

**TUGAS BESAR 3
IF2211 STRATEGI ALGORITMA
SEMESTER II TAHUN 2022/2023**

**PENERAPAN STRING MATCHING DAN REGULAR EXPRESSION
DALAM PEMBUATAN CHAT GPT SEDERHANA**



DISUSUN OLEH

Daniel Egiant Sitanggang	13521056
Ilham Akbar	13521068
Asyifa Nurul Shafira	13521125

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

DAFTAR ISI

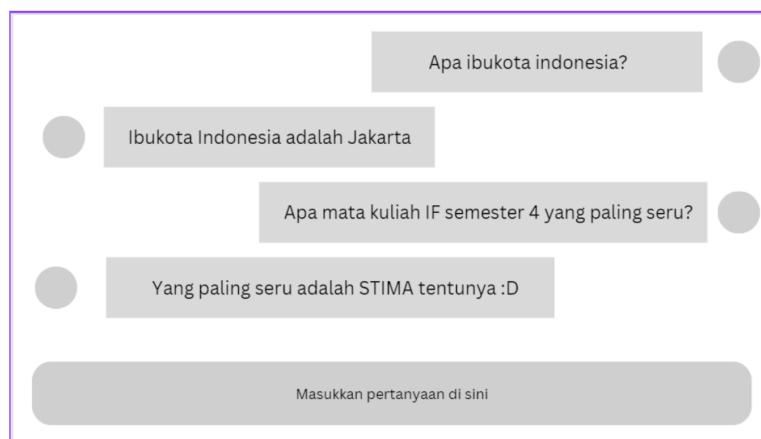
DAFTAR ISI.....	1
BAB 1 DESKRIPSI TUGAS.....	2
BAB 2 LANDASAN TEORI.....	4
2.1 Algoritma Knuth-Morris-Pratt.....	4
2.2 Algoritma Boyer-Moore.....	4
2.3 Regular Expression.....	5
2.4 Penjelasan Singkat Aplikasi Web yang Dibangun.....	6
BAB 3 ANALISIS PEMECAHAN MASALAH.....	7
3.1 Langkah Penyelesaian Masalah.....	7
3.1.1 Fitur pertanyaan teks.....	7
3.1.2 Fitur kalkulator.....	7
3.1.3 Fitur tanggal.....	8
3.1.4 Tambah pertanyaan dan jawaban ke database.....	8
3.1.5 Hapus pertanyaan dari database.....	9
3.2 Fitur Fungsional dan Arsitektur Aplikasi Web.....	10
3.2.1 Fitur Fungsional.....	10
3.2.2 Arsitektur Aplikasi Web.....	10
BAB 4 IMPLEMENTASI DAN PENGUJIAN.....	11
4.1 Spesifikasi Teknis Program.....	11
4.1.1 String Matching.....	11
4.1.1.1 Kelas PatternMatcher sebagai base class.....	11
4.1.1.2 Kelas BM.....	13
4.1.1.3 Kelas KMP.....	15
4.1.2 Date.....	17
4.1.3 MathEvaluator.....	18
4.2 Tata Cara Penggunaan Program.....	20
4.3 Hasil Pengujian.....	21
4.4 Analisis Hasil Pengujian.....	25
BAB 5 KESIMPULAN DAN SARAN.....	26
5.1 Kesimpulan.....	26
5.2 Saran.....	26
5.3 Komentar dan Refleksi.....	26
DAFTAR PUSTAKA.....	27

BAB 1

DESKRIPSI TUGAS

Pada tugas besar ini, kami membangun sebuah aplikasi ChatGPT sederhana dengan menerapkan pendekatan QA yang paling sederhana. Untuk menemukan pertanyaan yang paling mirip dengan input pengguna, kami memanfaatkan algoritma pencocokan string Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM) serta Regex untuk menentukan format dari pertanyaan. Jika tidak ada pertanyaan pada database yang tepat sesuai dengan input pengguna, aplikasi akan menggunakan pertanyaan paling mirip dengan kesamaan setidaknya 90%. Apabila tidak ada, maka chatbot akan memberikan maksimum 3 pilihan pertanyaan yang paling mirip untuk dipilih oleh pengguna.

Aplikasi ChatGPT sederhana ini memiliki beberapa fitur. Pertama, fitur pertanyaan teks yang dapat mencocokkan input pengguna dengan pertanyaan di database menggunakan algoritma KMP atau BM. Kedua, fitur kalkulator yang dapat menghitung operasi matematika sederhana seperti tambah, kurang, kali, bagi, pangkat, dan kurung. Ketiga, fitur tanggal yang akan memberitahukan hari pada tanggal tertentu yang di input oleh pengguna. Keempat, fitur tambah pertanyaan dan jawaban ke database, dimana pengguna dapat menambahkan pertanyaan dan jawabannya sendiri ke database, dan menggunakan algoritma string matching untuk memeriksa apakah pertanyaan sudah ada. Jika sudah, maka jawaban akan diperbarui. Terakhir, fitur hapus pertanyaan dari database, dimana pengguna dapat menghapus sebuah pertanyaan dari database dengan query dan menggunakan algoritma string matching untuk mencari pertanyaan yang sesuai dengan input pengguna. Berikut adalah contoh ilustrasi program.



Gambar 1.1 Ilustrasi Fitur Pertanyaan teks kasus exact

Program yang dibuat adalah sebuah aplikasi berbasis website yang terdiri dari *Frontend* dan *Backend* yang jelas terpisah. Backend diimplementasikan menggunakan Node.js, sedangkan Frontend menggunakan Next.js. Basis data yang digunakan untuk menyimpan informasi adalah

MySQL. Algoritma pencocokan string seperti KMP dan Boyer-Moore, serta Regex diimplementasikan pada sisi Backend. Informasi yang wajib disimpan pada basis data meliputi tabel pasangan pertanyaan dan jawaban, serta tabel history. Skema basis data setidaknya mencakup kedua informasi tersebut. Proses string matching tidak case sensitive, dan pencocokan yang dilakukan adalah dalam satu kesatuan string pertanyaan utuh, bukan kata per kata. Misalnya, pertanyaan "Apa ibukota Filipina?" akan dicocokkan sebagai sebuah string utuh, bukan sebagai tiga kata terpisah "apa", "ibukota", dan "Filipina".

BAB 2

LANDASAN TEORI

2.1 Algoritma Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt atau KMP adalah metode pencocokan pola yang didasarkan pada algoritma brute-force dengan kemampuan untuk melakukan pergeseran posisi yang lebih banyak daripada algoritma brute-force. Hal ini membuat KMP lebih efisien dalam memeriksa karakter. Jika terdapat ketidakcocokan pada $P[j]$ antara teks dan pola P , maka pergeseran posisi pola terbesar yang dapat dilakukan adalah ukuran prefix $P[0..j-1]$ yang juga merupakan suffix dari $P[1..j-1]$. Nilai pergeseran ini disebut sebagai fungsi pinggiran KMP atau border function $b(k)$ yang dihitung sebelum melakukan pencocokan string. Setelah fungsi pinggiran dihitung, pencarian string dimulai dengan pergeseran posisi yang mengikuti fungsi tersebut ketika terdapat ketidakcocokan. Berikut ini adalah contoh dari proses pencocokan pola dengan menggunakan algoritma KMP. Kompleksitas dari algoritma KMP adalah $O(m + n)$

T:	a	b	a	c	a	a	b	a	c	c	a	b	a	c	a	b	a	a	b	b	
	1	2	3	4	5	6															
P:	a	b	a	c	a	b															
							7														
								a	b	a	c	a	b								
									8	9	10	11	12								
										a	b	a	c	a	b						
															13						
																a	b	a	c	a	b
																14	15	16	17	18	19
																a	b	a	c	a	b
j	0	1	2	3	4	5															
$P[j]$	a	b	a	c	a	b															
k	-	0	1	2	3	4															
$b(k)$	-	0	0	1	0	1															

Gambar 2.1 Contoh Pencocokan Pola dengan Algoritma KMP

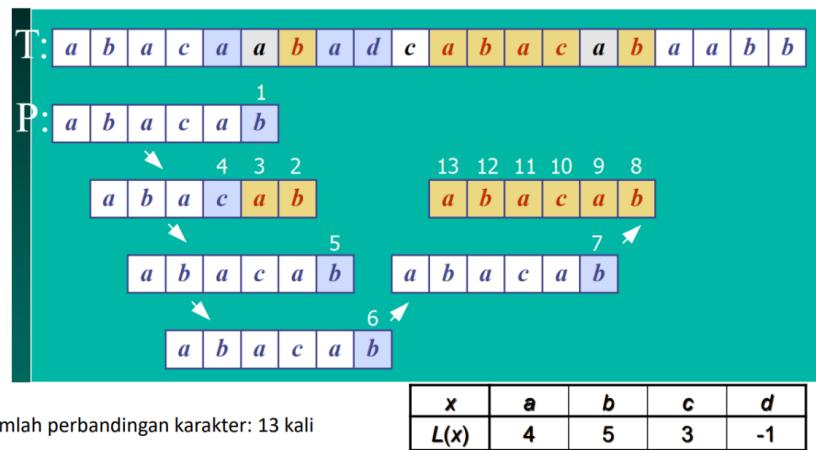
2.2 Algoritma Boyer-Moore

Algoritma pencocokan pola Boyer-Moore mengkombinasikan 2 teknik, yaitu looking-glass technique dan character-jump technique. Looking-glass technique mencocokkan pola P pada T dimulai dari akhir P secara mundur, sedangkan character-jump technique memeriksa ketidakcocokan karakter dan melakukan salah satu dari 3 tindakan berikut:

1. jika pola P mengandung karakter yang menyebabkan ketidakcocokan pada T , geser P ke kanan sehingga karakter di T sejajar dengan kemunculan terakhir karakter yang sama pada P ,
2. jika pergeseran ke kanan tidak memungkinkan menghasilkan kondisi (1), geser P sebanyak 1 karakter,

3. jika pola P tidak memenuhi kondisi (1) dan (2), geser P sehingga karakter awal pola sejajar dengan satu karakter setelah karakter pada T yang menimbulkan ketidak cocokan,

Berikut adalah ilustrasi pencocokan pola menggunakan algoritma Boyer-Moore.



Gambar 2.2 Contoh Pencocokan Pola dengan Algoritma BM

Sama seperti algoritma KMP, algoritma Boyer-Moore juga melakukan tahap pra-pemrosesan sebelum mencocokkan pola. Langkah ini melibatkan perhitungan fungsi penampakan terakhir (last occurrence function) $L()$ untuk setiap karakter dalam alfabet. Algoritma ini berkinerja cepat untuk alfabet yang besar, tetapi relatif lambat untuk alfabet yang kecil seperti pada bilangan biner. Kompleksitas dari algoritma Boyer-Moore adalah $O(nm + A)$.

2.3 Regular Expression

Regular Expression (RE) adalah suatu notasi yang digunakan untuk menggambarkan pola dari suatu kata yang ingin ditemukan. Konsep mengenai RE muncul pada tahun 1951 ketika seorang ilmuwan matematika bernama Stephen Cole Kleene merumuskan definisi tentang bahasa formal. Dalam industri pemrograman, RE dimanfaatkan untuk berbagai keperluan seperti validasi data, pencarian, fitur pencarian dan penggantian, dan sebagainya. Beberapa contoh RE yang umum digunakan.

Character classes	
.	any character except newline
\w \d \s	word, digit, whitespace
\W \D \S	not word, digit, whitespace
[abc]	any of a, b, or c
[^abc]	not a, b, or c
[a-g]	character between a & g
Anchors	
^abc\$	start / end of the string
\b	word boundary
Escaped characters	
\. * \\	escaped special characters
\t \n \r	tab, linefeed, carriage return
\u00A9	unicode escaped ©
Groups & Lookaround	
(abc)	capture group
\1	backreference to group #1
(?:abc)	non-capturing group
(?=abc)	positive lookahead
(?!abc)	negative lookahead
Quantifiers & Alternation	
a* a+ a?	0 or more, 1 or more, 0 or 1
a{5} a{2,}	exactly five, two or more
a{1,3}	between one & three
a+? a{2,}?match as few as possible	
ab cd	match ab or cd

Gambar 2.3 Beberapa Regular Expression yang Sering Dipakai

2.4 Penjelasan Singkat Aplikasi Web yang Dibangun

Aplikasi berbasis web yang dibangun pada tugas besar ini adalah sebuah aplikasi ChatGPT sederhana. Pada aplikasi ini, terdapat beberapa fitur yang dapat diakses oleh pengguna, diantaranya fitur pertanyaan teks, kalkulator, tanggal, tambah pertanyaan dan jawaban ke database, serta hapus pertanyaan dari database. Aplikasi ini mengklasifikasikan query dari input pengguna menggunakan regex dan algoritma string matching KMP dan BM yang dicocokan dengan query pertanyaan yang terdapat pada database. Ketika input pengguna sesuai dengan query pertanyaan yang terdapat pada database atau setidaknya memiliki kemiripan minimal 90%, maka aplikasi akan mengeluarkan jawaban yang sesuai. Namun jika tidak ada, maka pengguna dapat memilih 3 pertanyaan termirip yang ditawarkan oleh aplikasi. Selain itu, jika pengguna juga dapat menambahkan pertanyaan atau jawaban yang belum terdapat di database.

BAB 3

ANALISIS PEMECAHAN MASALAH

3.1 Langkah Penyelesaian Masalah

3.1.1 Fitur pertanyaan teks

Berikut adalah langkah-langkah penyelesaian masalah untuk mencocokkan pertanyaan dari input pengguna ke pertanyaan di database menggunakan algoritma KMP atau BM:

1. Pertama-tama, dapatkan database pertanyaan teks yang akan digunakan sebagai referensi dalam pencocokan. Pastikan database tersebut memiliki informasi yang cukup dan relevan dengan topik yang akan dibahas.
2. Selanjutnya, dapatkan input dari pengguna yang akan dicocokkan dengan database pertanyaan teks. Pastikan input tersebut sudah diolah dan dihilangkan kata-kata yang tidak relevan seperti kata tanya, kata hubung, dan lain-lain.
3. Pilih algoritma yang akan digunakan dalam pencocokan, dalam hal ini bisa menggunakan algoritma KMP atau BM. Keduanya adalah algoritma string matching yang digunakan untuk mencocokkan pola dalam sebuah string.
4. Terapkan algoritma tersebut pada database pertanyaan teks. Pada saat ini, setiap pertanyaan akan diubah menjadi string dan akan diproses dengan algoritma yang dipilih.
5. Setelah proses pencocokan selesai dilakukan, dapatkan hasil dari proses tersebut. Hasilnya bisa berupa indeks dari pertanyaan yang cocok dengan input pengguna atau dapat berupa seluruh pertanyaan yang cocok dengan input pengguna.

3.1.2 Fitur kalkulator

Berikut adalah langkah-langkah penyelesaian masalah untuk membuat fitur kalkulator yang mampu menerima input query berupa persamaan matematika:

1. Pertama-tama, dapatkan input query dari pengguna dalam bentuk string. Pastikan bahwa input tersebut valid, yaitu hanya mengandung karakter numerik, tanda operasi matematika (+, -, *, /, ^), dan tanda kurung.
2. Lakukan parsing atau analisis sintaksis pada input query tersebut. Parsing adalah proses mengubah input query menjadi suatu struktur data yang dapat diproses oleh kalkulator. Dalam hal ini, parsing bertujuan untuk mengenali tanda kurung dan menentukan urutan operasi yang harus dilakukan.
3. Implementasikan algoritma untuk menyelesaikan persamaan matematika tersebut. Pada dasarnya, langkah-langkah dalam menyelesaikan persamaan matematika

adalah sebagai berikut: a. Hitung operasi pangkat terlebih dahulu. b. Hitung operasi perkalian dan pembagian secara bergantian, dari kiri ke kanan. c. Hitung operasi penjumlahan dan pengurangan secara bergantian, dari kiri ke kanan.

4. Tampilkan hasil kalkulasi pada pengguna. Pastikan bahwa hasil yang ditampilkan sesuai dengan format yang diinginkan.

3.1.3 Fitur tanggal

Berikut adalah langkah-langkah penyelesaian masalah untuk membuat fitur tanggal yang dapat memberikan respon hari pada tanggal tertentu:

1. Pertama-tama, dapatkan input tanggal dari pengguna dalam bentuk string. Pastikan bahwa input tersebut valid, yaitu dalam format tanggal yang benar seperti dd/mm/yyyy atau mm/dd/yyyy.
2. Lakukan parsing atau analisis sintaksis pada input tanggal tersebut untuk memastikan bahwa tanggal tersebut benar dan valid. Pada tahap ini, pastikan bahwa input tanggal tersebut mengandung tiga bagian yaitu tanggal, bulan, dan tahun serta nilai-nilai pada bagian tersebut juga valid, misalnya tanggal tidak melebihi jumlah hari pada bulan yang bersangkutan.
3. Konversikan input tanggal tersebut menjadi bentuk yang dapat diproses oleh program. Dalam hal ini, konversikan tanggal tersebut menjadi bentuk angka atau bilangan yang dapat direpresentasikan sebagai nilai integer.
4. Hitung hari pada tanggal tersebut. Terdapat beberapa metode yang dapat digunakan untuk menghitung hari pada tanggal tertentu, salah satunya adalah dengan menggunakan algoritma Zeller. Algoritma Zeller dapat menghitung hari pada tanggal tertentu dengan memanfaatkan formula matematika.
5. Tampilkan hasil perhitungan pada pengguna dalam bentuk hari. Pastikan bahwa hasil yang ditampilkan sesuai dengan format yang diinginkan, misalnya dalam bahasa Indonesia atau bahasa Inggris.

3.1.4 Tambah pertanyaan dan jawaban ke database

Berikut adalah langkah-langkah penyelesaian masalah untuk membuat fitur menambahkan pertanyaan dan jawaban ke database dengan algoritma string matching:

1. Pertama-tama, dapatkan input query dari pengguna dalam bentuk string. Pastikan bahwa input tersebut valid dan sesuai dengan format yang diinginkan, yaitu "Tambahkan pertanyaan xxx dengan jawaban yyy".
2. Lakukan parsing atau analisis sintaksis pada input query tersebut untuk memisahkan bagian pertanyaan dan jawaban yang ingin ditambahkan. Dalam hal

ini, pastikan bahwa input query tersebut mengandung tiga bagian yaitu "Tambahkan pertanyaan", isi pertanyaan, dan isi jawaban.

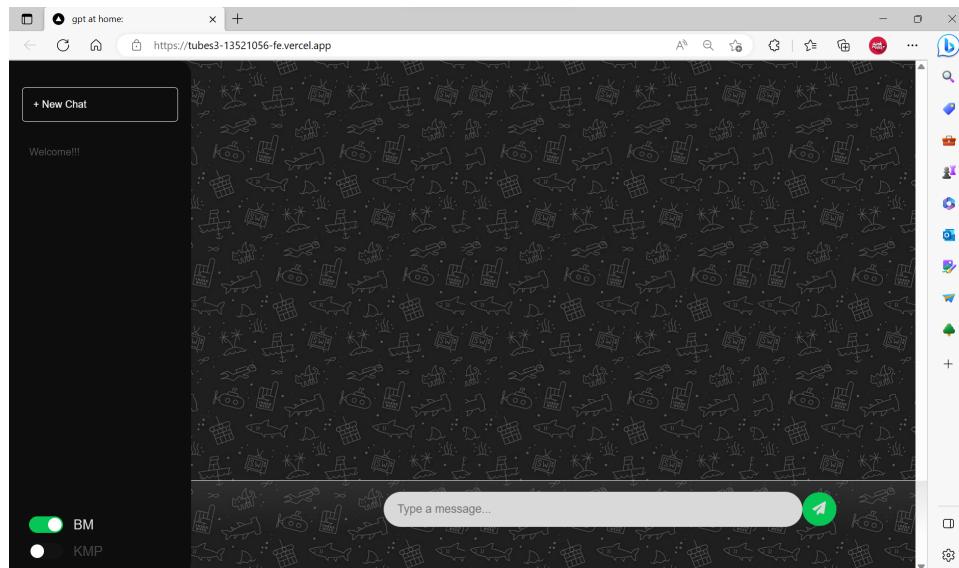
3. Gunakan algoritma string matching, seperti KMP atau BM, untuk mencari apakah pertanyaan yang ingin ditambahkan sudah ada di dalam database. Dalam hal ini, pastikan bahwa pengguna tidak menambahkan pertanyaan yang sudah ada di dalam database, sehingga data di dalam database tetap konsisten.
4. Jika pertanyaan sudah ada di dalam database, perbarui jawaban untuk pertanyaan tersebut. Jika pertanyaan belum ada di dalam database, tambahkan pertanyaan dan jawaban baru ke dalam database.
5. Tampilkan konfirmasi kepada pengguna bahwa pertanyaan dan jawaban telah ditambahkan atau diperbarui di dalam database.

3.1.5 Hapus pertanyaan dari database

Berikut adalah langkah-langkah penyelesaian masalah untuk membuat fitur menghapus pertanyaan dari database:

1. Pertama-tama, dapatkan input query dari pengguna dalam bentuk string. Pastikan bahwa input tersebut valid dan sesuai dengan format yang diinginkan, yaitu "Hapus pertanyaan xxx".
2. Lakukan parsing atau analisis sintaksis pada input query tersebut. Dalam hal ini, pastikan bahwa input query tersebut mengandung dua bagian yaitu "Hapus pertanyaan" dan isi pertanyaan.
3. Gunakan algoritma string matching, seperti KMP atau BM, untuk mencari apakah pertanyaan yang ingin dihapus terdapat pada database.
4. Jika pertanyaan ada di dalam database, hapus pertanyaan serta jawaban untuk pertanyaan tersebut. Jika pertanyaan tidak ada di dalam database, maka akan muncul pesan kesalahan.
5. Tampilkan konfirmasi kepada pengguna bahwa pertanyaan dan jawaban telah berhasil dihapus dari database.

3.2 Fitur Fungsional dan Arsitektur Aplikasi Web



Gambar 3.1 Screenshot Aplikasi gpt at home:

3.2.1 Fitur Fungsional

Pada aplikasi ini terdapat beberapa fitur fungsional yaitu fitur utama chat bot, fitur toggle untuk memilih algoritma yang akan digunakan, dan fitur history chat yang menyimpan *history* chat dengan program di masa lampau.

3.2.2 Arsitektur Aplikasi Web

Aplikasi web ini dibagi menjadi tiga bagian utama, yaitu frontend, backend, dan basis data.

Bagian frontend adalah bagian antarmuka program yang berinteraksi langsung dengan pengguna. Bagian ini terdiri dari seluruh komponen yang terlihat pada aplikasi ketika dijalankan. Pada aplikasi ini, bagian front-end dibangun dengan bahasa pemrograman Java Script dengan framework Next.js.

Bagian backend adalah bagian yang bertugas untuk memberikan respons terhadap masukan pengguna. Respons ini termasuk melakukan validasi terhadap masukan, mengkalkulasi kecocokan input yang dimasukkan oleh pengguna, serta memberi perintah ke basis data.

Bagian basis data bertugas untuk menampung data yang diperlukan pada *secondary memory*. Data yang disimpan berupa pertanyaan beserta jawabannya, dan history dari percakapan.

BAB 4

IMPLEMENTASI DAN PENGUJIAN

4.1 Spesifikasi Teknis Program

4.1.1 String Matching

4.1.1.1 Kelas PatternMatcher sebagai *base class*

```
class PatternMatcher {
    constructor(pattern) {
        this.pattern = pattern;
    }

    levenshteinDistance(content) {
        /*
            function to find lavenshteinDistance of transforming
            the pattern to
            given content or otherwise
        */
        content = content.toLowerCase();
        const pattern = this.pattern;
        const patternLength = pattern.length;
        const contentLength = content.length;

        /*
            initiliazing DP table [i, j]: levenshtein-distance
            transforming the
            pattern[i..end] to the content[j..end]
        */
        const distanceMatrix = [];
        /*
            base case, when the pattern or content is "" thus the
            distance is the length
            of string itself (the cost of deletion)
        */
        for (let i = 0; i <= patternLength; i++) {
```

```

        distanceMatrix[i] = [i];
    }
    for (let j = 0; j <= contentLength; j++) {
        distanceMatrix[0][j] = j;
    }

    /*
     reccurence:
         suppose we want to evaluate the distance
transforming pattern[i..end]
            to content[j..end]
            if the char at i and char at j is the same, when
the cost is the cost
                of transforming pattern[i-1..end] to
content[j-1..end]
            if we delete char at i of the pattern, then the
cost is 1 + distance
                from pattern[i-1..end] to content[j..end]
            if we insert char at i to the pattern, then the
cost is 1 + distance
                from pattern[i..end] to content[j..end]
            if we substitute char at i to match char at j
then the cost is
                1 + distance from from pattern[i-1..end] to
content[j-1..end]
    */
    for (let i = 1; i <= patternLength; i++) {
        for (let j = 1; j <= contentLength; j++) {
            const substitutionCost = pattern[i - 1] ===
content[j - 1] ? 0 : 1;
            distanceMatrix[i][j] = Math.min(
                distanceMatrix[i - 1][j] + 1,
// deletion
                distanceMatrix[i][j - 1] + 1,
// insertion
                distanceMatrix[i - 1][j - 1] +
substitutionCost,      // substitution
            );
        }
    }

```

```

        }
        const maxLength = Math.max(patternLength,
contentLength);
        return (1 -
distanceMatrix[patternLength] [contentLength] / maxLength);

    }
}

module.exports = PatternMatcher;

```

4.1.1.2 Kelas BM

```

const { Module } = require('module');
const PatternMatcher = require('./PatternMatcher');

class BM extends PatternMatcher {
    constructor(pattern) {
        super(pattern);
        /*
         (pre-)compute the jump count/ last-occurrence
        */
        this.lastOccurrence = new Map();
        for (let i = 0; i < pattern.length; i++) {
            this.lastOccurrence.set(pattern[i], i);
        }
    }

    getLastOccurrence(char) {
        /*
         function that return the value of last occurrence/jump
        count that had been
         pre-computed when the pattern initialized
        */
        if (!this.lastOccurrence.has(char)) {
            return -1;
        }
        return this.lastOccurrence.get(char);
    }
}

```

```

    }

    match(content) {
        /*
         * match the pattern to given content using Boyer-Moore
         * technique
        */
        let patternLength = this.pattern.length;
        let contentLength = content.length;

        if (patternLength >= contentLength) {
            return false;
        }
        content = content.toLowerCase();
        let pattern = this.pattern;
        let patternPointer = patternLength - 1;
        let contentPointer = patternLength - 1;
        do {
            if (pattern[patternPointer] ===
content[contentPointer]) {
                if (patternPointer == 0) {
                    return true;
                }
                /* Looking-Glass Technique */
                patternPointer--;
                contentPointer--;
            } else {
                /* Character Jump Technique */
                let lastOccurence =
this.getLastOccurrence(content[contentPointer])
                let jump = patternLength -
Math.min(patternPointer, 1 + lastOccurence);
                contentPointer += jump;
                patternPointer = patternLength - 1;
            }
        } while (contentPointer <= contentLength - 1);

        return false;
    }
}

```

```
}
```

```
module.exports = BM;
```

4.1.1.3 Kelas KMP

```
const PatternMatcher = require('./PatternMatcher');

class KMP extends PatternMatcher{
    constructor(pattern) {
        super(pattern);

        /*
        (pre-)compute the border-function
        */
        this.border = new Array(pattern.length + 1);
        this.border[0] = -1;
        this.border[1] = 0;
        let i = 1;
        let prefixLen = 0;
        while (i < pattern.length) {
            if (pattern[prefixLen] == pattern[i]) {
                prefixLen++;
                i++;
                this.border[i] = prefixLen;
            } else if (prefixLen > 0) {
                prefixLen = this.border[prefixLen];
            } else {
                i++;
                this.border[i] = 0;
            }
        }
    }

    match(content) {
        /*
        match the pattern to given content using
        Knuth-Morris-Pratt technique
    }
}
```

```

*/
let patternLength = this.pattern.length;
let contentLength = content.length;

if (patternLength >= contentLength) {
    return false;
}

content = content.toLowerCase();
let contentPointer = 0;
let patternPointer = 0;
let pattern = this.pattern;
while (contentPointer < contentLength) {
    const currentCharMatch = pattern[patternPointer]
==== content[contentPointer];
    if (currentCharMatch) {
        patternPointer++;
        contentPointer++;
        const allPatternCorrect = patternPointer ==
patternLength
        if (allPatternCorrect) {
            return true;
        }
    } else {
        patternPointer = this.border[patternPointer];
        if (patternPointer < 0) {
            contentPointer++;
            patternPointer++;
        }
    }
}
return false;
}

module.exports = KMP;

```

4.1.2 Date

```
class Date {
    static dayName(day, month, year) {
        /*
            function return name of the day from given date using
        Zeller's congruence
        */
        const days = ['Sabtu', 'Minggu', 'Senin', 'Selasa', 'Rabu',
        'Kamis', 'Jumat'];

        if (month < 3) {
            month += 12;
            year--;
        }

        let q = day;
        let m = month;
        let k = year % 100;
        let j = Math.floor(year / 100);
        let h = (
            q
            + Math.floor((13 * (m + 1)) / 5)
            + k
            + Math.floor(k / 4)
            + Math.floor(j / 4)
            + 5 * j
        ) % 7;

        return days[h];
    }

    static isValid(day, month, year) {
        const daysInMonth = [31, 28, 31, 30, 31, 30, 31, 31, 30,
        31, 30, 31];
        if (month < 1 || month > 12) {
            return false;
        }
        if (month === 2 && (year % 400 === 0 || (year % 100 !== 0
        && year % 4 === 0)) && day === 29) {
    
```

```

        return true;
    }
    return (day >= 0 && day <= daysInMonth[month - 1]);
}
}

module.exports = Date;

```

4.1.3 MathEvaluator

```

class MathEvaluator {

    constructor(expression) {
        this.operandStack = [];
        this.operatorStack = [];
        this.token = expression.match(/\d+|[+\-*\/()]/g);
        this.precedence = {
            '+': 1,
            '-': 1,
            '*': 2,
            '/': 2,
        };
    }

    doOperation() {
        let rightOperand = this.operandStack.pop();
        let leftOperand = this.operandStack.pop();
        let operator = this.operatorStack.pop();
        switch (operator) {
            case '+':
                this.operandStack.push(leftOperand + rightOperand);
                break;
            case '-':
                this.operandStack.push(leftOperand - rightOperand);
                break;
            case '*':
                this.operandStack.push(leftOperand * rightOperand);
                break;
        }
    }
}

```

```

        break;
    case '/':
        this.operandStack.push(leftOperand / rightOperand);
        break;

    }
}

evaluate() {
    this.token.forEach(currToken => {
        const currNumber = parseFloat(currToken);
        const currTokenIsNumber =
!isNaN(parseFloat(currNumber)) && isFinite(currNumber);
        if (currTokenIsNumber) {
            this.operandStack.push(currNumber);
        } else if (currToken === '(') {
            this.operatorStack.push(currToken);
        } else if (currToken === ')') {
            while (this.operatorStack[this.operatorStack.length - 1] !== '(') {
                this.doOperation();
            }
            this.operatorStack.pop();
        } else {
            while (this.operatorStack.length > 0 &&
this.precedence[this.operatorStack[this.operatorStack.length - 1]] >= this.precedence[currToken]
                ) {
                this.doOperation();
            }
            this.operatorStack.push(currToken);
        }
    });
    while (this.operatorStack.length > 0) {
        this.doOperation();
    }

    if (this.operatorStack.length !== 0) {

```

```
        return "Invalid Math Expression";
    }
    return this.operandStack.pop();
}
}

module.exports = MathEvaluator;
```

4.2 Tata Cara Penggunaan Program

4.2.1 Alternatif 1, vercel-app

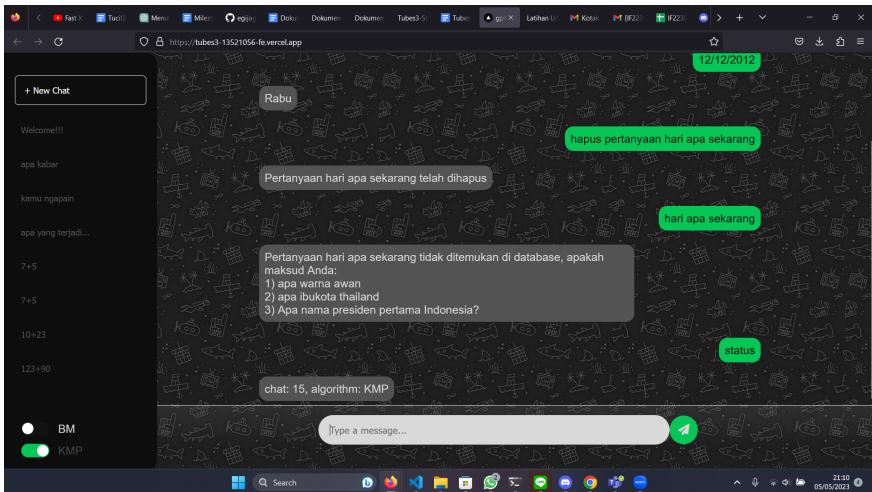
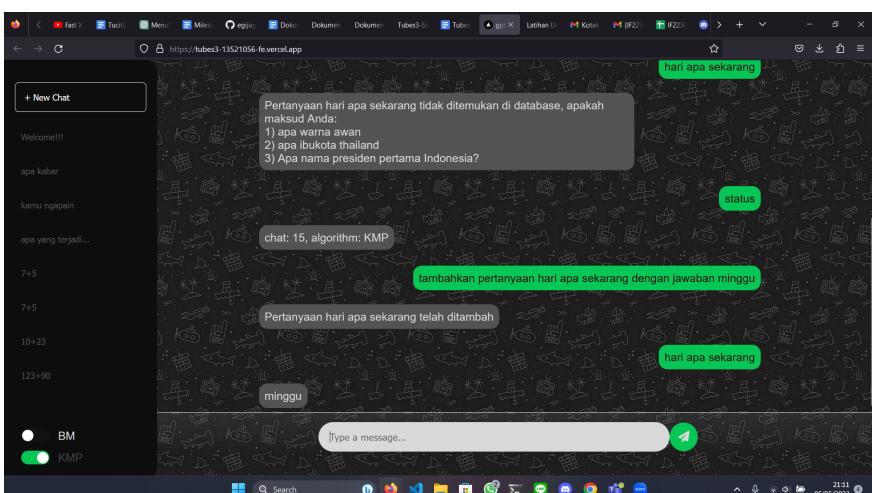
- buka link <https://tubes3-13521056-fe.vercel.app/> pada browser

4.2.2 Alternatif 2, hosting secara local

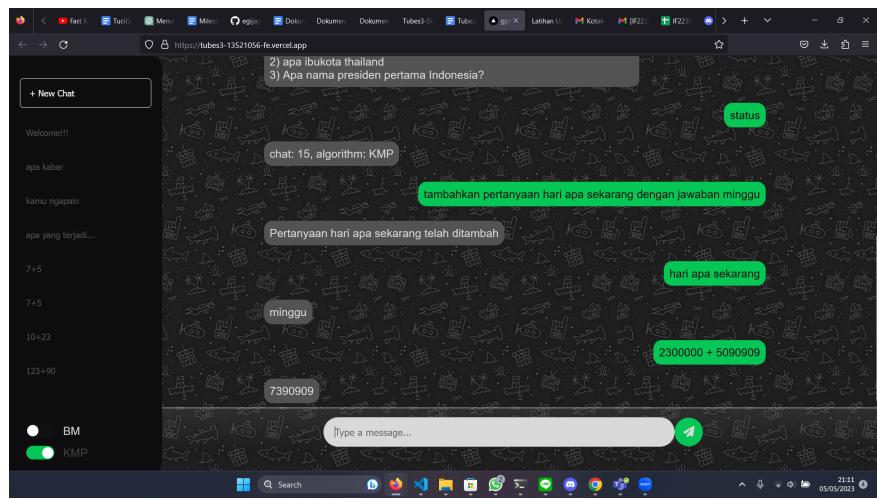
- Pastikan semua requirements: next.js, node.js, mysql telah tersedia
- clone repository back-end dengan git clone https://github.com/egijago/Tubes3_13521056 lalu buka branch local-hosting
- clone repository front-end dengan git clone https://github.com/egijago/Tubes3_13521056_fe lalu buka branch local-hosting
- setup database mysql lokal dengan memasukan query pada 'setup.sql'
- pastikan credential pada './src/DBController' benar
- masukkan npm install pada repository back-end dan front-end untuk menginstall dependencies, jika gagal hapus node_module dan package-lock.json
- masukkan node index.js pada repository back-end dan npm run dev pada repository front-end
- program dapat dilihat <http://localhost:3000/>

seluruh query pada prompt mengikuti spesifikasi pada <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2022-2023/Tubes3-Stima-2023.pdf>

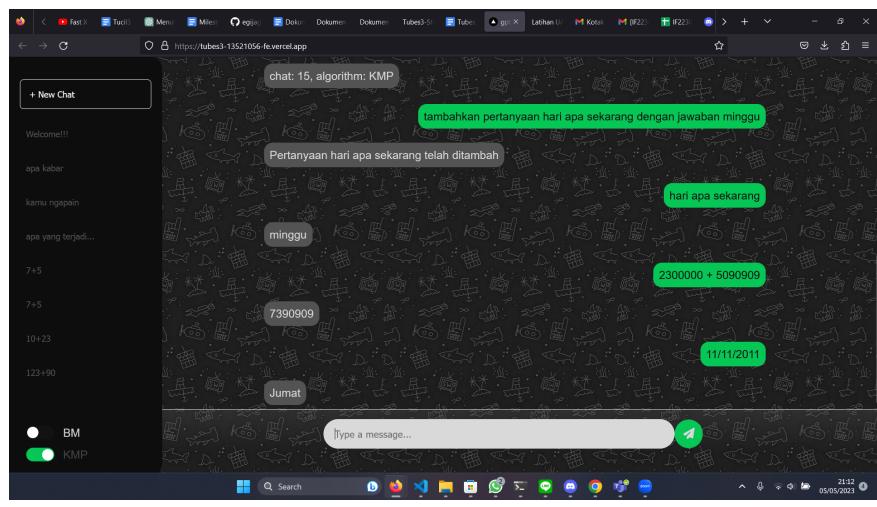
4.3 Hasil Pengujian

Hasil Pengujian dengan Menggunakan Algoritma Knuth-Morris-Pratt	
Cek Status Algoritma KMP	 <p>The screenshot shows a web browser window with a chat interface. On the left, there is a sidebar with various input fields and a toggle switch between 'BM' and 'KMP'. The main area shows a conversation history with messages like 'Welcome!!!', 'apa kabar', 'kamu ngapain', 'apa yang terjadi...', '7+5', '7+5', '10+23', and '123+90'. A message from the user 'Rabu' says 'Pertanyaan hari apa sekarang telah dihapus'. A response message says 'Pertanyaan hari apa sekarang tidak ditemukan di database, apakah maksud Anda: 1) apa warna awan 2) apa ibukota thailand 3) Apa nama presiden pertama Indonesia?'. A green button labeled 'status' is visible. The status bar at the bottom indicates 'chat: 15, algoritm: KMP'.</p>
Menambahkan Pertanyaan	 <p>The screenshot shows a similar web-based chat application. The sidebar includes 'BM' and 'KMP' buttons. The conversation history is identical to the first screenshot. A message from the user 'Rabu' says 'tambahkan pertanyaan hari apa sekarang dengan jawaban minggu'. A response message says 'Pertanyaan hari apa sekarang telah ditambah'. A green button labeled 'status' is visible. The status bar at the bottom indicates 'chat: 15, algoritm: KMP'.</p>

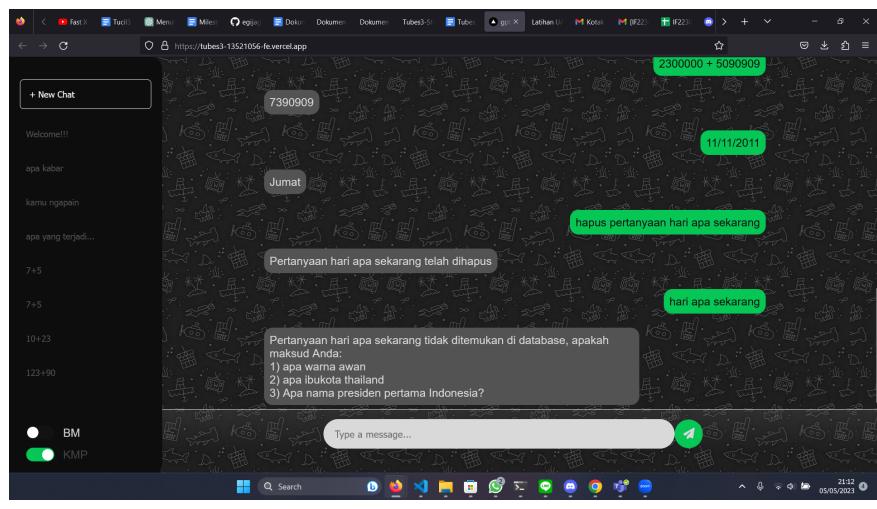
Kalkulator



Tanggal

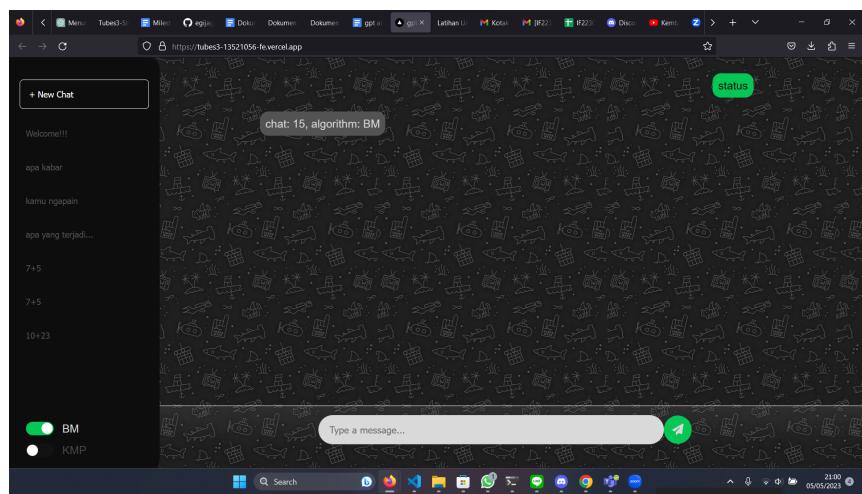


Menghapus Pertanyaan

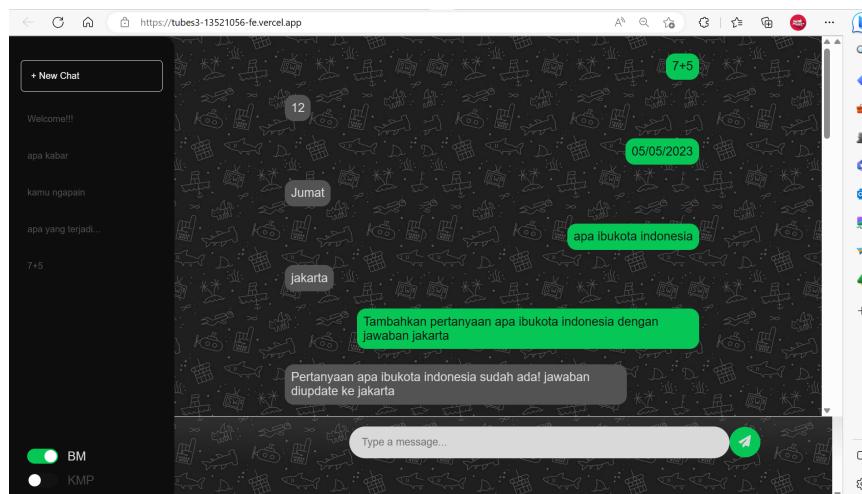


Hasil Pengujian dengan Menggunakan Algoritma Boyer-Moore

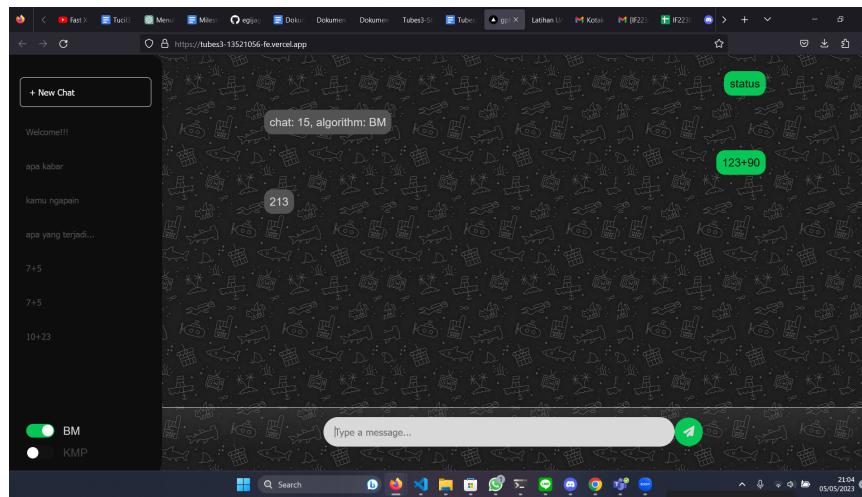
Cek Status Algoritma BM



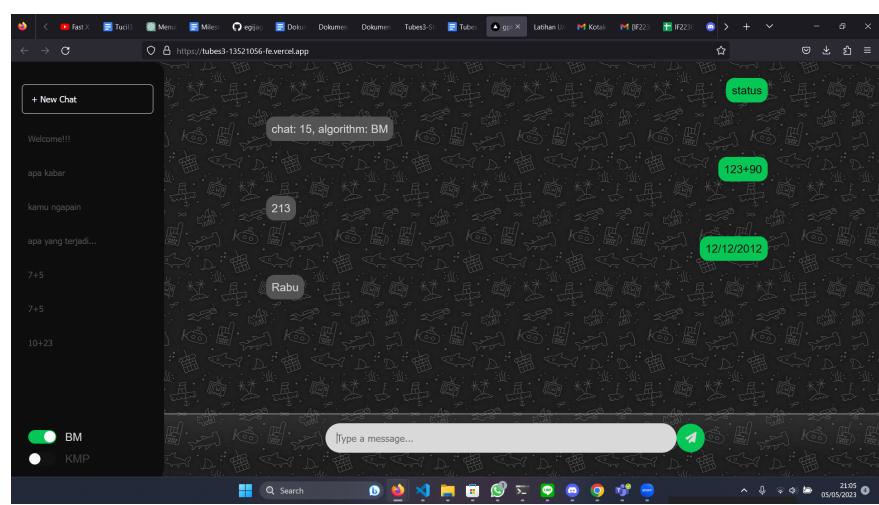
Menambahkan Pertanyaan



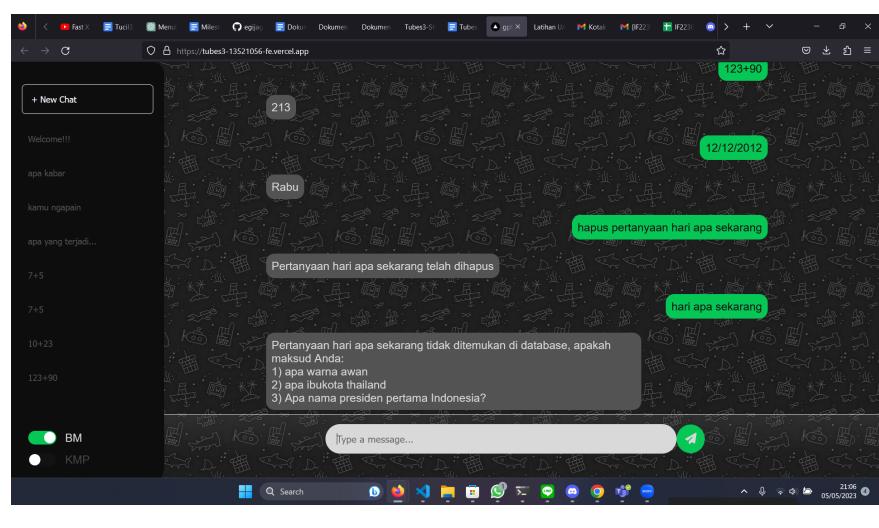
Kalkulator



Tanggal



Menghapus Pertanyaan



4.4 Analisis Hasil Pengujian

Berdasarkan hasil pengujian di atas, terlihat bahwa aplikasi web yang kami buat dapat melakukan beberapa hal berikut.

- Dapat mendeteksi suatu pertanyaan berupa operasi matematika dan menampilkan hasil operasinya.
- Dapat mendeteksi suatu pertanyaan berupa tanggal dan dapat menampilkan hari pada tanggal tersebut.
- Dapat mendeteksi suatu pertanyaan berupa teks yang dicocokan dengan pertanyaan pada database menggunakan algoritma string matching BM ataupun KMP. Jika pertanyaan terdapat pada database, maka aplikasi akan menampilkan pasangan jawabannya. Namun, jika pertanyaan tidak ada pada database, aplikasi akan menampilkan jawaban dari pertanyaan pada database yang memiliki kemiripan setidaknya 90% dengan pertanyaan pengguna, tapi jika tidak ditemukan pertanyaan yang memiliki kemiripan sebesar itu, maka aplikasi akan menampilkan tiga pertanyaan termirip yang dapat dipilih oleh pengguna.
- Dapat menambahkan pertanyaan dan jawaban baru ke database berdasarkan input dari pengguna. Aplikasi akan melakukan pengecekan apakah pertanyaan tersebut sudah ada di dalam database dengan menggunakan algoritma BM ataupun KMP. Jika pertanyaan belum ada, maka pertanyaan tersebut akan ditambahkan dan aplikasi menampilkan pesan berhasil menambahkan. Namun, jika pertanyaan tersebut sudah ada, maka aplikasi akan menampilkan pesan bahwa pertanyaan sudah ada dan meng-update jawaban yang baru sesuai dengan input pengguna.
- Dapat menghapus pertanyaan beserta jawaban dari database berdasarkan input pengguna. Aplikasi akan melakukan pengecekan apakah pertanyaan tersebut sudah ada di dalam database dengan menggunakan algoritma BM ataupun KMP. Jika pertanyaan ada, maka aplikasi akan menghapus pertanyaan tersebut beserta jawabannya dari database. Namun, jika pertanyaan tidak ada, maka aplikasi akan menampilkan pesan bahwa pertanyaan tersebut tidak ada pada database.
- Dapat memberitahukan algoritma apa yang sedang digunakan untuk melakukan string matching dengan mengetik “status”.

BAB 5

KESIMPULAN DAN SARAN

5.1 Kesimpulan

String Matching dan Regular Expression digunakan untuk menentukan keluaran dari aplikasi berbasis web berdasarkan masukan pengguna. Dalam aplikasi ini, pengguna dapat memilih algoritma yang digunakan yaitu BM dan KMP dengan menekan toggle yang berada di sidebar. Pengguna juga dapat memasukkan pernyataan dalam database sebagai jawaban baru. Rekomendasi pertanyaan akan dikeluarkan apabila masukan pengguna memiliki kasus kemiripan < 90% dari pernyataan yang terdapat di database. Pengguna dapat menekan tombol newchat untuk memasuki sesi baru dan dapat menekan tombol history untuk kembali ke sesi yang sebelumnya.

5.2 Saran

Pada pengerjaan tugas besar kali ini, terdapat kendala waktu karena diperlukan waktu untuk menghubungkan backend dan frontend sehingga memerlukan waktu yang lebih lama untuk menyelesaikan aplikasi. Agar waktu pengerjaan tugas cukup, berikut adalah hal yang perlu dilakukan:

1. Segera pelajari fungsi fungsi yang belum dipahami untuk menyelesaikan aplikasi.
2. Segera membagi tugas agar pekerjaan dapat selesai tepat waktu.
3. Tidak menunda pengerjaan tugas.

5.3 Komentar dan Refleksi

Pada tugas besar kali ini, sangat menuntut mahasiswa untuk memikirkan implementasi dari algoritma String Matching agar mengeluarkan hasil yang terbaik. Pada tugas ini juga diperlukan koordinasi antar anggota kelompok agar dapat menyelesaikan aplikasi dengan baik dan tepat waktu. Meskipun begitu, mungkin masih terdapat kekurangan dari aplikasi ini dan masih mungkin terjadi kesalahan.

DAFTAR PUSTAKA

- Referensi :
 - Munir, Rinaldi dan Maulidevi, Nur Ulfa. (2021). “Pencocokan String (String/Pattern Matching)”. Diakses online dari [PowerPoint Presentation \(itb.ac.id\)](#) pada 04 Mei 2023.
 - Khodra, Masayu Leylia (2021). “String Matching dengan Regular Expression”. Diakses online dari [String Matching dengan Regular Expression \(itb.ac.id\)](#) pada 04 Mei 2023.
- Link Repository :
 - Backend: https://github.com/egijago/Tubes3_13521056
 - Frontend: https://github.com/egijago/Tubes3_13521056_FE
- Link Deployment :
 - <https://tubes3-13521056-fe.vercel.app/>
- Link Youtube:
 - https://youtu.be/3Et_hr3oYcw