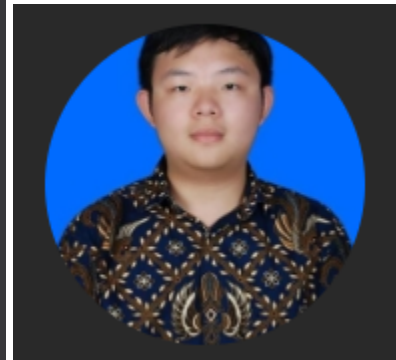
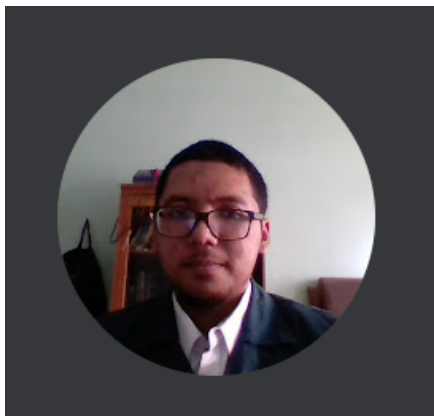


# **LAPORAN TUGAS BESAR 3**

**IF2211 Strategi Algoritma**

## **Penerapan String Matching dan Regular Expression dalam DNA Pattern Matching**



Disusun oleh:

Kelompok 50

1. Ignasius Ferry Priguna (13520126)
2. Bayu Samudra (13520128)
3. Muhammad Gilang Ramadhan (13520137)

**PROGRAM STUDI TEKNIK INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
BANDUNG  
2022**

# DAFTAR ISI

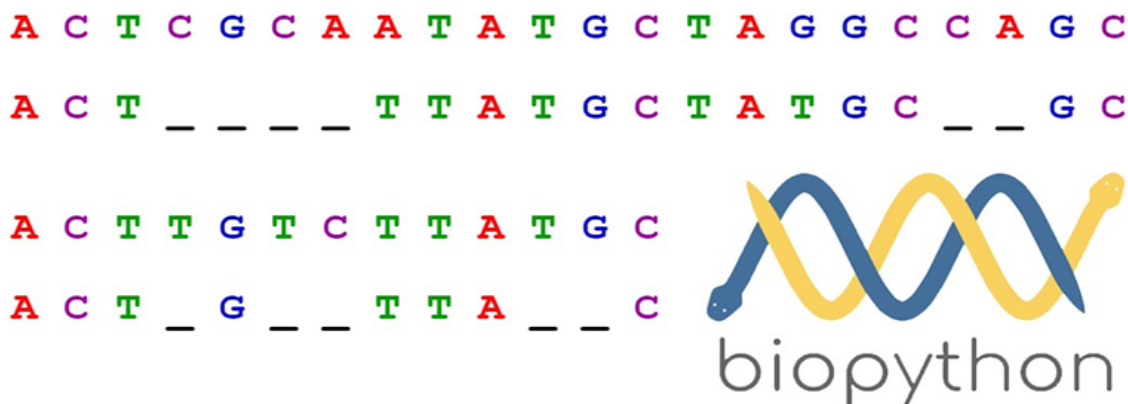
<b>DAFTAR ISI</b>	<b>2</b>
<b>DESKRIPSI TUGAS</b>	<b>3</b>
1.1 Latar Belakang	3
1.2 Deskripsi Tugas	4
1.3 Fitur-Fitur Aplikasi	5
1.4 Spesifikasi Program	10
<b>LANDASAN TEORI</b>	<b>11</b>
2.1 Algoritma Knuth-Morris-Pratt	11
2.2 Algoritma Boyer-Moore	12
2.4 Gambaran Umum Aplikasi Web yang Dibangun	14
<b>ANALISIS PEMECAHAN MASALAH</b>	<b>15</b>
3.1 Langkah Penyelesaian Masalah	15
3.2 Fitur Fungsional dan Arsitektur Aplikasi Web	16
<b>IMPLEMENTASI DAN PENGUJIAN</b>	<b>19</b>
4.1 Spesifikasi Teknis Program	19
<b>KESIMPULAN DAN SARAN</b>	<b>23</b>
5.1 Kesimpulan	23
5.2 Saran	23
<b>DAFTAR PUSTAKA</b>	<b>24</b>

# BAB I

## DESKRIPSI TUGAS

### 1.1 Latar Belakang

Manusia umumnya memiliki 46 kromosom di dalam setiap selnya. Kromosom-kromosom tersebut tersusun dari DNA (deoxyribonucleic acid) atau asam deoksiribonukleat. DNA tersusun atas dua zat basa purin, yaitu Adenin (A) dan Guanin (G), serta dua zat basa pirimidin, yaitu sitosin (C) dan timin (T). Masing-masing purin akan berikatan dengan satu pirimidin. DNA merupakan materi genetik yang menentukan sifat dan karakteristik seseorang, seperti warna kulit, mata, rambut, dan bentuk wajah. Ketika seseorang memiliki kelainan genetik atau DNA, misalnya karena penyakit keturunan atau karena faktor lainnya, ia bisa mengalami penyakit tertentu. Oleh karena itu, tes DNA penting untuk dilakukan untuk mengetahui struktur genetik di dalam tubuh seseorang serta mendeteksi kelainan genetik. Ada berbagai jenis tes DNA yang dapat dilakukan, seperti uji pra implantasi, uji pra kelahiran, uji pembawa atau carrier testing, uji forensik, dan DNA sequence analysis.



**Gambar 1.1** *Ilustrasi Sekuens DNA*

<https://towardsdatascience.com/pairwise-sequence-alignment-using-biopython->

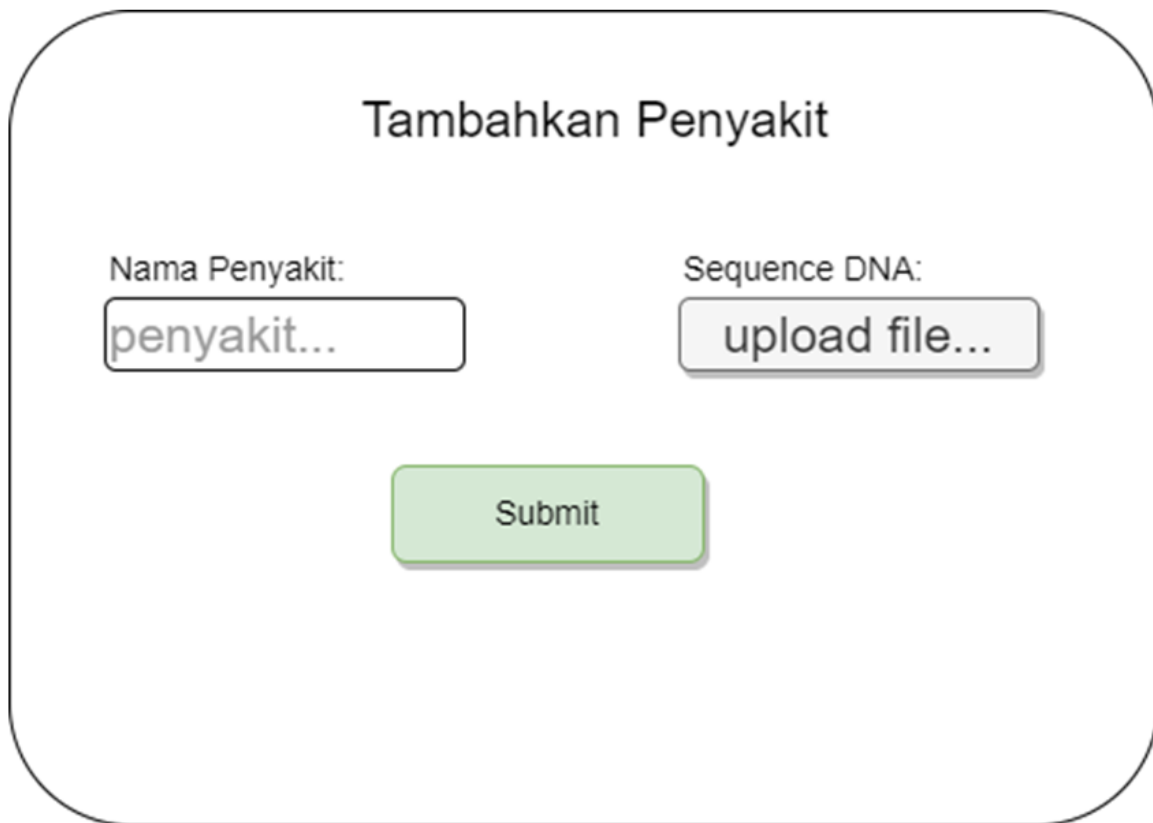
Salah satu jenis tes DNA yang sangat berkaitan dengan dunia bioinformatika adalah DNA sequence analysis. DNA sequence analysis adalah sebuah cara yang dapat digunakan untuk memprediksi berbagai macam penyakit yang tersimpan pada database berdasarkan urutan sekuens DNA-nya. Sebuah sekuens DNA adalah suatu representasi string of nucleotides yang disimpan pada suatu rantai DNA, sebagai contoh: ATTCGTAAGTAAAGTTA. Teknik pattern matching memegang peranan penting untuk dapat menganalisis sekuens DNA yang sangat panjang dalam waktu singkat. Oleh karena itu, mahasiswa Teknik Informatika berniat untuk membuat suatu aplikasi web berupa DNA Sequence Matching yang menerapkan algoritma String Matching dan Regular Expression untuk membantu penyedia jasa kesehatan dalam memprediksi penyakit pasien. Hasil prediksi juga dapat ditampilkan dalam tabel dan dilengkapi dengan kolom pencarian untuk membantu admin dalam melakukan filtering dan pencarian.

## **1.2 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.

### 1.3 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).
  - c. Contoh *input* penyakit:



The image shows a web form titled "Tambahkan Penyakit" (Add Disease). It contains two input fields: "Nama Penyakit:" (Disease Name) with a placeholder text "penyakit..." and "Sequence DNA:" with a placeholder text "upload file...". Below these fields is a green "Submit" button.

**Gambar 1.2** *Ilustrasi Input Penyakit*

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*.
- f. Contoh tampilan web:

### Tes DNA

Nama Pengguna:

Sequence DNA:

Prediksi Penyakit:

---

#### Hasil Tes

<Tanggal> - <pengguna> - <penyakit> - <True/False>

### Gambar 1.3 Ilustrasi Prediksi

3. Aplikasi memiliki halaman yang menampilkan urutan hasil prediksi dengan kolom pencarian di dalamnya. Kolom pencarian bekerja sebagai filter dalam menampilkan hasil.
- Kolom pencarian dapat menerima masukan dengan struktur: <tanggal\_prediksi><spasi><nama\_penyakit>, contoh “13 April 2022 HIV”. **Format penanggalan dibeaskan**, jika bisa menerima >1 format lebih baik.
  - Kolom pencarian dapat menerima masukan hanya tanggal ataupun hanya nama penyakit. Fitur ini diimplementasikan menggunakan **regex**.
  - Contoh ilustrasi
    - Masukan tanggal dan nama penyakit

The screenshot shows a web application interface. At the top, there is a search input field containing the text "13 April 2022 HIV". Below the input field, there is a list of six search results, each displayed in a separate box. The results are numbered 1 through 6 and show a date, a name, and a prediction result (True or False).

No	Search Input	Result
1	13 April 2022 - Fulan - HIV - True.	True
2	13 April 2022 - Kamal - HIV - False.	False
3	13 April 2022 - Entah - HIV - False.	False
4	13 April 2022 - Jamal - HIV - True.	True
5	13 April 2022 - Yubai - HIV - True.	True
6	13 April 2022 - Hika - HIV - False.	False

Gambar 1.4 Ilustrasi Interaksi 1

2) Masukan hanya tanggal

13 April 2022

1. 13 April 2022 - Fulan - Diabetes - True.
2. 13 April 2022 - Kamal - Sinusitis - False.
3. 13 April 2022 - Entah - Down Syndrome - False.
4. 13 April 2022 - Jamal - Polio - True.
5. 13 April 2022 - Yubai - TBC - True.
6. 13 April 2022 - Hika - Hepatitis A - False.

**Gambar 1.5** *Ilustrasi Interaksi 2*

3) Masukan hanya nama penyakit

HIV

1. 13 April 2022 - Fulan - HIV - True.
2. 14 April 2022 - Kamal - HIV - False.
3. 15 April 2022 - Entah - HIV - False.
4. 16 April 2022 - Jamal - HIV - True.
5. 17 April 2022 - Yubai - HIV - True.
6. 18 April 2022 - Hika - HIV - False.

**Gambar 1.6** *Ilustrasi Interaksi 3*



4. **(Bonus)** Menghitung tingkat kemiripan DNA pengguna dengan DNA penyakit pada tes DNA
- a. Ketika melakukan tes DNA, terdapat persentase kemiripan DNA dalam hasil tes. **Contoh hasil tes: 1 April 2022 - Mhs IF - HIV - 75% - False**
  - b. Perhitungan tingkat kemiripan dapat dilakukan dengan menggunakan Hamming distance, Levenshtein distance, LCS, atau algoritma lainnya (dapat dijelaskan dalam laporan).
  - c. Tingkat kemiripan DNA dengan nilai lebih dari atau sama dengan 80% dikategorikan sebagai **True**. Perlu diperhatikan mengimplementasikan atau tidak mengimplementasikan bonus ini tetap dilakukan pengecekan *string matching* terlebih dahulu.
  - d. Contoh tampilan:

### Tes DNA

Nama Pengguna:  
<pengguna>

Sequence DNA:  
upload file...

Prediksi Penyakit:  
<penyakit>

Submit

---

Hasil Tes

<Tanggal> - <pengguna> - <penyakit> - <similarity> - <True/False>

**Gambar 1.7** *Ilustrasi Bonus*

## 1.4 Spesifikasi Program

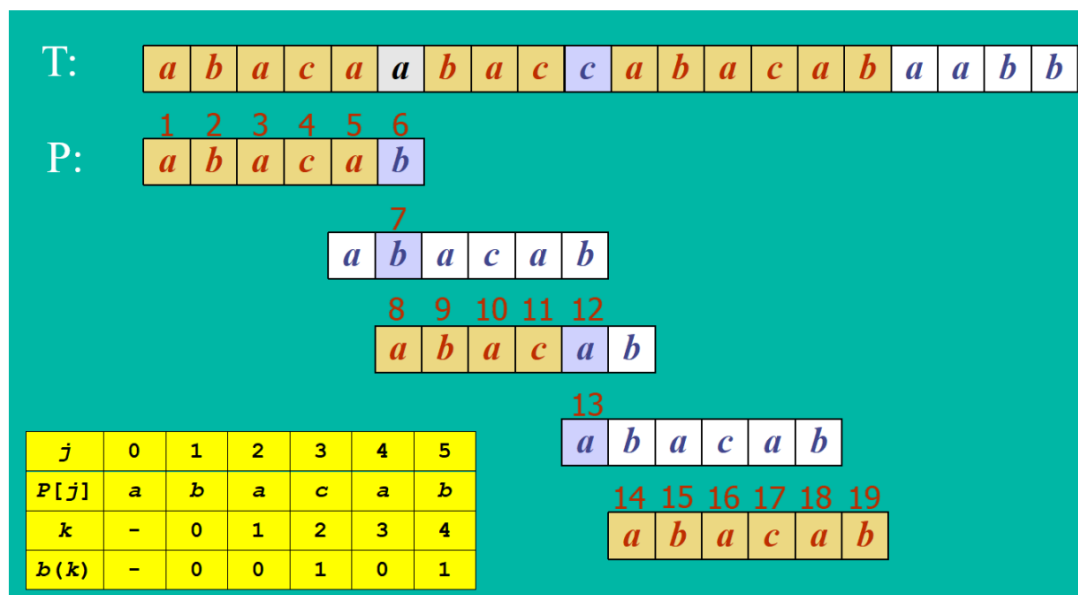
1. Aplikasi berbasis website dengan pembagian Frontend dan Backend yang jelas.
2. Implementasi Backend **wajib** menggunakan Node.js / Golang, sedangkan Frontend **disarankan** untuk menggunakan React / Next.js / Vue / Angular. Lihat referensi untuk selengkapnya.
3. Penyimpanan data **wajib** menggunakan basis data (MySQL / PostgreSQL / MongoDB).
4. Algoritma pencocokan string (KMP dan Boyer-Moore) **wajib** diimplementasikan pada sisi Backend aplikasi.
5. Informasi yang **wajib** disimpan pada basis data:
  - a. Jenis Penyakit
    - Nama penyakit
    - Rantai DNA penyusun
  - b. Hasil Prediksi
    - Tanggal prediksi
    - Nama pasien
    - Penyakit prediksi
    - Status terprediksi
6. Jika mengerjakan bonus tingkat kemiripan DNA, simpan hasil tingkat kemiripan tersebut pada basis data.

## BAB II

### LANDASAN TEORI

#### 2.1 Algoritma Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt (KMP) adalah algoritma pencocokan pola yang merupakan modifikasi dari algoritma brute-force. Jika pada algoritma brute-force pergeseran dilakukan sebanyak satu posisi setiap ditemukan kesalahan, algoritma KMP bisa melakukan pergeseran posisi lebih banyak sehingga jumlah karakter yang dicek akan lebih sedikit. Jika ditemukan ketidakcocokan antara teks dan pola P pada  $P[j]$ , jumlah pergeseran posisi pola terbesar yang dapat dilakukan untuk menghindari perbandingan yang sia-sia adalah ukuran prefix  $P[0 \dots j-1]$  yang juga merupakan suffix dari  $P[1 \dots j-1]$ . Untuk memperoleh nilai tersebut untuk setiap posisi di pola, dilakukan *preprocessing* dengan perhitungan fungsi pinggiran KMP atau *border function*  $b(k)$ . Setelah fungsi pinggiran dihitung, pencarian string dimulai dengan pergeseran posisi pada saat terdapat ketidakcocokan mengikuti fungsi tersebut. Berikut adalah contoh pencocokan pola dengan algoritma KMP



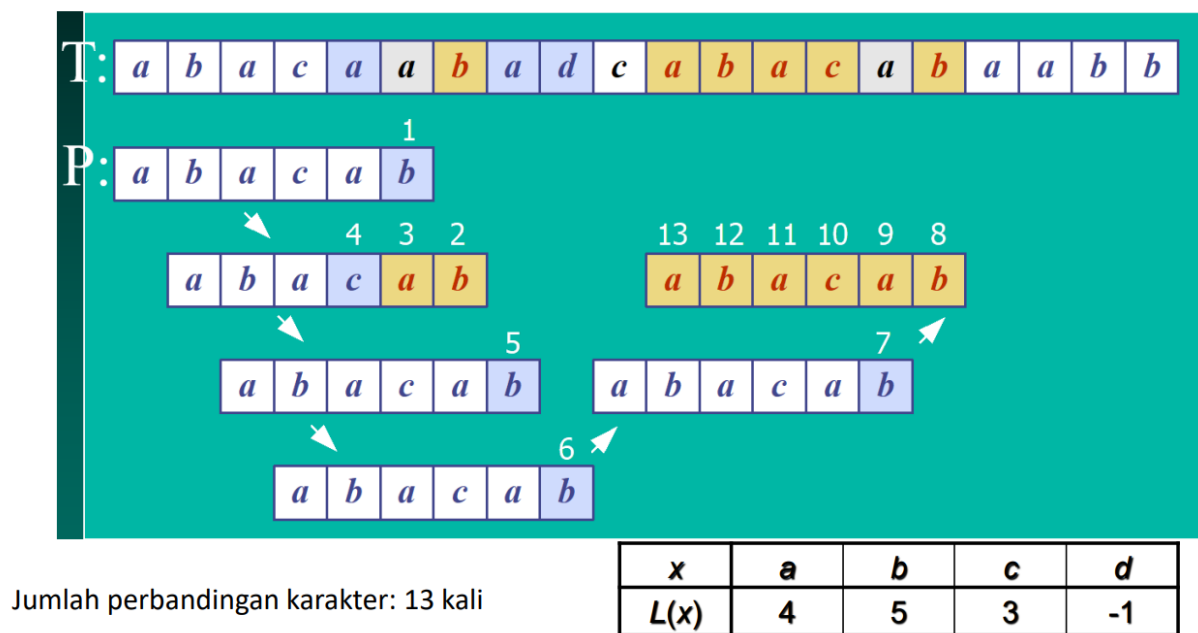
Gambar 2.1 Contoh Pencocokan Pola dengan Algoritma KMP

## 2.2 Algoritma Boyer-Moore

Algoritma Boyer-Moore adalah algoritma pencocokan pola yang menggabungkan 2 teknik, yaitu *looking-glass technique* dan *character-jump technique*. Pada *looking-glass technique*, suatu pola P di T dicocokkan mulai dari akhir P secara mundur. Pada *character-jump technique* dilakukan 1 diantara 3 hal berikut bila ditemukan ketidakcocokan karakter:

1. Jika pola P mengandung karakter yang menyebabkan ketidakcocokan di teks tersebut, geser P ke kanan sehingga karakter di teks tadi sejajar dengan kemunculan terakhir karakter yang sama pada pola P.
2. Jika pola P mengandung karakter yang menyebabkan ketidakcocokan di teks tersebut, tapi pergeseran ke kanan tidak mungkin menghasilkan kondisi sebelumnya, geser P sebanyak 1 karakter
3. Jika pola P tidak memenuhi kondisi 1 dan 2, pola P digeser sehingga karakter awal pola sejajar dengan satu karakter setelah karakter pada teks yang menimbulkan ketidakcocokan.

Berikut adalah contoh pencocokan pola dengan algoritma Boyer-Moore



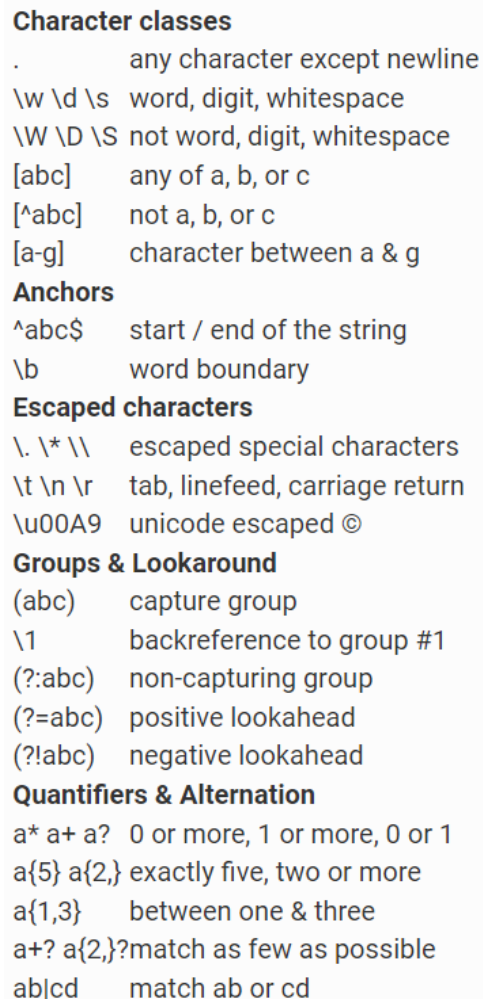
**Gambar 2.2** Contoh Pencocokan Pola dengan Algoritma Boyer-Moore

Sama seperti di algoritma KMP, algoritma Boyer-Moore juga melakukan *preprocessing* sebelum pencocokan dilakukan. *Last occurrence function* L() dihitung untuk setiap alfabet.

Algoritma ini cepat untuk jumlah alfabet yang besar, namun lambat untuk jumlah alfabet yang kecil seperti bilangan biner.

## 2.3 Regular Expression

*Regular Expression* (RE) adalah sebuah notasi yang dapat digunakan untuk mendeskripsikan pola dari kata yang ingin dicari. Konsep tentang regex pertamakali muncul di tahun 1951, ketika seorang ilmunan matematikan bernama Stephen Cole Kleene memformulasikan definisi tentang bahasa formal. Dalam dunia pemrograman, RE dimanfaatkan untuk berbagai kebutuhan seperti validasi data, pencarian, fitur *find and replace*, dan lain-lain. Berikut ini adalah beberapa RE yang sering dipakai



**Character classes**

- `.` any character except newline
- `\w \d \s` word, digit, whitespace
- `\W \D \S` not word, digit, whitespace
- `[abc]` any of a, b, or c
- `[^abc]` not a, b, or c
- `[a-g]` character between a & g

**Anchors**

- `^abc$` start / end of the string
- `\b` word boundary

**Escaped characters**

- `\. \* \\` escaped special characters
- `\t \n \r` tab, linefeed, carriage return
- `\u00A9` unicode escaped ©

**Groups & Lookaround**

- `(abc)` capture group
- `\1` backreference to group #1
- `(?:abc)` non-capturing group
- `(?=abc)` positive lookahead
- `(?!abc)` negative lookahead

**Quantifiers & Alternation**

- `a* a+ a?` 0 or more, 1 or more, 0 or 1
- `a{5} a{2,}` exactly five, two or more
- `a{1,3}` between one & three
- `a+? a{2,}? match as few as possible`
- `ab|cd` match ab or cd

**Gambar 2.3** Beberapa *Regular Expression* yang Sering Dipakai

## 2.4 Gambaran Umum Aplikasi Web yang Dibangun

Aplikasi Web yang dibangun untuk tugas ini adalah aplikasi web pencocokan pola DNA. Pada aplikasi ini, pengguna dapat mendaftarkan *sequence* DNA suatu penyakit. Dari *sequence* DNA penyakit yang telah didaftarkan tersebut, dapat dilakukan tes kecocokan dengan *sequence* DNA seseorang. Hasil tes akan disimpan dalam sebuah basis data yang datanya dapat diakses berdasarkan tanggal tes, nama penyakit yang dites, atau pun gabungan keduanya.

## BAB III

### ANALISIS PEMECAHAN MASALAH

#### 3.1 Langkah Penyelesaian Masalah

##### a. Fitur *Input* Penyakit Baru

1. Menu input penyakit baru ditampilkan ke pengguna sehingga pengguna dapat melakukan pengisian nama penyakit dan file sequence DNA.
2. Saat pengguna menekan tombol submit, nama dan file yang diinput akan dikirimkan ke backend.
3. Di backend, dilakukan pengecekan terhadap kelengkapan input. Jika nama atau file tidak ada, maka proses di backend akan dihentikan dan dikembalikan status gagal.
4. Jika input lengkap, dilakukan validasi terhadap file menggunakan regex untuk memastikan bahwa isinya hanya huruf ACGT tanpa spasi. Jika validasi gagal, proses backend dihentikan dan dikembalikan status gagal.
5. Jika file valid, data penyakit akan dicoba ditambahkan ke basis data. Jika terjadi error di basis data, akan dikembalikan status gagal. Jika penambahan berhasil, akan dikembalikan status berhasil dan ditampilkan bahwa input penyakit berhasil di frontend.

##### b. Fitur Prediksi Penyakit berdasarkan *Sequence* DNA

1. Menu prediksi penyakit baru ditampilkan ke pengguna sehingga pengguna dapat melakukan input sequence DNA dan penyakit yang akan diprediksi
2. Setelah dilakukan input pengguna dapat melakukan proses prediksi tersebut dengan cara menekan tombol submit.
3. Pertama-tama akan dilakukan validasi terhadap input dari pengguna tersebut apakah sudah benar apa belum di frontendnya.
4. Setelah valid, input dari user akan diproses dari frontend ke backend. Kemudian, dilakukan pengecekan terlebih dahulu terhadap penyakit yang diinput oleh user apakah ada di database.
5. Setelah itu, barulah dilakukan pengecekan Pattern Matching terhadap input sequence DNA dari user dan sequence DNA dari penyakit tersebut.

6. Setelah melakukan proses pengecekan pada sequence DNA dari inputan user dengan sequence DNA dari penyakit yang akan diprediksi, maka akan ditampilkan kepada user yaitu hasilnya di layar. Adapun tampilan hasil dari proses tersebut ialah <Tanggal> - <Nama pengguna> - <Nama penyakit> - <Similarity> - <True/False>.

c. Fitur Kolom Pencarian Hasil Prediksi

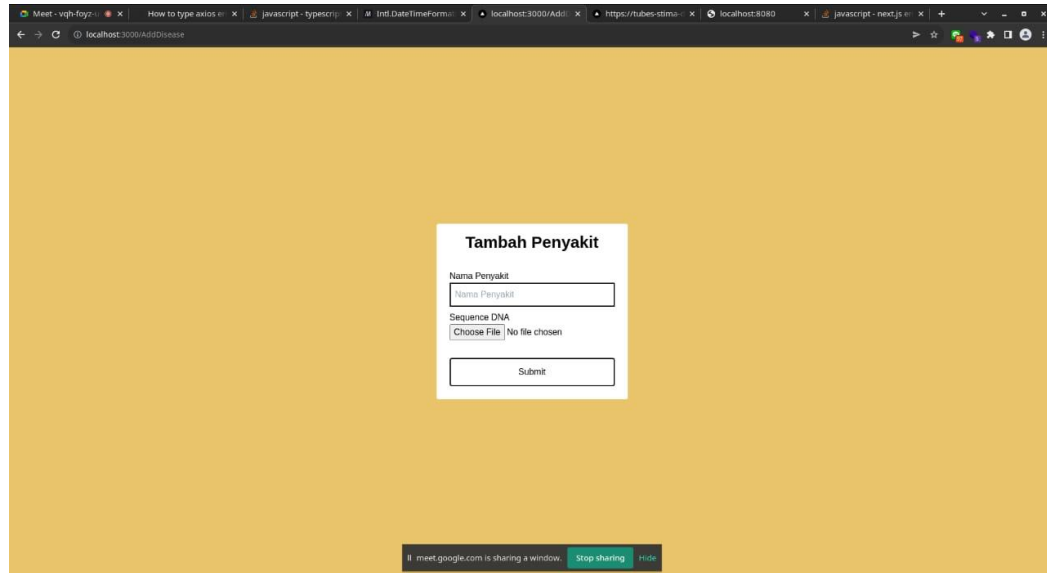
1. Menu kolom pencarian hasil prediksi ditampilkan sehingga pengguna dapat melakukan input syarat pencarian.
2. Saat pengguna menekan tombol submit, syarat pencarian akan dikirim ke backend untuk diproses.
3. Di backend, dilakukan pengecekan terhadap input syarat pencarian. Jika input kosong, akan dikembalikan status gagal dan pemrosesan di backend dihentikan.
4. Input syarat pencarian dicocokkan menggunakan regex dengan pola <Tanggal> <Penyakit>, <Tanggal>, dan <Penyakit>. Jika tidak ada yang sesuai, akan dikembalikan status gagal dan pemrosesan di backend dihentikan.
5. Jika ada regex yang memenuhi, input syarat pencarian menjadi filter dari data hasil prediksi yang akan ditampilkan. Data diambil dari basis data berdasarkan filter tersebut.
6. Data yang telah difilter dikirimkan ke frontend dan ditampilkan ke layar pengguna.

### 3.2 Fitur Fungsional dan Arsitektur Aplikasi Web

Aplikasi web ini dibagi menjadi tiga bagian utama, yaitu frontend, backend, dan basis data. Bagian frontend bertugas untuk menunjukkan antarmuka bagi pengguna. Bagian backend bertugas untuk melakukan validasi terhadap masukan, mengkalkulasi kecocokan pola DNA yang dimasukkan oleh pengguna, serta memberi perintah ke basis data. Bagian basis data bertugas untuk menampung data penyakit serta riwayat pengecekan pengguna. Berikut adalah tampilan fitur yang tersedia di aplikasi web ini

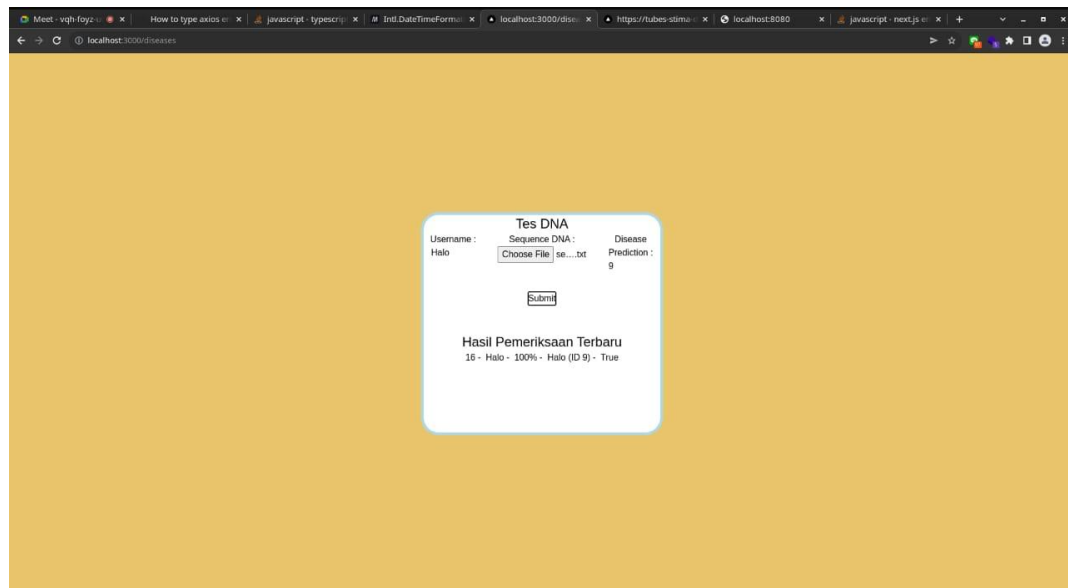
1. Tampilan fitur penambahan penyakit





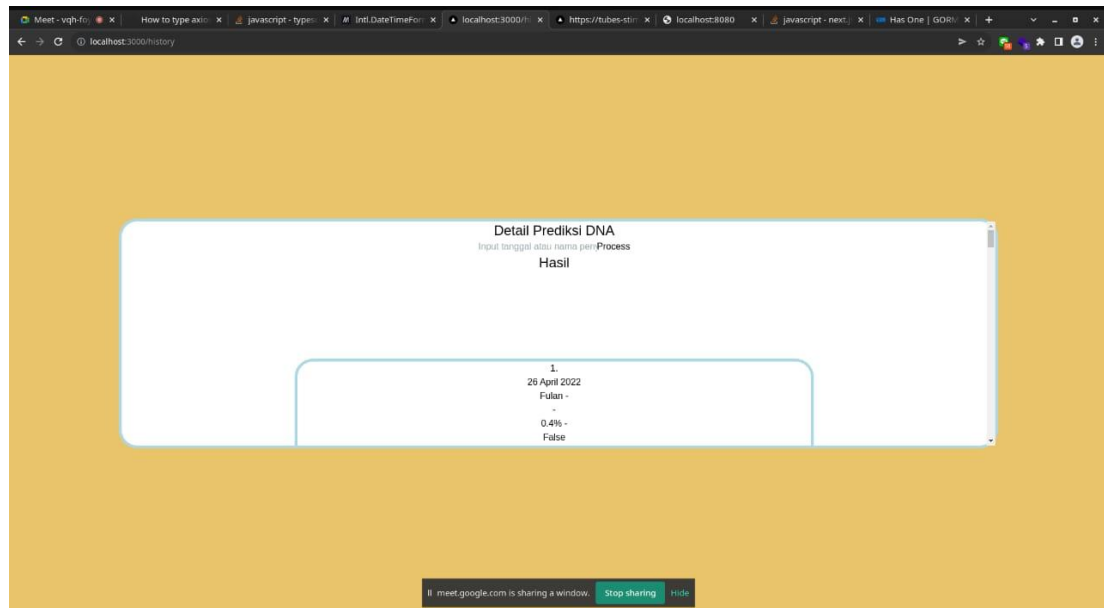
**Gambar 3.2.1** Tampilan fitur tambah penyakit

## 2. Tampilan fitur prediksi penyakit berdasarkan sequence DNA



**Gambar 3.2.2** Tampilan fitur prediksi penyakit

## 3. Tampilan fitur pencarian hasil prediksi



**Gambar 3.2.3** Tampilan Pencarian detail prediksi

## BAB IV

### IMPLEMENTASI DAN PENGUJIAN

#### 4.1 Spesifikasi Teknis Program

Fungsi utama yang digunakan untuk mendukung fungsionalitas program ini terletak di backend, tepatnya di `/controller/validation.go` dan di `/lib`. File `validation.go` berisi fungsi `FileValidation` yang berfungsi untuk memastikan file hanya berisi huruf ACTG tanpa spasi menggunakan regex. Fungsi tersebut mengembalikan string yang kosong jika validasi gagal dan isi file jika berhasil serta boolean yang menyatakan berhasil atau gagalnya validasi.

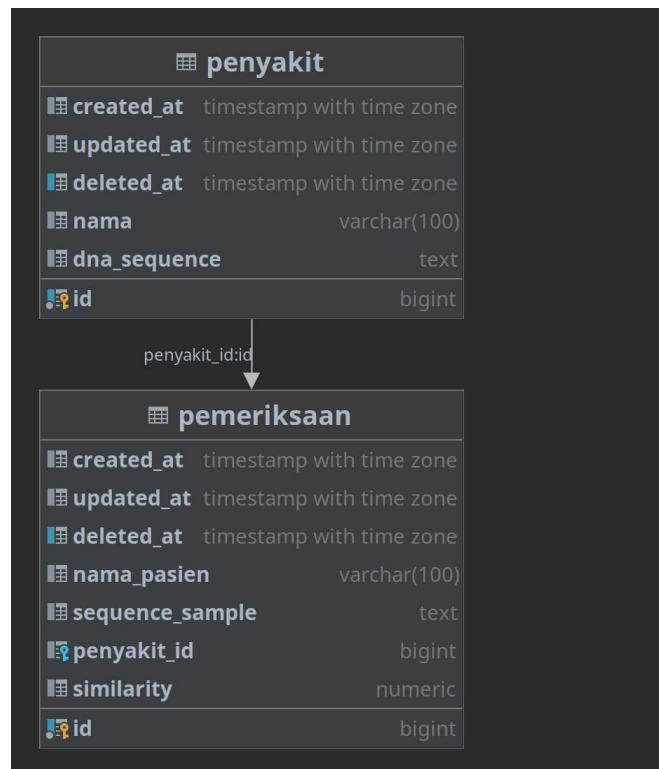
File yang berada di dalam folder `lib` berisi algoritma pencarian pola dan fungsi-fungsi pembantunya serta fungsi pengecekan regex untuk input syarat pencarian. Fungsi `BoyerMoore` digunakan untuk melakukan pencarian suatu pola string dalam teks menggunakan algoritma `BoyerMoore`. Jika fungsi menemukan kecocokan, akan dikembalikan posisi ditemukannya kecocokan sedangkan jika tidak akan mengembalikan nilai `-1`. Fungsi `BoyerMoore` memanfaatkan fungsi pembantu `BuildLast` yang disimpan di `util.go` untuk menyimpan posisi kemunculan terakhir setiap alfabet dalam pola. Struktur data yang digunakan untuk fungsi pembantu ini adalah `map`.

Fungsi `KMP` digunakan untuk melakukan pencarian pola string dalam teks menggunakan algoritma `Knuth-Morris-Pratt`. Sama seperti fungsi `BoyerMoore`, fungsi `KMP` mengembalikan posisi ditemukannya pola jika kecocokan ditemukan dan `-1` jika tidak ditemukan. Fungsi ini memanfaatkan fungsi pembantu `PrefixFunction` yang tersimpan di `util.go`. `PrefixFunction` mengembalikan array of integer yang menyatakan panjang prefix terbesar yang juga merupakan suffix untuk suatu bagian pola pada posisi tertentu.

Fungsi `Similarity` digunakan untuk melakukan perhitungan tingkat kecocokan suatu pola dibandingkan dengan teks. Fungsi ini memanfaatkan fungsi bantuan `KmpTableGenerator` yang akan menghasilkan jump table yang dipakai dalam proses pencarian serta fungsi `Lcs`. Sebenarnya fungsi `similarity` ini merupakan pengembangan dari fungsi `KMP`.

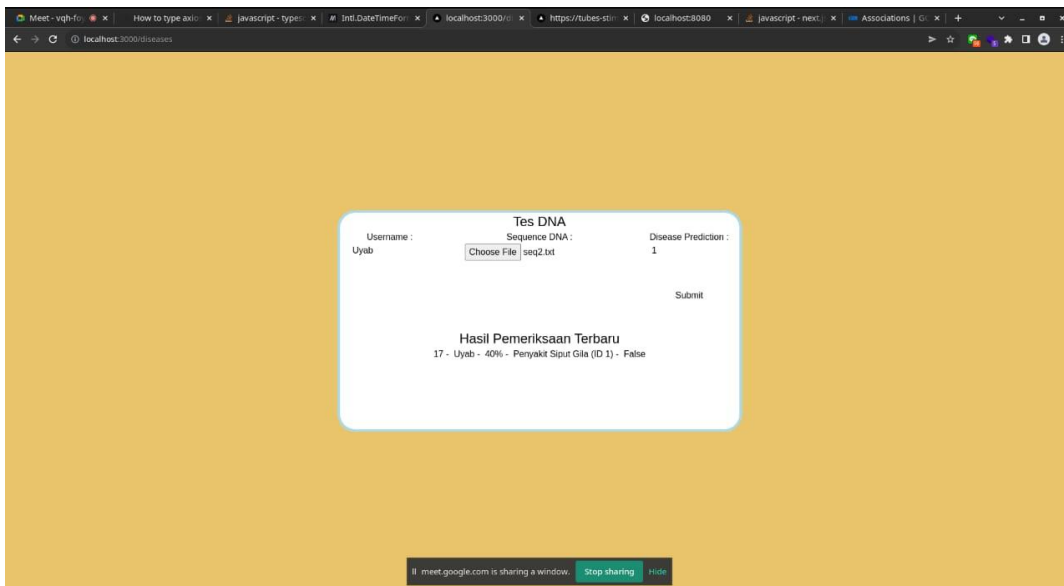
## 4.2 Tata Cara Penggunaan Program

Program terdiri atas dua komponen, yaitu backend dan frontend. Program backend menggunakan environment Go lang dan sql, sedangkan program frontend menggunakan Typescript dan framework React. Dipakai juga Docker dan Heroku sebagai untuk deploy dan build websitenya, dengan dependency yang diperlukan kedua program terdapat pada gambar di bawah. Database Management System yang digunakan adalah MariaDB (development di lokal) dan MySQL (deployment di server). Skema relasional basis data terdapat pada gambar di bawah.

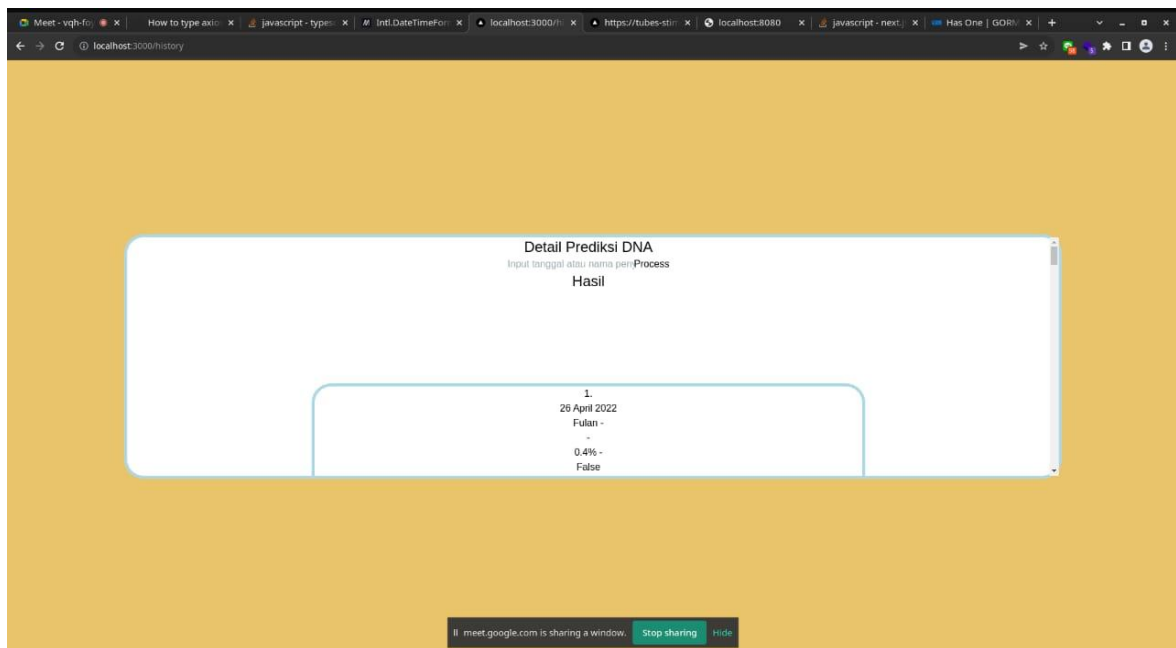


**Gambar 4.2.1** Skema relasional basis data

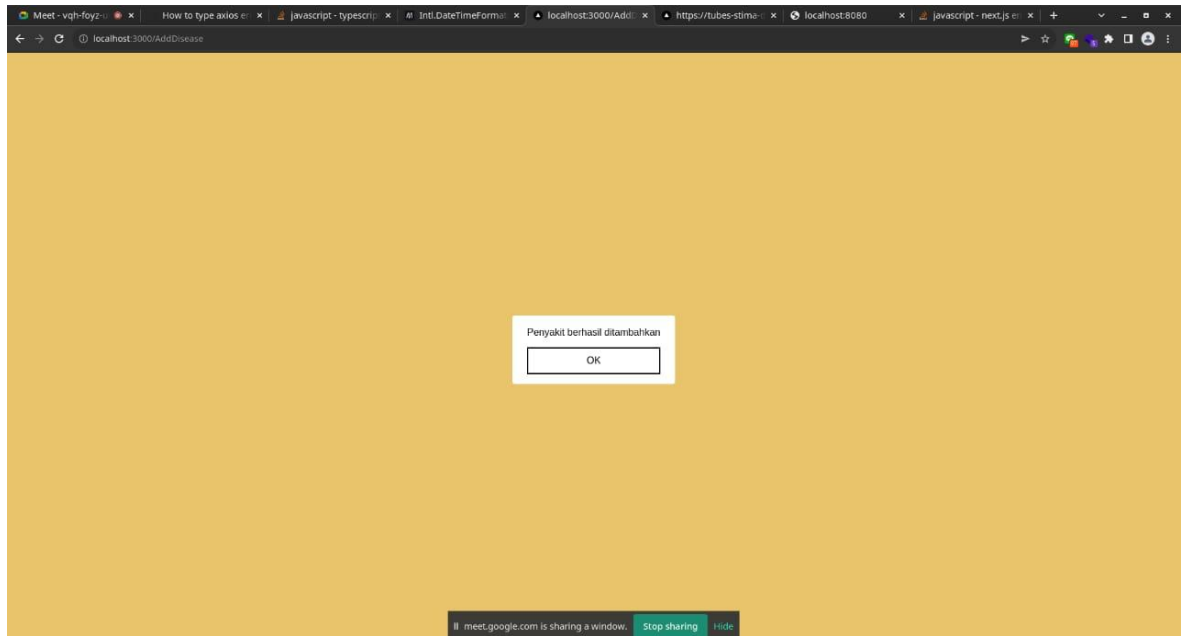
## 4.3 Hasil Pengujian



**Gambar 4.3.1** Melakukan hasil pemeriksaan



**Gambar 4.3.2** Hasil detail prediksi penyakit



**Gambar 4.3.3** Penyakit berhasil ditambahkan

## 4.4 Analisis Hasil Pengujian

Berdasarkan hasil pengujian diatas diketahui bahwa fitur penambahan penyakit dan prediksi penyakit telah berjalan dengan baik. Penyakit berhasil ditambahkan ke basis data dan hasil prediksi serta tingkat kecocokannya berhasil ditampilkan. Untuk pengecekan riwayat belum sempat diujikan.

## BAB V

### KESIMPULAN DAN SARAN

#### 5.1 Kesimpulan

Dalam tugas besar ke-3 IF2211 Strategi Algoritma kali ini yaitu terkait dengan penerapan berbagai Algoritma *Pattern Matching* seperti Algoritma KMP, *Hamming distance*, dan *Regular Expression* (Regex). Kelompok kami dapat menyimpulkan bahwa Algoritma yang paling cocok untuk pattern matching kali ini ialah Algoritma KMP. Hal tersebut dikarenakan string dari *sequence* DNA hanya terdiri 4 huruf. Menurut kompleksitasnya, untuk panjang string *pattern* yang cukup kecil Algoritma KMP lebih efisien dibanding dengan Algoritma *Pattern Matching* lain.

#### 5.2 Saran

Dalam pengerjaan tugas besar ini, website sudah dapat terdeploy dengan sebagai mana mestinya, walaupun memang bukan merupakan hasil deploy yang terbaik. Dengan demikian, dari hasil tersebut, diharapkan untuk kedepannya agar website yang kami buat dapat dikembangkan lebih lanjut dan dibuat agar lebih menarik dari sebelumnya.

## DAFTAR PUSTAKA

1. <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf>
2. <https://socs.binus.ac.id/2018/11/26/regular-expression/>
3. <https://www.petanikode.com/regex/>
4. <https://www.regexpal.com/>

Link Repository Github :

[https://github.com/bayusamudra5502/Tubes3\\_13520126](https://github.com/bayusamudra5502/Tubes3_13520126)

Link Website Deployment :

<https://tubes-stima-dna.netlify.app/> (Frontend)

<https://stima-tubes-dna.herokuapp.com/> (Backend)