IF2211
Strategi Algoritma

LAPORAN TUGAS BESAR

STRATEGI ALGORITMA

PENERAPAN STRING MATCHING DAN REGULAR EXPRESSION DALAM PEMBUATAN CHATGPT SEDERHANA

Kelas Mahasiswa K03

Dosen Pengampu: Ir. Rila Mandala, M.Eng., Ph.D.

Diajukan sebagai tugas besar Mata Kuliah IF2211 Strategi Algoritma pada

Semester II Tahun Akademik 2022/2023

Disusun Oleh:

Kelompok 22 ChatterBot-STIM(AI)

Jason Rivalino	**13521008**
Muhhamad Syauqi Jannatan	**13521014**
Muhammad Zulfiansyah Bayu Pratama	**13521028**

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2022

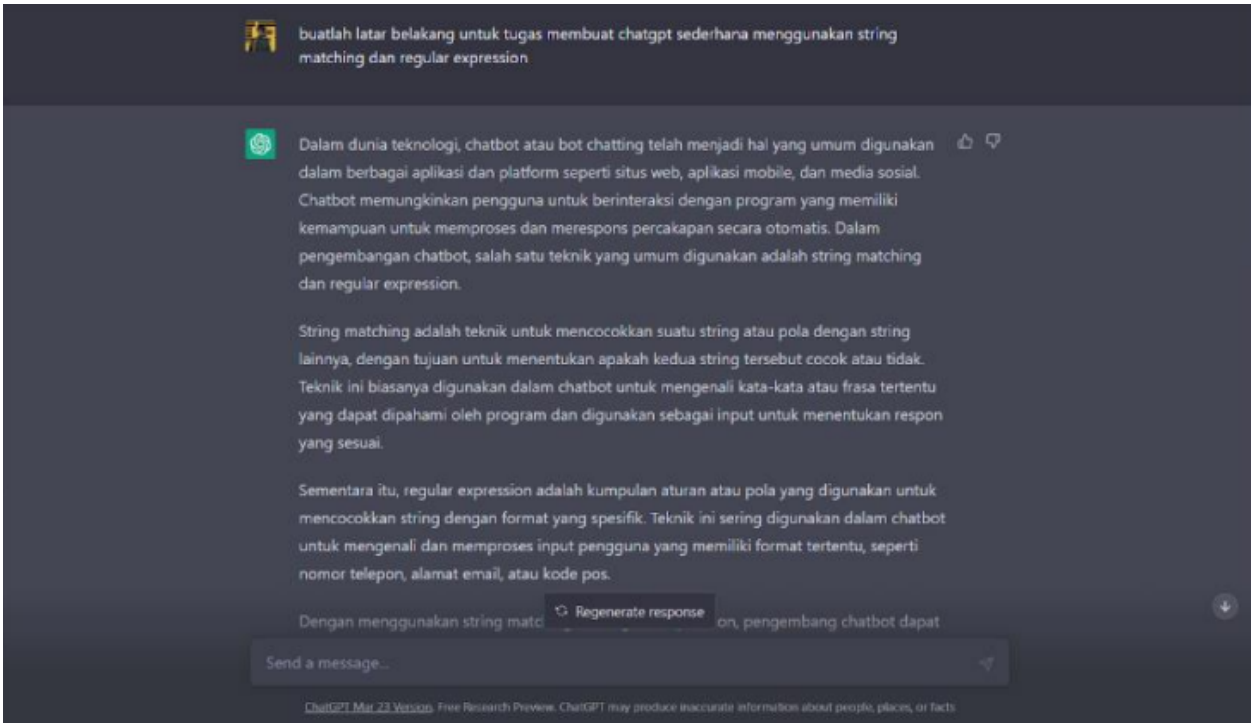
IF2211
Strategi Algoritma

DAFTAR ISI

BAB I

DESKRIPSI TUGAS

Dalam dunia teknologi, chatbot telah menjadi hal yang umum digunakan dalam berbagai aplikasi dan platform seperti situs web, aplikasi mobile, dan media sosial. Chatbot memungkinkan pengguna untuk berinteraksi dengan program yang memiliki kemampuan untuk memproses dan merespons percakapan secara otomatis. Salah satu contoh chatbot yang sedang booming saat ini adalah ChatGPT.



Gambar 1. Ilustrasi Chatbot ChatGPT
Sumber: https://chat.openai.com/chat

Pembangunan chatbot dapat dilakukan dengan menggunakan berbagai pendekatan dari bidang Question Answering (QA). Pendekatan QA yang paling sederhana adalah menyimpan sejumlah pasangan pertanyaan dan jawaban, menentukan pertanyaan yang paling mirip dengan pertanyaan yang diberikan pengguna, dan memberikan jawabannya kepada pengguna. Untuk

mencocokkan input pengguna dengan pertanyaan yang disimpan pada database, kalian bisa menggunakan string matching.

String matching adalah teknik untuk mencocokkan suatu string atau pola dengan string lainnya, dengan tujuan untuk menentukan apakah kedua string tersebut cocok atau tidak. Teknik ini biasanya digunakan dalam chatbot untuk mengenali kata-kata atau frasa tertentu yang dapat dipahami oleh program dan digunakan sebagai input untuk menentukan respon yang sesuai. Sementara itu, regular expression adalah kumpulan aturan atau pola yang digunakan untuk pencocokan string dengan format yang spesifik. Teknik ini sering digunakan dalam chatbot untuk mengenali dan memproses input pengguna yang memiliki format tertentu, seperti nomor telepon, alamat email, atau kode pos.

Dalam tugas besar ini, tugas yang diminta adalah untuk membangun sebuah aplikasi ChatGPT sederhana dengan mengaplikasikan pendekatan QA yang paling sederhana tersebut. Pencarian pertanyaan yang paling mirip dengan pertanyaan yang diberikan pengguna dilakukan dengan algoritma pencocokan string Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM). Regex digunakan untuk menentukan format dari pertanyaan. Jika tidak ada satupun pertanyaan pada database yang exact match dengan pertanyaan pengguna melalui algoritma KMP ataupun BM, maka gunakan pertanyaan termirip dengan kesamaan setidaknya 90% Apabila tidak ada pertanyaan yang kemiripannya di atas 90%, maka chatbot akan memberikan maksimum 3 pilihan pertanyaan yang paling mirip untuk dipilih oleh pengguna.

Adapun fitur-fitur yang akan diimplementasikan dalam aplikasi adalah sebagai berikut:

1. Fitur pertanyaan teks yang didapat dari database

   Mencocokkan pertanyaan dari input pengguna ke pertanyaan di database menggunakan algoritma KMP atau BM.

2. Fitur kalkulator

   Pengguna memasukkan input query berupa persamaan matematika. Contohnya adalah 2*5 atau 5+9*(2+4). Operasi cukup Tambah, kurang, kali, bagi, pangkat, kurung.

3. Fitur tanggal

   Pengguna memasukkan input berupa tanggal, lalu chatbot akan merespon dengan hari apa di tanggal tersebut. Contohnya adalah 25/08/2023 maka chatbot akan menjawab dengan hari senin.

4. Menambah pertanyaan dan jawaban ke database

   Pengguna dapat menambahkan pertanyaan dan jawabannya sendiri ke database dengan query contoh “Tambahkan pertanyan xxx dengan jawaban yyy”. Menggunakan algoritma string matching untuk mencari tahu apakah pertanyaan sudah ada. Apabila sudah, maka jawaban akan diperbaharui.

5. Menghapus pertanyaan dari database

   Pengguna dapat menghapus sebuah pertanyaan dari database dengan query contoh “Hapus pertanyaan xxx”. Menggunakan string algoritma string matching untuk mencari pertanyaan xxx tersebut pada database.

Klasifikasi dilakukan menggunakan regex dan terklasifikasi layaknya bahasa sehari-hari. Algoritma string matching KMP dan BM digunakan untuk klasifikasi query teks. Tersedia toggle untuk memilih algoritma KMP atau BM. Semua pemrosesan respons dilakukan pada sisi backend. Jika ada pertanyaan yang tidak sesuai dengan fitur, maka ditampilkan jawaban “Pertanyaan tidak dapat diproses”.

Seperti ChatGPT, di sebelah kiri disediakan history dari hasil pertanyaan. Cukup tampilkan 5-10 pertanyaan terbaru di toolbar kiri. Sistem history disini disamakan dengan ChatGPT, sehingga satu history yang diklik menyimpan seluruh pertanyaan pada sesi itu. Apabila history diclick, maka akan merestore seluruh pertanyaan dan jawaban di halaman utama.

BAB II

LANDASAN TEORI

I. Teori Dasar Algoritma Knuth-Morris-Pratt (KMP), Boyer-Moore (BM), dan *Regular Expression* (Regex)

a) Teori Dasar Algoritma Knuth-Morris-Pratt (KMP)

Algoritma Knuth-Morris-Pratt atau KMP merupakan algoritma pencarian pola pencocokan string dengan urutan pencarian dimulai dari kiri ke kanan. Algoritma ini mirip dengan algoritma *brute force*, namun memiliki cara kerja yang lebih cerdas dan efisien. Algoritma ini termasuk ke dalam jenis *Exact String Matching Algorithm* yang merupakan pencocokan string secara tepat dengan susunan karakter dalam string yang dicocokkan memiliki jumlah maupun urutan karakter dalam string yang sama. Contohnya adalah kata 'informatika', akan menunjukkan kecocokan hanya dengan kata 'informatika'. Dalam algoritma KMP, kita menyimpan informasi untuk pergeseran yang lebih jauh sehingga berbeda dengan algoritma *Brute Force* yang melakukan pergeseran sebanyak satu karakter saja.

Beberapa definisi yang terdapat pada algoritma KMP antara lain:

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

Adapun *pseudocode* dari algoritma KMP sendiri adalah sebagai berikut:

```
procedure KMPsearch(input m, n : integer, input P : array[1..m] of
char,
                    input T : array[1..n] of char,
                    output idx : integer)
{ Mencari kecocokan pattern P di dalam teks T dengan algoritma Knuth-
Morris-Pratt. Jika ditemukan P di dalam T, lokasi awal kecocokan
disimpan di dalam peubah idx.
  Masukan: pattern P yang panjangnya m dan teks T yang panjangnya n.
           Teks T direpresentasika sebagai string (array of character)
  Keluaran: posisi awal kecocokan (idx). Jika P tidak ditemukan, idx =
-1.
}
Deklarasi
  i, j : integer
  ketemu : boolean
  b : array[1..m] of integer

  procedure HitungPinggiran(input m : integer, P : array[1..m] of
char, output b : array[1..m] of integer)
  { Menghitung nilai b[1..m] untuk pattern P[1..m] }

Algoritma:
 HitungPinggiran(m, P, b)
  j←0
  i←1
  ketemu←false
  while (i ≤ n and not ketemu) do

    while((j > 0) and (P[j+1]≠T[i])) do
      j←b[j]
    endwhile

    if P[j+1]=T[i] then
      j←j+1
    endif
    if j = m then
      ketemu←true
    else
      i←i+1
    endif
 endwhile
 if ketemu then
   idx←i-m+1   { catatan: jika indeks array dimulai dari 0, maka
idx←i-m }
 else
   idx←-1
 endif
```

Algoritma KMP sendiri memiliki kompleksitas sebesar O(m+n) dengan O(m) menyatakan kompleksitas untuk menghitung fungsi pinggiran dan O(n) menyatakan kompleksitas untuk pencarian *string*.

Kelebihan dari algoritma ini sendiri adalah algoritma ini tidak memerlukan pencarian mundur untuk teks input sehingga efektif untuk memproses *file* yang berukuran besar dari *device* eksternal dan kekurangannya adalah dapat terjadi ketidaktepatan pada algoritma KMP untuk alfabet yang berukuran besar.

b) Teori Dasar Algoritma Boyer-Moore (BM)

Algoritma Boyer-Moore atau BM merupakan algoritma pencarian pola pencocokan string dengan urutan pencarian dimulai dari kanan ke kiri. Adapun teknik pencocokan yang ada dalam algoritma ini dibagi menjadi dua, yaitu:

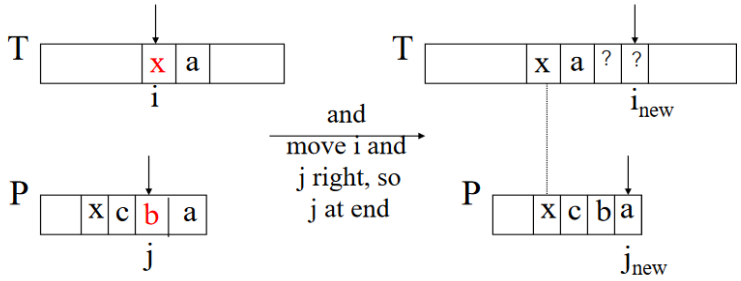
1. The Looking-Glass Technique

   Melakukan teknik pencocokan dengan bergerak dari akhir menuju ke awal (kanan ke kiri), jika teks pada T[i] tidak sesuai dengan *pattern* pada P[j], *pattern* akan digeser kebelakang T[i] dengan teknik *character-jump* karena sudah pasti tidak cocok.

2. The Character-Jump Technique

   Teknik untuk menghitung banyaknya pergeseran *pattern* saat pemeriksaan teks (ketika terjadi kondisi mismatch saat T[i] = = x). Untuk kondisi dari teknik ini sendiri dapat dibagi menjadi tiga antara lain:

   - Jika pattern P mengandung karakter x dengan *last-occurence* berada di kiri *pattern* yang diperiksa, geser P ke kanan dan menyesuaikan *last-occurence* x di P dengan T[i]

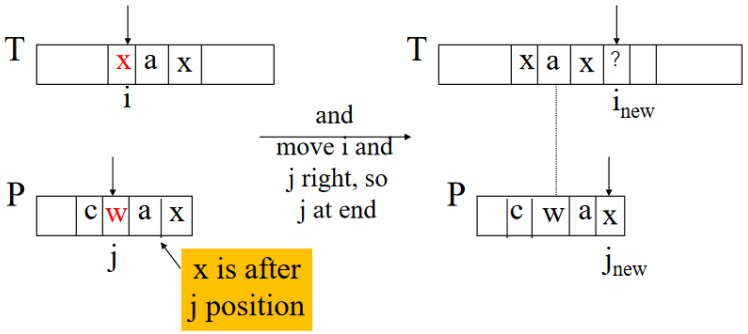


Gambar 2.1 Kondisi pertama Character-Jump

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

Dalam kondisi gambar tersebut, awalnya posisi x pada *pattern* P berada disebelah kiri *pattern* yang diperiksa, sehingga *pattern* P kemudian digeser hingga lokasi mismatch pada teks sejajar dengan lokasi *last-occurance* pada *pattern*.

- Jika pattern P mengandung karakter x dengan *last-occurence* berada di kanan *pattern* yang diperiksa, geser P ke kanan sebanyak satu karakter terhadap teks (menuju ke T[i+1])

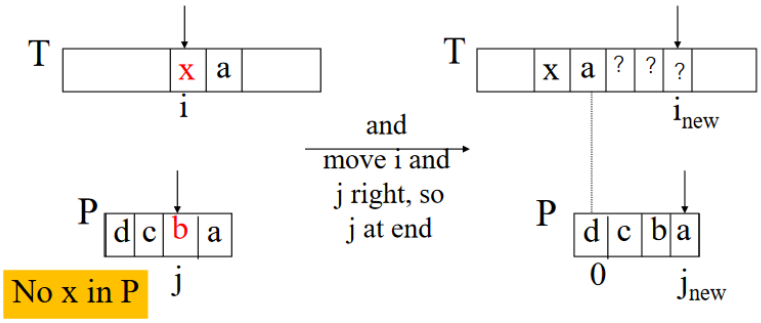


Gambar 2.2 Kondisi kedua Character-Jump

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

Dalam kondisi gambar tersebut, awalnya posisi x pada *pattern* P berada disebelah kanan *pattern* yang diperiksa, sehingga *pattern* P kemudian digeser sebanyak satu karakter terhadap teks.

- Jika tidak ada karakter x dalam P, geser P hingga posisi P[0] berada tepat satu karakter setelah mismatch teks yang diperiksa (sejajar dengan T[i+1]



Gambar 2.3 Kondisi ketiga Character-Jump

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

Dalam kondisi gambar tersebut, tidak ada karakter x pada *pattern* P berada sehingga *pattern* P kemudian digeser hingga karakter *pattern* P yang paling kiri (P[0]) sejajar dengan satu karakter disebelah kanan mismatch di teks T.

Algoritma Boyer-Moore sendiri memiliki kompleksitas sebesar O(mn+A) dengan O(m) menyatakan kompleksitas untuk menghitung fungsi pinggiran dan O(n) menyatakan kompleksitas untuk pencarian *string*, serta A menyatakan variasi karakter. Algoritma ini akan semakin baik jika variasi pada karakter semakin banyak.

c) Teori Dasar *Regular Expression* (Regex)

Regular Expression atau Regex adalah notasi standar yang mendeskripsikan suatu pola (pattern) berupa urutan karakter atau string. Regex digunakan untuk pencocokan string (string matching) dengan efisien. Regex didefinisikan berdasarkan aturan teori bahasa formal.

II. Penjelasan Aplikasi Website yang Dibangun

Aplikasi yang akan dibangun dalam tugas besar ini merupakan aplikasi berbasis *web* dengan pengguna dapat memberikan sebuah pertanyaan dan juga pengguna dapat

memilih metode algoritma yang dipergunakan untuk menjawab pertanyaan tersebut. Pemilihan metode algoritma dilakukan dengan *radio button* dengan mengklik algoritma yang ingin dipilih antara KMP ataupun BM.

Setelah memasukkan pertanyaan dan mengklik tombol submit, program nantinya akan memproses pertanyaan pengguna dan mencocokan pertanyaan yang dimasukkan dengan pertanyaan yang terdapat pada database ataupun pada algoritma program dan kemudian memberikan balikan jawaban berdasarkan data yang terdapat didalam database. Pengguna juga dapat menanyakan perhitungan kalkulator dan juga hari tanggal yang kemudian program akan memprosesnya untuk menghasilkan jawaban yang sesuai.

Selain itu, pengguna juga dapat menambahkan ataupun menghapus pertanyaan yang terdapat di dalam database dengan struktur query tertentu yang kemudian akan diproses dengan string matching. Terakhir, pengguna juga dapat melakukan pengecekan terhadap *history* pertanyaan yang sudah ditanyakan sebelumnya dengan mengklik tombol penyimpanan pada menu *history*.

Untuk aplikasinya, aplikasi ini dibuat dengan memanfaatkan beberapa bahasa pemrograman. Untuk pembuatan frontend dari website program, kelompok kami menggunakan bahasa HTML, CSS, dan JavaScript dengan memakai React framework. Untuk backend dari program, dibuat secara keseluruhan dengan menggunakan bahasa Golang. Sedangkan, untuk penyimpanan data, menggunakan basis data mySQL.

BAB III

ANALISIS PEMECAHAN MASALAH

I. Langkah-langkah Pemecahan Masalah pada Setiap Fitur

1) Langkah pemecahan masalah fitur tanggal
   - Membuat fungsi untuk melakukan pengecekan tanggal apakah valid atau tidak. Pengecekan dilakukan berdasarkan kondisi kalender dan menyesuaikan apakah tahun kabisat atau tidak
   - Membuat fungsi untuk mencari hari dari Senin sampai Minggu
   - Melakukan pengecekan berdasarkan fungsi, jika kondisi terpenuhi, maka fungsi pencarian hari dipanggil dan dikembalikan dalam bentuk hari berdasarkan tanggal

2) Langkah pemecahan masalah fitur kalkulator
   - Membuat fungsi untuk mengecek apakah token merupakan angka dan pengecekan apakah notasi sudah valid atau belum
   - Membuat fungsi untuk membaca operasi dan mengubah infix menjadi postfix
   - Membuat fungsi untuk menghitung operasi matematika dalam bentuk postfix dan mengembalikan hasil perhitungan
   - Melakukan pengecekan berdasarkan fungsi, jika kondisi terpenuhi, ubah infix menjadi postfix dan fungsi perhitungan dipanggil dan dikembalikan dalam bentuk bilangan matematika

3) Langkah pemecahan masalah fitur menjawab pertanyaan dari database
   a) Algoritma Knuth-Morris-Pratt
      - Baca input string yang akan dicari polanya (pat) dan input string di mana pola akan dicari (txt).
      - Konversi kedua string menjadi lowercase menggunakan fungsi strings.ToLower.

- Hitung panjang kedua string pat dan txt, yaitu M dan N.
- Buat array lps berukuran M untuk menyimpan nilai panjang prefiks terpanjang pada pola yang sama di depan dan belakangnya. Nilai lps[0] diinisialisasi dengan 0.
- Hitung nilai lps untuk setiap karakter pada pat menggunakan fungsi computeLPSArray.
- Lakukan pencarian pola pada txt menggunakan variabel i dan j.
- Saat j kurang dari M dan i kurang dari N, bandingkan karakter ke-j pada pat dengan karakter ke-i pada txt.
- Jika kedua karakter sama, maka tingkatkan i dan j masing-masing satu. Jika j sama dengan M, maka kembalikan indeks (i - j) sebagai hasil pencarian.
- Jika kedua karakter tidak sama, maka atur j ke lps[j-1] jika j bukanlah 0, atau tingkatkan i satu.
- Jika tidak ada pola yang ditemukan, kembalikan -1 sebagai hasil pencarian.

b) Algoritma Boyer-Moore
   - Baca input string yang akan dicari polanya (pat) dan input string di mana pola akan dicari (txt).
   - Konversi kedua string menjadi lowercase menggunakan fungsi strings.ToLower.
   - Hitung panjang kedua string pat dan txt, yaitu m dan n.
   - Buat array badchar berukuran 256 (limit) untuk menyimpan nilai indeks terakhir dari setiap karakter pada pola pat.
   - Hitung nilai badchar menggunakan fungsi badCharHeuristic.
   - Lakukan pencarian pola pada txt menggunakan variabel s sebagai indeks mulai pencarian.
   - Saat s kurang dari atau sama dengan (n - m), bandingkan karakter pada pat dari belakang ke depan dengan karakter pada txt mulai dari s+j ke depan.
   - Jika semua karakter pada pat cocok dengan karakter pada txt, kembalikan indeks s sebagai hasil pencarian.

- Jika tidak ada cocokan karakter, maka geser pola sejauh jarak antara karakter pada badchar[txt[s+j]] dengan karakter pat yang sama pada posisi j.
- Jika jarak antara karakter pada badchar[txt[s+j]] dengan karakter pat yang sama pada posisi j kurang dari atau sama dengan 1, maka geser pola sejauh 1 karakter.
- Ulangi langkah 7 hingga 10 sampai seluruh teks telah diproses.
- Jika tidak ada pola yang ditemukan, kembalikan -1 sebagai hasil pencarian.

II. Fitur Fungsional dan Arsitektur Aplikasi Web yang Dibangun

Pengembangan Aplikasi ChatGPT sederhana ini menggunakan berbagai macam arsitektur aplikasi yaitu sebagai berikut:

1) Frontend: HTML, CSS, Javascript (Framework React)
2) Backend: Golang
3) Database: MySQL

Untuk pembagian bagiannya, kelompok kami mengelompokkan antara frontend, backend, dan juga database ke dalam folder yang berbeda-beda antara satu dengan yang lainnya dengan tujuan agar pekerjaan lebih terorganisir.

Untuk folder frontend sendiri memiliki isi sebagai berikut:

a) Folder node_modules: folder yang dibuat oleh npm untuk melacak setiap paket yang telah diinstall
b) Folder public: terdapat file HTML yang dapat diubah untuk mengubah gambar dan title dari tab yang terdapat pada browser yang digunakan
c) Folder src: berisi seluruh program frontend yang telah dibuat. Dalam folder ini sendiri, terdapat beberapa sub-folder dan file dengan isi sebagai berikut:
    i) Folder components: di dalam folder ini, terdapat berbagai macam file javascript yang digunakan untuk membangun frontend dari program. Beberapa file yang ada antara lain:
        - Folder assets: berisi gambar-gambar yang dibutuhkan sebagai *profile* dari aplikasi ChatGPT yang dibangun

- ChatHistory.js: menu *history chat* dari program
- ChatInput.js: menu untuk memasukkan pesan
- ChatItem.js: menu untuk menampilkan *bubble chat* pada program
- ChatList.js: menu untuk *container* dari berbagai macam *bubble chat* dalam program
- RadioButton.js: menu *radioButton* untuk memilih antara algoritma KMP atau BM

ii) Folder styles: di dalam folder ini, terdapat berbagai macam file css yang digunakan untuk mengatur tampilan pada program (*styling*). Beberapa file yang ada antara lain:

- ChatHistory.css: *styling* menu *history chat*
- ChatInput.css: *styling* menu input chat
- ChatItem.css: *styling* menu *bubble chat*
- ChatList.css: *styling container* untuk *bubble chat*
- RadioButton.css: *styling radioButton*

iii) App.js: merupakan main program yang berfungsi untuk menjalankan program yang ada secara keseluruhan

d) File package.json: berfungsi untuk memberikan deskripsi pengaturan dari project Javascript yang dikerjakan. Isi dari file ini memuat nama projek, versi, kontributor, dan lainya.

Untuk folder backend sendiri memiliki isi sebagai berikut:

a) Folder component: di dalam folder ini, terdapat berbagai macam file golang untuk algoritma melakukan pengecekan hari dan tanggal dan juga operasi matematika serta untuk menjawab pertanyaan berdasarkan algoritma KMP dan juga BM. Beberapa file yang ada antara lain:

i) Bm.go: algoritma untuk menjawab pertanyaan dengan Boyer-Moore
ii) Checkdateformat.go: algoritma pengecekan hari dan tanggal
iii) Kmp.go: algoritma untuk menjawab pertanyaan dengan Knuth-Morris-Pratt
iv) Lcs.go: algoritma untuk menghitung tingkat kemiripan dengan Longest Common Subsequence

v) Mathoperation.go: algoritma untuk perhitungan kalkulator

vi) Searchanswer.go: algoritma untuk mencari jawaban dari pengecekan hari tanggal, perhitungan kalkulator, menambah dan menghapus file, serta untuk menjawab pertanyaan berdasarkan database dengan memanfaatkan *regular expression*

b) Folder controllers: di dalam folder ini, terdapat berbagai macam file untuk mengelola list pertanyaan yang terdapat di dalam database. Beberapa file yang ada antara lain:

i) Listquestioncontroller.go: algoritma untuk mengelola data pada tabel data listQuestion.

ii) Riwayatcontroller.go: algoritma untuk mengelola data pada tabel data riwayat

c) Folder models: di dalam folder ini, terdapat berbagai macam file untuk

d) Go.mod: berisi berbagai macam *modules* yang diimport untuk membentuk program

e) Go.sum: generated module file pada program, terbentuk saat menjalankan go.run pada program

f) Main.go: Main program utama untuk menjalankan *backend* secara keseluruhan

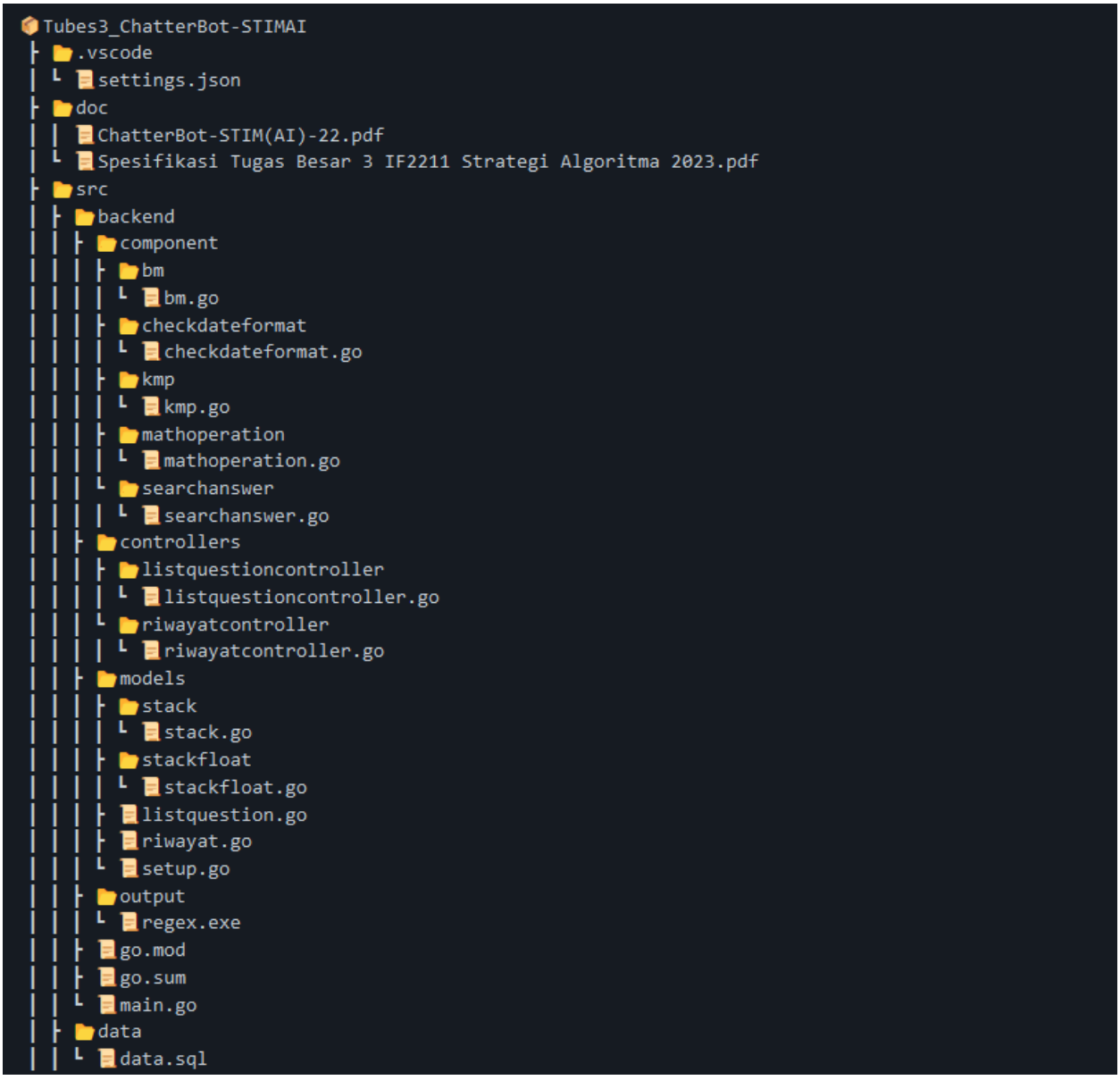
BAB IV

IMPLEMENTASI DAN PENGUJIAN

I. Spesifikasi Teknis Program

A. Struktur Program

Secara garis besar, struktur program untuk aplikasi web ini dapat dilihat pada ilustrasi di bawah sebagai berikut:



Program yang ada sendiri terbagi menjadi dua folder utama yaitu doc yang berisi spesifikasi tugas dan juga laporan tugas besar dalam bentuk pdf dan src yang merupakan folder berisi source code program aplikasi yang dibangun yang terdiri atas folder backend untuk merancang algoritma dan frontend untuk merancang tampilan aplikasi program. Selain itu, terdapat juga file ReadME yang berfungsi untuk memberikan informasi terkait deskripsi program, struktur file, kebutuhan instalasi, cara setup dan mengoperasikan program, dll.

B. Struktur Data

Struktur Data yang ada pada program yang dibuat ini menggunakan MySQL dengan dua tabel data yaitu ListQuestion yang menyimpan pertanyaan dan jawaban serta AskHistory yang menyimpan informasi terkait pertanyaan yang telah ditanyakan.

C. Fungsi dan Prosedur

Beberapa fungsi dan prosedur penting yang ada dalam program kami adalah sebagai berikut:

1) Bm.go

Fungsi/Prosedur	Kegunaan
func badCharHeuristic(str string, size int, badchar [limit]int)	Fungsi untuk membuat tabel heuristik bad character
func BoyerMooreSearch(txt string, pat string) int64	Fungsi untuk melakukan pencarian dengan algoritma Boyer-Moore

2) CheckDateFormat.go

Fungsi/Prosedur	Kegunaan
func isLeapYear(year int) bool	Fungsi untuk menentukan tahun kabisat atau tidak
func IsValidDate(dateStr string) bool	Fungsi untuk menentukan kevalidan tanggal
func GetDayOfWeek(dateStr string) string	Fungsi untuk mendapatkan hari dalam seminggu

3) Kmp.go

Fungsi/Prosedur	Kegunaan
func computeLPSArray(pat string, M int, lps []int)	Fungsi untuk menghitung Longest Prefix Suffix dari pola ‘pat’
func KMPSearch(pat string, txt string)	Fungsi untuk melakukan pencarian

int64	dengan algoritma Knuth-Morris-Pratt

4) Lcs.go

Fungsi/Prosedur	Kegunaan
func lcs(a, b string) int	Fungsi untuk menghitung panjang LCS dengan DP (Dynamic Programming) secara bottom-up
func max(a, b int) int	Fungsi untuk mengembalikan nilai maksimum dari dua bilangan
func Similarity(a, b string) float64	Fungsi untuk menentukan kemiripan antara dua string

5) Mathoperation.go

Fungsi/Prosedur	Kegunaan
func IsNumber(token string) bool	Fungsi untuk mengecek apakah token merupakan angka
func IsInfixValid(infix string) bool	Fungsi untuk mengembalikan nilai maksimum dari dua bilangan
func Similarity(a, b string) float64	Fungsi untuk menentukan kemiripan antara dua string
func Precedence(operator string) int	Fungsi menentukan operator
func InfixToPostfix(infix string) string	Fungsi untuk mengubah pengurutan dari secara infix ke postfix
func CalculatePostfix(postfix string) (float64, error)	Fungsi menghitung operasi matematika yang diberikan dalam bentuk postfix.

6) Searchanswer.go

Fungsi/Prosedur	Kegunaan
func SearchAnswer(question string,	Fungsi untuk mencari jawaban

stringMatching int8) string	berdasarkan masukan pertanyaan

7) listquestion.go

Fungsi/Prosedur	Kegunaan
type ListQuestion struct	Tipe data ini digunakan untuk merepresentasikan sebuah pertanyaan dan jawaban yang disimpan dalam database
func (ListQuestion) TableName() string	Fungsi ini digunakan untuk mengembalikan nama tabel yang digunakan untuk menyimpan data ListQuestion

8) riwayat.go

Fungsi/Prosedur	Kegunaan
type Riwayat struct	Tipe data ini merepresentasikan entitas data riwayat pertanyaan dan jawaban pada program
func (Riwayat) TableName() string	Fungsi ini digunakan untuk mengembalikan nama tabel yang sesuai dengan struktur data Riwayat pada database

9) setup.go

Fungsi/Prosedur	Kegunaan
func ConnectDatabase()	Fungsi ConnectDatabase() digunakan untuk melakukan koneksi ke database MySQL dan melakukan otomatisasi migrasi tabel.

II. Penjelasan Tata Cara Penggunaan Program

Untuk dapat menjalankan program, diperlukan beberapa spesifikasi sebagai berikut:

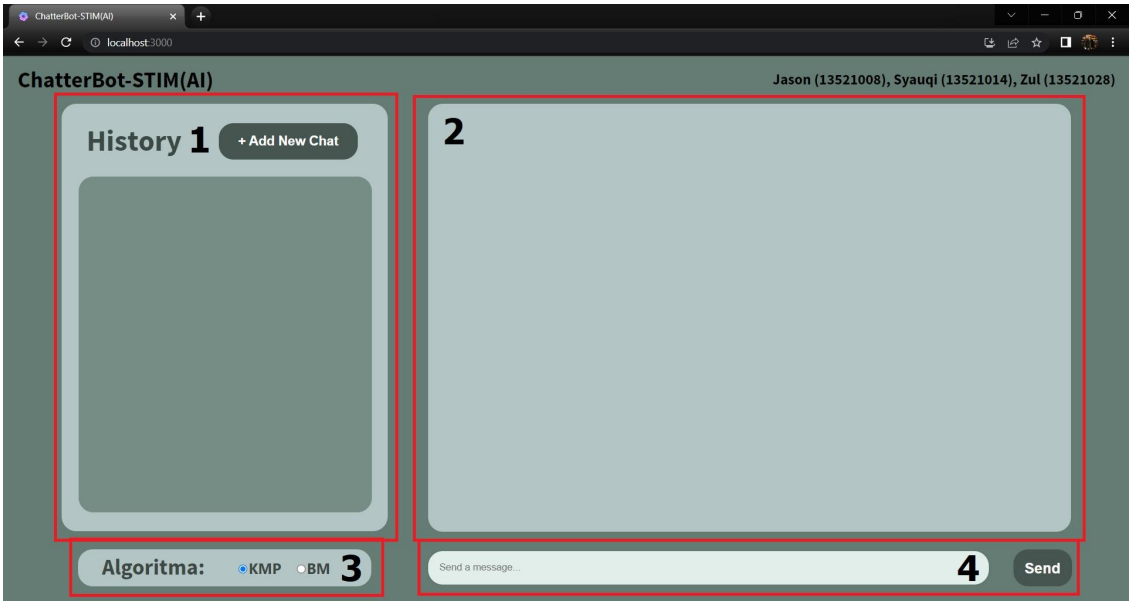
1. Visual Studio Code
2. Bahasa Pemrograman Golang
3. Node.js untuk running frontend

Setelah semua spesifikasi terpenuhi, langkah-langkah untuk dapat menjalankan program adalah sebagai berikut:

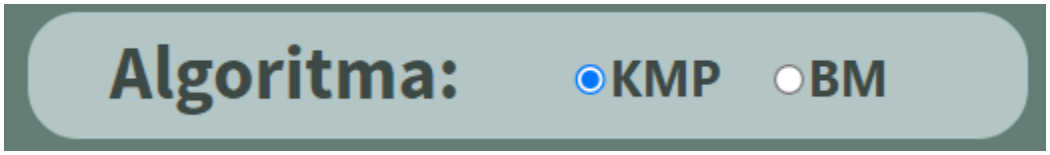
1) Clone repository ini
2) Menjalankan frontend yang ada dengan mengetikkan *npm start* (dijalankan pada direction folder src yang terdapat dalam frontend)
3) Setelah program dijalankan, akan muncul tampilan sebagai berikut:

4) Keterangan yang ada pada tools program
    1. Menu *history* untuk menyimpan *chat* yang sudah dilakukan
    2. Box untuk menampilkan *chat* antara pengguna dengan *bot*

3. *RadioButton* untuk memilih antara algoritma KMP atau BM untuk menjawab pertanyaan dari *database*
4. Menu *textbar* untuk memasukkan chat dan mengirimkannya

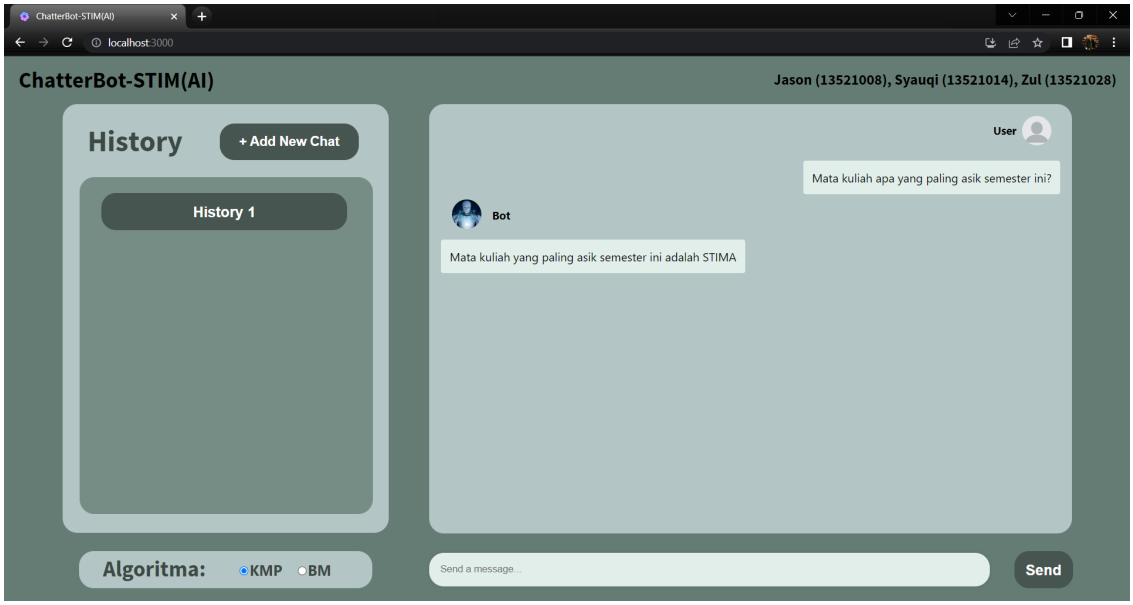


5) Memilih opsi algoritma untuk menjawab pertanyaan dalam *database*

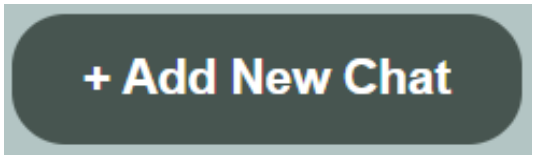


6) Memasukkan chat pada menu *textbar* yang ada dibawah dan kemudian mengirimkan chatnya

7) Program akan menampilkan chat yang dimasukkan pengguna dan juga jawaban yang diberikan oleh program berdasarkan algoritma



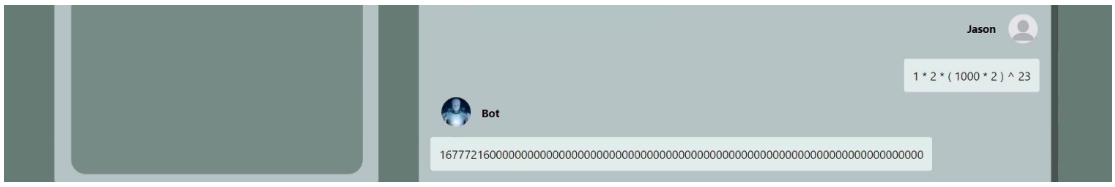
8) Jika pengguna ingin menambahkan sesi *history* baru, bisa mengklik tombol Add New Chat ini dan sesi *history* akan bertambah



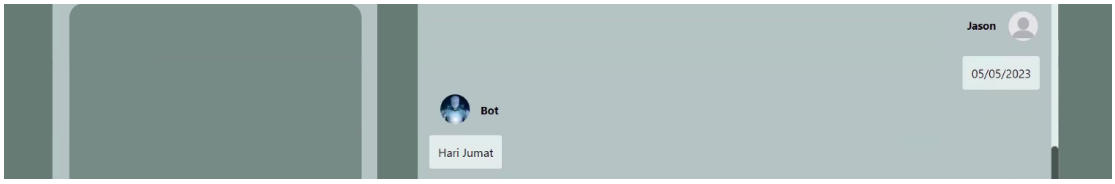
III. Hasil Pengujian Program

1) Pengujian pertanyaan berdasarkan *database* menggunakan Algoritma KMP dan algoritma BM (yang atas algoritma KMP dan yang bawah algoritma BM)

2) Pengujian pertanyaan perhitungan dengan kalkulator



3) Pengujian pertanyaan hari dan tanggal



4) Pengujian memasukkan pertanyaan dan jawaban ke dalam *database*

5) Pengujian menghapus pertanyaan dari *database*

IV. Analisis Hasil Pengujian

Berdasarkan hasil diatas, terdapat perbedaan antara pembacaan antara algoritma KMP dan BM. Hal itu disebabkan karena kedua algoritma pencocokan string (string matching) tersebut memiliki pendekatan yang berbeda dalam mencari pola pada sebuah string. Pada algoritma KMP (Knuth-Morris-Pratt), pola dicocokkan secara bertahap dengan string utama, sedangkan pada algoritma BM (Boyer-Moore), pencocokan dimulai

dari akhir pola dan dilakukan dengan menggeser pola ke posisi berikutnya berdasarkan karakter yang tidak cocok.

Untuk algoritma lain, secara keseluruhan, pengujian berdasarkan algoritma yang ada dan berdasarkan pengecekan juga sudah sesuai baik untuk kalkulator, penanggalan, menambah pertanyaan, ataupun untuk menghapus pertanyaan.

BAB V

KESIMPULAN DAN SARAN

I. Kesimpulan

Dari tugas besar ini dapat disimpulkan hal-hal sebagai berikut,

1) Algoritma Knuth-Morris-Pratt dan Boyer-Moore dapat digunakan untuk melakukan *string matching* untuk mencari jawaban berdasarkan pertanyaan yang ada di dalam *database*
2) Penggunaan *regular expression* dapat diaplikasikan untuk proses perhitungan kalkulator dan juga pencarian informasi tanggal

II. Saran

Saran untuk penyelesaian masalah ini untuk kedepannya adalah sebagai berikut,

1) Terdapat beberapa kondisi pertanyaan tertentu yang bisa memungkinkan terjadinya *double process* dari *database* (dua kondisi dianggap valid), sehingga diperlukan *handling* yang lebih baik lagi dalam pembuatan program
2) Terdapat pertanyaan yang mungkin memiliki struktur dua *query* secara sekaligus yang mengakibatkan kesalahan dalam pembacaan program yang seharusnya bisa dihandle juga dengan lebih baik

III. Refleksi

Dari pengerjaan tugas besar ini, hal penting yang didapat adalah mengenai pentingnya komunikasi dan kerja sama yang baik antar anggota kelompok.. Komunikasi penting mengingat anggota kelompok perlu menghubungkan *front-end* dengan *back-end*, diperlukan komunikasi yang baik agar program dapat menyatu dengan baik. Selain itu, dalam tugas besar ini, juga memberikan pembelajaran mengenai algoritma *string matching* dan *regular expression* yang tentunya bermanfaat dalam berbagai macam aplikasi dalam dunia Teknik Informatika.

IV. Tanggapan

Tugas yang diberikan sudah sangat membantu mahasiswa dalam mengerti terkait algoritma *string matching* dan *regular expression*. Tugasnya sangat relate dengan teknologi yang sering dipakai mahasiswa sekarang haha. Sudah oke dan mantap, tetap dipertahankan dan bahkan ditingkatkan untuk pemberian tugas-tugas kedepannya agar dapat semakin lebih baik lagi.

DAFTAR PUSTAKA

[1] R. Munir (2021). Pencocokan String (String/Pattern Matching) [Powerpoint Slides]. Available:https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

[2] R. Munir (2019). String Matching dengan Regular Expression [Powerpoint Slides]. Available:https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf

[3] R. Munir (2004). Algoritma Pencarian String(String Matching). https://dokumen.tips/download/link/1-string-matching-rinaldi-munir.html (Diakses pada tanggal 26 April 2023)

IF2211
Strategi Algoritma

LAMPIRAN

Pranala Github : https://github.com/jasonrivalino/Tubes3_ChatterBot-STIMAI

Video Demonstrasi : bit.ly/tubes-stima-jason