

**LAPORAN TUGAS BESAR III
IF2211 STRATEGI ALGORITMA**

**PENERAPAN *STRING MATCHING* DAN *REGULAR
EXPRESSION* DALAM DNA *PATTERN MATCHING***



Disusun oleh:

Kelompok -Tania (44)

Grace Claudia	13520078
Patrick Amadeus Irawan	13520109
Nelsen Putra	13520130

**Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung**

2022

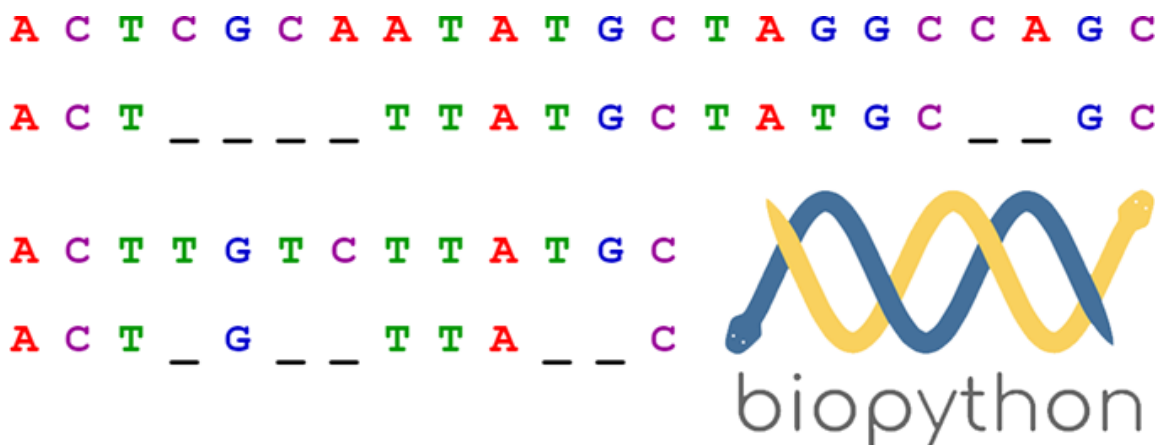
DAFTAR ISI

DAFTAR ISI	<i>i</i>
BAB I	1
BAB II	9
Dasar Teori	9
Penjelasan Singkat Aplikasi Web yang Dibangun	11
BAB III	13
Langkah-Langkah Penyelesaian Masalah Setiap Fitur	13
Fitur Fungsional dan Arsitektur Aplikasi Web	14
BAB IV	18
Spesifikasi Teknis Program	18
Tata Cara Penggunaan Program	23
Hasil Pengujian	27
Analisis Hasil Pengujian	39
BAB V	41
Kesimpulan	41
Saran	42
Refleksi	44
LAMPIRAN	46
DAFTAR PUSTAKA	47

BAB I

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

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: ATTCGTAAGTAAAGTTA. Teknik

pattern matching memegang peranan penting untuk dapat menganalisis sekuens DNA yang sangat panjang dalam waktu singkat. Oleh karena itu, mahasiswa Teknik Informatika 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 admin dalam melakukan *filtering* dan pencarian.

Dalam tugas besar ini, setiap kelompok 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:

Tambahkan Penyakit

Nama Penyakit:

Sequence DNA:

Gambar 1.2 Contoh *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:

Tes DNA

Nama Pengguna:

Sequence DNA:

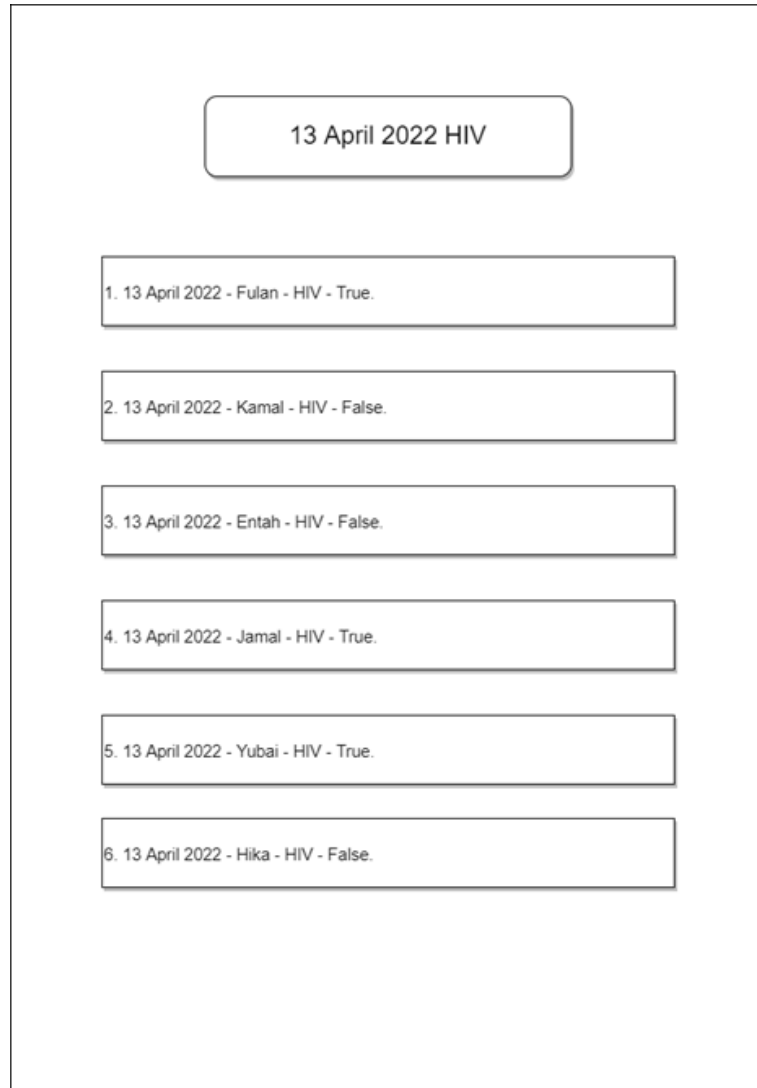
Prediksi Penyakit:

Hasil Tes

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

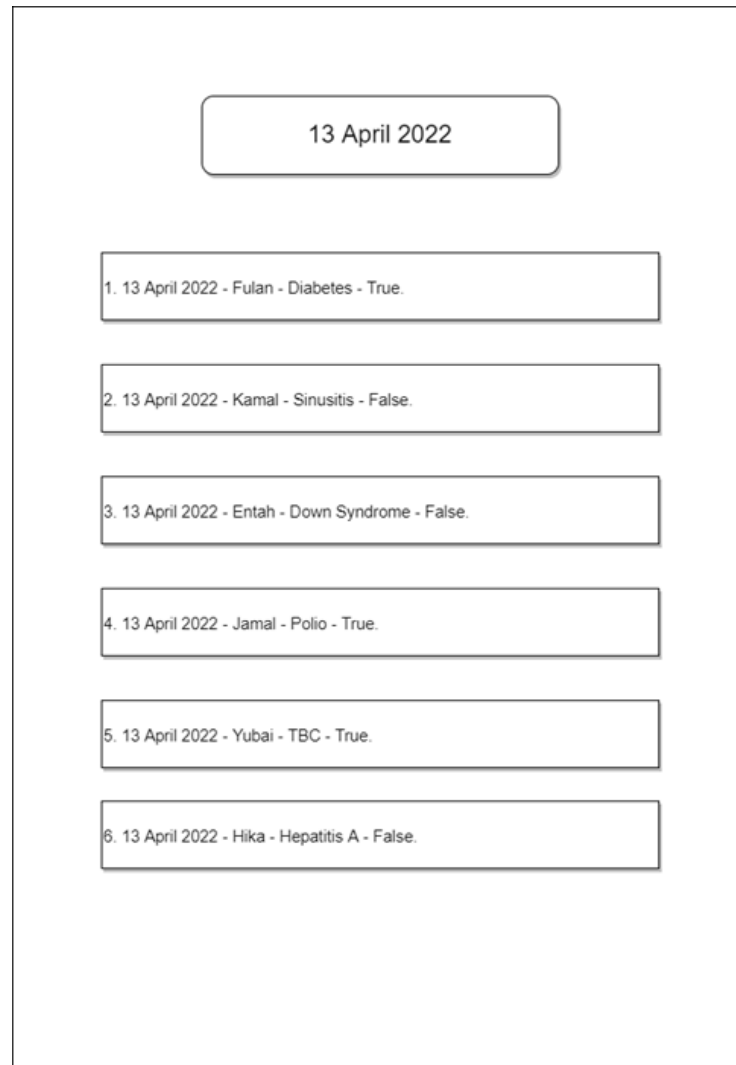
Gambar 1.3 Ilustrasi Prediksi

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: <tanggal_prediksi><spasi><nama_penyakit>, 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



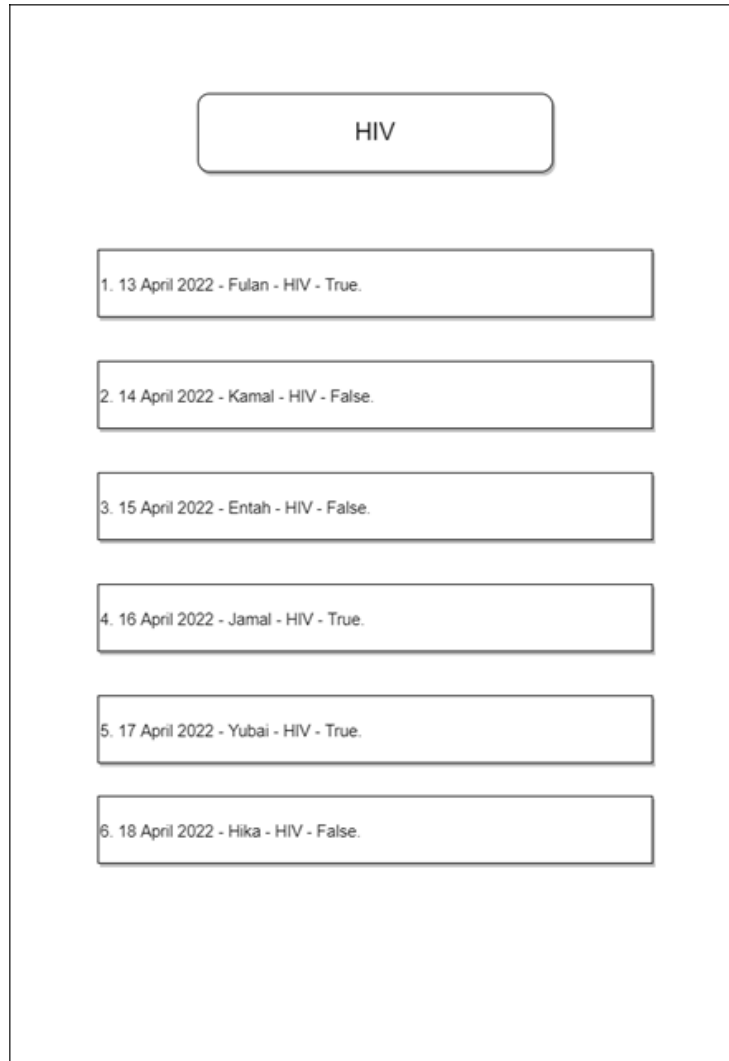
Gambar 1.4 Ilustrasi Interaksi 1

ii. Masukkan hanya tanggal



Gambar 1.5 Ilustrasi Interaksi 2

iii. Masukan hanya nama penyakit



Gambar 1.6 Ilustrasi Interaksi 3

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

The image shows a web application interface titled "Tes DNA". It features three input fields at the top: "Nama Pengguna:" with a placeholder "<pengguna>", "Sequence DNA:" with a placeholder "upload file...", and "Prediksi Penyakit:" with a placeholder "<penyakit>". Below these fields is a green "Submit" button. A horizontal dashed line separates the input section from the output section, which is titled "Hasil Tes". The output section displays a string: "<Tanggal> - <pengguna> - <penyakit> - <similarity> - <True/False>".

Gambar 1.7 Tampilan Hasil dengan Persentase Kemiripan

BAB II

LANDASAN TEORI

2.1. Dasar Teori

2.1.1. Algoritma Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt (KMP) merupakan salah satu jenis algoritma pencarian pola/pencocokan string dengan pencarian dimulai dari kiri ke kanan (seperti pada algoritma *Brute Force*). Akan tetapi, algoritma ini bekerja lebih cerdas daripada algoritma *Brute Force*. Algoritma ini merupakan jenis *Exact String Matching Algorithm* yang merupakan pencocokan string secara tepat dengan susunan karakter dalam string yang dicocokkan memiliki jumlah maupun urutan karakter dalam string yang sama. Contohnya, kata 'algoritmik' akan menunjukkan kecocokan hanya dengan kata 'algoritmik'. Pada algoritma KMP, kita menyimpan informasi yang digunakan untuk melakukan pergeseran lebih jauh, tidak hanya satu karakter seperti algoritma *Brute Force*. Terdapat beberapa definisi pada algoritma KMP:

1. Misalkan A adalah alfabet dan $x = x_1x_2\dots x_k$ adalah string yang panjangnya k yang dibentuk dari karakter-karakter di dalam alfabet A.
 - Awalan (*prefix*) dari x adalah upa-string (*substring*) u dengan $u = x_1x_2\dots x_{k-1}$, $k \in \{1, 2, \dots, k-1\}$ dengan kata lain, x diawali dengan u.
 - Akhiran (*suffix*) dari x adalah upa-string (*substring*) u dengan $u = x_k - b \dots x_k$, $k \in \{1, 2, \dots, k-1\}$ dengan kata lain, x diakhiri dengan v.
 - Pinggiran (*border*) dari x adalah upa-string r sedemikian sehingga $r = x_1x_2\dots x_{k-1}$ dan $u = x_k - b \dots x_k - b + 1 \dots x_k$, $k \in \{1, 2, \dots, k-1\}$, dengan kata lain, pinggiran dari x adalah upa-string yang keduanya awalan dan juga akhiran sebenarnya dari x.
2. Fungsi Pinggiran $b(j)$ didefinisikan sebagai ukuran awalan terpanjang dari P yang merupakan akhiran dari $P[1..j]$

2.1.2. Algoritma Boyer-Moore

Algoritma Boyer Moore (BM) merupakan suatu solusi pencarian yang efisien dapat melakukan perbandingan *pattern* mulai dari kanan ke kiri. Jika terjadi ketidakcocokan string dari kanan *pattern* maka ketidakcocokan akan membantu kita untuk menggerakkan *pattern* tersebut dengan jarak yang lebih jauh. Gerakan melompat ini akan memberikan informasi berapa banyak *pattern* harus digeser untuk mencocokkan karakter terakhir yang cocok dengan kemunculan awal *pattern*. Artinya, akan lebih signifikan dalam mengurangi proses perbandingan, jika kita bisa melompati atau tidak melakukan perbandingan karakter yang diprediksi akan gagal. Algoritma Boyer Moore mempunyai keunggulan dalam waktu menemukan *pattern* yang akan dicari dalam ukuran *file* yang lebih besar. Pada algoritma ini pencocokan kata dimulai dari karakter terakhir kata kunci menuju karakter awalnya. Jika terjadi perbedaan antara karakter terakhir kata kunci dengan kata yang dicocokkan maka karakter-karakter dalam potongan kata yang dicocokkan tadi akan diperiksa satu per satu. Hal ini dimaksudkan untuk mendeteksi apakah ada karakter dalam potongan kata tersebut yang sama dengan karakter yang ada pada kata kunci. Apabila terdapat kesamaan, maka kata kunci akan digeser sedemikian rupa sehingga posisi karakter yang sama terletak sejajar, dan kemudian dilakukan kembali pencocokan karakter terakhir dari kata kunci. Sebaliknya jika tidak terdapat kesamaan karakter, maka seluruh karakter kata kunci akan bergeser ke kanan sebanyak m karakter, di mana m adalah panjang karakter dari kata kunci. Algoritma Pattern Matching Boyer-Moore ini berbasis pada 2 metode, yaitu:

1. The Looking-Glass Technique

The Looking-Glass Technique melakukan perbandingan suatu karakter akhir pada kata w dengan suatu karakter pada teks s . Jika karakter tersebut sama maka jendela karakter akan berjalan mundur pada kedua string dan mengecek kembali kedua karakter. Mencari suatu kecocokan string pada teks dengan pola yang akan dicari dengan cara memindahkan atau menggesernya sampai teks string selesai.

2. The Character-Jump Technique

Character-Jump Technique melakukan suatu aksi ketika perbandingan antara dua karakter yang berbeda. Ada dua aksi yang tergantung pada teks s dan kata w yang dimiliki; jika p yaitu karakter pada s yang sedang diproses yang tidak cocok maka

ada dua kemungkinan aksi. Mencari karakter yang sesuai dan cara penggeseran sebuah karakter perbandingan terakhir.

2.1.3. Regular Expression

Regular expression merupakan notasi yang digunakan untuk mendeskripsikan himpunan karakter string. *Regular expression* didefinisikan berdasarkan aturan teori bahasa formal. *Regular expression* terdiri atas konstanta dan operator yang menunjukkan himpunan-himpunan string dan operasi antar himpunan string tersebut secara berurutan. Konstanta yang telah didefinisikan, antara lain:

1. Himpunan kosong, diberi notasi \emptyset .
2. String kosong, diberi notasi ϵ .
3. Karakter, diberi notasi sesuai dengan karakter bahasa yang digunakan.

Di samping itu, operator yang telah didefinisikan, antara lain:

1. Konkatenasi, misal $\{"ab", "c"\} \{"d", "ef"\} = \{"abd", "abef", "cd", "cef"\}$.
2. Alternasi, misal $\{"ab", "c"\} \mid \{"ab", "d", "ef"\} = \{"ab", "c", "d", "ef"\}$.
3. *Kleene star*, menunjukkan semua himpunan yang dapat dibuat dengan melakukan konkatenasi 0 atau lebih banyak string dari string yang dilakukan operasi ini. Misal $\{"ab", "c"\}^* = \{\epsilon, "ab", "c", "abab", "abc", "cab", "cc", "ababab", "abcab", \dots\}$. Untuk mengurangi jumlah tanda kurung, diasumsikan *Kleene Star* memiliki prioritas terbesar dilanjutkan dengan operasi konkatenasi baru operasi alternasi.

2.2. Penjelasan Singkat Aplikasi Web yang Dibangun

Aplikasi yang dibangun merupakan aplikasi berbasis web berupa *DNA-matcher* yang mampu menganalisis DNA yang dimasukkan oleh pengguna untuk memprediksi suatu penyakit genetik tertentu. Selain itu, aplikasi juga dapat menerima daftar penyakit baru dengan menerima masukan pola *sequence* DNA yang baru untuk berbagai penyakit yang kemudian disimpan ke dalam suatu *database*. Aplikasi web kami sendiri memiliki tampilan ‘Home’ sebagai *landing page* ketika *website* pertama kali diakses. Selain itu, terdapat beberapa halaman lain seperti ‘Input New Disease’, ‘DNA Prediction Test’, dan ‘Search Test Result’ yang masing-masing memiliki fitur dan kegunaan untuk menambahkan penyakit baru berdasarkan *sequence* DNA, melakukan tes prediksi

penyakit genetik berdasarkan *sequence* DNA, serta mencari hasil tes prediksi DNA yang tersimpan di dalam *database* yang terhubung di aplikasi *website* kami. Adapun informasi yang berhasil disimpan ke dalam *database* tersebut ialah jenis penyakit (*disease*) yang terdiri atas nama penyakit dan rantai DNA penyusun, serta hasil tes prediksi (*user_record*) yang terdiri atas tanggal tes, nama pasien, penyakit prediksi, status terprediksi, serta tingkat kemiripan.

Aplikasi dibuat dengan memanfaatkan beberapa bahasa pemrograman sekaligus, yakni Golang, JavaScript, CSS, dan HTML. Golang sendiri digunakan secara signifikan untuk mengimplementasikan algoritma pencocokan string yang disisipkan di bagian *backend* program. Di samping itu, JavaScript, CSS, dan HTML digunakan dalam pembuatan *frontend* dari aplikasi web yang dibangun dengan memanfaatkan *React framework*. Agar dapat diakses *online*, web ini juga di-*deploy* dengan *deployment platform* heroku.

BAB III

ANALISIS PEMECAHAN MASALAH

3.1. Langkah-Langkah Penyelesaian Masalah Setiap Fitur

3.1.1. Fitur *String Matching* dengan Algoritma KMP

- Tentukan fungsi pinggiran dari *pattern* yang hendak dicari, ditentukan dengan mencari panjang *substring prefix* dan *suffix* terpanjang yang sama.
- Untuk setiap *adjacent substring* pada word yang hendak dicari patternnya, digunakan algoritma berikut.
- Cari indeks terakhir dimana *substring* di *word* sama dengan *substring pattern*.
- Apabila indeks sama dengan panjang *pattern* yang hendak dicari, kembalikan **true**.
- Apabila tidak, gunakan fungsi pinggiran dari *pattern* yang ditentukan di awal untuk menentukan mekanisme pergeseran karakter.
- Ulangi untuk substring selanjutnya, apabila hingga akhir tidak ditemukan kesamaan *pattern*, kembalikan **false**.

3.1.2. Fitur *String Matching* dengan Boyer Moore

- Tentukan fungsi indeks tempat kemunculan terakhir setiap karakter di *word* pada karakter di *pattern*.
- Untuk setiap *adjacent substring* pada *word* yang hendak dicari *pattern*-nya, digunakan algoritma berikut.
- Cari indeks ter kiri dimana terjadi ketidaksamaan karakter antara *substring word* yang sedang dicek dengan *pattern*.
- Apabila indeks -1, yang berarti “tidak ada” indeks ter kiri yang berbeda, kembalikan **true**.
- Apabila tidak, gunakan fungsi indeks kemunculan terakhir dan bandingkan nilai kembalian fungsi sesuai dengan kasus yang terbagi menjadi 3, yakni:

- a. Apabila kembalian = -1, dimana karakter *word* tidak ditemukan di karakter *pattern*, indeks maju sebanyak panjang *pattern* - 1.
- b. Apabila kembalian indeks fungsi lebih kecil dari kemunculan indeks terkiri di *pattern*, indeks maju sebanyak indeks terkiri - kembalian indeks fungsi.
- c. Apabila kembalian indeks fungsi lebih besar dari kemunculan indeks terkiri di *pattern*, indeks maju sebanyak 1.

3.1.3. Fitur *Similarity* dengan *Levenshtein & Moving Pattern*

- Buat matriks kosong dengan ukuran panjang *word* x panjang *pattern*.
- Inisialisasi seluruh baris pertama dan kolom pertama pada matriks sesuai dengan indeks yang berkorespondensi.
- Dengan menggunakan *surrounding algorithm* isi block yang kosong dengan syarat sebagai berikut.
 - a. Apabila *block* atas kotak kosong sama dengan *block* kiri kotak kosong, isi kotak kosong dengan $\min(\text{block kiri} + 1, \text{block atas} + 1, \text{block diagonal})$
 - b. Apabila *block* atas kotak kosong berbeda dengan *block* kiri kotak kosong, isi kotak kosong dengan $\min(\text{block kiri}, \text{block atas}, \text{block diagonal}) + 1$
- Setelah semua kotak terisi, bandingkan nilai dengan angka pada pojok kanan bawah matriks.
- Untuk setiap substring pada *word* dengan panjang *pattern*, lakukan algoritma di atas dan kembalikan *similarity* dengan nilai maksimal.

3.2. Fitur Fungsional dan Arsitektur Aplikasi Web

3.2.1. Fitur Fungsional

Aplikasi web *DNA Pattern Matching* yang telah dibuat memiliki berbagai fitur fungsional, antara lain sebagai berikut.

1. Fitur 1: Register bentuk *disease* baru beserta *DNA sequence*-nya.

2. Fitur 2: Register *user record* untuk pengecekan *disease* tertentu yang terdaftar di *database*.
3. Fitur 3: *Search record* pengujian penyakit untuk *user* dengan filter tanggal, nama penyakit, maupun keduanya dengan implementasi *Regular Expression* untuk berbagai format penanggalan.

3.2.2. Arsitektur Aplikasi Web

Aplikasi web *DNA Pattern Matching* dibangun dan dikembangkan dengan arsitektur sebagai berikut.

1. Backend: Golang
2. Frontend: JavaScript, HTML, CSS, React
3. Kakas basis data: MongoDB

Backend dari program dipisah menjadi beberapa *folder*, di antaranya sebagai berikut.

1. *Configs*: untuk melakukan *configuration* dan *connect* ke *database*
2. *Controllers*: untuk mengatur penerimaan *input* dan menjalankan fungsi tertentu
3. *Models*: berisi *struct* perjanjian untuk data yang di-*passing*
4. *Responses*: berisi *struct* hasil pemrosesan yang dilakukan di *backend* untuk diberikan ke *frontend*
5. *Route*: berisi *endpoint* yang di-*expose* untuk digunakan *frontend*

Frontend dari program dipisah menjadi beberapa *folder* dan *file*, di antaranya sebagai berikut.

1. *Assets*: *folder* penyimpanan gambar yang digunakan
2. *Styles*:
 - *App.css*: *styling* untuk *App*
 - *Home.css*: *styling* untuk *Home*
 - *index.css*: *styling* untuk keseluruhan aplikasi
 - *InputDisease.css*: *styling* untuk fitur *Input New Disease*

- SearchResult.css: *styling* untuk fitur *Search Test Result*
 - TestDisease.css: *styling* untuk fitur *Test Disease*
3. App.js: *layer* aplikasi paling bawah, tempat adanya *routing*
 4. Home.js: *home page web* kami
 5. index.js: *rendering* App.js
 6. InputDisease.js: komponen berbasis *page* yang berkaitan dengan fitur *register disease* baru beserta *DNA sequence*-nya.
 7. Navbar.js: komponen *navigation bar* yang berfungsi sebagai petunjuk pergantian *page website*
 8. SearchList.js: komponen berupa daftar hasil pencarian *user record* yang berkaitan dengan fitur pencarian hasil uji penyakit sesuai dengan *filter*.
 9. SearchResult.js: komponen berbasis *page* yang berkaitan dengan fitur pencarian hasil uji penyakit sesuai dengan *filter*
 10. TestDisease.js: komponen untuk fitur *Test Disease*

Untuk melakukan *request* dan menerima *response* dari *frontend* ke *backend* dan sebaliknya, kami menggunakan *library* *axios*. Contoh penggunaannya adalah sebagai berikut.

```
}  
if (state) {  
  axios({  
    method: "post",  
    url: URL + "/disease",  
    data: datas,  
  }).then(function(res) {  
    setModalSucessShow(true);  
    console.log(res);  
  }).catch(function(err) {  
    console.log(err.message);  
  })  
}  
}).catch(err => {  
  console.log(err);  
})  
}
```

Dengan demikian, *frontend* akan melakukan *request method* POST ke URL = "https://dna-matching-be.herokuapp.com/disease" lalu ke `.then` akan mendapatkan *response* jika berhasil atau ke `.catch` jika `err`. Pada *backend*, yang akan diakses adalah sebagai berikut.

```
func DiseaseRoute(e *echo.Echo) {  
    e.POST("/disease", controllers.AddDisease)
```

BAB IV

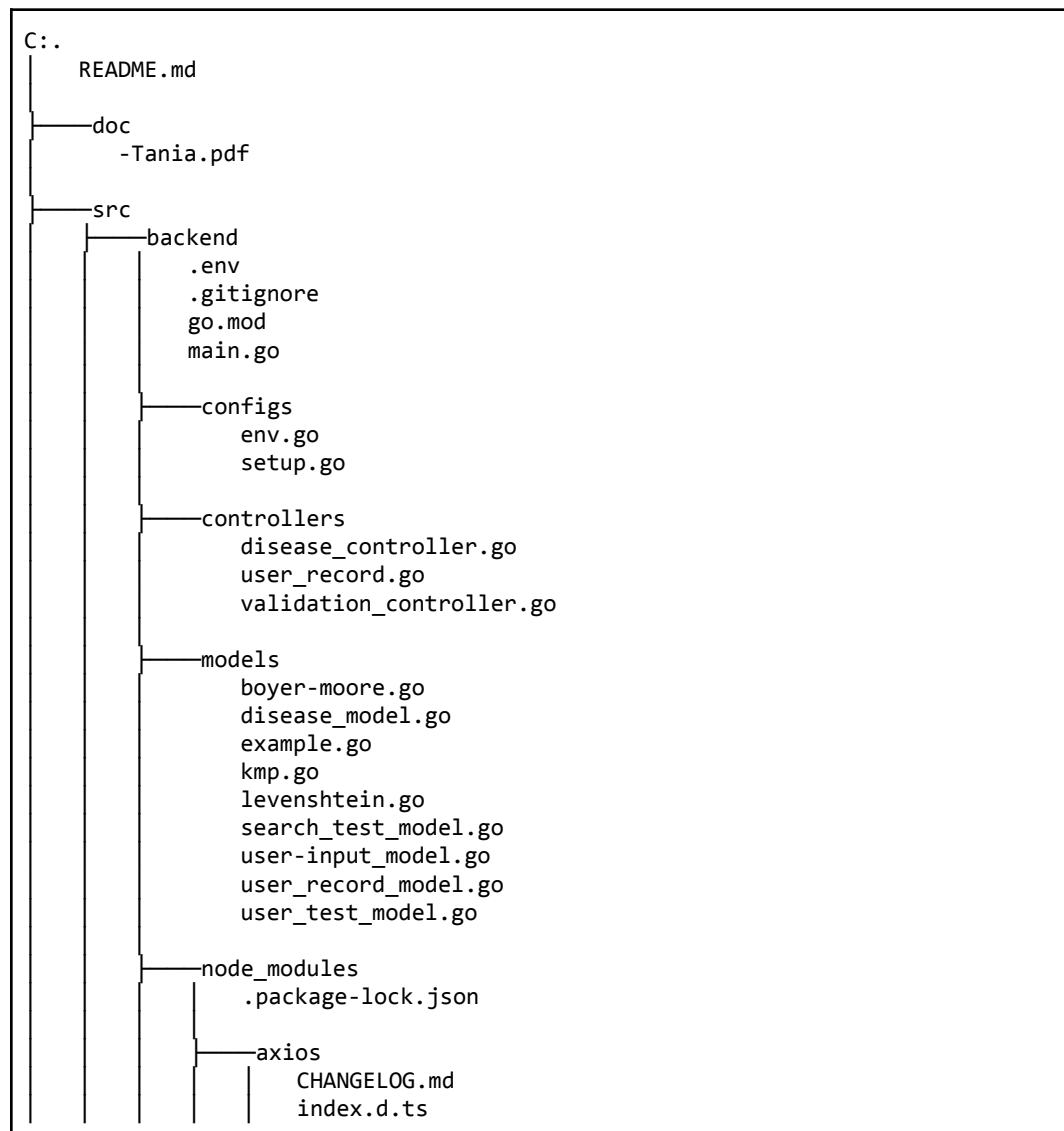
IMPLEMENTASI DAN PENGUJIAN

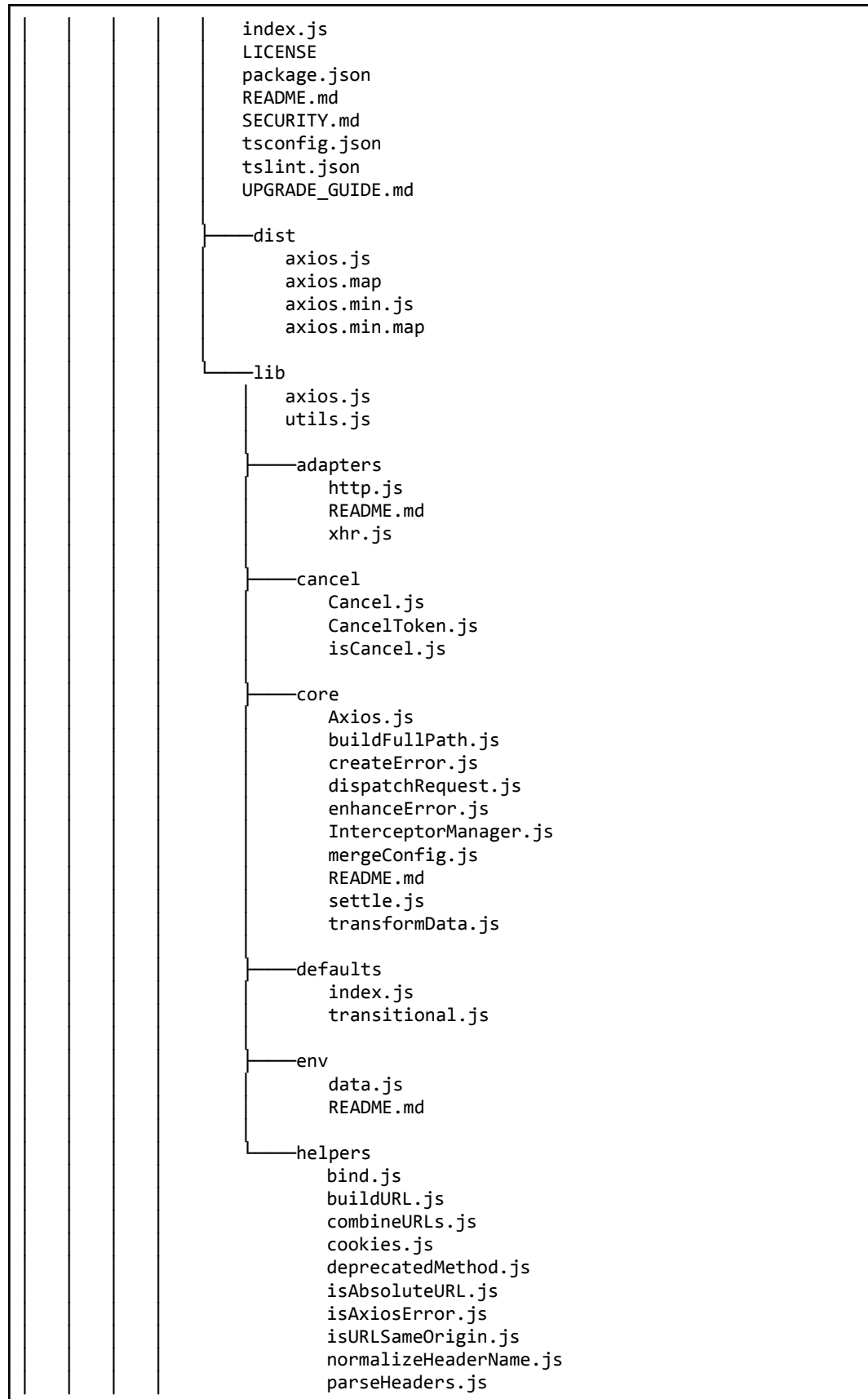
4.1. Spesifikasi Teknis Program

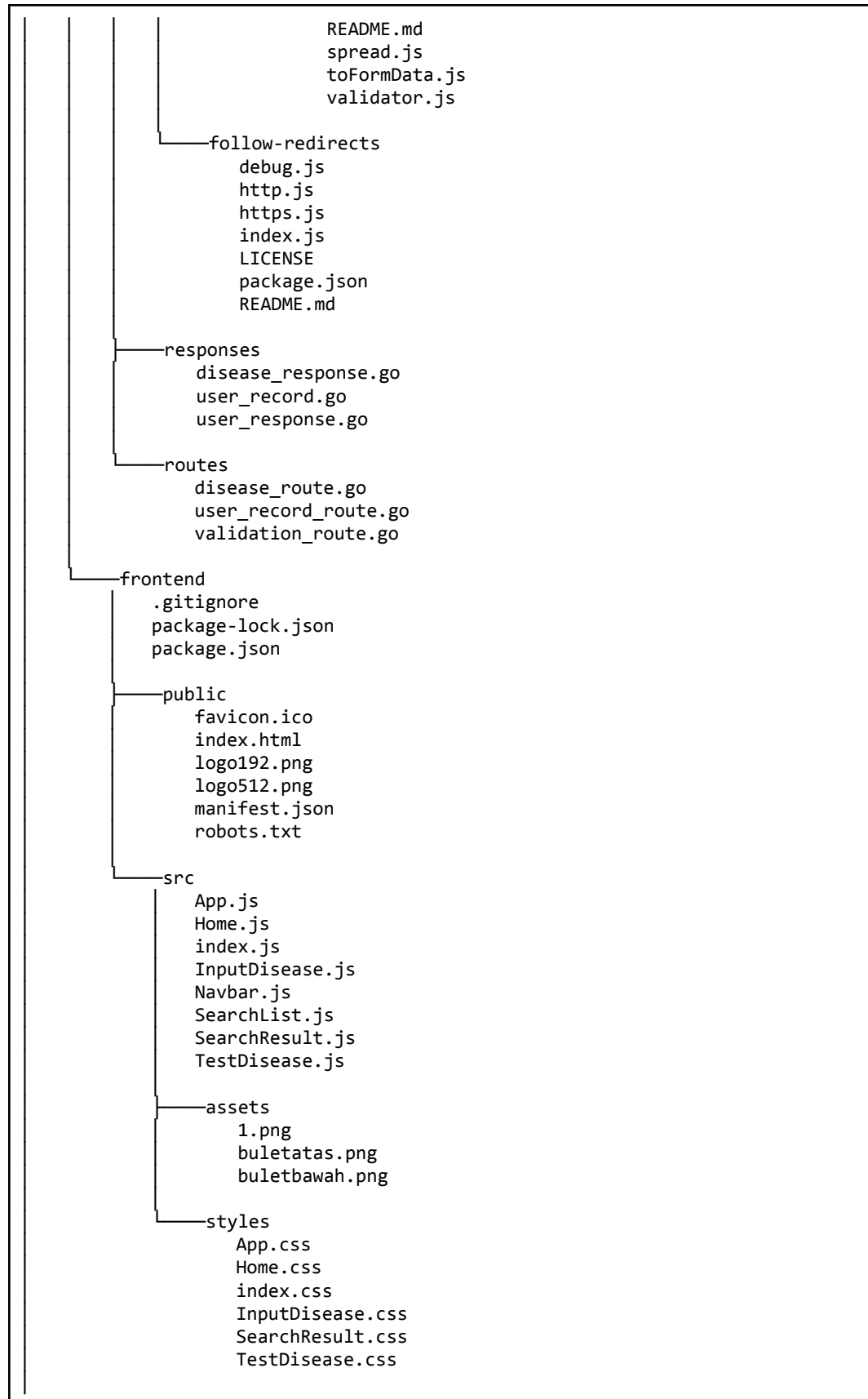
Spesifikasi teknis dari program terdiri atas struktur data, serta fungsi dan prosedur yang berhasil diimplementasikan di dalam program. Pada program aplikasi web DNA *Pattern Matching* buatan kelompok kami, spesifikasi teknisnya adalah sebagai berikut.

4.1.1. Struktur Program

Berikut ini merupakan struktur dari program aplikasi web DNA *Pattern Matching* yang telah dibangun.







```
└──test
    disease1.txt
    disease2.txt
    disease3.txt
    disease_invalid.txt
    person1.txt
    person2.txt
    person3.txt
    person_invalid.txt
```

Program terdiri atas 3 folder utama dan 1 buah README *text file* dengan uraian sebagai berikut.

1. Folder src, berisi *source code* dari program aplikasi web DNA *Pattern Matching* itu sendiri yang terdiri atas kode program untuk tampilan *frontend* dan kode program untuk sisi *backend*.
2. Folder test, berisi beberapa *file .txt* yang digunakan sebagai *test case* pada proses pengujian program aplikasi web DNA *Pattern Matching* yang telah dibangun.
3. Folder doc, berisi *file* laporan tugas besar dalam bentuk pdf.
4. File README, berisi informasi seputar program aplikasi web DNA *Pattern Matching* terkait latar belakang pembuatan program, teknologi dan kakas yang dipakai dalam implementasi, fitur-fitur dalam program, hingga cara menjalankan program.

4.1.2. Struktur Data

Program aplikasi web DNA *Pattern Matching* kami mengimplementasikan stuktur data berupa basis data yang disimpan dengan bantuan kakas MongoDB. Terdapat dua tabel basis data yang diimplementasikan, yakni *disease* dan *user_record*. Selain implementasi basis data, program kami juga memanfaatkan *file .txt* untuk digunakan sebagai *file input* DNA *sequence* penyakit maupun pasien. Implementasi *file .txt* tersebut dapat ditemukan pada folder test dari struktur program aplikasi web DNA *Pattern Matching* kami.

4.1.3. Fungsi dan Prosedur

Berikut ini merupakan fungsi-fungsi dan prosedur yang diimplementasikan dalam program.

1. Fungsi KMP

Fungsi KMP digunakan untuk algoritma string matching dengan metode KMP dan mengembalikan boolean.

2. Fungsi kmp_boundary

Fungsi kmp_boundary digunakan untuk menentukan fungsi pinggiran untuk *pattern matching* KMP *array of integer*.

3. Fungsi BoyerMoore

Fungsi BoyerMoore digunakan untuk algoritma *string matching* dengan metode Boyer-Moore dan mengembalikan boolean.

4. Fungsi value

Fungsi value digunakan untuk menentukan map kemunculan indeks karakter terakhir pada *pattern* dengan kembalian berupa HashMap string dan integer.

5. Fungsi levenshtein

Fungsi levenshtein merupakan fungsi untuk menentukan taraf *similarity* dengan metode levenshtein dengan kembalian berupa float64.

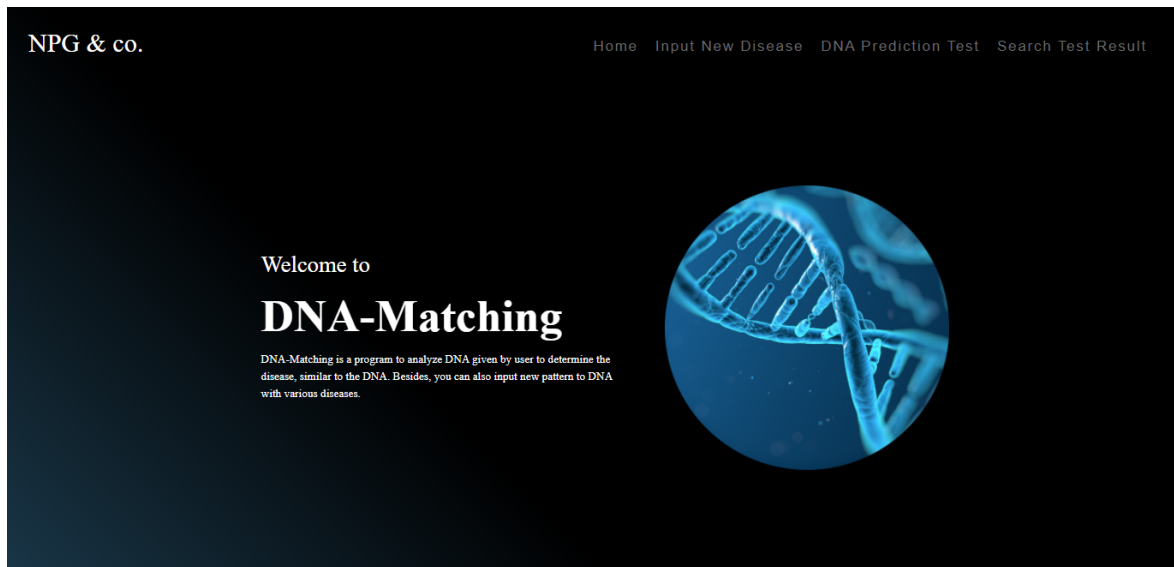
6. Fungsi MovingPatternSimilarity

Fungsi ini merupakan implementasi pergeseran *brute-force* untuk setiap *substring* berimpit guna mencari *similarity* maksimum, mengembalikan nilai float64.

7. Fungsi min

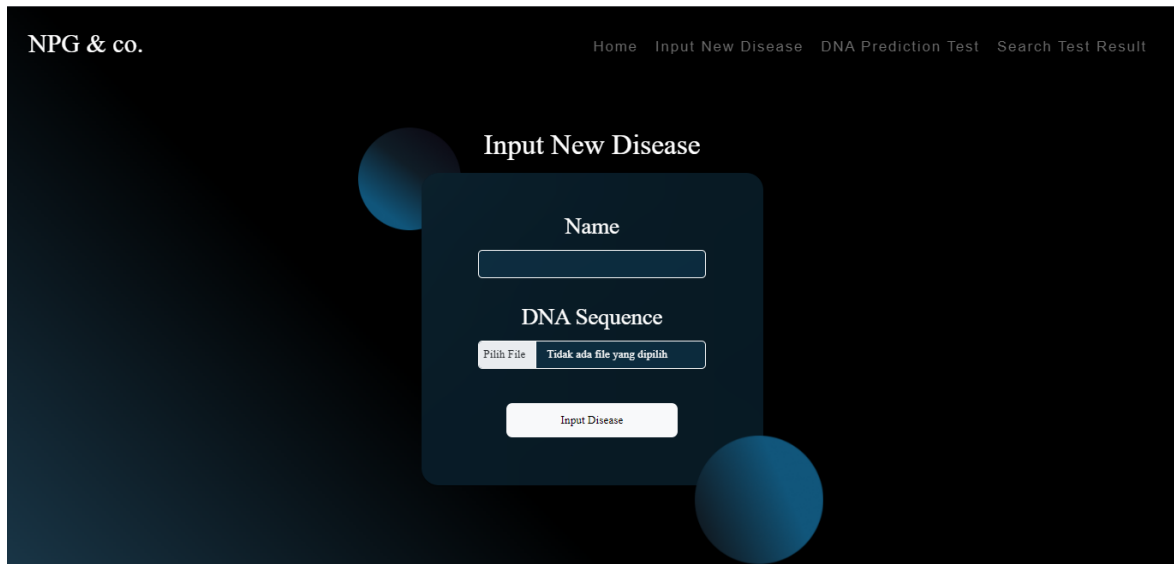
Fungsi untuk menentukan nilai minimum antara dua int / float64 sebagai representasi number, mengembalikan int / float64.

4.2. Tata Cara Penggunaan Program



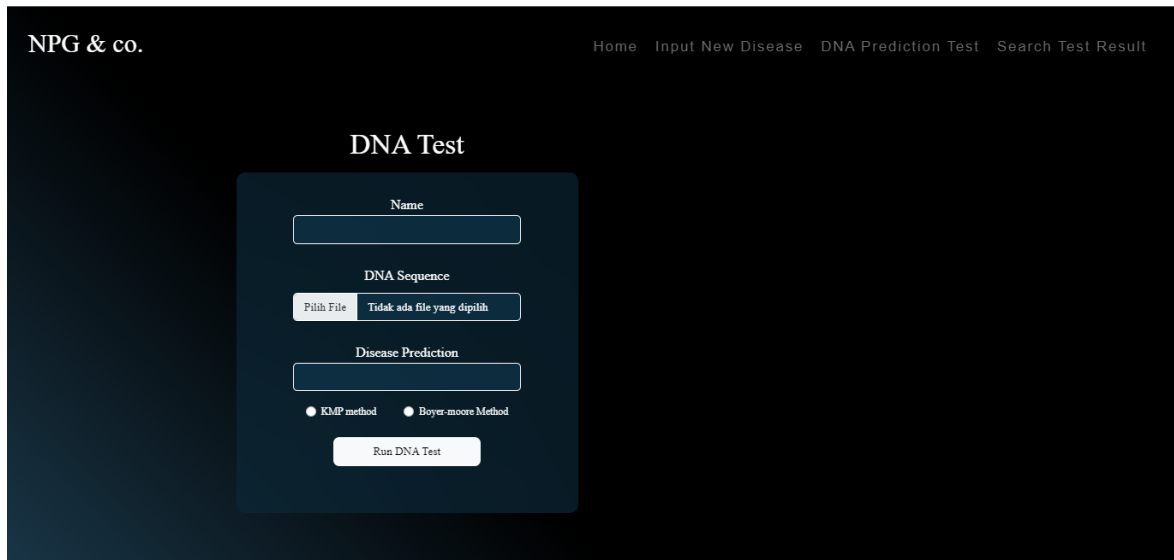
Gambar 4.2.1 *Interface Program: Tampilan Menu Home*

Pada saat program pertama kali dijalankan, tampilan yang muncul akan seperti pada Gambar 4.2.1. *Website* memiliki tampilan halaman '*Home*' yang merupakan *landing page* dari *website* itu sendiri. Pada bagian tengah tampilan '*Home*', terdapat nama dan deskripsi singkat *website* yang dibuat bergerak hingga sedemikian rupa. Pada bagian kiri atas, terdapat nama *author* pembuat *website* DNA Pattern Matching ini. Sedangkan pada bagian kanan atas, terdapat beberapa pilihan menu yang nantinya menjadi bagian dari fitur-fitur yang terdapat di dalam aplikasi. Beberapa pilihan menu tersebut, antara lain menu '*Input New Disease*', menu '*DNA Prediction Test*', dan menu '*Search Test Result*'.

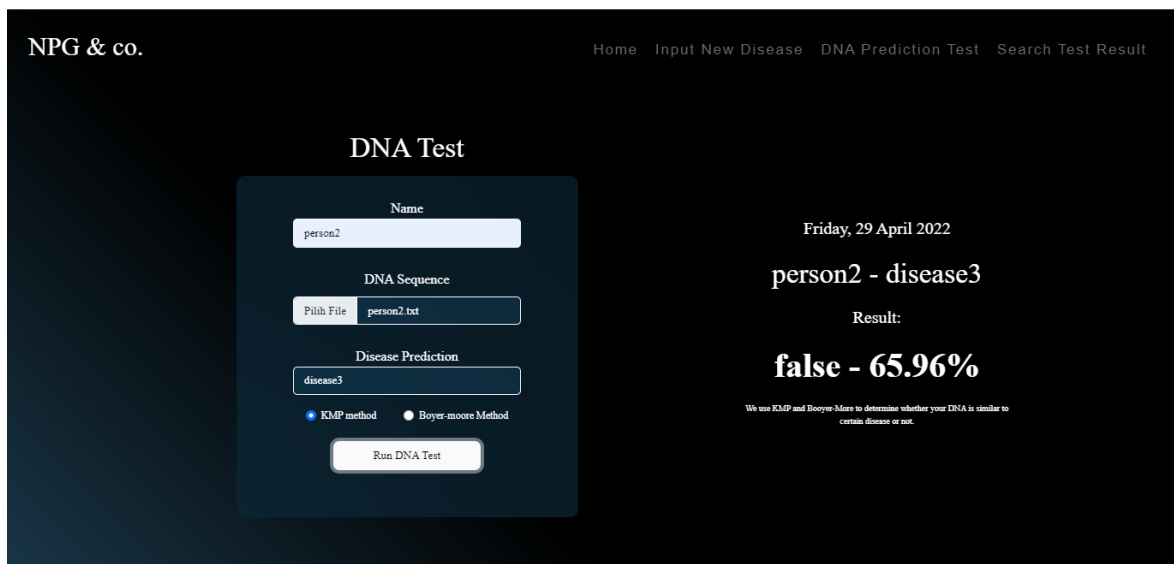


Gambar 4.2.2 *Interface Program: Tampilan Menu Input New Disease*

Selanjutnya, apabila pengguna memilih untuk menekan tombol menu ‘*Input New Disease*’, tampilan *website* akan berubah menjadi seperti Gambar 4.2.2. Pada halaman ini, pengguna bisa menggunakan fitur untuk menambahkan penyakit baru ke dalam *database* yang terhubung dengan aplikasi. Kolom ‘*Name*’ digunakan untuk mengisi nama penyakit baru, sedangkan tombol ‘Pilih File’ apabila ditekan, akan beralih ke *file explorer* di komputer lokal untuk memilih *file* berisi *DNA sequence* dari penyakit baru yang ingin ditambahkan. *File* yang dipilih harus berisi *sequence* DNA yang valid karena terdapat proses sanitasi *input* yang telah dilakukan dengan menerapkan konsep *Regular Expression*. Apabila kolom nama sudah diisi dan *file DNA* sequence sudah di-*input*, pengguna dapat segera mendaftarkan penyakit barunya ke *database* dengan menekan tombol ‘*Input Disease*’.



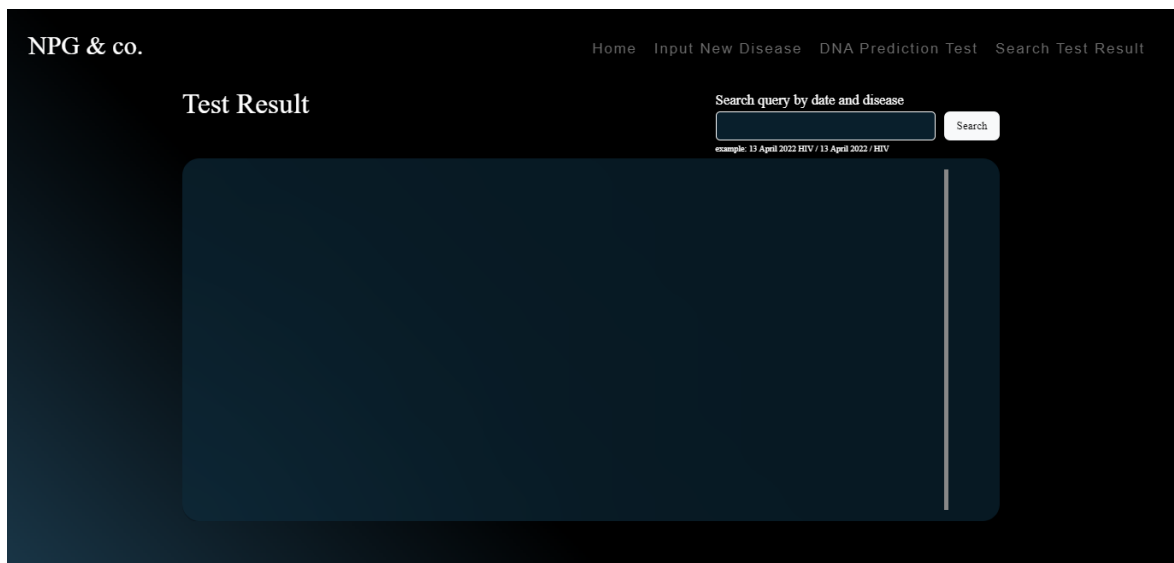
Gambar 4.2.3 *Interface Program: Tampilan Menu DNA Prediction Test*



Gambar 4.2.4 *Interface Program: Tampilan Hasil Tes Prediksi DNA*

Kemudian, apabila pengguna memilih untuk menekan tombol menu ‘*DNA Prediction Test*’, tampilan *website* akan berubah menjadi seperti Gambar 4.2.3. Pada halaman ini, pengguna bisa menggunakan fitur untuk melakukan tes prediksi DNA terhadap suatu penyakit genetik tertentu yang mungkin dialami oleh pasien/pengguna. Tes dapat dilakukan dengan memasukkan nama pasien/pengguna yang ingin dicek ke kolom ‘*Name*’, menambahkan *file DNA sequence* yang dimiliki oleh pasien untuk nanti dicocokkan dengan *DNA sequence* penyakit yang diprediksi dengan menekan tombol ‘Pilih File’, serta menuliskan nama penyakit yang diprediksi pada bagian kolom ‘*Disease*

Prediction'. Sebagai catatan, nama penyakit yang dimasukkan harus merupakan nama penyakit yang telah terdaftar ke dalam *database* aplikasi/*website*. Kemudian, pengguna dapat memilih metode atau algoritma yang ingin dipakai untuk menyelesaikan pencocokan string antara DNA *sequence* milik pengguna dengan DNA *sequence* penyakit yang diprediksi. Untuk melangsungkan tesnya, pengguna dapat langsung menekan tombol '*Run DNA Test*' dan aplikasi akan secara langsung melakukan proses prediksi dan menampilkan hasil prediksi seperti yang terdapat pada Gambar 4.2.4.



Gambar 4.2.5 *Interface* Program: Tampilan Menu *Search Test Result*

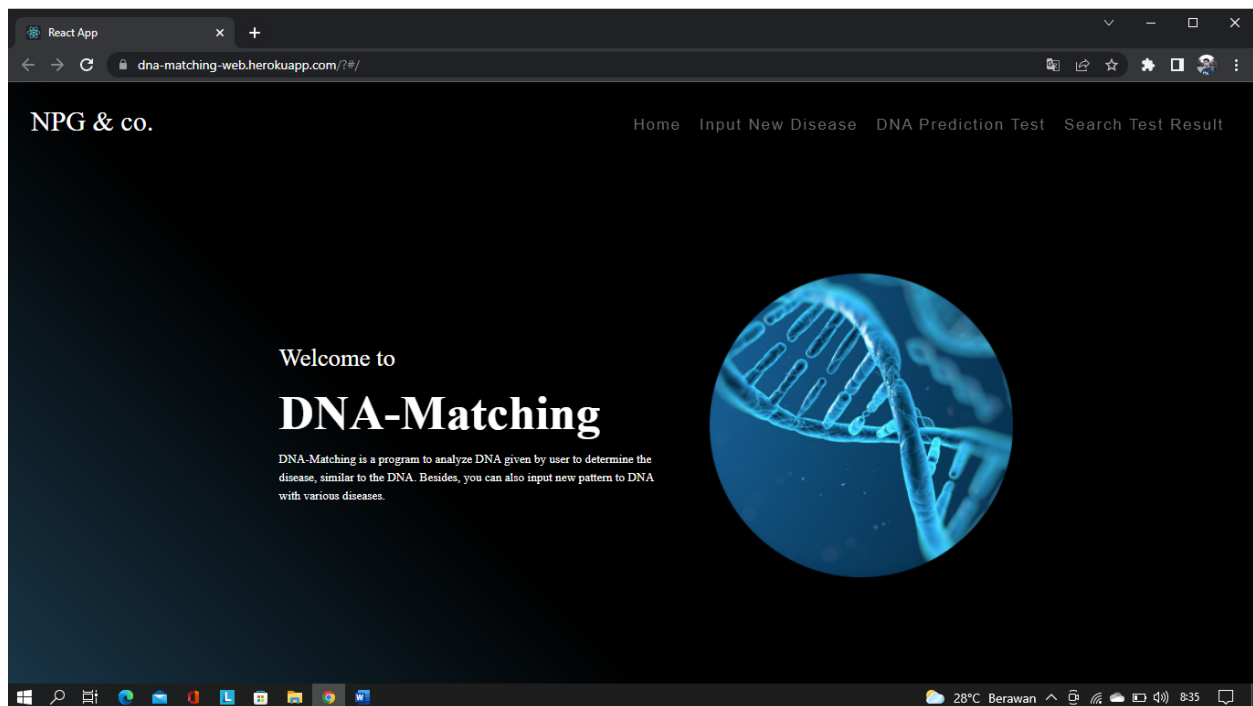


Gambar 4.2.6 *Interface* Program: Tampilan Daftar Hasil Tes Prediksi DNA

Terakhir, apabila pengguna memilih untuk menekan tombol menu ‘*Search Test Result*’, tampilan *website* akan berubah menjadi seperti Gambar 4.2.5. Pada halaman ini, pengguna bisa menggunakan fitur untuk melihat daftar hasil tes prediksi DNA yang telah dilakukan oleh pengguna. Daftar hasil tes ini didasarkan pada *record* yang telah tersimpan ke dalam *database* yang telah terhubung dengan aplikasi dengan bantuan kakas MongoDB. Pengguna dapat memasukkan *query* berupa tanggal tes dengan contoh format ‘13 April 2022’, nama penyakit dengan contoh format ‘HIV’, atau keduanya dengan contoh format ‘13 April 2022 HIV’. Setelah memasukkan *query* pada kolom ‘*Search query by date and disease*’ yang terdapat di kanan atas halaman *website*, pengguna dapat menekan tombol ‘*Search*’ untuk kemudian meminta program melakukan pencarian dan menampilkan daftar hasil tes terkait seperti yang terdapat pada Gambar 4.2.6.

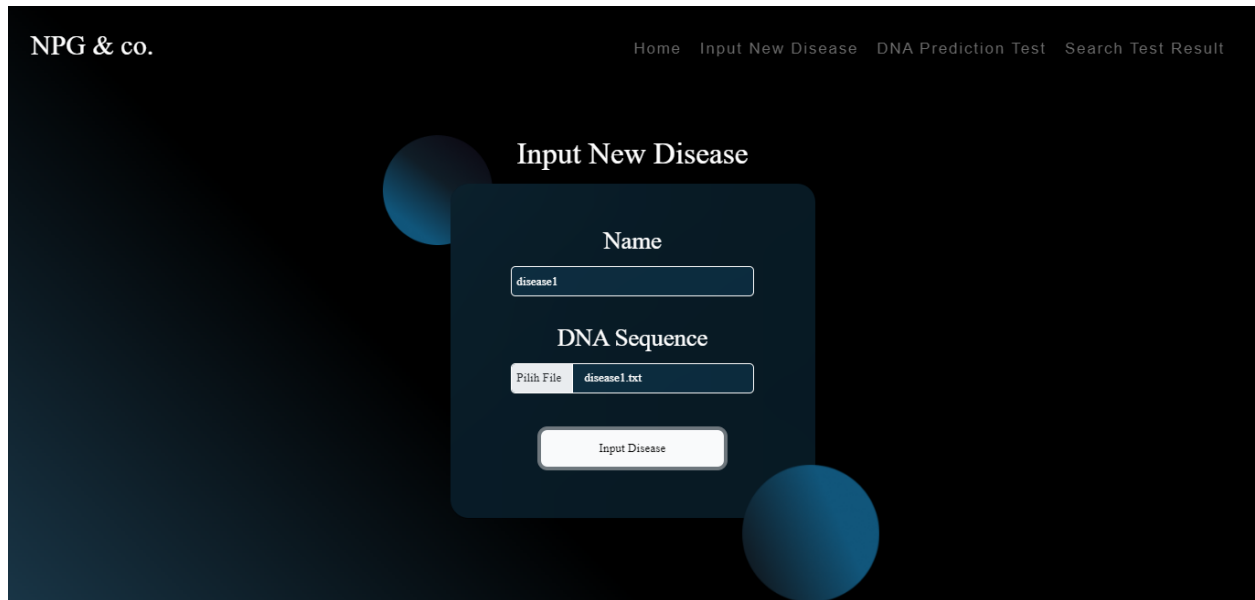
4.3. Hasil Pengujian

4.3.1. Men-deploy Aplikasi ke Website



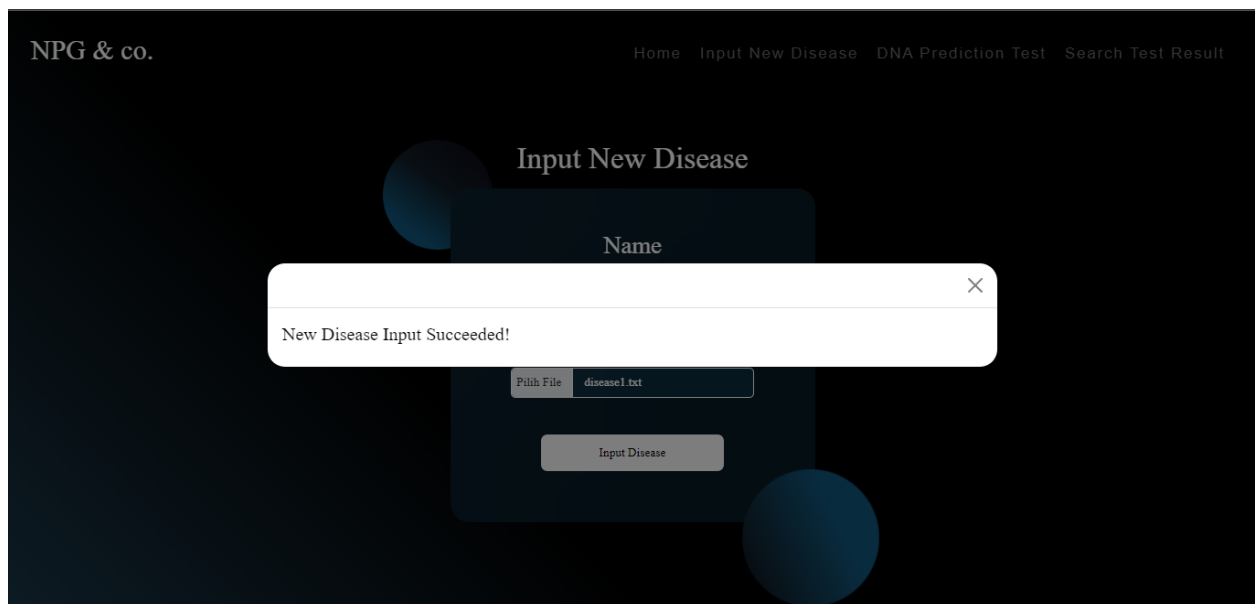
Gambar 4.3.1.1 Deploy Aplikasi ke Website

4.3.2. Menambahkan Penyakit Baru

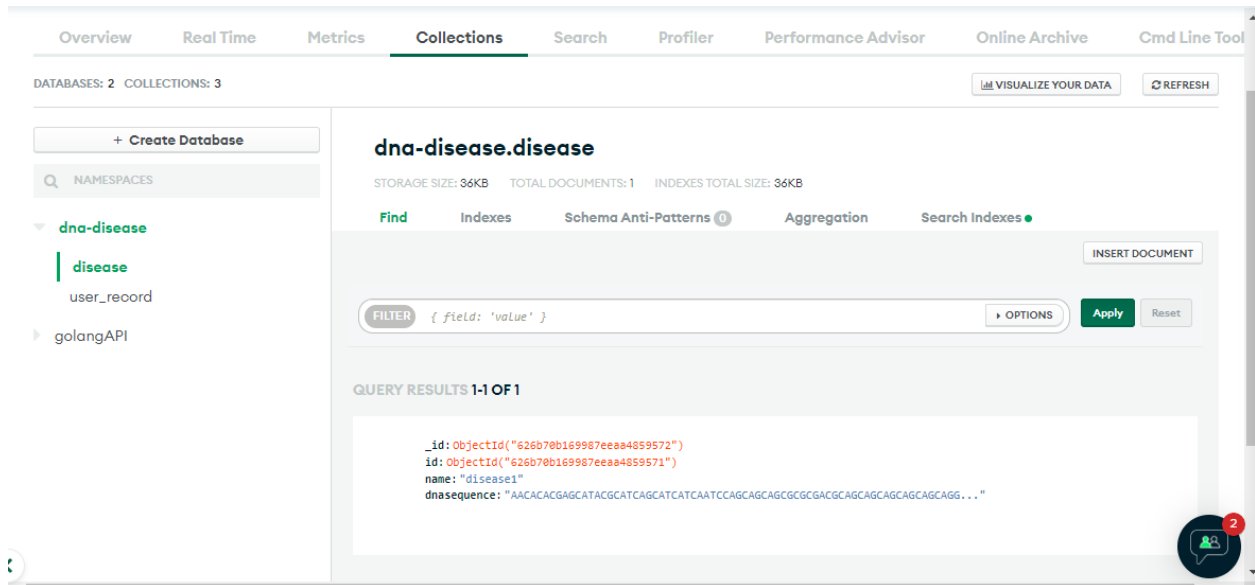


The screenshot shows the 'Input New Disease' form on the NPG & co. website. The form is centered on a dark blue background with decorative circles. It contains a 'Name' input field with the text 'disease1', a 'DNA Sequence' section with a 'Pilih File' button and a file input field containing 'disease1.txt', and an 'Input Disease' button at the bottom.

Gambar 4.3.2.1 Menambahkan Penyakit Baru

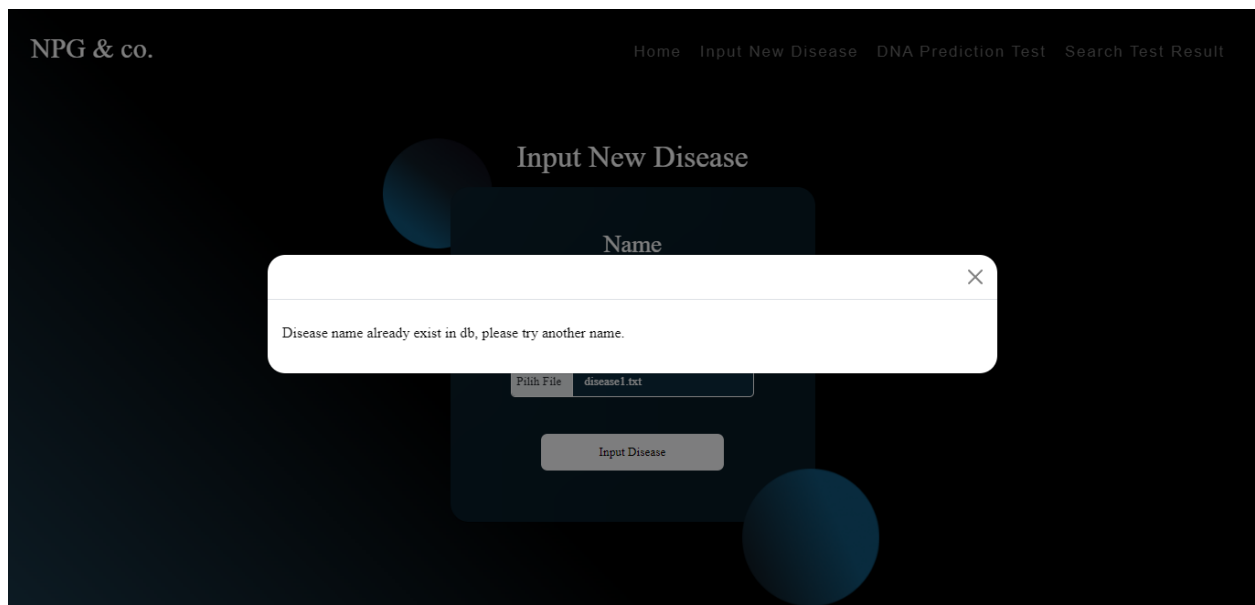


Gambar 4.3.2.2 *Pop-Up Message* saat Menambahkan Penyakit Baru



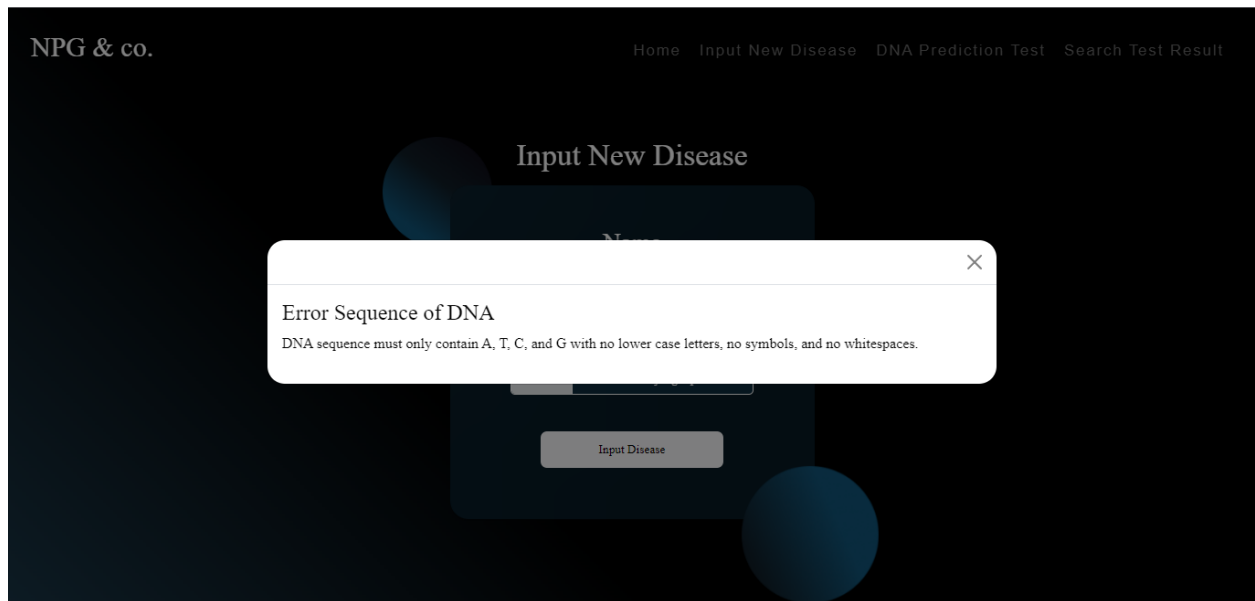
Gambar 4.3.2.3 Database Setelah Menambahkan Penyakit Baru

4.3.3. Menambahkan Penyakit dengan Nama yang Sudah Ada di Database



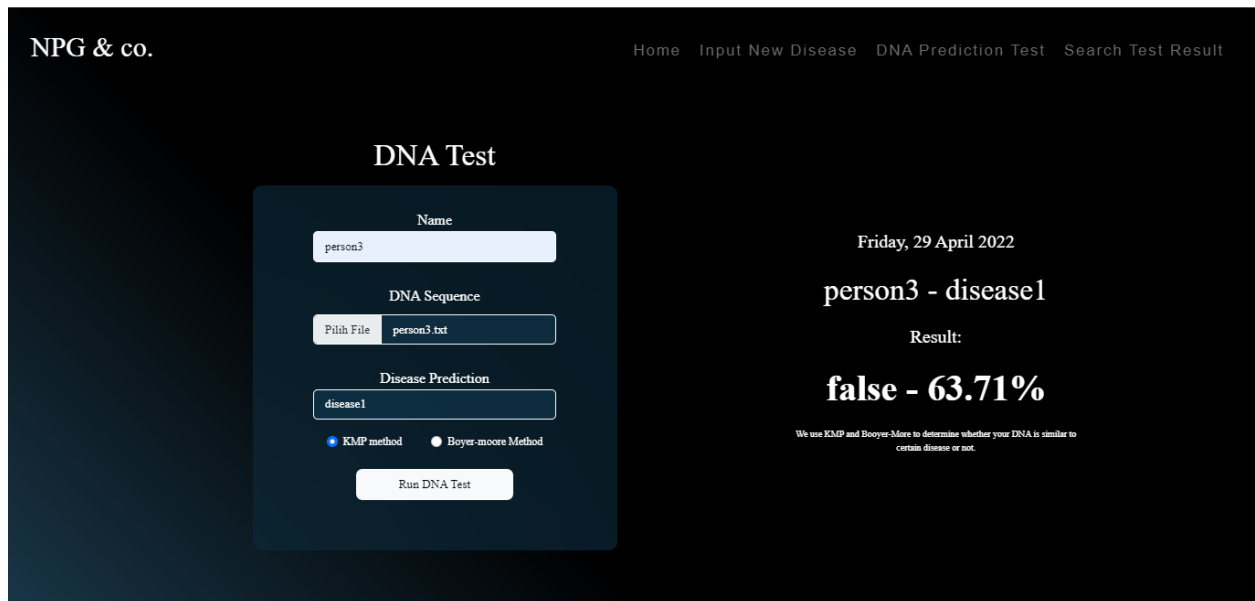
Gambar 4.3.3.1 Menambahkan Penyakit dengan Nama yang Sudah Ada di Database

4.3.4. Menambahkan Penyakit dengan DNA *Sequence* yang Tidak Valid

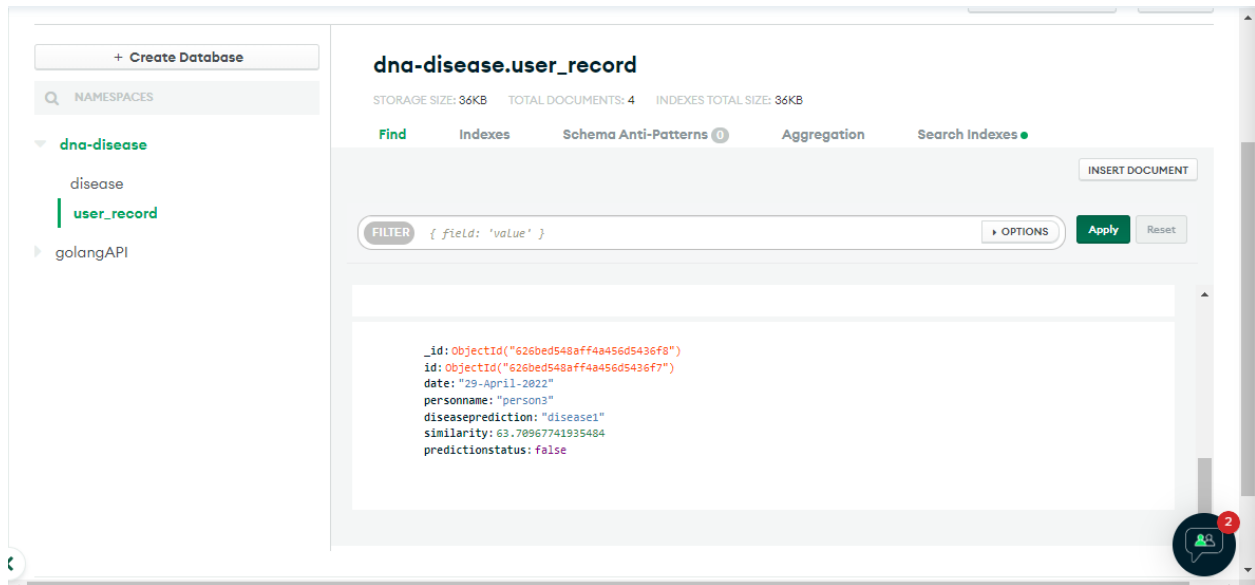


Gambar 4.3.4.1 Menambahkan Penyakit dengan DNA *Sequence* yang Tidak Valid

4.3.5. Melakukan Tes Prediksi DNA dengan Metode KMP

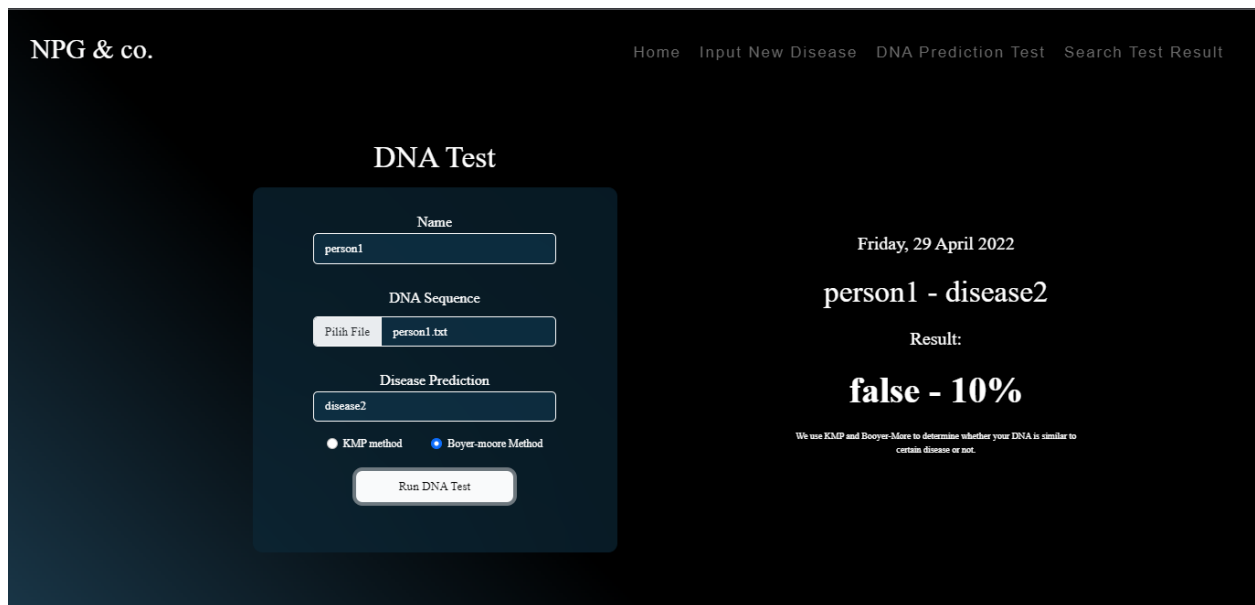


Gambar 4.3.5.1 Melakukan Tes Prediksi DNA dengan Metode KMP

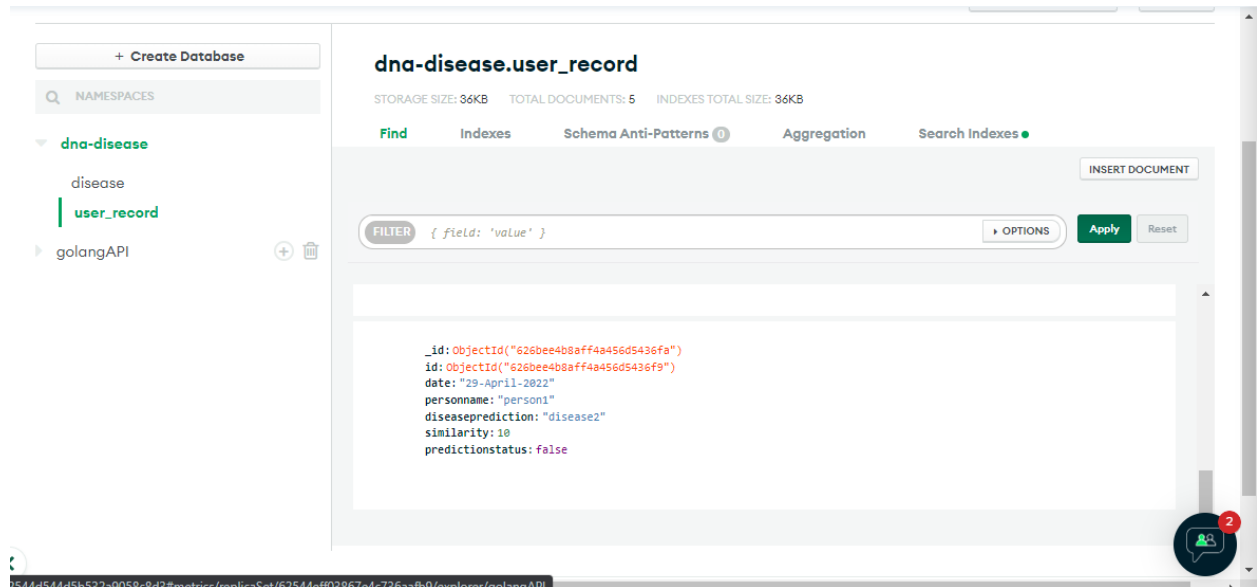


Gambar 4.3.5.2 Database Setelah Melakukan Tes Prediksi DNA dengan Metode KMP

4.3.6. Melakukan Tes Prediksi DNA dengan Metode BM

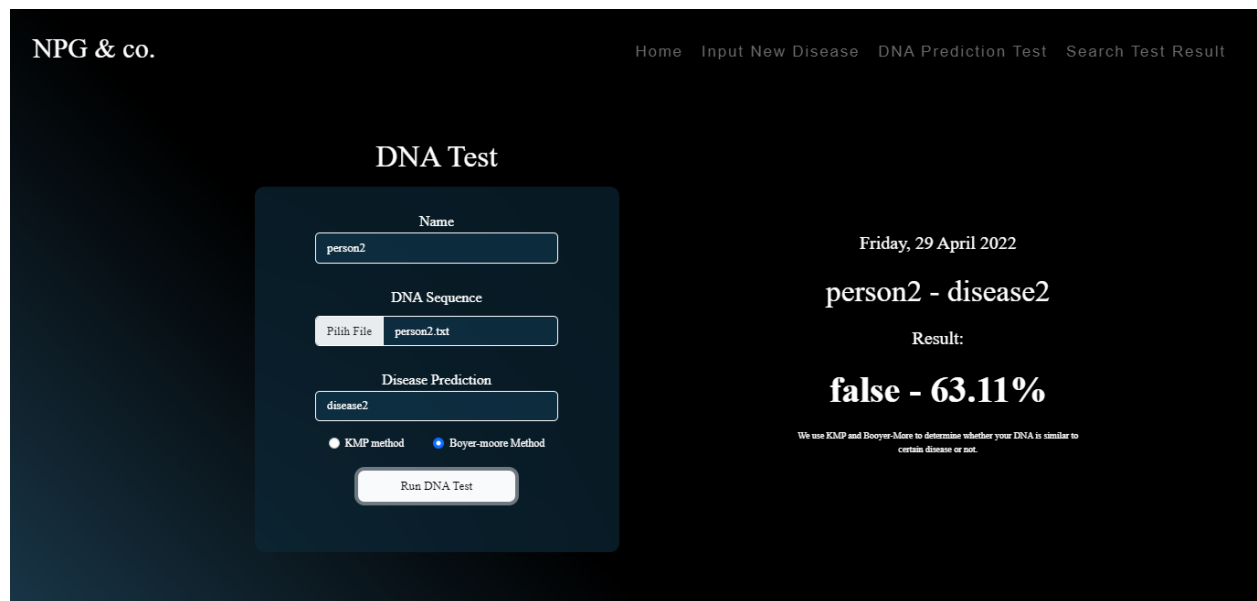


Gambar 4.3.6.1 Melakukan Tes Prediksi DNA dengan Metode BM

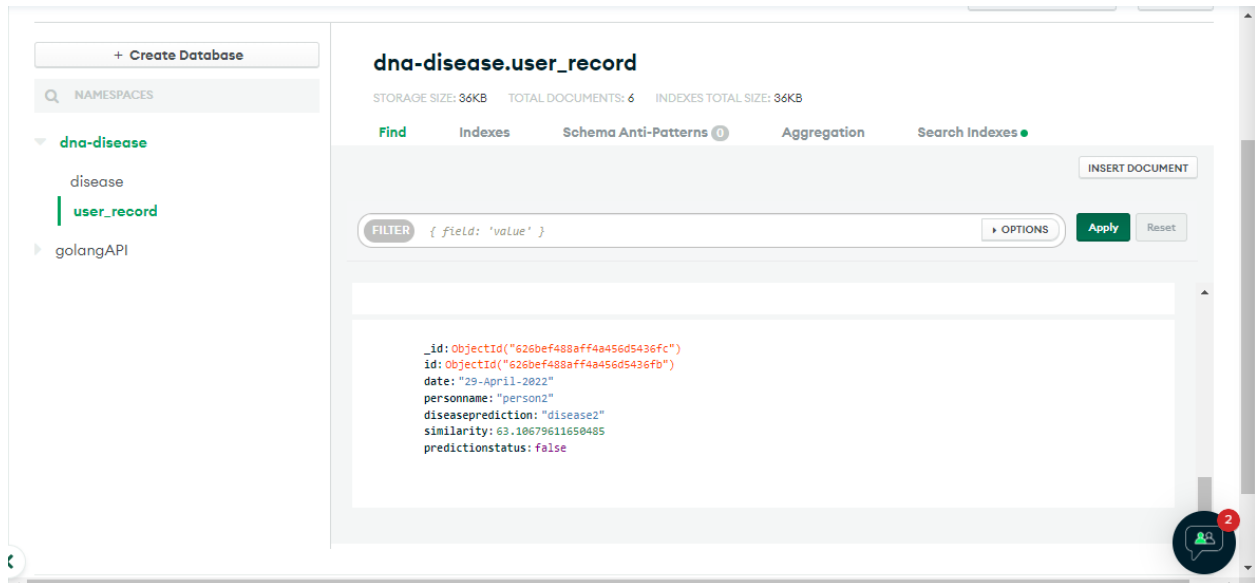


Gambar 4.3.6.2 Database Setelah Melakukan Tes Prediksi DNA dengan Metode BM

4.3.7. Melakukan Tes Prediksi DNA dengan Hasil Status Terprediksi *False*

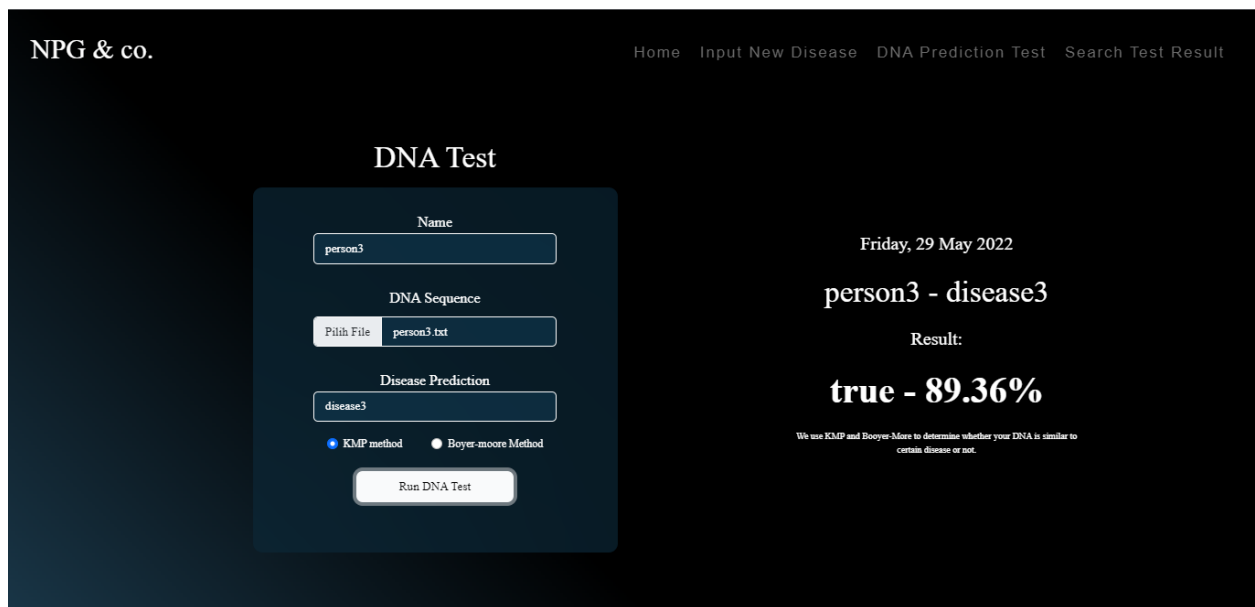


Gambar 4.3.7.1 Melakukan Tes Prediksi DNA dengan Hasil Status Terprediksi *False*

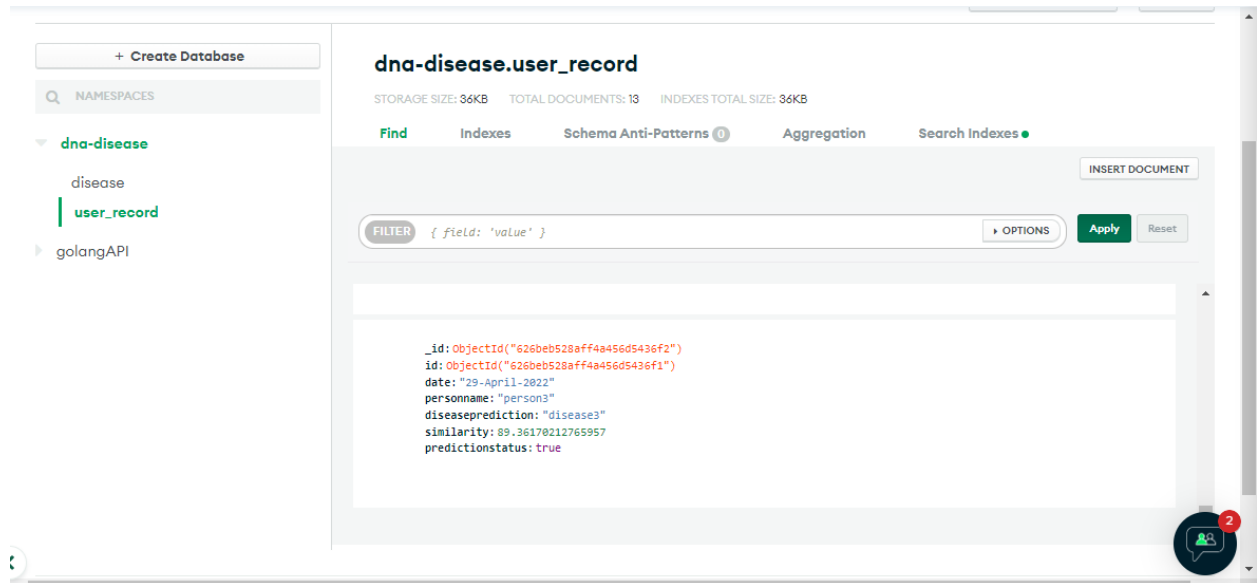


Gambar 4.3.7.1 Database Setelah Melakukan Tes Prediksi DNA dengan Hasil Status Terprediksi *False*

4.3.8. Melakukan Tes Prediksi DNA dengan Hasil Status Terprediksi *True*

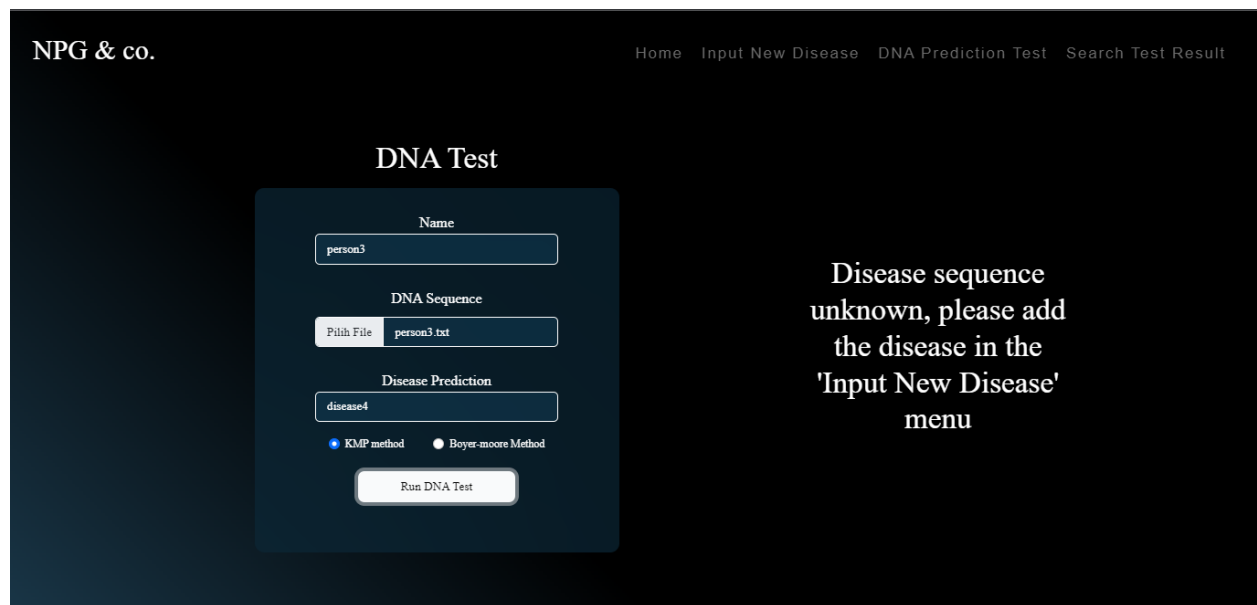


Gambar 4.3.8.1 Melakukan Tes Prediksi DNA dengan Hasil Status Terprediksi *True*



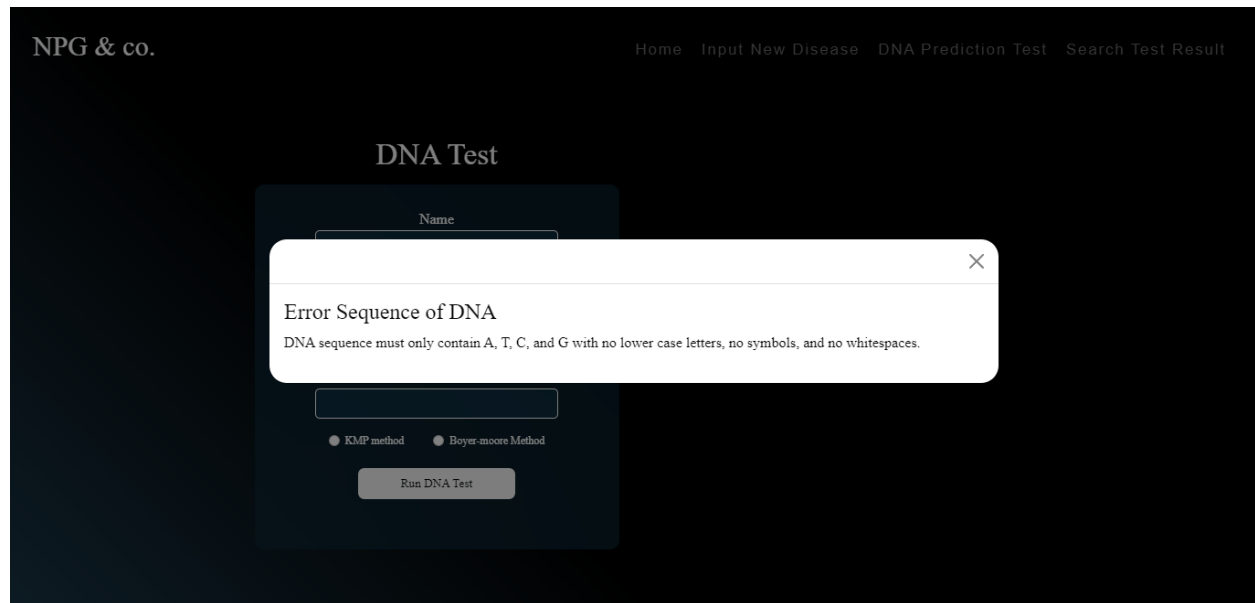
Gambar 4.3.8.2 Database Setelah Melakukan Tes Prediksi DNA dengan Hasil Status Terprediksi *True*

4.3.9. Melakukan Tes Prediksi DNA dengan Nama Penyakit yang Belum Terdaftar



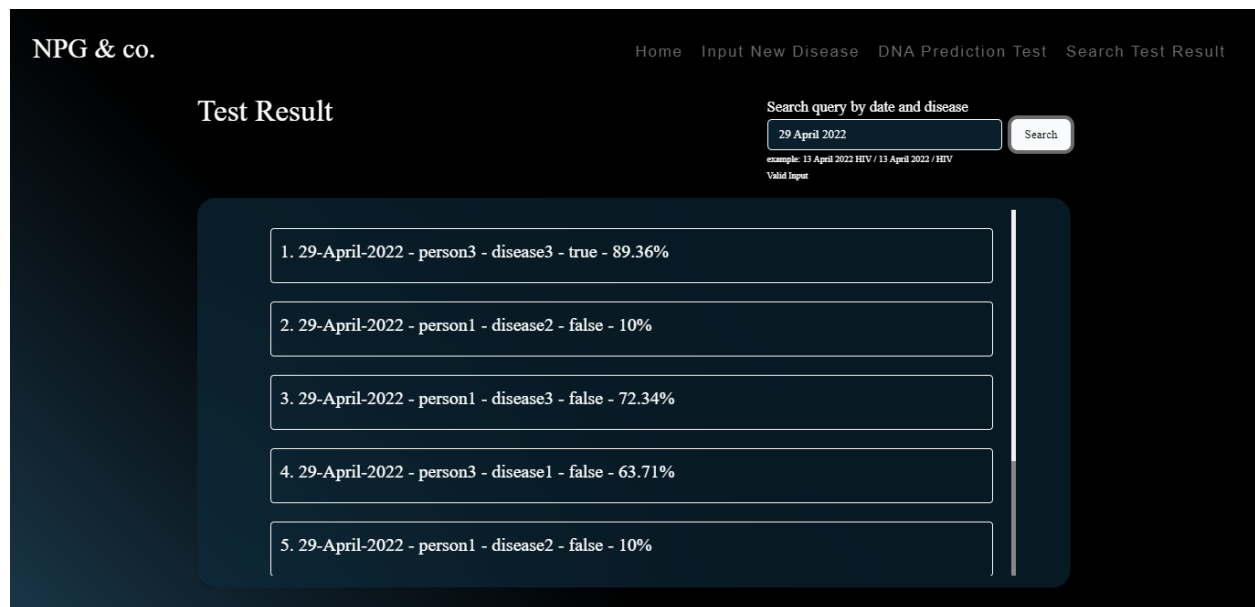
Gambar 4.3.9.1 Melakukan Tes Prediksi DNA dengan Nama Penyakit yang Belum Terdaftar

4.3.10. Melakukan Tes Prediksi DNA dengan DNA *Sequence* Pengguna Tidak Valid



Gambar 4.3.10.1 Melakukan Tes Prediksi DNA dengan DNA *Sequence* Pengguna Tidak Valid

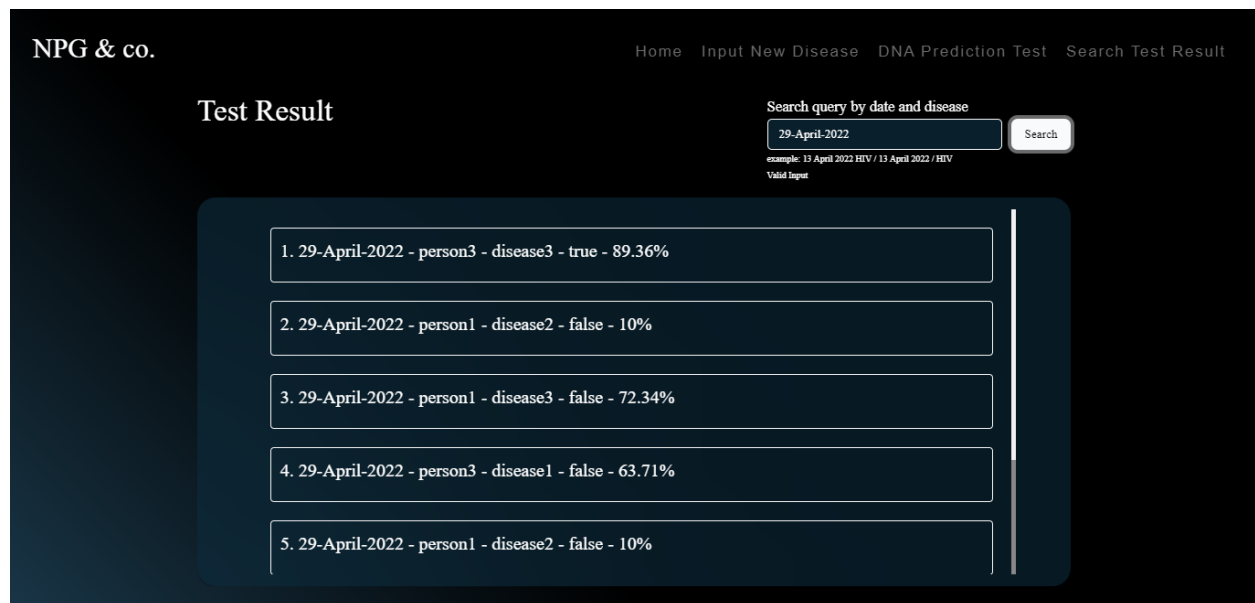
4.3.11. Melakukan Pencarian Hasil Tes Prediksi DNA dengan *Query* Tanggal Tes



Gambar 4.3.11.1 Melakukan Pencarian Hasil Tes Prediksi DNA dengan *Query* Tanggal Tes dengan Separator *Whitespaces*



Gambar 4.3.11.2 Melakukan Pencarian Hasil Tes Prediksi DNA dengan *Query* Tanggal Tes dengan Separator Garis Miring



Gambar 4.3.11.3 Melakukan Pencarian Hasil Tes Prediksi DNA dengan *Query* Tanggal Tes dengan Separator Tanda Strip

4.3.12. Melakukan Pencarian Hasil Tes Prediksi DNA dengan *Query* Nama Penyakit

The screenshot shows the 'Test Result' page of the NPG & co. application. The header includes the logo 'NPG & co.' and navigation links: 'Home', 'Input New Disease', 'DNA Prediction Test', and 'Search Test Result'. The main heading is 'Test Result'. On the right, there is a search section titled 'Search query by date and disease' with a text input field containing 'disease2' and a 'Search' button. Below the input field, there is an example: 'example: 13 April 2022 HIV / 13 April 2022 / HIV' and a note 'Valid Input'. The results are displayed in a list on the left, showing three entries:

1. 29-April-2022 - person1 - disease2 - false - 10%
2. 29-April-2022 - person1 - disease2 - false - 10%
3. 29-April-2022 - person2 - disease2 - false - 63.11%

Gambar 4.3.12.1 Melakukan Pencarian Hasil Tes Prediksi DNA dengan *Query* Nama Penyakit

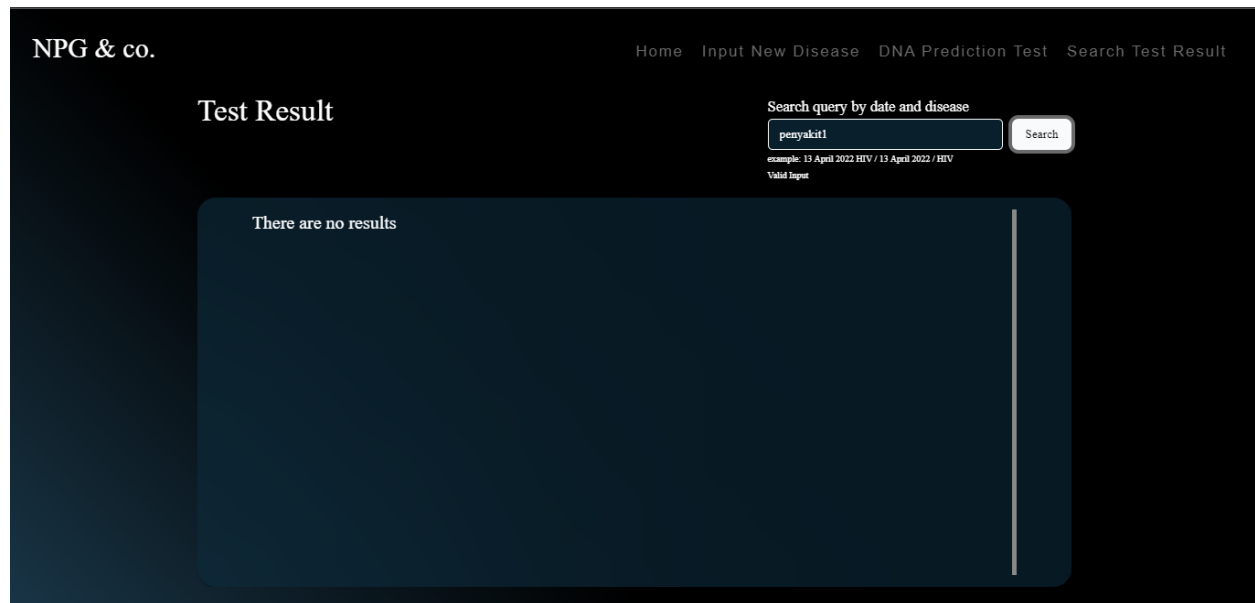
4.3.13. Melakukan Pencarian Hasil Tes Prediksi DNA dengan *Query* Tanggal Tes dan Nama Penyakit

The screenshot shows the 'Test Result' page of the NPG & co. application. The header includes the logo 'NPG & co.' and navigation links: 'Home', 'Input New Disease', 'DNA Prediction Test', and 'Search Test Result'. The main heading is 'Test Result'. On the right, there is a search section titled 'Search query by date and disease' with a text input field containing '29 April 2022 disease1' and a 'Search' button. Below the input field, there is an example: 'example: 13 April 2022 HIV / 13 April 2022 / HIV' and a note 'Valid Input'. The results are displayed in a list on the left, showing one entry:

1. 29-April-2022 - person3 - disease1 - false - 63.71%

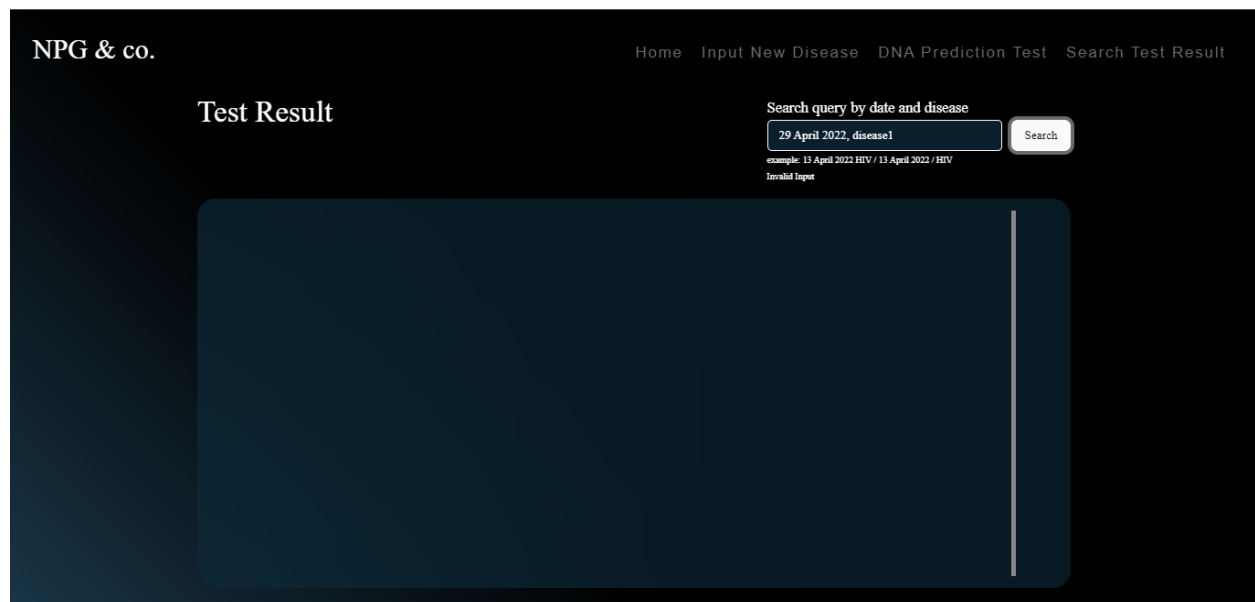
Gambar 4.3.13.1 Melakukan Pencarian Hasil Tes Prediksi DNA dengan *Query* Tanggal Tes dan Nama Penyakit

- 4.3.14. Melakukan Pencarian Hasil Tes Prediksi DNA dengan *Query* Valid, namun Tidak Ditemukan Hasil Terkait



Gambar 4.3.14.1 Melakukan Pencarian Hasil Tes Prediksi DNA dengan *Query* Valid, namun Tidak Ditemukan Hasil Terkait

- 4.3.15. Melakukan Pencarian Hasil Tes Prediksi DNA dengan *Query* yang Tidak Valid



Gambar 4.3.15.1 Melakukan Pencarian Hasil Tes Prediksi DNA dengan *Query* yang Tidak Valid

4.4. Analisis Hasil Pengujian

Berdasarkan proses pengujian program yang telah kelompok kami lakukan, aplikasi web *DNA Pattern Matching* yang kami buat dapat berjalan dengan baik, baik itu pada saat dijalankan di lokal, maupun pada saat di-*deploy* di *website*. Tentu, terdapat perbedaan kecepatan pada saat menjalankan program di lokal dan di *website*, di mana *running* program di lokal memerlukan waktu yang lebih singkat dibandingkan dengan pada saat di-*deploy* di *website*. Pengujian telah dilakukan terhadap seluruh fitur yang tersedia di aplikasi dengan mempertimbangkan berbagai kemungkinan skenario yang dapat terjadi.

Pada saat pertama kali dijalankan, program berhasil masuk ke menu halaman '*Home*' dengan tampilan yang sedemikian rupa telah didesain. Program berhasil menerima masukan berupa nama penyakit baru beserta dengan *sequence* DNA spesifiknya dalam bentuk *file input*. Hal ini dapat dilakukan pada aplikasi bagian halaman '*Input New Disease*'. Nama penyakit yang didaftarkan kemudian tersimpan ke dalam *database* yang telah terhubung dengan bantuan kakas MongoDB, khususnya di bagian tabel '*disease*'. Nama penyakit yang telah ditambahkan ke dalam *database* tidak bisa di-*input* kembali oleh pengguna.

Selanjutnya, hasil pengujian program juga menunjukkan bahwa aplikasi dapat melakukan tes prediksi DNA terhadap suatu penyakit genetik tertentu. Aplikasi mampu membuktikan apakah *sequence* DNA yang dimiliki seseorang cocok atau mirip dengan DNA penyakit tertentu sehingga memberikan status *true/false* untuk prediksi penyakit tersebut. Proses pencocokan ini dapat dilakukan dengan 2 opsi algoritma pencocokan string yang tersedia. Selain itu, tingkat kemiripan dari DNA juga dapat ditampilkan dan kemudian disimpan bersamaan dengan hasil tes ke dalam *database*, khususnya pada tabel *user_record*. Tes prediksi DNA dapat ditolak apabila penyakit yang dimasukkan belum terdaftar ke dalam *database* sehingga kemudian memberikan pesan notifikasi untuk menambahkan *sequence* DNA penyakit tersebut.

Terakhir, program berhasil diuji dalam hal pencarian hasil tes yang telah disimpan di dalam *database* dari tes-tes prediksi DNA yang pernah dilakukan sebelumnya. Pencarian dapat dilakukan dengan memasukkan *input query* berupa tanggal tes, nama penyakit, atau lebih spesifik, yakni keduanya sesuai dengan contoh yang tertera pada halaman '*Search Test Result*'.

Aplikasi akan menampilkan hasil dari *query* yang bersesuaian. Apabila *input query* tidak sesuai dengan yang ada pada contoh, aplikasi akan memberikan pesan kesalahan bahwa *input* yang dimasukkan tidak valid. Sanitasi input yang diimplementasikan dengan *Regular Expression* telah me-*filter* masukan-masukan pengguna yang dianggap tidak valid. Selain itu, aplikasi juga bisa saja tidak menampilkan data apa pun, hanya menampilkan pesan yang menunjukkan tidak ada data yang sesuai apabila *query* yang diminta tidak mengarah ke *record* mana pun yang telah tersimpan di *database*.

Secara keseluruhan, pengujian telah memberikan gambaran mengenai fungsionalitas serta tampilan dari aplikasi web yang telah kelompok kami buat. Pengujian ini memberikan testimoni yang kuat kepada calon pengguna bahwa aplikasi yang dibangun telah mampu berjalan dan mengeksekusi tiap fitur-fiturnya dengan baik. Dengan demikian, hasil pengujian ini telah memvalidasi kebergunaan dan keberjalanan aplikasi web *DNA Pattern Matching* buatan kami.

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Telah berhasil diimplementasikan sebuah program berupa aplikasi berbasis web yang dirancang dan dikembangkan secara interaktif untuk mendeteksi apakah seorang pasien mempunyai penyakit genetik tertentu sesuai dengan yang diminta dalam Spesifikasi Tugas Besar 3 IF2211 Strategi Algoritma Semester 2 Tahun 2021/2022. Hal mengenai program aplikasi DNA *Pattern Matching* yang berhasil diimplementasikan dalam program ini meliputi:

1. Konsep algoritma Knuth-Morris-Pratt dan Boyer-Moore, serta penerapannya dalam implementasi program aplikasi DNA *Pattern Matching*,
2. Konsep pencocokan string dengan menggunakan *Regular Expression*,
3. Perhitungan tingkat kemiripan yang dilakukan dengan menggunakan *Levenshtein Distance Algorithm*,
4. Implementasi *frontend* menggunakan React dan *backend* menggunakan Golang,
5. Pemanfaatan basis data MongoDB sebagai tempat penyimpanan data hasil *input* dari *user* di aplikasi web yang telah dibuat.

Semua implementasi dari konsep-konsep di atas kemudian berhasil digunakan untuk menyelesaikan seluruh fitur yang ada di dalam spesifikasi. Fitur-fitur tersebut telah terdapat pada aplikasi DNA *Pattern Matching* yang kami buat. Setidaknya terdapat 4 fitur utama yang dapat digunakan pada aplikasi kami, antara lain:

1. Aplikasi dapat menerima input penyakit baru berupa nama penyakit dan *sequence* DNA-nya (dan dimasukkan ke dalam *database*),
2. Aplikasi dapat memprediksi seseorang menderita penyakit tertentu berdasarkan *sequence* DNA-nya,
3. Aplikasi memiliki halaman yang menampilkan urutan hasil prediksi dengan kolom pencarian di dalamnya yang bekerja sebagai filter dalam menampilkan hasil,

4. Aplikasi dapat menampilkan tingkat kemiripan DNA pengguna dengan DNA penyakit pada tes DNA.

Fitur-fitur ini dapat digunakan pada aplikasi web kami yang sudah diciptakan sedemikian rupa dengan memanfaatkan tampilan *frontend* yang *user-friendly* dan menarik untuk dapat mempermudah interaksi antara *user* dengan aplikasi.

Dengan pengimplementasian berbagai algoritma pencocokan string/pencarian pola yang telah disisipkan pada bagian *backend* aplikasi, khususnya dengan menggunakan konsep algoritma Knuth-Morris-Pratt dan Boyer-Moore, kita dapat memecahkan permasalahan di bidang kesehatan dan kedokteran yang dapat kita temukan dalam kehidupan sehari-hari, yakni melakukan pendeteksian terhadap seseorang apakah ia memiliki penyakit genetik tertentu. Konsep yang telah diajarkan di perkuliahan IF2211 dapat dengan baik diterapkan dalam pengerjaan Tugas Besar 3 IF2211 Strategi Algoritma ini. Selain melakukan eksplorasi dan implementasi terhadap berbagai alternatif algoritma pencocokan string yang dapat dipakai, kelompok juga berhasil melakukan sanitasi input menggunakan *Regular Expression* untuk memastikan bahwa masukan merupakan *sequence* DNA yang valid sesuai dengan yang diminta di spesifikasi.

Dengan demikian, kelompok menyimpulkan bahwa dengan mengerjakan Tugas Besar 3 IF2211 Strategi Algoritma Semester 2 Tahun 2021/2022 ini, dapat diketahui bahwa untuk menyelesaikan suatu masalah yang mungkin ditemukan dalam kehidupan sehari-hari, dalam hal ini misalnya, memprediksi apakah seseorang menderita penyakit genetik tertentu berdasarkan *sequence* DNA-nya, dapat diimplementasikan program berupa aplikasi berbasis web yang memodelkan fitur tes DNA untuk melakukan pendeteksian terhadap suatu penyakit genetik tertentu yang dialami oleh seseorang sebagai bentuk penerapan dari konsep algoritma *String Matching* dan *Regular Expression* yang telah dipelajari pada kuliah IF2211.

5.2. Saran

Tugas Besar 3 IF2211 Strategi Algoritma Semester 2 Tahun 2021/2022 menjadi salah satu proses pembelajaran bagi kelompok dalam menerapkan ilmu-ilmu yang

diajarkan pada kuliah maupun melakukan eksplorasi materi secara mandiri. Berikut ini merupakan sejumlah saran dari kelompok untuk pihak-pihak yang ingin melakukan atau mengerjakan hal serupa.

1. Program yang diminta adalah program yang terdiri atas tampilan *frontend* serta algoritma *backend* yang lengkap untuk membentuk suatu *website* *DNA Pattern Matching* dengan spesifikasi yang telah ditentukan. Selain itu, untuk memaksimalkan *output*, program juga perlu dikembangkan dengan beberapa bahasa pemrograman sekaligus sehingga diperlukan adanya kemampuan elaboratif dan koordinatif yang baik antar anggota kelompok dalam membuat program ini. Sebagian bahasa pemrograman dan *framework* yang kami gunakan merupakan hal yang baru kami kenali sejak pengerjaan tugas besar ini. Oleh karena itu, kelompok merekomendasikan agar disediakan waktu yang cukup untuk melakukan eksplorasi terkait bahasa pemrograman serta *framework* yang akan digunakan sebelum mengimplementasikannya ke dalam program. Hal ini akan meningkatkan efektivitas kerja tim dalam pembuatan suatu program. Di samping itu, perlu dipertimbangkan pula pemilihan bahasa serta *framework* yang tepat dan sesuai dengan kebutuhan program agar dapat meningkatkan efektivitas proses pengerjaan program.
2. Modularitas menjadi hal yang krusial dalam menciptakan suatu program secara efektif dan efisien. Dalam jangka waktu yang tidak panjang, pemrograman secara modular dapat membantu *programmer* untuk memudahkan proses pencarian kesalahan/*error* serta *debugging*. Pada dasarnya, memrogram secara modular berarti memecah-mecah program menjadi modul-modul kecil di mana masing-masing modul berinteraksi melalui antarmuka modul. Masalah yang awalnya kompleks dapat dibagi menjadi bagian-bagian kecil yang lebih sederhana dan dapat diselesaikan dalam lingkup yang lebih kecil. Akibatnya, apabila terdapat *error/bug* pada program, kesalahan dapat dengan mudah ditemukan karena alur logika yang jelas serta dapat dilokalisasi dalam satu modul. Lebih dari pada itu, modifikasi program dapat dilakukan tanpa mengganggu *body* program secara keseluruhan. Oleh karena itu, kelompok sangat menyarankan untuk melakukan pemrograman secara modular dalam mengimplementasikan algoritma

String Matching dan *Regular Expression* dalam pembuatan aplikasi *DNA Pattern Matching* ini.

3. Penting bagi kelompok untuk memiliki strategi serta distribusi tugas yang baik. Ketika membuat program dalam sebuah tim, kesamaan cara menulis kode serta kemampuan untuk menulis komentar menjadi hal yang sangat penting. Hal ini diperlukan agar memudahkan anggota kelompok dalam menyatukan dan melanjutkan sebuah program. Kemampuan tersebut tentunya didukung juga dengan adanya *version control system* yang baik yang dapat digunakan oleh *programmer* dalam membuat sebuah program secara bersama-sama. Untuk itu, kami sangat menyarankan ‘GitHub’ untuk digunakan sebagai *version control system* dalam pengerjaan tugas-tugas besar pada mata kuliah IF2211 ini, maupun pada pembuatan program dan pengerjaan proyek yang lainnya.
4. Kelompok menyadari bahwa pada implementasi program aplikasi *DNA Pattern Matching* yang telah kami buat, masih banyak aspek yang dapat dikembangkan lebih lagi. Salah satunya ialah dengan mengoptimalkan algoritma pencocokan string yang digunakan pada sisi *backend* program agar proses prediksi dan pendeteksian penyakit genetik berdasarkan *sequence* DNA dapat berlangsung dengan lebih cepat. Program juga dapat dikembangkan dari sisi *frontend*, yakni UI/UX yang telah diimplementasikan oleh kelompok dalam bentuk *website*. Selain itu, bisa pula dilakukan eksplorasi terkait fitur-fitur aplikasi terkait yang mungkin belum kelompok kami implementasikan lebih lanjut. Hal ini tentu menjadi ruang untuk *programmer* agar dapat melakukan improvisasi terhadap implementasi dan pengembangan aplikasi *DNA Pattern Matching* ini, terutama dalam hal pengembangan algoritma dan desain UI/UX.

5.3. Refleksi

Dalam pengerjaan Tugas Besar 3 IF2211 Strategi Algoritma Semester 2 Tahun 2021/2022 ini, banyak hal dan sudut pandang yang diperoleh dan dipelajari oleh kelompok mengenai konsep pencocokan string pada tes DNA seseorang. Timbul apresiasi yang besar dalam diri anggota kelompok kepada semua orang yang terlibat dalam penemuan algoritma pencocokan string, khususnya algoritma Knuth-Morris-Pratt

dan Boyer-Moore. Keberadaan kedua algoritma ini memiliki dampak dan manfaat yang signifikan, terutama dalam menyelesaikan suatu masalah yang berkaitan dengan kesehatan, yakni seperti apa yang telah diimplementasikan pada tugas besar ini, prediksi terhadap suatu penyakit genetik tertentu. Dengan begitu, kelompok menyadari bahwa orang-orang tersebut memiliki peran yang patut dihargai sebagai bagian dari kelompok yang telah melahirkan maupun mengkaji ilmu serta penerapan dari algoritma-algoritma itu sendiri. Konsep *String Matching* yang sarat akan kegunaannya tentu membuat salah satu penemuan ini patut untuk dipelajari serta ditekuni agar dapat diimplementasikan di kemudian hari.

Melalui Tugas Besar 3 IF2211 Strategi Algoritma Semester 2 Tahun 2021/2022 juga, kelompok belajar untuk saling bekerja sama sebagai sebuah tim. Kami belajar untuk dapat bersikap kolaboratif dan koordinatif dengan menyesuaikan diri terhadap situasi dan kondisi yang kami hadapi. Sebagai bentuk kerja sama dan penyusunan strategi yang baik dari kelompok kami, pada awal pengerjaan tugas besar ini, dilakukan eksplorasi mandiri terkait materi dan spesifikasi yang kemudian dilanjutkan dengan pembahasan teknis pengerjaan program secara bersama-sama. Kerja sama ini tentunya dilakukan juga dengan pembagian tugas masing-masing serta pemberian masukan solusi dari seluruh anggota apabila terdapat tugas yang dirasa sulit bagi salah seorang penanggung jawab. Selain itu, agar dapat menyamakan pemahaman alur dari kode yang dibuat oleh masing-masing anggota, setiap dari kami diperkenankan untuk menuliskan *comment* pada *source code* program serta melakukan komunikasi dengan baik melalui grup yang kami punya secara virtual untuk menjelaskan lebih detail terkait kode program yang cukup sulit dipahami oleh anggota kelompok yang lain. Masing-masing dari kami secara *non-technical* mempelajari betapa pentingnya manajemen waktu, tanggung jawab, serta koordinasi dalam mengerjakan suatu tugas bersama. Di samping itu, dari sisi *technical*, kami jadi belajar banyak mengenai *full-stack programming*, terutama dengan apa yang telah kami gunakan pada pengerjaan tugas besar ini. Berkat kemampuan individu serta kerja sama antar anggota yang baik, Tugas Besar 3 IF2211 Strategi Algoritma Semester 2 Tahun 2021/2022 ini dapat terselesaikan dengan tepat waktu.

LAMPIRAN

Link *repository* GitHub:

https://github.com/graceclaudia19/Tubes3_13520078_stima

Link hasil *deploy website*:

<https://dna-matching-web.herokuapp.com/?#/>

Link *video* demo:

<https://youtu.be/fdJkJTwWl3A>

DAFTAR PUSTAKA

Munir, Rinaldi. (2019). *String Matching dengan Regular Expression*. Institut Teknologi Bandung.

<http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf>. Diakses pada 21 April 2022.

Munir, Rinaldi. (2021). *Pencocokan String (Pattern/String Matching)*. Institut Teknologi Bandung.

<http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf>. Diakses pada 21 April 2022.