

PENERAPAN *STRING MATCHING* DAN *REGULAR EXPRESSION* DALAM PEMBUATAN ChatGPT SEDERHANA

Laporan Tugas Besar 3

Disusun untuk memenuhi tugas mata kuliah IF2211 Strategi Algoritma
pada Semester 2 (dua) Tahun Akademik 2022/2023



Oleh Kelompok C3GPT

| | |
|----------------------|----------|
| Jeffrey Chow | 13521046 |
| Bill Clinton | 13521064 |
| Chiquita Ahsanunnisa | 13521129 |

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

DAFTAR ISI

| | |
|---|-----------|
| DAFTAR ISI | 1 |
| BAB I DESKRIPSI TUGAS | 3 |
| 1.1 Latar Belakang | 3 |
| 1.2 Tugas | 3 |
| 1.3 Fitur | 4 |
| BAB II LANDASAN TEORI | 5 |
| 2.1 Algoritma Knuth-Morris-Pratt (KMP) | 5 |
| 2.2 Algoritma Boyer-Moore (BM) | 5 |
| 2.3 Regular Expression (Regex) | 7 |
| 2.4 Algoritma Levenshtein Distance | 8 |
| 2.5 Aplikasi Web yang Dibangun | 8 |
| BAB III ANALISIS PEMECAHAN MASALAH | 10 |
| 3.1 Langkah Penyelesaian Masalah secara Umum | 10 |
| 3.2 Langkah Penyelesaian Masalah Setiap Fitur | 11 |
| 3.2.1 Fitur Tanggal | 11 |
| 3.2.2 Fitur Kalkulator | 12 |
| 3.2.3 Fitur Tambah Pertanyaan | 12 |
| 3.2.4 Fitur Hapus Pertanyaan | 13 |
| 3.2.5 Fitur Cari ke Basis Data | 13 |
| 3.3 Fitur Fungsional | 13 |
| 3.4 Arsitektur Aplikasi Web yang Dibangun | 14 |
| BAB IV IMPLEMENTASI DAN PENGUJIAN | 15 |
| 4.1 Spesifikasi Teknis | 15 |
| 4.1.1 Struktur Data | 15 |
| 4.1.2 Fungsi dan Prosedur | 15 |
| 4.2 Tata Cara Penggunaan Program | 17 |
| 4.3 Hasil Pengujian dan Analisis | 18 |
| 4.3.1 Fitur Kalkulator | 19 |
| 4.3.2 Fitur Tanggal | 19 |
| 4.3.3 Fitur Tambahkan Pertanyaan | 20 |
| 4.3.4 Fitur Hapus Pertanyaan | 21 |
| 4.3.5 Fitur Cari ke Basis Data | 22 |
| 4.3.6 Multifitur | 24 |
| BAB V SIMPULAN DAN SARAN | 26 |
| 5.1 Simpulan | 26 |
| 5.2 Saran | 26 |
| 5.3 Tanggapan dan Refleksi | 26 |
| DAFTAR PUSTAKA | 28 |

BAB I

DESKRIPSI TUGAS

1.1 Latar Belakang

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.

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*, algoritma *string matching* dapat digunakan.

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.

1.2 Tugas

Dalam tugas besar ini, mahasiswa 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, 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 mahasiswa asalkan dijelaskan di laporan, namun disarankan menggunakan salah satu dari algoritma Hamming Distance, Levenshtein Distance, ataupun Longest Common Subsequence.

1.3 Fitur

ChatGPT sederhana yang dibuat wajib dapat melakukan beberapa fitur/klasifikasi *query* 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 pertanyaan 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 algoritma *string matching* untuk mencari pertanyaan xxx tersebut pada *database*.

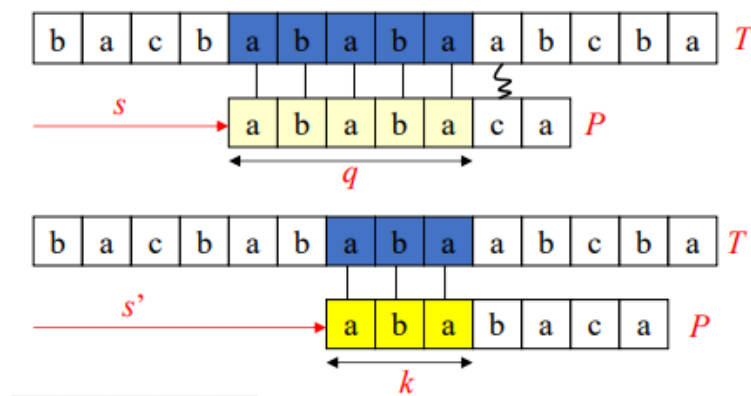
Klasifikasi dilakukan menggunakan *regex* dan terklasifikasi layaknya bahasa sehari-hari. Algoritma *string matching* KMP dan BM digunakan untuk klasifikasi *query* teks. Tersedia *toggle* untuk memilih algoritma KMP atau BM. Semua pemrosesan respons dilakukan pada sisi *backend*. Jika ada pertanyaan yang sesuai dengan fitur, maka tampilkan saja “Pertanyaan tidak dapat diproses”.

BAB II

LANDASAN TEORI

2.1 Algoritma Knuth-Morris-Pratt (KMP)

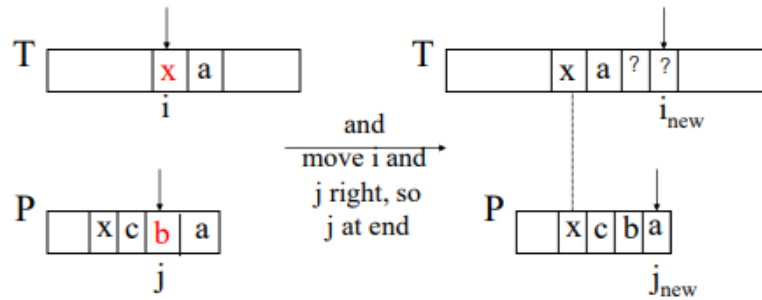
Algoritma Knuth-Morris-Pratt (KMP) adalah salah satu algoritma pencocokan string (*string matching*). Algoritma ini mencari sebuah *pattern* dalam sebuah text dengan urutan dari kiri ke kanan dengan mencocokkan setiap karakter satu per satu. Secara matematis, dalam pencariannya, jika ditemukan sebuah ketidakcocokan antara teks dan *pattern* P pada $P[j]$ ($T[i] \neq P[j]$), kita menggeser *pattern*-nya sejauh panjang *pattern* dikurangi panjang *prefix* terbesar dari $P[0..j-1]$ yang merupakan *suffix* dari $P[1..j-1]$. Panjang *prefix* terbesar yang merupakan *suffix* tersebut dapat disebut sebagai fungsi pinggiran KMP atau *KMP Border Function* (dinotasikan dengan $b(k)$ dengan $k = j-1$). Algoritma ini memiliki kompleksitas waktu untuk menghitung *border function*, yaitu $O(m)$ dan pencarian *string*, yaitu $O(n)$. Dengan demikian, kompleksitas waktu untuk algoritma KMP adalah $O(m + n)$.



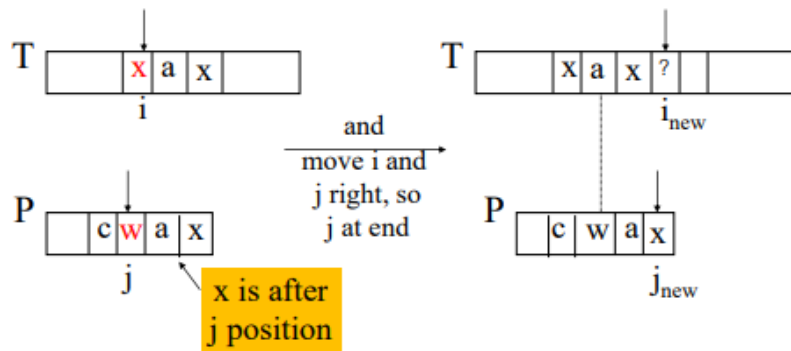
2.2 Algoritma Boyer-Moore (BM)

Algoritma Boyer-Moore juga merupakan salah satu algoritma pencocokan string (*string matching*). Algoritma ini didasarkan pada dua teknik, yaitu teknik *looking-glass* dan teknik *character-jump*. Teknik *looking-glass* adalah teknik yang dimana *pattern* P dicari dalam teks T dengan bergerak mundur, sedangkan teknik *character-jump* adalah teknik pergeseran akibat munculnya tiga kemungkinan yang terjadi saat ada ketidakcocokan pada $T[i] \neq x$. Tiga kemungkinan tersebut adalah sebagai berikut.

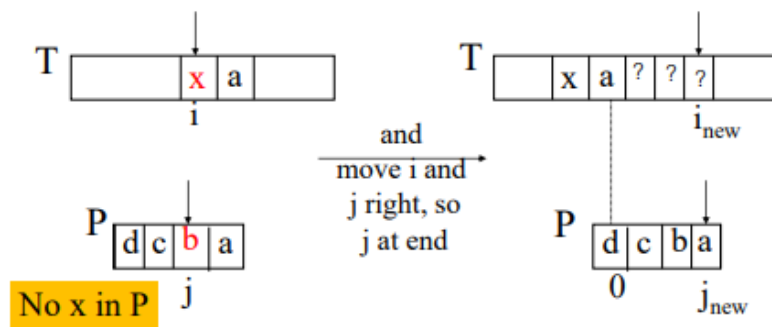
- Ketika *pattern* P mengandung x di suatu tempat, geser P ke kanan untuk menyejajarkan kemunculan terakhir x dalam P dengan $T[i]$



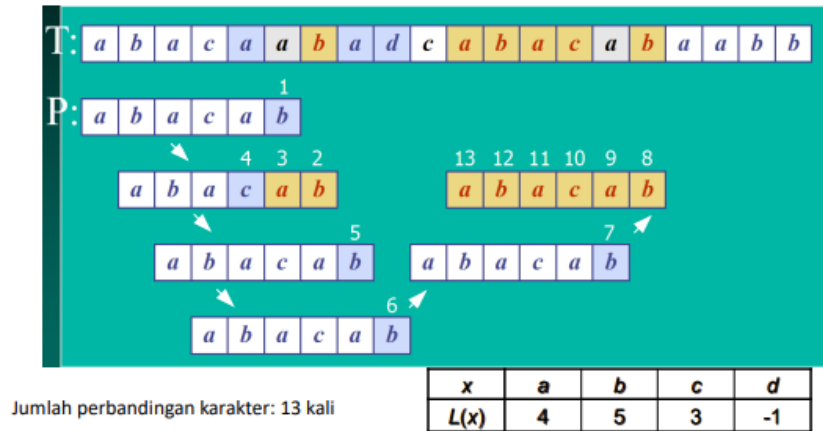
- Ketika *pattern* P mengandung x di suatu tempat dan pergeseran ke kanan ke kemunculan terakhir tidak mungkin, geser P sejauh 1 karakter ke $T[i+1]$.



- Jika keadaannya selain kedua kemungkinan sebelumnya, geser *pattern* P untuk menyelaraskan $P[0]$ dengan $T[i+1]$.



Fungsi untuk menghitung kemunculan terakhir dalam algoritma ini disebut *last occurrence function* (dinotasikan dengan $L(x)$). $L(x)$ didefinisikan sebagai index terbesar i sehingga $P[i] == x$ atau -1 jika indeks tersebut tidak ada.



2.3 Regular Expression (Regex)

Sebuah *Regular Expression* atau regex merupakan suatu filter yang mendeskripsikan string-string yang cocok dengan suatu *pattern* tertentu. Artinya, sebuah regex menerima beberapa string dan menolak string-string lainnya. Satu baris regex dapat menggantikan banyak baris kode pemrograman dengan mudah sehingga regex merupakan salah satu alat yang kuat dalam proses manipulasi string. Sebuah regex terdiri dari operator-operator (misalnya +, *, ?, |, ^), karakter-karakter, dan *metacharacter-metacharacter* (misalnya \d, \s, ., \D, \S).

Dalam program ini, regex digunakan untuk menentukan fitur yang akan digunakan serta untuk validasi pada beberapa fitur.

2.4 Algoritma Levenshtein Distance

Algoritma *Levenshtein Distance* merupakan suatu algoritma untuk menentukan ukuran kesamaan antara dua buah string. Algoritma ini didefinisikan sebagai perubahan minimum yang dibutuhkan untuk mengubah suatu string menjadi string yang lainnya. Perubahan tersebut dapat dilakukan melalui proses *insertion*, *deletion*, dan *substitution*. *Levenshtein distance* yang semakin sedikit menandakan kemiripan antara kedua string yang semakin tinggi. Sebaliknya, *Levenshtein distance* yang semakin tinggi menandakan kemiripan antara kedua string yang semakin rendah.

| |
|--------------|
| insertion |
| substitution |
| deletion |

Levenshtein distance between "GILY" and "GEELY" is 2.

| | | | | |
|---|---|---|---|---|
| G | I | | L | Y |
| G | E | E | L | Y |

| | | | | |
|---|---|---|---|---|
| G | | I | L | Y |
| G | E | E | L | Y |

2.5 Aplikasi Web yang Dibangun

Aplikasi web adalah perangkat lunak yang dapat diakses dari web *browser* dan berjalan pada suatu server web. Aplikasi web biasanya terdapat dua komponen utama yaitu *client-side* dan *server-side*. *Client-side* adalah bagian yang dijalankan pada web browser untuk mengatur tampilan dan interaksi pengguna dengan halaman web, seperti menampilkan teks, gambar, video, dan sebagainya. *Server-side* adalah bagian yang dijalankan pada server yang berfungsi untuk mengatur konten dinamis pada website seperti mengambil data dari database, API *routing* dan sebagainya.

Pada pengembangan web, terdapat beberapa konsep dan teknologi dasar yang umumnya digunakan yaitu:

1. HTML (*Hypertext Markup Language*) digunakan untuk membuat struktur dasar dari suatu halaman web seperti proses *layouting*.
2. CSS (*Cascading Style Sheet*) digunakan untuk mengatur tampilan dan keindahan dari website seperti pengaturan warna, ukuran, jenis font, dan animasi.

3. Javascript adalah bahasa pemrograman untuk mengatur interaksi yang terjadi pada web seperti pengaturan jika suatu tombol di klik, validasi, dropdown, bahkan untuk interaksi dengan database.
4. Database adalah tempat penyimpanan data yang akan digunakan oleh web. Data yang biasa disimpan adalah informasi tentang user atau berbagai data yang akan di-*render* pada web.
5. Web server adalah perangkat lunak yang menjalankan aplikasi web dan menyediakan akses ke halaman web melalui internet, contohnya Apache dan Nginx.
6. RESTful API adalah protokol yang berfungsi untuk menghubungkan suatu aplikasi web dengan layanan web lainnya. Biasanya digunakan untuk melakukan pengambilan data dari sumber eksternal atau yang disediakan oleh backend.
7. Framework adalah kerangka kerja yang digunakan untuk mengembangkan aplikasi web dengan lebih cepat contohnya Next.js, Django, Laravel.

BAB III

ANALISIS PEMECAHAN MASALAH

3.1 Langkah Penyelesaian Masalah secara Umum

Secara umum, berikut adalah urutan penyelesaian masalah yang dilakukan.

1. Program menerima pesan dari pengguna. Pesan tersebut bisa untuk satu fitur maupun banyak fitur.
2. Pesan tersebut akan di-*split* setiap ada karakter '?'. Tiap pesan yang sudah di-*split* akan dianggap sebagai pertanyaan individual.
3. Setiap pertanyaan diklasifikasikan berdasarkan karakteristik string nya. Klasifikasi ini dilakukan dengan memanfaatkan regex. Klasifikasi ini dilakukan berdasarkan fitur yang ada. Berikut adalah klasifikasi yang dilakukan, diurutkan berdasarkan prioritasnya.

- **Fitur Tanggal**

Pertanyaan diklasifikasikan sebagai pertanyaan untuk fitur kalkulator jika pesan mengandung tanggal dengan format DD/MM/YYYY. Dari pertanyaan, hanya akan diambil bagian tanggalnya saja.

- **Fitur Kalkulator**

Pertanyaan diklasifikasikan sebagai pertanyaan untuk fitur kalkulator jika pesan mengandung ekspresi matematika (mengandung bilangan, baik desimal maupun bulat, serta mengandung setidaknya salah satu dari operator kali (*), bagi (/), kurang (-), tambah(+), atau pangkat (^)). Dari pertanyaan, hanya akan diambil bagian ekspresi matematikanya saja.

- **Fitur Tambah Pertanyaan**

Pertanyaan diklasifikasikan sebagai pertanyaan untuk fitur tambah pertanyaan jika pesan memiliki format "Tambahkan pertanyaan [pertanyaan] dengan jawaban [jawaban]". Dari pertanyaan, hanya akan diambil bagian [pertanyaan] dan [jawaban] saja.

- **Fitur Hapus Pertanyaan**

Pertanyaan diklasifikasikan sebagai pertanyaan untuk fitur hapus pertanyaan jika pesan memiliki format "Hapus pertanyaan [pertanyaan]". Dari pertanyaan, hanya akan diambil bagian [pertanyaan] saja.

- **Fitur Cari ke Basis Data**

Pertanyaan yang tidak terklasifikasi sebagai salah satu dari keempat fitur di atas akan langsung dianggap sebagai pertanyaan untuk fitur cari ke basis data.

4. Setelah diklasifikasikan, pertanyaan ditangani oleh masing-masing fitur yang bertanggung jawab. Penanganan ini mencakup validasi dan pencarian.
5. Setelah ditangani, fitur yang bertanggung jawab akan mengembalikan jawaban dari pertanyaan yang dikirimkan.
6. Jawaban-jawaban untuk setiap pertanyaan disatukan dan dikembalikan lagi ke pengguna sebagai respons dari pesan yang dikirimkan pengguna.

3.2 Langkah Penyelesaian Masalah Setiap Fitur

3.2.1 Fitur Tanggal

Dari tanggal yang sudah diekstraksi dari pertanyaan, akan dilakukan validasi terlebih dahulu untuk mengecek apakah tanggal tersebut valid. Tanggal dianggap tidak valid jika setidaknya memenuhi salah satu kondisi di bawah ini.

- Bagian hari bernilai kurang dari 1 atau lebih dari 31.
- Bagian bulan bernilai kurang dari 1 atau lebih dari 12.
- Bagian tahun bernilai kurang dari 0.
- Bagian bulan bernilai 4, 6, 9, atau 11 tetapi memiliki bagian hari bernilai lebih dari 30.
- Bagian bulan bernilai 2, tetapi memiliki bagian hari bernilai lebih dari 29.
- Bagian bulan bernilai 2 dan bagian tahun tidak habis dibagi 4 (bukan tahun kabisat), tetapi memiliki bagian hari bernilai lebih dari 28.

Jika tanggal tidak valid, akan dikembalikan pesan “Tanggal tidak valid”. Jika tanggal valid, dilakukan perhitungan untuk mendapatkan hari dari tanggal tersebut. Perhitungan tersebut dilakukan dengan algoritma sebagai berikut.

1. *String* tanggal di-*split* oleh *splitter* ‘/’.
2. Pisahkan bagian hari, bulan, dan tahun, lalu *parse* menjadi bilangan.
3. Jika nilai bulan kurang dari 3, kurangi nilai tahun dengan 1.
4. Gunakan larik *magicNum* yang berisi [0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4]. Lakukan perhitungan di bawah ini untuk mendapatkan indeks hari (0 untuk Minggu, 1 untuk Senin, ..., 6 untuk Sabtu).

$$\text{Indeks Hari} = \text{tahun} + \left\lfloor \frac{\text{tahun}}{4} \right\rfloor - \left\lfloor \frac{\text{tahun}}{100} \right\rfloor + \left\lfloor \frac{\text{tahun}}{400} \right\rfloor + \text{magicNum}[\text{bulan} - 1] + \text{hari}$$

5. *Decode* indeks tersebut menjadi nama hari dan kembalikan nilainya.

3.2.2 Fitur Kalkulator

Dari ekspresi matematika yang sudah diekstraksi dari pertanyaan, akan dilakukan validasi dan perhitungan secara bersamaan dengan memanfaatkan struktur data *stack*. Berikut adalah algoritmanya.

1. Pisahkan ekspresi matematika menjadi token berdasarkan bilangan (mencakup bilangan desimal), operator, serta tanda kurung.
2. Siapkan *stack* operator dan *stack* bilangan.
3. Untuk setiap token, lakukan:
 - Jika token adalah bilangan, *push* token yang di-*parse* menjadi bilangan ke *stack* bilangan.
 - Jika token adalah operator dan *stack* operator kosong, *push* token tersebut ke *stack* operator.
 - Jika token adalah operator dan *stack* operator tidak kosong
 - 1) Jika prioritas token lebih besar daripada prioritas operator di *top stack* operator, *push* token tersebut ke *stack* operator.
 - 2) Jika tidak, lakukan proses perhitungan (dijelaskan di bawah) hingga *stack* operator kosong atau prioritas token lebih kecil sama dengan daripada prioritas operator di *top stack* operator. Lalu, *push* token tersebut ke *stack* operator.
 - Jika token adalah '(', *push* token tersebut ke *stack* operator.
 - Jika token adalah ')', lakukan proses perhitungan (dijelaskan di bawah) hingga ditemukan '('.
4. Lakukan proses perhitungan hingga *stack* operator kosong.
5. *Pop* *stack* bilangan, hasilnya adalah bilangan tersebut.

Proses perhitungan yang dimaksud pada bagian di atas yaitu melakukan *pop* dari *stack* bilangan sebanyak dua kali (misalnya menghasilkan *num1* dan *num2*) dan melakukan *pop* dari *stack* operator sebanyak sekali (misalnya menghasilkan *op*). Hitung hasil dari *num2 op num1* lalu *push* hasilnya ke *stack* bilangan.

3.2.3 Fitur Tambah Pertanyaan

Dari pertanyaan dan jawaban yang sudah diekstraksi dari pertanyaan, dilakukan langkah sebagai berikut.

1. Cari *exact match* dari pertanyaan tersebut ke basis data dengan memanfaatkan algoritma KMP atau BM (sesuai pilihan pengguna).

2. Jika ditemukan, ubah jawaban dari pertanyaan di basis data menjadi jawaban yang baru.

3.2.4 Fitur Hapus Pertanyaan

Dari pertanyaan yang diekstrak dari pertanyaan, dilakukan langkah sebagai berikut.

1. Cari *exact match* dari pertanyaan tersebut ke basis data dengan memanfaatkan algoritma KMP atau BM (sesuai pilihan pengguna).
2. Jika ditemukan, hapus pertanyaan tersebut dari basis data. Jika tidak ditemukan, kembalikan pesan kesalahan.

3.2.5 Fitur Cari ke Basis Data

Dari pertanyaan yang diekstrak dari pertanyaan, dilakukan langkah sebagai berikut.

1. Cari *exact match* dari pertanyaan tersebut ke basis data dengan memanfaatkan algoritma KMP atau BM (sesuai pilihan pengguna).
2. Jika tidak ditemukan *exact match*, cari data pertanyaan di basis data yang memiliki kemiripan lebih besar dari 90%. Kemiripan ini dihitung dengan memanfaatkan algoritma Levenshtein Distance.
3. Jika tidak ditemukan data pertanyaan dengan kemiripan lebih dari 90%, tampilkan tiga pertanyaan dengan tingkat kemiripan terbesar. Tiga pertanyaan ini seakan-akan 'ditawarkan' kepada pengguna.
4. Jika basis data kosong, dikembalikan pesan bahwa pertanyaan tidak sesuai dengan fitur.

3.3 Fitur Fungsional

Dalam web yang kami buat, ada beberapa fitur yang kami implementasikan. Fitur-fitur tersebut adalah sebagai berikut.

- Fitur Pertanyaan Teks

Pada fitur ini, pengguna bisa menanyakan jawaban dari suatu pertanyaan. Pertanyaan ini nantinya akan dicocokkan dengan pertanyaan yang telah ada dalam *database* dengan algoritma sesuai pilihan pengguna (Knuth-Morris-Pratt (KMP) atau Boyer-Moore (BM)). Jika pertanyaan tersebut ada atau mirip dengan yang ada dalam *database*, web akan menampilkan jawaban dari pertanyaan tersebut.

- Fitur Kalkulator

Pada fitur ini, pengguna bisa memasukkan pertanyaan berupa persamaan matematika (misalnya $2*3*5$ dan $4+6-7$). Jika persamaan tersebut valid, web akan menampilkan jawaban dari persamaan matematika tersebut.

- **Fitur Tanggal**

Pada fitur ini, pengguna bisa menanyakan nama hari pada suatu tanggal. Tanggal yang dimasukkan pengguna harus mengikuti format DD/MM/YYYY (misalnya 14/05/1989). Jika tanggal yang dimasukkan valid, web akan menampilkan nama hari pada tanggal tersebut.

- **Fitur Tambah Pertanyaan dan Jawaban**

Pada fitur ini, pengguna bisa menambahkan pertanyaan dan jawaban ke *database* sehingga jika pertanyaan tersebut ditanyakan lagi, web akan menampilkan jawaban sesuai yang dimasukkan pengguna sebelumnya. Untuk dapat menggunakan fitur ini, masukan pengguna harus mengikuti format “Tambahkan pertanyaan xxx dengan jawaban yyy” (xxx menandakan pertanyaan, sedangkan yyy menandakan jawaban).

- **Fitur Hapus Pertanyaan**

Pada fitur ini, pengguna bisa menghapus sebuah pertanyaan dari *database*. Untuk dapat menggunakan fitur ini, masukan pengguna harus mengikuti format “Hapus pertanyaan xxx” (xxx menandakan pertanyaan). Pertanyaan tersebut nantinya akan dicari terlebih dahulu di database, lalu dihapus.

3.4 Arsitektur Aplikasi Web yang Dibangun

Arsitektur aplikasi web yang kami bangun menggunakan *fullstack framework* Next.js dan berbasis *monorepo*. Database yang digunakan untuk aplikasi ini adalah MongoDB yang di *host* pada MongoDB Atlas. Aplikasi web kami di-*deploy* pada Vercel pada tautan [berikut](#).

Pada bagian *frontend* digunakan library React dan Tailwind CSS untuk mengatur fungsionalitas tombol, *text field*, dan sebagainya. Pada frontend juga digunakan library Axios untuk mengambil data dari API yang dikirim oleh *backend*.

Pada bagian *backend* digunakan library mongoose untuk melakukan koneksi ke database dan membuat skema database MongoDB.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Spesifikasi Teknis

4.1.1 Struktur Data

Data yang disimpan pada *database* disimpan menggunakan JSON(*Javascript Object Notation*). Implementasi struktur data dapat dilihat pada folder *models*. Berikut adalah contoh skema struktur data yang disimpan pada *database*:

| | |
|---------|---|
| QnA | <pre> _id: ObjectId('64547946f8fb828a0b84e55a') question: "apa mata kuliah terseu" answer: "stima" __v: 0 </pre> |
| History | <pre> _id: ObjectId('64547918f8fb828a0b84e553') created_time: 2023-05-05T03:33:44.579+00:00 __v: 0 </pre> |
| Chat | <pre> _id: ObjectId('64547945f8fb828a0b84e557') history_id: ObjectId('64547918f8fb828a0b84e553') message: "tambahkan pertanyaan apa mata kuliah terseu dengan jawaban stima" sender: "user" created_time: 2023-05-05T03:34:29.559+00:00 __v: 0 </pre> |

4.1.2 Fungsi dan Prosedur

| No | Fungsi/Prosedur | Keterangan |
|----|------------------------------|---|
| 1 | kmp(text,pattern) | Fungsi ini mengimplementasikan algoritma KMP dan mengembalikan indeks pertama dari text yang matching dengan pattern atau -1 jika tidak ada match |
| 2 | computeBorderKmp(pattern) | Fungsi ini berfungsi sebagai penunjang algoritma KMP untuk menemukan border function |
| 3 | exactMatchKMP(text, pattern) | Fungsi ini mengembalikan true jika text tepat sama dengan pattern |
| 4 | lastOcc(pattern) | Fungsi untuk mencari kemunculan terakhir pada pattern dan mengembalikan dictionary |
| 5 | bm(text, pattern) | Fungsi ini mengimplementasikan algoritma BM |

| | | |
|----|--|--|
| | | dan mengembalikan indeks pertama dari text yang matching dengan pattern atau -1 jika tidak ada match |
| 6 | exactMatchBM(str1, str2) | Fungsi ini mengembalikan true jika str1 tepat sama dengan str2 |
| 7 | isValidExp(exp) | Fungsi ini mengembalikan true jika exp adalah ekspresi matematika valid |
| 8 | calculate(num1, op, num2) | Fungsi ini melakukan operasi terhadap num1 dan num2 menggunakan operator op, mengembalikan hasil operasinya |
| 9 | evalExp(exp) | Fungsi ini mengevaluasi expresi dalam notasi infix dan mengembalikan hasil expresi |
| 10 | validateAndEvalExp(exp) | Fungsi ini melakukan validasi dan evaluasi terhadap ekspresi |
| 11 | isDateValid(date) | Fungsi ini mengembalikan true jika date valid |
| 12 | getDay(date) | Fungsi ini mengembalikan hari pada date |
| 13 | validateAndEvalDate(date) | Fungsi ini melakukan validasi dan evaluasi terhadap date. |
| 14 | isCalculatorQuery(input) | Fungsi ini mengecek apakah pada string input terdapat query untuk kalkulator dengan menggunakan regex. |
| 15 | isDateQuery(input) | Fungsi ini mengecek apakah pada string input terdapat query untuk tanggal dengan menggunakan regex. Formatnya adalah DD/MM/YYYY |
| 16 | isAddQuestionQuery(input) | Fungsi ini mengecek apakah pada string input terdapat query untuk menambahkan pertanyaan dengan regex. Formatnya adalah "tambahkan pertanyaan <pertanyaan> dengan jawaban <jawaban>" |
| 17 | isRemoveQuestionQuery(input) | Fungsi ini mengecek apakah pada string input terdapat query untuk menghapus pertanyaan dengan regex. Formatnya adalah "hapus pertanyaan <pertanyaan>" |
| 18 | evalQuestion(input, arr, fdel, fadd, fupdate, isKMP) | Fungsi ini mengevaluasi suatu pertanyaan dan melakukan <i>branching</i> pemanggilan fungsi terhadap jenis pertanyaan yang diajukan. |

| | | |
|----|--|--|
| 19 | splitQuestion(question, arr, fdel, fadd, fupdate, isKMP) | Fungsi ini melakukan pemisahan jika terdapat beberapa pertanyaan pada suatu query. |
| 20 | searchExactMatch(question, arr, isKMP) | Fungsi ini mencari question yang tepat sama dengan suatu pertanyaan pada arr |
| 21 | search90(question, arr) | Fungsi ini mencari question yang memiliki kemiripan diatas 90% dengan suatu pertanyaan pada arr |
| 22 | search3Nearest(question, arr) | Fungsi ini mencari 3 pertanyaan dengan kemiripan terdekat dengan question pada array |
| 23 | levenshteinDist(str1, str2) | Fungsi ini untuk mencari ukuran perbedaan dua buah string dengan menggunakan algoritma levenshtein |
| 24 | similarityPercentage(str1, str2) | Fungsi ini menghitung kemiripan antara dua buah string |

4.2 Tata Cara Penggunaan Program

Untuk menggunakan web kami, pertama pengguna harus menekan tombol +New Chat di bagian kiri atas. Setelah itu, pengguna dapat mengajukan pertanyaan yang ingin diketahui jawabannya dengan mengetikkan dan mengirimkannya pada kolom *chat*. Pengguna juga dapat mengganti algoritma yang digunakan untuk mencocokkan pertanyaan pengguna dengan pertanyaan dalam *database* (algoritma Knuth-Morris-Pratt (KMP) atau Boyer-Moore (BM)) dengan menekan pilihan yang diinginkan pada bagian kiri bawah (pada bagian lingkaran). Hasil penggantian tersebut akan mulai diterapkan setelah pengguna menekan tombol kirim pada kolom *chat*. Jika pertanyaan yang pengguna ajukan ada dalam *database*, web akan menampilkan jawaban dari pertanyaan tersebut melalui *chat*.

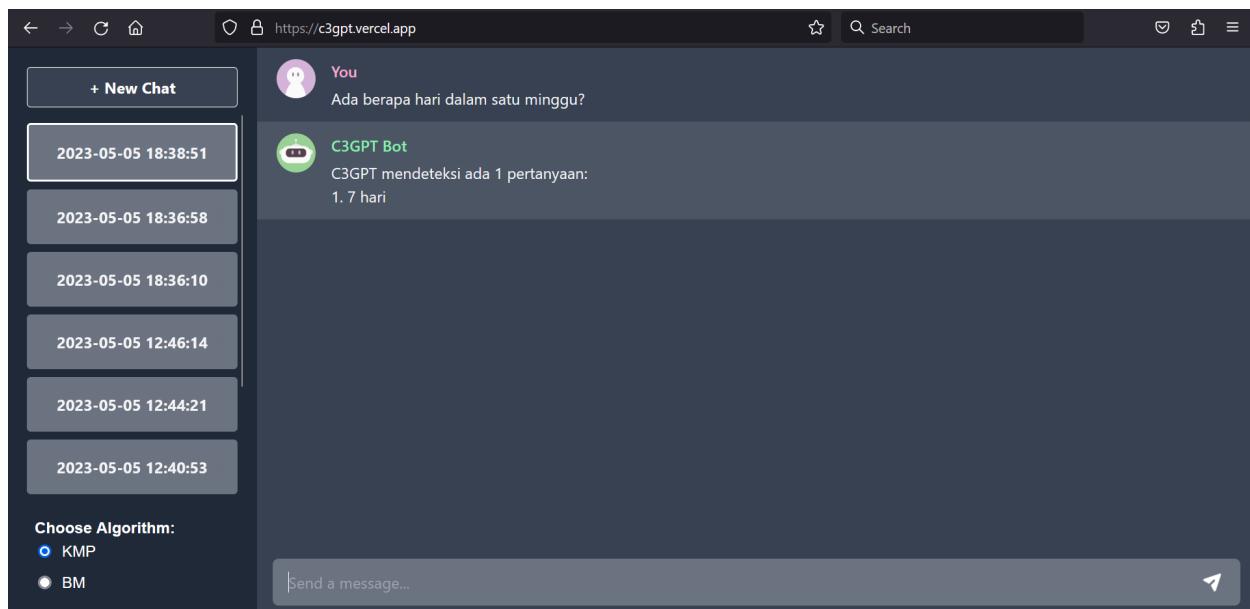
Selain itu fitur umum yang telah disebutkan sebelumnya, web kami juga memiliki beberapa fitur lain, seperti fitur kalkulator, fitur penentuan hari berdasarkan tanggal, fitur penambahan pertanyaan dan jawaban, dan fitur penghapusan pertanyaan dari *database*. Untuk menggunakan fitur kalkulator, pengguna harus memasukkan persamaan matematika yang valid, misalnya $2+3-1$. Jika persamaan ini valid, web akan menampilkan jawaban dari persamaan matematika tersebut melalui *chat*. Untuk menggunakan fitur penentuan hari berdasarkan tanggal, pengguna harus memasukkan input tanggal dengan format DD/MM/YYYY dengan DD merupakan tanggal, MM merupakan bulan, dan YYYY merupakan tahun. Jika input tersebut valid, web akan menampilkan nama hari yang sesuai dengan tanggal tersebut melalui *chat*. Untuk menggunakan fitur penambahan pertanyaan dan jawaban, pengguna harus

memasukkan input dengan format “Tambahkan pertanyaan xxx dengan jawaban yyy” (xxx menandakan pertanyaan, sedangkan yyy menandakan jawaban). Jika input ini valid, pasangan pertanyaan dan jawaban tersebut akan ditambahkan ke *database*. Terakhir, untuk menggunakan fitur penghapusan pertanyaan dari *database*, pengguna harus memasukkan input dengan format “Hapus pertanyaan xxx” (xxx menandakan pertanyaan). Jika pertanyaan tersebut ada dalam *database*, pertanyaan tersebut akan dihapus dari *database*.

Selain itu, jika pengguna merasa *chat* yang dilakukan sudah terlalu panjang, pengguna dapat memulai chat baru dengan menekan tombol +New Chat pada bagian kiri atas. Pengguna juga tidak perlu khawatir akan kehilangan informasi-informasi yang telah didapatkan setelah menanyakan pada web kami. Jawaban dari pertanyaan-pertanyaan sebelumnya dapat diakses dengan menekan kotak yang bertuliskan waktu penggunaan chat yang berisi pertanyaan-pertanyaan yang diajukan sebelumnya. Kotak tersebut terletak pada bagian kiri (di bawah tombol + New Chat). Terakhir, jika pengguna merasa telah membuat terlalu banyak *chat*, pengguna dapat menghapus kotak *chat* yang berisi percakapan sebelumnya tersebut dengan menekan tombol tong sampah pada bagian kanan kotak yang berisi waktu *chat*.

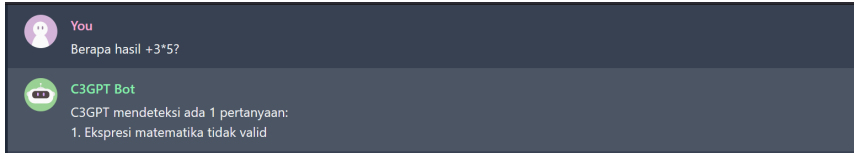


4.3 Hasil Pengujian dan Analisis

Berikut adalah tampilan *web application* kami secara umum.



Beberapa fitur yang ada dalam *web application* kami adalah sebagai berikut.

4.3.1 Fitur Kalkulator

| No. | Tangkapan Layar | Keterangan |
|-----|--|---|
| 1 |  | <i>Syntax</i> tidak valid |
| 2 |  | <i>Syntax</i> valid, namun ada pembagian dengan 0 |
| 3 |  | <i>Syntax</i> dan ekspresi valid |

Berdasarkan hasil pengujian di atas, dapat dilihat bahwa fitur kalkulator yang diimplementasikan sudah mampu menangani ekspresi matematika yang tidak valid (termasuk menangani kasus pembagian dengan 0) maupun yang valid. Sebagai catatan, fitur kalkulator tidak menggunakan algoritma KMP maupun BM.

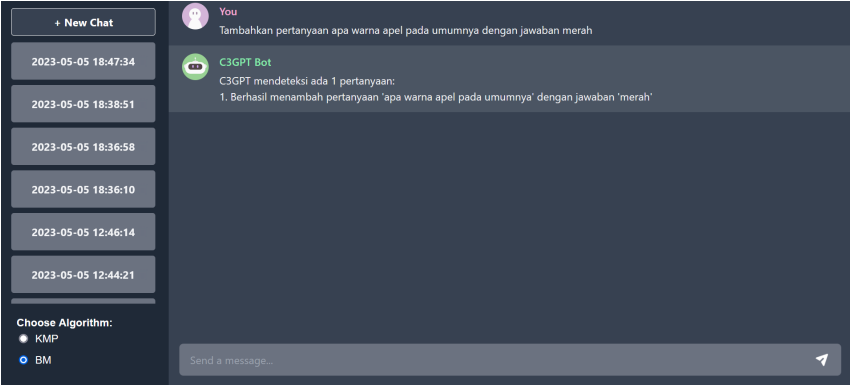
4.3.2 Fitur Tanggal

| No. | Tangkapan Layar | Keterangan |
|-----|--|---------------------|
| 1 |  | Tanggal tidak valid |
| 2 |  | Tanggal valid |

Berdasarkan hasil pengujian di atas, dapat dilihat bahwa fitur tanggal yang diimplementasikan sudah mampu menangani tanggal yang tidak valid maupun yang valid. Sebagai catatan, fitur tanggal tidak menggunakan algoritma KMP maupun BM.

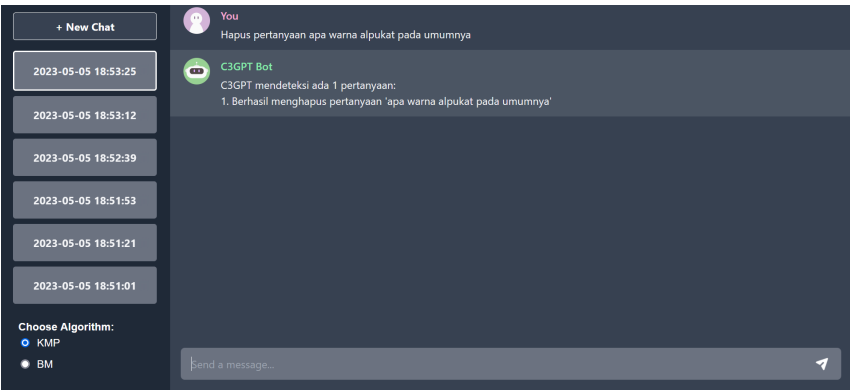
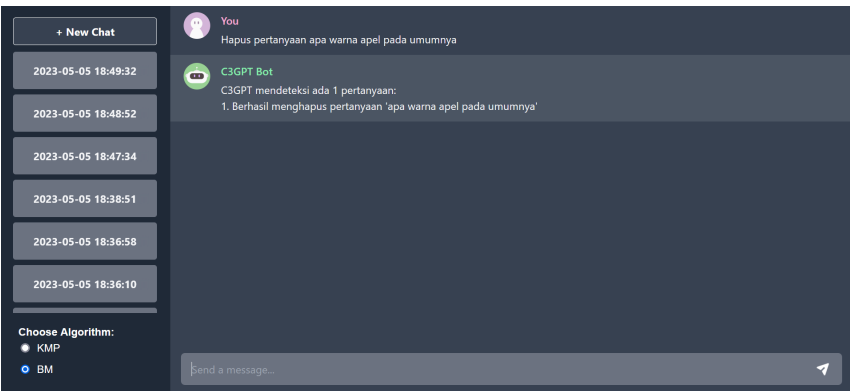
4.3.3 Fitur Tambahkan Pertanyaan

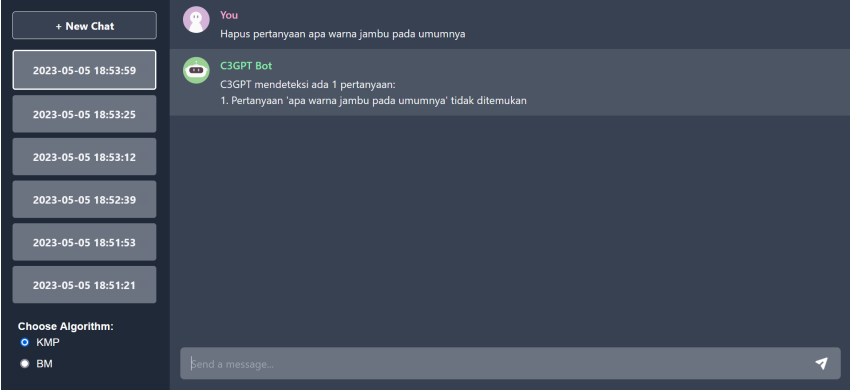

| No. | Tangkapan Layar | Keterangan |
|-----|---|--|
| 1 |  <p>The screenshot shows a chat window with a sidebar on the left containing a list of messages with timestamps. The main chat area shows a conversation where the user asks a question, and the C3GPT Bot responds. The 'Choose Algorithm' section at the bottom has 'KMP' selected. The input field contains the text 'Tambahkan pertanyaan'.</p> | Pertanyaan sudah ada di basis data (dicari dengan algoritma KMP) |
| 2 |  <p>The screenshot shows a chat window similar to the first one, but the 'Choose Algorithm' section at the bottom has 'BM' selected. The input field contains the text 'Tambahkan pertanyaan'.</p> | Pertanyaan sudah ada di basis data (dicari dengan algoritma BM) |
| 3 |  <p>The screenshot shows a chat window with a longer list of messages in the sidebar. The main chat area shows a sequence of questions and answers. The 'Choose Algorithm' section at the bottom has 'KMP' selected. The input field contains the text 'Tambahkan pertanyaan'.</p> | Pertanyaan belum ada di basis data (dicari dengan algoritma KMP) |

| | | |
|---|--|---|
| 4 |  | Pertanyaan belum ada di basis data (dicari dengan algoritma BM) |
|---|--|---|

Berdasarkan hasil pengujian di atas, dapat dilihat bahwa baik algoritma KMP maupun BM dapat mengaplikasikan *string matching* untuk *exact match* dengan baik. Fitur juga mampu menangani kasus saat pertanyaan yang akan ditambahkan sudah ada di basis data (melalui *update* jawaban) maupun belum.

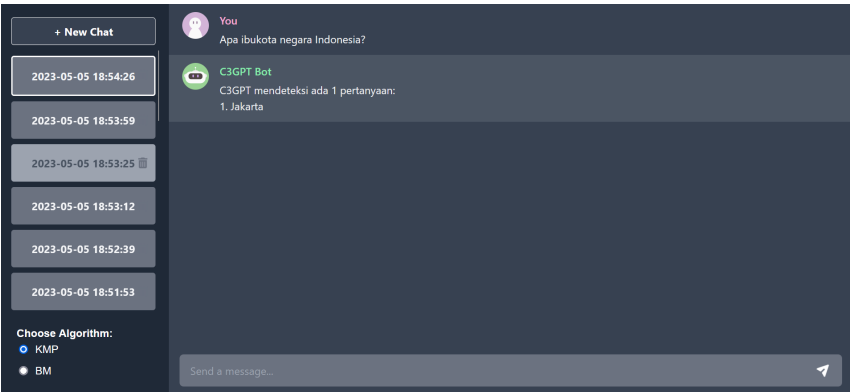
4.3.4 Fitur Hapus Pertanyaan

| No. | Tangkapan Layar | Keterangan |
|-----|--|--|
| 1 |  | Pertanyaan ada di basis data (dicari dengan algoritma KMP) |
| 2 |  | Pertanyaan ada di basis data (dicari dengan algoritma BM) |

| | | |
|---|---|--|
| 3 |  | Pertanyaan tidak ada di basis data (dicari dengan algoritma KMP) |
| 4 |  | Pertanyaan tidak ada di basis data (dicari dengan algoritma BM) |

Berdasarkan hasil pengujian di atas, dapat dilihat bahwa baik algoritma KMP maupun BM dapat mengaplikasikan *string matching* untuk *exact match* dengan baik. Fitur juga mampu menangani kasus saat pertanyaan yang akan dihapus ada di basis data maupun tidak (melalui pesan kesalahan).

4.3.5 Fitur Cari ke Basis Data


| No. | Tangkapan Layar | Keterangan |
|-----|--|---|
| 1 |  | Pertanyaan ada di basis data, dicari dengan algoritma KMP |

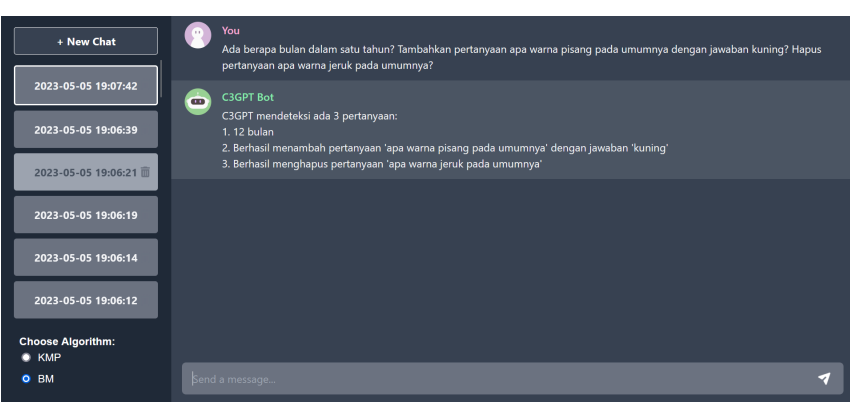
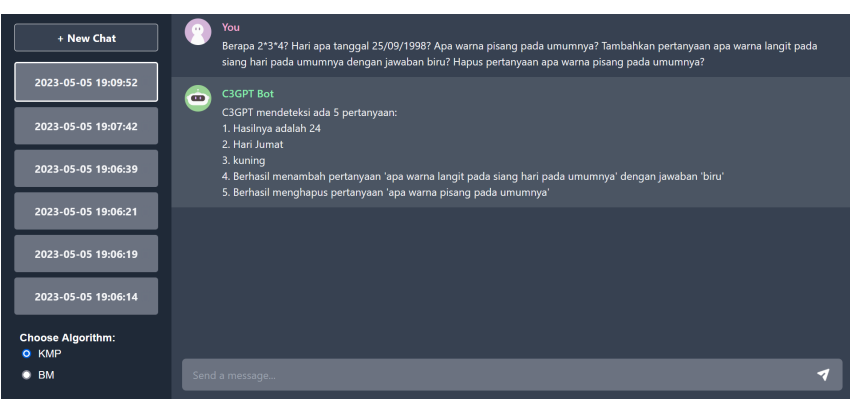
| | | |
|---|--|---|
| 2 |  | Pertanyaan ada di basis data, dicari dengan algoritma BM |
| 3 |  | Pertanyaan tidak ada di basis data (dicari dengan algoritma KMP), didapat pertanyaan dengan kemiripan > 90% |
| 4 |  | Pertanyaan tidak ada di basis data (dicari dengan algoritma BM), didapat pertanyaan dengan kemiripan > 90% |
| 5 |  | Pertanyaan tidak ada di basis data (dicari dengan algoritma KMP), tidak didapat pertanyaan dengan kemiripan > 90% |

| | | |
|---|--|--|
| 6 |  | Pertanyaan tidak ada di basis data (dicari dengan algoritma BM), tidak didapat pertanyaan dengan kemiripan > 90% |
|---|--|--|

Berdasarkan hasil pengujian di atas, dapat dilihat bahwa baik algoritma KMP maupun BM dapat mengaplikasikan *string matching* untuk *exact match* dengan baik. Fitur juga mampu menangani kasus saat pertanyaan yang ditanyakan belum ada di basis data, yaitu melalui pencarian pertanyaan dengan kemiripan di atas 90% (jika ada), ataupun melalui pencarian tiga pertanyaan termirip.

4.3.6 Multifitur

| No. | Tangkapan Layar | Keterangan |
|-----|--|---|
| 1 |  | Pertanyaan untuk fitur kalkulator, tanggal, serta cari ke database secara bersamaan |

| | | |
|---|--|--|
| 2 |  <p>The screenshot shows a chat window with a sidebar on the left containing a list of chat messages with timestamps. The main chat area shows a user message: 'Ada berapa bulan dalam satu tahun? Tambahkan pertanyaan apa warna pisang pada umumnya dengan jawaban kuning? Hapus pertanyaan apa warna jeruk pada umumnya?'. The bot response is: 'C3GPT mendeteksi ada 3 pertanyaan: 1. 12 bulan 2. Berhasil menambah pertanyaan 'apa warna pisang pada umumnya' dengan jawaban 'kuning' 3. Berhasil menghapus pertanyaan 'apa warna jeruk pada umumnya''. At the bottom, there is a 'Choose Algorithm:' section with radio buttons for 'KMP' and 'BM', and a 'Send a message...' input field.</p> | <p>Pertanyaan untuk fitur cari ke database, tambah pertanyaan, dan hapus pertanyaan secara bersamaan</p> |
| 3 |  <p>The screenshot shows a chat window similar to the one above. The user message is: 'Berapa 2*3*4? Hari apa tanggal 25/09/1998? Apa warna pisang pada umumnya? Tambahkan pertanyaan apa warna langit pada siang hari pada umumnya dengan jawaban biru? Hapus pertanyaan apa warna pisang pada umumnya?'. The bot response is: 'C3GPT mendeteksi ada 5 pertanyaan: 1. Hasilnya adalah 24 2. Hari Jumat 3. kuning 4. Berhasil menambah pertanyaan 'apa warna langit pada siang hari pada umumnya' dengan jawaban 'biru' 5. Berhasil menghapus pertanyaan 'apa warna pisang pada umumnya''. The 'Choose Algorithm:' section and input field are also visible.</p> | <p>Pertanyaan untuk keseluruhan fitur secara bersamaan</p> |

Berdasarkan hasil pengujian di atas, dapat dilihat bahwa program mampu menangani pesan yang memanfaatkan banyak fitur (bahkan keseluruhan yang berbeda-beda).

BAB V

SIMPULAN DAN SARAN

5.1 Simpulan

Dari hasil tugas besar III IF2211 Strategi Algoritma ini, kami berhasil memperoleh kesimpulan-kesimpulan sebagai berikut.

- Algoritma Knuth-Morris-Pratt (KMP) dan Bayer-Moore (BM) merupakan algoritma-algoritma yang digunakan untuk permasalahan pencocokan string (*string matching*).
- *Regular Expression* atau regex adalah alat yang dapat memfilter string-string yang cocok dengan suatu *pattern* tertentu.
- Algoritma *Levenshtein Distance* merupakan algoritma yang dapat digunakan untuk mencari tingkat kemiripan antara dua buah string.
- Next.js merupakan React framework yang sangat berguna dalam pembuatan web.
- Algoritma Knuth-Morris-Pratt (KMP), algoritma Boyer-Moore (BM), algoritma *Levenshtein Distance*, *Regular Expression*, dan Next.js berhasil kami gunakan untuk mengimplementasikan *web application* mirip ChatGPT sederhana ini.
- *Web application* ini berhasil kami *deploy*.

5.2 Saran

Dari pengerjaan tugas besar III IF2211 Strategi Algoritma ini, kami memberikan saran untuk mengadakan fitur *rename* untuk mengganti nama kotak yang menandakan *chat* tertentu.

5.3 Tanggapan dan Refleksi

Dari hasil tugas besar III IF2211 Strategi Algoritma ini, penulis memiliki tanggapan dan refleksi sebagai berikut.

- Penulis berhasil mengimplementasikan algoritma Knuth-Morris-Pratt (KMP), algoritma Boyer-Moore (BM), dan algoritma *Levenshtein Distance* dalam pembuatan *application* mirip ChatGPT sederhana ini.
- Penulis berhasil menggunakan *Regular Expression* (regex) untuk mengidentifikasi jenis masukan dari pengguna
- Penulis mendapatkan pengalaman bekerja secara berkelompok dalam mengimplementasikan *front-end* dan *back-end* dari suatu *web application*.

- Penulis berhasil membuat *interface* yang menarik untuk mendukung *web application* yang kami buat untuk tugas besar III IF2211 Strategi Algoritma ini.
- Penulis berhasil membuat *database* untuk mendukung *web application* yang kami buat untuk tugas besar III IF2211 Strategi Algoritma ini.

DAFTAR PUSTAKA

- Chua, Hock-Chuan. (2018). *Regular Expression (Regex) Tutorial*. Dilansir dari: <https://www3.ntu.edu.sg/home/ehchua/programming/howto/Regexe.html>. Diakses pada 3 Mei 2023.
- Cuelogic. (2017). *The Levenshtein Algorithm*. Dilansir dari: <https://www3.ntu.edu.sg/home/ehchua/programming/howto/Regexe.html>. Diakses pada 3 Mei 2023.
- Educative, Inc. (n.d.). *The Levenshtein Distance Algorithm*. Dilansir dari: <https://www.cuelogic.com/blog/the-levenshtein-algorithm>. Diakses pada 3 Mei 2023.
- Khodra, Masayu Leylia. (2018). *String Matching dengan Regular Expression*. Dilansir dari Homepage Rinaldi Munir: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf>. Diakses pada 1 Mei 2023.
- Munir, Rinaldi. (2020). *Pencocokan String*. Dilansir dari Homepage Rinaldi Munir: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf>. Diakses pada 1 Mei 2023.

LAMPIRAN

Lampiran 1 Pranala ke *Repository* GitHub

https://github.com/ashnchiquita/Tubes3_13521046

Lampiran 2 Pranala ke *Website* C3GPT

c3gpt.vercel.app

Lampiran 3 Pranala ke *Video Youtube*

<https://youtu.be/qRN3jSWH8fk>