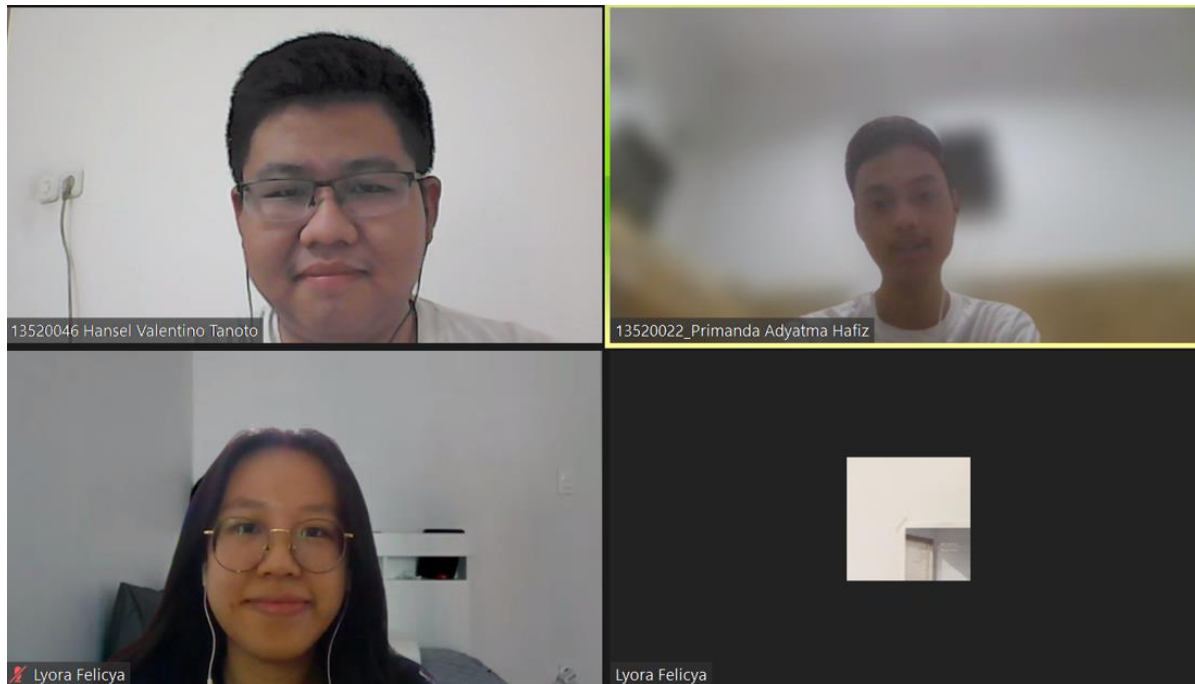


LAPORAN TUGAS BESAR 3

IF2211 STRATEGI ALGORITMA

Kelompok 22 – GWS



Disusun oleh:

Primanda Adyatma Hafiz	13520022
Hansel Valentino Tanoto	13520046
Lyora Felicya	13520073

TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

DAFTAR ISI

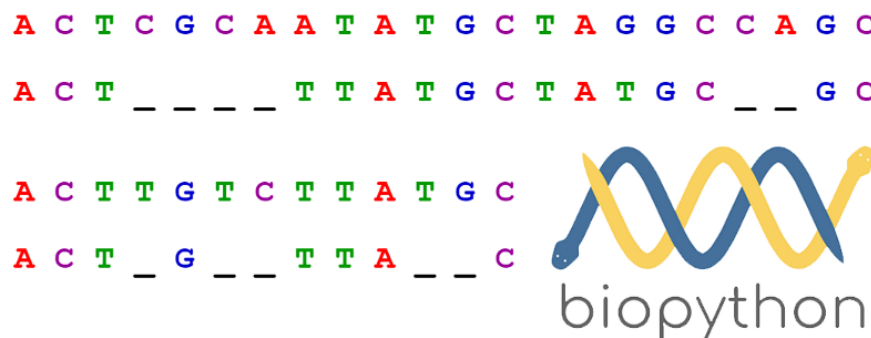
DAFTAR ISI	2
BAB 1 – DESKRIPSI TUGAS.....	3
BAB 2 – LANDASAN TEORI	10
A. ALGORITMA KMP (KNUTH–MORRIS–PRATT)	10
B. ALGORITMA BM (BOYER-MOORE)	10
C. REGULAR EXPRESSION (REGEX).....	11
D. APLIKASI WEB.....	11
BAB 3 – ANALISIS PEMECAHAN MASALAH.....	12
A. LANGKAH PENYELESAIAN MASALAH	12
B. FITUR FUNGSIONAL DAN ARSITEKTUR APLIKASI WEB	12
BAB 4 – IMPLEMENTASI DAN PENGUJIAN	13
A. SPESIFIKASI TEKNIS PROGRAM.....	13
B. TATA CARA PENGGUNAAN PROGRAM.....	14
C. HASIL PENGUJIAN.....	14
D. ANALISIS HASIL PENGUJIAN	18
BAB 5 – KESIMPULAN DAN SARAN	19
A. KESIMPULAN	19
B. SARAN.....	19
C. KOMENTAR / REFLEKSI	19
LINK <i>REPOSITORY</i> GITHUB	20
DAFTAR PUSTAKA.....	21

BAB 1

DESKRIPSI TUGAS

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. Ilustrasi Sekuens DNA

Sumber:

<https://towardsdatascience.com/pairwise-sequence-alignment-using-biopython-d1a9d0ba861f>

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.

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.

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:



Gambar 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 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.
 - a. Kolom pencarian dapat menerima masukan dengan struktur: <tanggal_prediksi><spasi><nama_penyakit>, 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**.
 - c. Contoh ilustrasi:
 - i. Masukan tanggal dan nama penyakit.

13 April 2022 HIV

1. 13 April 2022 - Fulan - HIV - True.

2. 13 April 2022 - Kamal - HIV - False.

3. 13 April 2022 - Entah - HIV - False.

4. 13 April 2022 - Jamal - HIV - True.

5. 13 April 2022 - Yubai - HIV - True.

6. 13 April 2022 - Hika - HIV - False.

Gambar 4. Ilustrasi Interaksi 1

- ii. 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.

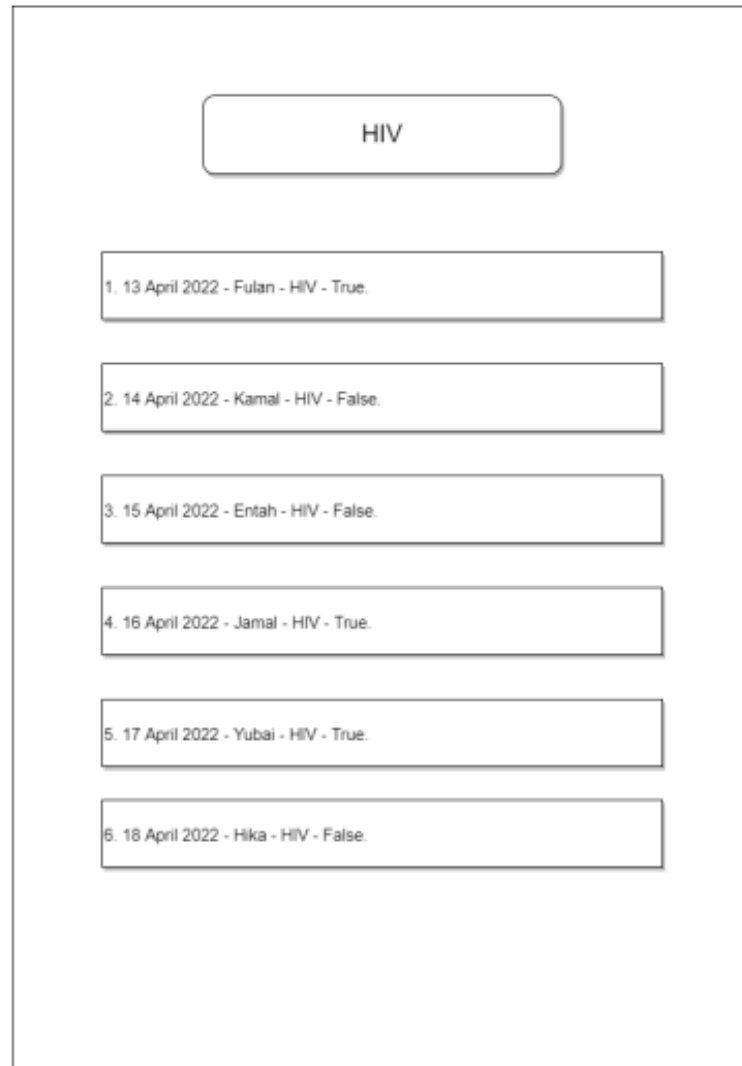
4. 13 April 2022 - Jamal - Polio - True.

5. 13 April 2022 - Yubai - TBC - True.

6. 13 April 2022 - Hika - Hepatitis A - False.

Gambar 5. Ilustrasi Interaksi 2

- iii. Masukan hanya nama penyakit



Gambar 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>

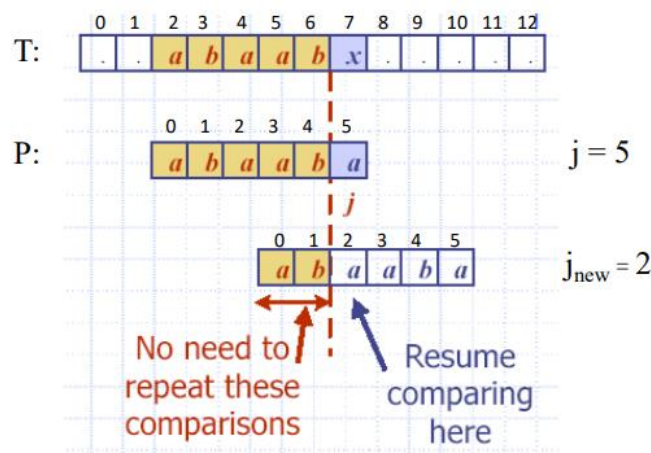
Gambar 7. Ilustrasi Prediksi jika Mengerjakan Bonus Menghitung Similarity

BAB 2

LANDASAN TEORI

A. ALGORITMA KMP (KNUTH–MORRIS–PRATT)

Algoritma KMP atau Knuth-Morris-Pratt merupakan salah satu algoritma yang digunakan dalam proses *string matching*, yaitu menemukan suatu pola teks dari suatu bacaan lain dengan membandingkan stringnya dari kiri ke kanan, sama seperti algoritma *brute force*. Perbedaannya terletak pada proses pergeseran string saat melakukan pencocokan. Jika algoritma *brute force* menempatkan pola teks dari awal sampai akhir bacaan, maka algoritma KMP ini akan melewati pengecekan apabila terdapat *prefix* dan *suffix* dari pola yang telah sama. Untuk lebih jelasnya, dapat dilihat dari gambar berikut.



Gambar 8. Ilustrasi Pergeseran String pada Algoritma KMP

Dari gambar tersebut, terlihat bahwa pola yang dicari, "abaab", memiliki *prefix* dan *suffix* yang sama, yaitu "ab". Oleh karena itu, kita dapat melewati langkah pengecekan terhadap "ab" dan mulai melakukan *string matching* dimulai dari indeks setelah pola tersebut. Melalui algoritma KMP ini, proses pengecekan *string matching* secara dapat menjadi lebih singkat dan efisien.

B. ALGORITMA BM (BOYER-MOORE)

Algoritma Boyer-Moore (BM) merupakan salah satu algoritma lainnya yang dapat digunakan untuk pencocokan *string*. Algoritma ini terbagi menjadi dua teknik, yaitu teknik *looking-glass* dan *character-jump*. Teknik *looking-glass* merupakan teknik pencocokan pola (*pattern*) terhadap teks secara terbalik (dari kanan ke kiri), yakni dimulai dengan huruf terakhir polanya. Sedangkan teknik *character-jump* merupakan teknik untuk menangani pergeseran pola terhadap teks ketika terjadi ketidaksesuaian pola terhadap teks. Terdapat tiga kasus yang mungkin terjadi di antaranya:

1. Kasus 1

Kasus 1 terjadi ketika huruf yang tidak sama pada teks (misalkan x) masih terdapat pada bagian kiri pola yang belum dicocokkan. Ketika hal ini terjadi maka geser pola ke kanan hingga huruf x pertama ditemukan pada pola sejajar dengan huruf x pada teks.

2. Kasus 2

Kasus 2 terjadi ketika huruf x yang dimisalkan pada kasus 1 tidak terdapat di bagian kiri pola (terdapat di bagian kanan pola sehingga sudah dicocokkan sebelumnya). Ketika kasus ini terjadi cukup geser pola ke kanan sejauh 1 huruf atau karakter.

3. Kasus 3

Kasus 3 terjadi ketika kasus 1 dan kasus 2 tidak terjadi. Ketika berhadapan dengan hal ini maka dilakukan pergeseran ke kanan hingga huruf awal pada pola berada satu huruf atau karakter lebih jauh ke kanan dari huruf x yang dimisalkan di atas.

C. REGULAR EXPRESSION (REGEX)

Regex merupakan sekumpulan notasi yang dapat digunakan untuk pencocokan string. Regex bisa berupa sebuah teks (*string*) yang mendefinisikan sebuah pola pencarian sehingga dapat membantu kita untuk melakukan *matching* (pencocokan), *locate* (pencarian), dan manipulasi teks. Pencocokan *string* yang dilakukan merupakan alternatif dari *exact matching* sehingga didapatkan hasil pencocokan yang lebih fleksibel tergantung notasi yang digunakan untuk mencocokkan pola dengan teks.

D. APLIKASI WEB

Aplikasi *DNA Pattern Matching* yang dibuat merupakan sebuah aplikasi interaktif yang dapat mendeteksi apakah seorang pasien mempunyai penyakit genetik tertentu. Hasil prediksi tersebut juga dapat disimpan pada basis data dan kemudian ditampilkan berdasarkan *query* pencarian. Untuk lebih detailnya, berikut adalah fitur-fitur dari aplikasi:

1. Menambahkan penyakit baru
Aplikasi dapat menerima input penyakit baru berupa nama penyakit dan *sequence* DNA dari penyakit tersebut.
2. Melakukan tes DNA
Aplikasi dapat memprediksi seseorang menderita penyakit tertentu berdasarkan *sequence* DNA-nya
3. Menampilkan hasil prediksi yang telah dilakukan
Aplikasi memiliki halaman yang menampilkan urutan hasil prediksi dengan kolom pencarian di dalamnya. Kolom pencarian bekerja sebagai filter dalam menampilkan hasil.

Seluruh proses pencocokan *string* dilakukan dengan algoritma *string matching* KMP, dan sanitasi input serta fitur pencarian diimplementasikan menggunakan regex.

BAB 3

ANALISIS PEMECAHAN MASALAH

A. LANGKAH PENYELESAIAN MASALAH

1. Fitur menambahkan penyakit baru
Pada fitur ini, digunakan regex untuk melakukan sanitasi input terlebih dahulu pada *sequence* DNA yang diterima sebelum ditambahkan ke *database*. Jika *sequence* DNA yang dibaca valid, serta nama penyakit yang ingin dimasukkan belum ada di *database*, maka data nama penyakit dan *sequence* DNA tersebut akan ditambahkan ke dalam *database*.
2. Fitur melakukan tes DNA
Pada fitur ini, digunakan KMP *string matching* untuk memeriksa adanya pola yang sesuai antara *sequence* DNA *user* dengan *sequence* DNA pada database penyakit. Jika ditemukan kecocokan atau *similarity* bernilai $> 80\%$, maka hasil tes akan menunjukkan bahwa prediksi penyakit pengguna adalah benar.
3. Fitur menampilkan hasil prediksi yang telah dilakukan
Pada fitur ini, digunakan regex untuk melakukan pemeriksaan terhadap tanggal, nama penyakit, atau keduanya yang dituliskan pada kolom pencarian. Setelah itu, jika ditemukan data yang cocok dari *database*, maka akan ditampilkan hasil pencariannya.

B. FITUR FUNGSIONAL DAN ARSITEKTUR APLIKASI WEB

Fitur fungsional yang dapat dilakukan oleh aplikasi ini adalah:

1. Menambahkan penyakit baru
2. Melakukan tes DNA
3. Menampilkan hasil prediksi yang telah dilakukan

Secara garis besar, *source code* dari aplikasi ini terbagi menjadi 2, yaitu *folder client* yang berisi seluruh kebutuhan *frontend* dan *folder server* yang berisi seluruh kebutuhan *backend* dari aplikasi.

Implementasi *backend* dari aplikasi ini menggunakan salah satu *framework* dari Node.js, yaitu Express.js. Terdapat 3 buah folder utama untuk *backend* yaitu folder *model* yang berisi skema dari database, *controller* yang berisi konfigurasi penambahan data dari *request website* ke database, serta *policy* yang berfungsi untuk melakukan validasi data *request* yang masuk.

Frontend menggunakan bahasa pemrograman Javascript dengan *framework* Vue.js. Terdapat 3 buah folder utama pada bagian *client*, yaitu *folder components* yang berisi kode untuk mengatur tampilan web, *folder routes* yang berisi index.js untuk melakukan navigasi antar halaman, serta *folder services* untuk melakukan *request* ke bagian *backend*.

Adapun *database* yang kami gunakan yaitu dengan *database* lokal di mysql. Terdiri atas 2 tabel yaitu tabel DNADiseases untuk menyimpan nama penyakit dan *sequence* DNA serta tabel HistoryTests untuk menyimpan hasil prediksi yang pernah dilakukan.

BAB 4

IMPLEMENTASI DAN PENGUJIAN

A. SPESIFIKASI TEKNIS PROGRAM

Berikut spesifikasi model beserta fungsi dan prosedur yang digunakan:

1) Model

Nama	Struktur	Kegunaan
DNADisease	DiseaseName: <i>String</i> DNASequence: <i>Text</i>	Menyimpan masukan data penyakit beserta sekuens DNA-nya
HistoryTest	Username: <i>String</i> DiseaseName: <i>String</i> DNASequence: <i>Text</i> TestDate: <i>Date</i> Percentage: <i>Float</i> Status: <i>Boolean</i>	Menyimpan informasi hasil tes DNA dari seorang pengguna

2) Fungsi dan Prosedur

Nama	Kegunaan
async void testDisease (req, res)	Melakukan pemrosesan data <i>request</i> untuk memperoleh tingkat kemiripan DNA pengguna dengan DNA penyakit masukan beserta statusnya
void KMP (DNAUser, DNADisease)	Melakukan pencocokan <i>string</i> dengan algoritma KMP
int[] KMPBorderFunction (DNADisease)	Menghitung <i>border function</i> dari algoritma KMP
void BoyerMoore (DNAUser, DNADisease)	Melakukan pencocokan <i>string</i> dengan algoritma Boyer Moore
int BMLastOccurence (DNADisease)	Mengkomputasi urutan kemunculan terakhir dari tiap karakter di string pattern
int HammingDistance (DNAUserSample)	Menghitung <i>hamming distance</i> (banyaknya perbedaan karakter) antara dua <i>string</i>
float largestSimilarity (DNAUser, DNADisease)	Menghitung nilai <i>similarity</i> terbesar yang ditemukan selama proses <i>string matching</i>
async void getHistoryTests (req,res)	Melakukan <i>filtering</i> data riwayat pengetesan berdasarkan paramater tanggal tes dan nama penyakit
async addData (req, res)	Menambah data penyakit dari <i>request</i> ke <i>database</i>
void checkDNA (req, res, next)	Melakukan validasi DNA dengan menggunakan regex
void checkQuery (req, res, next)	Melakukan validasi <i>query</i> dan <i>parsing query</i> ke format tanggal dan nama penyakit

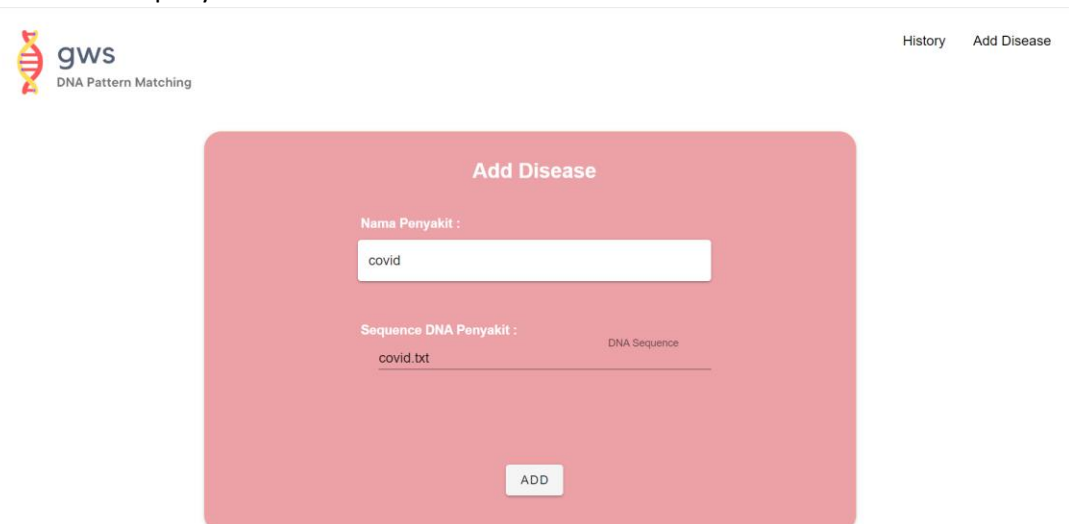
B. TATA CARA PENGGUNAAN PROGRAM

Langkah-langkah untuk menjalankan dan menggunakan aplikasi *DNA Pattern Matching* ini adalah sebagai berikut:

1. *Install* semua *dependency* yang diperlukan menggunakan *command npm install* pada terminal di *directory src/server* dan *src/client*.
2. Buat *database* lokal bernama **dbmysql** di *mysql*
3. Edit data *password* pada file **config.js** yang berada di *directory src/server/config/config.js*. Jika tidak menggunakan *password*, tidak perlu lakukan langkah ini.
4. Jalankan program *frontend* dengan *command npm start* di *directory src/client*.
5. Jalankan program *backend* dengan *command npm start* di *directory src/server*.
6. Buka *browser* dan masuk ke laman <http://localhost:8080>.
7. Akan muncul tampilan laman utama yang berisi *form* untuk melakukan tes DNA. Silakan masukan *input* sesuai yang diminta dan tekan tombol SUBMIT untuk melakukan tes DNA.
8. Pada daerah pojok kanan atas, terdapat menu History yang digunakan untuk melihat history tes DNA yang sudah dilakukan dan menu Add Disease untuk menambahkan penyakit baru beserta sekuens DNA-nya.
9. Untuk cek *history* dan menambahkan penyakit baru, silakan masukan *input* sesuai dengan yang diminta oleh *form*.
10. Tekan logo aplikasi untuk kembali ke laman awal (laman tes DNA).

C. HASIL PENGUJIAN

- 1) Pemambahan penyakit berhasil



Gambar 9. Tampilan ketika Berhasil Menambahkan Penyakit Baru

- 2) Penambahan penyakit gagal

Add Disease

Nama Penyakit :

Sequence DNA Penyakit : DNA Sequence

Invalid DNA

Gambar 10. Tampilan ketika Gagal Menambahkan Penyakit Baru

3) Tes yang gagal

DNA Test

Nama Pengguna :

Prediksi Penyakit :

Sequence DNA Pengguna : DNA User

Disease not found

Hasil Tes

Tanggal :

Pengguna :

Nama Penyakit :

Persentase :

Hasil :

Gambar 11. Tampilan ketika Gagal Melakukan pengetesan

4) Tes dengan hasil *false*

DNA Test

Nama Pengguna :

Prediksi Penyakit :

Sequence DNA Pengguna :

DNA User
DNA Sequence file is required

SUBMIT

Hasil Tes

Tanggal : 2022-04-29

Pengguna : alif

Nama Penyakit : covid

Persentase : 0

Hasil : false

Gambar 12. Tampilan ketika Melakukan Tes dengan Hasil False

dna alif - Notepad
— □ ×

File Edit Format View Help

ATGTCGTAAAT

Gambar 13. Isi File Sekuens DNA Penguin yang Dimasukkan (1)

5) Tes dengan hasil *true*

DNA Test

Nama Pengguna :

Prediksi Penyakit :

Sequence DNA Pengguna :

DNA User
DNA Sequence file is required

SUBMIT

Hasil Tes

Tanggal : 2022-04-29

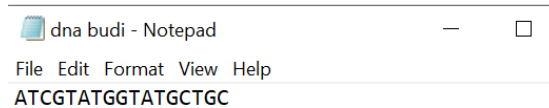
Pengguna : budi

Nama Penyakit : flu

Persentase : 1

Hasil : true

Gambar 14. Tampilan ketika Melakukan Tes dengan Hasil True



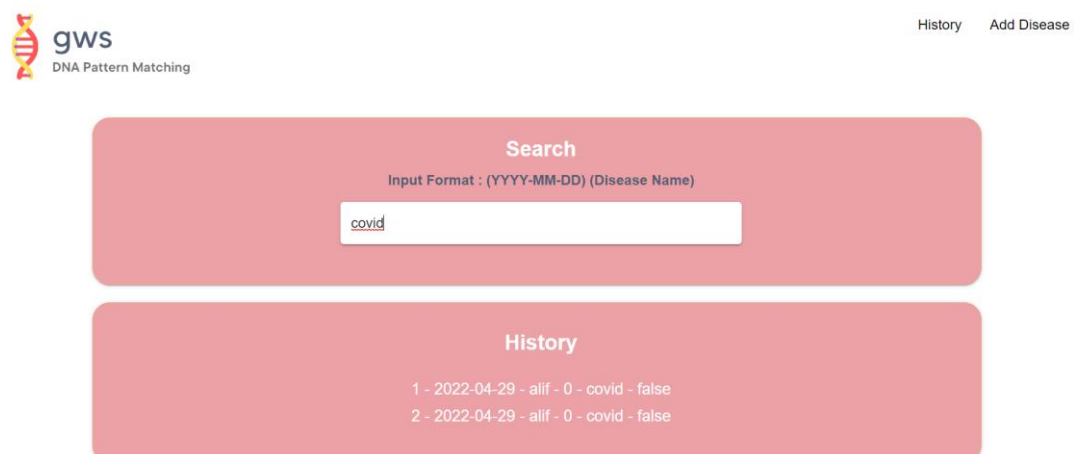
Gambar 15. Isi File Sekuens DNA Pengguna yang Dimasukkan (2)

6) Pencarian data tes berdasarkan tanggal



Gambar 16. Tampilan Hasil Pencarian History Berdasarkan Tanggal

7) Pencarian data tes berdasarkan nama penyakit



Gambar 17. Tampilan Hasil Pencarian History Berdasarkan Nama Penyakit

8) Pencarian data tes berdasarkan tanggal dan nama penyakit

Search
 Input Format : (YYYY-MM-DD) (Disease Name)

History
 1 - 2022-04-29 - budi - 0 - flu - false
 2 - 2022-04-29 - budi - 1 - flu - true

Gambar 18. Tampilan Hasil Pencarian History Berdasarkan Tanggal dan Nama Penyakit

9) Pencarian data tes dengan hasil pencarian tidak ditemukan

Search
 Input Format : (YYYY-MM-DD) (Disease Name)

History
 Data not found

Gambar 19. Tampilan Hasil Pencarian History yang Tidak Ditemukan

D. ANALISIS HASIL PENGUJIAN

Berdasarkan berbagai pengujian yang telah dilakukan pada bagian sebelumnya, dapat disimpulkan bahwa algoritma *string matching* menggunakan KMP dan regex yang diimplementasikan sudah berjalan dengan baik. Fitur-fitur yang disebutkan dalam spesifikasi seperti menambahkan penyakit baru, melakukan tes DNA untuk prediksi penyakit, dan melakukan pencarian hasil prediksi dapat berjalan dengan baik. Pesan kesalahan juga muncul jika hasil pemeriksaan *sequence* DNA yang dilakukan dengan menggunakan regex tidak sesuai. Untuk fitur pencarian juga format pencariannya fleksibel, dapat berdasarkan tanggal, nama penyakit, atau keduanya dengan urutan yang dibebaskan.

BAB 5

KESIMPULAN DAN SARAN

A. KESIMPULAN

Melalui tugas besar 3 mata kuliah IF2211 Strategi Algoritma ini, dapat disimpulkan bahwa implementasi *string matching* dan *regular expression* sangatlah luas dan dapat dimanfaatkan untuk berbagai hal, salah satunya yaitu dapat diimplementasikan pada sistem *DNA Sequence Analysis*, yaitu sebuah cara untuk memprediksi berbagai macam penyakit yang tersimpan pada *database* berdasarkan urutan *sequence* DNA-nya. Kelompok kami telah berhasil mengimplementasikan algoritma Knuth-Morris-Pratt (KMP) dan Boyer Moore, serta memanfaatkan algoritma KMP untuk digunakan pada proses pencocokan *sequence* DNA pada aplikasi. Implementasi regex juga berhasil mendeteksi *input* tanggal dan nama penyakit pada fitur pencarian, serta pada proses sanitasi *input*. *User Interface* dari aplikasi kami dibuat dengan *framework* Vue.js, dan algoritmanya dengan Express.js. Luaran yang dihasilkan juga sesuai dengan spesifikasi tugas yang diberikan.

B. SARAN

Untuk tampilan *web*, dapat dibuat lebih menarik lagi dengan memanfaatkan fitur-fitur lainnya yang telah disediakan oleh Vue.js, serta dibuat lebih interaktif dan halaman website yang responsif.

C. KOMENTAR / REFLEKSI

Tugas besar 3 dari mata kuliah IF2211 Strategi Algoritma ini mendorong kami untuk melakukan eksplorasi sendiri untuk mempelajari *framework backend* dan *frontend* untuk membangun sebuah *website*. Melalui tugas ini juga kami dapat mengimplementasikan secara langsung algoritma yang sebelumnya dipelajari di kelas.

LINK REPOSITORY GITHUB

Kode program DNA *Pattern Matching* ini dapat diakses pada *link* Github berikut ini:

https://github.com/primahafiz/Tubes3_13520022

Demo program dapat dilihat pada *link* YouTube berikut ini: <https://youtu.be/kriar1iHHI>

DAFTAR PUSTAKA

- [1] Levitin, A., 2012. *Introduction to the Design & Analysis of Algorithms*. Essex: Pearson.
- [2] <http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf>
- [3] <http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf>