

Tugas Besar 3 IF2211 Strategi Algoritma

Penerapan String Matching dan Regular Expression dalam DNA Pattern Matching



Disusun oleh:

fansberatCASSETTETAPE

13520075 - Samuel Christopher Swandi

13520081 - Andhika Arta Aryanto

13520083 - Sarah Azka Arief

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2022

DAFTAR ISI

BAB 1	3
DESKRIPSI TUGAS	3
BAB 2	4
LANDASAN TEORI	4
KMP	4
BM	6
Regex	7
Aplikasi Web	7
BAB 3	9
ANALISIS PEMECAHAN MASALAH	9
Langkah-langkah Pemecahan Masalah tiap Fitur	9
Fitur Fungsional dan Arsitektur Aplikasi Web yang Dibangun	12
BAB 4	17
IMPLEMENTASI DAN PENGUJIAN	17
Spesifikasi Teknis Program	17
Tata Cara Penggunaan Program	21
Hasil Pengujian	24
Analisis	31
BAB 5	33
KESIMPULAN DAN SARAN	33
Kesimpulan	33
Saran	33
DAFTAR PUSTAKA	35

BAB 1

DESKRIPSI TUGAS

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

DNA Pattern Matching ini merupakan bagian dari salah satu jenis tes DNA yaitu DNA *sequence analysis*. Tes ini dilakukan dengan mengambil suatu sekuens DNA, contoh : ATTCGTAACTAGTAAGTTA dan dilakukan String Matching dengan data penyakit untuk membantu penyedia jasa kesehatan dalam memprediksi penyakit pasien.

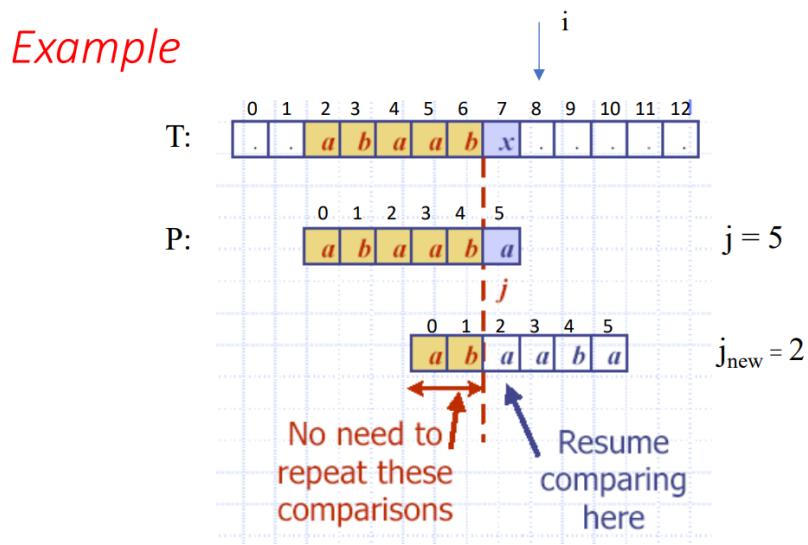
BAB 2

LANDASAN TEORI

2.1 KMP

Knuth-Morris-Pratt (KMP) algorithm adalah salah satu algoritma pattern matching yang melakukan pencarian pattern pada suatu *text* dari kiri ke kanan (mirip algoritma *Brute Force*), namun algoritma ini melakukan pemindahan pattern lebih baik daripada *brute force*.

Algoritma ini dimulai dengan pertanyaan : misal terdapat suatu text T dan pattern P dan ditemukan perbedaan pada $P[j] \neq T[i]$, bagaimana cara kita memindahkan pattern untuk menghindari perbandingan yang sia - sia. Dari sini didapat bahwa jawaban dari pertanyaan ini adalah prefiks terbesar $P[0..j-1]$ yang juga merupakan sufiks dari $P[1..j-1]$.



22

Gambar 2.1.1 Komponen dalam Pohon Berakar

Bila dilihat dari gambar di atas, misal pattern abaab, prefiks terbesar yang sekaligus sufiks adalah ab. Berarti panjang sebesar 2 dan bisa dilihat jumlah pergeserannya bukan mulai dari 0, namun dimulai lagi dari 2 (jumlah pergeseran = panjang pattern sampai j - panjang prefiks terpanjang = 3). Fungsi untuk mencari besar hal ini dinamakan *KMP Border Function*. KMP melakukan *preprocessing* pada pattern untuk menemukan semua prefiks terpanjang dalam

pattern untuk tiap posisi. Misal j adalah posisi terjadinya ketidaksamaan dan k adalah posisi sebelum ketidaksamaan ($k = j-1$) , $b(k)$ adalah panjang prefiks terpanjang dari $P[0..k]$ yang juga merupakan sufiks dari $P[1..k]$. Berikut contoh lebih jelas :

Border Function Example

➤ P: abaaba
j: 012345

j	0	1	2	3	4	5
P[j]	a	b	a	a	b	a
k	0	1	2	3	4	
b(k)	0	0	1	1	2	

$b(k)$ is the size of the largest border.

➤ In code, $b()$ is represented by an array, like the table.

Hint: The border function $b(k)$ is defined as the size of the largest prefix of $P[0..k]$ that is also a suffix of $P[1..k]$.

Gambar 2.1.2 Contoh Border Function

Lihat pada $b(4)$ bernilai 2 karena pada posisi itu prefiks terpanjang yang juga merupakan sufiks adalah pada pattern “abaab” yaitu “ab”, dengan panjang 2, dan berikut seterusnya.

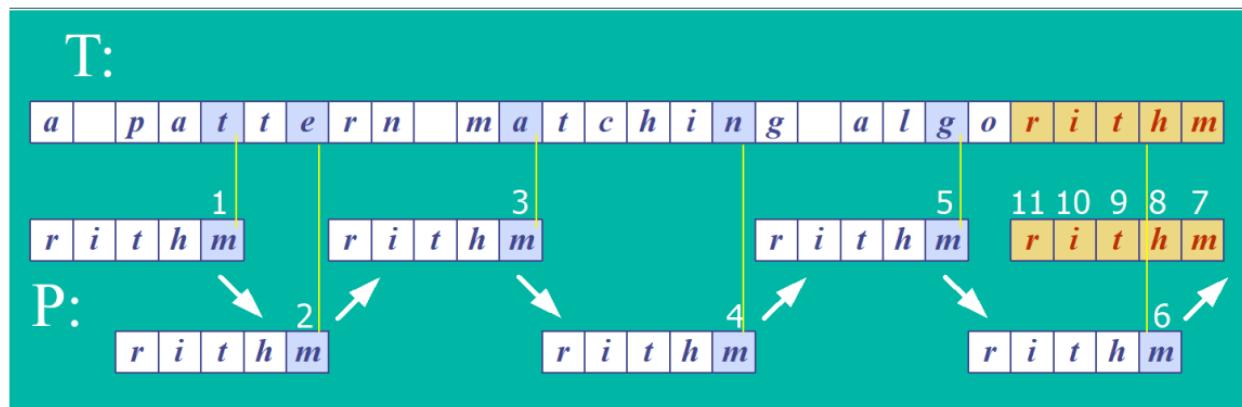
Jadi, pada intinya KMP mirip dengan *brute force*, namun terdapat *pre-processing* dan teknik pergeseran yang membuat dirinya lebih efisien dan lebih cepat. Kompleksitas waktu dari algoritma ini adalah $O(m+n)$. Kelebihan lain dari KMP adalah algoritma tidak akan perlu mundur di text input , T, jadi hal ini membuat algoritma sangat cocok untuk memproses file yang sangat besar. KMP juga memiliki kekurangan yaitu pada alfabet yang besar, hal ini karena apabila alfabet besar, kemungkinan ketidaksamaan seiring terjadi dan kemungkinan panjang dari prefiks juga lebih kecil. Sehingga akan sering terjadi pergeseran yang mungkin hanya menggeser 1.

2.2 BM

Boyer-Moore merupakan algoritma *pattern matching* berdasarkan dua teknik, yaitu *looking-glass technique* yaitu untuk pattern P mencari di T dari akhir dan digeser mundur serta *character-jump technique* berikut penjelasan lebih lanjut :

- Apabila terjadi ketidaksamaan pada $T[i]$ misal $T[i]$ adalah x dan $P[j]$ adalah b ada 3 kasus yang mungkin
 - Kasus 1 adalah saat P memiliki huruf x di suatu tempat, pada kasus ini, P akan dipindahkan ke kanan agar meluruskan kemunculan terakhir x pada P di $T[i]$.
 - Kasus 2 adalah P memiliki huruf x , namun tidak bisa dipindahkan ke kanan (misal karena x berada pada posisi pattern sekarang), pada kasus ini P hanya digeser ke kanan sebesar 1
 - Kasus 3 adalah saat kasus 1 dan 2 tidak bisa, jadi geser P untuk meluruskan $P[0]$ dengan $T[i+1]$

Berikut contoh gambar :



Gambar 2.2.1 Contoh Boyer Moore Algorithm

Sebelumnya sudah dibahas bahwa diperlukan mengetahui kemunculan terakhir suatu alfabet pada *pattern* P. Kemunculan ini dihitung dengan suatu algoritma *pre-processing* pada pattern bernama buildLast, algoritma ini akan mengembalikan suatu array yang berisi semua kemunculan terakhir semua karakter ASCII. Misal untuk kata “Halo” , a akan bernilai 2, dan seterusnya, untuk karakter lain yang tidak ada pada pattern, diisi nilai -1.

Untuk algoritma *Boyer-Moore*, kompleksitas kasus terburuk adalah $O(nm+A)$, namun algoritma ini sangat cepat apabila digunakan untuk alfabet yang besar, dengan kata lain, *Boyer-Moore* sangat baik untuk teks yang menggunakan alfabet Inggris, namun akan buruk untuk teks alfabet *Binary*.

2.3 Regex

Regular expression adalah konstruksi bahasa yang digunakan untuk mencocokkan suatu *set of strings* pada teks berdasarkan suatu pola tertentu. Regex kerap digunakan untuk kasus-kasus kompleks. Selain itu, regex juga seringkali digunakan untuk menguraikan kata alias *parsing*. Regex juga dapat digunakan untuk melakukan pencarian, substitusi, atau pemisahan *string* dalam kasus yang rumit.

Pada metode *string matching*, regex memainkan peran yang penting. Aplikasi dari regex sendiri dapat sebagai validator dari suatu teks yang dapat memastikan kebenaran dari format suatu teks, contohnya pada nama, nomor telepon, *email*, dan lain sebagainya. Berbeda dengan algoritma *string matching* seperti KMP dan Boyer-Moore, regex memiliki fleksibilitas yakni tidak selalu *exact match* alias dapat bergantung pada *pattern* dari regex tersebut. Sebagai contoh, regex yang dapat mencocokkan semua string yang berakhir dengan 01 adalah `.*01$` yang mana titik menandakan *match* semua karakter sebanyak 0 atau lebih kali, kemudian dilanjut dengan 01 yang merupakan *literal match* yang dicari, dan diakhiri dengan \$ yang menandakan 01 harus berada pada akhir *string*.

2.4 Aplikasi Web

Aplikasi berbasis *web* merupakan aplikasi yang dapat diakses oleh pengguna dengan perangkat apapun yang dimiliki. Selama ada internet, aplikasi tersebut dapat diakses kapanpun dan dimanapun. Berbeda dengan aplikasi *desktop* yang harus mengunduh instalasi program secara lokal, aplikasi *web* dapat diakses hanya dengan menggunakan *web browser* tanpa harus mengunduh aplikasi *client* khusus untuk menjalankannya. Dengan adanya *web browser*, aplikasi *web* dibuat semakin mudah dan tersedia bagi penggunanya. Adapun keuntungan yang dapat diberi oleh aplikasi berbasis web adalah akses informasi mudah, *setup server* yang lebih mudah,

informasi yang lebih mudah didistribusikan, serta platform yang bebas sehingga informasi dapat disajikan oleh browser web pada sistem operasi mana saja.

BAB 3

ANALISIS PEMECAHAN MASALAH

3.1. Langkah-langkah Pemecahan Masalah tiap Fitur

I. Tes DNA

Data dari *frontend* diterima oleh *backend* lalu diproses dan divalidasi. Jika sesuai dengan model yang telah dibuat, maka akan dipanggil fungsi CekDNA untuk menentukan apakah data yang diterima memperoleh prediksi yang benar atau tidak serta hasil kemiripan dari DNA. Berikut algoritma yang digunakan dalam fungsi CekDNA:

A. KMP

Fungsi ini akan menerima 2 parameter, yaitu *text* dan *pattern*, lalu akan digunakan fungsi *computeFail* (fungsi yang akan mengembalikan suatu array yang berisi prefiks terpanjang yang juga sufiksnya pada tiap indeks). Terdapat i yang berupa indeks untuk teks dan j untuk pattern, lalu akan dilakukan iterasi sepanjang text, pertama - tama akan dicek apabila pada posisi i dan j apakah teks dan *pattern* memuat huruf yang sama, apabila benar, geser masing - masing sebesar 1. Fungsi akan mengembalikan true apabila j akan sebesar panjang pattern, karena hal ini berarti tidak terjadi ketidakcocokan sama sekali. Apabila ditemukan suatu ketidakcocokan, akan dilihat pada array *computeFail* dan pengecekan akan dilakukan kembali namun perbandingan dimulai dari j yang baru didapat dari array *computeFail* tersebut.

B. BM

Fungsi ini akan menerima 2 parameter, yaitu *text* dan *pattern*, akan dipanggil fungsi *buildLast* yang akan mengembalikan array sepanjang ASCII *character* yang berisi nilai *last occurrence* masing masing karakter tersebut (bernilai -1 apabila karakter tidak ada pada *pattern*). Lalu seperti

pada KMP , i adalah indeks untuk teks dan j adalah indeks untuk *pattern*, akan dimulai suatu iterasi yang akan terus iterasi sampai i sampai sebesar panjang text. Dimulai perbandingan karakter dengan *Looking-Glass Technique* yang berarti perbandingan karakter dimulai dari indeks terakhir. Akan terus dilakukan pencocokan, dan apabila terjadi ketidaksamaan , i akan menjadi $i + \text{panjang pattern} + \text{nilai minimum antara } j \text{ atau } 1 + \text{posisi terakhir}$. Hal ini akan melakukan lompat seperti pada algoritma BM yang sudah dijelaskan sebelumnya. Misal apabila pada $i = 5$ dan $j = 2$ dan terdapat ketidaksamaan pada karakter x, apabila x ada dan lebih besar dari j (berarti ada di kanan) i akan lompat sebesar panjang pattern - last occurrence, dan apabila j lebih kecil, hanya akan lompat sebesar 1

C. Levenshtein

Levenshtein Distance adalah salah satu cara pencarian jarak antara 2 string, pada intinya *Distance* ini adalah berapa banyak edit pada karakter (misal insertion karakter baru, menggeser karakter, ataupun menghapus karakter yang ada) untuk membuat 1 string menjadi string lainnya yang dicari jaraknya. Implementasinya dilakukan pengecekan sebagai berikut :

- Dibuat sebuah matriks dengan ukuran panjang kata 1 x panjang kata 2, , lalu inisialisasi semua baris pertama dan kolom pertama dengan mengisi indeksnya, misal untuk baris 2 kolom 1 diisi 1 (karena baris 2 adalah indeks 1), baris 3 kolom 1 diisi 2, baris 1 kolom 2 diisi 1, dan seterusnya.
- Isi setiap sel dengan “*surrounding algorithm*” yaitu cek sekitar suatu sel , apabila sel atas = sel bawah (berarti pada posisi itu kata - kata sama) isi sel dengan nilai sama dengan sel diagonal karena berarti tidak dibutuhkan apapun dan distance tetap sama, apabila tidak sama, cek nilai paling kecil antara sel atas, sel bawah, dan sel diagonal ditambah 1, (apabila sel kiri paling kecil

berarti langkah terbaik adalah *deletion*, sel atas berarti *insertion*, dan sel diagonal berarti *substitusi*)

- Hasil pada pojok kanan matriks (berarti pada matriks[panjang teks 1] [panjang teks2] adalah jaraknya)
- Untuk setiap substring pada DNA pengguna, akan dihitung jaraknya dengan penyakit, lalu akan dikembalikan similarity paling besar (alias jarak paling kecil)

Setelah melalui fungsi CekDNA, hasil dari fungsi tersebut akan dimasukkan ke dalam model Riwayat, lalu dimasukkan ke database.

II. Tambah Penyakit

Data dari *frontend* diterima oleh *backend* lalu masuk ke controller bagian tambahPenyakit. Di fungsi tambahPenyakit, data akan divalidasi apakah sesuai dengan model yang sudah dibuat. Setelah validasi, data akan langsung dimasukkan ke dalam database.

III. Cek Riwayat

Data dari *frontend* diterima oleh *backend* lalu masuk ke controller bagian cekRiwayat. Di fungsi cekRiwayat, data akan divalidasi dengan model yang tersedia, Lalu ada beberapa jenis untuk cari riwayat yaitu:

1. Query tanggal

Dalam query tanggal, query dibagi menjadi 3 format yaitu:

1. YYYY-MM-DD (tanggal lengkap)
2. MM-DD (hanya bulan dan hari)
3. YYYY-MM (bulan dan tahun)

2. Query nama penyakit

Hanya berisi nama penyakit

Lalu kedua query tersebut digabung dan dikirimkan ke database, lalu hasilnya akan dikirimkan ke *frontend*.

3.2. Fitur Fungsional dan Arsitektur Aplikasi Web yang Dibangun

Aplikasi berbasis *website* yang kami buat pada Tugas Besar kali ini memiliki beberapa fitur fungsional, yakni sebagai berikut:

1. Menerima *input* penyakit baru berisi nama penyakit beserta *sequence DNA*-nya, *input* kolom pencarian berupa tanggal prediksi dan nama penyakit, serta *input* nama pengguna, *sequence DNA* penngguna, serta nama penyakit yang diuji untuk melakukan tes DNA
2. Memasukkan hasil tes DNA berisi tanggal, nama pengguna, nama penyakit yang diuji, persentase kemiripan, dan status hasil tes serta *input* penyakit baru berisi nama penyakit beserta *sequence DNA*-nya ke *database*
3. Melakukan sanitasi *input* dengan menggunakan *regular expression* untuk memastikan bahwa *sequence DNA* yang diterima valid
4. Memprediksi apakah seseorang menderita suatu penyakit tertentu dengan melakukan pencocokan *sequence DNA* yang dilakukan dengan algoritma *string matching*
5. Menampilkan riwayat/hasil prediksi tes DNA yang dapat difilter melalui *input* kolom pencarian
6. Menghitung tingkat kemiripan DNA pengguna dengan DNA penyakit pada tes DNA

Aplikasi web ini dapat dibagi menjadi dua bagian yakni *backend* dan juga *frontend*. Penerapan dari kedua bagian tersebut dilaksanakan dengan menggunakan Golang untuk *backend* serta React untuk *frontend* dengan penyimpanan data dilakukan dengan menggunakan MongoDB. Adapun struktur dari aplikasi kami tertata sebagai berikut:

```
C:.
|   README.md
|   tes.txt
|   test.txt
+
+---algorithm
|       algorithm.go
+
+---backend
|       .env
|       .gitignore
|       go.mod
|       go.sum
|       main.go
+
+---configs
|       env.go
|       setup.go
+
+---controllers
|       algorithm.go
|       errors.go
|       penyakit.go
|       riwayat.go
+
+---models
|       models.go
+
+---responses
|       responses.go
+
+---routes
|       routes.go
+
+---frontend
|       .gitignore
|       package-lock.json
|       package.json
|       README.md
```

Bagian *backend* dari aplikasi website kami bertanggung jawab sebagai tempat untuk memproses segala sesuatu yang berkaitan dengan penyimpanan file dan pemrosesan data. *Backend* dibuat dengan sebuah Echo framework dari bahasa Golang. Aplikasi *backend* sendiri menggunakan architecture berupa model view dan controller, yang dibagi lagi menjadi beberapa folder sebagai berikut:

1. Configs

Bertugas sebagai penghubung konfigurasi koneksi ke database

2. Controllers

Bertugas sebagai tempat pemrosesan data yang telah diterima dari *frontend*

3. Models

Berisi jenis-jenis *object* yang sering digunakan dalam pemrosesan data secara internal maupun eksternal

4. Responses

Berisi object yang akan diberikan kepada *frontend* lewat axios

5. Routes

Berisi endpoint URL untuk menentukan ke mana *frontend* ingin memroses data

Untuk lebih detail, alur pekerjaan yang diproses jika endpoint diakses dari *frontend* adalah:

1. Data diterima oleh routes dan routes akan memindahkan data ke controller yang telah *di-assign* oleh routes
2. Data diterima oleh controller, dan diproses. Pemrosesan terbagi menjadi 2 yaitu proses yang tidak memakai algoritma(seperti menambah penyakit) dan proses yang membutuhkan algoritma(seperti tes DNA).
3. Setelah data diproses oleh masing-masing controller, controller akan *me-return* sebuah json file dengan data yang sudah diproses di poin nomor 2.
4. Data akan dikirimkan ke *frontend*

Untuk bagian algoritma, algoritma terbagi menjadi 2 yaitu KMP dan BM yang sudah dijelaskan pada bagian 3.1.

Bagian *frontend* dari aplikasi website kami bertanggung jawab sebagai jembatan yang dapat menerima *input* pengguna dan menampilkan hasil dari olahan data dari *backend* kepada pengguna. *Frontend* kami dibangun dengan menggunakan React dengan menggunakan Axios sebagai perantara untuk membuat suatu HTTP request dari *frontend* ke *backend*. Aplikasi kami terdiri atas 5 halaman yakni:

1. Halaman utama
2. Halaman untuk menambahkan penyakit
3. Halaman untuk melakukan tes DNA
4. Halaman untuk menampilkan riwayat prediksi tes DNA

5. Halaman untuk menampilkan info tentang kami.

Setiap halaman pada aplikasi kami dianggap sebagai suatu komponen. Setiap komponen dibuat dalam bentuk *file JSX* yang menggunakan *Hooks* untuk menentukan *state* dari komponen tersebut dan juga *bootstrap*. Komponen tersebut disimpan di dalam *folder* ‘components’ yang dapat dilihat pada struktur berikut:

```
C:.
  index.css
  index.js
  logo.svg
  reportWebVitals.js
  setupTests.js
  tes.txt

  +---app
    App.css
    App.js
    App.test.js

  +---assets
    azka.JPG
    azka.png
    dhika.JPG
    dhika.png
    semi.JPG
    semi.png

  +---components
    Footer.jsx
    LamanUtama.jsx
    Navigation.jsx
    RiwayatPenyakit.jsx
    TambahPenyakit.jsx
    TentangKami.jsx
    TesDNA.jsx

  +---style
    style.css
```

Selain kelima halaman tersebut, terdapat 2 *file JSX* lainnya yang berfungsi sebagai *footer* dan juga *navigation bar* dari aplikasi web kami. Adapun *routing* antarhalaman dilakukan pada *App.js* yang terletak pada *folder* ‘app’ dengan menggunakan *react router dom*.

Untuk setiap data yang diberikan oleh *user*, data disimpan dalam *state* dari komponen yang menerima *input* tersebut. Apabila *user* ingin mengirim data, akan dilakukan validasi terlebih dahulu untuk memastikan bahwa data yang akan dikirim telah valid dan lengkap. Apabila sudah valid dan lengkap, data yang disimpan pada *state* akan dikirim dengan membuat HTTP request melalui axios dengan method ‘post’ dan url berupa rute yang dituju. Untuk beberapa komponen, *response* dari HTTP request tersebut akan digunakan untuk menentukan *state* terbaru dari komponen tersebut (contohnya pada

halaman RiwayatPenyakit.jsx dimana setelah mengirim data pengguna, komponen akan menyesuaikan *statenya* dengan hasil dari tes DNA pengguna).

BAB 4

IMPLEMENTASI DAN PENGUJIAN

4.1 Spesifikasi Teknis Program

4.1.1 Struktur Program dan Data

Berikut struktur data dari aplikasi kami:

```
C:.
|   README.md
|
+---backend
|   |   .env
|   |   .gitignore
|   |   go.mod
|   |   go.sum
|   |   main.go
|
|   +---configs
|       |       env.go
|       |       setup.go
|
|   +---controllers
|       |       algorithm.go
|       |       errors.go
|       |       penyakit.go
|       |       riwayat.go
|
|   +---models
|       |       models.go
|
|   +---responses
|       |       responses.go
|
|   +---routes
|       |       routes.go
|
|   +---vendor
|       |       modules.txt
|
+---frontend
```

```
.gitignore
package-lock.json
package.json

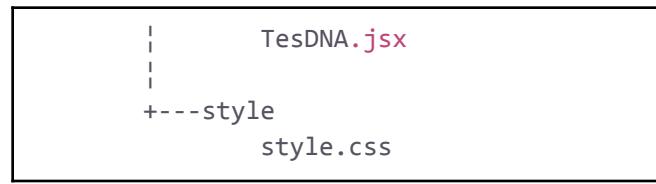
+---public
    covid-test.png
    dna.png
    favicon.ico
    hehe.jpg
    index.html
    logo192.png
    logo512.png
    manifest.json
    notes.png
    robots.txt
    SXF.png

+---src
    index.css
    index.js
    logo.svg
    reportWebVitals.js
    setupTests.js

+---app
    App.css
    App.js
    App.test.js

+---assets
    azka.JPG
    azka.png
    dhika.JPG
    dhika.png
    semi.JPG
    semi.png

+---components
    Footer.jsx
    LamanUtama.jsx
    Navigation.jsx
    RiwayatPenyakit.jsx
    TambahPenyakit.jsx
    TentangKami.jsx
```



Aplikasi kami terdiri atas dua folder utama bernama ‘backend’ dan ‘frontend’ serta sebuah *file* README dengan rincian folder berikut:

1. Folder Backend

Terdiri atas *folder* configs, controllers, models, dan responses yang membangun *backend* dari aplikasi dan tempat diimplementasikannya algoritma *string matching*. Terdapat *file* env untuk *setup* basis data.

2. Folder Frontend

Terdiri atas *folder* app, assets, components, dan style yang terdapat di dalam folder src serta *file* lainnya yang berfungsi sebagai dasar dari *frontend* aplikasi kami.

4.1.2 Fungsi dan Prosedur

Program kami terdiri atas beberapa fungsi dan prosedur yakni sebagai berikut:

1. max

Fungsi ini berfungsi untuk mencari nilai maksimum antara dua integer

2. min

Fungsi ini berfungsi untuk mencari nilai minimum antara dua integer

3. computeFail

Fungsi ini berfungsi untuk *pre-processing* pada KMP dan akan mengembalikan suatu array yang menyimpan nilai prefiks terpanjang yang juga merupakan sufiks pada tiap indeksnya

4. KMP

Fungsi ini berfungsi untuk mengecek apakah *pattern* dapat ditemukan dalam teks menggunakan algoritma KMP, fungsi akan mengembalikan nilai *boolean*

5. buildLast

Fungsi ini berfungsi untuk *pre-processing* algoritma BM yang akan mengembalikan suatu array dengan panjang sepanjang ASCII character dan tiap indeks merepresentasikan kemunculan terakhir karakter tersebut pada *pattern*, akan bernilai -1 bila karakter tidak dapat ditemukan

6. bmMatch

Fungsi ini berfungsi untuk mengecek apakah *pattern* dapat ditemukan dalam teks menggunakan algoritma BM, fungsi akan mengembalikan nilai *boolean*

7. levenshteinDistance

Fungsi ini berfungsi untuk menghitung jarak antara 2 string menggunakan algoritma *Levenshtein*

8. divideSubstring

Fungsi ini berfungsi untuk membagi suatu string menjadi substring substring sepanjang k, k merupakan parameter yang dimasukan pada fungsi

9. findSimilarity

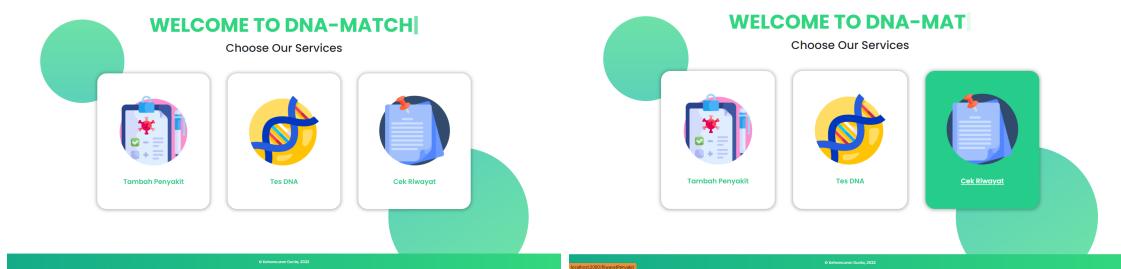
Fungsi ini berfungsi untuk memecah DNA yang panjang (dengan divideSubstring) lalu mencari jarak antara substring dan DNA penyakit, dan akan mengembalikan nilai double berupa kemiripan , dihitung dengan cara panjang *pattern* dikurang jarak terkecil dibagi panjang *pattern*

10. cekDNA

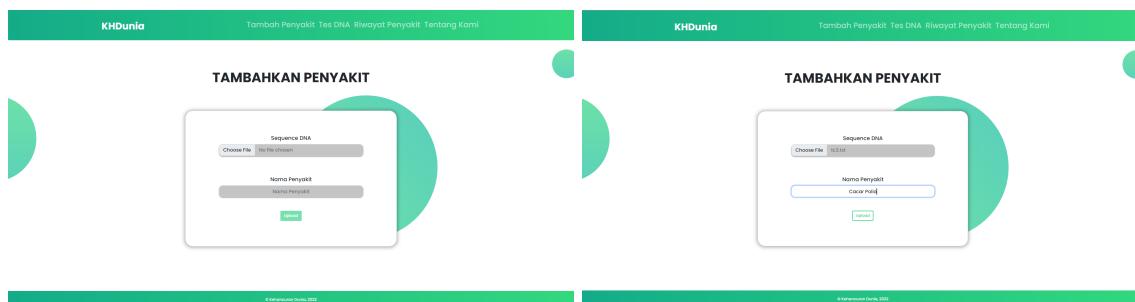
Fungsi ini berfungsi untuk menentukan metode pencocokan string(KMP atau BM) yang akan digunakan oleh user, sesuai input dari *frontend*. Lalu, memanggil fungsi findSimilarity yang akhirnya hasilnya akan dikirimkan lagi ke *backend* untuk diproses lebih lanjut.

4.2 Tata Cara Penggunaan Program

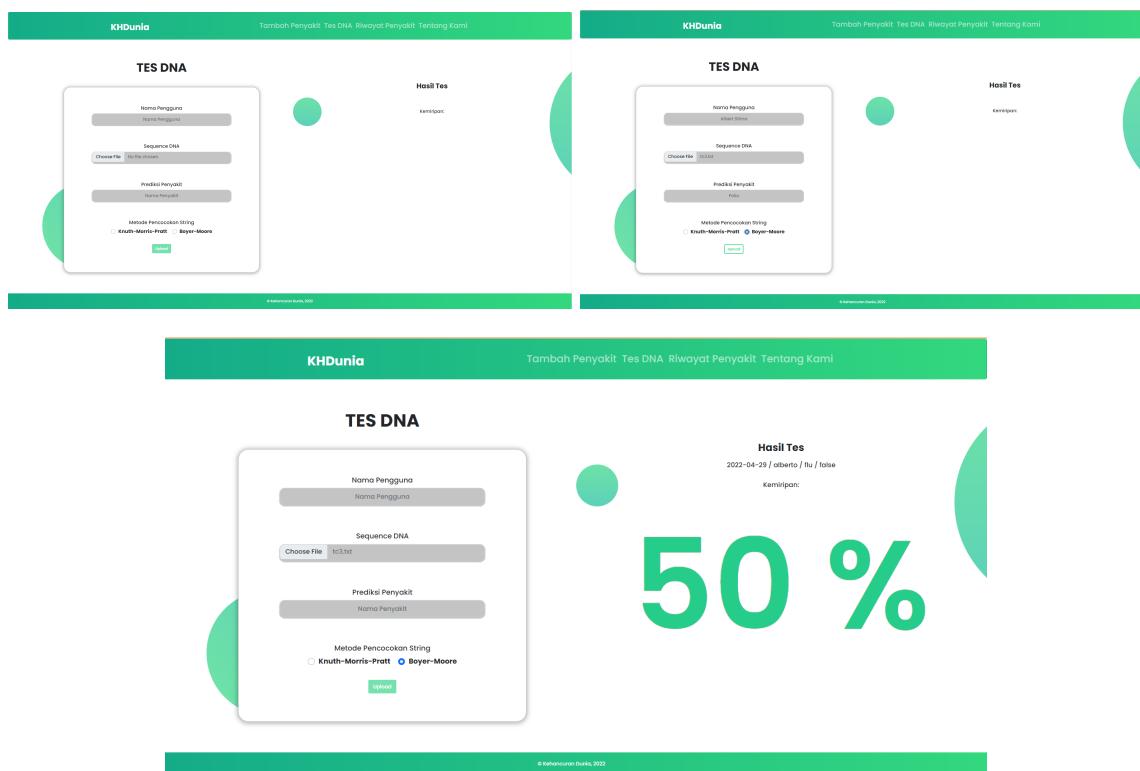
Aplikasi kami dimulai dari halaman utama. Saat pertama *load*, akan ada animasi *fade-in* untuk tiga tombol yang dapat ditekan untuk mengarahkan pengguna ke halaman tambah penyakit, halaman tes DNA, atau halaman cek riwayat. Berikut tampilan halaman utama aplikasi kami:



Apabila tombol tambah penyakit ditekan, pengguna akan diarahkan ke halaman tambah penyakit. Pada halaman tambah penyakit, pengguna dapat memasukkan *file txt* berisikan *sequence DNA* dan juga nama penyakit. Apabila *sequence DNA* tidak valid maka *input* tidak akan diterima dan dimasukkan ke *database*. Selain itu, apabila salah satu *slot input* belum terisi maka tombol tidak dapat dipencet sehingga tidak mungkin ada *input* kosong. Berikut tampilan halaman tambah penyakit pada aplikasi kami:



Apabila pengguna diarahkan ke halaman tes DNA, pengguna dapat memasukkan nama pengguna, file txt berisikan *sequence* DNA, dan nama penyakit yang akan diprediksi. Pengguna juga dapat memilih metode pencocokan *string* (Knuth-Morris-Pratt atau Boyer-Moore) dengan memencet tombol sebelum melakukan *upload*. Setelah *input* pengguna diolah, hasil tes DNA akan ditampilkan pada aplikasi seperti pada gambar di bawah:



Jika pengguna diarahkan ke halaman riwayat penyakit, saat pertama halaman *di-load* akan ditampilkan tabel berisikan riwayat hasil penyakit tes DNA. Apabila pengguna ingin mencari riwayat hasil penyakit tes berdasarkan suatu *query*, pengguna dapat memasukkan tanggal dan nama penyakit, tanggal, atau nama penyakit pada *search bar*. Format tanggal yang dapat diterima dapat berupa YYYY-MM-DD, YYYY <Nama Bulan>, <Nama Bulan> DD. Tampilan halaman riwayat tes dapat dilihat sebagai berikut:

RIWAYAT TES

Tambah Penyakit, Tes DNA, Riwayat Penyakit, Tentang Kami

KHDunia

No. Tanggal Nama Pengguna Nama Penyakit Hasil Tes DNA

1	2020-04-08	dhika	flu	true
2	2020-04-08	semi	sakit mata	false
3	2020-04-08	hero	flu	false
4	2020-04-08	azka	flu	false
5	2020-04-08	daphne	sakit purpung	false
6	2020-04-08	benedict	flu	true
7	2020-04-08	dhika	flu	true
8	2020-04-08	grina	sakit mata	false
9	2020-04-08	semir	sakit purpung	false
10	2020-04-08	jenny	sakit mata	false
11	2020-04-08	joni	flu	false
12	2020-04-08	lulu	sakit purpung	false
13	2020-04-08	hero	flu	true
14	2020-04-08	grina	flu	false

© Kedokteranku Studio, 2020

RIWAYAT TES

Tambah Penyakit, Tes DNA, Riwayat Penyakit, Tentang Kami

KHDunia

No. Tanggal Nama Pengguna Nama Penyakit Hasil Tes DNA

1	2020-04-19	dhika	flu	true
2	2020-04-19	semi	sakit mata	false
3	2020-04-19	hero	flu	false
4	2020-04-19	azka	flu	false
5	2020-04-19	daphne	sakit purpung	false
6	2020-04-19	benedict	flu	true
7	2020-04-19	dhika	flu	true
8	2020-04-19	grina	sakit mata	false
9	2020-04-19	semir	sakit purpung	false
10	2020-04-19	jenny	sakit mata	false
11	2020-04-19	joni	flu	false
12	2020-04-19	lulu	sakit purpung	false
13	2020-04-19	hero	flu	true
14	2020-04-19	grina	flu	false

© Kedokteranku Studio, 2020

RIWAYAT TES

Tambah Penyakit, Tes DNA, Riwayat Penyakit, Tentang Kami

KHDunia

No. Tanggal Nama Pengguna Nama Penyakit Hasil Tes DNA

1	2020-04-28	dhika	flu	true
2	2020-04-28	hero	flu	true
3	2020-04-28	azka	flu	false
4	2020-04-28	benedict	flu	true
5	2020-04-28	dhika	flu	true
6	2020-04-28	joni	flu	true
7	2020-04-28	hero	flu	true
8	2020-04-29	dhika	flu	false
9	2020-04-29	marche	flu	false
10	2020-04-29	oberto	flu	false

© Kedokteranku Studio, 2020

Pada halaman tentang kami, terdapat profil kami masing-masing dengan animasi *slide-in* untuk mempercantik tampilan. Untuk mengganti anggota yang di-*highlight*, dapat diklik salah satu kotak yang berisi foto berwarna abu-abu.

DHIKA (Male, 20)

"Sekali VTuber hidup sampai mati tapi"

AZKA

"Hidup cuma sekali tapi VTuber sampai mati"

SEMI

"Hidup VTuber tapi sekali mati sampai"

KHDunia

Tambah Penyakit, Tes DNA, Riwayat Penyakit, Tentang Kami

KHDunia

Tambah Penyakit, Tes DNA, Riwayat Penyakit, Tentang Kami

KHDunia

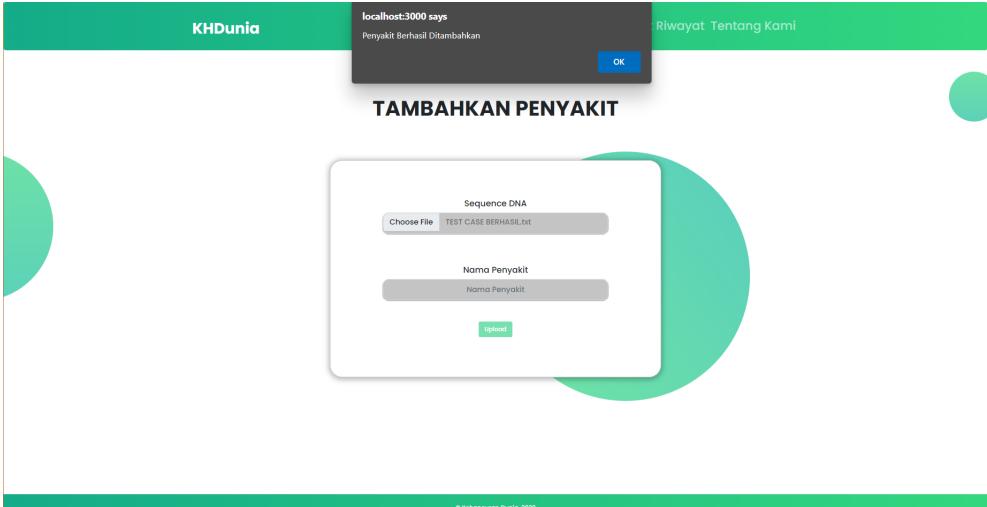
Tambah Penyakit, Tes DNA, Riwayat Penyakit, Tentang Kami

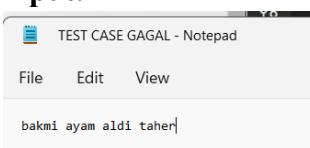
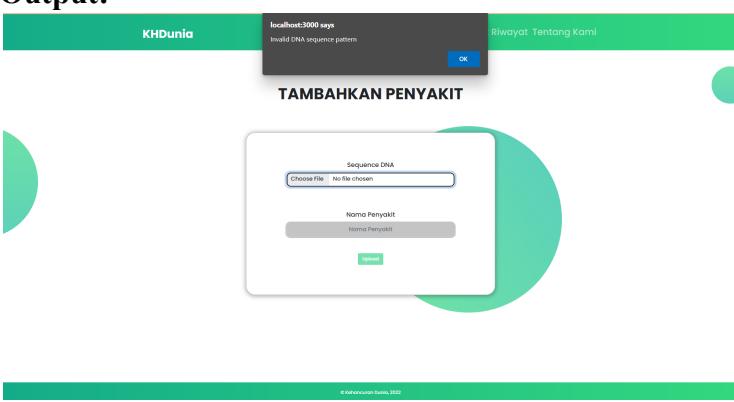
© Kedokteranku Studio, 2020

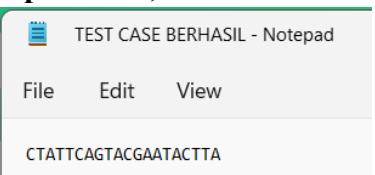
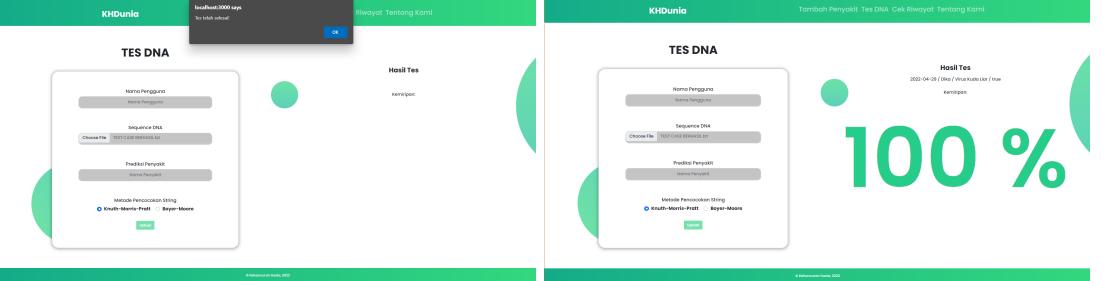
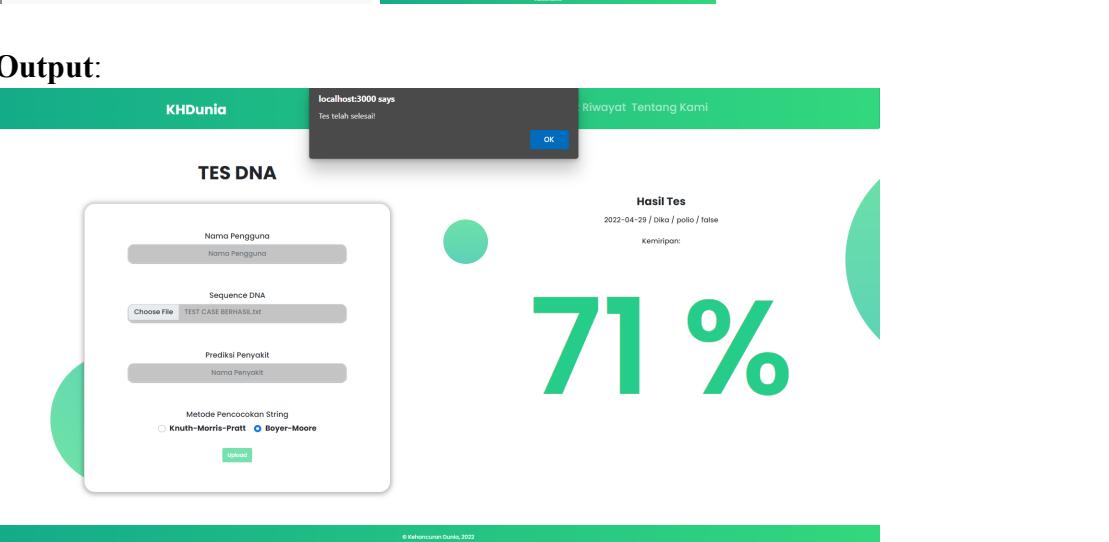
© Kedokteranku Studio, 2020

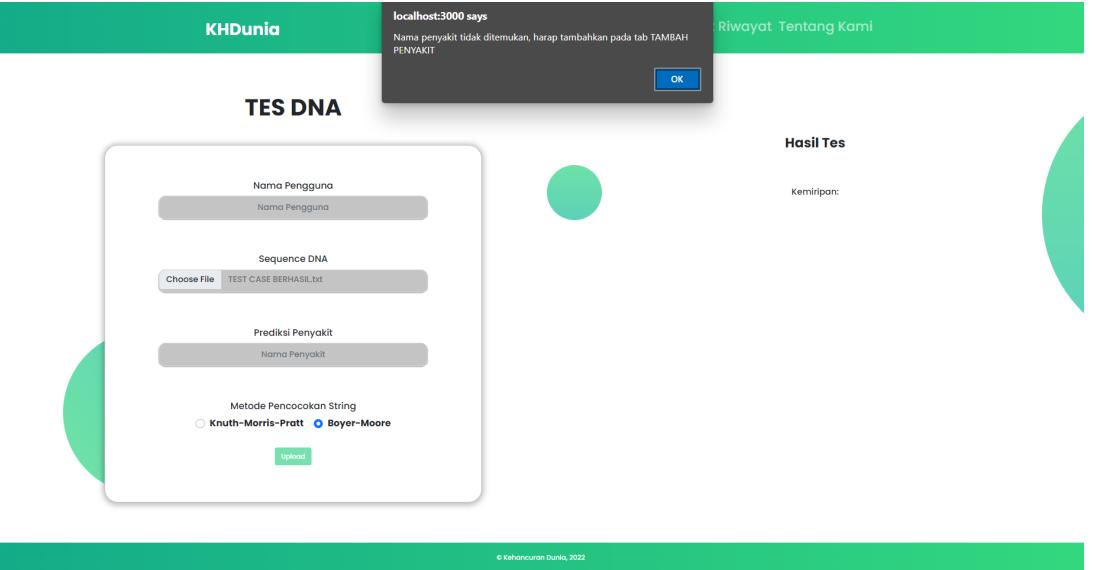
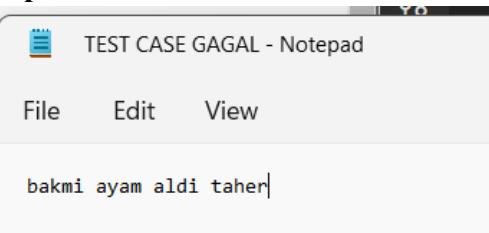
© Kedokteranku Studio, 2020

4.3 Hasil Pengujian

1.	<p>Skenario: Menambah penyakit baru ke database</p> <p>Input: Virus Kuda Liar, "CTATTCAGTACGAATACTTA"</p> <p>Output:</p>  <p>The screenshot shows a modal dialog titled "TAMBAHKAN PENYAKIT". It contains fields for "Sequence DNA" (with a "Choose File" button and a file path "TEST-CASE-BERHASIL.txt"), "Nama Penyakit" (with a "Nama Penyakit" placeholder), and an "Upload" button. Above the modal, a success message from "localhost:3000 says" states "Penyakit Berhasil Ditambahkan" with an "OK" button. The background has a green circular watermark.</p> <p>DB Before:</p> <p>DNA.penyakit</p> <p>STORAGE SIZE: 36KB TOTAL DOCUMENTS: 5 INDEXES TOTAL SIZE: 36KB</p> <p>Find Indexes Schema Anti-Patterns Aggregation Search Indexes</p> <p>FILTER { field: 'value' }</p> <pre>_id: ObjectId("626aa99e984a19ba0ad20ffd") namapenyakit: "sakit punggung" sequencedna: "ACGT" _id: ObjectId("626aa99e984a19ba0ad20ff0") namapenyakit: "sakit mata" sequencedna: "ATGC" _id: ObjectId("626aa99e984a19ba0ad20ff1") namapenyakit: "Bisul Naga" sequencedna: "GACTGATCGATGGCGTACGATCTAGCATCGACGTA" _id: ObjectId("626bb0db11c910ead25f7753") namapenyakit: "cocor polio" sequencedna: "AGCTCGA"</pre>
----	---

	<p>DB After:</p> <p>DNA.penyakit</p> <p>STORAGE SIZE: 36KB TOTAL DOCUMENTS: 6 INDEXES TOTAL SIZE: 36KB</p> <p>Find Indexes Schema Anti-Patterns 0 Aggregation</p> <p>FILTER { field: 'value' }</p> <pre>_id: ObjectId("626aa99e984a19ba0ad20ffd") namapenyakit: "sakit mata" sequencedna: "ATGC" _id: ObjectId("626aa99e984a19ba0ad20ffd") namapenyakit: "Bisul Naga" sequencedna: "GACTGATCGATGGCGTACGATCTAGCATCGACGTA" _id: ObjectId("626bb0db11c910ead25f7753") namapenyakit: "cocor polio" sequencedna: "AGCTCGA" _id: ObjectId("626c03530c3397c2eb613199") namapenyakit: "Virus Kuda Liar" sequencedna: "CTATTCACTACGAATACTTA"</pre>
2.	<p>Skenario: Menambah penyakit baru dengan error sequence dna</p> <p>Input:</p>  <p>Output:</p> 

3.	<p>Skenario: Tes DNA dengan KMP hasil true</p> <p>Input: Dika, Virus Kuda Liar</p>  <p>Output:</p> 
4.	<p>Skenario: Tes DNA dengan Boyer-Moore hasil false</p> <p>Input: Dika, polio</p>  <p>Output:</p> 

5.	<p>Skenario: Tes DNA untuk nama penyakit yang tidak ada di db</p> <p>Input: bakmi ayam aldi tahir Output:</p> 
6.	<p>Skenario: Tes DNA untuk orang dengan sequence-dna tidak valid</p> <p>Input:</p>  <p>Output:</p>

KHDunia localhost:3000 says
Invalid DNA sequence pattern

Riwayat Tentang Kami

OK

TES DNA

Nama Pengguna: ahaha

Sequence DNA: Choose File No file chosen

Prediksi Penyakit: Nama Penyakit

Metode Pencocokan String:
 Knuth-Morris-Pratt Boyer-Moore

Hasil Tes

Kemiripan:

7. **Skenario:** Pencarian Riwayat Tes dengan tanggal

Input: 2022-04-28 (YYYY-MM-DD)

Output:

KHDunia Tambah Penyakit Tes DNA Cek Riwayat Tentang Kami

RIWAYAT TES

Tanggal / Nama Penyakit **CARI**

No.	Tanggal	Nama Pengguna	Nama Penyakit	Hasil Tes DNA
1	2022-04-28	dhika	flu	true
2	2022-04-28	semi	sakit mata	true
3	2022-04-28	azka	flu	true
4	2022-04-28	KUY	flu	false
5	2022-04-28	daphne	sakit punggung	false
6	2022-04-28	benedict	flu	true
7	2022-04-28	dhika	flu	true
8	2022-04-28	anton	sakit mata	false
9	2022-04-28	wonton	sakit punggung	false
10	2022-04-28	jeremy	sakit mata	false
11	2022-04-28	joni	flu	true
12	2022-04-28	lys	sakit punggung	false
13	2022-04-28	hero	flu	true

Input: April 2022 (Bulan Year)

Output:

KHDunia

Tambah Penyakit. Tes DNA Cek Riwayat. Tentang Kami

RIWAYAT TES

Tanggal / Nama Penyakit

No.	Tanggal	Nama Pengguna	Nama Penyakit	Hasil Tes DNA
1	2022-04-28	dhika	flu	true
2	2022-04-28	semi	sakit mata	true
3	2022-04-28	ekta	flu	true
4	2022-04-28	KUY	flu	false
5	2022-04-28	daphne	sakit punggung	false
6	2022-04-28	benedict	flu	true
7	2022-04-28	dhika	flu	true
8	2022-04-28	anton	sakit mata	false
9	2022-04-28	wonton	sakit punggung	false
10	2022-04-28	jeremy	sakit mata	false
11	2022-04-28	joni	flu	true
12	2022-04-28	lulu	sakit punggung	false
13	2022-04-28	heru	flu	true
14	2022-04-28	dhika	flu	false
15	2022-04-28	manche	flu	false
16	2022-04-29	manche	sakit mata	false
17	2022-04-29	manche	sakit punggung	false
18	2022-04-29	suleen	sakit mata	false
19	2022-04-29	alberto	flu	false
20	2022-04-29	ojeng	cocor pollo	true

© Kedokteranku, 2022

KHDunia

Tambah Penyakit. Tes DNA Cek Riwayat. Tentang Kami

RIWAYAT TES

Tanggal / Nama Penyakit

No.	Tanggal	Nama Pengguna	Nama Penyakit	Hasil Tes DNA
1	2022-04-28	dhika	flu	true
2	2022-04-28	semi	sakit mata	true
3	2022-04-28	ekta	flu	true
4	2022-04-28	KUY	flu	false
5	2022-04-28	daphne	sakit punggung	false
6	2022-04-28	benedict	flu	true
7	2022-04-28	dhika	flu	true
8	2022-04-28	anton	sakit mata	false
9	2022-04-28	wonton	sakit punggung	false
10	2022-04-28	jeremy	sakit mata	false
11	2022-04-28	joni	flu	true
12	2022-04-28	lulu	sakit punggung	false
13	2022-04-28	heru	flu	true
14	2022-04-28	dhika	flu	false
15	2022-04-29	manche	flu	false
16	2022-04-29	manche	sakit mata	false
17	2022-04-29	manche	sakit punggung	false
18	2022-04-29	suleen	sakit mata	false
19	2022-04-29	alberto	flu	false
20	2022-04-29	ojeng	cocor pollo	true

© Kedokteranku, 2022

Input: 2022 ApRiL (Year Bulan)

Output:

KHDunia

Tambah Penyakit. Tes DNA Cek Riwayat. Tentang Kami

RIWAYAT TES

Tanggal / Nama Penyakit

No.	Tanggal	Nama Pengguna	Nama Penyakit	Hasil Tes DNA
1	2022-04-28	dhika	flu	true
2	2022-04-28	semi	sakit mata	true
3	2022-04-28	ekta	flu	true
4	2022-04-28	KUY	flu	false
5	2022-04-28	daphne	sakit punggung	false
6	2022-04-28	benedict	flu	true
7	2022-04-28	dhika	flu	true
8	2022-04-28	anton	sakit mata	false
9	2022-04-28	wonton	sakit punggung	false
10	2022-04-28	jeremy	sakit mata	false
11	2022-04-28	joni	flu	true
12	2022-04-28	lulu	sakit punggung	false
13	2022-04-28	heru	flu	true
14	2022-04-28	dhika	flu	false
15	2022-04-29	manche	flu	false
16	2022-04-29	manche	sakit mata	false
17	2022-04-29	manche	sakit punggung	false
18	2022-04-29	suleen	sakit mata	false
19	2022-04-29	alberto	flu	false
20	2022-04-29	ojeng	cocor pollo	true

© Kedokteranku, 2022

Input: 22 April (Hari Bulan)

Output:

--	--

8. **Skenario:** Pencarian riwayat tes dengan nama penyakit

Input: flu

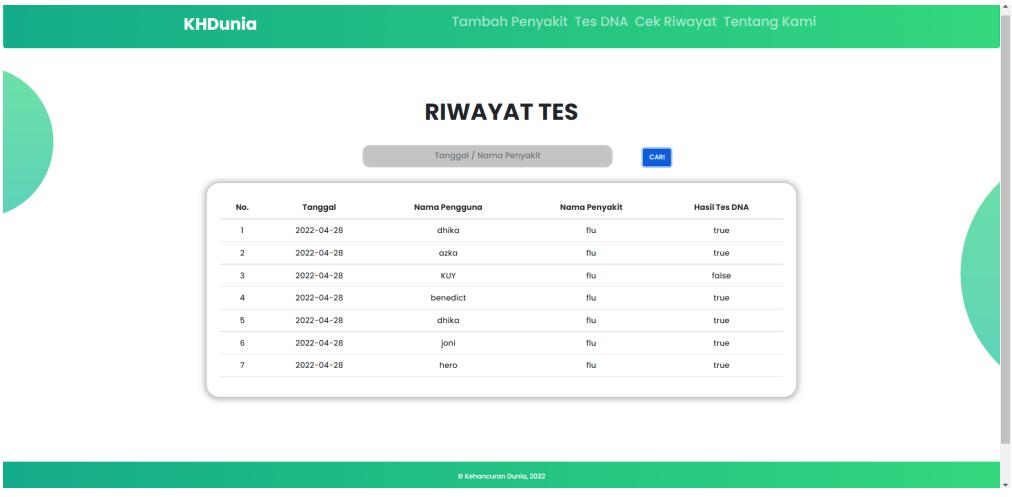
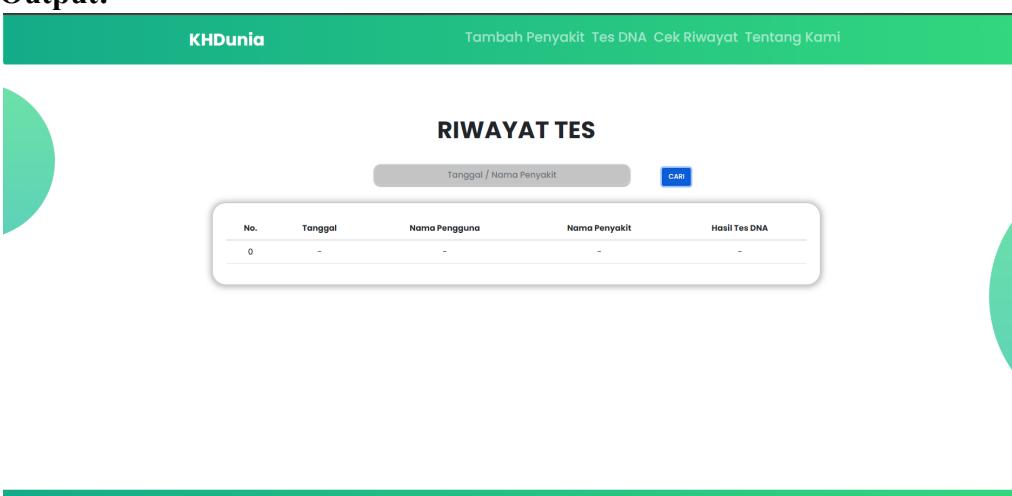
Output:

--	--

9. **Skenario:** Pencarian riwayat tes dengan nama penyakit dan tanggal

Input: 2022-04-28 flu

Output:

	 <p>KHDunia</p> <p>Tambah Penyakit Tes DNA Cek Riwayat Tentang Kami</p> <h3>RIWAYAT TES</h3> <p>Tanggal / Nama Penyakit</p> <p>CARI</p> <table border="1"> <thead> <tr> <th>No.</th> <th>Tanggal</th> <th>Nama Pengguna</th> <th>Nama Penyakit</th> <th>Hasil Tes DNA</th> </tr> </thead> <tbody> <tr><td>1</td><td>2022-04-28</td><td>dhika</td><td>flu</td><td>true</td></tr> <tr><td>2</td><td>2022-04-28</td><td>azka</td><td>flu</td><td>true</td></tr> <tr><td>3</td><td>2022-04-28</td><td>KUY</td><td>flu</td><td>false</td></tr> <tr><td>4</td><td>2022-04-28</td><td>benedict</td><td>flu</td><td>true</td></tr> <tr><td>5</td><td>2022-04-28</td><td>dhika</td><td>flu</td><td>true</td></tr> <tr><td>6</td><td>2022-04-28</td><td>joni</td><td>flu</td><td>true</td></tr> <tr><td>7</td><td>2022-04-28</td><td>hero</td><td>flu</td><td>true</td></tr> </tbody> </table> <p>© Kehancurun Dunia, 2022</p>	No.	Tanggal	Nama Pengguna	Nama Penyakit	Hasil Tes DNA	1	2022-04-28	dhika	flu	true	2	2022-04-28	azka	flu	true	3	2022-04-28	KUY	flu	false	4	2022-04-28	benedict	flu	true	5	2022-04-28	dhika	flu	true	6	2022-04-28	joni	flu	true	7	2022-04-28	hero	flu	true
No.	Tanggal	Nama Pengguna	Nama Penyakit	Hasil Tes DNA																																					
1	2022-04-28	dhika	flu	true																																					
2	2022-04-28	azka	flu	true																																					
3	2022-04-28	KUY	flu	false																																					
4	2022-04-28	benedict	flu	true																																					
5	2022-04-28	dhika	flu	true																																					
6	2022-04-28	joni	flu	true																																					
7	2022-04-28	hero	flu	true																																					
10.	<p>Skenario: Pencarian riwayat tes noneksisten melalui nama penyakit invalid</p> <p>Input: sarah</p> <p>Output:</p>  <p>KHDunia</p> <p>Tambah Penyakit Tes DNA Cek Riwayat Tentang Kami</p> <h3>RIWAYAT TES</h3> <p>Tanggal / Nama Penyakit</p> <p>CARI</p> <table border="1"> <thead> <tr> <th>No.</th> <th>Tanggal</th> <th>Nama Pengguna</th> <th>Nama Penyakit</th> <th>Hasil Tes DNA</th> </tr> </thead> <tbody> <tr><td>0</td><td>-</td><td>-</td><td>-</td><td>-</td></tr> </tbody> </table> <p>© Kehancurun Dunia, 2022</p>	No.	Tanggal	Nama Pengguna	Nama Penyakit	Hasil Tes DNA	0	-	-	-	-																														
No.	Tanggal	Nama Pengguna	Nama Penyakit	Hasil Tes DNA																																					
0	-	-	-	-																																					

4.4 Analisis

Aplikasi web yang kami buat berhasil berjalan dengan baik sesuai yang diharapkan. Aplikasi kami dapat diakses melalui lokal maupun remote yaitu dengan menggunakan deployment platform untuk FE pada vercel sedangkan BE pada heroku. Untuk penyimpanan

database sendiri, kami menggunakan mongodb. Aplikasi kami memiliki fitur diantaranya homepage, “Tambah Penyakit”, “Tes DNA”, dan “Cek Riwayat”. Untuk kembali ke homepage kami, user dapat memencet logo “KHdunia”

Kami memanfaatkan regex dalam mensanitasi inputan user diantaranya input DNA sequence pada fitur “Tambah Penyakit” dan “Tes DNA” serta query pada fitur “Cek Riwayat”. Untuk DNA sequence, yang diperbolehkan hanya AGCT, jika tidak sesuai dengan regex maka web kami akan mengeluarkan alert. Untuk query pada “Cek Riwayat”, web kami dapat mencari berdasarkan tanggal dan nama penyakit. Untuk tanggal kelompok kami dapat YYYY-MM-DD, MM-DD, dan DD-YYYY.

Fitur “Tambah Penyakit” dapat menambah penyakit baru beserta DNA sequence nya ke database kami. Sebagai catatan, apabila keluar alert karena DNA sequence salah, penyakit tidak akan tertambah ke db. Fitur ini juga dapat digunakan untuk mengupdate DNA sequence penyakit tertentu. Di db, akan tersimpan lengkap DNA Sequence awal dan perubahan-perubahannya, sedangkan di backend akan dilakukan pengambilan DNA Sequence paling terakhir untuk dimanfaatkan pada fitur “Tes Riwayat”.

Fitur “Tes DNA” dapat digunakan untuk mengetes apakah seseorang mengidap suatu penyakit. Pengecekan akan dilakukan dengan pemilihan metode pencocokan string yaitu algoritma KMP atau Boyer Moore sesuai dengan keinginan user. Untuk similarity digunakan algoritma Levenshtein. Hasil dengan similarity diatas 80 memiliki makna bahwa seseorang mengidap penyakit tersebut.

Fitur “Cek Riwayat” dapat digunakan untuk mengecek riwayat tes yang pernah dilakukan oleh seluruh user. User dapat mencari sesuai query yang diinginkan, yaitu tanggal maupun nama penyakit. Tanggal dapat dituliskan dengan format YYYY-MM-DD, MM-DD, dan DD-YYYY. Jika query yang diinginkan tidak ada pada data kami, maka tidak akan ada data yang muncul.

BAB 5

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Melalui pengerjaan Tugas Besar 3 IF2211 Strategi Algoritma, kelompok berhasil mengimplementasikan secara langsung konsep-konsep yang telah diajarkan pada perkuliahan IF2211, yakni mengenai algoritma pencocokan *string* seperti Knuth-Morris-Pratt dan Boyer-Moore serta *Regular Expression*. Aplikasi berbasis *website* yang digunakan untuk memprediksi suatu penyakit genetik tertentu berhasil diimplementasikan dengan tampilan *frontend* yang memanfaatkan *framework* React serta algoritma *backend* yang menggunakan bahasa Golang. Selain itu, aplikasi juga berhasil melakukan penyimpanan ke database daftar penyakit dan hasil tes yaitu MongoDB. Adapun fitur-fitur aplikasi yang berhasil diimplementasikan adalah sebagai berikut.

1. Aplikasi dapat menerima input user berupa nama penyakit beserta DNA Sequencenya lalu dimasukkan ke DB
2. Aplikasi dapat mengeluarkan hasil bahwa seseorang mengidap penyakit tertentu atau tidak berdasarkan DNA Sequence dan dimasukkan ke DB dengan menghitung juga tingkat kemiripan antara DNA penyakit dengan DNA pengguna yang di tes.
3. Aplikasi dapat melakukan pencarian dari hasil tes yang telah dilakukan dengan tampilan mengikuti filter sesuai input user.

5.2 Saran

Berdasarkan proses pengerjaan tugas besar yang telah kelompok kami lakukan, kami menimbang bahwa *programmer* harus memiliki kemampuan dalam manajemen, khususnya dalam hal membagi dan mendekomposisi program ke dalam bagian-bagian tertentu. Bagian *backend* dan *frontend* program dirasa perlu dibagi dengan jelas agar proses pengembangan dapat terhindar dari konflik sehingga keberlangsungan dari proses pembuatan aplikasi *website* dapat berjalan lebih lancar dan efektif. Selain itu, kami

menyarankan untuk melakukan pemrograman secara modular. Modularitas akan membantu *programmer*, baik itu dalam proses pembuatan kode program, maupun pada saat pencarian *bug* atau *error* pada program.

5.3 Refleksi

Pengerjaan Tugas Besar 3 IF2211 Strategi Algoritma telah memberikan wadah bagi kelompok untuk belajar mengenai koordinasi, kerja sama, serta manajemen waktu yang baik. Kelompok menyadari bahwa dalam proses pengerjaan tugas besar, masih terdapat berbagai tantangan dan kendala yang mungkin kami alami secara pribadi, seperti misalnya, bahasa pemrograman yang baru, kesulitan dalam pemilihan *framework* dan *tools* yang tepat untuk program, pembagian kerja, hingga kesulitan dalam mengatur waktu. Akan tetapi, berkat pengajaran yang cukup dari perkuliahan IF2211, bimbingan dari asisten, serta koordinasi yang baik antar anggota kelompok, kami berhasil menyelesaikan Tugas Besar 3 IF2211 Strategi Algoritma ini dengan tepat waktu. Proses pengerjaan tugas besar ini tentu akan menjadi bekal bagi kelompok untuk menghadapi tugas-tugas dan proyek-proyek kami yang berikutnya sebagai mahasiswa jurusan Teknik Informatika.

DAFTAR PUSTAKA

<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>

<https://medium.com/@ethannam/understanding-the-levenshtein-distance-equation-for-beginners-c4285a5604f0>

<https://dasarpemrogramangolang.novalagung.com/>

<https://reactjs.org/docs/getting-started.html#react-for-beginners>

<https://blog.logrocket.com/how-to-make-http-requests-like-a-pro-with-axios/>

<https://github.com/axios/axios>

<https://www.taniarascia.com/getting-started-with-react/>