LAPORAN TUGAS BESAR III

IF2211 Strategi Algoritma

Penerapan String Matching dan Regular Expression dalam Pembuatan ChatGPT Sederhana

Fakhri Muhammad M	13521045
Hosea Nathanael A	13521057
Razzan Daksana Y	13521087

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022/2023

Daftar Isi

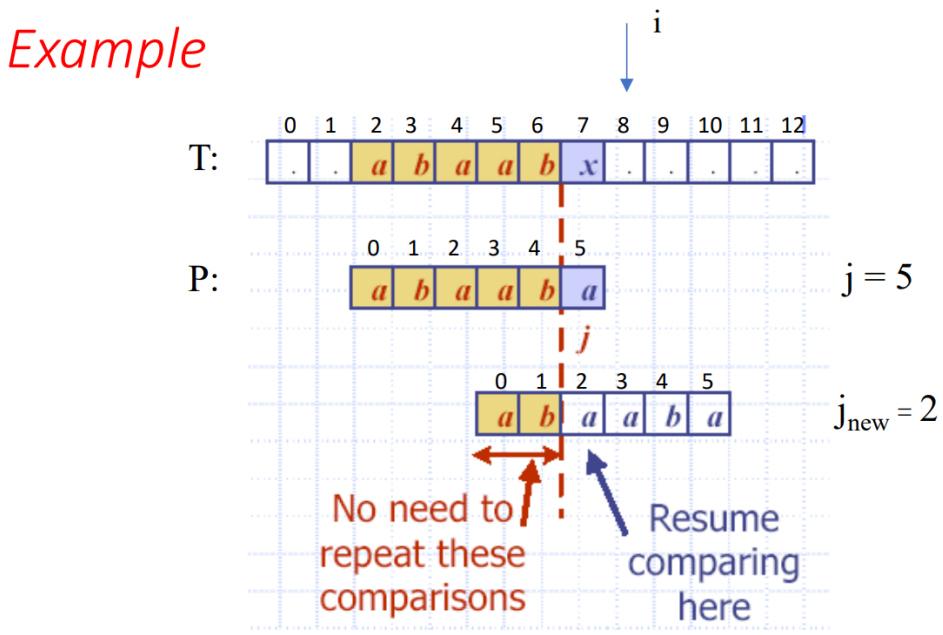
BAB I : Deskripsi Tugas

Dalam tugas besar 3 ini, anda diminta untuk membangun sebuah aplikasi Chat GPT 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.

BAB II : Landasan Teori

2.1. Algoritma Knuth-Morris-Pratt

Merupakan algoritma pencocokan *string* dengan urutan *left-to-right*. Pada algoritma ini, jika sebuah ketidakcocokan ditemukan, *pattern* pencocokan akan digeser sebanyak panjang karakter yang sudah dicocokkan dikurang ukuran terbesar *prefix* yang merupakan *suffix* pada *pattern* yang sudah dicocokkan. Contoh:



Gambar 2.1.1 Pencocokan *string* menggunakan algoritma KMP

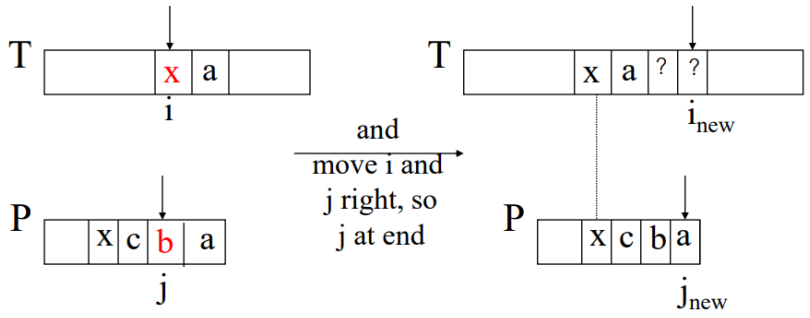
Pada gambar 2.1.1 dapat dilihat bahwa terdapat ketidakcocokan pada karakter ke-6, sehingga banyak karakter yang sudah diperiksa adalah 5. Pada *pattern* tersebut, *prefix* yang merupakan *suffix* juga adalah string “ab”. Maka ketika ditemukan ketidak cocokan, *pattern* akan digeser sejauh 5 - 3 = 2 kemudian pencocokan dilakukan kembali.

2.2. Algoritma Boyer-Moore

Algoritma Boyer-Moore merupakan algoritma pencocokan *string* dengan teknik *the looking-glass*, yaitu pencocokan dilakukan secara mundur pada *pattern* yang diperiksa, dan *character-jump*. Untuk melakukan pencocokan *string* dengan algoritma ini, diperlukan data posisi paling terakhir (*last occurence*) dari karkater yang ada pada *patern*.

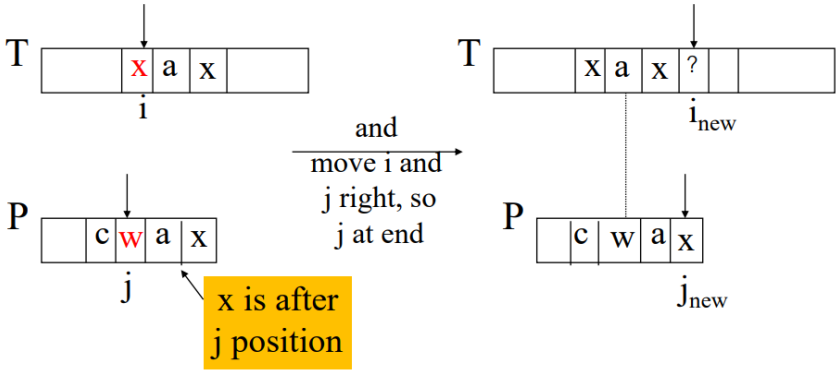
Terdapat 3 kasus yang dapat terjadi ketika ditemukan ketidakcocokan. Kasus pertama, ketika pada teks ditemukan ketidakcocokan, maka *pattern* digeser sehingga karakter yang tidak

cocok tersebut sejajar dengan *last occurence* dari karakter tersebut pada *pattern* kemudian pencocokan kembali dilanjutkan.



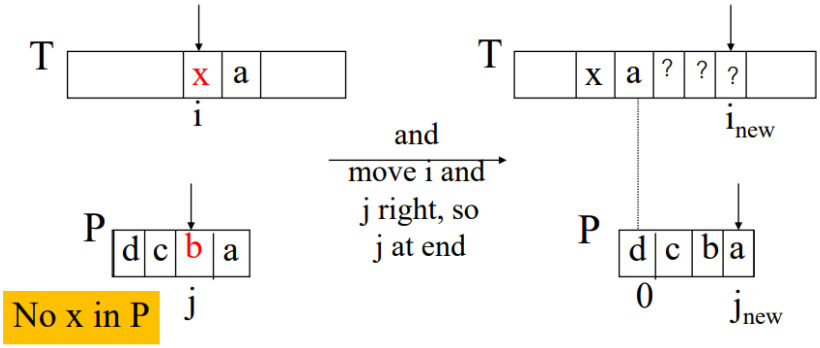
Gambar 2.2.1 Kasus pertama ketidakcocokan pada algoritma BM

Kasus kedua adalah ketika terjadi ketidakcocokan pada teks tetapi *last occurence* dari karakter tersebut sudah dilewati, maka *pattern* hanya digeser satu karakter ke kanan dan pencocokan dilakukan kembali.



Gambar 2.2.2 Kasus kedua ketidakcocokan pada algoritma BM

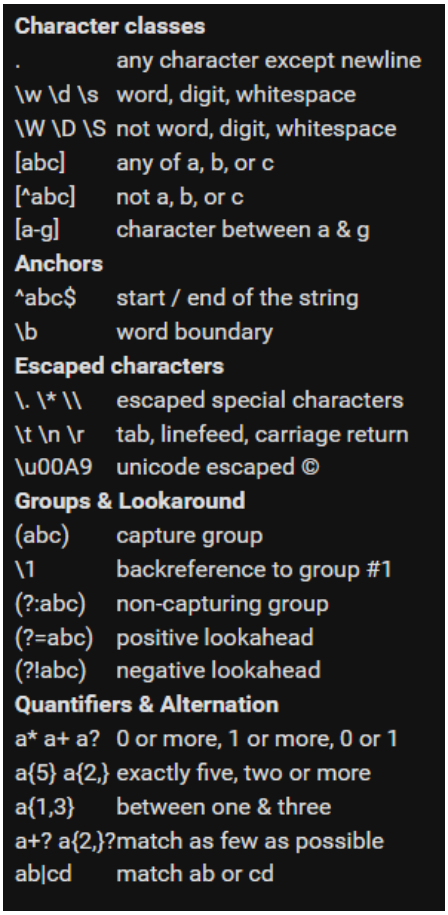
Kasus terakhir adalah ketika kasus pertama dan kedua tidak terpenuhi atau dalam kata lain bahwa ketika ketidakcocokan ditemukan pada teks dan karakter tersebut tidak terdapat pada *pattern*, maka *pattern* digeser sehingga karakter pertama *pattern* berada satu karakter di depan karakter yang tidak cocok tersebut.



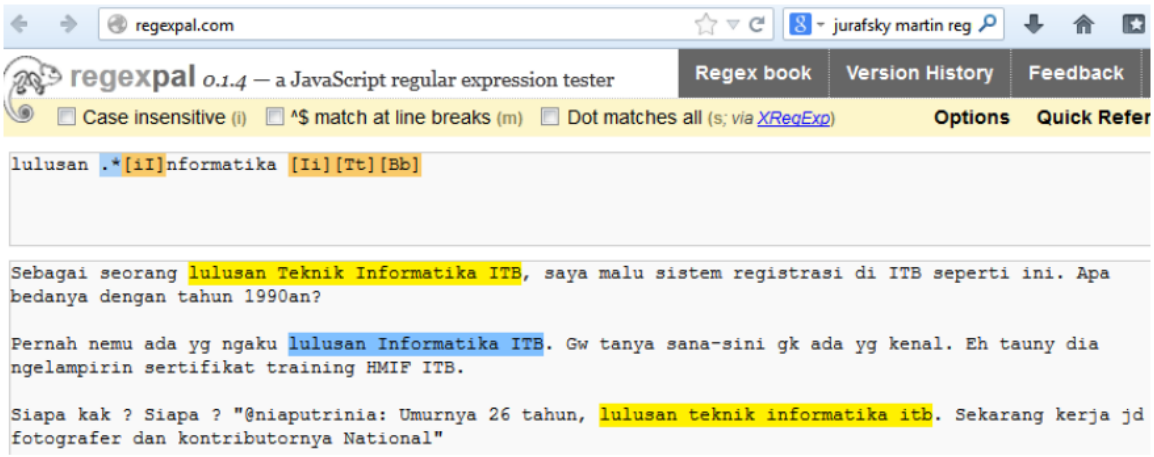
Gambar 2.2.3 Kasus ketiga ketidakcocokan pada algoritma BM

2.3. Regular Expression

Regular Expression atau Regex merupakan *string* yang menyatakan pola untuk menyocokkan *string* yang sesuai dengan pola tersebut. Pada algoritma-alogritma sebelumnya, pencocokan hanya dilakukan untuk menemukan *exact match* pada *string*. Dengan menggunakan Regular Expression, kita dapat menemukan *string* pada sebuah teks yang sesuai dengan pola yang diinginkan. Berikut beberapa aturan penulisan pola Regular Expression:



Gambar 2.3.1 Aturan penulisan pola pada Regex



Gambar 2.3.2 Contoh *string matching* dengan Regex

BAB III : Analisis Pemecahan Masalah

3.1. Pemecahan Masalah

Pada tugas ini, diperlukan sebuah website yang menjadi sarana interaksi pengguna dengan sistem dengan menanyakan pertanyaan. Program ini juga harus dapat menyimpan pembicaraan antara sistem dan pengguna yang dapat diakses kembali di lain waktu. Pertanyaan-pertanyaan yang dapat ditanyakan kepada sistem dapat ditambahkan atau dikurangi oleh pengguna melalui pesan yang disampaikan ke sistem dan ketika pengguna menanyakan pertanyaan yang tidak terdapat exact match dan berada di bawah 90% kecocokan, maka sistem akan menyarankan pertanyaan kepada pengguna, jika lebih dari atau sama dengan 90% maka jawaban dari pertanyaan yang paling sesuai ditampilkan oleh sistem. Selain pertanyaan yang ditambahkan oleh pengguna pada sistem, sistem juga harus dapat menjawab pertanyaan perhitungan sederhana seperti kalkulator dan juga menentukan hari dari tanggal yang diberikan pengguna dengan tidak case sensitive. Program juga harus bisa menyimpan history pembicaraan antara pengguna dengan sistem pada basis data.

3.2. Penerapan Algoritma KMP, BM, dan Regex

3.3.1 KMP dan BM

Algoritma KMP digunakan untuk mencari exact match antara pertanyaan yang diberikan pengguna dan pertanyaan yang ada di database. Jika ditemukan kesamaan persis, maka jawaban yang yang berkorespondensi dengan pertanyaan tersebut di database yang akan di output kepada pengguna. Hal yang sama juga dilakukan pada Algoritma BM jika pengguna memilih pilihan algoritma tersebut.

3.3.2 Regex

Regex digunakan untuk mencari format pertanyaan yang sesuai sehingga program bisa memakai algoritma yang sesuai untuk menghandle kasus tersebut.

Sebagai contoh untuk mengidentifikasi sebuah pertanyaan matematika untuk di handle calculator, regex yang digunakan merupakan seperti ini

```
^((hitung|hasil|berapa)?\s*([0-9\+\-*/^\s\(\)\.]+)(=?[\?\s]*))$
```

Regex tersebut bisa dipakai untuk menentukan banyak format ekspresi matematika yang mengandung digit, +, -, *, /, ., (, atau) dan mungkin berawalan 'hitung', 'hasil, atau 'berapa' serta memiliki akhiran = atau ?.

Selain itu regex juga dipakai pada file calculator.ts untuk mengidentifikasi format digit yang memiliki titik (.) dan yang tidak memiliki titik. Hal ini dipakai lebih lanjut untuk mengevaluasi persamaan matematika di calculator.ts

3.3. Pemecahan Masalah Setiap Fitur

Untuk membentuk *website* ini, digunakan Next js dan Chakra UI untuk mengimplementasikan *Frontend* dan Node.js pada *Backend*. Penyimpanan data pertanyaan dan *history* diimplementasikan menggunakan basis data dengan DBMS MySQL. Pencocokan pertanyaan pada *input* pengguna menggunakan algoritma KMP, BM, dan Regex, dilakukan pada keseluruhan *input* pengguna dan dilakukan pada *Backend*.

3.4. Analisis Kemungkinan Kasus

Pada tugas besar ini terdapat beberapa permasalahan yang harus diidentifikasi dan diatasi akan dibahas masalah-masalah tersebut dan

Pada sebuah permasalahan tentu terdapat beberapa kasus unik yang memiliki resiko tidak tertangani oleh algoritma program. Dalam konteks tugas besar ini, file test case yang diberikan akan mengandung kasus unik yang menguji batasan algoritma program. Berikut contoh kemungkinan kasus unik dan upaya dalam menanganinya.

1. Unary operator

   Dalam kasus ini program sudah menangani kasus dengan penanganan yang tepat
2. Pertanyaan tidak terdapat di *database*

   Program akan mengembalikan tiga pertanyaan yang paling dekat dengan pertanyaan yang ada di *database*

BAB IV : Implementasi dan Pengujian

4.1. Menjalankan Program

Untuk memulai program, pastikan berada pada *root directory* dari *repository* kemudian gunakan *command* **npm i** untuk instalasi keperluan program dan gunakan **npm run dev** pada terminal untuk memulai programnya. Pastikan sudah memiliki Node.js sebelum memulai program.

4.2. Spesifikasi dan Struktur Data, Fungsi, dan Prosedur Program

4.2.1 Fungsi dan Prosedur

1. **BM**

BM	
String matching dengan algoritma Boyer-Moore	
Input	**Output**
Text : String Pattern : String	Index karakter pertama pada string yang sudah cocok dengan pattern, -1 jika tidak ditemukan

2. **computeLastOccurrence**

computeLastOccurrence	
Membuat map yang menyimpan data kemunculan terakhir dari sebuah karakter	
Input	**Output**
Pattern : string	Map<string, number> yang menyimpan kemunculan terakhir suatu karakter pada pattern

3. **KMP**

KMP
String matching dengan algoritma Knuth-Morris-Pratt

Input	Output
Text : String Pattern : String	Mengembalikan indeks pertama dari pattern di text, -1 jika tidak ditemukan

4. **levenshetain_distance**

levenshetain_distance	
String matching dengan algoritma levenshtein	
Input	**Output**
s : String t : String	Mengembalikan levenshtein distance dari pencocokan string

5. **similarityScore**

similarityScorre	
Menentukan tingkat kesamaan string dengan pattern	
Input	**Output**
S : String T : String	Mengembalikan nilai kesamaan string dengan pattern

6. **dateQuestionHandler**

dateQuestionHandler	
Memproses pertanyaan tentang tanggal dari pengguna	
Input	**Output**
Text : String	Mengembalikan hari dari tanggal yang diberikan

7. **validateDate**

validateDate
Memvalidasi input tanggal yang diberikan

Input	Output
Date : number[]	Mengembalikan boolean valid atau tidak dan pesannya

8. **gaussAlgorithm**

gaussAlgorithm	
Menentukan hari dari tanggal yang telah diberikan	
Input	**Output**
Date : number[]	Mengembalikan teks hari pada tanggal yang ditentukan

9. **op**

op	
Melakukan penghitungan pada dua angka dan operatornya	
Input	**Output**
A : number B : number Op : string	Mengembalikan hasil operasi angka *a* dan *b* dengan operator *op*

10. **inputPrecendence**

inputPrecedence	
Menentukan prioritas operator	
Input	**Output**
C : string	Mengembalikan nilai prioritas operator

11. **stackPrecedence**

stackPrecedence	
Menentukan prioritas operator pada stack	
Input	**Output**

C : string	Mengembalikan nilai prioritas operator

12. processInfix

processInfix	
Memproses input string yang diberikan untuk sesuai dengan ketentuan operasi matematika	
Input	**Output**
Exp : string	Mengembalikan string yang dapat diproses perhitungannya

13. infixToPostfix

infixToPostfix	
Memproses perhitungan dengan implementasi stack	
Input	**Output**
Exp : string	Mengembalikan stack perhitungan

14. evaluatePostfix

evaluatePostfix	
Memproses perhitungan pada stack	
Input	**Output**
Exp : string	Mengembalikan hasil perhitungan matematika

15. isDot

isDot	
Menentukan apakah sebuah karakter merupakan titik	
Input	**Output**

C : string	Mengembalikan true jika c merupakan titik dan false jika tidak

16. isDigitWithDot

isDigitWithDot	
Menentukan apakah string merupakan angka desimal	
Input	**Output**
C : string	Mengembalikan true jika c merupakan string angka desimal

17. isOperator

isOperator	
Menentukan apakah karakter sebuah operator	
Input	**Output**
C : string	Mengembalikan true jika c adalah operator dan false jika tidak

18. isParenthesis

isParenthesis	
Menentukan apakah karakter tanda kurung	
Input	**Output**
C : string	Mengembalikan true jika c adalah tanda kurung dan false jika tidak

19. isDigit

isDigit
Menentukan apakah string merupakan sebuah angka

Input	Output
C : string	Mengembalikan true jika string adalah angka dan false jika tidak

20. isNumber

isDigit	
Menentukan apakah string merupakan angka	
Input	**Output**
C : string	Mengembalikan true jika string adalah angka dan false jika tidak

21. isLeftParenthesis

isLeftParenthesis	
Menentukan apakah karakter merupakan tanda buka kurung	
Input	**Output**
C : string	Mengembalikan true jika karakter adalah tanda buka kurung dan false jika tidak

22. isRightParenthesis

isRightParenthesis	
Menentukan apakah karakter merupakan tanda tutup kurung	
Input	**Output**
C : string	Mengembalikan true jika karakter adalah tanda tutup kurung dan false jika tidak

23. addQnAHandler

addQnAHandler	
Memproses string penambahan set pertanyaan dan jawaban dari pengguna	
Input	**Output**
pattern : string	Mengembalikan set pertanyaan dan jawaban yang akan disimpan ke basis data

24. deleteQnAHandler

deleteQnAHandler	
Memproses string penghapusan set pertanyaan dan jawaban dari pengguna	
Input	**Output**
pattern : string	Mengembalikan pertanyaan yang akan dihapus dari basis data

25. bracketMatcher

bracketMatcher	
Memproses input tanda kurung sehingga selalu berpasangan	
Input	**Output**
text : string	Mengembalikan true jika penggunaan kurung sudah sesuai dan false jika tidak

26. mathQuestionHandler

mathQuestionHandler	
Memproses string perhitungan ekspresi matematika	
Input	**Output**
text : string	Mengembalikan teks hasil perhitungan atau pesan error

27. deleteQnA

deleteQnA	
Menghapus set pertanyaan dan jawaban pada basis data	
Input	**Output**
Id_reference : number	-

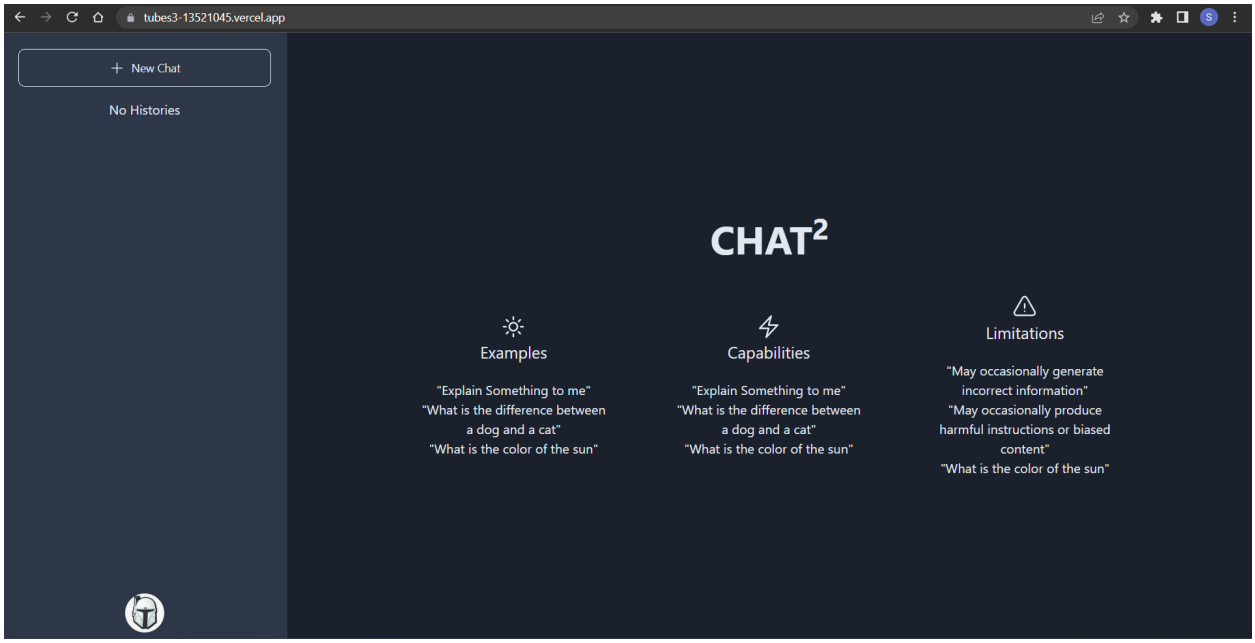
28. addQnA

addQnA	
Menambahkan set pertanyaan dan jawaban pada basis data	
Input	**Output**
newQuestion : string newAnswer : string	-

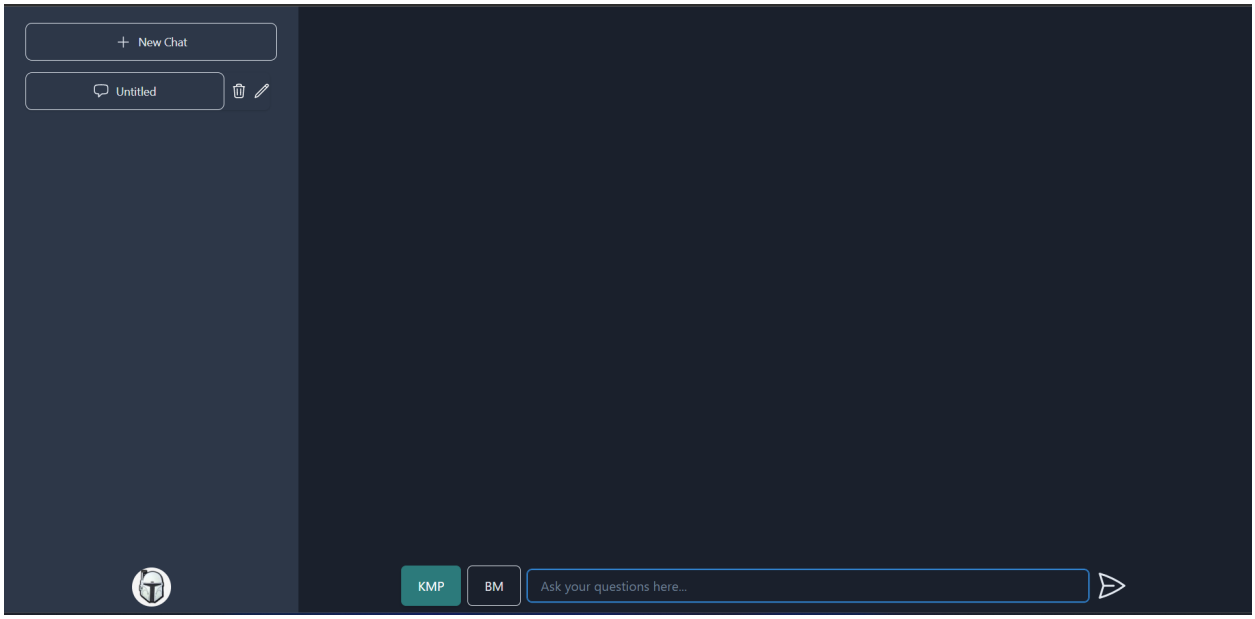
29. mathQuestionHandler

mathQuestionHandler	
Prosesor utama input pengguna untuk menentukan jawaban dari sistem	
Input	**Output**
Pattern : string isKMP : boolean	Jawaban dari pertanyaan, pesan sukses atau gagal penambahan atau penghapusan pertanyaan

4.3. Tata Cara Penggunaan



Pada laman ini ditampilkan laman utama, pengguna dapat menekan tombol New Chat untuk membuka satu sesi *chat* untuk menggunakan program

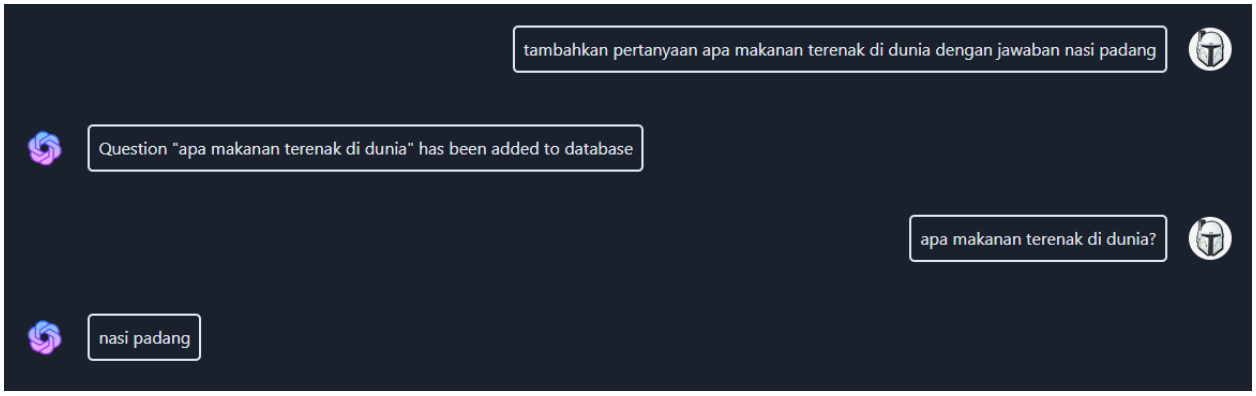


Pengguna dapat mengubah ataupun menghapus sesi chat dengan menekan tombol trash dan pencil. Dengan menekan satu sesi *chat* pengguna akan ditampilkan untuk menggunakan program yang penulis buat. Lalu masukkan text ke dalam input box dan

pengguna dapat menekan tombol enter pada keyboard ataupun tombol pesawat untuk mengirim pertanyaan ke program, pengguna juga dapat mengubah toggle algoritma yaitu terletak di sebelah kiri input box.

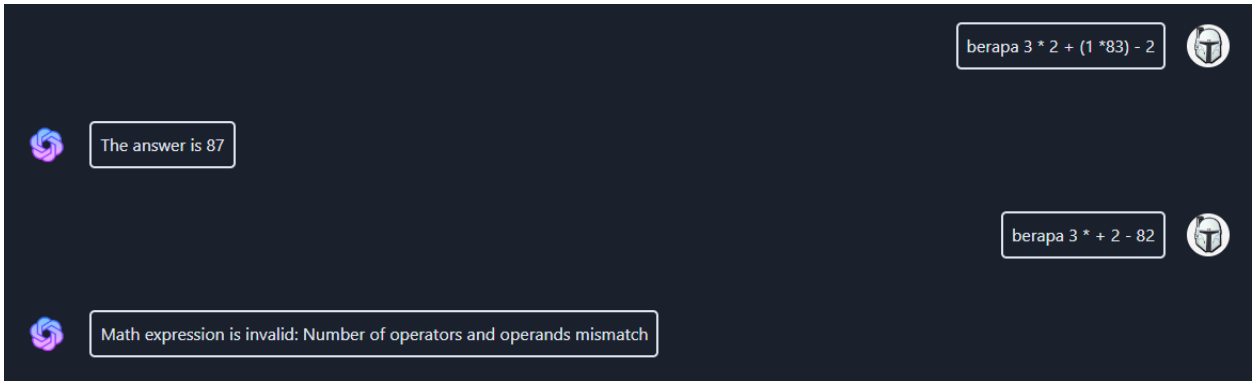
4.4. Eksperimen

4.5.1. Penambahan pertanyaan

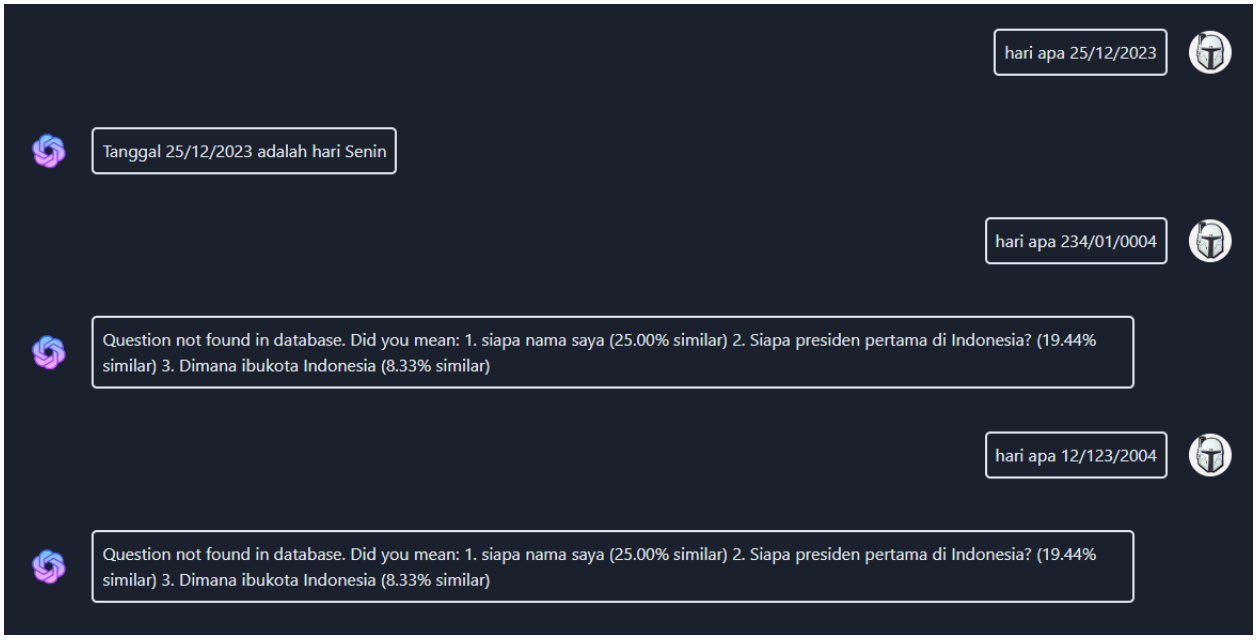


4.5.2. Penghapusan pertanyaan

4.5.5. Kalkulator



4.5.6. Penentuan hari



4.5. Analisis Hasil Eksperimen

BAB V : Kesimpulan dan Saran

5.1. Kesimpulan

Dapat disimpulkan bahwa implementasi program pembicaraan antara pengguna dengan sistem dapat diterapkan dengan penggunaan algoritma Knuth-Morris-Pratt, Boyer-Moore, dan Regular Expression menggunakan aplikasi berbasis *web*. Dengan algoritma KMP, BM, dan RE, *input* pertanyaan dan perintah lainnya dari pengguna dapat dicocokan sehingga sistem dapat menemukan jawaban yang sesuai. Interaksi dari pengguna dengan sistem juga dapat diimplementasikan menggunakan *web* dan basis data untuk ditampilkan dan disimpan.

5.2. Saran

Penulis menyarankan untuk pengembang selanjutnya untuk kode yang dibuat pada program ini dikembangkan dari sisi modularitas, strategi, penampilan dan kerapihannya. Penulis juga menyarankan untuk eksplorasi dan juga pemanfaatan waktu yang baik pada pengembangan program.

5.3. Refleksi

Penulis merasa dengan adanya tugas ini, penulis dapat mempelajari dan memperdalam pengetahuannya tentang algoritma *string matching* dan juga *web development*. Tugas ini juga membantu penulis untuk mengeksplorasi dan berhasil mengimplementasikan teori yang telah dipelajari pada permasalahan yang sebenarnya.

Daftar Pustaka

1. Spesifikasi Tugas Besar 3 Strategi Algoritma
2. Bahan Kuliah IF2211 Strategi Algoritma - Pencocokan String (String/Pattern Matching)
3. Bahan Kuliah IF2211 Strategi Algoritma - Pencocokan String dengan Regular Expression
4. Wagner Fisher Algorithm
5. Menentukan hari pada tanggal

Lampiran

Link Repository GitHub : https://github.com/razzanYoni/tubes3_13521045.git

Link deployment : https://tubes3-13521045.vercel.app/

Link Video : https://youtu.be/iigJfOSzfbU