TUGAS BESAR III

PENERAPAN STRING MATCHING DAN REGULAR EXPRESSION DALAM DNA PATTERN MATCHING

LAPORAN

Diajukan sebagai salah satu tugas mata kuliah

IF2211 Strategi Algoritma pada Semester II

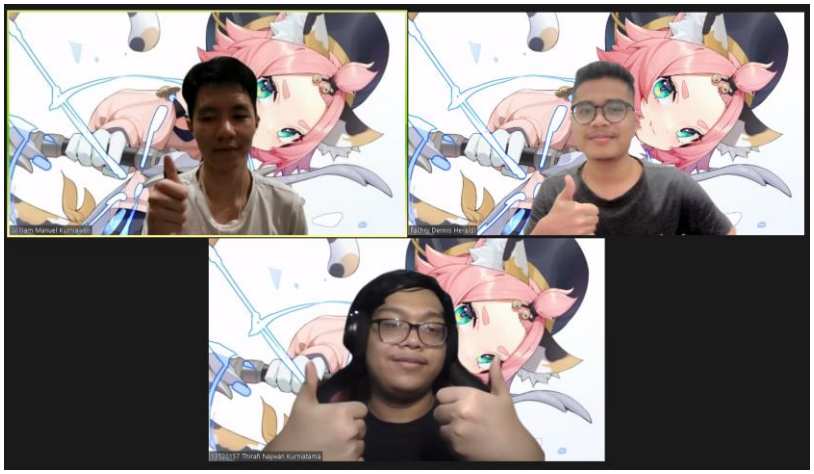
Tahun Akademik 2021-2022

Oleh

William Manuel Kurniawan 13520020

Fachry Dennis Heraldi 13520139

Thirafi Najwan Kurniatama 13520157



SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2022

DAFTAR ISI

BAB I

DESKRIPSI TUGAS

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

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 II

LANDASAN TEORI

2.1 Algoritma Knuth-Morris-Pratt (KMP)

Algoritma Knuth-Morris-Pratt atau yang lebih sering disebut dengan KMP adalah salah satu algoritma *string matching* untuk mencari suatu pola tertentu pada suatu teks dari kiri ke kanan. Pergerakan algoritma ini mirip dengan algoritma *brute force* namun pergeseran dari kiri ke kanan pada algoritma KMP lebih cerdas. Pada algoritma *brute force* pergeseran dilakukan secara satu persatu untuk setiap perbandingan, sedangkan pada algoritma KMP, pergeseran dilakukan sesuai dengan prefiks bagian string perbandingan yang sudah cocok sufiksnya, gunanya agar mengurangi jumlah perbandingan pola yang tidak berguna. Pola yang dicari digeser ke kanan sampai ditemukan prefiks pada pola yang bertetangga dengan sufiks pada bagian string yang dicari.

Pada algoritma ini dikenal juga *border function* b(*k*) yang didefinisikan sebagai ukuran terbesar dari prefiks yang juga merupaka sufiks pada pola string yang dicari. Melalui perhitungan *border function* ini, dapat diketahui pada indeks keberapa perbandingan dimulai kembali ketika terjadi *mismatch* agar tidak terjadi perulangan perbandingan.

Algoritma ini cocok diaplikasikan dalam ukuran file yang besar atau *streaming* karena tidak pernah bergerak “mundur” (mengulang pemeriksaan) seperti algoritma *brute force*. Meskipun demikian, algoritma ini kurang cocok digunakan ketika variasi teks beragam karena akan lebih sering terjadi *mismatch* dikarenakan algortima KMP tidak memperhitungkan karakter apa yang membuat terjadinya *mismatch*.

2.2 Algoritma Boyer-Moore (BM)

Algoritma Booyer-Moore atau yang lebih sering disebut dengan algoritma BM merupakan algoritma *string matching* berbasis dua teknik. Dua teknik tersebut antara lain, *the looking-glass technique* dan *the character-jump technique*.

The looking-glass technique adalah teknik memeriksa kecocokan pattern P dengan teks T, dimulai dari indeks terakhir pada P. Pemerikasaan terhadap T tetap dimulai dari awal, dalam hal ini indeks i dimulai pada nilai m-1 (jika panjang P adalah m).

The character-jump technique dilakukan ketika terjadi *mismatch* (P[i] ≠ T[i] dengan T[i] = x). Terdapat tiga kasus yang perlu diperiksa:

1. Jika P memiliki x di suatu tempat, geser P ke kanan untuk menyesuaikan kemunculan terakhir dari x dalam P dengan T[i]
2. Jika P memiliki x di suatu tempat, tapi suatu pergeseran ke kanan ke kemunculan terakhir tidak bisa dilakukan, geser P ke kanan 1 karakter ke T[i+1].
3. Jika tidak memenuhi kasus 1 dan kasus 2, maka geser P untuk menyesuaikan P[0] dengan T[i+1].

Algoritma BM memiliki fungsi bantuan yang disebut dengan fungsi *last occurrence* L(x). Tujuan dari fungsi ini adalah untuk memeriksa dan menandai indeks kemunculan terakhir seluruh karakter pada T dalam pola P.

Algoritma BM sangat cocok digunakan pada *string matching* yang memiliki teks dengan variasi karakter yang sangat beragam seperti karakter dalam bahasa Inggris namun kurang cocok untuk teks dengan variasi karakter yang tidak beragam seperti karakter binary.

2.3 Regular Expression (Regex)

Regular Expression atau yang sering disebut dengan Regex adalah sekumpulan notasi dan karakter yang digunakan untuk mendeskripsikan suatu pola pada pencarian string. Pola tersebut digunakan dalam pencocokan string untuk melakukan operasi pencarian dan penggantian kata pada string. Regex juga digunakan untuk memvalidasi sebuah masukan string yang akan dicocokan dengan pola yang diterima.

2.4 Aplikasi Web *DNA Pattern Matching*

Aplikasi Web DNA *Pattern/Sequence matching* merupakan salah satu bentuk pengaplikasian ilmu bioinformatika. Pembangunan dan pengembangan aplikasi ini menerapkan algoritma *string matching* dan *regular expression* dengan tujuan untuk membantu penyedia jasa kesehatan dalam memprediksi penyakit pasien. Penyedia jasa kesehatan nantinya dapat mendaftarkan rantai DNA dari penyakit terkait dan nantinya dapat dilakukan prediksi dengan mencocokan rantai DNA penyakit tersebut dengan rantai DNA pasien. Hasil prediksi dapat ditampilkan dalam tabel dan dilengkapi dengan kolom pencarian untuk membantu admin dalam melakukan penyaringan dan pencarian.

Aplikasi Web ini dibangun dengan basis bahasa Go untuk *backend* dan *library* yang dibantu dengan *framework* gin dan gorm. Untuk keperluan database, digunakan MySQL sebagai DBMS yang menyimpan data jenis penyakit dan hasil prediksi dalam bentuk tabel relasi. Untuk bagian design UI pada bagian *frontend* digunakan *framework* React. Adapun kode program yang menangani logic pada web, seperti fetching response dari inputan user berupa request ke API pada *backend* ditulis dalam bahasa JavaScript.

BAB III
ANALISIS PEMECAHAN MASALAH

3.1 Langkah Penyelesaian Masalah Setiap Fitur

3.1.1 Fitur Menerima Input Penyakit Baru

Fitur ini berupa halaman yang dapat melakukan penerimaan input penyakit baru berupa nama penyakit dan sequence DNA dan dimasukkan ke dalam database. Langkah-langkah penyelesaian masalah dalam pembuatan fitur ini adalah sebagai berikut.

1. Menyiapkan basis data dengan bantuan DBMS MySQL. Dibuat tabel baru bernama `penyakits` yang memiliki atribut ‘nama_penyakit` untuk menyimpan nama penyakit dan atribut `dna_penyakit` untuk menyimpan sekuens DNA penyakit tersebut.
2. Menyiapkan API pada *backend* aplikasi yang ditulis dalam bahasa Go menggunakan bantuan framework gin untuk kebutuhan HTTP response dan framework gorm untuk kebutuhan manipulasi basis data. Arsitektur API dibuat berlapis (fetch-handler-service-repository-database).
3. Membuat REST API method POST untuk menerima masukan JSON yang memiliki atribut ‘nama_penyakit’ untuk informasi nama penyakit dan atribut ‘dna_penyakit’ untuk informasi sekuens DNA penyakit tersebut.
4. Menyiapkan *library* untuk melakukan sanitasi terhadap masukan DNA penyakit yang diinput oleh pengguna. Untuk melakukan sanitasi terhadap DNA penyakit tersebut, digunakan Regex.
5. Menyiapkan Web UI pada *frontend* aplikasi dengan bantuan framework React dengan basis Node.JS. Dibuat tampilan halaman web untuk menambahkan penyakit. Halaman tersebut akan menerima input nama penyakit dan file .txt sekuens DNA dari penyakit tersebut.
6. Membuat fungsi untuk melakukan fetching response method POST yang ditulis dalam bahasa JavaScript. Ketika terdapat inputan penyakit, maka inputan tersebut akan ditranslasi dalam bentuk JSON, kemudian dipanggil fungsi untuk merespon method POST.
7. Memanggil handler method POST pada bagian *backend* untuk menerima request dalam bentuk JSON. Masukan DNA penyakit terlebih dahulu dicek menggunakan *library* sanitasi dengan Regex. Jika masukan DNA penyakit valid, maka handler akan memanggil service untuk melakukan penambahan data baru. Penerimaan input

penyakit baru dinyatakan berhasil ketika record baru berhasil ditambahkan pada basis data.

3.1.2 Fitur Prediksi Penyakit

Fitur ini berupa halaman yang menerima inputan dari pengguna untuk melakukan prediksi terhadap seseorang menderita penyakit tertentu berdasarkan *sequence* DNA-nya. Langkah-langkah penyelesaian masalah dalam pembuatan fitur ini sejalan dengan fitur sebelumnya, langkah-langkahnya adalah sebagai berikut.

1. Memastikan tabel yang berisi nama-nama penyakit telah terdefinisi pada basis data berdasarkan pembuatan fitur sebelumnya.
2. Membuat tabel baru bernama `riwayats` yang memiliki atribut `tanggal_pred` untuk menyimpan informasi tanggal dilakukan prediksi, atribut `nama_pasien` untuk menyimpan informasi nama pasien yang diprediksi, atribut `nama_penyakit` untuk menyimpan informasi nama penyakit yang diprediksi, dan atribut `status` untuk menyimpan informasi hasil prediksi.
3. Menyiapkan response baru pada API *backend* yang telah dibuat pada fitur sebelumnya dengan membuat REST API method POST untuk menerima masukan JSON yang memiliki atribut ‘nama_pasien’ untuk informasi nama pasien, atribut ‘dna_pasien’ untuk informasi sekuens DNA pasien tersebut, dan atribut ‘nama_penyakit`
4. Menyiapkan *library* untuk sanitasi inputan DNA pasien. Disiapkan pula *library* untuk melakukan *string matching* baik menggunakan algoritma KMP maupun BM. Namun untuk melakukan *string matching pattern* DNA, kami menggunakan algoritma KMP karena dinilai lebih cepat.
5. Menyiapkan Web UI pada *frontend* aplikasi dengan bantuan framework React dengan basis Node.JS. Dibuat tampilan halaman web untuk memprediksi penyakit. Halaman tersebut akan menerima input nama pasien, sekuens dna pasien dalam file txt, dan nama penyakit yang ingin diprediksi.
6. Membuat fungsi untuk melakukan fetching response method POST yang ditulis dalam bahasa JavaScript. Ketika terdapat inputan untuk memprediksi, maka inputan tersebut akan ditranslasi dalam bentuk JSON, kemudian dipanggil fungsi untuk merespon method POST.
7. Memanggil handler method POST pada bagian *backend* untuk menerima request dalam bentuk JSON. Handler kemudian akan mengolah masukan dari JSON, mula-mula akan divalidasi terlebih dahulu masukan DNA pasien menggunakan *library* sanitasi dengan

Regex. Jika DNA pasien valid, selanjutnya akan divalidasi masukan nama penyakit, nama penyakit valid jika ditemukan data DNA penyakit berdasarkan pencarian masukan nama penyakit. Selanjutnya dilakukan string matching menggunakan algoritma KMP dengan DNA pasien sebagai *text* dan DNA penyakit sebagai *pattern*, jika pattern ditemukan, maka akan mengembalikan status "True". Handler kemudian akan memanggil service untuk melakukan penambahan data baru. Penerimaan input prediksi baru dinyatakan berhasil ketika record baru berhasil ditambahkan pada basis data.

3.1.3 Fitur Menampilkan Hasil Prediksi

Fitur ini berupa halaman yang menerima inputan query dan menampilkan urutan hasil prediksi dengan kolom pencarian di dalamnya. Langkah-langkah penyelesaian masalah dalam pembuatan fitur ini adalah sebagai berikut.

1. Memastikan tabel yang berisi riwayat prediksi telah terdefinisi pada basis data berdasarkan pembuatan fitur sebelumnya.
2. Menyiapkan response baru pada API *backend* yang telah dibuat pada fitur sebelumnya dengan membuat REST API method GET untuk menerima query berupa tanggal prediksi dan atau nama penyakit.
3. Menyiapkan *library* untuk melakukan validasi dan *parsing* terhadap query menggunakan Regex. Inputan query akan dicocokkan dengan *pattern* yang sesuai. Kemudian dilakukan *parsing* apakah query merupakan tanggal atau nama penyakit atau dapat keduanya.
4. Menyiapkan Web UI pada *frontend* aplikasi dengan bantuan framework React dengan basis Node.JS. Dibuat tampilan halaman web untuk melakukan pencarian hasil prediksi berdasarkan input query tanggal dan atau nama penyakit.
5. Membuat fungsi untuk melakukan fetching response method GET yang ditulis dalam bahasa JavaScript. Inputan dari halaman web nantinya ditranslasi ke dalam parameter query, kemudian dipanggil fungsi untuk merespon method GET.
6. Memanggil handler method GET pada bagian *backend* untuk mengembalikan data dalam bentuk JSON. Handler akan memanggil *service* yang bersesuaian dengan parameter query kemudian mengambil informasi dari basis data dan mentranslasikannya dalam bentuk JSON. Bentuk JSON ini akan di-*pass* kembali ke bagian *frontend* untuk menampilkan data hasil pencarian.

7. Membuat fungsi yang mentranslasikan bentuk JSON ke bentuk teks yang dapat terbaca pada tampilan halaman Web pada bagian *backend*. Fungsi ini menerima JSON dari pemanggilan response GET. Fungsi untuk menampilkan ini berhasil jika muncul tulisan hasil prediksi untuk query yang valid.

3.1.4 Fitur Menghitung Tingkat Kemiripan DNA

Fitur ini merupakan tambahan dari fitur menampilkan hasil prediksi berupa menampilkan hasil perhitungan tingkat kemiripan DNA pengguna dengan DNA penyakit pada tes DNA. Langkah-langkahnya mirip pada fitur prediksi penyakit, namun ada penambahan sebagai berikut.

1. Pada basis data untuk tabel yang berisi riwayat prediksi ditambahkan atribut baru, yaitu `similarity`
2. Pada *library* ditambahkan fungsi untuk menghitung tingkat kemiripan dengan metode Longest Common Subsequence (LCS). Fungsi ini akan membandingkan tingkat kecocokan *pattern* di dalam *text*.
3. Pada tampilan hasil prediksi ditambahkan persentase tingkat kemiripan berdasarkan perhitungan dengan metode LCS.

3.2 Fitur Fungsional dan Arsitektur Aplikasi Web yang Dibangun

3.2.1 Fitur Fungsional

Fitur fungsional yang terdapat pada aplikasi web ini adalah:

1. Aplikasi dapat menerima input penyakit baru berupa nama penyakit dan sequence DNA-nya
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. Kolom pencarian bekerja sebagai filter dalam menampilkan hasil
4. (Bonus) Aplikasi dapat menghitung tingkat kemiripan DNA pengguna dengan DNA penyakit pada tes DNA

3.2.2 Arsitektur Web

Backend:

- Go
- Framework: gin, gorm

Frontend:

- HTML+CSS+JavaScript
- Framework: ReactJS, Material-UI-Skin

Database:

- MySQL

Deployment:

DBMS: remotemysql

API: Heroku

Frontend: Netlify

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Spesifikasi Teknis Program

1. Database DNA Matching

   Menyimpan data dari pengguna yang menggunakan aplikasi. Data yang disimpan adalah data penyakit dan data riwayat penyakit pengguna. Data penyakit memiliki atribut (nama_penyakit, dna_penyakit). Data riwayat penyakit pengguna memiliki atribut (tanggal_pred, nama_pasien, nama_penyakit, similarity, status)

2. Repository Penyakit
   a) NewRepository

   Fungsi membuat database penyakit baru menggunakan library gorm.

   b) FindAll

   Fungsi menerima parameter database penyakit yang dibuat dan mengeluarkan semua data yang ada dalam database tersebut

   c) FindByName

   Fungsi menerima parameter database penyakit yang dibuat serta nama penyakit dan mengeluarkan data penyakit tersebut jika ditemukan dalam database

   d) Create

   Fungsi membuat penyakit baru dan memasukkan penyakit ke dalam database

3. Repository Penyakit
   a) NewRepository

   Fungsi membuat database penyakit baru menggunakan library gorm.

   b) FindAll

   Fungsi menerima parameter database penyakit yang dibuat dan mengeluarkan semua data yang ada dalam database tersebut

   c) FindByPenyakit

   Fungsi menerima parameter database penyakit yang dibuat serta nama penyakit dan mengeluarkan data riwayat tersebut jika ditemukan dalam database

   d) FindByTanggal

   Fungsi menerima parameter database penyakit yang dibuat serta tanggal prediksi dan mengeluarkan data riwayat tersebut jika ditemukan dalam database

   e) FindByTanggalPenyakit

Fungsi menerima parameter database penyakit yang dibuat serta tanggal prediksi dan nama penyakit lalu mengeluarkan data riwayat tersebut jika ditemukan dalam database

f) Create

Fungsi membuat data riwayat baru dan memasukkan penyakit ke dalam database

4. Library
   a) BM

Fungsi menerima 2 string sebagai parameter dengan salah satu string lebih panjang dari yang lainnya. Fungsi memeriksa jika string yang lebih pendek berada di dalam string lain yang lebih panjang menggunakan metode Boyer-Moore.

b) KMP

Fungsi menerima 2 string sebagai parameter dengan salah satu string lebih panjang dari yang lainnya. Fungsi memeriksa jika string yang lebih pendek berada di dalam string lain yang lebih panjang menggunakan metode Knuth-Morris-Pratt

c) LcsCount

Fungsi menerima 2 string sebagai parameter dengan salah satu string lebih panjang dari yang lainnya. Fungsi mencari longest common subsequence (LCS) dari kedua string tersebut dan mengeluarkan hasil jumlah huruf proses LCS.

d) LcsResult

Fungsi menerima 2 string sebagai parameter dengan salah satu string lebih panjang dari yang lainnya. Fungsi menggunakan fungsi LcsCount dan mengeluarkan hasil perbandingan jumlah huruf proses LCS dan string yang lebih panjang.

e) Sanitasi

Fungsi menerima sebuah string sebagai parameter dan menggunakan regex untuk memeriksa jika data yang dimasukkan valid untuk dimasukkan ke database

f) CheckQuery

Fungsi menerima sebuah string sebagai parameter kemudian menggunakan regex untuk validasi. Selanjutnya string tersebut diparse menjadi dua buah string sebagai inputan tanggal dan nama penyakit

4.2 Tata Cara Penggunaan Program

Program akan dijalankan secara lokal di perangkat. Sebelum memulai menggunakan program, pastikan telah menginstal program-program berikut:

- go
- MySQL/MariaDB
- Node.js

Setelah program tersebut berhasil diinstal, selanjutnya lakukan langkah-langkah berikut secara berurutan.

- Menyiapkan basis data
  1. Jalankan koneksi database dan masuk ke dalam database monitor
  2. Buat database baru bernama `dnamatching` dengan menjalankan query berikut:
     ```
     create database dnamatching
     ```
  3. Selanjutnya akses database yang telah dibuat dengan menjalankan query berikut:
     ```
     use database dnamatching
     ```
  4. Buat tabel baru bernama `penyakits` dan `riwayats` dengan menjalankan dua query berikut:

     ```
     create table penyakits(
     nama_penyakit varchar(100) not null,
     dna_penyakit varchar(255) not null,
     primary key(nama_penyakit)
     );

     create table riwayats(
     tanggal_pred date not null,
     nama_pasien varchar(100) not null,
     nama_penyakit varchar(100) not null,
     similarity decimal(10,2) not null,
     status varchar(11) not null,
     foreign key(nama_penyakit) references
     penyakits(nama_penyakit)
     );
     ```

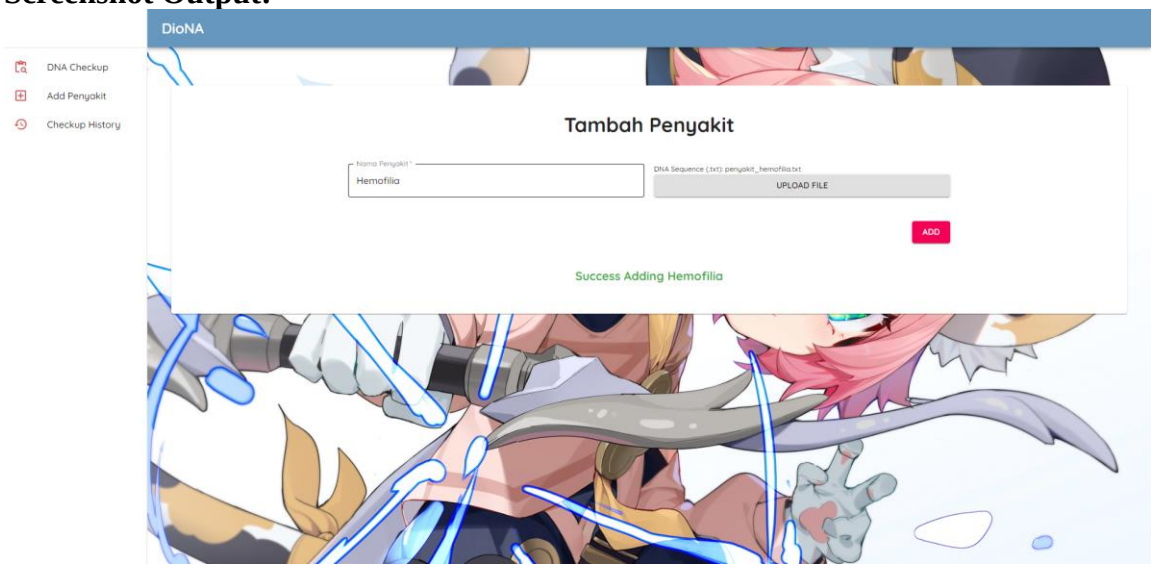
  5. Jika tidak muncul pesan error, maka basis data telah berhasil dibuat dan siap digunakan.
- Menjalankan backend
  1. Akses direktori `backend/api`.
  2. Pastikan
  3. Buka terminal pada direktori tersebut lalu jalankan perintah `go run main.go`.
  4. Jika tidak muncul pesan error, maka backend telah berhasil dijalankan.
- Menjalankan frontend
  1. Membuka folder frontend menggunakan terminal.
  2. Menjalankan perintah “npm install”.

3. Menjalankan perintah “npm start” setelah database sudah disiapkan dan backend sudah dijalankan.
4. Jika berhasil dijalankan, maka akan terbuka window baru yang diarahkan pada halaman web aplikasi.

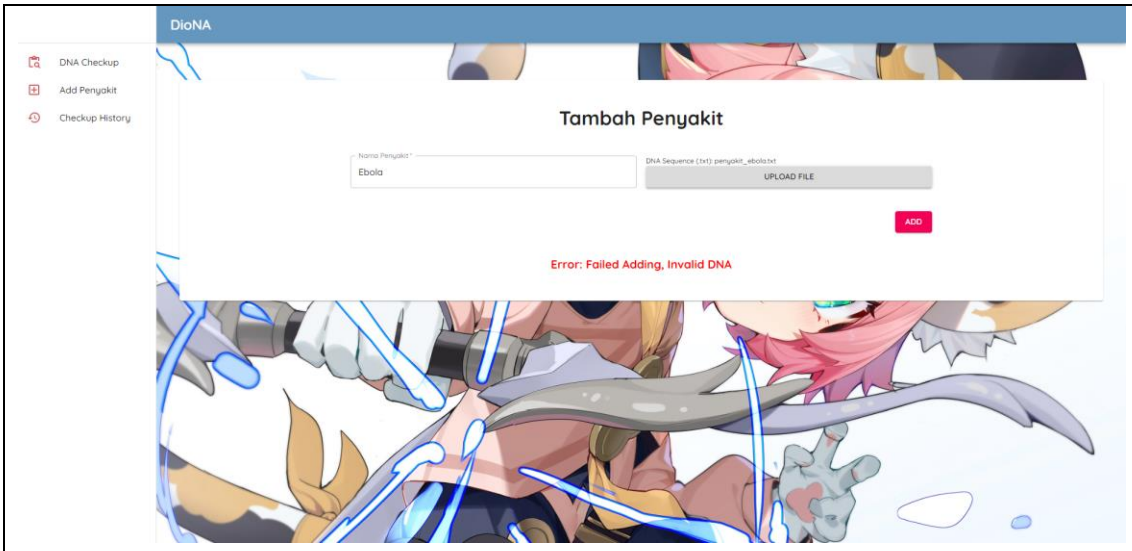
Jika ingin menjalankan web aplikasi secara daring, dapat mengunjungi situs berikut ini: dnamatch-fend.netlify.app

4.3 Hasil Pengujian

4.3.1 Pengujian Fitur Menerima Input Penyakit Baru

Input: Nama Penyakit: Hemofilia File DNA penyakit (penyakit_hemofilia.txt): ACTGGTACACCATG
Screenshot Output: 
Penjelasan: Inputan valid, penyakit Hemofilia berhasil ditambahkan

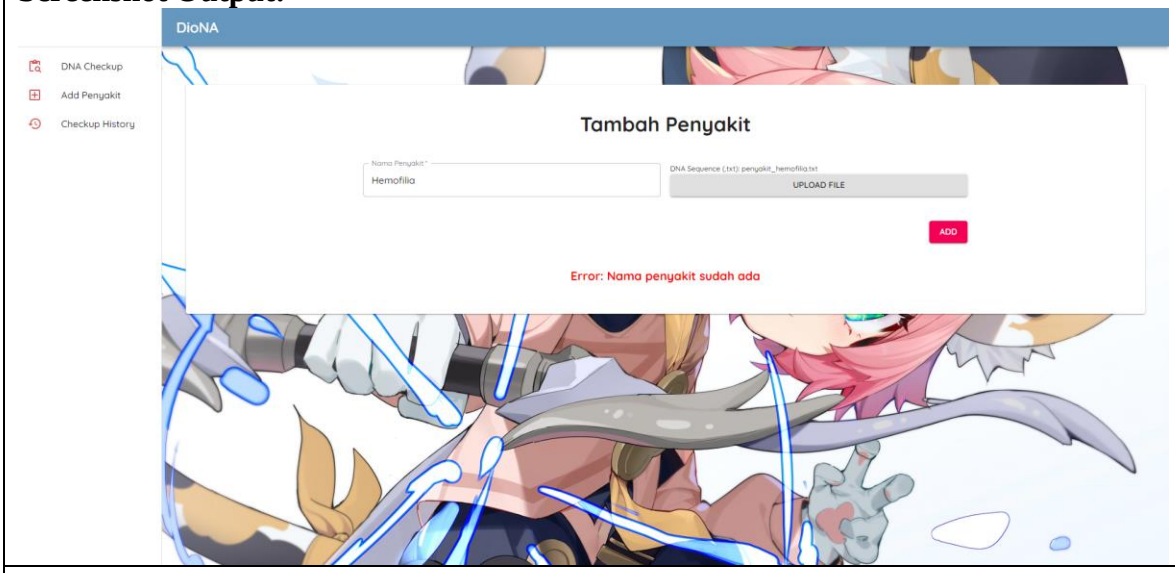
Input: Nama Penyakit: Ebola File DNA penyakit (penyakit_ebola.txt): TAUATUTTATTUUG
Screenshot Output:



Penjelasan:
Inputan tidak valid karena input DNA sequence Ebola tidak memenuhi standar.

Input:
Nama Penyakit: Hemofilia
File DNA penyakit (penyakit_hemofilia.txt):
ACTGGTACACCATG

Screenshot Output:



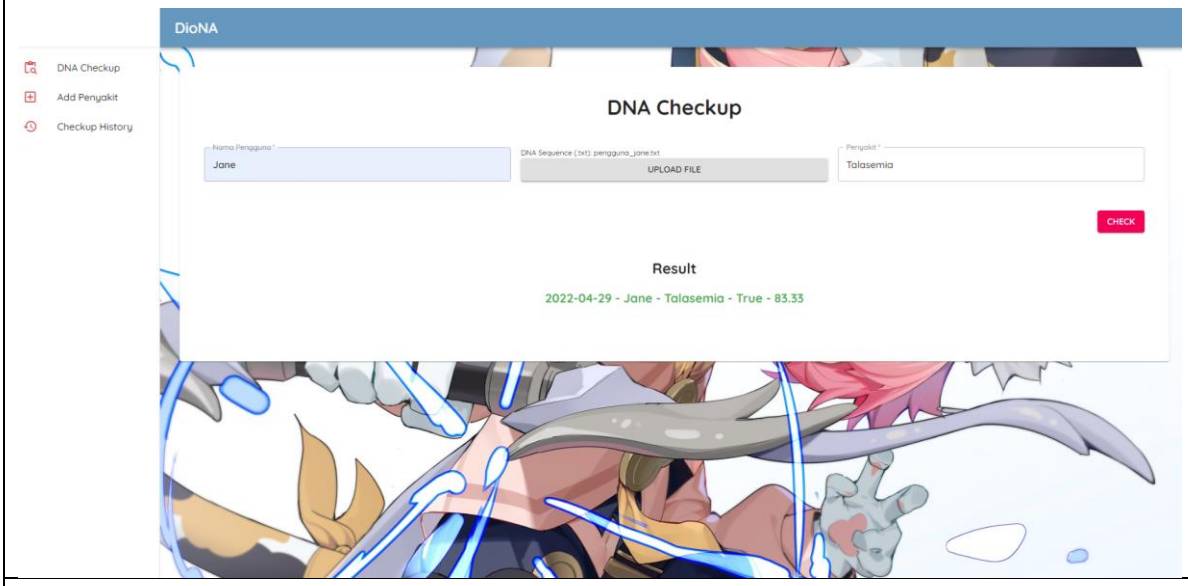
Penjelasan:
Inputan tidak valid karena tidak dapat menambahkan penyakit yang sudah ada.

4.3.2 Pengujian Fitur Prediksi Penyakit

Input:
Nama Pengguna: Jane
File DNA pengguna (pengguna_jane.txt):
GCCCACTGGTGTCAACACCGCATATG

Nama penyakit: Talasemia

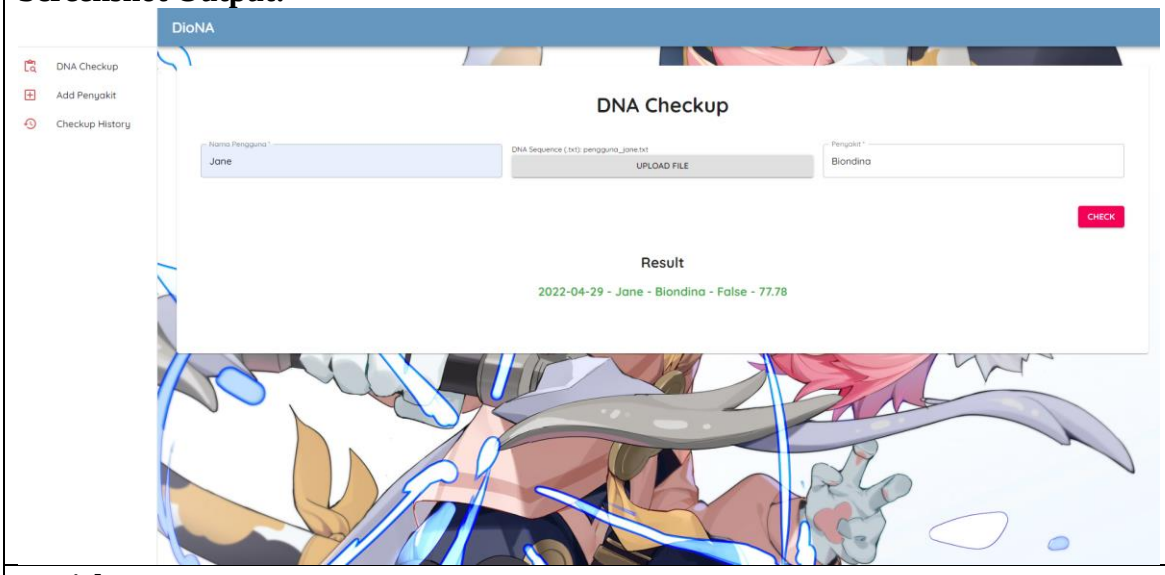
Screenshot Output:



Penjelasan:
Inputan valid dan memberikan hasil prediksi True berdasarkan masukan.

Input:
Nama pengguna: Jane
File DNA pengguna (pengguna_jane.txt):
GCCCACTGGTGTCAACACCGCATATG
Nama penyakit: Biondina

Screenshot Output:

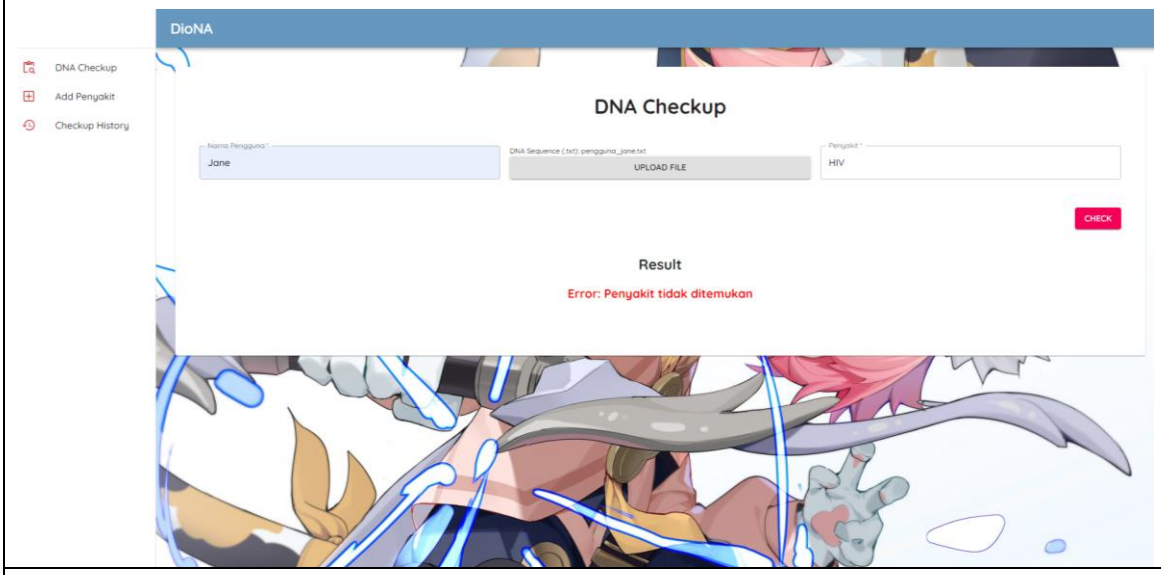


Penjelasan:
Inputan valid dan memberikan hasil prediksi False berdasarkan masukan.

Input:
Nama pengguna: Jane
File DNA pengguna (pengguna_jane.txt):

GCCCACTGGTGTCAACACCGCATATG
Nama penyakit: HIV

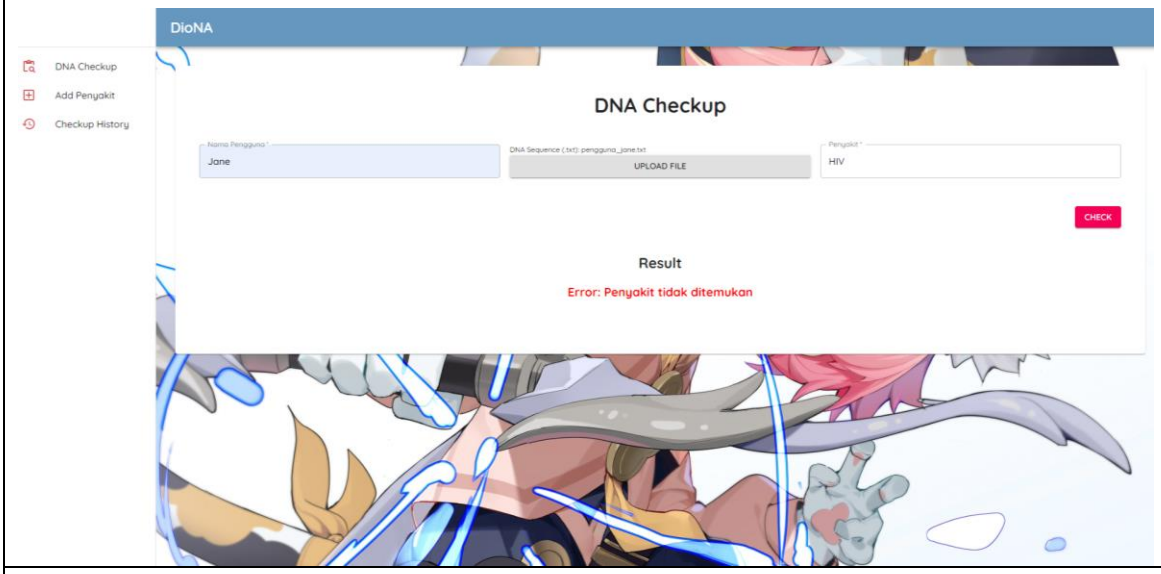
Screenshot Output:



Penjelasan:
Pesan error muncul karena input masukan penyakit tidak valid, penyakit tidak ada di basis data.

Input:
Nama pengguna: Jane
File DNA pengguna (pengguna_jane.txt):
GCCCACTGGTGTCAACACCGCATATG
Nama penyakit: HIV

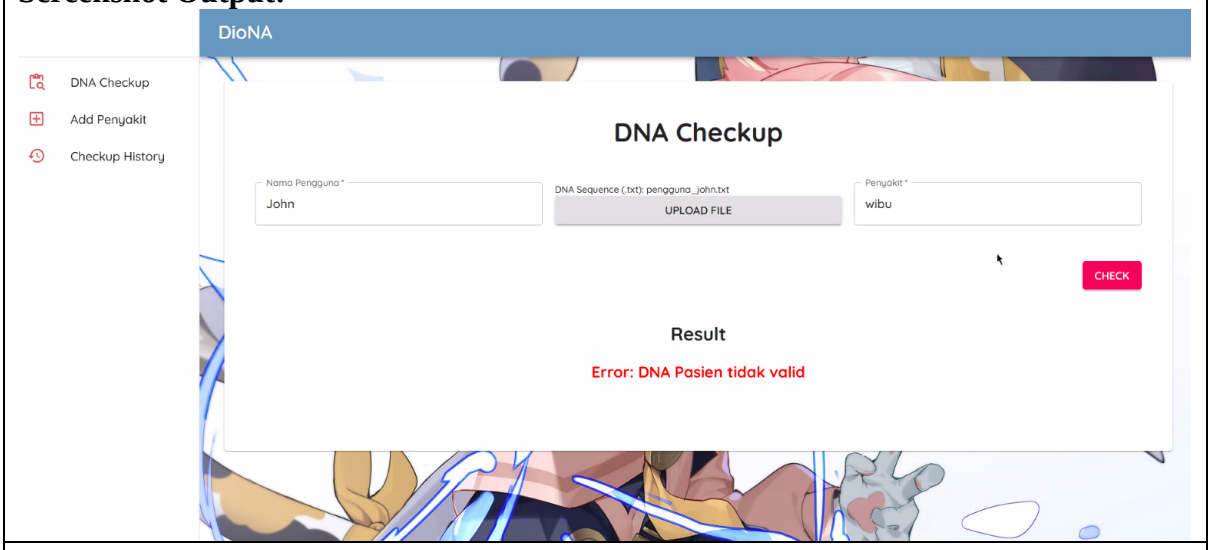
Screenshot Output:



Penjelasan:
Pesan error muncul karena input masukan penyakit tidak valid, penyakit tidak ada di basis data.

Input:
Nama pengguna: John
File DNA pengguna (pengguna_john.txt):
GUAAAATUUTUGGTATGUTAA
Nama penyakit: Wibu

Screenshot Output:

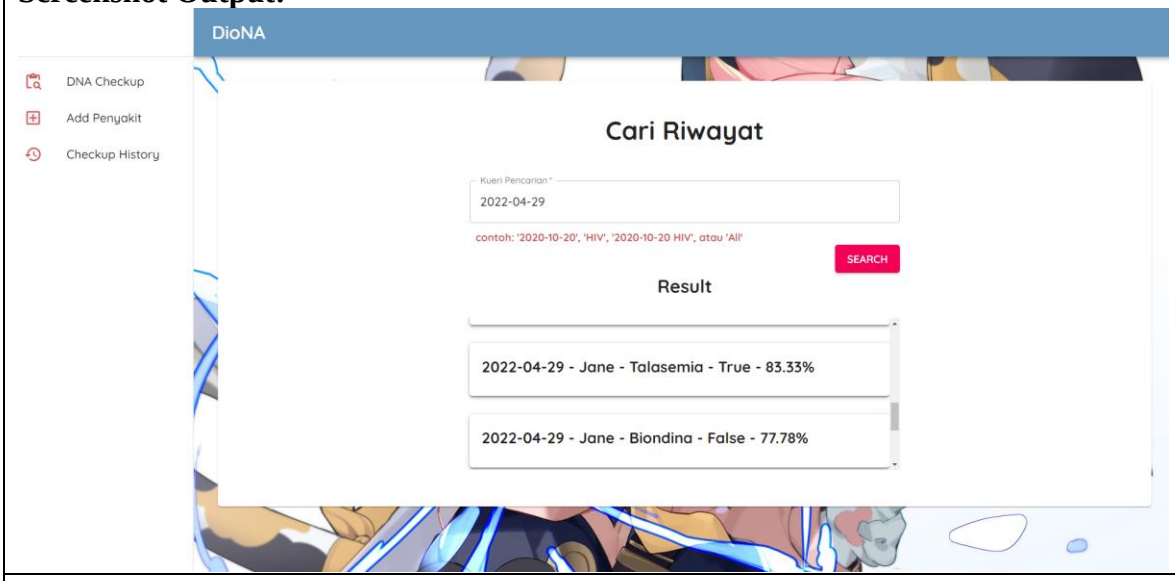


Penjelasan:
Pesan error muncul karena input masukan DNA Sequence tidak valid, penyakit tidak ada di basis data.

4.3.3 Pengujian Fitur Menampilkan Hasil Prediksi

Input:
Kueri pencarian: 2022-04-29

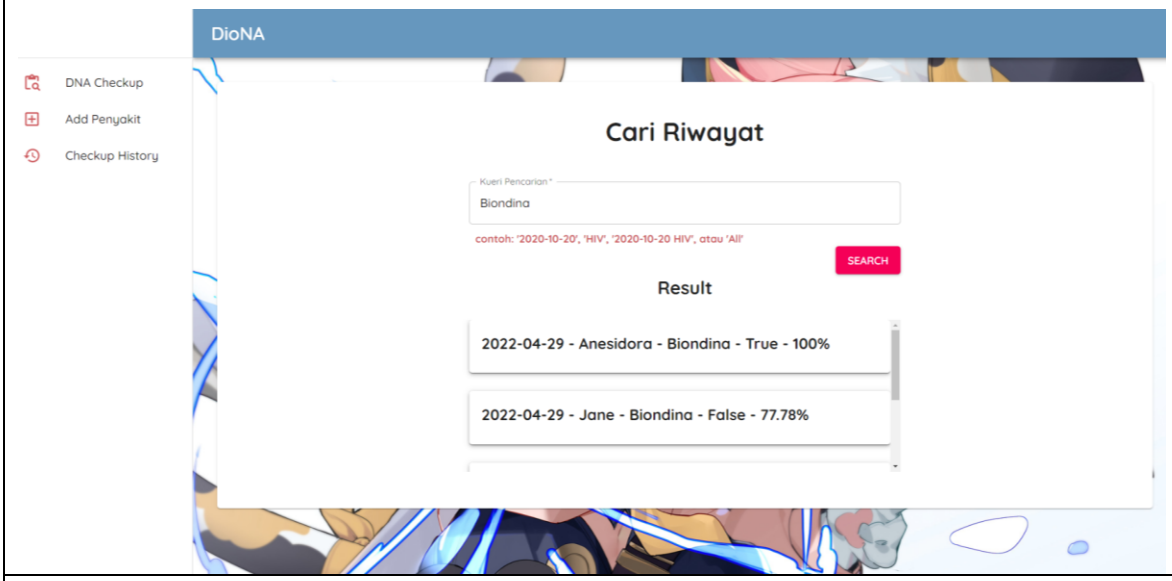
Screenshot Output:



Penjelasan:
Menampilkan hasil pencarian berdasarkan masukan hanya tanggal

Input:
Kueri pencarian: Biondina

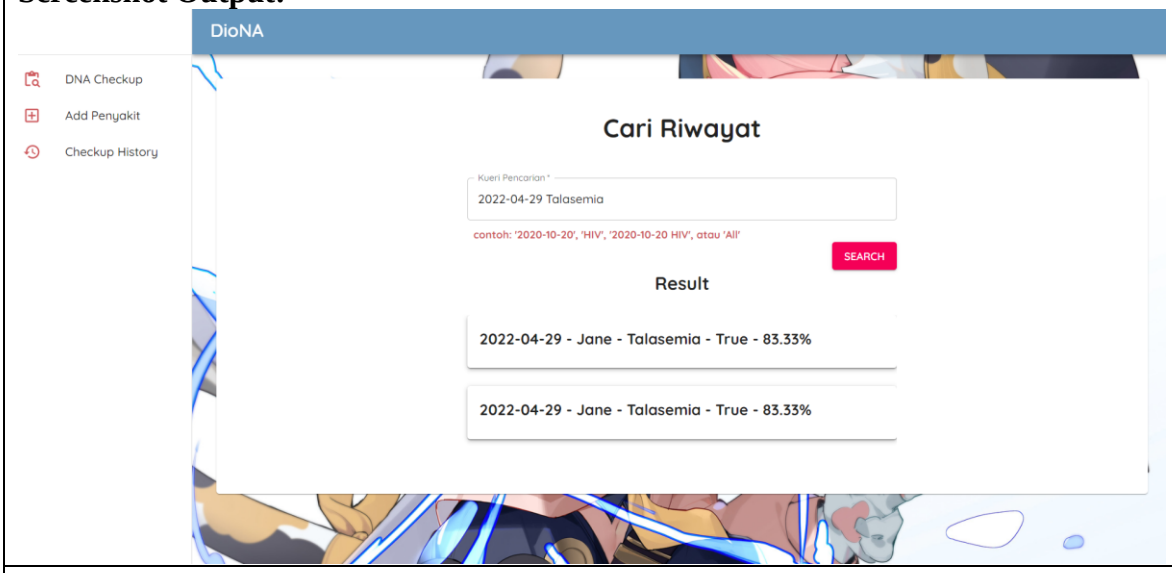
Screenshot Output:



Penjelasan:
Menampilkan hasil pencarian berdasarkan masukan tanggal dan nama penyakit

Input:
Kueri pencarian: 2022-04-29 Talasemia

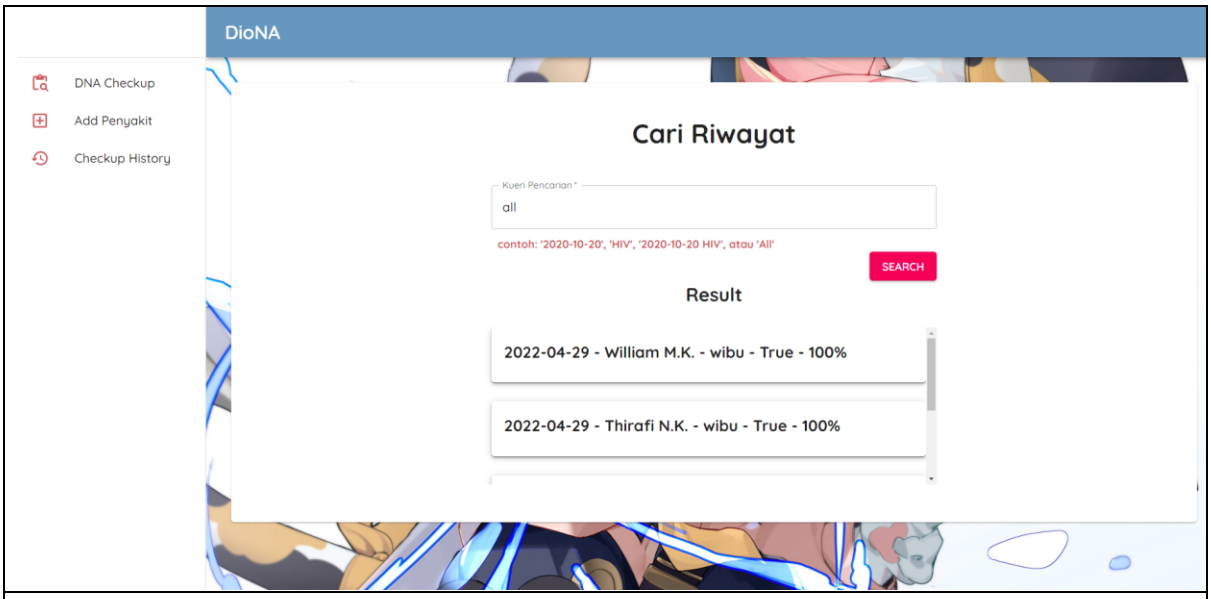
Screenshot Output:



Penjelasan:
Menampilkan hasil pencarian berdasarkan masukan tanggal dan nama penyakit

Input:
Kueri pencarian: all

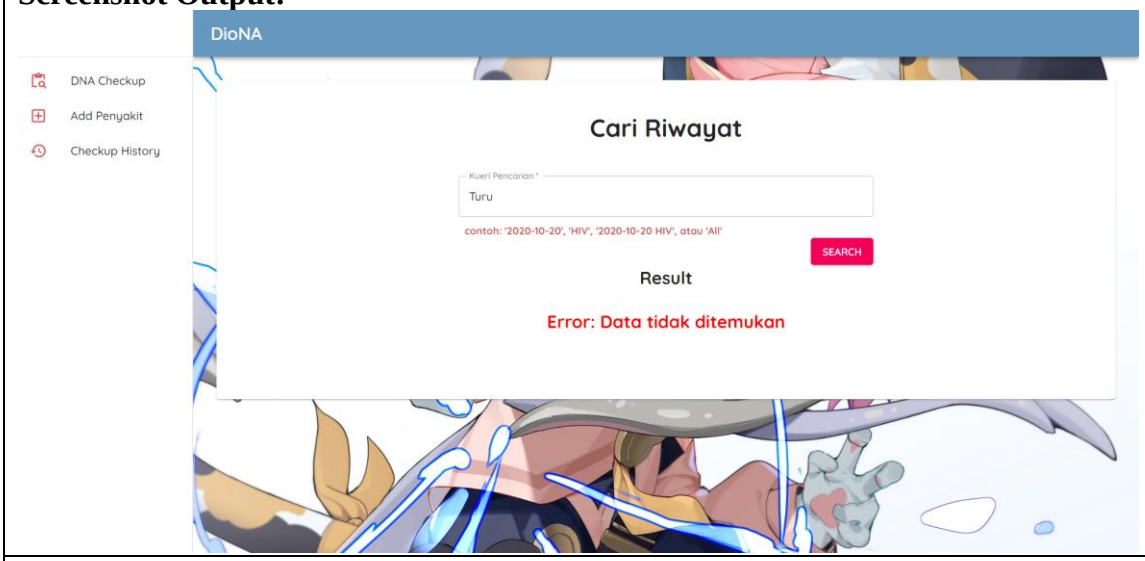
Screenshot Output:



Penjelasan:
Menampilkan seluruh data dengan kueri all

Input:
Kueri pencarian: Turu

Screenshot Output:



Penjelasan:
Pesan error muncul karena nama penyakit tidak ditemukan

4.4 Analisis Hasil Pengujian

Pada pengujian fitur menambahkan penyakit baru, pengguna harus menginput nama penyakit yang valid. Nama penyakit yang valid adalah nama penyakit yang belum ada di basis data. Pengguna juga harus menginput file DNA sequence yang valid yaitu hanya memuat huruf A, G, T, dan C. Pada pengujian, file DNA yang tidak valid akan ditolak dan ditampilkan pesan error.

Pada pengujian fitur prediksi penyakit, pengguna harus menginput DNA sequence pasien yang valid dan nama penyakit yang valid. DNA sequence yang valid hanya memuat huruf A, G, T, dan C. Pada pengujian, file DNA yang tidak valid akan ditolak dan ditampilkan pesan error. Nama penyakit yang valid adalah nama penyakit yang terdapat pada basis data. Pada pengujian, nama penyakit yang tidak valid akan memunculkan pesan error, nama penyakit tidak valid. Pada pengujian, jika masukan valid, maka akan muncul hasil prediksi berupa tanggal prediksi, nama pasien, nama penyakit, hasil prediksi, dan tingkat kemiripannya.

Pada pengujian fitur menampilkan hasil prediksi, pengguna dapat melakukan input kueri tanggal dan atau nama penyakit. Atau dapat juga menampilkan semua kueri. Jika kueri tidak valid maka akan muncul pesan error. Kueri yang valid akan menampilkan hasil pencarian yang sesuai.

Berdasarkan ketiga pengujian fitur tersebut dapat disimpulkan aplikasi yang kami buat telah memenuhi spesifikasi yang diberikan karena menampilkan hasil yang sesuai dengan tiap kasus ujinya.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan implementasi program dan eksperimen yang telah dilakukan, didapatkan kesimpulan sebagai berikut.

1. Telah dibangun sebuah aplikasi berbasis web interaktif dengan fitur-fitur yang dapat membantu untuk mendeteksi apakah seorang pasien mempunyai penyakit genetik tertentu berdasarkan input sekuens DNA-nya.
2. String Matching dapat dilakukan dengan algoritma KMP, algoritma BM, dan Regex. Algoritma KMP memiliki kompleksitas yang lebih baik dibandingkan algoritma BM dalam hal DNA Pattern Matching dikarenakan variasi hurufnya relatif lebih sedikit. Sementara itu, Regex dapat diaplikasikan sebagai sanitasi input untuk memastikan setiap inputan dari pengguna selalu valid.

5.2 Saran

Berdasarkan implementasi program dan eksperimen yang telah dilakukan, adapun saran dari penulis sebagai berikut.

1. Alokasikan waktu sebaik mungkin dalam pengerjaan tugas besar ini dikarenakan pengembangan aplikasi berbasis web memakan waktu yang cukup besar. Pengembangan aplikasi harus dilakukan secara iteratif dan paralel, sehingga harus bisa bertanggung jawab dalam pengembangan bagian *backend*, *frontend*, dan *library*.
2. Perbanyak mencari sumber belajar terutama dalam mempelajari bahasa baru Go.
3. Dibutuhkan kreativitas dan keinginan untuk berkreasi dalam mengerjakan tugas besar ini.

5.3 Komentar dan Refleksi

Berdasarkan implementasi program dan eksperimen yang telah dilakukan, adapun beberapa hal yang kami sadari dan ingin memberi komentar sebagai berikut.

1. Kami mengetahui bagaimana ilmu informatika diaplikasikan dalam ilmu biologi. Kami menyadari teknologi merupakan cara yang pintar untuk melakukan prediksi dan tentu dapat menyelamatkan seseorang jika suatu penyakit dapat diprediksi lebih awal.

2. Kami mempelajari bagaimana dasar membangun aplikasi berbasis web dari segi *backend* maupun *frontend* dan juga mengimplementasikan algoritma dan metode yang telah diajarkan.

LINK REPOSITORY DAN VIDEO

Repository GitHub

https://github.com/dennisheraldi/Tubes3_13520020

Video Demo

https://www.youtube.com/watch?v=H0G9fCfD1so

Link Deployment

https://dnamatch-fend.netlify.app/

Untuk Keperluan Basis Data dan API :

db: 4isuGYYya5:Xxmdskrzr6@tcp(remotemysql.com:3306)/4isuGYYya5?

api: https://dnamatch-api.herokuapp.com/

DAFTAR PUSTAKA

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

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