

TUGAS BESAR III
**PEMANFAATAN ALGORITMA *STRING MATCHING* SERTA REGULAR
EXPRESSION DALAM APLIKASI *DNA PATTERN MATCHING***

LAPORAN
Diajukan sebagai salah satu tugas mata kuliah
IF2211 Strategi Algoritma Semester II
Tahun Akademik 2021-2022

Oleh
Kelompok 51 - DNATrain
Dzaky Fattan Rizqullah 13520003
Yohana Golkaria Nainggolan 13520053
Rifqi Naufal Abdjul 13520062



SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2022

DAFTAR ISI

DAFTAR ISI	2
BAB I DESKRIPSI TUGAS	3
BAB II LANDASAN TEORI	5
2.1 Algoritma	5
2.1.1 Algoritma Knuth-Morris-Pratt	5
2.1.2 Algoritma Boyer-Moore	6
2.2 Aplikasi Web yang Dibangun	7
BAB III ANALISIS PEMECAHAN MASALAH	8
3.1 Langkah Pemecahan Masalah	8
3.2 Fitur Fungsional Web	9
3.2.1. Fungsional Front-end	9
3.2.2 Fungsional Back-end	10
BAB IV IMPLEMENTASI DAN PENGUJIAN	12
4.1 Spesifikasi Teknis Program	12
4.1.1 Struktur Data	12
4.1.2 Fungsi dan Prosedur	13
4.2 Tata Cara Penggunaan Program	14
4.4 Hasil Pengujian	16
4.5 Analisis Hasil Pengujian	25
BAB V KESIMPULAN DAN SARAN	26
5.1 Kesimpulan	26
5.2 Saran	26
5.2 Komentar	27
LINK TERKAIT	27
DAFTAR PUSTAKA	27

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: , 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.
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.

Spesifikasi Program:

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

BAB II

LANDASAN TEORI

2.1 Algoritma

2.1.1 Algoritma Knuth-Morris-Pratt

Pada algoritma *brute force*, setiap kali ditemukan ketidakcocokan *pattern* dengan teks, maka *pattern* digeser satu karakter ke kanan, sedangkan pada algoritma KMP, dilakukan pemeliharaan informasi yang berguna untuk menghitung jumlah pergeseran yang dibutuhkan. Algoritma ini menggunakan informasi tersebut untuk membuat pergeseran yang lebih jauh, tidak hanya satu karakter seperti pada algoritma *brute force*.

Algoritma KMP mengembangkan algoritma *brute force* menggunakan prefix dan suffix. Sebagai contoh, pada *pattern* $T = \text{'abcabd'}$, jika terjadi ketidakcocokan pada d ($j = 5$), kita cari suffix dari $T[0..j-1]$ yang juga merupakan prefix dari $T[0..j-1]$. Dalam kasus ini, prefix ‘ab’ ($T[0..1]$) sama dengan suffix ‘ab’ ($T[3..4]$). Kita tahu bahwa prefix ‘ab’ sudah cocok, maka kita geser T sehingga prefix ‘ab’ ada di posisi suffix ‘ab’ dengan menggeser T sejauh 3 karakter.

Contoh:

i	:	012345678
S	:	abcabcabd
T	:	abcabd
j	:	012345

Terjadi ketidakcocokan pada $d(j=5)$. Geser T sehingga prefix ‘ab’ ada di posisi suffix ‘ab’

i	:	012345678
S	:	abcabcabd
T	:	abcabd
j	:	012345

Kita sudah tahu bahwa $T[0..1]$ cocok dengan $S[3..4]$, sehingga pencocokan dimulai dari $j = 2$. Pada akhirnya ditemukan bahwa T cocok dengan S di $i = 3$.

Untuk mempercepat pencocokan, dilakukan preprocessing pada pattern untuk membentuk tabel yang berisi berapa jauh pattern akan digeser jika terjadi ketidakcocokan pada j tertentu berdasarkan aturan $\text{prefix } T[0..j-1] = \text{suffix } T[0..j-1]$. Preprocessing ini dinamakan border function.

2.1.2 Algoritma Boyer-Moore

Algoritma Boyer-Moore mengembangkan brute force dengan cara mencari sejauh apa pattern bisa digeser jika terjadi ketidakcocokan berdasarkan posisi karakter pada pattern. Proses pencocokan string dengan pattern dilakukan dari ujung kanan pattern.

Contoh:

i : 0123456789

S: abcceabcb

T: abcbd

j : 01234

Terjadi ketidakcocokan pada $i = 3$. Kita lihat bahwa $S[3] = 'c'$. Lalu kita cari dimana posisi 'c' terakhir di T, yang ada pada $j = 2$. Kita geser T sehingga 'c' terakhir di T ada pada posisi ketidakcocokan, yaitu dengan menggeser T sejauh satu karakter.

i : 0123456789

S: abcceabcb

T: abcbd

j : 01234

Terjadi ketidakcocokan pada $i = 5$. Kita lihat bahwa $S[5] = 'a'$ dan 'a' terakhir ada di $j = 0$. Kita geser T sejauh 4 karakter.

i : 0123456789

S: abcceabcb

T: abcbd

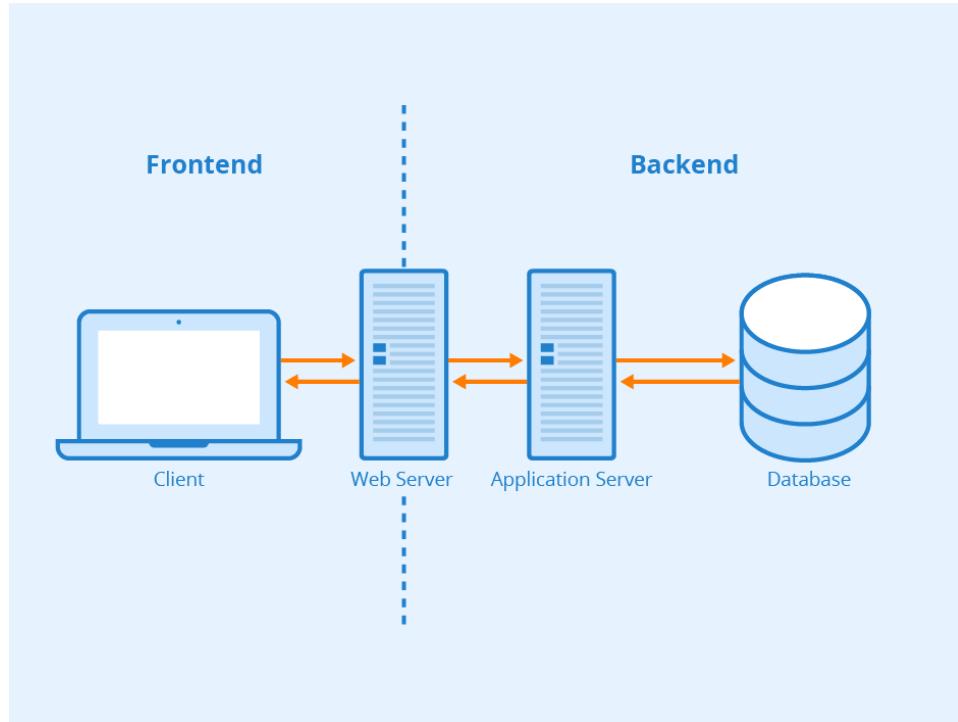
j : 01234

Sehingga ditemukan kecocokan T pada S.

Untuk mempercepat pencocokan, dilakukan preprocessing untuk mencari posisi terakhir tiap karakter yang ada pada pattern. Karakter yang tidak ada pada pattern bernilai -1. Preprocessing ini disebut last occurrence function.

2.2 Aplikasi Web yang Dibangun

Web yang kami bangun untuk penyelesaian tugas ini berstruktur *microservices*, yang berarti web dibagi menjadi beberapa bagian (dalam kasus ini, frontend, dan backend) sesuai dengan tugasnya masing masing.



Gambar 2.2.1 Penggambaran struktur web arsitektur *microservice*

Bagian frontend berfungsi sebagai bagian interface dari aplikasi yang dapat digunakan oleh pengguna. Frontend ini biasa bersifat *client-side*, berarti dijalankan dan di *compile* dalam mesin pengguna, berdasarkan program yang dikirimkan dari server. Bagian backend berfungsi sebagai bagian server dari aplikasi yang menerima dan memproses kiriman dari bagian frontend pengguna. Backend dijalankan pada server dan dapat menerima *request* dari frontend dan berkomunikasi dengan database.

Keuntungan struktur *microservices* ini adalah adanya sekat jelas antara bagian yang dijalankan oleh klien (frontend) dan bagian server (backend) sehingga bagian frontend dapat dijalankan *client-side* yang dapat meringankan beban backend. Selain itu dikarenakan dipisahkan, pembangunan web dalam tim lebih mudah karena pembagian tugas yang jelas.

BAB III

ANALISIS PEMECAHAN MASALAH

3.1 Langkah Pemecahan Masalah

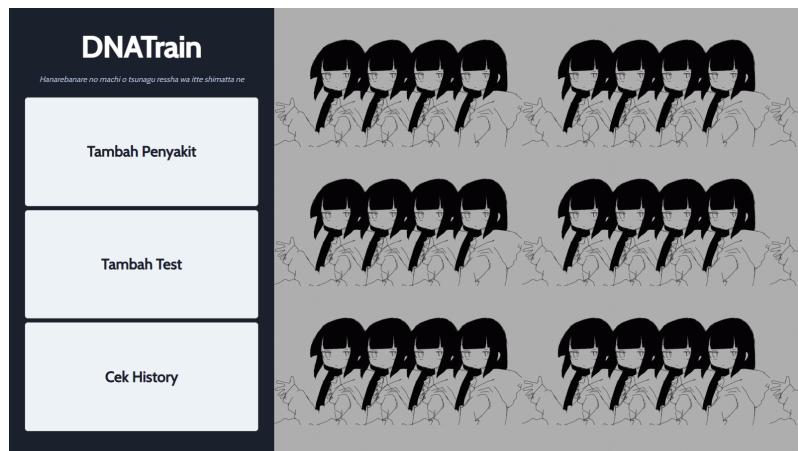
Pemecahan masalah dimulai dengan memahami spesifikasi aplikasi secara keseluruhan, serta memahami basis pengembangan aplikasi tersebut, yang dalam hal ini aplikasi diimplementasikan dengan berbasis website dengan pembagian *frontend* dan *backend* yang jelas. Fitur-fitur aplikasi yang dibuat serta implementasi alternatif solusinya adalah sebagai berikut.

- a. **Aplikasi dapat menerima input penyakit baru berupa nama penyakit dan sequence DNA-nya.** Masukan sequence DNA diberikan dalam bentuk input ketik ataupun file yang interaksi masukannya akan difasilitasi pada *frontend* website. Kemudian dilakukan sanitasi input menggunakan regex untuk memastikan masukan merupakan sequence DNA yang valid. Proses ini dilakukan pada *backend*. Modul regex pada Golang dimanfaatkan untuk melakukan sanitasi input tersebut. Selanjutnya *input* disimpan pada basis data di website.
- b. **Aplikasi dapat memprediksi seseorang menderita penyakit tertentu berdasarkan sequence DNA-nya.** Tes DNA dilakukan dengan menerima input nama pengguna, sequence DNA pengguna, dan nama penyakit yang diuji. Asumsi yang digunakan sequence DNA pengguna > sequence DNA penyakit. Penerimaan input dan sanitasinya mirip dengan fitur sebelumnya. Selanjutnya input diteruskan ke *backend* untuk dilakukan pencocokan *sequence* DNA menggunakan algoritma pencocokan *string* yang sudah dijelaskan pada sub bab 2.1. Selanjutnya, hasil pencocokan ditampilkan, yang berupa tanggal tes, nama pengguna, nama penyakit yang diuji, dan status hasil tes, yang difasilitasi pada *frontend*.
- c. **Aplikasi memiliki halaman yang menampilkan urutan hasil prediksi dengan kolom pencarian di dalamnya.** Kolom pencarian bekerja sebagai filter dalam menampilkan hasil. Masukan pencarian difasilitasi dalam *frontend* yang selanjutnya diteruskan ke *backend* yang memanfaatkan regex agar mampu menerima berbagai *format input* dan menghasilkan keluaran yang sesuai dengan masukannya.
- d. **Menghitung tingkat kemiripan DNA pengguna dengan DNA penyakit pada tes DNA.** Bonus spesifikasi tugas dibuat dengan memodifikasi algoritma pencocokan *string* agar pengecekan juga dilakukan untuk mendapatkan tingkat kemiripan DNA tersebut. Perhitungannya menggunakan *Hamming Distance* karena cukup menggambarkan *behaviour* DNA yang sebenarnya. Algoritmanya ialah dengan mencocokkan karakter (dalam hal ini, A, C, T, atau G) pada DNA pengguna dengan DNA penyakit berdasarkan posisi *index*-nya saja. Misalnya, substring “ACTG” untuk rantai “GGT GTA GGT CAG TGC” memiliki tingkat kemiripan 80%, dengan memosisikan *index* pertama substring pada *index* ke-10 pada rantai.

3.2 Fitur Fungsional Web

3.2.1. Fungsional Front-end

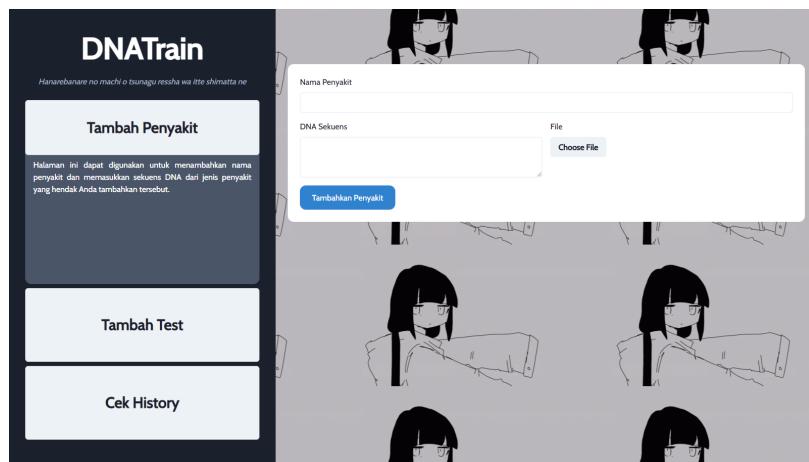
Pengimplementasian frontend aplikasi web ini menggunakan bahasa javascript dengan next.js framework. Untuk penampilan, kami menggunakan UI framework Chakra UI agar pengembangan lebih cepat dan rapih. Next.js mempunyai fitur routing yang berdasarkan file directory yang ada, berikut adalah page routing yang ada pada frontend website,



Gambar 3.2.1.1 Tampilan mainpage dari website

1. /tambah-penyakit

Halaman ini dapat digunakan untuk menambahkan nama penyakit dan memasukkan sekuen DNA dari jenis penyakit yang hendak ditambahkan tersebut.



Gambar 3.2.1.2 Tampilan /tambah-penyakit dari website

2. /tambah-test

Halaman ini dapat digunakan untuk menambahkan nama penyakit dan memasukkan sekuen DNA dari jenis penyakit yang hendak Anda tambahkan tersebut.

Gambar 3.2.1.3 Tampilan /tambah-test dari website

3. /cek-history

Halaman ini menampilkan urutan hasil prediksi dengan kolom pencarian di dalamnya. Pengguna dapat melihat history hasil prediksinya dengan menggunakan kolom pencarian dengan menginput nama pengguna.

NO	TANGGAL	NAMA PASIEN	PENYAKIT	TINGKAT KECOCOKAN	HASIL
1	23/04/2022, 22:03:51	iqi	pilek	1.00	Benar
2	16/04/2022, 15:16:19	rifqi	badut	0.95	Benar
3	16/04/2022, 15:16:25	rifqi	badut	0.95	Benar
4	27/04/2022, 15:35:58	Mayano	gamerz	0.83	Benar
5	27/04/2022, 05:46:51	jeki	pilek	0.67	Salah
6	23/04/2022, 21:59:47	rifqi	badut	0.50	Salah
7	27/04/2022, 05:47:40	kucing	pilek	0.50	Salah
8	27/04/2022, 12:15:52	badut	badut	0.50	Salah
9	28/04/2022, 10:25:38	Sy	gamerz	0.50	Salah
10	23/04/2022, 22:01:37	iqi	pilek	0.43	Salah
11	27/04/2022, 05:37:10	Rifqqq	pilek	0.33	Salah
12	27/04/2022, 05:41:51	Rifqqq	pilek	0.33	Salah
13	27/04/2022, 05:45:43	Rifqqq	pilek	0.33	Salah

Gambar 3.2.1.4 Tampilan /cek-history dari website

3.2.2 Fungsional Back-end

Pengimplementasian backend aplikasi web ini menggunakan bahasa golang dengan echo framework. Backend juga di-deploy dengan heroku menggunakan heroku cli pada url <https://tubes-3-stima-backend.herokuapp.com>. Dengan menggunakan framework echo tersebut, dapat dibuat api endpoint routes sesuai dengan penjelasan di bawah ini,

1. /penyakit/get (GET)

Digunakan untuk mendapatkan seluruh nama penyakit yang berada pada database

2. /penyakit/create (POST)

Digunakan untuk menambahkan penyakit dengan mengirimkan request berisikan nama penyakit dan dna sequence dari penyakit tersebut

3. /test/get (GET)

Digunakan untuk mendapatkan seluruh hasil test yang telah dilakukan dalam aplikasi web

4. /test/create (POST)

Digunakan untuk melakukan test DNA sequence terhadap suatu penyakit dengan mengirimkan request berisikan nama pasien, nama penyakit, metode penyocokan, dan dna dari pasien tersebut.

Untuk menyimpan data penyakit dan hasil tes, diimplementasikan database MySQL pada Planetscale (<https://planetscale.com>). Dalam interaksi antara server dan database, menggunakan modul go-sql-driver yang memberikan mysql driver untuk melakukan koneksi dan query pada database melalui server.

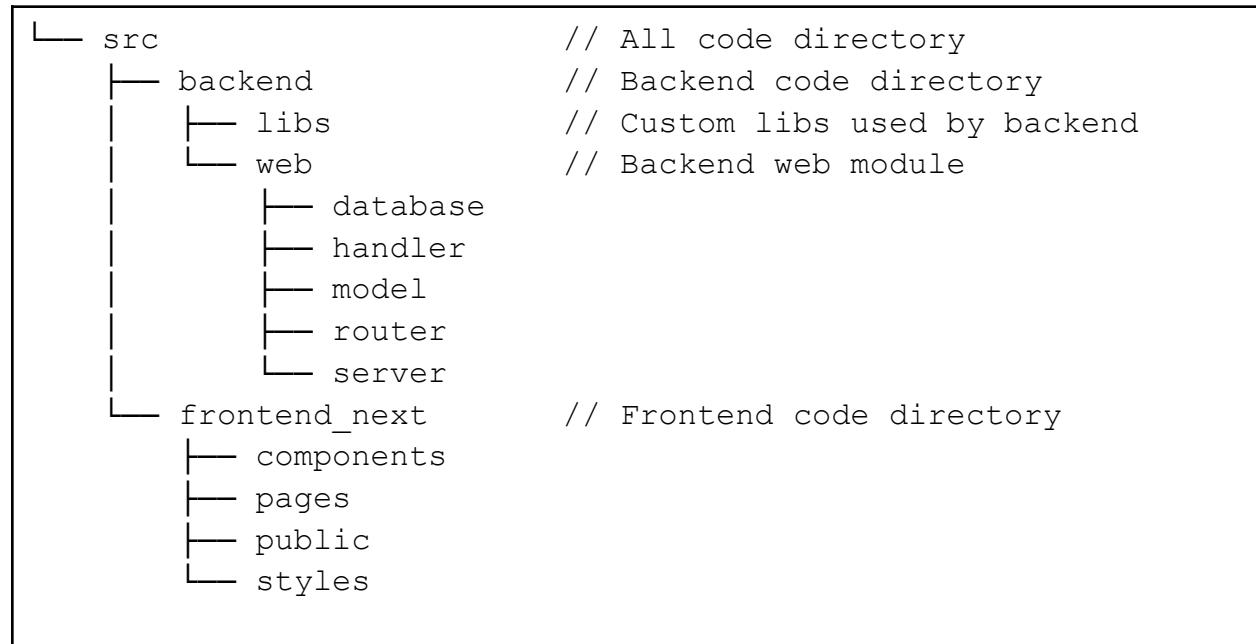
BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Spesifikasi Teknis Program

4.1.1 Struktur Data

Secara garis besar, struktur data untuk aplikasi web ini dapat dilihat pada ilustrasi di bawah (Juga dapat dilihat pada repositori yang linknya tertera pada bab Link Repository).



Gambar 4.1.1 Bagan struktur data aplikasi web

- a. *backend/libs*, merupakan folder yang berisi seluruh *custom library* yang dibutuhkan aplikasi web untuk melakukan pencocokan *string*. Di dalamnya terdapat modul *kmp*, *boyermoore*, *hd* (untuk perhitungan *hamming distance*), dan *regex*.
- b. *backend/web*, merupakan folder yang berisi seluruh file dan modul yang digunakan untuk menjalankan fungsi *backend web*. Di dalamnya terdapat:
 - i. *database*, yang mengatur fungsi koneksi web ke basis data;
 - ii. *handler*, yang mengatur fungsi *backend* secara keseluruhan, di sini fungsi-fungsi pada *libs* dipanggil untuk dimanfaatkan;
 - iii. *model*, mengatur pemodelan struktur basis data dan struktur lainnya yang diperlukan untuk keberjalanannya *backend*;
 - iv. *router*, mengatur navigasi antar sub-laman pada web;
 - v. *server*, mengatur keberjalanannya *server backend*.
- c. *frontend_next*, merupakan folder yang berisi seluruh file dan modul yang dibutuhkan untuk menjalankan *frontend web*. Folder-foler penting di antaranya,

components mengatur fitur forms dan sidebar web, sementara pages mengatur sub-halaman dan keseluruhan fungsi pada *frontend* web.

4.1.2 Fungsi dan Prosedur

Fungsi dan Prosedur yang akan dijelaskan hanyalah yang berkaitan langsung dengan fitur-fitur aplikasi web, penjabarannya sebagai berikut.

- a. BoyerMoore(string seq, string pattern) -> double, fungsi yang menjalankan algoritma pencocokan *string* dengan metode *Boyer-Moore*, menerima masukan *sequence DNA* pasien dan *pattern DNA* penyakit (keduanya bertipe *string*) dan keluarannya bertipe *double* yang menunjukkan tingkat kemiripan. (0 jika tidak mirip sama sekali, 1 jika *exact*).
- b. KMP(string seq, string pattern) -> double, fungsi yang menjalankan algoritma pencocokan *string* dengan metode *Boyer-Moore*, menerima masukan *sequence DNA* pasien dan *pattern DNA* penyakit (keduanya bertipe *string*) dan keluarannya bertipe *double* yang menunjukkan tingkat kemiripan. (0 jika tidak mirip sama sekali, 1 jika *exact*).
- c. HammingDistance(string str1, string str2) -> int, fungsi yang menjalankan perhitungan *hamming distance* antara kedua masukan str1 dan str2 yang bertipe *string*, dan keluarannya berupa *integer* tingkat kecocokan.
- d. SanitizeDNA(string dna) -> error, fungsi yang menjalankan sanitasi masukan *DNA*, menerima *string DNA* dan keluarannya berupa *error* yang bernilai *null* jika masukan sesuai, dan tidak *null* jika tidak sesuai.
- e. SplitSearch(string q) -> (string, string, error), fungsi yang mengatur masukkan kata kunci pencarian.
- f. Connect() -> *sql.DB, fungsi yang mengatur koneksi *backend* ke basis data.
- g. Init(), prosedur menjalankan *backend* secara keseluruhan.
- h. CreatePenyakit(echo.Context c) -> error, fungsi yang menjalankan fitur penambahan penyakit baru.
- i. GetPenyakit(echo.Context c) -> error, fungsi yang menjalankan fitur menerima suatu penyakit.
- j. GetTest(echo.Context c) -> error, fungsi yang menjalankan fitur menerima suatu test.
- k. CreateTest(echo.Context c) -> error, fungsi yang menjalankan fitur penambahan tes baru.

4.2 Tata Cara Penggunaan Program

Program dapat digunakan baik dengan menjalankan server di lokal maupun mengakses situs <https://tubes-3-stima.vercel.app/>. Bila ingin menjalankan server di lokal, jalankan *backend* dan *frontend* dengan membuka *Terminal* dan melakukan langkah-langkah berikut.

- *Backend*

1. Pindah direktori ke `src/backend`;
2. Duplikasi “`.env.example`” dan *rename* menjadi “`.env`”. Masukkan `DB_URL` menggunakan url milik sendiri. Misal, untuk contoh ini: `root@tcp(localhost:1234)/dnatrain?parseTime=true`. `dnatrain` Ini merupakan nama basis data lokal yang akan diakses *backend*. Pastikan argumen terakhir (`?parseTime=true`) tertera;
3. Install Golang *Packages* dengan menjalankan perintah `go get`;
4. Jalankan server dengan menjalankan perintah `go run main.go`;
5. Cek apakah server sudah berjalan pada `localhost:$PORT`.

- *Frontend*

1. Pindah direktori ke `src/frontend_next`;
2. Install Node *Packages* dengan menjalankan perintah `npm i`;
3. Buka file `axios.js` pada folder `src/frontend_next/lib`, pada variabel string `baseURL`, ubah nilai string (dari `"https://tubes-3-stima-backend.herokuapp.com"`) menjadi `"http://localhost:$PORT/"`, nilai `$PORT` diubah menjadi angka port yang sebenarnya pada file “`.env`” yang telah dijelaskan pada bagian *backend*.
4. Jalankan server dengan menjalankan perintah `npm run dev`;
5. Cek apakah server sudah berjalan pada `localhost:$PORT`.

Bila menggunakan basis data lokal, kemungkinan akan membutuhkan persiapan tambahan. Langkah-langkahnya sebagai berikut. Tiap langkah dapat disesuaikan dengan DBMS yang digunakan (sesuaikan dengan `DB_URL` yang telah dimasukkan dalam file `.env` sebelumnya).

1. Buat *database* baru bernama `dnatrain`. (Cocokkan dengan nama database yang tertulis pada `DB_URL`).
2. Tambahkan tabel `penyakit` dengan atribut `nama` dengan tipe `varchar(255)` (string) dan atribut `dna` dengan tipe `varchar(255)` juga.
3. Tambahkan tabel `test` dengan atribut `tanggal` dengan tipe `datetime`, `nama` dengan tipe `varchar(255)`, `nama_penyakit` dengan tipe `varchar(255)`, `kecocokan` dengan tipe `double`, dan atribut `cocok` dengan tipe `boolean` juga.

```
C:\Windows\System32\cmd.exe - mysql -u root
MariaDB [dnatrain]> describe penyakit;
+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| nama  | varchar(255) | YES  |     | NULL    |         |
| dna   | varchar(255) | YES  |     | NULL    |         |
+-----+-----+-----+-----+-----+
2 rows in set (0.010 sec)

MariaDB [dnatrain]> describe test;
+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| tanggal    | datetime  | YES  |     | NULL    |         |
| nama       | varchar(255) | YES  |     | NULL    |         |
| nama_penyakit | varchar(255) | YES  |     | NULL    |         |
| kecocokan  | double    | YES  |     | NULL    |         |
| cocok      | tinyint(1) | YES  |     | NULL    |         |
+-----+-----+-----+-----+-----+
5 rows in set (0.010 sec)

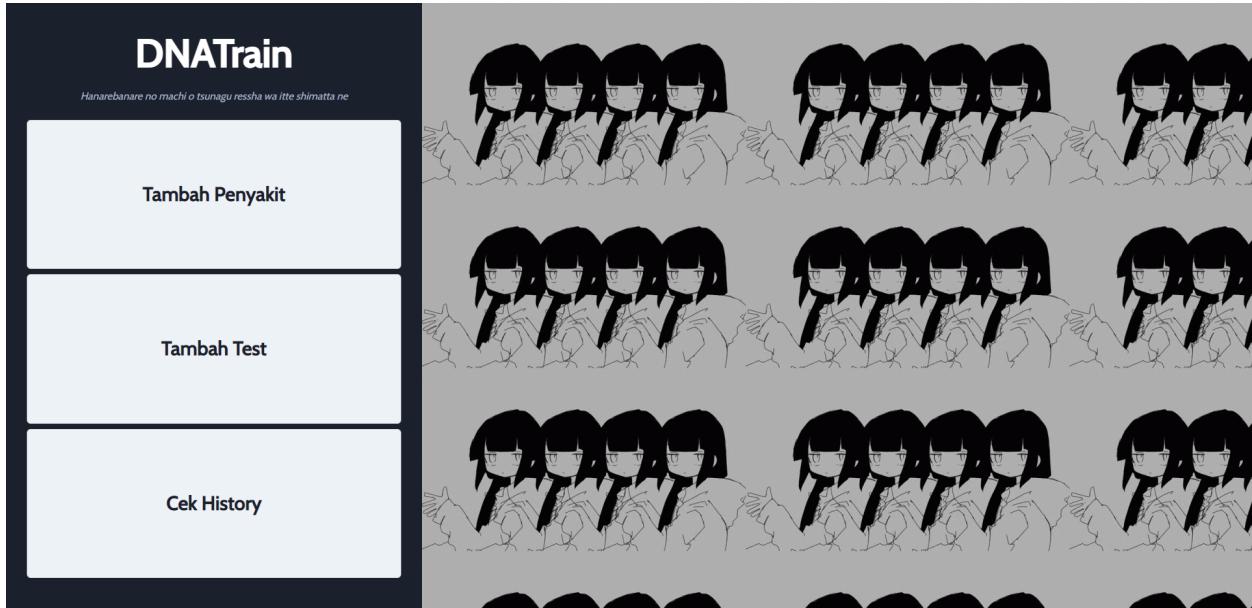
MariaDB [dnatrain]>
```

Gambar 4.2.1 Tampilan hasil pembuatan basis data di lokal

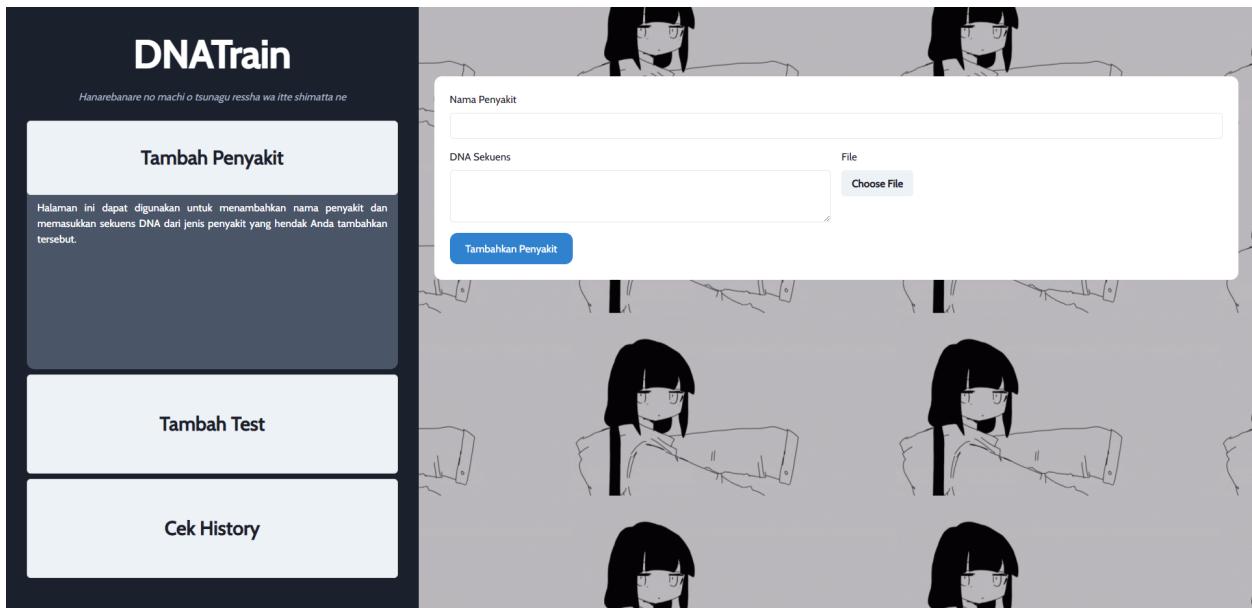
Aplikasi web yang berhasil berjalan akan menampilkan *main page* yang terlihat seperti pada gambar pada sub bab 4.4. Aplikasi web dibuat sesederhana mungkin sehingga keseluruhan fitur aplikasi web dapat diakses hanya dengan memilih menu yang sesuai di bagian kiri halaman dan mengisi masukan yang dibutuhkan.

4.4 Hasil Pengujian

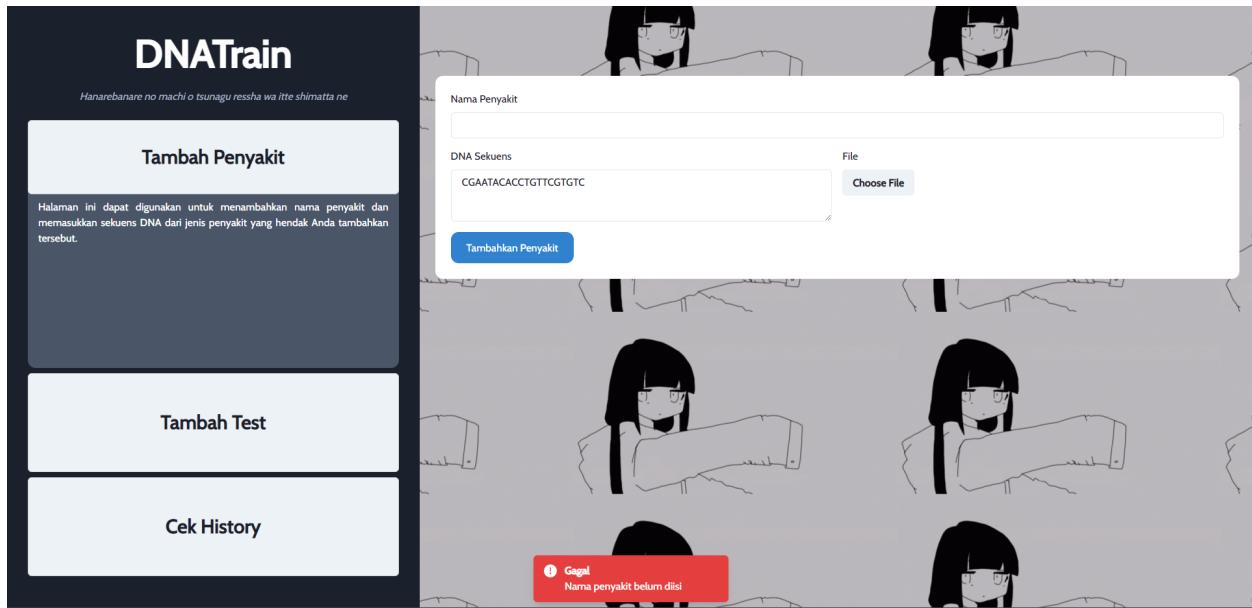
Berikut akan ditampilkan tangkapan layar saat pengujian aplikasi web. Perlu diperhatikan bahwa pengujian dilakukan pada website hasil *deploy*, bukan di lokal.



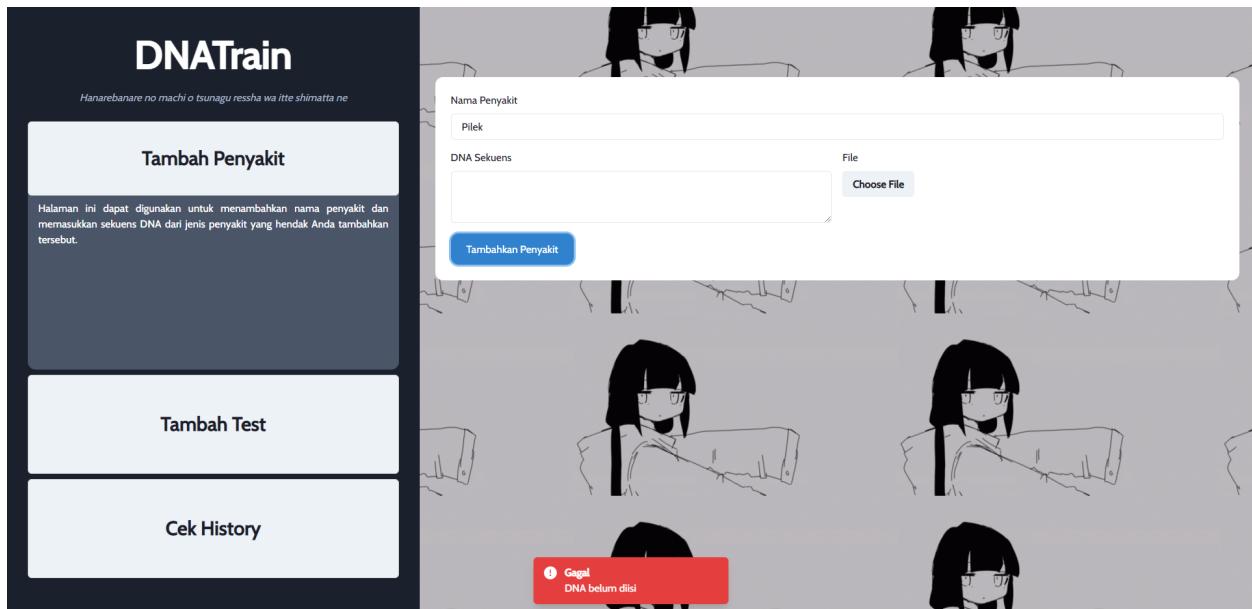
Gambar 4.4.1 Tampilan laman utama



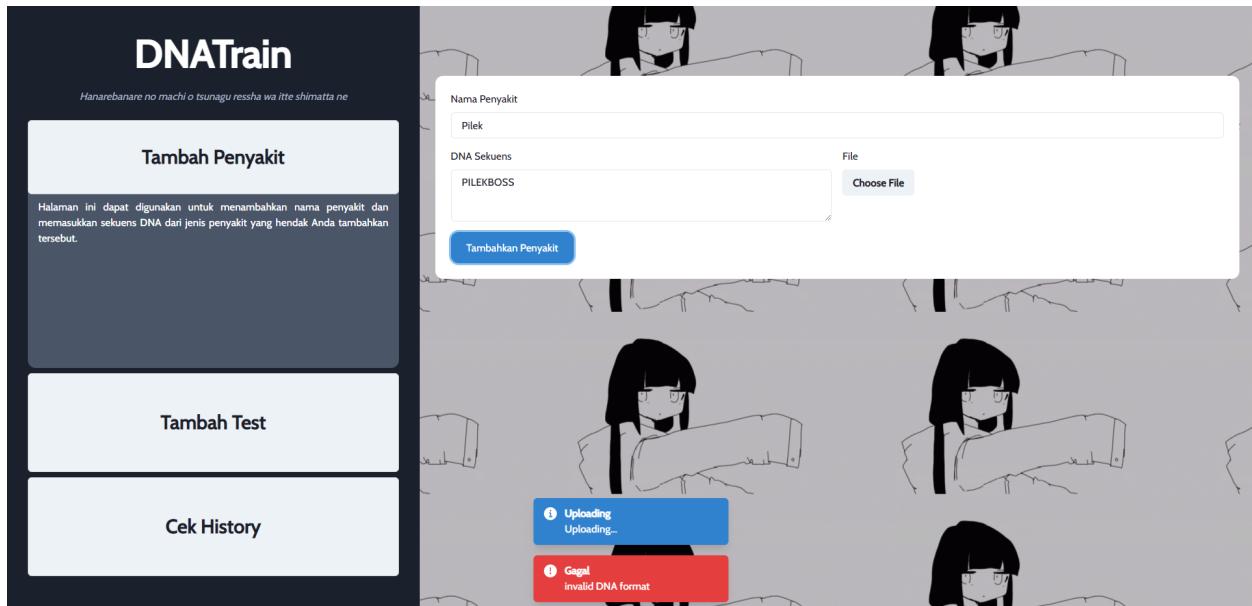
Gambar 4.4.2 Tampilan laman “Tambah Penyakit”



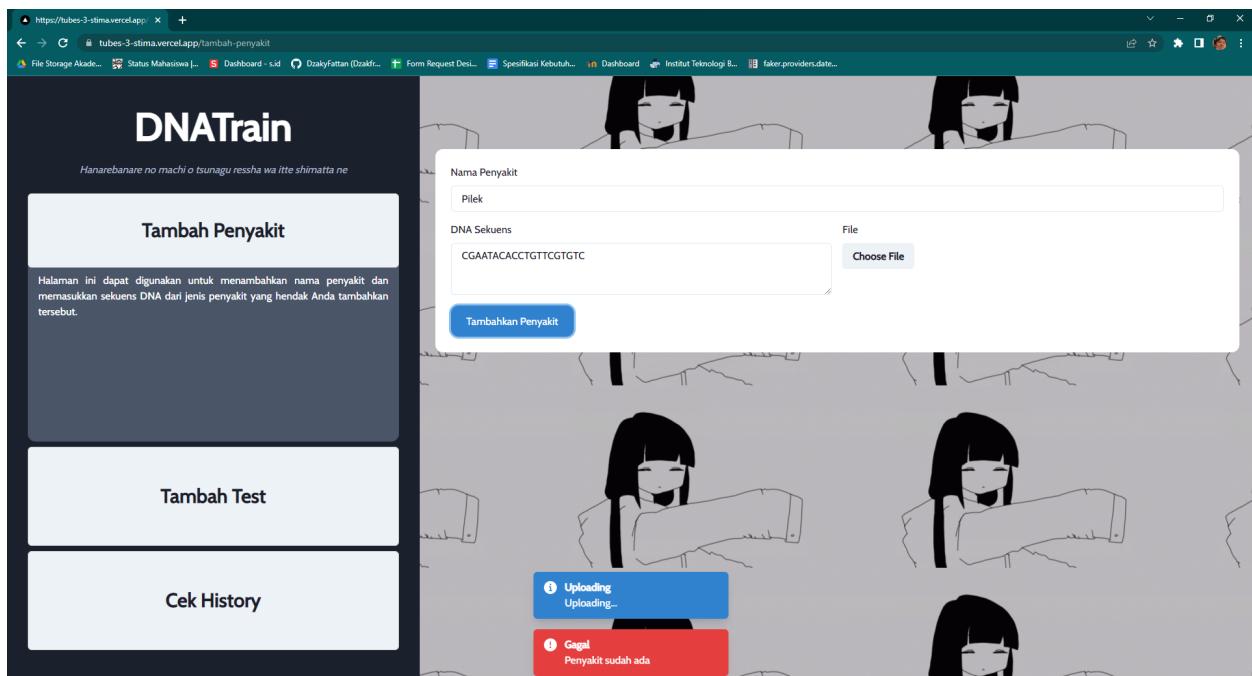
Gambar 4.4.3 Gagal menambah penyakit karena nama penyakit belum dimasukkan



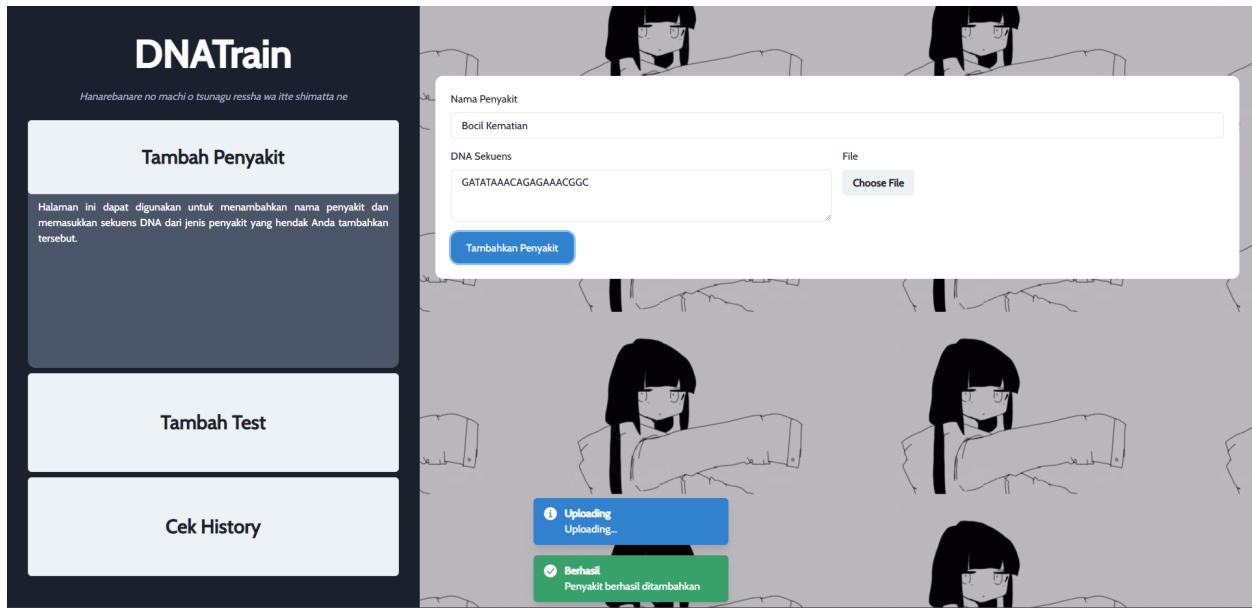
Gambar 4.4.4 Gagal menambah penyakit karena *sequence* DNA belum dimasukkan



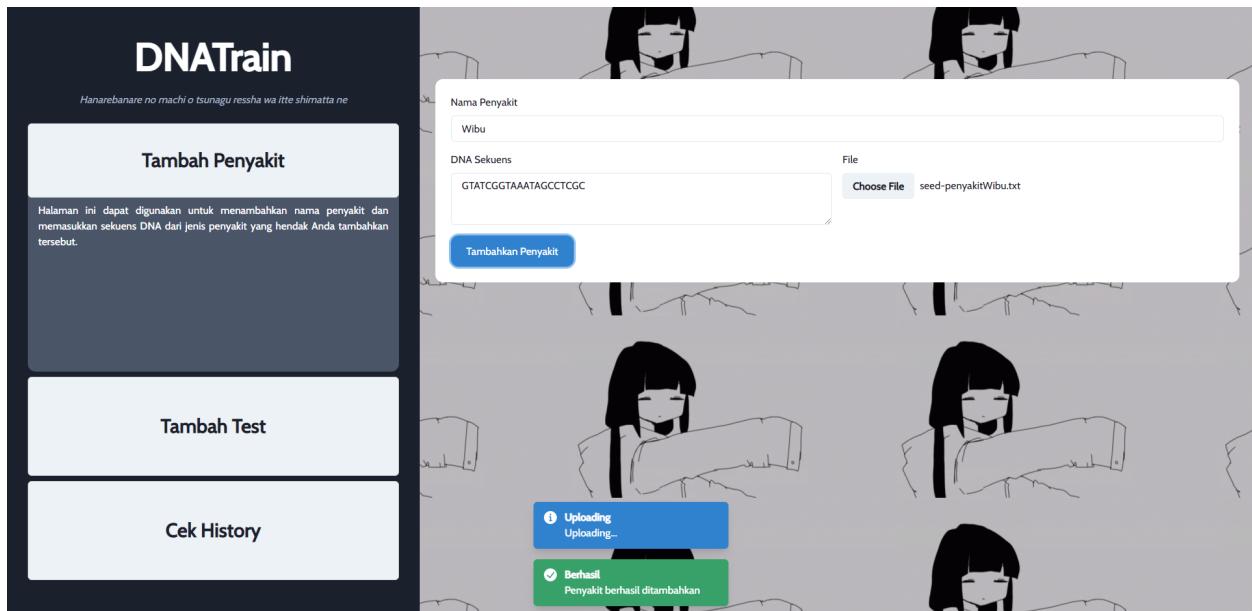
Gambar 4.4.5 Gagal menambah penyakit karena *sequence* DNA tidak valid



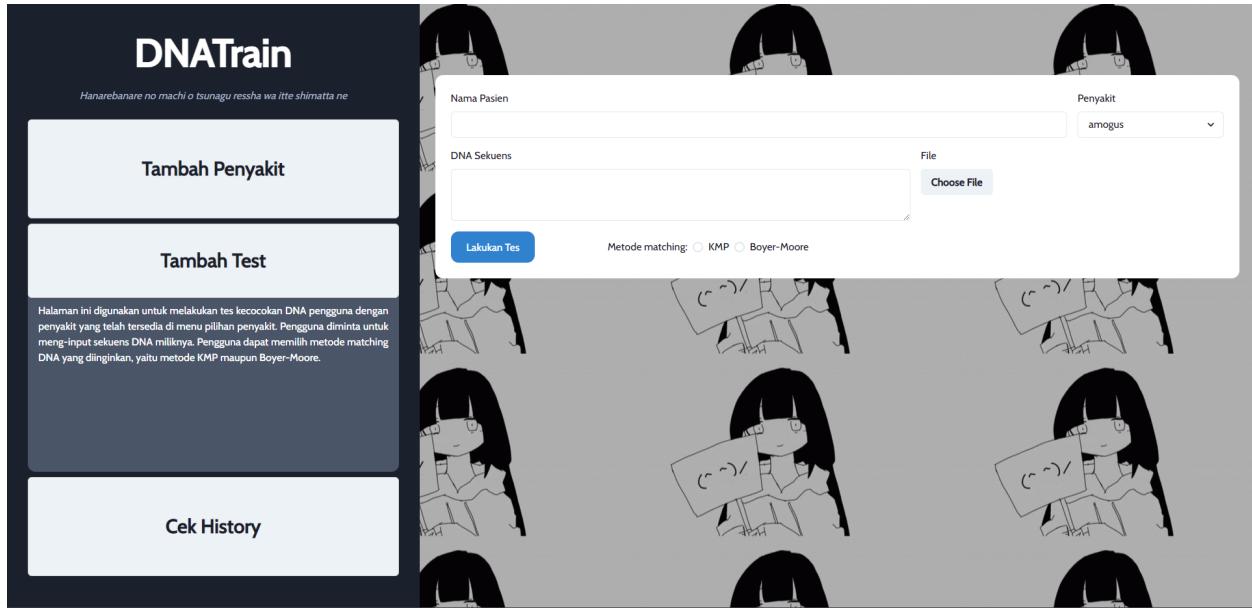
Gambar 4.4.6 Gagal menambah penyakit karena penyakit sudah ada



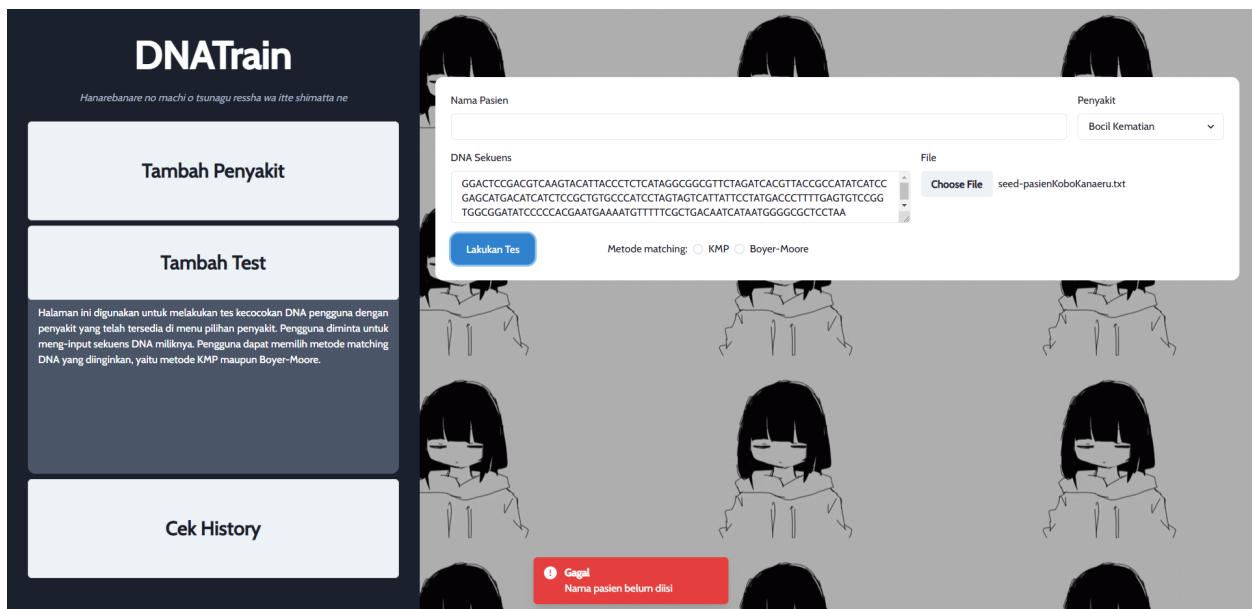
Gambar 4.4.7 Berhasil menambahkan penyakit dari masukan pada kotak masukan



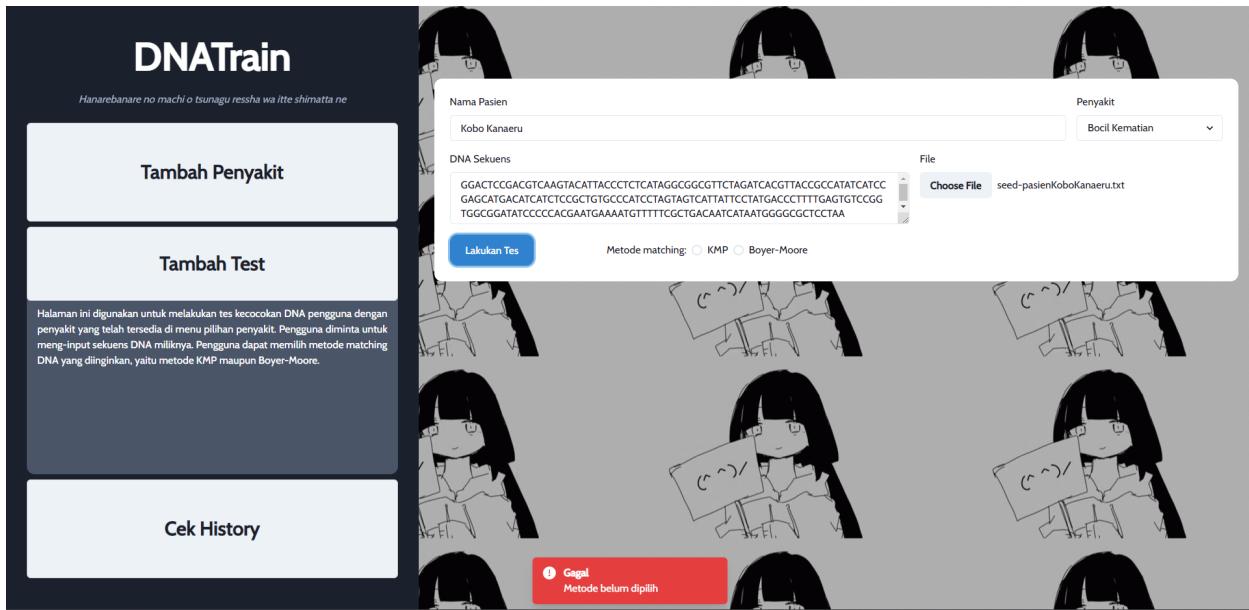
Gambar 4.4.8 Berhasil menambahkan penyakit dari masukan teks (*open file*)



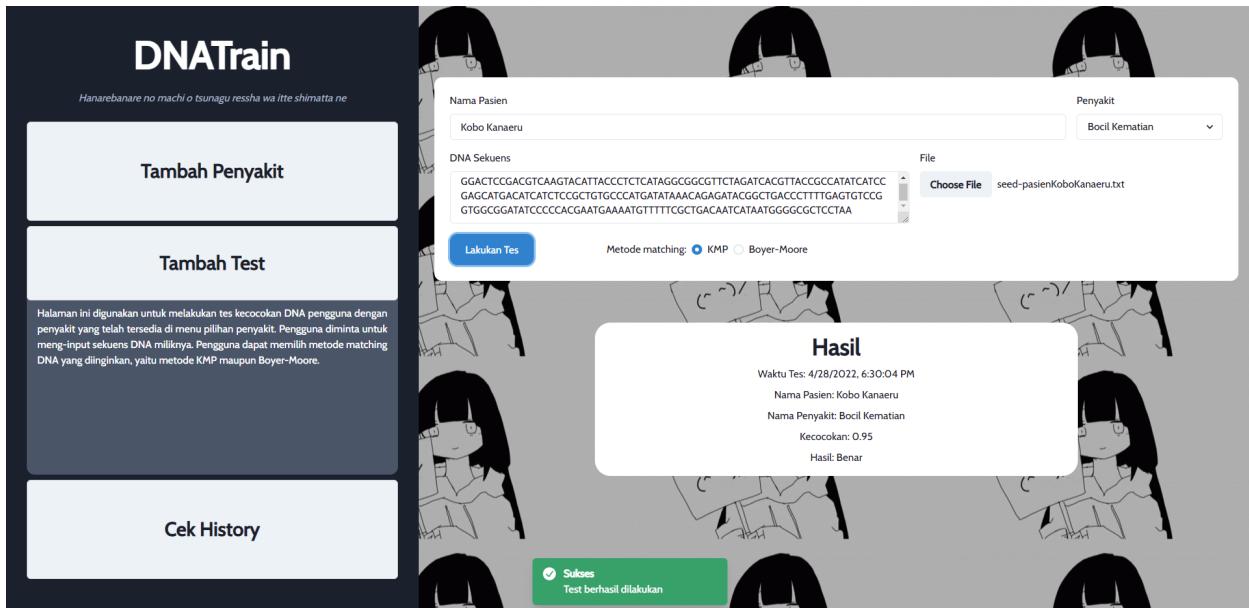
Gambar 4.4.9 Tampilan laman “Tambah Test”



Gambar 4.4.10 Gagal menambah penyakit karena nama pasien belum dimasukkan



Gambar 4.4.11 Gagal menambah penyakit karena metode *matching* belum dipilih



Gambar 4.4.12 Test berhasil dengan hasil “Benar”

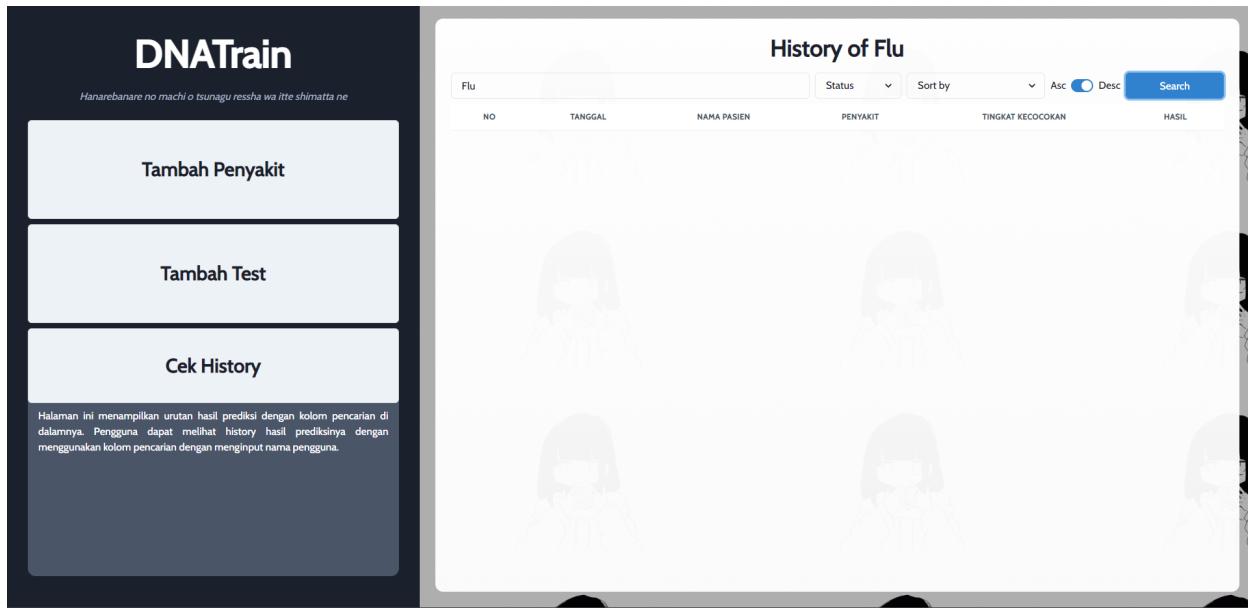
The screenshot shows the DNATrain web application interface. On the left, there are three main buttons: "Tambah Penyakit" (Add Disease), "Tambah Test" (Add Test), and "Cek History" (Check History). The "Cek History" button is highlighted. The central part of the screen displays a DNA test form. It has fields for "Nama Pasien" (Patient Name) set to "Jeki", "Penyakit" (Disease) set to "Bocil Kematian", and "DNA Sekuens" (DNA Sequence) containing a long string of nucleotides. Below these fields are buttons for "Lakukan Tes" (Perform Test) and "Choose File". A "Metode matching:" dropdown is set to "Boyer-Moore". To the right, a large speech bubble-like area titled "Hasil" (Result) shows the test results: "Waktu Tes: 4/28/2022, 6:31:36 PM", "Nama Pasien: Jeki", "Nama Penyakit: Bocil Kematian", "Kecocokan: 0.5", and "Hasil: Salah". Below this, a blue progress bar says "Uploading Uploading..." and a green success message says "Sukses Test berhasil dilakukan". The background features a repeating pattern of cartoon characters.

Gambar 4.4.13 Test berhasil dengan hasil “Salah”

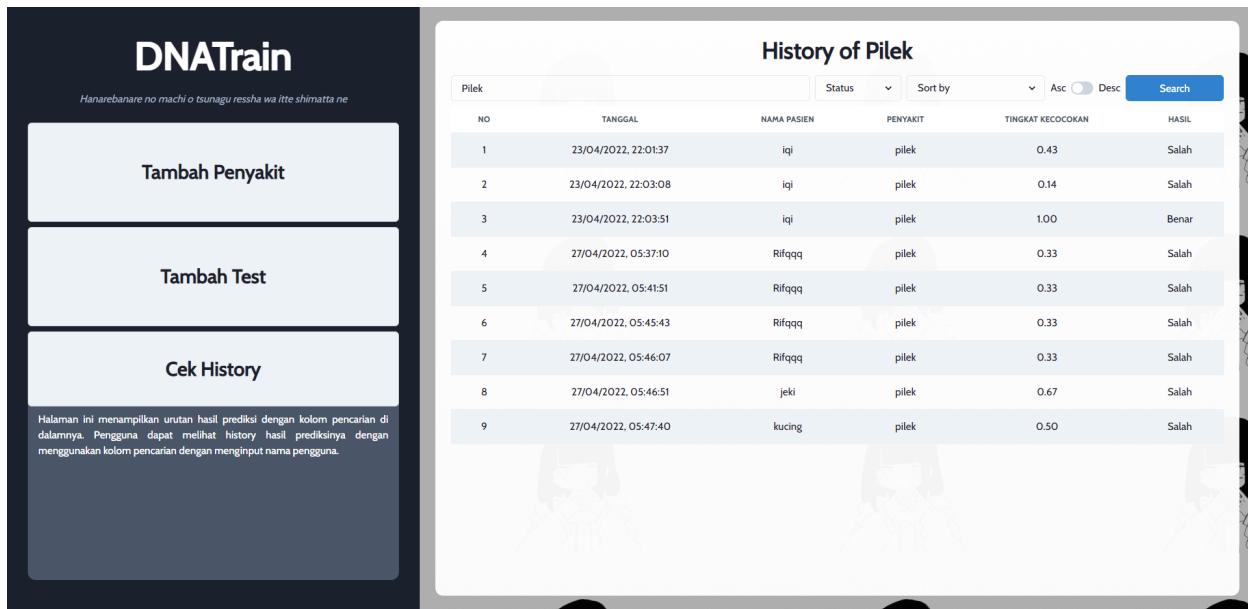
The screenshot shows the "History" page of the DNATrain web application. On the left, there is a sidebar with the same three buttons: "Tambah Penyakit", "Tambah Test", and "Cek History". The "Cek History" button is highlighted. The main content area is a table titled "History" with columns: NO, TANGGAL, NAMA PASIEN, PENYAKIT, TINGKAT KECOCOKAN, and HASIL. The table lists 13 rows of test results. The first few rows show results for patient "rifqi" with "badut" as the disease and "Benar" as the result. Rows 3 through 13 show results for patient "iqi" with "pilek" as the disease and "Salah" as the result. The last row shows a result for patient "jeki" with "kucing" as the disease and "Salah" as the result. The table includes search and sort functionality at the top.

NO	TANGGAL	NAMA PASIEN	PENYAKIT	TINGKAT KECOCOKAN	HASIL
1	16/04/2022, 15:16:19	rifqi	badut	0.95	Benar
2	16/04/2022, 15:16:25	rifqi	badut	0.95	Benar
3	23/04/2022, 21:59:47	rifqi	badut	0.50	Salah
4	23/04/2022, 22:01:37	iqi	pilek	0.43	Salah
5	23/04/2022, 22:03:08	iqi	pilek	0.14	Salah
6	23/04/2022, 22:03:51	iqi	pilek	1.00	Benar
7	27/04/2022, 05:37:10	Rifqqq	pilek	0.33	Salah
8	27/04/2022, 05:41:51	Rifqqq	pilek	0.33	Salah
9	27/04/2022, 05:45:43	Rifqqq	pilek	0.33	Salah
10	27/04/2022, 05:46:07	Rifqqq	pilek	0.33	Salah
11	27/04/2022, 05:46:51	jeki	pilek	0.67	Salah
12	27/04/2022, 05:47:40	kucing	pilek	0.50	Salah
13	27/04/2022, 12:01:52	badut	badut	0.50	Salah

Gambar 4.4.14 Tampilan laman “Cek History”



Gambar 4.4.15 Mencari berdasarkan nama penyakit, tidak ditemukan



Gambar 4.4.16 Mencari berdasarkan nama penyakit, ditemukan

The screenshot shows the DNATrain application's main interface on the left and a detailed view of a search result on the right.

Main Interface (Left):

- Header:** DNATrain
- Text:** Hanarebanare no machi o tsunagu ressha wa itte shinmatta ne
- Buttons:**
 - Tambah Penyakit
 - Tambah Test
 - Cek History
- Note:** Halaman ini menampilkan urutan hasil prediksi dengan kolom pencarian di dalamnya. Pengguna dapat melihat history hasil prediksinya dengan menggunakan kolom pencarian dengan menginput nama pengguna.

Search Result (Right):

Header: History of 30/04/2022

Search Bar: 30/04/2022, Status, Tanggal, Asc, Desc, Search

Table Headers: NO, TANGGAL, NAMA PASIEN, PENYAKIT, TINGKAT KECOCOKAN, HASIL

Data: There is no data present in the table.

Gambar 4.4.17 Mencari berdasarkan tanggal test, tidak ditemukan

The screenshot shows the DNATrain application's main interface on the left and a detailed view of a search result on the right.

Main Interface (Left):

- Header:** DNATrain
- Text:** Hanarebanare no machi o tsunagu ressha wa itte shinmatta ne
- Buttons:**
 - Tambah Penyakit
 - Tambah Test
 - Cek History
- Note:** Halaman ini menampilkan urutan hasil prediksi dengan kolom pencarian di dalamnya. Pengguna dapat melihat history hasil prediksinya dengan menggunakan kolom pencarian dengan menginput nama pengguna.

Search Result (Right):

Header: History of 28/04/2022

Search Bar: 28/04/2022, Status, Kecocokan, Asc, Desc, Search

Table Headers: NO, TANGGAL, NAMA PASIEN, PENYAKIT, TINGKAT KECOCOKAN, HASIL

Data:

NO	TANGGAL	NAMA PASIEN	PENYAKIT	TINGKAT KECOCOKAN	HASIL
1	28/04/2022, 17:55:03	Jeki	badut	1.00	Benar
2	28/04/2022, 18:30:05	Kobo Kanaeru	Bocil Kematian	0.95	Benar
3	28/04/2022, 10:25:38	Sy	gamerz	0.50	Salah
4	28/04/2022, 18:28:49	Kobo Kanaeru	Bocil Kematian	0.50	Salah
5	28/04/2022, 18:31:36	Jeki	Bocil Kematian	0.50	Salah
6	28/04/2022, 10:10:36	Saya	badut	0.25	Salah

Gambar 4.4.18 Mencari berdasarkan tanggal test, ditemukan

4.5 Analisis Hasil Pengujian

Hasil pengujian menunjukkan aplikasi web berhasil menjalankan spesifikasi yang sudah dijelaskan sebelumnya. Penjabarannya sebagai berikut.

- a. **Aplikasi dapat menerima input penyakit baru berupa nama penyakit dan sequence DNA-nya.** Aplikasi web dapat mengecek kebenaran input dan dapat menyimpan input tersebut
- b. **Aplikasi dapat memprediksi seseorang menderita penyakit tertentu berdasarkan sequence DNA-nya.** Aplikasi web dapat mengutilisasi algoritma *string matching* dengan baik, baik itu algoritma *KMP* ataupun *Boyer-Moore*.
- c. **Aplikasi memiliki halaman yang menampilkan urutan hasil prediksi dengan kolom pencarian di dalamnya.** Sanitasi masukan dengan regex berjalan dengan baik dan keluaran juga ditampilkan dengan baik.
- d. **Menghitung tingkat kemiripan DNA pengguna dengan DNA penyakit pada tes DNA.** Baik itu algoritma *KMP* ataupun *Boyer-Moore*, keduanya menghasilkan hasil yang mirip untuk kasus pencocokan di bawah 100% dan menghasilkan *verdict* yang benar pula.
- e. **Aplikasi web ter-deploy dengan baik.** Aplikasi web dapat diakses melalui pranala <https://tubes-3-stima.vercel.app/>.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan implementasi algoritma pada program dan eksperimen yang telah dilakukan, didapatkan kesimpulan sebagai berikut.

1. Telah dibangun suatu aplikasi berbasis web “DNATrain” yang merupakan sebuah aplikasi DNA *pattern matching*, berfungsi untuk mencocokkan pola rantai DNA dalam suatu *sequence DNA*.
2. Aplikasi web dapat menjalankan keseluruhan fungsi wajib yang telah dijelaskan pada spesifikasi program dalam sub bab Deskripsi Tugas.
3. Aplikasi web juga dibangun dengan fitur tambahan dapat mengukur tingkat kemiripan DNA pengguna dengan DNA penyakit pada tes DNA.
4. Aplikasi web memiliki *Tech Stacks* sebagai berikut.
 - a. Front-End : React
 - b. Back-End : Golang with Echo
 - c. Database : MySQL on Planetscape
5. Aplikasi web di-deploy pada *hosting provider* Vercel dan dapat berjalan dengan baik.

5.2 Saran

Berdasarkan implementasi algoritma pada program dan eksperimen yang telah dilakukan, penulis memberikan beberapa saran sebagai berikut.

1. Pada dasarnya, baik algoritma KMP maupun *Boyer-Moore* saja tidak dapat melakukan pencocokan string secara parsial, sehingga diperlukan modifikasi terhadap kedua algoritma tersebut. Meskipun demikian, pada saat testing, keduanya menampilkan hasil yang berbeda dikarenakan modifikasi tersebut. Algoritma *brute force* justru menghasilkan hasil yang lebih akurat untuk pencocokan parsial. Hal ini perlu dipertimbangkan agar algoritma pencocokan *string* dapat mengeluarkan hasil tes yang akurat.
2. Dikarenakan program ini dimanfaatkan dalam bidang kedokteran yang sangat mengutamakan ketepatan hasil, maka perlu ditekankan kembali bahwa akurasi hasil pencocokan sangat penting. Idealnya, kita tidak ingin aplikasi ini justru merugikan pengguna.
3. Referensi terkait DNA dan penyakit yang lebih komprehensif akan sangat membantu dalam mplementasikan algoritma pencocokan DNA ini.
4. Aplikasi web kami secara keseluruhan belum tentu bisa dikatakan sangat baik. Masih ada bagian-bagian kecil yang dapat diperbaiki agar tentunya menghasilkan aplikasi web yang lebih baik dari yang sekarang.

5.2 Komentar

Berikut beberapa komentar yang dapat penulis sampaikan terkait tugas besar ketiga yang telah dilaksanakan ini.

1. Untuk sebagian dari kami, tugas ini merupakan pengalaman pertama kami dalam memprogram dalam bahasa Golang.
2. Untuk sebagian dari kami pula, tugas ini merupakan pengalaman pertama kami dalam membuat aplikasi web *full-stack*, ditambah dengan integrasi database untuk keberlangsungan operasionalnya.
3. Referensi cara kerja DNA dan keterkaitannya dengan penyakit cukup membantu dalam mengambil keputusan mengenai algoritma pencocokan yang digunakan.

LINK TERKAIT

1. Repository: https://github.com/rifqi2320/Tubes3_13520003
2. Link Video: <https://youtu.be/H0xK1Q9C8Vk>
3. Link Website hasil *deploy*: <https://tubes-3-stima.vercel.app/>

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://dasarpemrogramangolang.novalagung.com/>

<https://github.com/labstack/echo>

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

<https://sequelize.org/docs/v6/getting-started/>

Waggner, Bill (1995). Pulse Code Modulation Techniques. Springer. p. 206. ISBN 9780442014360. Diakses 27 April 2022.