

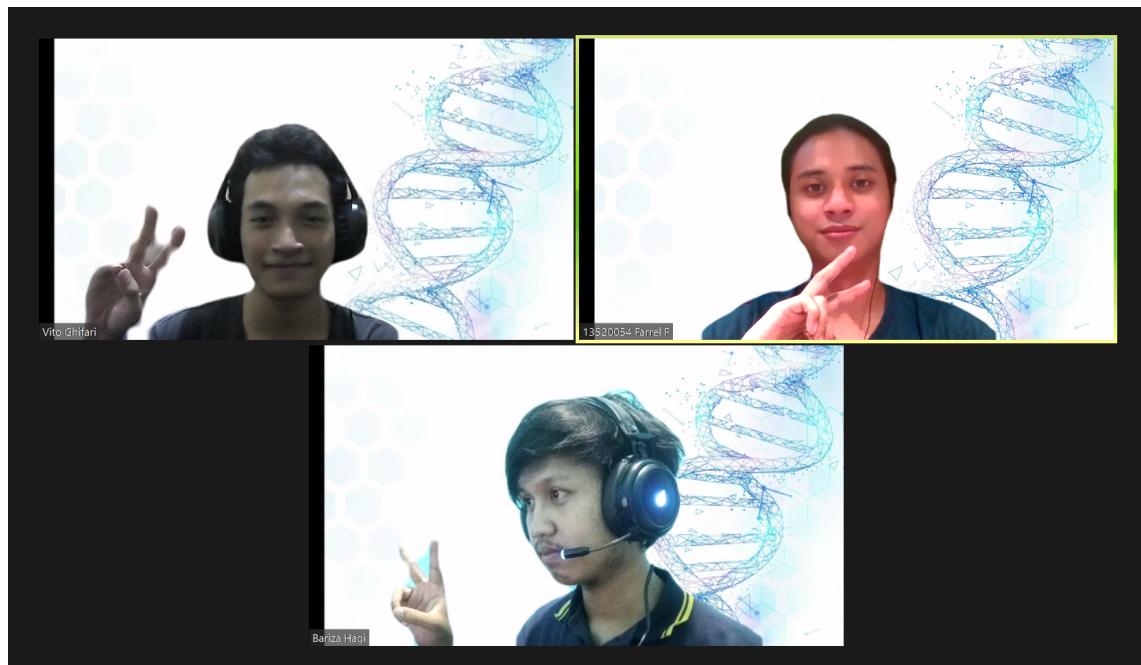
# **TUGAS BESAR I IF2211 STRATEGI ALGORITMA**

**SEMESTER II TAHUN 2021/2022**

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

Disusun Oleh Kelompok:

Kelompok 30 “DNA at Work!”



Bariza Haqi  
13520018

Farrel Farandieka  
Fibriyanto  
13520054

Vito Ghifari  
13520153

**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

## Daftar Isi

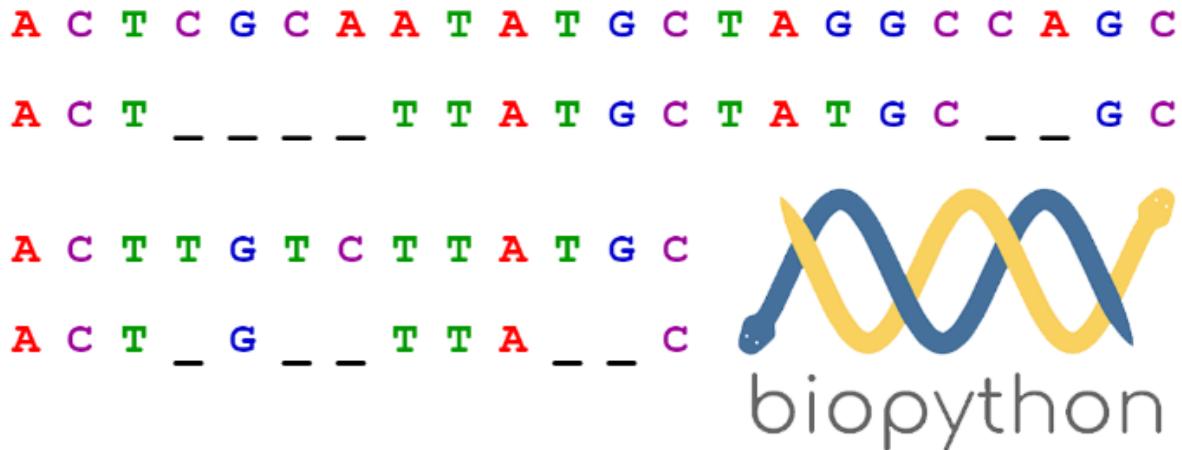
<b>Daftar Isi</b>	<b>2</b>
<b>Bab 1</b>	
<b>Deskripsi Tugas</b>	<b>4</b>
<b>Bab 2</b>	<b>6</b>
2.1 Algoritma Knuth-Morris-Pratt	6
2.2 Algoritma Boyer-Moore	7
2.3 Algoritma Regular Expression	9
2.4 Aplikasi Berbasis Web	9
<b>Bab 3</b>	<b>10</b>
3.1 Langkah Penyelesaian Masalah	10
3.1.1 Input Sekuens DNA ke Aplikasi	10
3.1.2 Pengecekan Sekuens DNA Pasien	10
3.1.3 Pencarian Hasil dari Pengecekan Sekuens DNA	11
3.2 Fitur Fungsional dan Arsitektur Aplikasi	12
3.2.1 Fitur Fungsional	12
3.2.2 Arsitektur Aplikasi	12
<b>Bab 4</b>	<b>13</b>
4.1 Spesifikasi Teknis Program	13
4.1.1 Struktur Data	13
4.1.2 Fungsi dan Prosedur	14
4.2 Tata Cara Penggunaan Program	16
4.2.1 Antarmuka Program	16
4.2.2 Fitur yang Disediakan Program	18
4.3 Hasil Pengujian	19

4.3.1 Hasil Pengujian Penambahan DNA Penyakit	19
4.3.2 Hasil Pengujian Prediksi Penyakit Pasien	21
4.3.3 Hasil Pengujian Cari Prediksi DNA	23
4.4 Analisis Hasil Pengujian	25
<b>Bab 5</b>	<b>26</b>
5.1 Kesimpulan	26
5.2 Saran	26
<b>Daftar Pustaka</b>	<b>27</b>
<b>Link Repositori dan Video Penjelasan</b>	<b>27</b>

## Bab 1

### Deskripsi Tugas

Manusia umumnya memiliki 46 kromosom di dalam setiap selnya. Kromosom-kromosom tersebut tersusun dari DNA (deoxyribonucleic acid) atau asam deoksiribonukleat. DNA tersusun atas dua zat basa purin, yaitu Adenin (A) dan Guanin (G), serta dua zat basa pirimidin, yaitu sitosin (C) dan timin (T). Masing-masing purin akan berikatan dengan satu pirimidin. DNA merupakan materi genetik yang menentukan sifat dan karakteristik seseorang, seperti warna kulit, mata, rambut, dan bentuk wajah. Ketika seseorang memiliki kelainan genetik atau DNA, misalnya karena penyakit keturunan atau karena faktor lainnya, ia bisa mengalami penyakit tertentu. Oleh karena itu, tes DNA penting untuk dilakukan untuk mengetahui struktur genetik di dalam tubuh seseorang serta mendeteksi kelainan genetik. Ada berbagai jenis tes DNA yang dapat dilakukan, seperti uji pra implantasi, uji pra kelahiran, uji pembawa atau carrier testing, uji forensik, dan DNA sequence analysis.



Gambar 1.1 Ilustrasi Sekuens DNA

Sumber: <https://towardsdatascience.com/pairwise-sequence-alignment-using-biopython-d1a9d0ba861f>

Salah satu jenis tes DNA yang sangat berkaitan dengan dunia bioinformatika adalah DNA sequence analysis. DNA sequence analysis adalah sebuah cara yang dapat digunakan untuk memprediksi berbagai macam penyakit yang tersimpan pada database berdasarkan urutan sekuens DNA-nya. Sebuah sekuens DNA adalah suatu representasi string of nucleotides yang disimpan pada suatu rantai DNA, sebagai contoh: ATTCGTAACTAGTAAGTTA. Teknik pattern matching memegang peranan penting untuk dapat menganalisis sekuens DNA yang sangat panjang dalam waktu

singkat. Oleh karena itu, kami berniat untuk membuat suatu aplikasi web berupa DNA Sequence Matching yang menerapkan algoritma String Matching dan Regular Expression untuk membantu penyedia jasa kesehatan dalam memprediksi penyakit pasien. Hasil prediksi juga dapat ditampilkan dalam tabel dan dilengkapi dengan kolom pencarian untuk membantu pengguna dalam melakukan filtering dan pencarian.

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

## Bab 2

### Landasan Teori

#### 2.1 Algoritma Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt atau algoritma KMP merupakan salah satu algoritma pencocokan pola (*pattern*) pada suatu teks. Pada dasarnya, algoritma KMP melakukan pencocokan dari kiri ke kanan teks seperti algoritma *brute force*. Namun, algoritma KMP melakukan *shifting* yang bergantung pada pola yang ingin dicari pada teks. Kompleksitas algoritma KMP pun berbeda dengan *brute force*. Pada *brute force*, kompleksitas algoritmanya  $O(mn)$ , sedangkan KMP kompleksitasnya adalah  $O(m+n)$ .

Pada algoritma KMP, dikenal dua istilah untuk membantu pencarian *string* pada teks, yaitu *prefix* dan *suffix*. Contoh *prefix* dan *suffix* dapat di lihat di bawah ini.

Pola: abcdeabc

Prefiks: a, abc, abcd, abcde, abcdea, abcdeab

Sufiks: c, bc, abc, eabc, deabc, cdeabc, bcdeabc

Algoritma KMP mencari prefiks terpanjang yang juga merupakan sufiks. Pada contoh tersebut untuk prefiks *abc*, sufiks terpanjang yang memenuhi adalah *abc*. Dalam hal ini, panjang sufiks terpanjang yang memenuhi untuk prefiks *abc* adalah 3. Pencarian sufiks terpanjang dilakukan untuk semua prefiks yang terdapat pada pola. Ini disebut dengan *preprocessing* algoritma KMP, yaitu mendapatkan *KMP border function* atau fungsi pinggiran. Gambar 2.1.1 menjelaskan *border function* yang terbentuk untuk pola *abaaba*. Lambang *j* merupakan indeks dari huruf pada pola dan  $P[j]$  adalah huruf pada indeks ke *j*. String *P* akan membentuk *border function*  $b(k)$ .

<i>j</i>	0	1	2	3	4	5
$P[j]$	a	b	a	a	b	a

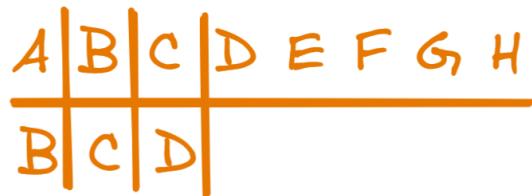
<i>k</i>	0	1	2	3	4
$b(k)$	0	0	1	1	2

Gambar 2.1.1 *Border function* untuk pola *abaaba*

Cara penggunaannya untuk mencari pola pada teks adalah dengan membandingkan huruf pertama pola dengan teks. Jika tidak ditemukan kecocokan pada huruf pertama, dilanjutkan dengan huruf kedua dan seterusnya. Apabila ditemukan kecocokan pada huruf pertama pola, lanjutkan dengan membandingkan huruf kedua pola dan seterusnya. Jika terjadi ketidakcocokan antara pola dan teks, catat indeks ketidakcocokan tersebut (indeks  $j$  pada pola), lalu petakan ke fungsi pinggiran  $b(k)$  dengan  $k = j-1$ . Pencarian dilanjutkan dengan memulai perbandingan pada indeks  $j$ .

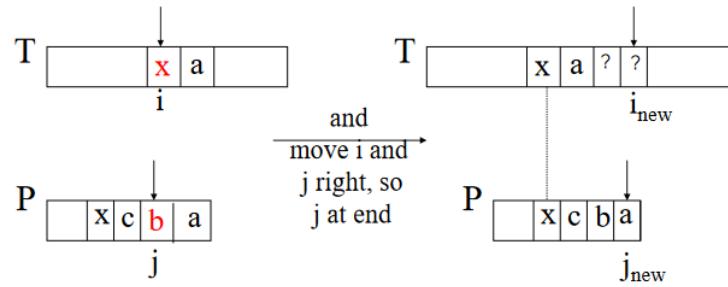
## 2.2 Algoritma Boyer-Moore

Algoritma Boyer-Moore merupakan algoritma pencocokan pola berbasis dua teknik, yaitu teknik *Looking-Glass* dan teknik *Character-Jump*. Teknik *Looking-Glass* merupakan teknik dengan mengecek kecocokan karakter pada kalimat yang sedang dicek dengan karakter yang ingin dicari dari belakang kalimat, bukan dari awal.



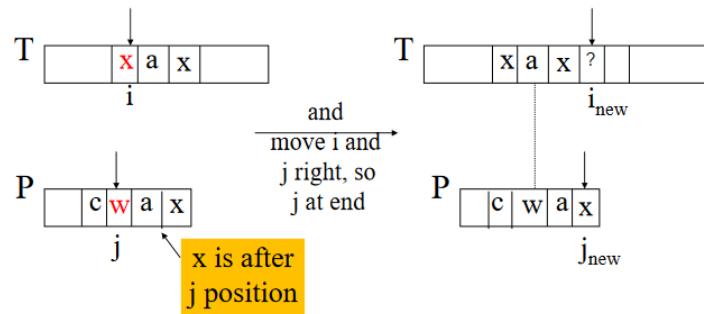
Gambar 2.2.1 contoh *Looking-Glass*

Contohnya, bila kita lihat pada gambar di atas, kita akan memulai pengecekan dari paling belakang kalimat BCD. Maka akan dicek apakah karakter D sama dengan karakter C. Teknik *Character-Jump* merupakan teknik untuk memindahkan indeks karakter yang sedang dibaca dari kalimat utama sebanyak sejumlah karakter tertentu. Terdapat tiga kasus *Character-Jump* dalam algoritma Boyer-Moore, Kasus Pertama merupakan jika pola  $P$  terdiri dari  $x$  pada suatu posisi, maka coba untuk menggerakkan  $P$  ke kanan untuk mensejajarkan dengan kemunculan  $x$  terakhir di  $P$  dengan  $T[i]$ .



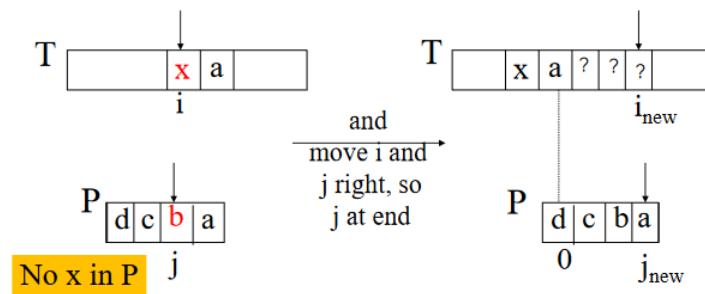
Gambar 2.2.2 Ilustrasi Kasus Pertama

Kasus kedua, jika pola P terdiri dari x pada suatu posisi, tetapi tidak memungkinkan untuk melakukan pergeseran ke kanan ke kemunculan x terakhir, maka geser P ke kanan 1 karakter ke  $T[i+1]$ .



Gambar 2.2.3 Ilustrasi Kasus Kedua

Kasus Ketiga, apabila kasus pertama dan kedua tidak dapat diterapkan, maka geser P untuk mensejajarkan  $P[0]$  dengan  $T[i+1]$ .



Gambar 2.2.4 Ilustrasi Kasus Ketiga

Pada algoritma yang diimplementasi, akan dibuat fungsi Last Occurrence untuk mencari indeks terakhir kemunculan sebuah karakter. Fungsi tersebut kemudian

akan dipakai untuk membuat tabel Bad Match yang akan dipakai untuk menentukan berapa banyak karakter yang perlu dilompati.

### 2.3 Algoritma Regular Expression

Regular Expression, atau singkatnya regex atau regexp, adalah teknik yang sangat efisien dalam mencari dan memanipulasi teks, terutama dalam memproses file teks. Satu baris regex dapat dengan mudah menggantikan beberapa lusin baris kode pemrograman. Teknik regex biasa ditemukan dalam algoritma *find* atau *find and replace* serta sebagai algoritma validasi input. Regex dikembangkan dalam ilmu komputer dan teori bahasa formal.

Regex dapat didefinisikan sebagai suatu pola syntax standar dan spesifik yang mewakili sebuah pola text yang “diterima”. Setiap karakter dalam regex merupakan sebuah karakter yang bermakna literal atau sebuah metakarakter yang memiliki makna khusus menurut syntax penulisan regex yang digunakan. Sebagai contoh, regex `b.` terdiri dari karakter `b` yang memiliki makna literal huruf `b` dan karakter `.` yang memiliki makna metakarakter yang mencocokkan semua karakter kecuali newline, sehingga `b.` menerima `b%`, `bx`, dan `b5` tetapi tidak menerima `b\n`.

### 2.4 Aplikasi Berbasis Web

Aplikasi Berbasis Web merupakan program aplikasi yang disimpan pada suatu server jarak jauh dan dikirimkan melalui jaringan internet dan ditampilkan melalui antarmuka suatu browser. Aplikasi Web dirancang untuk berbagai kegunaan dan dapat digunakan oleh siapa saja. Aplikasi Web yang umum merupakan webmail, kalkulator daring, toko e-niaga, dst.

Karena Aplikasi Web tidak perlu diunduh dan dijalankan secara lokal, suatu aplikasi web membutuhkan server web, server aplikasi, dan server database agar dapat beroperasi. Server web mengelola permintaan yang datang dari klien, server aplikasi bertugas menyelesaikan tugas yang diminta, dan database dipakai untuk menyimpan informasi yang dibutuhkan.

## Bab 3

### Analisis Pemecahan Masalah

#### 3.1 Langkah Penyelesaian Masalah

Kami mengidentifikasi tiga sub-masalah yang bisa diidentifikasi dari Bab 1, menginput sekuens DNA suatu penyakit ke aplikasi, melakukan pengecekan sekuens DNA pasien terhadap sekuens DNA penyakit yang terdapat pada aplikasi, dan melakukan pencarian hasil dari pengecekan sekuens DNA yang sudah pernah dilakukan.

##### 3.1.1 Input Sekuens DNA ke Aplikasi

Secara garis besar, langkah pemecahan sub-masalah ini akan berlaku sebagai berikut:

1. Menentukan masukan dari pengguna ke aplikasi
2. Memastikan masukan pengguna merupakan masukan yang valid dan unik
3. Memasukkan masukan pengguna ke dalam basis data aplikasi

Kami menentukan masukan paling minimum dari pengguna merupakan nama penyakit dan sekuens DNA penyakit tersebut. Aplikasi kemudian akan mengecek apakah sekuens DNA penyakit yang dimasukkan merupakan valid, yaitu hanya terdiri dari karakter “A”, “C”, “G”, dan “T”. Untuk implementasi validasi karakter, akan dijalankan suatu ekspresi Regular Expression terhadap sekuens DNA yang dimasukkan. Apabila tidak terjadi masalah terhadap ekspresi tersebut, maka kita bisa asumsikan masukan valid. Sekuens DNA kemudian dapat dimasukkan ke dalam basis data aplikasi, key yang akan disimpan merupakan nama penyakit serta sekuens DNA penyakit tersebut.

##### 3.1.2 Pengecekan Sekuens DNA Pasien

Garis besar pemecahan sub-masalah pengecekan sekuens DNA pasien akan berlaku sebagai berikut:

1. Menentukan masukan dari pengguna ke aplikasi
2. Mengecek apakah sekuens DNA pengguna merupakan sekuens yang valid
3. Mengecek menggunakan suatu algoritma String Matching apakah terdapat sekuens DNA penyakit dalam sekuens DNA dari masukan

4. Mengeluarkan hasil pencarian sekuens DNA kepada pengguna dan memasukkan hasil pencarian ke dalam basis data aplikasi

Kami menentukan masukan paling minimum merupakan nama pasien, algoritma String Matching, sekuens DNA pasien, serta nama penyakit yang ingin dicek. Sekuens DNA Pasien kemudian perlu dicek terlebih dahulu apakah merupakan sekuens DNA yang valid atau tidak. Apabila valid, maka akan dilakukan String Matching menggunakan algoritma yang telah ditentukan pengguna. Apabila terdapat substring yang sama dengan sekuens DNA penyakit, nyatakan bahwa pasien terkena penyakit tersebut dengan hasil kesamaan substring 100%. Apabila tidak, lakukan pengecekan berikutnya yaitu menggunakan partial string matching. Apabila hasil partial string matching lebih besar sama dengan 80%, maka bisa ditetapkan bahwa pasien sebenarnya terkena penyakit tersebut. Setelah pengecekan string matching partial maupun full selesai, akan diberikan hasil pengecekan kepada pengguna. Hasil pengecekan pun pada akhirnya akan disimpan pada basis data dengan key waktu dilakukan pengecekan, nama pasien, nama penyakit, tanggal dilakukan pengecekan, hasil kesamaan, dan apakah pasien terkena penyakit tersebut atau tidak.

### 3.1.3 Pencarian Hasil dari Pengecekan Sekuens DNA

Garis besar pemecahan sub-masalah pencarian hasil pengecekan sekuens DNA adalah sebagai berikut:

1. Menentukan masukan dari pengguna ke aplikasi
2. Memisahkan nilai Tanggal serta nilai Penyakit yang ingin dicari dari masukan pengguna
3. Mencari entri basis data yang cocok dengan nilai Tanggal dan Penyakit

Kami menentukan masukan dari pengguna hanya berupa satu string panjang berisikan tanggal dan nama penyakit, kita panggil input string. Pemisahan kedua *value* tersebut akan menggunakan ekspresi Regex. Pertama Regex dipakai untuk mendeteksi dan mengambil *value* Tanggal yang kemudian *value* tersebut akan dihilangkan dari string input. Kemudian akan digunakan pula ekspresi Regex untuk mendeteksi dan mengambil *value* Penyakit dari string input yang sudah dihilangkan *value* tanggal. Kedua *value* tersebut kemudian akan dipakai dalam pencarian *query* di basis data. Setelah hasil *query* dikembalikan basis data, hasil *query* akan dikirimkan kembali ke pengguna untuk ditampilkan.

## 3.2 Fitur Fungsional dan Arsitektur Aplikasi

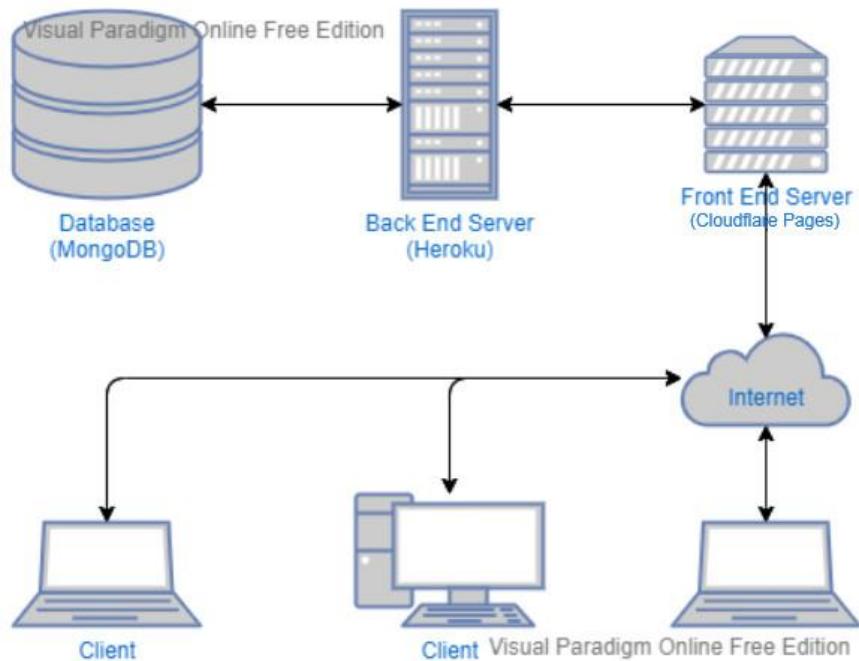
### 3.2.1 Fitur Fungsional

Secara fungsional, berikut merupakan daftar fitur yang bisa dilakukan aplikasi web kami

1. Menambahkan sekuen DNA suatu penyakit
2. Mengecek apakah terdapat sekuen DNA suatu penyakit pada sekuen DNA suatu pasien
3. Mencari hasil pengecekan berdasarkan *query* tanggal atau nama penyakit

### 3.2.2 Arsitektur Aplikasi

Arsitektur aplikasi kami adalah menggunakan MongoDB sebagai basis data, aplikasi back-end dengan bahasa Go yang di-deploy pada platform Heroku, dan aplikasi front-end berupa VueJS yang di-deploy pada platform Cloudflare Pages. Gambar 3.2.2.1 menjelaskan diagram dari arsitektur aplikasi berbasis web ini.



Gambar 3.2.2.1 Arsitektur aplikasi berbasis web “Dna at Work!”

## Bab 4

### Implementasi dan Pengujian

#### 4.1 Spesifikasi Teknis Program

##### 4.1.1 Struktur Data

Pada implementasi dari aplikasi berbasis web untuk pengujian DNA, terdapat beberapa struktur data yang terlibat. Struktur data tersebut adalah

###### 1. Struct

Pada bahasa Go, dikenal sebuah struktur data bernama struct. Struktur data ini digunakan untuk melakukan pengelompokan beberapa jenis data menjadi sebuah grup. Kami menggunakan struct sebagai skema basis data agar *document* yang diperoleh maupun yang dimasukkan ke basis data tetap konsisten.

Kami menggunakan struct pada dua skema, yaitu Disease dan Result. Struct Disease mempunyai tiga data di dalamnya yaitu ID, Name, dan DNA. Pada struct Result, terdapat tujuh data yaitu ID, Date, PatientName, DiseaseName, hasDisease, dan Likeness.

```
type Disease struct {
    ID   primitive.ObjectID `bson:"_id,omitempty"`
    Name string            `bson:"name,omitempty"`
    DNA  string            `bson:"dna,omitempty"`
}

type Result struct {
    ID        primitive.ObjectID `bson:"_id,omitempty"`
    Date     primitive.DateTime `bson:"date,omitempty"`
    PatientName string          `bson:"patientName,omitempty"`
    DiseaseName string          `bson:"diseaseName,omitempty"`
    HasDisease bool             `bson:"hasDisease"`
    Likeness  float32          `bson:"likeness"`
}
```

Gambar 4.1.1.1 Struct Disease dan Result

###### 2. Map

Struktur data lain pada bahasa pemrograman Go adalah map. Tipe

struktur data ini berbentuk *key-value pair*. Setiap data/value yang disimpan, akan disiapkan juga *key*-nya. Sementara itu, setiap *key* pada map harus unik karena sebagai penanda untuk pengaksesan *value* yang bersangkutan. Penggunaan map pada kasus ini adalah untuk mencari *status message* berdasarkan *status code*.

```
var text = map[string]string{
    InvalidDNA:      "DNA sequence contains illegal characters",
    InvalidAlgorithm: "The specified algorithm invalid. Algorithms available: KMP, Boy
    DiseaseExists:   "Disease name already exists",
    DiseaseNotExists: "DNA information from the disease requested is not yet available
    PatientNameEmpty: "Patient name empty",
    DiseaseEmpty:    "Disease name empty",
    FileError:        "An error occurred while reading the file",
    ServerError:      "The server encountered an error",
    WrongSearchQuery: "Wrong search query",
}
```

Gambar 4.1.1.2 Struktur data map pada Back-end

### 3. Array

Struktur data yang lebih umum adalah array. Tipe struktur data array pada Go berisi kumpulan data yang berjenis sama dan kapasitasnya ditentukan saat inisialisasi. Array pada program yang telah kami buat diimplementasikan untuk menampung hasil prediksi yang sebelumnya telah disimpan pada basis data.

## 4.1.2 Fungsi dan Prosedur

Struktur program yang kami buat terbagi atas dua bagian, yaitu front-end dan back-end. Karena itu, fungsi dan prosedur yang dijelaskan akan dipisah menjadi dua bagian.

### 1. Back-end

- AddDiseaseController: Fungsi ini digunakan untuk melakukan pemrosesan pada penambahan DNA penyakit dari POST request. Data yang valid akan ditambahkan ke database.
- PredictPatientController: Digunakan untuk melakukan pemrosesan pada request yang berisi prediksi penyakit yang diderita pasien, nama pasien, serta DNA pasien. Data yang valid akan ditambahkan ke basis data.
- GetResultController: Digunakan untuk melakukan pencocokan pola dengan teks dari DNA penyakit dengan DNA pasien.

- Connect: Melakukan koneksi pada basis data MongoDB.
- BoyerMoore: Menerima pola dan teks yang akan dicocokkan dengan algoritma Boyer-Moore. Fungsi ini nilai boolean jika teks tersebut mengandung pola atau tidak.
- KMP: Menerima pola dan teks yang akan dicocokkan dengan algoritma KMP. Fungsi ini nilai boolean jika teks tersebut mengandung pola atau tidak.
- Levenshtein: Sebagai algoritma tambahan apabila tidak ditemukan kecocokan pada algoritma Boyer-Moore maupun KMP.
- ValidateDNA: Melakukan validasi pada DNA menggunakan regex.
- SanitizeInput: Melakukan sanitasi pada request untuk mendapatkan hasil. Sanitasi ini termasuk melakukan *parsing* pada tanggal dan nama penyakit.
- AddDiseaseRoute: Membuat rute POST pada controller AddDiseaseController.
- PredictPatientRoute: Membuat rute POST pada controller PredictPatientController.
- GetResultRoute: Membuat rute GET pada controller GetResultController.
- Main: Mengeksekusi program utama dari back-end, yaitu inisialisasi router dan melakukan *grouping* pada rute. Setelah itu, router dijalankan dan aplikasi back-end dapat digunakan.

## 2. Front-end

- onChange pada file Prediksi.vue: Memasukkan file DNA ke aplikasi web front-end.
- onSubmit pada Prediksi.vue: Membuat request dari user menjadi formData lalu mengirim nama, file, dan nama penyakit ke back-end dengan metode POST.
- onChange pada file Input.vue: Memasukkan file DNA ke aplikasi web front-end.
- onSubmit pada Input.vue: Membuat request dari user menjadi formData lalu mengirim file dan nama penyakit ke back-end dengan metode POST.
- onSubmit pada Hasil.vue: Meminta hasil prediksi pasien dari back-end.

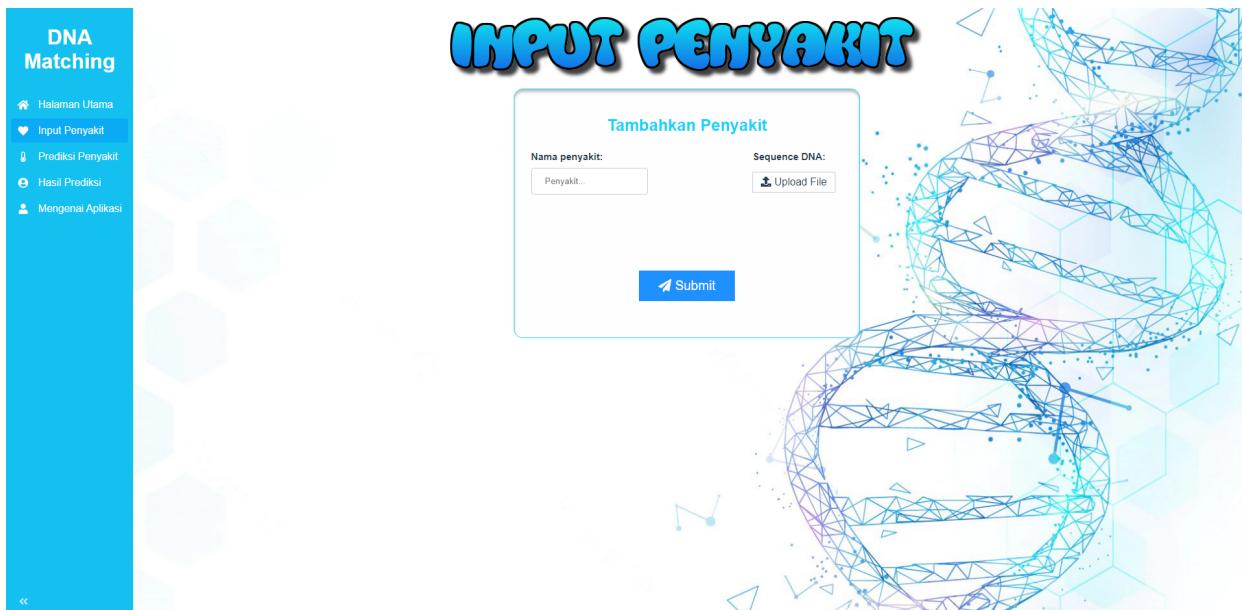
## 4.2 Tata Cara Penggunaan Program

### 4.2.1 Antarmuka Program

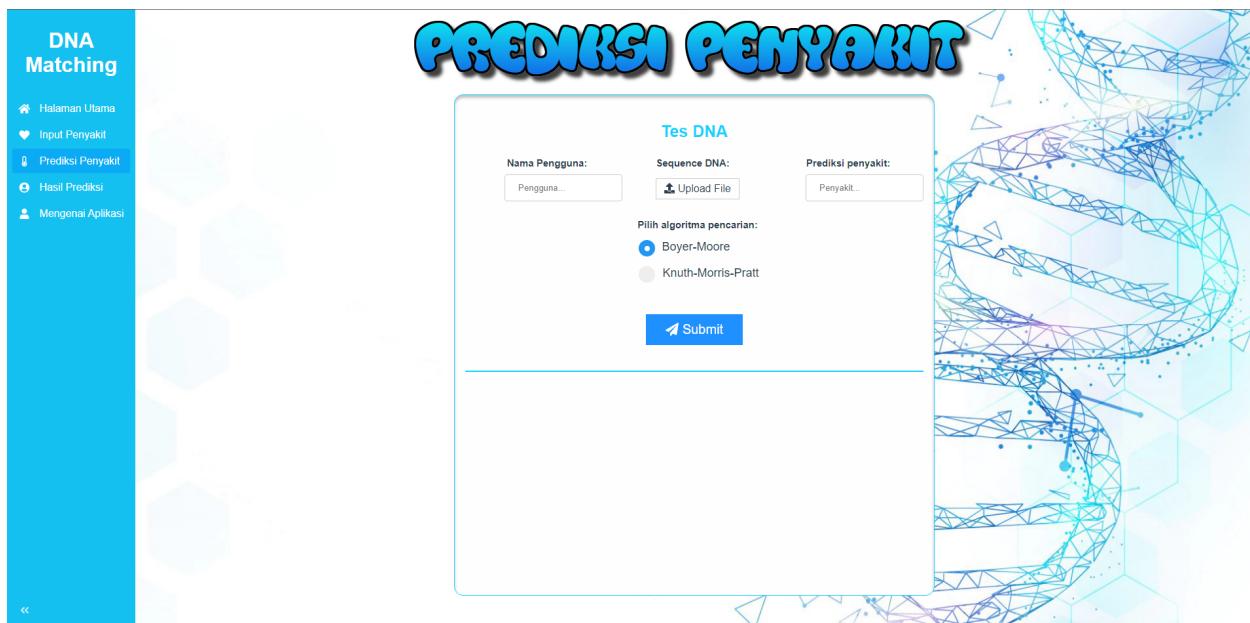
#### 1. Halaman Utama



#### 2. Input Penyakit



### 3. Prediksi Penyakit



PREDIKSI PENYAKIT

Tes DNA

Nama Pengguna: Pengguna...

Sequence DNA:

Prediksi penyakit: Penyakit...

Pilih algoritma pencarian:

Boyer-Moore

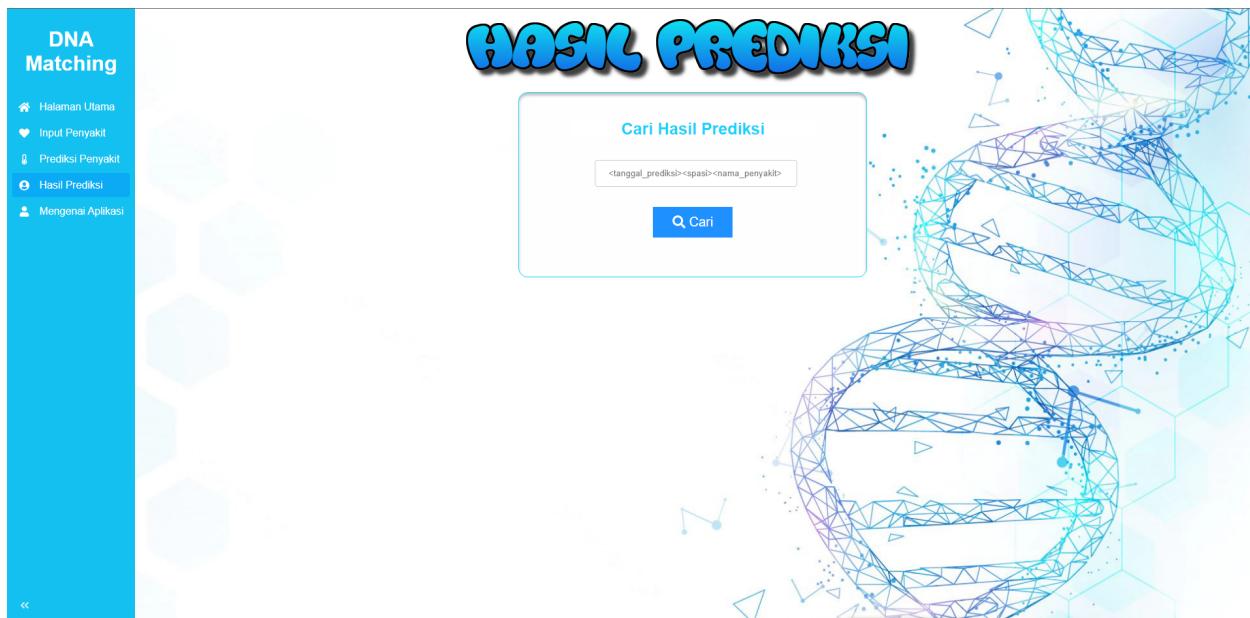
Knuth-Morris-Pratt

Submit

DNA Matching

- Halaman Utama
- Input Penyakit
- Prediksi Penyakit
- Hasil Prediksi
- Mengenai Aplikasi

### 4. Hasil Prediksi



HASIL PREDIKSI

Cari Hasil Prediksi

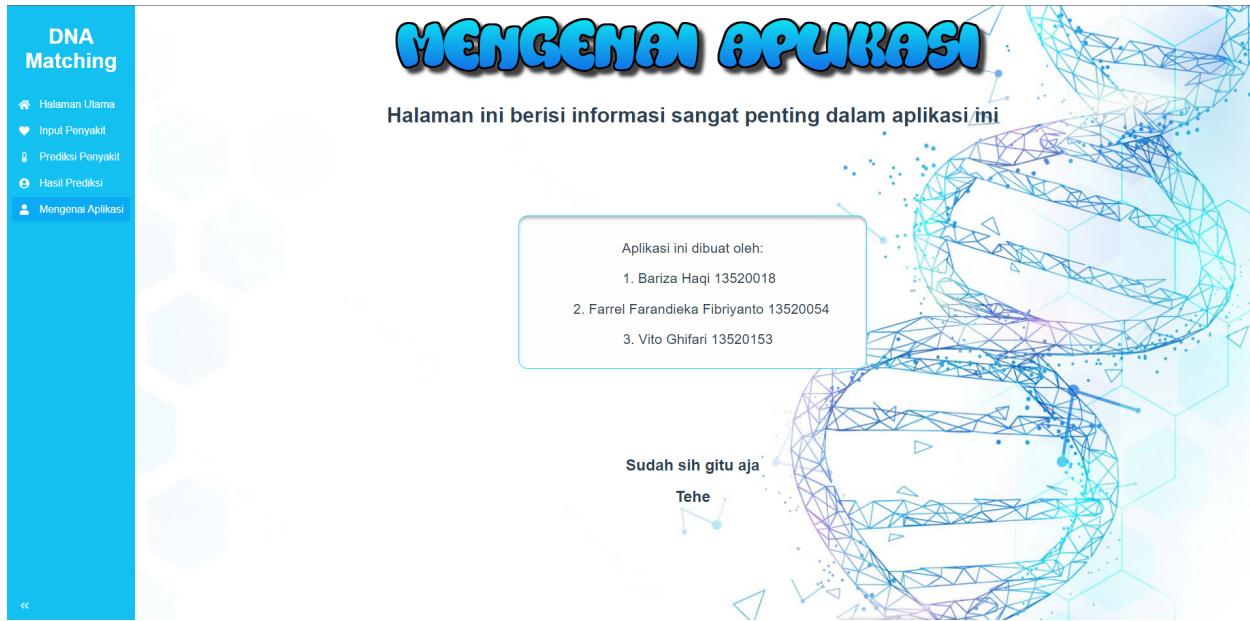
<tanggal\_prediksi><spasi><nama\_penyakit>

Cari

DNA Matching

- Halaman Utama
- Input Penyakit
- Prediksi Penyakit
- Hasil Prediksi
- Mengenai Aplikasi

### 5. Mengenai Aplikasi



#### 4.2.2 Fitur yang Disediakan Program

##### 1. Input Penyakit

Fitur ini dapat menerima input penyakit baru berupa nama penyakit dan sequence DNA-nya. Untuk penggunaannya isi nama penyakit pada kolom 'Nama Penyakit' dan upload sequence DNA-nya berupa txt melalui tombol 'Upload File' kemudian tekan tombol submit.

##### 2. Prediksi Penyakit

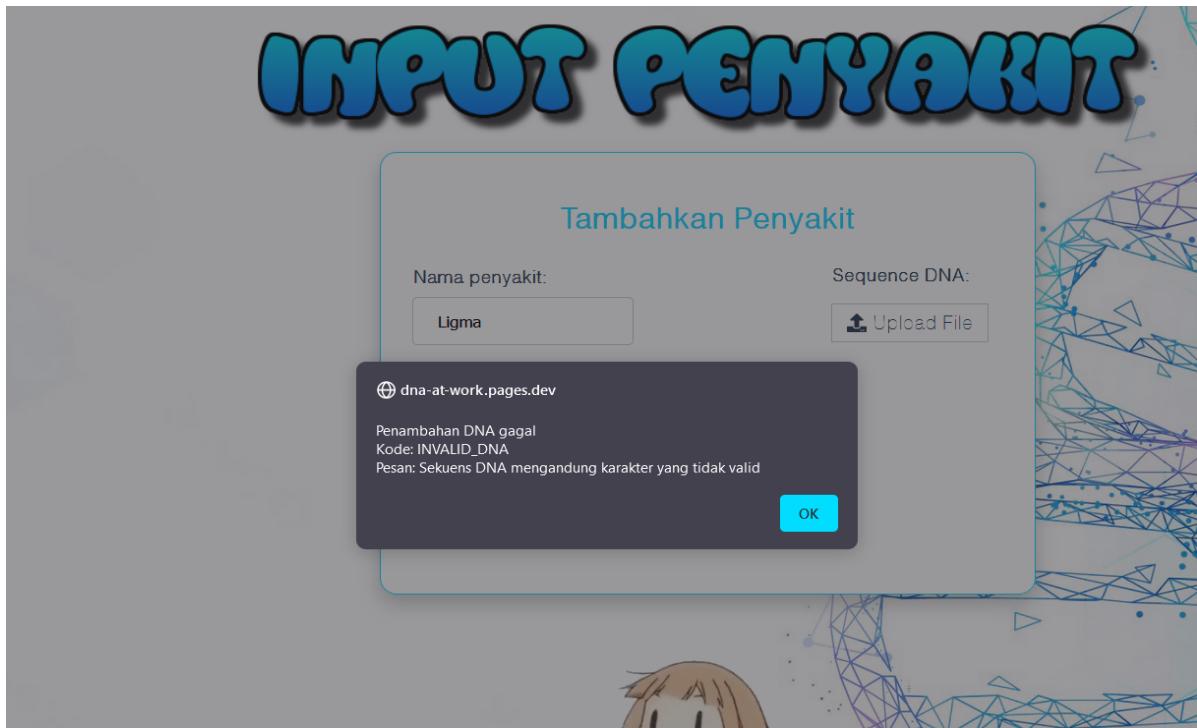
Fitur ini dapat memprediksi seseorang menderita penyakit tertentu berdasarkan sequence DNA-nya. Untuk penggunaannya isi nama pengguna pada kolom 'Nama Pengguna', upload sequence DNA-nya berupa txt melalui tombol 'Upload File', isi nama penyakit pada kolom 'Prediksi Penyakit' dan pilih algoritma pencarian antara Boyer-Moore atau Knuth-Morris-Pratt kemudian tekan tombol submit.

##### 3. Hasil Prediksi

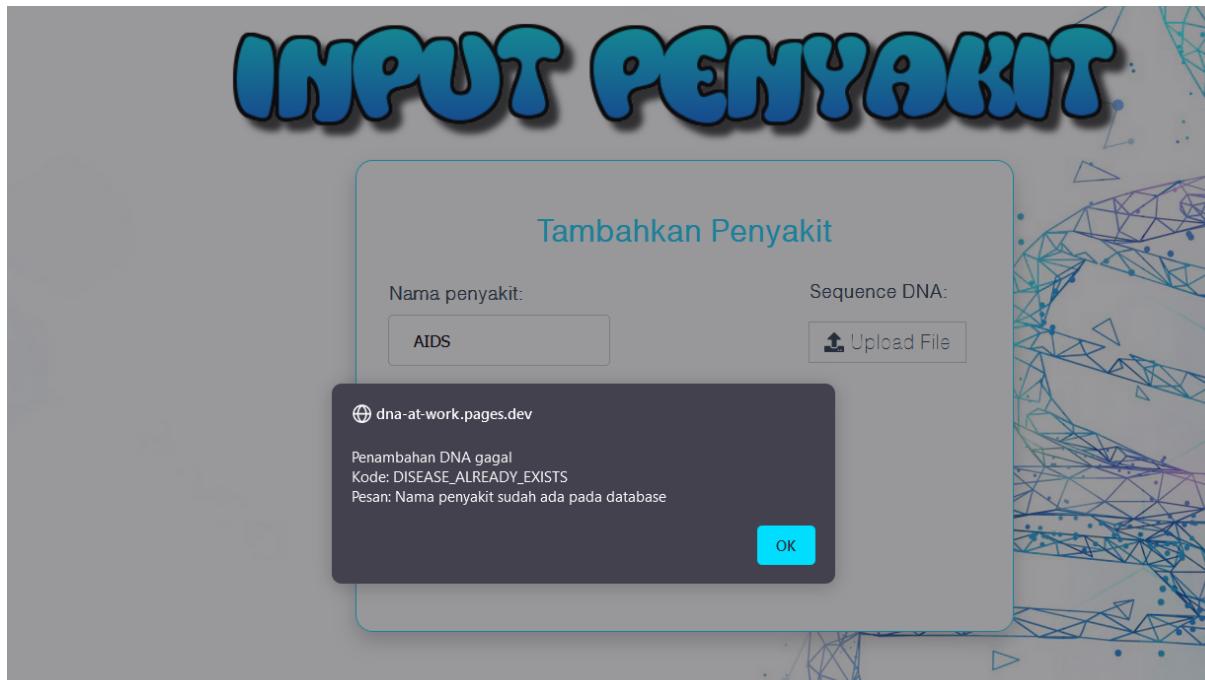
Fitur ini dapat menampilkan urutan hasil prediksi dari pencarian yang dilakukan. Untuk penggunaannya isi kolom pencarian dengan tanggal dengan format "tahun-bulan-tanggal" atau nama penyakit atau tanggal disertai nama penyakit dengan format "<tahun-bulan-tanggal><spasi><nama\_penyakit>"..

## 4.3 Hasil Pengujian

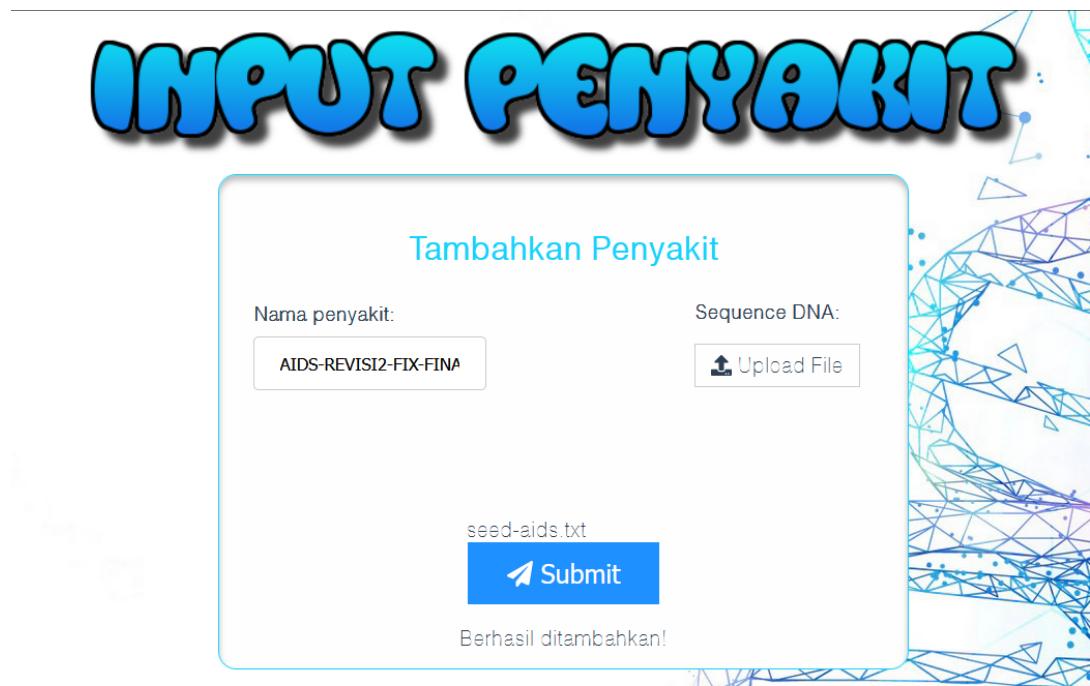
### 4.3.1 Hasil Pengujian Penambahan DNA Penyakit



Gambar 4.3.1.1 Kasus Penambahan DNA Penyakit Gagal Karena File Mengandung Karakter Selain ACGT



Gambar 4.3.1.2 Kasus penambahan DNA penyakit gagal karena nama penyakit sudah ada pada database



Gambar 4.3.1.3 Kasus penambahan DNA penyakit berhasil

#### 4.3.2 Hasil Pengujian Prediksi Penyakit Pasien

# PREDIKSI PENYAKIT

**Tes DNA**

Nama Pasien:  Sequence DNA:  Prediksi penyakit:

Pilih algoritma pencarian:  
 Boyer-Moore  
 Knuth-Morris-Pratt

seed-aids-patient.txt

Proses Selesai!

**Hasil Tes**

AntiVax - AIDS - TRUE - 100%



Gambar 4.3.2.1 Kasus Pengujian Prediksi Penyakit Pasien yang Positif dan 100% Cocok

# PREDIKSI PENYAKIT

**Tes DNA**

Nama Pasien:  Sequence DNA:  Prediksi penyakit:

Pilih algoritma pencarian:  
 Boyer-Moore  
 Knuth-Morris-Pratt

seed-aids-patient-likely.txt

Proses Selesai!

**Hasil Tes**

VaxedOnce - AIDS - TRUE - 90%



Gambar 4.3.2.2 Kasus Pengujian Prediksi Penyakit Pasien Positif tetapi Tidak 100% Cocok

# PREDIKSI PENYAKIT

**Tes DNA**

Nama Pasien:

Sequence DNA:

Prediksi penyakit:

Pilih algoritma pencarian:

Boyer-Moore

Knuth-Morris-Pratt

seed-aids-patient-not-likely.txt

Proses Selesai!

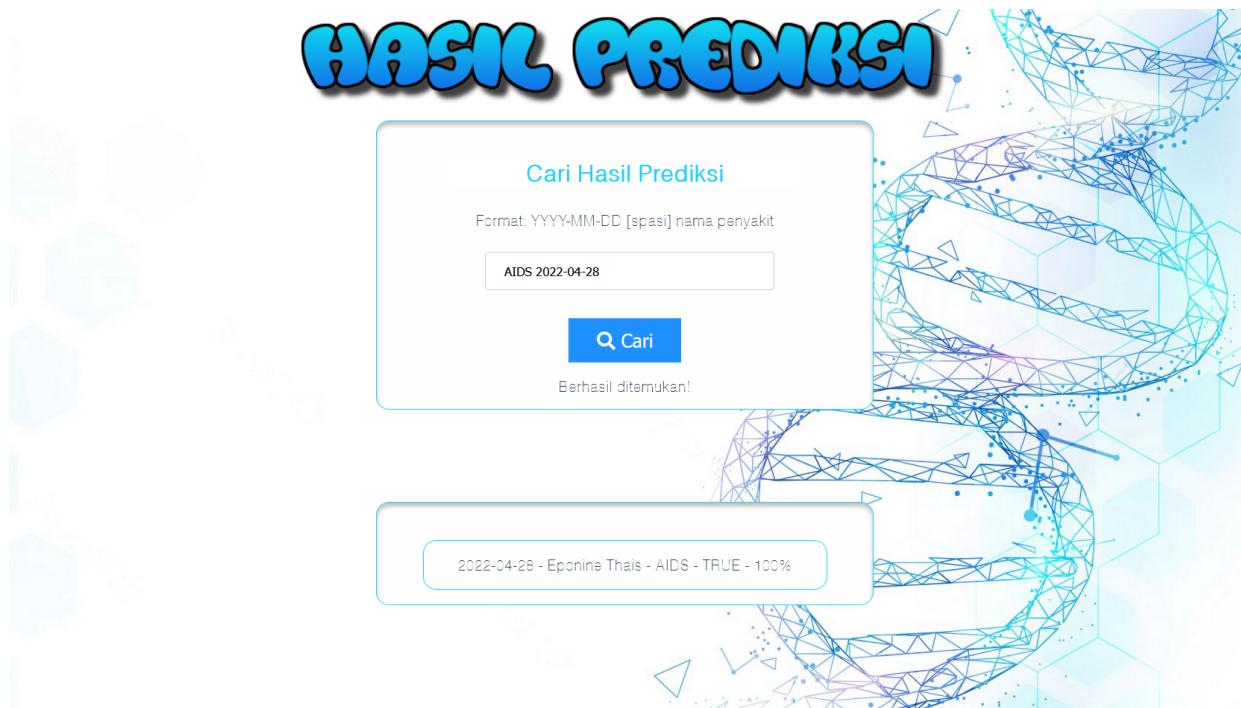
**Hasil Tes**

BoosterEnjoyer - AIDS - FALSE - 60.000004%

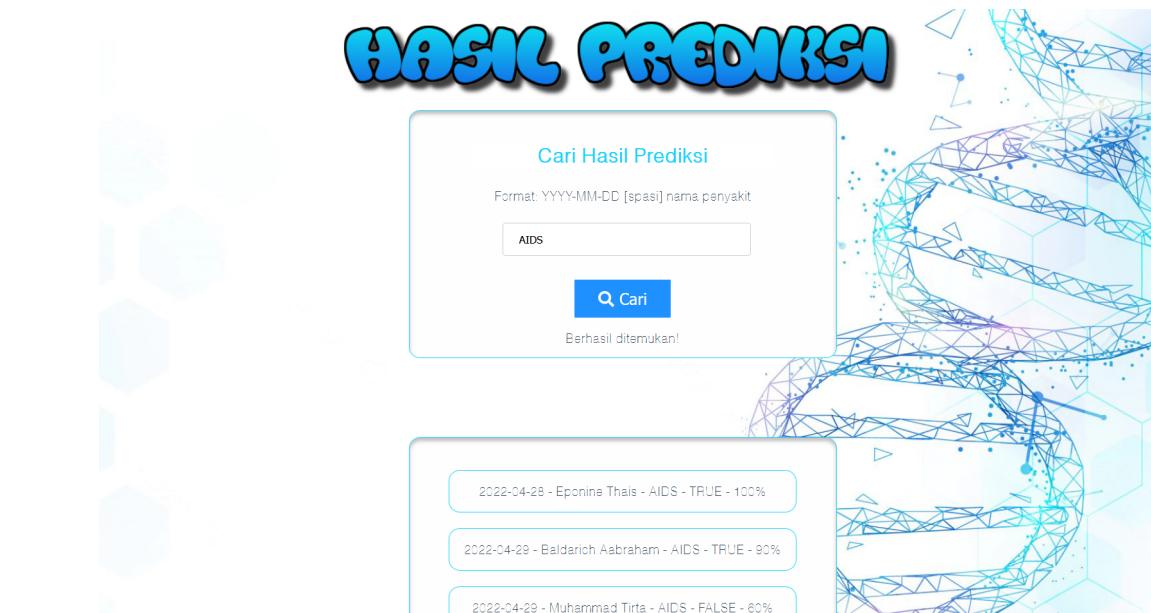


Gambar 4.3.2.3 Kasus Pengujian Prediksi Penyakit Pasien yang di Bawah 80% Kemiripan

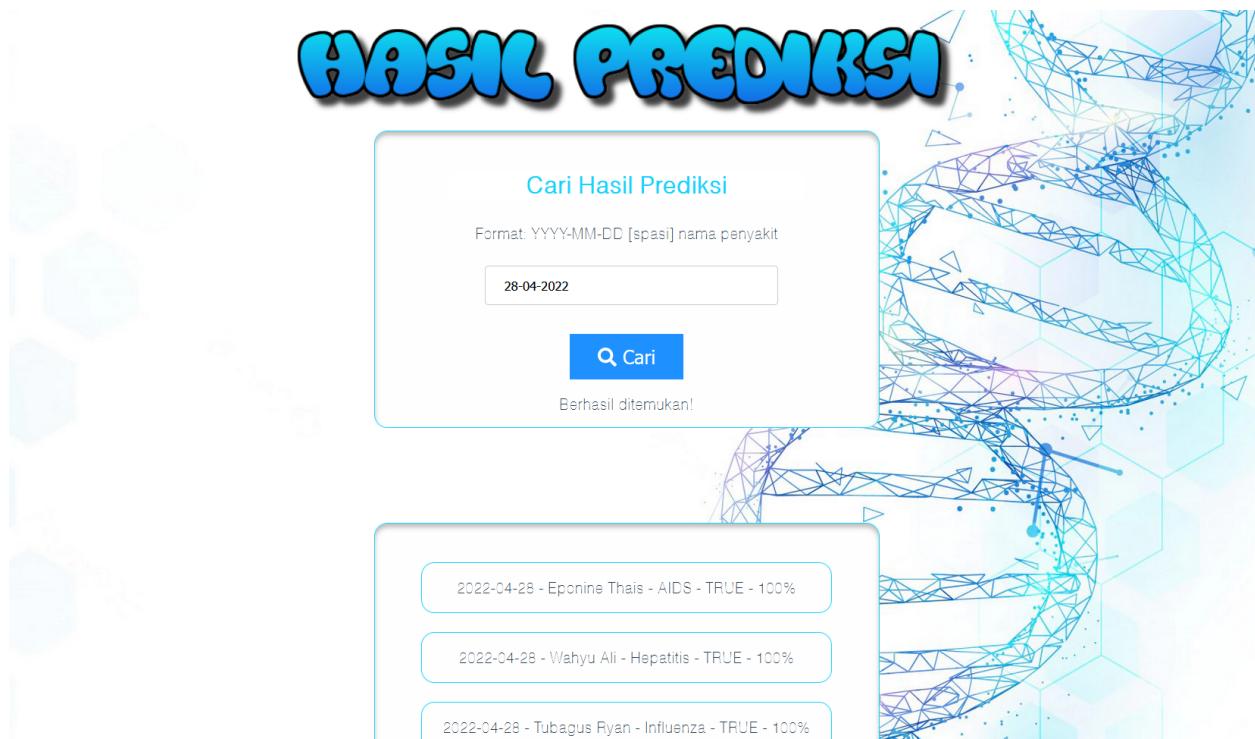
#### 4.3.3 Hasil Pengujian Cari Prediksi DNA



Gambar 4.3.3.1 Hasil Prediksi dengan Format Nama Penyakit Diikuti dengan Tanggal Prediksi



Gambar 4.3.3.2 Hasil Prediksi dengan Format Nama Penyakit Saja



Gambar 4.3.3.2 Hasil Prediksi dengan Format Tanggal Prediksi Saja

#### 4.4 Analisis Hasil Pengujian

Dari hasil analisis pengujian, aplikasi dapat melakukan input penyakit, prediksi penyakit dan hasil prediksi dengan benar. Selain itu aplikasi dapat menghitung tingkat kemiripan DNA pengguna dengan DNA penyakit pada tes DNA menggunakan Levenshtein distance dengan tepat. Aplikasi juga dapat memunculkan pesan error dengan benar bila terjadi error pada bagian spesifik dari program.

Untuk sanitasi input menggunakan regex agar memastikan bahwa masukan merupakan sequence DNA yang valid sudah sesuai. Untuk pilihan algoritma string matching, baik algoritma Boyer-Moore maupun Knuth-Morris-Pratt akan menghasilkan hasil yang sama pada prediksi penyakit. Untuk waktu yang dibutuhkan, masing-masing algoritma memiliki keunggulan masing-masing tergantung input yang dimasukkan.

## Bab 5

### Kesimpulan

#### 5.1 Kesimpulan

Berdasarkan implementasi String Matching dan Regular Expression pada program dan eksperimen yang telah dilakukan, didapatkan kesimpulan sebagai berikut.

1. Aplikasi “DNA At Work!” dapat melakukan input penyakit, prediksi penyakit dan pencarian hasil prediksi
2. Aplikasi berhasil menjalankan fungsinya dengan memanfaatkan algoritma String Matching dan Regular Expression.
3. Aplikasi juga berhasil menghitung tingkat kemiripan DNA pengguna dengan DNA penyakit pada tes dengan memanfaatkan Levenshtein distance.

#### 5.2 Saran

Berdasarkan implementasi String Matching dan Regular Expression pada program dan eksperimen yang telah dilakukan, kami memberikan beberapa saran sebagai berikut.

1. Melakukan perbandingan dari segi waktu eksekusi pada algoritma KMP, Boyer-Moore, maupun Levenshtein.
2. Menerima query pencarian dalam format lain, misalnya selain menulis tanggal dengan format angka semua, kita bisa menulis nama bulannya juga.

## Daftar Pustaka

- Rinaldi Munir, Diakses 27 April 2022  
<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/stima21-22.htm>
- Boytsov, Leonid (Mei 2011). *Indexing methods for approximate dictionary searching*. Journal of Experimental Algorithmics
- Wikipedia, Diakses 28 April 2022  
[https://en.wikipedia.org/wiki/Regular\\_expression](https://en.wikipedia.org/wiki/Regular_expression)
- <https://github.com/gin-gonic/gin> Diakses 28 April 2022
- <https://dasarpemrogramangolang.novalagung.com/A-map.html> Diakses 28 April 2022
- [https://en.wikipedia.org/wiki/Damerau%E2%80%93Levenshtein\\_distance](https://en.wikipedia.org/wiki/Damerau%E2%80%93Levenshtein_distance)  
Diakses 28 April 2022

## Link Repositori dan Video Penjelasan

- Repositori : [https://github.com/VanillaMacchiato/Tubes3\\_13520018](https://github.com/VanillaMacchiato/Tubes3_13520018)
- Video Penjelasan : <https://youtu.be/MA41hOydeHo>