Tugas Besar III IF2211 Strategi Algoritma

PENERAPAN STRING MATCHING DAN REGULAR EXPRESSION DALAM PEMBUATAN CHATGPT SEDERHANA

Diajukan sebagai pemenuhan tugas besar III.

Oleh:

Kelompok 10 (**GPT-(-1)**)

1. 13521100 - Alexander Jason
2. 13521114 - Farhan Nabil Suryono
3. 13521137 - Michael Utama

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

DAFTAR ISI

BAB 1
DESKRIPSI MASALAH

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.

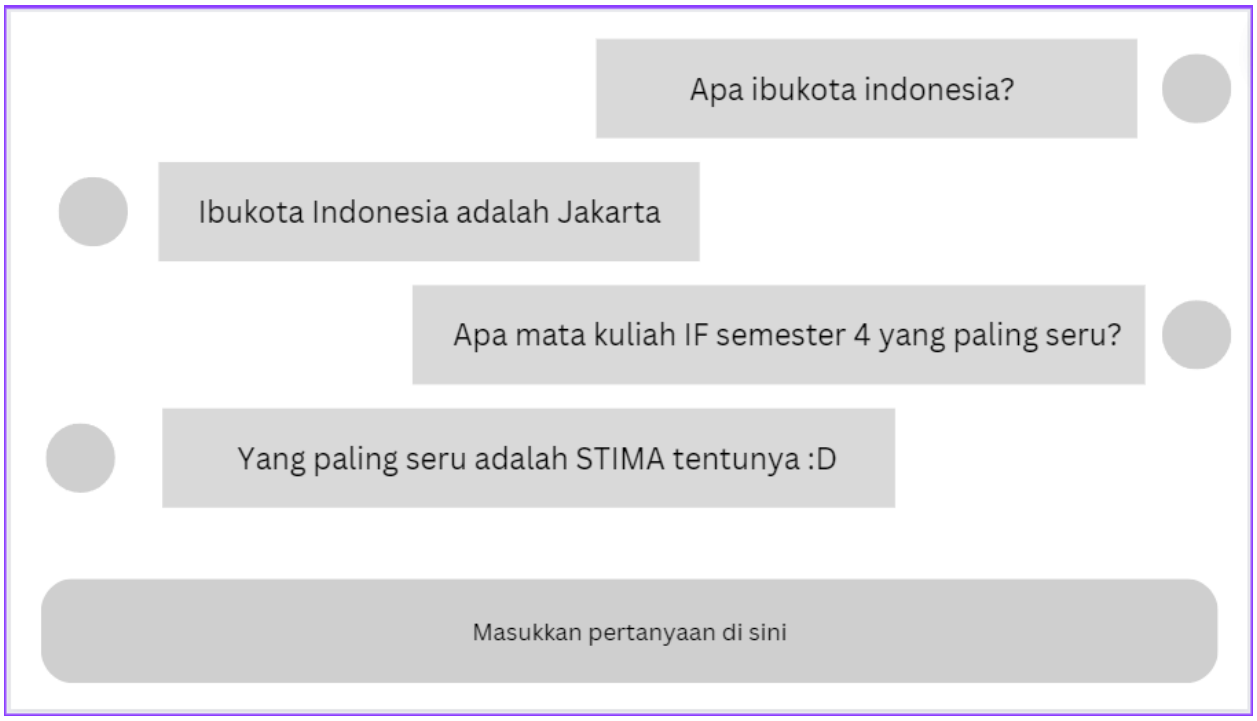
Perhitungan tingkat kemiripan dibebaskan kepada anda asalkan dijelaskan di laporan, namun disarankan menggunakan salah satu dari algoritma Hamming Distance, Levenshtein Distance, ataupun Longest Common Subsequence.

Fitur-Fitur Aplikasi:

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

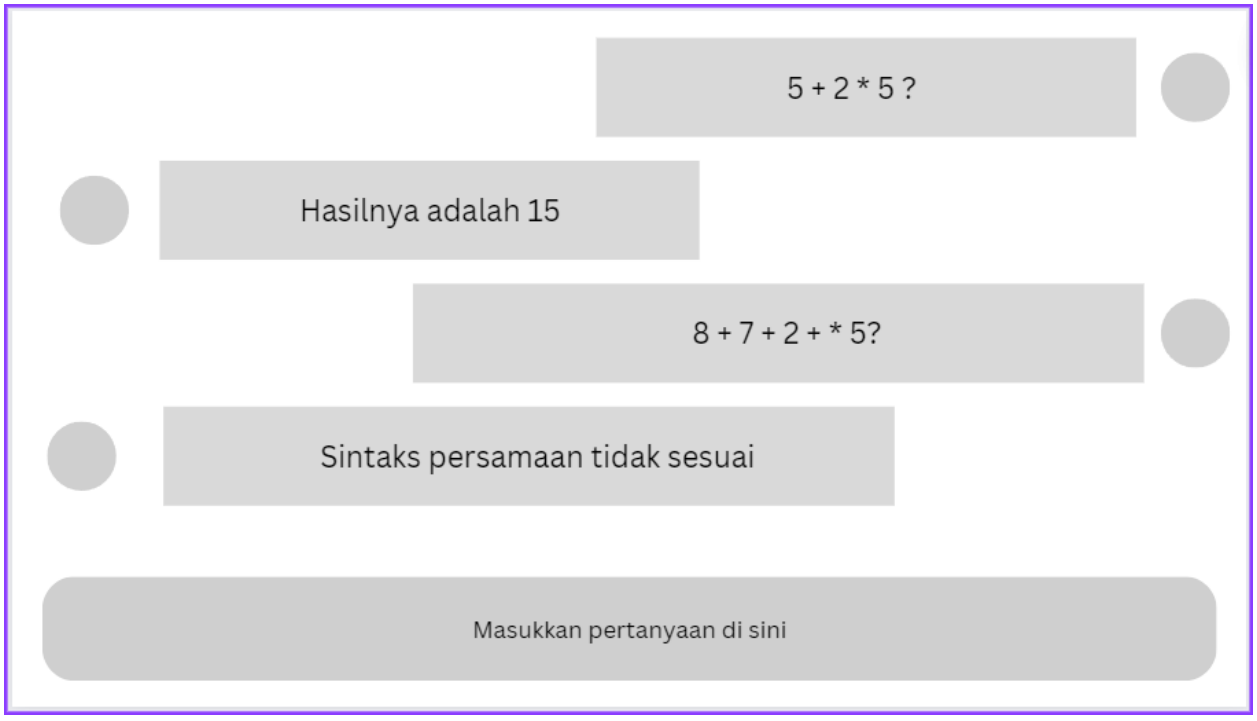
1. **Fitur pertanyaan teks (didapat dari database)**
   Mencocokkan pertanyaan dari input pengguna ke pertanyaan di database menggunakan algoritma **KMP atau BM**.
2. **Fitur kalkulator**
   Pengguna memasukkan input query berupa persamaan matematika. Contohnya adalah 2*5 atau 5+9*(2+4). Operasi cukup 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 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** pada website bagi pengguna 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**)**



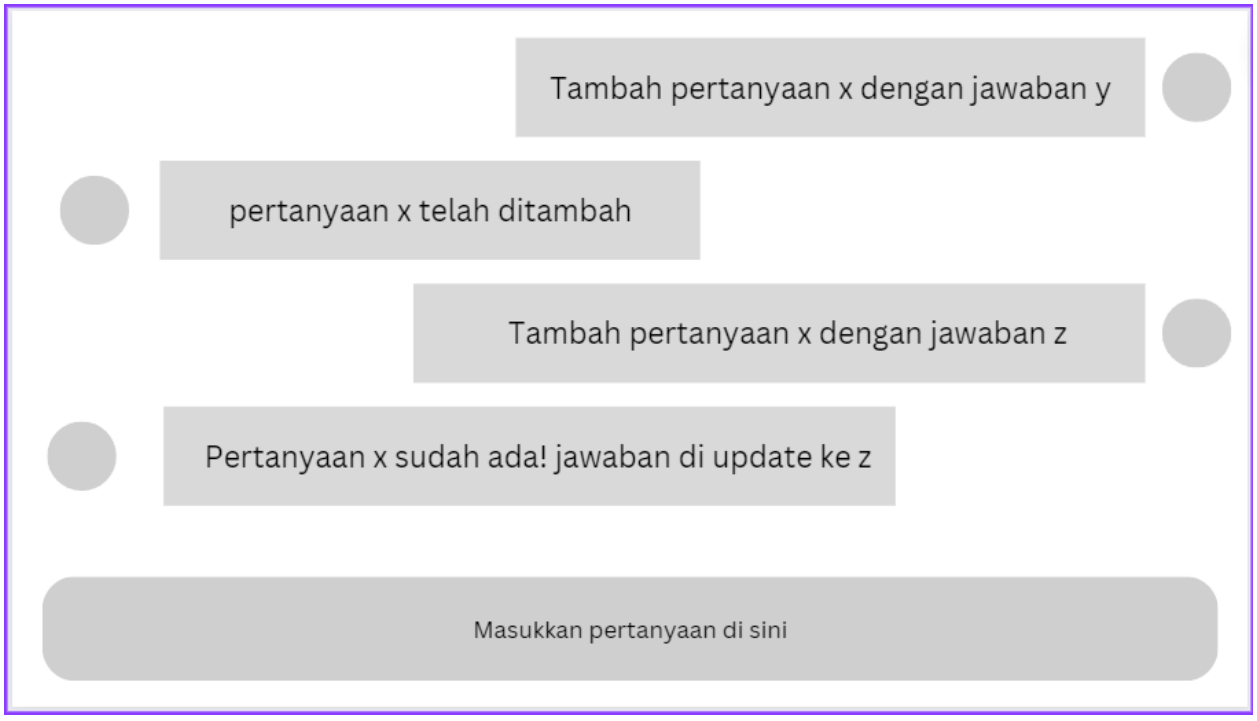
Gambar 2. Ilustrasi Fitur Pertanyaan teks kasus exact

Gambar 3. Ilustrasi Fitur Pertanyaan teks kasus tidak exact

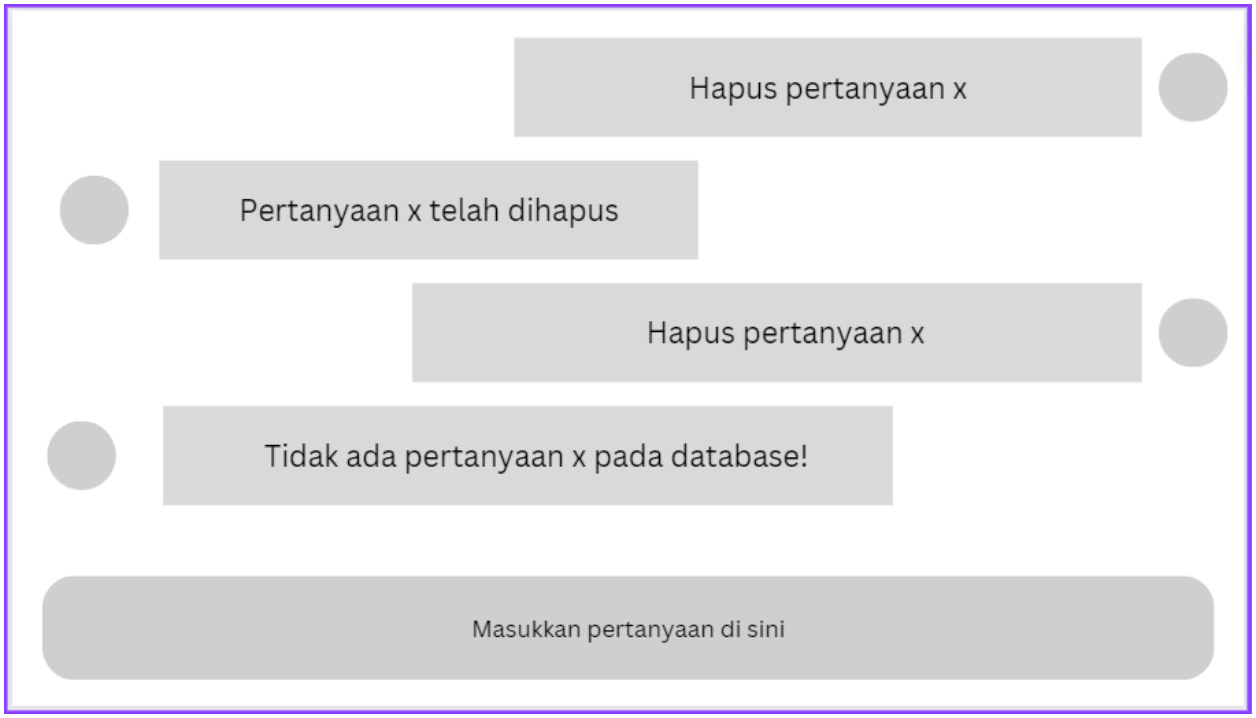


Gambar 4. Ilustrasi Fitur Kalkulator

Gambar 5. Ilustrasi Fitur Tanggal

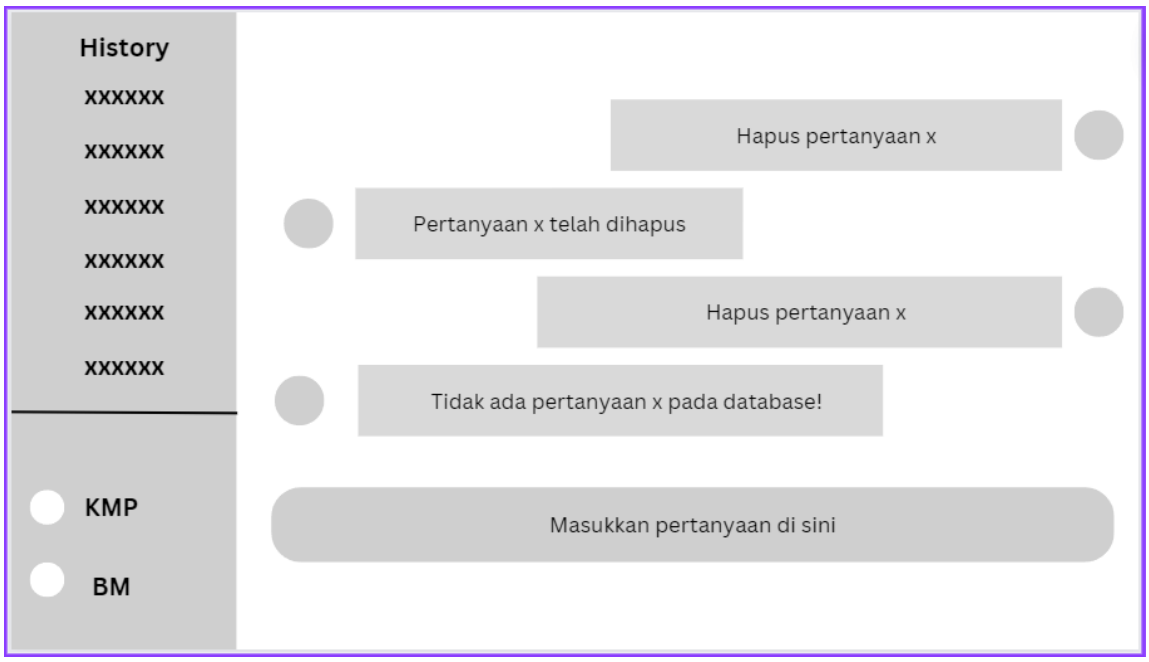


Gambar 6. Ilustrasi Fitur Tambah Pertanyaan



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

BAB 2
LANDASAN TEORI

2.1. KMP (Knuth-Morris-Pratt)

.Algoritma Knuth-Morris-Pratt merupakan algoritma pencarian pola pada sebuah string. Algoritma KMP bekerja dengan mencari pola dari kiri ke kanan dengan teknik pergeseran yang lebih efisien dibanding *brute force*.

Algoritma KMP dimulai dengan pencocokan string mulai dari posisi paling kiri. Ketika terdapat ketidaksamaan, pencarian dilanjutkan dengan menggeser pola sejauh mungkin ke kanan. Pergeseran terjauh ini dapat dihitung dengan mencari *prefix* terbesar pola yang juga merupakan *proper suffix* bagian pola yang sudah diperiksa.

Secara singkat, algoritma KMP berjalan dengan urutan:

1. Preproses string pola untuk mendapat array lps
    a. Inisialisasi lps[0] ← 0, last ← 0, i ← 1 (untuk traversal)
    b. Jika pola[i] = pola[last], last ← last + 1, lps[i] ← last, i ← i + 1
       Jika pola[i] ≠ pola[last] dan last > 0, last ← pola[last-1]
       Jika bukan keduanya, lps[i] ← last, i ← i + 1
    c. Ulangi langkah (b) hingga i = panjang pola
2. Cari pola pada string txt dengan menggunakan array lps
    a. Inisialisasi idx_txt ← 0, idx_pola ← 0
    b. Jika txt[idx_txt] = pola[idx_pola], increment idx_txt dan idx_pola
       Jika txt[idx_txt] ≠ pola[idx_pola], idx_pola ← lps[idx_pola-1]
    c. Ulangi langkah () - () hingga idx_txt = panjang txt atau idx_pola = panjang pola (ditemukan)

Algoritma KMP dapat berjalan dalam kompleksitas waktu worst case = O(n), dengan average case sama dengan worst case. Pembacaan teks dengan KMP tidak memerlukan langkah mundur, sehingga waktu untuk I/O minimal.

2.2. BM (Boyer-Moore)

.Algoritma Boyer-Moore merupakan algoritma pencarian pola pada string dengan teknik *looking-glass*. Algoritma ini mencari pola pada sebuah teks dimulai dengan cek bagian akhir pola, lalu bergerak ke depan. Ketika terjadi *mismatch* pola[j] != txt[i], ada tiga kasus yang mungkin:

1. Pola mengandung txt[i], geser pola ke kanan hingga txt[i] berimpit dengan kemunculan terakhir txt[i] pada pola.
2. Jika pada kasus (1) tidak dapat bergeser ke kanan, geser sejumlah 1 karakter
3. Selain kasus 1 dan 2, geser pola ke kanan hingga txt[i+1] berimpit dengan pola[0]

Algoritma ini berjalan dengan kompleksitas waktu worst case O(nm). Meskipun kompleksitas waktu worst case sama dengan brute force, algoritma Boyer-Moore jauh lebih cepat dibanding brute force untuk mencari pola pada kasus ukuran alfabet besar dan teks cukup acak, contohnya pengecekan bahasa inggris.

2.3. Regex (Regular Expression)

Regular Expression merupakan sekumpulan karakter yang digunakan untuk mencari pola *non-exact* dari suatu string. Pada pengembangan aplikasi, Regex dapat digunakan untuk ekstraksi nilai penting dari sebuah string dan untuk validasi input. Pola umum pada regular expression meliputi *brackets* ([], menandakan disjungsi), *hyphen*(-, menandakan range pada disjungsi), *caret*(^, menandakan negasi), dan titik(., menandakan karakter bebas).

2.4. Aplikasi Web yang Dibangun

Aplikasi yang dibuat berbasis web yang terdiri dengan *front-end* dan *back-end*. Implementasi *front-end* menggunakan *framework* NextJs dengan bahasa typescript dan didesign dengan menggunakan *framework* Tailwind. Untuk *back-end* juga menggunakan *framework* NextJs, lalu untuk database distore menggunakan Prisma. Pada *back-end* tersedia algoritma *pattern matching* dengan algoritma Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM). Untuk pemrosesan string yang diterima menggunaka Regular Expression.

BAB 3
ANALISIS PEMECAHAN MASALAH

3.1. Langkah-langkah Penyelesaian Masalah Setiap Fitur

Secara umum, teks pertanyaan dari pengguna akan diklasifikasi ke 5 kategori, yaitu pertanyaan teks, kalkulator, tanggal, tambah pertanyaan dan jawaban ke database, dan hapus jawaban dari database. Klasifikasi memanfaatkan regex ditambah pengecekan manual untuk fitur kalkulator (karena regex tidak mampu validasi tanda kurung)

Jika terdapat *overlap,* fitur yang akan diprioritaskan adalah menambah pertanyaan dan jawaban ke database, lalu menghapus jawaban dari database, tanggal, kalkulator, lalu pertanyaan teks.

3.1.1. Fitur Pertanyaan Teks

Pertanyaan berupa teks dari pengguna akan dicocokkan dengan pertanyaan dari database dengan *exact match*. Jika terdapat tepat satu pertanyaan dari database yang cocok dengan pertanyaan pengguna, jawaban pada database akan ditayangkan. Jika tidak, akan dicari pertanyaan dari database semirip mungkin dengan kemiripan setidaknya 0.9 berdasar *levenshtein distance*. Jika tidak ada yang memenuhi, akan dikembalikan maksimal 3 pertanyaan paling mirip dari database untuk dipilih pengguna.

Metode exact match dapat dipilih antara Knuth-Morris-Pratt atau Boyer-Moore, meskipun perbedaan kedua metode hanya terdapat pada pemrosesan kata (pengguna tidak dapat mengetahui perbedaannya). Untuk mencari *similarity*, digunakan rumus

$$similarity = \frac{len - levDist}{len}$$

Dengan len = panjang string terpanjang, levDist = *levenshtein distance*.

3.1.2 Fitur Kalkulator

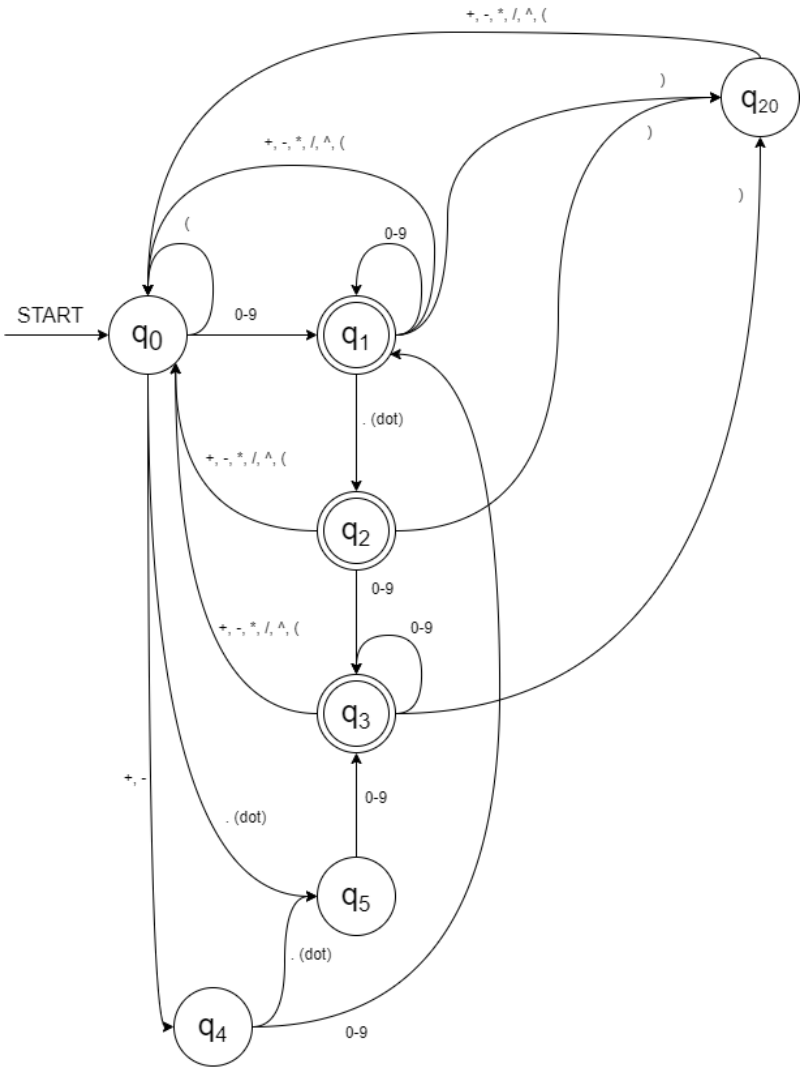
Secara umum, cara kerja fitur kalkulator adalah:

1. Menerima masukan notasi infix dalam bentuk string
2. Parse string menjadi bagian-bagian atomik (angka dan operator)
3. Mengubah hasil parse dalam notasi infix menjadi postfix
4. Menghitung hasil dari notasi postfix

Parse string notasi infix dilakukan dengan bantuan DFA (Deterministic Finite Automata) untuk memisahkan angka (dengan koma) dan operator. Token yang dihasilkan kemudian diubah menjadi bentuk postfix dengan algoritma Shunting Yard. Berikut adalah konfigurasi yang digunakan pada kalkulator.

Tabel 3.1. Operasi Matematika

Operasi	Simbol	Precedence	Association
Penjumlahan	+	1	Left
Pengurangan	-	1	Left
Perkalian	*	2	Left
Pembagian	/	2	Left
Perpangkatan	^	3	Right



Gambar 3.1 Diagram transisi parser kalkulator

3.1.3 Fitur Tanggal

Fitur tanggal bekerja dengan menerima hasil parse regex berupa hari, bulan, dan tanggal, lalu menggunakan fungsi getDay() dari kelas Date.

3.1.4 Fitur Tambah Pertanyaan dan Jawaban ke Database

Fungsi yang melayani fitur ini bekerja dengan menerima hasil parse regex berupa pertanyaan dan jawaban, lalu memasukkan data ke database.

3.1.5 Fitur Hapus Pertanyaan dari Database

Fungsi ini bekerja dengan menerima hasil parse regex berupa pertanyaan yang akan dihapus, lalu pertanyaan tersebut akan dihapus dari database.

3.2. Fitur Fungsional dan Arsitektur Aplikasi Web yang Dibangun

Front-end NextJs dijalankan dari file page.tsx. Di dalam file tersebut mengandung 2 komponen utama yaitu SideBar dan MainPage. Seluruh komponen pada aplikasi ini disimpan pada folder *components*. Pada komponen SideBar memiliki beberapa tombol yaitu NewChat, untuk menambahkan *chat session* baru; ClearAll, untuk menghapus seluruh *history chat;* Radio button, untuk memilih algoritma *pattern matching* apa yang ingin digunakan; serta HistoryList, yang mengandung seluruh *history chat* yang disimpan pada database.

MainPage merupakan halaman utama pada aplikasi ini yang memiliki beberapa komponen. Terdapat ChatBubble yang berfungsi untuk menampilkan pertanyaan dari masukan pengguna serta jawaban dari program. Lalu, terdapat textbox dan Send button untuk menerima masukan dari pengguna. Setiap kali pengguna membuka aplikasi web, akan pasti dialokasikan sebagai *chat session* baru.

3.2.1. *New Chat*

Fitur NewChat terletak pada file NewChat.tsx di folder *button* yang memiliki parent folder *components*. Fitur ini membuat *chat session* dengan id baru dan membuat data baru.

3.2.2. *Clear All*

Fitur ClearAll terletak pada file Clear.tsx di folder *button* yang memiliki parent folder *components*. Fitur ini menghapus seluruh *chat session* pada *history list* dan database.

3.2.3. *Delete*

Fitur Delete terletak pada file ChatHistory.tsx di folder *button* yang memiliki parent folder *components*. Fitur ini menghapus *chat session* yang dipilih pada *history list* dan database.

3.2.4. *History List*

Fitur HistoryList terletak pada file HistoryList.tsx di folder *components*. Fitur ini menampilkan seluruh *chat session* pada database. Pengguna dapat memilih *chat session* yang tersedia pada list, dan aplikasi akan menampilkan seluruh chat pada *session* tersebut.

3.2.5. *Chatting*

Fitur Chatting merupakan fitur utama dari aplikasi ini. Fitur ini terletak pada file *MainPage.tsx* pada folder *components*. Pertama, pengguna memasukkan pertanyaan pada textbox, lalu klik *enter* untuk menampilkan jawaban. Pengguna dapat memilih algoritma apa yang ingin digunakan dalam proses *pattern matching*.

BAB 4
IMPLEMENTASI DAN PENGUJIAN

4.1. Spesifikasi Teknis Program

4.1.1. Fungsi classifyInputMessage

Menerima string berupa message dari user
Mengembalikan respon berupa tipe pertanyaan dan jawaban

Menggunakan regex dan exception untuk menentukan tipe pertanyaan

4.1.2. Fungsi getMostSimilarString

Menerima string input dan array string pertanyaan dari database
Mengembalikan array string berukuran 1-3 sesuai kondisi input

Menggunakan fungsi exact matching (kmp atau bm, sesuai pilihan). Jika fungsi tersebut mengembalikan tepat 1 match, return hasil match tersebut. Jika tidak, kembalikan pertanyaan yang paling mirip dan kemiripan minimal 90%. Jika tidak ada, kembalikan maksimal 3 pertanyaan yang paling mirip.

4.1.3. Fungsi kmp

Menerima string pattern dan array string teks yang akan diperiksa
Mengembalikan array string berisi teks yang memiliki substring pattern

4.1.4. Fungsi bm

Menerima string pattern dan array string teks yang akan diperiksa
Mengembalikan array string berisi teks yang memiliki substring pattern

4.1.5. Fungsi levenshteinSimilarity

Menerima dua string
Mengembalikan nilai antara 0-1; 0 berarti kedua string sangat berbeda, 1 berarti kedua string sama.

Menggunakan 2d dynamic programming untuk mendapat distance, lalu bobotnya dicari dengan membagi distance dengan panjang string terpanjang.

4.1.6. Fungsi mathParser

Menerima string berisi ekspresi matematika infix, contoh: 3+(2 * 5).
Mengembalikan sebuah number berupa hasil evaluasi ekspresi matematika

Algoritma yang digunakan terdapat pada 3.1.2

4.1.7. Fungsi getDayFromDate

Menerima tanggal, bulan, dan tahun
Mengembalikan hari pada tanggal, bulan, dan tahun yang bersangkutan

4.1.8. Fungsi API Database

Menerima request yang dipanggil dengan metode fetch
Mengembalikan response

4.2. Tata Cara Penggunaan Program

Terdapat dua cara penggunaan program yang telah digunakan. Pertama adalah dengan cara menjalankan aplikasi web di *localhost*. Caranya:

1) Masuk ke folder 'gpt-1' dengan *command* **cd gpt-1**
2) Masukkan *command* **npm install** untuk menginstall *package* dari framework
3) Jalankan program dengan *command* **npm run dev**

Cara kedua adalah dengan membuka halaman website yang sudah dideploy menggunakan vercel melalui link berikut: https://gpt-1-nu.vercel.app/

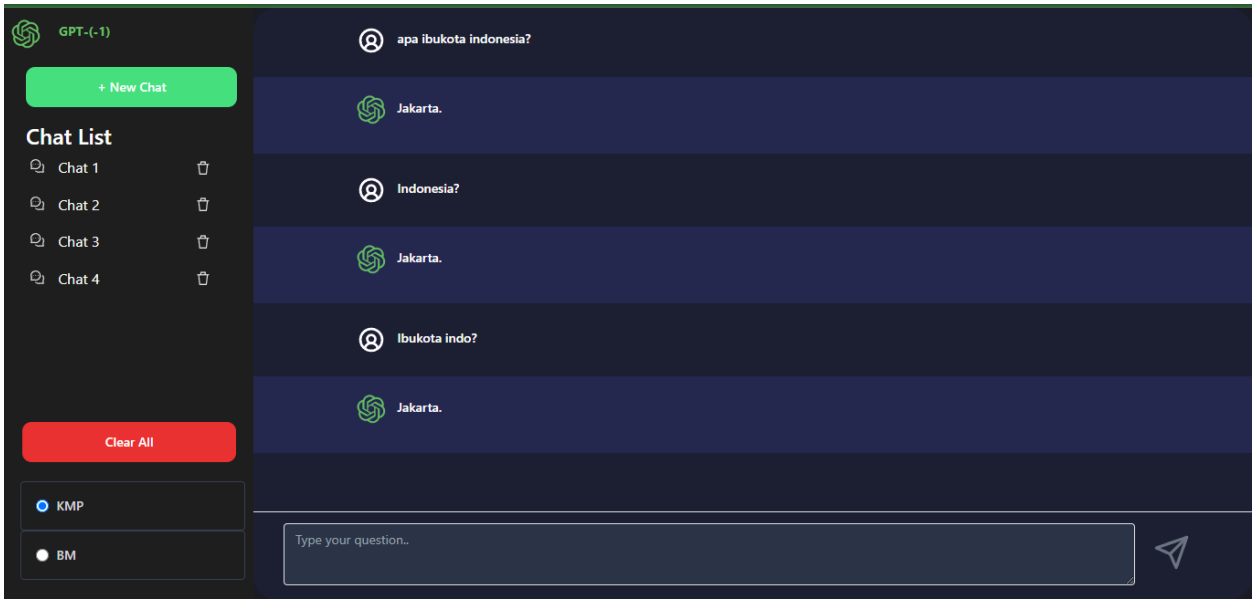
Cara penggunaan aplikasi web adalah sebagai berikut:

1) Pengguna membuka aplikasi dan akan masuk ke *chat session* baru
2) Pengguna dapat memasukkan pertanyaan pada *textbox* berdasarkan fitur yang tersedia
3) Ketika pengguna sudah selesai mengetik pertanyaan, pengguna dapat menekan tombol *send* yang terletak disebelah *textbox*
4) Pengguna dapat menambahkan sesi baru dengan tombol *New Chat* pada *side bar*
5) Pengguna juga dapat memilih *chat session* yang tersedia pada *side bar*
6) Pengguna dapat menghapus seluruh atau salah satu *chat session* dengan tombol *clear all* atau *icon* bergambar *trash bin*

4.3. Hasil Pengujian

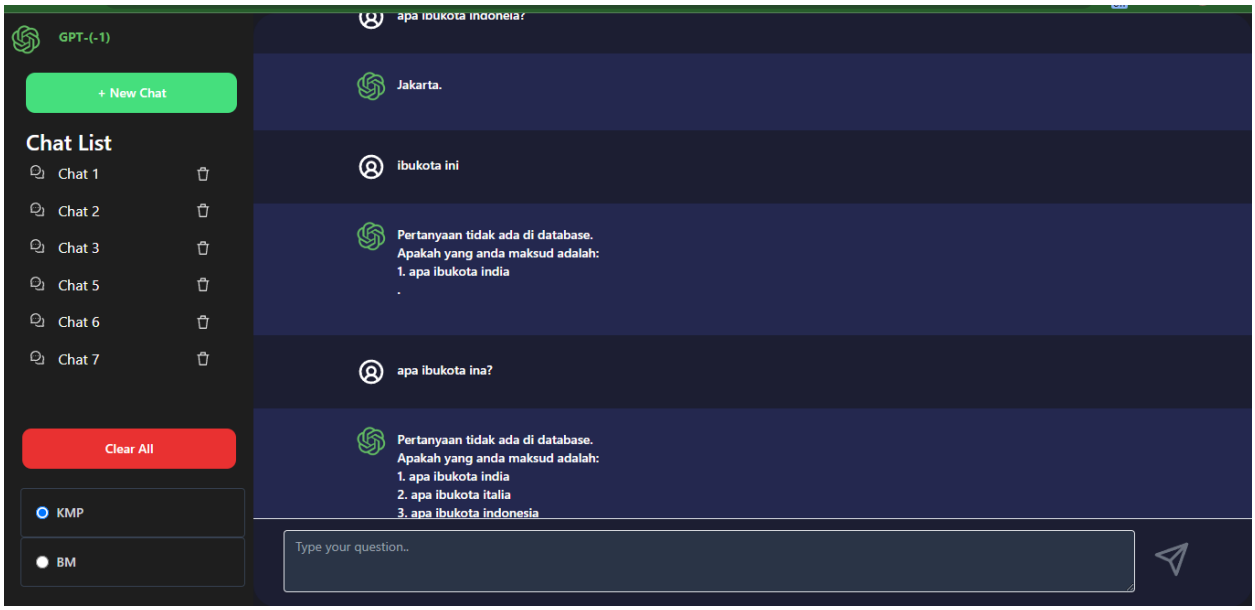
4.3.1. Pengujian 1

Exact matching (total & parsial)



4.3.2. Pengujian 2

String similarity (total)

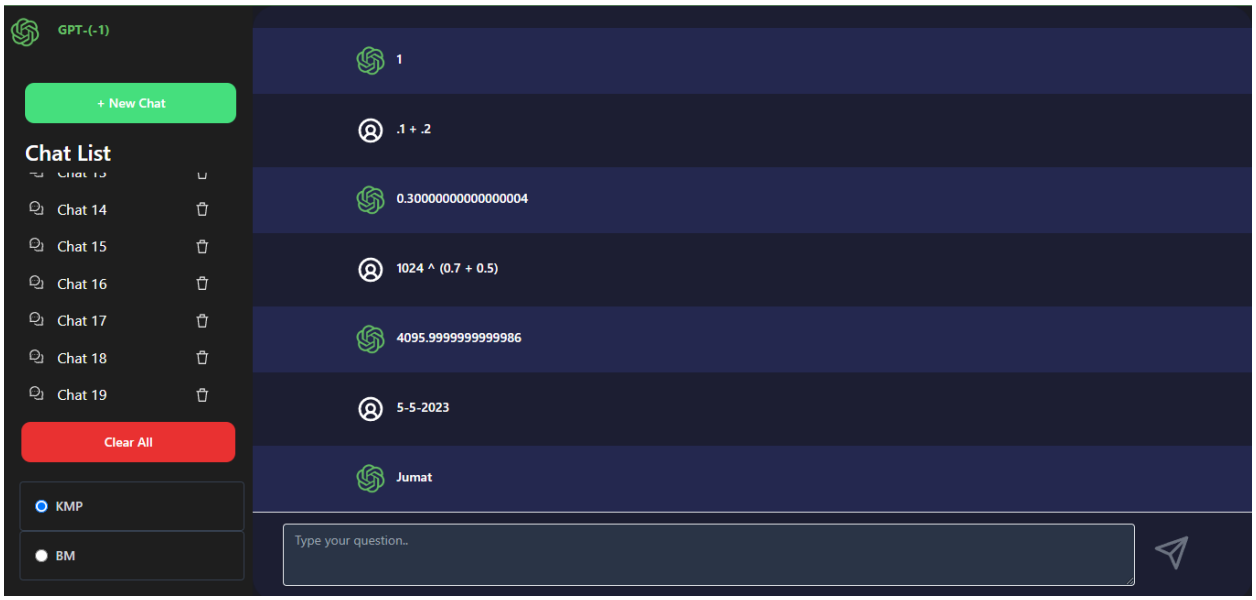


4.3.3 Pengujian 3

Penambahan dan pengurangan pertanyaan

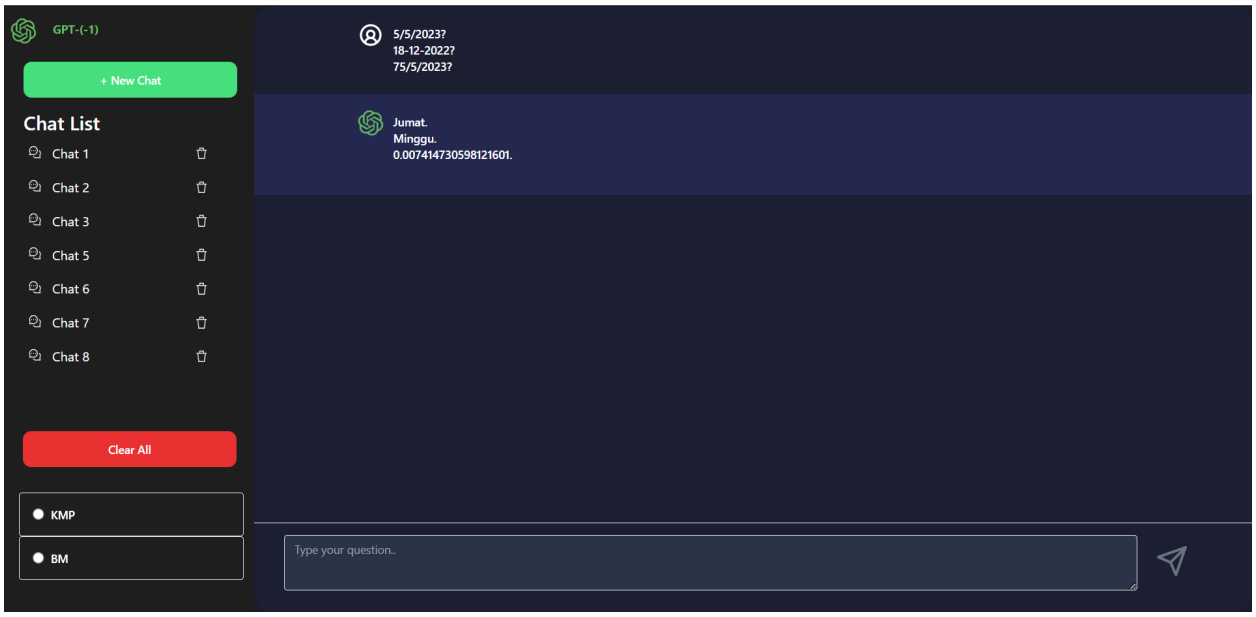
4.3.4 Pengujian 4

Kalkulator



4.3.5 Pengujian 5

Tanggal



4.4. Analisis Hasil Pengujian

4.4.1 Analisis Hasil Pengujian 1

Jawaban yang dikembalikan seluruhnya adalah jawaban pertanyaan ‘apa ibukota indonesia’, karena seluruh pertanyaan merupakan substring dari ‘apa ibukota indonesia’ dan tidak ada pertanyaan yang lebih mirip.

4.4.2 Analisis Hasil Pengujian 2

Pertanyaan ‘apa ibukota indoneia’ mengembalikan jawaban Jakarta, karena kemiripan dengan ‘apa ibukota indonesia’ lebih dari 90%.

Pertanyaan ‘ibukota ini’ mengembalikan pilihan ‘apa ibukota india’ karena tidak ada pertanyaan di database dengan kemiripan lebih dari 90%, tetapi pertanyaan ‘apa ibukota india’ mirip setidaknya 60%

Pertanyaan ‘apa ibukota ina’ mengembalikan pilihan ‘apa ibukota india’, ‘apa ibukota italia’, dan ‘apa ibukota indonesia’. Artinya, ketiga pilihan tersebut punya kemiripan antara 60%-90% dan urutannya mulai dari yang tingkat kemiripannya paling tinggi.

4.4.3 Analisis Hasil Pengujian 3

Penambahan dan pengurangan pertanyaan berfungsi seperti yang diinginkan

4.4.4 Analisis Hasil Pengujian 4

(4-2)(4+2)(4-(-2)) bernilai 72 karena terdapat implicit multiplication

0^0 bernilai 1 karena mengikuti operasi perpangkatan dari typescript.

.1 + .2 tidak tepat bernilai .3 karena rounding error dan tidak terdapat pembulatan

5-5-2023 tidak dikenali sebagai fungsi kalkulator, tetapi dapat dipaksa dengan menambah tanda kurung untuk melingkupi persamaan

4.4.5 Analisis Hasil Pengujian 5

Format DD-MM-YYYY dan DD/MM/YYYY keduanya dapat digunakan untuk fitur tanggal. 75/5/2023 mengembalikan nilai 0.007414730598121601 karena bukan merupakan tanggal yang valid sehingga dianggap sebagai operasi matematika.

BAB 5
PENUTUP

5.1. Kesimpulan

Dalam tugas besar IF2211 Strategi Algoritma ini, kami berhasil membuat aplikasi web. Dari pengerjaan tugas ini, kami mendapatkan beberapa kesimpulan, yaitu

- Chatbot dapat dibuat dengan algoritma sederhana yang memanfaatkan pattern matching (exact match dan regex match)
- Pemisahan front end dan back end adalah best practice dalam pengembangan web based application.
- Tidak ada perbedaan signifikan terkait performa pada pemilihan algoritma Knuth-Morris-Pratt atau Boyer-Moore, karena keduanya beroperasi dalam kompleksitas rata-rata yang linear untuk teks bahasa Indonesia.

5.2. Saran

Beberapa saran yang kami dapatkan dari pengerjaan pengembangan aplikasi ini adalah sebagai berikut:

1. Database dibuat berbeda setiap user agar setiap user dapat melihat history chat masing-masing
2. Pattern matching sebaiknya diganti dengan *machine learning*

5.3. Refleksi

Gambar 5.3 Refleksi Diri

Dalam pengerjaan tugas ini sebaiknya tidak dikerjakan mepet deadline agar ga burnout.

DAFTAR PUSTAKA

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2022-2023/Tubes3-Stima-2023.pdf

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

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

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

LAMPIRAN

LINK REPOSITORY

Link repository GitHub : https://github.com/AJason36/Tubes3_13521100

Link website hasil deploy : https://gpt-1-nu.vercel.app/

Link Youtube : https://youtu.be/1G4_VxmXKYE

PEMBAGIAN TUGAS

NIM	Nama	Tugas
13521100	Alexander Jason	Front-end
13521114	Farhan Nabil Suryono	Database, connect Front-end to Back-end, deploy
13521137	Michael Utama	Back-end