

IF2211 Strategi Algoritma
**IMPLEMENTASI ALGORITMA KNUTH-MORRIS-PRATT (KMP)
DAN BOYER MOORE (BM) DALAM PEMBUATAN CHATBOT SEDERHANA**
Laporan Tugas Besar III

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



Oleh
Arsa Izdihar Islam 13521101
Vanessa Rebecca Wiyono 13521151
Alisha Listya Wardhani 13521171

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

DAFTAR ISI

BAB I	
DESKRIPSI MASALAH	3
BAB II	
LANDASAN TEORI	4
2.1. Algoritma Knuth–Morris–Pratt (KMP)	4
2.2 Algoritma Boyer-Moore (BM)	4
2.3 Regular Expression	5
2.4 Perhitungan Tingkat Kemiripan	6
2.5 Arsitektur Aplikasi Web yang Dibangun	6
BAB III	8
RANCANGAN PEMECAHAN MASALAH	8
3.1. Langkah Pemecahan Masalah	8
3.2. Pemodelan Algoritma Pencarian	10
3.3. Arsitektur Web	12
BAB IV	
ANALISIS PEMECAHAN MASALAH	13
4.1. Implementasi Program	13
4.2. Struktur Data Program	14
4.3. Pengujian dan Analisis Hasil pengujian Program	19
BAB V	
KESIMPULAN DAN SARAN	25
5.1. Kesimpulan	25
5.2. Saran	25
LAMPIRAN	27
Repository Github	27

BAB I

DESKRIPSI MASALAH

ChatBot merupakan sebuah program komputer yang dirancang untuk melakukan percakapan dengan manusia melalui media seperti pesan teks ataupun suara. ChatBot yang lebih canggih, seperti ChatGPT, memiliki kemampuan teknologi pemrosesan bahasa alami sehingga mampu memahami konteks dan menyusun jawaban sesuai dengan query pertanyaan yang diberikan. Dalam Tugas Besar III Strategi Algoritma, diberikan permasalahan untuk membuat ChatBot sederhana dengan pendekatan Question-Answering.

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.

Berikut merupakan fitur-fitur yang wajib diimplementasi pada ChatBot:

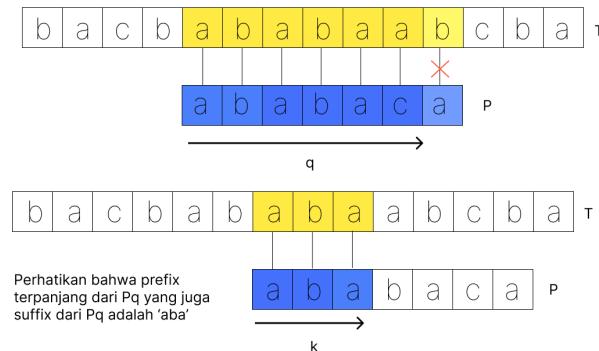
1. **Fitur pertanyaan teks (didapat dari database)**, mencocokkan pertanyaan dari input pengguna ke pertanyaan di database menggunakan algoritma KMP atau BM.
2. **Fitur kalkulator**, pengguna memasukkan input query berupa persamaan matematika. Contohnya adalah $5+9*(2+4)$. Operasi cukup Tambah, kurang, 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.

BAB II

LANDASAN TEORI

2.1. Algoritma Knuth–Morris–Pratt (KMP)

Algoritma Knuth–Morris–Pratt (KMP) merupakan algoritma *string matching* yang mencari kemunculan suatu pola atau upa-string dalam sebuah teks dari kiri-ke-kanan. Algoritma ini berperilaku seperti algoritma *brute-force*, tetapi dengan pemindaiyan pola yang lebih efektif daripada algoritma *brute-force*.



Pada algoritma KMP, dilakukan pembentukan tabel prefix (*failure function*). Tabel ini menyimpan informasi mengenai kemungkinan kemunculan pola pada suatu posisi pada teks, dengan mengidentifikasi kemungkinan prefiks terpanjang dari pola yang cocok. Dalam proses pencarian, algoritma ini memindai pola dari karakter pertama pada *string*. Jika terjadi ketidakcocokan pada karakter tertentu, algoritma memanfaatkan informasi yang telah dikumpulkan sebelumnya pada tabel prefiks untuk menentukan posisi berikutnya. Dengan demikian, jumlah pengulangan untuk pemindaiyan dapat diminimalisir. Proses ini berlanjut sampai seluruh teks selesai diproses.

2.2 Algoritma Boyer-Moore (BM)

Algoritma Boyer-Moore (BM) merupakan algoritma *string matching* yang menggunakan 2 teknik. Algoritma ini memproses pola yang dicari (pattern), bukan teks tempat pola tersebut dicari. Pada algoritma ini, terdapat 2 aturan utama dalam melakukan pergeseran pencarian pola.

A. Bad Character Heuristic

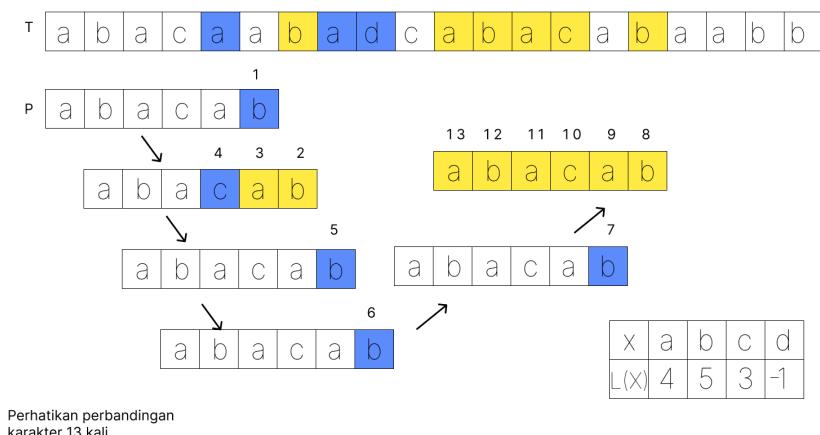
Aturan ini mempertimbangkan karakter dalam T dimana proses perbandingan gagal (dengan asumsi kegagalan terjadi). Kemunculan karakter tersebut di kiri P ditemukan,

dan terjadi pergeseran sehingga kemunculan karakter tersebut sejajar dengan karakter dimana proses perbandingan gagal. Jika karakter tersebut tidak muncul di kiri P, maka pergeseran terjadi sebesar P melewati tempat terjadinya kegagalan perbandingan.

B. Good Suffix Heuristic

Aturan ini menggunakan fitur perbandingan algoritma yang dimulai pada akhir pola dan berlanjut pada awal pola. Misalkan t adalah substring dari teks T yang dicocokan dengan substring dari *pattern* P. Penggeresan pattern dilakukan sehingga:

1. Kemunculan lain dari t dalam P cocok dengan t dalam T
2. Awalan P, yang cocok dengan akhiran t
3. P bergerak melewati t, karena tidak ada pattern yang ditemukan dalam kemunculan sebelumnya.



2.3 Regular Expression

Regular Expression (RegEx) merupakan pola yang digunakan untuk mencocokkan kombinasi karakter dalam suatu *string*. Dalam JavaScript, regular expression digunakan dengan metode `exec()` dan `test()` dari RegExp dan metode `match()`, `matchAll()`, `replace()`, `search()`, dan `split()` dari String.

Sebuah pola regular expression terdiri dari karakter sederhana seperti `/abc/`, atau kombinasi karakter sederhana dan karakter khusus, seperti `/ab*c/` atau `/Bab(\d+)\.\d*/`. Contoh terakhir menggunakan tanda kurung, yang digunakan sebagai penyimpan memori. Kecocokan yang dibuat dengan pola ini kemudian diingat untuk digunakan nantinya.

2.4 Perhitungan Tingkat Kemiripan

Perhitungan tingkat kemiripan dilakukan pada kemiripan substring. Pada tugas besar III ini, algoritma yang dipakai merupakan algoritma Levenshtein Distance.

2.3.2 Algoritma Levenshtein Distance

Algoritma Levenshtein Distance merupakan algoritma yang diciptakan oleh Vladimir Levenshtein pada tahun 1965. Levenshtein Distance merupakan sebuah angka yang merepresentasikan perbedaan diantara dua *string*. Semakin besar angka tersebut maka semakin berbeda kedua *string*. Levenshtein Distance diukur menggunakan persamaan berikut.

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

Dengan a adalah string 1, b adalah string 2, i dan j merupakan posisi karakter terminal pada masing-masing string. Untuk kasus yang besar, Levenshtein Distance dapat memanfaatkan matriks untuk menemukan perbedaannya. Berdasarkan matriks, perbedaan Levenshteinnya 3, karena angka tersebut

	K	I	T	T	E	N
0	1	2	3	4	5	6
S 1	1	2	3	4	5	6
I 2	2	1	2	3	4	5
T 3	3	2	1	2	3	4
T 4	4	3	2	1	2	3
I 5	5	4	3	2	2	3
N 6	6	5	4	3	3	2
G 7	7	6	5	4	4	3

Dengan mengevaluasi dari kolom baris kiri atas,

$$\text{lev}_{a,b}(1, 1) = \min \begin{cases} \text{lev}_{a,b}(0, 1) + 1 = 2 \\ \text{lev}_{a,b}(1, 0) + 1 = 2 \\ \text{lev}_{a,b}(0, 0) + 1_{(\text{jika } a_1 \neq b_1)} = 0 + 1 = 1 \end{cases}$$

Didapat bahwa nilai minimalnya 1, kemudian masukkan ke matriks.

Selanjutnya kalkulasi kolom dan baris berikutnya,

$$\text{lev}_{a,b}(1, 2) = \min \begin{cases} \text{lev}_{a,b}(0, 2) + 1 = 3 \\ \text{lev}_{a,b}(1, 1) + 1 = 2 \\ \text{lev}_{a,b}(0, 1) + 1_{(\text{jika } a_1 \neq b_1)} = 1 + 1 = 2 \end{cases}$$

Didapat bahwa nilai minimalnya 2, kemudian masukkan ke matriks.

Selanjutnya kalkulasi kolom dan baris berikutnya,

berada pada kolom paling kanan dan baris paling bawah.

2.5 Arsitektur Aplikasi Web yang Dibangun

Pada Tugas besar III Strategi Algoritma ini, program diimplementasikan dalam bahasa Typescript dan Node.JS. Node.JS merupakan sebuah *open-sourced environment* JavaScript lintas platform yang dapat mengeksekusi kode JavaScript diluar web browser. Node.js dibangun berdasarkan JavaScript V8 Engine. Salah satu fitur utama Node.js adalah arsitektur event-driven, yang memungkinkan developer untuk menulis kode *asynchronous* yang dapat menangani berbagai permintaan dalam suatu waktu.

2.5.1. Next.js

Next.js merupakan *framework* berbasis React yang memungkinkan pembuatan aplikasi *server-rendered* dan halaman website statis. *Framework* ini menyediakan banyak *tool* dan fitur yang memudahkan penggunanya untuk membangun web berkinerja tinggi, termasuk perenderan sisi server, pembuatan situs statis, pemisahan kode secara otomatis, dan navigasi *client-side* yang *seamless*.

2.5.2. Vercel

Vercel, sebelumnya dikenal dengan Zeit, adalah platform cloud yang menyediakan layanan hosting dan implementasi untuk aplikasi Next.js. Aplikasi ini dirancang untuk bekerja *seamlessly* dengan Next.js dan menyediakan fitur yang memudahkan proses *deployment* dan pengelolaan aplikasi web. Vercel juga mendukung fitur seperti generator sertifikat SSL secara otomatis, CDN global, domain yang dapat diubah sendiri. Fitur-fitur tersebut membantu meningkatkan keamanan dan performansi dari aplikasi web.

2.5.3. PostgreSQL

PostgreSQL merupakan salah satu *Database Management System* (DBMS) berbasis SQL *open source*. PostgreSQL cukup populer dengan banyak fitur-fitur yang memudahkan proses manajemen data.

2.5.4. Typescript

Typescript merupakan bahasa pemrograman *open source* yang merupakan superset dari bahasa Javascript. Pada bahasa Typescript, terdapat fitur-fitur tambahan dengan fitur utama yaitu tipedata statis. Typescript lebih baik untuk digunakan dalam pemrograman web dibanding Javascript karena lebih secure dan aman dari *type bug*.

BAB III

RANCANGAN PEMECAHAN MASALAH

3.1. Langkah Pemecahan Masalah

Dalam pemecahan permasalahan tugas besar ini, penulis mengawali penggerjaan dengan analisis permasalahan untuk memudahkan penggerjaan. Analisis permasalahan tersebut menghasilkan beberapa bagian yang dijabarkan sebagai berikut:

1. Membuat Graphical User Interface (GUI) dalam bentuk aplikasi web yang dapat menerima *input* masukan dari pengguna berupa pertanyaan, pilihan metode algoritma KMP-BM, serta menampilkan riwayat percakapan dengan ChatBot. Dalam implementasinya, kami memanfaatkan *framework* Next JS sebagai *tool* untuk membantu memvisualisasikan GUI.
2. Membuat fungsi pencarian *exact match* yang menerapkan algoritma Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM). Penjelasan mengenai algoritma akan dijelaskan pada poin dibawah. Pada algoritma, digunakan struktur data matriks untuk membuat tabel.
3. Perincian fitur-fitur menjadi beberapa modul. Perincian tersebut antara lain: fitur kalkulator, fitur kalender, fitur tambah dan hapus pertanyaan, dan fitur menjawab pertanyaan berdasarkan *database*.

Pada implementasinya, fitur-fitur tersebut diprioritaskan berdasarkan kerumitan *regular expression*. Prioritas tersebut dari yang terbesar antara lain: fitur kalender, fitur kalkulator, fitur tambah pertanyaan, fitur hapus pertanyaan, dan kemudian fitur menjawab pertanyaan dari basis data. Berikut merupakan langkah penyelesaian dari setiap fitur yang telah dijabarkan sebelumnya.

3.1.1. Fitur Kalkulator

1. Fitur menerima masukan pengguna yang bersesuaian dengan *regular expression* yang menyatakan suatu sintaks persamaan matematika. Masukan harus mengandung angka dan operator yang direpresentasikan dengan regex `/^[\d+\-*/*()]\s]+(\?)?\$/`.
2. Fitur kemudian mengetes apakah sintaks persamaan yang dimasukan valid. Contoh sintaks yang tidak valid yaitu $2 + - 4$ atau $(2 +) * 34 (= 2)$.
3. Jika sintaks persamaan yang dimasukkan valid, fitur akan menghitung hasil menggunakan struktur data stack. Jika karakter yang diolah pada saat itu merupakan angka atau tanda kurung, maka akan dimasukkan (push) ke dalam stack. Kemudian, stack menerima sebuah operator dan operand lainnya dan menghasilkan hasil operasi tersebut menggunakan pop.

3.1.2. Fitur Kalender

1. Fitur menerima masukan pengguna yang bersesuaian dengan *regular expression* yang menyatakan suatu tanggal. Tanggal berada dalam format dd/mm/yyyy tapi juga memungkinkan menerima d/m/yyyy. Masukan harus mengandung angka dan garis miring direpresentasikan dengan regex `^(?:Hari apa)?(\d{1,2}\/\d{1,2}\/\d{4}) (\?)?$/i`.
2. Fitur kemudian mengetes apakah masukan merupakan tanggal yang valid. Contoh tanggal yang tidak valid yaitu 00/00/0000 atau 32/01/2023.
3. Jika masukan merupakan tanggal valid, fitur akan mengkonstruksi hasil *parsing* pada regex ke dalam kelas Date dan mendapatkan hari pada tanggal tersebut.

3.1.3. Fitur Tambah Pertanyaan

1. Fitur menerima masukan pengguna yang diawali frasa ‘Tambah pertanyaan’ dan mengandung frasa ‘dengan jawaban’. Masukan ini direpresentasikan dengan regex `/^Tambah pertanyaan +(.+) dengan jawaban +(.+)$/i`
2. Fitur akan mengecek jika pertanyaan yang dimasukkan beririsan dengan fitur lain. Misalnya pertanyaan ‘2+3’ beririsan dengan fitur kalkulator. Jika beririsan, fitur akan mengeluarkan pesan error dan pertanyaan tidak ditambah ke basis data. Hal ini dilakukan dengan konsiderasi bahwa pertanyaan beririsan yang ditambah tidak akan bisa diakses kemudian. Jika pengguna mencoba mengakses pertanyaan tersebut, sistem akan otomatis mengalihkan ke fitur lain.
3. Fitur akan mengecek jika pertanyaan sudah terdapat pada basis data. Jika sudah ada, fitur akan memperbarui jawaban ka jawaban yang baru dimasukan.
4. Jika pertanyaan belum ada dalam basis data, pertanyaan akan ditambahkan pada basis data dengan jawaban yang bersesuaian.
5. Perlu diperhatikan, batasan pada fitur ini yaitu fitur akan mendeteksi kalimat pertanyaan setelah frasa ‘Tambah Pertanyaan’ yang pertama kali ditemukan, dan mendeteksi jawaban setelah frasa ‘dengan jawaban’ yang terakhir kali ditemukan. Maka, `Tambah pertanyaan Tambah pertanyaan Tambah pertanyaan dengan jawaban dengan jawaban dengan jawaban` akan menghasilkan pertanyaan ‘Tambah pertanyaan Tambah pertanyaan dengan jawaban’ dengan jawaban ‘dengan jawaban’.

3.1.4. Fitur Hapus Pertanyaan

1. Fitur menerima masukan pengguna yang diawali frasa ‘Hapus pertanyaan’. Masukan ini direpresentasikan dengan regex `^Hapus pertanyaan (.+)`.
2. Fitur akan mengecek jika pertanyaan terdapat pada basis data. Jika iya, fitur akan menghapus pertanyaan pada basis data.
3. Jika pertanyaan yang dimaksud tidak ada pada basis data, maka fitur akan mengembalikan jawaban ‘Tidak ada pertanyaan y pada database.’
4. Fitur kemudian mengembalikan rekomendasi pertanyaan dengan jarak terdekat untuk dihapus dan mengembalikan jawaban ‘Mungkin maksud kamu pertanyaan ‘x?’’. Perlu diperhatikan bahwa jarak terdekat dikalkulasikan menggunakan algoritma Levenshtein Distance.

3.1.5. Fitur Menjawab Pertanyaan dari Database

1. Fitur menerima masukan pengguna.
2. Menggunakan algoritma KMP atau BM, dilakukan pencocokan masukan pengguna dengan basis data. Pencocokan ini menggunakan pendekatan *exact matching*.
3. Jika tidak ditemukan pertanyaan yang *exact*, maka hitunglah jarak kemiripan dengan menggunakan algoritma Levenshtein Distance.
4. Jika terdapat kemiripan diatas atau sama dengan 90%, maka dilakukan asumsi bahwa pengguna menanyakan pertanyaan tersebut. Fitur kemudian mengembalikan jawaban dengan jawaban termirip tersebut.
5. Jika tidak ditemukan pertanyaan dengan kemiripan diatas 90%, fitur akan mengembalikan

3.2. Pemodelan Algoritma Pencarian

Berikut merupakan algoritma pencarian yang digunakan dalam *string matching* pada program. Pada Tugas Besar ini, digunakan dua macam algoritma yaitu KMP dan BM. Kedua algoritma digunakan untuk mencari sebuah pattern substring pada sebuah teks. Tetapi, penggunaan pada program ini agak berbeda. Kedua algoritma digunakan untuk mencari **exact match**, yang berarti kesamaan sama persis antara kedua teks yang dibandingkan. Berikut merupakan flow pencarian substring:

1. Carilah exact match substring dengan menggunakan algoritma KMP atau BM
2. Jika tidak ditemukan exact match, carilah pertanyaan termirip dengan menggunakan algoritma Levenshtein Distance
3. Levenshtein Distance akan mengukur semua kemiripan teks, jika terdapat data dengan kemiripan $\geq 90\%$, maka sistem akan mengasumsi pengguna menanyakan pertanyaan tersebut
4. Jika tidak, maka sistem akan merekomendasikan tiga pertanyaan termirip kepada user.

2.2.1. Algoritma Knuth-Morris-Pratt (KMP)

1. Dengan menggunakan *pattern*, buatlah tabel prefiks. Tabel prefiks ini nantinya digunakan sebagai acuan dalam pergeseran mencocokan *pattern* dan *string*.
2. Inisialisasi nilai *j* dan *k* menjadi nol. Nilai *j* merupakan indeks pada teks yang dicari, sedangkan nilai *i* merupakan indeks pada *pattern*.
3. Lakukan perbandingan dari kiri-ke-kanan. Jika karakter yang dibandingkan sama, tambah nilai *j* dan *k*. Jika ditemukan ketidaksesuaian karakter, geser *pattern* ke *prefix* terbesar dari teks [0..*k*-1] yang merupakan *suffix* dari $P[1..k-1]$. Pencarian pergeseran ini mengacu kepada tabel *prefix* yang telah dibuat sebelumnya.
4. Lakukanlah pencarian hingga *pattern* ditemukan atau teks yang dicari sudah sampai akhir.

3.2.2. Algoritma Boyer-Moore (BM)

1. Algoritma ini melakukan perbandingan dari kanan. Inisialisasi nilai *i* dan *j* menjadi 0. Nilai *i* merupakan indeks pada *pattern*, sedangkan nilai *j* merupakan indeks pada teks.
2. Misalkan *P* merupakan *pattern* yang dicari, sedangkan *T* merupakan teks tempat *pattern* dicari.
3. Jika ditemukan ketidaksesuaian karakter, pergeseran dilakukan sehingga kemunculan terakhir *x* di *P* bersesuaian dengan $T[i]$.
4. Jika tidak terdapat kemunculan terakhir, maka geser *pattern* sehingga $P[0]$ bersesuaian dengan $T[i+1]$.
5. Lakukanlah pencarian hingga *pattern* ditemukan atau teks yang dicari sudah sampai akhir.

3.2.3 Perhitungan Kemiripan

Perhitungan kemiripan *string* menggunakan algoritma Levenshtein Distance. Algoritma ini hampir selalu digunakan untuk membandingkan teks secara kontekstual. Pada Levenshtein Distance, karakter yang harus ditambahkan atau dihapus akan diberi bobot. Bobot tersebut ditambahkan di antara dua string untuk memberikan jarak akhir.

1. Buatlah representasi jarak antara kedua *string* dengan struktur data matriks. Inisialisasi matriks tersebut dengan nol.
2. Matriks menggunakan indexing dari angka satu, yang berarti karakter pertama dihitung sebagai indeks ke-satu. Oleh karena itu, indeks ke-nol pada matriks diisi dengan *substitution cost*, yang bersesuaian dengan urutan pada indeks.
3. Menggunakan algoritma Levenshtein, hitunglah jarak perbedaan dan simpan hasilnya pada matriks. Kalkulasikan setiap *cell* sampai semua matriks terisi.

4. Levenshtein Distance merupakan hasil dari Distance pada matriks dengan kolom paling kanan dan baris paling bawah.

3.3. Arsitektur Web

Website yang dibuat menggunakan *framework* Next.js baik untuk *frontend* maupun *backend* dengan beberapa library tambahan seperti *next connect*, *tailwindcss*, *headlessui*, dll. Client dapat menggunakan fungsional server melalui REST API yang disediakan di api folder Next.js. Website tersebut di-deploy pada Vercel. Arsitektur pada aplikasi website terbagi menjadi 4 fitur fungsional utama, yang akan dideskripsikan sebagai berikut:

1. Fitur Pemilihan Algoritma

Fitur ini memungkinkan pengguna untuk memilih algoritma yang digunakan dengan menggunakan dua buah tombol. Algoritma yang dipilih kemudian akan dipassing ke pencarian.

2. Fitur Riwayat Percakapan

Fitur ini dapat menyimpan riwayat percakapan pengguna sebelumnya secara otomatis. Perlu diperhatikan bahwa tidak terdapat autentifikasi pengguna, maka riwayat bersifat global.

3. Fitur Pemilihan Ruang Percakapan

Pengguna dapat memilih ruang percakapan berdasarkan riwayat percakapan sebelumnya, ataupun memilih untuk membuat ruang percakapan baru.

4. Fitur Bercakap dengan ChatBot

Pengguna dapat memasukan pertanyaan, kemudian chatbot akan mengembalikan jawaban yang bersesuaian. Pemilihan jawaban berdasarkan fitur yang telah dijabarkan sebelumnya.

BAB IV

ANALISIS PEMECAHAN MASALAH

4.1. Implementasi Program

Berikut merupakan pseudocode algoritma utama yang memuat logika utama pada program.

```
{ Algoritma Pengecekan }

procedure check(input: string text, string pattern, boolean exact, output: boolean) {
    if (exact AND pattern.length != text.length) then
        -> false
    if (algorithm == "BM") then
        checkBM(text)
    else then
        checkKMP(text)

{ Algoritma KMP }
{ Pembuatan tabel prefiks }

prefixTable : matrix
i <- 0
j traversal [1..pattern.length]
    while (i>0 AND pattern[i] != pattern[j]) do
        i <- prefixTable[i-1]
    if (pattern[i] == pattern[j]) then
        i <- i+1
    prefixTable[j] <- i

{ Pengecekan algoritma KMP }
function checkKMP(input: string text, output: boolean) {
    j <- 0
    i <- 0
    while (j < text.length) {
        if (pattern[k] == text[j]) then
            j <- j + 1
            k <- k + 1
        if (k == pattern.length) then
            -> true
        else if ( j<text.length AND pattern[k] != text[j]) then
            if (k != 0) then
                k <- prefixTable[k-1]
            else then
                j <- j + 1
    { Jika sudah sampai akhir teks }
    -> false

{ Algoritma BM }

function checkBM(input: string text, output: boolean)
    if (pattern.length > text.length) then
        return false
    i traversal [0..text.length]
        last.set(text[i], i)
    i <- pattern.length - 1
    j <- i
    do {
        if (pattern[j] == text[i] then
            if (j==0) then
                -> true
            i <- i - 1
```

```
    j <- j - 1
else then
    lastOccurrence <- last.get(pattern[i] - 1
    i <- i + pattern.length - min(j, 1 + lastOccurrence)
    j <- pattern.length - 1
} while ( i <= text.length - 1)

{ Jika sudah sampai akhir teks }
-> false
```

```
{ Algoritma Levenshtein Distance }

function checkLevenshtein(input: string a, string b, output: int distance)
m <- a.length
n <- b.length
d : Matrix

i traversal [1..m]
d[i][0] <- i
j traversal [1..n]
d[0][j] <- j

j traversal [1..n]
i traversal [1..m]
substitutionCost <- 1
if (a[i] == b[j])
    substitutionCost <- 0
d[i][j] <- min (
    d[i - 1][j] + 1,
    d[i][j - 1] + 1,
    d[i - 1][j - 1] + substitutionCost
-> d[m][n]
```

4.2. Struktur Data Program

Pada tugas besar ini, terdapat beberapa struktur data yang digunakan, yaitu:

1. **Stack**, Stack merupakan struktur data yang menggunakan prinsip LIFO (*Last In First Out*) yang penggunaannya dengan memanfaatkan metode Push (memasukkan elemen) dan Pop (mengejekarkan elemen). Struktur data ini diimplementasikan pada fitur kalkulator untuk mengevaluasi persamaan.
2. **Array**, array merupakan struktur data yang menyimpan elemen atau objek dalam bentuk list. Struktur data ini digunakan dalam pembuatan tabel prefiks.
3. **Matriks**, digunakan untuk menghitung Levenshtein Distance.

Dalam pengembangan program ini, dibuat kelas-kelas yang mengimplementasikan struktur data tersebut. Kelas tersebut dikategorikan dalam beberapa folder yang berbeda. Berikut merupakan penjelasan dari setiap kelas yang dibuat.

4.2.1. Folder Algorithm

Tabel 4.2.1. Kelas Base

Nama kelas: BaseAlgorithm (interface)	
Method	
getResponse()	Mengembalikan string hasil respons dari algoritma
isMatch()	Mengembalikan boolean jika masukan user sesuai dengan pre-kondisi fitur, misalnya RegEx sesuai

Tabel 4.2.2. Kelas Calculator

Nama kelas: Calculator (implements BaseAlgorithm)	
Atribut	
regex	Regular expression yang menjadi prekondisi fitur kalkulator
Method	
public getResponse()	Mengembalikan string hasil respons dari algoritma
public isMatch()	Mengembalikan boolean jika masukan user sesuai dengan pre-kondisi fitur, misalnya RegEx sesuai
private evaluate(string expression)	Mengevaluasi persamaan sintaks matematika
private getPrecedence(string operator)	Mengembalikan urutan prioritas dari operator
private operation(number a, number b, string operator)	Mengembalikan hasil operasi dari operator ^ * / + -

Tabel 4.2.3. Kelas DateQuestion

Nama kelas: DateQuestion (implements BaseAlgorithm)	
Atribut	
regex	Regular expression yang menjadi prekondisi fitur hapus
Method	
getResponse()	Mengembalikan string hasil respons dari algoritma
isMatch()	Mengembalikan boolean jika masukan user sesuai dengan pre-kondisi fitur, misalnya RegEx sesuai

Tabel 4.2.4. Kelas Hapus

Nama kelas: Hapus	
Atribut	
regex	Regular expression yang menjadi prekondisi fitur hapus
ChatStorage[] _data	Menyimpan data riwayat percakapan pengguna
Method	
getResponse()	Mengembalikan string hasil respons dari algoritma

<code>isMatch()</code>	Mengembalikan boolean jika masukan user sesuai dengan pre-kondisi fitur, misalnya RegEx sesuai
------------------------	--

Tabel 4.2.5. Kelas Tambah

Nama kelas: Tambah	
Atribut	
<code>regex</code>	Regular expression yang menjadi prekondisi fitur tambah
<code>ChatStorage[] _data</code>	Menyimpan data riwayat percakapan pengguna
Method	
<code>getResponse()</code>	Mengembalikan string hasil respons dari algoritma
<code>isMatch()</code>	Mengembalikan boolean jika masukan user sesuai dengan pre-kondisi fitur, misalnya RegEx sesuai

Tabel 4.2.5. Kelas Question

Nama kelas: Question	
Atribut	
<code>ChatStorage[] _data</code>	Menyimpan data riwayat percakapan pengguna
Method	
<code>getResponse(string input)</code>	Mengembalikan string hasil respons dari algoritma
<code>isMatch()</code>	Mengembalikan boolean jika masukan user sesuai dengan pre-kondisi fitur, misalnya RegEx sesuai
<code>checkLevenshtein(string input)</code>	Mengembalikan nilai Levenshtein Distance dari kedua string

Tabel 4.2.5. Kelas StringMatching

Nama kelas: StringMatching	
Atribut	
<code>number[] prefixTable</code>	Menyimpan tabel prefix
<code>string algorithm</code>	Algoritma yang digunakan
<code>string pattern</code>	Pola substring yang dicari
Method	
<code>check(string text, boolean exact)</code>	Mengembalikan hasil pencarian pattern pada teks sesuai algoritma
<code>checkBM(string text)</code>	Mengembalikan true jika substring pattern ditemukan pada text dengan menggunakan algoritma Boyer-Moore
<code>checkKMP(string text)</code>	Mengembalikan true jika substring pattern ditemukan pada text dengan menggunakan algoritma Knuth-Morris-Pratt
<code>getLevenshteinDistance(string a, string b)</code>	Mengembalikan Levenshtein Distance berdasarkan masukan string a dan string b.

4.2.1. Folder Components

Folder ini merupakan folder yang menangani komponen-komponen di dalam aplikasi web. Berikut merupakan file yang terdapat dalam folder tersebut beserta deskripsinya.

Nama file	Deskripsi
AlgorithmContext.tsx	Menangani algoritma yang dipilih pengguna (KMP atau BM)
Chat.tsx	Menangani tampilan percakapan antara bot dengan pengguna
ChatSection.tsx	Menangani tampilan daftar ruang riwayat percakapan pengguna
HistoryNav.tsx	Menangani riwayat percakapan pengguna
MessageForm.tsx	Menangani komponen tempat pengguna memasukkan chat
NavBar.tsx	Menangani tampilan navigation bar, dimana terdapat <i>toggle</i> sidebar dan tombol new chat untuk memulai chat baru
NavButton.tsx	Menangani tombol pada navigation bar yang merupakan <i>toggle</i> sidebar
Sidebar.tsx	Menangani tampilan sidebar

4.2.1. Folder Hooks

Folder ini berisi satu file, useNewChat.ts, yang menangani jika pengguna menginginkan ruang percakapan yang baru.

4.2.1. Folder Pages

Folder ini berisi file utama tempat pengembangan aplikasi web. File dalam folder ini juga berkaitan dengan perpindahan laman pada aplikasi web. Perpindahan laman ini disebabkan oleh perpindahan ruang percakapan dengan milih riwayat percakapan tertentu.

4.2.1. Folder Server

Folder ini berisi dua file: api-handler.ts dan db.ts. Seperti namanya, file pada folder ini menangani API yang berkaitan dengan basis data (termasuk API Request dan API Response).

4.2.1. Folder Styles

Folder ini berisi satu file: globals.css yang menangani *styling* pada halaman web secara keseluruhan.

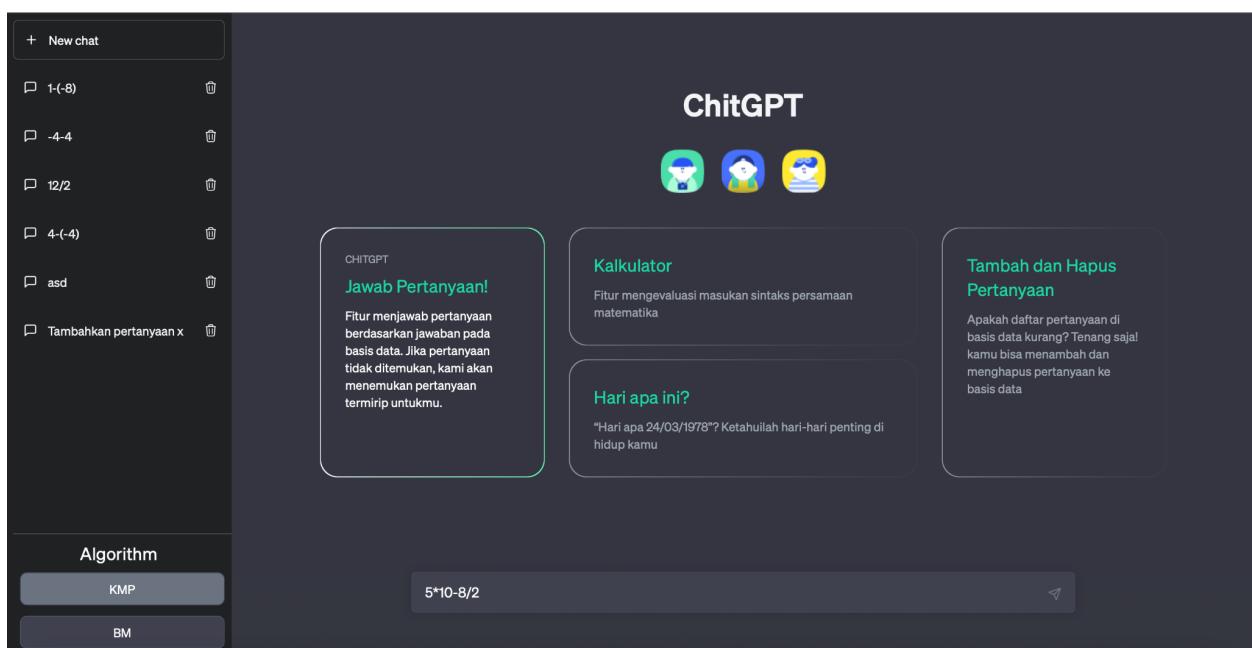
4.2.1. Folder Utils

Folder ini berisi file yang menangani berbagai keperluan seperti mempos pesan kepada percakapan, *fetching* riwayat percakapan, dan *fetching* percakapan.

4.3. Tata Cara Penggunaan Program

Program telah di-deploy ke web dengan tautan yang dapat diakses di <https://chitgpt.arsaizdihar.com>.

Berikut merupakan bentuk program ketika baru dibuka.



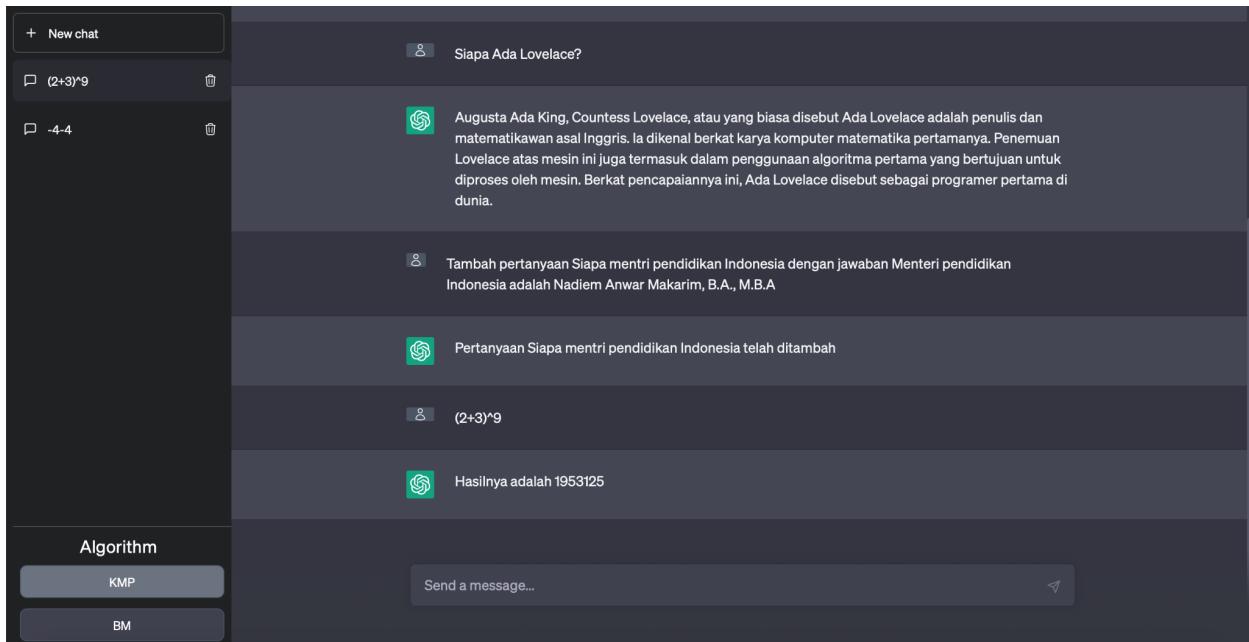
Gambar 3.3.1. Tampilan Utama Program

Fitur fungsional pada laman web terbagi menjadi 4 bagian utama.

1. Button “Algorithm” yang mengarahkan pengguna memilih algoritma antara KMP atau BM.
2. Sidebar berisi riwayat percakapan yang telah dilakukan sebelumnya. Setiap riwayat berperilaku sebagai button. Jika dipilih, akan mengarahkan pengguna ke ruang riwayat percakapan tersebut.
3. Button “New Chat” yang mengarahkan pengguna ke ruang percakapan baru.
4. Textbox dengan placeholder “Send a message...”. Perlu diperhatikan, jika ingin menanyakan 2 hal sekaligus, pengguna perlu menggunakan *newline* (dua pertanyaan berada dalam dua baris yang berbeda).

Homepage juga menyediakan sebuah antarmuka yang interaktif dan memberikan informasi mengenai fitur yang terdapat pada program.

Untuk memulai percakapan, pengguna hanya perlu memasukkan pertanyaan lewat textbox. Jawaban yang bersesuaian nantinya akan keluar dengan sendirinya. Rincian setiap *query* akan dijelaskan pada bagian pengujian program.



Gambar 4.3.2. Tampilan Utama Program

4.3. Pengujian dan Analisis Hasil pengujian Program

Berikut merupakan pengujian program dengan menggunakan penyelesaian ChitGPT.

Tabel 4.3.1. Pengujian Program Fitur Kalkulator

Permasalahan	Penyelesaian
$2 + 3$	<p>2+3</p> <p>Hasilnya adalah 5</p> <p>Analisis: ChatBot dapat menerima masukan dengan sintaks persamaan matematika dan mengembalikan jawaban yang benar.</p>
$(1 - (2^3)/4.3 + 104 - 0.2*0.5 + (-1.2)) / (-2.983) ?$	<p>(1 - (2^3)/4.3 + 104 - 0.2*0.5 + (-1.2)) / (-2.983) ?</p> <p>Hasilnya adalah -34.14</p> <p>Analisis: ChatBot dapat menerima masukan dengan sintaks persamaan matematika dengan akhiran tanda tanya. Selain itu, fitur kalkulator dapat menangani 5 operan yaitu +, -, /, *, ^ dan tanda kurung. Fitur juga menangani angka desimal. Jawaban yang dikembalikan telah terkonfirmasi benar.</p>

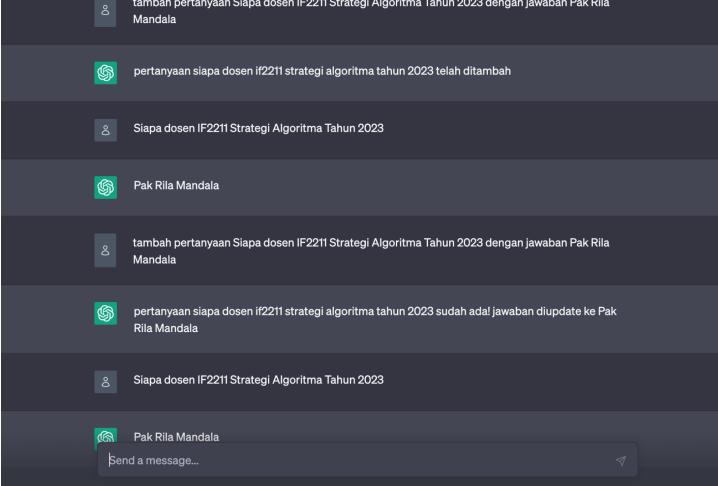
2 + - 3	<p>Analisis: ChatBot dapat menerima masukan dengan sintaks persamaan matematika yang tidak sesuai. Pada permasalahan ini, tidak bisa mengevaluasi $2 + - 3$.</p>
$(1 - (2^3)/4.3 + 104 - 0.2*0.5 + (-1.2)) / 0$	<p>Analisis: ChatBot dapat menerima masukan sintaks persamaan yang sesuai, tetapi hasilnya tidak terdefinisi karena hasil persamaan dibagi dengan nol.</p>

Tabel 4.3.2. Pengujian Program Fitur Kalender

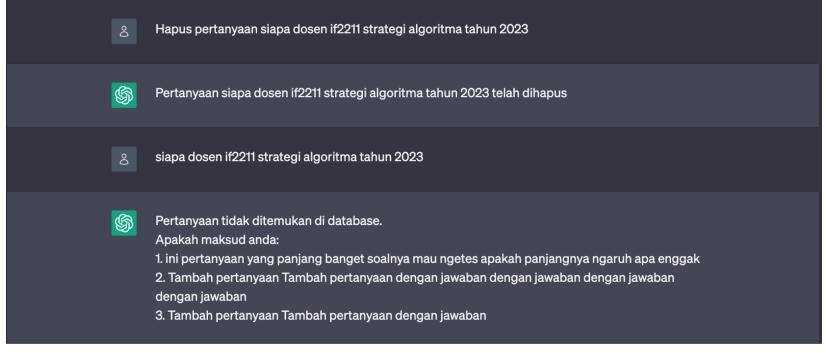
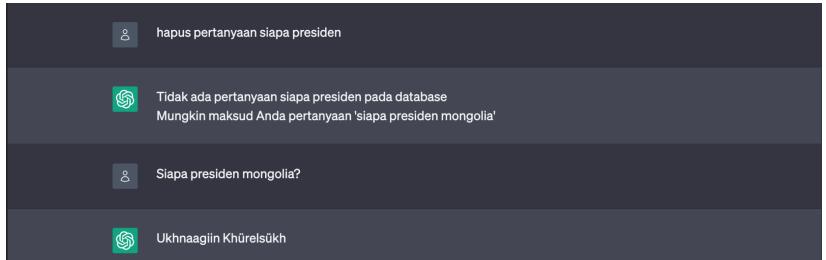
Permasalahan	Penyelesaian
Hari apa 19/03/2023?	<p>Analisis: ChatBot dapat menerima masukan dengan awalan ‘Hari apa’ yang diikuti oleh tanggal dengan format dd/mm/yyyy. Hasil jawaban chatbot juga terkonfirmasi benar.</p>
22/02/1999?	<p>Analisis: ChatBot dapat menerima masukan tanpa awalan ‘Hari apa’, hanya menanyakan tanggal. Hasil jawaban chatbot terkonfirmasi benar.</p>
2/2/2002?	<p>Analisis: ChatBot dapat menerima format d/m/yyyy. Hasil jawaban chatbot terkonfirmasi benar.</p>
32/04/1974?	<p>Analisis: ChatBot menerima format dd/mm/yyyy dengan benar, tetapi tanggal yang dimasukkan tidak sesuai karena tidak ada tanggal 32. Maka, hasil jawaban chatbot merupakan error.</p>
07/00/2003?	

	Analisis: ChatBot menerima format dd/mm/yyyy dengan benar, tetapi tanggal yang dimasukkan tidak sesuai karena tidak ada bulan 00. Maka, hasil jawaban chatbot merupakan error.
--	--

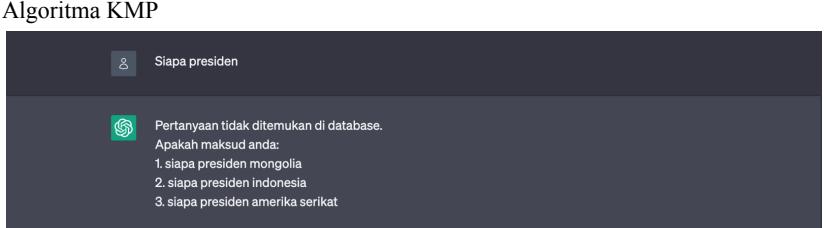
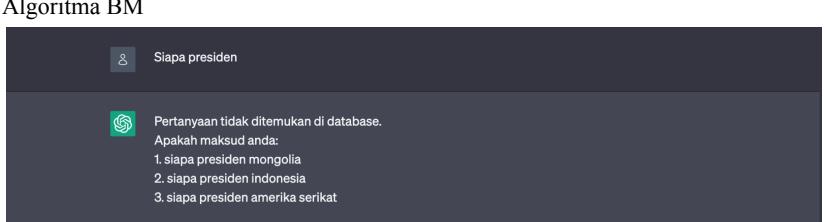
Tabel 4.3.3. Pengujian Program Fitur Tambah Pertanyaan

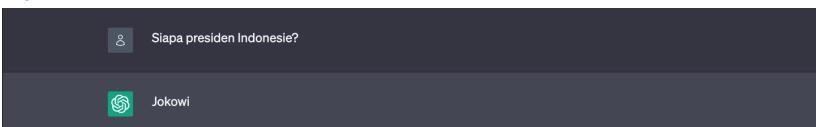
Permasalahan	Penyelesaian
Tambah pertanyaan Siapa dosen IF2211 Strategi Algoritma Tahun 2023? dengan jawaban Pak Rinaldi Munir. (pertanyaan belum ada pada basis data)	 <p>Analisis: ChatBot menerima masukan yang sesuai untuk fitur tambah pertanyaan. Pertanyaan belum ada pada basis data, maka pertanyaan ditambahkan. Setelah itu, pertanyaan diuji dan didapatkan jawaban yang sesuai.</p>
Tambah pertanyaan Siapa dosen IF2211 Strategi Algoritma Tahun 2023? dengan jawaban Pak Rila Mandala. (pertanyaan sudah ada pada basis data, jawaban berbeda)	 <p>Analisis: ChatBot menerima masukan yang sesuai untuk fitur tambah pertanyaan. Pertanyaan sudah ada pada basis data, maka pertanyaan diperbarui jawabannya. Setelah itu, pertanyaan diuji dan didapatkan jawaban terbaru yang sesuai.</p>
Tambah pertanyaan Siapa dosen IF2211 Strategi Algoritma Tahun 2023? dengan jawaban Pak Rila Mandala. (pertanyaan sudah ada pada basis data, jawaban sama)	 <p>Analisis: ChatBot menerima masukan yang sesuai untuk fitur tambah pertanyaan. Pertanyaan sudah ada pada basis data, maka pertanyaan diperbarui jawabannya. Setelah itu, pertanyaan diuji dan karena jawabannya sama, maka didapatkan jawaban yang sesuai.</p>

Tabel 4.3.4. Pengujian Program Fitur Hapus Pertanyaan

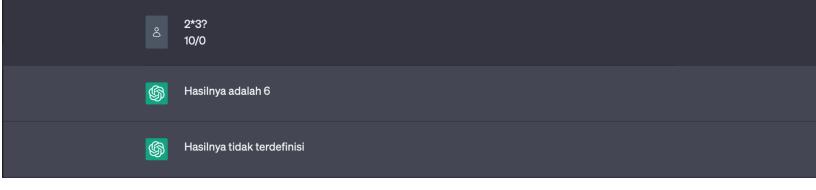
Permasalahan	Penyelesaian
<p>Hapus pertanyaan siapa dosen if2211 strategi algoritma tahun 2023 (pertanyaan ada di database)</p>	 <p>Analisis: ChatBot menerima masukan yang sesuai untuk fitur hapus pertanyaan. Pertanyaan terdapat pada basis data, maka pertanyaan dihapus dari basis data. Setelah itu, pertanyaan diuji dan tidak ditemukan (penghapusan berhasil).</p>
<p>Hapus pertanyaan siapa presiden (pertanyaan tidak ada di database)</p>	 <p>Analisis: ChatBot menerima masukan yang sesuai untuk fitur hapus pertanyaan. Pertanyaan tidak terdapat pada basis data, tetapi ChatBot merekomendasikan pertanyaan termirip. Walaupun demikian, ChatBot hanya merekomendasikan, namun belum menghapus.</p>

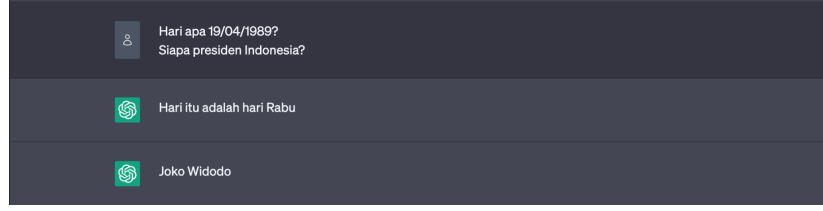
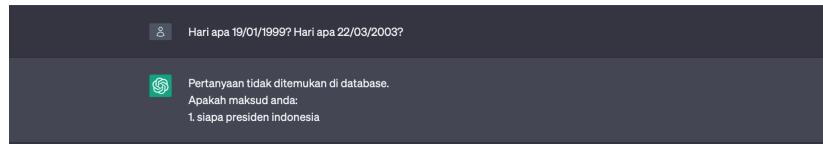
Tabel 4.3.5. Pengujian Program Fitur Pertanyaan pada Database

Permasalahan	Penyelesaian
<p>Siapa presiden? (jawaban tidak terdapat pada basis data)</p>	<p>Algoritma KMP</p>  <p>Algoritma BM</p> 

	<p>Analisis: ChatBot menerima masukan untuk menjawab dari database. Pertanyaan tidak terdapat pada basis data dan tidak terdapat pertanyaan dengan threshold kemiripan diatas 90%, maka ChatBot akan merekomendasikan 3 pertanyaan termirip. Kedua algoritma menghasilkan hasil yang sama.</p>
Siapa presiden Mongolia? (jawaban terdapat pada basis data)	<p>Algoritma KMP</p>  <p>Algoritma BM</p>  <p>Analisis: ChatBot menerima masukan untuk menjawab dari database. Pertanyaan terdapat pada basis data dan ChatBot menjawab sesuai dengan basis data. Kedua algoritma menghasilkan hasil yang sama.</p>
Siapa presiden Indonesia? (terdapat jawaban dengan threshold kemiripan diatas 90%)	<p>Algoritma KMP</p>  <p>Algoritma BM</p>  <p>Analisis: ChatBot menerima masukan untuk menjawab dari database. Pertanyaan tidak terdapat pada basis data tetapi terdapat pertanyaan dengan threshold diatas 90%. Maka ChatBot akan menjawab sesuai dengan pertanyaan termirip tersebut. Kedua algoritma menghasilkan hasil yang sama.</p>

Tabel 4.3.6. Pengujian Program Dengan Permasalahan Menarik

Permasalahan	Penyelesaian
$2*3?$ $10/0$ (Hal ini merepresentasikan 2 Query dengan 2 Fitur sama)	 <p>Analisis: ChatBot menerima 2 query masukan. Perlu diperhatikan bahwa kedua query harus dipisahkan dengan newline. Pada kasus ini, query menerima fitur yang sama, yaitu kalkulator. Kedua kalkulator mengembalikan hasil yang valid.</p>

<p>Hari apa 19/04/1989? Siapa presiden Indonesia?</p> <p>(Hal ini merepresentasikan 2 Query dengan fitur berbeda)</p>	
<p>Hari apa 19/01/1999? Hari apa 22/03/2003?</p> <p>(Hal ini merepresentasikan 2 Query tapi tidak dipisah newline)</p>	
<p>2*5-10 Siapa presiden Indonesia? Hari apa 05/12/1978? Tambah pertanyaan Apakah pak Judhi akan masuk kelas besok dengan jawaban Tidak tahu Siapa Saul Sayers?</p> <p>(Hal ini merepresentasikan 5 Query)</p>	

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Pada tugas besar II IF2211 Strategi Algoritma ini telah diimplementasikan algoritma Knuth-Morris-Pratt dan Boyer-Moore beserta kelas-kelas pendukung dengan tujuan untuk menyelesaikan permasalahan *string matching* yang tertera pada spek. Selain itu, juga diimplementasikan pencarian *string matching* menggunakan *regular expression*. Kelas-kelas pendukung mencakup hal yang menangani GUI (Graphical User Interface) menggunakan *framework* Next.js dan kelas interface.

Pada tabel 3.3.1 sampai tabel 3.3.4 terlihat bahwa setiap fitur terimplementasikan dengan baik. Pada tabel 3.3.5, terlihat bahwa algoritma KMP dan BM selalu berhasil menyelesaikan permasalahan yang valid. Didapat bahwa kedua algoritma membutuhkan waktu yang relatif sama, sehingga tidak dapat dipastikan algoritma mana yang menampilkan hasil yang lebih optimal. Dengan demikian, penulis menyimpulkan bahwa melalui Tugas Besar III IF2211 Strategi Algoritma ini, dapat dibuat sebuah algoritma berbasis KMP, BM dan Regular Expression yang mendukung pembuatan sebuah ChatBot.

5.2. Saran

Tugas Besar III IF2211 Strategi Algoritma Semester II Tahun 2022/2023 menjadi salah satu tugas yang memberikan pelajaran baru bagi penulis. Berdasarkan pengalaman penulis mengerjakan tugas ini, berikut merupakan saran untuk pembaca yang ingin melakukan atau mengerjakan hal serupa.

1. Keefektifan dalam kerja sama tim merupakan hal yang penting dalam mengerjakan tugas ini. Selain pembagian tugas yang merata, penulis terbantu oleh beberapa kali kerja kelompok dan pemakaian *real-time collaboration app*, seperti Google Docs. Selain itu, penggunaan version control (Github) disarankan untuk memudahkan mengelola pekerjaan secara asinkron.
2. Perancangan algoritma dan struktur program perlu diperhatikan dalam mengerjakan tugas ini. Implementasi struktur program yang tepat akan memudahkan dalam pembuatan Graphical User Interface (GUI) pada *platform* web.

5.3. Refleksi dan Tanggapan

Tugas Besar II IF2211 Strategi Algoritma merupakan salah satu tugas besar yang memerlukan usaha lebih dari biasanya. Meskipun demikian, kami mengapresiasi tugas besar ini karena memberikan kami kesempatan untuk bereksplorasi dan mengimplementasikan algoritma yang menarik.

DAFTAR PUSTAKA

Khodra, L (2022). *Strategi Algoritma: String Matching dengan Regular Expression* Dilansir dari Homepage Rinaldi Munir:
<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf>. Diakses pada 1 Mei 2023.

Munir, R. (2022). Strategi Algoritma: Pencocokan String (*String/Pattern Matching*)
Dilansir dari Homepage Rinaldi Munir:
<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf>.
Diakses pada 1 Mei 2023.

LAMPIRAN

Repository Github

https://github.com/arsaizdihar/Tubes3_13521101

Link Deploy Website

<https://chitgpt.vercel.app/>

Link Video Youtube

<https://youtu.be/vODGhiRgkB0>

CheckList Fitur

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan	v	
2. Program berhasil running	v	
3. Program dapat membaca input pengguna	v	
4. Semua fitur terimplementasikan	v	
5. Program berhasil di deploy	v	