LAPORAN TUGAS BESAR III

Penerapan String Matching dan Regular Expression dalam Pembuatan ChatGPT Sederhana

Oleh:

13521088 Puti Nabilla Aidira

13521132 Dhanika Novlisariyanti

13521170 Haziq Abiyyu Mahdy

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022/2023

DAFTAR ISI

BAB 1: 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 KMKMP 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 pertanyan 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)

BAB 2: Landasan Teori

2.1 Dasar teori

a. Knuth-Morris-Pratt (KMP)

Knuth-Morris-Pratt adalah algoritma *string matching* yang mencari *pattern* dalam sebuah text dari kiri ke kanan dengan menghindari komparasi elemen string S yang sudah di komparasi sebelumnya dengan pattern p. Untuk mencari KMP, dibutuhkan sebuah *longest proper suffix* untuk memberikan informasi berapa banyak karakter yang dapat di *skip*.

Algoritma LPS:

1. Pertama, inisialisasi array kosong dan elemen array pertama diisi nol. Lalu inisialisasi sebuah variabel untuk menyimpan banyak perhitungan prefix dan suffix
2. Jika array[len] dan array[i] memiliki kata yang sama variabel tadi ditambah 1 dan dimasukan ke dalam array[i]
3. Jika array[len] dan array[i] tidak sama, maka variabel tadi di set kembali menjadi nol dan mengupdate variabel trus menjadi array[len-1]

Algoritma KMP:

1. Komparasi pattern dengan string dimulai dari i = 0
2. Looping terus dilakukan antara pattern dan string sampai salah satu kata dari pattern dan string *match*
3. Ketika terjadi *mismatch,*
    a. Setelah melewati looping tadi, sebuah substring dari string dan pattern sampai index yang *mismatch* adalah sama sehingga tidak perlu di *match* lagi dan dapat dimulai dari index tersebut untuk *match*
    b. Jika selanjutnya tidak match, maka index akan pindah lagi ke nol.

b. Boyer-Moore (BM)

Boyer-Moore adalah algoritma *string matching* yang menyamakan *pattern* dan *string* dari karakter terakhir *pattern.* Untuk memproses boyer-moore, pertama harus membuat array berisi *Bad Character Heuristic*.

Algoritma *Bad Character Heuristic*:

1. Karakter yang tidak sama dengan karakter *pattern* disebut *bad character*
2. Selama karakter tidak sama, pattern akan digeser sampai *match* dan pattern tersebut bergerak melati karakter yang *mismatch*

Algoritma Boyer-Moore:

1. Pertama kita mencari posisi dari *last occurrence* dari karakter yang *mismatch* dan jika karakter tersebut terdapat dalam *pattern* maka *pattern* akan digeser sampai dia sejajar dengan karakter yang *mismatch* dalam string
2. Jika posisi dari *last occurrence* dari *mismatch* karakter dalam *pattern* jika tidak ada maka *pattern* akan digeser melewati *mismatch* karakter.

c. Levenshtein distance

Levenshtein distance adalah angka yang menghitung seberapa berbedanya dua *string.* Seberapa berbedanya kedua string ditentukan dari seberapa banyak *string* kedua membutuhkan berapa kali perubahan agar sama dengan *string* pertama. Banyaknya perubahan yang terjadi diambil dari huruf terakhir dalam matriks mengalami berapa kali perubahan.

$$\operatorname{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \operatorname{lev}_{a,b}(i-1,j) + 1 \\ \operatorname{lev}_{a,b}(i,j-1) + 1 \\ \operatorname{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

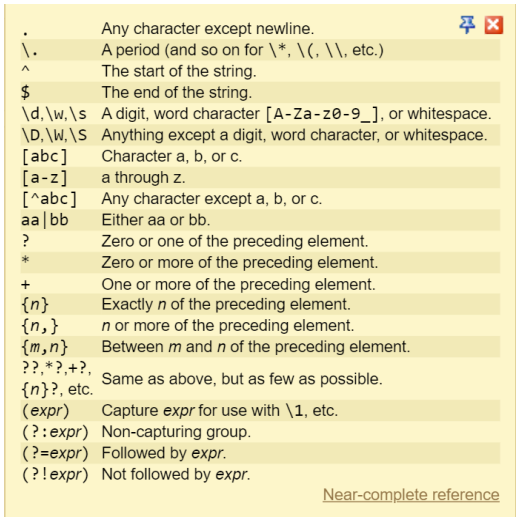
Original

Sumber:

https://medium.com/@ethannam/understanding-the-levenshtein-distance-equation-for-beginners-c4285a5604f0

d. *Regular expression* (regex)

Regular expression merupakan suatu pola yang berisi sekumpulan karakter yang biasa digunakan untuk melakukan pencocokan *string*. *Regular expression* sering digunakan pada algoritma pencarian *string*, operasi 'find and replace' pada *text editor*, atau pada validasi input. *Regular expression* ditemukan pada 1951 ketika matematikawan Stephen Cole Kleene menjelaskan bahasa reguler menggunakan notasi matematikanya yang disebut *regular events*. Notasi *regular expression* merupakan bagian dari ilmu teori komputasi dalam subbidang teori automata. Berikut adalah sintaks yang umum digunakan pada *regular expression*



`.`	Any character except newline.
`\.`	A period (and so on for `\*`, `\(`, `\\`, etc.)
`^`	The start of the string.
`$`	The end of the string.
`\d`,`\w`,`\s`	A digit, word character `[A-Za-z0-9_]`, or whitespace.
`\D`,`\W`,`\S`	Anything except a digit, word character, or whitespace.
`[abc]`	Character a, b, or c.
`[a-z]`	a through z.
`[^abc]`	Any character except a, b, or c.
`aa|bb`	Either aa or bb.
`?`	Zero or one of the preceding element.
`*`	Zero or more of the preceding element.
`+`	One or more of the preceding element.
`{n}`	Exactly *n* of the preceding element.
`{n,}`	*n* or more of the preceding element.
`{m,n}`	Between *m* and *n* of the preceding element.
`??`,`*?`,`+?`, `{n}?`, etc.	Same as above, but as few as possible.
`(expr)`	Capture *expr* for use with `\1`, etc.
`(?:expr)`	Non-capturing group.
`(?=expr)`	Followed by *expr*.
`(?!expr)`	Not followed by *expr*.

Near-complete reference

Sumber: https://stevenlevithan.com/regexpal/

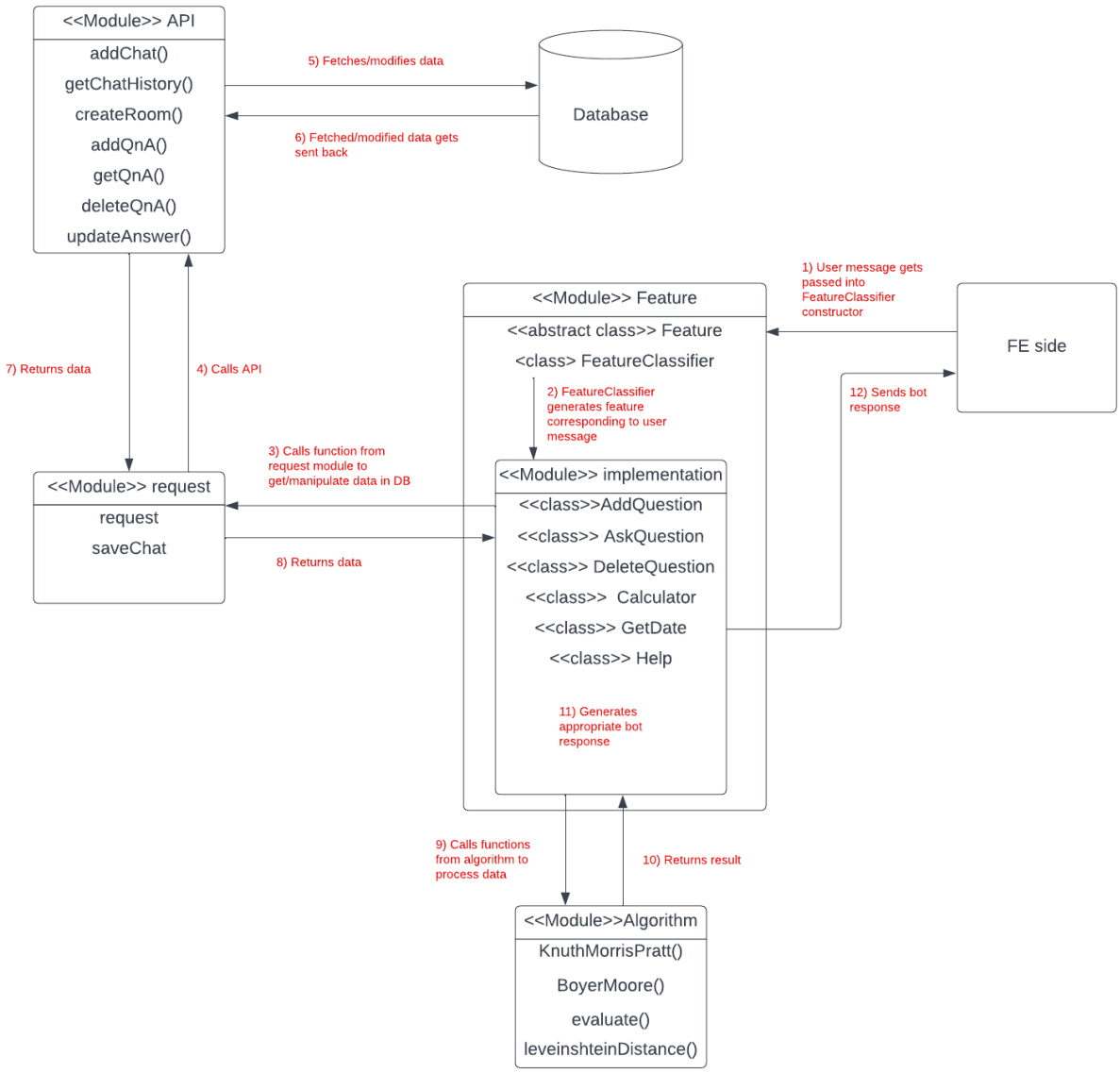
Secara umum, cara kerja *regular expression* adalah dengan menggunakan *regex processor* yang dapat menerjemahkan *regular expression* menjadi representasi internal yang dapat dimanfaatkan komputer untuk melakukan pencocokan *string*. Salah satu pendekatan yang dapat dilakukan adalah Thompson's *construction algorithm* yang dapat membangun *Nondeterministic Finite Automata* (NFA) dari suatu *regular expression* kemudian NFA tersebut diubah menjadi *Deterministic Finite Automata* (DFA). *String* yang akan dicocokkan akan dimasukkan ke dalam DFA, kemudian DFA akan memutuskan apakah *string* tersebut dikenali atau tidak.

Bahasa TypeScript menyediakan kelas RegExp dapat digunakan untuk melakukan operasi pencocokan *string* dengan *regular expression*. Kelas RegExp memiliki *method* match() yang menerima parameter *string* dan mencocokkan *string* tersebut dengan *regular expression*.

Jika tidak ditemukan kecocokan, *method* tersebut akan mengembalikan null dan jika ditemukan kecocokan, *method* tersebut akan mengembalikan informasi *string* yang ditemukan kecocokannya.

2.2 Penjelasan singkat aplikasi *web* yang dibangun

Aplikasi *web* yang dibangun menggunakan *framework* Next.js dan bahasa pemrograman TypeScript. Untuk *styling,* digunalan framework CSS *Tailwind*. Sebagai *database management systems*, digunakan PostgreSQL dengan ORM prisma, dan supabase sebagai *platform hosting server database*. Untuk *deployment* digunakan platform hosting vercel. Berikut adalah arsitektur aplikasi secara umum.


<<Module>> API
addChat()
getChatHistory()
createRoom()
addQnA()
getQnA()
deleteQnA()
updateAnswer()
5) Fetches/modifies data
6) Fetched/modified data gets sent back
Database
1) User message gets passed into FeatureClassifier constructor
FE side
<<Module>> Feature
<<abstract class>> Feature
<class> FeatureClassifier
2) FeatureClassifier generates feature corresponding to user message
12) Sends bot response
7) Returns data
4) Calls API
3) Calls function from request module to get/manipulate data in DB
<<Module>> request
request
saveChat
8) Returns data
<<Module>> implementation
<<class>>AddQuestion
<<class>> AskQuestion
<<class>> DeleteQuestion
<<class>> Calculator
<<class>> GetDate
<<class>> Help
11) Generates appropriate bot response
9) Calls functions from algorithm to process data
10) Returns result
<<Module>>Algorithm
KnuthMorrisPratt()
BoyerMoore()
evaluate()
leveinshteinDistance()

BAB 3: Analisis Penyelesaian Masalah

3.1 Langkah-langkah pemecahan masalah

1. **Mencocokan pertanyaan input dan memberikan respon**

   Pengguna akan mengetik pertanyaan pada kolom *textfield* jika pengguna belum memilih ruang percakapan maka akan terbentuk otomatis ruang percakapan baru. Ketika masukan sudah diterima, input akan di parsing untuk memeriksa apakah dalam satu input terdapat beberapa pertanyaan. Ketika berhasil di parsing, input akan diklasifikasikan ke dalam regex. Jika terdapat kata 'tambahkan', maka program akan otomatis menambahkan pertanyaan dan jawaban yang diberikan input ke dalam database. Jika terdapat kata 'hapus', maka program akan melakukan hal sebaliknya. Jika input yang diidentifikasi adalah sebuah tanggal/pertanyaan yang mengandung tanggal, maka program akan mengeluarkan hari yang bersesuaian dengan tanggal tersebut. Jika input yang diidentifikasi adalah sebuah ekspresi matematika, maka program akan mengeluarkan hasil perhitungan ekspresi tersebut. Jika input yang diidentifikasi adalah string biasa, maka akan diidentifikasi lagi. Jika tidak termasuk salah satu regex diatas, maka program akan menjawab pertanyaan sesuai database. Program akan menjawab pertanyaan dengan algoritma KMP atau BM sesuai pilihan pengguna. Jika tidak terdapat *exact match*, maka program akan menggunakan *levenshtein distance* dan mengeluarkan tiga pertanyaan termirip dari input. Jika input pengguna tidak dapat diidentifikasi sama sekali, maka program akan mengeluarkan *output* "Pertanyaan tidak dapat diproses".

2. **Menambah dan menghapus pertanyaan ke dalam database**

   Seperti penjelasan sebelumnya, pengguna dapat menambahkan dan menghapus pertanyaan ke dalam database sekaligus memberikan jawabannya. Pengguna dapat menggunakan *keyword* 'tambahkan' atau 'hapus' untuk fitur ini.

3. **Membuat ruang percakapan baru**

   Pengguna dapat membuat ruang percakapan baru dengan menekan tombol "+ New Chat" pada bagian kiri atas. Jika ketika pengguna membuka aplikasi di awal dan tidak memilih ruang maka program akan menciptakan ruang percakapan baru otomatis.

3.2 Fitur

Terdapat beberapa fitur dalam aplikasi web:

1. Memberikan respon sesuai klasifikasi input berdasarkan regex
    a. Memberikan respon untuk input pertanyaan teks. Respon didapat menggunakan algoritma pencocokan KMP atau BM dan algoritma kemiripan Levenshtein Distance. Pencocokan dilakukan ke database kumpulan pertanyaan dan jawaban.
    b. Memberikan respon untuk input ekspresi matematika (fitur kalkulator). Respon berupa hasil evaluasi ekspresi yang didapat menggunakan algoritma Shunting Yard.
    c. Memberikan respon untuk input tanggal. Respon berupa hari yang bersesuaian dengan input tanggal.
    d. Fitur ‘help’ untuk menjelaskan cara menggunakan aplikasi
2. Menambah dan menghapus pertanyaan dalam database
3. Membuat ruang percakapan baru

Arsitektur yang digunakan dalam web adalah prisma dan Node.js untuk backend, Next.js dan React js untuk frontend. Database disimpan dalam bentuk PostgreSQL menggunakan supabase dan deploy frontend menggunakan vercel. Penghubungan antara backend dan frontend menggunakan bantuan Next.js serta prisma

BAB 4: Implementasi dan pengujian.

4.1 Spesifikasi teknis program

a. Struktur Data

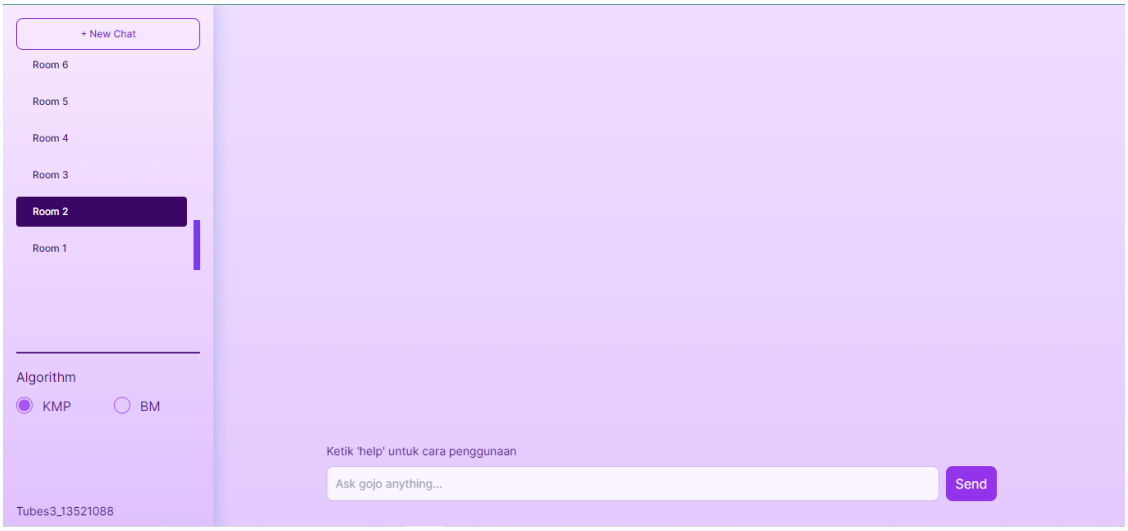
Struktur Data	Keterangan
string	Sebagai parameter input dalam algoritma KMP, BM, levenshtein, dan regex.
QuestionAndAnswer	Sebagai parameter return dari database berisi id, question, dan answer.
foundQuestion	Sebagai parameter return dari database berisi question dan percentage.
Messages	Sebagai parameter return database dengan messageId, question, answer, chatHistoryRoomID.
chatRoom	Sebagai parameter return dari database dengan roomId, roomName, messages.
Questions	Sebagai representasi format chat pada GUI yang juga digunakan untuk keperluan penyimpanan chat pada database. Field: id, text, responses.

b. Fungsi dan Prosedur

LPS(toMatch: string)	Fungsi untuk mencari lowest prefix dan suffix yang dibutuhkan untuk memproses algoritma KMP dengan return pattern array berisi angka
knuthMorrisPratt(pattern:string, data:QuestionAndAnswer[])	Fungsi untuk pencocokan string dengan KMP dengan return exact match
badCharacter(pattern:string)	Fungsi untuk mencari bad character dari sebuah pattern untuk memproses algoritma BM dengan return pattern array berisi angka
boyerMoore(pattern:string, data:QuestionAndAnswer[])	Fungsi untuk pencocokan string dengan BM dengan return exact match
levenshteinDistance(pattern: string, data: QuestionAndAnswer[]):	Fungsi untuk mencari string yang memiliki persen pencocokan kurang dari 100

evaluate (expression:string, regex:RegExp): number | undefined	Fungsi untuk mengevaluasi ekspresi matematika dengan menggunakan algoritma Shunting Yard.
getFeature()	Method kelas featureClassifer yang mengembalikan fitur (berupa objek) yang sesuai dengan hasil pencocokan regex.
getResponse()	Method abstrak kelas feature yang mengembalikan string respon yang sesuai. Implementasi method ini dilakukan pada kelas turunan feature sesuai dengan fungsionalitas tiap fitur.
getQuestionandAnswer()	Mengembalikan data pertanyaan dan jawaban dari basis data dengan return QuestionandAnswer[]
getQnA()	Mengembalikan data pertanyaan dan jawaban dari basis data dengan return Promise<QuestionAndAnswer>
addQnA(question:string, answer:string)	Fungsi yang digunakan untuk menambah pertanyaan dan jawaban ke dalam basis data
updateAnswer(id: number, newAnswer:string)	Fungsi yang digunakan untuk meng-*update* jawaban ke dalam basis data
deleteQnA(id:number)	Fungsi yang digunakan untuk menghapus pertanyaan dan jawaban ke dalam basis data
getAllRoom()	Fungsi yang mengembalikan semua ruang percakapan ke dalam program dengan return Promise<chatRoom[]>
getRoomChatHistory(roomId:number)	Fungsi yang mengembalikan ruang percakapan ke dalam program berdasarkan idRoom yang ingin dikembalikan dengan return Promise<chatRoom>
createRoom()	Fungsi yang digunakan untuk membuat ruang percakapan baru dan menambahkannya ke dalam basis data
addChat(question:string, answer:string, chatHistoryRoomId:number)	Fungsi yang digunakan untuk menambahkan percakapan berupa pertanyaan dan jawaban beserta roomId ke dalam basis data

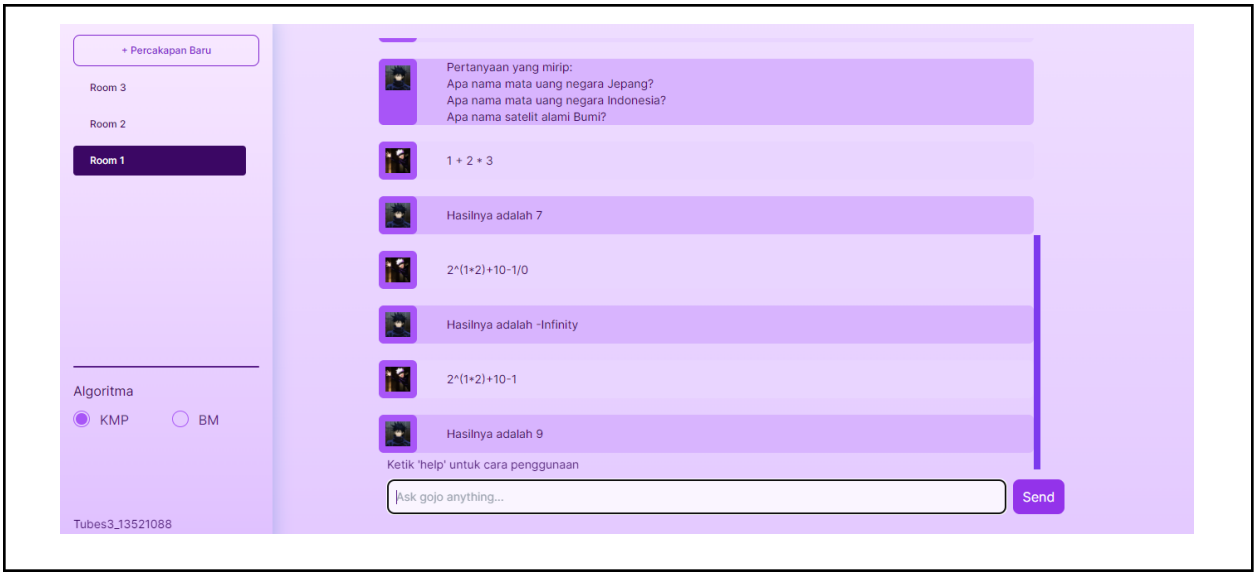
4.2 Penjelasan tata cara penggunaan program



Secara umum tampilan program seperti diatas, pengguna akan memilih ruang percakapan yang tersedia atau membuat ruang percakapan baru. Jika pengguna belum memilih ruang percakapan, maka tidak dapat mengirim input apa-apa. Jika sudah memilih ruang percakapan, maka pengguna akan memilih algoritma yang dipilih. Jika pengguna mengirim input terlebih dahulu tanpa memilih algoritma, input tidak akan bisa diproses kecuali memasukan operasi matematika sederhana. Program akan mengeluarkan "Pertanyaan tidak dapat di proses". Jika pengguna sudah memilih algoritma dan mengirim input, maka program akan mengeluarkan tampilan sesuai dengan algoritma yang dipilih dan fitur berdasarkan hasil input.

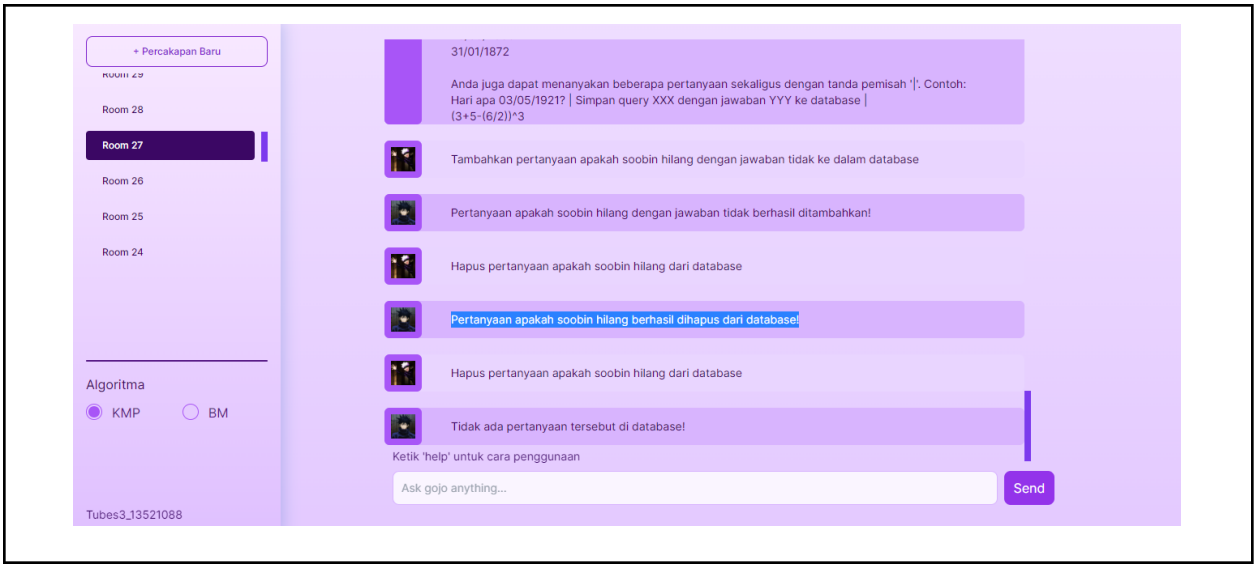
4.3 Hasil pengujian

TESTING 1	
Skenario pengujian	Calculator
Pesan input	2^(1*2)+10-1/0
Respon *bot*	Hasilnya adalah -Infinity
Tampilan aplikasi	



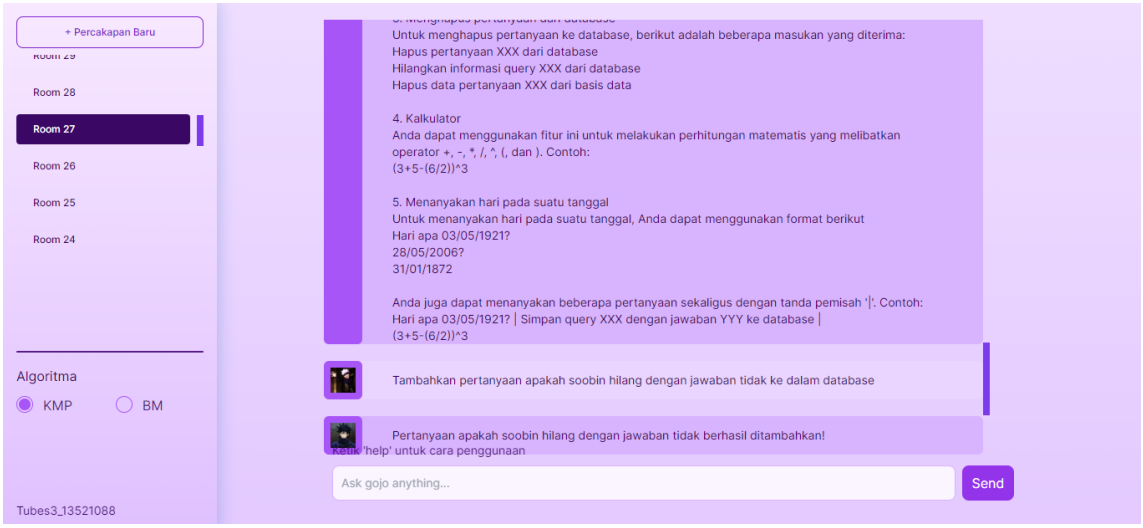
Tabel 4.3.1 Testing 1

TESTING 2	
Skenario pengujian	Hapus pertanyaan
Pesan input	Hapus pertanyaan apakah soobin hilang dari database
Respon *bot*	Pertanyaan apakah soobin hilang berhasil dihapus dari database!
Tampilan aplikasi	



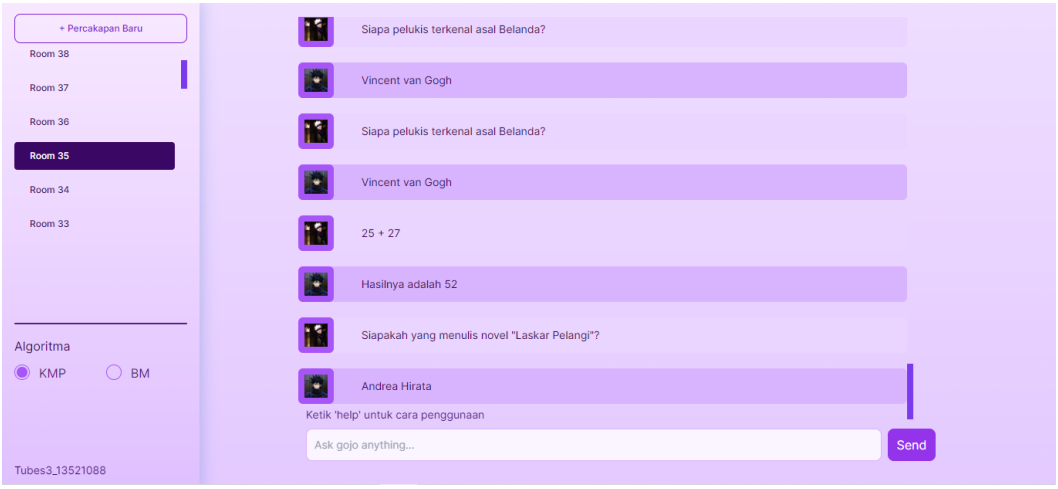
Tabel 4.3.1 Testing 2

TESTING 3	
Skenario pengujian	Tambahkan Pertanyaan
Pesan input	Tambahkan pertanyaan apakah soobin hilang dengan jawaban tidak ke dalam database
Respon *bot*	Pertanyaan apakah soobin hilang dengan jawaban tidak berhasil

	ditambahkan!
Tampilan aplikasi 	

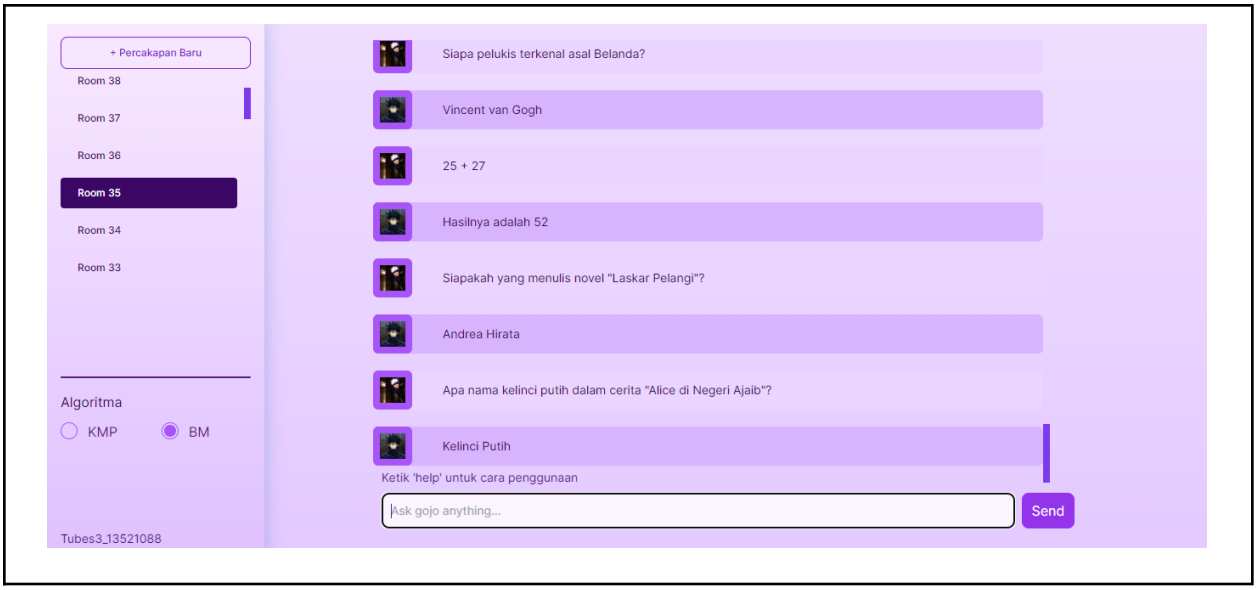
Tabel 4.3.1 Testing 3

TESTING 4	
Skenario pengujian	Algoritma KMP Exact Match
Pesan input	Siapakah yang menulis novel "Laskar Pelangi"?
Respon *bot*	Andrea Hirata
Tampilan aplikasi	



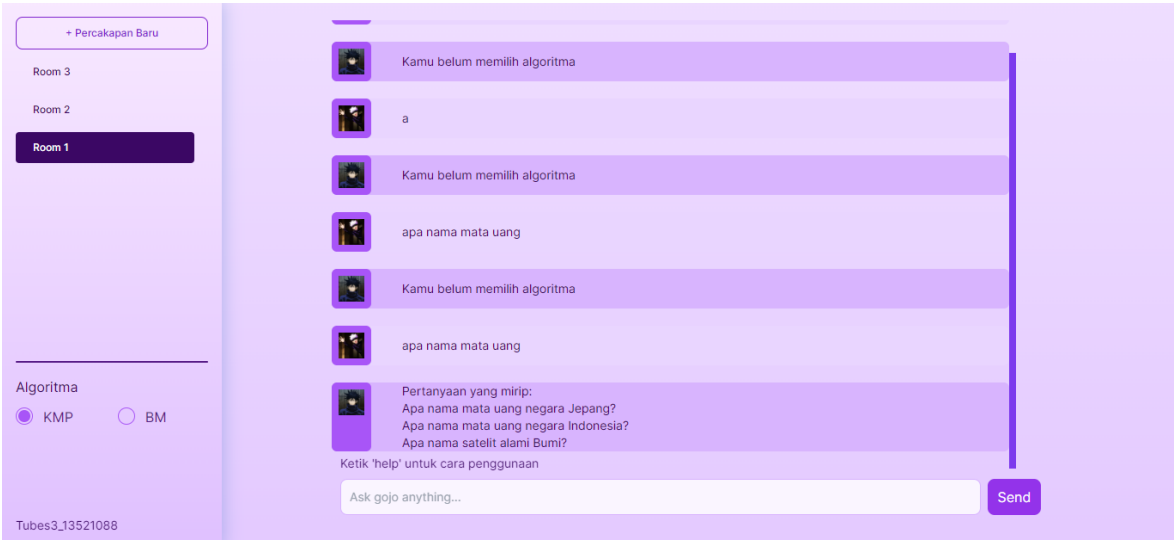
Tabel 4.3.4 Testing

TESTING 5	
Skenario pengujian	Algoritma BM Exact Match
Pesan input	Apa nama kelinci putih dalam cerita "Alice di Negeri Ajaib"?
Respon *bot*	Kelinci Putih
Tampilan aplikasi	



Tabel 4.3.5 Testing

TESTING 6	
Skenario pengujian	Algoritma KMP/BM Tidak Exact Match
Pesan input	Apa nama mata
Respon *bot*	Pertanyaan yang mirip: Apa nama mata uang negara Jepang?

	Apa nama mata uang negara Indonesia? Apa nama satelit alami Bumi?
Tampilan aplikasi 	

Tabel 4.3.6 Testing 6

4.3 Analisis hasil pengujian

Berdasarkan hasil pengujian, program berhasil menerima input dan tersimpan dalam *history room chat* dengan klasifikasi menggunakan regex. Aplikasi juga dapat memberikan jawaban pertanyaan dari input, menghitung operasi matematika sederhana, memberi tahu tanggal, bulan, dan hari serta menambahkan dan menghapus pertanyaan dari basis data.

BAB 5: Kesimpulan dan Saran

5.1. Kesimpulan

Pattern yang diklasifikasikan menggunakan regular expression dapat berjalan dengan baik dan berjalan sesuai regular expression yang tersedia. Algoritma KMP dan BM dapat berfungsi untuk mencari pattern yang exact match. Jika tidak terdapat exact match, pattern akan dicari menggunakan levenshtein distance. Levenshtein distance akan mengembalikan tiga *match* terbaik sesuai perhitungan.

5.2. Saran dan refleksi

Untuk tugas berikutnya akan lebih baik jika implementasi autentikasi agar *history* tiap laptop berbeda dan mendesain tampilan yang lebih baik dan responsif. Tugas ini lumayan seru dan menantang karena pada tugas ini kami belajar untuk membuat *web app dan mendeploy website kami.*

Daftar Pustaka dan Link Terkait

Link github: https://github.com/haziqam/Tubes3_13521088.git
Link youtube: https://www.youtube.com/watch?v=QdpfhQ4mdHM&ab_channel=PutiNabillaAidira
Link deploy: https://tubes3-13521088-sepia.vercel.app/

https://www.geeksforgeeks.org/boyer-moore-algorithm-for-pattern-searching/

https://www.geeksforgeeks.org/kmp-algorithm-for-pattern-searching/

https://medium.com/@ethannam/understanding-the-levenshtein-distance-equation-for-beginners-c4285a5604f0