

LAPORAN TUGAS BESAR III

IF2211 STRATEGI ALGORITMA

Penerapan String Matching dan Regular Expression dalam DNA Pattern Matching



Disusun oleh:

13520002 Muhammad Fikri Ranjabi

13520051 Flavia Beatrix Leoni A. S.

13520066 Putri Nurhaliza

TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

SEMESTER II TAHUN 2021/2022

DAFTAR ISI

DAFTAR ISI	i
BAB I DESKRIPSI TUGAS	1
BAB II LANDASAN TEORI	3
A. Algoritma Knuth–Morris–Pratt	3
B. Algoritma Boyer-Moore	3
C. Algoritma LCS (Longest Common Subsequence)	4
D. <i>Regular Expression</i>	4
E. Deskripsi Singkat Web.....	5
BAB III ANALISIS PEMECAHAN MASALAH	6
A. Langkah Penyelesaian Masalah Setiap Fitur	6
B. Fitur Fungsional dan Arsitektur Aplikasi Web.....	7
1. Fitur Fungsional.....	7
2. Arsitektur	7
BAB IV IMPLEMENTASI DAN PENGUJIAN	9
C. Spesifikasi Teknis Program.....	9
1. Struktur Data.....	9
2. Fungsi dan Prosedur.....	9
D. Tata Cara Penggunaan Program	12
E. Hasil Pengujian.....	13
1. Fitur Tambahkan Penyakit	13
2. Fitur Prediksi Penyakit	15
3. Fitur Pencarian Hasil Tes DNA.....	18
F. Analisis Hasil Pengujian.....	19
BAB V KESIMPULAN, SARAN, DAN REFLEKSI	20
LAMPIRAN	21
DAFTAR PUSTAKA	22

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.

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: <tanggal_prediksi><spasi><nama_penakit>, 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.

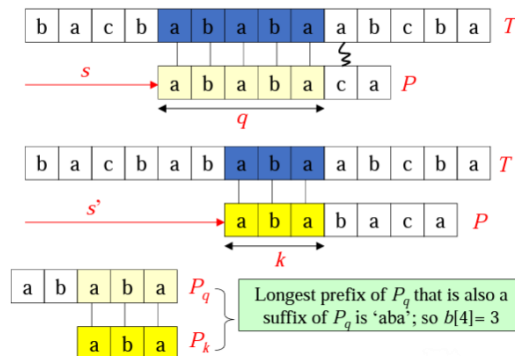
- c. Contoh ilustrasi:
 - i. Masukan tanggal dan nama penyakit
 - ii. Masukan hanya tanggal
 - iii. Masukan hanya nama penyakit
- 4. (Bonus) Menghitung tingkat kemiripan DNA pengguna dengan DNA penyakit pada tes DNA
 - a. Ketika melakukan tes DNA, terdapat persentase kemiripan DNA dalam hasil tes. Contoh 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.

BAB II

LANDASAN TEORI

A. Algoritma Knuth–Morris–Pratt

Algoritma Knuth-Morris-Pratt merupakan salah satu algoritma pencarian pattern dalam text dari kiri ke kanan. Algoritma ini lebih baik dibanding brute force dengan pendekatan yang lebih efektif pada pergeseran pattern. Jika terjadi ketidakcocokan antara text dan pattern P , misalkan $T[i] \neq P[j]$, maka pergeseran pattern dapat dilakukan pada prefix dari $P[0 \dots j-1]$ yang merupakan suffix dari $P[1 \dots j-1]$. Pergeseran ini akan mengurangi operasi perbandingan. KMP menggunakan fungsi pinggiran (*border function*) $b(k)$ yang didefinisikan sebagai ukuran prefix dari $P[0 \dots k]$ terbesar yang merupakan suffix $P[1 \dots k]$ dengan $k = j-1$ dan $b(k)$ sebagai nilai j yang baru. Algoritma KMP mengembalikan indeks dimana pattern dimulai pada text jika ditemukan. Jika tidak, akan mengembalikan nilai -1.

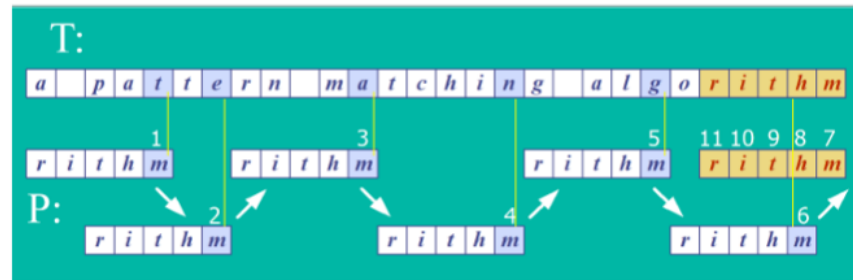


B. Algoritma Boyer-Moore

Selain algoritma KMP, Algoritma Boyer-Moore juga digunakan untuk pencocokan text dan pattern. Algoritma ini menggunakan dua metode, yaitu *looking-glass technique* dan *character-jump technique*. P ditemukan pada T dengan bergerak mundur melalui P , mulai dari ujung akhir. Lalu, *character-jump* dilakukan saat terjadi ketidakcocokan saat $T[i] = x$ dan karakter di pattern $P[j] \neq T[i]$. Terdapat 3 kasus yang mungkin, yaitu sebagai berikut.

1. Jika P memiliki x , maka geser P ke kanan hingga kemunculan terakhir x di P sejajar dengan $T[i]$.
2. Jika P memiliki x , namun tidak memungkinkan melakukan pergeseran di atas, maka geser P satu karakter ke kanan ($T[i+1]$).
3. Jika tidak memenuhi kedua kasus di atas, maka geser P sehingga $P[0]$ sejajar dengan $T[i+1]$.

Algoritma Boyer-Moore memproses pattern P dan alfabet A untuk membangun fungsi kemunculan terakhir L. Misal terdapat x pada A, maka $L(x)$ adalah index i terbesar yang memenuhi $P[i] = x$. $L(x)$ akan bernilai -1 jika tidak terdapat x pada A.



C. Algoritma LCS (Longest Common Subsequence)

LCS merupakan salah satu algoritma untuk mengukur kemiripan dua string. Misal diberikan text dan pattern, akan ditentukan panjang subsequence terpanjang yang ada pada keduanya. Sebuah subsequence adalah urutan huruf yang muncul dalam urutan relatif yang sama, tetapi tidak harus tepat bersebelahan. Misalnya, "abc", "abg", "bdf", "aeg", "acefg", .. dll adalah subsequence dari "abcdefg". Misalkan text $T[0..m-1]$ dan pattern $P[0..n-1]$ dengan panjang m dan n berturut-turut. Dan misalkan $L(T[0..m-1], P[0..n-1])$ adalah panjang LCS dari T dan P.

1. Jika karakter terakhir dari kedua sequence cocok ($T[m-1] = P[n-1]$) maka $L(T[0..m-1], P[0..n-1]) = 1 + L(T[0..m-2], P[0..n-2])$.
2. Jika karakter terakhir dari kedua urutan tidak cocok ($T[m-1] \neq P[n-1]$) maka $L(T[0..m-1], P[0..n-1]) = \text{MAX} (L(T[0..m-2], P[0..n-1]), L(T[0..m-1], P[0..n-2]))$

Karena pada tugas besar ini diasumsikan sequence DNA pengguna lebih panjang dari sequence DNA penyakit, maka DNA pengguna akan diterapkan sebagai text dan DNA penyakit sebagai pattern. Similaritas ditentukan dari panjang LCS dibagi panjang DNA penyakit.

D. Regular Expression

Regular expression (regex) adalah notasi standar yang mendeskripsikan suatu pola (*pattern*) berupa urutan karakter atau string. Regex digunakan untuk pencocokan string (*string matching*) dengan efisien. Regex sudah menjadi standar yang tersebar di semua tools dan bahasa pemrograman sehingga penting untuk dipelajari. Regular expression menyediakan beberapa *special character* yang dapat digunakan untuk mencocokkan karakter dengan pola-pola tertentu.

a. Wildcard

Simbol titik '.' disebut sebagai wildcard yang cocok dengan satu karakter apapun. Misalnya terdapat regex `/n.p/` maka kata yang cocok dapat berupa `nlp`, `n3p`, dll.

b. Optionality

Simbol tanda tanya ‘?’ menandakan bahwa regular expression yang diberikan sebelum simbol tersebut bersifat opsional. Misalnya terdapat regex `/colou?r/` maka kata yang cocok adalah `colour` dan `color`.

c. Repeatability

Terdapat dua jenis perulangan yang dapat digunakan, yaitu dengan menggunakan simbol ‘+’ dan ‘*’. Simbol ‘+’ menandakan bahwa regex yang diberikan sebelum simbol tersebut dapat diulang. Misalnya terdapat regex `/coo+l/` maka kata yang cocok dapat berupa `cool`, `coool`, `coool`, dst. Sedangkan simbol ‘*’ menandakan bahwa regex yang diberikan sebelum simbol tersebut bersifat opsional atau dapat berulang. Misalnya terdapat regex `/coo*/` maka kata yang cocok dapat berupa `col`, `cool`, `coool`, dst.

d. Choice

Dengan simbol ‘[]’, pola kata yang cocok akan terbatas pada regex yang diberikan di dalam simbol tersebut. Misalnya terdapat regex `/n[a,l,o]p/` maka kata yang cocok adalah `nap`, `nlp`, dan `nop` saja.

e. Range

Simbol ‘-’ digunakan untuk menunjukkan suatu range, misalnya kata yang cocok dari regex `/n[a-z]p/` adalah semua kombinasi huruf dari huruf a-z (`nap`, `nbp`, `nep`, dst).

f. Complementation

Simbol ‘^’ digunakan sebagai negasi. Misalnya regex `/[aiueo]/` adalah regex untuk menghasilkan karakter vocal, maka regex `/^[aiueo]/` adalah regex untuk menghasilkan karakter konsonan saja.

g. Common special symbol

Simbol ‘^’ dan ‘\$’ digunakan untuk mencocokkan awalan dan akhiran dari sebuah baris. Simbol ‘^’ memiliki dua arti, jika simbol tersebut digunakan menjadi sebuah awalan pada class character seperti pada bagian f, maka simbol tersebut berarti negasi, sementara selain itu simbol tersebut menandakan awal dari sebuah baris.

h. Other special character

`\b` : *word boundary (zero width)*
`\d` : *any decimal digit (equivalent to [0-9])*
`\D` : *any non-digit character (equivalent to [^0-9])*
`\s` : *any whitespace character (equivalent to [\t\n\r\f\v])*
`\S` : *any non-whitespace character (equivalent to [^\t\n\r\f\v])*
`\w` : *any alphanumeric character (equivalent to [a-zA-Z0-9_])*
`\W` : *any non-alphanumeric character (equivalent to [^a-zA-Z0-9_])*

E. Deskripsi Singkat Web

Website dibuat dengan pembagian front end menggunakan bahasa pemrograman Javascript dengan framework React Js dan back end menggunakan Node Js dengan framework Express. Pada website yang dibuat, dapat dilihat pemisahan front end dan backend pada repository yang ada. Folder *client* memuat front end website yang dideploy ke layanan Netlify dan folder *server* membuat back end website yang dideploy ke layanan Heroku. Front end dan back end website berkomunikasi melalui API yang memuat database PostgreSQL mengenai tabel daftar penyakit dan hasil tes prediksi penyakit.

BAB III

ANALISIS PEMECAHAN MASALAH

A. Langkah Penyelesaian Masalah Setiap Fitur

1. Fitur Tambahkan Penyakit

- a. Dibuat komponen bernama ‘AddDisease’ di bagian *front end*. Dalam komponen tersebut tersebut terdapat tiga *state* yaitu *diseaseName*, *DNASequence*, dan *fileContent*.
- b. Pada tampilan, terdapat 1 buah form yang akan menerima masukan nama penyakit, 1 buah tombol *upload file* yang dapat mengunggah file teks berisikan barisan DNA, dan 1 buah tombol *submit* untuk mengirimkan masukan nama dan barisan DNA penyakit.
- c. Dilakukan validasi masukan. Beberapa validasi yang kami buat yaitu masukan nama penyakit harus unik dan barisan DNA dilakukan sanitasi terlebih dahulu. Apabila barisan DNA tidak valid atau nama penyakit sudah ada di database, maka akan terdapat peringatan untuk memasukkan ulang nama penyakit serta barisan DNA.
- d. Data yang dimasukkan akan dikirimkan ke backend melalui POST Request. Backend akan menerima POST Request tersebut.
- e. Dijalankan SQL Query yang memasukkan nama dan barisan DNA penyakit ke dalam database PostgreSQL.

2. Fitur Tes Prediksi DNA

- a. Dibuat komponen bernama ‘TestDNA’ di bagian *front end*. Dalam komponen tersebut tersebut terdapat *state* *enteredDisease*, *enteredUsername*, *enteredDNASequence*, *similarity*, *isInfected*, *date*, *fileContent*, *stringMatcher*, dan *showTestResult*.
- b. Pada *interface*, terdapat sebuah form untuk menerima masukan nama pengguna, DNA sequence pengguna dan nama penyakit yang ingin diuji. Karena pada tugas besar ini diharuskan untuk mengimplementasikan kedua algoritma KMP dan Boyer-Moore, maka pilihan algoritma *string matching* dipilih melalui radio button. Terdapat sebuah tombol submit yang akan menjalankan fungsi di sisi *backend* dan kemudian menampilkan hasil test.
- c. Dilakukan validasi masukan. Nama pengguna dan penyakit tidak boleh kosong. DNA sequence hanya boleh mengandung A, C, G, T. Jik penyakit yang ingin dites tidak terdapat di database, maka akan muncul peringatan.
- d. Data yang dimasukkan akan dikirimkan ke backend melalui POST Request. Backend akan menerima POST Request tersebut.
- e. Dijalankan SQL Query yang memasukkan nama dan barisan DNA penyakit ke dalam database PostgreSQL.

3. Fitur Pencarian Hasil Tes DNA

- a. Dibuat komponen Bernama 'FindResult' di bagian *front end*. Dalam komponen tersebut, terdapat 2 *state* yaitu *searchTerm* dan *searchResult*.
- b. Pada *interface*, terdapat sebuah *field* untuk menerima masukan input query dengan format <tanggal_prediksi><spasi><nama_penyakit> atau <nama_penyakit><spasi><tanggal_prediksi> atau <tanggal_prediksi> atau <nama_penyakit>.
- c. Data yang dimasukkan akan dikirimkan ke backend sebagai query parameter melalui GET Request. Backend akan menerima GET Request tersebut.
- d. Dilakukan pengecekan dengan regular expression untuk mengidentifikasi tanggal_prediksi dan/atau nama_penyakit berdasarkan masukan.
- e. Dijalankan SQL Query yang mencari hasil prediksi berdasarkan tanggal_prediksi dan/atau nama_penyakitnya.
- f. Hasil pencarian akan diterima oleh front end dan akan ditampilkan hasilnya.

B. Fitur Fungsional dan Arsitektur Aplikasi Web

1. Fitur Fungsional

- a. Fitur Tambahkan Penyakit
Menambahkan daftar penyakit beserta barisan DNA. Barisan DNA dimasukkan melalui *input file* berformat *.txt*.
- b. Fitur Tes Prediksi DNA
Menampilkan hasil tes prediksi DNA pengguna terhadap suatu penyakit. Masukan berupa nama pengguna, barisan DNA melalui *input file* berformat *.txt*, prediksi nama penyakit, serta metode string matching yang digunakan.
- c. Fitur Pencarian Hasil Tes DNA
Mencari hasil tes DNA yang tersedia pada hasil prediksi DNA
- d. Fitur Menampilkan Isi Database
Terdapat tombol *Show Database* yang berada di paling bawah tampilan. Jika diklik, maka akan muncul basis data yang digunakan pada website ini.
- e. Advanced Mode
Terdapat slider *On/Off* pada navigasi bar yang terletak di paling atas tampilan website. Jika diaktifkan maka barisan DNA dapat dimasukkan melalui ketikan pada form. (memudahkan saat debugging).

2. Arsitektur

Arsitektur yang digunakan dalam front end website adalah framework React dalam bahasa pemrograman Javascript. Sedangkan untuk arsitektur backend digunakan library Node.js dengan framework Express. Arsitektur ini dipilih karena kami sudah familiar dalam penggunaannya. Pada front end, setiap fitur fungsional memiliki komponennya masing-masing (dapat dilihat di direktori *Tubes3_13520002/src/client/src/components/*). Kemudian pada backend, terdapat beberapa API endpoints yang digunakan untuk post/get request sebagai berikut:

API Endpoint:

- [/api/diseases-list GET](#)
- [/api/insert-diseases-list POST](#)
- [/api/test-result GET](#)
- [/api/insert-test-result POST](#)

Jika backend dijalankan dari heroku, maka contoh salah satu API endpoint yaitu <https://dna-tester.herokuapp.com/api/diseases-list>. API endpoint ini digunakan sebagai alat komunikasi penghubung antara front end dengan back end. Dengan ini, terdapat perbedaan yang jelas fungsionalitas antara back end dengan front end.

API endpoint yang tersedia memuat data berformat JSON yang berisikan isi dari basis data yang digunakan. Beberapa query yang digunakan untuk dimuat ke dalam API endpoint yaitu sebagai berikut.

```
"INSERT INTO test_result (dates, username, disease, dna_sequence, similarity, isInfected) VALUES ($1, $2, $3, $4, $5, $6)", [dates, username, disease, dna_sequence, similarity, isInfected],
```

```
"SELECT * FROM diseases"
```

```
"SELECT * FROM test_result"
```

```
"SELECT dna_sequence FROM diseases WHERE disease_name = $1", [disease_name]
```

Query SQL lebih lengkap terdapat pada bab IV.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

C. Spesifikasi Teknis Program

1. Struktur Data.

- a. DiseaseList
 - disease_name : nama penyakit
 - dna_sequence : sequence DNA penyakit
- b. TestResult
 - username : nama pengguna
 - dna_sequence : sequence DNA pengguna
 - disease : nama penyakit yang ingin diprediksi
 - dates : tanggal uji
 - similarity : tingkat kemiripan DNA pengguna dengan DNA penyakit
 - isInfected : status apakah pengguna terinfeksi penyakit tersebut

2. Fungsi dan Prosedur

a. String Matching

kmpMatch(text, pattern)	Implementasi algoritma KMP, mereturn posisi index pertama pattern pada text jika ditemukan (<i>match</i>), -1 jika tidak
computeFail(pattern)	<i>Border function</i> untuk algoritma KMP
bmMatch(text, pattern)	Implementasi algoritma Boyer-Moore, mereturn posisi index pertama pattern pada text jika ditemukan (<i>match</i>), -1 jika tidak
buildLast(pattern)	<i>Last occurrence function</i> untuk algoritma Boyer-Moore
isInfected(text, pattern)	Melakukan pencocokan dengan algoritma yang diinginkan (KMP atau BM). Jika cocok, akan mereturn 1. Jika tidak, maka dicek similaritasnya. Jika similaritas $\geq 80\%$, maka akan return 1, 0 jika tidak.

b. LCS (Longest Common Subsequence)

lcs(text, pattern)	Mereturn panjang <i>longest common subsequence</i> antara text dan pattern
rateLCS(text, pattern)	Mereturn <i>rate</i> similaritas text dan pattern

c. Query

Fungsi dan prosedur dibawah mengimplementasikan Query SQL untuk menambahkan data ataupun mendapatkan data yang terhubung dengan API.

getDiseasesList()	"SELECT * FROM diseases"
-------------------	--------------------------

	Memberikan daftar penyakit beserta DNA squence nya
insertDiseasesList()	<pre>"INSERT INTO diseases (disease_name, dna_sequence) VALUES (\$1, \$2)", [disease_name, dna_sequence]</pre> Menambahkan penyakit baru
getDiseasesDNASequence (diseaseName)	<pre>"SELECT dna_sequence FROM diseases WHERE disease_name = \$1", [disease_name]</pre> Mereturn DNA sequence dari sebuah penyakit
insertTestResult()	<pre>"INSERT INTO test_result (dates, username, disease, dna_sequence, similarity, isInfected) VALUES (\$1, \$2, \$3, \$4, \$5, \$6)", [dates, username, disease, dna_sequence, similarity, isInfected]</pre> Menambahkan hasil test baru
getTestRestult()	<pre>"SELECT * FROM test_result"</pre> Membeikan daftar hasil test
searchTestResult()	<pre>SELECT * FROM test_result WHERE dates = \$1", [dateSearch]</pre> Mengembalikan data hasil test pada tanggal tertentu <pre>SELECT * FROM test_result WHERE dates = \$1 AND disease LIKE \$2", [dateSearch, nameSearch+'%']</pre> Mengembalikan data hasil test pada tanggal tertentu dengan nama penyakit tertentu <pre>SELECT * FROM test_result WHERE disease LIKE \$1", [nameSearch+'%']</pre> Mengembalikan data hasil test dengan nama penyakit tertentu

d. Regular Expression untuk sanitasi input

isValidDNASequence(dnaSequence)	<p>Mereturn <i>true</i> jika dnaSequence yang diinput valid (hanya mengandung A, C, G, T). <i>False</i> jika tidak.</p> <p>Regex: <code>/^[ACGT]+\$</code></p>
---------------------------------	--

e. Regular Expression untuk menampilkan riwayat test

regexSearchTerm(str)	<p>Mengembalikan tanggal dan nama penyakit yang akan dicari serta metode pencarian yang akan dilakukan.</p> <p>Regex:</p> <ul style="list-style-type: none"> - Hanya tanggal <p><code>/^\d{1,2}(\ -)\d{1,2}(\ -)\d{4}\$</code></p>
----------------------	---

	<p>/^d{1,2}s((j J)an(?:uary)? (f F)eb(?:ruary)? (m M)ar(?:ch)? (a A)pr(?:il)? (May may) (j J)un(?:e)? (j J)ul(?:y)? (a A)ug(?:ust)? (s S)ep(?:t)?(?:tember)? (o O)ct(?:ober)? (n N)ov(?:ember)? (d D)ec(?:ember)?)+s(d{4})\$/</p> <p>/^d{4}(\\ -)d{1,2}(\\ -)d{1,2}\$/</p> <p>/^d{4}s((j J)an(?:uary)? (f F)eb(?:ruary)? (m M)ar(?:ch)? (a A)pr(?:il)? (May may) (j J)un(?:e)? (j J)ul(?:y)? (a A)ug(?:ust)? (s S)ep(?:t)?(?:tember)? (o O)ct(?:ober)? (n N)ov(?:ember)? (d D)ec(?:ember)?)+s(d{1,2})\$/</p> <p>- Tanggal dan nama penyakit</p> <p>/^d{1,2}(\\ -)d{1,2}(\\ -)d{4}s[s\\S]+\$/\$</p> <p>/^d{1,2}s((j J)an(?:uary)? (f F)eb(?:ruary)? (m M)ar(?:ch)? (a A)pr(?:il)? (May may) (j J)un(?:e)? (j J)ul(?:y)? (a A)ug(?:ust)? (s S)ep(?:t)?(?:tember)? (o O)ct(?:ober)? (n N)ov(?:ember)? (d D)ec(?:ember)?)+s(d{4})s[s\\S]+\$/\$</p> <p>/^d{4}(\\ -)d{1,2}(\\ -)d{1,2}s[s\\S]+\$/\$</p> <p>/^d{4}s((j J)an(?:uary)? (f F)eb(?:ruary)? (m M)ar(?:ch)? (a A)pr(?:il)? (May may) (j J)un(?:e)? (j J)ul(?:y)? (a A)ug(?:ust)? (s S)ep(?:t)?(?:tember)? (o O)ct(?:ober)? (n N)ov(?:ember)? (d D)ec(?:ember)?)+s(d{1,2})s[s\\S]+\$/\$</p> <p>/^[s\\S]+s(d{1,2}(\\ -)d{1,2}(\\ -)d{4})\$/</p> <p>/^[s\\S]+s(d{1,2}s((j J)an(?:uary)? (f F)eb(?:ruary)? (m M)ar(?:ch)? (a A)pr(?:il)? (May may) (j J)un(?:e)? (j J)ul(?:y)? (a A)ug(?:ust)? (s S)ep(?:t)?(?:tember)? (o O)ct(?:ober)? (n N)ov(?:ember)? (d D)ec(?:ember)?)+s(d{4})\$/</p> <p>/^[s\\S]+s(d{4}(\\ -)d{1,2}(\\ -)d{1,2})\$/</p> <p>/^[s\\S]+s(d{4}s((j J)an(?:uary)? (f F)eb(?:ruary)? (m M)ar(?:ch)? (a A)pr(?:il)? (May may) (j J)un(?:e)? (j J)ul(?:y)? (a A)ug(?:ust)? (s S)ep(?:t)?(?:tember)? (o O)ct(?:ober)? (n N)ov(?:ember)? (d D)ec(?:ember)?)+s(d{1,2})\$/</p> <p>- Hanya nama penyakit</p> <p>/^[s\\S]+\$/\$</p>
--	--

D. Tata Cara Penggunaan Program

Terdapat dua cara untuk menjalankan program. Cara pertama yaitu dengan membuka website secara langsung yang memuat halaman front end program. Konfigurasi default program yaitu backend yang dijalankan di Cloud Platform Heroku. Database yang digunakan juga disimpan di layanan tersebut. Front end dijalankan secara otomatis pada layanan Netlify. Kode sumber pada front end sudah secara otomatis dihubungkan dengan alamat back end. Akan tetapi jika ingin menjalankan program secara offline, maka dapat membaca langkah-langkah yang ada di bawah ini.

Menjalankan Website yang Sudah Di-deploy

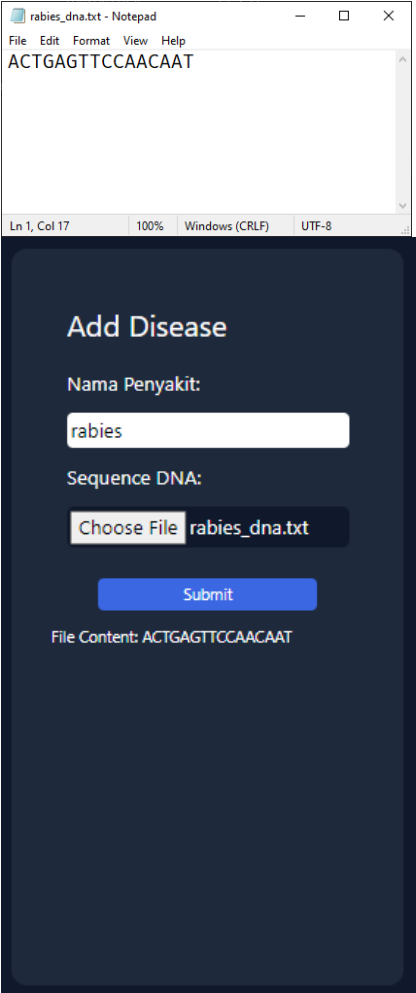
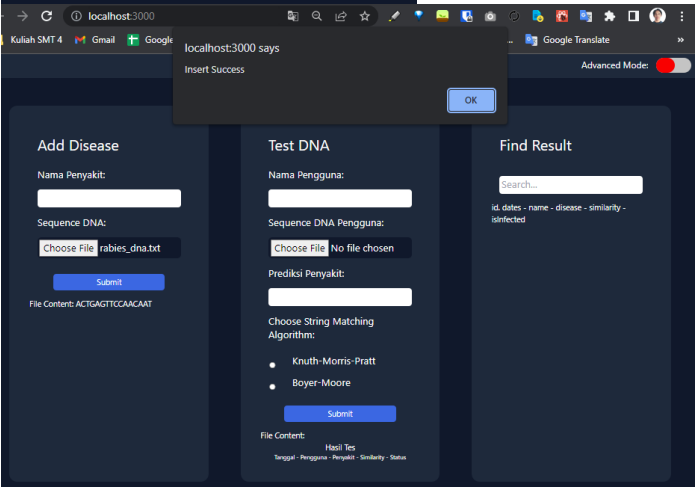
1. Buka <https://dna-tester.netlify.app/> untuk melihat tampilan front end. Semua interaksi terhadap fungsionalitas website dapat dilakukan di alamat ini.
2. Buka <https://dna-tester.herokuapp.com/> jika ingin melihat melihat API yang ada di back end. Halaman ini memuat daftar alamat API yang digunakan. API digunakan untuk memuat data yang ada pada database Postgre SQL serta sebagai komunikasi antara front end dengan back end. Karena keterbatasan pengetahuan dan penggunaan layanan gratis, maka terdapat kasus random yang menyebabkan back end menjadi crash saat digunakan. Jika hal ini terjadi, maka website harus dijalankan secara lokal.

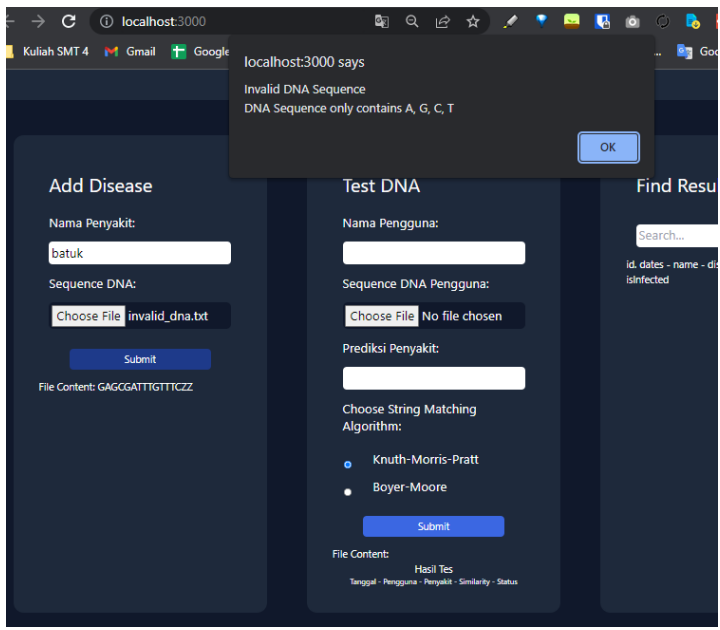
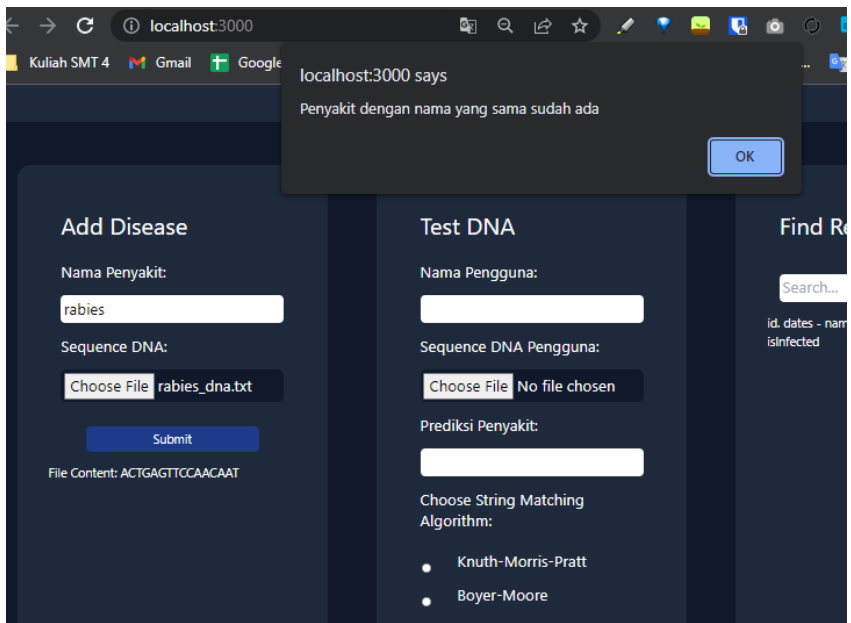
Menjalankan Website secara Lokal

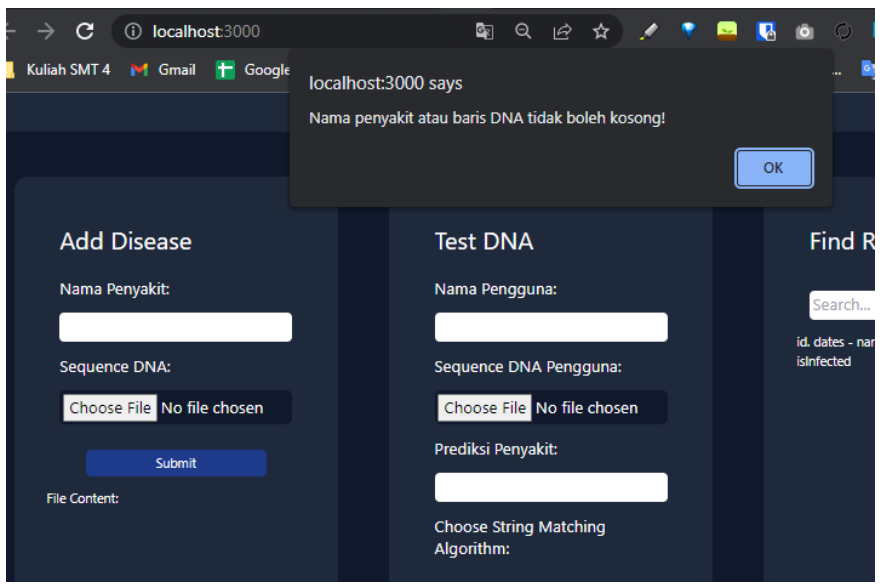
1. *Clone repository*, https://github.com/Putriliza/Tucil3_13520066.git
2. Buka 2 terminal di direktori folder hasil *clone repository*
3. Di terminal pertama jalankan perintah `'cd src/server'` `'npm install'` `'nodemon start'`. Setelah itu buka <http://localhost:3001/> di browser untuk melihat daftar API yang terdapat pada back end website. Back end akan dijalankan pada port 3001.
4. Di terminal kedua, jalankan perintah `'cd src/client'` `'npm install'` `'npm start'`. Secara otomatis browser akan membuka alamat website yang merupakan bagian dari front end. Jika website tidak otomatis dijalankan, maka buka <http://localhost:3000/> pada browser. Front end akan dijalankan pada port 3000.

E. Hasil Pengujian

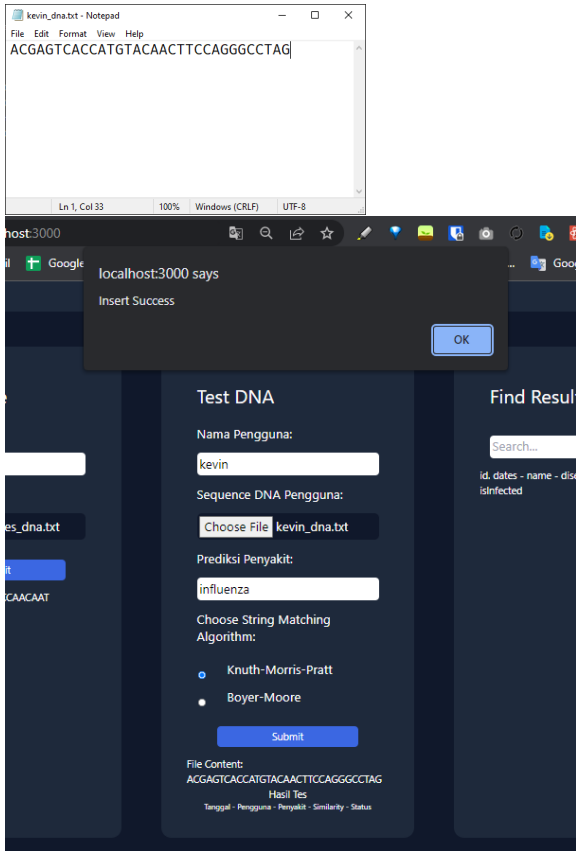
1. Fitur Tambahkan Penyakit

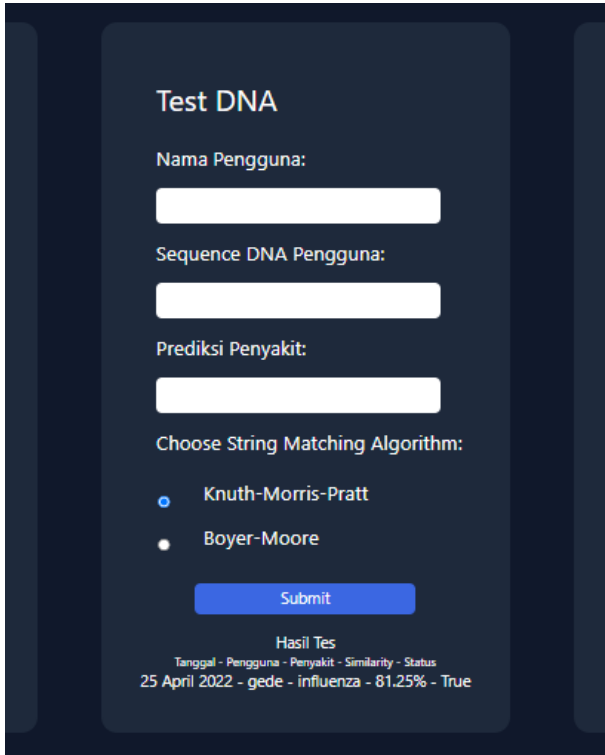
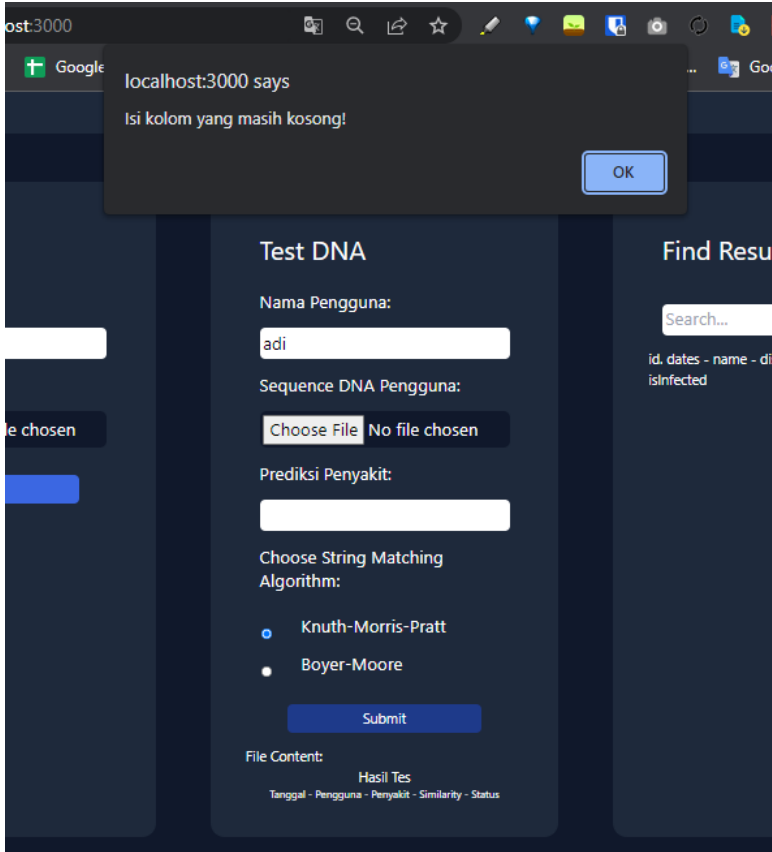
Kasus	Input/Output
Berhasil	 

<p>DNA sequence tidak valid</p>	 <p>The screenshot shows the 'Add Disease' form in a web application. The 'Nama Penyakit' field contains 'batuk'. The 'Sequence DNA' field contains 'invalid_dna.txt'. A modal dialog box is displayed over the form, stating 'localhost:3000 says Invalid DNA Sequence DNA Sequence only contains A, G, C, T'. The 'File Content' field shows 'GAGGATTGTTTCZZ'.</p>
<p>Sudah terdapat data dengan nama penyakit yang sama</p>	 <p>The screenshot shows the 'Add Disease' form in a web application. The 'Nama Penyakit' field contains 'rabies'. The 'Sequence DNA' field contains 'rabies_dna.txt'. A modal dialog box is displayed over the form, stating 'localhost:3000 says Penyakit dengan nama yang sama sudah ada'. The 'File Content' field shows 'ACTGAGTTCCAACAAT'.</p>

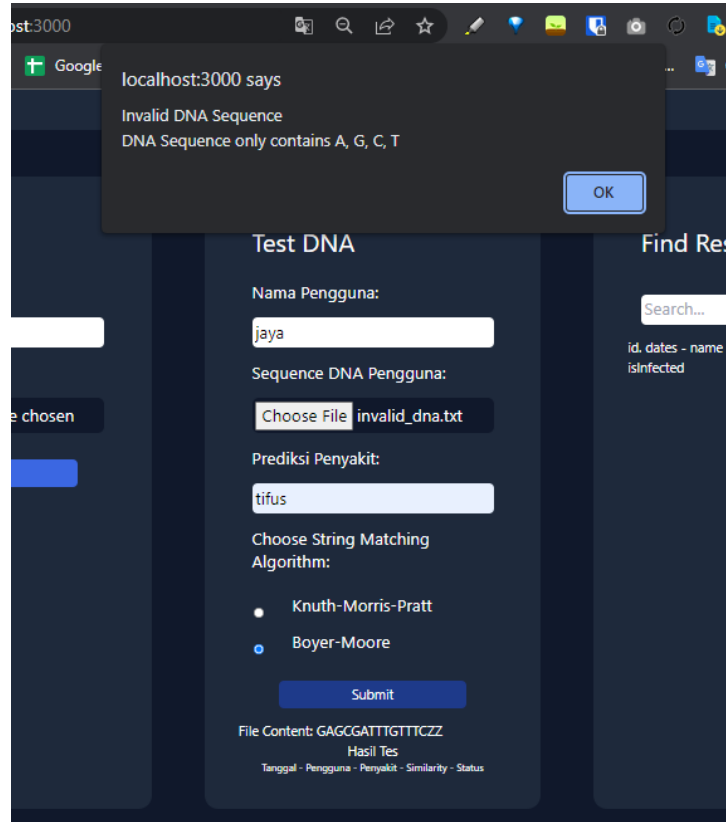
Form kosong	
-------------	--

2. Fitur Prediksi Penyakit

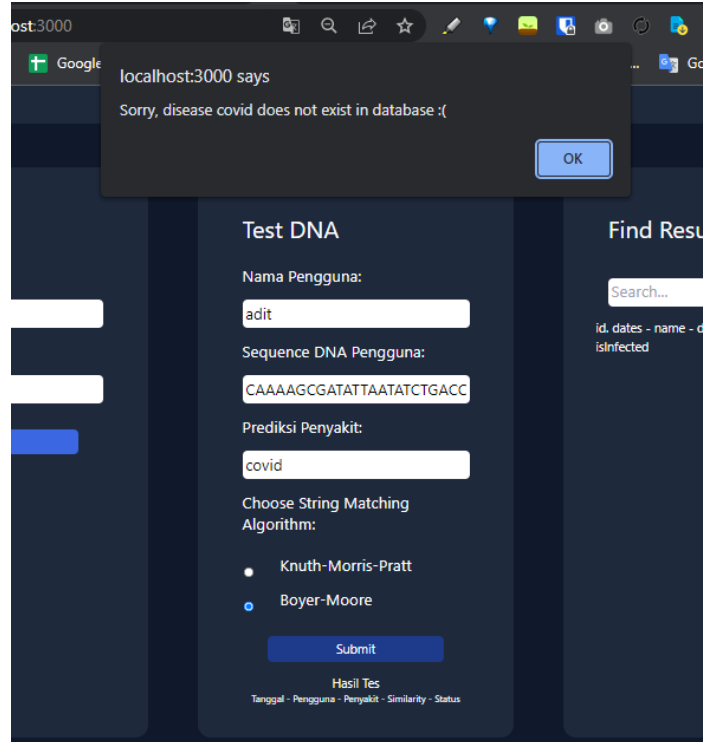
Kasus	Input/Output
Berhasil	

	 <p>The screenshot shows a web application titled "Test DNA". It contains four input fields: "Nama Pengguna:" (filled with "gede"), "Sequence DNA Pengguna:" (filled with "influenza"), "Prediksi Penyakit:" (filled with "81.25%"), and "Choose String Matching Algorithm:" (with "Knuth-Morris-Pratt" selected). A blue "Submit" button is at the bottom. Below the button, the text "Hasil Tes" is displayed, followed by a table header "Tanggal - Pengguna - Penyakit - Similarity - Status" and a single row of data: "25 April 2022 - gede - influenza - 81.25% - True".</p>
Form kosong	 <p>The screenshot shows the same "Test DNA" form but with empty input fields. The "Nama Pengguna:" field contains "adi". The "Sequence DNA Pengguna:" field has a "Choose File" button and the text "No file chosen". The "Prediksi Penyakit:" field is empty. The "Choose String Matching Algorithm:" section has "Knuth-Morris-Pratt" selected. A blue "Submit" button is present. Below the button, the text "File Content:" is displayed, followed by "Hasil Tes" and the same table header as above. An error message dialog box is overlaid on the form, stating "localhost:3000 says" and "Isi kolom yang masih kosong!" (Fill the columns that are still empty!). The dialog has an "OK" button.</p>

DNA Sequence tidak valid

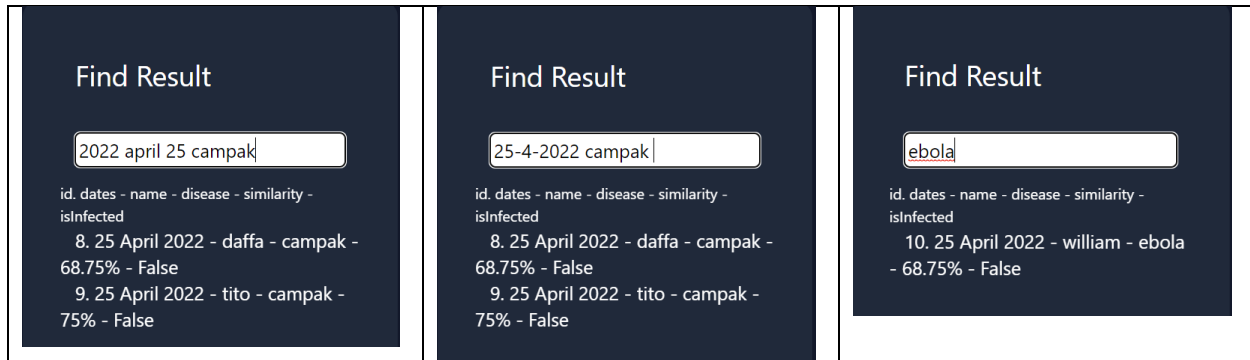


Memprediksi penyakit yang belum ada di database



3. Fitur Pencarian Hasil Tes DNA

<p>Format dd/mm/yyyy</p> <p>Find Result</p> <p>25/04/2022</p> <p>id. dates - name - disease - similarity - isInfected</p> <p>1. 25 April 2022 - kevin - influenza - 81.25% - False</p> <p>2. 25 April 2022 - adit - influenza - 87.5% - False</p> <p>3. 25 April 2022 - gede - influenza - 81.25% - False</p> <p>4. 25 April 2022 - lisna - tifus - 56.25% - False</p> <p>5. 25 April 2022 - daffa - cacar air - 37.5% - False</p> <p>6. 25 April 2022 - tito - tifus - 100% - False</p> <p>7. 25 April 2022 - girvin - tifus - 75% - False</p> <p>8. 25 April 2022 - daffa - campak - 68.75% - False</p> <p>9. 25 April 2022 - tito - campak - 75% - False</p> <p>10. 25 April 2022 - william - ebola - 68.75% - False</p>	<p>Format dd-mm-yyyy</p> <p>Find Result</p> <p>25-04-2022</p> <p>id. dates - name - disease - similarity - isInfected</p> <p>1. 25 April 2022 - kevin - influenza - 81.25% - False</p> <p>2. 25 April 2022 - adit - influenza - 87.5% - False</p> <p>3. 25 April 2022 - gede - influenza - 81.25% - False</p> <p>4. 25 April 2022 - lisna - tifus - 56.25% - False</p> <p>5. 25 April 2022 - daffa - cacar air - 37.5% - False</p> <p>6. 25 April 2022 - tito - tifus - 100% - False</p> <p>7. 25 April 2022 - girvin - tifus - 75% - False</p> <p>8. 25 April 2022 - daffa - campak - 68.75% - False</p> <p>9. 25 April 2022 - tito - campak - 75% - False</p> <p>10. 25 April 2022 - william - ebola - 68.75% - False</p>	<p>Format tanggal bulan tahun</p> <p>Find Result</p> <p>25 april 2022</p> <p>id. dates - name - disease - similarity - isInfected</p> <p>1. 25 April 2022 - kevin - influenza - 81.25% - False</p> <p>2. 25 April 2022 - adit - influenza - 87.5% - False</p> <p>3. 25 April 2022 - gede - influenza - 81.25% - False</p> <p>4. 25 April 2022 - lisna - tifus - 56.25% - False</p> <p>5. 25 April 2022 - daffa - cacar air - 37.5% - False</p> <p>6. 25 April 2022 - tito - tifus - 100% - False</p> <p>7. 25 April 2022 - girvin - tifus - 75% - False</p> <p>8. 25 April 2022 - daffa - campak - 68.75% - False</p> <p>9. 25 April 2022 - tito - campak - 75% - False</p> <p>10. 25 April 2022 - william - ebola - 68.75% - False</p>
<p>Format yyyy/mm/dd</p> <p>Find Result</p> <p>2021/1/1</p> <p>id. dates - name - disease - similarity - isInfected</p> <p>11. 1 January 2021 - gregous - cacar air - 61% - False</p>	<p>Format yyyy-mm-dd</p> <p>Find Result</p> <p>2021-1-1</p> <p>id. dates - name - disease - similarity - isInfected</p> <p>11. 1 January 2021 - gregous - cacar air - 61% - False</p>	<p>Format tahun bulan tanggal</p> <p>Find Result</p> <p>2021 jan 1</p> <p>id. dates - name - disease - similarity - isInfected</p> <p>11. 1 January 2021 - gregous - cacar air - 61% - False</p>
<p>Format namaPenyakit dd/mm/yyyy</p> <p>Find Result</p> <p>influenza 25/4/2022</p> <p>id. dates - name - disease - similarity - isInfected</p> <p>1. 25 April 2022 - kevin - influenza - 81.25% - False</p> <p>2. 25 April 2022 - adit - influenza - 87.5% - False</p> <p>3. 25 April 2022 - gede - influenza - 81.25% - False</p>	<p>Format namaPenyakit tanggal bulan tahun</p> <p>Find Result</p> <p>influenza 25 apr 2022</p> <p>id. dates - name - disease - similarity - isInfected</p> <p>1. 25 April 2022 - kevin - influenza - 81.25% - False</p> <p>2. 25 April 2022 - adit - influenza - 87.5% - False</p> <p>3. 25 April 2022 - gede - influenza - 81.25% - False</p>	<p>Format namaPenyakit tahun bulan tanggal</p> <p>Find Result</p> <p>cacar air 2022 april 25</p> <p>id. dates - name - disease - similarity - isInfected</p> <p>5. 25 April 2022 - daffa - cacar air - 37.5% - False</p>
<p>Format tahun bulan tanggal nama_penyakit</p>	<p>Format dd-mm-yyyy nama_penyakit</p>	<p>Format namaPenyakit</p>



F. Analisis Hasil Pengujian

Uji kasus dilakukan dengan panjang DNA pengguna sejumlah 16 karakter dan panjang DNA penyakit sejumlah 32 karakter. Uji kasus dilakukan dengan 11 data pengguna beserta penyakit yang dimiliki. Didapat 4 pengguna memiliki tingkat kemiripan DNA dengan DNA penyakit lebih dari sama dengan 80% sehingga pengguna tersebut dinyatakan mengidap penyakit yang diuji.

Exact matching dilakukan dengan algoritma Knuth-Morris-Pratt atau algoritma Boyer-Moore. Kedua algoritma ini memberikan hasil yang persis sama. Jika *string matching* ini gagal, akan dihitung similaritasnya. Dari percobaan yang telah dilakukan, panjang DNA pengguna dan panjang DNA penyakit berpengaruh pada persentase kemiripan yang dihitung dengan algoritma LCS. Algoritma LCS mengkalkulasikan *subsequence* dimana urutan huruf yang muncul pada DNA pengguna dan DNA penyakit dalam urutan relatif yang sama, tetapi tidak harus tepat bersebelahan. DNA sequence hanya mengandung A, C, G, atau T. Sehingga, jika DNA pengguna dan DNA penyakit cukup acak, maka semakin panjang DNA pengguna dibanding DNA penyakit, tingkat similaritas keduanya pun dapat semakin tinggi.

Pada fitur pencarian hasil tes DNA, pengguna dapat memberi masukan berupa <tanggal_prediksi> <spasi> <nama_penakit> atau <nama_penakit> <spasi> <tanggal_prediksi> atau <tanggal_prediksi> atau <nama_penakit>. Terdapat 6 format tanggal yang dapat diterima, yaitu dd-mm-yyyy, dd/mm/yyyy, dd month yyyy, yyyy-mm-dd, yyyy/mm/dd, dan yyyy month dd. Oleh karena itu, terdapat total 19 kombinasi format masukan yang dapat diterima. Pada implementasinya, regular expression untuk format tanggal dd-mm-yyyy dan dd/mm/yyyy digabung menjadi satu sehingga terdapat sebanyak 13 regular expression yang dibuat untuk mengolah masukan pencarian dari pengguna. Regular expression juga berhasil diimplementasikan untuk sanitasi input.

Secara keseluruhan, website yang kami dibuat sudah memenuhi spesifikasi dari tugas besar 3 dengan tiga fitur utama yaitu menambahkan baris DNA suatu penyakit, melakukan tes prediksi penyakit, dan mencari hasil tes penyakit.

BAB V

KESIMPULAN, SARAN, DAN REFLEKSI

Beberapa algoritma yang populer untuk mengimplementasikan string matching adalah Algoritma Knuth-Morris-Pratt dan Boyer-Moore. Pada Tugas Besar II Strategi Algoritma 2022 ini, telah diimplementasikan String Matching dan Regular Expression pada DNA pattern matching. Tiga fitur utama dari web yang dibangun adalah menambahkan penyakit baru, melakukan pengujian DNA pengguna terhadap suatu penyakit, serta melihat hasil test yang disimpan dalam database. Web diimplementasikan dengan bahasa pemrograman Node.js (Backend) dan React (Frontend). Berdasar pengujian yang telah dilakukan, solusi yang kami buat memberikan hasil yang baik dan memenuhi seluruh spesifikasi tugas. Sebagai saran, tujuan dari tugas ini sebenarnya dapat dicapai dengan komputasi yang lebih sederhana. Contohnya, algoritma yang digunakan untuk mendapatkan tingkat similaritas juga mencakup *exact matching* jika hasilnya 100% sehingga tidak diperlukan dua algoritma *string matching* yang memberikan hasil yang sama.

Dalam pengerjaan tugas besar ini, kami menyadari segala kendala dan keterbatasan pada program yang kami buat. Untuk itu, kami mengharapkan kritik dan saran dari pihak-pihak yang turut mengevaluasi program maupun laporan ini. Tentunya, Web ini kedepannya dapat dikembangkan dengan fitur fungsional yang lebih lengkap dan UI/UX yang lebih baik.

LAMPIRAN

Repository GitHub:

https://github.com/ranjabi/Tubes3_13520002

Video Kelompok:

<https://youtu.be/ie0cpFhRJm4>

DAFTAR PUSTAKA

Munir, Rinaldi (2021). *Pencocokan String String/Pattern Matching*. Diakses pada 17 April 2022, dari <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf>

Khodra, Masayu L (2019). *String Matching dengan Regular Expression*. Diakses pada 17 April 2022, dari <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf>

Alsmadi, Izzat. Nuser, Maryam (2012). *String Matching Evaluation Methods for DNA Comparison*, International Journal of Advanced Science and Technology, Vol. 47, October, 2012

Bird, S., & Klein, E. (2006). *Regular expressions for natural language processing*. University of Pennsylvania.