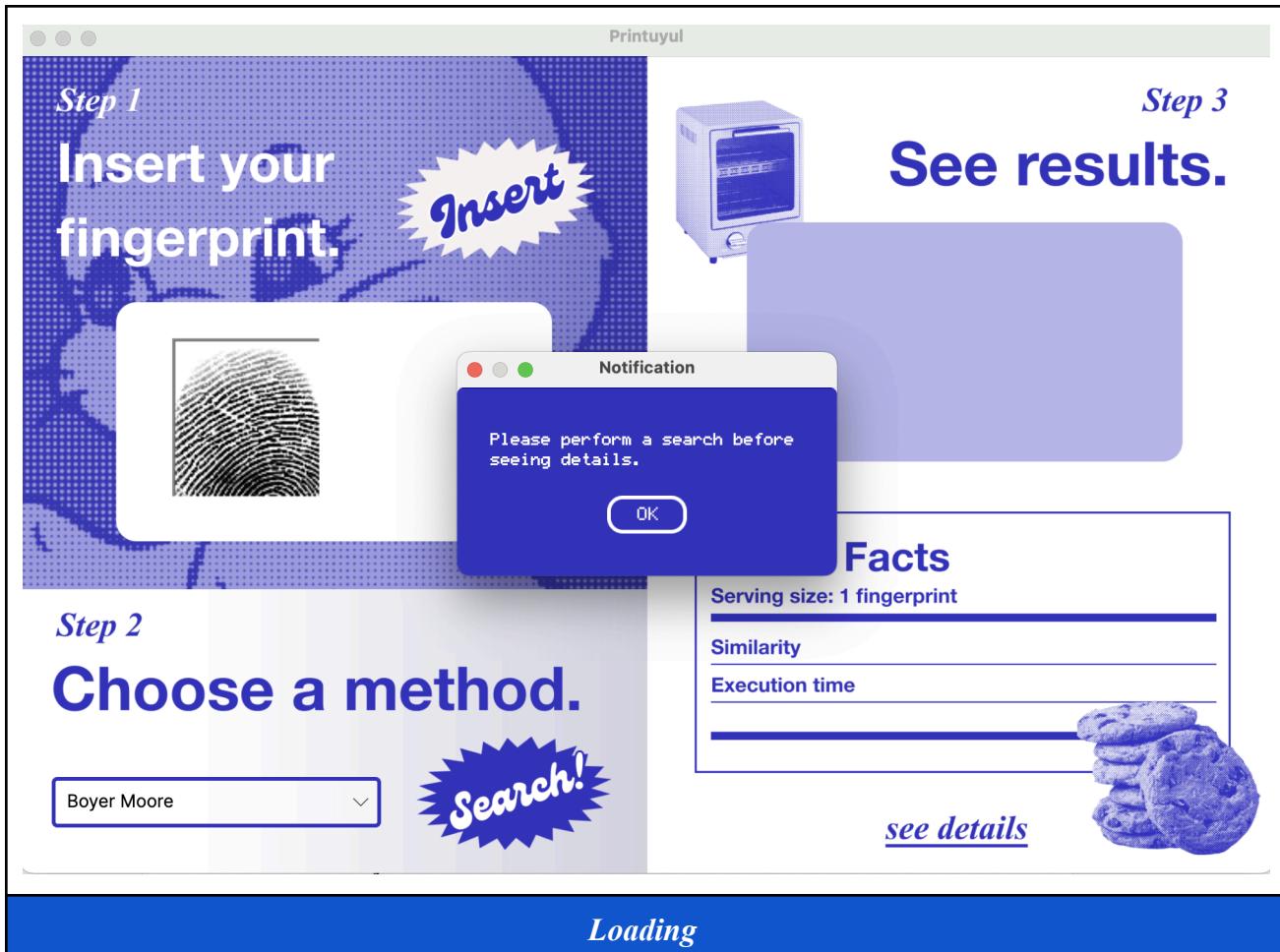


#### 4.5.2 Field Input

Search Ketika Belum Upload Image











#### 4.5.3 Uji Coba Kasus

**1: Algoritma BM, exact match**

**Step 1**  
Insert your fingerprint.

**Step 2**  
Choose a method.  
Boyer Moore

**Step 3**  
See results.

Result Facts  
Serving size: 1 fingerprint  
Similarity 100 %  
Execution time 12951 ms  
[see details](#)

**Result Details for Ruth Price**

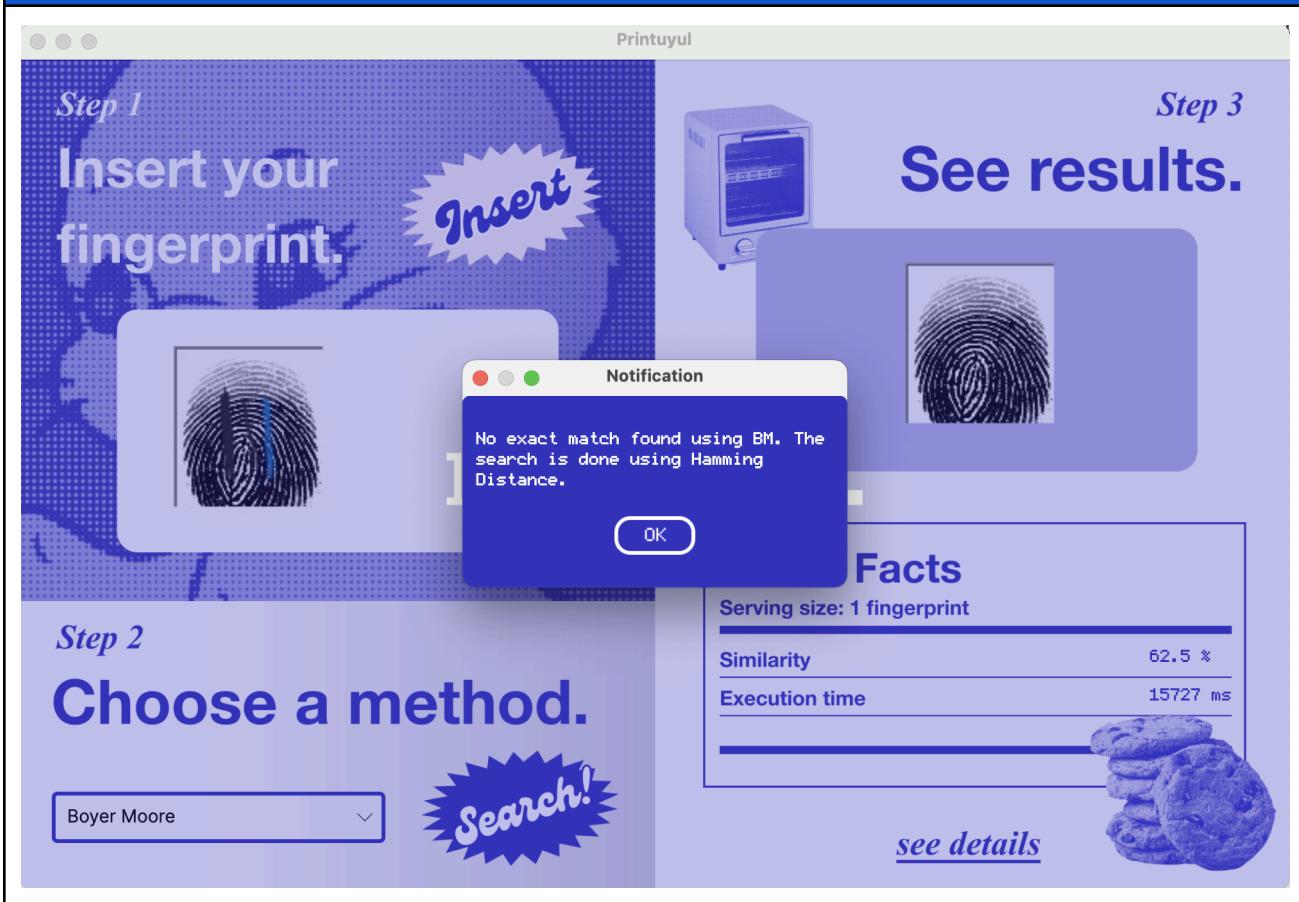
NIK	: 0647889245898855
TEMPAT LAHIR	: Banyuwangi
TANGGAL LAHIR	: 1988-01-05
JENIS KELAMIN	: Laki-Laki
GOLONGAN DARAH	: A
ALAMAT	: Jl. Rth pRc3 No. 87
AGAMA	: Katolik
STATUS PERKAWINAN	: Menikah
PEKERJAAN	: Tukang Batu
KEWARGANEGARAAN	: Perancis

BACK    RETRY    THANK YOU FOR USING OUR PROGRAM!

**Gambar jari yang digunakan berbeda dengan 1, tetapi milik orang yang sama**



2: Algoritma BM, tidak *exact match* (dicoret, kemiripan masih  $\geq 50\%$ , hasil data orangnya benar)





Gambar jari yang digunakan sama dengan 2, tetapi dicoret lebih parah (kemiripan 50%) sehingga data orang yang dikeluarkan salah



**Step 1**  
Insert your fingerprint.  
**Insert!**

**Step 2**  
Choose a method.  
Boyer Moore  
**Search!**

**Step 3**  
**See results.**

**Result Facts**  
Serving size: 1 fingerprint

Similarity	50 %
Execution time	41139 ms

[see details](#)

**Printuyul**

NAMA : Nicole Wheeler

NIK : 992673988131032  
TEMPAT LAHIR : Sumenep  
TANGGAL LAHIR : 1995-09-01  
JENIS KELAMIN : Laki-Laki  
GOLONGAN DARAH : A  
ALAMAT : Jl. NICOLE Wh3LER No. 47  
AGAMA : Protestan  
STATUS PERKAWINAN : Cerai  
PEKERJAAN : Teknisi  
KEWARGANEGARAAN : Laos

BACK       RETRY

THANK YOU FOR USING OUR PROGRAM!

### 3: Algoritma KMP, exact match

**Step 1**  
Insert your fingerprint.  
**Insert!**

**Step 2**  
Choose a method.  
Knuth Morris Pratt  
**Search!**

**Step 3**  
**See results.**

**Result Facts**  
Serving size: 1 fingerprint

Similarity	100 %
Execution time	14059 ms

[see details](#)

**Printuyul**

NAMA : Ruth Price

NIK : 0647889245898855  
TEMPAT LAHIR : Banguwangi  
TANGGAL LAHIR : 1988-01-05  
JENIS KELAMIN : Laki-Laki  
GOLONGAN DARAH : A  
ALAMAT : Jl. Rth pRc3 No. 87  
AGAMA : Katolik  
STATUS PERKAWINAN : Menikah  
PEKERJAAN : Tukang Batu  
KEWARGANEGARAAN : Perancis

BACK       RETRY

THANK YOU FOR USING OUR PROGRAM!

Gambar jari yang digunakan berbeda dengan 1, tetapi milik orang yang sama

**Step 1**  
Insert your fingerprint.  
**Insert!**

**Step 2**  
Choose a method.  
Knuth Morris Pratt  
**Search!**

**Step 3**  
**See results.**

**Result Facts**  
Serving size: 1 fingerprint

Similarity	100 %
Execution time	13760 ms

[see details](#)

**Printuyul**

NAMA : Ruth Price

NIK : 0647889245898855  
TEMPAT LAHIR : Banguwangi  
TANGGAL LAHIR : 1988-01-05  
JENIS KELAMIN : Laki-Laki  
GOLONGAN DARAH : A  
ALAMAT : Jl. Rth pRc3 No. 87  
AGAMA : Katolik  
STATUS PERKAWINAN : Menikah  
PEKERJAAN : Tukang Batu  
KEWARGANEGARAAN : Perancis

BACK       RETRY

THANK YOU FOR USING OUR PROGRAM!

### 4: Algoritma KMP, tidak *exact match* (dicoret, kemiripan masih $\geq 50\%$ , hasil data orangnya benar)

Printuyul

**Step 1**

Insert your fingerprint.

**Step 2**

Choose a method.

**Step 3**

See results.

Notification

No exact match found using KMP. The search is done using Hamming Distance.

OK

Facts

Serving size: 1 fingerprint

Similarity 62.5 %

Execution time

[see details](#)

Knuth Morris Pratt

Search!

Printuyul

Result Facts	
Serving size: 1 fingerprint	
Similarity	62.5 %
Execution time	19324 ms
<a href="#">see details</a>	

Printuyul

Result Facts	
NAMA	: Ruth Price
NIK	: 0647889245898855
TEMPAT LAHIR	: Banyuwangi
TANGGAL LAHIR	: 1988-01-05
JENIS KELAMIN	: Laki-Laki
GOLONGAN DARAH	: A
ALAMAT	: Jl. Rth pRc3 No. 87
AGAMA	: Katolik
STATUS PERKAWINAN	: Menikah
PEKERJAAN	: Tukang Batu
KEWARGANEGARAAN	: Perancis

BACK

RETRY

THANK YOU FOR USING OUR PROGRAM!

**Gambar jari yang digunakan sama dengan 4, tetapi dicoret lebih parah (kemiripan 50%) sehingga data orang yang dikeluarkan salah**





## 4.6 Analisis Hasil Pengujian

Penggunaan antara metode KMP dan BM menggunakan pendekatan yang berbeda. Namun keduanya sama-sama digunakan untuk pencocokan string. Efisiensi kedua algoritma ini tergantung dari kasus persoalannya, namun pencarian dengan algoritma KMP cenderung lebih lama apabila dibandingkan dengan BM. Berikut merupakan hasil analisis terhadap pengujian yang telah dilakukan.

Dari semua pengujian *test case* yang dilakukan, didapatkan bahwa algoritma KMP dan BM sama-sama menghasilkan solusi yang sama. Yang membedakan keduanya adalah waktu eksekusi pencarian. Ketika tidak ditemukan kecocokan tepat (*exact match*) antara pola gambar yang diunggah dan seluruh gambar referensi dalam database, program beralih menggunakan Hamming Distance untuk mencari solusi. Hal ini membuat waktu eksekusi program cenderung lebih lama, karena pencarian dilakukan terlebih dahulu dengan algoritma KMP atau BM, dan baru kemudian dilanjutkan dengan Hamming Distance. Selain itu, perbandingan waktu eksekusi antara KMP dan BM tidak terlalu signifikan, dengan BM rata-rata lebih cepat sekitar 2 detik.

Baik algoritma KMP dan BM memiliki kelebihan dan kekurangannya masing-masing. KMP memiliki kelebihan dalam efisiensi dan stabilitas, menghindari perulangan perbandingan yang tidak perlu dan menunjukkan waktu eksekusi yang relatif konsisten. Namun, pembuatan tabel border memerlukan waktu *preprocessing* tambahan. Di sisi lain, BM menonjol dalam kecepatan dan efisiensi pencarian dalam kasus umum serta pada teks yang sangat panjang. Namun, kinerjanya dapat menurun pada pola atau teks dengan banyak pengulangan karakter, dan proses *preprocessing* untuk membuat tabel heuristik memerlukan waktu tambahan.

Dalam pengujian tambahan terhadap *test case* yang terdiri dari satu blok warna yang sama, seperti gambar *full* merah atau *full* putih, ditemukan bahwa program belum mampu menangani kasus semacam itu. Hasil pengujian menunjukkan bahwa program mengembalikan *exact match* padahal gambar tersebut sama sekali tidak terdapat dalam database gambar referensi.

Masalah ini muncul karena algoritma pemotongan gambar yang dilakukan dalam program tidak cukup sensitif untuk membedakan pola gambar dengan blok warna yang sama. Pola ASCII yang dihasilkan dari gambar-gambar tersebut cenderung serupa, bahkan ketika gambar sebenarnya berbeda secara visual. Sebagai akibatnya, program menganggap pola tersebut sebagai *exact match* meskipun gambar yang sebenarnya tidak ada dalam database referensi.

## **BAB V**

### **PENUTUP**

#### **5.1 Kesimpulan**

Dalam tugas ini, digunakan algoritma Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM) untuk pencarian pola dalam proses identifikasi sidik jari. Algoritma KMP dan BM dipilih karena efisiensinya dalam pencarian string, di mana KMP memanfaatkan tabel border untuk menghindari perbandingan yang tidak perlu dan BM menggunakan tabel heuristik untuk melakukan lompatan pencarian yang lebih jauh. Jika kedua algoritma ini tidak menemukan kecocokan yang tepat, maka digunakan pendekatan Hamming Distance untuk mengukur kemiripan dengan tingkat tertentu. Hamming Distance memberikan solusi alternatif dengan menghitung perbedaan posisi antara karakter pada dua string.

Kami berhasil mengimplementasikan aplikasi desktop berbasis GUI menggunakan C# dan Avalonia yang memanfaatkan algoritma KMP, BM, dan Hamming Distance untuk mencari kemiripan gambar sidik jari. Penggunaan algoritma-algoritma tersebut telah membantu dalam menemukan sidik jari yang mirip dengan efisien dan efektif, masing-masing dengan kelebihan dan kekurangannya.

Pengembangan aplikasi dilakukan dengan menggunakan bahasa pemrograman C# untuk backend dan Avalonia untuk frontend. Kombinasi teknologi ini memungkinkan kami untuk menciptakan aplikasi desktop yang responsif dan mudah digunakan. Melalui tugas ini, kami banyak belajar tentang penggunaan framework dan teknologi pengembangan aplikasi desktop terkini.

#### **5.2 Saran**

Berdasarkan hasil pengembangan aplikasi ini, kami memiliki beberapa saran untuk perbaikan dan pengembangan di masa yang akan datang:

1. Optimisasi Algoritma:

Algoritma KMP, BM, dan Hamming Distance, serta dapat dioptimalkan lebih lanjut dengan teknik-teknik preprocessing untuk memastikan pencarian gambar tidak terhambat oleh variabel seperti perbedaan ukuran, rotasi gambar, maupun noise gambar. Metode perhitungan persentase kemiripan pun dapat ditelaah lebih lanjut agar memberikan efektivitas yang maksimal.

## 2. Peningkatan Efisiensi Backend:

Meskipun aplikasi ini telah berfungsi dengan baik, ada peluang untuk meningkatkan efisiensi backend dengan penggunaan teknologi caching atau optimalisasi akses data. Implementasi caching pada sisi aplikasi dapat mempercepat waktu respon dan mengurangi beban sistem. Dapat juga diperhatikan lebih lanjut pada waktu yang banyak terbuang hanya untuk overhead fetch database agar dapat diminimalisir.

## 5.3 Refleksi

Dalam proses pengembangan aplikasi desktop ini, kami belajar banyak tentang strategi algoritma khususnya pada pendekatan Knuth-Morris-Pratt (KMP), Boyer-Moore (BM), dan Hamming Distance, serta penggunaan bahasa pemrograman C# dan pengembangan antarmuka pengguna menggunakan Avalonia. Kami juga memperoleh pemahaman yang lebih mendalam tentang pentingnya merancang algoritma pencarian yang efisien dalam menyelesaikan permasalahan identifikasi sidik jari. Di sini, kami juga belajar bagaimana mengintegrasikan kedua bagian aplikasi ini.

Selain itu, kolaborasi antar anggota kelompok dalam pengembangan aplikasi ini memberikan pengalaman berharga dalam koordinasi dan integrasi antara kedua bagian sistem. Kami berharap bahwa aplikasi yang kami buat dapat bermanfaat bagi kami sendiri dan bagi orang lain dalam proses identifikasi sidik jari, menjadi bahan pembelajaran mengenai algoritma pencarian, serta menjadi landasan untuk pengembangan lebih lanjut di masa yang akan datang.

## 5.4 Tanggapan

Menurut kami, tugas besar 3 ini merupakan sarana yang tepat untuk mengaplikasikan konsep algoritma *string matching*, baik algoritma KMP, BM, maupun *regex*. Dari tugas besar 3 ini juga, kami dapat menerapkan ilmu yang kami peroleh dari mata kuliah Basis Data dalam pembuatan *database* biodata dan sidik jari. Kami juga memperoleh kemampuan menggunakan bahasa C# dan melakukan pengembangan dalam .NET *environment*.

Banyak ucapan terimakasih kepada dosen dan kakak-kakak asisten, terutama dalam perpanjangan tenggat waktu dari yang awalnya 25 Mei menjadi 9 Juni. *Overall*, tugas ini dapat menjadi wadah eksplorasi kami dalam perjalanan menjadi *programmer* yang lebih baik.

## LAMPIRAN

Link *repository*: [https://github.com/NopalAul/Tubes3\\_Printuyul](https://github.com/NopalAul/Tubes3_Printuyul)

Link video: <https://youtu.be/80WNFNvIH18>

## **DAFTAR PUSTAKA**

Munir, Rinaldi. 2024. String Matching dengan Regular Expression.

Diakses dari

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

Munir, Rinaldi. 2024. Pencocokan String.

Diakses dari

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