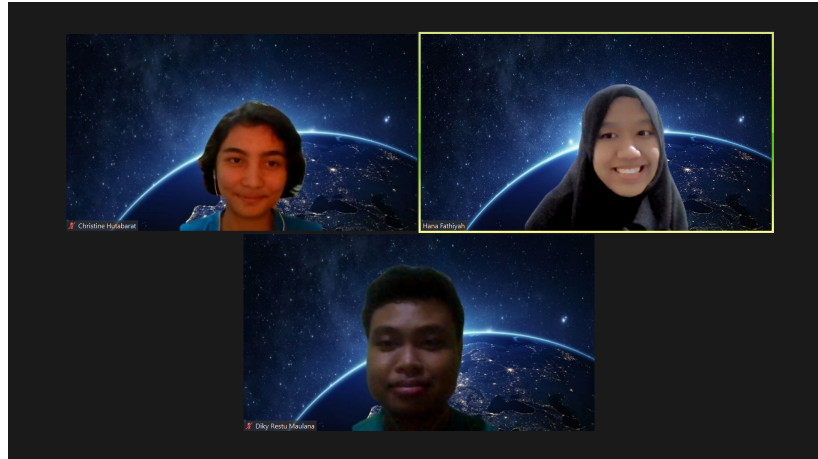


## Laporan Tugas Besar 3

IF2211 Strategi Algoritma

# Penerapan *String Matching* dan *Regular Expression* dalam *DNA Pattern Matching*



Disusun oleh:

Kelompok **HDC** tapi ada badaknya

**Christine Hutabarat      13520005**

**Diky Restu Maulana      13520017**

**Hana Fathiyah              13520047**

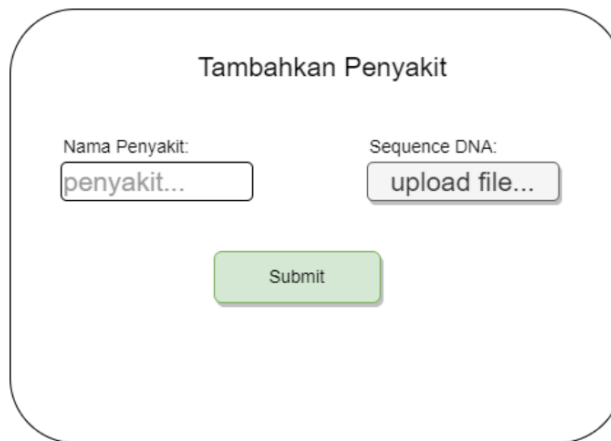
**PROGRAM STUDI TEKNIK INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
BANDUNG  
2022**

# 1. Deskripsi Tugas

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

Fitur-Fitur Aplikasi:

1. Aplikasi dapat menerima input penyakit baru berupa nama penyakit dan sequence DNA-nya (dan dimasukkan ke dalam database).
  - a. Implementasi input sequence DNA dalam bentuk file.
  - b. Dilakukan sanitasi input menggunakan regex untuk memastikan bahwa masukan merupakan sequence DNA yang valid (tidak boleh ada huruf kecil, tidak boleh ada huruf selain AGCT, dan tidak ada spasi).
  - c. Contoh input penyakit:



The image shows a web form titled "Tambahkan Penyakit". It has two input fields: "Nama Penyakit:" with a placeholder text "penyakit..." and "Sequence DNA:" with a placeholder text "upload file...". Below these fields is a green "Submit" button.

Gambar 2. Ilustrasi Input Penyakit

2. Aplikasi dapat memprediksi seseorang menderita penyakit tertentu berdasarkan sequence DNA-nya.
  - a. Tes DNA dilakukan dengan menerima input nama pengguna, sequence DNA pengguna, dan nama penyakit yang diuji. Asumsi sequence DNA pengguna > sequence DNA penyakit.
  - b. Dilakukan sanitasi input menggunakan regex untuk memastikan bahwa masukan merupakan sequence DNA yang valid (tidak boleh ada huruf kecil, tidak boleh ada huruf selain AGCT, tidak ada spasi, dll).
  - c. Pencocokan sequence DNA dilakukan dengan menggunakan algoritma string matching.
  - d. Hasil dari tes DNA berupa tanggal tes, nama pengguna, nama penyakit yang diuji, dan status hasil tes.  
Contoh: 1 April 2022 - Mhs IF - HIV - False
  - e. Semua komponen hasil tes ini dapat ditampilkan pada halaman web (refer ke poin 3 pada "Fitur-Fitur Aplikasi") dan disimpan pada sebuah tabel database.
  - f. Contoh tampilan web:

*Gambar 3. Ilustrasi Prediksi*

3. Aplikasi memiliki halaman yang menampilkan urutan hasil prediksi dengan kolom pencarian di dalamnya. Kolom pencarian bekerja sebagai filter dalam menampilkan hasil.
  - a. Kolom pencarian dapat menerima masukan dengan struktur: , 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

Index	Tanggal	Nama	Penyakit	Prediksi
1.	13 April 2022	Fulan	HIV	True
2.	13 April 2022	Kamal	HIV	False
3.	13 April 2022	Entah	HIV	False
4.	13 April 2022	Jamal	HIV	True
5.	13 April 2022	Yubai	HIV	True
6.	13 April 2022	Hika	HIV	False

*Gambar 4. Ilustrasi Interaksi 1*

- ii. Masukan hanya tanggal

13 April 2022

1. 13 April 2022 - Fulan - Diabetes - True
2. 13 April 2022 - Kamal - Sinusitis - False
3. 13 April 2022 - Entah - Down Syndrome - False
4. 13 April 2022 - Jamal - Polio - True
5. 13 April 2022 - Yubai - TBC - True
6. 13 April 2022 - Hika - Hepatitis A - False

**Gambar 5. Ilustrasi Interaksi 2**

iii. Masukan hanya nama penyakit

HIV

1. 13 April 2022 - Fulan - HIV - True
2. 14 April 2022 - Kamal - HIV - False
3. 15 April 2022 - Entah - HIV - False
4. 16 April 2022 - Jamal - HIV - True
5. 17 April 2022 - Yubai - HIV - True
6. 18 April 2022 - Hika - HIV - False

**Gambar 6. Ilustrasi Interaksi 3**

4. (Bonus) Menghitung tingkat kemiripan DNA pengguna dengan DNA penyakit pada tes DNA
  - a. Ketika melakukan tes DNA, terdapat persentase kemiripan DNA dalam hasil tes. Contoh hasil tes: 1 April 2022 - Mhs IF - HIV - 75% - False
  - b. Perhitungan tingkat kemiripan dapat dilakukan dengan menggunakan Hamming distance, Levenshtein distance, LCS, atau algoritma lainnya (dapat dijelaskan dalam laporan).

- c. Tingkat kemiripan DNA dengan nilai lebih dari atau sama dengan 80% dikategorikan sebagai True. Perlu diperhatikan mengimplementasikan atau tidak mengimplementasikan bonus ini tetap dilakukan pengecekan string matching terlebih dahulu.
- d. Contoh tampilan:

### Tes DNA

Nama Pengguna:

Sequence DNA:

Prediksi Penyakit:

---

Hasil Tes

<Tanggal> - <pengguna> - <penyakit> - <similarity> - <True/False>

#### 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: - Nama penyakit - Rantai DNA penyusun.
  - b. Hasil Prediksi: - Tanggal prediksi - Nama pasien - Penyakit prediksi - Status terprediksi.
6. Jika mengerjakan bonus tingkat kemiripan DNA, simpan hasil tingkat kemiripan tersebut pada basis data.

## 2. Landasan Teori

### a. Algoritma KMP (*Knuth-Morris-Pratt*)

Algoritma Knuth-Morris-Pratt adalah salah satu algoritma pencarian string, dikembangkan secara terpisah oleh Donald E. Knuth pada tahun 1967 dan James H. Morris bersama Vaughan R. Pratt pada tahun 1966, tetapi keduanya mempublikasikannya secara bersamaan pada tahun 1977.

Jika kita melihat algoritma brute force lebih mendalam, kita mengetahui bahwa dengan mengingat beberapa perbandingan yang dilakukan sebelumnya kita dapat meningkatkan besar pergeseran yang dilakukan. Hal ini akan menghemat perbandingan, yang selanjutnya akan meningkatkan kecepatan pencarian.

Algoritma Knuth-Morris-Pratt (KMP) mencari pola dalam teks dalam urutan kiri-ke-kanan (seperti algoritma brute force). Namun, KMP menggeser pola dengan cara yang lebih cerdas daripada algoritma brute force.

KMP melakukan praproses pola untuk menemukan kecocokan awalan pola dengan pola itu sendiri.

$j$  = posisi tidak cocok di  $P[]$

$k$  = posisi sebelum mismatch ( $k = j - 1$ ).

Fungsi Pinggiran / *Border Function*  $b(k)$  didefinisikan sebagai ukuran terbesar awalan  $P[0..k]$  yang juga merupakan akhiran dari  $P[1..k]$ .

Nama lainnya : *failure-function* (disingkat : fail)

### b. Algoritma BM (*Boyer Moore*)

Algoritma Boyer-Moore adalah salah satu algoritma pencarian string, dipublikasikan oleh Robert S. Boyer, dan J. Strother Moore pada tahun 1977.

Algoritma ini dianggap sebagai algoritma yang paling efisien pada aplikasi umum. Tidak seperti algoritma pencarian string yang ditemukan sebelumnya, algoritma Boyer-Moore mulai mencocokkan karakter dari sebelah kanan pattern. Ide di balik algoritma ini adalah bahwa dengan memulai pencocokan karakter dari kanan, dan bukan dari kiri, maka akan lebih banyak informasi yang didapat.

Algoritma ini didasarkan kepada dua teknik, yaitu

1. *The Looking-Glass Technique*

Mencari *pattern*  $P$  di dalam teks  $T$  dengan cara bergerak mundur menyusuri  $P$  dimulai dari ujung paling belakang.

2. *The Character-Jump Technique*

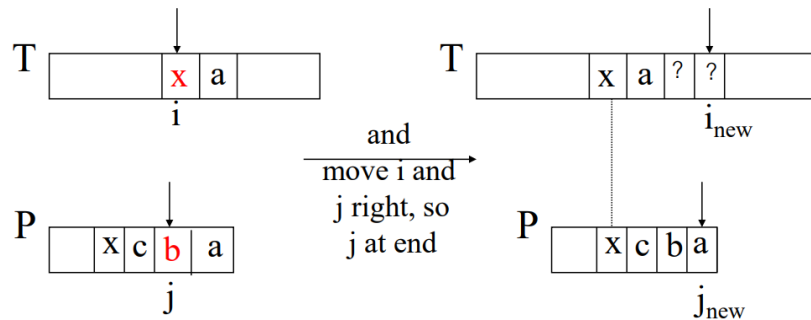
Ketika terjadi ketidakcocokan di  $T[i] \neq x$

Karakter di  $P[j]$  tidak sama dengan  $T[i]$

Terdapat tiga kasus, yaitu

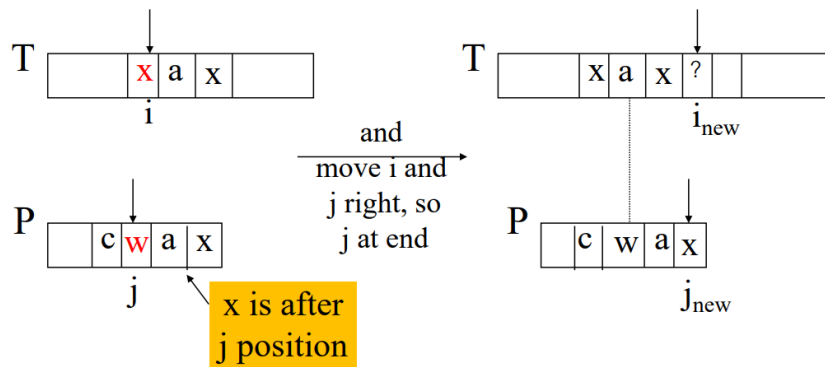
1. Kasus 1

Jika P mengandung x di suatu tempat, kemudian kemudian coba geser P ke kanan untuk menyelaraskan kemunculan terakhir x di P dengan T[i].



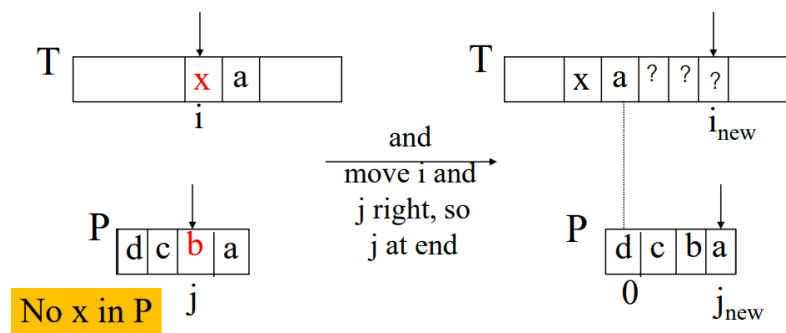
2. Kasus 2

Jika P berisi x di suatu tempat, tetapi bergeser ke kanan ke yang terakhir kemunculannya tidak mungkin, lalu geser P ke kanan sebanyak 1 karakter ke T[i+1].



3. Kasus 3

Jika kasus 1 dan 2 tidak berlaku, maka geser P untuk meratakan P[0] dengan T[i+1].



Algoritma Boyer-Moore melakukan pra-proses pola P dan alfabet A untuk membangun fungsi kemunculan terakhir / *Last Occurrence* dinotasikan sebagai L(). L() memetakan semua huruf di A ke bilangan bulat.

L(x) dengan x adalah huruf di A didefinisikan sebagai:

- indeks terbesar  $i$  sedemikian rupa sehingga  $P[i] == x$ , atau
- -1 jika indeks tidak ditemukan

c. Algoritma Regex (*Regular Expression*)

Ekspresi reguler adalah serangkaian karakter yang mendefinisikan sebuah pola pencarian. Pola tersebut biasanya digunakan oleh algoritma pencarian string untuk melakukan operasi "cari" atau "cari dan ganti" pada string, atau untuk memeriksa string masukan. Ekspresi reguler merupakan teknik yang dikembangkan dalam bidang ilmu komputer teori dan teori bahasa formal.



### 3. Analisis Pemecahan Masalah

#### a. Langkah Penyelesaian Masalah

Program menerima masukan dari pengguna berupa nama pasien, rantai DNA pasien yang didapatkan dari file masukan, dan nama penyakit yang akan diperiksa dari rantai DNA pasien. Selain itu, pengguna juga memilih metode pemeriksaan yang akan digunakan. Untuk mendapatkan pola rantai DNA dari penyakit yang dimasukkan, program akan memberikan query ke basis data yang telah terhubung. Jika penyakit ditemukan, maka pencocokan rantai DNA pasien dengan pola rantai penyakit akan dilakukan dengan menggunakan metode terpilih. Namun jika tidak, maka program tidak akan melakukan pencocokan.

Jika pencocokan dilakukan dan hasil pencocokan menghasilkan 'False', yaitu tidak ditemukan adanya pola DNA penyakit pada rantai DNA pasien, maka akan dipertimbangkan kemiripan antara kedua rantai DNA. Perhitungan dari kemiripan kedua rantai dilakukan dengan menggunakan perhitungan jarak Levenshtein. Jika kedua rantai memiliki tingkat kemiripan di atas 80%, akan diasumsikan bahwa prediksi penyakit menghasilkan hasil positif. Hasil pencocokan dan perhitungan kemiripan kemudian akan ditampilkan pada pengguna, dan riwayat pencarian akan dimasukkan ke dalam basis data.

#### b. Fitur Fungsional dan Arsitektur

Folder root

1. docker-compose.yaml pada folder root  
Berfungsi untuk melakukan instalasi postgresQL untuk membentuk database
2. heroku.yml  
Berfungsi untuk melakukan deployment backend ke dalam heroku

Folder backend

1. server.js  
Terintegrasi dengan express. Berfungsi dalam penggantian route.  
app.get berfungsi untuk membaca suatu data  
app.post berfungsi untuk menambahkan suatu data

Folder backend/controller

1. penyakit.controller.js  
Berisi aksi yang dilakukan oleh *backend* terhadap tabel penyakit pada *database*
2. prediksi.controller.js  
Berisi aksi yang dilakukan oleh *backend* terhadap tabel prediksi pada *database*.

Folder backend/lib

3. boyermooore.js  
Berisi implementasi dari algoritma *string matching* boyermooore.js
4. kmp.js  
Berisi implementasi dari algoritma *string matching* KMP
5. similarity.js  
Berisi implementasi dari perhitungan kemiripan dua *string* Levenshtein.

Folder backend/prisma

1. schema.prisma

Berisi perintah pembentukan tabel di dalam database

Folder frontend

1. vite.config.js

React dibentuk menggunakan vite dengan menggunakan create vite@latest frontend

Folder frontend/src/pages

1. Home.jsx

File yang mengatur halaman home.

2. Tambah.jsx

File yang mengatur halaman tambah, seperti integrasi dengan tabel penyakit pada database, serta menampilkan record yang telah tersimpan pada tabel penyakit

3. Test.jsx

File yang mengatur halaman test, seperti integrasi dengan tabel prediksi pada database, serta menampilkan record yang telah tersimpan pada tabel prediksi

Folder frontend/src

1. App.css

File untuk melakukan pengeditan tampilan pada halaman

2. App.jsx

File untuk melakukan integrasi antarhalaman

3. main.jsx

Memanggil App.jsx untuk menjalankan tampilan frontend

## 4. Implementasi dan Pengujian

### a. Spesifikasi Teknis Program

1. Frontend
  - React dengan library MUI
2. Backend
  - Nodejs
3. Database
  - PostgreSQL menggunakan prisma
4. Deploy
  - Frontend: netlify
  - Backend: heroku

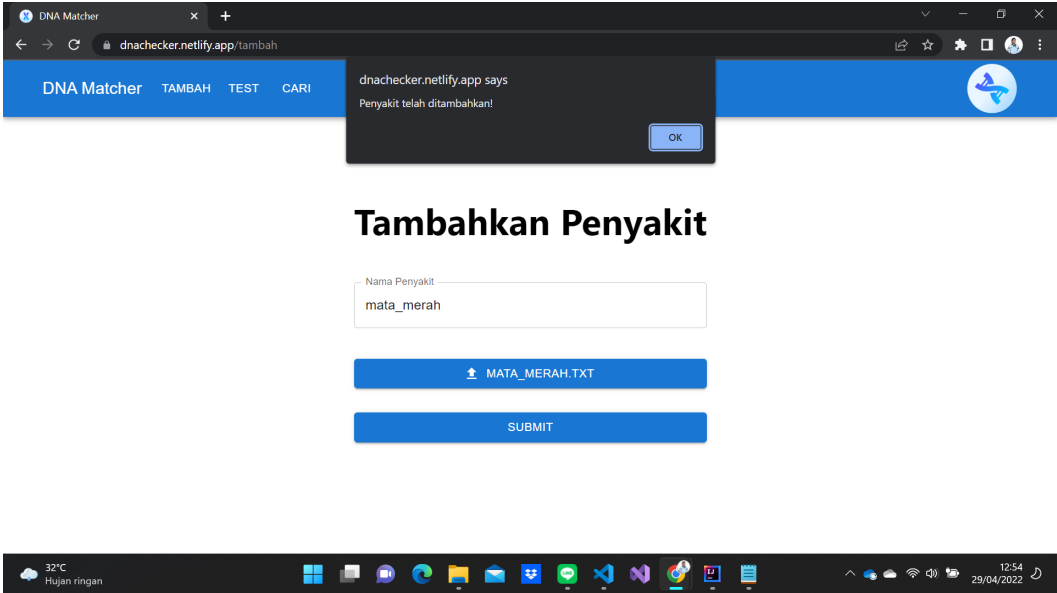
### b. Cara Penggunaan Program

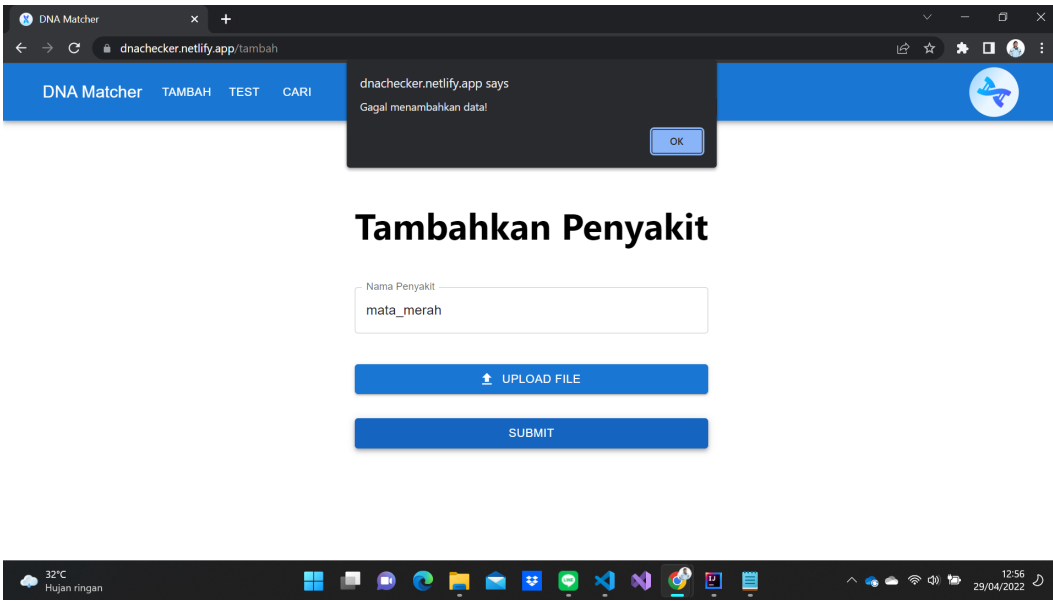
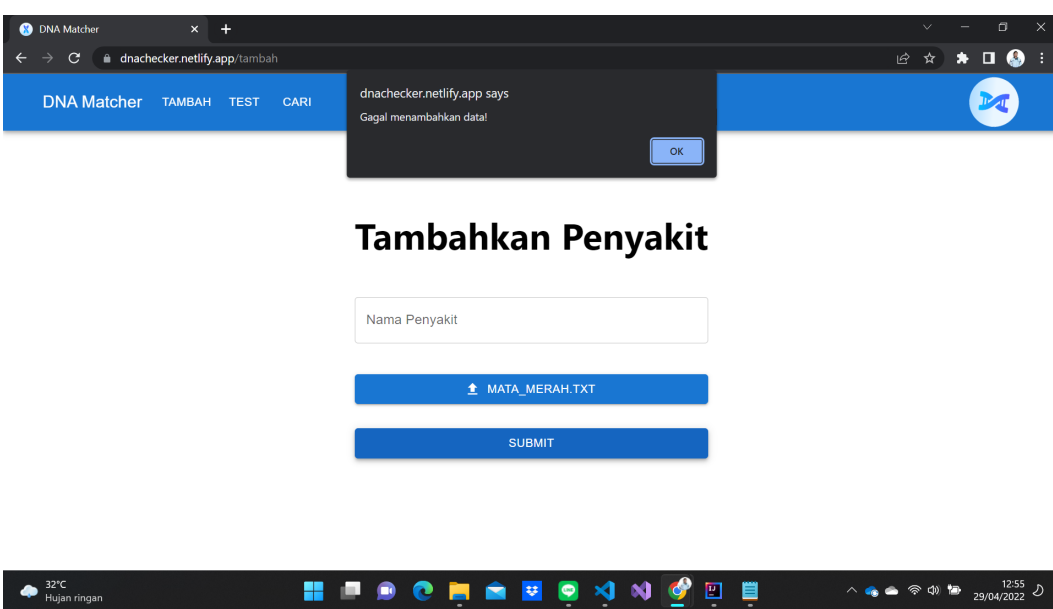
1. Pengujian di lokal
  - Frontend
    - Ubah direktori ke ./src/frontend
    - Jalankan: "npm run dev"
  - Backend

- Ubah direktori ke ./src/backend  
Jalankan: “npm start”
  - Database  
Ubah direktori ke ./src/backend  
Jalankan: “npx prisma studio”
  - Docker  
Ubah direktori ke ./src/backend  
Jalankan: “docker-compose up”
2. Pengujian di situs hasil deployment
- Frontend  
Buka situs:  
<https://dnachecker.netlify.app/>
  - Backend  
Buka situs  
<https://dnachecker.herokuapp.com>  
<https://dnachecker.herokuapp.com/penyakit>  
<https://dnachecker.herokuapp.com/prediksi>

### c. Hasil Pengujian

#### 1. Menambahkan Penyakit Baru

Berhasil	
----------	--

<p>Gagal (tidak ada input file)</p>	
<p>Gagal (nama penyakit kosong)</p>	

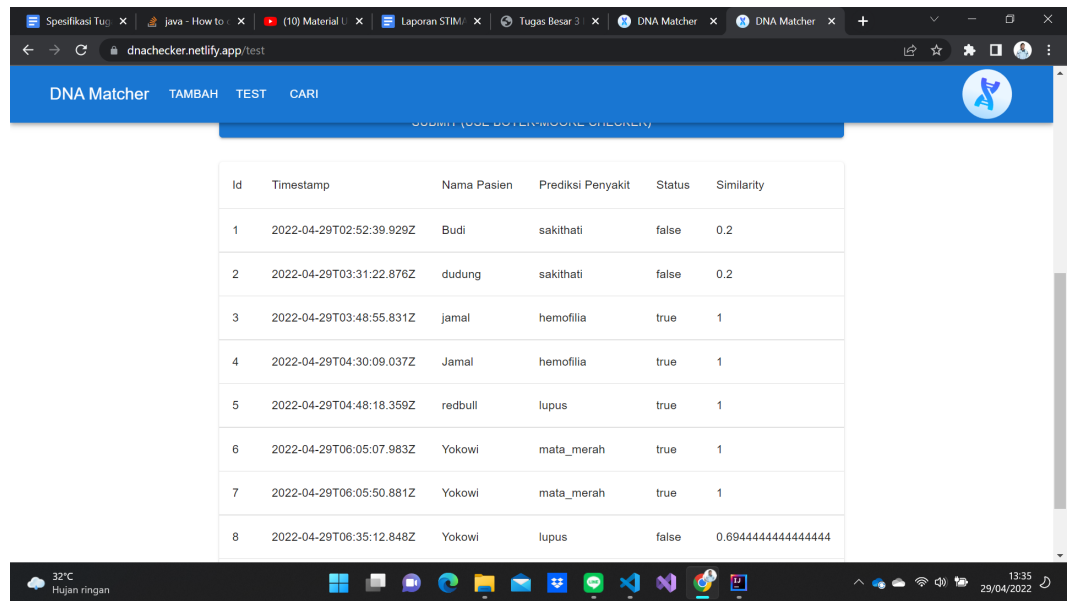
## 2. Memprediksi Penyakit

Berhasil  
(menderita  
penyakit)

The screenshot displays the DNA Matcher web application interface. The top navigation bar includes 'DNA Matcher', 'TAMBAH', 'TEST', and 'CARI'. A notification box states 'dnachecker.netlify.app says: Prediksi telah ditambahkan!' with an 'OK' button. The input section contains a 'Nama Pengguna' field with the value 'Yokowi', a file upload button 'YOKOWI.TXT', a 'Prediksi Penyakit' dropdown menu showing '8 mata\_merah GATACTT', and two submit buttons: 'SUBMIT (USE KMP CHECKER)' and 'SUBMIT (USE BOYER-MOORE CHECKER)'. Below the input section is a table with the following data:

Id	Timestamp	Nama Pasien	Prediksi Penyakit	Status	Similarity
1	2022-04-29T02:52:39.929Z	Budi	sakithati	false	0.2
2	2022-04-29T03:31:22.876Z	dudung	sakithati	false	0.2
3	2022-04-29T03:48:55.831Z	jamal	hemofilia	true	1
4	2022-04-29T04:30:09.037Z	Jamal	hemofilia	true	1
5	2022-04-29T04:48:18.359Z	redbull	lupus	true	1
6	2022-04-29T06:05:07.983Z	Yokowi	mata_merah	true	1

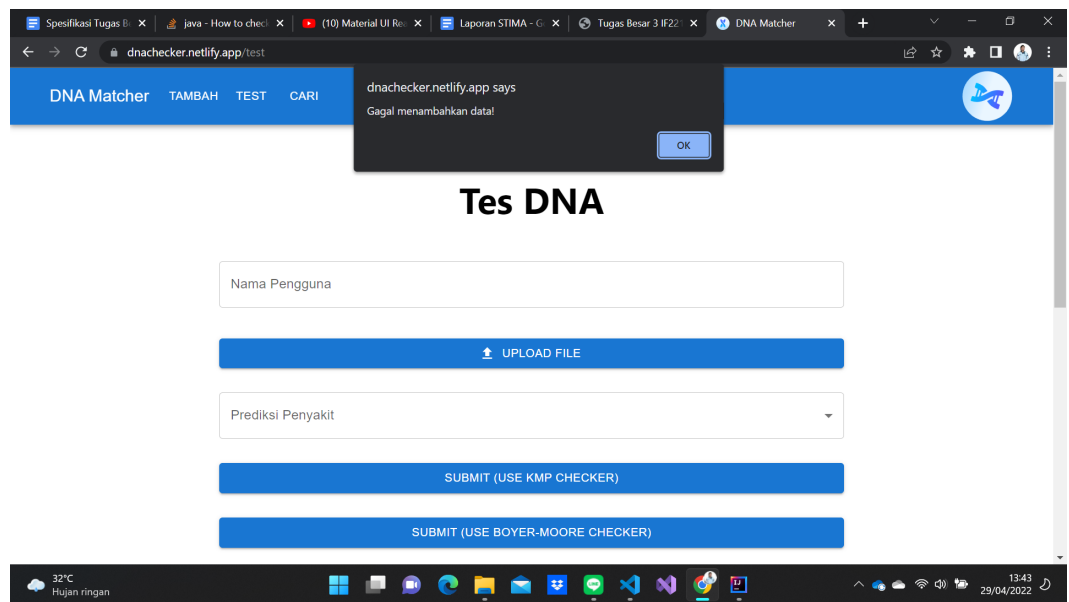
Berhasil  
(tidak  
menderita)



The screenshot shows the 'DNA Matcher' application interface. At the top, there's a navigation bar with 'DNA Matcher', 'TAMBAH', 'TEST', and 'CARI' buttons. Below this is a table with 8 rows of patient data. The table has columns for Id, Timestamp, Nama Pasien, Prediksi Penyakit, Status, and Similarity. The data shows various patients with different predicted diseases and similarity scores.

Id	Timestamp	Nama Pasien	Prediksi Penyakit	Status	Similarity
1	2022-04-29T02:52:39.929Z	Budi	sakithati	false	0.2
2	2022-04-29T03:31:22.876Z	dudung	sakithati	false	0.2
3	2022-04-29T03:48:55.831Z	jamal	hemofilia	true	1
4	2022-04-29T04:30:09.037Z	Jamal	hemofilia	true	1
5	2022-04-29T04:48:18.359Z	redbull	lupus	true	1
6	2022-04-29T06:05:07.983Z	Yokowi	mata_merah	true	1
7	2022-04-29T06:05:50.881Z	Yokowi	mata_merah	true	1
8	2022-04-29T06:35:12.848Z	Yokowi	lupus	false	0.6944444444444444

Gagal (ada  
field yang  
kosong)



The screenshot shows the 'DNA Matcher' application interface with an error message overlay. The error message says 'dnachecker.netlify.app says: Gagal menambahkan data!' with an 'OK' button. Below the error message, the 'Tes DNA' form is visible. The form has a 'Nama Pengguna' input field, an 'UPLOAD FILE' button, a 'Prediksi Penyakit' dropdown menu, and two 'SUBMIT' buttons: 'SUBMIT (USE KMP CHECKER)' and 'SUBMIT (USE BOYER-MOORE CHECKER)'.

**Tes DNA**

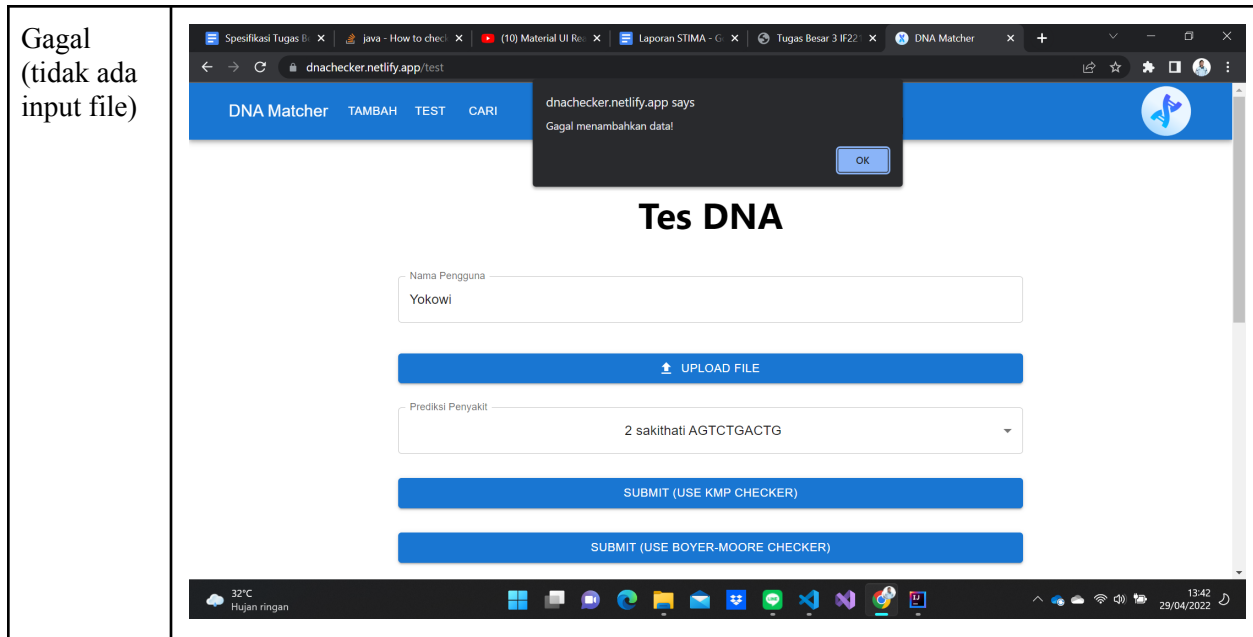
Nama Pengguna

UPLOAD FILE

Prediksi Penyakit

SUBMIT (USE KMP CHECKER)

SUBMIT (USE BOYER-MOORE CHECKER)



Gagal  
(tidak ada  
input file)

### 3. Mencari Hasil Prediksi

#### d. Analisis Hasil Pengujian

Pengujian fitur menambahkan penyakit baru dilakukan dalam tiga kasus. Pertama, jika seluruh masukan lengkap dan valid, akan keluar pesan bahwa penyakit berhasil ditambahkan ke dalam database. Kedua, jika nama penyakit tidak diisi, akan keluar pesan bahwa penyakit gagal ditambahkan. Begitu pula jika file inputnya kosong, proses akan digagalkan.

Pengujian fitur memprediksi penyakit dilakukan dalam empat kasus. Pertama, jika DNA pasien yang dimasukkan cocok dengan penyakit yang diprediksi, hasil prediksi akan mengembalikan “true” dan akan disimpan ke dalam database. Kedua, jika DNA pasien tidak cocok dengan penyakit yang diprediksi, maka hasil prediksinya akan tetap dimasukkan ke dalam database, tetapi dengan hasil “false”. Kedua kasus ini juga akan memberikan hasil berupa bilangan antara 0 hingga 1 yang merupakan tingkat kemiripan DNA pasien dengan rantai DNA penyakit. Kasus lainnya adalah ketika nama pasien atau prediksi penyakit belum diisi, maka proses akan digagalkan dan menampilkan pesan error. Begitu pula jika file inputnya kosong, proses akan digagalkan.

Kami melakukan pengujian dengan menambahkan sebuah penyakit bernama “mata\_merah” dengan rantai DNA sebagai berikut

GATACTT

Kemudian, dilakukan prediksi terhadap pasien bernama Yokowi dengan rantai DNA sebagai berikut.

AGGCTGCGACGATGGGCCATTATAG**GATACTT**TATCCC

Terlihat jelas bahwa Yokowi menderita penyakit mata\_merah dan itu terbukti dengan hasil evaluasi yang mengembalikan true dengan tingkat kemiripan sebesar satu.

## 5. Kesimpulan dan Saran

Tugas Besar 3 IF2211 Strategi Algoritma Tahun 2021/2022 berjudul “Penerapan *String Matching* dan *Regular Expression* dalam DNA *Pattern Matching*” telah berhasil diselesaikan dengan hasil yang memuaskan. Kami telah membuat sebuah aplikasi berbasis web untuk melakukan pencocokan rantai DNA terhadap sebuah potongan rantai DNA penyakit dan melakukan evaluasi. Aplikasi ini menyimpan seluruh potongan rantai DNA penyakit dalam sebuah database. Rantai DNA pasien yang telah dicek beserta hasil evaluasinya juga akan disimpan ke dalam database tersebut. Pengguna bisa melakukan pencarian riwayat pengecekan berdasarkan tanggal dan nama penyakit. Tampilan antarmuka pengguna dibuat menarik dan berwarna supaya tidak membosankan dalam penggunaannya.

Tim pengembang menyadari masih banyak kekurangan di dalam aplikasi ini. Oleh karena itu, pengembangan lebih lanjut sangat diharapkan untuk memperluas manfaat. Ide pembuatan aplikasi ini juga bisa menjadi inspirasi bagi para pengembang aplikasi di luar sana yang tertarik di bidang kesehatan.



## 6. Tautan Github

[https://github.com/hanafathiyah/Tubes3\\_13520005/](https://github.com/hanafathiyah/Tubes3_13520005/)

## 7. Tautan Youtube

<https://www.youtube.com/watch?v=9ozzSwnEuTI>

## 8. Tautan Hasil Deploy (Frontend)

<https://dnachecker.netlify.app/>

## 9. Tautan Hasil Deploy (Backend)

<https://dnachecker.herokuapp.com/>

## 10. Daftar Pustaka

React : <https://id.reactjs.org/docs/getting-started.html>

React Router : <https://reactrouter.com/docs/en/v6/getting-started/overview>

HTML : <https://www.w3schools.com/html/>

Material UI : <https://mui.com/material-ui/getting-started/installation/>