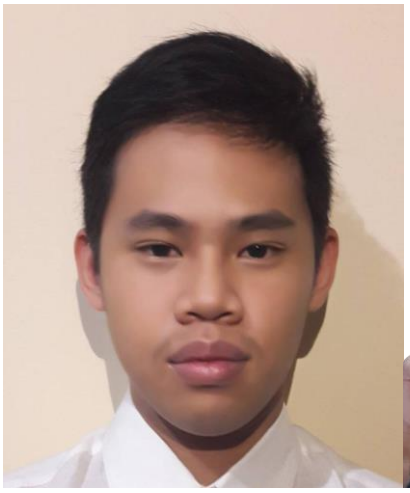
PENERAPAN *STRING MATCHING* DAN *REGULAR EXPRESSION* DALAM DNA *PATTERN MATCHING*

LAPORAN TUGAS BESAR III

Diajukan untuk memenuhi Tugas Besar III
IF2211 Strategi Algoritma



Oleh

Fransiskus Davin Anwari	13520025
Damianus Clairvoyance Diva Putra	13520035
Nayotama Pradipta	13520089

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI 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.

Fitur-Fitur Aplikasi:

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

BAB II
LANDASAN TEORI

1. Deskripsi Singkat Algoritma KMP, BM, dan Regex

a. KMP

Algoritma KMP atau Knuth-Morris-Pratt merupakan salah satu algoritma string matching dengan urutan kiri ke kanan. Pergeseran pada algoritma KMP dilakukan berdasarkan pola maksimal untuk menghindari perbandingan yang tidak diperlukan. Untuk menerapkan algoritma ini, diperlukan sebuah fungsi pinggiran/*border function*. Border function merupakan ukuran prefix P[0...k] yang juga suffix dari P[1..k]. Kompleksitas waktu algoritma KMP adalah O(m+n).

b. Boyer Moore

Algoritma Boyer Moore merupakan algoritma string matching berdasarkan dua teknik, yaitu *looking-glass* dan *character-jump*. Pencocokan karakter dengan Boyer Moore dimulai dari yang paling terakhir. Sebelum pencocokan dimulai, diperlukan sebuah fungsi bernama *Last Occurrence Function* yang menerima pola dan mengembalikan sebuah tabel berisi index terakhir masing-masing alfabet pada pola ditemukan.

c. Regex

Regular Expression atau Regex merupakan sebuah rangkaian karakter yang mendefinisikan sebuah pola pencarian dengan tujuan untuk melakukan pencocokan. Pada tugas besar ini, Regex digunakan untuk pencocokan input DNA dan searching hasil prediksi.

2. Penjelasan Singkat Mengenai Aplikasi Web yang Dibangun

Pada bagian backend, digunakan bahasa Go dengan Gin sebagai HTTP web framework. Backend berisi router-router GET dan POST untuk melakukan query terhadap database menggunakan MYSQL DBMS. Database hanya terdiri atas dua tabel yaitu jenis penyakit dan hasil prediksi, disesuaikan dengan spesifikasi tugas. Untuk bagian front end, ...

BAB III
ANALISIS PEMECAHAN MASALAH

1. Langkah Penyelesaian Masalah Setiap Fitur

Fitur pencarian String Matching diimplementasikan pada sebuah folder yang berisi dua tiga buah file: *boyerMoore*.go, *kmp*.go, dan *fileProcess*.go. Boyer Moore diimplementasikan dengan bantuan fungsi build last yang menerima sebuah pola dna penyakit dan mengembalikan array of integer yang berisi indeks terakhir sebuah karakter ditemukan pada disease. Fungsi StartBoyerMoore menerima sebuah string dna pasien, string penyakit yang ingin dicek, serta mengembalikan boolean. KMP diimplementasikan dengan bantuan fungsi b atau border function yang menerima sebuah string penyakit dan mengembalikan array of integer. Fungsi StartKMP menerima sebuah string dna pasien, string penyakit yang ingin dicek, serta mengembalikan boolean.

Untuk fitur upload penyakit, program akan menerima sebuah file dan menjalankan regex parsing pada fungsi IsValidString. Apabila valid, maka string akan disimpan pada database. Untuk fitur cek penyakit, user akan diminta untuk mengisi nama, upload file DNA, dan isi penyakit yang ingin dites. Program akan mengecek terlebih dahulu apakah nama penyakit yang ingin di tes ada di database. Jika ada, maka program akan mengecek isi dari file DNA dan melakukan sanitasi dengan regex IsValidString, jika tidak ada maka program akan mengeluarkan error. Apabila file DNA valid, maka program akan menjalankan algoritma boyer moore dan kmp, update database, dan mengeluarkan hasill pada website. Fitur cari riwayat dilakukan dengan melakukan regex parsing terlebih dahulu. Format yang dapat digunakan adalah date, disease, date & disease, dan disease & date. Apabila input sudah sesuai, maka program akan melakukan query pada database untuk mengembalikan semua hasil riwayat yang ada pada tabel hasilriwayat.

2. Fitur Fungsional dan Arsitektur Aplikasi Web yang Dibangun

Fitur fungsional yang dibangun meliputi Boyer Moore string matching, KMP string matching, get file from user, regex parsing, query to database, routing, dll. Algoritma string matching telah dijelaskan pada bagian 3.1 sehingga tidak akan dibahas lebih lanjut pada bagian ini. Regex parsing diterapkan pada file fileProcess. File tersebut terdiri atas beberapa fungsi untuk memastikan bahwa input file dan user sudah sesuai format. Fungsi IsValidString digunakan untuk mengecek file DNA, fungsi IsValidSearchDiseaseOnly digunakan untuk mengecek input pencarian hanya penyakit, fungsi IsValidSearchDateOnly digunakan untuk mengecek input pencarian hanya tanggal, fungsi IsValidSearchDateAndDisease digunakan untuk mengecek input pencarian tanggal dan penyakit (atau kebalikannya). DBMS yang digunakan adalah mysql, dan query diterapkan pada router GET dan POST yang berada di file main.go.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

1. Spesifikasi Teknis Program (Struktur Data, Fungsi, dan Prosedur)

A. Struktur Data

```
// Data Pada Go
type jenispenyakit struct {
      Id_penyakit               int       `json:"id_penyakit"`
      Nama_penyakit             string    `json:"nama_penyakit"`
      Rantai_dna                string    `json:"rantai_dna"`
}
type hasilprediksi struct {
      Tanggal_prediksi          string    `json:"tanggal_prediksi"`
      Nama_pasien               string    `json:"nama_pasien"`
      Penyakit_terprediksi      string    `json:"penyakit_terprediksi"`
      Status_terprediksi        string    `json:"status_terprediksi"`
}
```

```
// Data pada SQL

CREATE TABLE jenispenyakit
(id_penyakit int NOT NULL AUTO_INCREMENT,
nama_penyakit varchar(255) NOT NULL,
rantai_dna varchar(255) NOT NULL,
PRIMARY KEY (id_penyakit, nama_penyakit)
);

CREATE TABLE hasilprediksi
(tanggal_prediksi varchar(255) NOT NULL,
nama_pasien varchar(255) NOT NULL,
penyakit_terprediksi varchar(255) NOT NULL,
status_terprediksi varchar(255) NOT NULL
PRIMARY KEY (tanggal_prediksi, nama_pasien, penyakit_terprediksi)
);
```

B. Fungsi dan Prosedur

```
// Boyer Moore
package algorithm

func Min(a, b int) int {
      if a < b {
            return a
      }
      return b
}

func BuildLast(disease string) [128]int {
      // Mengembalikan sebuah array of integer berukuran 128 (Banyaknya
```

```
karakter ASCII).
	// Jika index ke-i bernilai -1, maka tidak ada karakter tersebut di
pattern
	// Jika tidak bernilai -1, maka nilai tersebut menunjukkan index
terakhir karakter di pattern
	var lastOccurence [128]int
	for i := 0; i < 128; i++ {
		lastOccurence[i] = -1
	}
	for i := 0; i < len(disease); i++ {
		lastOccurence[disease[i]] = i
	}
	return lastOccurence
}

func StartBoyerMoore(dna string, disease string) bool {
	// Boyer-Moore Algorithm
	// I.S. Input DNA valid (tidak ada huruf kecil, tidak ada huruf
selain AGCT, tidak ada spasi)
	// F.S. Boolean true or false

	// Deklarasi Variabel
	var lastOccurence [128]int
	var dnaLength int
	var diseaseLength int
	var i int

	// Simpan last occurence untuk masing-masing karakter pada disease
	lastOccurence = BuildLast(disease)

	dnaLength = len(dna)
	diseaseLength = len(disease)
	i = diseaseLength - 1
	if i > dnaLength-1 {
		return false
	}
	var j int
	j = diseaseLength - 1
	for ok := true; ok; ok = (i <= dnaLength) {
		if disease[j] == dna[i] {
			if j == 0 {
				return true
			} else {
				i--
				j--
			}
		} else {
			var lo int = lastOccurence[dna[i]]
			i = i + diseaseLength - Min(j, lo+1)
			j = diseaseLength - 1
		}
	}
	return false
}
```

```
// KMP
package algorithm
```

```
// Border Function / fail
func b(disease string) []int{
	var diseaseLength = len(disease)
	b := make([]int, diseaseLength)
	var prevLength int = 0
	var i int = 1
	b[0] = 0
	for ok := true;ok; ok = (i < diseaseLength){
		if (disease[i] == disease[prevLength]){
			prevLength++
			b[i] = prevLength
			i++
		} else {
			if (prevLength != 0){
				prevLength = b[prevLength - 1]
			} else {
				b[i] = prevLength
				i++
			}
		}
	}
	return b
}



func StartKMP(dna string, disease string) bool{
	var dnaLength int
	var diseaseLength int
	dnaLength = len(dna)
	diseaseLength = len(disease)
	var b = b(disease)
	var i int = 0 // Index disease
	var j int = 0 // Index dna
	// Looping hingga disease ditemukan atau sudah search semua dna
	for ok := true; ok; ok = (j < dnaLength) {
		if disease[i] == dna[j]{
			i++
			j++
		}
		if i == diseaseLength {
			return true
		} else if j < dnaLength && disease[i] != dna[j] {
			if (i != 0){
				i = b[i-1]
			} else {
				j++
			}
		}
	}
	return false
}
```

```
// fileProcess
package algorithm
// File untuk pengecekan regex
import (
	"io/ioutil"
	"path/filepath"
	"regexp"
	"os"
)

func GetDNASequenceFromFile(filename string) string{
	mydir, _ := os.Getwd()
	absPath, _ := filepath.Abs(filepath.Dir(filepath.Dir(mydir)) +
"/test/" + filename)
	b, err := ioutil.ReadFile(absPath)
	var str = ""
	if (err == nil){
		str := string(b)
		return str
	}
	return str
}

func IsValidString(dna string) bool {
	// Regex Parsing
	match, err := regexp.MatchString(`^[ACGT]+$`, dna)
	if err == nil {
		return match
	} else {
		return false
	}
}

func IsValidSearchDiseaseOnly(search string) bool {
	// Regex Parsing
	match, err := regexp.MatchString(`([A-Z][a-z]+|[A-Z]+)`, search)
	if err == nil {
		if match {
			return true
		} else {
			return false
		}
	} else {
		panic(err.Error())
	}
}

func IsValidSearchDateOnly(search string) bool {
	match, err := regexp.MatchString(`^(3[01]|[12][0-9]|0?[1-
9])\s(Jan(uary)?|Feb(ruary)?|Mar(ch)?|Apr(il)?|May|Jun(e)?|Jul(y)?|Aug(us
t)?|Sep(tember)?|Oct(ober)?|Nov(ember)?|Dec(ember)?)\s\d{4}$`, search)
	if err == nil {
		if match {
			return true
		} else {
			return false
```

```
            }
      } else {
            panic(err.Error())
      }
}

func IsValidSearchDateAndDisease(search string) bool {
      match, err := regexp.MatchString(`((3[01]|[12][0-9]|0?[1-
9])\s(Jan(uary)?|Feb(ruary)?|Mar(ch)?|Apr(il)?|May|Jun(e)?|Jul(y)?|Aug(us
t)?|Sep(tember)?|Oct(ober)?|Nov(ember)?|Dec(ember)?)\s\d{4}\s([A-Z][a-
z]+|[A-Z]+))|(([A-Z][a-z]+|[A-Z]+)\s(3[01]|[12][0-9]|0?[1-
9])\s(Jan(uary)?|Feb(ruary)?|Mar(ch)?|Apr(il)?|May|Jun(e)?|Jul(y)?|Aug(us
t)?|Sep(tember)?|Oct(ober)?|Nov(ember)?|Dec(ember)?)\s\d{4})`, search)
      if err == nil {
            if match {
                  return true
            } else {
                  return false
            }
      } else {
            panic(err.Error())
      }
}
```

```
// Router
      router.GET("/disease/:penyakit_terprediksi", func(c *gin.Context) {
            var (
                  hp hasilprediksi
                  hPs []hasilprediksi
                result gin.H
            )
            penyakit_terprediksi := c.Param("penyakit_terprediksi")
            if
(algorithm.IsValidSearchDiseaseOnly(penyakit_terprediksi)){
                  rows, err := db.Query("SELECT tanggal_prediksi,
nama_pasien, penyakit_terprediksi, status_terprediksi FROM hasilprediksi
WHERE penyakit_terprediksi = ?;", penyakit_terprediksi)
                  if err != nil {
                        result = gin.H{
                              "data" : "null",
                              "Error" : err.Error(),
                        }
                  } else {
                        for rows.Next(){
                              err = rows.Scan(&hp.Tanggal_prediksi,
&hp.Nama_pasien, &hp.Penyakit_terprediksi, &hp.Status_terprediksi)
                              hPs = append(hPs, hp)
                              if err != nil {
                                    result = gin.H{
                                          "data" : "null",
                                          "Error" : err.Error(),
                                    }
                              } else {
                                    result = gin.H {
                                          "data" : hPs,
                                    }
```

```
                              }
                        }
                        defer rows.Close()
                  }
                  c.JSON(http.StatusOK, result)
            } else {
                  result = gin.H{
                        "data": "Invalid",
                  }
                  c.JSON(http.StatusOK, result)
            }


      })
      router.GET("/date/:tanggal_prediksi", func(c *gin.Context) {
            var (
                  hp hasilprediksi
                  hPs []hasilprediksi
                result gin.H
            )
            tanggal_prediksi := c.Param("tanggal_prediksi")
            if (algorithm.IsValidSearchDateOnly(tanggal_prediksi)){
                  rows, err := db.Query("SELECT tanggal_prediksi,
nama_pasien, penyakit_terprediksi, status_terprediksi FROM hasilprediksi
WHERE tanggal_prediksi = ?;", tanggal_prediksi)
                  if err != nil {
                        result = gin.H{
                              "data" : "null",
                              "Error" : err.Error(),
                        }
                  } else {
                        for rows.Next(){
                              err = rows.Scan(&hp.Tanggal_prediksi,
&hp.Nama_pasien, &hp.Penyakit_terprediksi, &hp.Status_terprediksi)
                              hPs = append(hPs, hp)
                              if err != nil {
                                    result = gin.H{
                                          "data" : "null",
                                          "Error" : err.Error(),
                                    }
                              } else {
                                    result = gin.H {
                                          "data" : hPs,
                                    }
                              }
                        }
                        defer rows.Close()
                  }
                  c.JSON(http.StatusOK, result)
            } else {
                  result = gin.H{
                        "data": "Invalid",
                  }
                  c.JSON(http.StatusOK, result)
            }
      })
      router.GET("/dnd/:penyakit_terprediksi/:tanggal_prediksi", func(c
*gin.Context) {
            var (
                  hp hasilprediksi
```

```
                  hPs []hasilprediksi
                result gin.H
            )
            penyakit_terprediksi := c.Param("penyakit_terprediksi")
            tanggal_prediksi := c.Param("tanggal_prediksi")
            combine := tanggal_prediksi + " " + penyakit_terprediksi
            if (algorithm.IsValidSearchDateAndDisease(combine)){
                  rows, err := db.Query("SELECT tanggal_prediksi,
nama_pasien, penyakit_terprediksi, status_terprediksi FROM hasilprediksi
WHERE tanggal_prediksi = ? AND penyakit_terprediksi = ?;",
tanggal_prediksi, penyakit_terprediksi)
                  if err != nil {
                        result = gin.H{
                              "data" : "null",
                              "Error" : err.Error(),
                        }
                  } else {
                        for rows.Next(){
                              err = rows.Scan(&hp.Tanggal_prediksi,
&hp.Nama_pasien, &hp.Penyakit_terprediksi, &hp.Status_terprediksi)
                              hPs = append(hPs, hp)
                              if err != nil {
                                    result = gin.H{
                                          "data" : "null",
                                          "Error" : err.Error(),
                                    }
                              } else {
                                    result = gin.H {
                                          "data" : hPs,
                                    }
                              }
                        }
                        defer rows.Close()
                  }
                  c.JSON(http.StatusOK, result)
            } else {
                  result = gin.H{
                        "data": "Invalid",
                        "test": tanggal_prediksi,
                  }
                  c.JSON(http.StatusOK, result)
            }

      })

      router.POST("/insertDisease", func(c *gin.Context){
            id_penyakit, _ := strconv.Atoi(c.PostForm("id_penyakit"))
            nama_penyakit := c.PostForm("nama_penyakit")
            rantai_dna := c.PostForm("rantai_dna")
            if (algorithm.IsValidString(rantai_dna)){
                  query := fmt.Sprintf(`INSERT INTO tesdna.jenispenyakit
(id_penyakit, nama_penyakit, rantai_dna) SELECT * FROM (SELECT '%d',
'%s', '%s') AS tmp WHERE NOT EXISTS (SELECT nama_penyakit FROM
tesdna.jenispenyakit WHERE nama_penyakit='%s') LIMIT 1`, id_penyakit,
nama_penyakit, rantai_dna, nama_penyakit)
                  insert, err := db.Prepare(query)

                  if err != nil {
                        fmt.Print(err.Error())
                  }
```

```
                  _, err = insert.Exec(id_penyakit, nama_penyakit,
rantai_dna)
                  if err != nil {
                        c.JSON(http.StatusOK, gin.H {
                              "Message" : "Insertion of new disease is
unsuccessful",
                        })
                  }
                  c.JSON(http.StatusOK, gin.H {
                        "Message" : "Insertion of new disease is
successful",
                  })
            } else {
                  c.JSON(http.StatusOK, gin.H{"Message": "Invalid Rantai
DNA"})
            }
      })
      router.POST("/insertDNA", func(c *gin.Context){
            nama_penyakit := c.PostForm("nama_penyakit")
            query := fmt.Sprintf(`SELECT rantai_dna FROM jenispenyakit
WHERE nama_penyakit = "%s"`, nama_penyakit)
            var rdna string
            err = db.QueryRow(query).Scan(&rdna)
            if err != nil {
                  c.JSON(http.StatusOK, gin.H {
                        "Message" : err.Error(),
                  })
            } else {
                  nama_pasien := c.PostForm("nama_pasien")
                  rantai_dna := c.PostForm("rantai_dna")
                  if algorithm.IsValidString(rantai_dna) { // Input valid
dan disease yang dicari ada di database
                        now := time.Now()
                        y, m, d := now.Date()
                        date := strconv.Itoa(d) + " " + m.String() + " "
+ strconv.Itoa(y)
                        var diagnosis string
                        if algorithm.StartBoyerMoore(rantai_dna, rdna) &&
algorithm.StartKMP(rantai_dna, rdna) {
                              diagnosis = "True"
                        } else {
                              diagnosis = "False"
                        }
                        query := fmt.Sprintf(`INSERT IGNORE INTO
hasilprediksi VALUES ('%s', '%s', '%s', '%s')`, date, nama_pasien,
nama_penyakit, diagnosis)
                        insert, err := db.Query(query)
                        if err != nil {
                              c.JSON(http.StatusOK, gin.H {
                                    "Message" : "Invalid Input file!",
                              })
                        } else {
                              var hP hasilprediksi
                              hP.Tanggal_prediksi = date
                              hP.Nama_pasien = nama_pasien
                              hP.Penyakit_terprediksi = nama_penyakit
                              hP.Status_terprediksi = diagnosis
                              c.JSON(http.StatusOK, gin.H {
                                    "HasilPrediksi" : hP,
                              })
```

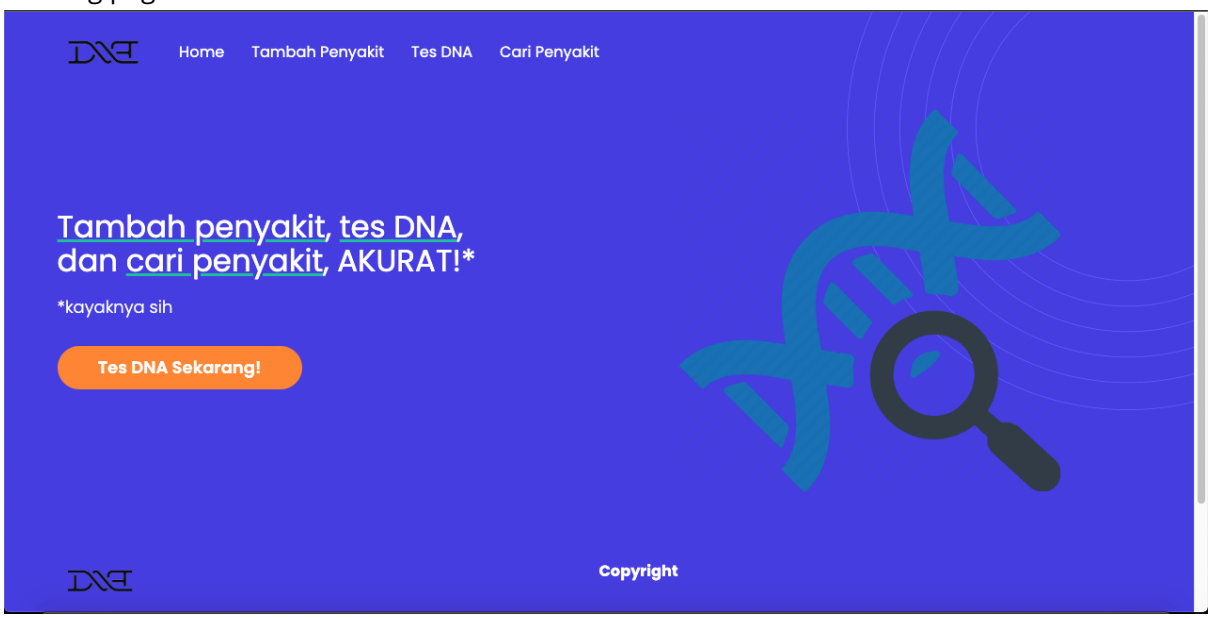
```
                        }
                        defer insert.Close()
                } else {
                        c.JSON(http.StatusOK, gin.H {
                                "Message" : "Invalid Input file!",
                        })
                }
        }
})
```

2. Penjelasan Tata Cara Penggunaan Program (*Interface* Program, Fitur Program, dan Sebagainya)

Berhubung padatnya jadwal pada saat pengerjaan program dan faktor eksternal lain, tidak semua spesifikasi program dapat diimplementasikan secara sempurna. Secara fungsionalitas, program dapat bekerja dengan baik dan sudah complete, akan tetapi secara website, beberapa fitur hanya bisa diimplementasikan dari sisi back end saja ataupun secara langsung lewat terminal. Untuk menggunakan program, pertama jalankan main.go untuk menyalakan router. Selanjutnya, untuk menjalankan website, dapat dilakukan dengan membuka index.html pada browser. Website akan menampilkan fitur pencarian riwayat dna. Ketika input dimasukkan, maka website akan melakukan reroute ke link lain yang ada di backend router. Hasil dari riwayat adalah dalam bentuk JSON. Fitur selain pencarian riwayat dapat dilakukan lewat terminal secara langsung atau menggunakan Postman.

3. Hasil Pengujian (*Screenshot* Antarmuka dan Skenario Kasus)

Landing page:



Database Awal:

id_penyakit	nama_penyakit	rantai_dna
1	Klinefelter	CTAACCTTCATCCTT
2	Trisomy	TCGACTACGACTACGTACG
3	Neurofibromatosis	GACTGTAGCTACAGCT

3 rows in set (0.000 sec)

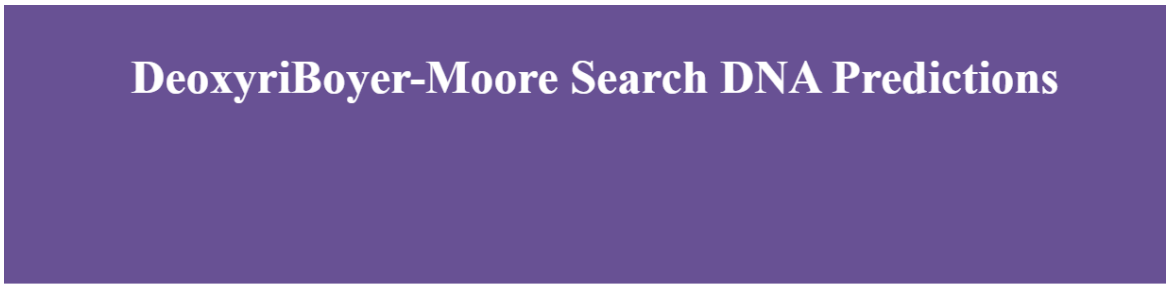
tanggal_prediksi	nama_pasien	penyakit_terprediksi	status_terprediksi
15 December 2020	Seryuu Ubiquitous	Trisomy	True
17 August 2010	Malty Melromarc	Klinefelter	True
24 October 2021	Chizuru Ichinose	Klinefelter	False
24 October 2021	Danzo Shimura	Neurofibromatosis	True

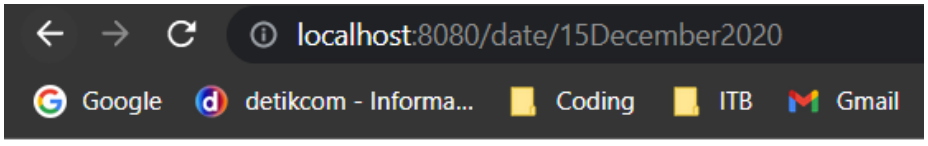
Cari riwayat pada website:

Skenario:

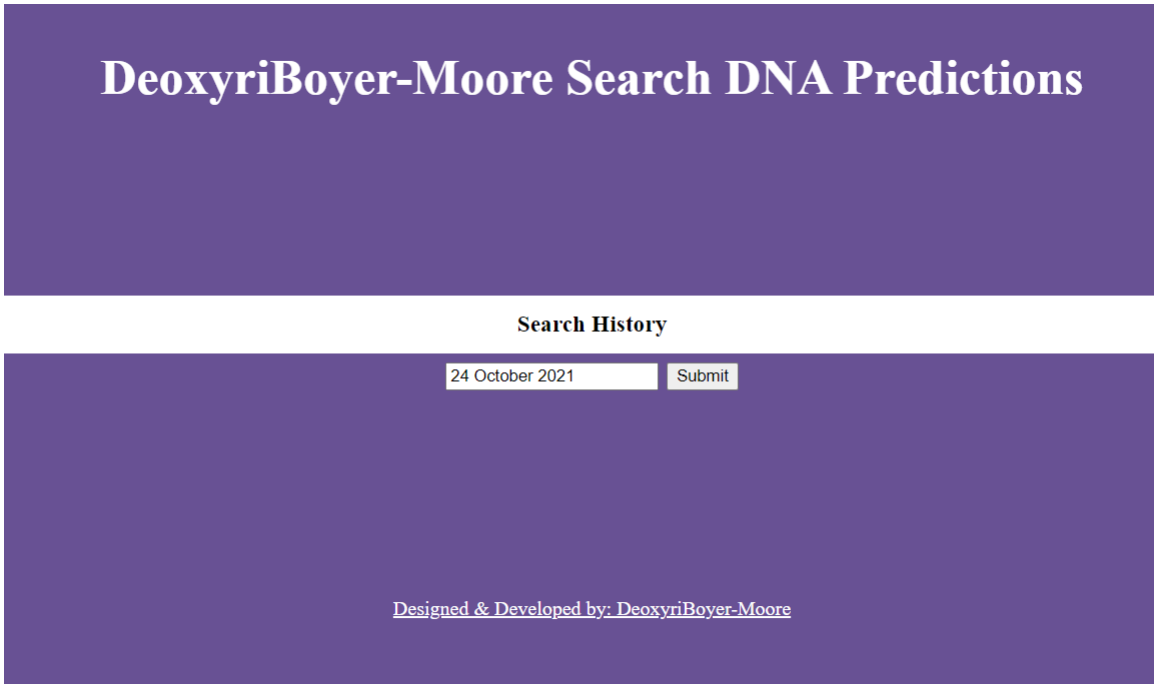
1. Hanya tanggal

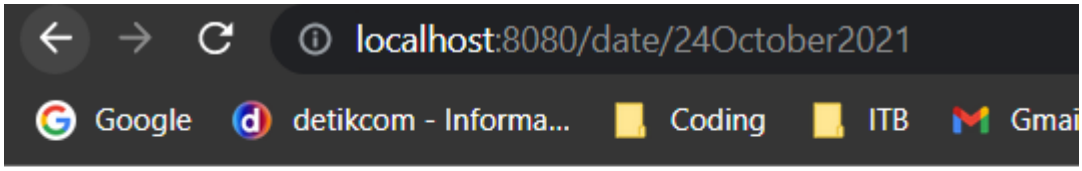
a. Input ada di Database





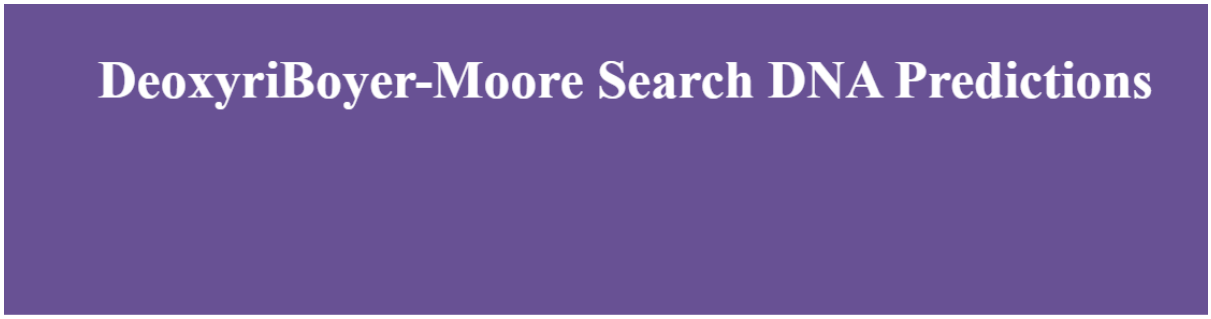

```
{
    "data": [
        {
            "tanggal_prediksi": "15 December 2020",
            "nama_pasien": "Seryuu Ubiquitous",
            "penyakit_terprediksi": "Trisomy",
            "status_terprediksi": "True"
        }
    ]
}
```



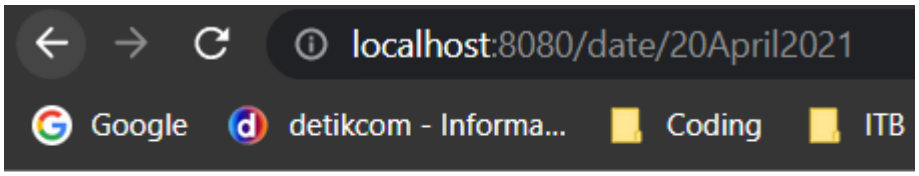



```
{
    "data": [
        {
            "tanggal_prediksi": "24 October 2021",
            "nama_pasien": "Chizuru Ichinose",
            "penyakit_terprediksi": "Klinefelter",
            "status_terprediksi": "False"
        },
        {
            "tanggal_prediksi": "24 October 2021",
            "nama_pasien": "Danzo Shimura",
            "penyakit_terprediksi": "Neurofibromatosis",
            "status_terprediksi": "True"
        }
    ]
}
```


b. Input tidak ada di database

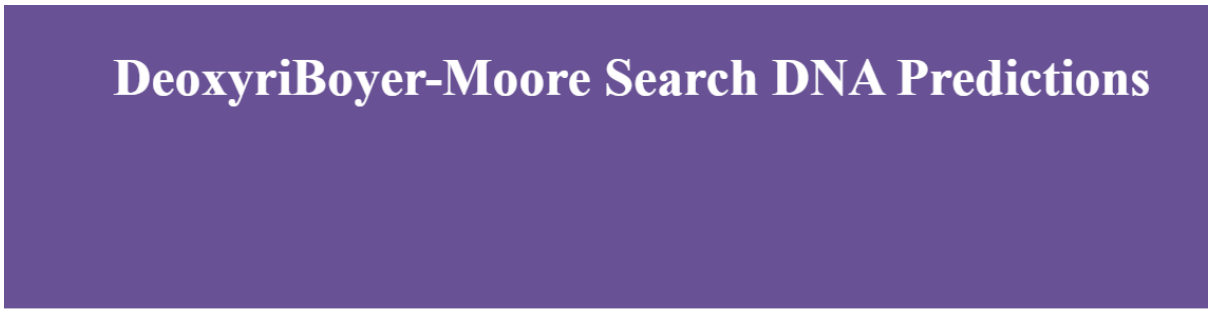


Search History

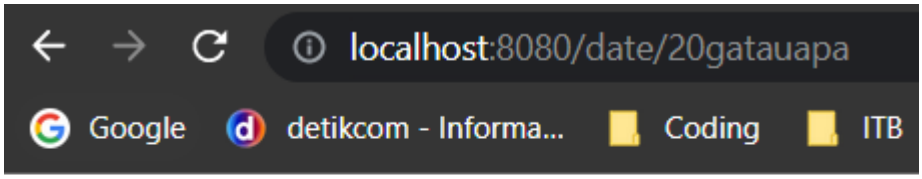



```
null
```

c. Input salah



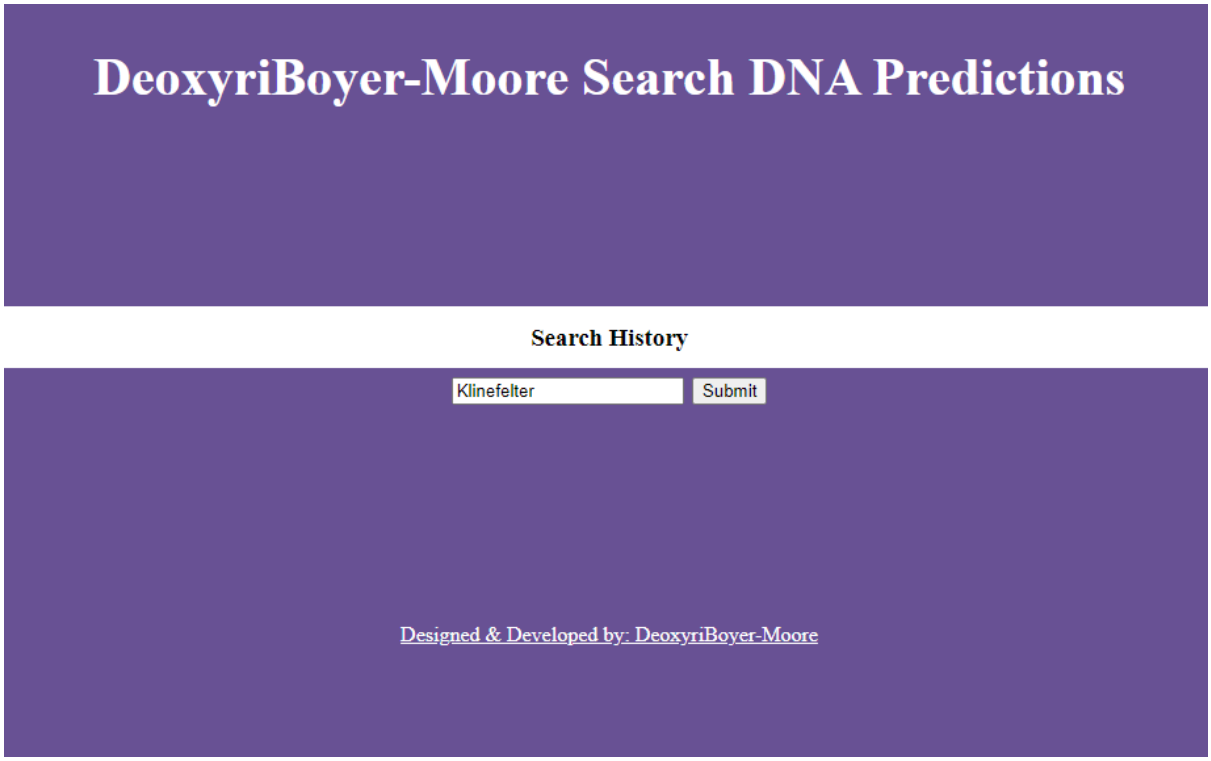
Search History

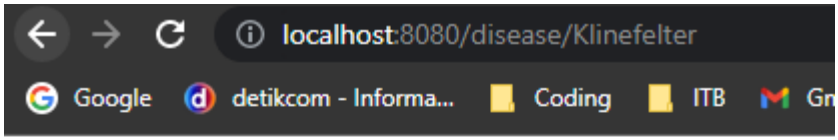



```
{
    "data": "Invalid"
}
```


2. Hanya penyakit

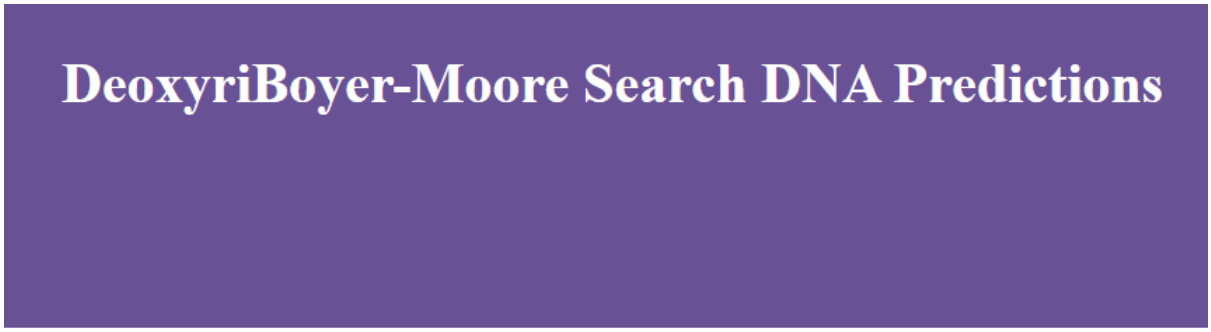
a. Input ada di database



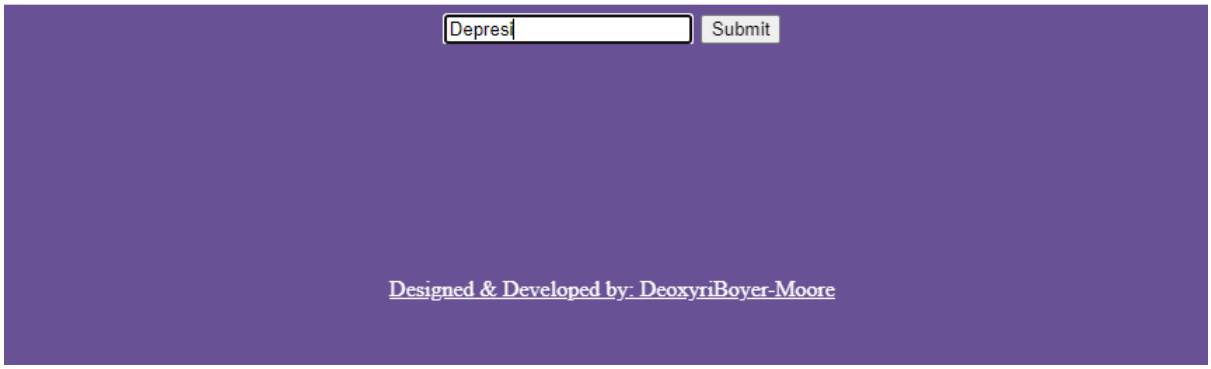


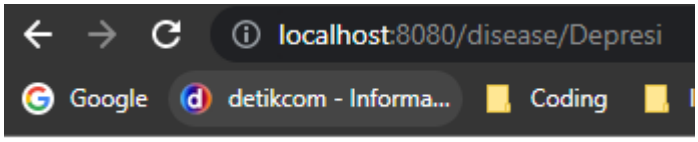

```
{
    "data": [
        {
            "tanggal_prediksi": "17 August 2010",
            "nama_pasien": "Malty Melromarc",
            "penyakit_terprediksi": "Klinefelter",
            "status_terprediksi": "True"
        },
        {
            "tanggal_prediksi": "24 October 2021",
            "nama_pasien": "Chizuru Ichinose",
            "penyakit_terprediksi": "Klinefelter",
            "status_terprediksi": "False"
        }
    ]
}
```


b. Input tidak ada di database



Search History

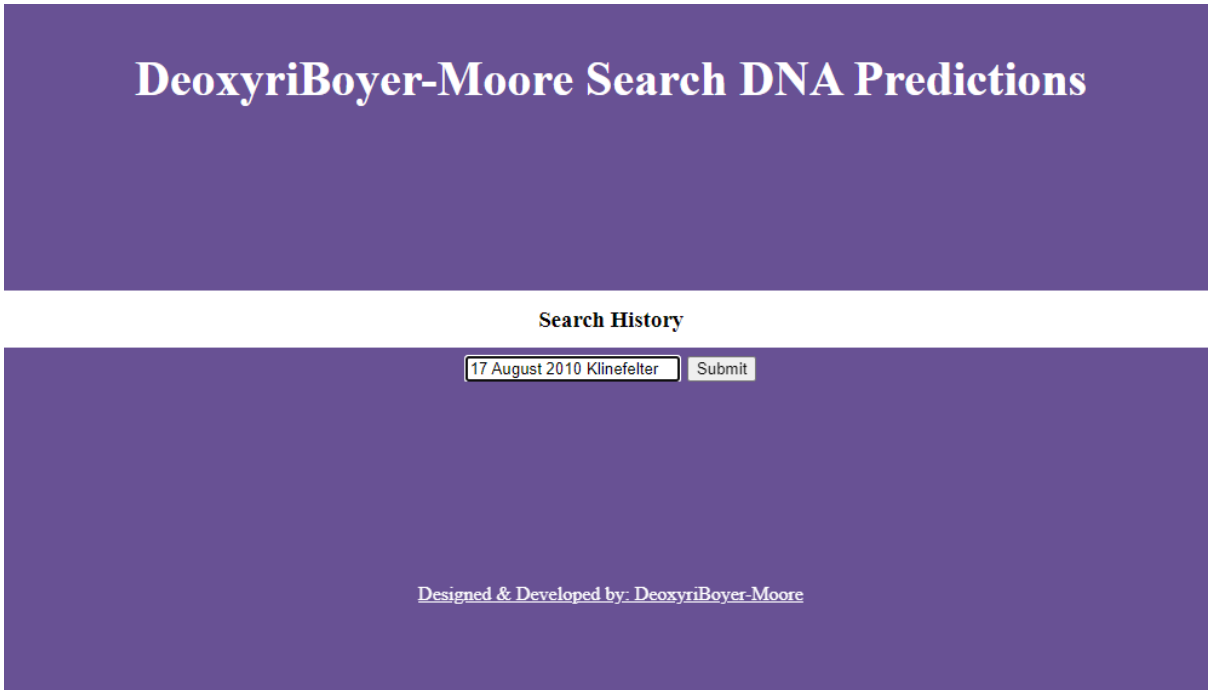


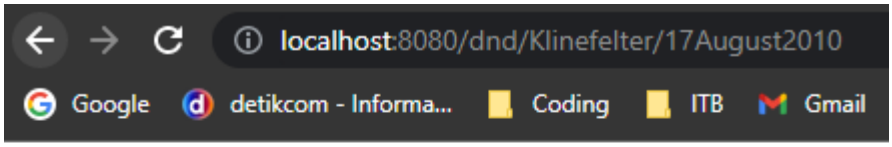



```
null
```

3. Tanggal + Penyakit

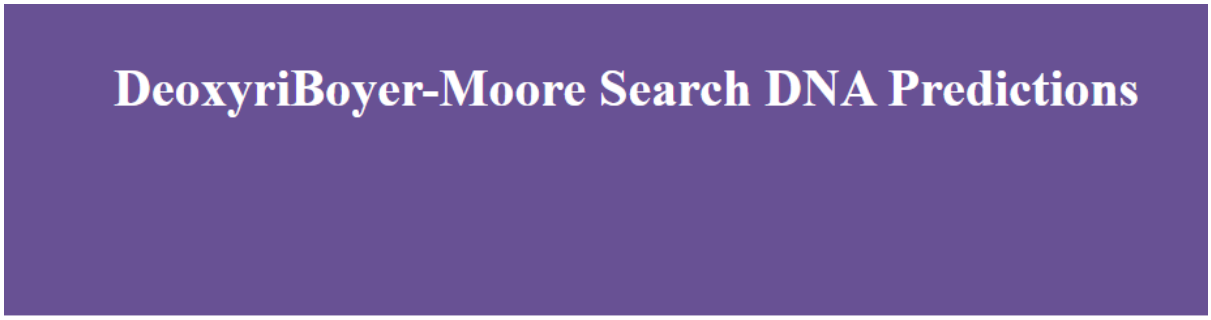
a. Input ada di database



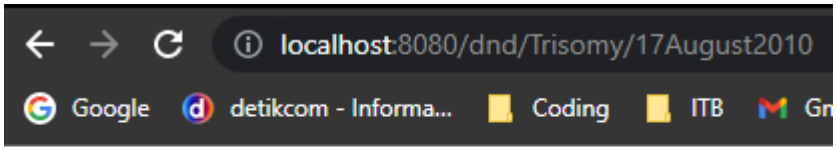



```
{
    "data": [
        {
            "tanggal_prediksi": "17 August 2010",
            "nama_pasien": "Malty Melromarc",
            "penyakit_terprediksi": "Klinefelter",
            "status_terprediksi": "True"
        }
    ]
}
```


b. Input tidak ada di database

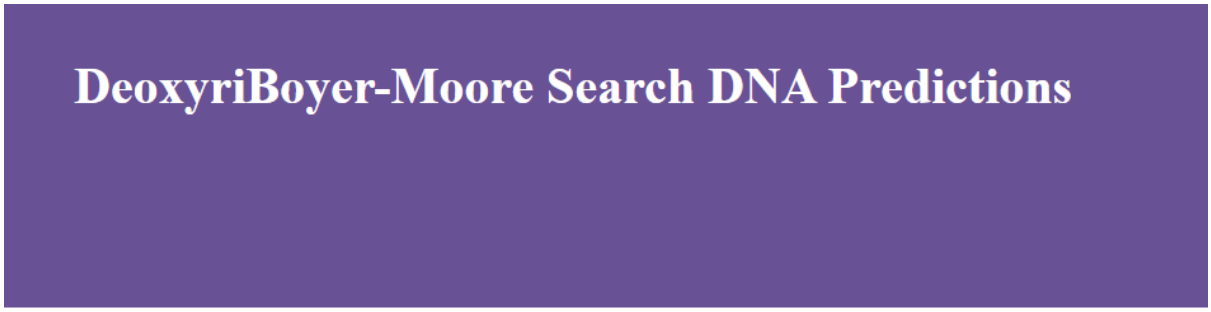


Search History

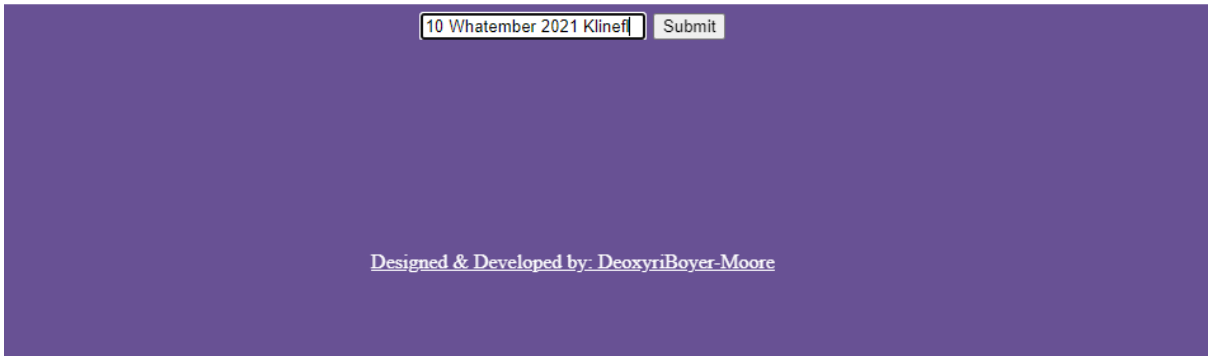


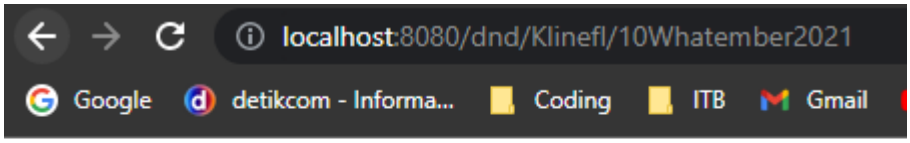

```
null
```

c. Input invalid



Search History

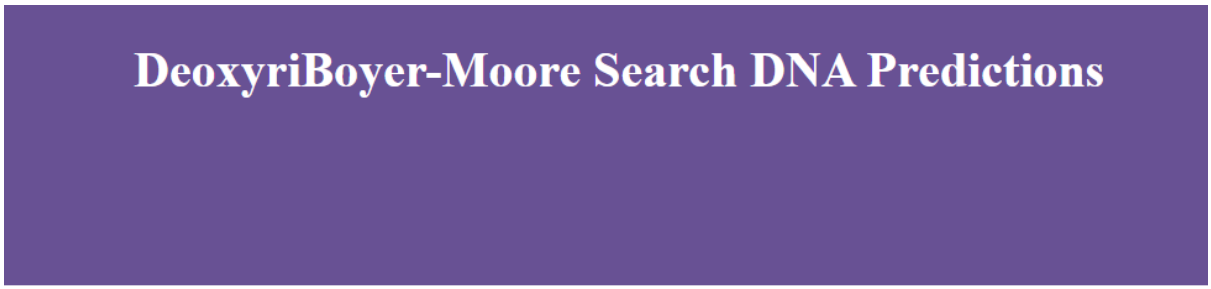


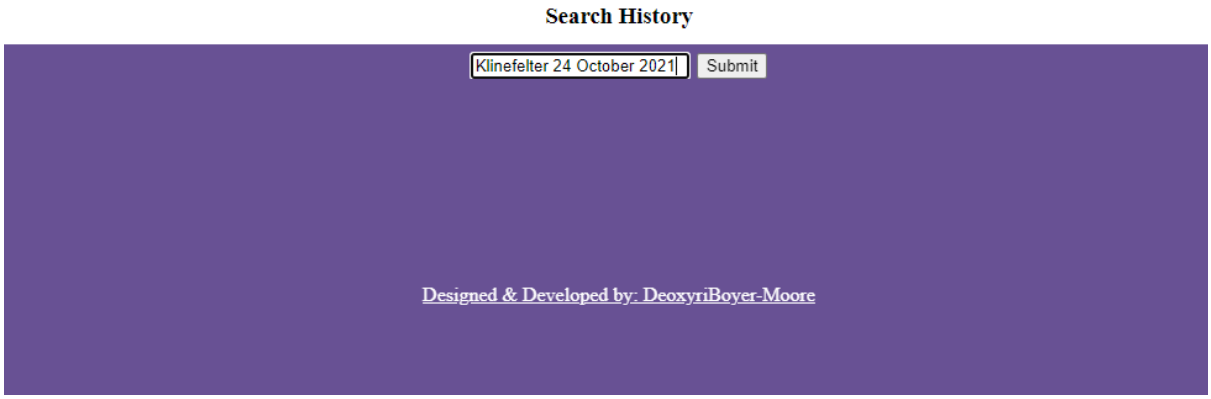


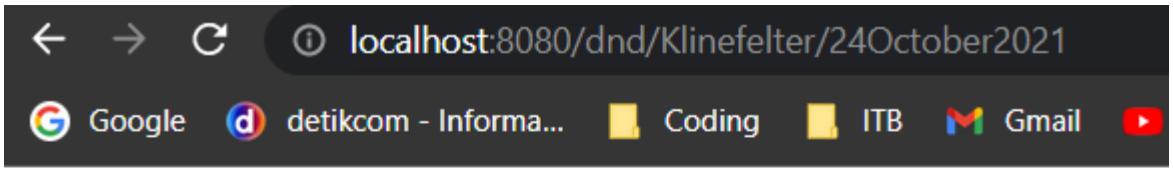

```
{
    "data": "Invalid",
    "test": "10 Whatember 2021"
}
```


4. Penyakit + Tanggal

a. Input ada di database







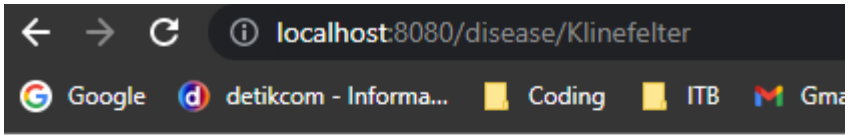

```
{
    "data": [
        {
            "tanggal_prediksi": "24 October 2021",
            "nama_pasien": "Chizuru Ichinose",
            "penyakit_terprediksi": "Klinefelter",
            "status_terprediksi": "False"
        }
    ]
}
```


Fitur tes pasien

Input dna: TTATCGAGCTATCGATCTAGTATACGATCTAACCTTCATCCTT

```
PS C:\Users\Lenovo\Desktop\ITB\IF\Semester 4\Kuliah\Strategi Algoritma\Tubes 3\DeoxyriBoyer-Moore\src\main> go run .
Action:
1. Add disease DNA to database:
2. Check DNA for disease
2
Insert nama:
apaaja
Insert file DNA:
dnaInput.txt
Insert disease to check:
Klinefelter
INSERT SUCCESS
```

Hasil pada website:




```
{
    "data": [
        {
            "tanggal_prediksi": "17 August 2010",
            "nama_pasien": "Malty Melromarc",
            "penyakit_terprediksi": "Klinefelter",
            "status_terprediksi": "True"
        },
        {
            "tanggal_prediksi": "24 October 2021",
            "nama_pasien": "Chizuru Ichinose",
            "penyakit_terprediksi": "Klinefelter",
            "status_terprediksi": "False"
        },
        {
            "tanggal_prediksi": "29 April 2022",
            "nama_pasien": "apaaja",
            "penyakit_terprediksi": "Klinefelter",
            "status_terprediksi": "True"
        }
    ]
}
```


Fitur tambah penyakit

Filename: test.txt

Isi : ACTGTACGATCGATCGATGCA

```
Action:
1. Add disease DNA to database:
2. Check DNA for disease
1
Insert disease name:
test
Insert filename:
test.txt
test
INSERT SUCCESS
```

Hasil pada database

```
MariaDB [tesdna]> SELECT * from jenispenyakit;
+-------------+-------------------+-----------------------+
| id_penyakit | nama_penyakit     | rantai_dna            |
+-------------+-------------------+-----------------------+
|           1 | Klinefelter       | CTAACCTTCATCCTT       |
|           2 | Trisomy           | TCGACTACGACTACGTACG    |
|           3 | Neurofibromatosis | GACTGTAGCTACAGCT      |
|           4 | test              | ACTGTACGATCGATCGATGCA |
+-------------+-------------------+-----------------------+
4 rows in set (0.001 sec)

MariaDB [tesdna]>
```

4. Analisis Algoritma Hasil Pengujian

Algoritma yang diterapkan untuk string matching adalah KMP, Boyer-Moore, dan Regex. Berhubung KMP dan Boyer Moore harus dipakai, maka kami menggunakan kedua algoritma tersebut untuk melakukan string matching sebuah DNA. Algoritma Boyer Moore memiliki kompleksitas O(mn + A) sedangkan algoritma KMP memiliki kompleksitas O(m+n). Oleh karena itu, ketika keduanya digunakan, maka program kami memiliki kompleksitas O(m + n + mn + A). Kompleksitas untuk regex adalah O(n). Algoritma Boyer-Moore dapat menjadi lebih cepat daripada algoritma KMP apabila terdapat banyak karakter yang di skip, akan tetapi sequence DNA sifatnya cenderung random sehingga pernyataan ini tidak dapat menyimpulkan mana yang lebih cepat. KMP memiliki keunggulan dari Boyer-Moore pada string dengan variasi alfabet yang cukup kecil. Sequence DNA hanya terdiri atas 4 variasi karakter (A,C,G,T) sehingga dapat dipastikan bahwa kemungkinan untuk menggunakan ulang pola yang sudah ada cukup tinggi.

BAB V
KESIMPULAN DAN SARAN

1. Kesimpulan

Algoritma KMP, Boyer-Moore, dan Regex masing-masing memiliki keunggulan dan kekurangan yang tergantung pada jenis input. Untuk tugas besar ini, dapat disimpulkan bahwa algoritma KMP memiliki keunggulan terhadap Boyer-Moore karena inputnya hanya memiliki empat variasi karakter. Akan tetapi, baik KMP, Boyer-Moore, maupun Regex dapat menyelesaikan permasalahan string matching dengan cepat apabila menggunakan komputer.

2. Saran

Saran-saran yang dapat kami berikan untuk tugas besar IF2211 Strategi Algoritma Semester 2 2021/2022 adalah:

a. Algoritma yang digunakan pada tugas besar ini agak sedikit memaksa sehingga kurang optimal, program akan berjalan lebih efektif apabila string matching hanya dilakukan menggunakan algoritma KMP
b. Memperjelas beberapa spesifikasi yang ada, terutama yang berhubungan dengan poin a. Kami agak bingung dengan cara penerapan kedua string matching algorithm, apakah user dapat memilih, atau apakah menggunakan salah satu jika ada suatu kondisi, dll.
c. Memberikan jarak deadline dengan tubes-tubes yang lain :)

3. Komentar/Refleksi

Pengerjaan selanjut sebaiknya dilakukan secara bertahap dan mencicil daripada dilakukan semua pada hari-hari ujung dekat deadline. Jaga kesehatan juga, jangan sampai sakit karena akan rugi kehabisan banyak waktu berharga.

BAB VI
TAUTAN VIDEO DEMO DAN *REPOSITORY*

Video demo dapat diakses melalui tautan berikut.
TBD.
Repository GitHub dapat diakses melalui tautan berikut.
https://github.com/NayotamaPradipta/DeoxyriBoyer-Moore

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