

Tugas Besar 3 IF2211 Strategi Algoritma

Semester II tahun 2022/2023

Penerapan String Matching dan Regular Expression dalam Pembuatan ChatGPT Sederhana



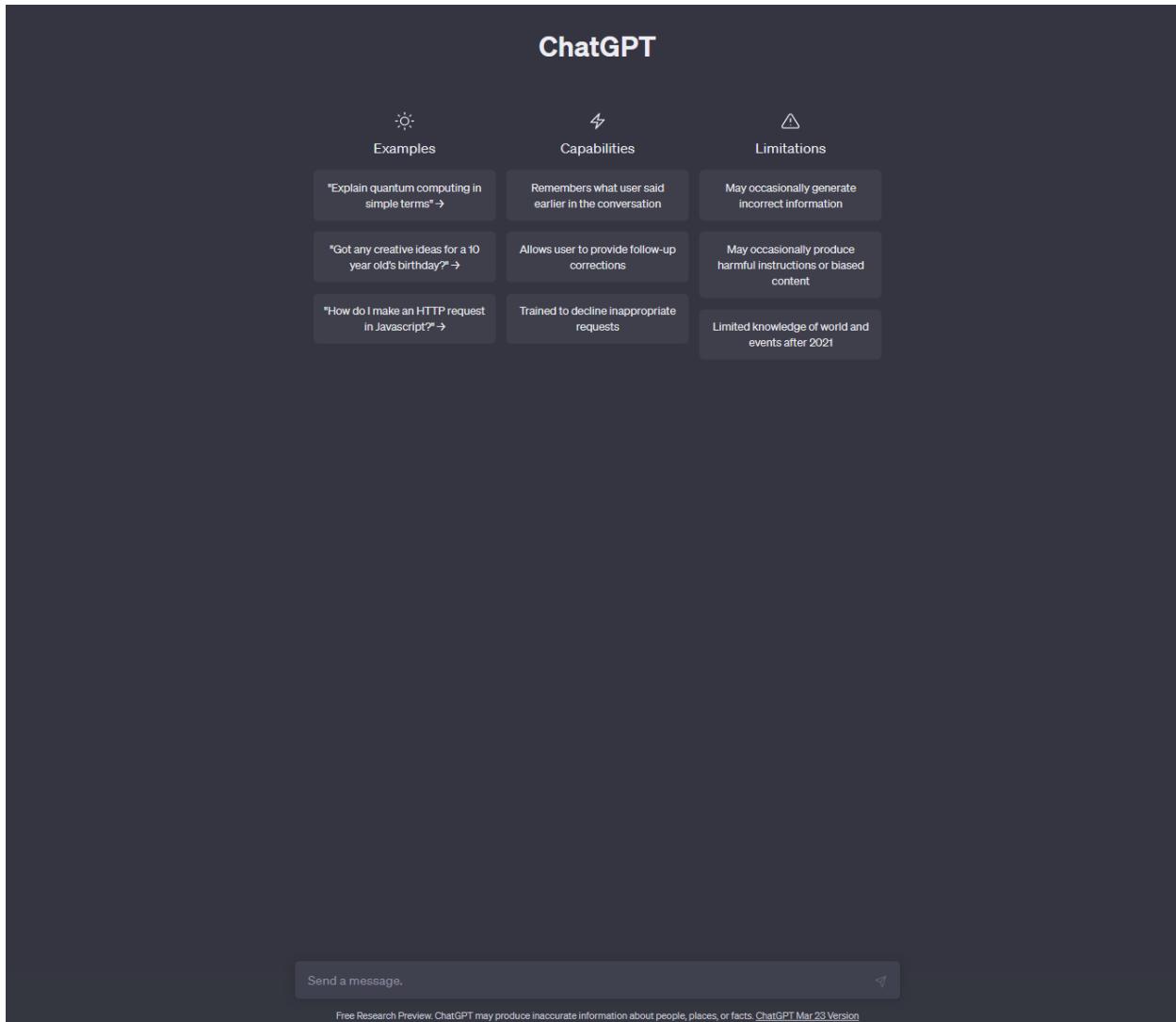
Disusun oleh :
13521104 Muhammad Zaydan A
13521134 Rinaldy Adin
13521142 Enrique Alifio Ditya

**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2022/2023**

BAB I

DESKRIPSI MASALAH

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

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. 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.

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 pertanyaan xxx dengan jawaban yyy”. Menggunakan algoritma string matching untuk mencari tahu apakah pertanyaan sudah ada. Apabila sudah, maka jawaban akan diperbarui.

5. Hapus pertanyaan dari database

Pengguna dapat menghapus sebuah pertanyaan dari database dengan query contoh “Hapus pertanyaan xxx”. Menggunakan string algoritma string matching untuk mencari pertanyaan xxx tersebut pada database.

Klasifikasi dilakukan menggunakan regex dan terklasifikasi layaknya bahasa sehari - hari. Algoritma string matching KMP dan BM digunakan untuk klasifikasi query teks. Tersedia toggle untuk memilih algoritma KMP atau BM. Semua pemrosesan respons dilakukan pada sisi backend. Jika ada pertanyaan yang sesuai dengan fitur, maka tampilkan saja “Pertanyaan tidak dapat diproses”. Berikut adalah beberapa contoh ilustrasi sederhana untuk tiap pertanyaannya. (Note: Tidak wajib mengikuti ilustrasi ini, tampilan disamakan dengan chatGPT juga boleh)

Spesifikasi Program:

1. Aplikasi berbasis website dengan pembagian Frontend dan Backend yang jelas.
2. Implementasi Backend wajib menggunakan Node.js / Golang, sedangkan Frontend dibebas tetapi disarankan untuk menggunakan React / Next.js / Vue / Angular. Lihat referensi untuk selengkapnya.
3. Penyimpanan data wajib menggunakan basis data (MySQL / PostgreSQL / MongoDB).

4. Algoritma pencocokan string (KMP dan Boyer-Moore) dan Regex wajib diimplementasikan pada sisi Backend aplikasi.
5. Informasi yang wajib disimpan pada basis data: a. Tabel pasangan pertanyaan dan Jawaban b. Tabel history
6. Skema basis data dibebaskan asalkan mencakup setidaknya kedua informasi di atas.
7. Proses string matching pada tugas ini Tidak case sensitive.
8. Pencocokan yang dilakukan adalah dalam satu kesatuan string pertanyaan utuh (misal “Apa ibukota Filipina?”), bukan kata per kata (“apa”, “ibukota”, “Filipina”).

Lain-lain:

1. Anda dapat menambahkan fitur-fitur lain yang menunjang program yang anda buat (unsur kreativitas).
2. Tugas dikerjakan berkelompok, minimal 2 orang dan maksimal 3 orang, boleh lintas kelas namun tidak boleh sekelompok dengan orang yang sama dengan tubes ataupun tucil stima sebelumnya.
3. Semua kelompok harap mengisi data kelompok mereka pada link <https://bit.ly/KelompokTubes3Stima>
4. Batas akhir pengisian anggota kelompok adalah 16 April, Pukul 22:11.. Mahasiswa yang belum mendapatkan kelompok setelah tanggal ini akan diacak kelompoknya.
5. Anda harus membuat aplikasi dan program ini sendiri kecuali library regex, tetapi belajar dari contoh-contoh program serupa yang sudah ada tidak dilarang (tidak boleh melakukan plagiasi source code dari program orang lain). Program harus dibuat sendiri, tidak boleh sama dengan teman.
6. Program harus modular dan mengandung komentar yang jelas.
7. Dilarang menggunakan kode program yang diunduh dari Internet. Mahasiswa harus membuat program sendiri, tetapi belajar dari program yang sudah ada tidak dilarang.
8. Batas akhir pengumpulan tugas adalah 5 Mei, Pukul 23:59.. Keterlambatan dalam mengumpulkan akan diberi penalti pengurangan skor yang cukup signifikan.
9. Semua pertanyaan menyangkut tugas ini dapat dikomunikasikan lewat QnA yang bisa diakses pada bit.ly/TugasStimaQnA
10. Bonus (maksimal 10 poin):
 - a. Mendeploy aplikasi web yang telah dibangun (hosting provider dibebaskan). Deployment website harus dipertahankan sampai demo tugas besar.
 - b. Setiap kelompok membuat video aplikasi yang mereka buat kemudian mengunggahnya ke Youtube. Video yang dibuat harus memiliki audio dan menampilkan wajah dari setiap anggota kelompok. Pada waktu demo aplikasi di

depan asisten, mahasiswa mengakses video Youtube tersebut dan memutarnya di depan asisten. Beberapa contoh video tubes tahun-tahun sebelumnya dapat dilihat di YouTube dengan menggunakan kata kunci “Tubes Stima”, “Tugas besar stima”, “strategi algoritma”, dll.

11. Demo akan dilakukan, tunggu informasi lanjut setelah waktu penggerjaan tugas berakhir.
12. Setiap anggota kelompok harus memahami seluruh program, termasuk bagian yang bukan bagian mereka.
13. Program disimpan dalam folder Tubes3_NIM (jika menggunakan repository dapat mengubah nama repository-nya) dengan NIM merupakan NIM anggota terkecil. Berikut merupakan struktur dari isi folder tersebut.
 - a. Folder src berisi source code.
 - b. Folder doc berisi laporan tugas besar dengan format nama_kelompok.pdf
 - c. README selengkap mungkin (dapat menjelaskan cara menjalankan backend dan frontend di lokal). Referensi README dapat diakses pada <https://github.com/ritaly/README-cheatsheet> atau referensi lain yang serupa.

BAB II

LANDASAN TEORI

2.1 Algoritma KMP

Algoritma KMP adalah suatu teknik pencocokan string yang lebih cerdas dibandingkan dengan algoritma *brute force*. Dalam pencocokan string, algoritma KMP juga mencocokkan string dari kiri ke kanan, namun perbedaannya terletak pada cara men-shift pattern yang dicari. Algoritma KMP menggunakan teknik yang lebih cerdas dalam menentukan pergeseran pattern yang dibandingkan dengan algoritma *brute force* yang hanya melakukan pergeseran satu persatu.

Misalkan, dalam pencocokan string antara text T dan pattern P, jika terdapat perbedaan karakter di antara keduanya pada $T[i]$ dan $P[j]$, maka untuk mendapatkan pergeseran pattern yang optimal, akan dilakukan pergeseran sebesar panjang prefix terbesar dari $P[0..k]$ yang juga sama dengan suffix dari $P[1..k]$, di mana $k = j-1$.

Fungsi pinggiran (border function) adalah bagian penting dari algoritma KMP. Fungsi ini didefinisikan dengan parameter pattern P, posisi terjadinya ketidakcocokan pada pattern yang disimbolkan dengan j, dan k yang sama dengan $j-1$. Border function $b(k)$ digunakan untuk menghitung panjang prefix terbesar dari $P[0..k]$ yang juga merupakan suffix dari $P[1..k]$. Fungsi ini tidak dipanggil jika $j=0$, dan menghasilkan nilai 0 jika $j=1$. Fungsi pinggiran juga dapat disebut sebagai failure function.

Misalkan terdapat pattern P dengan nilai abcabe dan terjadi ketidakcocokan pada karakter e di posisi $j=5$. Untuk menemukan pergeseran pattern yang optimal, maka perlu dilakukan perhitungan pada border function dengan $k=4$. Border function akan mencari ukuran prefix terbesar dari $P[0..4]$ yaitu abcab, yang juga sama dengan suffix dari $P[1..4]$ yaitu bcab. Dari hasil pencarian ini diperoleh nilai 2 dari ab, yang merupakan prefix terbesar dari abcab dan juga suffix dari bcab. Dalam pencarian selanjutnya, nilai j akan diubah menjadi nilai $b(k)$ yaitu 2. Dengan demikian, pada pencocokan pattern selanjutnya, karakter ab akan dilewatkan karena sudah terjamin sama. Algoritma KMP akan berhenti jika berhasil mencapai indeks terakhir dari P tanpa terjadi ketidakcocokan yang berarti berhasil menemukan kecocokan antara P dan T. Namun, jika telah mencapai akhir dari T tanpa menemukan kecocokan, maka dapat disimpulkan bahwa tidak ada kecocokan antara P dan T.

2.2 Algoritma BM

Algoritma Boyer-Moore (BM) merupakan algoritma yang digunakan untuk melakukan pencocokan string. Algoritma ini memanfaatkan dua teknik, yaitu *the looking-glass technique* dan *the character-jump technique*. *The looking-glass technique* dilakukan dengan cara memeriksa tiap karakter di pattern P dari belakang ke depan untuk mencocokkannya dengan teks T. Sementara itu, *the character-jump technique* digunakan untuk menangani jika terjadi

ketidakcocokan pada karakter tertentu di teks T. Teknik ini mengubah nilai indeks untuk membuat seolah-olah pattern P bergeser agar dapat cocok dengan teks T.

Dalam the character-jump technique, diterapkan sebuah metode yang memanfaatkan last occurrence function $L(x)$ yang menerima karakter x sebagai argumen dan berfungsi untuk menghitung indeks terakhir x muncul di dalam teks T. Fungsi ini kemudian digunakan untuk melakukan perhitungan dan penyesuaian pada indeks ketika terjadi ketidakcocokan antara karakter di teks T dengan pattern P. Fungsi $L(x)$ akan menghasilkan nilai -1 jika karakter x tidak ada di dalam pattern P.

Dalam karakter-jump technique, terdapat 3 skenario yang mungkin terjadi saat terjadi ketidakcocokan antara karakter pada teks T dan pattern P. Skenario pertama terjadi jika karakter x pada T juga ada pada P namun posisinya lebih kecil dari posisi karakter yang sedang dibandingkan pada P. Skenario kedua terjadi jika karakter x pada T juga ada pada P namun posisinya lebih besar dari posisi karakter yang sedang dibandingkan pada P. Skenario ketiga terjadi jika karakter x pada T tidak ada pada P. Setiap skenario akan memerlukan penyesuaian posisi P, yang dapat dilakukan dengan menggeser P ke kanan sebanyak 1 kali pada skenario kedua dan ke posisi $P[0]$ pada skenario ketiga.

Algoritma BM menggunakan teknik the looking-glass untuk memeriksa kecocokan pola P pada teks T dengan memeriksa karakter-karakter di P dari belakang ke depan. Teknik the character-jump digunakan ketika terjadi ketidakcocokan pada karakter $T[i]$ dan $P[j]$, di mana biasanya nilai i akan diubah agar seolah-olah P digeser ke kanan. Perlu diingat bahwa dalam algoritma BM, pencocokan pola P dimulai dari indeks terakhir menurut teknik looking-glass. Jika pencocokan berhasil mencapai indeks awal P tanpa terjadi ketidakcocokan, maka dianggap telah ditemukan kecocokan antara P dan T. Namun jika pencocokan tidak berhasil mencapai akhir teks T, maka dianggap tidak ada kecocokan antara P dan T.

2.3 Regex

Regex atau regular expression adalah sebuah pola pencarian yang terdiri dari kumpulan karakter yang digunakan untuk mencocokkan atau memvalidasi input teks dalam algoritma string matching dan aplikasi lainnya. Regex merupakan sebuah teknik yang berasal dari teori bahasa formal, dan memiliki format penulisan yang telah ditentukan. Setiap karakter dalam regex dapat berupa karakter literal atau metakarakter yang memiliki arti khusus dalam regex. Sebagai contoh, karakter 'a' dalam regex adalah karakter literal yang merepresentasikan huruf a, sedangkan karakter '.' adalah karakter literal yang digunakan untuk mencocokkan karakter apa pun kecuali newline.

Dengan menggunakan regex, seseorang dapat membuat pola pencarian yang lebih fleksibel dan luas dalam mencari sebuah teks. Sebagai contoh, jika seseorang ingin mencari pattern kata "ITB" dalam sebuah teks tanpa memperdulikan huruf kapital, dapat menggunakan regex '/itb/i'. Penggunaan regex ini jauh lebih efisien dibandingkan dengan metode exact string matching, karena tidak perlu mencoba banyak pattern dari tiap variasi penulisan.

2.4 Website

Suatu website memiliki dua bagian yang perlu dikembangkan, yaitu frontend dan backend. Frontend merupakan tampilan dari website itu sendiri yang mencakup semua komponen *user interface*, interaktivitas dengan pengguna, serta desain dan estetika dari *user interface* tadi. Pengembangan frontend dari suatu website pada umumnya berbasis HTML untuk penataan komponen, CSS untuk desain dan estetika komponen, dan Javascript untuk interaktivitas dari komponen tersebut. Akan tetapi, pada masa kini sudah terdapat banyak kakas yang membantu dan mempermudah pengembangan web sehingga pengembangan frontend website dapat lebih cepat dan efisien, contoh dari kakas/framework tersebut adalah React, Angular, Svelte, dan lain-lain.

Bagian backend dari suatu website mencakup logika bisnis, penyimpanan data, dan logika dan sistem lain yang tidak ditampilkan langsung oleh user interface. Backend dari sebuah website berkomunikasi dengan frontend website melalui http request terhadap API endpoint yang diekspos oleh server backend tersebut. Request yang diterima oleh backend tersebut akan diproses sehingga dapat mengembalikan data, mengubah data, atau melakukan pemrosesan lainnya sehingga dapat diterima dan ditampilkan kembali oleh frontend. Framework backend yang umum digunakan adalah Express.js, Flask, dan lain-lain.

Penyimpanan data juga dapat dilakukan dengan memanfaatkan berbagai jenis database yang memiliki sifat, keunggulan, dan kekurangan masing-masing. Beberapa jenis basis data yang sering digunakan untuk keperluan web adalah PostgreSQL, MongoDB, dan lain-lain.

Setelah suatu website dikembangkan, website tersebut perlu di-deploy agar semua orang dapat mengakses website tersebut melalui internet terbuka. Oleh karena itu, diperlukan layanan yang dapat menjalankan dan meng-hosting aplikasi dan server website yang telah dibuat. Beberapa layanan yang umum digunakan adalah vercel, heroku, railway, dan lain-lain

2.5 Levenshtein Distance

Levenshtein Distance adalah sebuah metode yang digunakan untuk mengukur seberapa jauh dua buah string dalam segi kesamaannya. Berikut adalah langkah-langkah yang dapat dilakukan untuk mencari Levenshtein Distance:

1. Tentukan dua buah string yang akan dihitung jarak kesamaannya.
2. Buat sebuah matriks dengan ukuran $(m + 1) \times (n + 1)$, dengan m dan n masing-masing merupakan panjang string pertama dan kedua. Matriks ini akan digunakan untuk menyimpan hasil dari perhitungan jarak kesamaan antara string pertama dan kedua.
3. Inisialisasi elemen matriks pada baris dan kolom pertama dengan nilai 0 hingga matriks terisi penuh.
4. Iterasi baris dan kolom pada matriks, dimulai dari baris kedua dan kolom kedua. Pada setiap elemen matriks, hitung jarak kesamaannya dengan membandingkan karakter pada string pertama dan kedua pada indeks yang sama. Jika kedua karakter tidak sama, ambil nilai minimum dari tiga elemen sebelumnya (diagonal kiri atas, atas, dan diagonal kiri) pada matriks, dan tambahkan 1 sebagai jarak kesamaannya. Jika kedua karakter sama, ambil nilai diagonal kiri atas pada matriks sebagai jarak kesamaannya.
5. Setelah selesai iterasi, nilai elemen pada posisi terakhir pada matriks merupakan jarak kesamaan antara dua buah string.

Jarak kesamaan antara dua string juga dapat dihitung dengan menggunakan rumus berikut:

$$LD(a, b) = \max(|a|, |b|) - LCS(a, b)$$

Dengan:

a : String pertama

b : String kedua

$|a|$: Panjang string pertama

$|b|$: Panjang string kedua

$LCS(a, b)$: Panjang subsequence terpanjang dari kedua string

$LCS(a, b)$ dapat dihitung dengan menggunakan algoritma dynamic programming yang serupa dengan algoritma Levenshtein Distance, namun dengan menghitung nilai elemen pada diagonal kiri atas pada matriks.

BAB III

ANALISIS PEMECAHAN MASALAH

3.1 Langkah Penyelesaian Masalah

3.1.1 Fitur Tanggal

Pada aplikasi yang telah dibuat, fitur tanggal dapat diakses dengan memasukkan tanggal dalam format DD/MM/YYYY. Program akan otomatis mendeteksi apabila input bermaksud untuk mengakses fitur tanggal melalui pattern matching oleh Javascript Regex yang didefinisikan sebagai berikut.

```
/(?<! [+-*\^.\d])\b\d+\d+\d+\b(?! [+-*\^(\)])/
```

Berikut adalah penjelasan tiap bagian dari regex tersebut:

- **(?<! [+-*\^.\d])** : Adalah *negative lookbehind assertion* untuk memastikan tidak ada operator matematika yang mendahului string. Hal ini ditujukan untuk menghindari pencocokan tanggal yang merupakan bagian dari ekspresi matematika yang lebih besar, seperti "2/2/2022" pada persamaan "1+2/2/2022".
- **\b\d+\d+\d+\b** : Adalah pola utama pada fitur tanggal, yakni tiga bilangan bulat terpisah oleh *slash*.
- **(?! [+-*\^(\)])** : Adalah *negative lookahead assertion* untuk memastikan tidak ada operator matematika yang mengakhiri string. Sama seperti alasan sebelumnya, hal ini dilakukan untuk menghindari penangkapan pola tanggal pada persamaan matematika yang lebih besar.

Setelah klasifikasi dilakukan, potongan string yang cocok pada pattern regex kemudian divalidasi dan diproses untuk dicari nama hari yang berkorespondensi dengan tanggal tersebut. Validasi dilakukan layaknya tanggal seperti biasa, yakni memastikan bahwa bulan berada pada rentang nilai 1-12, serta tanggal pada rentang 1-31 untuk bulan Januari, Maret, Mei, Juli, Agustus, Oktober, dan Desember, rentang 1-30 untuk bulan April, Juni, September, dan November, dan khusus untuk bulan Februari memiliki rentang 1-29 pada tahun kabisat dan 1-28 pada tahun lainnya. Terakhir, untuk mengevaluasi hari pada tanggal yang diberikan, digunakan kelas bawaan Date dari Javascript untuk mentransformasikan tanggal menjadi bentuk string dengan format "<Bulan> <Tanggal>, <Tahun> falls on a <Hari>".

3.1.2 Fitur Persamaan Matematika

Selanjutnya, dilakukan pemrosesan terhadap persamaan matematika. Input akan otomatis terklasifikasi sebagai persamaan matematika apabila cocok dengan pola regex berikut.

```
\-?\(*\s*(-?\d+(\.\d+)?)\s*(\(*\s*[+\-\^]\s*((-\?\d+(\.\d+)?))|)+\(+\)*)+/;
```

Berikut adalah penjelasan tiap bagian dari regex tersebut:

- `\-?\(*\s*(-?\d+(\.\d+)?)\s*` : Merupakan awalan dari persamaan matematika, yakni bisa diawali dengan simbol negatif, kurung buka, ataupun angka.
- `\(*\s*[+\-\^]\s*` : Merupakan bagian tengah dari persamaan matematika, yakni berupa operator valid untuk operasi biner
- `((-\?\d+(\.\d+)?))|)+\(+\)*` : Merupakan bagian akhir dari persamaan matematika, yakni angka (termasuk angka negatif), ataupun repetisi kurung tutup
- Gabungan dari bagian `\(*\s*[+\-\^]\s*` dan `((-\?\d+(\.\d+)?))|)+\(+\)*` dapat berepetisi lebih dari satu kali sehingga dikumpulkan dan diberi klosur dalam *capturing group*
`(\(*\s*[+\-\^]\s*((-\?\d+(\.\d+)?))|)+\(+\)*+`

Setelah ditemukan pola yang cocok dengan regex tersebut dalam string, dilakukan pemrosesan berupa validasi dari persamaan matematika. Evaluasi dilakukan dengan pengecekan keterseimbangan tanda kurung, kemudian memastikan apabila tiap angka pada persamaan diikuti oleh operator dan operator diikuti oleh angka atau tanda kurung.

Terakhir, dilakukan evaluasi dengan *Shunting Yard Algorithm*. Algoritma *Shunting Yard* merupakan sebuah metode untuk *parsing* persamaan matematika dalam notasi infix menjadi bentuk *Reverse Polish Notation* (RPN) atau biasa dikenal dengan notasi postfix, yang kemudian dapat dievaluasi dengan struktur data *stack*. Algoritma ini secara dasarnya bekerja dengan memindai string persamaan dari kiri ke kanan, kemudian mem-*push* operator pada *stack* atau *output queue* berdasarkan presedens dan aturan asosiatif. Cara selengkapnya adalah sebagai berikut.

1. Definisikan operator dan prioritas dari tiap operator tersebut.
2. Hilangkan *whitespace* dari string persamaan.
3. Pisahkan persamaan menjadi *array of tokens* dengan pemisah berupa operator matematika, yakni (+, -, *, /, ^) atau tanda kurung.
4. Jika tanda minus (-) terdeteksi dan simbol tersebut adalah token pertama dalam ekspresi atau segera mengikuti operator lain, tanda minus akan diperlakukan sebagai *unary minus* dan digabungkan dengan token berikutnya untuk membentuk angka negatif. Hal ini dilakukan dengan memeriksa token sebelumnya dan berikutnya dalam array dan menggabungkannya apabila diperlukan.

5. Filter *array of tokens* untuk menghilangkan token-token yang kosong
6. Lakukan loop untuk iterasi tiap token
7. Apabila token adalah angka, masukkan ke *output queue*
8. Apabila token adalah operator, *push* ke dalam *stack* operator. Apabila *top* dari *stack* memiliki prioritas yang lebih tinggi dari operator pada token, maka *stack* akan dilakukan *pop* terlebih dahulu sampai ditemui operator yang lebih rendah preseden nya atau dicapai *bottom* dari *stack*.
9. Apabila token merupakan tanda kurung buka, tambahkan pada *stack* operator.
10. Apabila token merupakan tanda kurung tutup, semua operator akan di-*pop* dari *stack* operator dan ditambahkan ke *output queue* hingga tanda kurung tutup ditemukan. Kurung buka kemudian akan di-*pop* dari *stack*.
11. Apabila seluruh token telah diproses namun masih ada operator tersisa, maka seluruh operator akan dimasukkan pada *output queue*.
12. *Output queue* akan dievaluasi menggunakan *stack based approach*, yakni:
 - a. Apabila token merupakan angka, *push* ke *stack operand*
 - b. Apabila token merupakan operator, dua elemen teratas pada *stack operand* akan di-*pop* dan dievaluasi berdasarkan operator pada token. Hasil evaluasi tersebut kemudian di-*push* kembali pada *stack*.
13. Hasil akhir merupakan elemen terakhir yang tersisa pada *stack*. Kembalikan nilai tersebut.

Sebelum hasil evaluasi ditampilkan, terdapat beberapa kondisi yang perlu diperiksa:

- Apabila hasil adalah Infinity atau -Infinity, sangat mungkin terjadi pembagian dengan nol pada ekspresi matematika. Tampilkan keluaran ‘*Undefined*’ sebagai hasil evaluasi.
- Apabila hasil adalah NaN, mungkin terjadi bentuk ekspresi matematika yang tidak *disupport* oleh parser. Tampilkan keluaran *request* untuk meminta pengguna memformat kembali persamaan yang diberi (Memasukkan negatif ke ekspresi dalam tanda kurung, menambahkan operator perkalian untuk perkalian antar kurung, cek saltik).
- Apabila persamaan tidak lolos validasi, tampilkan pesan kesalahan.

Jika persamaan melewati kondisi yang telah dirincikan, maka persamaan benar dan akan ditampilkan hasil evaluasi dari ekspresi matematika tersebut.

3.1.3 Fitur Tambah Pertanyaan

Pengguna dapat mengakses fitur menambah pertanyaan ke database dengan memasukkan pola <Tambahkan pertanyaan "{question}" dengan jawaban "{answer}"> dan menyesuaikannya dengan pertanyaan dan jawaban yang ingin disimpan. Sebuah input akan terklasifikasi ingin mengakses fitur tambah pertanyaan apabila pola keyword berikut ditemukan pada string *query*.

```
/tambah(kan)?\s*pertanyaan/i
```

Input kemudian akan dicek kembali apabila telah mengikuti format yang telah dispesifikan sebelumnya dengan regex berikut.

```
/tambah(kan)?\s*pertanyaan\s*"([^\"]+)"\s*dengan\s*jawaban\s*"([^\"]+)"/i
```

Berikut penjelasan dari regex tersebut:

- `/tambah(kan)?\s*pertanyaan\s*` : Merupakan bagian awal dari kata kunci menambahkan pertanyaan, dengan akhiran ‘kan’ pada kata ‘tambahkan’ bersifat opsional.
- `\s*dengan\s*jawaban\s*` : Merupakan bagian tengah dari perintah menambahkan pertanyaan. Cukup *straightforward* dan wajib ada pada *query*.
- `"([^\"]+)"` : Merupakan bagian pertanyaan atau jawaban yang ingin ditambah ke basis data. Pertanyaan dan jawaban wajib tidak memiliki tanda kutip di dalamnya untuk menghindari kesalahan batasan pertanyaan dan jawaban. Apabila menyalahi aturan tersebut, akan ditampilkan permohonan untuk mengganti format pertanyaan dan jawaban.

Selanjutnya, dilakukan validasi terhadap pertanyaan dan jawaban yang akan dimasukkan ke basis data, yakni tidak boleh mengandung karakter *new line* ‘\n’ sebab *new line* digunakan sebagai pemisah antar fitur, serta tidak boleh memiliki konten yang beririsan dengan fitur spesial lainnya pada chatbot (persamaan matematika, format tanggal, keyword menambahkan/menghapus pertanyaan) untuk menghindari ambiguitas.

Terakhir, dilakukan operasi terhadap *database*. Pertama, akan dilakukan pencarian terhadap pertanyaan yang sama pada basis data. Apabila ditemui, jawaban terhadap pertanyaan tersebut akan diperbarui berdasarkan input terbaru. Jika pertanyaan tersebut baru, pasangan pertanyaan dan jawaban akan ditambah ke *database*.

3.1.4 Fitur Hapus Pertanyaan

Fitur menghapus pertanyaan bekerja hampir sama seperti menambah pertanyaan. Input pertama akan diklasifikasi ingin mengakses fitur ini apabila terdapat kecocokan terhadap pola regex berikut.

```
/hapus\s*pertanyaan\s*"([^\"]+)"/i
```

Pertanyaan yang dihapus kemudian akan dicari *exact match* pada basis data. Apabila tidak ditemukan, akan diberikan pesan kesalahan. Sebaliknya, apabila ditemukan, akan ditampilkan pesan bahwa operasi penghapusan telah sukses.

3.1.5 Fitur Pertanyaan dari *Database*

Fitur ini merupakan fungsionalitas utama dari aplikasi chatbot, yakni mampu menjawab pertanyaan yang telah terdefinisi oleh perintah tambah pertanyaan pada basis data. Fitur ini hanya bisa diakses apabila input tidak terkласifikasi sebagai salah satu dari fitur spesial yang telah dideskripsikan sebelumnya. Cara kerja fitur ini adalah sebagai berikut:

1. Menerima input string
2. Fetch seluruh pertanyaan dari database.
3. Iterasi seluruh pertanyaan, bandingkan dengan algoritma KMP atau BM berdasarkan pilihan pengguna pada *dropdown* tampilan.
4. Apabila ditemukan *exact match*, kembalikan jawaban dari pertanyaan tersebut.
5. Apabila tidak ditemukan, hitung jarak *levenshtein* untuk seluruh pertanyaan lalu filter pertanyaan yang memiliki tingkat kemiripan di atas 50%.
6. Urutkan pertanyaan berdasarkan jarak *levenshtein*. Apabila pertanyaan termirip memiliki tingkat kemiripan 90% ke atas, maka kembalikan jawaban dari pertanyaan tersebut.
7. Apabila tidak, tampilkan maksimal tiga pertanyaan termirip yang tersisa pada daftar pertanyaan.

3.1.6 Pengaksesan Beberapa Fitur Bersamaan

Untuk mengatasi input pengguna yang ingin mengakses beberapa fitur bersamaan, dilakukan *enforcing* aturan masukan berupa pemisahan dengan karakter new line '\n' melalui *keybinding* shift+enter pada *text box* antarmuka. Dengan ini, pengguna dapat mengakses keempat fitur tanpa ambiguitas.

Pemrosesan input multi fitur, dilakukan dengan tokenisasi input berdasarkan pemisah karakter *new line*, kemudian dilakukan pemrosesan untuk seluruh token secara terpisah. Secara umum, pemrosesan mencakup klasifikasi fitur yang ingin diakses token, *handling* tiap fitur yang terakses, dan terakhir menambahkan jawaban ke *array* jawaban yang akan digabung di akhir.

3.2 Arsitektur Aplikasi Web

3.2.1 Client Side

Client side atau *frontend* dari website yang kami kembangkan memanfaatkan framework React untuk membangun tampilan, desain, dan interaktivitas tampilan sisi pengguna. Kami menggunakan React untuk mempercepat dan mempermudah pengembangan *frontend*, serta karena terdapat banyak tools yang dapat digunakan dengan react sehingga dapat menambah fitur-fitur pada tampilan pengguna.

3.2.2 Server Side

Server side dari aplikasi web kami menggunakan Express.js yang merupakan framework Javascript yang berbasis node yang umum digunakan untuk membuat sebuah server aplikasi web. Aplikasi backend berbasis Express.js ini menerima request dari *frontend*, mengolah request tersebut sehingga menerima, mengubah, atau menghilangkan data dari database, dan akhirnya mengembalikan informasi kepada *frontend* untuk ditampilkan. Fungsionalitas dan logika dari pemanfaatan string matching dan regex diletakkan pada backend karena perlu adanya interaksi dengan database.

3.2.3 Database

Database yang digunakan dalam aplikasi ini adalah MongoDB yang sudah di host dan deploy dengan MongoDB atlas. MongoDB merupakan program basis data yang terstruktur dalam bentuk Collections dan Documents sehingga terstruktur seperti objek-objek yang terletak dalam suatu array. Query yang dilakukan untuk mengakses, mengubah, dan menulis data ke MongoDB bersifat NoSQL sehingga dapat lebih mudah digunakan. Penggunaan MongoDB juga memudahkan pengembangan karena basis data yang digunakan merupakan basis data yang sudah ter-deploy oleh MongoDB Atlas sehingga kami tidak memerlukan setup dan pengaturan tambahan untuk mendeploy basis data kami.

3.2.4 Deployment

Deployment merupakan tahap akhir pada pengembangan web yang bertujuan agar website yang sudah dikembangkan dapat diakses menggunakan internet. Website kami di deploy menggunakan Vercel dan Railway. Vercel digunakan untuk men-deploy *frontend*. *Frontend* kami juga menggunakan framework Next.Js agar memudahkan deployment menggunakan vercel. Backend kami di deploy menggunakan railway karena railway menyediakan trial gratis per bulan untuk mendeploy suatu service, contohnya suatu server backend. Untuk basis data, sudah dijelaskan sebelumnya bahwa database MongoDB aplikasi kami di deploy oleh MongoDB Atlas.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Spesifikasi Teknis Program

Program pertama kali diinisiasi ketika pengguna memasukkan query baru pada chatbot. Request yang disubmisi pada antarmuka kemudian akan *dipassing* ke sisi *backend* dan akan dilakukan pemrosesan selanjutnya. Secara end-to-end, pemrosesan dilakukan mayoritas oleh fungsi `inputHandler` yang akan mengembalikan respons berupa array jawaban yang perlu ditampilkan pada antarmuka. Berikut langkah rinci pemrosesan yang dilakukan.

1. Inisialisasi senarai (*list*) untuk menyimpan kandidat jawaban.
2. Cek apabila input ingin mengakses beberapa fitur. Jika ya, input harus memiliki pemisah berupa *newline character*.
3. Tokenisasi input berdasarkan *newline*. Tahap ini didukung oleh fungsi `tokenizeString` yang mengembalikan list dari token berdasarkan pemisahan terhadap karakter *newline*.
4. Iterasi setiap token.
5. Klasifikasi fitur yang ingin diakses token. Tahap ini didukung oleh fungsi `classifyInput` yang akan mengembalikan *list* berisi *flag* biner. List tersebut berisi empat buah angka nol atau satu, dengan satu menandai fitur ingin diakses. Indeks pertama pada list tersebut menandai fitur tambah pertanyaan, kedua menandai fitur hapus pertanyaan, ketiga untuk fitur tanggal dan keempat untuk fitur kalkulator. Apabila seluruh fitur memiliki *flag* set ke nol, maka akan dicari pertanyaan yang sesuai pada *database*.
6. Cek apabila input ingin mengakses fitur tambah atau hapus pertanyaan dari database namun tidak sesuai dengan format. Hal ini sebab input tidak akan terklasifikasi dengan baik apabila tidak mengikuti format. Sehingga, akan dikembalikan pesan kesalahan apabila hal ini terjadi.
7. *Handle* input dengan array klasifikasi yang memiliki *flag* seluruhnya nol. Sebagaimana dijelaskan sebelumnya, hal ini berarti harus dilakukan pencarian pertanyaan pada *database*. Fungsi pendukung tahap ini adalah `getAnswer` yang merupakan controller untuk fetch jawaban dari *database* apabila ditemukan. Proses penyocokan mengikuti penjelasan yang telah dideskripsikan pada bagian III. Hasil dari fitur ini kemudian dimasukkan pada *array* jawaban.
8. *Handle* setiap fitur dengan *flag* set ke true (1). Pemrosesan tiap fitur dilakukan sebagaimana pada bagian III. Fungsi dan prosedur pendukung untuk tiap fitur antara lain:
 - a. `parseAddQuestionRegex` untuk menarik konten pertanyaan dan jawaban yang ingin ditambah dari perintah tambah pertanyaan.
 - b. `addQuestion` merupakan controller untuk menambah pertanyaan pada basis data.

- c. `parseDeleteQuestionRegex` untuk menarik konten pertanyaan yang ingin dihapus dari perintah menghapus pertanyaan.
 - d. `deleteQuestion` merupakan controller untuk menghapus pertanyaan pada basis data.
 - e. `validateDate` untuk validasi tanggal masukan sebagaimana dijelaskan pada bagian III
 - f. `formatDate` untuk memformat keluaran hasil fitur tanggal
 - g. `validateMathExpression` untuk validasi persamaan matematika sebagaimana dijelaskan pada bagian III
 - h. `evaluateExpression` untuk mengevaluasi persamaan matematika dengan notasi infix
9. Apabila *array* jawaban masih kosong, kembalikan pesan bahwa input tidak dapat diproses
 10. Kembalikan array kumpulan jawaban valid.
 11. *Passing* data *array* jawaban ke sisi *frontend* untuk ditampilkan ke pengguna.

Adapun struktur data yang digunakan dalam tahap pemrosesan antara lain:

1. *List*, untuk menyimpan kumpulan jawaban dan token
2. *Stack*, untuk evaluasi persamaan matematika
3. *Queue*, untuk menyimpan hasil parsing infix ke postfix dari persamaan matematika

4.2 Tata Cara Penggunaan Program

Program kami dapat diakses/digunakan melalui dua cara, yaitu dengan membuka website kami yang sudah di deploy atau dengan menjalankan aplikasi frontend dan server backend. Website kami dapat diakses melalui link <https://crdgpt-rinaldy-adin.vercel.app>. Untuk menjalankan aplikasi kami dalam environment local, perlu dilakukan clone terhadap repository github frontend dan backend kami, yang terdapat pada lampiran laporan ini. Setelah kedua repository tersebut di-clone perlu dilakukan `npm i` pada kedua folder dan ditambahkan file `.env.local` dan `.env` pada root folder frontend dan backend, isi file `.env.local` dan `.env` adalah sebagai berikut,

.env.local (frontend)

```
NEXT_PUBLIC_BACKEND_API_URL="http://localhost:3080"
```

.env (backend)

```
MONGODB_URI="mongodb+srv://stima-local:SFHYBghV0lH9ZaxN@cluster0.9lnshi2.mongodb.net"
```

```
t/crdgpt?retryWrites=true&w=majority"
PORT=3080
```

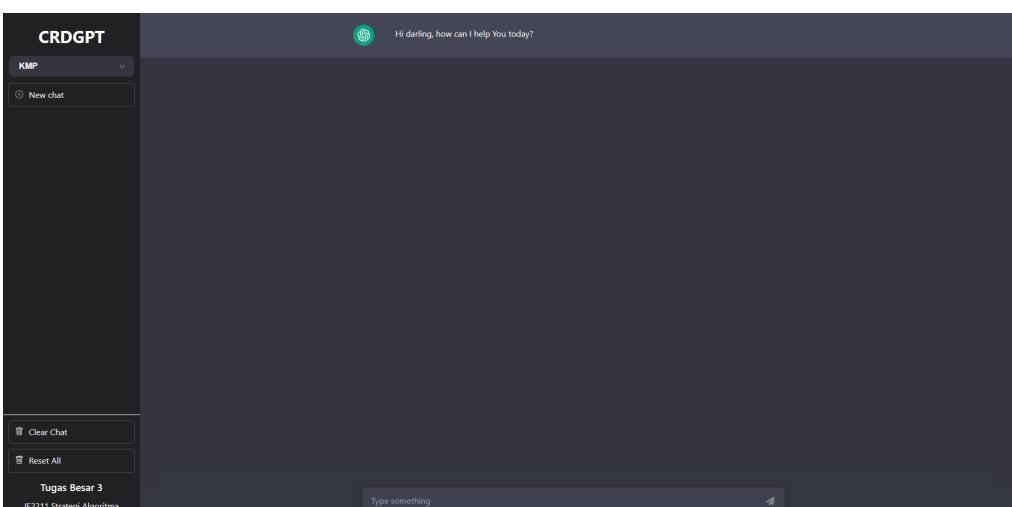
Setelah kedua file tersebut sudah ditambahkan, frontend dapat dijalankan dengan menjalankan `npm run dev` dan backend dapat dijalankan dengan `npm run serverstart`.

Setelah program dijalankan, web frontend dapat dibuka dengan mengikuti link yang terdapat pada terminal output frontend (umumnya <http://localhost:3000>). Setelah dibuka, pengguna dapat mengirim chat ke aplikasi, menambah chat session baru, atau menghapus chat session yang sudah ada.

4.3 Hasil dan analisis pengujian

1. Test Case 1

Test Case 1 - Kondisi Awal Website

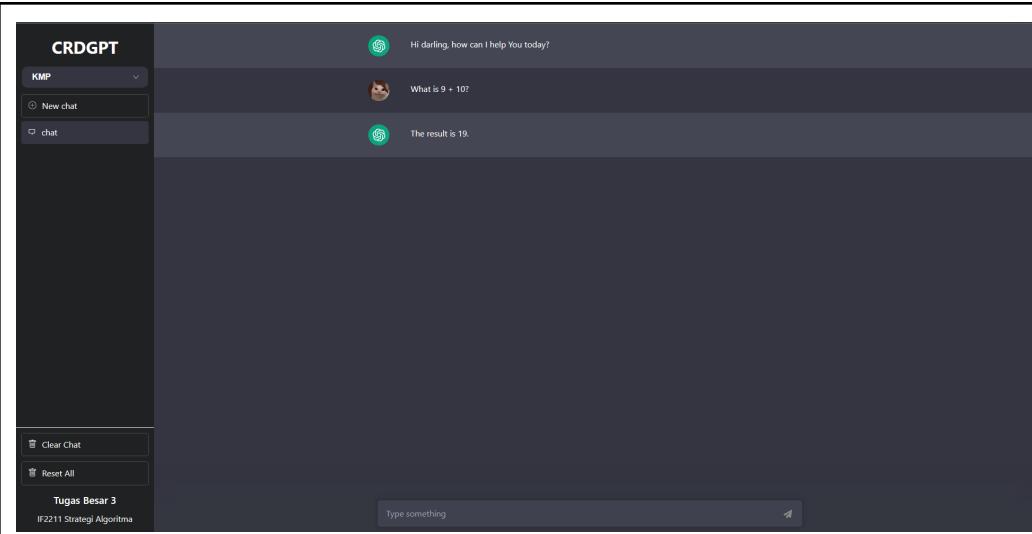


The screenshot shows the CRDGPT web application. On the left, there is a sidebar titled 'CRDGPT' with a dropdown menu set to 'KMP'. Below the dropdown is a button labeled 'New chat'. At the bottom of the sidebar are two buttons: 'Clear Chat' and 'Reset All'. A small note at the bottom reads 'Tugas Besar 3 IF2211 Strategi Algoritma'. The main area of the page has a dark background. In the top right corner, there is a green circular icon with a white symbol and the text 'Hi darling, how can I help You today?'. Below this, there is a text input field with the placeholder 'Type something' and a magnifying glass icon. The overall interface is minimalist and modern.

Ket : -

2. Test Case 2

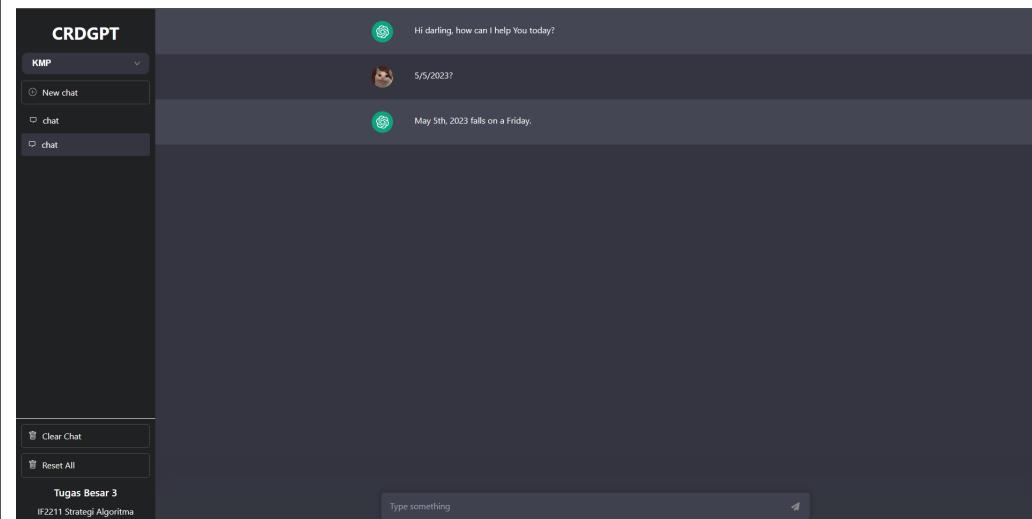
Test Case 2 - Simple Math Prompt



Ket : Program mengembalikan jawaban ekspresi matematika dengan benar

3. Test Case 3

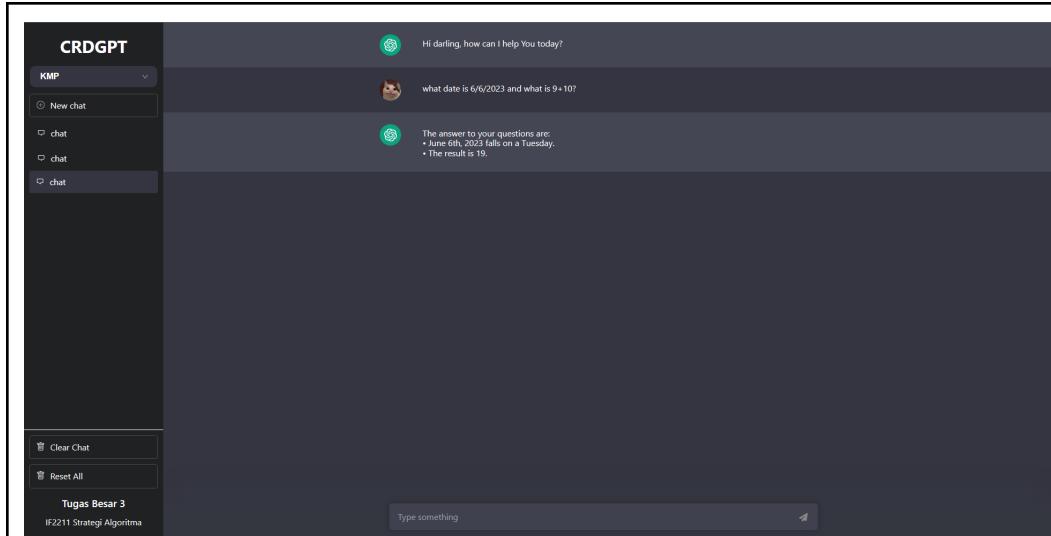
Test Case 3 - Date Prompt



Ket : Program mengembalikan jawaban hari dari tanggal dengan benar

4. Test Case 4

Test Case 4 - Simple Math & Date Prompt



Ket : Berhasil menggunakan fitur Kalkulator dan Date pada satu Query

5. Test Case 5

Test Case 5

Query :

1. 5/5/2023
2. what date does 17/8/1945 fall on?
3. what day does 19/5/2010 fall on?

The screenshot shows a dark-themed chat interface titled "CRDGPT". On the left, there's a sidebar with a dropdown menu set to "KMP" and buttons for "New chat", "chat", "chat", "Clear Chat", and "Reset All". Below this is a section for "Tugas Besar 3" and "IF2211 Strategi Algoritma". The main area is a conversation log:

- A message from the AI: "Please include a new line to access multiple features.
Note: Accessing dates might require a new line to differ from math expressions."
- A message from the user: "5/5/2023"
- A message from the AI: "May 5th, 2023 falls on a Friday."
- A message from the user: "what date does 17/8/1945 fall on?"
- A message from the AI: "August 17th, 1945 falls on a Friday."
- A message from the user: "what day does 19/5/2010 fall on?"
- A message from the AI: "Please include a new line to access multiple features.
Note: Accessing date feature requires a new line to differ from math expressions."
- A message from the user: "what day does 19/5/2010 fall on?"
- A message from the AI: "May 19th, 2010 falls on a Wednesday."

In the bottom right corner of the main area, there's a text input field with the placeholder "Type something" and a send icon.

Ket : Program berhasil menjawab pertanyaan mengenai tanggal dengan benar

6. Test Case 6

Test Case 6 - Tambah dan Hapus Pertanyaan

The screenshot shows a GPT-based chat interface titled "CRDGPT". On the left, there is a sidebar with a dropdown menu set to "KMP" and a list of "chat" entries. Below the sidebar are buttons for "Clear Chat" and "Reset All", followed by the text "Tugas Besar 3" and "IF2211 Strategi Algoritma".

The main area displays a conversation:

- User: Hi darling, how can i help You today?
- Bot: Tambahkan pertanyaan "x" dengan jawaban "y"
- Bot: Question "y" added successfully.
- User: x
- Bot: y
- User: Hapus pertanyaan "x"
- Bot: Question "x" successfully deleted.
- User: x
- Bot: Could not find question in the database.
- User: Hari apa 5/5/2023?
- Bot: May 5th, 2023 falls on a Friday.

A text input field at the bottom right contains the placeholder "Type something".

Ket : Program berhasil menambah dan menghapus pertanyaan

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Algoritma string matching dan regex telah diimplementasikan dan berhasil digunakan untuk mendeteksi perintah dari masukan user untuk chatbot. Selain itu, regex juga berhasil digunakan untuk mengekstraksi informasi dari masukan user. Semua kode yang dihasilkan dapat terhubung dengan GUI dengan baik, sehingga tercipta sebuah aplikasi yang memenuhi semua spesifikasi tugas yang diberikan.

5.2 Saran

Saran yang dapat kami berikan setelah mengerjakan tugas besar ini adalah

- Mulai mengerjakan tugas besar dari jauh hari, sehingga tidak berdekatan oleh deadline tubes lainnya
- Coba untuk eksplorasi berbagai library dan platform terlebih dahulu yang dapat memudahkan pembuatan program.

5.3 Refleksi

Materi yang dipelajari dalam mata kuliah Strategi Algoritma telah membantu dalam pembuatan algoritma string matching dan penggunaan regex pada program ini menjadi lebih mudah. Meskipun JavaScript bukan bahasa pemrograman yang sulit, namun cukup efektif digunakan dalam pembuatan program ini. Selain itu, penggunaan React dan MongoDB juga membantu dalam memudahkan pembuatan program.

DAFTAR PUSTAKA

- <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf>
- <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf>
- <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2019-2020/Modul-Praktikum-NLP-Regex.pdf>

LAMPIRAN

Link Website : <https://crdgpt-rinaldy-adin.vercel.app>

Repository FE : https://github.com/Rinaldy-Adin/Tubes3_13521104_FE.git

Repository BE : https://github.com/Rinaldy-Adin/Tubes3_13521104_BE.git

Link Video Youtube (Bonus) : <https://youtu.be/Rw9ze9E7Tqs>