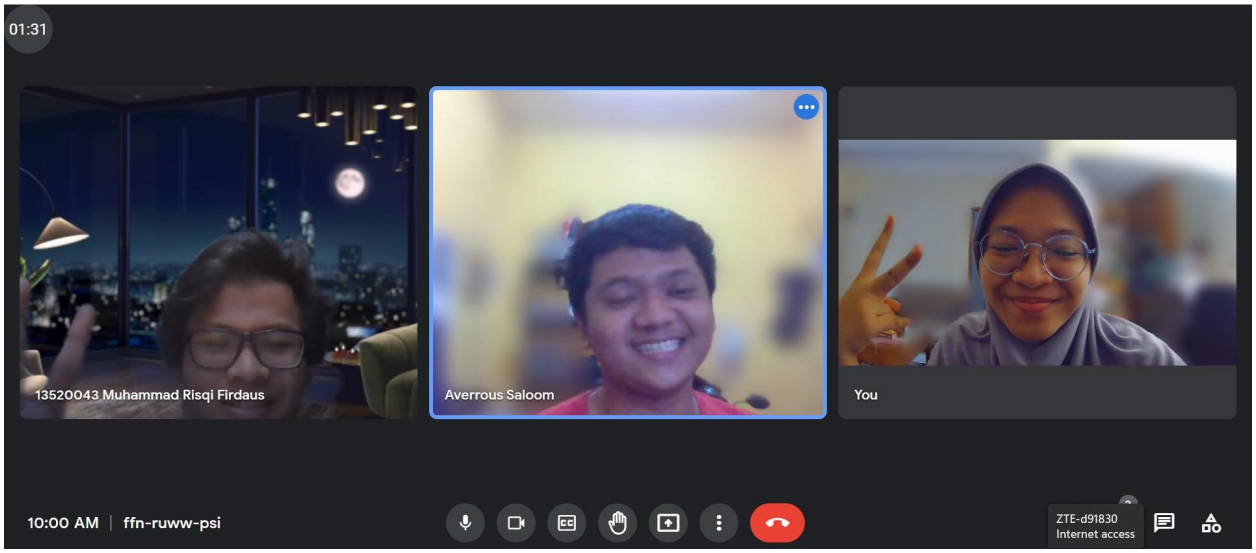
LAPORAN TUGAS BESAR III
IF2211 STRATEGI ALGORITMA

Laporan dibuat untuk memenuhi salah satu tugas mata kuliah
IF2211 Strategi Algoritma



Disusun oleh:

Kelompok

13520043 Muhammad Risqi Firdaus

13520045 Addin Nabilal Huda

13520100 Averrous Saloom

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

SEMESTER 2 TAHUN 2021/2022

DAFTAR ISI

DAFTAR ISI **2**

BAB I **3**

BAB II **4**

Aplikasi web yang dibangun 4

BAB III **5**

Langkah Penyelesaian Masalah 5

Fitur input penyakit 5

Fitur prediksi apakah seseorang menderita penyakit tertentu berdasarkan sekuens DNA miliknya 5

BAB IV **6**

BAB V **7**

Kesimpulan 7

Saran 7

TAUTAN REPOSITORY GITHUB **8**

DAFTAR PUSTAKA **8**

BAB I

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.

BAB II

LANDASAN TEORI

Algoritma KMP

Algoritma Knuth-Morris-Pratt (KMP) merupakan algoritma pencocokan string yang mencari pola string terurut dari kiri ke kanan yang dimodifikasi dari algoritma brute-force. Algoritma ini melakukan pencocokan String menggunakan degenerating property, yaitu pola yang memiliki sub-pola yang sama muncul lebih dari sekali dalam pola. Ide dasar algoritma ini adalah setiap kali algoritma mendeteksi ketidakcocokan, beberapa karakter dalam string selanjutnya sudah diketahui. Informasi ini digunakan untuk menghindari pencocokan karakter pada karakter yang sudah diketahui tidak akan cocok. Dengan konsep ini, kita dapat menghindari wasteful comparisons.

Algoritma KMP melakukan preprocessing pattern untuk menemukan kecocokan prefix pola dengan pola itu sendiri. Misal ketidakcocokan terjadi pada karakter ke-j di pola P dan k merupakan posisi sebelum posisi ketidakcocokan (k = j-1). Border function KMP, b(k), didefinisikan sebagai nilai prefix terbesar P[0...j-1] yang juga merupakan suffix dari P[1...j-1]. Nilai border function ini merupakan nilai terbesar untuk melakukan loncaran pemeriksaan untuk menghindari wasteful comparisons.

Kompleksitas waktu untuk mengitung border function/fungsi pinggiran adalah O(m), sedangkan untuk melakukan pencarian string adalah O(n). Maka, kompleksitas waktu algoritma KMP adalah O(m+n). Bila dibandingkan dengan kompleksitas algoritma brute force untuk pencocokan string, algoritma KMP sangat cepat.

Keuntungan dari penggunaan algoritma KMP adalah kita tidak perlu melakukan "move backwards" sehingga algoritma ini cocok digunakan untuk memroses file dengan ukuran sangat besar yang dibaca dari device eksternal melalui network stream. Namun, algoritma KMP tidak berjalan dengan baik ketika ukuran variasi alfabet meningkat karena akan lebih banyak kemungkinan ketidakcocokan yang terjadi di awal pola. KMP cepat ketika ketidakcocokan tidak terjadi di awal pola.

Algoritma Boyer-Moore (BM)

Algoritma Boyer-Moore merupakan algoritma pencocokan string yang didasari oleh dua teknik, yaitu:

1. *The looking-glass technique*

Teknik mencari pola P dalam teks T dengan bergerak mundur melalui P, dimulai dari belakang

2. *The character-jump technique*
   Bila terdapat ketidakcocokan pada T[i] == x dan karakter di pola P[j] !=T[i], terdapat 3 kemungkinan:
   - Jika pada pola P terdapat karakter x, geser P ke kanan untuk meng-align dengan kemunculan terakhir dari karakter x di P dengan T[i]
   - Jika pada pola P terdapat karakter x, tetapi pergeseran ke kanan tidak memungkinkan, geser P ke kanan sebanyak 1 karakter ke T[i+1]
   - Jika dua kemungkinan sebelumnya tidak terjadi, geser P untuk meng-align P[0] dengan T[i+1]

Algoritma ini melakukan *preprocessing* pola P dengan alfabet A untuk membentuk *last occurrence function* L() yang mereturn indeks terbesar i sehingga P[i] == x, dengan x adalah karakter di alfabet A dan -1 bila tidak ada indeks yang memenuhi ketentuan tersebut.

Worst case algoritma ini memiliki kompleksitas O(nm+A), yaitu pada kasus seperti melakukan pencocokan teks T: “aaaaa…a” dengan pola P: “baaaaa”. Algoritma ini cepat untuk memproses variasi alfabet A besar, dan sebaliknya. Maka, algoritma Boyer-Moore cocok digunakan untuk English text, dan buruk untuk binary.

Regular Expression

Regular expression merupakan template data berdasarkan pola tertentu. Dalam pencarian string, regular Expression (RE) berfungsi sebagai notasi yang dapat digunakan untuk mendeskripsikan pola dari kata yang ingin dicari.

Regex digunakan berdasarkan pencocokan format string atau teks dengan rumus yang dimasukkan oleh pengguna. Notasi regex bersifat universal, sehingga untuk berbagai bahasa pemrograman terdapat notasi sejenis.

Regex dapat digunakan untuk memfiltrasi teks maupun mengeksaminasi dan memisahkan suatu pola substring dalam teks yang lebih panjang. Penggunaan regex terbilang efektif, karena dapat mengeksaminasi string dengan cepat serta simpel.

Aplikasi web yang dibangun

Aplikasi web yang dibangun terdiri dari backend dan frontend. Backend memproses data sequence, menyimpannya di basis data, serta menyediakan endpoint sebagai antarmuka yang digunakan untuk berkomunikasi dengan frontend. Frontend akan mengirimkan permintaan kepada endpoint yang telah disediakan untuk melakukan operasi seperti menambahkan data penyakit, memeriksa apakah seseorang memiliki penyakit, dan mengecek riwayat. Backend akan mengembalikan respons yang menunjukkan apakah operasi berhasil dilakukan atau mengandung data yang diminta oleh frontend.

Backend diimplementasi dengan NodeJS dengan bantuan pustaka ExpressJS. Sementara frontend menggunakan ReactJS dengan bantuan pustaka penghias SemanticUI.

BAB III

ANALISIS PEMECAHAN MASALAH

Langkah Penyelesaian Masalah

Fitur input penyakit

- Frontend menerima nama penyakit serta sekuens dna dalam bentuk file
- Sebelum mengirim ke backend, frontend memastikan sekuens dna sesuai dengan aturan, yakni tidak mengandung huruf kecil, huruf selain AGCT, dan spasi menggunakan regex. Jika masih mengandung hal tersebut, permintaan ke backend tidak akan dilakukan.
- Jika sekuens valid, frontend mengirimkan permintaan *post* dengan menyertakan *payload* berisi nama dan sekuens dna dalam bentuk string.
- Backend akan menerima permintaan dan menyimpan data penyakit ke dalam basis data.
- Setelah berhasil menyimpan, kirim respon kembali ke frontend.

Fitur prediksi apakah seseorang menderita penyakit tertentu berdasarkan sekuens DNA miliknya

- Frontend menerima nama pasien, sekuens DNA pasien, serta nama penyakit yang ingin diperiksa.
- Sebelum mengirim ke backend, frontend memastikan sekuens dna sesuai dengan aturan, yakni tidak mengandung huruf kecil, huruf selain AGCT, dan spasi menggunakan regex. Jika masih mengandung hal tersebut, permintaan ke backend tidak akan dilakukan.
- Jika sekuens valid, frontend mengirimkan permintaan *post* dengan menyertakan *payload* berisi nama pasien, sekuens DNA pasien, serta nama penyakit yang ingin diperiksa.
- Backend akan menerima permintaan, dan melakukan tahapan berikut:
    - Jika nama penyakit tidak ada di database, kembalikan respon dengan status 404
    - Jika ada, proses apakah sekuens pasien mengandung sekuens penyakit menggunakan algoritma yang dipilih.
    - Jika sekuens dari pengguna tidak mengandung sekuens penyakit yang di-*input*-kan, maka program akan menghitung nilai kemiripannya dengan membagi nilai levnthsiten distance dengan panjang sekuens. Sekuens yang dipilih sebagai penyebut adalah sekuens yang lebih panjang antara sekuens penyakit dan sekuens dari masukan pengguna.
    - Simpan hasil proses ke dalam basis data
- Setelah berhasil menyimpan, backend mengirim respon kembali ke frontend yang berisi tanggal, nama pasien, nama penyakit, serta nilai boolean apakah sekuens pasien mengandung sekuens penyakit
- Frontend menampilkan hasil ke layar

Aplikasi memiliki halaman yang menampilkan urutan hasil prediksi dengan kolom pencarian di dalamnya. Kolom pencarian bekerja sebagai filter dalam menampilkan hasil.

- Frontend akan menerima tanggal atau nama penyakit atau keduanya dalam satu kolom input
- Masukan akan diproses menggunakan RegEx untuk memisahkan tanggal dan nama penyakit. Jika nama penyakit tidak ada pada masukan, akan dianggap sebagai string kosong. Begitu pula sebaliknya.
- Jika minimal salah satu dari dua jenis kueri yang dapat digunakan terdapat pada string masukan, frontend mengirimkan permintaan *post* ke backend dengan menyertakan *payload* nama penyakit dan tanggal. Untuk kueri yang tidak ada dalam string masukan, dianggap sebagai string kosong.
- Backend akan menerima permintaan, dan melakukan tahapan berikut:
    - Jika kueri hanya nama penyakit, cari instans pada relasi hasil_test yang memiliki nama penyakit yang sama.
    - Jika kueri hanya tanggal, cari instans pada relasi hasil_test yang memiliki tanggal yang sama.
    - Jika kueri keduanya, cari instans pada relasi hasil_test yang memiliki nama penyakit dan tanggal yang sama.
- Respon berisi list hasil tes yang sesuai dengan kueri dikirimkan ke frontend
- Frontend menampilkan list tersebut

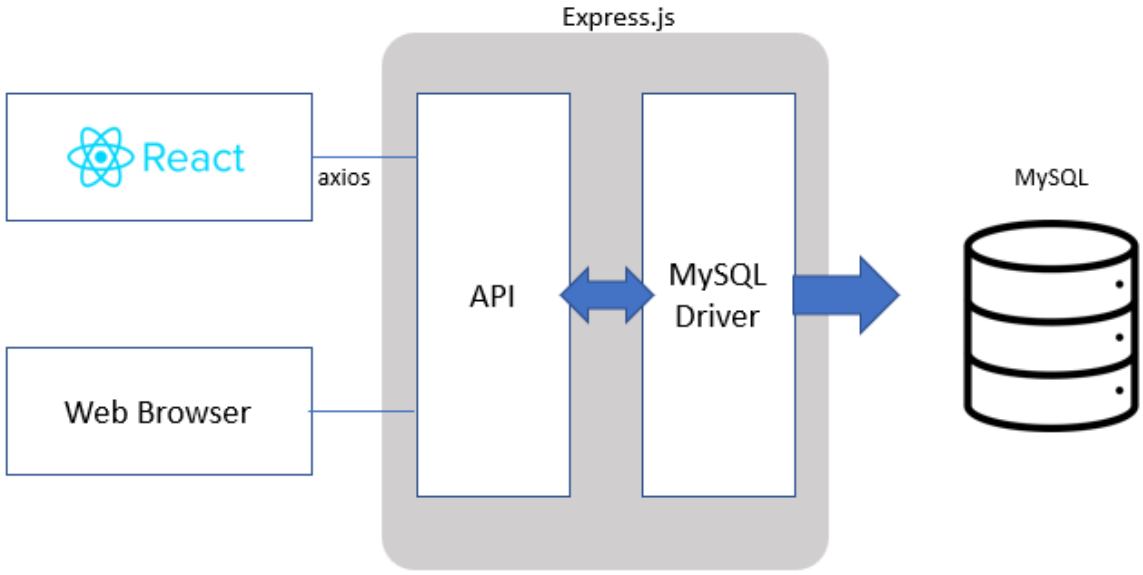
Fitur Fungsional dan Arsitektur Aplikasi Web yang Dibangun

Web yang dibangun memiliki fitur fungsional sebagai berikut:

- Web dapat menerima input penyakit baru berupa nama penyakit dan sequence DNA-nya (dan dimasukkan ke dalam database)
- Web dapat memprediksi seseorang menderita penyakit tertentu berdasarkan sequence DNA-nya.
- Web memiliki halaman yang menampilkan urutan hasil prediksi dengan kolom pencarian di dalamnya. Kolom pencarian merupakan filter dengan input tanggal, penyakit, atau tanggal dan penyakit
- Menghitung tingkat kemiripan DNA pengguna dengan DNA penyakit pada tes DNA

Arsitektur aplikasi web yang dibangun adalah sebagai berikut:

- Client: web browser
- Database server: MySQL
- Framework pengembangan frontend: React.js
- Framework pengembangan backend: Express.js


Express.js
React
axios
API
MySQL
Driver
MySQL
Web Browser

-

BAB IV

IMPLEMENTASI DAN PENGUJIAN

Spesifikasi Teknis Program

a. Struktur Data

Program menggunakan dua jenis struktur data, yakni untuk penyakit (Penyakit) dan struktur untuk hasil pencarian (PencarianP):

i. Penyakit

```
Interface Penyakit{
        id: number,
        nama: string,
        sequence: string
}
```

Id akan menyimpan id dari penyakit yang dimasukkan. Nama menyimpan nama penyakit yang dimasukkan, serta sequence menyimpan sequence dari penyakit.

Pada database, bentuk dari database yang menyimpan penyakit adalah sebagai berikut.

```
CREATE TABLE IF NOT EXISTS penyakit(
        id INT AUTO_INCREMENT PRIMARY KEY,
        nama VARCHAR(255) NOT NULL,
        sequence text NOT NULL
);
```

ii. PencarianP

Struct PencarianP akan menyimpan dari hasil pencarian yang pernah dilakukan menggunakan program.

```
interface PencarianP{
        namaPenyakit: string;
        tanggal: Date;
        namaPengguna: string;
        hasil: number;
        kemiripan: number;
}
```

iii. KnuthMorrisPrattPattern

Struct KnuthMorrisPrattPattern akan menyimpan string pola yang akan dicari, beserta nilai border dengan struktur data list dengan indeks yang mengindikasikan indeks k dan nilai list sebagai besar border.

```
interface KnuthMorrisPrattPattern {
```

```
  pattern: string;
  borderValue: number[];
}
```

namaPenyakit akan menyimpan namaPenyakit yang dicari, pada database, akan disimpan id dari penyakit yang diambil dari relasi penyakit. Tanggal akan diisi tanggal dilakukannya pencarian, namaPengguna, adalah nama yang diinputkan pada program, hasil akan menyimpan nilai kecocokan, 1 untuk nilai true dan 0 untuk nilai false, sedangkan kemiripan akan menyimpan nilai 1-(jarakLevithense/panjangSequence). Pada mysql akan dibentuk relasi sebagai berikut,

```
CREATE TABLE IF NOT EXISTS hasil_tes(
    id INT PRIMARY KEY AUTO_INCREMENT,
    id_penyakit INT, tanggal DATE NOT NULL,
    nama_pengguna VARCHAR(30) NOT NULL,
    hasil TINYINT(1) NOT NULL,
    kemiripan FLOAT,
    FOREIGN KEY (id_penyakit) REFERENCES penyakit(id)
);
```

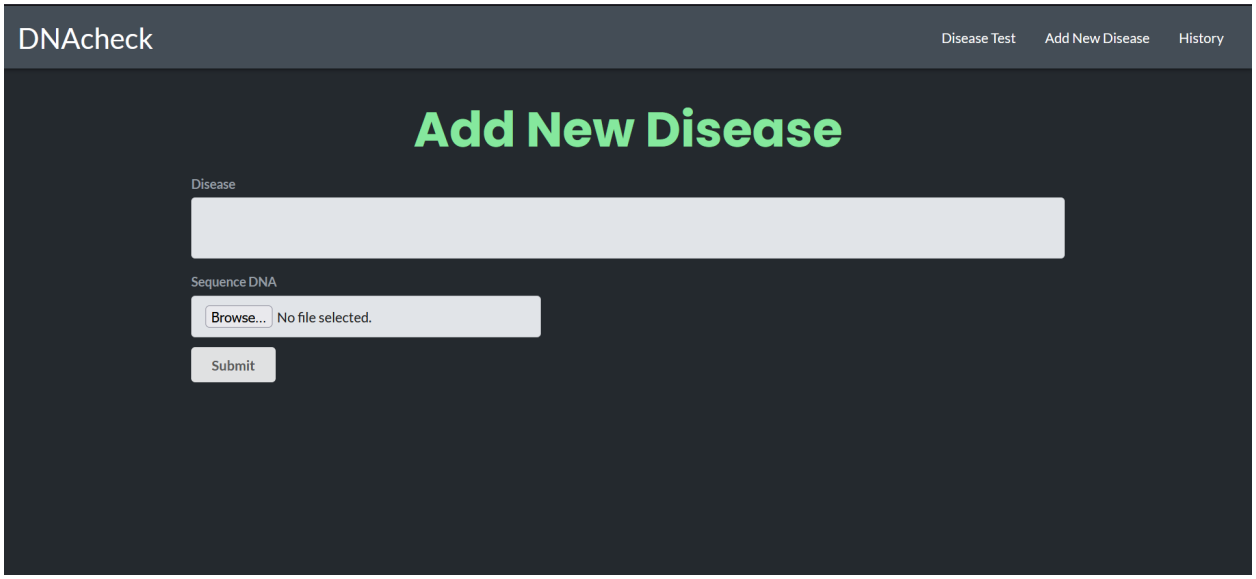
b. Fungsi yang digunakan

No	Nama Fungsi	Deskripsi
1.	Procedure findByNamaPenyakitAndTanggal(namaPenyakit: string, tanggal: string, callback: Function)	Prosedur ini akan masukkan array berisi hasil history pencariaan yang sesuai dengan parameter yng diberikan
2.	insertPenyakit(namaPenyakit: string, sequence:string, callback: Function)	Prosedur ini akan memasukkan data penyakit baru serta sequencenya ke dalam database
3.	findSimilar(namaPengguna: string, namaPenyakit: string, sequence: string, callback: Function)	Prosedur ini akan memasukkan nilai callback berupa sebuah array yang berisi penyakit yang bersuaian dengan data yang diinput, serta memberika nilai kesesuaian penyakit dan nilai kemiripannya.
4	KnuthMorrisPratt.find(pattern: KnuthMorrisPrattPattern, text: string): number	Fungsi ini akan mengembalikan nilai indeks dari awal pola ditemukan. Mengembalikan -1 jika tidak ditemukan

5.	KnuthMorrisPratt.border(pattern: string): number[]	Fungsi menerima string pola dan mengembalikan list border
6.	function levensthein(pattern: string, text: string)-> number	Fungsi ini akan mengembalikan nilai jarak levensthein dari dua buah string sequence yang dicocokkan.

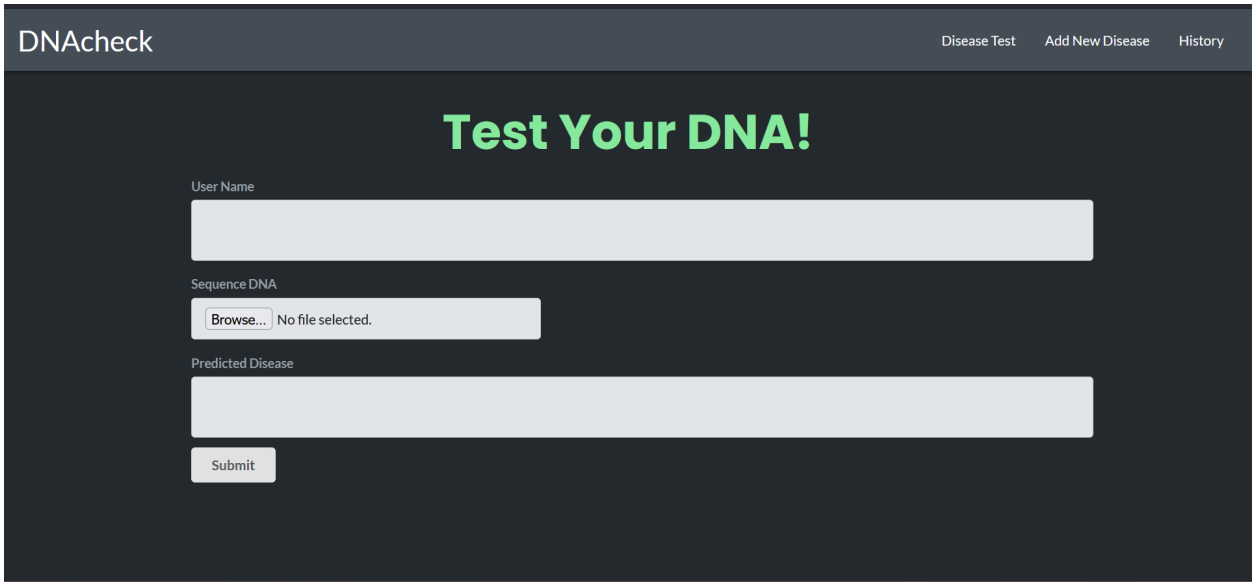
Penjelasan tata cara penggunaan program

Fitur input penyakit



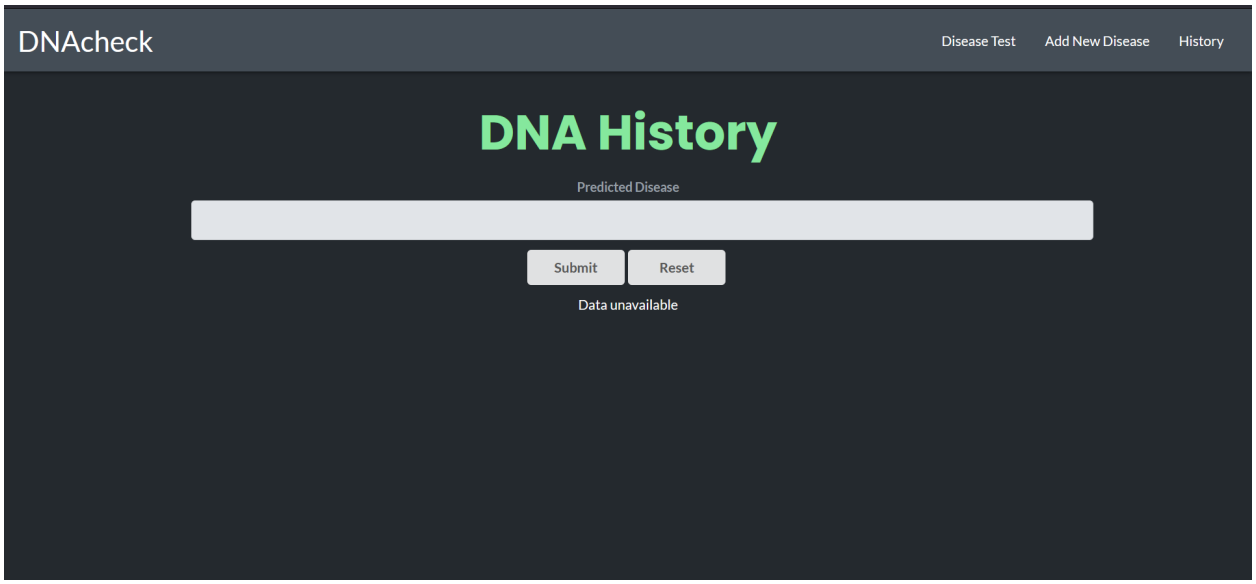
1. Buka halaman “Add New Disease” yang dapat diakses dari navigation bar. Halaman ini berupa form yang berisi field untuk input nama penyakit dan sequence DNA dari penyakit tersebut
2. Isi nama penyakit di text input field dengan label “Disease”
3. Masukkan file .txt ke file input field dengan label “Sequence DNA”
4. Tekan tombol “Submit”
5. Bila sequence DNA invalid, web akan menampilkan pesan error

Fitur cek penyakit



1. Buka halaman “Disease Test” yang dapat diakses dari navigation bar. Halaman ini berupa form yang berisi field untuk input nama pengguna, sequence DNA, dan prediksi penyakit
2. Isi nama pengguna di text input field dengan label “Disease”
3. Masukkan file .txt ke file input field dengan label “Sequence DNA”
4. Isi nama prediksi penyakit di text input field dengan label “Predicted Disease”
5. Tekan tombol “Submit”Bila sequence DNA invalid, web akan menampilkan pesan error
6. Bila sequence DNA valid, web akan menampilkan hasil pengecekan berupa nama pengguna, tanggal, hasil tes, dan persen kemiripan

Fitur history

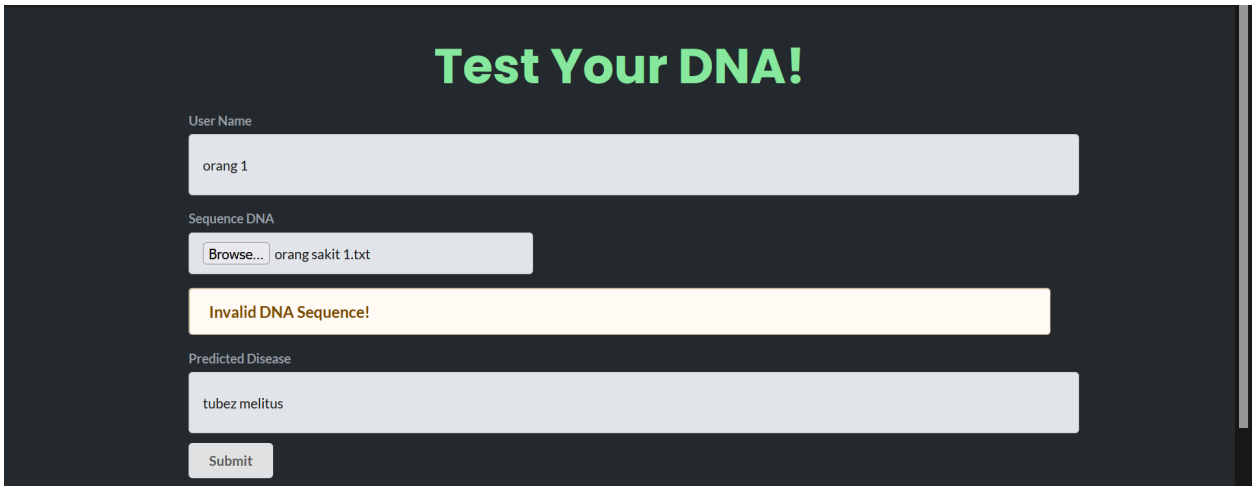


1. Buka halaman “History” yang dapat diakses dari navigation bar

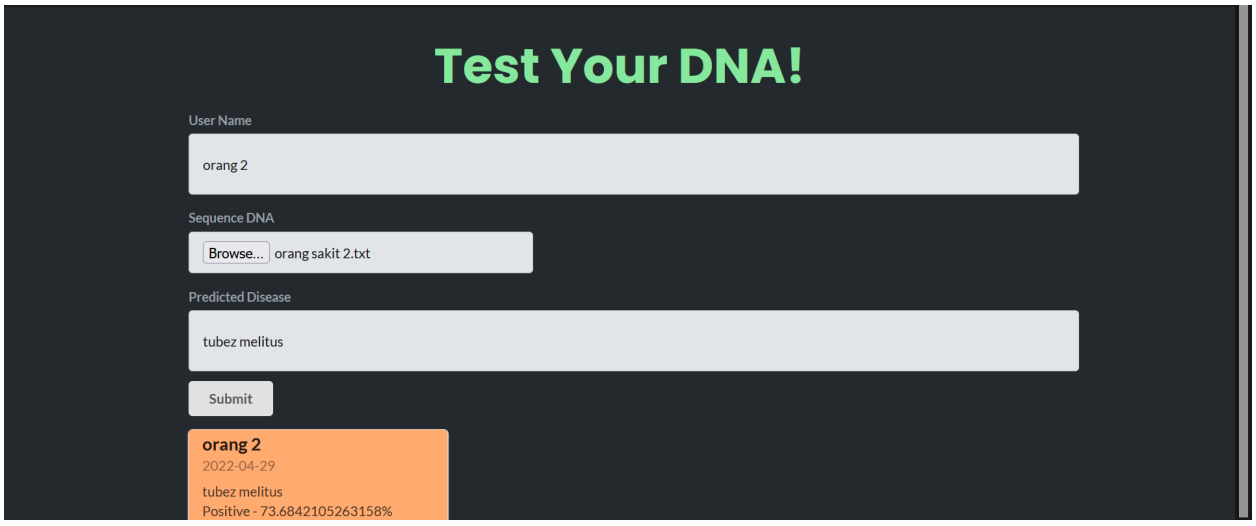
2. Isi query dengan tanggal, penyakit, atau tanggal dan nama penyakit. Misal “2022-04-29”, “HIV”, atau “2022-04-29 HIV”
3. Tekan tombol “Submit”
4. Bila tidak ada data yang sesuai, web akan menampilkan pesan
5. Bila ada data yang sesuai, akan tampil daftar data yang bersesuaian

Hasil Pengujian

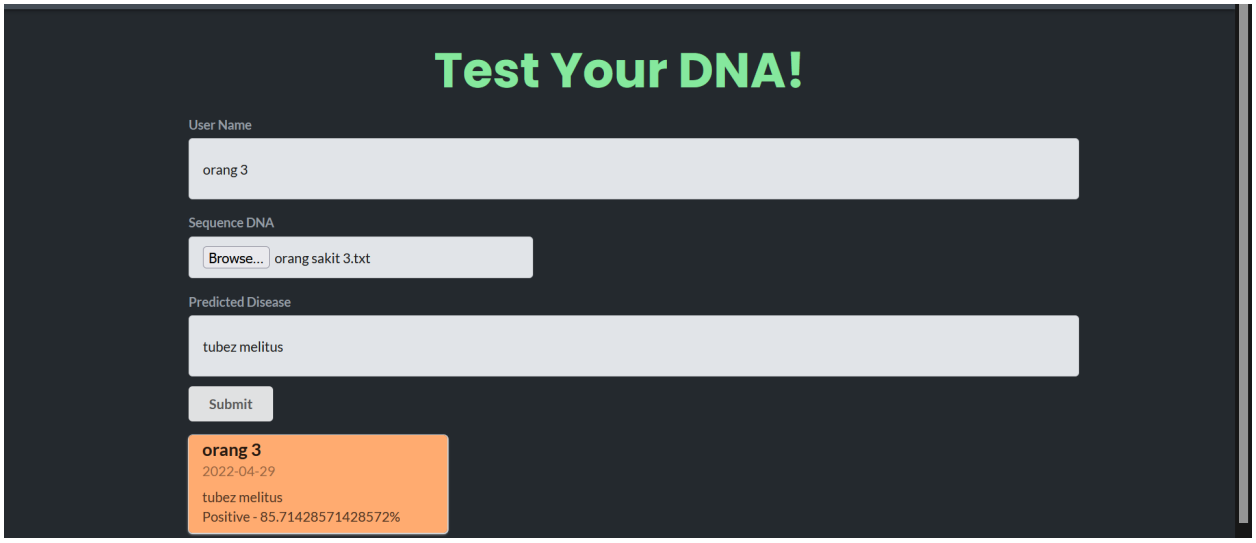
Menggunakan Sequence = ACGTAGCTAGATCG



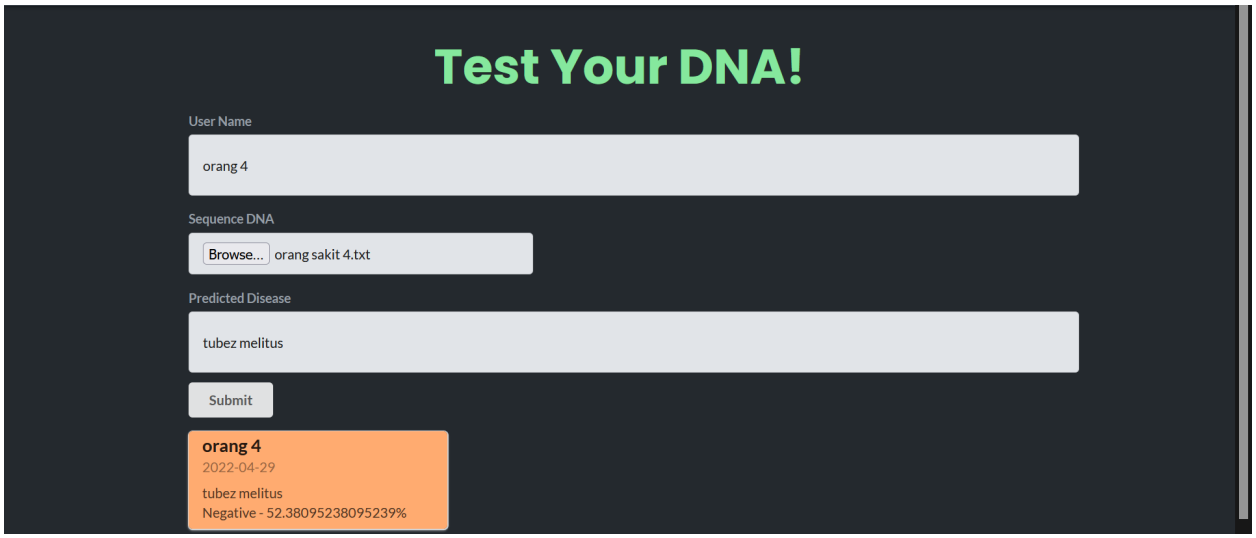
Kasus invalid DNA sequence AUGGGGACTAGCCI.



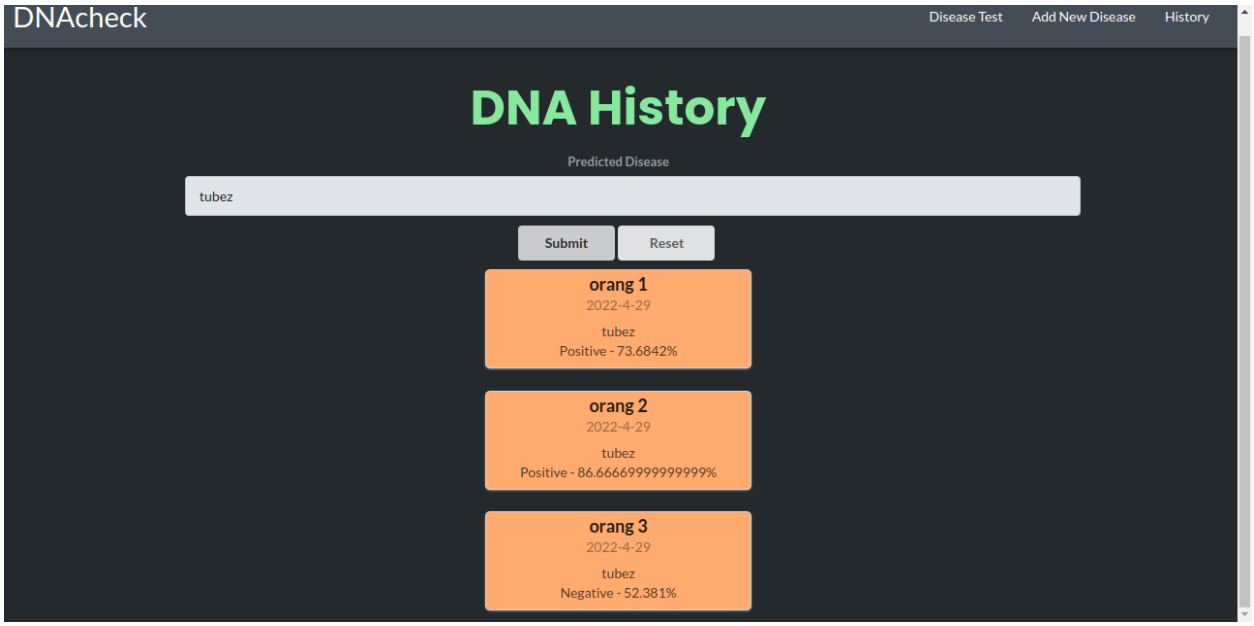
Orang 2 = ACGTAGCTAGATCGCGCTA

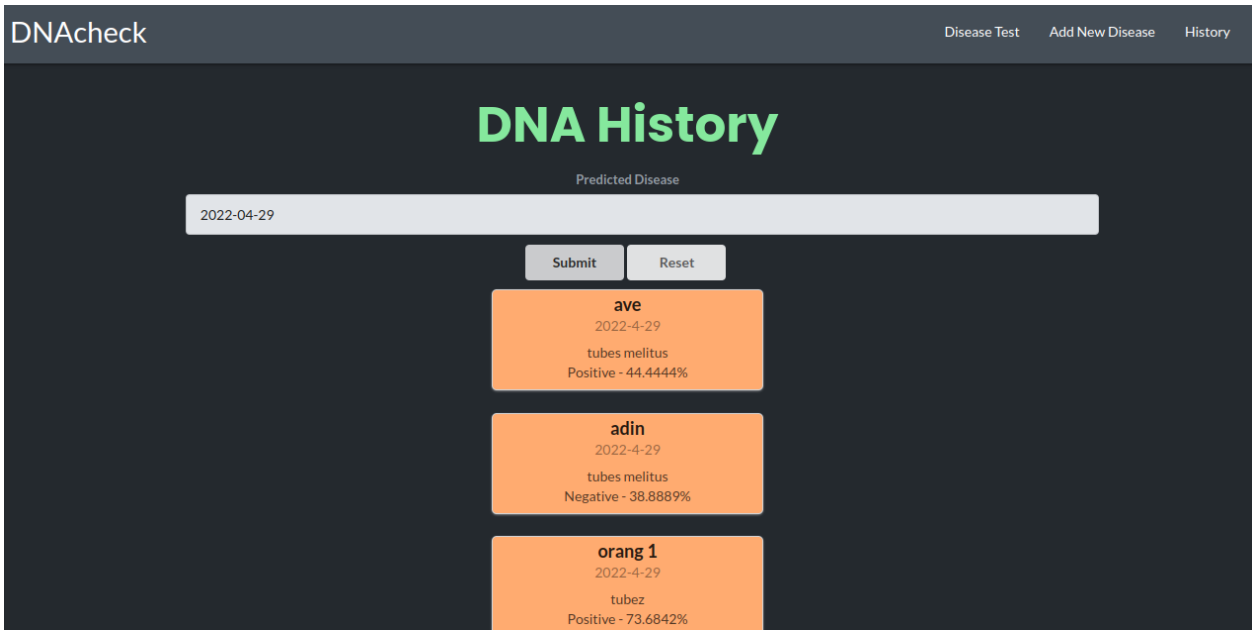


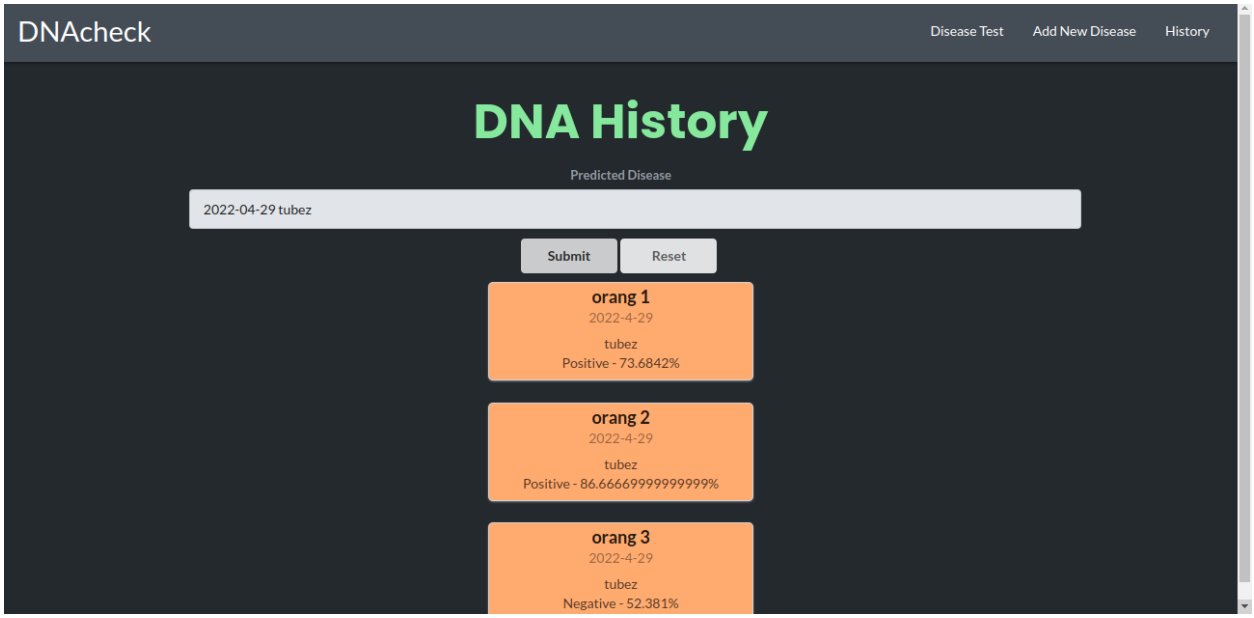
Orang 3 = ACGTAGCTAGAGTA



Orang 4 = AAGGGTACGTATAGCTAGCTA


DNAcheck
Disease Test
Add New Disease
History
DNA History
Predicted Disease
tubez
Submit
Reset
orang 1
2022-4-29
tubez
Positive - 73.6842%
orang 2
2022-4-29
tubez
Positive - 86.66669999999999%
orang 3
2022-4-29
tubez
Negative - 52.381%


DNAcheck
Disease Test
Add New Disease
History
DNA History
Predicted Disease
2022-04-29
Submit
Reset
ave
2022-4-29
tubes melitus
Positive - 44.4444%
adin
2022-4-29
tubes melitus
Negative - 38.8889%
orang 1
2022-4-29
tubez
Positive - 73.6842%



Analisis Hasil Uji

Berdasarkan pengujian yang dilakukan, website telah dapat melakukan identifikasi penyaringan DNA dengan Regex, melakukan string matching terhadap keberadaan substring DNA penyakit dalam DNA pengguna, menghitung kemiripan DNA dengan algoritma levenshtein serta menyimpan data dalam data base.

Terbukti pada pengguna dengan DNA bersubstring penyakit atau yang memiliki kemiripan DNA dengan DNA penyakit lebih dari 80 persen akan dianggap positif. Sedangkan, DNA yang memiliki kemiripan di bawah 80 persen dan tidak memiliki substring DNA penyakit amaka akan dianggap negatif. DNA yang tak sesuai dengan format seharusnya akan ditolak oleh sistem.

Program cukup efisien karena nilai border dari algoritma KMP serta nilai kemiripan berdasarkan levenshtein disimpan dalam database, serta pengguna join table dari database terbilang minim. Pengujian juga menunjukkan program telah berjalan dengan akurat. Meskipun tak menampik kemungkinan sedikitnya format tanggal yang ada pada pencarian historis.

BAB V

KESIMPULAN DAN SARAN

Kesimpulan

Saat ini, string atau kumpulan karakter menjadi cara termudah menyampaikan pesan. Ada tiga metode pencocokan string yang utama yakni Regex, Knuth Morris Path dan Boyer Moore. Ketiga metode pencocokan string ini memiliki kelebihan dan kekurangan masing-masing, dengan begitu, penggunaannya secara efektif pun akan berbeda.

Pada Program ini Regex digunakan untuk memfilter masukkan, KMP digunakan untuk melakukan string matching DNA. Regex sangat efektif untuk melakukan filtrasi dan pengelompokan pada berbagai jenis teks, sedangkan KMP sangat efektif digunakan pada string yang substringnya terkesan redundan atau hanya terdiri atas substring yang kurang bervariasi.

Penggunaan keduanya dapat digunakan sears efektif untuk memfilter masukan dari pengguna. Mengingat, pengembang tak bisa memastikan bahwa masukan dari pengguna pasti benar. Sehingga, untuk hasil pencarian yang maksimal, kata kunci pencarian perlu dieksaminasi terlebih dahulu.

Saran

Program masih perlu diakuratkan dari sisi string maching, mengingat algoritma levenshtein masih belum bisa menghitung jarak dari sebuah string yang mendekati substring dari string utama. Akibatnya, Jika ada string yang mendekati substring utama, dan perbedaan ukuran antara string yang dicek dan string utama terlampau jauh, maka jarak yang dihasilkan oleh algrotima levenstein akan bernilai besar.

Selain itu, perlu ada optimasi penggunaan algoritma string matching. Pada kasus ini algoritma yang digunakan hanya KMP sedangkan penggunaan algorithm booyer moore dirasa kurang efektif pada string matching DNA. Sehingga perlu adanya penggunaan lebih seperti pada pencarian nama orang atau penyakit dari database.

Dalam penggunaan regex pada pencarian historis pun belum maksimal. Hanya satu dari sekian format yang telah diimplementasikan pada program. Pada pengembangan lanjutan, perlu diperbanyak implementasi regex untuk berbagai jenis format masukan tanggal.

TAUTAN REPOSITORY GITHUB

https://github.com/mrfirdauss-20/Tubes3_13520043.git
https://youtu.be/2BRyCaDM-n4

DAFTAR PUSTAKA

https://www.geeksforgeeks.org/kmp-algorithm-for-pattern-searching/
https://socs.binus.ac.id/2018/11/26/regular-expression/
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf