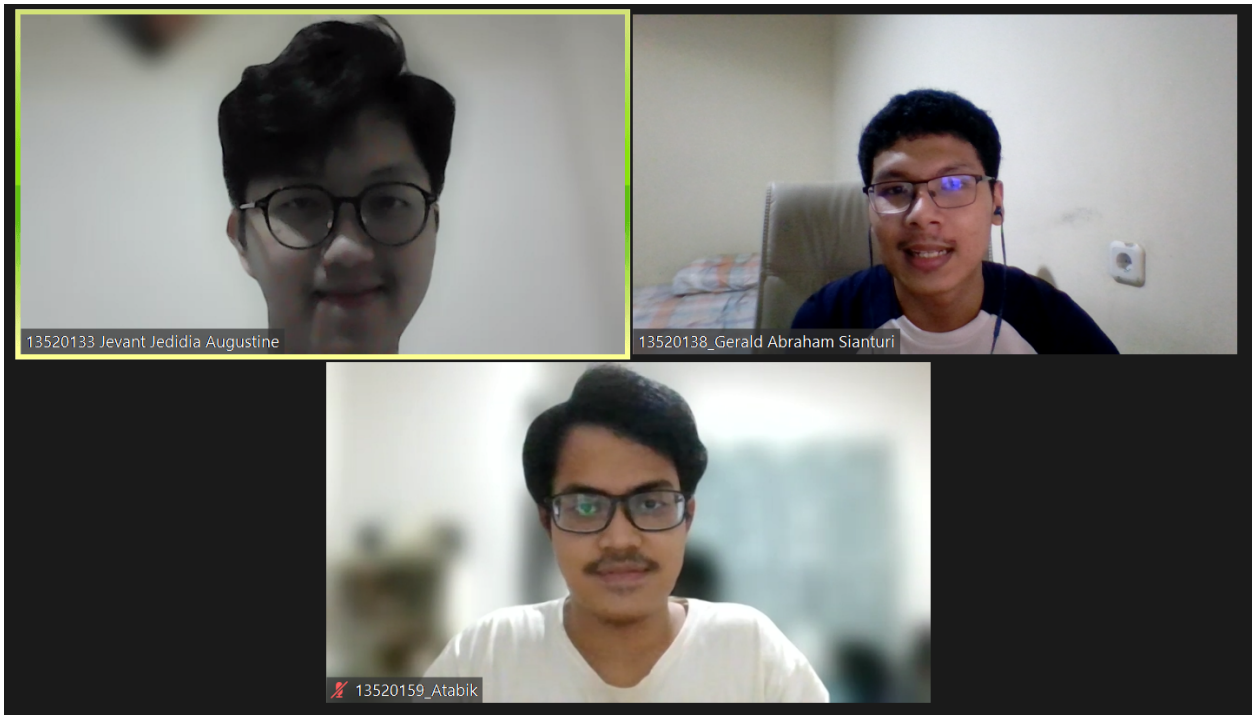
Laporan Tugas Besar 3
IF2211 Strategi Algoritma
Penerapan *String Matching* dan *Regular Expression* dalam DNA *Pattern Matching*

Semester II Tahun 2021/2022



Disusun oleh:

Kelompok 39 - RNAnotDNA

Jevant Jedidia Augustine	13520133
Gerald Abraham Sianturi	13520138
Atabik Muhammad Azfa Shofi	13520159

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2022

Daftar Isi

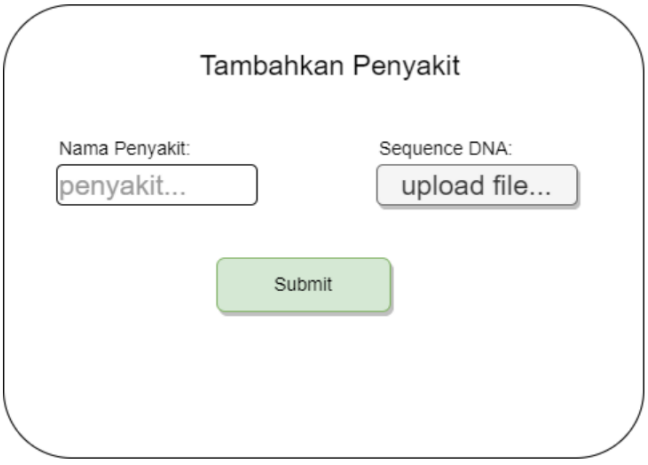
Bab 1

Deskripsi Tugas

Dalam tugas besar ini, anda diminta untuk membangun sebuah aplikasi *DNA Pattern Matchin*g. 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.

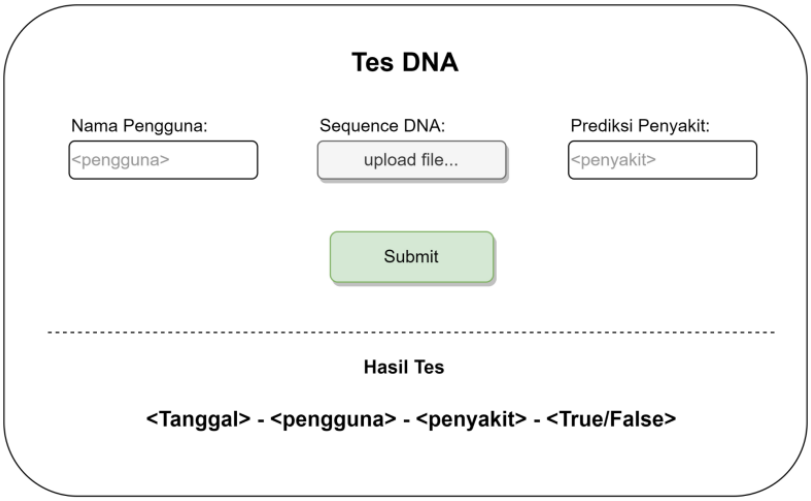
Fitur-Fitur Aplikasi:

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:

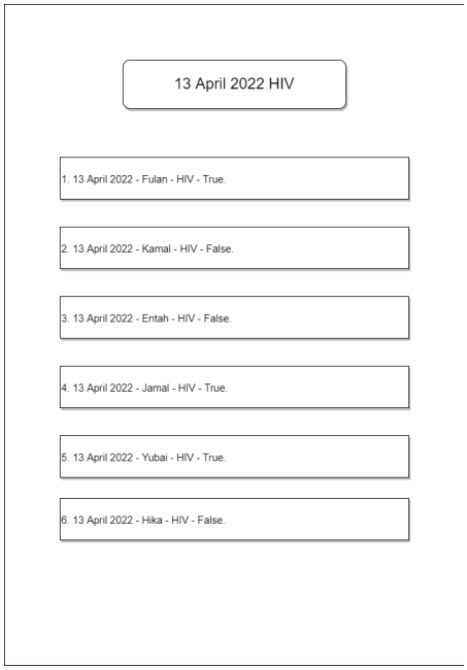


2. Aplikasi dapat memprediksi seseorang menderita penyakit tertentu berdasarkan *sequence* DNA-nya.
    a. Tes DNA dilakukan dengan menerima input nama pengguna, *sequence* DNA pengguna, dan nama penyakit yang diuji. Asumsi *sequence* DNA pengguna > *sequence* DNA penyakit.
    b. Dilakukan sanitasi *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).

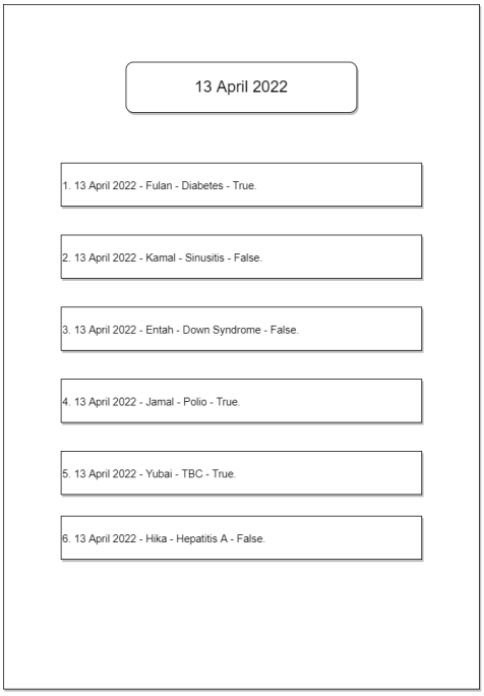
c. Pencocokan *sequence* DNA dilakukan dengan menggunakan algoritma **string matching**.
d. Hasil dari tes DNA berupa tanggal tes, nama pengguna, nama penyakit yang diuji, dan status hasil tes. **Contoh: 1 April 2022 - Mhs IF - HIV - False**
e. Semua komponen hasil tes ini dapat ditampilkan pada halaman web (*refer* ke poin 3 pada "Fitur-Fitur Aplikasi") dan disimpan pada sebuah tabel *database*.
f. Contoh tampilan web:



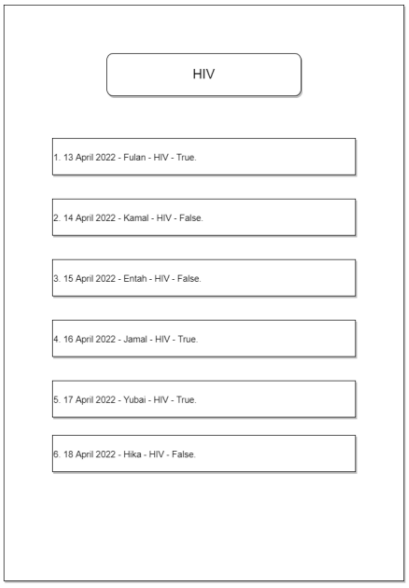
3. Aplikasi memiliki halaman yang menampilkan urutan hasil prediksi dengan kolom pencarian di dalamnya. Kolom pencarian bekerja sebagai filter dalam menampilkan hasil.
    a. Kolom pencarian dapat menerima masukan dengan struktur: <tanggal_prediksi><spasi><nama_penyakit>, contoh "13 April 2022 HIV". **Format penanggalan dibebaska**n, 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



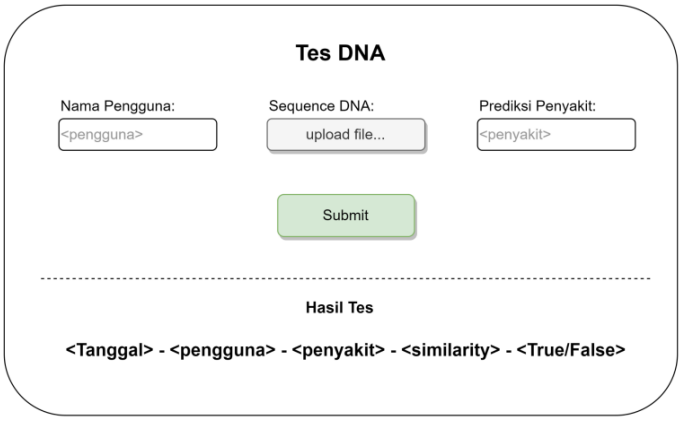
ii. Masukan hanya tanggal



iii. Masukan hanya nama penyakit



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 pengecekkan *string matching* terlebih dahulu.
    d. Contoh tampilan:



Spesifikasi Program:

1. Aplikasi berbasis website dengan pembagian Frontend dan Backend yang jelas.

2. Implementasi Backend wajib menggunakan Node.js / Golang, sedangkan Frontend disarankan untuk menggunakan React / Next.js / Vue / Angular. Lihat referensi untuk selengkapnya
3. Penyimpanan data wajib menggunakan basis data (MySQL / PostgreSQL / MongoDB)
4. Algoritma pencocokan string (KMP dan Boyer-Moore) wajib diimplementasikan pada sisi Backend aplikasi.
5. Informasi yang wajib disimpan pada basis data:
    a. Jenis Penyakit:
        - Nama penyakit
        - Rantai DNA penyusun.
    b. Hasil Prediksi:
        - Tanggal prediksi
        - Nama pasien
        - Penyakit prediksi
        - Status terprediksi.
6. Jika mengerjakan bonus tingkat kemiripan DNA, simpan hasil tingkat kemiripan tersebut pada basis data.

Bab 2
Landasan Teori

2.1 Penjelasan Singkat Algoritma

2.1.1 Algoritma KMP

Algoritma Knuth-Morris-Pratt (KMP) merupakan algoritma *pattern matching* yang mencari pola pada text dengan urutan kiri ke kanan seperti layaknya algoritma *Brute Force* tetapi bergerak dengan lebih cerdas dibandingkan dengan algoritma *Brute Force*. Apabila terdapat sebuah ketidaksamaan pada text dengan pattern pada indeks j, maka akan dilakukan pergeseran berdasarkan prefix (P[0..j-1]) terkecil dari pattern yang juga merupakan suffix (P[1..j]) dari pattern tersebut. Untuk melakukan hal tersebut, maka akan dibuat *border function* dari pattern tersebut. *Border function* b(k) didefinisikan sebagai ukuran terbesar prefix dari P[0..k-1] yang juga merupakan suffix dari P[1..k]. Secara detailnya, algoritma KMP memodifikasi algoritma *brute force* dengan algoritma berikut:

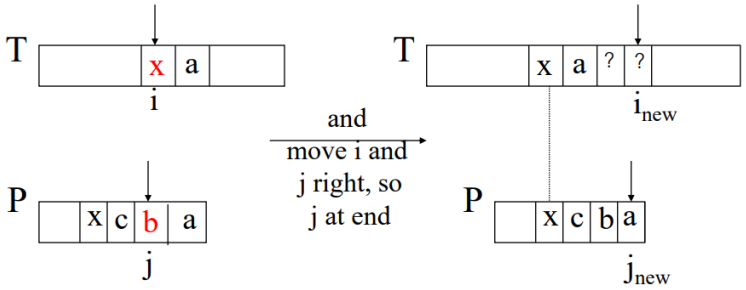
```
Apabila terjadi ketidaksamaan elemen pada P[j](P[j]!=T[i]), maka
     k = j-1;
     j = b(k)
```

Kompleksitas waktu dari algoritma KMP adalah O(m+n), yang merupakan dari hasil pertambahan algoritma perhitungan fungsi pinggiran (O(m)) dan algoritma pencarian string (O(n)). Dari kompleksitas waktunya, algoritma KMP merupakan algoritma yang lebih efisien dibandingkan dengan *brute force* dalam *pattern matching*.

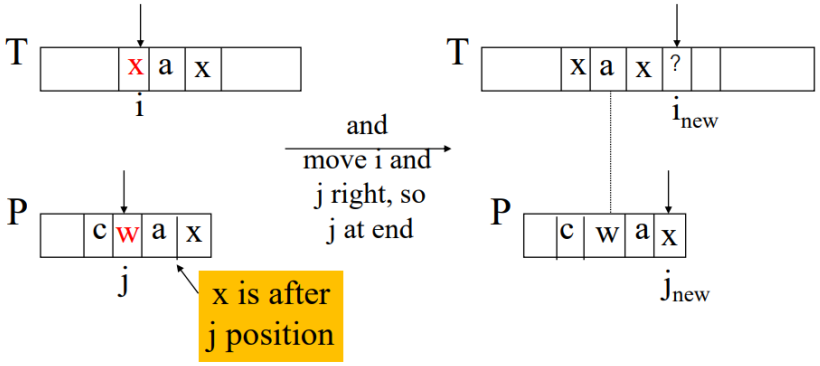
2.1.2 Algoritma BM

Algoritma Boyer-Moore (BM) merupakan algoritma *pattern matching* yang mencari pola pada text berdasarkan dua teknik yaitu *looking-glass technique* atau mencari dengan urutan mundur (dari kanan ke kiri) dan *character-jump technique* yaitu melakukan lompatan yang sesuai jika pattern ditemukan tidak sesuai dengan bagian text yang sedang dicocokkan. Idenya adalah dengan menyimpan data *last occurence* dari tiap karakter yang terdapat pada pattern. Ada 3 kemungkinan ketidakcocokan yang dapat ditemukan pada algoritma ini.

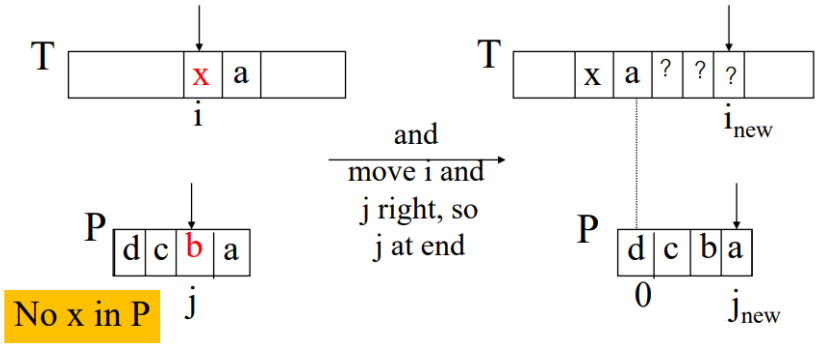
1. Jika ketidakcocokan terjadi pada karakter X(misal) pada text, lalu pada pattern terdapat X dan mungkin dilakukan pergeseran, maka pencocokan diulang pada posisi X pattern sejajar dengan X text.



2. Jika pada pattern terdapat X namun posisi X tidak memungkinkan melakukan pergeseran, maka pencocokan diulang dengan pattern digeser satu karakter ke kanan dari text.



3. Jika pada pattern tidak terdapat X, maka pencocokan diulang dengan posisi patter[0] (karakter pertama pattern) berada pada posisi X + 1.



2.1.3 Algoritma Regex

Regular Expression (*Regex*) adalah sebuah teks yang mendefinisikan sebuah pola pencarian yang dapat digunakan untuk mencocokan, mencari, dan memanipulasi suatu teks dengan efisien. Algoritma pencocokan string dengan Regex singkatnya adalah mencocokkan setiap isi dari text dengan isi ekspresi Regex yang diberikan, jika ada yang tidak sesuai maka text akan ditolak.

Bab 3
Analisis Pemecahan Masalah

3.1 Langkah penyelesain masalah

Penyelesaian masalah dilakukan dengan membagi aplikasi menjadi 3 fitur utama dengan penyelesaian persoalan masing-masing. Fitur-fitur tersebut antara lain fitur Input Disease, fitur DNA Test, dan fitur DNA Test Result. Lebih lengkapnya akan dijelaskan pada subbab di bawah.

3.1.1 Penyelesaian fitur Input Disease

Fitur Input Disease dibuat untuk menambahkan data penyakit dari masukan user. Oleh karena itu, akan dibuat sebuah form entry untuk menerima nama penyakit dari user, dan sebuah form file request untuk meminta file .txt yang berisikan sequence DNA yang dibutuhkan. Selain itu, dilakukan pembuatan fungsi untuk melakukan pengecekan sequence DNA tersebut dengan Regex yang sudah dibuat.

3.1.2 Penyelesaian fitur DNA Test

Fitur DNA Test dibuat untuk melakukan tes DNA berdasarkan masukan user. Oleh karena itu, akan dibuat sebuah form entry untuk menerima nama pasien dari user, sebuah form file request untuk meminta file .txt berisi sequence DNA yang dimiliki pasien, dan sebuah form entry untuk meminta ID penyakit yang akan ditest. Selain itu, dilakukan pembuatan fungsi untuk melakukan kecocokan pattern DNA dari penyakit dengan text DNA dari pasien menggunakan algoritma string matching KMP atau BM.

3.1.1 Penyelesaian fitur DNA Test Result

Fitur DNA Test Result dibuat untuk melihat data-data hasil tes berdasarkan kondisi dari pengguna. Oleh karena itu akan dibuat sebuah form entry yang dapat menerima tanggal masukan user dan masukan opsional nama penyakit. Selain itu, akan dibuat fungsi untuk menyimpan data 10 penyakit teratas berdasarkan masukan tanggal dan nama penyakit (opsional), juga fungsi next dan prev untuk melihat daftar 10 penyakit berikut dan sebelumnya.

3.2 Fitur fungsional dan arsitektur aplikasi web yang dibangun

Aplikasi web yang dibangun memuat homepage sebagai menu utama ketika aplikasi pertama dijalankan. Lalu, ada tab untuk masing-masing fitur yang dibangun dengan fitur fungsional sesuai kebutuhan masing-masing fitur seperti yang telah dijelaskan pada subbab sebelumnya.

Bab 4

Implementasi dan Pengujian

4.1 Spesifikasi teknis program

Program dibuat dengan bahasa pemrograman Golang untuk bagian backend menggunakan Go Fiber Gorm untuk koneksi server dengan database, dan menggunakan framework React.js untuk bagian frontend.

Program dibagi menjadi beberapa bagian dengan beberapa bagian utama yang paling penting misalnya package Utils, package UserHandler dan Package Model yang masing-masing akan dijelaskan lebih lanjut pada subbab di bawah.

4.1.1 Util

Package Utils memuat file-file yang memuat fungsi utilitas yang akan dibutuhkan pada program utama seperti fungsi-fungsi string matching, dan fungsi checkInput. Berikut adalah file-file yang terdapat pada package ini :

- KMP.go, adalah file yang memuat fungsi untuk melakukan string matching menggunakan algoritma KMP.
- BM.go, adalah file yang memuat fungsi untuk melakukan string matching menggunakan algoritma BM.
- FileUtil.go, adalah file yang memuat terkait kebutuhan File Handling.
- Util.go, adalah file yang memuat berbagai utilitas checkInput misalnya checkDNA untuk melakukan cek DNA berdasarkan regex yang diberikan.

4.1.2 UserHandler

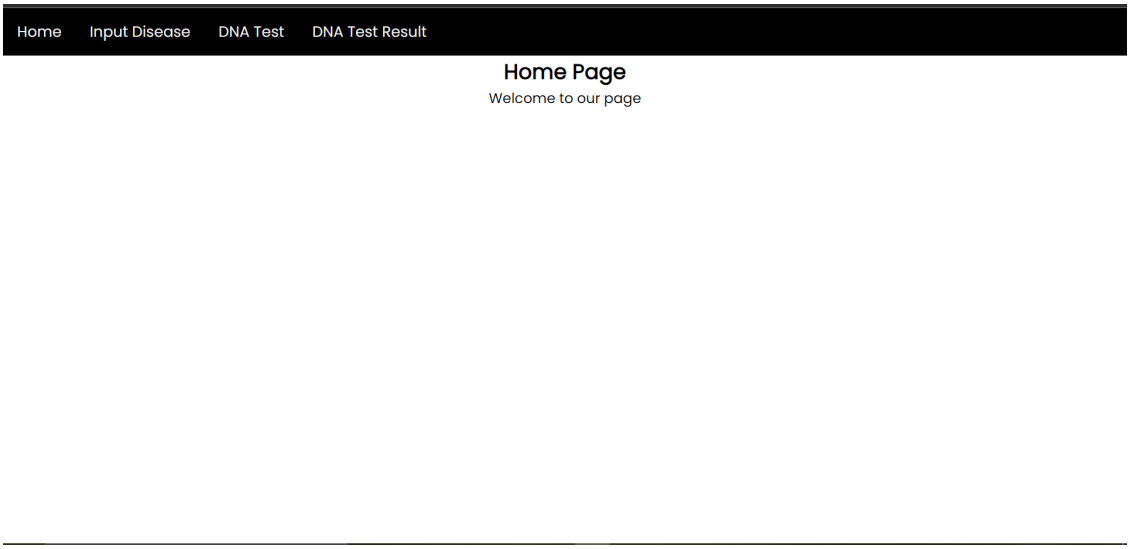
Package UserHandler dibuat sebagai Controller antara backend dan frontend, memuat fungsi-fungsi untuk handler input User dan menghubungkannya dengan database. Nantinya, fungsi-fungsi di bagian ini akan digunakan pada bagian frontend. Misalnya, fungsi HandlerGetAllDisease untuk mendapatkan seluruh data penyakit, HandlerCreateDisease untuk membuat data penyakit baru yang akan digunakan pada fitur Input Disease.

4.1.3 Model

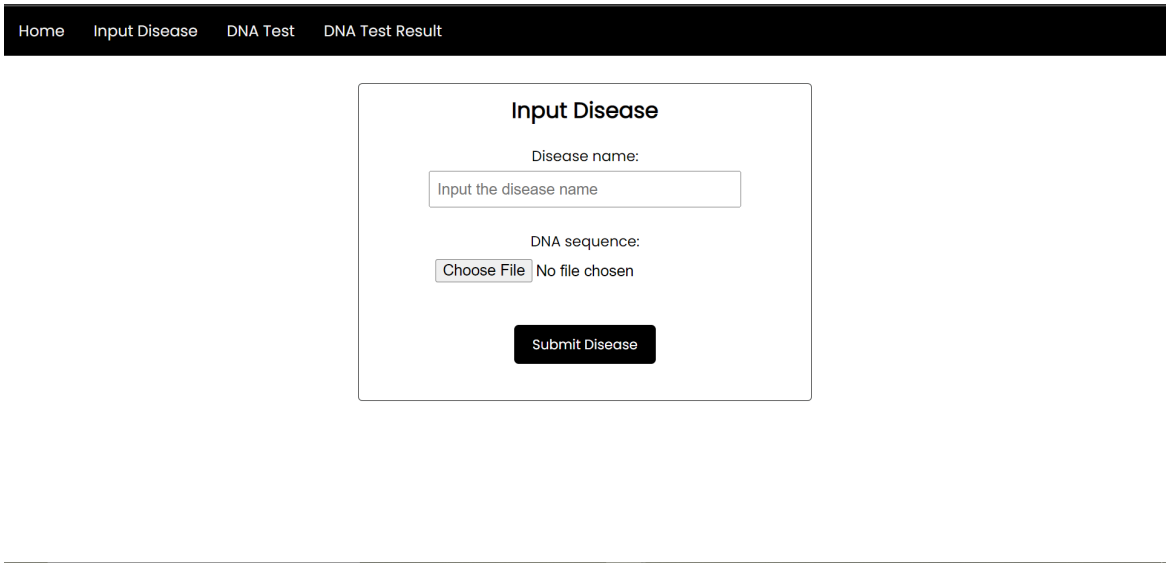
Package Model adalah bagian untuk mengurus terkait tabel pada database, memuat file .go yang berisikan struktur untuk membuat tabel basis data yang dibutuhkan menggunakan Go Fiber Gorm.

4.2 Arsitektur dan Fitur-Fitur Aplikasi

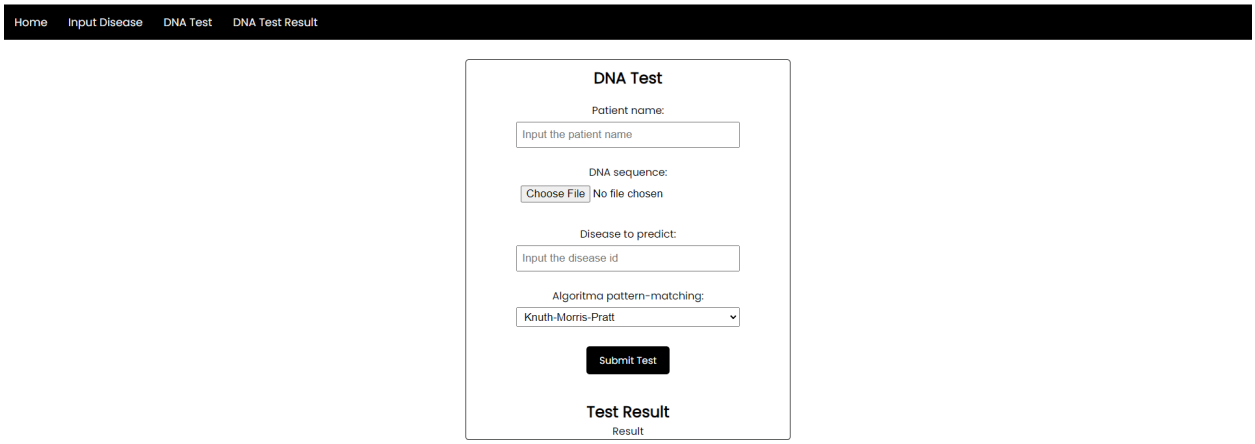
4.2.1 Interface program



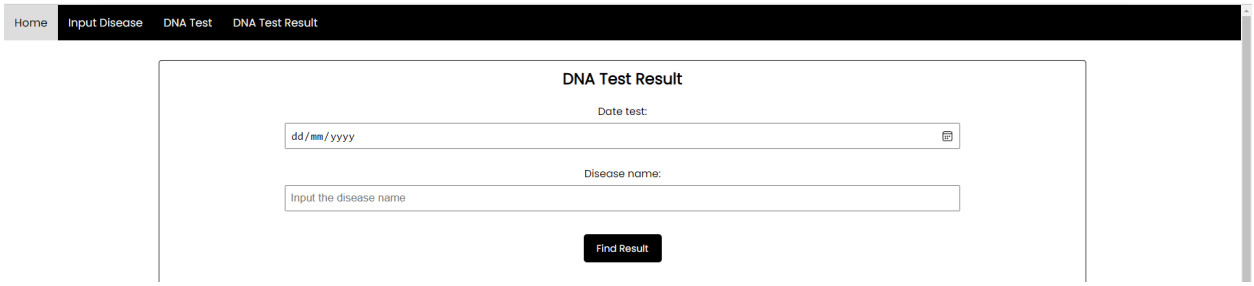
Tampilan Menu Utama



Tampilan Input Disease



Tampilan DNA Test



Tampilan DNA Test Result

4.2.2 Fitur yang disediakan program

Program menyediakan tiga (3) fitur utama, yaitu fitur Input Disease, fitur Tes Dna, dan fitur DNA Test Result yang masing-masing akan dijelaskan lebih lanjut pada subbab di bawah.

4.2.2.1 Fitur Input Disease

Fitur Input Disease dibuat untuk menambahkan data penyakit dari masukan user. User akan diminta memasukkan data nama penyakit yang akan dibuat serta sebuah file .txt berisikan sequence DNA dari penyakit yang bersangkutan. Penambahan data penyakit akan gagal apabila sudah ada nama penyakit yang sama dalam database, ataupun sequence DNA tidak sesuai Regex yang diberikan. Selain itu, penambahan penyakit akan berhasil jika seluruh input sudah sesuai.

4.2.2.2 Fitur DNA Test

Fitur DNA Test dibuat untuk melakukan tes DNA berdasarkan masukan user. User akan diminta memasukkan data nama pasien dan ID penyakit yang akan dilakukan tes DNA, serta sebuah file .txt berisikan sequence DNA pasien yang sesuai. Input akan gagal apabila sequence DNA tidak sesuai Regex

yang diberikan. Selain itu, input akan berhasil dan dilakukan string matching lalu hasilnya akan disimpan dalam database prediction result.

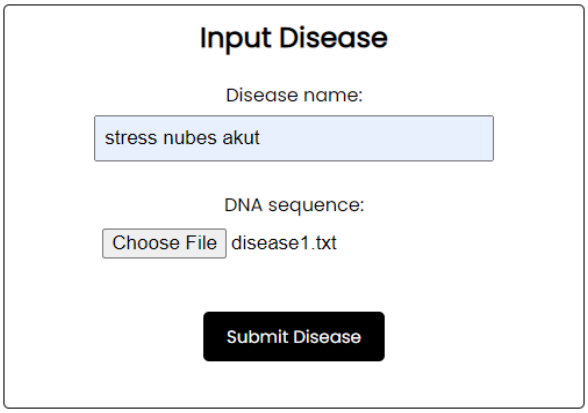
4.2.2.3 Fitur DNA Test Result

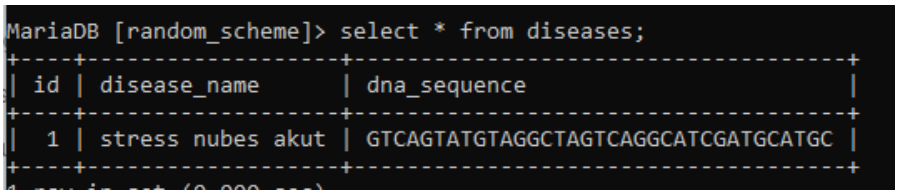
Fitur DNA Test Result dibuat untuk melihat data-data hasil tes berdasarkan kondisi dari pengguna. User akan diminta untuk memasukkan tanggal untuk melihat daftar hasil tes pada tanggal tersebut, dan nama penyakit (opsional) jika ingin melihat hasil tes dari penyakit tertentu. Tanggal masukan harus dalam format yyyy-mm-dd.

4.3 Pengujian program

4.3.1 Kasus Success Input Disease

Input Disease dianggap sukses apabila data penyakit pada tabel disease bertambah sesuai input pengguna

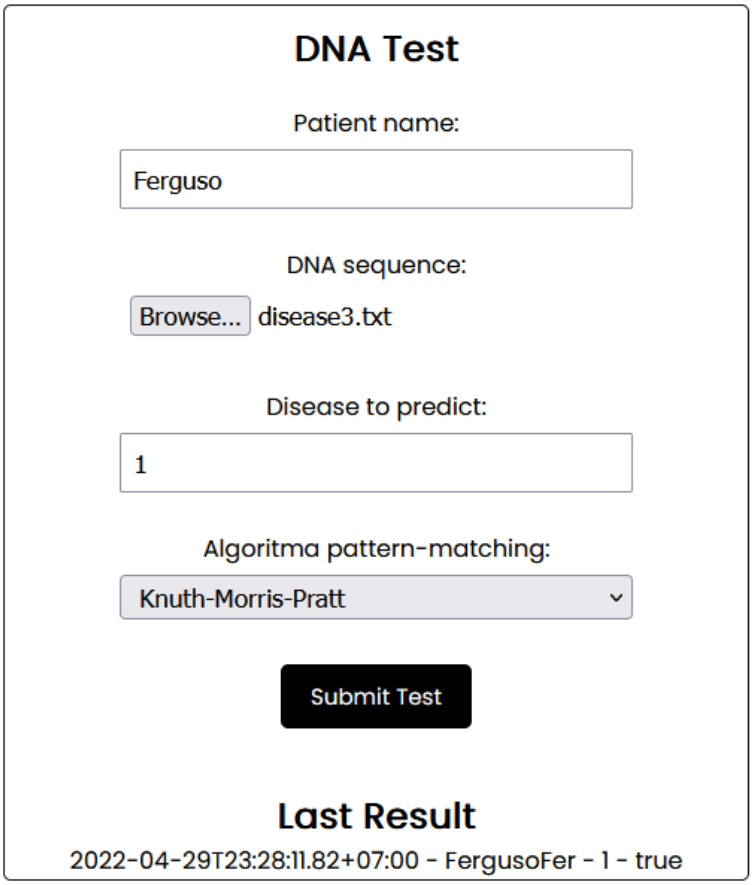




Dapat dilihat pada database akan bertambah data penyakit masukan user sehingga pengujian input disease untuk kasus ini dapat dikatakan berhasil.

4.3.2 Kasus Success DNA Test

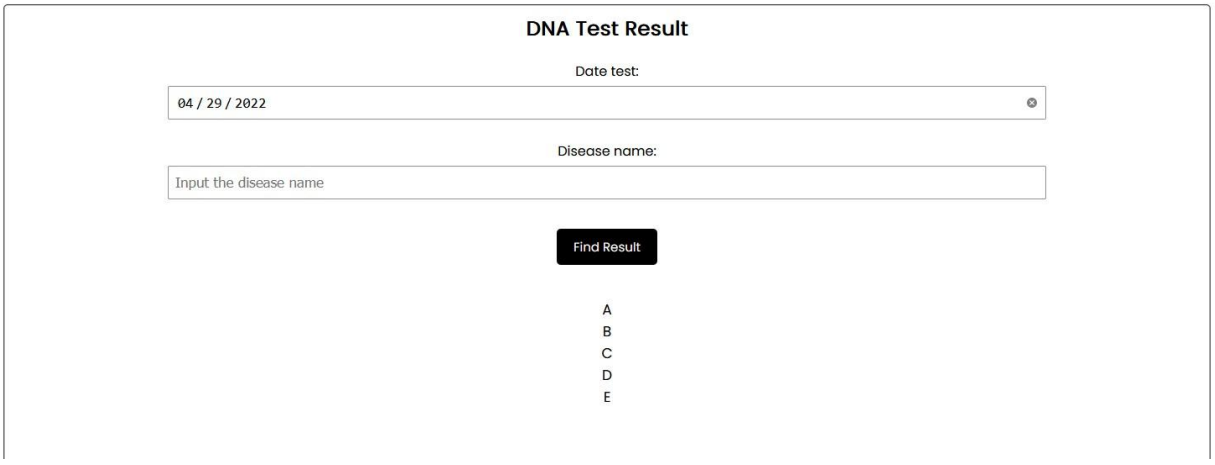
DNA Test dianggap sukses apabila bagian Last Result dapat memunculkan hasil baik test DNA tidak sesuai (false) maupun sesuai (true)



Dapat dilihat bahwa hasil yang ditunjukkan sudah sesuai dengan harapan, yaitu pasien bernama FergusoFer menderita penyakit dengan id 1, sehingga kasus untuk percobaan ini dapat dikatakan berhasil.

4.3.3 Kasus Success DNA Test Result

DNA Test Result dianggap sukses apabila dapat menampilkan daftar hasil pencarian sesuai dengan format yang diberikan



Dapat dilihat bahwa hasil yang ditampilkan belum sesuai, maka dapat disimpulkan bahwa fitur DNA Test Result masih belum sempurna sehingga masih perlu dilakukan pengembangan.

4.4 Analisis hasil pengujian

Dapat dilihat untuk kasus sukses fitur pertama sudah sesuai harapan, sehingga program dapat dikatakan sudah dapat berjalan sudah baik. Untuk fitur DNA Test seharusnya bagian Last Result menunjukkan cukup tanggal saja, namun karena keterbatasan cara akses maka yang muncul adalah waktu penuh. Selain itu, fitur DNA Test Result masih jauh dari harapan.

Bab 5
Kesimpulan

5.1 Kesimpulan

Pembuatan aplikasi berbasis web menggunakan bahasa pemrograman Golang dan React.js dapat berjalan dengan baik sesuai spesifikasi kebutuhan program. Dengan menggunakan regex, input DNA sudah dapat disanitasi sehingga DNA yang dimasukkan sudah sesuai dengan format DNA. Pemeriksaan apabila DNA manusia yang dimasukkan mengandung DNA penyakit yang dicari juga sudah dapat ditemukan dengan menggunakan *string matching* baik dengan algoritma KMP maupun Boyer-Moore. Hasil pengujian menunjukkan bahwa seluruh fitur yang direncanakan sudah berjalan dengan baik sesuai dengan keluaran yang diinginkan.

5.2 Saran

1. Untuk pengembangan selanjutnya, struktur program dapat lebih diperbaiki dan dirapikan untuk mempermudah proses pembuatan dan maintenance

5.3 Komentar dan refleksi

"Bikin aplikasi berbasis web ternyata ribet banget, banyak yang harus diurus dan memakan banyak waktu, pikiran dan tenaga"~

"Yang penting semua rangkaian tugas Stima udah kelarr yeyy!!"~

"gws"

Link Repository

https://github.com/geraldabrhm/Stima03.git