LAPORAN TUGAS BESAR 3
IF2211 STRATEGI ALGORITMA
SEMESTER II 2022-2023

Penerapan String Matching dan Regular Expression dalam Pembuatan ChatGPT Sederhana

Disusun oleh:

Kelompok 5 (SabdaUstadzFakih)

Christian Albert Hasiholan	13521078
Fakih Anugerah Pratama	13521091
Bintang Dwi Marthen	13521144

PROGRAM STUDI
TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO
DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG 2023

1. DESKRIPSI TUGAS

Dalam tugas besar ketiga mata kuliah IF2211 – Strategi Algoritma, mahasiswa diminta untuk membuat sebuah aplikasi ChatGPT sederhana. Pembuatan ChatGPT sederhana tersebut menggunakan pendekatan QA yang sederhana. Pencarian pertanyaan yang paling mirip antara pertanyaan pengguna dan pertanyaan pada basis data akan menggunakan algoritma pencocokan string Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM). Selain algoritma pencocokan string, digunakan pula ekspresi regular (*regular expression [regex]*) untuk menentukan format dari pertanyaan apakah pertanyaan teks, kalkulator, tanggan, maupun sebuah kueri ke basis data. Bila tidak terdapat pertanyaan yang persis pada basis data, maka dapat digunakan algoritma *Hamming Distance*, *Levenshtein Distance*, ataupun *Longest Common Subsequence* untuk mencari pertanyaan yang memiliki kesamaan setidaknya 90%. Apabila tidak ada pertanyaan yang memiliki kemiripan di atas 90%, maka akan diberikan maksimal tiga pilihan pertanyaan paling mirip untuk dipilih oleh pengguna.

2. LANDASAN TEORI

2.1. Algoritma KMP

Algoritma KMP (*Knuth-Morris-Pratt)* merupakan optimasi dari algoritma *bruteforce* untuk pencocokan string. KMP meng-*shift* pola ketika pencocokan dengan lebih cerdas. Ketika terjadi ketidakcocokan antara teks dan pola P di P[j] (semisal T[i] != P[j] maka akan pola akan dishit sebanyak prefiks terbesar dari P[0..j-1] yang merupakan sufiks dari P[1..j-1]).

Sebagai contoh, dimiliki suatu teks “abcabcabd” dan pola “abcabd”. Ketika dilakukan pencocokan, terdapat ketidakcocokan pada huruf “d” (indeks j = 5) maka didapatkan prefiks terbesar adalah “ab” (T[0..1]) dengan sufiks “ab” (T[3..4]). Karena diketahui bahwa sufiks dan prefiks tersebut sama, maka dapat dikatakan bahwa prefiks tersebut telah cocok, maka T digeser hingga prefiks berada pada posisi sufiks (sebanyak tiga indeks). Berikut merupakan ilustrasinya.

Kondisi teks dan pola sebelum pergeseran:

a	b	c	a	b	c	a	b	d
a	b	c	a	b	d			

Kondisi teks dan pola setelah pergeseran:

a	b	c	a	b	c	a	b	d
			a	b	c	a	b	d

Setelah dilakukan pergeseran, didapati bahwa pola dan teks persis satu banding satu sehingga didapati bahwa teks T cocok dengan pola P ketika indeks awalnya adalah tiga (i = 3).

Terdapat optimasi untuk KMP yang disebut dengan *border function*. Optimasi ini dilakukan dengan membuat tabel berisikan sejauh berapa indeks pola akan digeser bila terjadi ketidakcocokan pada indeks j tertentu dengan aturan prefiks(T[0..j-1]) = sufiks(T[0..j-1]).

Algoritma KMP memiliki keuntungan bahwa algoritma ini tidak perlu mundur dalam pengecekan teks T (mempercepat pengecekan teks yang besar). Akan tetapi, algoritma KMP ini tidak begitu baik bila komponen teks (alfabet) yang dicek semakin besar (semakin sering terjadi ketidakcocokan yang biasanya terjadi di awal pola [KMP lebih optimal bila ketidakcocokan berada di akhir pola]).

2.2. Algoritma BM

Algoritma BM *(Boyer-Moore*) merupakan optimasi dari algoritma *bruteforce* dengan prinsip mencari sejauh apa pola P dapat digeser bila terjadi ketidakcocokan ketika terjadi pencocokan dengan teks T. Berbeda dengan pencocokan *bruteforce* dan KMP, proses pencocokan dilakukan dari ujung kanan pola.

Kembali menggunakan contoh pola P “abcbd” dan teks T “abcceabcbd”, akan terdapat ketidakcocokan pada indeks i = 3. Terlihat bahwa indeks ketiga T adalah “c” sementara indeks ketiga P adalah “b”, terakhir kali “c” muncul pada P adalah pada indeks kedua. Dilakukan pergeseran sehingga “c” terakhir pada P akan berada pada posisi ketidakcocokan (indeks i = 3) sehingga dilakukanlah satu pergeseran. Setelah itu, didapati lagi terdapat ketidakcocokan pada indeks i = 5. Terlihat bahwa indeks kelima T adalah “a” dan “a” pada P terakhir muncul pada awal atau indeks kenol sehingga dilakukanlah pergeseran sebanyak empat. Berikut merupakan ilustrasinya.

Kondisi teks dan pola sebelum pergeseran:

a	b	c	c	e	a	b	c	b	d
a	b	c	b	d					

Kondisi teks dan pola setelah pergeseran pertama:

a	b	c	c	e	a	b	c	b	d
	a	b	c	b	d				

Kondisi teks dan pola setelah pergeseran kedua:

a	b	c	c	e	a	b	c	b	d
					a	b	c	b	d

Setelah dilakukan pergeseran maka didapati bahwa teks T dan pola P persis satu banding satu ketika indeks awalnya adalah lima (i = 5).

Terdapat optimasi bagi BM, yaitu dilakukan pemrosesan posisi terakhir munculnya tiap karakter yang ada pada pola P. Untuk setiap karakter yang tidak pernah muncul pada pola P dapat diberi tanda khusus, sebagai contoh adalah nilai -1. Optimasi ini disebut dengan *last occurrence function*. Algoritma BM merupakan alternatif bagi algoritma KMP ketika alfabet teks T besar. Akan tetapi, algoritma BM akan menjadi lambat ketika alfabat teks T kecil.

2.3. Ekspresi Regular

Ekspresi regular sebuah pola yang dinyatakan dalam suatu notasi tertentu. Dalam ekspresi regular, kurung siku ([ ]) melambangkan disjungsi (atau), garis sambung (-) di dalam kurung siku melambangkan

range, caret (^) melambangkan negasi, tanda tanya (?) melambangkan opsional, dan titik (.) melambangkan karakter apapun. Ekspresi regular merupakan salah satu metode pencocokan yang mengecek sebuah teks dengan ekspresi regular yang ada.

Ekspresi regular dapat digunakan dimanapun, sebagai contoh adalah validasi alamat surel. Salah satu contoh ekspresi regular yang dapat digunakan adalah ^[a-zA-Z0-9._%+-]+@(a-zA-Z0-9.-)+\.[a-zA-Z]{2,}$. Bagian [a-zA-Z0-9._%+-] merupakan bagian nama pengguna dari alamat surel (semisal alamat surelnya adalah asdf@abc.com maka asdf adalah nama penggunanya). Bagian @(a-zA-Z0-9.-)+\.[a-zA-Z]{2,} merupakan bagian *domain* alamat surel dan memastikan bahwa *top-level domain* harus memiliki setidaknya dua karakter (sehingga alamat surel seperti asdf@abc.c dilarang).

2.4. Penjelasan Singkat Aplikasi Web yang Dibangun

Aplikasi web yang dibangun pada tugas besar ini memiliki struktur *microservices* sehingga aplikasi dibagi menjadi beberapa bagian, tepatnya *frontend* dan *backend*. Bagian *frontend* merupakan bagian yang bertanggungjawab atas tampilan antar muka dan aplikasi yang dapat digunakan oleh pengguna. Bagian *frontend* merupakan bagian yang digunakan oleh pengguna. Bagian *backend* merupakan bagian yang bertanggungjawab mengurusi fungsionalitas aplikasi seperti koneksi ke basis data dan algoritma. Bagian *backend* tidak dapat digunakan dan dilihat oleh pengguna, bagian ini berinteraksi dengan bagian *frontend* melalui *request* dan *response*.

Pemilihan struktur *microservices* dilatarbelakangi oleh adanya pembagian yang jelas antara bagian yang digunakan oleh pengguna (*frontend*) dan bagian *server* (*backend*) sehingga pengguna tidak terlalu diberatkan untuk menjalankan bagian *backend*. Selain itu, karena pembagiannya yang jelas maka dalam proses *development*-nya dapat menjadi lebih modular karena pembagian tugasnya yang jelas.

3. ANALISIS PEMECAHAN MASALAH

4.1 Langkah Penyelesaian Masalah Setiap Fitur

Dalam implementasi yang kami buat, dari query yang diterima pertama-tama akan dibuat sebuah list yang berisi pertanyaan-pertanyaan yang terdapat pada query itu. Untuk memperolehnya, dilakukan pemisahan string query dengan regex yang memisahkan berdasarkan question mark, exclamation mark, dan comma. Setelah diperoleh list pertanyaan, tiap pertanyaan itu kemudian diklasifikasikan dengan regex dan diproses berdasarkan tipenya. Setelah seluruh pertanyaan diproses, maka akan diperoleh output dari tiap pertanyaan, yang kemudian akan ditampilkan pada user.

3.1.1. Fitur Pertanyaan Teks (didapat dari database)

Untuk fitur ini, pertanyaan user akan dicocokan dengan pertanyaan-pertanyaan pada database. Metode pattern matching yang dipakai tergantung pada metode yang user pilih. Namun karena pengecekan dilakukan dengan metode KMP atau BM, bila terdapat sebuah pertanyaan pada database yang substringnya merupakan pertanyaan user, maka pertanyaan tersebutlah yang akan dipilih oleh program. Untuk mencegah kasus tersebut dan mencari exact match maka ukuran dari string pertanyaan pada database haruslah sama dengan string pertanyaan user.
Bila tidak diperoleh satupun kasus exact match, maka tiap string pada database dikalkulasi kemiripannya dengan Levensthein Distance dan diurutkan menurun berdasarkan persentase kemiripannya. Bila pertanyaan dengan persentase kemiripan tertinggi memiliki persentase lebih dari 90%, maka pertanyaan tersebut dipilih sebagai solusi. Namun bila tidak ada, maka dipilih 3 pertanyaan dengan persentase tertinggi.

3.1.2. Fitur Kalkulator

Fitur kalkulator dapat digunakan bila terdapat keyword “hitung”, “berapa”, dan equal mark. Sebelum dilakukan perhitungan, keyword yang terdapat pada pertanyaan akan dihapus terlebih dahulu, sehingga setelah itu akan menyisakan operasi matematika saja. Operasi tersebut kemudian dibuat menjadi list of string dan diubah bentuknya menjadi postfix. Setelah diperoleh bentuk postfixnya, maka perhitungan akan dilakukan dengan menggunakan stack. Pada perhitungan ini juga akan dihandle kasus dimana operasi matematika yang dimasukkan invalid.

3.1.3. Fitur Tanggal

Fitur tanggal dapat digunakan bila pada pertanyaan terdapat tanggal dengan bentuk D/M/Y. Setelah pertanyaan dibersihkan dari string lain dan didapat tanggalnya saja maka kemudian akan dilakukan validasi tanggal. Bila tanggal valid, kemudian akan ditentukan hari pada tanggal tersebut.

3.1.4. Tambah Pertanyaan dan Jawaban ke Database

Fitur tambah pertanyaan dapat digunakan dengan keyword “Tambah pertanyaan … dengan jawaban … “. Setelah itu pertanyaan dan jawaban yang ingin ditambahkan diekstrak dari perintah user. Pertanyaan yang ingin ditambahkan kemudian dicek exact matchnya di database dengan metode yang user pilih. Bila ditemukan exact matchnya maka artinya pertanyaan tersebut telah ada dan hanya jawabannya saja yang perlu diupdate. Namun bila tidak ditemukan exact matchnya, maka pasangan pertanyaan dan jawaban itu akan ditambahkan ke dalam database.

3.1.5. Hapus Pertanyaan dari Database

Fitur hapus pertanyaan dapat digunakan dengan keyword "Hapus pertanyaan … ". Pertanyaan yang ingin dihapus kemudian diekstrak dari perintah user. Setelah itu dilakukan pengecekan exact match berdasarkan metode yang user pilih dengan pertanyaan-pertanyaan pada database. Bila ditemukan exact matchnya maka pertanyaan dan jawabannya akan dihapus dari database. Namun bila tidak maka akan ditampilkan respon tidak ditemukannya pertanyaan pada database kepada user.

4.2 Fitur Fungsional dan Arsitektur Aplikasi Web yang Dibangun

Fungsional dari aplikasi *web* yang dibangun sebagian besar terletak pada bagian *backend*. Fungsionalitas utama dari aplikasi web tersebut adalah pencocokan pola dengan menggunakan KMP, BM, maupun ekspresi regular yang terdapat dalam folder algorithm pada folder backend. Selain algoritma, terdapat pula beberapa algoritma penunjang untuk menjawab pertanyaan pada bagian 3.1 seperti algoritma untuk menghitung hasil query kalkulator dan tanggal.

Selain algoritma, fungsionalitas yang terdapat jugalah CRUD bagi pertanyaan dan sejarah dari pertanyaan dan jawaban dari aplikasi. CRUD yang digunakan disini menggunakan REST API. Read menggunakan routes GET, seperti /history (GET) dan /question (GET) untuk mendapatkan sejarah dan pertanyaan tertentu. Selain itu, create dan update dari pertanyaan menggunakan POST karena ia akan mencoba untuk membuat pertanyaan baru, bila telah ada maka jawabannya akan diganti. Menghapus pertanyaan menggunakan routes DELETE, begitu pula dengan sejarah (POST untuk membuat sejarah baru di basisdata dan DELETE untu menghapusnya). Penyimpanan basis data dari aplikasi web yang dibangun terdapat pada Planetscale dan terhubung dengan *backend* melalui *library* mysql2.

4. IMPLEMENTASI DAN PENGUJIAN

4.1 Struktur Data

Berikut merupakan tabel basis data yang digunakan pada aplikasi web ini:

Tabel history

Field	Type	Null	Key	Default	Extra
id	bigint	NO	PRI		auto_increment
answer	text	NO			
question	text	NO			
time	timestamp	NO			

Tabel questions

Field	Type	Null	Key	Default	Extra
question	text	NO	PRI		
answer	text	NO			

Selain itu, terdapat pula beberapa struktur data penunjang dalam App.vue pada bagian *frontend*, antara lain:

1. messages → merepresentasikan isi dari *chat bubble* antara pengguna dan aplikasi
2. useKMP → mencatat apakah pengguna memilih KMP atau BM
3. chatHistory → sejarah dari chat antara pengguna dan aplikasi

4.2 Fungsi dan Prosedur

Fungsi dan Prosedur

Fungsi dan prosedur dibawah ini adalah yang penting atau sangat berkaitan dengan program

1. mainAlgorithm
   Fungsi utama algoritma. Fungsi ini menerima query, database, dan metode. Kemudian memanggil fungsi dan prosedur lain untuk memperoleh jawaban untuk semua pertanyaan dan menyimpannya dalam array. Setelah semua pertanyaan terjawab, maka list jawaban direturn.

2. splitQuestion
   Fungsi ini menerima query pertanyaan, kemudian memisahkannya berdasarkan constraint yang telah ditetapkan dan dimasukan dalam sebuah array. Fungsi ini kemudian mereturn list pertanyaan.

3. classifyQuestion dan createAnswer
   classifyQuestion merupakan fungsi yang mengklasifikasikan setiap pertanyaan yang diterima. Setelah itu pertanyaan akan dihandle sesuai tipenya di createAnswer. Fungsi ini kemudian mereturn list jawaban kepada mainAlgorithm.

4. kmpMatch dan bmMatch
   Fungsi ini bertujuan untuk melakukan pattern matching dengan metodenya masing-masing, KMP untuk kmpMatch dan BM untuk bmMatch, antara pertanyaan user dengan pertanyaan pada database, selain itu dicek juga ukuran kedua string tersebut supaya diperoleh exact match. Jika tidak exact match maka fungsi ini mereturn -1.

5. levenstheinDistance
   Fungsi ini tujuannya untuk menghitung kemiripan antara pertanyaan user dengan pertanyaan-pertanyaan pada database dengan metode Levensthein Distance. Fungsi ini akan mereturn persentase kemiripannya.

6. addQuestion
   Fungsi ini adalah fungsi yang menangani query untuk menambahkan pertanyaan. Pada fungsi ini pertanyaan yang akan ditambahkan dicari

terlebih dahulu di database. Bila ditemukan, maka jawaban dari pertanyaan tersebut akan diupdate, namun bila tidak ada maka pasangan pertanyaan dan jawaban tersebut akan ditambahkan.

7. deleteQuestion
   Fungsi ini adalah fungsi yang menangani query untuk menghapus pertanyaan. Pada fungsi ini pertanyaan yang akan dihapus dicari terlebih dahulu di database. Bila ditemukan maka pertanyaan tersebut akan dihapus dari database.

8. getDateAnswer
   Fungsi ini adalah fungsi yang menangani tipe pertanyaan hari. Fungsi ini akan memanggil fungsi lain untuk memvalidasi tanggal yang diterima kemudian bila valid akan mereturn hari pada tanggal tersebut.

9. getMathAnswer
   Fungsi ini adalah fungsi yang menangani tipe pertanyaan matematika. Fungsi ini kemudian memanggil fungsi-fungsi lain untuk menyelesaikan persoalan matematika tadi.

10. Controllers/index.js
    Berisi kumpulan fungsi-fungsi penting yang menghubungkan database dengan algoritma dan dengan frontend. Diantaranya terdapat get_ question dan get_history, yang berfungsi untuk mendapatkan data pertanyaan dan history dari database.

Dari *frontend* sendiri, beberapa fungsi berikut memiliki peran yang relatif lebih vital daripada fungsi lain, di antaranya :

1. submitMessage(msg)
   Fungsi ini bertugas menangani *event* pengguna melakukan *submit* pada pertanyaan yang telah ditulis. Pertama, fungsi ini akan melakukan query pada *backend* untuk meminta jawaban yang tepat untuk pertanyaan yang diberikan dan menampilkannya ke layar tampilan *chat*.

   Setelah itu, fungsi ini memastikan bahwa pasangan pertanyaan dan jawaban yang baru saja dibuat akan dikirim ke *database* sebagai sebuah riwayat pertanyaan.

   Terakhir, riwayat-riwayat pertanyaan baru akan kembali di-*fetch* dan ditetapkan sebagai nilai baru variabel yang akan diakses oleh tampilan riwayat chat.

2. DeleteMessage(id)
   Fungsi ini bertugas menangani permintaan penghapusan riwayat pesan dengan *id* tertentu. Setelah riwayat yang dimaksud telah dihapus dari

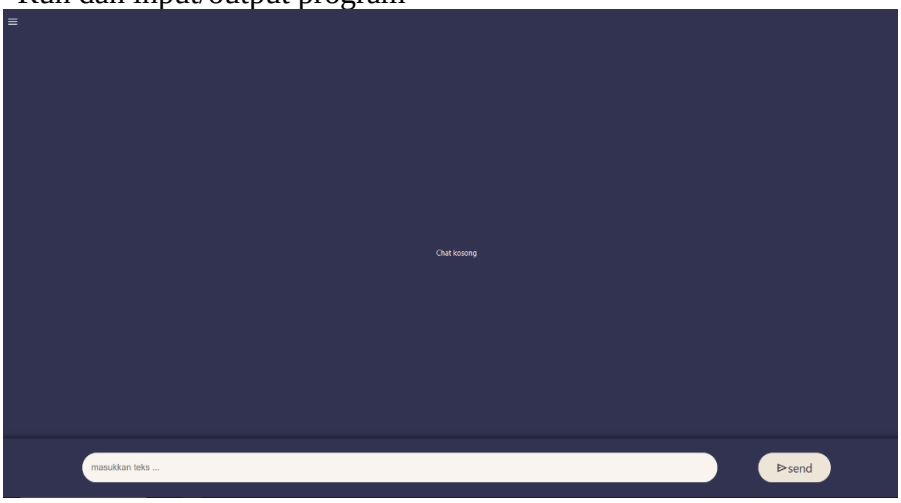
database, fungsi ini akan meminta daftar riwayat pertanyaan yang baru dan menetapkannya sebagai nilai baru tampilan riwayat chat. Sebagai catatan, fungsi ini dipanggil ketika pengguna menekan tombol sampah yang ada pada setiap item riwayat pertanyaan.

4.3 Tata Cara Menggunakan Program

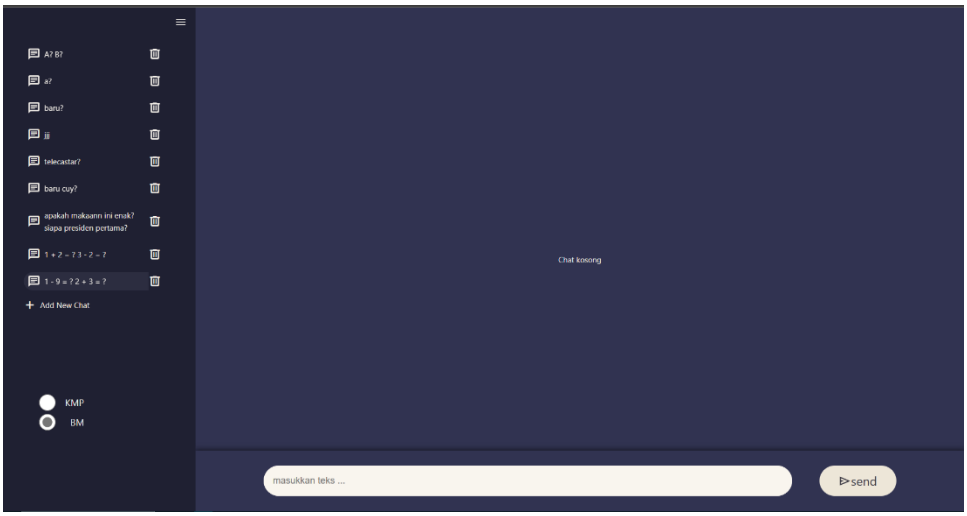
1. Arahkan terminal ke dalam folder *backend*
2. Jalankan perintah npm install
3. Arahkan terminal ke dalam folder *frontend*
4. Jalankan perintah npm install
5. Jalankan *backend* dengan mengarahkan terminal ke dalam folder *backend* dan jalankan perintah npm run start
6. Jalankan *frontend* dengan mengarahkan terminal ke dalam folder *frontend* dan jalankan perintah npm run host
7. Seharusnya aplikasi web akan berjalan pada beberapa pranala yang diberikan ketika perintah npm run host dijalankan (contoh: localhost:5173)

4.4 Hasil Pengujian

1. Run dan input/output program

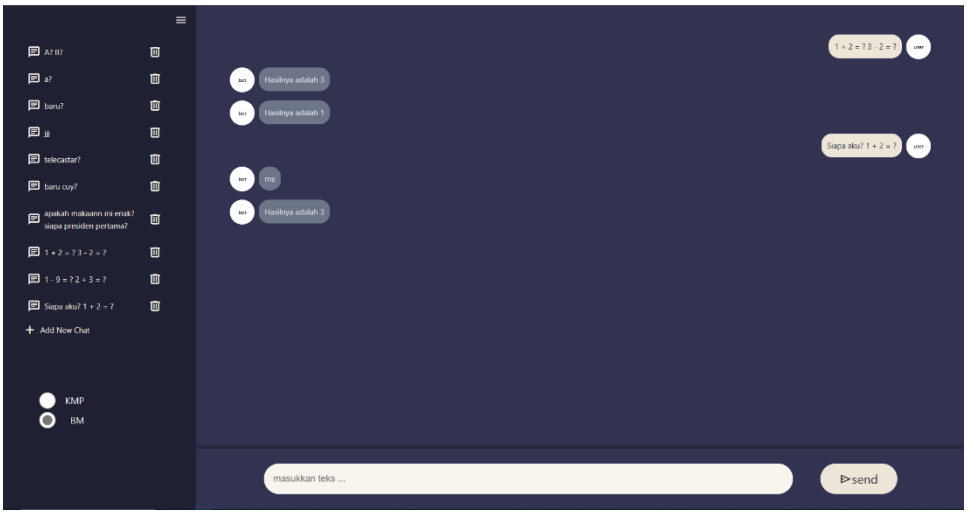


Saat pengguna pertama kali mengakses web, aplikasi akan menampilkan keterangan bahwa chat masih kosong. Pengguna dapat mengisi tampilan chat dengan mengirimkan pertanyaan baru atau melakukan akses pada histori pertanyaan yang pernah ditanyakan sebelumnya.

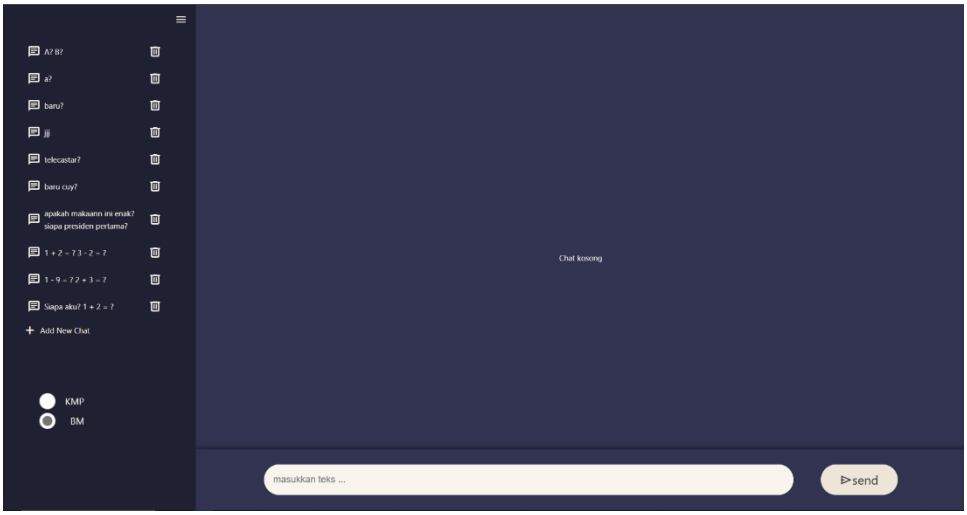


Setelah memilih salah satu *history* pertanyaan, tampilan akan berubah menampilkan riwayat pertanyaan tersebut dalam chat. *History* pertanyaan bersifat satu-satu (1 riwayat untuk 1 chat pertanyaan).

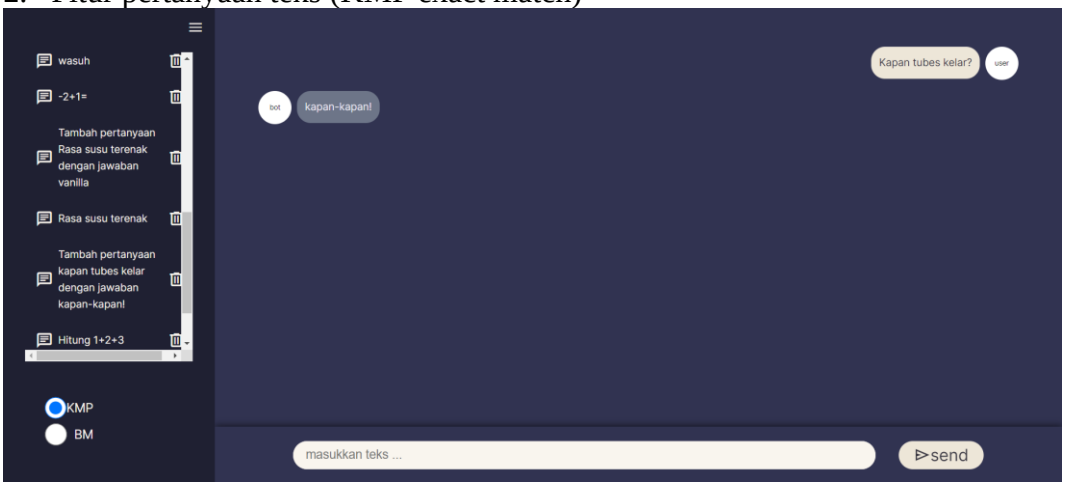
Setelah memasukkan dan mengirimkan pertanyaan, aplikasi akan menuliskan langsung jawaban pada layar chat tanpa membersihkan riwayat chat yang sebelumnya dipilih.



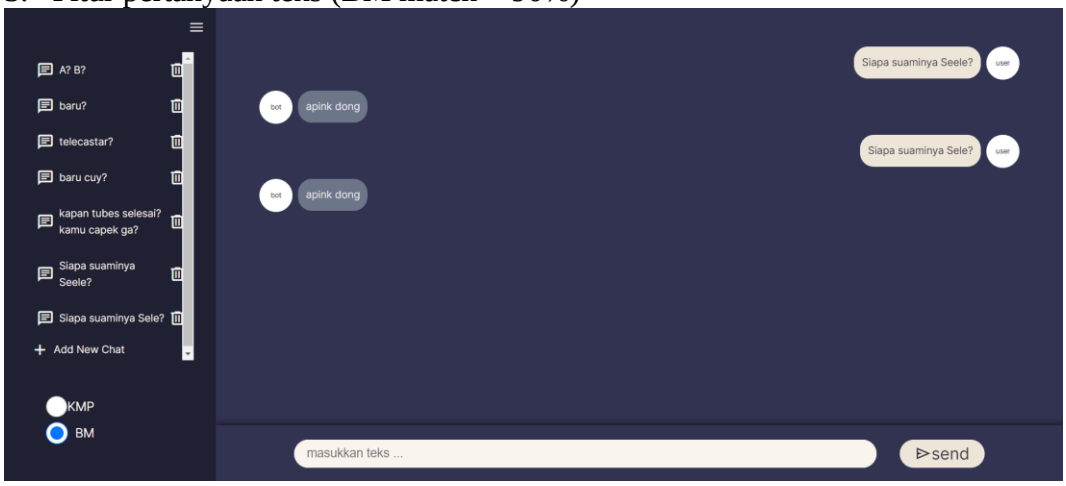
Pengguna dapat membersihkan layar (memulai layar baru) dengan meng-klik tombol Add New Chat.



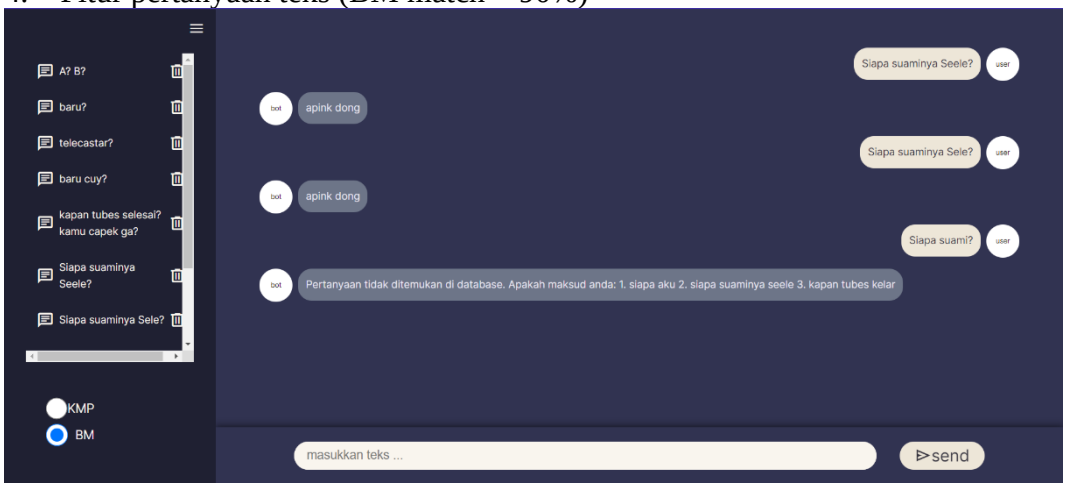
2. Fitur pertanyaan teks (KMP exact match)



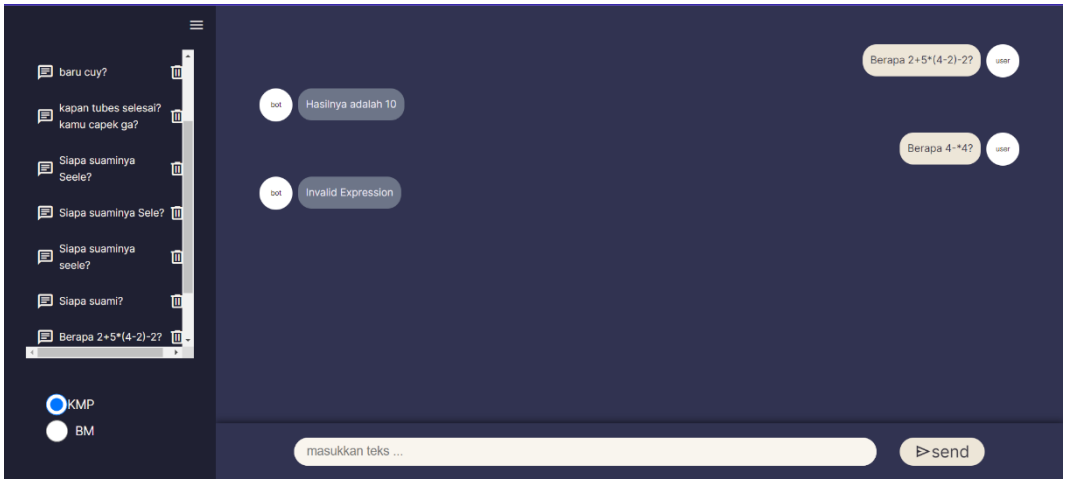
3. Fitur pertanyaan teks (BM match > 90%)



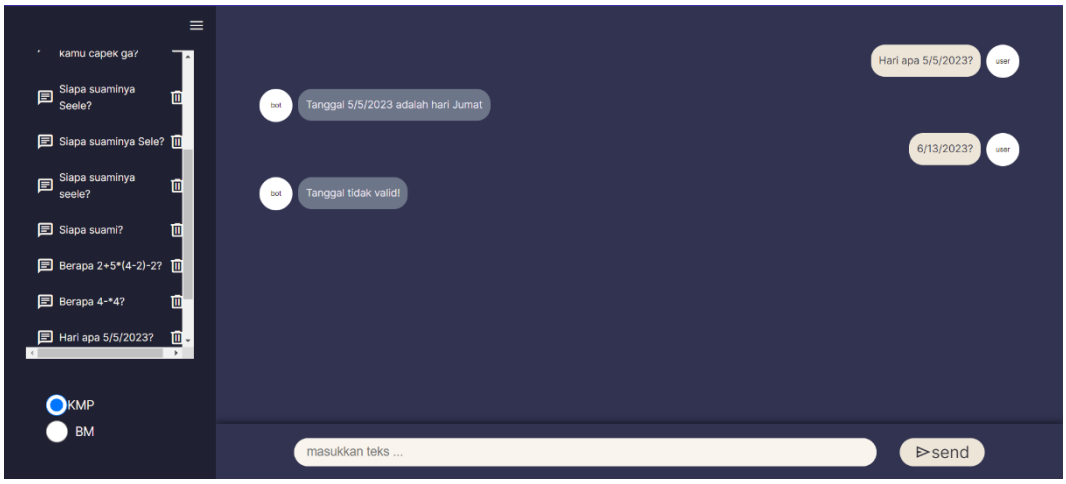
4. Fitur pertanyaan teks (BM match < 90%)



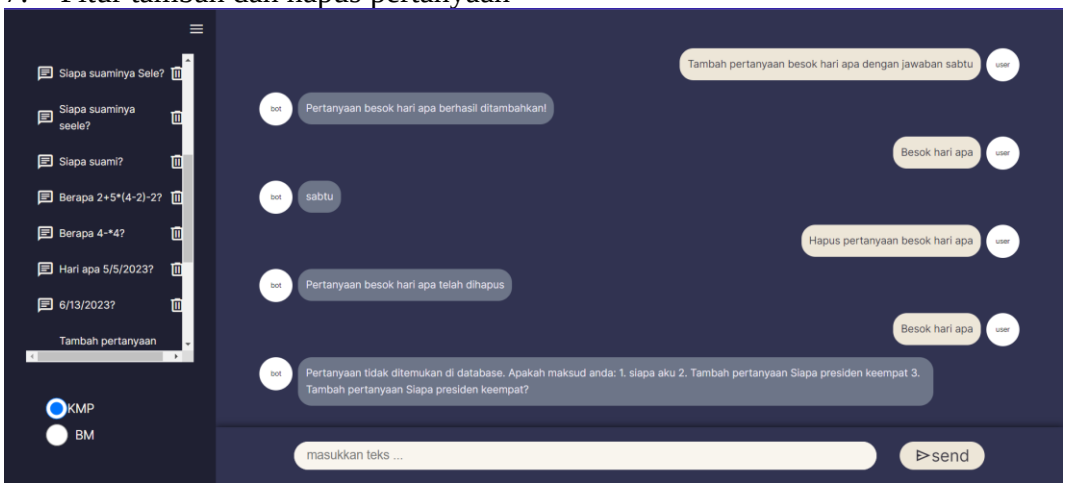
5. Fitur kalkulator



6. Fitur tanggal



7. Fitur tambah dan hapus pertanyaan



4.5 Analisis Hasil Pengujian

Pada pengujian 1 ditunjukkan bahwa program dapat berjalan. Selain itu ditunjukkan juga history chat pada sidebar. Dapat dilihat bahwa program dapat menerima input dari user, kemudian memberikan output/respon kepada user.

Pada pengujian 2 ditunjukkan fitur pertanyaan teks dengan kasus exact match. Karena pada database terdapat pertanyaan yang setelah dicek merupakan exact match, maka program kemudian mengirim output berupa jawaban dari pertanyaan tersebut.

Di pengujian 3 ditunjukkan kasus exact match dan kasus ketika terdapat sebuah typo pada pertanyaan. Meskipun terdapat typo namun kemiripannya dengan salah satu pertanyaan di database lebih dari 90% sehingga program mengiring jawaban dari pertanyaan itu sebagai output.

Pada pengujian 4 ditunjukkan kasus dimana tidak ada pertanyaan di database yang kemiripannya lebih dari 90%, oleh karena itu program mencari maksimal 3 pertanyaan dengan kemiripan tertinggi dan mengirimnya sebagai output.

Pada pengujian 5 ekspresi matematika pertama valid, sehingga program menghitung hasilnya dan mengirimkan hasilnya. Namun ekspresi matematika kedua invalid karena "-*", oleh karena itu program memberi output "Invalid Expression"

Pada pengujian 6 ditunjukkan fitur tanggal. Tanggal pertama valid sehingga program menentukan hari dari tanggal tersebut dan mengirimnya sebagai output. Sedangkan bulan pada tanggal kedua invalid dan program memberi output tanggal invalid.

Terakhir, dilakukan pengujian pada tambah dan hapus pertanyaan. Dapat dilihat bahwa pertanyaan yang ditambahkan, tersimpan di database dan dapat langsung ditanyakan oleh pengguna. Setelah itu pertanyaan tadi dihapus dan ketika pengguna mencoba menanyakannya maka tidak diperoleh jawabannya.

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

2. Metode KMP dan BM sangat reliable untuk mencari string pada sebuah text. Namun dalam kasus exact match, perlu dilakukan langkah tambahan untuk mencarinya, bahkan terkadang ditemukan substring yang exact match, yang menyebabkan terjadi kesalahan matching.
3. Regex sangat mudah digunakan untuk membedakan satu string dengan lainnya. Selain itu regex juga memiliki banyak manfaat lain dalam memanipulasi string, misalnya dalam menghapus maupun mereplace

5.2 Saran

2. Manfaat dari metode KMP dan BM kurang dirasakan karena pattern matching memerlukan kasus exact match.
3. Tidak perlu terlalu overthinking dalam membuat sebuah program. Misalnya program kalkulator yang menangani expresi matematika yang sangat aneh.
4. Gunakan function dan object yang telah ada untuk mempermudah dan efisiensi program, contohnya menggunakan object Date dari Javascript
5. Pengerjaan tugas besar ini beriringan dengan beberapa tugas besar lainnya. Oleh karena itu gunakan dan bagilah waktu sebaik mungkin agar tidak stress pada akhir pengerjaan tugas ini.

6. Referensi

Algoritma, A. S. (2023, May 5). *QnA Stima.* Retrieved from Google Sheets: https://docs.google.com/spreadsheets/d/1Q7anspE-qIu8G64wuuTfDt9KERPHaXbU2kolIE4q-Yw/edit#gid=2088364611

Algoritma, A. S. (2023, May 5). *Tugas Besar 3 IF2211 Stima 22/23.docx.* Retrieved from Google Docs: https://docs.google.com/document/d/1d1eimK5XlJ7QYSb4iUE4ZmBBTsg6e7vF/edit

Munir, R. (2023, May 5). *Pencocokan String (String/Pattern Matching).* Retrieved from PowerPoint Presentation: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

Munir, R. (2023, May 5). *String Matching dengan* . Retrieved from PowerPoint Presentation: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf

7. Lampiran

7.1. Tautan *Repository* GitHub
https://github.com/fakihap/Tubes3_13521078

7.2. Tautan Video YouTube
https://youtu.be/xj9HWPttJ8U