LAPORAN TUGAS BESAR 3 IF2211 STRATEGI ALGORITMA

Semester II Tahun 2022/2023

"Penerapan String Matching dan Regular Expression dalam Pembuatan ChatGPT Sederhana"

LAPORAN TUGAS BESAR 1 IF2123 ALJABAR LINIER DAN GEOMETRI

SISTEM PERSAMAAN LINIER, DETERMINAN, DAN APLIKASINYA

Disusun oleh:

Kelompok 16 – "ChatBjorka"

Rizky Abdillah Rasyid – 13521109

Saddam Annais Shaquille – 13521121

Hanif Muhammad Zhafran – 13521157

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

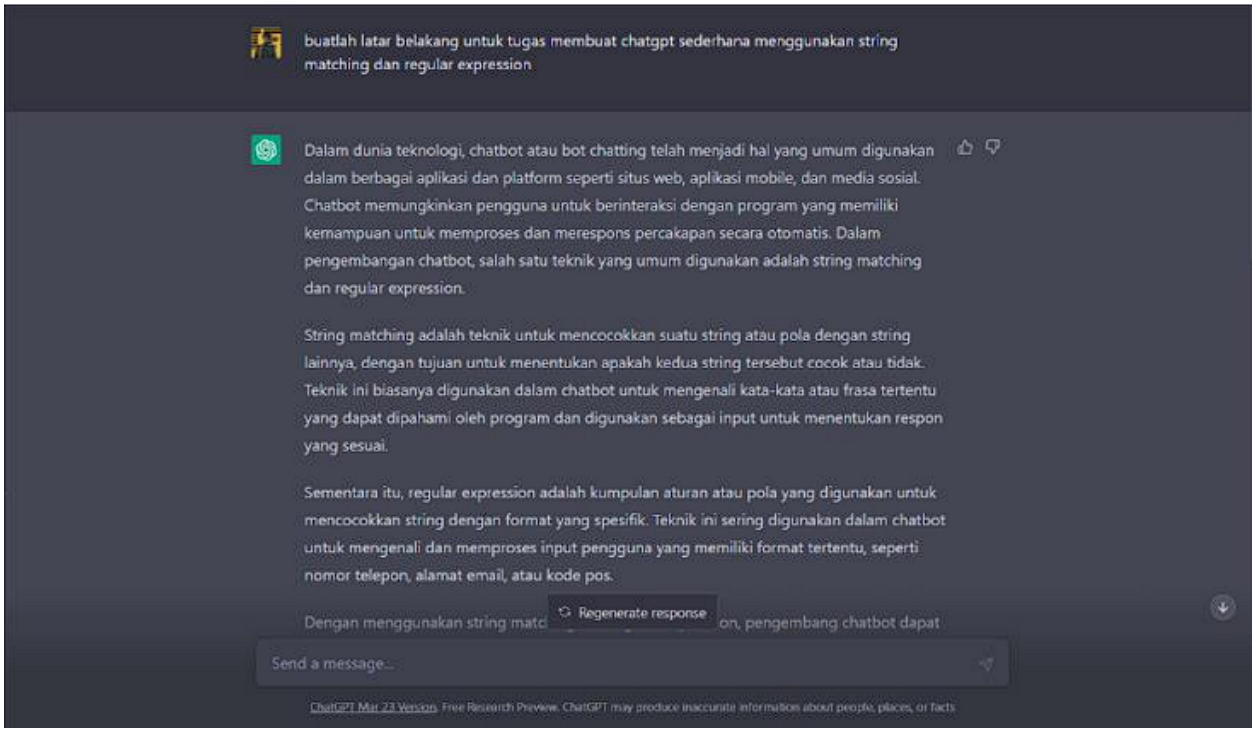
INSTITUT TEKNOLOGI BANDUNG

2023

DAFTAR PUSTAKA

BAB 1
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 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.

BAB 2
LANDASAN TEORI

2.1. Algoritma Knuth-Morris-Pratt (KMP)

Algoritma Knuth-Morris-Pratt (KMP) adalah algoritma untuk mencari kecocokan antara sebuah pola (pattern) dan sebuah teks. Algoritma ini bekerja dengan cara memanfaatkan informasi tentang kecocokan yang telah dilakukan sebelumnya untuk menghindari pencocokan ulang pada posisi yang sama dalam teks. Cara kerja algoritma KMP adalah dengan membangun sebuah border function untuk pola yang akan dicari. Border function ini menentukan panjang maksimal dari sebuah proper border (prefix yang sama dengan suffix) dari sebuah sub-pola dari pola. Border function dapat dibangun dengan menggunakan teknik dynamic programming.

Ketika mencocokkan pola dengan teks, algoritma KMP memeriksa karakter-karakter dari pola dan teks secara berurutan dari kiri ke kanan. Jika terdapat ketidakcocokan pada karakter tertentu, algoritma akan memindahkan pola sejauh border function dari sub-pola terakhir yang cocok dengan teks. Algoritma KMP menggeser pola ke posisi yang tepat, sehingga tidak perlu mencocokkan kembali karakter-karakter yang telah cocok sebelumnya. Dengan demikian, algoritma KMP lebih efisien dibandingkan brute force dalam melakukan pencocokan string.

Contoh, misalnya jika pola "aba" ingin dicari dalam teks "abacabadabacaba", maka border function dibangun untuk pola "aba" yang memberikan hasil "0 0 1". Selanjutnya, pola dicocokkan dengan teks dari kiri ke kanan. Pada posisi pertama, "a" pada pola cocok dengan "a" pada teks. Pada posisi kedua, "b" pada pola juga cocok dengan "b" pada teks. Namun, pada posisi ketiga, "a" pada pola tidak cocok dengan "c" pada teks. Karena sub-pola terakhir dari pola yang cocok dengan teks adalah "ab", pola dapat dipindahkan sejauh 1 karakter menggunakan border function, sehingga posisi pola berubah menjadi kedua. Dengan demikian, karakter-karakter "a" dan "b" yang telah cocok sebelumnya tidak perlu dicocokkan kembali. Pencocokan kemudian dilanjutkan dari posisi kedua dan seterusnya hingga seluruh kemunculan pola "aba" dalam teks ditemukan.

2.2. Algoritma Boyer-Moore (BM)

Algoritma Boyer-Moore (BM) adalah salah satu algoritma untuk mencari kecocokan antara sebuah pola (pattern) dan sebuah teks. Algoritma ini bekerja dengan cara memanfaatkan informasi tentang kecocokan yang telah dilakukan sebelumnya untuk menghindari pencocokan ulang pada posisi yang sama dalam teks. Cara kerja algoritma BM adalah dengan memanfaatkan dua heuristik, yaitu heuristik karakter terakhir (last character heuristic) dan heuristik good suffix. Heuristik karakter terakhir memanfaatkan informasi tentang karakter terakhir dari pola yang dicari untuk menggeser pola ke posisi yang tepat pada teks. Heuristik good suffix menggunakan informasi tentang proper suffix dari sub-pola yang cocok dengan teks untuk menggeser pola ke posisi yang tepat.

Algoritma BM melakukan pencocokan pola dengan teks secara berurutan dari kanan ke kiri. Ketika terdapat ketidakcocokan pada karakter tertentu, algoritma akan memindahkan pola

ke posisi yang tepat menggunakan heuristik karakter terakhir dan heuristik good suffix. Algoritma BM juga memanfaatkan tabel karakter (character table) dan tabel good suffix (good suffix table) untuk mempercepat pencocokan string. Tabel karakter berisi informasi tentang posisi karakter terakhir dari setiap karakter yang muncul dalam pola, sedangkan tabel good suffix berisi informasi tentang panjang maksimal dari proper suffix dari sub-pola yang cocok dengan teks.

2.3. Regular Expression (Regex)

Regex atau Regular Expression adalah sebuah notasi untuk merepresentasikan sebuah pola atau kumpulan pola yang digunakan untuk mencocokkan dan memanipulasi teks. Dalam konteks pemrograman, Regex biasanya dimasukkan ke dalam sebuah string dan digunakan bersama dengan fungsi atau metode untuk mencari, mengganti atau memanipulasi teks yang sesuai dengan pola yang telah ditentukan.

Regex terdiri dari karakter-karakter literal dan karakter-karakter metakarakter. Karakter-karakter literal merepresentasikan karakter-karakter yang harus cocok secara harfiah dengan teks yang ingin dicocokkan, sedangkan karakter-karakter metakarakter merepresentasikan karakter-karakter yang memiliki arti khusus dan digunakan untuk merepresentasikan jenis karakter tertentu seperti angka, huruf, spasi atau karakter-karakter lainnya.

Beberapa karakter metakarakter yang sering digunakan dalam Regex antara lain:

'^' untuk merepresentasikan awal sebuah string
'$' untuk merepresentasikan akhir sebuah string
'.' untuk merepresentasikan satu karakter
'*' untuk merepresentasikan nol atau lebih karakter
'+' untuk merepresentasikan satu atau lebih karakter
'?' untuk merepresentasikan nol atau satu karakter
'[]' untuk merepresentasikan kumpulan karakter yang diizinkan
'|' untuk merepresentasikan alternatif antara dua pola
'()' untuk merepresentasikan grup dari pola

Sebagai contoh, sebuah pola Regex seperti '^Hello' akan mencocokkan setiap string yang dimulai dengan kata "Hello", sedangkan pola Regex seperti '[0-9]+' akan mencocokkan setiap angka yang muncul dalam sebuah string. Regex berguna dalam hal seperti validasi input pengguna pada formulir web, pencarian teks dalam sebuah file, atau penggantian teks yang cocok dengan sebuah pola dengan teks baru.

2.4. Aplikasi Web

Aplikasi web adalah suatu program komputer yang dirancang untuk berjalan pada platform web, seperti browser web, dan biasanya diakses melalui internet menggunakan protokol HTTP

atau HTTPS. Aplikasi web dapat berupa website sederhana seperti blog atau situs berita, atau dapat berupa aplikasi yang lebih kompleks seperti e-commerce, aplikasi perbankan online, atau aplikasi media sosial.

Aplikasi web umumnya terdiri dari kombinasi tiga komponen utama, yaitu:

1. Client-side script: kode program yang dijalankan pada browser pengguna, biasanya ditulis dalam bahasa pemrograman seperti JavaScript, dan digunakan untuk mengatur interaksi antara pengguna dan aplikasi.
2. Server-side script: kode program yang dijalankan pada server, biasanya ditulis dalam bahasa pemrograman seperti PHP, Python, atau Ruby, dan digunakan untuk menghasilkan halaman web yang akan ditampilkan pada browser pengguna.
3. Database: tempat penyimpanan data yang digunakan oleh aplikasi web, biasanya terletak pada server, dan dapat diakses melalui server-side script.

Aplikasi web memiliki beberapa keuntungan dibandingkan dengan aplikasi desktop tradisional, di antaranya:

1. Aplikasi web dapat diakses dari mana saja, selama terhubung dengan internet.
2. Aplikasi web dapat diakses dari berbagai perangkat, seperti komputer desktop, laptop, tablet, atau smartphone.
3. Aplikasi web tidak memerlukan instalasi pada perangkat pengguna, sehingga lebih mudah dalam hal distribusi dan pemeliharaan.

Namun, aplikasi web juga memiliki beberapa kelemahan, di antaranya:

1. Ketergantungan pada koneksi internet, sehingga jika koneksi terputus, pengguna tidak dapat mengakses aplikasi.
2. Keamanan yang lebih rentan terhadap serangan online seperti hacking atau phishing.
3. Keterbatasan dalam hal akses ke fitur-fitur perangkat keras pada perangkat pengguna, seperti kamera atau mikrofon.

Secara umum, pengembangan aplikasi web dibagi menjadi dua, yaitu Front-end Development yang berfokus pada Client-Side Script dan Back-end Development yang berfokus pada Server-Side Script dan Database.

2.4.1 Front-end Development

Front-end development adalah proses pengembangan aplikasi web yang fokus pada bagian tampilan atau user interface (UI) yang dilihat dan diakses oleh pengguna. Front-end development melibatkan penggunaan bahasa markup seperti HTML, styling menggunakan CSS, dan interaksi menggunakan JavaScript untuk membuat tampilan yang menarik dan responsif.

Front-end development meliputi desain tampilan, interaksi pengguna, dan pengembangan aplikasi web yang responsif dan cepat. Front-end developer bertanggung jawab untuk mengembangkan bagian depan dari aplikasi web, termasuk antarmuka pengguna, interaksi, dan pengalaman pengguna yang baik.

React adalah salah satu framework front-end yang sangat populer di kalangan pengembang web. React dikembangkan oleh Facebook dan digunakan untuk membangun aplikasi web modern dengan tampilan yang dinamis dan responsif. React menggunakan konsep komponen yang memungkinkan pengembang untuk memecah tampilan menjadi bagian-bagian yang lebih kecil dan mudah dikelola.

Beberapa konsep pengembangan front-end menggunakan React antara lain:

1. **Komponen**: Konsep dasar dalam pengembangan aplikasi web dengan React adalah komponen. Komponen adalah bagian-bagian kecil dari aplikasi web yang dapat digabungkan untuk membuat tampilan yang lebih kompleks. Komponen dapat dianalogikan seperti Lego yang dapat digabungkan untuk membuat struktur yang lebih kompleks.
2. **Virtual DOM**: React menggunakan Virtual DOM (Document Object Model) untuk mempercepat proses rendering tampilan. Virtual DOM adalah representasi JavaScript dari DOM yang sebenarnya. Ketika suatu komponen diubah, React tidak langsung memperbarui DOM sebenarnya, tetapi membuat Virtual DOM baru dan membandingkannya dengan Virtual DOM sebelumnya untuk menemukan perbedaan dan melakukan perubahan yang diperlukan pada DOM yang sebenarnya.
3. **State dan Props**: React menggunakan konsep State dan Props untuk mengelola data dalam aplikasi. State adalah data yang dapat berubah dalam komponen, sedangkan Props adalah data yang diteruskan dari komponen induk ke komponen anak. Dengan mengelola State dan Props, aplikasi yang responsif dan mudah diatur dapat dibuat.
4. **JSX**: React menggunakan JSX (JavaScript XML) untuk menggabungkan kode JavaScript dengan tag HTML. JSX memungkinkan pengembang untuk menuliskan kode dengan cara yang lebih dekat dengan HTML, membuat kode lebih mudah dibaca dan dimengerti.

2.4.2 Back-end Development

Back-end development adalah proses pengembangan aplikasi web yang fokus pada bagian yang tidak terlihat oleh pengguna, seperti server, database, dan logika bisnis. Back-end development melibatkan penggunaan bahasa pemrograman seperti Golang, Python, Ruby, atau PHP, dan penggunaan database untuk menyimpan dan mengelola data. Konsep dasar dalam back-end development meliputi pengembangan API, penggunaan framework, pengaturan server, dan pengembangan logika bisnis.

Golang adalah bahasa pemrograman back-end yang relatif baru. Golang dikembangkan oleh Google dan dirancang untuk memudahkan pengembangan aplikasi yang efisien, skalabel, dan mudah dikelola.

Beberapa konsep pengembangan back-end menggunakan Golang antara lain:

1. **Goroutines**: Goroutines adalah konsep paralelisme dalam Golang yang memungkinkan pengembang untuk menjalankan beberapa tugas secara bersamaan tanpa mengganggu kinerja aplikasi.
2. **Channels**: Channels adalah konsep komunikasi yang digunakan dalam Golang untuk menghubungkan Goroutines. Channels memungkinkan Goroutines untuk berkomunikasi dengan aman dan efisien, menghindari masalah seperti race condition atau deadlock.
3. **Struct**: Struct adalah konsep dalam Golang yang mirip dengan kelas dalam bahasa pemrograman lainnya. Struct digunakan untuk mengelompokkan data dan fungsi yang terkait dalam satu objek.
4. **Error Handling**: Error handling adalah konsep dalam Golang untuk mengatasi kesalahan yang terjadi dalam aplikasi. Golang memiliki sistem error handling yang sederhana dan efektif, dengan penggunaan return value untuk mengindikasikan apakah fungsi berhasil atau gagal.

Golang memiliki banyak framework yang bisa digunakan untuk mengembangkan back-end seperti Gin-Gonic, Chi, Echo, Mux-Gorilla dan lainnya. Penggunaan framework ini untuk membantu dalam membuat routing untuk membuat API (Application Program Interface) dan menghubungkan database dengan menggunakan driver database atau ORM (Object Relational Mapping) dari library yang ada di community Go.

PostgreSQL adalah salah satu database relasional yang dapat digunakan dengan Golang. PostgreSQL adalah database open source untuk menyimpan dan mengelola data. PostgreSQL memiliki fitur-fitur seperti dukungan untuk transaksi, pengindeksan yang kuat, dan dukungan untuk JSON dan data semi-struktur lainnya.

.

BAB 3
ANALISIS PEMECAHAN MASALAH

3.1. Langkah Pemecahan Masalah Setiap Fitur

3.1.1 Fitur Chat History dan User ID

Program menampilkan *chat history* berdasarkan user id yang disimpan di dalam browser. Jika user id tidak ada di browser, maka akan dianggap sebagai user baru. Jika user id ada, maka akan dikirim *chat history* user id yang bersesuaian

3.1.2 Fitur Edit Chat Title

Program dapat mengedit judul *chat history*. Ketika suatu judul *chat history* diedit, suatu API akan dipanggil untuk memberitahu bahwa terjadi perubahan pada judul chat.

3.1.3 Fitur Delete Chat History

Program dapat menghapus suatu *chat history*. Ketika suatu *chat history* dihapus, suatu API akan dipanggil untuk memberitahu bahwa suatu chat telah dihapus.

3.1.4 Fitur Chat Content

Program dapat menampilkan isi dari suatu chat ketika judul chatnya ditekan. Ketika judul chat ditekan, suatu API akan dipanggil dan akan didapatkan isi dari percakapan sebelumnya.

3.1.5 Send Message

User dapat mengirimkan pesan ke dalam program dan program akan mengembalikan respon bot. Ketika user mengirim pesan, suatu API akan dipanggil dan akan didapatkan respon dari algoritma bot dan akan ditampilkan ke user.

3.1.6 Chose Algorithm

User dapat memilih algoritma yang akan digunakan untuk mengkomputasi respon dari bot. Ketika tombol untuk algoritma ditekan, program akan menyimpan algoritma apa yang sedang digunakan saat ini dan akan digunakan ketika user mengirim pesan.

3.2. Fitur Fungsional Aplikasi

3.2.1 Fitur Pertanyaan Teks

Fitur ini memungkinkan bot untuk menjawab pertanyaan yang diberikan oleh user berdasarkan pasangan pertanyaan dan jawaban yang tersimpan dalam database. Fitur ini akan ditangkap oleh bot pada kasus *else* dari klasifikasi regex. Sehingga, seluruh string yang tidak

masuk ke dalam klasifikasi fitur lainnya akan masuk ke dalam fitur pertanyaan teks. Pencocokan string pertanyaan user dan pertanyaan pada database dilakukan dengan menggunakan algoritma KMP/BM untuk memeriksa *exact match* atau tidak. Ketika kedua pertanyaan tidak ada yang *exact match*, maka digunakan Levenshtein Distance untuk mengukur kemiripan kedua string pertanyaan. Jika ditemukan pada database pertanyaan yang memiliki kemiripan >90%, maka jawaban dari pertanyaan tersebut akan langsung dikeluarkan oleh bot. Jika seluruh pertanyaan pada database memiliki kemiripan <90% tetapi masih diatas >30%, maka akan ditampilkan tiga pertanyaan dengan tingkat kemiripan paling tinggi. Jika seluruh pertanyaan pada database memiliki kemiripan <30%, maka pertanyaan tersebut dianggap terlalu acak (bot akan mengeluarkan pesan bahwa pertanyaan user tidak dapat di-*handle*)

3.2.2 Fitur Kalkulator

Fitur ini memungkinkan bot untuk menjawab hasil dari suatu ekspresi aritmatika sederhana yang memiliki operator tambah, kurang, bagi, kali, dan kurung. Fitur kalkulator memiliki urutan prioritas ke-2 dalam klasifikasi menggunakan regex. Berikut regex yang digunakan untuk klasifikasi fitur kalkulator:

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

Ketika suatu pertanyaan user terklasifikasi sebagai kalkulasi oleh regex, namun sintaks ekspresi aritmatikanya ada yang keliru, bot akan mengeluarkan pesan bahwa sintaks ekspresi aritmatikanya salah.

3.2.3 Fitur Tanggal

Fitur ini memungkinkan bot untuk memberitahu hari dari suatu tanggal dengan format penulisan YYYY/MM/DD. Fitur tanggal memiliki prioritas urutan ke-1 dalam klasifikasi menggunakan regex. Bot juga akan memvalidasi apakah tanggal yang diinputkan oleh user merupakan tanggal yang ada atau tidak. Jika tanggal tidak ada, maka bot akan mengeluarkan pesan bahwa tanggal tidak valid. Berikut regex yang digunakan untuk klasifikasi fitur tanggal:

```
^\s*(hari|hari apa)?\s*[0-9]{4}/[0-9]{2}/[0-9]{2}\s*\?*\s*$
```

3.2.4 Fitur Tambah Pertanyaan dan Jawaban ke Database

Fitur ini memungkinkan bot untuk menambahkan pertanyaan beserta jawabannya ke database. Fitur ini memiliki urutan prioritas ke-3 dalam klasifikasi menggunakan regex. Mirip dengan fitur pertanyaan teks (3.2.1), fitur tambah pertanyaan juga menggunakan algoritma KMP, BM, dan Levenshtein Distance untuk mengecek kesamaan pertanyaan user dengan database. Apabila seluruh pertanyaan pada database memiliki kemiripan <90%, maka pertanyaan dan jawaban baru dimasukkan ke dalam database. Jika pertanyaan database paling mirip memiliki kemiripan >90% atau bahkan *exact match*, jawaban dari pertanyaan tersebut akan diperbarui sesuai dengan input pengguna. Berikut regex yang digunakan untuk klasifikasi fitur tambah pertanyaan dan jawaban:

```
^\s*tambah pertanyaan (.+) dengan jawaban (.+)$
```

3.2.5 Fitur Hapus Pertanyaan dari Database

Fitur ini memungkinkan bot untuk menghapus pertanyaan beserta jawabannya ke database. Fitur ini memiliki urutan prioritas ke-4 dalam klasifikasi menggunakan regex. Mirip dengan fitur pertanyaan teks (3.2.1), fitur tambah pertanyaan juga menggunakan algoritma KMP, BM, dan Levenshtein Distance untuk mengecek kesamaan pertanyaan user dengan database. Apabila seluruh pertanyaan pada database memiliki kemiripan <90%, maka bot akan mengeluarkan pesan berupa tiga pertanyaan dengan kemiripan tertinggi. Jika pertanyaan database paling mirip memiliki kemiripan >90% atau bahkan *exact match*, pertanyaan tersebut akan dihapus dari databse. Berikut regex yang digunakan untuk klasifikasi fitur hapus pertanyaan:

```
^\s*hapus pertanyaan (.+)$
```

Selain fitur-fitur yang sudah disebutkan, bot juga dapat meng*handle* banyak pertanyaan sekaligus. Karakter pembatas yang membedakan antar pertanyaan adalah ‘?’, ‘.’, ‘;’, dan ‘!’.

3.3 Arsitektur Aplikasi

Frontend aplikasi ini menggunakan bahasa pemrograman JavaScript dengan framework React. Framework React dipilih karena memiliki performa yang tinggi dan memungkinkan pengembang untuk membangun aplikasi berbasis komponen dengan mudah. Sementara itu, untuk membantu styling, digunakan library Chakra UI. Chakra UI menyediakan komponen-komponen yang siap pakai dan mudah dikostumisasi sesuai kebutuhan aplikasi.

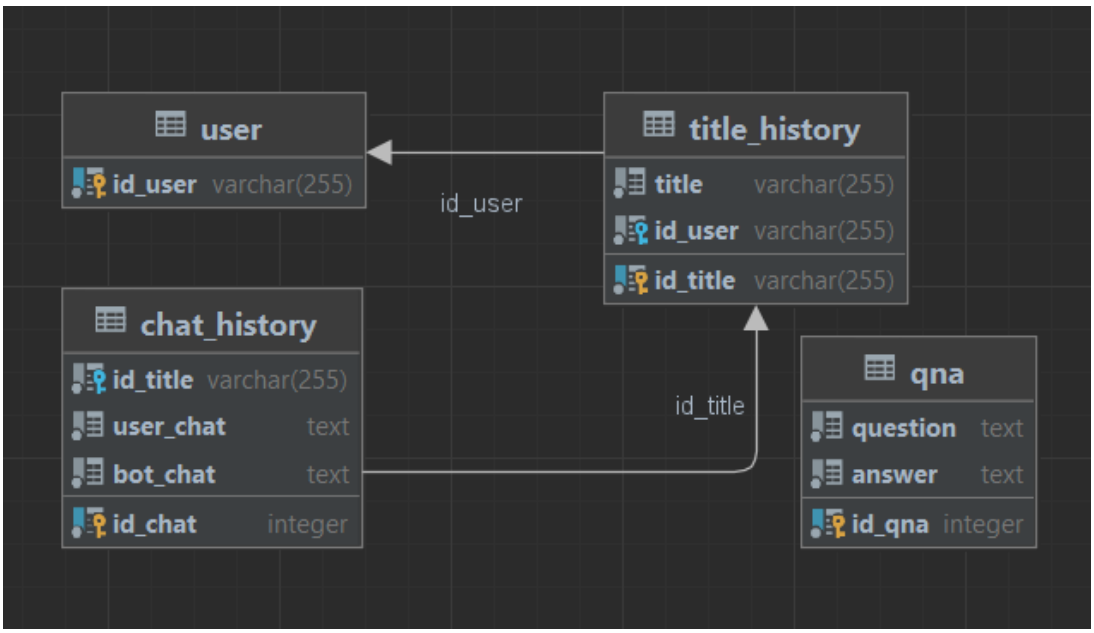
Untuk melakukan hit endpoint API, digunakan library bantuan dari axios untuk mempermudah. Axios adalah sebuah library untuk melakukan request HTTP pada browser. Library axios tersebut dipanggil langsung pada komponen atau modul yang membutuhkan akses ke API. Setelah selesai dibuat, aplikasi ini di-deploy ke Github Pages agar dapat diakses secara online. Github Pages adalah fitur dari Github yang memungkinkan pengguna untuk membuat website statis dan meng-hostingnya secara gratis.

Backend aplikasi ini menggunakan bahasa pemrograman Golang dengan framework Gin-Gonic. Framework ini dipilih karena memiliki performa yang baik dan memungkinkan pengembang untuk membangun aplikasi dengan pendekatan REST-ful dengan Clean architecture. Clean architecture adalah sebuah konsep arsitektur software yang menempatkan bisnis logic (algoritma *string matching*) pada pusat perhatian dan menjaganya agar tidak tercampur aduk dengan detail implementasi teknis lainnya seperti database, framework atau library. Aplikasi ini menyediakan 5 service utama, yaitu :

1. *Service load data history* (/history/{user_id}), service ini berfungsi untuk mengambil semua data history room chat yang pernah dibuat oleh pengguna.

2. *Service load data chat history* (/chat_history/{title_id}), service ini berfungsi untuk mengambil data chat history user pada room chat yang pernah dibuatnya.
3. *Service user input and bot respond* (/user_respond/{user_id}/{title_id}), service ini berfungsi untuk menerima masukan pengguna dan akan dilakukan string matching untuk mendapatkan hasil jawaban dari *chatbot* dan mengirimkannya kembali ke pengguna.
4. *Service rename history* (/rename_history/{title_id}), service untuk mengubah judul *chatroom* yang pernah dibuat oleh pengguna.
5. *Service delete history* (/delete_history/{title_id}), service untuk menghapus *chat room* yang pernah dibuat oleh pengguna.

Kami menggunakan database relasional PostgreSQL untuk menyimpan data yang diperlukan. Berikut adalah struktur basis data relasional yang kami buat,



Relasional Diagram Database ChatBjorka

Untuk menghubungkan dan mengelola basis data kami menggunakan database driver untuk PostgreSQL yaitu *pgx*. Aplikasi backend dan database server kami lakukan deployment pada cloud service Azure. Kami menggunakan service *App Service* untuk melakukan deployment pada aplikasi backend dan menggunakan service *Azure Database for PostgreSQL server* sebagai server database kami.

BAB 4
IMPLEMENTASI DAN PENGUJIAN

4.1. Spesifikasi Teknis Program

4.1.1. Library

Spesifikasi teknis progam untuk library terdiri dari fungsi-fungsi berikut:

1. func HandleQueries(r repository.Repo, c context.Context, str string, algo string) string
   Fungsi yang melakukan mengelola string input dari pengguna yang akan menghasilkan string jawaban dari bot
2. func preprocessQuery(str string) string
   Fungsi yang melakukan pre-proses untuk membersihkan string input pengguna
3. func solveExpression(str string) string
   Menyelesaikan input ekspresi aritmatika
4. func dateToDay(str string) string
   Fungsi yang mengolah input tanggal menjadi hari
5. func levenshteinDistance(str1 string, str2 string) int
   Fungsi untuk mendapatkan Levenshtein distance
6. func distToPercentage(levDist int, str1 string, str2 string) float64
   Fungsi untuk konversi dari Levenshtein distance ke persen
7. func bmSearch(text string, pattern string) int
   Fungsi untuk menjalankan algoritma BM search
8. func buildLast(pattern string) [256]int
   Fungsi untuk menghitung tabel last occurence untuk BM search
9. func kmpSearch(text string, pattern string) int
   Fungsi untuk menjalankan algoritma KMP search
10. func computeBorder(pattern string) []int
    Fungsi untuk menghitung border function KMP search

4.1.2. Backend

Spesifikasi teknis program untuk backend terdiri dari fungsi-fungsi berikut:

1. func GetAllHistory(uc usecase.UseCase) gin.HandlerFunc
   Handler untuk *service load data history.* Dengan *endpoint* GET
2. func DeleteHistory(uc usecase.UseCase) gin.HandlerFunc
   Handler untuk *service delete history.* Dengan *endpoint* DELETE
3. func RenameTitle(uc usecase.UseCase) gin.HandlerFunc
   Handler untuk *service rename history.* Dengan *endpoint* POST
4. func GetChatHistory(uc usecase.UseCase) gin.HandlerFunc
   Handler untuk *service load data chat history.* Dengan *endpoint* GET
5. func PostUserRespond(uc usecase.UseCase) gin.HandlerFunc
   Handler untuk *service user input and bot respond.* Dengan *endpoint* POST
6. func (u usecase) GenerateAnswer(c context.Context, idTitle string, idUser string, input entity.UserInput) (entity.BotOutput, error)

Fungsi dari usecase layer yang memanggil library untuk mendapatkan hasil atau jawaban dari bot

7. func (u usecase) GetChatById(c context.Context, idTitle string) ([]entity.ChatHistoryOutput, error)
   Fungsi dari usecase layer yang memanggil fungsi dari repository layer untuk mengambil data chat berdasarkan masukan idTitle
8. func (u usecase) DelChatById(c context.Context, idTitle string) error
   Fungsi dari usecase layer yang memanggil fungsi dari repository layer untuk menghapus data chat berdasarkan masukan idTitle
9. func (u usecase) AddChat(c context.Context, idTitle string, userChat string, botChat string) error
   Fungsi dari usecase layer yang memanggil fungsi dari repository layer untuk menambahkan data chat
10. func (u usecase) AddHistory(c context.Context, idTitle string, title string, idUser string) error
    Fungsi dari usecase layer yang memanggil fungsi dari repository layer untuk menambahkan riwayat berdasarkan idUser
11. func (u usecase) DelHistoryById(c context.Context, idTitle string) error
    Fungsi dari usecase layer yang memanggil fungsi dari repository layer untuk menghapus riwayat berdasarkan idTitle
12. func (u usecase) SetTitleById(c context.Context, idTitle string, newTitle string) error
    Fungsi dari usecase layer yang memanggil fungsi dari repository layer untuk memperbarui nama *chat room* atau *history* yang ada
13. func NewUseCase(repo repository.Repo) usecase
    Inisialisasi layer usecase yang menyimpan data repository layer untuk dilakukan pemaggilan fungsi repository
14. func (r repo) GetAllData(c context.Context) ([]entity.Qna, error)
    Fungsi repository layer yang mengambil semua data pertanyaan dan jawaban dari tabel “qna” di database
15. func (r repo) AddData(c context.Context, quest string, ans string) error
    Fungsi repository layer yang menambahkan data pertanyaan dan jawaban pada tabel “qna” di database
16. func (r repo) DeleteDataById(c context.Context, idQna int) error
    Fungsi repository layer yang menghapus data pertanyaan dan jawaban pada tabel “qna” di database berdasarkan idQna
17. func (r repo) GetChatById(c context.Context, idTitle string) (chat entity.ChatHistory, err error)
    Fungsi repository layer yang mengambil semua data chat pada tabel “chat_history” di database berdasarkan idTitle
18. func (r repo) DelChatById(c context.Context, idTitle string) error
    Fungsi repository layer yang menghapus semua data chat pada tabel “chat_history” di database berdasarkan idTitle
19. func (r repo) AddChat(c context.Context, idTitle string, userChat string, botChat string) error

Fungsi repository layer yang menambahkan data chat pada tabel “chat_history” di database

20. func (r repo) GetAllHistory(c context.Context, idUser string) ([]entity.History, error)
    Fungsi repository layer yang mengambil semua data riwayat atau chat room di tabel “title_history” di database
21. func (r repo) GetHistoryById(c context.Context, idTitle string, idUser string) (result entity.History, err error)
    Fungsi repository layer yang mengambil data riwayat atau chat room di tabel “title_history” di database berdasarkan idTitle
22. func (r repo) AddHistory(c context.Context, idTitle string, title string, idUser string) error
    Fungsi repository layer yang menambah data riwayat atay chat room di tabel “title_history” di database
23. func (r repo) DelHistoryById(c context.Context, idTitle string) error
    Fungsi repository layer yang menghapus data riwayat atau chat room di tabel “title_history” di database berdasarkan idTitle
24. func (r repo) SetTitleById(c context.Context, idTitle string, newTitle string) error
    Fungsi repository layer yang memperbarui data nama (title) riwayat atau chat room di tabel “title_history” di database berdasarkan idTitle
25. func NewRepo(db *pgxpool.Pool) repo
    Inisialisasi layer repository yang menyimpan koneksi ke database untuk dilakukan pengolahan database
26. func (r repo) AddUser(c context.Context, idUser string) error
    Fungsi repositroy layer untuk menambahkan user id ke tabel “user” di database

4.1.3. Frontend

Spesifikasi teknis program untuk front-end meliputi hal-hal yang berhubungan dengan pemanggilan API. Sebagai berikut:

1. getUserId()
   Mendapatkan user id yang tersimpan di *local browser* pengguna
2. loadHistory(maxRetries, retryDelay, userId)
   Melakukan metode GET pada API untuk mendapatkan chat history
3. loadChatHandler(chatId, maxRetries, retryDelay)
   Melakukan metode GET pada API untuk mendapatkan isi dari suatu chat
4. newUsrMsgHandler(newMsg, maxRetries, retryDelay)
   Melakukan metode POST pada API untuk mengirimkan pesan user ke dan yang kemudian mendapatkan respon balik pesan bot.
5. editTitlePayloadHandler(newTitle, maxRetries, retryDelay)
   Melakukan metode POST pada API untuk mengirimkan judul baru yang akan menggantikan judul lama.
6. deleteChatHandler(deleteId, maxRetries, retryDelay)
   Melakukan metode DELETE pada API untuk memberitahu bahwa suatu chat dihapus.
7. newChatSignalHandler()
   Menyiapkan state yang ada di program untuk menerima chat kosong baru.
8. Class Sidebar

Merupakan kelas component untuk sidebar program. Sidebar program merupakan components yang terletak di bagian kiri layar yang berisi history, tombol untuk chat baru, dan tombol untuk memilih algoritma yang akan digunakan.

9. Class Chat
   Merupakan kelas component untuk bagian utama isi dari chat konten. Bagian ini terdiri dari kotak pengiriman pesan user dan juga isi percakapan antara user dengan bot.

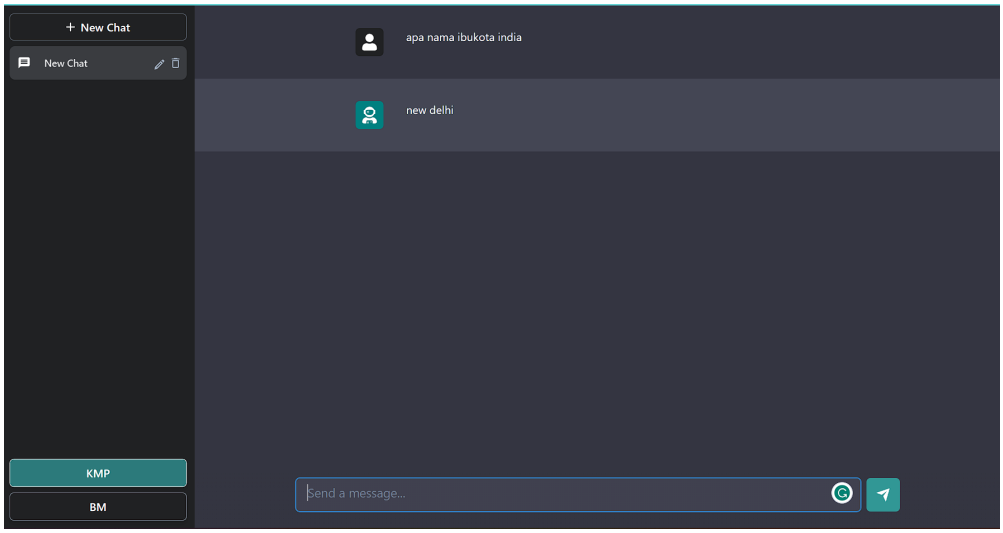
4.2. Tata Cara Penggunaan

Karena program dideploy, program dapat langsung dibuka pada https://saddamannais.github.io/Tubes3_13521109/. Adapun cara untuk membuka tautan melalui browser adalah sebagai berikut:

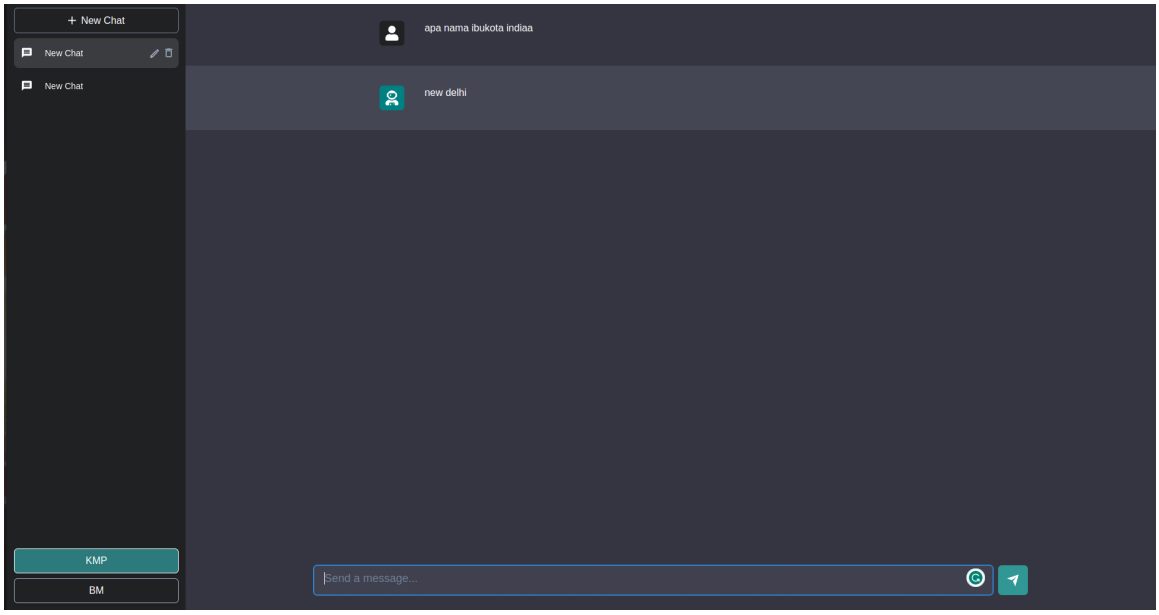
1. Pertama-tama pastikan bahwa perangkat yang digunakan terhubung dengan internet.
2. Buka aplikasi browser seperti Google Chrome, Mozilla Firefox, atau Microsoft Edge.
3. Masukkan alamat “https://saddamannais.github.io/Tubes3_13521109/” pada kolom URL di browser.
4. Tekan tombol Enter pada keyboard atau klik tombol Go untuk membuka link tersebut.
5. Tunggu beberapa saat hingga halaman web selesai dimuat.
6. Setelah halaman selesai dimuat, Anda akan melihat tampilan awal dari program tersebut.

4.3. Hasil Pengujian dan Analisis

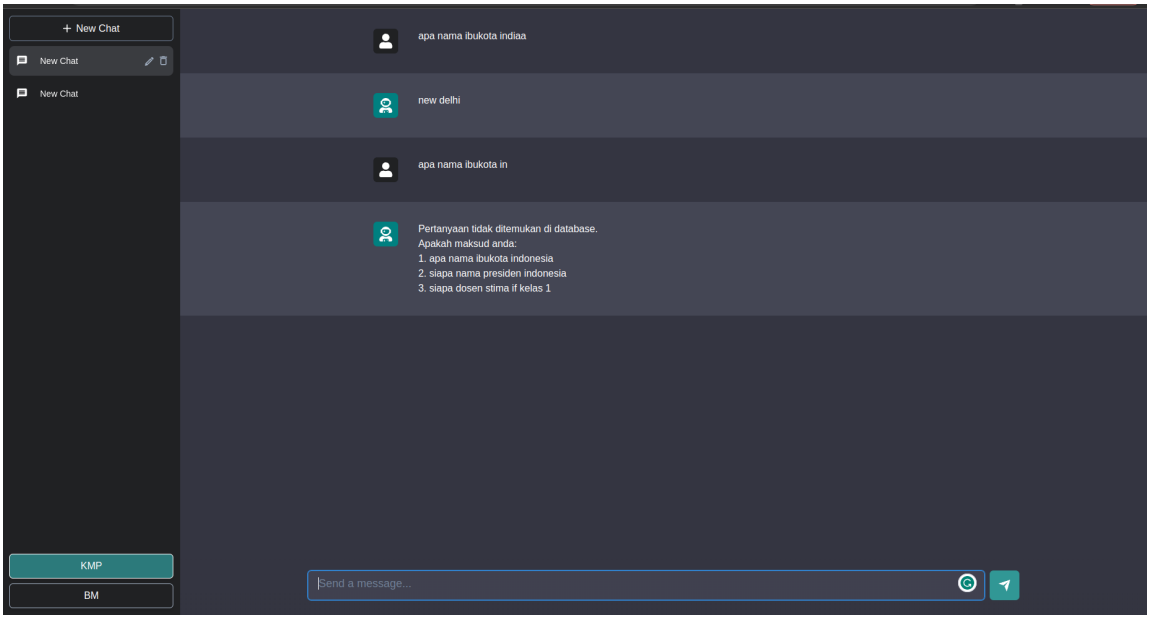
4.3.1. Fitur pertanyaan teks



Karena exact match, bot akan mengeluarkan jawaban yang sesuai dengan yang ada di database.

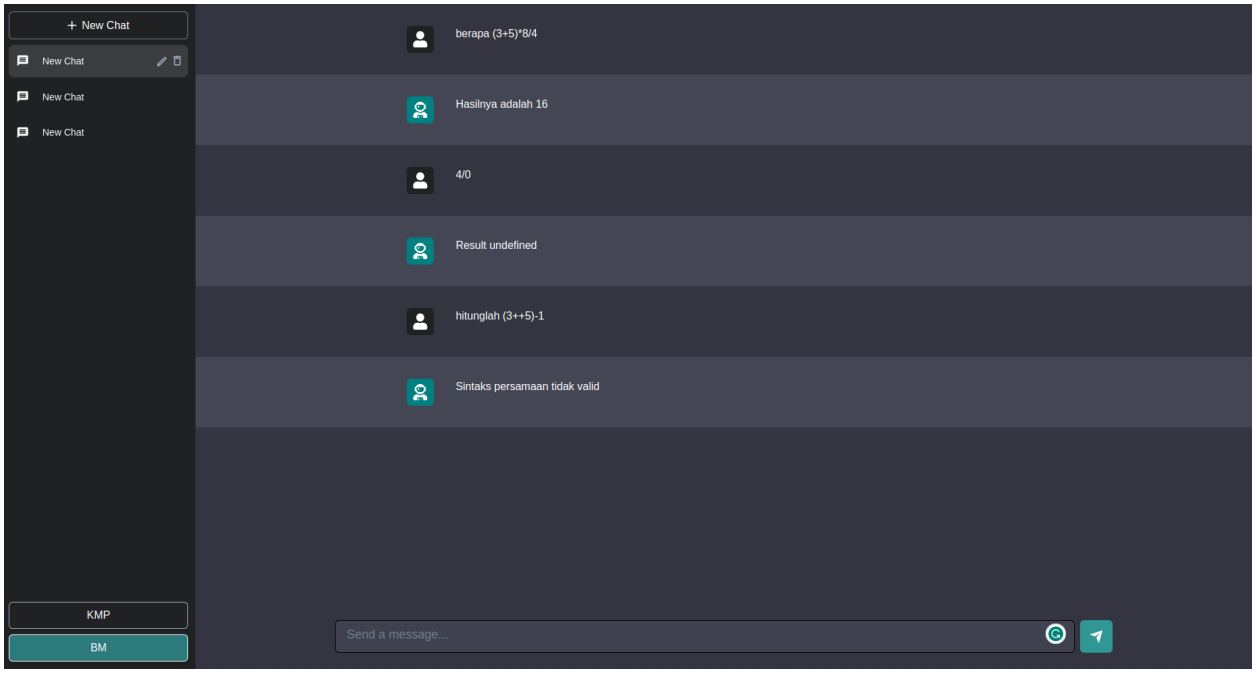


Walaupun tidak exact match, karena kemiripan diatas >90%, masih dianggap sebagai pertanyaan yang sesuai.



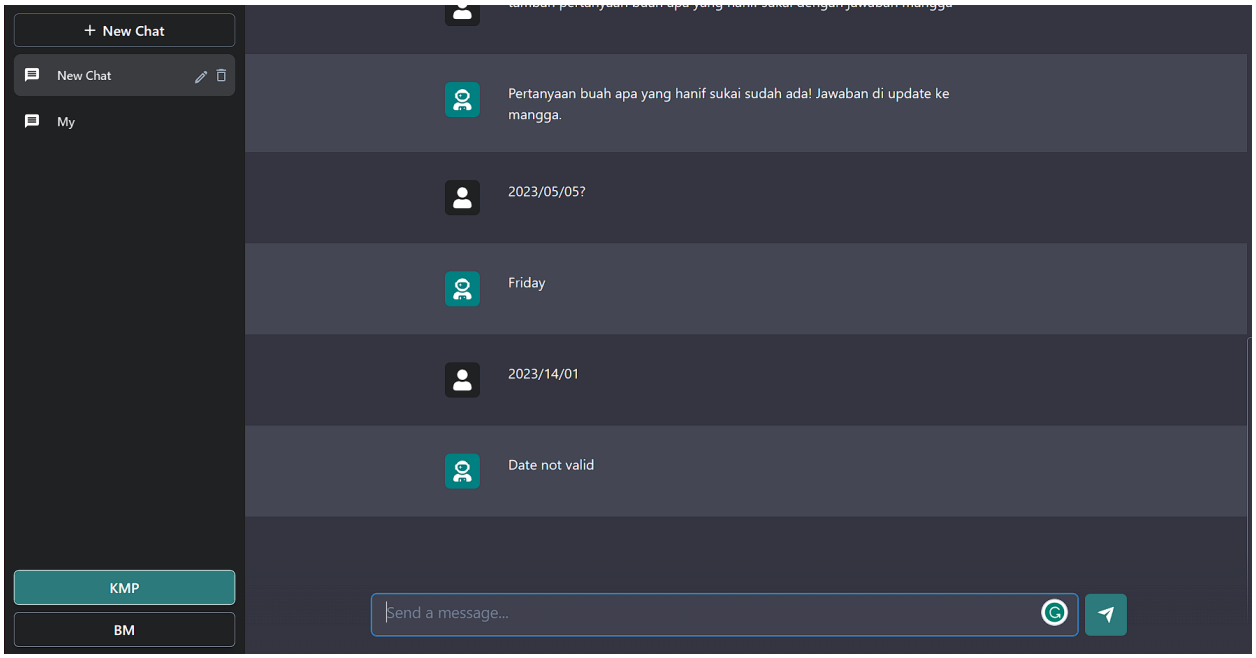
Kemiripan dibawah >90%, namun masih >30%, sehingga bot akan memberikan tiga pertanyaan dengan kemiripan tertinggi.

4.3.2. Fitur kalkulator



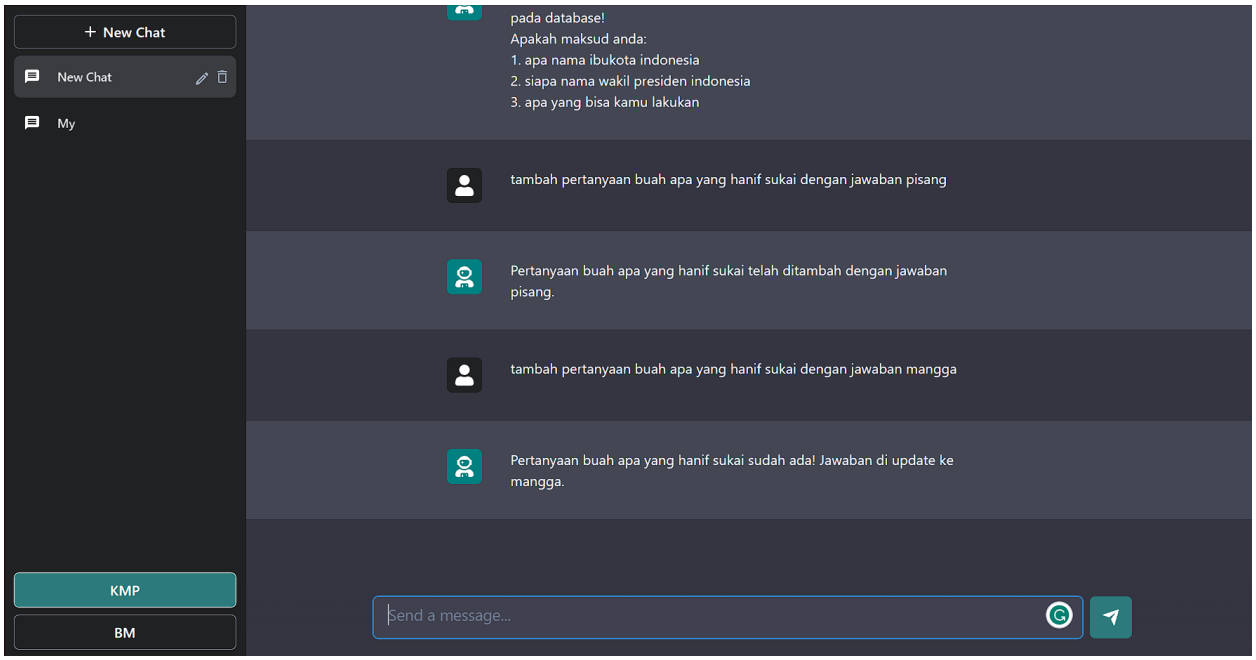
Bot dapat menghitung ekspresi aritmatika yang valid seperti pada pertanyaan pertama. Pada pertanyaan kedua, terdapat ekspresi aritmatika yang menghasilkan sesuatu yang tidak terdefinisi sehingga bot akan mengeluarkan result undefined. Pada pertanyaan terakhir, terdapat kesalahan sintaks, sehingga bot akan mengeluarkan pesan sintaks persamaan tidak sesuai

4.3.3 Fitur Tanggal



Bot dapat memvalidasi mana tanggal yang benar dan yang salah. Ketika tanggal tersebut benar, bot akan mengeluarkan hari dari tanggal tersebut. Jika salah, bot akan mengeluarkan tanda bahwa tanggal tidak valid.

4.3.4. Fitur Tambah pertanyaan



Bot dapat menambahkan pertanyaan baru pada database ketika kemiripan <90%. Jika >90%, maka untuk pertanyaan yang sama, jawaban akan ditimpa (*overwrite*)

4.3.5. Fitur Hapus Pertanyaan

Bot dapat menghapus pertanyaan yang ada pada database, pada input pertama pertanyaan memiliki kemiripan > 90% dengan pertanyaan yang ada didatabase dan berhasil menghapus

data pertanyaan dan jawabannya. Sedangkan input kedua pertanyaan memiliki kemiripan < 90% dengan pertanyaan yang ada di database

BAB 5
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Algoritma pencocokan string memiliki beragam variasi, di antaranya adalah algoritma Knuth-Morris-Pratt (KMP) dan Booyer-Moore (BM). Kedua algoritma ini berguna untuk mencocokkan dua buah string. Selain itu, regular expression juga dapat digunakan untuk melakukan pengecekan apakah suatu string sudah memenuhi suatu format.

Sebuah chat bot berbasis web yang memanfaatkan algoritma KMP dan BM berhasil dibuat. Sebelum melakukan pencocokan, kami menggunakan regular expression untuk mengecek input yang diterima apakah memenuhi suatu format atau tidak. Kemudian, hasil tersebut dilanjutkan dengan mengggunakan algoritma KMP dan BM untuk mengetahui input tersebut masuk ke dalam fitur mana. Setelah diketahui, input tersebut dikomputasi sesuai dengan golongan fitur tadi kemudian menghasilkan suatu respon bot yang akan dikirimkan balik ke user

5.2. Saran

Algoritma KMP dan BM tidak terlalu terpakai pada kasus pemrograman ini karena kedua algoritma tersebut berfokus untuk mencari pola dalam sebuah teks. Cara yang lebih cocok untuk kasus ini adalah menggunakan perhitungan Levenshtein distance atau algoritma kemiripan string lainnya. Levenshtein distance adalah ukuran jarak antara dua string, yaitu jumlah minimum dari operasi (penambahan, penghapusan, atau penggantian satu karakter) yang diperlukan untuk mengubah satu string menjadi string lainnya.

Link Repositori : https://github.com/SaddamAnnais/Tubes3_13521109