

TUGAS BESAR III IF2211 STRATEGI ALGORITMA
SEMESTER II TAHUN 2022/2023
Penerapan String Matching dan Regular Expression dalam Pembuatan
ChatGPT Sederhana



Disusun Oleh:

Kelompok 666

13521102 Jimly Firdaus

13521127 Marcel Ryan Antony

13521145 Kenneth Dave Bahana

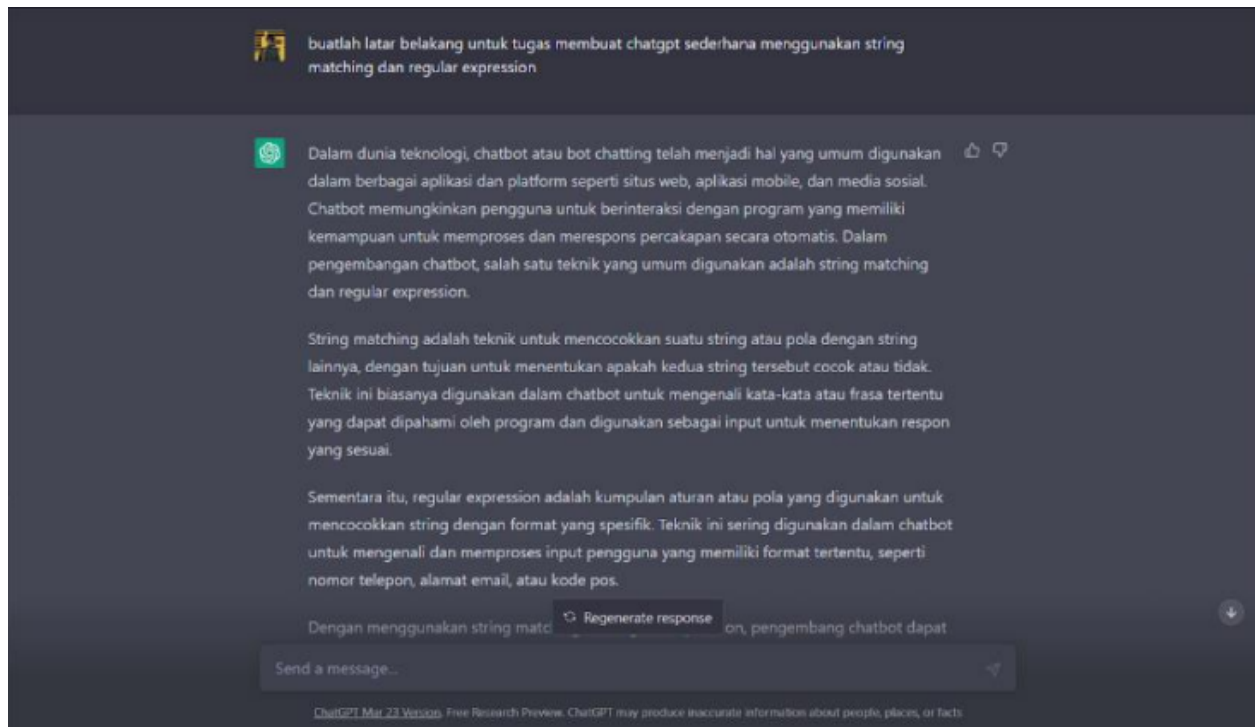
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2023

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 (funfact latar belakang spek ini dari 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.

Deskripsi tugas:

Dalam tugas besar ini, anda diminta 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 (akan dijelaskan lebih lanjut pada bagian fitur aplikasi). **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.

Perhitungan tingkat kemiripan dibebaskan kepada anda asalkan dijelaskan di laporan, namun disarankan menggunakan salah satu dari algoritma Hamming Distance, Levenshtein Distance, ataupun Longest Common Subsequence.

Fitur-Fitur Aplikasi:

ChatGPT sederhana yang anda membuat wajib dapat melakukan beberapa fitur / klasifikasi query seperti berikut:

1. **Fitur pertanyaan teks (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. **Tambah pertanyaan dan jawaban ke database**

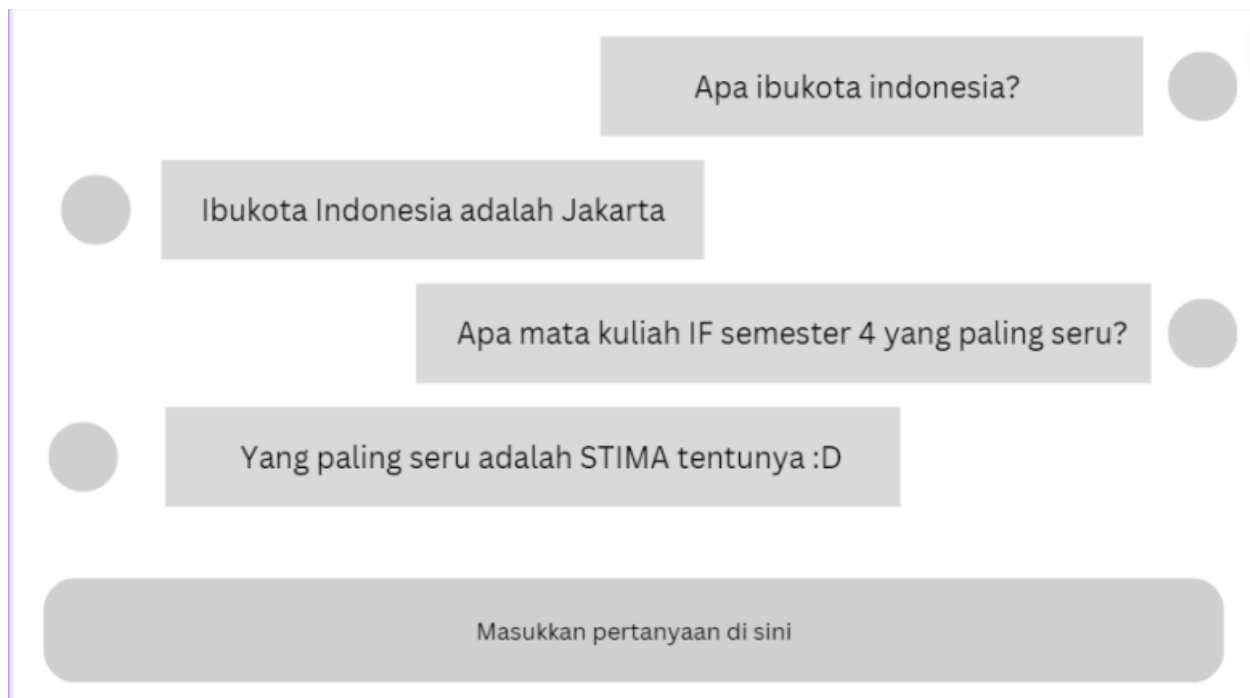
Pengguna dapat menambahkan pertanyaan dan jawabannya sendiri ke database dengan query contoh "Tambahkan pertanyaan xxx dengan jawaban yyy". Menggunakan algoritma string

matching untuk mencari tahu apakah pertanyaan sudah ada. Apabila sudah, maka jawaban akan diperbaharui.

5. **Hapus 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.

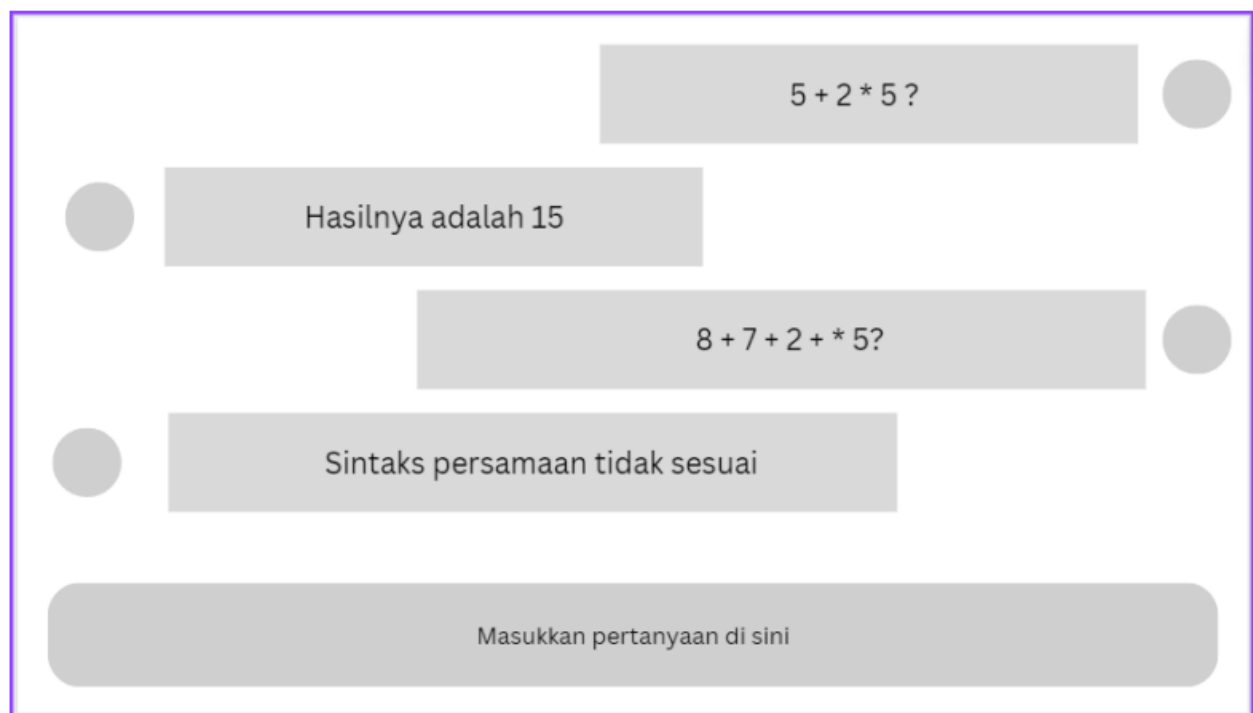
Klasifikasi dilakukan menggunakan **regex** dan terklasifikasi layaknya bahasa sehari - hari. Algoritma string matching KMP dan BM digunakan untuk klasifikasi query teks. Tersedia toggle untuk memilih algoritma KMP atau BM. Semua pemrosesan respons dilakukan pada sisi **backend**. Jika ada pertanyaan yang sesuai dengan fitur, maka tampilkan saja “Pertanyaan tidak dapat diproses”. Berikut adalah beberapa **contoh** ilustrasi sederhana untuk tiap pertanyaannya. (**Note**: Tidak wajib mengikuti ilustrasi ini, tampilan disamakan dengan chatGPT juga boleh)



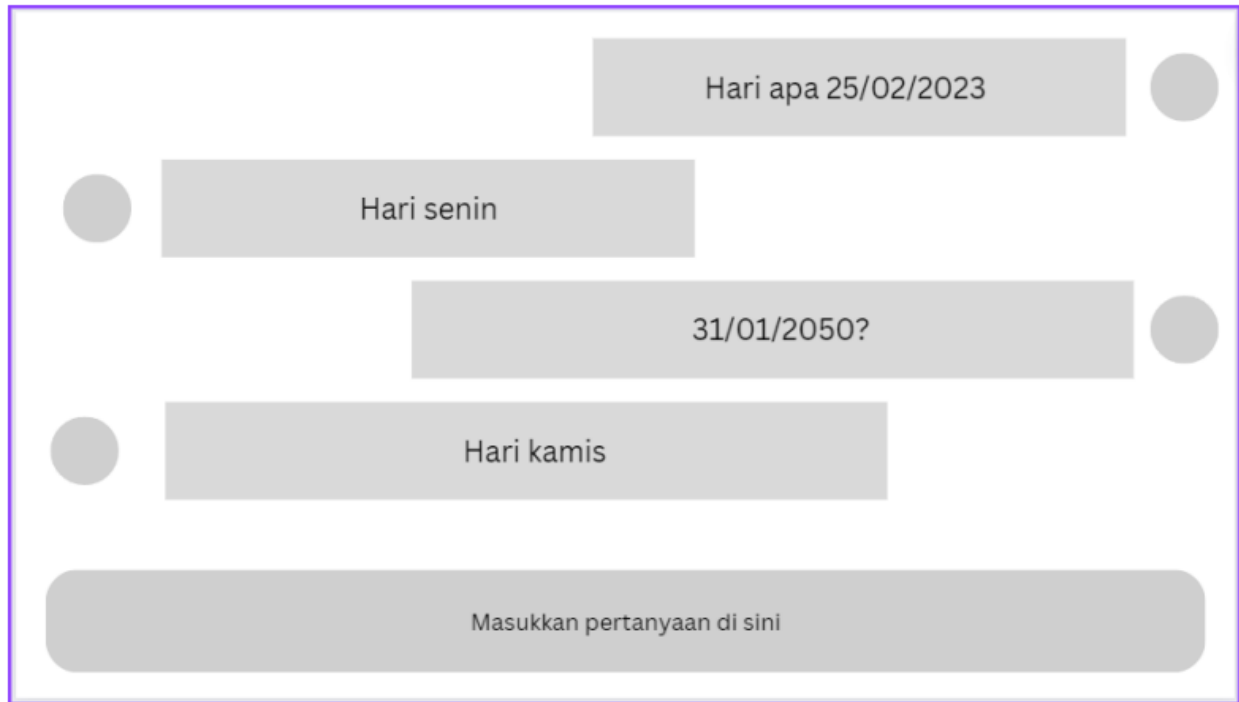
Gambar 2. Ilustrasi Fitur Pertanyaan teks kasus exact



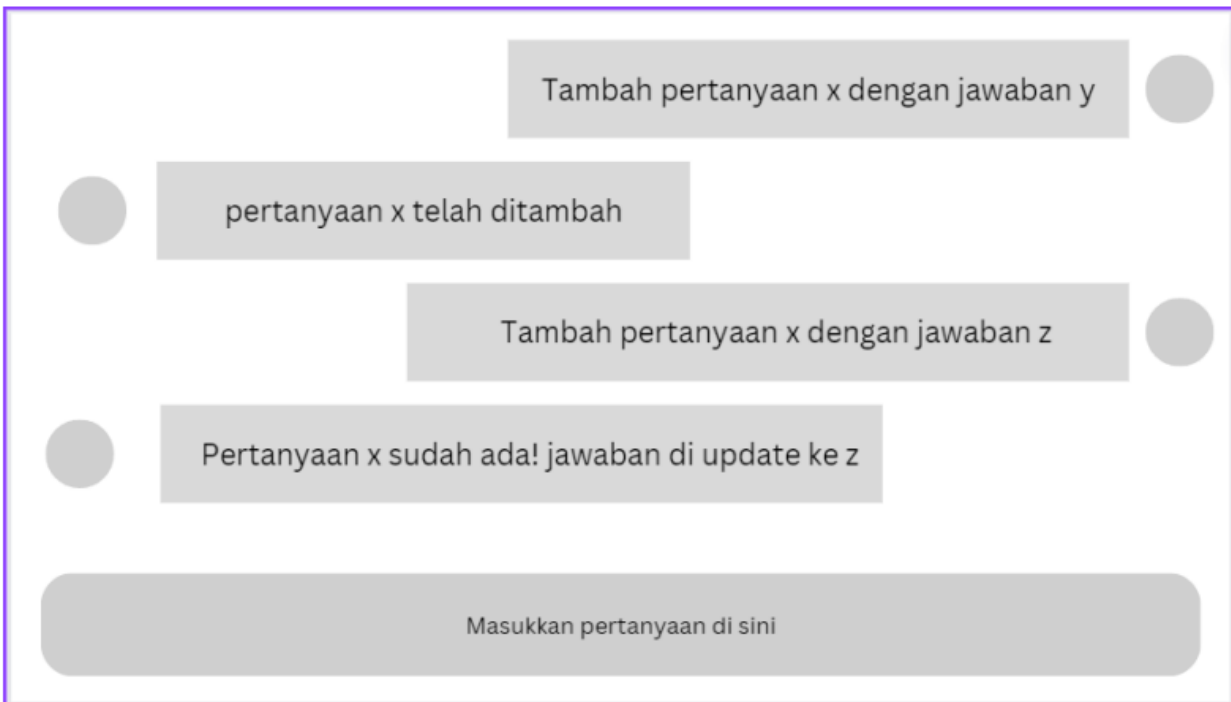
Gambar 3. Ilustrasi Fitur Pertanyaan teks kasus tidak exact



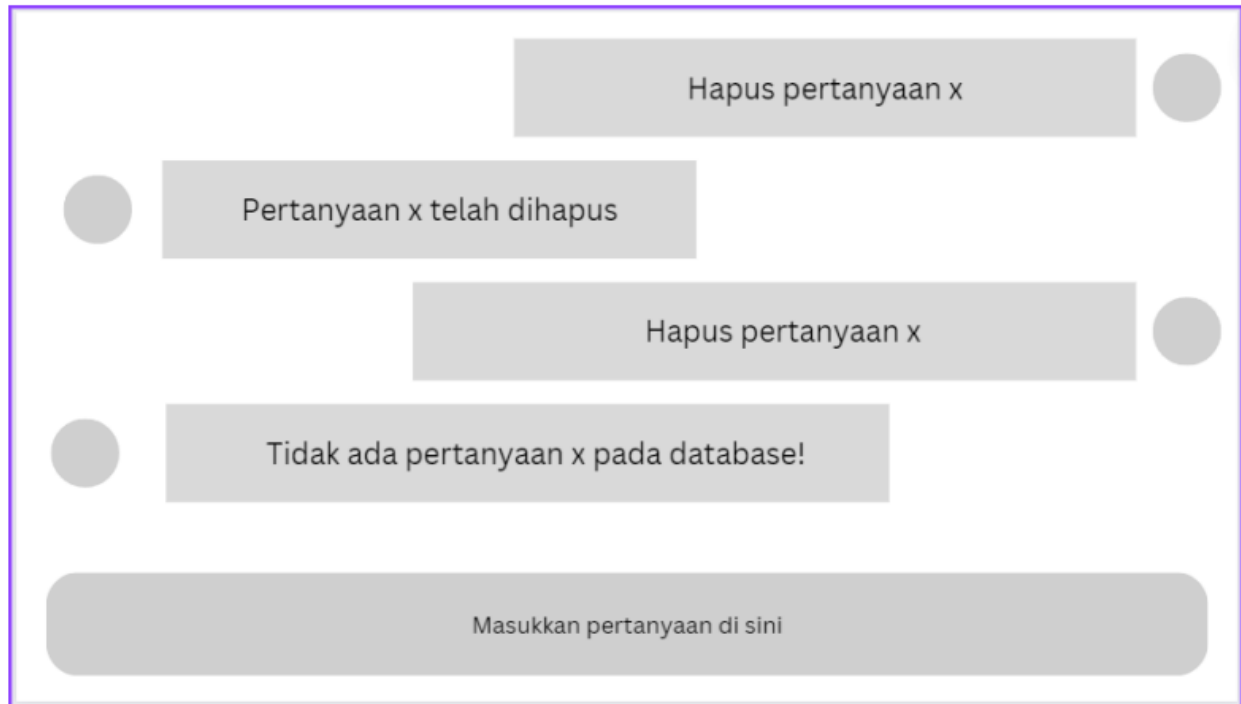
Gambar 4. Ilustrasi Fitur Kalkulator



Gambar 5. Ilustrasi Fitur Tanggal

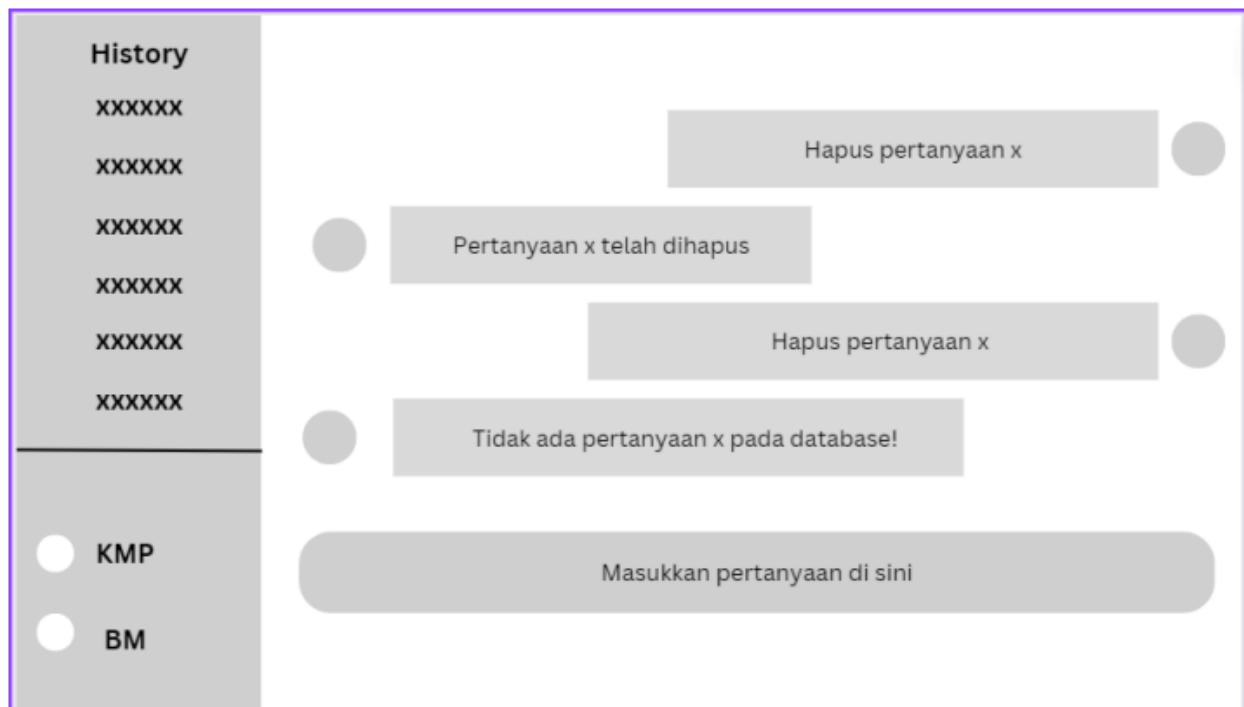


Gambar 6. Ilustrasi Fitur Tambah Pertanyaan



Gambar 7. Ilustrasi Fitur Hapus Pertanyaan

Layaknya **ChatGPT**, di sebelah kiri disediakan **history** dari hasil pertanyaan anda. Cukup tampilkan 5-10 pertanyaan terbaru di toolbar kiri. Perhatikan bahwa 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. Contoh ilustrasi keseluruhan:



Gambar 8. Ilustrasi Keseluruhan

Spesifikasi Program:

1. Aplikasi berbasis website dengan pembagian Frontend dan Backend yang jelas.
2. Implementasi Backend **wajib** menggunakan Node.js / Golang, sedangkan Frontend dibebaskan tetapi **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) dan Regex wajib diimplementasikan pada sisi Backend aplikasi.
5. Informasi yang **wajib** disimpan pada basis data:
 - a. Tabel pasangan pertanyaan dan Jawaban
 - b. Tabel history
6. Skema basis data dibebaskan asalkan mencakup setidaknya kedua informasi di atas.
7. Proses string matching pada tugas ini **Tidak case sensitive**.
8. Pencocokan yang dilakukan adalah dalam satu kesatuan string pertanyaan utuh (misal “Apa ibukota Filipina?”), bukan kata per kata (“apa”, “ibukota”, “Filipina”).

Lain-lain:

1. Anda dapat menambahkan fitur-fitur lain yang menunjang program yang anda buat (unsur kreativitas).
2. Tugas dikerjakan berkelompok, minimal 2 orang dan maksimal 3 orang, boleh lintas kelas namun **tidak boleh sekelompok** dengan **orang yang sama dengan tubes ataupun tucil stima sebelumnya**.
3. Semua kelompok harap mengisi data kelompok mereka pada link <https://bit.ly/KelompokTubes3Stima>
4. Batas akhir pengisian anggota kelompok adalah **16 April, Pukul 22:11.** Mahasiswa yang belum mendapatkan kelompok setelah tanggal ini akan diacak kelompoknya.
5. Anda harus membuat aplikasi dan program ini sendiri kecuali library regex, tetapi belajar dari contoh-contoh program serupa yang sudah ada tidak dilarang (tidak boleh melakukan plagiasi source code dari program orang lain). Program harus dibuat sendiri, tidak boleh sama dengan teman.
6. Program harus modular dan mengandung komentar yang jelas.
7. Dilarang menggunakan kode program yang diunduh dari Internet. Mahasiswa harus membuat program sendiri, tetapi belajar dari program yang sudah ada tidak dilarang.
8. Batas akhir pengumpulan tugas adalah **5 Mei, Pukul 23:59.** Keterlambatan dalam mengumpulkan akan diberi penalti pengurangan skor yang cukup signifikan.

9. Semua pertanyaan menyangkut tugas ini dapat dikomunikasikan lewat QnA yang bisa diakses pada bit.ly/TugasStimaQnA

10. **Bonus (maksimal 10 poin):**

a. Mendeploy aplikasi web yang telah dibangun (hosting provider dibebaskan). Deployment website harus dipertahankan sampai demo tugas besar.

b. Setiap kelompok membuat video aplikasi yang mereka buat kemudian mengunggahnya ke Youtube. Video yang dibuat harus memiliki audio dan menampilkan wajah dari setiap anggota kelompok. Pada waktu demo aplikasi di depan asisten, mahasiswa mengakses video Youtube tersebut dan memutarinya di depan asisten. Beberapa contoh video tubes tahun-tahun sebelumnya dapat dilihat di YouTube dengan menggunakan kata kunci “Tubes Stima”, “Tugas besar stima”, “strategi algoritma”, dll.

11. Demo akan dilakukan, tunggu informasi lanjut setelah waktu pengerjaan tugas berakhir.

12. Setiap anggota kelompok harus memahami seluruh program, termasuk bagian yang bukan bagian mereka.

13. Program disimpan dalam folder **Tubes3_NIM** (jika menggunakan repository dapat mengubah nama repository-nya) dengan NIM merupakan NIM anggota terkecil. Berikut merupakan struktur dari isi folder tersebut.

a. Folder **src** berisi **source code**.

b. Folder **doc** berisi **laporan tugas besar** dengan format **nama_kelompok.pdf**

c. README selengkap mungkin (dapat menjelaskan cara menjalankan backend dan frontend di lokal). Referensi README dapat diakses pada <https://github.com/ritaly/README-cheatsheet> atau referensi lain yang serupa.

BAB II

LANDASAN TEORI

2.1. Algoritma Knuth-Morris-Pratt (KMP)

Algoritma Knuth-Morris-Pratt merupakan salah satu algoritma yang digunakan untuk untuk pencocokan pola (*pattern-matching*). Tujuan algoritma ini menguji apakah suatu pola *string* dapat ditemukan dalam suatu teks *string* utamanya dimana pola tersebut merupakan *sub-string* atau bukan dari *string* yang diuji. Algoritma ini menguji pola tersebut dari kiri ke kanan, namun tidak semata-mata secara *brute-force* menguji hurufnya satu-satu.

Strategi utama dalam algoritma KMP adalah pembentukkan tabel kecocokan (*border function table*). Tabel kecocokan menyimpan informasi mengenai pola yang digunakan untuk mencocokkan karakter dalam teks. Tabel ini digunakan untuk mempercepat proses pencocokan pola dengan menghindari pengulangan yang tidak perlu, sehingga algoritma dapat “melompat” beberapa karakter berdasarkan perhitungan pada tabel kecocokan untuk mempersingkat pengecekan.

Contoh sederhana, apabila terdapat sebuah pola dengan string “abaaba”, maka berikut adalah tabel yang dibentuk.

j	0	1	2	3	4	5
P[j]	a	b	a	a	b	a
k	-	0	1	2	3	4
b(k)	-	0	0	1	1	2

Pada tabel pada gambar tersebut, tabel j merepresentasikan pola dan tabel k merepresentasikan tabel kecocokan untuk algoritma KMP. Nilai b(k) didapatkan dari prefix terbesar dari P[0..k] terbesar yang sama dengan suffix dari P[1..k]. Contoh sederhana pada b(3), prefix dari “abaa” dan suffix dari “baa” yang terbesar dan sama adalah hanya “a”, yaitu karakter pertama pada “abaa” dan karakter terakhir dari “baa” saja yang memungkinkan sama. Contoh pada b(4), prefix dari “abaab” dan suffix dari “baab” yang terbesar dan sama adalah “ab” pada awalan “abaab” dan “ab” pada akhiran “baab”.

Pergeseran yang dilakukan pada algoritma Knuth-Morris-Pratt adalah berdasarkan fungsi pembatas yang sudah didefinisikan dari pattern. Persyaratannya adalah sebagai berikut:

- Apabila terjadi ketidakcocokan karakter pada pola $P[j]$ yaitu $P[j] \neq T[i]$ pada teks, dan $k=j-1$, maka nilai j menjadi $b(k)$
- Apabila terjadi kecocokan karakter pada pola $P[j]$ yaitu $P[j]=T[i]$ pada teks, maka nilai i menjadi $i+1$, dan nilai j menjadi $j+1$

Contoh perhitungan pergeseran pola “abaaba” pada teks “ababaacabaababd” sebagai berikut.

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
T[i]	a	b	a	b	a	a	c	a	b	a	a	b	a	b	d

Pengecekan pada teks ini dimulai pada $i = 0$ dan $j = 0$ dimana j merupakan indeks untuk pola $P[j]$. pada pola dan teks, dua karakter pertama dianggap sama sehingga setiap karakter yang sama akan dilakukan penambahan i dan j hingga $i = 3$ dan $j = 3$ karena pada indeks 0 hingga 2 terjadi kecocokan. Namun, pada $T[3]$, terdapat huruf “b” yang berbeda dengan $P[3]$, yaitu “a” sehingga pola akan digeser pada $j = b[j-1]$ (yaitu $b[k]$), sehingga $j = 0$ dan $i = 3$ karena $b[2] = 0$ dan dilakukan pengecekan ulang pola. Karena mengulang dari awal, indeks ke 4 merupakan karakter yang sama dilakukan penambahan pada i dan j sehingga $j = 1$ dan $i = 4$. Pada indeks selanjutnya ($i = 5$) terjadi ketidakcocokan lagi sehingga $j = b[j-1]$, yaitu nilai j kembali ke nilai nol kembali ($j = b[0]$). Kemudian $T[6]$, yaitu bernilai “c” berbeda dengan awalan pola sehingga i digeser dan nilai $j = 0$, $i = 6$. Pada indeks ke-7, ditemukan kesamaan hingga indeks ke-12 sehingga dilakukan terus penambahan pada i dan j , yaitu dari $j = 0$ dan $i = 7$, hingga $j = 5$ dan $i = 12$. Karena panjang pola 6, maka j mencapai kesamaan, yaitu sebesar indeks terakhir dari panjang pola dan ditemukan kecocokan pola dengan teks.

Algoritma KMP ini sendiri memiliki kompleksitas waktu $O(m+n)$ dengan keterangan m sebagai perhitungan fungsi dari pola dan n sebagai pencarian string atau teks. Hal ini menunjukkan KMP lebih efektif dibandingkan algoritma *brute force* yang memiliki kompleksitas $O(mn)$ pada kasus terburuk dikarenakan pengecekan untuk setiap huruf pada pattern dalam perbandingan hanya satu karakter pada teks.

2.2. Algoritma Boyer-Moore (BM)

Algoritma BM (Boyer-Moore) adalah salah satu algoritma pencarian pola (pattern matching) yang digunakan untuk mencari kemunculan suatu pola tertentu dalam sebuah teks. Algoritma ini sering digunakan dalam pencarian string karena efisien dalam pencarian pola yang panjangnya cukup besar pada teks yang panjang.

Algoritma Boyer-Moore didasarkan pada dua teknik pencocokan string, yaitu pencocokan mundur (backward matching) dan pencocokan maju (forward matching). Pencocokan mundur dilakukan dengan cara memulai pencocokan dari karakter terakhir pada pola, sedangkan pencocokan maju dilakukan dengan cara memindahkan pola sejauh jarak yang sesuai dengan karakter pada teks yang tidak cocok dengan pola.

Perumusan dasar Boyer-Moore adalah sebagai berikut:

- Membuat tabel bad-character shift (tabel geser karakter yang tidak cocok) Tabel ini berisi informasi tentang jarak geser (shift) yang harus dilakukan apabila karakter pada pola tidak cocok dengan karakter pada teks. Tabel ini disusun dengan cara menghitung jarak antara karakter pada pola yang tidak cocok dengan karakter pada teks, lalu menentukan jarak geser yang sesuai.
- Membuat tabel good-suffix shift (tabel geser susunan yang cocok) Tabel ini berisi informasi tentang jarak geser (shift) yang harus dilakukan apabila bagian pada pola cocok dengan bagian pada teks. Tabel ini disusun dengan cara menghitung jarak antara bagian pada pola yang cocok dengan bagian pada teks, lalu menentukan jarak geser yang sesuai.
- Melakukan pencocokan pola pada teks Pencocokan dimulai dari karakter terakhir pada pola. Apabila karakter pada pola tidak cocok dengan karakter pada teks, maka dilakukan geser menggunakan informasi pada tabel bad-character shift. Apabila bagian pada pola cocok dengan bagian pada teks, maka dilakukan geser menggunakan informasi pada tabel good-suffix shift. Proses ini dilakukan sampai pola ditemukan atau sampai seluruh teks telah dicocokkan.

Boyer-Moore sendiri memiliki kompleksitas yang sama dengan *brute force* dimana $O(mn)$ adalah kasus terburuk yang dapat terjadi apabila terjadi pengulangan pengecekan yang cocok dari karakter akhir ke awal bersamaan dengan teks, tetapi berbeda dikata pertama seperti teks “aaaaaa” dan pattern “zaa”, kompleksitasnya akan mirip dengan *brute force*. Namun, Boyer-Moore sangat efektif dalam pengecekan kata - kata yang digunakan sehari - hari baik bahasa Inggris, Indonesia, maupun kata - kata yang menggunakan huruf variatif. Hal ini dikarenakan penggunaan pergeseran tabel *bad-character* lebih banyak terpakai saat pencarian kata, melainkan kumpulan huruf yang seragam. Namun, rata - rata

kompleksitas Boyer-Moore sendiri $O(m+n)$ dan pada seringkali memiliki kompleksitas $O(m)$ dalam pengecekan kata - kata sehari-hari

2.3. Regular Expression (Regex)

Regular expression (*Regex*) adalah sekuens dari karakter-karakter yang mendefinisikan suatu pola pencarian. *Regex* merupakan salah satu metode pencocokan string (*string matching*) yang sudah banyak digunakan dan menjadi standar pada tools-tools dan bahasa pemrograman yang banyak dipakai saat ini, dikarenakan *regex* sendiri adalah metode yang efisien untuk *string matching*.

Beberapa jenis-jenis simbol penting dalam penggunaan *regex* yang perlu diketahui adalah sebagai berikut :

1. *Repeaters* : ($\{$, $+$, $*$) : Simbol-simbol berikut digunakan untuk memberitahu komputer bahwa karakter bisa digunakan lebih dari 1 kali
2. *Wildcard* : ($.$) : Simbol titik ini digunakan untuk memberitahu komputer bahwa karakter apapun dapat digunakan untuk menggantikan posisi simbol titik tersebut
3. *Optional character* : ($?$) : Simbol ini digunakan untuk memberitahu komputer bahwa sebuah karakter boleh terdapat atau tidak terdapat pada sebuah string
4. *Caret* : ($^$) : Simbol ini digunakan untuk memberitahu komputer bahwa pattern tersebut harus muncul pada bagian awal sebuah string
5. *Dollar* : ($\$$) : Simbol ini digunakan untuk memberitahu komputer bahwa pattern tersebut harus muncul pada bagian akhir sebuah string

Masih banyak simbol-simbol yang terdapat pada *regex* dan kegunaannya yang bervariasi. Namun pada landasan teori ini hanya akan dijelaskan beberapa dikarenakan untuk kegunaan simbol-simbol diatas dan yang lainnya dapat dilihat lebih jelas pada referensi yang diberikan di bawah.

2.4. Penjelasan singkat mengenai aplikasi web yang dibangun

Aplikasi web yang dibangun merupakan aplikasi menyerupai *ChatBot* (yang lebih dikenal sekarang adalah ChatGPT). Aplikasi web ini dibangun dengan menggunakan teknologi *Quasar Framework*, *Vue 3 Typescript* untuk membangun dari sisi depannya / *front-end*. Untuk membangun sisi belakang dari aplikasi web ini, kami menggunakan bahasa pemrograman *Golang*. Untuk *data store* yang kami gunakan adalah *MariaDB*. Aplikasi *ChatBot* ini memiliki 2 metode untuk melakukan pencocokan string yang dikirim oleh user dalam bentuk pertanyaan kepada *Bot*. Metode tersebut berupa

Knuth-Morris-Pratt (KMP) dan **Boyer-Moore (BM)**. Setiap string yang dikirim oleh *user* dalam bentuk pertanyaan akan divalidasi dari sisi *backend* untuk menentukan jawaban yang paling tepat berdasarkan pertanyaan dan jawaban yang telah ada di *data store Bot*. *User* juga dapat mengirimkan lebih dari 1 sekaligus dengan melakukan *Shift + Enter* pada keyboard untuk *newline* pada *input field* yang disediakan. *Bot* akan menjawab semua pertanyaan tersebut dan mengembalikannya kepada *user*. *User* juga dapat memilih ingin menggunakan metode pencocokan, antara KMP atau BM.

BAB III

ANALISIS PEMECAHAN MASALAH

3.1. Penjelasan Algoritma Fitur Kalkulator

Untuk algoritma fitur kalkulator sendiri, digunakan kelas *Stack* untuk mempermudah pengerjaan fitur ini. Berikut langkah-langkah penyelesaian fitur kalkulator :

1. Pertama, akan dibuat dua *stack* yang mempunyai tugas berupa, *stack* pertama sebagai penyimpan angka-angka yang terdapat pada ekspresi matematika dan *stack* kedua sebagai penyimpan operator-operator yang terdapat pada ekspresi matematika
2. Kedua, akan diambil terlebih dahulu angka-angka yang terdapat pada ekspresi matematika dan diletakkan pada sebuah *array*. Hal ini dilakukan agar mempermudah operasi evaluasi nantinya karena angka bisa memiliki desimal dibelakangnya
3. Ketiga, akan dilakukan iterasi pada *string* sampai seluruh *character* pada *string* telah di cek. Selama iterasi dilakukan, akan dilakukan pengecekan kondisi yang menentukan aksi yang akan dilakukan. Berikut kondisi-kondisinya :
 - a. Apabila sebuah karakter yang di cek sekarang merupakan sebuah angka, masukkan angka tersebut kedalam *value stack* dan iterasi akan dilewati sampai seluruh angka telah dilewati pada *string*
 - b. Apabila karakter yang di cek sekarang adalah operator *)*, maka lakukan *pop* dua kali pada *value stack* dan *pop* satu kali pada *operator stack* dan lakukan operasi matematika sesuai dengan dua *value* dan *operator* yang di *pop* tadi, serta masukkan kembali *value* yang didapat ke dalam *value stack*. Lakukan hal berikut sampai operator *)* berhasil ditemukan pada *top* dari *operator stack*
 - c. Apabila karakter yang di cek sekarang adalah operator *(*, maka *push* karakter tersebut ke *operator stack*.
 - d. Apabila karakter yang di cek sekarang adalah operator *+*, *-*, */*, ***, maka selama *operator stack* tidak kosong dan *top* dari *operator stack* memiliki tingkat kepentingan yang sama atau lebih besar dari operator yang di cek sekarang (*P.S. Tingkat kepentingan yang dimaksud disini adalah contohnya operator *** dan */* harus didahulukan dari operator *+* dan *-**), Lakukan hal yang sama seperti pada kondisi b tadi. Setelah iterasi selesai, *push* karakter ke dalam *operator stack*
4. Keempat, apabila setelah langkah ketiga selesai dan *operator stack* masih belum kosong, maka lakukan hal yang sama seperti pada langkah 3b sampai *operator stack* kosong.

5. Terakhir, apabila *operator stack* telah kosong, maka *value* terakhir pada *value stack* adalah hasil ekspresi matematika dari *string* tersebut

Note penting, untuk menggunakan angka negatif pada kalkulator yang disediakan ini, harus digunakan tanda kurung sebagai contoh $(-2) + 3$, apabila penulisan angka negatif tidak menggunakan tanda kurung program kalkulator akan menganggap sintaks sebagai sintaks yang tidak valid

3.2. Penjelasan Algoritma Fitur Tanggal

Untuk algoritma fitur tanggal, digunakan **Zeller's Rule** yang memiliki penjelasan sebagai berikut :

- Rumus yang digunakan **Zeller's Rule** adalah sebagai berikut :

$$F = k + [(13 * m - 1) / 5] + D + [D/4] + [C/4] - 2 * C$$

k : tanggal hari, m : tanggal bulan, D : dua digit terakhir dari tanggal tahun,

C : dua digit pertama dari tanggal tahun

- Berdasarkan **Zeller's Rule** angka yang merepresentasikan bulan adalah sebagai berikut :
 - Maret : 1
 - April : 2
 - Mei : 3
 - Juni : 4
 - Juli : 5
 - Agustus : 6
 - September : 7
 - Oktober : 8
 - November : 9
 - Desember : 10
 - Januari : 11
 - Februari : 12
- Karena bulan Maret merepresentasikan angka 1 pada **Zeller's Rule** maka tahun akan dimulai pada bulan Maret dan berakhir pada bulan Februari. Sehingga apabila hari yang ingin dicari berada pada bulan Januari atau Februari, kurangkan nilai tahun dengan 1. Sebagai contoh : apabila tanggal yang ingin dicari adalah 2 Februari 2023, maka nilai tahun akan menjadi 2022.
- Setelah mendapatkan *value* dari F dengan menggunakan rumus diatas, bagikan *value* F dengan 7, dan ambil sisa dari pembagian *value* F dengan 7. *Value* tersebut akan mewakili hari yang dicari dengan aturan sebagai berikut :
 - *Value* 0 berarti Hari Minggu
 - *Value* 1 berarti Hari Senin

- *Value 2* berarti Hari Selasa
- *Value 3* berarti Hari Rabu
- *Value 4* berarti Hari Kamis
- *Value 5* berarti Hari Jumat
- *Value 6* berarti Hari Sabtu

Note penting untuk fitur tanggal yang dimasukkan harus dalam format DD/MM/YYYY, jika tidak maka tidak akan dianggap sebagai tanggal yang valid

3.3. Fitur Fungsional dan Arsitektur Aplikasi *Web* yang Dibangun

Untuk fitur fungsional aplikasi *web* mengacu pada kemampuan *web-app* untuk dapat melakukan tugas-tugas tertentu yang diinginkan oleh pengguna. Fitur-fitur ini mencakup kemampuan untuk membuat, membaca, memperbarui, menghapus data, serta kemampuan untuk berinteraksi dengan pengguna melalui antarmuka yang intuitif. Fitur fungsional lainnya yaitu fitur untuk mencocokkan *string matching* yang perlu digunakan untuk memberikan *response* kepada pengguna. Selain fitur *string matching*, juga terdapat fitur “GPT” dimana *bot* akan me-*response* layaknya seperti *chatGPT*.

Framework yang kami gunakan sebagai *front-end* adalah *Quasar Framework*. *Quasar* adalah *framework front-end* yang dapat membantu kami dalam membangun aplikasi web yang responsif dan menarik. *Framework* ini menyediakan berbagai komponen siap pakai yang dapat kami gunakan untuk membangun antarmuka pengguna dengan cepat. *Quasar* juga mendukung *Vue 3*, yang merupakan versi terbaru dari *framework front-end* populer *Vue.js*. Dengan menggunakan *Vue 3*, kami dapat memanfaatkan fitur-fitur terbaru dari *Vue.js* dalam membangun aplikasi kami.

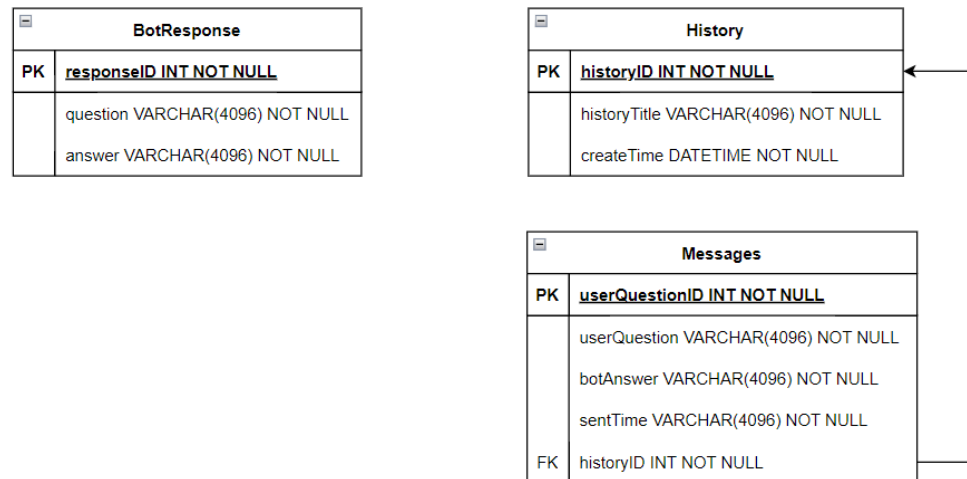
Selain itu, *Quasar* juga mendukung penggunaan *TypeScript*, yang merupakan bahasa pemrograman yang menambahkan fitur-fitur seperti pengecekan tipe statis ke *JavaScript*. Dengan menggunakan *TypeScript*, kami dapat menulis kode yang lebih mudah di-*maintenance* dan di-*develop* serta mengurangi kemungkinan terjadinya kesalahan saat *runtime*.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1. Spesifikasi Teknis Program

Berikut adalah *relational model* dari tabel-tabel yang diimplementasikan pada database



Gambar Relational Model database

Berikut struktur data yang digunakan pada program yang telah kami buat

No	Nama	Struktur data	Keterangan
1	BotResponse	<ul style="list-style-type: none"> - ResponseID (integer) - Question (string) - Answer (string) 	Struktur data <i>bot response</i> berisi pasangan pertanyaan dan jawaban yang akan digunakan untuk mencocokkan pertanyaan user melalui fitur pertanyaan teks.
2	Message	<ul style="list-style-type: none"> - Id (integer) - Text (string) - Response (string) - SentTime (string) 	Struktur message merepresentasikan keterangan data sebuah pertanyaan

		<ul style="list-style-type: none"> - HistoryId (integer) - HistoryTimeStamp (string) 	user beserta tanggapan dari <i>chatbot</i> .
3	History	<ul style="list-style-type: none"> - HistoryID (integer) - Topic (string) - Conversation (array of message) 	Struktur history merepresentasikan satu <i>chat</i> yang dilakukan user. Dengan topik sebagai judul dan perincian isi chat pada <i>conversation</i>
4	HistoryRequest	<ul style="list-style-type: none"> - HistoryID (integer) - HistoryTopic (string) 	Struktur yang digunakan untuk <i>frontend</i> meminta (<i>request</i>) suatu <i>history chat</i> untuk dimunculkan pada tampilan antarmuka
5	Request	<ul style="list-style-type: none"> - Message (message) - Method (string) 	Struktur request digunakan untuk <i>frontend</i> untuk mendapatkan respon dengan algoritma <i>backend</i> .

Berikut merupakan fungsi atau prosedur yang dibuat dalam implementasi *front-end* dari program kami.

Function	Penjelasan
fetchAllTopic(): Promise<void>	Untuk mengambil semua data history (topik dan id) dari database ke <i>front-end</i>
fetchHistory(): Promise<void>	Untuk mengambil data history berupa isi <i>conversation</i> berdasarkan history id yang dikirim ke <i>backend</i>
generateTimestamp(): string	Mengenerate waktu pada saat itu
sendMessage(): Promise<void>	Mengirim <i>message</i> user ke <i>back-end</i> untuk diproses
generateMessageId(): number	Mengembalikan history id yang bisa digunakan

Berikut beberapa fungsi atau prosedur yang dibuat dalam implementasi *back-end* dari program kami.

Fungsi	Penjelasan
FilterMessage(req *Request, stat *string, db *sql.DB, regex []*regexp.Regexp): void	Melakukan filter pertanyaan atau teks menggunakan splitter dan regex untuk menentukan kategori fitur serta jumlah fitur yang digunakan pada pertanyaan tersebut
GetResponse(req *Request, question string, index int, stat *string, db *sql.DB): string	Mengembalikan nilai respon yang seharusnya untuk suatu pertanyaan yang sudah dikategorikan fiturnya berdasarkan index.
StringMatching(req *Request, text string, db *sql.DB, questions []strings)	Mengembalikan kondisi apakah ditemukan <i>exact match</i> berdasarkan metode algoritma KMP / Boyer Moore.
CalculateExpression(expression string)	Mengembalikan hasil ekspresi matematika yang ada pada parameter dan juga mengembalikan error apabila terdapat error
FindDayName(date string)	Mengembalikan nama hari pada tanggal yang diberikan
CreateRegex(): []*regexp.Regexp	Membuat tabel regex dengan query fitur - fitur yang diperlukan.
AddMessage(req *structs.Request, db *sql.DB): void	Memasukkan pasangan teks user dan respon bot ke dalam database.
GetAllHistoryMessage(req *structs.Request, db *sql.DB): *structs.Request	Melakukan <i>fetch</i> atau mengambil seluruh <i>history</i> untuk dikirimkan ke <i>front-end</i>
ParseUserMessage(req *structs.Request, db *sql.DB, regex []*regexp.Regexp): void	Fungsi <i>post</i> untuk menerima <i>text</i> pengguna dari <i>front-end</i> untuk diolah responnya pada <i>back-end</i> dan dikembalikan lagi untuk ditampilkan pada <i>front-end</i>

4.2. Tata Cara Penggunaan Program

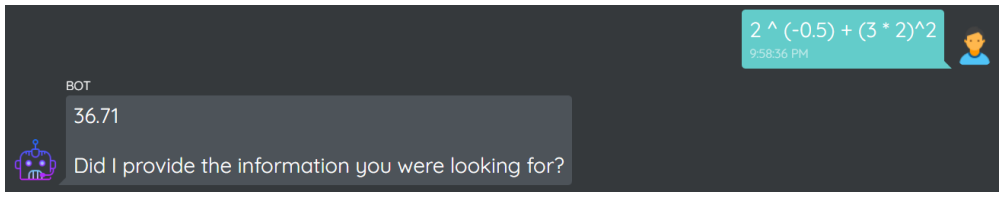
Karena ada beberapa fitur yang dapat digunakan pada program, pada bagian ini akan dijelaskan cara menggunakan program secara *overall* dan penggunaan fitur-fitur yang ada. Berikut langkah-langkahnya :

1. Untuk menambahkan chat baru, dapat dengan menekan tombol *new chat* yang ada pada bagian kiri atas web

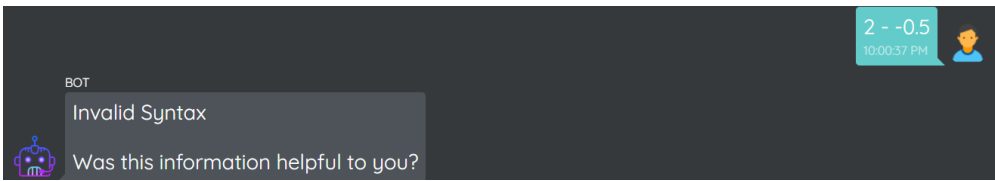
2. Apabila ingin melihat *chat* pada suatu *history* yang ada, dapat dilakukan dengan memilih *history* yang ingin dilihat *chatnya* pada bagian kiri
3. Kemudian untuk memilih algoritma *string matching* dapat dipilih pada bagian kiri bawah
4. Apabila ingin memakai fitur kalkulator, gunakan sintaks berikut “berapakah {persamaan}” atau “berapa {persamaan}” atau “hasil dari {persamaan}”, pada sintaks-sintaks ini huruf besar dan kecil tidak diperhatikan
5. Apabila ingin memakai fitur tanggal, gunakan sintaks berikut “hari apa {tanggal}” atau “hari apakah {tanggal}”, pada sintaks-sintaks ini huruf besar dan kecil tidak diperhatikan
6. Apabila ingin memakai fitur menambahkan pertanyaan maka, gunakan sintaks berikut “Tambahkan pertanyaan x dengan jawaban y”, dengan x adalah pertanyaan yang ingin ditambahkan dan y adalah jawaban pertanyaan.
7. Apabila ingin memakai fitur menghapus pertanyaan maka, gunakan sintaks berikut “Hapus pertanyaan x”, dengan x adalah pertanyaan yang ingin dihapus
8. Apabila ingin memakai fitur pertanyaan teks, maka tanyakan saja kepada bot pertanyaan apapun, apabila terdapat pertanyaan yang sesuai pada database maka bot akan menampilkan jawaban dari pertanyaan tersebut

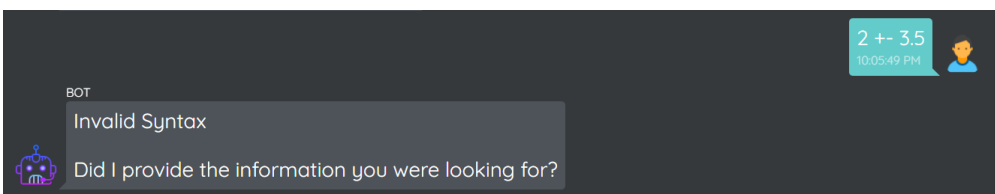
4.3. Hasil Pengujian dan Analisis Pengujian

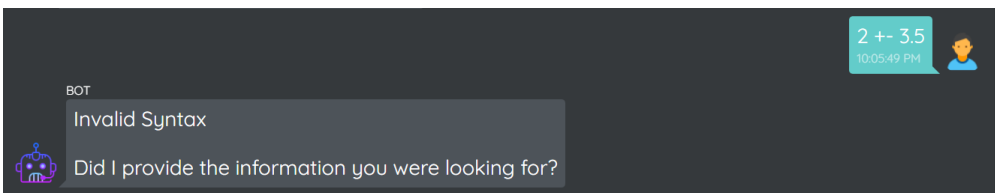
4.3.1. Pengujian Fitur Kalkulator

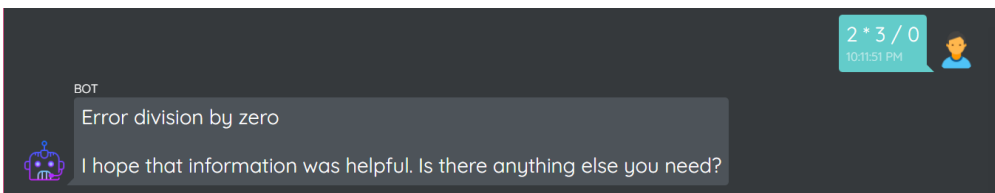
Query	$2^{(-0.5)} + (3 * 2)^2$
Jawaban	36.71
Screenshot	
Penjelasan	Karena persamaan yang diberikan telah memiliki sintaks yang valid, maka kalkulator berhasil dijalankan

Query	$2 - -0.5$
Jawaban	Invalid Syntax

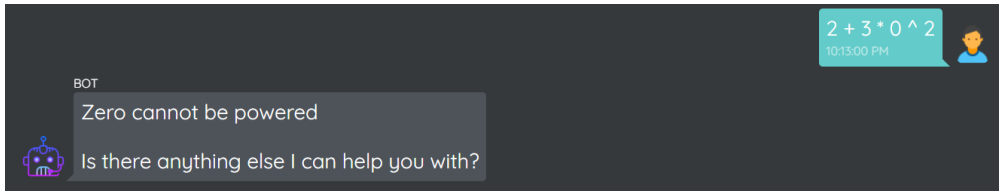
Screenshot	
Penjelasan	Meskipun sintaks tersebut terlihat benar, namun pada fitur kalkulator kami, kami mewajibkan untuk menggunakan tanda kurung untuk penggunaan nilai negatif

Query	2 +- 3.5
Jawaban	Invalid Syntax
Screenshot	
Penjelasan	Karena sintaks yang diberikan salah maka bot akan mengirim Invalid Syntax

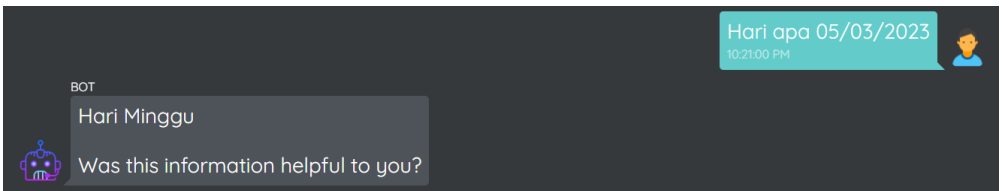
Query	2 +- 3.5
Jawaban	Invalid Syntax
Screenshot	
Penjelasan	Karena sintaks yang diberikan salah maka bot akan mengirim Invalid Syntax

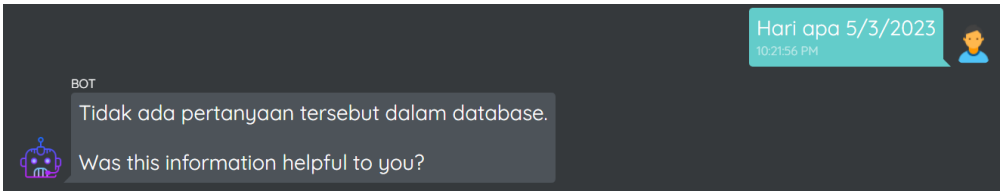
Query	2 * 3 / 0
Jawaban	Error division by zero
Screenshot	
Penjelasan	Karena angka berapapun tidak boleh dibagi 0 maka akan dikeluarkan <i>message</i> dari

	bot
--	-----

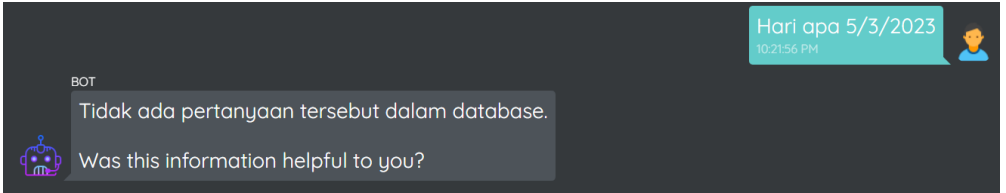
Query	$2 + 3 * 0 ^ 2$
Jawaban	Zero cannot be powered
Screenshot	
Penjelasan	Karena angka 0 tidak boleh dipangkat dengan angka apapun maka akan dikeluarkan <i>message</i> dari bot

4.3.2. Pengujian Fitur Tanggal

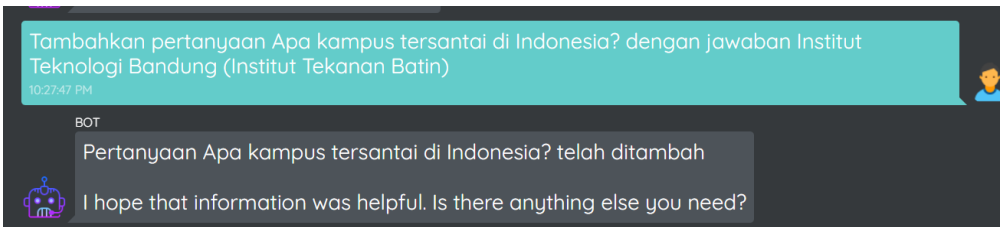
Query	Hari apa 05/03/2023
Jawaban	Hari Minggu
Screenshot	
Penjelasan	Karena sintaks yang diberikan sesuai maka fitur tanggal dapat bekerja dengan baik

Query	Hari apa 5/3/2023
Jawaban	Tidak ada pertanyaan tersebut dalam database
Screenshot	
Penjelasan	Karena sintaks penulisan tanggal masih salah sehingga chatbot akan menggunakan fitur pertanyaan teks dikarenakan untuk masuk ke fitur tanggal, format tanggal yang

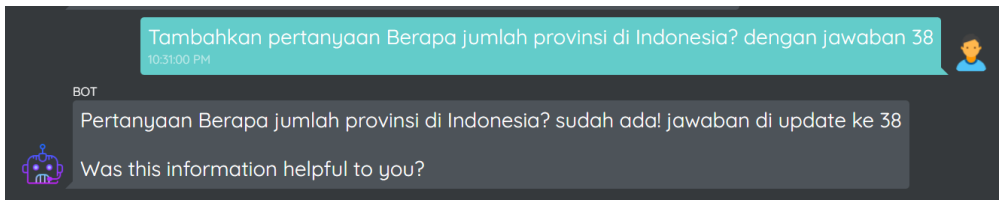
	harus ditulis adalah DD/MM/YYYY bukan D/M/YYYY
--	--

Query	Hari apa 31/02/2023
Jawaban	Invalid Date
Screenshot	
Penjelasan	Karena tanggal yang diberikan bukan merupakan tanggal yang valid maka bot akan mengirimkan <i>message</i>

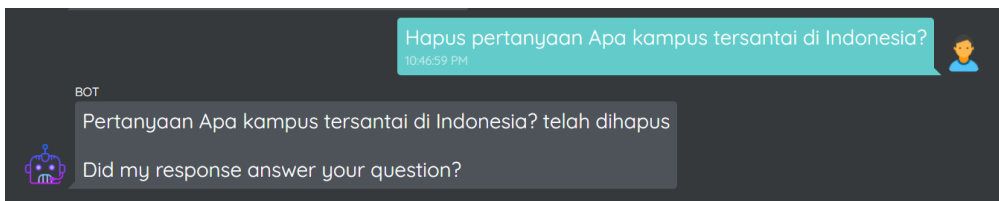
4.3.3. Pengujian fitur tambah pertanyaan

Query	Tambahkan pertanyaan Apa kampus tersantai di Indonesia? dengan jawaban Institut Teknologi Bandung (Institut Tekanan Batin)
Jawaban	Pertanyaan Apa kampus tersantai di Indonesia? telah ditambah
Screenshot	
Penjelasan	Karena sintaks yang diberikan sudah sesuai maka pertanyaan ditambahkan ke dalam database

Query	Tambahkan pertanyaan Berapa jumlah provinsi di Indonesia? dengan jawaban 38
Jawaban	Pertanyaan Berapa jumlah provinsi di Indonesia? sudah ada! jawaban di update ke 38

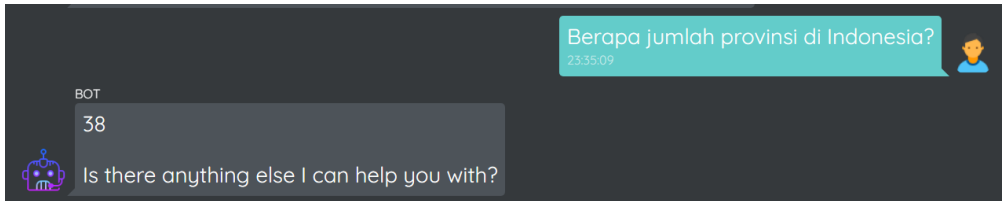
Screenshot	
Penjelasan	Karena pertanyaan sudah ada di dalam database maka jawaban akan di-update dengan jawaban yang baru

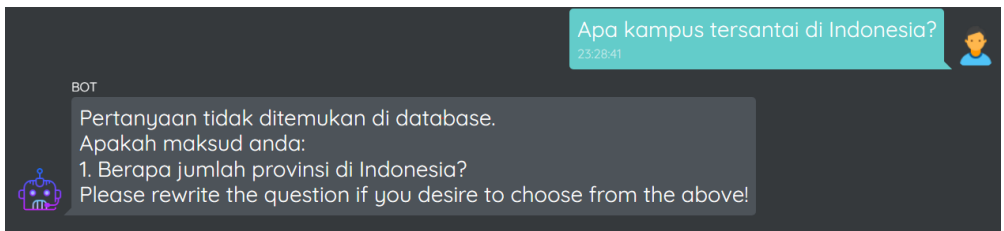
4.3.4. Pengujian fitur hapus pertanyaan

Query	Hapus pertanyaan Apa kampus tersantai di Indonesia?
Jawaban	Pertanyaan Apa kampus tersantai di Indonesia? telah dihapus
Screenshot	
Penjelasan	Karena sintaks yang diberikan sudah sesuai maka pertanyaan dihapus dari database

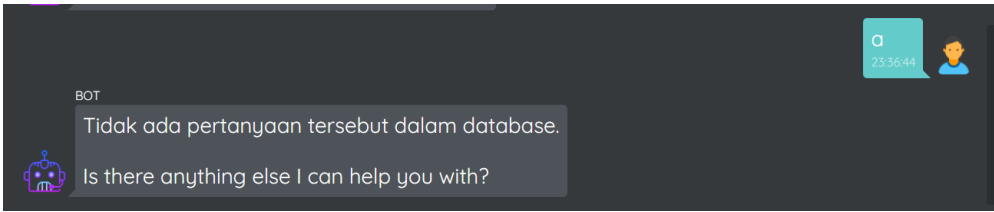
Query	Hapus pertanyaan Apa kampus tersantai di Indonesia?
Jawaban	Tidak ada pertanyaan Apa kampus tersantai di Indonesia? pada database!
Screenshot	
Penjelasan	Karena pertanyaan Apa kampus tersantai di Indonesia? telah dihapus sebelumnya maka apabila dicoba untuk dihapus lagi bot tidak akan menemukan pertanyaan tersebut pada database

4.3.5. Pengujian fitur pertanyaan teks

Query	Berapa jumlah provinsi di Indonesia?
Jawaban	38
Screenshot	
Penjelasan	Pada database, telah disimpan pertanyaan “Berapa jumlah provinsi di Indonesia?” dengan jawaban “38”. Karena terjadi exact matching, maka bot merespon dengan jawaban 38.

Query	Apa kampus tersantai di Indonesia?
Jawaban	Pertanyaan tidak ditemukan di database. Apakah maksud anda: 1. Berapa jumlah provinsi di Indonesia?
Screenshot	
Penjelasan	Karena pertanyaan Apa kampus tersantai di Indonesia? telah dihapus sebelumnya maka apabila dicoba untuk ditanyakan pertanyaan tidak ditemukan di database dan akan dimunculkan pertanyaan - pertanyaan lain yang memiliki kecocokan lebih dari 50%

Query	a
Jawaban	Tidak ada pertanyaan tersebut dalam database.

Screenshot	 <p>The screenshot shows a chat window with a dark background. In the top right corner, there is a status bar with a green square containing the letter 'a', the time '23:56:44', and a small profile icon of a person with orange hair. A message bubble from the bot, labeled 'BOT' in small text, contains the text 'Tidak ada pertanyaan tersebut dalam database.' (There is no such question in the database.) Below this, a smaller message bubble with a robot icon asks 'Is there anything else I can help you with?'.</p>
Penjelasan	<p>Jika query menunjukkan a saja, maka kemungkinan besar akan terjadi exact matching terhadap lebih dari 2 pertanyaan atau tidak sama sekali sehingga dengan algoritma <i>leviathan distance</i>, tidak ditemukan pertanyaan pada database yang setidaknya memiliki tingkat kemiripan 50% dan ditampilkan tidak ada pertanyaan tersebut dalam database.</p>

BAB V

PENUTUP

5.1. Kesimpulan

Algoritma **Knuth-Morris-Pratt (KMP)** dan **Boyer-Moore (BM)** dapat digunakan untuk membuat sebuah *webapp* yang menyerupai *chatGPT*. Kedua algoritma tersebut dapat menemukan semua kemunculan pola dalam teks pada waktu linier. Algoritma KMP melakukannya dengan memproses pola untuk membuat tabel kecocokan parsial yang memungkinkannya menghindari perbandingan yang tidak perlu saat terjadi *unmatch*. Algoritma BM melakukannya dengan memproses pola untuk membuat dua heuristik: aturan karakter buruk dan aturan akhiran baik. Heuristik ini memungkinkannya melompati bagian besar teks saat terjadi *unmatch*. Dengan menggunakan algoritma KMP atau BM, *ChatBot* dapat mencocokkan input pengguna dengan pola yang telah ditentukan secara efisien dan mengenali maksud dari input tersebut. Ini memungkinkan *ChatBot* untuk memberikan response yang sesuai dan meningkatkan kualitas interaksi dengan pengguna.

5.2. Saran

Penggunaan algoritma Knuth-Morris-Pratt serta Boyer Moore memiliki keterbatasan yang cukup besar dalam program *chatbot* dikarenakan *exact-matching* suatu teks kemungkinan besar akan menghasilkan lebih dari satu *exact matching* dengan data seluruh pertanyaan ataupun dengan sedikit kesalahan penulisan juga dapat menyebabkan tidak ada yang *exact match* sehingga seringkali terbatas untuk menentukan pertanyaan yang sesuai dan penggunaan perhitungan tingkat kemiripan salah satunya seperti algoritma *Levenshtein distance* jauh lebih efektif dalam implementasi *chatbot*. Algoritma Knuth-Morris-Pratt serta Boyer Moore lebih cocok dalam implementasi seperti algoritma *searching* pencarian sebuah kata pada teks atau arsip.

5.3. Penutup

Kami dari kelompok 666 mengucapkan terima kasih kepada asisten yang sudah mengeluarkan spesifikasi untuk Tugas Besar Strategi Algoritma III dan kami juga berterima kasih kepada dosen pengampu Strategi Algoritma atas Tugas Besar ini.

DAFTAR PUSTAKA

- Munir, Rinaldi. 2023. Pencocokan string (String matching/pattern matching). Homepage Rinaldi Munir Sekolah Teknik Elektro dan Informatika (STEI) ITB.
<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf>
- Munir, Rinaldi. 2023. Pencocokan string dengan Regular Expression (Regex). Homepage Rinaldi Munir Sekolah Teknik Elektro dan Informatika (STEI) ITB.
<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf>
- Singh, Chaitanya. 2012. Day Calculation from date. BeginnersBook.
<https://beginnersbook.com/2013/04/calculating-day-given-date/>
- GeeksforGeeks. 2022. Expression evaluation.
<https://www.geeksforgeeks.org/expression-evaluation/>
- Regexpal.
<https://www.regexpal.com/>

LAMPIRAN

Tautan repository tugas besar : https://github.com/Jimly-Firdaus/Tubes3_13521102

Tautan *web-app* : <https://pemuladigital.github.io/#/>