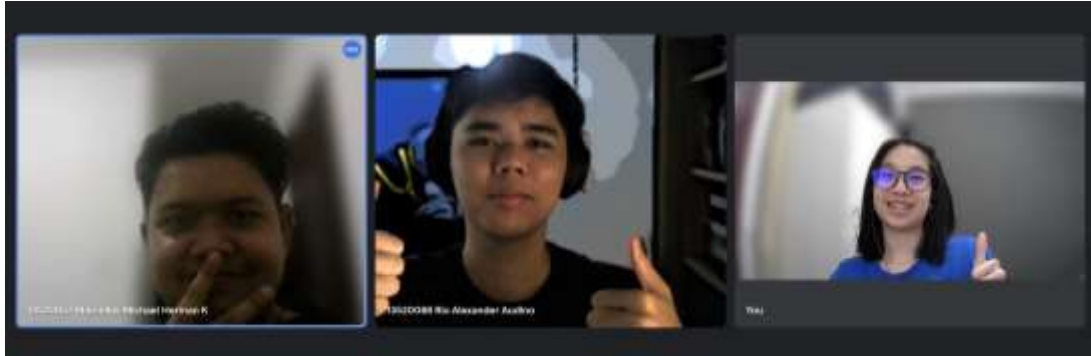


## **LAPORAN TUGAS BESAR**

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

Ditujukan untuk memenuhi salah satu tugas besar mata kuliah IF2211 Strategi Algoritma  
pada Semester II Tahun Akademik 2021/2022



Disusun oleh:

**Marcellus Michael H. K. (K3)      13520094**

**Rio Alexander Audino (K1)      13520088**

**Maria Khelli (K1)      13520115**

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

**2022**

## DAFTAR ISI

BAB I DESKRIPSI TUGAS .....	2
BAB II LANDASAN TEORI.....	4
2.1.1 Algoritma Knuth-Morris-Pratt .....	4
2.1.2 Algoritma Boyer-Moore .....	4
2.1.3 Regular expression (regex).....	4
2.1.4 Teori Aplikasi Web .....	4
2.1.5 Aplikasi Web yang Dibangun.....	5
BAB III ANALISIS PEMECAHAN MASALAH.....	6
3.1. Langkah Penyelesaian Masalah .....	6
3.2. Fitur Fungsional dan Arsitektur Aplikasi Web.....	7
BAB IV IMPLEMENTASI DAN PENGUJIAN.....	8
4.1. Spesifikasi Teknis Program.....	8
4.2. Tata Cara Penggunaan Program .....	10
4.3. Hasil Pengujian.....	13
4.4. Analisis Hasil Pengujian.....	19
BAB V KESIMPULAN DAN SARAN.....	20
5.1. Kesimpulan.....	20
5.2. Saran .....	20
LINK PENTING .....	21
DAFTAR PUSTAKA .....	22

## BAB I DESKRIPSI TUGAS

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

Fitur-Fitur Aplikasi:

1. Aplikasi dapat menerima input penyakit baru berupa nama penyakit dan sequence DNA-nya (dan dimasukkan ke dalam database).
  - a. Implementasi input sequence DNA dalam bentuk file.
  - b. Dilakukan sanitasi input menggunakan regex untuk memastikan bahwa masukan merupakan sequence DNA yang valid (tidak boleh ada huruf kecil, tidak boleh ada huruf selain AGCT, dan tidak ada spasi).
2. Aplikasi dapat memprediksi seseorang menderita penyakit tertentu berdasarkan sequence DNA-nya.
  - a. Tes DNA dilakukan dengan menerima input nama pengguna, sequence DNA pengguna, dan nama penyakit yang diuji. Asumsi sequence DNA pengguna > sequence DNA penyakit.
  - b. Dilakukan sanitasi input menggunakan regex untuk memastikan bahwa masukan merupakan sequence DNA yang valid (tidak boleh ada huruf kecil, tidak boleh ada huruf selain AGCT, tidak ada spasi, dll).
  - c. Pencocokan sequence DNA dilakukan dengan menggunakan algoritma string matching.
  - d. Hasil dari tes DNA berupa tanggal tes, nama pengguna, nama penyakit yang diuji, dan status hasil tes. Contoh: 1 April 2022 - Mhs IF - HIV - False
  - e. Semua komponen hasil tes ini dapat ditampilkan pada halaman web (refer ke poin 3 pada “Fitur-Fitur Aplikasi”) dan disimpan pada sebuah tabel database.
3. Aplikasi memiliki halaman yang menampilkan urutan hasil prediksi dengan kolom pencarian di dalamnya. Kolom pencarian bekerja sebagai filter dalam menampilkan hasil.
  - a. Kolom pencarian dapat menerima masukan dengan struktur: <tanggal\_prediksi><spasi><nama\_penakit>, contoh “13 April 2022 HIV”. Format penanggalan dibebaskan, jika bisa menerima >1 format lebih baik.
  - b. Kolom pencarian dapat menerima masukan hanya tanggal ataupun hanya nama penyakit. Fitur ini diimplementasikan menggunakan regex.
4. (Bonus) Menghitung tingkat kemiripan DNA pengguna dengan DNA penyakit pada tes DNA

- a. Ketika melakukan tes DNA, terdapat persentase kemiripan DNA dalam hasil tes. Contoh hasil tes: 1 April 2022 - Mhs IF - HIV - 75% - False
- b. Perhitungan tingkat kemiripan dapat dilakukan dengan menggunakan Hamming distance, Levenshtein distance, LCS, atau algoritma lainnya (dapat dijelaskan dalam laporan).
- c. Tingkat kemiripan DNA dengan nilai lebih dari atau sama dengan 80% dikategorikan sebagai True. Perlu diperhatikan mengimplementasikan atau tidak mengimplementasikan bonus ini tetap dilakukan pengecekan string matching terlebih dahulu.

## BAB II LANDASAN TEORI

### 2.1.1 Algoritma Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt (KMP) adalah algoritma pencocokan string yang mencari kata *pattern* dengan urutan kiri ke kanan. Algoritma KMP termasuk dalam algoritma *exact string matching*, berarti pencocokan kata dilakukan secara tepat baik secara urutan dan huruf. Proses pencocokan string mirip dengan algoritma Brute Force. Bedanya, algoritma KMP menggunakan heuristik, yaitu kemiripan suffix dan prefix yang bernama KMP Border Function sehingga mengurangi pengecekan berulang. Fungsi inilah yang nantinya digunakan dalam proses pengecekan.

### 2.1.2 Algoritma Boyer-Moore

Algoritma Boyer-Moore (BM) adalah salah satu algoritma *exact string matching* yang mencari kata *pattern* dengan urutan dari kanan ke kiri. Proses pergeseran index pencocokan didasarkan pada hasil pemetaan Last Occurrence Function. Algoritma Boyer-Moore lebih baik daripada algoritma Brute Force. Namun, jika dibandingkan dengan algoritma KMP, algoritma BM lebih cepat jika alfabet dalam kasus bervariasi. Sebaliknya, BM lebih lambat daripada KMP jika alfabet dalam kasus tidak bervariasi.

### 2.1.3 Regular expression (regex)

*Regex* adalah sebuah rangkaian string untuk mencari sebuah pola dalam sebuah teks. Rangkaian string ini dibuat berdasarkan simbol dengan aturan-aturan tertentu, misalnya tanda bintang (\*) berarti nol atau banyak kemunculan. Hampir semua bahasa pemrograman menyediakan pustaka untuk implementasi *regex*. *Regex* biasanya digunakan untuk persoalan *string-matching* dalam pemrosesan tulisan.

### 2.1.4 Teori Aplikasi Web

Menurut Simarmata (2010:56), aplikasi web adalah sebuah sistem informasi yang mendukung interaksi pengguna melalui antarmuka berbasis web. Aplikasi web dapat berjalan pada jaringan internet maupun intranet (LAN). Ada berbagai jenis aplikasi web, seperti web media sosial, web berbasis sistem informasi, web jual beli dan bisnis, web pencarian, web informasi dan berita, aplikasi web server, dan aplikasi web browser.

Dalam pembuatan website, *client* dan *server* menggunakan HTTP atau *Hypertext Transfer Protocol*. Menurut Hidayatullah dan Kawistara (2015:5), *Hypertext Transfer Protokol* (HTTP) adalah *protocol* agar *client* dan *server* bisa berkomunikasi dengan gaya *request-response*. Penggunaan HTTP dapat menentukan bagaimana format pesan dan cara mengirimkannya, serta bagaimana web dapat beraksi terhadap suatu perintah.

Untuk berkomunikasi antara *client* dan *server*, digunakan API. API atau *Application Programming Interface* adalah *interface* yang dapat menghubungkan satu aplikasi dengan aplikasi lainnya. API dapat digunakan untuk berkomunikasi dengan berbagai bahasa pemrograman yang berbeda.

### 2.1.5 Aplikasi Web yang Dibangun

Arsitektur metode komunikasi yang digunakan dalam pembuatan web adalah REST. REST (Representational State Transfer) adalah sebuah arsitektur metode komunikasi yang menggunakan protokol HTTP untuk pertukaran data dimana metode ini sering diterapkan dalam pengembangan aplikasi.

Pembuatan aplikasi web juga menggunakan *framework* sebagai alat bantu. Menurut Hakim (2010:3), *framework* adalah koleksi atau kumpulan potongan-potongan program yang disusun atau diorganisasikan sedemikian rupa, sehingga dapat digunakan untuk membantu membuat aplikasi utuh tanpa harus membuat semua kodenya dari awal.

Untuk melakukan penyimpanan data-data yang telah dimasukkan oleh pengguna, digunakan basis data. Menurut Connolly dan Begg (2010:65), basis data adalah sebuah kumpulan data yang secara logis terkait dan dirancang untuk memenuhi suatu kebutuhan informasi dari sebuah organisasi. Dalam mengelola data yang terdapat di basis data, digunakan DBMS atau *Database Management System*. Menurut Laudon, K.C. & Laudon, Jane.P. (2012), *Database Management System* (DBMS) adalah perangkat lunak yang memungkinkan sebuah organisasi untuk memusatkan data, mengelola secara efisien, dan memberikan akses ke data yang disimpan oleh program aplikasi. Contoh dari DBMS adalah MySQL, PostgreSQL, dan SQLite.

Pada aplikasi web yang dikembangkan untuk tugas besar ini, digunakan bahasa pemrograman Javascript menggunakan framework Next Js sebagai *client* dan bahasa pemrograman Golang dengan menggunakan framework Gin sebagai *server*. Antara *client* dan *server* berkomunikasi menggunakan RESTful API. Untuk membantu melakukan penyimpanan pada basis data, dapat digunakan MySQL sebagai DBMS.

## BAB III

### ANALISIS PEMECAHAN MASALAH

#### 3.1. Langkah Penyelesaian Masalah

Pada program yang kami buat, terdapat empat fitur utama yang dijalankan, yakni

- a. Penerimaan DNA penyakit dan DNA manusia.
- b. Mendeteksi keberadaan DNA penyakit dengan string matching.
- c. Menampilkan hasil deteksi penyakit berdasarkan jenis penyakit.
- d. Terakhir menghitung tingkat kemiripan DNA pengguna dengan DNA penyakit test.

Berikut adalah penjelasan untuk fitur a :

Pada bagian antarmuka (*front-end*), pengguna akan memasukkan file DNA penyakit (jika ingin menambahkan penyakit atau diagnosis) atau DNA manusia (hanya jika ingin menambahkan diagnosis). Kemudian, *front-end* akan membaca isi file teks (.txt) dari pengguna. *Front-end* tidak melakukan pengecekan apakah isi file sudah benar atau tidak. Pengecekan pola yang valid (hanya boleh huruf besar dari “AGCT”) dilakukan pada *back-end*.

*Front-end* akan melakukan *hit* pada *endpoint* yang disediakan *back-end* sesuai dengan methodnya dan akan mengirimkan data dalam format JSON jika dibutuhkan. Ketika di-*hit*, *back-end* akan mengirimkan respons kepada client dalam bentuk kode status HTTP dan data berformat JSON.

Berikut adalah penjelasan untuk fitur b, c, dan d :

Untaian DNA pengguna dan DNA penyakit yang sudah diterima selanjutnya akan diproses melalui algoritma string matching yang dipilih. Terdapat lima algoritma umum yang terdapat dalam program kami, 4 algoritma adalah algoritma pencocokan kata dan satu algoritma adalah algoritma mencari persentase kecocokan. 4 algoritma pencocokan kata adalah Boyers-Moore, Brute Force (Naïve), KMP, dan Regular Expressions. Algoritma pencari persentase kemiripan RNA yang kami gunakan adalah algoritma waterman-smith. Dari empat algoritma yang disediakan, akan dipilih salah satu untuk dijalankan. Selain menjalankan algoritma pencocokan kata, program juga akan menjalankan algoritma pencari kemiripan kata. Memanfaatkan konkurensi dari bahasa golang, kedua algoritma tersebut dapat dijalankan secara konkuren sehingga prosesnya bisa lebih cepat.

Algoritma pencocokan kata akan menerima *pattern* (DNA penyakit) dan juga *text* (DNA manusia pengguna). Dari kedua parameter ini selanjutnya akan diproses dan mengembalikan index kemunculan pertama *pattern*. Jika tidak ditemukan kata yang sama, algoritma akan mengembalikan -1. Di saat yang bersamaan, algoritma pencari persentase kemiripan kata juga dijalankan. Algoritma ini nantinya akan mengembalikan

persentasi kemiripan tertinggi yang ditemukan. Dari nilai index dan juga tingkat kemiripan, selanjutnya akan diproses apakah dna manusia tertentu terdeteksi penyakit. Proses pendeteksian penyakit cukup mudah, selama index ditemukannya bukan -1 atau tingkat kemiripan lebih dari 80%, program akan menyatakan DNA tersebut positif penyakit test.

### **3.2. Fitur Fungsional dan Arsitektur Aplikasi Web**

#### *3.2.1 Front-end*

Arsitektur pada bagian frontend terdiri dari 4 halaman, yaitu halaman indeks sebagai halaman utama, halaman tes-dna sebagai halaman untuk melakukan pengetesan DNA, halaman tambah-penyakit sebagai halaman untuk menambahkan penyakit baru beserta untai DNA-nya, dan halaman cari-riwayat-tes sebagai halaman untuk mencari riwayat dilakukannya pengetesan. Selain itu, terdapat folder umum yang berisi komponen-komponen bantu untuk membuat website.

#### *3.2.2 Back-end*

Arsitektur yang diimplementasikan *back-end* adalah arsitektur MVC, yaitu Model-View-Controller. View sendiri sudah ditangani oleh Front-end (menampilkan data). Model adalah definisi struct dari kolom-kolom tabel sebuah instansi data (data logic), misalnya penyakit atau diagnosis. Kemudian, untuk menangani data flow dan respons untuk *endpoint*, digunakan controller.



## BAB IV

### IMPLEMENTASI DAN PENGUJIAN

#### 4.1. Spesifikasi Teknis Program

##### 4.1.1 Struktur data

- *Hashmap* : digunakan dalam beberapa hal, salah satunya untuk menyimpan hasil pencarian pencocokan kata beserta tingkat kemiripannya. Hasmap yang digunakan dalam program ini antara lain map *string to float32*, map *byte to bool*, dan map *byte to int*.
- *Golang Channel* : digunakan untuk mengakomodir proses konkurensi algoritma pencocokan kata dan algoritma pencarian persentase kemiripan kata. Untuk memudahkan proses konkurensi, Golang Channel yang digunakan bertipe integer
- *2-D Integer Matrix* : digunakan untuk proses pencarian persentase kemiripan kata.
- *Array of Integer* : digunakan saat pengisian 2-D Matrix pada proses pencarian kemiripan kata
- *Integer, Float, byte, string, char* : digunakan secara umum

##### 4.1.2 Fungsi

###### 4.1.2.1 Backend

- a. Fungsi *PatternIsValid(text: string) → boolean*

Untuk mengecek apakah masukan pola DNA sudah sesuai, yaitu hanya terdiri dari “AGCT” yang berhuruf besar. Fungsi ini menggunakan *regex* dengan pattern “[^AGCT]”. Jika ada yang cocok, berarti masukan salah.

- b. Fungsi *ExtractQuery(text: string) → (string, string)*

Fungsi ini untuk mengolah kueri pencarian agar menjadi tuple (tanggal, nama) dari masukan search bar oleh pengguna sebelum dilakukan pencarian pada database.

###### 4.1.2.2 Algoritma String Matching

Fungsi utama algoritma string matching adalah *DNA\_String\_Matching()*. Fungsi ini akan menerima beberapa parameter untuk proses pengolahan string. Fungsi ini akan mengembalikan sebuah float dan boolean yang menyatakan secara beturut-turut tingkat kemiripan pattern dan keterangan kepemilikan penyakit. Dalam fungsi ini terdapat beberapa fungsi lagi yang menangani kalkulasi tiap jenis algoritma. Selain algoritma pencocokan kata, pada fungsi ini juga memanggil fungsi tingkat kemiripan kata. Kedua algoritma ini dijalankan secara konkuren menggunakan konkurensi dari bahasa go. Berikut penjelasan tiap jenis algoritmanya.

- a. Algoritma Boyer-Moore

*BmooreMatcher()* adalah fungsi yang melakukan pencocokan string menggunakan algoritma Bmoore. Di dalam fungsi ini akan dipanggil lagi

fungsi mapping pattern *bmMapper()* yang dipetakan berdasarkan Last Occurrence Function. Ketika proses pencocokan kata sudah selesai, fungsi akan mengembalikan indeks ditemukannya dna penyakit atau -1 jika tidak ditemukan dna yang cocok.

b. Algoritma KMP

*KMPMatcher()* adalah fungsi yang melakukan pencocokan string menggunakan algoritma KMP. Di dalam fungsi ini akan dipanggil lagi fungsi mapping pattern *kmpMapper()* yang dipetakan berdasarkan kecocokan prefix dan suffixnya. Ketika proses pencocokan kata sudah selesai, fungsi akan mengembalikan indeks ditemukannya dna penyakit atau -1 jika tidak ditemukan dna yang cocok.

c. Algoritma Regex

*RegexMatch()* adalah fungsi yang melakukan pencocokan string menggunakan algoritma Regular Expression. Proses pemetaan aturan dan pola Regex dilakukan menggunakan algoritma external “regex”. Ketika proses pencocokan kata sudah selesai, fungsi akan mengembalikan indeks ditemukannya dna penyakit atau -1 jika tidak ditemukan dna yang cocok.

d. Algoritma Bruteforce

*BruteForceMatching()* adalah fungsi yang melakukan pencocokan string menggunakan algoritma *Brute-Force* atau *Naïve*. Proses pencocokan string dilakukan satu per satu di setiap substringnya sampai huruf substring terakhir atau sampai ditemukannya dna yang sama. Ketika proses pencocokan kata sudah selesai, fungsi akan mengembalikan indeks ditemukannya dna penyakit atau -1 jika tidak ditemukan dna yang cocok.

e. String similarity

Proses pencarian persentasi kemiripan dna dilakukan menggunakan algoritma Smith-Waterman Algorithm. Algoritma ini termasuk dalam algoritma Local Alligment Algorithm. Algoritma ini ditangani dalam file *smith\_waterman.go* dan *utils.go* sebagai pendukung. DNA penyakit dan DNA manusia akan dipetakan terlebih dahulu dalam sebuah matriks. Setiap matriks selanjunya akan diisi berdasarkan aturan pengisian Smith-Waterman. Selama pengisian juga dicatat nilai maksimum yang berhasil diperoleh. Ketika proses selesai, akan dicari substring dengan kemiripan tertinggi. Substring ini yang selanjutnya dikembalikan ke program utama.

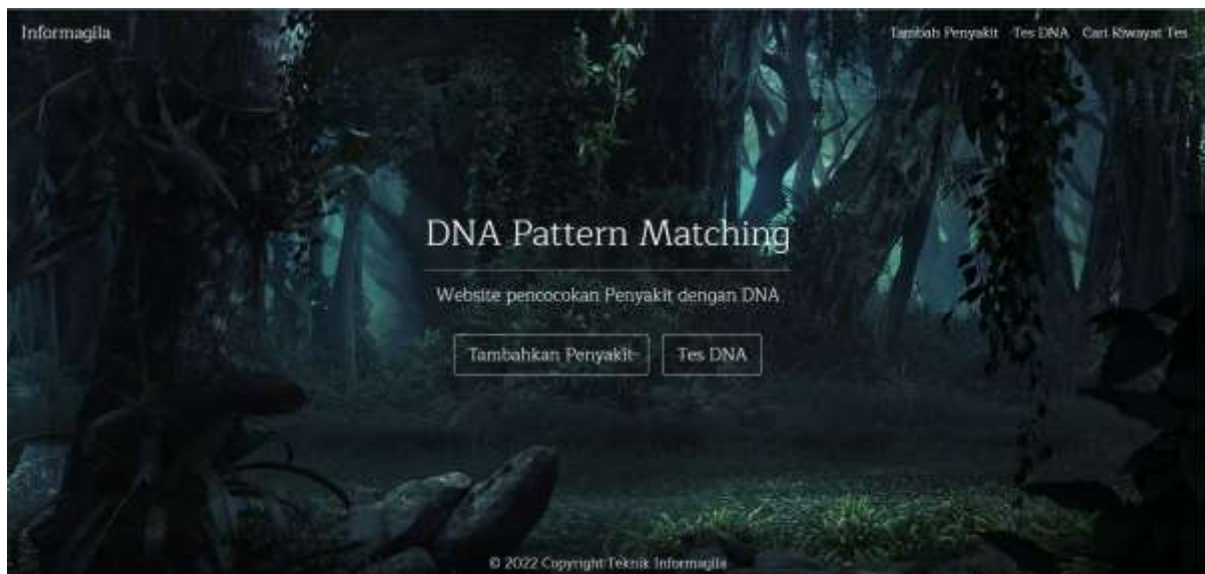
#### 4.1.3 Prosedur

##### 4.1.3.1 Frontend

###### a. Procedure tambahPenyakit(Input: Event e)

Prosedure akan menerima input file dan membaca file tersebut, kemudian menulis isi file tersebut menjadi sebuah string yang kemudian akan diberikan ke server melalui protokol HTTP.

## 4.2. Tata Cara Penggunaan Program



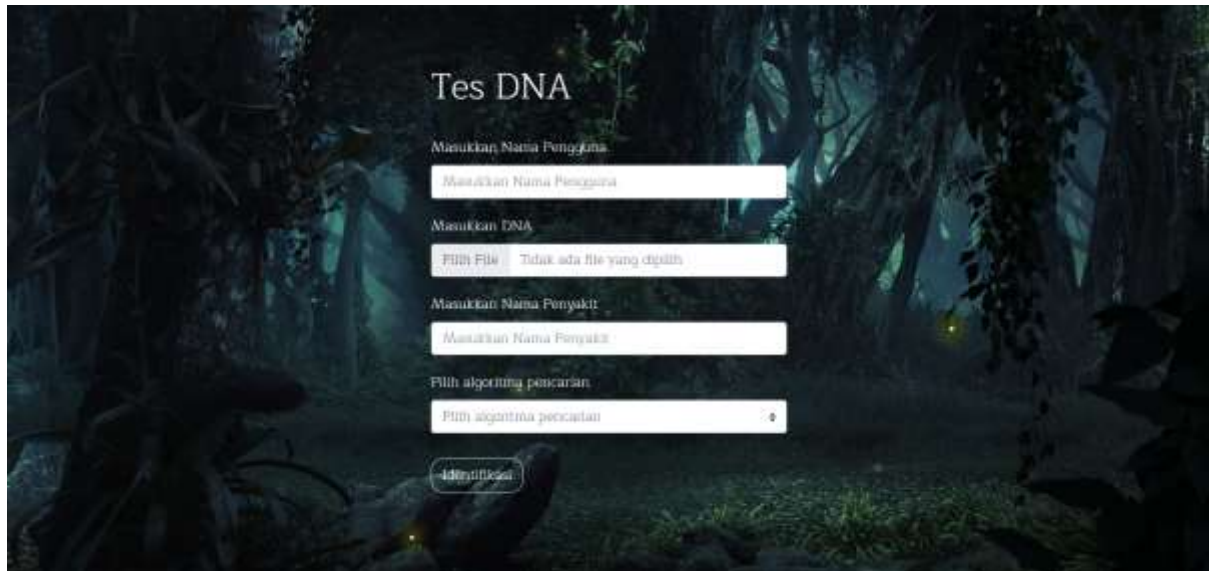
Gambar 4.2.1. Tampilan utama website

Saat memasuki website pertama kali, pengguna akan masuk ke tampilan utama website. Terdapat navigation bar yang dapat membawa pengguna ke halaman lain seperti halaman untuk menambahkan penyakit, mengetes apakah seseorang terkena penyakit, dan melihat riwayat tes.



Gambar 4.2.2. Tampilan menambahkan penyakit

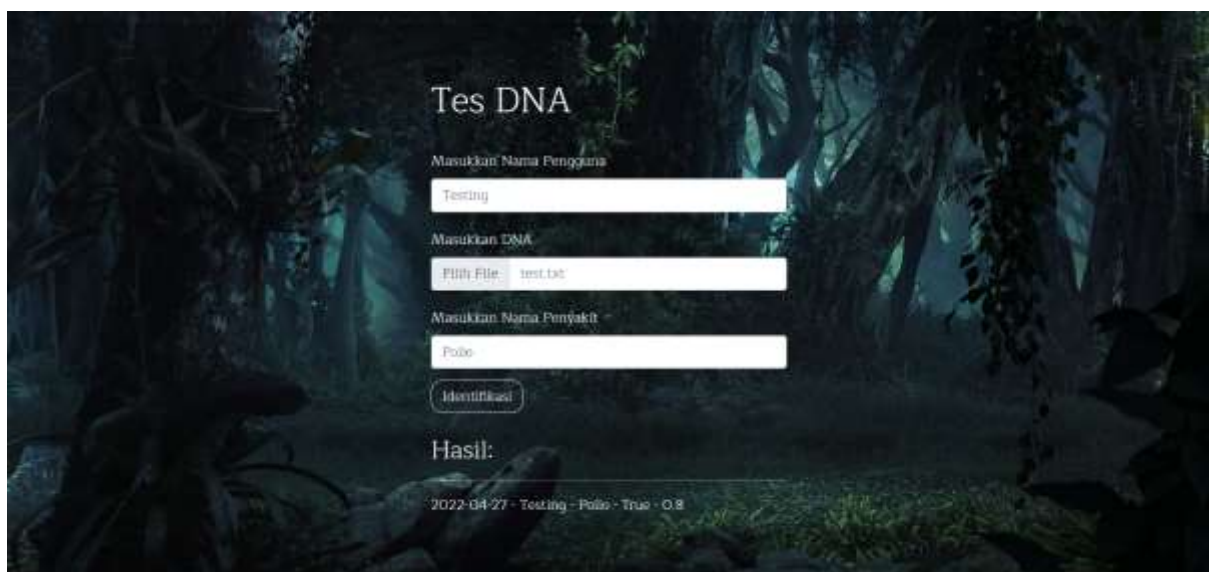
Pengguna dapat menambahkan penyakit pada halaman ini. Terdapat dua buah kolom input yang menerima nama penyakit dan untai DNA berupa file berekstensi .txt. Jika pengguna menekan tombol tambahkan, data akan tersimpan pada basis data.



The screenshot shows a web form titled "Tes DNA" set against a dark, forest-like background. The form contains four input fields: "Masukkan Nama Pengguna" (empty), "Masukkan DNA" (with a "Pilih File" button and the text "Tidak ada file yang dipilih"), "Masukkan Nama Penyakit" (empty), and "Pilih algoritma pencarian" (with a dropdown menu showing "Pilih algoritma pencarian"). Below these fields is an "Identifikasi" button.

Gambar 4.2.3. Tampilan mengetes DNA

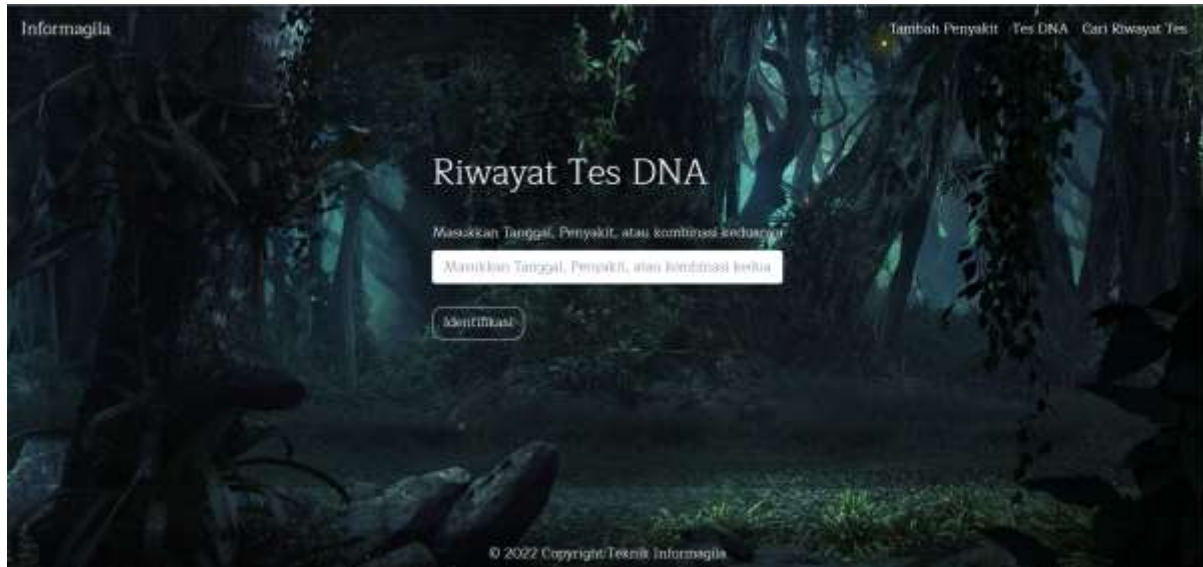
Pengguna dapat melakukan pengetesan apakah DNA pada penyakit yang pengguna masukan sama dengan DNA pada basis data. Terdapat empat buah input, yaitu nama pengguna yang akan di tes, untai DNA, nama penyakit, dan jenis algoritma pencarian. Jika pengguna menekan identifikasi, website akan melakukan pencocokan antara DNA penyakit dan nama penyakit dengan data yang ada di basis data dan akan mengeluarkan hasil seperti berikut.



The screenshot shows the same "Tes DNA" form, but now with filled input fields: "Masukkan Nama Pengguna" contains "Testing", "Masukkan DNA" has a "Pilih File" button and "test.txt", and "Masukkan Nama Penyakit" contains "Polio". The "Identifikasi" button is still present. Below the form, the text "Hasil:" is displayed, followed by a line of data: "2022-04-27 - Testing - Polio - True - 0.8".

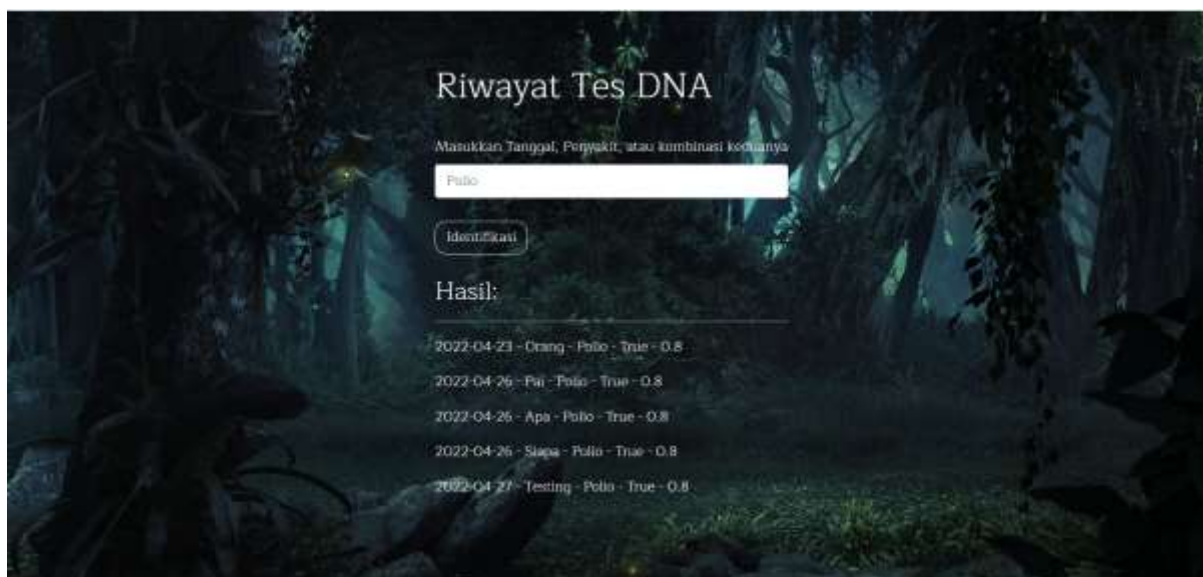
Gambar 4.2.4. Hasil mengetes DNA

Pada hasil pengetesan, secara berturut-turut dari kiri, ditampilkan tanggal dilakukannya pengetesan, nama pengguna, nama penyakit, hasil apakah DNA yang dimasukkan cocok atau tidak dengan nama penyakit dan untai DNA di basis data, dan tingkat kemiripannya. Jika memiliki tingkat kemiripan lebih dari atau sama dengan 80% atau 0.8, dianggap bahwa terdapat kecocokan antara input pengguna dengan data yang ada di basis data.



Gambar 4.2.5. Riwayat Tes DNA

Pada halaman pencarian riwayat tes DNA, terdapat satu kolom input. Pengguna dapat memasukkan kata kunci berupa tanggal, nama penyakit, ataupun kombinasi keduanya. Kemudian, pada website, akan ditampilkan hasil pencarian sesuai dengan masukkan pengguna.





#### Gambar 4.2.6. Hasil riwayat Tes DNA

Pada hasil riwayat tes DNA, ditampilkan hasil secara berturut-turut dari kiri, tanggal dilakukannya pengetesan, nama pengguna, nama penyakit, hasil apakah DNA yang dimasukkan cocok atau tidak dengan nama penyakit dan untai DNA di basis data, dan tingkat kemiripannya. Hasil yang ditampilkan disesuaikan dengan masukkan pengguna.

### 4.3. Hasil Pengujian

Untuk melakukan pengujian, digunakan data nama penyakit sebagai “Testing” dan untai DNA sebagai berikut.

```
CCGAAGGCCTTGTCTGGAACTTCTCTACAGAGGGTCAGACACCGCTCTACGTAAGACG
ACTCCTGACGTAGTCAGAAGACGCGTCGCCAGATGGTTGGGTCATGGTCCGATGCTGTGC
CGTCTCCTATGGAGGGAGTACGGTCGACCATAGTCACGATACTGTATTTTCGACAGCGGA
GTTCGAGTCAGCAAGTGTGTTTTATGAAACCCGTCCTTAGTCATGGTCTCGAATAAGGCG
GATACCGGCCATATGGGTTCAGAGCGGTGTCTGGGAACAGAGACTGTATCCCTTGAGTC
ATTCAGGTATATGATAGGGCGATCACAAAGTTATTAGTTAAAACCCACACTTTATGATTA
TGACTGCTCCACACACGCCAGACGACGACTTGGAAGATAGAAAAACATCACCCAGAGTT
CACTAAGCGTCGCTTAGAATTAAAGATGTTATGAAACCTCGTTTTGGTACGAGCGGGCCC
CGTTGCATGAGCACTCAAATATGCAGGTAATAAGCCAGATGTCTAAGGGTTCGAAATGG
CAGCTCGTGGGCACACACCATCTAAGGGTTCATTCCTTGGTAGACATATTGAGGGATACG
CTTCTGGTCTAGTCCGACGGGGACAAGCCGAGGCTACACTGCCTAGTTCAGCTGAGGTAA
AGCCGGACCGTTACAGGCGGAATCGCCCGTTCCGCGGACATGCAAGCTAATCAATGTGA
CATTGAGGTCTCAAGTACAGGACTCCGTAGGTATGGTCCCCTTTAATGTTTCGTTTAGCAT
CCAAATTGAACGACTTACTTACACAACGGGCGTTCATACGCTCATGGTAAGATGTTTAGT
CGGCGCGTTTCGTCCACGGTTGGTGGGGCCTATAGTGCTCGCTGTCTTTACTGCCTTGGTG
CGGAGCCGGTCTGTCACTGCCTGTGATTCTTTTCATCGCCGAAATGTGAAAAGTTATCAG
TAGTCAGGCCAGTCTTAAATGTGACAACCTAGTTAAGTACGA
```

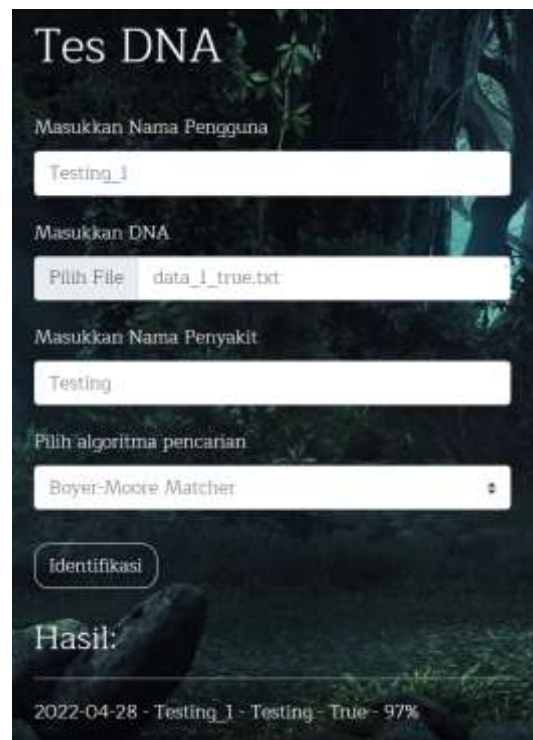
#### 4.3.1. Booyer-Moore Matcher

Input :

```
CCGAAGGCCTTGTCTGGAACTTCTCTACAGAGGGTCAGACACCGCTCTACGTAAGACG
ACTCCTGACGTAGTCAGAAGACGCGTCGCCAGATGGTTGGGTCATGGTCCGATGCTGTGC
CGTCTCCTATGGAGGGAGTACGGTCGACCATAGTCACGATACTGTATTTTCGACAGCGGA
GTTCGAGTCAGCAAGTGTGTTTTATGAAACCCGTCCTTAGTCATGGTCTCGAATAAGGCG
GATACCGGCCATATGGGTTCAGAGCGGTGTCTGGGAACAGAGACTGTATCCCTTGAGTC
ATTCAGGTATATGATAGGGCGATCACAAAGTTATTAGTTAAAACCCACACTTTATGATTA
TGACTGCTCCACACACGCCAGACGACGACTTGGAAGATAGAAAAACATCACCCAGAGTT
CACTAAGCGTCGCTTAGAATTAAAGATGTTATGAAACCTCGTTTTGGTACGAGCGGGCCC
CGTTGCATGAGCACTCAAATATGCAGGTAATAAGCCAGATGTCTAAGGGTTCGAAATGG
CAGCTCGTGGGCACACACCATCTAAGGGTTCATTCCTTGGTAGACATATTGAGGGATACG
CTTCTGGTCTAGTCCGACGGGGACAAGCCGAGGCTACACTGCCTAGTTCAGCTGAGGTAA
AGCCGGACCGTTACAGGCGGAATCGCCCGTTCCGCGGACATGCAAGCTAATCAATGTGA
CATTGAGGTCTCAAGTACAGGACTCCGTAGGTATGGTCCCCTTTAATGTTTCGTTTAGCAT
CCAAATTGAACGACTTACTTACACAACGGGCGTTCATACGCTCATGGTAAGATGTTTAGT
```

CGGCGCGTTTCGTCCACGGTTGGTGGGGCCTATAGTGCTCGCTGTCTTTACTGCCTTGGTG  
CGGAGCCGGTCTGTCACTGCCTGTCGATTCTTTCATCGCCGAAATGTGAAAAGTTATCAG  
TAGTCAGGCCAGTCT

Output :



#### 4.3.2. Data\_2\_true KMP Matcher

Input :

CTGACGTAGTCAGAAGACGCGTCGCCAGATGGTTGGGTCATGGTCCGATGCTGTGCCGTC  
TCCTATGGAGGGAGTACGGTCGACCATAGTCACGATACTGTATTTTCGACAGCGGAGTTC  
GAGTCAGCAAGTGTGTTTTATGAAACCCGTCCTTAGTCATGGTCTCGAATAAGGCGGATA  
CCGGCCATATGGGTTTCAGAGCGGTGTCTGGGAACAGAGACTGTATCCCTTGAGTCATTC  
AGGTATATGATAGGGCGATCACAAAGTTATTAGTTAAAACCCACACTTTATGATTATGAC  
TGCTCCACACACGCCAGACGACGACTTGGAAGATAGAAAAACATCACCCAGAGTTCACT  
AAGCGTCGCTTAGAATTAAAGATGTTATGAAACCTCGTTTTTGGTACGAGCGGGCCCCGTT  
GCATGAGCACTCAAATATGCAGGTACTAAGCCAGATGTCTAAGGGTTCGAAATGGCAGC  
TCGTGGGCACACACCATCTAAGGGTTCATTCCTTGGTAGACATATTGAGGGATACGCTTC  
TGGTCTAGTCCGACGGGGACAAGCCGAGGCTACACTGCCTAGTTCAGCTGAGGTAAAGC  
CGGACCGTTACAGGCGGAATCGCCCGTTCCGCGGACATGCAAGCTAATCAATGTGACAT  
TGAGGTCTCAAGTACAGGACTCCGTAGGTATGGTCCCCTTTAATGTTTCGTTTAGCATCC  
AAATTGAACGACTTACTTACACAACGGGCGTTCATACGCTCATGGTAAGATGTTTAGTCG  
GCGCGTTTCGTCCACGGTTGGTGGGGCCTATAGTGCTCGCTGTCTTTACTGCCTTGGTGCG  
GAGCCGGTCTGTCACTGCCTGTCGATTCTTTCATCGCCGAAATGTGAAAAGTTATCAGTA  
GTCAGGCCAGTCTTAAATGTGACAAGTAAAGTACGACCGAAGGCCTTGTCTGGAAA  
CTTCTCTACAGAGGGTCAGACACCGCTCTACGTAAGACGACTC

Output :

**Tes DNA**

Masukkan Nama Pengguna  
Testing\_2

Masukkan DNA  
Pilih File: data\_2\_true.txt

Masukkan Nama Penyakit  
Testing

Pilih algoritma pencarian  
KMP Matcher

Identifikasi

**Hasil:**

2022-04-28 - Testing\_2 - Testing - True - 93%

#### 4.3.3. Data\_3\_false Brute Force

CCGAAGGCCTTGTCTGGAACTTCTCTACAGAGGGTCAGACACCGCTCTACGTAAGACG  
ACTCCTGACGTAGTCAGAAGACGCGTCGCCAGATGGTTGGGTCATGGTCCGATGCTGTGC  
CGTCTCCTATGGAGGGAGTACGGTCGACCATAGTCACGATACTGTATTTTCGACAGCGGA  
GTTCGAGTCCGAAGGCCTTGTCTGGAACTTCTCTACAGAGGGTCAGACACCGCTCTACG  
TAAGACGACTCCTGACGTAGTCAGAAGACGCGTCGCCAGATGGTTGGGTCATGGTCCGA  
TGCTGTGCCGTCTCCTATGGAGGGAGTACGGTCGACCATAGTCACGATACTGTATTTTCG  
ACAGCGGAGTTCGAGTCCGAAGGCCTTGTCTGGAACTTCTCTACAGAGGGTCAGACAC  
CGCTCTACGTAAGACGACTCCTGACGTAGTCAGAAGACGCGTCGCCAGATGGTTGGGTC  
ATGGTCCGATGCTGTGCCGTCTCCTATGGAGGGAGTACGGTCGACCATAGTCACGATACT  
GTATTTTCGACAGCGGAGTTCGAGTCCGAAGGCCTTGTCTGGAACTTCTCTACAGAGGG  
TCAGACACCGCTCTACGTAAGACGACTCCTGACGTAGTCAGAAGACGCGTCGCCAGATG  
GTTGGGTCATGGTCCGATGCTGTGCCGTCTCCTATGGAGGGAGTACGGTCGACCATAGTC  
ACGATACTGTATTTTCGACAGCGGAGTTCGAGTCCGAAGGCCTTGTCTGGAACTTCTCT  
ACAGAGGGTCAGACACCGCTCTACGTAAGACGACTCCTGACGTAGTCAGAAGACGCGTC  
GCCAGATGGTTGGGTCATGGTCCGATGCTGTGCCGTCTCCTATGGAGGGAGTACGGTCGA  
CCATAGTCACGATACTGTATTTTCGACAGCGGAGTTCGAGTCCGAAGGCCTTGTCTGGAA  
ACTTCTCTACAGAGGGTCAGACACCGCTCTACGTAAGACGACTCCTGACGTAGTCAGAA  
GACGCGTCGCCAGATGGTTGGGTCATGGTCCGATGCTGTGCCGTCTCCTATGGAGGGAGT  
ACGGTCGACCATAGTCACGATACTGTATTTTCGACAGCGGAGTTCGAGTCCGAAGGCCTT  
GTCTGGAACTTCTCTACAGAGGGTCAGACACCGCTCTACGTAAGACGACTCCTGACGTA



GTCAGAAGACGCGTCGCCAGATGGTTGGGTCATGGTCCGATGCTGTGCCGTCTCCTATGG  
AGGGAGTACGGTCGACCATAGTCACGATACTGTATTTTCGACAGCGGAGTTCGAGT

Output :

Tes DNA

Masukkan Nama Pengguna

Testing\_3

Masukkan DNA

Pilih File: data\_9\_false.txt

Masukkan Nama Penyakit

Testing

Pilih algoritma pencarian

Brute Force Matching

Identifikasi

Hasil:

2022-04-28 - Testing\_3 - Testing - False - 47%

#### 4.3.4. Data\_4\_true Regex Match

Input :

ACGAAGGCCTCCCCCGTCTGGAACTTCTCTACAGAGGGTCAGACACCGCTCTACGTAAG  
ACGACTCCTGACGTAGTCAGAAGACGCGTCGCCAGATGGTTGGGTCATGGTCCGATGCT  
GTGCCGTCTCCTATGGAGGGAGTACGGTCGACCATAGTCACGATACTGTATTTTCGACAG  
CGGAGTTCGAGTCAGCAAGTGTGTTTTATGAAACCCGTCCTTAGTCATGGTCTCGAATAA  
GGCGGATACCGGCCATATGGGTTTCAGAGCGGTGTCCTGGGAACAGAGACTGTATCCCTT  
GAGTCATTTCAGGTATATGATAGGGCGATCACAAAGTTATTAGTTAAAACCCACACTTTAT  
GATTATGACTGCTCCACACACGCCAGACGACGACTTGGAAGATAGAAAAACATCACCCA  
GAGTTCACTAAGCGTCGCTTAGAATTAAAGATGTTATGAAACCTCGTTTTGGTACGAGCG  
GGCCCCGTTGCATGAGCACTCAAATATGCAGGTACTAAGCCAGATGTCTAAGGGTTCGA  
AATGGCAGCTCGTGGGCACACACCATCTAAGGGTTCATTCCTTGGTAGACATATTGAGGG  
ATACGCTTCTGGTCTAGTCCGACGGGGACAAGCCGAGGCTACACTGCCTAGTTCAGCTGA  
GGTAAAGCCGGACCGTTACAGGCGGAATCGCCCGTTCCGCGGACATGCAAGCTAATCAA  
TGTGACATTGAGGTCTCAAGTACAGGACTCCGTAGGTATGGTCCCCTTTAATGTTTCGTTT

AGCATCCAAATTGAACGACTTACTTACACAACGGGCGTTCATACGCTCATGGTAAGATGT  
TTAGTCGGCGCGTTCGTCCCACGGTTGGTGGGGCCTATAGTGCTCGCTGTCTTTACTGCCT  
TGGTGCGGAGCCGGTCTGTCACTGCCTGTCGATTCTTTTCATCGCCGAAATGTGAAAAGTT  
ATCAGTAGTCAGGCCAGTCTTAAATGTGACAACACTAGTTAAGTACGA

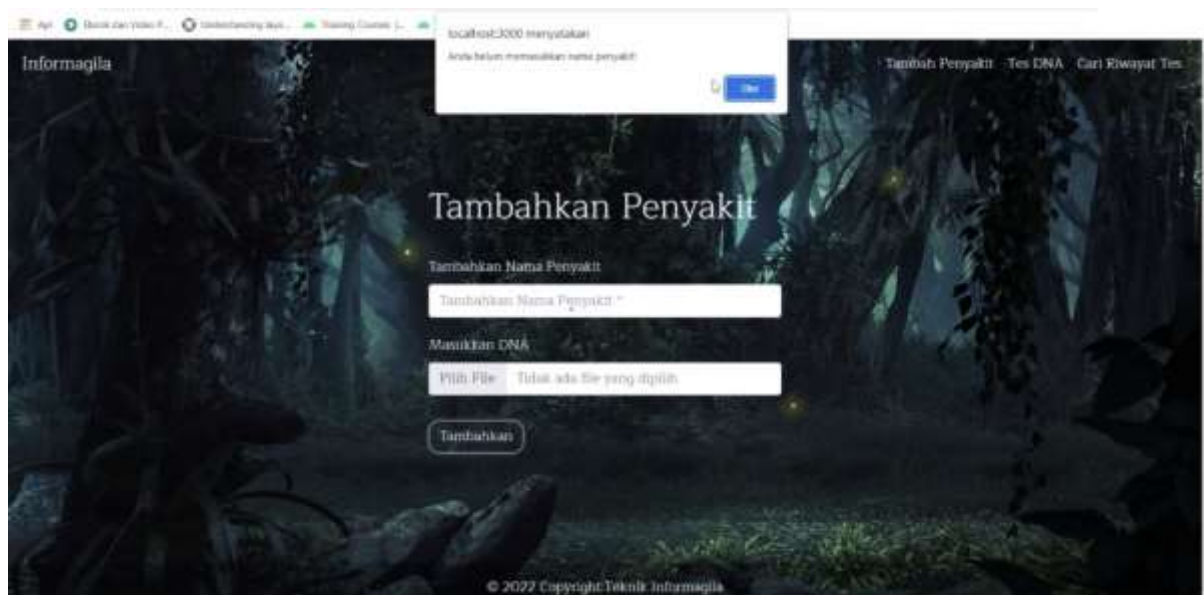
Output :



The screenshot shows the 'Tes DNA' web application interface. It has a dark, forest-themed background. The form includes the following elements:

- Masukkan Nama Pengguna:** A text input field containing 'Testing\_4'.
- Masukkan DNA:** A section with a 'Pilih File' button and a text input field containing 'data\_4\_true.txt'.
- Masukkan Nama Penyakit:** A text input field containing 'Testing\_'.
- Pilih algoritma pencarian:** A dropdown menu with 'Regex Match' selected.
- Identifikasi:** A button to submit the form.
- Hasil:** A section showing the result: '2022-04-28 - Testing\_4 - Testing - True - 99%'.

4.3.5. Tidak memasukkan input apapun ke tambah penyakit

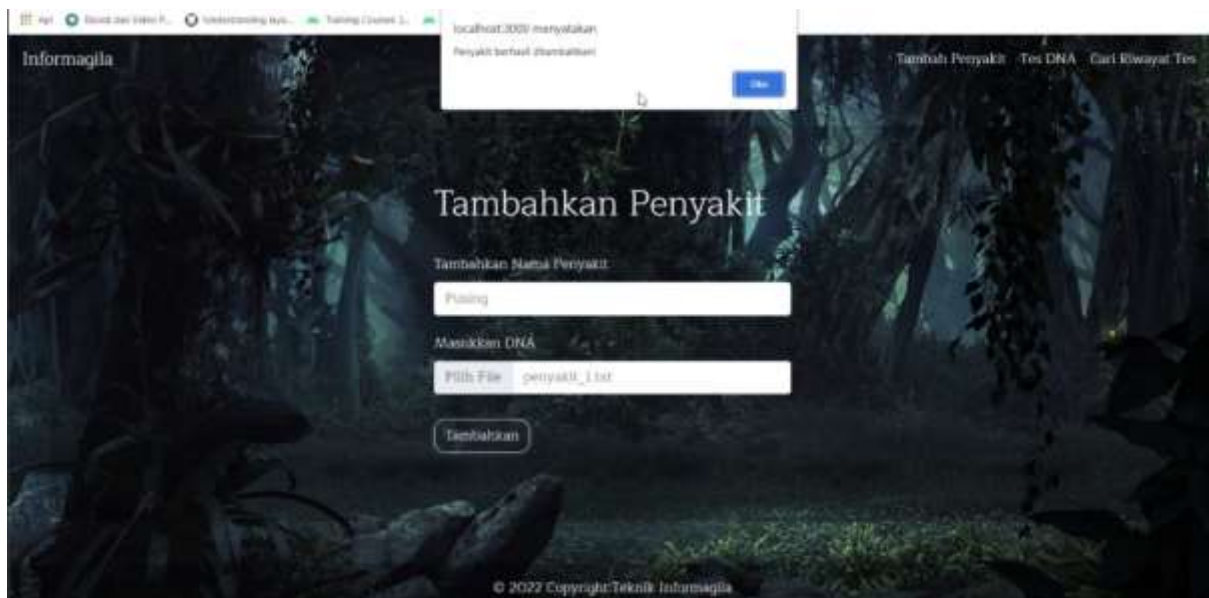


The screenshot shows the 'Tambahkan Penyakit' web application interface. It has a dark, forest-themed background. The form includes the following elements:

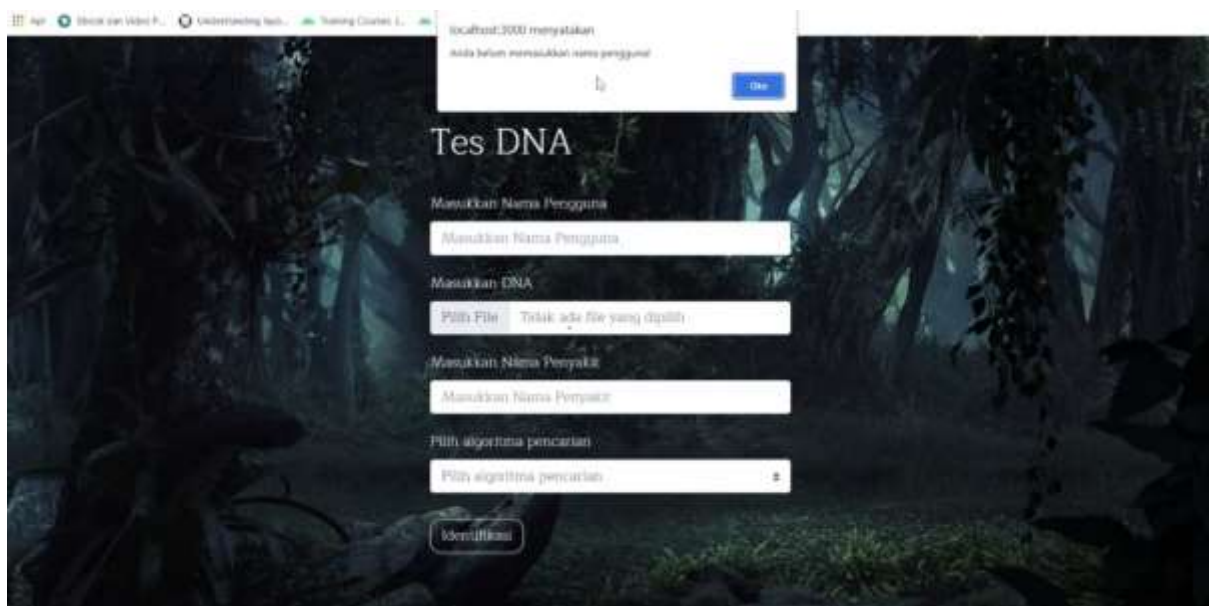
- Tambahkan Nama Penyakit:** A text input field containing 'Tambahkan Nama Penyakit ' '.
- Masukkan DNA:** A section with a 'Pilih File' button and a text input field containing 'Tidak ada file yang dipilih'.
- Tambahkan:** A button to submit the form.

A notification box at the top right says: 'localhost:3000 menyatakan: Anda belum memasukkan nama penyakit!'.

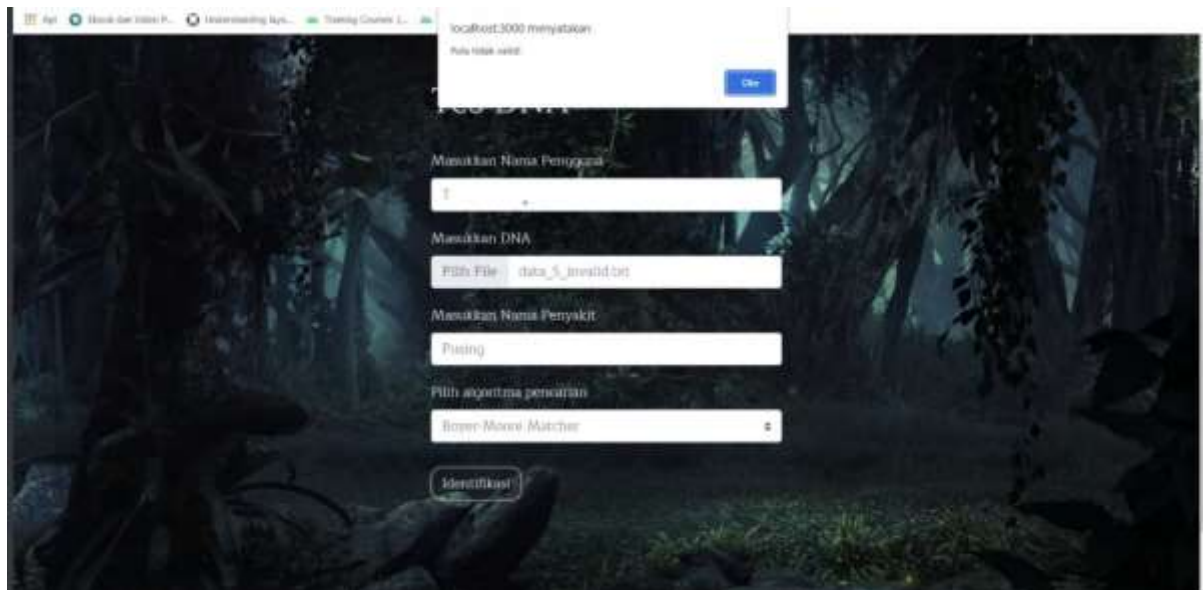
4.3.6. Menambahkan penyakit baru



#### 4.3.7. Tidak memasukkan input apapun pada tes DNA



#### 4.3.8. Memasukkan file dengan pola untai DNA yang tidak valid



#### 4.4. Analisis Hasil Pengujian

Dari hasil pengujian pada bagian sebelumnya, dapat disimpulkan bahwa aplikasi web yang dibuat telah memenuhi spesifikasi fitur yang wajib, yaitu:

1. Berhasil membuat nama penyakit dan sequence DNA masukan pengguna dan memasukkan ke dalam database.
2. Berhasil mengirimkan pesan kesalahan jika masukan tidak valid.
3. Berhasil memprediksi seseorang menderita penyakit tertentu atau tidak (diagnosis) berdasarkan DNA-nya.
4. Berhasil melakukan kueri pencarian dengan menggunakan tanggal dan/atau nama penyakit.
5. Berhasil menghitung tingkat kemiripan DNA dengan threshold kebenaran 80% (artinya, lebih dari sama dengan 80% dianggap tes positif).

## BAB V

### KESIMPULAN DAN SARAN

#### 5.1. Kesimpulan

Algoritma KMP dan BM merupakan algoritma pencocokan string dengan pendekatan yang berbeda. Keduanya merupakan algoritma yang lebih optimal daripada algoritma *brute-force*. Setiap algoritma memiliki performa yang optimal sesuai dengan kasusnya masing-masing, misalnya KMP optimal untuk variasi alfabet yang sedikit, sedangkan BM optimal untuk variasi alfabet yang besar. KMP mencocokkan string dari kiri ke kanan, sedangkan BM mencocokkan string dari kanan ke kiri. Selain KMP dan BM, terdapat juga kakas pencocokan string lain yaitu *regular expression* yang tersedia dalam berbagai macam bahasa pemrograman.

Aplikasi pencocokan string dapat digunakan dalam bidang Biologi, yaitu saat melakukan pencocokan DNA. Simulasi pencocokan ini dapat disajikan dalam bentuk aplikasi web dengan *tech stack* yang beragam.

#### 5.2. Saran

Untuk pengembangan berikutnya, dari sisi algoritma, dapat dicari algoritma pencocokan string lain yang bisa lebih optimal dan efisien daripada KMP atau BM untuk kasus ini. Selain itu, penjalanan algoritma dapat dicoba menggunakan komputasi paralel sehingga pencocokan string dapat lebih cepat dan efisien. Dari sisi implementasi aplikasi web, dapat ditambahkan fitur-fitur baru pada aplikasi web, misalnya visualisasi algoritma ketika melakukan pencocokan dan pencarian berdasarkan nama orang.

**LINK PENTING**

Github Repository: <https://github.com/khelli07/DNA-string-matching>

## DAFTAR PUSTAKA

- Connolly, Thomas and Begg, Carolyn. (2010). *Database Systems A Practical Approach to Design, Implementation, and Management Fifth Edition*. Boston: Pearson Education.
- Hakim, Lukmanul. 2010. *Membangun Web Berbasis PHP dengan Framework Codeigniter*. Yogyakarta : Lokomedia.
- Hidayatullah, Priyanto., Jauhari Khairul Kawistara. 2014. *Pemrograman WEB*. Bandung : Informatika Bandung.
- Janner, Simarmata. 2010. *Rekayasa Perangkat Lunak*. Yogyakarta: Penerbit Andi
- Laudon, K, & J.P. Laudon. (2010). *Management Informtaion System: Managing the Digital Firm, 11th edition*. New Jersey: Prentice Hall.