PENERAPAN STRING MATCHING DAN REGULAR EXPRESSION DALAM PEMBUATAN CHATGPT SEDERHANA

Diajukan sebagai pemenuhan tugas besar 3.

Disusun oleh:

Kelompok 30 (**pAIn**)

1. 13521052 - Melvin Kent Jonathan
2. 13521074 - Eugene Yap Jin Quan
3. 13521162 - Antonio Natthan Krishna

Dosen Pengampu : Dr. Nur Ulfa Maulidevi, S.T, M.Sc.

IF2211 - Strategi Algoritma

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

DAFTAR ISI

DAFTAR ISI **2**
BAB 1
DESKRIPSI MASALAH **3**
BAB 2
TEORI DASAR **5**
2.1. Algoritma Knuth-Morris-Pratt (KMP) 5
2.2. Algoritma Boyer-Moore (BM) 6
2.3. Regular Expression (REGEX) 7
2.4. Aplikasi Web Chat 8
BAB 3
ANALISIS PEMECAHAN MASALAH **10**
3.1. Deskripsi Singkat Aplikasi 10
3.2. Langkah Pemecahan Masalah 10
3.3. Fitur Fungsional dan Arsitektur Aplikasi Web 12
BAB 4
IMPLEMENTASI DAN PENGUJIAN **14**
4.1. Spesifikasi Teknis 14
4.2. Tata Cara Penggunaan Program 29
4.3. Hasil Pengujian 31
4.4. Analisis Hasil Pengujian 34
BAB 5
PENUTUP **36**
5.1. Kesimpulan 36
5.2. Saran, Komentar, dan Refleksi 36
DAFTAR PUSTAKA **37**
LAMPIRAN **38**

BAB 1
DESKRIPSI MASALAH

Pendekatan Question Answering (QA) pada chatbot 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. Sementara itu, regular expression adalah kumpulan aturan atau pola yang digunakan untuk pencocokan string dengan format yang spesifik.

Masalah dalam tugas besar ini adalah 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 digunakan 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, namun disarankan menggunakan salah satu dari algoritma **Hamming Distance, Levenshtein Distance,** ataupun **Longest Common Subsequence**.

Fitur-Fitur Aplikasi:

- Fitur pertanyaan teks (didapat dari database)
    - Mencocokkan pertanyaan dari input pengguna ke pertanyaan di database menggunakan algoritma KMP atau BM.
- 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.
- 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.
- Tambah pertanyaan dan jawaban ke database
    - Pengguna dapat menambahkan pertanyaan dan jawabannya sendiri ke database dengan query contoh “Tambahkan pertanyan xxx dengan jawaban yyy”. Menggunakan algoritma string matching untuk mencari tahu apakah pertanyaan sudah ada. Apabila sudah, maka jawaban akan diperbaharui.
- Hapus pertanyaan dari database
    - Pengguna dapat menghapus sebuah pertanyaan dari database dengan query contoh “Hapus pertanyaan xxx”. Menggunakan string algoritma string matching untuk mencari pertanyaan xxx tersebut pada database.
- Klasifikasi dilakukan menggunakan regex dan terklasifikasi layaknya bahasa sehari - hari. Algoritma string matching KMP dan BM digunakan untuk klasifikasi query teks. Tersedia toggle pada website bagi pengguna untuk memilih algoritma KMP atau BM. Semua pemrosesan respons dilakukan pada sisi backend. Jika ada pertanyaan yang sesuai dengan fitur, maka tampilkan saja “Pertanyaan tidak dapat diproses”.

BAB 2
TEORI DASAR

2.1. Algoritma *Knuth-Morris-Pratt (KMP)*

Algoritma Knuth-Morris Pratt (selanjutnya disebut KMP) merupakan algoritma pencarian string cukup populer. Algoritma ini merupakan perbaikan dari algoritma brute-force sehingga dimungkinkan pergeseran lebih dari satu karakter. Konsep yang diajukan oleh algoritma ini cukup sederhana, yaitu dengan menghitung panjang maksimum dari prefix suatu pattern dengan suffix pattern tersebut.

a. Prefix

   Substring yang diambil dari karakter pertama sebuah pattern. Contoh, pattern “STIMA” dapat memiliki prefix “S”, “ST”, “STI”, “STIM”, dan “STIMA”, namun tidak “SIM”

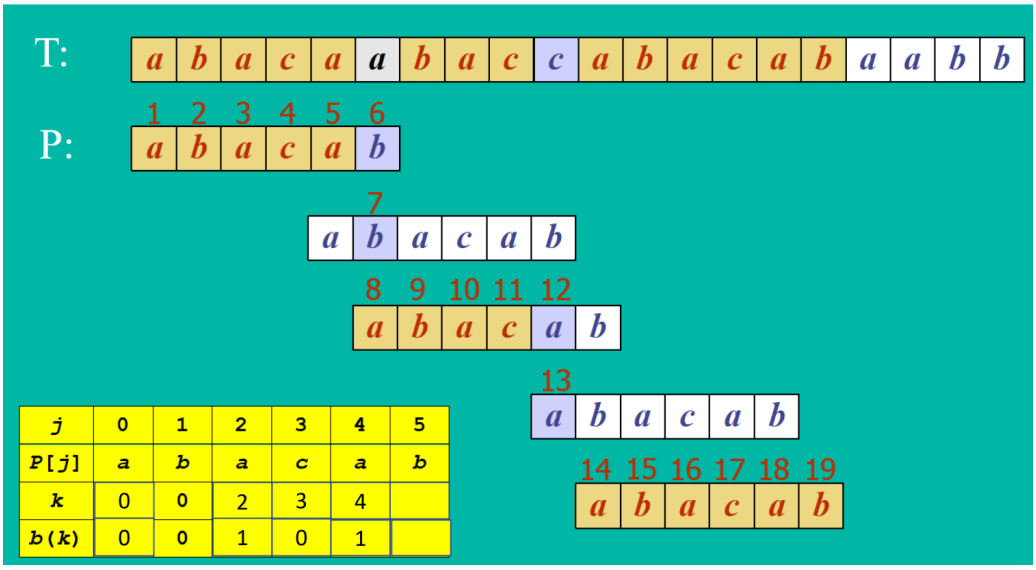
b. Suffix

   Contoh, pattern “STIMA” dapat memiliki suffix “A”, “MA”, “IMA”, “TIMA”, dan “STIMA”, namun tidak “TIA”.

Sebelum pencocokan string dimulai, dilakukan inisialisasi pattern untuk menentukan kecocokan prefix dan suffix pattern tersebut ketika pattern memiliki mismatch pada karakter di index i. Kemudian nilai nilai tersebut disimpan dalam sebuah array yang kemudian menjadi panjang pergeseran ketika terjadi mismatch karakter ke i pada pattern.

Dengan algoritma ini, pencocokan string pada text seakan akan tidak pernah mundur. Pattern akan menyesuaikan pencocokan karakter selanjutnya dengan menggeser sebanyak mungkin berdasarkan kesamaan prefix dan suffix yang ada pada pattern tersebut. Salah satu keuntungan menggunakan KMP adalah pembacaan ini memudahkan external devices melakukan pekerjaannya karena pembacaan tidak akan dilakukan secara mundur. Namun, KMP tidak cocok untuk size alphabet yang besar, karena akan meningkatkan terjadinya mismatch di awal yang membuat KMP tidak jauh lebih baik dibandingkan brute force.

Contoh berikut merupakan simulasi pencocokan pattern “ABACAB” pada string “ABACAABACCABACABAABB” menggunakan algoritma KMP.



Gambar 2.1.1 Ilustrasi Pencocokan String dengan KMP.

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

2.2. Algoritma *Boyer-Moore (BM)*

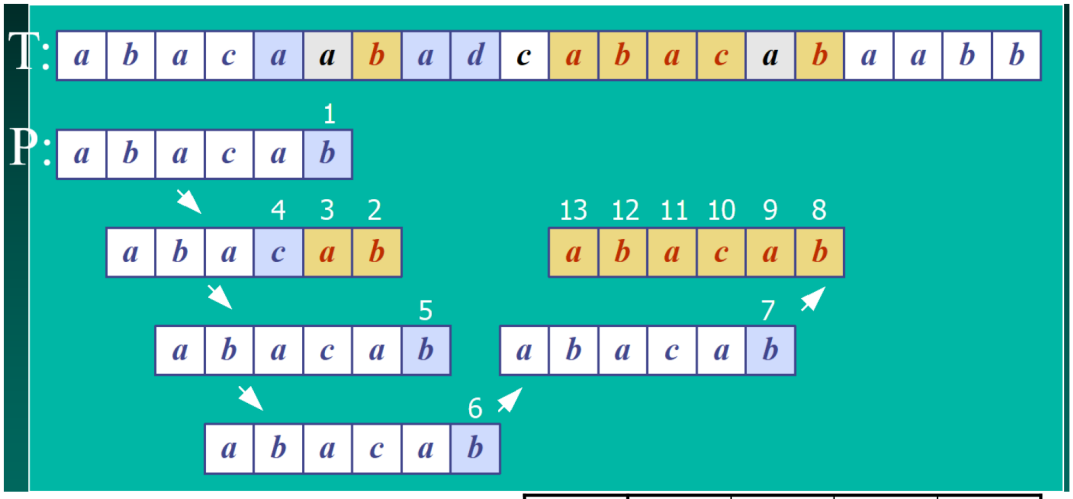
Algoritma Boyer-Moore (selanjutnya disebut BM) merupakan algoritma pencocokan string yang efektif untuk range kamus yang besar. Algoritma ini memungkinkan untuk menggeser pattern secara keseluruhan apabila karakter pada text tidak terdapat pada pattern target. Berlawanan dengan KMP, BM memiliki best case ketika mismatch terjadi pada awal pencocokan (KMP akan menjadi efektif ketika mismatch berada di akhir).

Dasar dari algoritma BM adalah dengan inisialisasi pencarian last occurrence dari suatu karakter pada suatu string. Data ini akan disimpan pada suatu array yang akan menjadi pedoman pergeseran pencocokan string. Pada proses pencocokan, BM memiliki pendekatan yang cukup berbeda, yaitu dengan melakukan pencocokan pada karakter terakhir pattern terlebih dahulu. Terdapat 3 kasus yang mungkin terjadi pada proses pencocokan string menggunakan algoritma BM,

1. Apabila terjadi mismatch, karakter text terdapat pada pattern dengan last occurrence nya karakter tersebut belum diperiksa. Pindahkan last occurrence karakter pada pattern menjadi sejajar dengan karakter text saat ini.
2. Apabila terjadi mismatch, karakter text terdapat pada pattern dengan last occurrence nya karakter tersebut sudah diperiksa. Geser pattern 1 ke kanan.
3. Apabila terjadi mismatch dan karakter text tidak terdapat pada pattern. Geser seluruh pattern ke kanan karakter text.

Informasi last occurrence didapat dari preprocessing yang dilakukan sebelum proses pencocokan dilakukan.

` Berikut merupakan ilustrasi pencocokan pattern “ABACAB” pada text “ABACAABADCABACABAABB” dengan menggunakan algoritma BM,



Jumlah perbandingan karakter: 13 kali

x	a	b	c	d
L(x)	4	5	3	-1

Gambar 2.2.1 Ilustrasi Pencocokan String dengan BM.

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

2.3. *Regular Expression (REGEX)*

Regular Expression (selanjutnya disebut REGEX) adalah salah satu bentuk bahasa formal untuk menentukan string matching. Hingga saat ini, REGEX menjadi dasar pencocokan string terhadap suatu pola tertentu. Penulisan REGEX cukup sederhana, mudah dipahami, dan bersifat general sehingga REGEX merupakan salah satu kit penting dalam beberapa bahasa pemrograman.

REGEX juga dapat membantu dalam proses validasi pola seperti contoh validasi email pengguna. Email memiliki struktur yang jelas, (nama pengguna)@(nama domain).(ekstensi_domain). Dengan menggunakan REGEX, pencocokan pola tersebut dapat dengan mudah dilakukan dengan satu baris kode! Dalam bahasa Javascript, REGEX sudah menjadi modul bawaan bahasa tersebut sehingga tidak diperlukan import modul tambahan (hal ini mungkin berbeda dalam bahasa pemrograman lainnya seperti Python yang mewajibkan untuk melakukan `import re` jika ingin menggunakan REGEX)

```
text.search([[a-zA-z]\w*@[a-z]+.[a-z]{2, 3}])
```

Notasi yang digunakan sudah terstandarisasi. Berikut merupakan beberapa diantaranya,

.	Mencari single character, kecuali newline atau karakter pengakhir line
\w	Mencari sebuah karakter word
\W	Mencari sebuah karakter non-word
\d	Mencari sebuah digit
\D	Mencari sebuah karakter bukan digit
\s	Mencari sebuah whitespace character
\S	Mencari sebuah karakter bukan whitespace
\b	Mencari pola di awal atau di akhir pattern
\B	Mencari pola selain di awal dan di akhir pattern
\0	Mencari karakter NULL
\n	Mencari karakter newline
\t	Mencari karakter tab
\v	Mencari karakter vertical tab

Tabel 1. Aturan REGEX

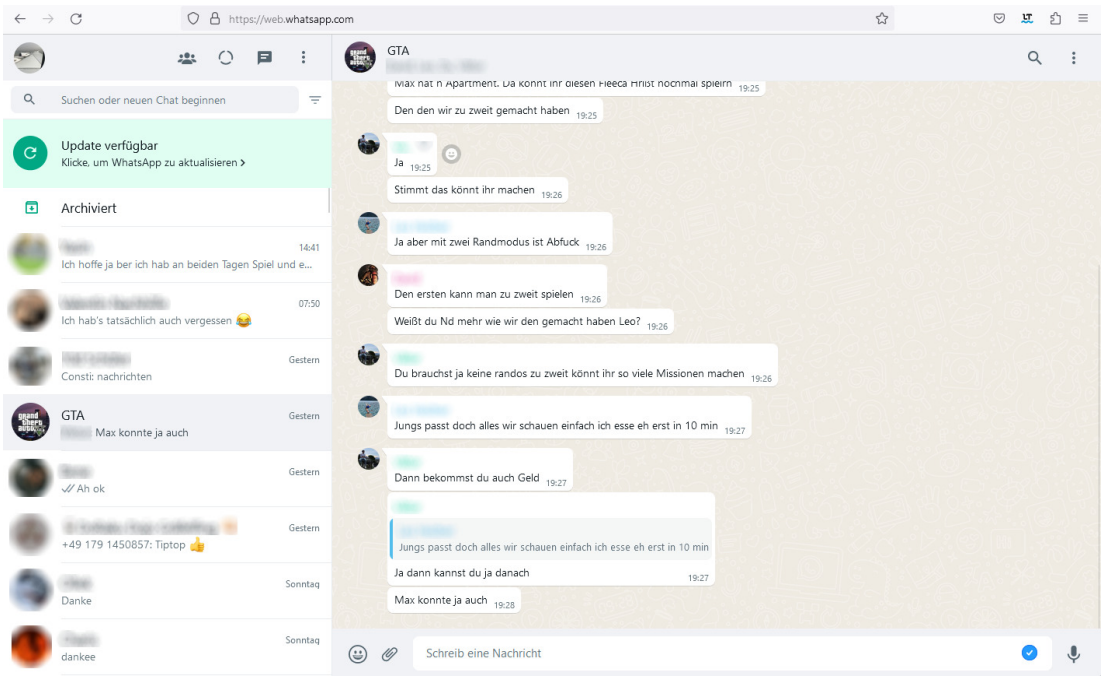
Sumber: https://www.w3schools.com/jsref/jsref_obj_regexp.asp

2.4. Aplikasi Web Chat

Aplikasi web (aplikasi web) adalah program aplikasi yang disimpan di server jarak jauh dan dikirimkan melalui internet melalui antarmuka browser. Layanan web pada dasarnya merupakan aplikasi web menurut definisinya, meskipun tidak semua situs web memiliki fungsionalitas seperti aplikasi.

Pengembang mendesain aplikasi web untuk berbagai macam penggunaan dan pengguna, mulai dari organisasi hingga perorangan. Aplikasi web yang umum digunakan dapat mencakup webmail, kalkulator online, atau toko e-commerce. Meskipun pengguna hanya dapat mengakses beberapa aplikasi web dengan browser tertentu, sebagian besar web apps dapat diakses melalui browser apapun.

Salah satu contoh web apps yang ada saat ini adalah WhatsApp Web. Kita dapat mengirim pesan melalui server WhatsApp melalui browser. Kita dapat bertukar pesan, mengirim dan menerima informasi melalui WhatsApp Web. Hal ini lah yang menjadi dasar pengembangan aplikasi yang ada pada laporan ini



Gambar 2.4.1 Ilustrasi Pencocokan String dengan BM.

Sumber: https://www.chip.de/downloads/webapp-WhatsApp-Web_164054150.html

BAB 3
ANALISIS PEMECAHAN MASALAH

3.1. Deskripsi Singkat Aplikasi

Aplikasi ChatGPT pada dasarnya merupakan sebuah aplikasi Web Chat. Perbedaan terletak pada lawan komunikasi dari pengguna. Pengguna tidak berbicara kepada manusia asli, melainkan kepada sebuah kecerdasan buatan (Artificial Intelligence). Aplikasi ini tidak hanya mengklasifikasi pertanyaan dan menjawabnya, namun juga belajar dari input pengguna. Aplikasi ini akan menjadi semakin powerful ketika digunakan oleh lebih banyak orang karena secara tidak langsung aplikasi tersebut telah mendapat sumber belajar yang lebih banyak.

Aplikasi web yang kami bangun merupakan aplikasi chatbot yang berfungsi menjawab pertanyaan dari pengguna secara interaktif berdasarkan pengetahuan atau *knowledge* yang dimilikinya yang disimpan pada basis data. Selain menjawab pertanyaan terbuka dari pengguna berdasarkan kecocokan pertanyaan pengguna dengan kumpulan pertanyaan yang disimpannya, aplikasi web chat ini juga dapat menjawab pertanyaan pengguna terkait dengan operasi matematika (tambah, kurang, kali, bagi, pangkat, dan kurung) serta pencarian nama hari pada tanggal yang dimasukan oleh pengguna. Adapun pengguna dapat pula menambah pertanyaan dan jawaban ke *database*, maupun menghapus pertanyaan dari *database*.

3.2. Langkah Pemecahan Masalah

Langkah pemecahan masalah adalah sebagai berikut,

1. Input pengguna akan diproses dan diklasifikasikan dengan melakukan pencocokan pola menggunakan REGEX. Klasifikasi ini akan menentukan apakah pertanyaan tersebut meminta jawaban dari fitur Aplikasi (seperti kalkulator dan tanggal) atau bertanya berdasarkan data yang telah ada di dalam database.
2. Sebuah objek fitur akan dibangun berdasarkan hasil klasifikasi REGEX dan akan dilakukan message passing terhadap objek tersebut untuk melakukan evaluasi terhadap string Input.
3. Objek akan melakukan evaluasi sesuai algoritmanya masing masing, dan memanfaatkan algoritma KMP atau BM (sesuai dengan pilihan pengguna) apabila diperlukan algoritma string Matching didalamnya.

4. Objek akan mengembalikan string hasil evaluasi untuk ditampilkan dalam GUI aplikasi web.

Berikut merupakan penyelesaian masalah dari setiap fitur yang ada pada aplikasi yang kami kembangkan,

Nama Fitur	Algoritma
Kalkulator	1. Lakukan parsing angka, operator, dan kurung dari string input pengguna 2. Lakukan evaluasi berdasarkan prioritas perhitungan matematika (kali didahulukan dari tambah, pangkat didahulukan dari kali). 3. Apabila tidak valid kirimkan balikan NaN 4. Apabila valid, lakukan evaluasi terhadap angka operator dan kurung
Tanggal	1. Lakukan parsing angka, operator, dan kurung dari string input pengguna 2. Lakukan evaluasi berdasarkan prioritas perhitungan matematika (kali didahulukan dari tambah, pangkat didahulukan dari kali). 3. Apabila tidak valid kirimkan balikan NaN 4. Apabila valid, lakukan evaluasi terhadap angka operator dan kurung
Randomize	1. Lakukan parsing angka jumlah pilihan. Kirimkan pesan kesalahan apabila bukan angka 2. Parsing pilihan menjadi array 3. Apabila panjang array lebih kecil daripada jumlah pilihan, kirimkan pesan kesalahan 4. Apabila valid, pilih secara random sebanyak jumlah pilihan .
Tambah Pertanyaan	1. Mencari pertanyaan pada database 2. Proses pencarian dilakukan dengan string matching KMP dan BM 3. Apabila string yang match berhasil ditemukan, perbarui jawaban 4. Apabila string tidak match, catat Levenshtein Distance dari string tersebut 5. Apabila tidak ditemukan string match sampai akhir, Apabila pertanyaan dengan Levenshtein tertinggi berada di atas 90%, maka perbarui jawaban. Apabila tidak, tambahkan pertanyaan tersebut
Hapus Pertanyaan	1. Mencari pertanyaan pada database 2. Proses pencarian dilakukan dengan string matching KMP dan BM 3. Apabila string yang match berhasil ditemukan, hapus jawaban 4. Apabila string tidak match, catat Levenshtein Distance dari

	string tersebut 5. Apabila tidak ditemukan string match sampai akhir, Apabila pertanyaan dengan Levenshtein tertinggi berada di atas 90%, maka hapus pertanyaan dan jawaban. Apabila tidak, kembalikan pesan kesalahan beserta 3 pertanyaan dengan Levenshtein Distance tertinggi.
Cari Pertanyaan	1. Mencari pertanyaan pada database apabila string tidak memenuhi fitur lainnya 2. Proses pencarian dilakukan dengan string matching KMP dan BM 3. Apabila string yang match berhasil ditemukan, kembalikan jawaban yang bersesuaian 4. Apabila string tidak match, catat Levenshtein Distance dari string tersebut 5. Apabila tidak ditemukan string match sampai akhir, Apabila pertanyaan dengan Levenshtein tertinggi berada di atas 90%, maka kembalikan jawaban pertanyaan tersebut. Apabila tidak, kembalikan 3 pertanyaan dengan Levenshtein Distance tertinggi.

3.3. Fitur Fungsional dan Arsitektur Aplikasi Web

Fitur fungsional yang dapat dilakukan pada aplikasi web yang kami kembangkan adalah,

Nama Fitur	Deskripsi	Query
Kalkulator	Menghitung kalkulasi dari perhitungan sederhana (+-/*^)	• `Hitung <ekspresi matematika>` • `<ekspresi matematika>`
Tanggal	Menentukan hari dari tanggal terkait	• `Hari apa DD-MM-YYYY` • `Hari apa DD/MM/YYYY` • `DD-MM-YYYY` • `DD/MM/YYYY`
Randomize	Memilih dari pilihan	• `Pilih 1 dari 54-90` • `Pilih 3 dari a, b, c, d`
Tambah Pertanyaan	Menambahkan pertanyaan ke dalam database atau memperbarui pertanyaan	• `Tambah pertanyaan Siapa presiden Indonesia dengan jawaban Jokowi`
Hapus Pertanyaan	Menghapus pertanyaan dari database	• `Hapus pertanyaan Siapa presiden Indonesia`
Cari Pertanyaan	Mencari pertanyaan dari database	• `Siapa presiden Indonesia`

Aplikasi yang kami kembangkan dengan menggunakan NextJS (frontend) dibantu dengan NodeJS (backend) dan menggunakan PostgreSQL untuk pengelolaan basis data yang dikoneksikan melalui Prisma.

BAB 4
IMPLEMENTASI DAN PENGUJIAN

4.1. Spesifikasi Teknis

Implementasi aplikasi web kami menggunakan next.js/node.js. Untuk penyimpanan database, kami menggunakan DBMS PostgreSQL yang dijalankan pada *platform* Supabase, dan koneksi ke aplikasi menggunakan ORM Prisma.

Implementasi algoritma pencocokan terletak pada file-file yang tersedia pada “src/backend/algo”. Integrasi antara algoritma pencocokan dan aplikasi web dilakukan dengan membuat instance dari agent.js pada sisi web.

Implementasi algoritma pencocokan string terdapat pada file-file berikut.

bm.js

```
// Class Boyer Moore
import {Array} from "./struct/Array.js";

export default class BoyerMoore {
    constructor() {
        this.charTable = new Array();
    }

    charProcess(str) {
        let strProcess = String(str);
        let strlen = strProcess.length;

        for (let i = 0; i < 256; i++) {
            this.charTable.setElement(i, -1);
        }

        for (let i = 0; i < strlen; i++) {
            this.charTable.setElement(strProcess[i].charCodeAt(0),
i);
        }
    }

    calculate(text, pattern) {
        let textInput = String(text),
            textLength = textInput.length,
            patternInput = String(pattern),
            patternLength = patternInput.length;

        this.charProcess(patternInput);

        let idx = 0;
```

```
        while(idx <= (textLength - patternLength)) {
            let j = patternLength - 1;
            while(j >= 0 && patternInput[j] == textInput[idx + j])
{
                j--;
            }

            if (j < 0) {
                return idx;
            } else {
                idx += Math.max(1, j -
this.charTable.getElement(textInput[idx + j].charCodeAt(0)));
            }
        }
        return -1;
    }
}
```

calculator.js

```
// Class Calculator
import {Array} from "./struct/Array.js";

export default class Calculator {
    constructor() {
        this.number = new Array();
        this.operator = new Array();
        this.bracket = new Array();
    }

    evaluate(text) {
        let strtext = String(text);
        let textlength = strtext.length;
        let currentIndex = 0;
        let currentchar = strtext[currentIndex];
        let firstNumberNegative = false;

        if ("1234567890+-/.*()^".indexOf(currentchar) == -1) {
            return NaN;
        } else if (currentchar == "-") {
            firstNumberNegative = true;
            currentIndex++;
        } else if (currentchar == "+") {
            currentIndex++;
        }
```

```
        while (currentIndex < textlength) {
            let number1 = 0;
            currentchar = strtext[currentIndex];
            let parseNumber = false;
            let divisor = 1;

            while (!isNaN(strtext[currentIndex])) {
                parseNumber = true;
                if (divisor == 1) {
                    number1 = (number1*10) +
parseInt(strtext[currentIndex]);
                } else {
                    number1 = (number1) +
parseFloat(strtext[currentIndex]/divisor);
                }

                currentIndex++;
                if (currentIndex == strtext.length) {
                    break;
                }
                currentchar = strtext[currentIndex];
                if (currentchar == "." && divisor == 1) {
                    currentIndex++;
                    if (currentIndex == strtext.length) {
                        break;
                    }
                    currentchar = strtext[currentIndex];
                    divisor *= 10;
                } else if (currentchar == "." && divisor != 1) {
                    return NaN;
                } else if (divisor != 1) {
                    divisor *= 10
                }
            }
            if (parseNumber) {
                this.number.push(number1);
            }

            if (currentIndex == strtext.length) {
                break;
            }

            if (currentchar == "(") {
                if (this.number.length() != 0) {
                    this.operator.push("*");
                }
            } else {
                this.operator.push(strtext[currentIndex]);
                currentIndex++;
                currentchar = strtext[currentIndex];
            }
```

```
            if (currentchar == "(") {
                this.bracket.push("(");
                let subtext = "";
                currentIndex++;
                while (currentIndex < textlength &&
this.bracket.length() != 0) {
                    currentchar = strtext[currentIndex];
                    if (currentchar == ")") {
                        this.bracket.pop();
                        if (this.bracket.length() != 0) {
                            subtext += currentchar;
                        }
                    } else if (currentchar == "(") {
                        subtext += currentchar;
                        this.bracket.push("(");
                    } else {
                        subtext += currentchar;
                    }
                    currentIndex++;
                }
                if (this.bracket.length() == 0) {
                    let newstack = new Calculator();
                    this.number.push(newstack.evaluate(subtext));
                    if (currentIndex < textlength &&
"1234567890".indexOf(strtext[currentIndex]) != -1) {
                        this.operator.push("*");
                    }
                } else {
                    return NaN
                }
            }
        }

        if (this.number.length() != this.operator.length() + 1) {
            return NaN;
        } else {
            if (firstNumberNegative) {
                this.number.setElement(0, this.number.getElement(0)
* -1);
            }
            while(this.operator.indexOf("^") != -1) {
                let idx = this.operator.indexOf("^");
                let number1 = this.number.getElement(idx);
                let number2 = this.number.getElement(idx+1);

                this.number.setElement(idx, Math.pow(number1,
number2));

                this.operator.remove(idx);
                this.number.remove(idx+1);
```

```
            }

            while(this.operator.indexOf("*") != -1 ||
this.operator.indexOf("/") != -1) {
                let idx1 = this.operator.indexOf("*");
                let idx2 = this.operator.indexOf("/");
                let idx;
                if (idx1 == -1) {
                    idx = idx2;
                } else if (idx2 == -1) {
                    idx = idx1;
                } else {
                    idx = idx1 < idx2 ? idx1 : idx2;
                }

                let number1 = this.number.getElement(idx);
                let number2 = this.number.getElement(idx+1);

                this.number.setElement(idx, idx == idx1 ? number1 *
number2 : number1 / number2);

                this.operator.remove(idx);
                this.number.remove(idx+1);
            }

            while(this.operator.indexOf("+") != -1 ||
this.operator.indexOf("-") != -1) {
                let idx1 = this.operator.indexOf("+");
                let idx2 = this.operator.indexOf("-");
                let idx;
                if (idx1 == -1) {
                    idx = idx2;
                } else if (idx2 == -1) {
                    idx = idx1;
                } else {
                    idx = idx1 < idx2 ? idx1 : idx2;
                }

                let number1 = this.number.getElement(idx);
                let number2 = this.number.getElement(idx+1);

                this.number.setElement(idx, idx == idx1 ? number1 +
number2 : number1 - number2);

                this.operator.remove(idx);
                this.number.remove(idx+1);
            }
            return this.number.getElement(0);
        }
    }
}
```

date.js

```
export default class Calendar {

    constructor() {
        this.weekday =
["Minggu","Senin","Selasa","Rabu","Kamis","Jumat","Sabtu"]
    }

    isLeapYear(y) {
        if (y % 400 == 0) {
            return true;
        } else if (y % 100 == 0) {
            return false;
        } else if (y % 4 == 0) {
            return true;
        } else {
            return false;
        }
    }

    countLeapYear(low, high){
        let i = low;
        let count = 0;
        while (i <= high) {
            if (this.isLeapYear(i)) {
                count++;
            }
            i++;
        }
        return count;
    }

    getDayMonth(m, y) {
        if (m == 1 || m == 3 || m == 5 || m == 7 || m == 8 || m ==
10 || m == 12) {
            return 31;
        } else if (m == 2 && this.isLeapYear(y)) {
            return 29;
        } else if (m == 2 && !this.isLeapYear(y)) {
            return 28;
        } else {
            return 30;
        }
```

```
    }

    getDayCount(d, m, y) {
        let plus = 0;
        let i = 0;
        if (Math.abs(y - 2023) > 1) {
            if (y < 2023) {
                plus += (2022 - y) * 365;
                plus += this.countLeapYear(y+1, 2022);
                plus++;
            } else {
                plus += (y - 2024) * 365;
                plus += this.countLeapYear(2024, y-1);
            }
        }
        if (y == 2023) {
            i = 1;
            while (i < m) {
                plus += this.getDayMonth(i, y);
                i++;
            }
            plus += d;
            plus--;
        } else if (y > 2023) {
            plus += 365;
            i = 1
            while (i < m) {
                plus += this.getDayMonth(i, y);
                i++;
            }
            plus += d;
            plus--;
        } else {
            i = m + 1
            while (i <= 12) {
                plus += this.getDayMonth(i, y);
                i++;
            }
            plus += this.getDayMonth(m, y) - d;
            plus *= -1;
        }
        return plus;
    }

    getDay(d, m, y) {
        let intd = parseInt(d), intm = parseInt(m), inty =
parseInt(y);
        let dayCount;
        if (intd > 0 && intd <= 31 && intm > 0 && intm <= 12) {
            if (intm == 2) {
                if (this.isLeapYear(inty) && intd > 29) {
```

```
                    return "Tanggal tidak valid";
                } else if (!this.isLeapYear(inty) && intd > 28) {
                    return "Tanggal tidak valid";
                }
            } else if (intm == 4 || intm == 6 || intm == 9 || intm
== 11) {
                if (intd > 30) {
                    return "Tanggal tidak valid";
                }
            }
            dayCount = this.getDayCount(intd, intm, inty);
            return d + "/" + m + "/" + y + " adalah hari " +
this.weekday[dayCount % 7 < 0 ? dayCount % 7 + 7 : dayCount % 7];
        } else {
            return "Tanggal tidak valid";
        }
    }
}
```

kmp.js

```
import {Array} from "./struct/Array.js";

// Knuth Morris Pratt Pattern Matching Algorithm

export default class KMP {

    constructor() {
        this.patternTable = new Array();
    }

    reinitializeTable(pattern, length) {
        this.patternTable.push(0);

        let len = 0;
        let i = 1;
        while (i < length) {
            if (pattern[i] == pattern[len]) {
                len++;
                this.patternTable.setElement(i, len);
                i++;
            } else {
                if (len != 0) {
                    len = this.patternTable.getElement(len-1);
                } else {
                    this.patternTable.setElement(i, 0);
                    i++;
                }
```

```
            }
        }
    }

    calculate(text, pattern) {
        let textInput = String(text),
            textLength = textInput.length,
            patternInput = String(pattern),
            patternLength = patternInput.length;

        this.reinitializeTable(patternInput, patternLength);

        let i = 0,
            j = 0;
        while ((textLength - i) >= (patternLength - j)) {
            if (patternInput[j] == textInput[i]) {
                j++;
                i++;
            }
            if (j == patternLength) {
                return (i - j);
            } else if (i < textLength && patternInput[j] !=
textInput[i]) {
                if (j != 0)
                    j = this.patternTable.getElement(j - 1);
                else
                    i++;
            }
        }
        return -1;
    }
}
```

levensthein.js

```
// Class Levensthein

export default class Levensthein {
    calculate(text1, text2) {
        let text1str = String(text1);
        let text2str = String(text2);
        let minlen, denom;
        let val = 0;

        if (text1str.length < text2str.length) {
            minlen = text1str.length;
```

```
            denom = text2str.length
        } else {
            minlen = text2str.length;
            denom = text1str.length;
        }

        for (let i = 0; i < minlen; i++) {
            if (text1str[i] != text2str[i]) {
                val++;
            }
        }

        val += Math.abs(denom - minlen);

        return val/denom;
    }
}
```

randomize.js

```
// Class Randomize
import {Array} from "./struct/Array.js";

export default class Randomize {

    constructor() {
        this.solution = new Array();
    }

    pick(num, textchoice, length) {
        if (num <= length) {
            for (let i = 0; i < num; i++) {
                let temp = Math.floor(Math.random() * length);
                if (this.solution.indexOf(temp) == -1) {
                    this.solution.push(temp);
                } else {
                    i--;
                }
            }
            let ret = "Saya memilih ";
            for (let i = 0; i < this.solution.length(); i++) {
                ret += textchoice[this.solution.getElement(i)]
                if (i < this.solution.length()-2) {
                    ret += ", ";
                }
                if (i == this.solution.length()-2 &&
this.solution.length() > 2) {
```

```
                    ret += ", dan ";
                }
                if (i == this.solution.length()-2 &&
this.solution.length() > 1) {
                    ret += " dan ";
                }
            }
            return ret;
        } else {
            return "Ga bisa milih! Jumlah pilihan terlalu sedikit"
        }
    }
}
```

agent.js

```
import Calculator from "./calculator.js";
import Calendar from "./date.js";
import Levensthein from "./levensthein.js";
import KMP from "./kmp.js"
import BoyerMoore from "./bm.js"
import Randomize from "./randomize.js";

export default class Agent {
    constructor() {
        this.prompt = "";
        this.mode = true; // TRUE FOR KMP, FALSE FOR BM
        this.kmp = new KMP();
        this.bm = new BoyerMoore();
        this.lev = new Levensthein();
    }

    async process(text, mode) {
        let tool;
        let res;
        this.prompt = text;
        this.mode = mode;
        if
(this.prompt.search(/[0-9]{1,2}[\/-][0-9]{1,2}[\/-][0-9]{4}/i) !=
-1 ||
this.prompt.search(/\bhariapa[0-9]{1,2}[\/-][0-9]{1,2}[\/-][0-9]{4}
/i) != -1) {
            this.prompt = this.prompt.replaceAll(" ", "");
            tool = new Calendar();
            if
(this.prompt.search(/\bhariapa[0-9]{1,2}[\/-][0-9]{1,2}[\/-][0-9]{4
}/i) != -1) {
                this.prompt = this.prompt.substring(7);
```

```
            }
            let d = this.prompt.substring(0,
this.prompt.search(/[\/-]/));
            this.prompt =
this.prompt.substring(this.prompt.search(/[\/-]/) + 1);
            let m = this.prompt.substring(0,
this.prompt.search(/[\/-]/));
            this.prompt =
this.prompt.substring(this.prompt.search(/[\/-]/) + 1);
            return tool.getDay(d, m, this.prompt);
        } else if (this.prompt.search(/\bhitung/i) != -1 ||
this.prompt.search(/\bberapa/i) != -1) {
            this.prompt = this.prompt.replaceAll(" ", "");
            tool = new Calculator(mode);
            if (this.prompt.search(/\bhitung/i) != -1) {
                res =
tool.evaluate(this.prompt.substring(this.prompt.search(/\bhitung/i)
+ 6));
            } else {
                res = tool.evaluate(this.prompt);
            }
            return isNaN(res) ? "Perhitungan tidak valid": res;
        } else if (this.prompt.search(/\bpilih.*[0-9]+.*dari/i) !=
-1) {
            this.prompt = this.prompt.replaceAll(" ", "");
            this.prompt = this.prompt.substring(5);
            console.log(this.prompt);
            let num = 0;
            if (this.prompt.search(/[0-9]+dari/i) == 0) {
                num = parseInt(this.prompt.substring(0,
this.prompt.search(/dari/i)));
            } else {
                return "Jumlah pilihan bukan sebuah angka!";
            }
            this.prompt =
this.prompt.substring(this.prompt.search(/dari/) + 4);
            let choice = [];
            if (this.prompt.search(/[0-9]+-[0-9]+/i) == 0) {
                let low = parseInt(this.prompt.substring(0,
this.prompt.search(/-/i)));
                let high =
parseInt(this.prompt.substring(this.prompt.search(/-/i)+1,
this.prompt.length));
                if (low <= high) {
                    tool = new Randomize()
                    for (let i = low; i < high; i++) {
                        choice.push(i);
                    }
                    return tool.pick(num, choice, choice.length);
                } else {
                    return "Batas atas lebih tinggi dari batas
```

```
bawah!";
                }
            } else if (this.prompt.search(/([\w]+,)+\w+/i) == 0) {
                this.prompt = this.prompt.replaceAll(" ", "");
                while (this.prompt.search(',') != -1) {
                    choice.push(this.prompt.substring(0,
this.prompt.search(',')));
                    this.prompt =
this.prompt.substring(this.prompt.search(',')+1);
                    tool = new Randomize()
                }
                return tool.pick(num, choice, choice.length);
            } else {
                return "Bukan sebuah range atau enumerasi!";
            }

        } else if (this.prompt.search(/Tambahkan pertanyaan.*dengan
jawaban/i) == 0) {
            let a =
this.prompt.substring(this.prompt.search(/dengan jawaban/i) + 15);
            this.prompt = this.prompt.substring(0,
this.prompt.search(/dengan jawaban/i));
            let q = this.prompt.substring(21);

            const getQuestion = await fetch(`/api/questions`, {
                method: "GET",
            });
            const array = await getQuestion.json();

            let ans = -1;
            let idx = 0;
            let found = false;
            let levdis = []
            array.forEach(element => {
                let i;
                if (this.mode) {
                    i = this.kmp.calculate(element.question, q);
                } else {
                    i = this.bm.calculate(element.question, q);
                }
                if (i != -1 && !found) {
                    ans = idx;
                    found = true;
                }
                levdis.push(this.lev.calculate(element.question,
q));
                idx++;
            });
            if (ans != -1) {
                const updateQA = await fetch(`/api/questions`, {
                    method: "PUT",
```

```
                    body: JSON.stringify({
                        question: q,
                        answer: a,
                    })
                })
                return "Jawaban pertanyaan berhasil update";
            } else {
                if (Math.min(levdis) < 0.1) {
                    const updateQA = await fetch(`/api/questions`,
{
                        method: "PUT",
                        body: JSON.stringify({
                            question:
array[levdis.indexOf(Math.min(...levdis))].question,
                            answer: a,
                        })
                    })
                    return "Jawaban pertanyaan berhasil update";
                } else {
                    const insertQuestion = await
fetch(`/api/questions`, {
                                                            method:
"POST",
                                                            body:
JSON.stringify({

question: q,
                                                                answer:
a,
                                                            })
                                                        })
                    return "Pertanyaan berhasil ditambahkan";
                }
            }
        }else if (this.prompt.search(/Hapus pertanyaan/i) == 0) {
            let q = this.prompt.substring(17);

            const getQuestion = await fetch(`/api/questions`, {
                method: "GET",
            });
            const array = await getQuestion.json();

            let ans = -1;
            let idx = 0;
            let found = false;
            let levdis = []
            array.forEach(element => {
                let i;
                if (this.mode) {
                    i = this.kmp.calculate(element.question, q);
                } else {
```

```
                    i = this.bm.calculate(element.question, q);
                }
                if (i != -1 && !found) {
                    ans = idx;
                    found = true;
                }
                levdis.push(this.lev.calculate(element.question,
q));
                idx++;
            });
            if (ans != -1) {
                console.log(q);
                const deleteQA = await
fetch(`/api/questions/${encodeURIComponent(q)}`, {
                    method: "DELETE",
                })
                return "Pertanyaan " + q + " sedang di hapus";
            } else {
                if (Math.min(levdis) < 0.1) {
                    const deleteQA = await
fetch(`/api/questions/${array[levdis.indexOf(Math.min(...levdis))].
question}`, {
                        method: "DELETE",
                    })
                    return "Pertanyaan " + q + " sedang di hapus";
                } else {
                    let text = "Pertanyaan gagal dihapus! Mungkin
maksud Anda:\n";
                    for (let i = 0; i < 3; i++) {
                        let temp =
array[levdis.indexOf(Math.min(...levdis))];
                        text += String(i+1) + ". " +
String(temp.question) + "\n";
                        levdis[levdis.indexOf(Math.min(...levdis))]
= 1;
                    }
                    return text;
                }
            }
        } else {
            const getQuestion = await fetch(`/api/questions`, {
                method: "GET",
            });
            const array = await getQuestion.json();

            let ans = -1;
            let idx = 0;
            let found = false;
            let levdis = []
            array.forEach(element => {
                let i;
```

```
                if (this.mode) {
                    i = this.kmp.calculate(element.question,
this.prompt);
                } else {
                    i = this.bm.calculate(element.question,
this.prompt);
                }
                if (i != -1 && !found) {
                    ans = idx;
                    found = true;
                }
                levdis.push(this.lev.calculate(element.question,
this.prompt));
                idx++;
            });
            if (ans != -1) {
                return String(array[ans].answer);
            } else {
                if (Math.min(levdis) < 0.1) {

console.log(array[levdis.indexOf(Math.min(...levdis))]);
                    return
array[levdis.indexOf(Math.min(...levdis))].answer;
                } else {
                    let text = "Masukkan tidak valid! Mungkin
maksud Anda:\n";
                    for (let i = 0; i < 3; i++) {
                        let temp =
array[levdis.indexOf(Math.min(...levdis))];
                        text += String(i+1) + ". " +
String(temp.question) + "\n";
                        levdis[levdis.indexOf(Math.min(...levdis))]
= 1;
                    }
                    return text;
                }
            }
        }
    }
}
```

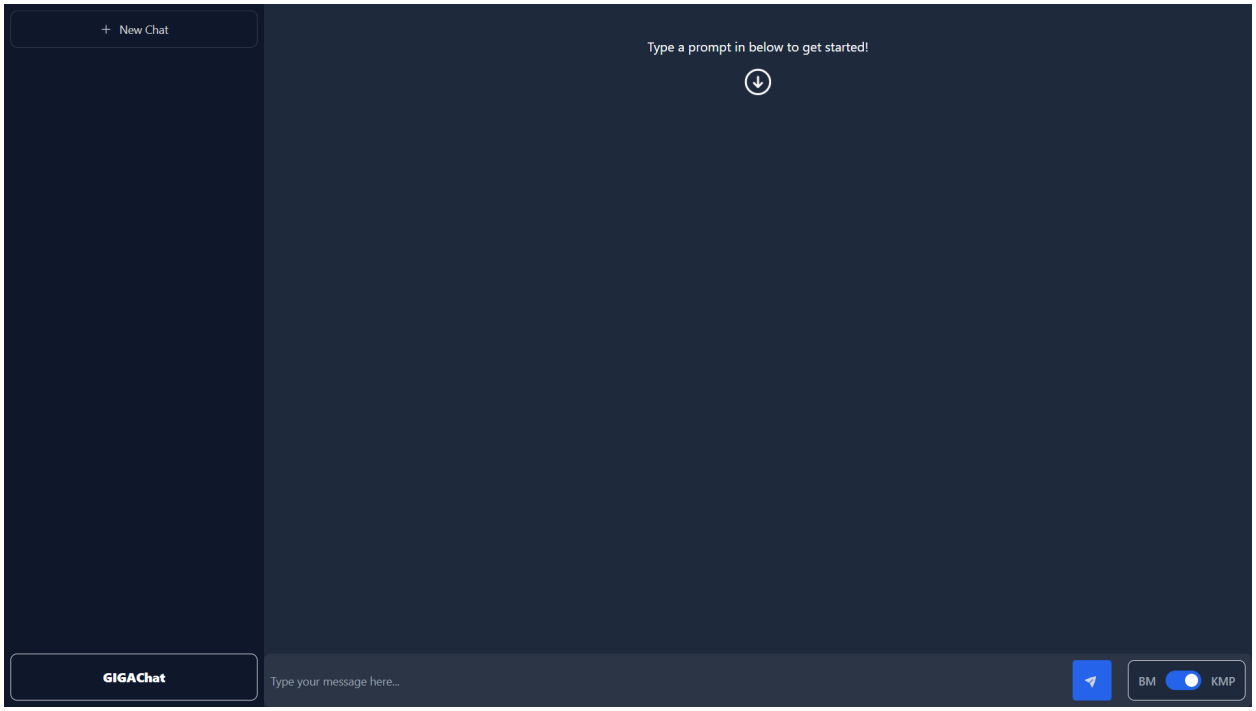
4.2. Tata Cara Penggunaan Program

Untuk menjalankan program ini secara lokal, lakukan clone terhadap repository kemudian navigasi ke folder src. Pastikan juga bahwa Next.Js dan Node.Js sudah dikonfigurasi pada komputer. Selanjutnya, buatlah file “.env” pada folder src dan isi file .env dengan sebagai berikut,dengan password diisi dengan.

.env

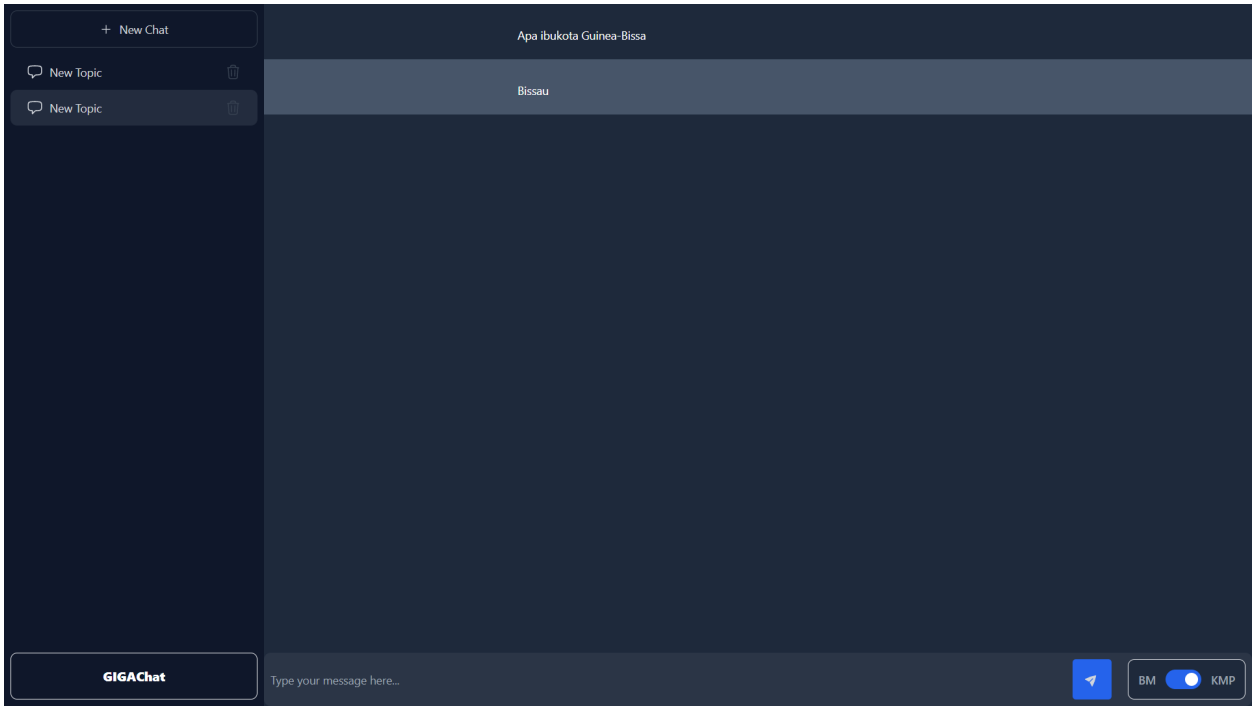
```
# .env
DATABASE_URL="postgresql://postgres:{PASSWORD}@db.hredeexujnmpkmlidzrg.supabase.co:5432/postgres"
```

Setelah itu, jalankan perintah `npm install` untuk menginstal semua *dependency*. Terakhir, jalankan `npm run dev` dan program akan dijalankan pada localhost:3000 jika port tidak dipakai. Program ini kemudian dapat dibuka pada *browser*.



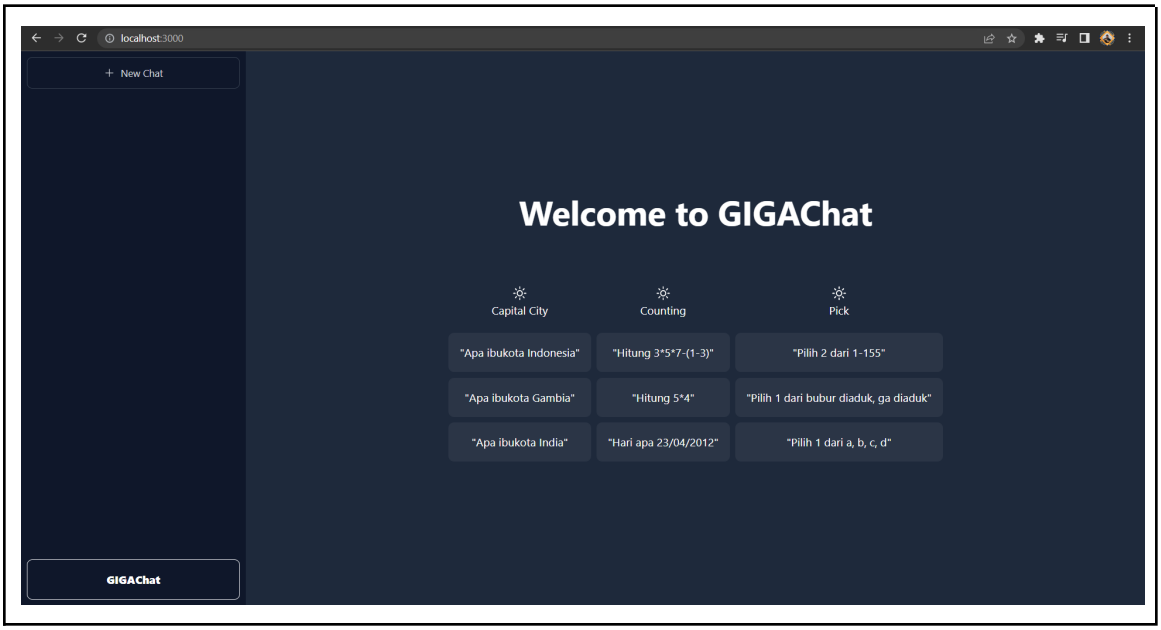
Pengguna dapat menanyakan pertanyaan dengan pertama membuat New Chat dan mengetik pertanyaan pada input teks. Tersedia juga pilihan algoritma BM dan KMP di pojok kanan bawah.

Karena keterbatasan implementasi, elemen UI pada aplikasi masih bersifat statik, sehingga membutuhkan **refresh pada browser** untuk menampilkan perubahan UI, seperti penambahan chat atau pesan.

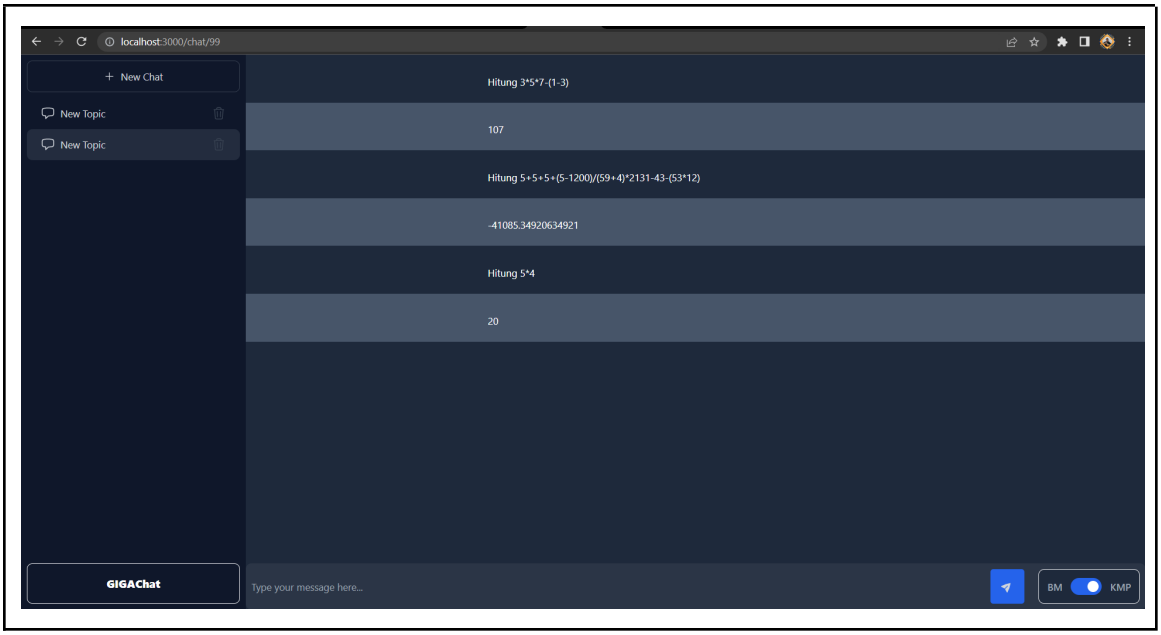


4.3. Hasil Pengujian

1. Landing Page



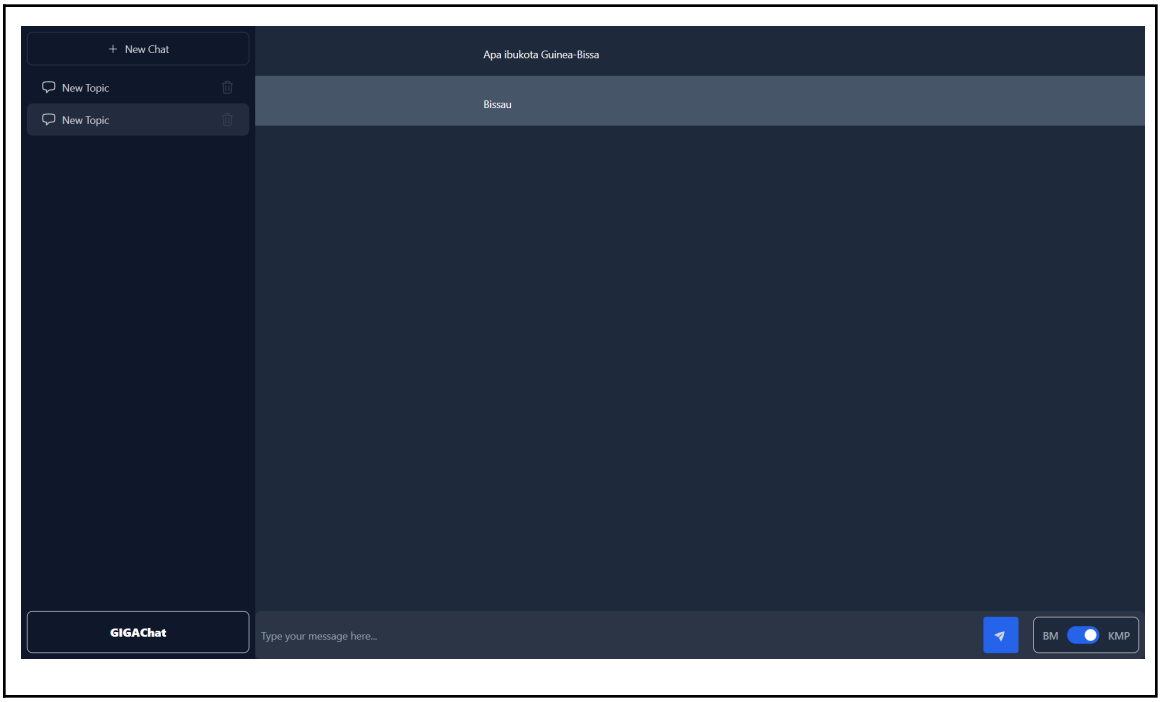
2. Fitur Kalkulator



3. Fitur Tanggal

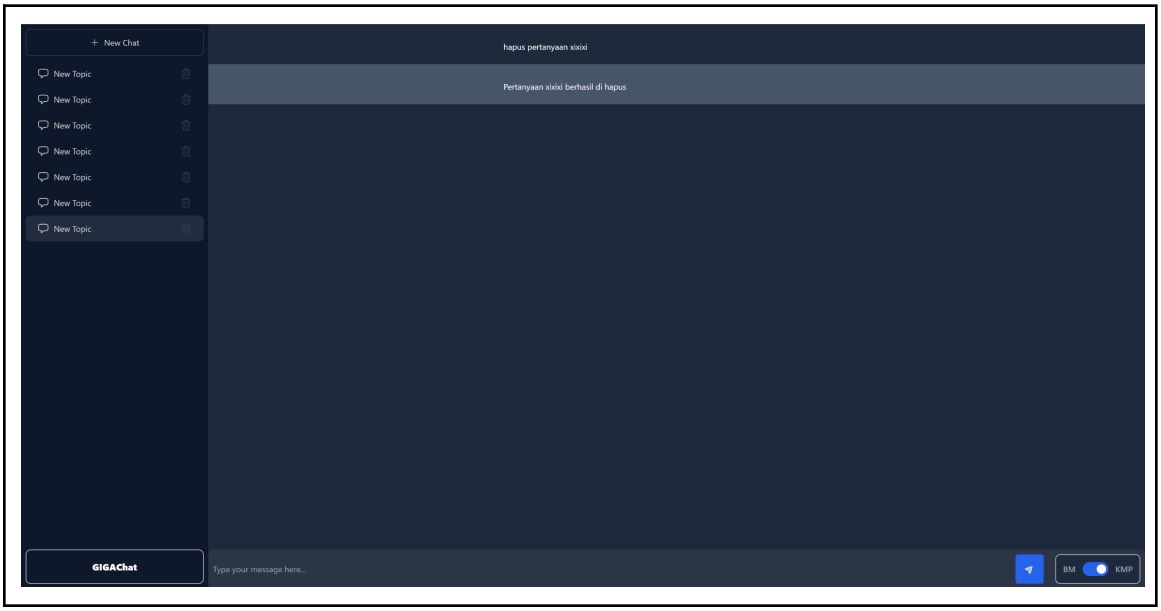
4. Fitur Randomize

5. Fitur Pencarian Pertanyaan



6. Fitur Insert/Update Pertanyaan

7. Fitur Delete Pertanyaan



4.4. Analisis Hasil Pengujian

Regular Expression memiliki peranan penting di dalam pengembangan aplikasi ini. Hal ini dikarena fleksibilitas yang ditawarkan oleh Regular Expression. Namun, khusus untuk fitur yang memerlukan akses ke database, algoritma KMP dan BM memiliki peranan yang penting karena telah melakukan pemrosesan String Matching

yang manual dan efektif. Hal ini menggantikan fungsi indexOf yang dimiliki oleh string (fungsi bawaan tipe string).

Dari sisi pengguna, pengguna algoritma KMP dan BM tidak terlalu terlihat karena KMP dan BM berada pada sisi backend aplikasi (yang memang seharusnya tidak perlu dilihat oleh pengguna). Algoritma KMP dan BM melakukan pemrosesan awal sebelum melakukan proses string matching. Pra pemrosesan ini sangat membantu untuk proses pencarian string sehingga proses string matching dapat dilakukan dengan lebih efektif.

Proses delete masih terkendala database, terdapat kesalahan yang tidak diketahui yang membuat database tidak bisa menghapus data yang ia miliki namun bisa membaca data yang berada didalamnya. Walaupun demikian, aplikasi tetap dapat berfungsi dengan baik meskipun terdapat banyak sekali kesalahan yang ada.

BAB 5
PENUTUP

5.1. Kesimpulan

Knuth-Morris-Pratt dan Boyer Moore merupakan algoritma string matching yang jauh lebih efektif dibandingkan dengan algoritma brute force. Algoritma KMP dan BM melakukan pra pemrosesan sebelum melakukan proses pencocokan string pada text. Pra pemrosesan ini sangat berdampak pada kompleksitas algoritma pencocokan sehingga kita tidak perlu mencocokkan string satu per satu.

5.2. Saran, Komentar, dan Refleksi

Tugas besar kali ini merupakan tugas yang cukup menantang. Tugas ini membuat eksplorasi mandiri yang kami lakukan lebih jauh daripada yang kami kira. Tugas ini sangat membantu kami dalam memahami algoritma string matching melalui pengembangan aplikasi chatbot berbasis web ini. Aplikasi ini telah membantu kami dalam memahami dasar dasar pengembangan perangkat lunak berbasis web yang tentunya sangat menambah ilmu pengetahuan kami

Kami sadar masih sangat banyak kekurangan yang ada pada program ini. Hal ini disebabkan dengan banyaknya tugas beririsan yang ada pada waktu yang sama dengan pengembangan aplikasi web ini. Kami sangat senang dengan progress terbaik yang telah kami lakukan. Kami sadar bahwa tugas ini akan memiliki hasil yang lebih baik dengan pengaturan waktu yang lebih baik. Oleh karena itu, kami sangat senang dapat membuat semua hal ini dengan sangat baik.

DAFTAR PUSTAKA

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

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

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf

LAMPIRAN

LINK REPOSITORY

Link repository GitHub : https://github.com/melvinkj/Tubes3_13521052

Link Bonus :

https://docs.google.com/document/d/17mtIrw4KpDgNtbBX543V8qWw1wzaXsozTYwY2hfpmPg/edit?usp=sharing

PEMBAGIAN TUGAS

NIM	Nama	Tugas
13521052	Melvin Kent Jonathan	Frontend
13521074	Eugene Yap Jin Quan	Database
13521162	Antonio Natthan Krishna	Backend