

# **Tugas Besar III IF2211 Strategi Algoritma: Penerapan String Matching dan Regular Expression dalam DNA Pattern Matching**



## **Anggota Kelompok :**

Dwi Kalam Amal Tauhid	13519210
Aditya Prawira Nugroho	13520049
Nathanael Santoso	13520129

PROGRAM STUDI TEKNIK INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
BANDUNG  
2022

## DAFTAR ISI

DAFTAR ISI.....	1
BAB 1 .....	2
BAB 2 .....	3
A.    Algoritma KMP.....	3
B.    Algoritma Boyer-Moore.....	3
C.    Regex.....	4
D.    Website yang Dibangun .....	5
BAB 3 .....	5
A.    Langkah-langkah Pemecahan Masalah .....	5
A.1 Penambahan Penyakit ke Database .....	5
A.2 Pencocokan DNA pada Tes .....	6
A.3 Pencarian Hasil Tes.....	6
B.    Fitur Fungsional dan Arsitektur Aplikasi Web yang Dibangun.....	6
BAB 4 .....	7
A.    Spesifikasi Teknis Program.....	7
A.1 Struktur Data .....	7
A.2 Fungsi dan Prosedur.....	7
B.    Penjelasan Tata Cara Penggunaan Program .....	7
B.1 Penambahan Penyakit ke Database .....	7
B.2 Tes DNA Pengguna.....	8
B.3 Pencarian Hasil Tes .....	9
C.    Hasil Pengujian .....	9
C.1 Menambahkan Penyakit .....	9
C.2 Mengetes DNA.....	10
D.    Analisis Hasil Pengujian .....	15
BAB 5 .....	16
A.    Kesimpulan.....	16
B.    Saran.....	16
C.    Komentar dan Refleksi.....	16
DAFTAR PUSTAKA .....	17
LAMPIRAN.....	18

# BAB 1

## DESKRIPSI TUGAS

Dalam tugas besar ini, anda diminta untuk membangun sebuah aplikasi DNA Pattern Matching. Dengan memanfaatkan algoritma String Matching dan Regular Expression yang telah anda pelajari di kelas IF2211 Strategi Algoritma, anda diharapkan dapat membangun sebuah aplikasi interaktif untuk mendeteksi apakah seorang pasien mempunyai penyakit genetik tertentu. Hasil prediksi tersebut dapat disimpan pada basis data untuk kemudian dapat ditampilkan berdasarkan query pencarian. Contoh masukan aplikasi:

Fitur-Fitur Aplikasi:

1. Aplikasi dapat menerima input penyakit baru berupa nama penyakit dan sequence DNA-nya (dan dimasukkan ke dalam database).
  - a. Implementasi input sequence DNA dalam bentuk file.
  - b. Dilakukan sanitasi input menggunakan regex untuk memastikan bahwa masukan merupakan sequence DNA yang valid (tidak boleh ada huruf kecil, tidak boleh ada huruf selain AGCT, dan tidak ada spasi).
2. Aplikasi dapat memprediksi seseorang menderita penyakit tertentu berdasarkan sequence DNA-nya.
  - a. Tes DNA dilakukan dengan menerima input nama pengguna, sequence DNA pengguna, dan nama penyakit yang diuji. Asumsi sequence DNA pengguna > sequence DNA penyakit.
  - b. Dilakukan sanitasi input menggunakan regex untuk memastikan bahwa masukan merupakan sequence DNA yang valid (tidak boleh ada huruf kecil, tidak boleh ada huruf selain AGCT, tidak ada spasi, dll).
  - c. Pencocokan sequence DNA dilakukan dengan menggunakan algoritma string matching.
  - d. Hasil dari tes DNA berupa tanggal tes, nama pengguna, nama penyakit yang diuji, dan status hasil tes. Contoh: 1 April 2022 - Mhs IF - HIV – False
  - e. Semua komponen hasil tes ini dapat ditampilkan pada halaman web (refer ke poin 3 pada “Fitur-Fitur Aplikasi”) dan disimpan pada sebuah tabel database.
3. Aplikasi memiliki halaman yang menampilkan urutan hasil prediksi dengan kolom pencarian di dalamnya. Kolom pencarian bekerja sebagai filter dalam menampilkan hasil.
  - a. Kolom pencarian dapat menerima masukan dengan struktur: , contoh “13 April 2022 HIV”. Format penanggalan dibebaskan, jika bisa menerima >1 format lebih baik.
  - b. Kolom pencarian dapat menerima masukan hanya tanggal ataupun hanya nama penyakit. Fitur ini diimplementasikan menggunakan regex.

## BAB 2

### LANDASAN TEORI

#### A. Algoritma KMP

Algoritma KMP atau Knuth-Morris-Pratt mencari pola dalam sebuah teks dari kiri ke kanan (seperti algoritma *brute force*). Akan tetapi, algoritma ini memindahkan pola sedikit lebih bagus dibandingkan *brute force*. Secara singkat, algoritma ini akan mencari kemunculan dari sebuah kata dalam serangkaian string dengan cara membuat pengamatan sedemikian sehingga ketika terjadi ketidakcocokan, algoritma ini akan menyimpan informasi dari serangkaian string untuk mengira dimana kata akan ditemukan. Sehingga akan melompati tahap pengecekan yang redundan.

Algoritma ini melakukan preproses pada pola untuk mencari prefiks yang sama dari pola dengan pola itu sendiri. Preproses tersebut disebut dengan fungsi pinggiran KMP. Misalkan terdapat sebuah pola/string  $P[]$ ,  $j$  adalah posisi ketidakcocokan dalam  $P[]$ ,  $k$  adalah posisi sebelum  $j$  ( $k = j - 1$ ). Maka, fungsi pinggiran  $b(k)$  didefinisikan sebagai ukuran dari prefiks terbesar dari  $P[0..k]$  yang juga merupakan sufiks dari  $P[1..k]$ . Fungsi ini juga disebut sebagai *failure function* (disingkat: *fail*).

Guna dari fungsi pinggiran adalah ketika ditemukan ketidakcocokan pada  $j$ , maka pengecekan tidak selalu dilakukan pada  $P[j+1]$ , melainkan digeser sejauh panjang  $P$  dikurangi panjang sufiks terpanjang pada prefiks pola yang akan dicari.

Kompleksitas waktu yang dimiliki oleh KMP adalah  $O(m + n)$ . Hal tersebut didapatkan dari perhitungan fungsi pinggiran, yaitu  $O(m)$  dan pencarian string, yaitu  $O(n)$ . Keuntungan dari algoritma KMP adalah algoritma ini tidak perlu bergerak mundur pada input, sehingga algoritma ini cocok untuk memproses file berukuran besar. Akan tetapi, KMP tidak terlalu baik dalam menyelesaikan pencarian string apabila range alfabet pada string bertambah.

#### B. Algoritma Boyer-Moore

Algoritma Boyer-Moore merupakan sebuah algoritma yang sangkil dan menjadi standar pembandingan dalam literatur *string-search*. Algoritma ini didasarkan dua teknik, yaitu *the looking-glass technique* dan *the character-jump technique*. *Looking glass technique* adalah teknik yang mencari sebuah pola  $P$  dalam teks  $T$  dengan bergerak dari belakang ke depan. *Character jump technique* adalah teknik yang melompati pencarian ketika terjadi ketidakcocokan pada pola  $P$  ke- $j$  dan Teks  $T$  ke- $i$ .

Terdapat 3 kasus untuk algoritma KMP. Kasus pertama adalah jika pola  $P$  mengandung sebuah karakter  $x$ , kemudian  $P$  digeser ke kanan untuk menyamakan kemunculan terakhir dari  $x$  di  $P$  dengan teks  $T$  ke- $i$ . Kasus kedua adalah jika  $P$  mengandung sebuah karakter  $x$ , tetapi penggeseran ke kanan hingga kemunculan terakhir tidak mungkin, maka geser  $P$  ke kanan dari 1 karakter hingga  $T[i+1]$ . Jika kasus pertama dan kedua tidak dapat digunakan, maka geser  $P$  untuk menyamakan  $P[0]$  dengan  $T[i+1]$ .

Seperti KMP yang memiliki *failure function*, Boyer-Moore memiliki fungsi *last occurrence*. Fungsi ini akan melakukan preproses pada pola P dan teks T untuk mencari semua kemunculan terakhir alfabet A-Z. *Worst case* dari Boyer-Moore adalah  $O(nm + A)$ . Boyer-Moore akan cepat ketika range alfabet pada teks tinggi, sebaliknya jika rendah maka akan lambat. Jadi Boyer-Moore akan cepat dalam pencarian English Text, tetapi lambat dalam pencarian biner yang hanya terdiri dari 0 dan 1. Meskipun begitu, Boyer-Moore jauh lebih cepat dibandingkan dengan brute force.

### C. Regex

Regular expression atau yang biasa disebut regex adalah sebuah sekuens karakter yang memberi tahu pola pencarian dalam teks. Pada umumnya, regex digunakan untuk pencarian string atau validasi input. Dalam regex, digunakan notasi untuk memberi informasi pola yang dicari harus seperti apa. Notasi-notasi tersebut adalah sebagai berikut:

- $\wedge$  Posisi mulai string
- $\cdot$  Semua karakter tunggal kecuali *newline*
- $[ ]$  Kurung iku menandakan sama dengan salah satu karakter dalam kurung siku
- $[^ ]$  Semua karakter kecuali yang ada di dalam kurung siku
- $\$$  Menandakan akhir string
- $( )$  Subekspresi
- $*$  Karakter muncul minimal 0
- $+$  Karakter muncul minimal 1 kali
- $\{m,n\}$  karakter muncul minimal m kali maksimal n kali

### D. Algoritma Jarak Levenshtein

Jarak Levenshtein adalah jumlah perubahan yang dapat dibuat terhadap suatu string “y” sehingga mendapatkan string “x”. Misalkan jika ada string “Hello” dan ada string “Hola”, maka dengan mengganti “e” menjadi “o”, “o” pada akhir menjadi “a” dan menghilangkan salah satu “l” maka bisa mendapatkan string “Hola” dari “Hello”. Pada contoh tersebut, jarak levenshtein adalah 3.

Jarak Levenshtein dapat dikomputasi dengan efisien menggunakan matriks sebesar  $m \times n$  dengan m sepanjang sequence pertama, dan n sepanjang sequence kedua. Kemudian, mengaplikasikan algoritma:

$$\begin{aligned}
 C_{i,0} &= i \\
 C_{0,j} &= j \\
 C_{i,j} &= \text{if } (x_i = y_j) \text{ then } C_{i-1,j-1} \\
 &\quad \text{else } 1 + \min(C_{i-1,j}, C_{i,j-1}, C_{i-1,j-1})
 \end{aligned}$$

Gambar 1. Algoritma Jarak Levenshtein

Dikutip dari: Navarro, Gonzalo (2001). "A guided tour to approximate string matching" [2]

Pada tugas ini, jarak levenshtein dipakai untuk menghitung kemiripan antara DNA pasien dan DNA penyakit. Pada algoritma Boyer-Moore dan Knuth-Morris-Pratt ditambahkan index awal dari substring yang paling mirip dengan sequence, dihitung dari index awal sequence saat gagal matching dengan panjang substring yang matching terpanjang. Kemudian diambil string sepanjang sequence penyakit mulai dari index awal substring dan dimasukkan ke dalam Algoritma Levenshtein. Jarak yang didapatkan dibagi dengan panjang sequence penyakit, dikalikan 100, dan dikurangkan terhadap 100 persen untuk mendapatkan tingkat kemiripan.

## E. Website yang Dibangun

Aplikasi web yang dibangun menggunakan bahasa pemrograman javascript dan MySQL untuk database penyimpanan penyakit dan hasil tes DNA. Untuk *frontend*, digunakan *framework* Next.js dan express untuk *backend*. Algoritma pencocokan string yang diimplementasikan adalah KMP dan Boyer-Moore. Sedangkan algoritma Levenshtein Distance diimplementasikan untuk menghitung kemiripan sekuens DNA.

Pada *frontend*, terdapat 4 halaman web, yaitu *homepage*, halaman pencarian hasil tes, halaman memasukkan penyakit baru ke database, dan halaman tes DNA. Pada pencarian, digunakan regex untuk pencocokan tanggal dan nama penyakit. Pada bagian *backend*, terdapat beberapa API yang dibuat untuk memberikan data pasien, menerima sekuens DNA, memberikan hasil tes. Setelah menerima sekuens DNA, dilakukan sanitasi untuk mengecek apakah DNA yang diunggah valid.

## BAB 3

### ANALISIS PEMECAHAN MASALAH

#### A. Langkah-langkah Pemecahan Masalah

##### A.1 Penambahan Penyakit ke Database

Ketika menambahkan penyakit ke dalam database, pengguna akan memasukkan nama penyakit dan file berformat .txt yang berisi sekuens DNA penyakit tersebut. Web akan melakukan pembacaan file dengan cara *client-side*. Setelah pembacaan dilakukan, *frontend* mengirimkan data nama dan sekuens DNA dalam bentuk JSON. Pada *backend* akan dilakukan sanitasi input terlebih dahulu. Jika valid, maka akan memanggil query yang memasukkan data baru ke dalam database. Jika belum ada nama penyakit yang sama, maka akan diberikan respons sukses. Sebaliknya, akan diberikan respons error jika terdapat duplikasi nama. Jika sekuens DNA tidak valid, akan diberikan respons error bahwa sekuens DNA bermasalah.

## A.2 Pencocokan DNA pada Tes

Saat pengguna ingin melakukan tes DNA, pengguna akan memasukkan nama, sekuens DNA berupa file .txt, prediksi penyakit yang dimiliki, dan memilih algoritma yang digunakan untuk melakukan testing, yaitu KMP atau Boyer-Moore. Setelah pengguna menekan tombol 'submit', *frontend* akan melakukan pembacaan file, kemudian mengirimkan sebuah body yang berisi nama, prediksi penyakit, sekuens DNA dalam bentuk JSON ke API yang sesuai pilihan algoritma. Pada bagian *backend* akan melakukan sanitasi sekuens DNA terlebih dahulu, jika tidak valid maka akan mengirimkan error. Jika valid, akan digunakan query mencari penyakit dengan nama penyakit yang sesuai pada database, jika tidak terdiksi ada penyakit tersebut, akan dikirimkan error. Jika ada, akan dilakukan pencocokan string sesuai algoritma yang dipilih. Jika hasilnya tidak sama, akan dilakukan algoritma kemiripan dengan penyakit. Hasil tes akan dimasukkan ke dalam database, kemudian server akan mengirimkan respons sukses.

## A.3 Pencarian Hasil Tes

Pengguna akan memasukkan query pencarian ke dalam *input field* yang disediakan. Setelah itu, pengguna harus menekan tombol 'Cari'. Format pencarian yang kami sediakan adalah <YYYY-MM-DD><spasi><Nama Penyakit>. Kemudian, *frontend* akan mencocokkan query dengan regex. Kemudian, akan dikirimkan body berisi query ke *backend*, kemudian dipanggil query ke database. Setelah didapatkan data, akan dikirim kembali ke *frontend*, kemudian akan dirender. Jika tidak ditemukan, akan menuliskan data tidak ditemukan.

## B. Fitur Fungsional dan Arsitektur Aplikasi Web yang Dibangun

Terdapat beberapa fitur dalam aplikasi web:

1. Menambahkan penyakit baru beserta sekuens DNA ke database
2. Mengetes DNA pengguna dengan penyakit yang diprediksi
3. Mencari hasil tes

Arsitektur yang digunakan dalam web adalah express dan Node.js untuk *backend*, Next.js dan React.js untuk *frontend*. Database disimpan dalam bentuk MySQL di *cloud* menggunakan layanan Azure.

Untuk menyambungkan *backend* dengan database, digunakan Promise untuk mempermudah. Selain itu, pengiriman data dari *backend* ke *frontend* menggunakan axios. Penanganan router dan perpindahan web ditangani oleh method dan fungsi bawaan Next.js. Pada *repository*, folder *backend* dan *frontend* dipisah.

## BAB 4

### IMPLEMENTASI DAN PENGUJIAN

#### A. Spesifikasi Teknis Program

##### A.1 Struktur Data

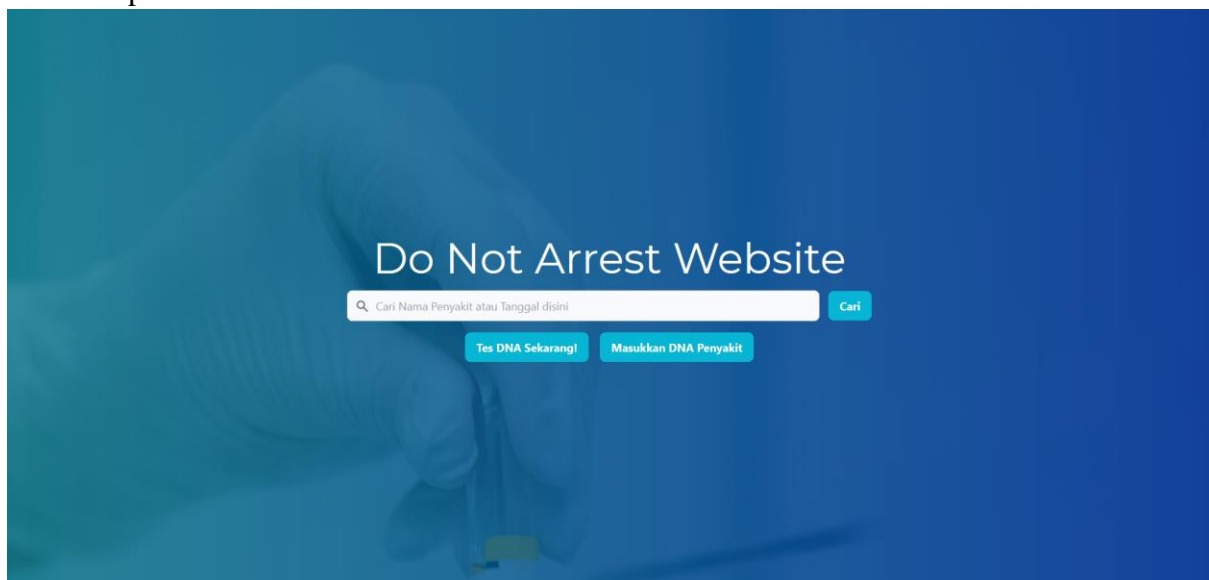
Struktur data yang digunakan dalam aplikasi web adalah string. Kemudian, string tersebut akan dibungkus menjadi objek JSON atau *Javascript Object Notation*. Kedua struktur ini memiliki peran yang sangat penting dalam perpindahan dan flow data web.

##### A.2 Fungsi dan Prosedur

No	Fungsi/Prosedur	Keterangan
1	sanitize(inputdna)	Melakukan sanitasi sekuens DNA dengan mencocokkan terhadap regex, return “” jika tidak valid, return inputDNA jika valid.
2	kmp(body, sequence)	Melakukan pencocokan string antara body dengan sequence menggunakan Knuth-Morris-Pratt algorithm
3	bm(body, sequence)	Melakukan pencocokan string antara body dengan sequence menggunakan Boyer-Moore algorithm
4	levenshtein(body, sequence, nearest)	Melakukan perhitungan tingkat kemiripan DNA
5	query(query, args)	Menghubungkan <i>backend</i> dengan database dan mendapatkan data hasil query

#### B. Penjelasan Tata Cata Penggunaan Program

Halaman pertama yang akan diakses oleh pengguna adalah *homepage* dengan tampilan seperti berikut



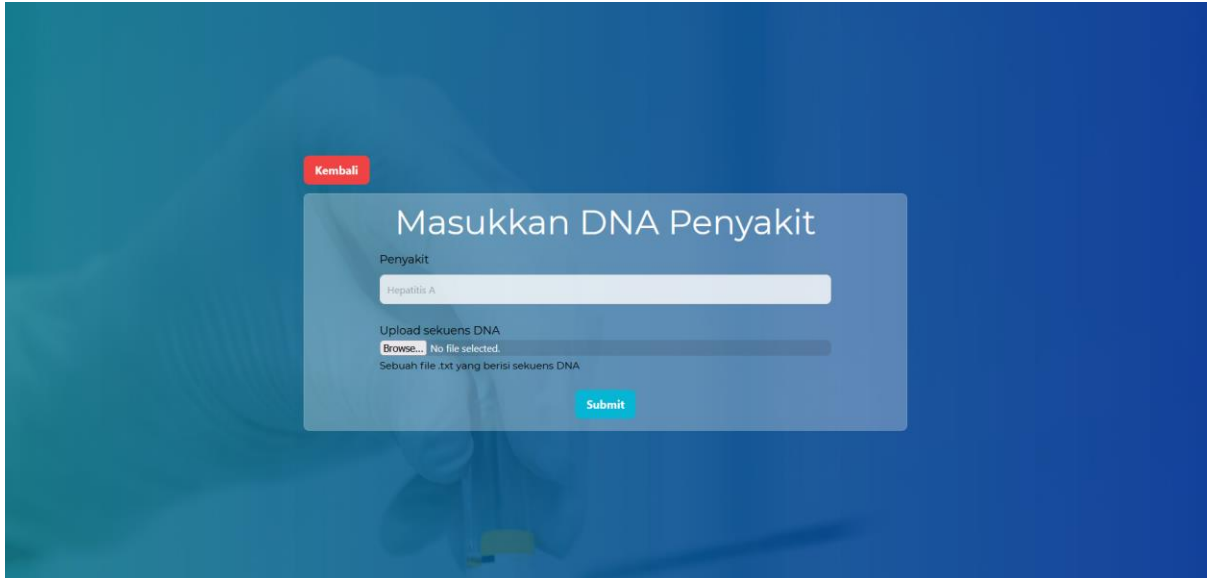
Gambar 2. *Homepage* aplikasi web

##### B.1 Penambahan Penyakit ke Database

1. Pengguna menekan tombol ‘Masukkan DNA Penyakit’ pada *homepage*



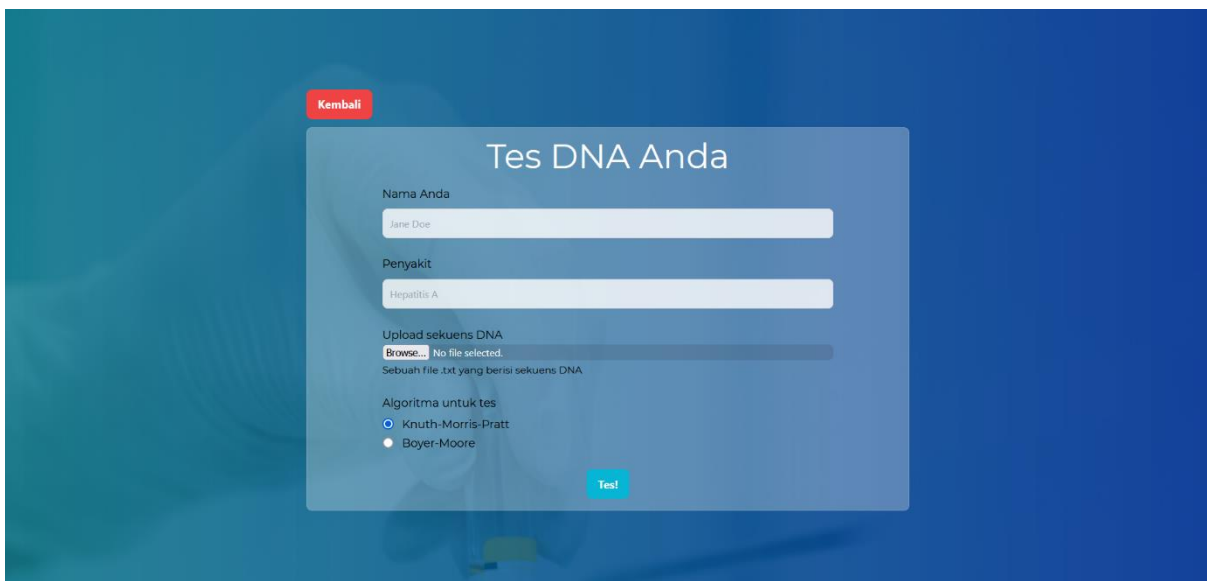
2. Pengguna memasukkan nama penyakit
3. Pengguna memasukkan file yang berisi sekuens DNA
4. Pengguna menekan tombol 'Submit'
5. Jika berhasil, akan muncul konfirmasi berhasil submit
6. Jika gagal, akan muncul konfirmasi gagal submit



Gambar 3. Halaman masukkan DNA Penyakit

## B.2 Tes DNA Pengguna

1. Pengguna menekan tombol 'Tes DNA Sekarang!' pada *homepage*
2. Pengguna memasukkan Nama, Prediksi Penyakit, file .txt yang berisi sekuens DNA, serta pilihan algoritma untuk tes.
3. Pengguna menekan tombol 'Tes!'
4. Jika berhasil tes, akan muncul hasil tes
5. Jika gagal, akan muncul pesan error



Gambar 4. Halaman tes DNA

### B.3 Pencarian Hasil Tes

1. Pengguna memasukkan query di *homepage*
2. Format query adalah <YYYY-MM-DD><spasi><Nama Penyakit>. Pengguna dapat memasukkan tanggal saja, penyakit saja, atau dua-duanya.
3. Pengguna menekan tombol 'Cari'
4. Jika ada hasil, akan muncul hasilnya
5. Jika tidak, akan muncul pesan tidak ditemukan

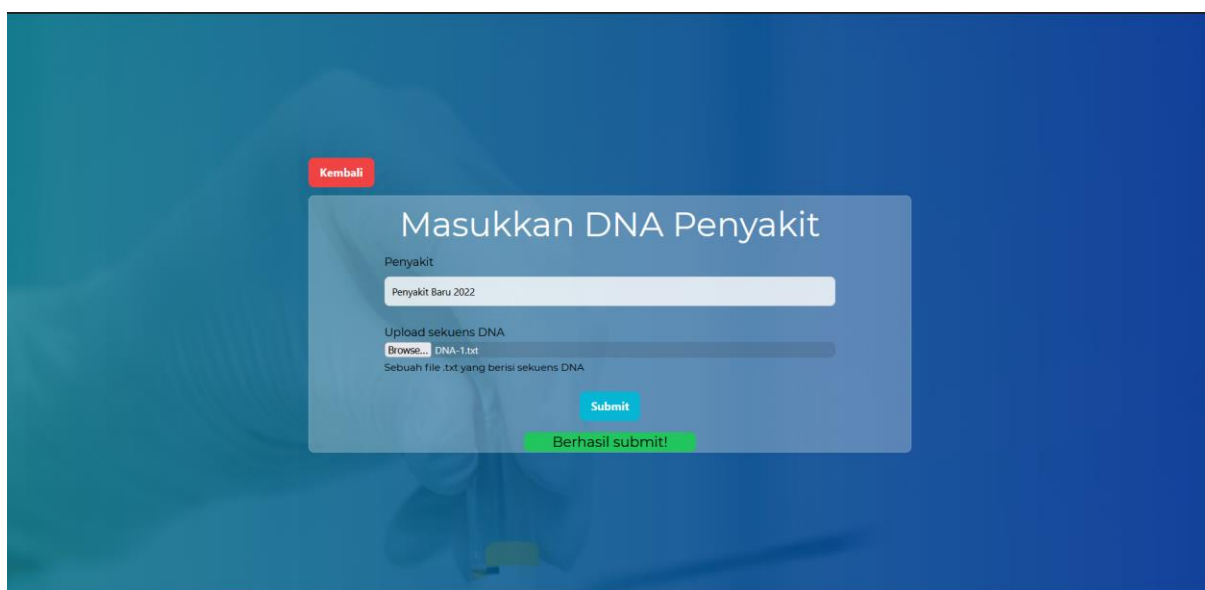


Gambar 5. Halaman pencarian sekaligus hasil tes

## C. Hasil Pengujian

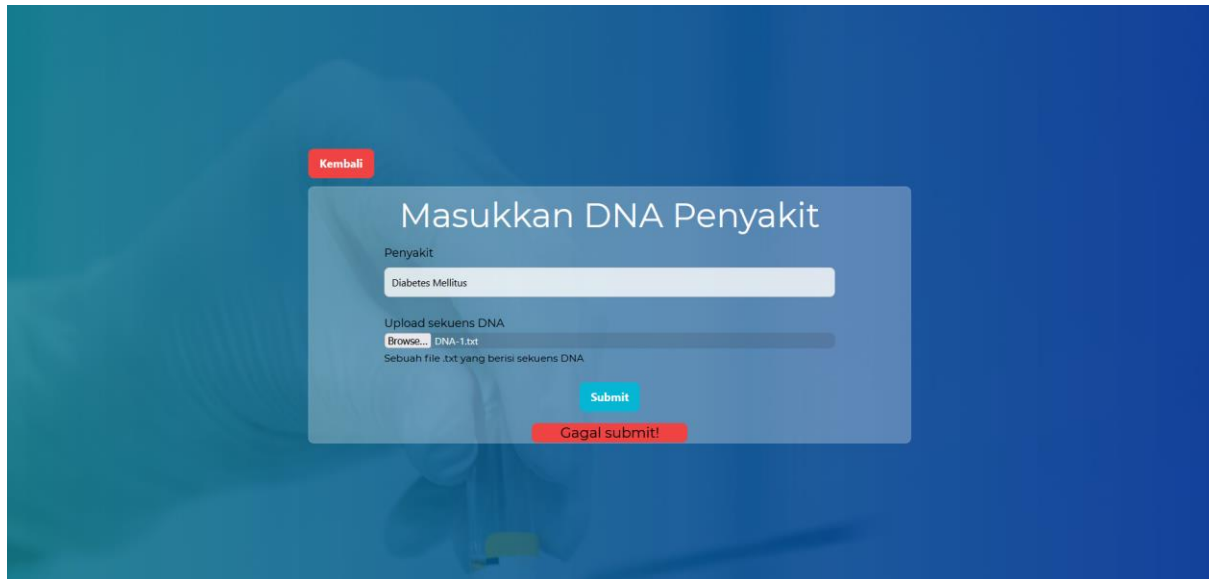
### C.1 Menambahkan Penyakit

1. Berhasil  
DNA Masukan: CACATTATAGGAGATGACATGACA



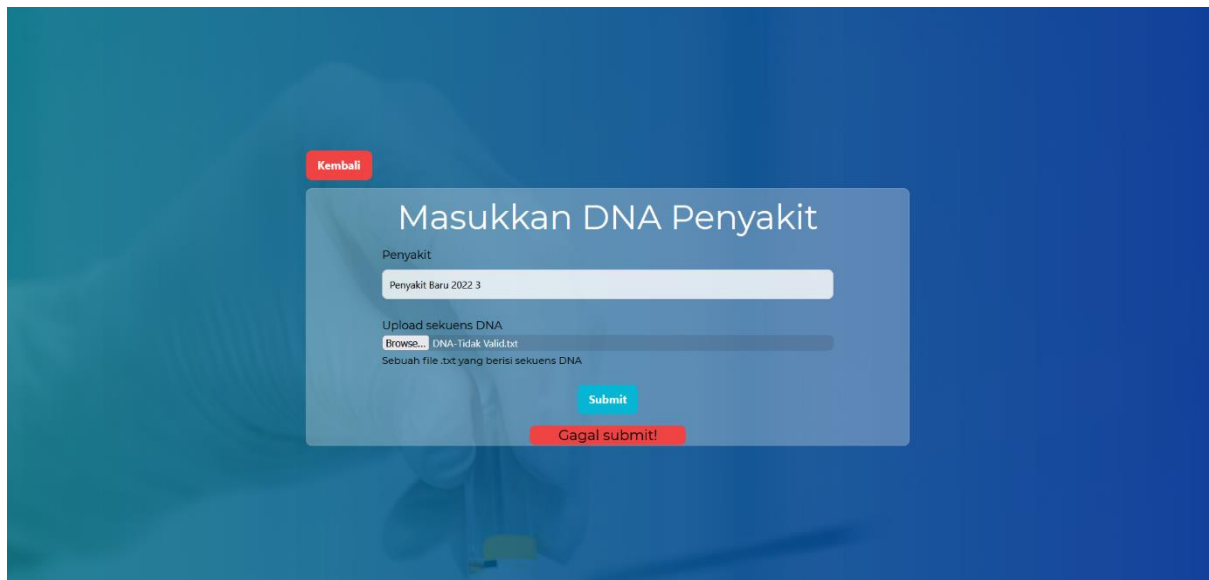
Gambar 6. Halaman berhasil submit penyakit baru

2. Gagal karena nama penyakit sudah ada  
DNA Masukan: CACATTATAGGAGATGACATGACA



Gambar 7. Halaman gagal submit karena sudah ada penyakit di database

3. Gagal karena DNA tidak valid  
DNA Masukan:  
CTGCGACGCCACGATAGTATCAATAAATGAATCTTTGATATGAG  
ATTGCGATCGGTGCAA -  
++TCAGCGAAACATGAGAACAGGTCGGTGTACTAATACGAAAG  
ACTAGAGATCTATAGATA



Gambar 8. Gagal submit karena DNA tidak valid

## C.2 Mengetes DNA

1. Exact Matching

DNA Masukan:

CTCTAATATCCCCTAAGCACATATTGACCTAAGCGCTTTTGTC  
GCCTAGGTTAAATTACGTAGGAGTCGATGGCAGAGTAACGACG  
GGAAGTGTACGTAGGGCTCTCTTCTCGCGTATCAGCTAGCATCA  
GCTACGATCAGCAGCTACGACATCAGCGACTACGAT

DNA Penyakit:

CTCTAATATCCCCTAAGCACATATTGACCTAAGCGCTTTTGTC  
GCCTAGGTTAAATTACGTAGGAGTCGATGGCAGAGTAACGACG  
GGAAGTGTACGTA



Gambar 9. Tes DNA dengan exact match

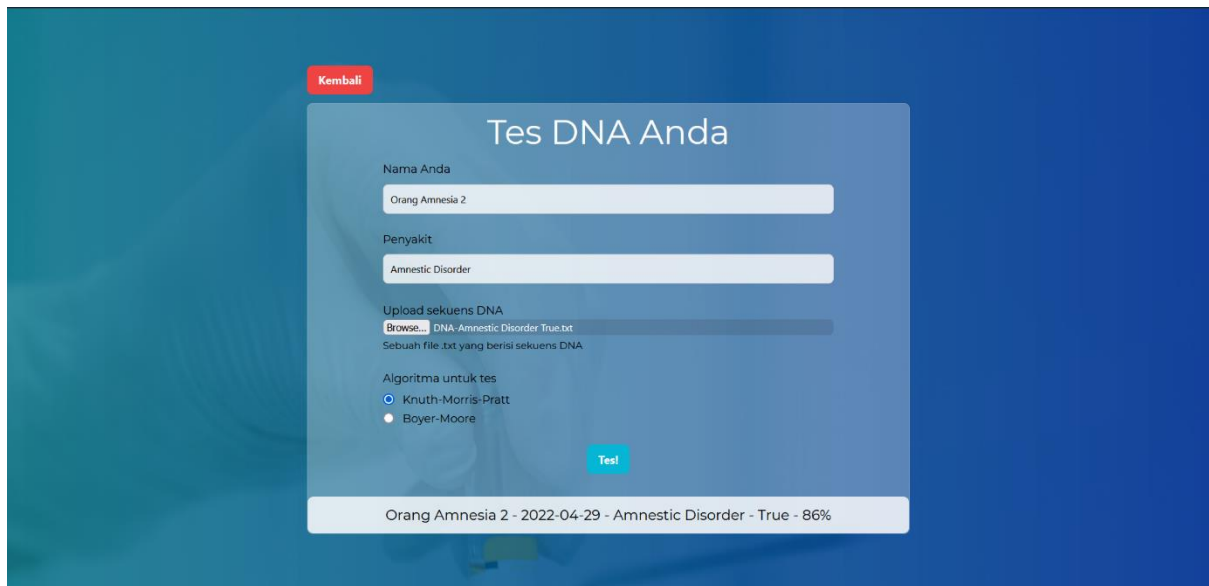
## 2. True, tetapi tidak exact matching

DNA Masukan:

CTCTAATATCCCCTAAGCACATATTGACCTAAGCGCTTTTGTC  
GCCTAGGTTAAATTACGTAGGAGTCGATGGCAGGGGCCAGAGT  
AACGACGGGAAGTGTACGTAGGGCTCTCTTCTCGCGTATCAGCT  
AGCATCAGCTACGATCAGCAGCTACGACATCAGCGACTACGAT

DNA Penyakit:

CTCTAATATCCCCTAAGCACATATTGACCTAAGCGCTTTTGTC  
GCCTAGGTTAAATTACGTAGGAGTCGATGGCAGAGTAACGACG  
GGAAGTGTACGTA



Gambar 10. Tes DNA True tetapi tidak exact

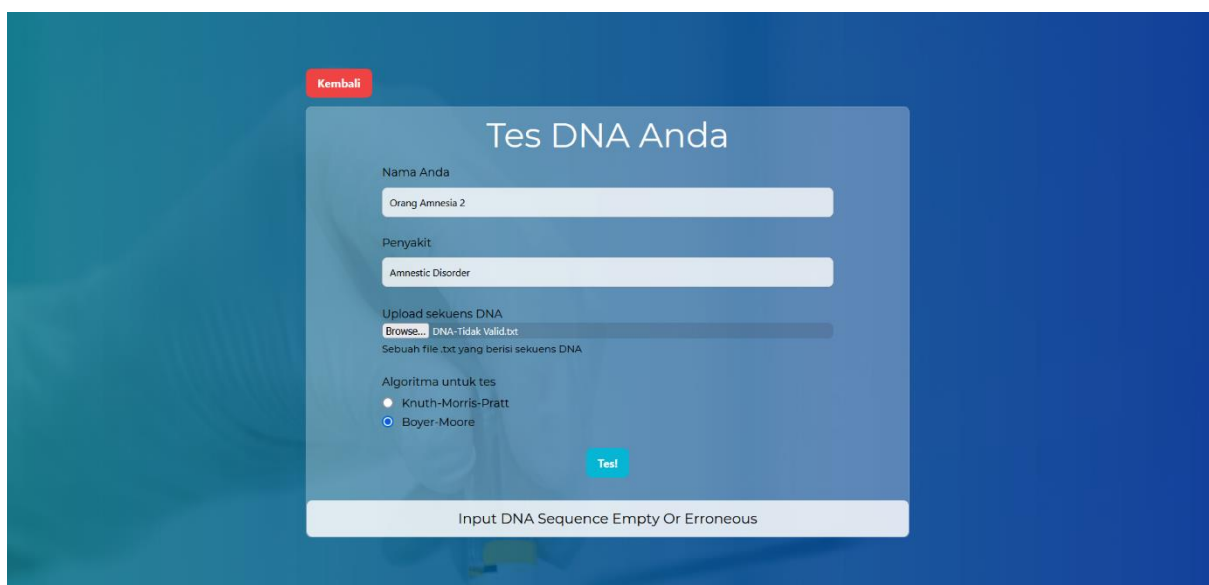
3. False karena tidak valid

DNA Masukan:

CTGCGACGCCACGATAGTATCAATAAATGAATCTTTGATATGAG  
ATTGCGATCGGTGCAA-  
++TCAGCGAAACATGAGAACAGGTCGGTGTACTAATACGAAAG  
ACTAGAGATCTATAGATA

DNA Penyakit:

CTCTAATATCCCACCTAAGCACATATTGACCTAAGCGCTTTTGTC  
GCCTAGGTTAAATTACGTAGGAGTCGATGGCAGAGTAACGACG  
GGAAGTGTACGTA



Gambar 11. DNA Tidak Valid

4. False karena similarity < 80%

DNA Masukan:

GGGGAGCATGCTAGCTAGCTAGCCGAGCTACGTACTGACGATG

CTAGCTAGCTAGCATGCTACGTACGATGCATGCGCGATGCGCCG  
 ATCGGATCGTACGATCGATCGATCGATGCATGCATGCATG  
 CTAGCATGCTGGCTAGCTACGATGCTACGTA

DNA Penyakit:

CTCTAATATCCCATAAGCACATATTGACCTAAGCGCTTTTGTC  
 GCCTAGGTAAATTACGTAGGAGTCGATGGCAGAGTAACGACG  
 GGAAGTGTACGTA

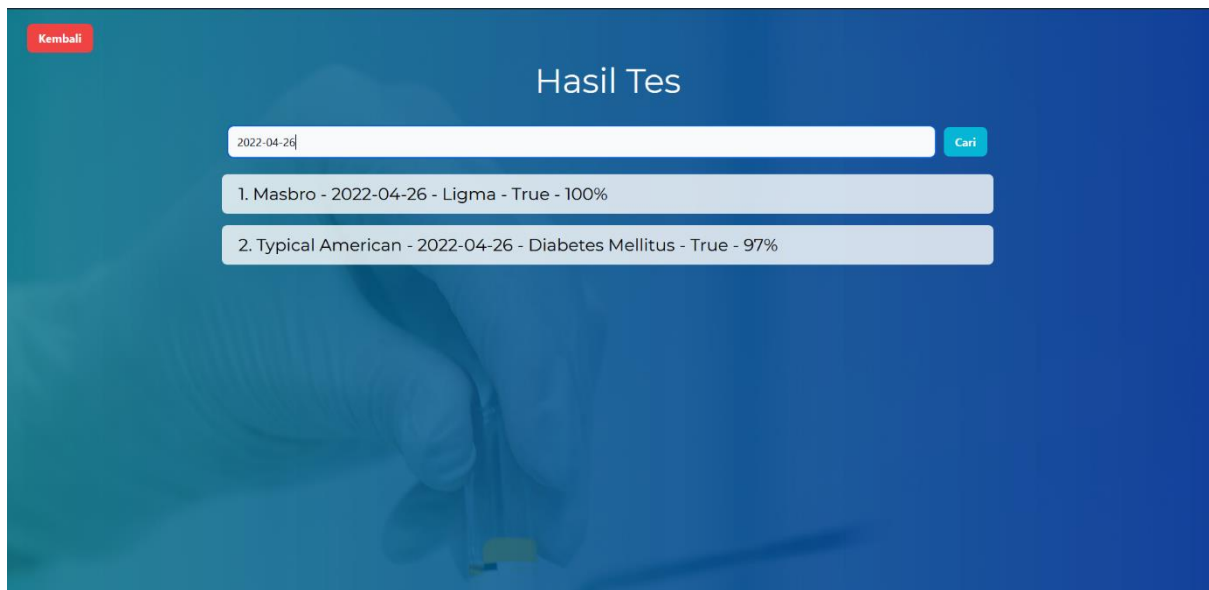
Gambar 12. Tes False karena kurang dari 80%

### C.3 Pencarian Hasil Tes

#### 1. Pencarian berdasarkan nama penyakit

Gambar 13. Pencarian hasil tes berdasarkan nama penyakit

## 2. Pencarian berdasarkan tanggal

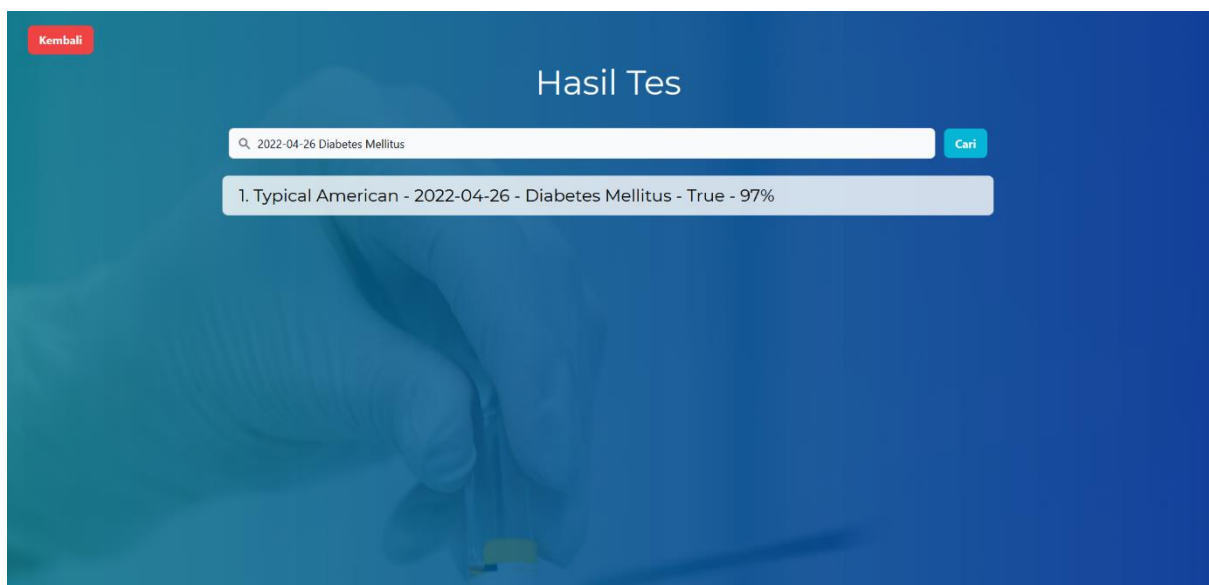


The screenshot shows a web application interface with a dark blue background. At the top left, there is a red button labeled "Kembali". In the center, the title "Hasil Tes" is displayed. Below the title, there is a search bar containing the text "2022-04-26" and a blue button labeled "Cari". Below the search bar, there are two search results displayed in light blue boxes:

- 1. Masbro - 2022-04-26 - Ligma - True - 100%
- 2. Typical American - 2022-04-26 - Diabetes Mellitus - True - 97%

Gambar 14. Pencarian hasil tes berdasarkan tanggal

## 3. Pencarian berdasarkan tanggal dan nama penyakit

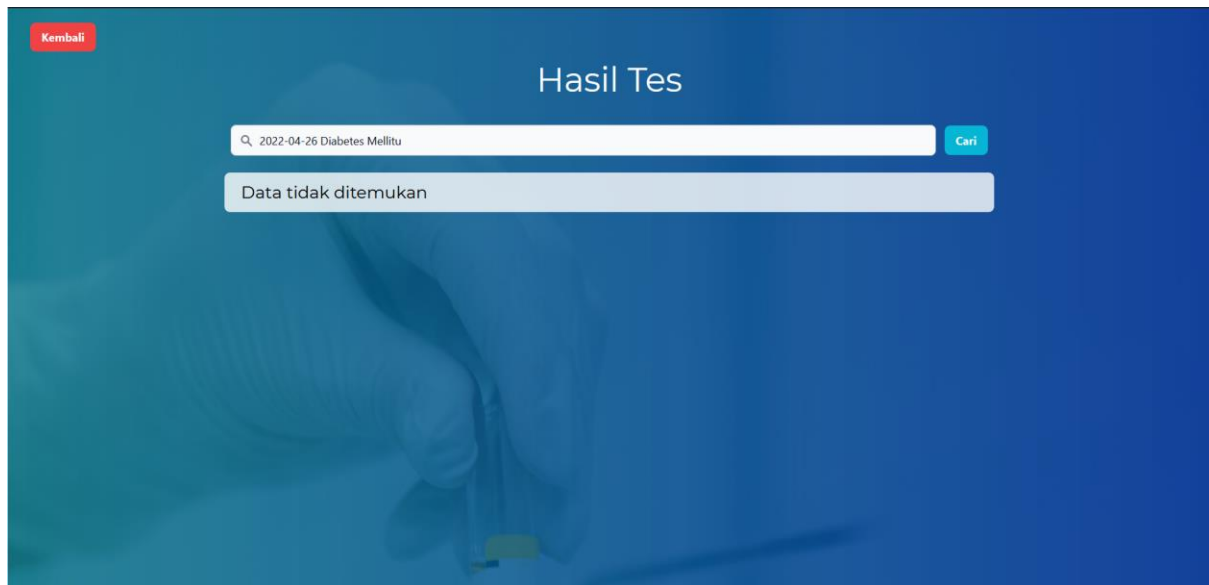


The screenshot shows the same web application interface as Gambar 14. The search bar now contains the text "2022-04-26 Diabetes Mellitus". The search results are filtered to show only one result:

- 1. Typical American - 2022-04-26 - Diabetes Mellitus - True - 97%

Gambar 15. Pencarian berdasarkan tanggal dan penyakit

#### 4. Pencarian tidak menemukan data



Gambar 16. Pencarian tidak menemukan data

#### D. Analisis Hasil Pengujian

Berdasarkan hasil pengujian, aplikasi berhasil menerima *input sequence* DNA dalam bentuk *file*. DNA tersanitasi dengan mengimplementasikan Regex. Aplikasi juga dapat menerima input nama pengguna, DNA-nya, dan nama penyakit yang diuji. DNA yang dimasukkan akan dicocokkan dengan DNA penyakit yang diuji. Aplikasi juga dapat menampilkan urutan hasil prediksi penyakit setelah pengguna memasukkan tanggal prediksi dan nama penyakit. Selain itu, aplikasi juga dapat menghitung tingkat kemiripan DNA pengguna dengan DNA penyakit pada tes DNA.



## BAB 5

### KESIMPULAN DAN SARAN

#### A. Kesimpulan

Algoritma KMP dan BM yang diimplementasikan dapat berfungsi dengan baik untuk mencari apakah ada pola yang dicari dalam string. Pemanfaatan algoritma tersebut untuk mengidentifikasi penyakit yang diderita seseorang berdasarkan kemiripan DNA juga berfungsi baik. Algoritma Levenshtein berguna untuk mencari tahu kemiripan DNA ketika sekuens DNA orang dan DNA penyakit tidak *exact match*. Dengan begitu, kita tetap bisa menentukan apakah orang tersebut menderita penyakit atau tidak dengan tingkat kemiripan di atas 80%. *Framework* yang digunakan baik untuk *frontend* maupun *backend* berfungsi baik untuk implementasi spesifikasi.

#### B. Saran

Untuk tugas selanjutnya, dapat didesain *layout* yang lebih baik dan desain yang lebih bagus. Selain itu, beberapa kode serasa tidak modular dan seharusnya bisa dibuat lebih modular.

#### C. Komentar dan Refleksi

Tugas besar kali ini sangat seru karena bahasa yang digunakan relatif baru bagi kami dan kami sebelumnya jarang membuat aplikasi web. Selain itu kami belajar banyak cara melakukan deploy website bagian *backend* dan *frontend*. Kami merasa bangga dengan karya kami, meskipun tidak sempurna.

## DAFTAR PUSTAKA

- [1] Munir, Rinaldi dan Maulidevi, Nur Ulfa. (2022), Pencocokan String Matching. Diakses online dari <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf> pada 15 April 2022.
- [2] Navarro, G. (2001). A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1), 31–88. <https://doi.org/10.1145/375360.375365>

## LAMPIRAN

Link repository github: [https://github.com/Adityapnn811/Tubes3\\_13519210](https://github.com/Adityapnn811/Tubes3_13519210)

Link REST API: <https://do-not-arrest.herokuapp.com/>

Link hasil deploy: <https://do-not-arrest.vercel.app/>