

**IF2211 STRATEGI ALGORITMA
TUGAS BESAR 3 SEMESTER II TAHUN 2021/2022**

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



Kelompok Reja Impek and Friends

**13520048 Arik Rayi Arkananta
13520060 Rheza Rizqullah Ecaldy
13520099 Vincent Prasetya Atmadja**

**PROGRAM STUDI SARJANA TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2021**

Daftar Isi

Daftar Isi	2
Bab 1	3
Bab 2	9
Landasan Teori	9
Algoritma Knuth-Morris-Pratt	9
Algoritma Boyer-Moore	10
Regex	11
Algoritma Longest Common Subsequence	12
Aplikasi Web MERN Stack	12
Bab 3	14
Langkah Penyelesaian Masalah	14
Fitur Fungsional dan Arsitektur Aplikasi	15
Bab 4	18
Spesifikasi Teknis Program	18
Tata Cara Penggunaan Program	24
Hasil Pengujian	27
Analisis Hasil Pengujian	30
Bab 5	31
Kesimpulan	31
Saran	31
Refleksi	31
Link Repository dan Deploy	32
Daftar Pustaka	33

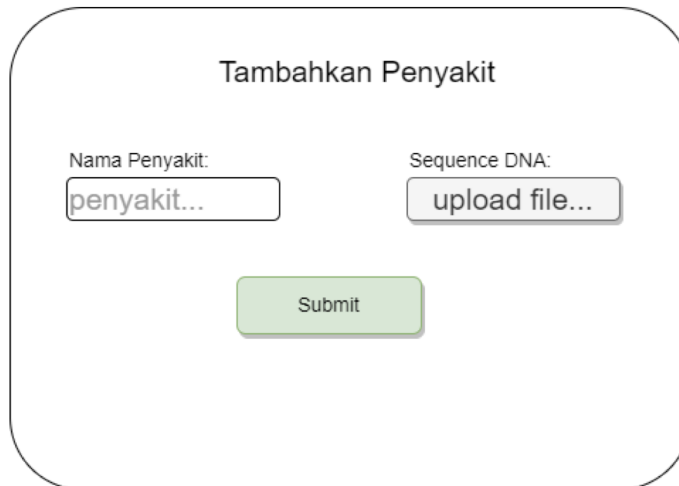
Bab 1

Deskripsi Tugas

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

Fitur-Fitur Aplikasi:

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



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

Gambar 1. Ilustrasi Input Penyakit

2. Aplikasi dapat memprediksi seseorang menderita penyakit tertentu berdasarkan sequence DNA-nya.

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

Tes DNA

Nama Pengguna:

Sequence DNA:

Prediksi Penyakit:

Hasil Tes

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

Gambar 2. 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

13 April 2022 HIV

1. 13 April 2022 - Fulan - HIV - True.

2. 13 April 2022 - Kamal - HIV - False.

3. 13 April 2022 - Entah - HIV - False.

4. 13 April 2022 - Jamal - HIV - True.

5. 13 April 2022 - Yubai - HIV - True.

6. 13 April 2022 - Hika - HIV - False.

Gambar 3. Ilustrasi Interaksi 1

II. Masukan hanya tanggal

13 April 2022

1. 13 April 2022 - Fulan - Diabetes - True.

2. 13 April 2022 - Kamal - Sinusitis - False.

3. 13 April 2022 - Entah - Down Syndrome - False.

4. 13 April 2022 - Jamal - Polio - True.

5. 13 April 2022 - Yubai - TBC - True.

6. 13 April 2022 - Hika - Hepatitis A - False.

Gambar 4. Ilustrasi Interaksi 2

III. Masukkan hanya nama penyakit

HIV

1. 13 April 2022 - Fulan - HIV - True.

2. 14 April 2022 - Kamal - HIV - False.

3. 15 April 2022 - Entah - HIV - False.

4. 16 April 2022 - Jamal - HIV - True.

5. 17 April 2022 - Yubai - HIV - True.

6. 18 April 2022 - Hika - HIV - False.

Gambar 5. 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.
 - Contoh tampilan:

Tes DNA

Nama Pengguna:

<pengguna>

Sequence DNA:

upload file...

Prediksi Penyakit:

<penyakit>

Submit

Hasil Tes

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

Bab 2

Landasan Teori

A. Landasan Teori

1. Algoritma Knuth-Morris-Pratt

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

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

Perhitungan pergeseran pada algoritma ini adalah sebagai berikut, bila terjadi ketidakcocokan pada saat pattern sejajar dengan $teks[i..i + n - 1]$, kita bisa menganggap ketidakcocokan pertama terjadi di antara $teks[i + j]$ dan $pattern[j]$, dengan $0 < j < n$. Berarti, $teks[i..i + j - 1] = pattern[0..j - 1]$ dan $a = teks[i + j]$ tidak sama dengan $b = pattern[j]$. Ketika kita menggeser, sangat beralasan bila ada sebuah awalan v dari pattern akan sama dengan sebagian akhiran u dari sebagian teks. Sehingga kita bisa menggeser pattern agar awalan v tersebut sejajar dengan akhiran dari u .

Dengan kata lain, pencocokkan string akan berjalan secara efisien bila kita mempunyai tabel yang menentukan berapa panjang kita seharusnya menggeser seandainya terdeteksi ketidakcocokkan di karakter ke- j dari pattern. Tabel itu harus memuat $next[j]$ yang merupakan posisi karakter $pattern[j]$ setelah digeser, sehingga kita bisa menggeser pattern sebesar $j - next[j]$ relatif terhadap teks.^[2]

Secara sistematis, langkah-langkah yang dilakukan algoritma Knuth-Morris-Pratt pada saat pencocokan string:

1. Algoritma Knuth-Morris-Pratt mulai mencocokkan pattern pada awal teks.

2. Dari kiri ke kanan, algoritma ini akan mencocokkan karakter per karakter pattern dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:

- i. Karakter di pattern dan di teks yang dibandingkan tidak cocok (mismatch).
- ii. Semua karakter di pattern cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.

3. Algoritma kemudian menggeser pattern berdasarkan tabel next, lalu mengulangi langkah 2 sampai pattern berada di ujung teks.

2. Algoritma Boyer-Moore

Algoritma Boyer-Moore adalah salah satu algoritma pencarian string, dipublikasikan oleh Robert S. Boyer, dan J. Strother Moore pada tahun 1977. Algoritma ini dianggap sebagai algoritma yang paling efisien pada aplikasi umum. Tidak seperti algoritma pencarian string yang ditemukan sebelumnya, algoritma Boyer-Moore mulai mencocokkan karakter dari sebelah kanan pattern. Ide di balik algoritme ini adalah bahwa dengan memulai pencocokan karakter dari kanan, dan bukan dari kiri, maka akan lebih banyak informasi yang didapat.

Misalnya ada sebuah usaha pencocokan yang terjadi pada $teks[i..i + n - 1]$, dan anggap ketidakcocokan pertama terjadi di antara $teks[i + j]$ dan $pattern[j]$, dengan $0 < j < n$. Berarti, $teks[i + j + 1..i + n - 1] = pattern[j + 1..n - 1]$ dan $a = teks[i + j]$ tidak sama dengan $b = pattern[j]$. Jika u adalah akhiran dari pattern sebelum b dan v adalah sebuah awalan dari pattern, maka pergeseran-pergeseran yang mungkin adalah:

1. Pergeseran good-suffix yang terdiri dari menyejajarkan potongan $teks[i + j + 1..i + n - 1] = pattern[j + 1..n - 1]$ dengan kemunculannya paling kanan di pattern yang didahului oleh karakter yang berbeda dengan $pattern[j]$. Jika tidak ada potongan seperti itu, maka algoritma akan menyejajarkan akhiran v dari $teks[i + j + 1..i + n - 1]$ dengan awalan dari pattern yang sama.
2. Pergeseran bad-character yang terdiri dari menyejajarkan $teks[i + j]$ dengan kemunculan paling kanan karakter tersebut di pattern. Bila karakter tersebut tidak ada di pattern, maka pattern akan disejajarkan dengan $teks[i + n + 1]$.

Secara sistematis, langkah-langkah yang dilakukan algoritma Boyer-Moore pada saat pencocokan string adalah:

1. Algoritma Boyer-Moore mulai mencocokkan pattern pada awal teks.
2. Dari kanan ke kiri, algoritma ini akan mencocokkan karakter per karakter pattern dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:
 1. Karakter di pattern dan di teks yang dibandingkan tidak cocok (mismatch).
 2. Semua karakter di pattern cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.
3. Algoritma kemudian menggeser pattern dengan memaksimalkan nilai pergeseran good-suffix dan pergeseran bad-character, lalu mengulangi langkah 2 sampai pattern berada di ujung teks.

3. Regex

Regular Expression atau regex adalah serangkaian karakter yang mendefinisikan sebuah pola pencarian. Pola regex terdiri dari karakter sederhana, seperti `/abc/`, atau kombinasi karakter sederhana dan khusus, seperti `/ab*c/` atau `/Chapter (\\d+)\\.\\d*/`. Contoh terakhir termasuk tanda kurung, yang digunakan sebagai perangkat memori.

Pola sederhana dibangun dari karakter yang ingin ditemukan kecocokannya secara langsung. Misalnya, pola `/abc/` cocok dengan kombinasi karakter dalam string hanya ketika urutan persis "abc" terjadi (semua karakter bersama-sama dan dalam urutan itu). Pertandingan seperti itu akan berhasil dalam string "Hai, apakah Anda tahu abc Anda?" dan "Desain pesawat terbaru berevolusi dari slabcraft.". Dalam kedua kasus, kecocokannya adalah dengan substring "abc". Tidak ada kecocokan dalam string "Ambil keping" karena meskipun berisi substring "ab c", string tersebut tidak berisi substring persis "abc".

Saat pencarian kecocokan membutuhkan sesuatu yang lebih dari sekadar kecocokan langsung, seperti menemukan satu atau lebih b, atau menemukan spasi, dapat disertakan karakter khusus dalam pola. Misalnya, untuk mencocokkan satu "a" diikuti oleh nol atau lebih "b" diikuti oleh "c", akan digunakan pola `/ab*c/`: * setelah "b" berarti "0 atau lebih kemunculan barang sebelumnya." Dalam string "cbbabbbbcdebc", pola ini akan cocok dengan substring "abbbbc".

4. Algoritma *Longest Common Subsequence*

Longest Common Subsequences atau LCS adalah masalah mencari uparangkaian bersama (*common subsequence*) terpanjang dari beberapa (biasanya hanya 2) buah rangkaian. Uparangkaian (*subsequence*) dari sebuah string S adalah sekumpulan karakter yang ada pada S yang urutan kemunculannya sama. Atau dalam definisi formalnya rangkaian Z adalah uparangkaian dari $X = \langle x_1, x_2, \dots, x_m \rangle$, jika terdapat urutan menaik $\langle i_1, i_2, \dots, i_k \rangle$ yang merupakan indeks X untuk semua $j=1, 2, \dots, k$, yang memenuhi $x_{i_j} = z_j$.

Misal pada $S = \text{GAATACA}$, beberapa uparangkaian yang mungkin adalah : GAT, TCA, dan GC. Uparangkaian bersama (*common subsequence*) dari 2 rangkaian adalah uparangkaian yang terdapat pada kedua rangkaian tersebut. Misal $S_1 = \text{GATTTAC}$ dan $S_2 = \text{AAGATC}$, maka GAT, GATC, maupun GAC adalah uparangkaian bersama dari S_1 dan S_2 . Uparangkaian bersama terpanjang (*longest common subsequences*) adalah uparangkaian bersama dari 2 rangkaian yang paling panjang.

Algoritma untuk masalah LCS ini berdasarkan teorema di bawah ini:

Misal $X = \langle x_1, x_2, \dots, x_m \rangle$ dan $Y = \langle y_1, y_2, \dots, y_n \rangle$ adalah rangkaian dan $Z = \langle z_1, z_2, \dots, z_k \rangle$ adalah suatu LCS dari X dan Y .

1. Jika $x_m = y_n$, maka $z_k = x_m = y_n$ dan Z_{k-1} adalah suatu LCS dari X_{m-1} dan Y_{n-1} .
2. Jika $x_m \neq y_n$ maka $z_k \neq x_m$ mengimplikasikan bahwa Z adalah suatu LCS dari X_{m-1} dan Y .
3. Jika $x_m \neq y_n$ maka $z_k \neq y_n$ mengimplikasikan bahwa Z adalah suatu LCS dari X dan Y_{n-1} .

B. Aplikasi Web MERN Stack

MERN Stack merupakan istilah dari arsitektur *full-stack* untuk membuat suatu aplikasi web. MERN Stack terdiri dari *frontend* berbasis React.js, *backend* berbasis Express.js & Node.js, dan basis data berbasis MongoDB. MERN Stack biasanya digunakan untuk aplikasi web yang “JSON-Heavy”, berbasis cloud, dan juga memiliki *interface dynamic*. Sehingga cocok dengan aplikasi kami.

1. React.Js

Tingkat teratas dari MERN Stack adalah React.js, yang merupakan *framework* JavaScript deklaratif untuk membuat aplikasi sisi klien yang *dynamic* dalam HTML. React memungkinkan untuk membangun antarmuka yang

kompleks melalui Komponen sederhana, menghubungkannya ke data di server *backend*, dan merendernya sebagai HTML.

Kuatnya React menangani *interface* yang berbasis data dengan kompleksitas kode minimal dan memiliki semua kakas yang diharapkan dari *framework* web modern: dukungan kuat untuk *form*, *error handling*, *events*, *lists*, dan banyak lagi.

2. Express.js dan Node.js

Tingkat selanjutnya adalah framework sisi server Express.js, yang berjalan di dalam server Node.js. Express.js disebut sebagai “kerangka web minimalis yang cepat, tidak beropini untuk Node.js”, dan memang seperti itulah adanya. Express.js memiliki model yang kuat untuk perutean URL (mencocokkan URL masuk dengan fungsi server), dan menangani permintaan dan tanggapan HTTP.

Dengan membuat XML HTTP Requests (XHRs) atau GET atau POST dari front-end React.js, sehingga dapat terhubung ke fungsi Express.js yang mendukung aplikasi. Fungsi-fungsi tersebut pada gilirannya menggunakan driver Node.js MongoDB, baik melalui panggilan balik untuk menggunakan Promises, untuk mengakses dan memperbarui data di database MongoDB.

3. MongoDB

MongoDB merupakan suatu program basis data yang berbasis dokumen seperti JSON. Keunggulan MongoDB ada jika aplikasi menyimpan data apa pun (profil pengguna, konten, komentar, unggahan, acara, dll.), maka disitu akan menginginkan database yang mudah digunakan seperti React, Express, dan Node.js.

Di situlah MongoDB masuk: Dokumen JSON yang dibuat di front end React.js aplikasi dapat dikirim ke server Express.js, di mana mereka dapat diproses dan (dengan asumsi mereka valid) disimpan langsung di MongoDB untuk pengambilan nanti.

Bab 3

Analisis Pemecahan Masalah

A. Langkah Penyelesaian Masalah

1. Input Penyakit Baru

Program akan meminta input berupa nama penyakit dan file txt berisi sequence DNA penyakit. Kemudian, akan dilakukan validasi terhadap kedua input yang telah diterima. Untuk nama penyakit, akan dilakukan pencarian pada database apakah input nama yang dimasukkan belum ada di database. Apabila nama sudah ada, maka input akan ditolak. Untuk file txt berisi sequence DNA, akan dilakukan pengecekan kevalidan file. File dianggap valid apabila tidak kosong dan hanya mengandung huruf A, C, G, dan T dengan format huruf kapital. Apabila kedua inputan valid, maka kedua input akan dimasukkan ke dalam database.

2. Prediksi Penyakit

Program akan meminta input berupa nama pengguna, prediksi penyakit, file txt berisi sequence DNA pengguna, dan pilihan algoritma antara boyer moore atau kmp. Kemudian, akan dilakukan validasi terhadap input yang telah diterima. Untuk prediksi penyakit, input dianggap valid apabila nama penyakit sudah ada di dalam database. Untuk file txt berisi sequence DNA, akan dilakukan pengecekan kevalidan file. File dianggap valid apabila tidak kosong dan hanya mengandung huruf A, C, G, dan T dengan format huruf kapital. Kemudian, dilakukan string matching antara sequence DNA penyakit pada database dan file txt dari pengguna dengan menggunakan algoritma yang sudah dipilih. Dihitung juga nilai similarity antara kedua sequence DNA dengan menggunakan algoritma LCS. Kemudian, ditampilkan hasil dari pencocokan kedua sequence DNA. Apabila nilai similarity dibawah 80%, maka akan dikembalikan nilai false yang berarti kedua sequence DNA dinilai kurang cocok.

3. Laman Pencarian

Program akan meminta input yang dapat berupa tanggal, nama penyakit, atau tanggal+nama penyakit. Akan dilakukan pembacaan input dengan menggunakan regex. Input tanggal dapat diterima apabila sesuai format berikut :

- a. dd Month yyyy, month dalam huruf, dd dan yyyy tidak ada awalan 0, yyyy dibatasi 2022
- b. dd Month yyyy, month dalam huruf, dd dan yyyy dengan awalan 0, yyyy dibatasi 2022 dan tidak ada tahun 0000
- c. dd mm yyyy, dd, mm, dan yyyy tidak ada awalan 0
- d. dd mm yyyy, dd, mm, dan yyyy dengan awalan 0
- e. dd-mm-yyyy, dd, mm, dan yyyy tidak ada awalan 0
- f. dd-mm-yyyy, dd, mm, dan yyyy dengan awalan 0
- g. dd/mm/yyyy, dd, mm, dan yyyy tidak ada awalan 0
- h. dd/mm/yyyy, dd, mm, dan, yyyy dengan awalan 0

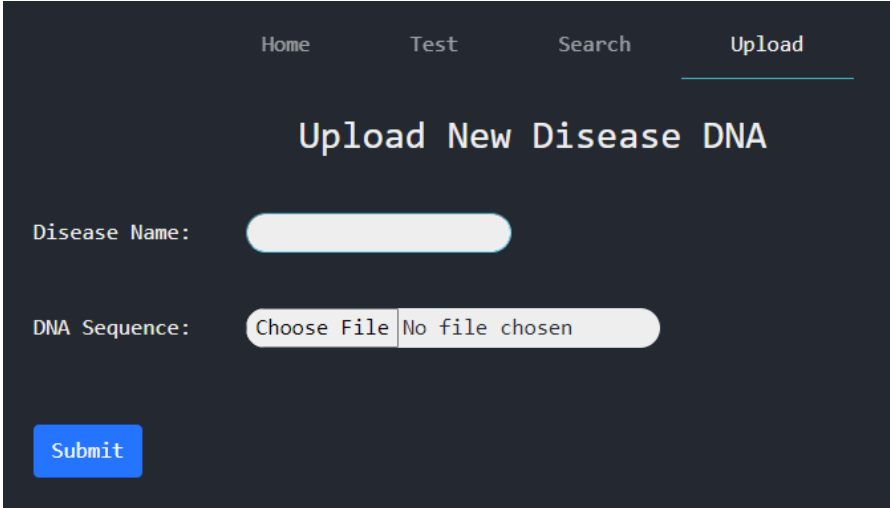
Hasil input yang telah dibaca akan dimapping agar didapatkan data dengan tanggal dan nama penyakit yang sesuai pada database

B. Fitur Fungsional dan Arsitektur Aplikasi

Aplikasi ini terdiri dari 4 halaman, yaitu halaman utama, halaman penambahan penyakit, halaman mengetes dna, dan halaman melihat riwayat pengetesan dna.

1. Halaman penambahan penyakit (/penyakit)


Pada halaman ini, pengguna akan diminta memasukkan nama penyakit dan DNA dalam bentuk file txt yang sesuai. Penyakit akan divalidasi unik dan DNA akan divalidasi dengan regex.



The screenshot shows a web interface with a dark blue background. At the top, there is a navigation bar with four links: 'Home', 'Test', 'Search', and 'Upload'. The 'Upload' link is highlighted with a red underline. Below the navigation bar, the title 'Upload New Disease DNA' is centered in a white, monospace-style font. The form contains two input fields: 'Disease Name:' followed by a white rounded rectangular text input, and 'DNA Sequence:' followed by a file upload button labeled 'Choose File' and a status indicator 'No file chosen'. At the bottom left of the form is a red 'Submit' button.

2. Halaman pengetesan DNA / pengetesan penyakit (/prediksi)

Pada halaman ini, pengguna akan diminta untuk memasukkan nama pengguna, DNA yang ingin dites, penyakit prediksi, dan metode pengujian yang ingin digunakan.



The screenshot shows a web interface with a dark blue background. At the top, there is a navigation bar with four links: 'Home', 'Test', 'Search', and 'Upload'. The 'Test' link is highlighted with a red underline. Below the navigation bar, the title 'Check Your DNA for Disease' is centered in a white, monospace-style font. The form contains four input fields: 'Your Name:' followed by a white rounded rectangular text input, 'Your DNA Sequence:' followed by a file upload button labeled 'Choose File' and a status indicator 'No file chosen', 'Disease Name:' followed by a white rounded rectangular text input, and 'Searching Algorithm:' followed by a dropdown menu currently showing 'KMP'. At the bottom left of the form is a red 'Submit' button.

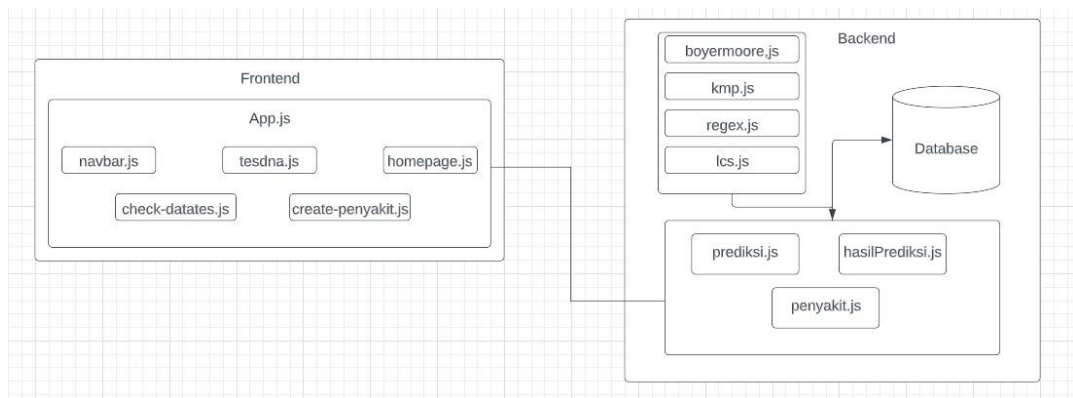
3. Halaman pencarian riwayat pengecekan DNA (/hasil)

Pada halaman ini, pengguna akan diminta memasukkan query untuk dicek. Query dapat dipisahkan menjadi tanggal dan nama penyakit atau salah satunya

saja. Pemisahan menggunakan regex. Format tanggal yang bisa diterima dapat dilihat pada Bab 3.

The screenshot shows a web application interface with a dark blue background. At the top, there is a navigation bar with four links: 'Home', 'Test', 'Search' (which is underlined), and 'Upload'. Below the navigation bar, the main heading is 'Search for Tests' in a large, white, monospace-style font. Underneath the heading is a prompt 'Enter a date and/or disease name:' followed by a large, light gray, rounded rectangular input field. At the bottom center of the form is a blue button with the text 'Submit' in white.

4. Arsitektur Program



Bab 4

Implementasi dan Pengujian

A. Spesifikasi Teknis Program

1. Struktur Data

Dalam pengerjaan tugas besar ini, kami memanfaatkan MongoDB sebagai database nya. Dalam database kami, kami menggunakan 2 buah schema MongoDB yaitu penyakit dan prediksi. Berikut adalah spesifikasi tiap schema tersebut.

a. Penyakit

Penyakit terdiri dari 2 atribut yaitu

1. penyakit, dengan tipe String, berisi nama penyakit. Atribut ini harus unik untuk setiap data yang dimasukkan/
2. dnaString, dengan tipe String, berisi dna dari penyakit. Atribut ini telah divalidasi dan dijamin memenuhi syarat dari DNA.

b. Prediksi

Prediksi terdiri dari 5 atribut yaitu

1. tanggalPrediksi, dengan tipe String, berisi tanggal prediksi penyakit dilakukan.
2. namaPasien, dengan tipe String, berisi nama dari pasien yang diuji dna nya
3. penyakitPasien, dengan tipe String, berisi penyakit yang diprediksi.
4. Status, dengan tipe String, berisi True/False yang menunjukkan apakah prediksi berhasil.
5. tingkatKemiripan, dengan tipe String, yang berisi nilai kemiripan dna yang dihitung dengan menggunakan LCS.

Selain atribut-atribut tersebut, dalam setiap schema, juga ada atribut bawaan dari MongoDB, yaitu Id (dengan tipe ObjectID, akan selalu unik), createdAt(dengan tipe Date, berisi tanggal data dimasukkan), dan updatedAt (dengan tipe Date, berisi tanggal data terakhir diubah).

2. Fungsi dan Prosedur Program

Berikut ini beberapa algoritma dan fungsi-fungsi utama yang kami gunakan

1. Fungsi String Matching dengan Boyer-Moore

```
function buildLast(pattern) {
  var m = pattern.length;
  var last = {
    A: -1,
    C: -1,
    G: -1,
    T: -1
  };
};

for (var i = 0; i < m; i++) {
  last[pattern[i]] = i;
}
return last;
}

module.exports = function bm(text, pattern) {
  var n = text.length;
  var m = pattern.length;
  var i = m-1;
  var j = m-1;
  var last = buildLast(pattern);

  if (i > n-1) {
    return -1;
  }
  do {
    if (pattern[j] == text[i]) {
      if (j == 0) {
        return true;
      }
      i--;
      j--;
    }
    else {
      i += i + m - Math.min(j, 1+last[text[i]]);
      j = m-1;
    }
  } while (i <= n-1);
  return false;
}
```

2. Fungsi String Matching dengan KMP

```
function computeFail(pattern) {  
    var m = pattern.length;  
    var fail = [];  
    var j = 0;  
    var i = 1;  
    fail[0] = 0;  
    while (i < m) {  
        if (pattern[j] == pattern[i]) {  
            fail[i] = j + 1;  
            i++;  
            j++;  
        }  
        else if (j > 0) {  
            j = fail[j - 1];  
        }  
        else {  
            fail[i] = 0;  
            i++;  
        }  
    }  
    return fail;  
}
```

```
module.exports = function kmp(text, pattern) {  
  var n = text.length;  
  var m = pattern.length;  
  var i = 0;  
  var j = 0;  
  var fail = computeFail(pattern);  
  while (i < n) {  
    if (pattern[j] == text[i]) {  
      if (j == m - 1) {  
        return true;  
      }  
      i++;  
      j++;  
    }  
    else if (j > 0) {  
      j = fail[j - 1];  
    }  
    else {  
      i++;  
    }  
  }  
  return false;  
}
```

3. Fungsi untuk melakukan perhitungan kemiripan dengan LCS

```
/**
 * Function untuk menghitung lcs dari string a dan b
 * @param {String} a
 * @param {String} b
 * @returns float yang menyatakan tingkat kemiripan
 */
module.exports = function lcs(a, b) {
  var aLength = a.length;
  var bLength = b.length;

  var dp = new Array(aLength + 1);

  for (let i = 0; i <= aLength + 1; i++) {
    dp[i] = new Array(bLength + 1);
  }

  for (let i = 0; i <= aLength; i++) {
    for (let j = 0; j <= bLength; j++) {
      if (i == 0 || j == 0) {
        dp[i][j] = 0;
      } else if (a[i - 1] == b[j - 1]) {
        dp[i][j] = dp[i - 1][j - 1] + 1;
      } else {
        dp[i][j] = Math.max(dp[i - 1][j], dp[i][j - 1]);
      }
    }
  }

  return (dp[aLength][bLength] / Math.min(aLength, bLength)) * 100;
}
```

4. Fungsi untuk memvalidasi DNA

```
function dnaMatching(dna){
  const pattern = /^[ACGT]/;

  var succedd = pattern.test(dna);
  return succedd;
}
/**
```

5. Fungsi untuk memisahkan tanggal

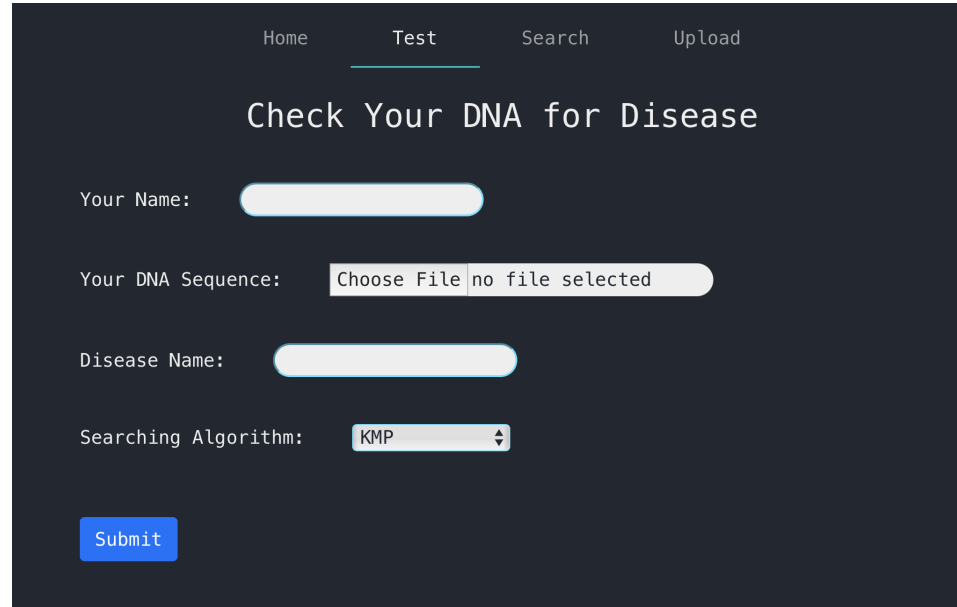
```
function matchTanggal(input){  
  /**  
   * Format Tanggal yang diterima  
   * 1. dd Month yyyy, month dalam huruf, dd dan yyyy tidak ada awalan 0, yyyy dibatasi 2022  
   * 2. dd Month yyyy, month dalam huruf, dd dan yyyy dengan awalan 0, yyyy dibatasi 2022 dan tidak ada tahun 0000  
   * 3. dd mm yyyy, dd, mm, dan yyyy tidak ada awalan 0  
   * 4. dd mm yyyy, dd, mm, dan yyyy dengan awalan 0  
   * 5. dd-mm-yyyy, dd, mm, dan yyyy tidak ada awalan 0  
   * 6. dd-mm-yyyy, dd, mm, dan yyyy dengan awalan 0  
   * 7. dd/mm/yyyy, dd, mm, dan yyyy tidak ada awalan 0  
   * 8. dd/mm/yyyy, dd, mm, dan yyyy dengan awalan 0  
   */  
  console.log(input)  
  const allPattern = [  
    /^(3[0-1])|(12)d?([4-9]) (Januari|maret|mei|juli|agustus|oktober|desember) (202[0-2]|20[0-1][0-9]|1[0-9](3)|1[0-9](1,2)[0-9])/gi,  
    /^(302|[12]d?)([4-9]) (februari|april|juni|september|november) (202[0-2]|20[0-1][0-9]|1[0-9](1,2)[0-9])/gi ,  
    /^(3[0-1])|(01-9)|1[0-9]|2[0-9]) (Januari|maret|mei|juli|agustus|oktober|desember) (202[0-2]|20[0-1][0-9]|1[0-9](3)|000[1-9]|00[1-9]|0[1-9](1,2)[0-9])/gi,  
    /^(30[01-9]|1[0-9]|2[0-9]) (februari|april|juni|september|november) (202[0-2]|20[0-1][0-9]|1[0-9](3)|000[1-9]|00[1-9]|0[1-9](1,2)[0-9])/gi,  
    /^(3[0-1])|(12)d?([4-9]) (1[3578]10|12) (202[0-2]|20[0-1][0-9]|1[0-9](3)|1[0-9](1,3))/gi,  
    /^(302|[12]d?)([4-9]) (2[469]11) (202[0-2]|20[0-1][0-9]|1[0-9](3)|1[0-9](1,3))/gi,  
    /^(3[0-1])|(01-9)|1[0-9]|2[0-9]) (0[13578]10|12) (202[0-2]|20[0-1][0-9]|1[0-9](3)|000[1-9]|00[1-9]|0[1-9](1,2)[0-9])/gi,  
    /^(30[01-9]|1[0-9]|2[0-9]) (0[2469]11) (202[0-2]|20[0-1][0-9]|1[0-9](3)|000[1-9]|00[1-9]|0[1-9](1,2)[0-9])/gi,  
    /^(3[0-1])|(12)d?([4-9])-(1[3578]10|12)-(202[0-2]|20[0-1][0-9]|1[0-9](3)|1[0-9](1,3))/gi,  
    /^(302|[12]d?)([4-9])-(2[469]11)-(202[0-2]|20[0-1][0-9]|1[0-9](3)|1[0-9](1,3))/gi,  
    /^(3[0-1])|(01-9)|1[0-9]|2[0-9])-(0[13578]10|12)-(202[0-2]|20[0-1][0-9]|1[0-9](3)|000[1-9]|00[1-9]|0[1-9](1,2)[0-9])/gi,  
    /^(30[01-9]|1[0-9]|2[0-9])-(0[2469]11)-(202[0-2]|20[0-1][0-9]|1[0-9](3)|000[1-9]|00[1-9]|0[1-9](1,2)[0-9])/gi,  
    /^(3[0-1])|(12)d?([4-9])√(1[3578]10|12)√(202[0-2]|20[0-1][0-9]|1[0-9](3)|1[0-9](1,3))/gi,  
    /^(302|[12]d?)([4-9])√(2[469]11)√(202[0-2]|20[0-1][0-9]|1[0-9](3)|1[0-9](1,3))/gi,  
    /^(3[0-1])|(01-9)|1[0-9]|2[0-9])√(0[13578]10|12)√(202[0-2]|20[0-1][0-9]|1[0-9](3)|000[1-9]|00[1-9]|0[1-9](1,2)[0-9])/gi,  
    /^(30[01-9]|1[0-9]|2[0-9])√(0[2469]11)√(202[0-2]|20[0-1][0-9]|1[0-9](3)|000[1-9]|00[1-9]|0[1-9](1,2)[0-9])/gi,  
  ];  
  
  for(let i = 0; i <= 15; i++){  
    if(input.match(allPattern[i]) != null){  
      // console.log(i);  
      return convertString(input.match(allPattern[i])[0], i);  
    }  
  }  
  return null;  
}
```

6. Fungsi untuk memisahkan nama penyakit

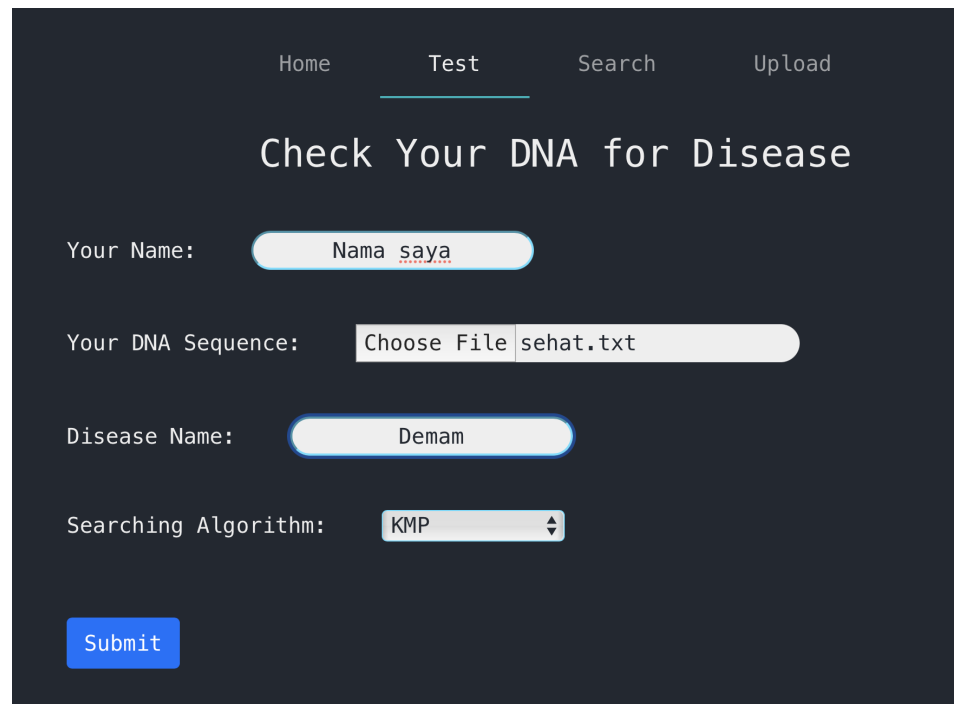
```
function matchPenyakit(input){  
  const pattern = /[a-z ]+$/gi;  
  
  return input.match(pattern);  
}
```

B. Tata Cara Penggunaan Program

1. Pencocokan DNA penyakit dengan DNA pengguna
 - a. Buka Laman Test pada aplikasi (/prediksi)

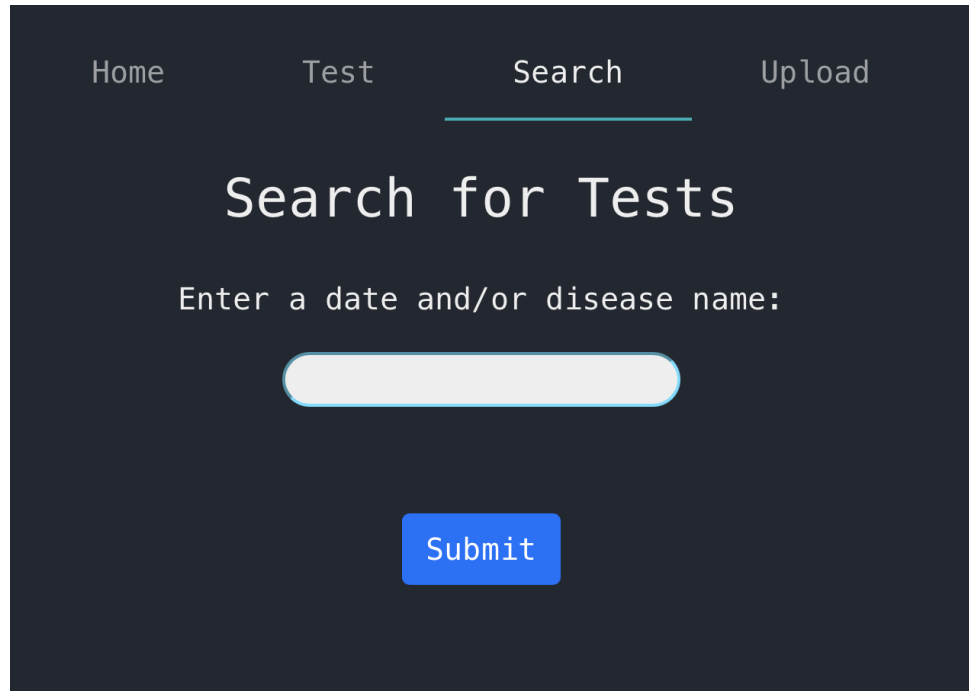


- b. Masukkan nama pengguna, file DNA pengguna (dalam format .txt), nama penyakit yang terdaftar, dan algoritma pencarian



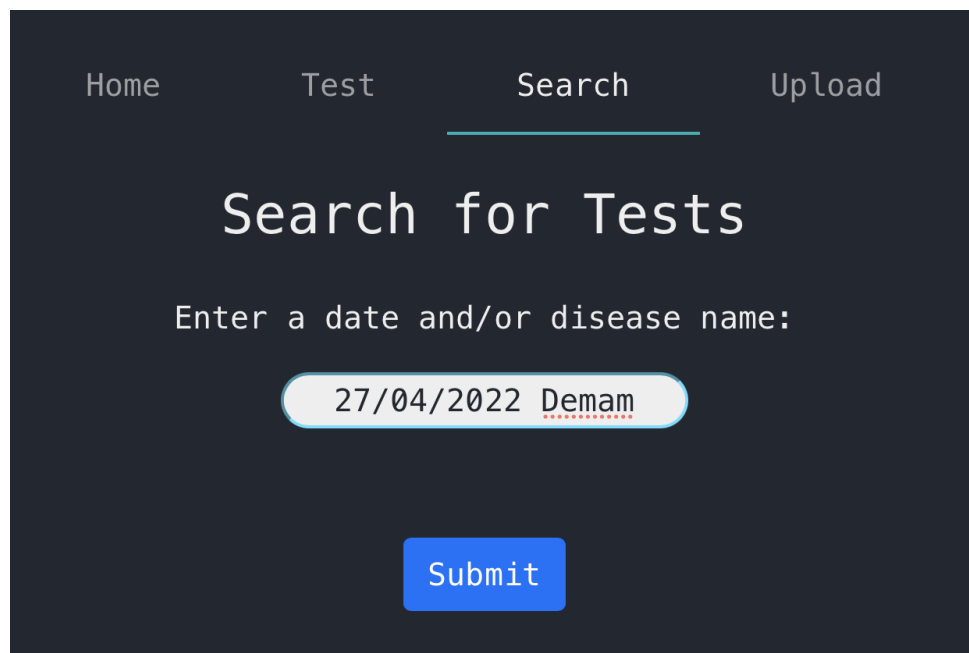
- c. Klik tombol submit untuk mendapatkan hasil dari pencocokan string (Hasil dari pengujian ada di subbab selanjutnya)
- d. Untuk melakukan tes DNA ulang, jangan lupa mereload laman.

2. Mencari riwayat pengecekan DNA
 - a. Masuk ke laman Search (/hasil)



The screenshot shows a dark-themed web interface with a navigation bar at the top containing 'Home', 'Test', 'Search' (highlighted with a teal underline), and 'Upload'. Below the navigation bar, the heading 'Search for Tests' is centered. Underneath the heading is the prompt 'Enter a date and/or disease name:'. A light gray, rounded rectangular input field is positioned below the prompt. At the bottom of the form is a blue button with the text 'Submit'.

- b. Masukkan tanggal dan/atau nama penyakit pada form
(Format tanggal bebas dan tambahkan spasi jika ingin mencari tanggal dan nama penyakit)



This screenshot shows the same 'Search for Tests' form as the previous one, but with the input field filled with the text '27/04/2022 Demam'. The text is split, with the date '27/04/2022' on the left and the disease name 'Demam' on the right. The 'Submit' button remains at the bottom.

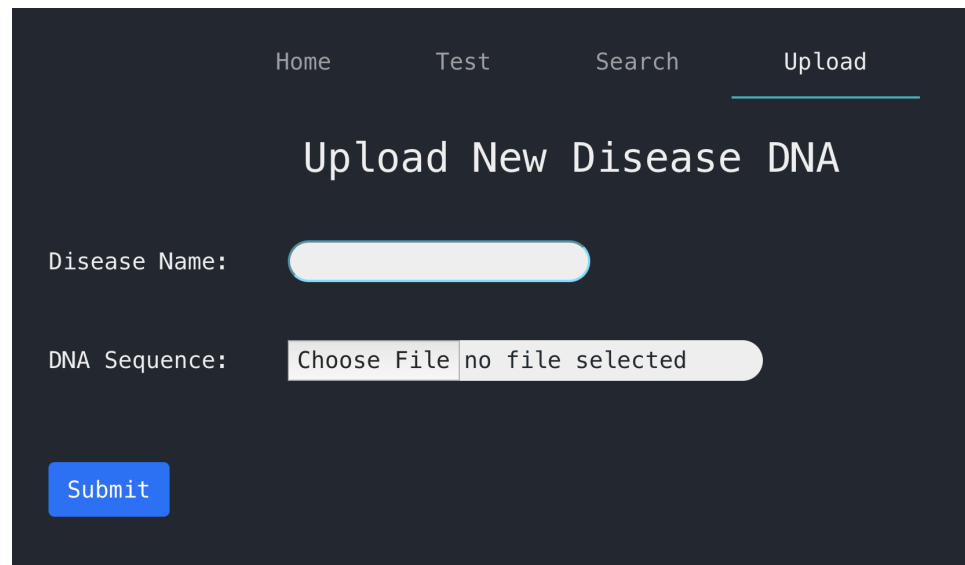
- c. Hasil pencarian akan muncul di bawah tombol submit

(Hasil dari pencarian ada di subbab selanjutnya)

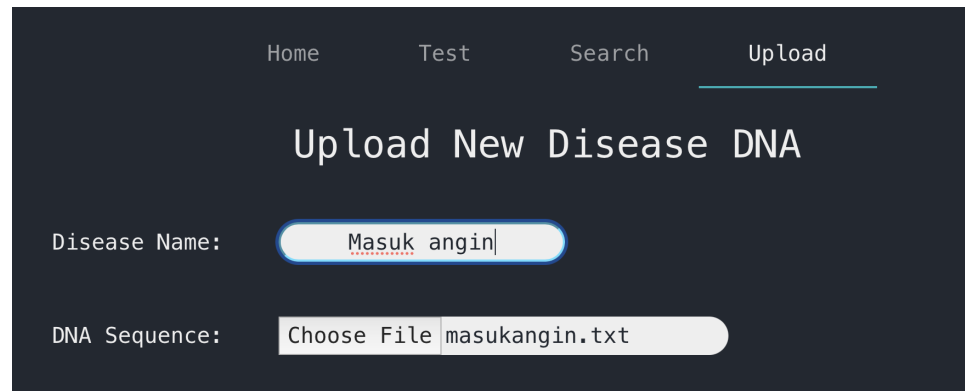
- d. Untuk melakukan pencarian ulang, jangan lupa reload laman.

3. Menambah penyakit baru

- a. Masuk ke laman Upload (/penyakit)



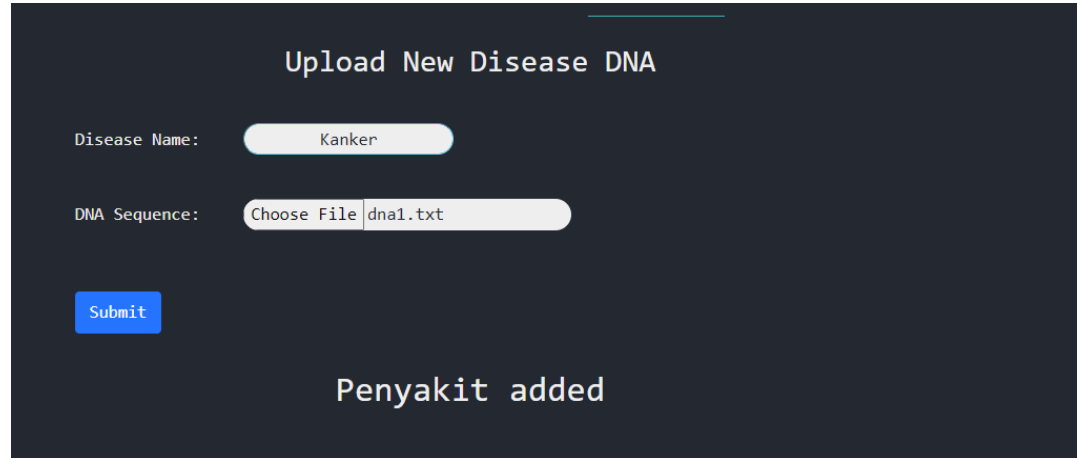
- b. Masukkan nama penyakit baru dan juga file sekuens DNA (dalam format .txt)



- c. Klik tombol submit
(Hasil dari pengujian ada di subbab selanjutnya)

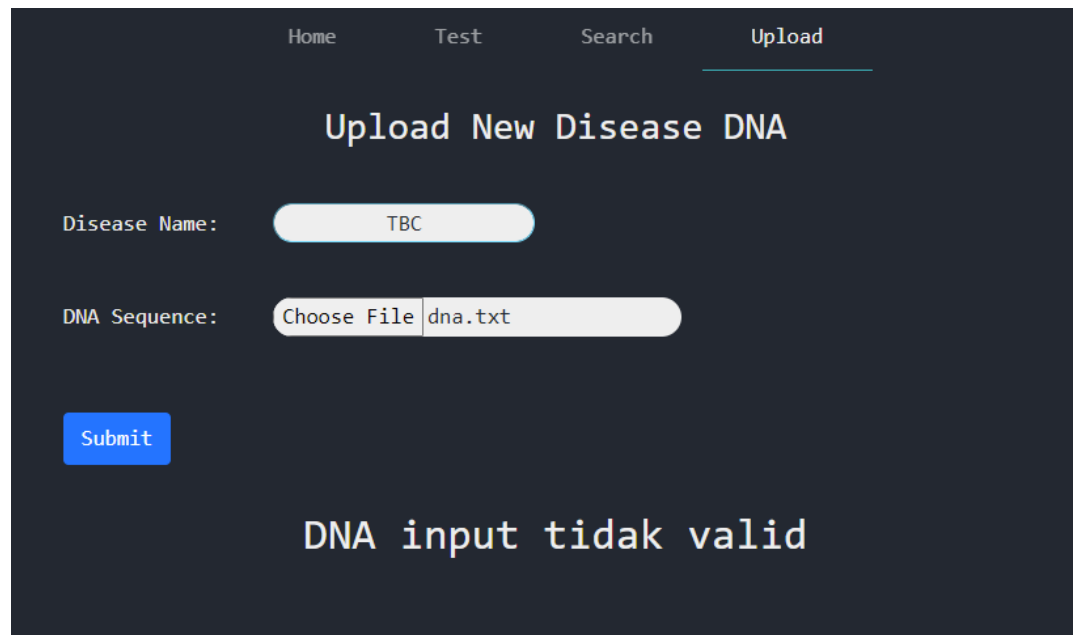
C. Hasil Pengujian

1. Penambahan penyakit Baru dengan DNA Valid



The screenshot shows a web interface with a dark background. At the top, there is a navigation bar with the word 'Upload' highlighted. Below it, the title 'Upload New Disease DNA' is centered. The form contains two input fields: 'Disease Name' with the value 'Kanker' and 'DNA Sequence' with a file selection button 'Choose File' and the filename 'dna1.txt'. A blue 'Submit' button is located below the inputs. At the bottom, a message 'Penyakit added' is displayed in a light green font.

2. Penambahan Penyakit baru dengan DNA tidak valid



The screenshot shows the same web interface as the previous one, but with the 'Disease Name' field set to 'TBC'. The 'DNA Sequence' field still shows 'Choose File' and 'dna.txt'. The blue 'Submit' button is present. At the bottom, a message 'DNA input tidak valid' is displayed in a light green font, indicating an error.

3. Pengujian untuk Test Penyakit yang gagal

Home Test Search Upload

Check Your DNA for Disease

Your Name: Vincent

Your DNA Sequence: Choose File dna.txt

Disease Name: demam

Searching Algorithm: KMP

Submit

Hasil Tes

Tanggal : 2022-04-29

Pengguna : Vincent

Penyakit : Demam

Similarity : 0%

True/False : False

4. Pengujian untuk Test Penyakit yang berhasil

Home Test Search Upload

Check Your DNA for Disease

Your Name: Reja

Your DNA Sequence: Choose File dna1.txt

Disease Name: Kanker

Searching Algorithm: KMP

Submit

Hasil Tes

Tanggal : 2022-04-29

Pengguna : Reja

Penyakit : Kanker

Similarity : 100%

True/False : True

5. Pengujian pencarian riwayat prediksi penyakit

Tanggal Prediksi	Pasien	Penyakit	Similarity	Hasil Tes
2022-04-29	Reja	Kanker	100%	True

D. Analisis Hasil Pengujian

Berdasarkan data pengujian, semua fitur yang diimplementasikan sudah bekerja dengan baik dan sesuai spesifikasi. Selain itu, terdapat juga poin bonus pengecekan kemiripan. Poin bonus deployment ke hosting remote juga telah diimplementasikan dan berjalan tanpa masalah.

Bab 5

Kesimpulan dan Saran

A. Kesimpulan

Dalam pengerjaan tugas ini, kelompok kami menerapkan menerapkan beberapa algoritma string matching seperti boyer-moore dan KMP, LCS, dan Regex. Dapat disimpulkan bahwa algoritma berikut cukup efisien dalam menyelesaikan masalah DNA Testing. Kelompok kami juga berhasil menggunakan struktur MERN dalam membuat website. Struktur ini sangat memudahkan kami dalam membuat website tersebut.

B. Saran

Dalam pengerjaan proyek ini, walaupun website sudah berjalan dengan baik dan berhasil di-deploy, website masih bisa dikembangkan lebih jauh lagi. Tampilan website bisa dibuat lebih menarik lagi. Jenis format tanggal dan nama penyakit yang dimasukkan pada halaman search dengan menggunakan algoritma regex bisa lebih didiversifikasi.

C. Refleksi

Proses pengerjaan tugas besar ini secara umum sudah cukup baik. Tiap anggota kelompok mengerjakan tugasnya masing-masing dengan baik dan sesuai dengan waktu yang ditentukan. Namun, mungkin di kesempatan berikutnya deadlinenya jangan berdekatan dengan tugas besar yang lain.

Link Repository dan Deploy

[Link Repository](#)

[Link Hasil Deploy](#)

Daftar Pustaka

https://id.wikipedia.org/wiki/Algoritma_Knuth-Morris-Pratt

https://id.wikipedia.org/wiki/Algoritma_Boyer-Moore

https://id.wikipedia.org/wiki/Ekspresi_reguler

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf>

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions#creating_a_regular_expression

<https://www.mongodb.com/mern-stack>