

**Laporan Tugas Besar 3 IF2211 Strategi Algoritma
Semester II Tahun 2022/2023**

**Penerapan String Matching dan Regular Expression dalam
Pembuatan ChatGPT Sederhana**



Disusun oleh:

Afnan Edsa Ramadhan 13521011

Laila Bilbina Khoiru Nisa 13521016

Syarifa Dwi Purnamasari 13521018

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

DAFTAR ISI

DAFTAR ISI	2
BAB I	4
DESKRIPSI TUGAS	4
BAB II	6
LANDASAN TEORI	6
2.1 Algoritma Knuth-Morris-Pratt	6
2.2. Algoritma Boyer-Moore	7
2.3. Regular Expression (Regex)	8
2.4. Penjelasan Aplikasi Web yang Dibangun	9
BAB III	10
ANALISIS PEMECAHAN MASALAH	10
3.1 Langkah Penyelesaian Masalah Tiap Fitur	10
3.1.1 Fitur Pertanyaan Teks	10
3.1.2 Fitur Kalkulator	10
3.1.4 Fitur Tambah Pertanyaan dan Jawaban	11
3.1.5 Fitur Hapus Pertanyaan	11
3.2 Fitur fungsional dan arsitektur aplikasi web yang dibangun	11
3.2.1 Fitur Fungsional	12
3.2.2 Arsitektur Aplikasi Web	12
BAB IV	13
IMPLEMENTASI DAN PENGUJIAN	13
4.1 Spesifikasi Teknis Program	13
4.1.1 add.go	13
4.1.2 bm.go	15
4.1.3 calculator.go	16
4.1.4 calender.go	23
4.1.5 dbQuery.go	26
4.1.6 fungsi.go	28
4.1.7 kmp.go	33
4.1.8 question.go	34
4.1.9 remove.go	35
4.2 Tata Cara Penggunaan Program	37
4.3 Hasil Pengujian dan Analisis	37
4.3.1 Fitur Pertanyaan Teks	37
4.3.2 Fitur Kalkulator	38
4.3.3 Fitur Tanggal	40
4.3.4 Fitur Tambah Pertanyaan dan Jawaban	42
4.3.5 Fitur Hapus Pertanyaan	42
BAB V	44

KESIMPULAN DAN SARAN	44
5.1 Kesimpulan	44
5.2 Saran	44
5.3 Komentar dan Refleksi	44
DAFTAR PUSTAKA	45
LAMPIRAN	46

BAB I

DESKRIPSI TUGAS

Dalam tugas besar ini, anda diminta untuk membangun sebuah aplikasi ChatGPT sederhana dengan mengaplikasikan pendekatan QA yang paling sederhana tersebut. Pencarian pertanyaan yang paling mirip dengan pertanyaan yang diberikan pengguna dilakukan dengan algoritma pencocokan string Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM). Regex digunakan untuk menentukan format dari pertanyaan (akan dijelaskan lebih lanjut pada bagian fitur aplikasi). Jika tidak ada satupun pertanyaan pada database yang exact match dengan pertanyaan pengguna melalui algoritma KMP ataupun BM, maka gunakan pertanyaan termirip dengan kesamaan setidaknya 90% Apabila tidak ada pertanyaan yang kemiripannya di atas 90%, maka chatbot akan memberikan maksimum 3 pilihan pertanyaan yang paling mirip untuk dipilih oleh pengguna. Perhitungan tingkat kemiripan dibebaskan kepada anda asalkan dijelaskan di laporan, namun disarankan menggunakan salah satu dari algoritma Hamming Distance, Levenshtein Distance, ataupun Longest Common Subsequence.

Fitur-Fitur Aplikasi:

ChatGPT sederhana yang anda membuat wajib dapat melakukan beberapa fitur / klasifikasi query seperti berikut:

1. Fitur pertanyaan teks (didapat dari database)

Mencocokkan pertanyaan dari input pengguna ke pertanyaan di database menggunakan algoritma KMP atau BM.

2. Fitur kalkulator

Pengguna memasukkan input query berupa persamaan matematika. Contohnya adalah $2*5$ atau $5+9*(2+4)$. Operasi cukup Tambah, kurang, kali, bagi, pangkat, kurung.

3. Fitur tanggal

Pengguna memasukkan input berupa tanggal, lalu chatbot akan merespon dengan hari apa di tanggal tersebut. Contohnya adalah 25/08/2023 maka chatbot akan menjawab dengan hari senin.

4. Tambah pertanyaan dan jawaban ke database

Pengguna dapat menambahkan pertanyaan dan jawabannya sendiri ke database dengan query contoh “Tambahkan pertanyaan xxx dengan jawaban yyy”. Menggunakan algoritma string matching untuk mencari tahu apakah pertanyaan sudah ada. Apabila sudah, maka jawaban akan diperbaharui.

5. Hapus pertanyaan dari database

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

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

BAB II

LANDASAN TEORI

2.1 Algoritma Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt (KMP) adalah algoritma pencocokan pola (pattern matching) dalam string yang dikembangkan oleh Donald Knuth, Vaughan Pratt, dan James Morris. Algoritma ini digunakan untuk mencari keberadaan sebuah string pola di dalam sebuah teks atau string utama.

Cara kerja algoritma KMP adalah dengan menghindari pencocokan ulang yang tidak perlu dari pola ke dalam teks. Hal ini dicapai dengan memanfaatkan informasi tentang pola yang telah dicocokkan sebelumnya dengan teks. Informasi ini dihasilkan dari pembangunan tabel preproses (preprocessing table) yang disebut dengan tabel π (pi).

Berikut adalah langkah-langkah dalam algoritma KMP:

1. Buat tabel π dengan menggunakan pola yang akan dicari. Tabel π menunjukkan panjang maksimum prefiks yang sama dan sufiks yang berakhir pada setiap indeks dalam pola. Proses pembuatan tabel π dapat dilakukan dengan cara berikut:
 - a. Setiap indeks pertama dalam tabel π diisi dengan 0.
 - b. Lakukan iterasi dari indeks 1 hingga panjang pola. Untuk setiap indeks i , cari panjang maksimum dari prefiks yang sama dengan sufiks dari potongan pola yang berakhir di indeks $i-1$. Simpan nilai ini di dalam tabel π pada indeks i .
2. Lakukan pencarian pola dalam teks dengan menggunakan tabel π . Proses pencarian dapat dilakukan dengan cara berikut:
 - a. Setel indeks $i = 0$ dan $j = 0$.
 - b. Lakukan iterasi selama $i < \text{panjang teks}$:
 - i. Jika karakter pada indeks i dalam teks sama dengan karakter pada indeks j dalam pola, maka increment i dan j .
 - ii. Jika $j = \text{panjang pola}$, maka pola telah ditemukan dalam teks pada indeks $i-j$. Setel $j = \pi[j-1]$ dan lanjutkan pencarian.
 - iii. Jika karakter pada indeks i dalam teks tidak sama dengan karakter pada indeks j dalam pola, maka setel $j = \pi[j-1]$. Jika $j != 0$, kembali ke langkah ii. Jika $j = 0$, increment i dan lanjutkan pencarian.

Dengan menggunakan algoritma KMP, pencarian pola pada teks dapat dilakukan dengan waktu $O(n + m)$, di mana n adalah panjang teks dan m adalah panjang pola. Ini lebih efisien daripada algoritma pencocokan pola sederhana yang menggunakan nested loop dan memerlukan waktu $O(n*m)$ dalam kasus terburuk.

2.2. Algoritma Boyer-Moore

Algoritma Boyer-Moore (BM) adalah algoritma pencocokan pola dalam string yang dikembangkan oleh Robert Boyer dan J Strother Moore pada tahun 1977. Algoritma ini digunakan untuk mencari keberadaan sebuah string pola dalam sebuah teks atau string utama.

Cara kerja algoritma BM adalah dengan mencocokkan pola dari belakang ke depan. Algoritma ini juga menggunakan dua heuristik: heuristik karakter dan heuristik penjajaran. Heuristik karakter memanfaatkan fakta bahwa jika ada karakter yang tidak cocok pada posisi tertentu, maka teks dapat digeser sejauh beberapa karakter ke kanan tanpa melewati kemungkinan keberadaan pola. Heuristik penjajaran memanfaatkan fakta bahwa jika ada beberapa karakter cocok, namun ada beberapa karakter yang tidak cocok, maka pola dapat digeser sejauh beberapa karakter ke kiri.

Berikut adalah langkah-langkah dalam algoritma BM:

1. Buat tabel karakter yang berisi indeks terakhir dari setiap karakter dalam pola. Jika sebuah karakter tidak ada dalam pola, maka indeksnya diatur menjadi -1.
2. Lakukan pencarian pola dalam teks dengan menggunakan heuristik karakter dan heuristik penjajaran. Proses pencarian dapat dilakukan dengan cara berikut:
 - a. Setel indeks $i = \text{panjang pola} - 1$ dan $j = \text{panjang pola} - 1$.
 - b. Lakukan iterasi selama $i < \text{panjang teks}$:
 - i. Jika karakter pada indeks i dalam teks sama dengan karakter pada indeks j dalam pola, maka decrement i dan j .
 - ii. Jika $j = -1$, maka pola telah ditemukan dalam teks pada indeks $i+1$. Geser teks sejauh panjang pola ke kanan dan lanjutkan pencarian.
 - iii. Jika karakter pada indeks i dalam teks tidak sama dengan karakter pada indeks j dalam pola, maka geser teks sejauh maksimum antara tabel karakter pada indeks tersebut dan heuristik penjajaran. Setel $j = \text{panjang pola} - 1$ dan kembali ke langkah i.

Dengan menggunakan algoritma BM, pencarian pola pada teks dapat dilakukan dengan waktu $O(n/m)$, di mana n adalah panjang teks dan m adalah panjang pola. Dalam kasus terburuk, algoritma ini dapat memerlukan waktu $O(n*m)$, tetapi hal ini sangat jarang terjadi dalam praktiknya. Algoritma BM merupakan salah satu algoritma pencocokan pola tercepat yang tersedia saat ini.

2.3. Regular Expression (Regex)

Regular Expression (Regex) adalah sebuah bahasa pemrograman yang digunakan untuk mencari, mencocokkan, dan memanipulasi teks berdasarkan pola tertentu. Regex digunakan untuk mencari pola-pola yang berulang dalam sebuah teks, seperti nomor telepon, email, atau kata kunci tertentu.

Regex terdiri dari karakter-karakter tertentu yang memiliki arti khusus. Karakter-karakter ini dapat digunakan untuk mencocokkan pola tertentu dalam sebuah string. Beberapa karakter khusus dalam regex di antaranya:

1. Karakter titik (.) : Mencocokkan karakter apa pun kecuali karakter baris baru.
2. Karakter tanda tanya (?) : Mencocokkan karakter sebelumnya nol atau satu kali.
3. Karakter asterisk (*) : Mencocokkan karakter sebelumnya nol atau lebih kali.
4. Karakter plus (+) : Mencocokkan karakter sebelumnya satu atau lebih kali.
5. Karakter tanda kurung buka dan tutup () : Mencocokkan sebuah grup karakter.
6. Karakter kurung kurawal {} : Mengatur jumlah karakter yang dicocokkan.
7. Karakter kurung siku [] : Mencocokkan sebuah karakter dalam set karakter tertentu.

Selain karakter khusus, regex juga memiliki beberapa fungsi atau metode yang dapat digunakan untuk mencocokkan pola pada sebuah string, seperti:

1. search() : mencari pola tertentu dalam sebuah string, dan mengembalikan objek Match jika pola ditemukan.
2. findall() : mencari semua kejadian pola tertentu dalam sebuah string, dan mengembalikan daftar berisi semua kejadian pola.
3. sub() : mengganti pola tertentu dalam sebuah string dengan pola baru.
4. split() : membagi sebuah string menjadi beberapa substring berdasarkan pola tertentu.

2.4. Penjelasan Aplikasi Web yang Dibangun

Program yang kami buat dalam Tugas Besar ini berstruktur *microservices*. Program berstruktur *microservices* adalah sebuah pendekatan dalam pengembangan perangkat lunak yang terdiri dari beberapa layanan kecil yang dapat berjalan secara mandiri dan saling berkomunikasi untuk membangun aplikasi yang lebih besar dan kompleks. Setiap layanan kecil memiliki fungsi spesifik dan berjalan dalam kontainer atau lingkungan terisolasi.

Dalam pendekatan *microservices*, aplikasi dibagi menjadi modul atau layanan kecil yang dapat dikembangkan dan dikelola secara terpisah, memungkinkan pengembang untuk memperbarui, memperbaiki, atau mengganti salah satu layanan tanpa memengaruhi seluruh aplikasi. Ini memudahkan pengembangan dan pemeliharaan aplikasi yang lebih besar dan kompleks. Dalam kasus ini, pembagian layanan program dibagi menjadi dua bagian yaitu backend dan frontend sesuai tugasnya masing-masing.

Bagian frontend merupakan bagian dari program yang diakses oleh pengguna akhir dan digunakan untuk berinteraksi dengan aplikasi tersebut. Isi frontend mencakup semua elemen visual dan interaktif yang ditampilkan di browser, seperti halaman web, button, text box, dan sebagainya. Frontend yang kami bangun menggunakan framework Next.js yang berbasis React dan JavaScript.

Bagian backend merupakan bagian dari sebuah aplikasi yang tidak dapat diakses oleh pengguna akhir, tetapi digunakan oleh pengembang untuk membangun dan mengelola fitur-fitur aplikasi. Backend mengendalikan fungsionalitas yang mendasari dan biasanya berada di belakang tampilan frontend. Isi backend mencakup segala hal yang terjadi di sisi server, seperti pengaturan server, pemrosesan permintaan dari klien, pengelolaan basis data, dan banyak lagi. Backend yang kami bangun menggunakan bahasa GoLang atau Go Language. Dalam backend terdapat algoritma yang digunakan untuk menyelesaikan permasalahan seperti Knuth-Morris-Pratt (KMP), Boyer-Moore (BM), Regex, dan Levenshtein.

BAB III

ANALISIS PEMECAHAN MASALAH

3.1 Langkah Penyelesaian Masalah Tiap Fitur

3.1.1 Fitur Pertanyaan Teks

Pada program ini, text string pertanyaan akan masuk ke dalam fitur ini ketika text tidak memenuhi syarat untuk masuk fitur tanggal, kalkulator, tambah pertanyaan, dan hapus pertanyaan. Hal pertama yang dilakukan adalah memproses text string pertanyaan menggunakan regex untuk menemukan prefix kata tanya pada text, seperti apakah, siapakah, dan sebagainya. Setelah mendapatkan pertanyaan yang sudah dihapus kata tanyanya, text akan dicocokkan dengan database lokal yang ada menggunakan algoritma Knuth-Morris-Pratt (KMP) atau Boyer-Moore (BM) sesuai dengan apa yang dipilih melalui toggle yang ada pada frontend. Jika text tepat sama dengan data pertanyaan yang disimpan dalam database, maka akan langsung mengeluarkan jawaban yang sesuai dengan id_data pertanyaan. Jika belum sesuai, text akan diproses menggunakan Levenshtein dan mengeluarkan jawaban yang presentase kemiripan pertanyaannya paling besar dengan pertanyaan yang ada dalam database.

3.1.2 Fitur Kalkulator

Hal pertama yang dilakukan sebelum masuk fitur ini adalah memproses text string pertanyaan menggunakan regex untuk menemukan prefix kata tanya pada text, seperti berapa dan sebagainya. Pada program ini, text string pertanyaan akan masuk ke dalam fitur ini ketika text tidak memenuhi syarat untuk masuk fitur tanggal dan panjang array string hasil FindAllString regex pattern kalkulator sama dengan satu. Jika panjang array string sama dengan lebih dari satu, program akan mengeluarkan jawaban “Sintaks tidak sesuai”. Setelah mendapatkan pertanyaan yang sudah dihapus kata tanyanya, maka text akan masuk ke dalam method calculator untuk mencari hasil dari ekspresi matematika yang didapatkan. Hasil dari ekspresi matematika yang didapatkan akan dikeluarkan melalui bubble chat yang ada pada halaman web.

3.1.3 Fitur Tanggal

Hal pertama yang dilakukan sebelum masuk fitur ini adalah memproses text string pertanyaan menggunakan regex untuk menemukan prefix kata tanya pada text, seperti hari, tanggal, dan sebagainya. Pada program ini, text string pertanyaan akan masuk ke dalam fitur ini ketika text memenuhi panjang array string hasil FindAllString regex pattern tanggal tidak sama dengan nol. Setelah mendapatkan pertanyaan yang sudah dihapus kata tanyanya, maka text akan masuk ke dalam method isDateValid yang mengeluarkan boolean. Jika mengeluarkan boolean true, fitur akan mengeluarkan hari apa yang sesuai dengan tanggal tersebut, sedangkan jika sebaliknya, maka string “Invalid Date” akan dikeluarkan melalui bubble chat yang ada pada halaman web.

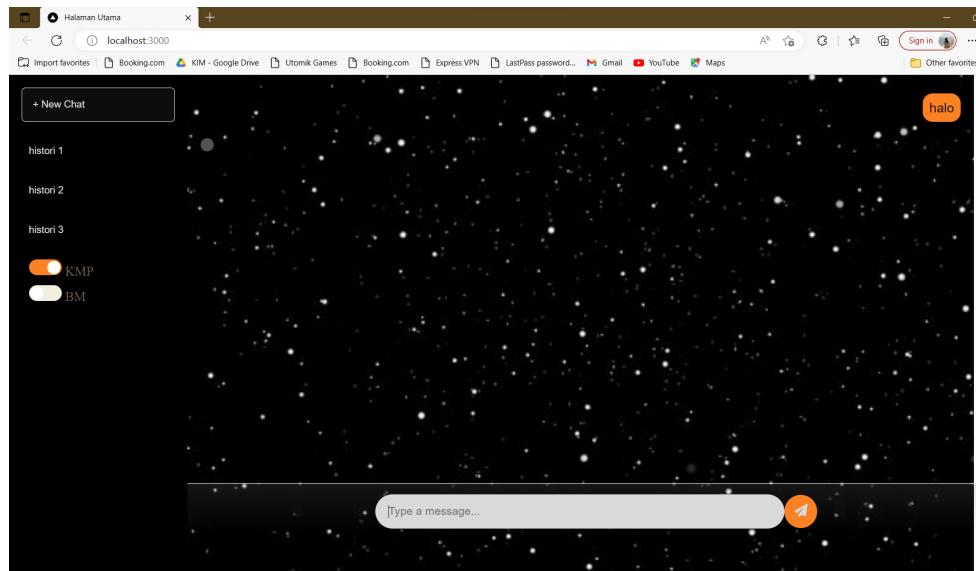
3.1.4 Fitur Tambah Pertanyaan dan Jawaban

Pada program ini, text string pertanyaan akan masuk ke dalam fitur ini ketika text tidak memenuhi syarat untuk masuk fitur tanggal dan kalkulator serta hasil ValidateAddDatabase tidak sama dengan "notFound". Pada fitur ini, jika sintaks pertanyaan yang dimasukkan sesuai dengan yang ada pada pattern yaitu "[T|t]ambah pertanyaan X dengan jawaban Y" dengan X merupakan pertanyaan yang ingin ditambahkan dan Y merupakan jawaban yang akan dimasukkan dalam database, maka bubble chat akan mengeluarkan jawaban "Pertanyaan X dan Jawaban Y berhasil ditambahkan" dan X serta Y akan dimasukkan dalam database. Jika X sudah ada dalam database, maka hanya Y yang akan di update dalam database.

3.1.5 Fitur Hapus Pertanyaan

Pada program ini, text string pertanyaan akan masuk ke dalam fitur ini ketika text tidak memenuhi syarat untuk masuk fitur tanggal kalkulator, dan tambah pertanyaan serta hasil ValidateRemoveDatabase tidak sama dengan "notFound". Pada fitur ini, jika sintaks pertanyaan yang dimasukkan sesuai dengan yang ada pada pattern yaitu "[H|h]apus pertanyaan X" dengan X merupakan pertanyaan yang ingin dihapus dari database, maka bubble chat akan mengeluarkan jawaban "Pertanyaan X berhasil dihapus" dan X akan dihapus dari database.

3.2 Fitur fungsional dan arsitektur aplikasi web yang dibangun



Gambar 3.1 Screenshot Aplikasi

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 IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Spesifikasi Teknis Program

4.1.1 add.go

```
package lib

import (
    "fmt"
    "regexp"
)

func ValidateAddDatabase(text string) string {
    knowledge_base := map[string]string{
        "[T|t]ambah pertanyaan (.*) dengan jawaban (.*)": "Pertanyaan X dan Jawaban Y berhasil ditambahkan",
    }
    notFound := "notFound"
    for key, value := range knowledge_base {
        m := regexp.MustCompile(key)
        if m.MatchString(text) {
            answer := value
            len_groups := len(m.FindString(text))
            if len_groups == 0 {
                fmt.Println("adfad")
                fmt.Println(answer)
                return answer
            } else {
                x := m.FindStringSubmatch(text)[1]
                fmt.Println(x)
                answer = regexp.MustCompile("X").ReplaceAllString(answer, x)
                y := m.FindStringSubmatch(text)[2]
                fmt.Println(y)
                answer = regexp.MustCompile("Y").ReplaceAllString(answer, y)
            }
        }
    }
}
```

```
    }
    return notFound
}

func AddDatabase(text string, listPertanyaan []string) string {
    knowledge_base := map[string]string{
        "[T|t]ambah pertanyaan (.*) dengan jawaban (.*)": "Pertanyaan X dan Jawaban Y berhasil
ditambahkan",
    }
    notFound := "notFound"
    for key, value := range knowledge_base {
        m := regexp.MustCompile(key)
        if m.MatchString(text) {
            answer := value
            len_groups := len(m.FindString(text))
            if len_groups == 0 {
                return answer
            } else {
                x := m.FindStringSubmatch(text)[1]
                fmt.Println(x)
                answer = regexp.MustCompile("X").ReplaceAllString(answer, x)
                y := m.FindStringSubmatch(text)[2]
                fmt.Println(y)
                answer = regexp.MustCompile("Y").ReplaceAllString(answer, y)

                return AddToDatabase(x, y, listPertanyaan)
            }
        }
    }
    return notFound
}
```

4.1.2 bm.go

```
package lib

// BM the pattern in the text
func BM(text, pattern string) int {
    charTable := [256]int{}
    // initialise default values for bad char table
    for i := range charTable {
        charTable[i] = len(pattern)
    }
    // creating the bad char table based on the pattern
    // using the formula len(pattern) - i - 1
    for i := 0; i < len(pattern)-1; i++ {
        charTable[pattern[i]] = len(pattern) - i - 1
    }
    // i is the index for text
    i := len(pattern) - 1

    // scan all the text
    for i < len(text) {
        j := len(pattern) - 1
        for j >= 0 && pattern[j] == text[i] {
            j--
            i--
        }
        if j < 0 {
            return i + 1
        }
        // shift the pattern based on badchar table
        i = i + charTable[text[i]]
    }

    // nothing has been found
    return -1
}

// func main() {
// fmt.Println("crazy brown fx", "own", BM("crazy brown fx", "own"))
```

```
// fmt.Println("crazy brown fox", "fox", BM("crazy brown fox", "fox"))
// fmt.Println("crazy brown fx", "fox", BM("crazy brown fx", "fox"))
// fmt.Println("crazy brown fox", "fx", BM("crazy brown fox", "fx"))
//}
```

4.1.3 calculator.go

```
package lib

import (
    "fmt"
    "regexp"
    "strconv"
)

func Contains(arr []string, str string) bool {
    for _, a := range arr {
        if a == str {
            return true
        }
    }
    return false
}

func IndexOf(arr []string, str string) int {
    for i, a := range arr {
        if a == str {
            return i
        }
    }
    return -1
}

func LastIndexOf(arr []string, val string) int {
    for i := len(arr) - 1; i >= 0; i-- {
        if arr[i] == val {
            return i
        }
    }
}
```

```
        }
    }
    return -1
}

func ConverArrStrToFloat(arr []string) []float64 {
    var arrFloat []float64
    var temp float64
    for i := 0; i < len(arr); i++ {
        temp, _ = strconv.ParseFloat(arr[i], 64)
        arrFloat = append(arrFloat, temp)
    }
    return arrFloat
}

func bracketValidation(operand []string) bool {
    var open, close int
    for i := 0; i < len(operand); i++ {
        if operand[i] == "(" {
            open++
        } else if operand[i] == ")" {
            close++
        }
    }
    if open >= close {
        return true
    } else {
        return false
    }
}

func addCloseBracket(operand []string) []string {
    var open, close int
    for i := 0; i < len(operand); i++ {
        if operand[i] == "(" {
            open++
        } else if operand[i] == ")" {
            close++
        }
    }
}
```

```
    }

    for i := 0; i < open-close; i++ {
        operand = append(operand, ")")
    }

    return operand
}

func findIndexCloseBracket(operand []string, IndexOpenBracket int) int {
    var open, close int
    for i := IndexOpenBracket; i < len(operand); i++ {
        if operand[i] == "(" {
            open++
        } else if operand[i] == ")" {
            close++
        }
        if open == close {
            return i
        }
    }
    return -1
}

func CountBracketBefore(operand []string, IndexOpenBracket int) int {
    var open, close int
    for i := 0; i < IndexOpenBracket; i++ {
        if operand[i] == "(" {
            open++
        } else if operand[i] == ")" {
            close++
        }
    }
    return open + close
}

func FindResult(operand []string, angka []float64) float64 {
    var result float64
    var idx int

    if bracketValidation(operand) == false {

```

```

        return -999999
    } else {
        operand = addCloseBracket(operand)

        // handle brackets recursively
        for Contains(operand, "(") {
            idxopen := LastIndexOf(operand, "(")
            idxclose := findIndexCloseBracket(operand, idxopen)
            fmt.Printf("idxopen: %v, idxclose: %v\n", idxopen, idxclose)

            operandIn := operand[idxopen+1 : idxclose]
            idxangka := idxopen - CountBracketBefore(operand, idxopen)
            angkaIn := angka[idxangka : idxangka+(idxclose-idxopen)]
            fmt.Printf("operandIn: %v, angkaIn: %v\n", operandIn,
            angkaIn)

            result := FindResult(operandIn, angkaIn)
            fmt.Printf("subResult: %v\n", result)
            operand = append(operand[:idxopen], operand[idxclose+1:]...)
            angka = append(angka[:idxangka+1],
            angka[idxangka+(idxclose-idxopen):]...)
            fmt.Printf("operand: %v, angka: %v, subResult: %v\n",
            operand, angka, result)
            angka[idxangka] = result
        }

        // handle remaining operations
        for len(operand) != 0 {
            idxp := IndexOf(operand, "+")
            idxs := IndexOf(operand, "-")
            idxm := IndexOf(operand, "*")
            idxd := IndexOf(operand, "/")

            if idxp < idxs && idxp > -1 {
                if Contains(operand, "-") {
                    idx = IndexOf(operand, "-")
                    result = angka[idx] - angka[idx+1]
                }
                if Contains(operand, "+") {

```

```

        idx = IndexOf(operand, "+")
        result = angka[idx] + angka[idx+1]
    }
} else {
    if Contains(operand, "+") {
        idx = IndexOf(operand, "+")
        result = angka[idx] + angka[idx+1]
    }
    if Contains(operand, "-") {
        idx = IndexOf(operand, "-")
        result = angka[idx] - angka[idx+1]
    }
}

if idxm < idxd && idxm > -1 {
    if Contains(operand, "/") {
        idx = IndexOf(operand, "/")
        result = angka[idx] / angka[idx+1]
    }
    if Contains(operand, "*") {
        idx = IndexOf(operand, "*")
        result = angka[idx] * angka[idx+1]
    }
} else {
    if Contains(operand, "*") {
        idx = IndexOf(operand, "*")
        result = angka[idx] * angka[idx+1]
    }
    if Contains(operand, "/") {
        idx = IndexOf(operand, "/")
        result = angka[idx] / angka[idx+1]
    }
}
}

operand = append(operand[:idx], operand[idx+1:]...)
angka = append(angka[:idx+1], angka[idx+2:]...)
angka[idx] = result
// fmt.Printf("operand: %v, angka: %v, subResult: %v\n",

```

```
operand, angka, result)
    }
    return angka[0]
}

func Calculator(text string) string{
    operand := []string{}
    angka := []string{}
    var temp string
    count := 0
    operation := []string{"+", "-", "*", "/", "(", ")"}

    for i := 0; i < len(text); i++ {
        if Contains(operation, string(text[i])) {
            operand = append(operand, string(text[i]))
        } else {
            if text[i] != ' ' {
                if i != len(text)-1 {
                    if count == 0 {
                        temp = string(text[i])
                        if Contains(operation, string(text[i+1])) ||
text[i+1] == ' ' {
                            angka = append(angka, (temp))
                            temp = ""
                            count = 0
                        }
                        count++
                    } else {
                        temp = temp + string(text[i])
                        if Contains(operation, string(text[i+1])) ||
text[i+1] == ' ' {
                            angka = append(angka, (temp))
                            temp = ""
                            count = 0
                        }
                    }
                } else {
                    temp = temp + string(text[i])
                }
            }
        }
    }
}
```

```

        angka = append(angka, (temp))
        temp = ""
        count = 0
    }
}
}
}

fmt.Println(operand)
fmt.Println(angka)
var angkaFloat []float64
angkaFloat = ConverArrStrToFloat(angka)
// fmt.Println(angkaFloat)

var result = FindResult(operand, angkaFloat)
if result == -999999{
    return "Sintaks tidak sesuai"
}
fmt.Println("Hasilnya adalah", result)
return ("Hasilnya adalah " + fmt.Sprintf("%.2f", result))
}

func FindPrefixCalculator(text string) string {
knowledge_base := map[string]string{
    "Berapa hasil (.*)":      "X",
    "(.*) hasilnya adalah": "X",
    "(.*) berapa hasilnya": "X",
    "Hasil dari (.*)":       "X",
    "(.*) ?":                "X",
}
notFound := "notFound"
for key, value := range knowledge_base {
    m := regexp.MustCompile(key)
    if m.MatchString(text) {
        answer := value
        len_groups := len(m.FindString(text))
        if len_groups == 0 {
            return answer
        } else {
            x := m.FindStringSubmatch(text)[1]

```

```
        answer =
regexp.MustCompile("X").ReplaceAllString(answer, x)
    return answer
}
}
}
return notFound
}
```

4.1.4 calendar.go

```
package lib

import (
    "fmt"
    "strconv"
    "strings"
    "regexp"
)

func IsDateValid(text string) bool {

    kalender := strings.Split(text, "/")

    tanggal, err := strconv.Atoi(kalender[0])
    bulan, err := strconv.Atoi(kalender[1])
    tahun, err := strconv.Atoi(kalender[2])
    if err != nil {
        fmt.Println(err.Error())
    }

    //Cek batas tanggalan
    if tanggal < 1 || tanggal > 31 {
        return false
    }
    if bulan < 1 || bulan > 12 {
        return false
    }
}
```

```
if tahun < 1 {
    return false
}

//Cek bulan 30 hari
if bulan == 4 || bulan == 6 || bulan == 9 || bulan == 11 {
    if tanggal > 30 {
        return false
    }
}

//Cek bulan februari
if bulan == 2 {
    if IsKabisat(tahun) {
        if tanggal > 29 {
            return false
        }
    } else {
        if tanggal > 28 {
            return false
        }
    }
}

return true
}

func GetDay(text string) string {
    kalender := strings.Split(text, "/")
    if len(kalender[0]) == 1 {
        kalender[0] = "0" + kalender[0]
    }
    if len(kalender[1]) == 1 {
        kalender[1] = "0" + kalender[1]
    }
    if len(kalender[2]) != 4 {
        for len(kalender[2]) != 4 {
            kalender[2] = "0" + kalender[2]
        }
    }
}
```

```

year, _ := strconv.Atoi(kalender[2])
month, _ := strconv.Atoi(kalender[1])
day, _ := strconv.Atoi(kalender[0])
a := (14 - month) / 12
y := year - a
m := month + 12*a - 2
weekday := (day + y + y/4 - y/100 + y/400 + (31*m)/12) % 7
dayName := []string{"Minggu", "Senin", "Selasa", "Rabu", "Kamis",
"Jumat", "Sabtu"} [weekday]
return dayName

}

func IsKabisat(tahun int) bool {
    if tahun%4 == 0 {
        if tahun%100 == 0 {
            if tahun%400 == 0 {
                return true
            }
            return false
        }
        return true
    }
    return false
}

func FindPrefixCalendar(text string) string {
    knowledge_base := map[string]string{
        "[H|h]ari apa (.*)": "X",
        "[H|h]ari apa tanggal (.*)": "X",
        "(.*) hari apa": "X",
        "[T!t]anggal (.*) hari apa": "X",
        "[H|h]ari (.*)": "X",
    }
    notFound := "notFound"
    for key, value := range knowledge_base {
        m := regexp.MustCompile(key)
        if m.MatchString(text) {
            answer := value

```

```

len_groups := len(m.FindString(text))
if len_groups == 0 {
    return answer
} else {
    x := m.FindStringSubmatch(text)[1]
    answer =
regexp.MustCompile("X").ReplaceAllString(answer, x)
    return answer
}
}
return notFound
}

```

4.1.5 dbQuery.go

```

package lib

import (
    "backend/models"
    "database/sql"
    "fmt"
)

var db *sql.DB

func GetAllData() []models.Data {
    var data []models.Data

    rows, _ := db.Query("SELECT * FROM data")

    defer rows.Close()
    for rows.Next() {
        var dataa models.Data
        rows.Scan(&dataa.Id_data, &dataa.Pertanyaan, &dataa.Jawaban)
        data = append(data, dataa)
    }
}

```

```
        return data
    }

func GetPertanyaan(data []models.Data) []string {
    var pertanyaan []string
    for i := 0; i < len(data); i++ {
        pertanyaan = append(pertanyaan, data[i].Pertanyaan)
    }
    return pertanyaan
}

func AddToDatabase(pertanyaan string, jawaban string, listPertanyaan
[]string) string {
    if ValidatePertanyaan(pertanyaan, listPertanyaan) == -1 {
        rows, err := db.Query("UPDATE data SET jawaban = ? WHERE
pertanyaan = ?", jawaban, pertanyaan)
        if err != nil {
            return "Gagal mengupdate jawaban"
        }
        defer rows.Close()
        return "Pertanyaan " + pertanyaan + " sudah ada!. jawaban di
update ke " + jawaban
    } else {
        rows, err := db.Query("INSERT INTO data (pertanyaan, jawaban)
VALUES (?, ?)", pertanyaan, jawaban)
        if err != nil {
            return "Gagal menambahkan pertanyaan dan jawaban"
        }
        defer rows.Close()
        return "Berhasil menambahkan pertanyaan dan jawaban"
    }
}

func RemoveFromDatabase(pertanyaan string, listPertanyaan []string)
string {
    fmt.Println(IsPertanyaanExist(pertanyaan, listPertanyaan))
    if IsPertanyaanExist(pertanyaan, listPertanyaan) == -1 {
        return "Tidak ada pertanyaan "+pertanyaan+" pada database."
    } else {
        rows, err := db.Query("DELETE FROM data WHERE pertanyaan = ?",

```

```

pertanyaan)
    if err != nil {
        return "Gagal menghapus pertanyaan"
    }
    defer rows.Close()
    return "Pertanyaan "+pertanyaan+" berhasil dihapus"
}

func ValidatePertanyaan(pertanyaan string, listPertanyaan []string) int
{
    for i := 0; i < len(listPertanyaan); i++ {
        if pertanyaan == listPertanyaan[i] {
            return -1
        }
    }
    return 0
}

func IsPertanyaanExist(pertanyaan string, listPertanyaan []string) int{
    for i := 0; i < len(listPertanyaan); i++ {
        if pertanyaan == listPertanyaan[i] {
            fmt.Println(i)
            return i
        }
    }
    return -1
}

```

4.1.6 fungsi.go

```

package lib

import (
    // "backend/controller"
    "fmt"
    "log"

```

```
"math"
"os"
"regexp"

"database/sql"

_ "github.com/go-sql-driver/mysql"
"github.com/joho/godotenv"
"github.com/texttheater/golang-levenshtein/levenshtein"
)

func SearchHighestPercentage(source string, listPertanyaan []string)
(float64, int) {
    var highest float64
    var index int
    for i := 0; i < len(listPertanyaan); i++ {
        distance := levenshtein.DistanceForStrings([]rune(source),
[]rune(listPertanyaan[i]), levenshtein.DefaultOptions)
        maxx := math.Max(float64(len(source)),
float64(len(listPertanyaan[i])))
        percentage := 100 - (float64(distance) / maxx * 100)
        if percentage > highest {
            highest = percentage
            index = i
        }
    }
    return highest, index
}

func SearchSimilarQuestion(source string, listPertanyaan []string) []int
{
    var index []int
    for i := 0; i < len(listPertanyaan); i++ {
        distance := levenshtein.DistanceForStrings([]rune(source),
[]rune(listPertanyaan[i]), levenshtein.DefaultOptions)
        maxx := math.Max(float64(len(source)),
float64(len(listPertanyaan[i])))
        percentage := 100 - (float64(distance) / maxx * 100)
        if percentage > 40 {
```

```
        index = append(index, i)
    }
}
return index
}

func getEnv(key string) string {
    err := godotenv.Load("models/.env")
    if err != nil {
        log.Fatalf("Error loading .env file")
    }
    return os.Getenv(key)
}

func findKMP(text string, listPertanyaan []string) int {
    var index int
    for i := 0; i < len(listPertanyaan); i++ {
        index = KmpMatch(text, listPertanyaan[i])
        if index != -1 {
            return i
        }
    }
    return index
}

func findBM(text string, listPertanyaan []string) int {
    var index int
    for i := 0; i < len(listPertanyaan); i++ {
        index = BM(text, listPertanyaan[i])
        if index != -1 {
            return i
        }
    }
    return index
}

func Utama(text string, val bool) string {
    text = text[1:len(text)]
    fmt.Println(text)
    var err error
```

```
var regexCalcu *regexp.Regexp
var regexCalen *regexp.Regexp

var textCalen = FindPrefixCalendar(text)
var textCalcu = FindPrefixCalculator(text)
regexCalcu, err =
regexp.MustCompile(`[\()?-+]?\\d*\\.?\\d+[\\)]?\\s*([-+*/] (\\s?) [\(\)?\\s*\\d*\\.?\\d+
[\\)]?\\s*)`)
regexCalen, err = regexp.MustCompile(`[0-9]{1,2}/[0-9]{1,2}/[0-9]{1,4}`)
if err != nil {
    fmt.Println(err.Error())
}

db, err = sql.Open("mysql",
getEnv("DBUSER") + ":" + getEnv("DBPASS") + "@tcp(localhost:" + getEnv("DBPORT")
+ ")/" + getEnv("DBNAME"))
if err != nil {
    panic(err.Error())
}

pingErr := db.Ping()
if pingErr != nil {
    log.Fatal(pingErr)
}
fmt.Println("Connected!")

rows := GetAllData()
pertanyaan := GetPertanyaan(rows)
var hasilCalcu = regexCalcu.FindAllString(textCalcu, -1)
var hasilCalen = regexCalen.FindAllString(textCalen, -1)
if len(hasilCalen) != 0 {
    fmt.Println("ini kalender")
    if IsDateValid(hasilCalen[0]) {
        return ("Hari " + GetDay(hasilCalen[0]))
    } else {
        return ("Invalid Date")
    }
} else if len(hasilCalcu) == 1 {
    fmt.Println("ini kalkulator")
```

```

        fmt.Println(hasilCalcu[0])
        fmt.Println(Calculator(hasilCalcu[0]))
        return Calculator(hasilCalcu[0])
    } else if ValidateAddDatabase(text) != "notFound" {
        fmt.Println(ValidateAddDatabase(text))
        return AddDatabase(text, pertanyaan)
    } else if ValidateRemoveDatabase(text) != "notFound" {
        fmt.Println(ValidateRemoveDatabase(text))
        return RemoveDatabase(text, pertanyaan)
    } else {
        fmt.Println("ini pertanyaan")
        text = FindPrefixQ(text)
        var retVal int
        if !val {
            retVal = findKMP(text, pertanyaan)
            fmt.Println("Masuk KMP")
        } else {
            retVal = findBM(text, pertanyaan)
            fmt.Println("Masuk BM")
        }
        fmt.Println(retVal)
        if retVal == -1 {
            fmt.Println("Masuk Levenshtein")
            percentage, index := SearchHighestPercentage(text,
pertanyaan)
            fmt.Println(percentage)
            if percentage < 41 {
                return ("Maaf, saya tidak mengerti")
            } else if percentage >= 41 && percentage < 80 {
                mirip := SearchSimilarQuestion(text, pertanyaan)
                retMirip := "Apakah maksud anda "
                for i := 0; i < len(mirip); i++ {
                    if i == len(mirip)-1 {
                        retMirip += rows[mirip[i]].Pertanyaan + " ?"
                    } else {
                        retMirip += rows[mirip[i]].Pertanyaan + ", "
                    }
                }
            }
        }
    }
}

```

```

        fmt.Println(retMirip)
        return retMirip
    } else {
        fmt.Println(percentage, index)
        fmt.Println(rows[index].Jawaban)
        return (rows[index].Jawaban)
    }
} else {
    fmt.Println("langsung")
    fmt.Println(rows[retVal].Jawaban)
    return (rows[retVal].Jawaban)
}
}
}
}

```

4.1.7 kmp.go

```

package lib

func KmpMatch(text string, pat string) int {
    border := ComputeBorder(pat)
    i := 0
    j := 0
    m := len(pat)
    n := len(text)

    for i < n {
        if text[i] == pat[j] {
            if j == m-1 {
                return i - m + 1
            }
            i++
            j++
        } else if j > 0 {
            j = border[j-1]
        } else {
            i++
        }
    }
}

```

```

    }

    return -1
}

func ComputeBorder(pat string) []int {
    var border []int = make([]int, len(pat))
    border[0] = 0
    var j int = 0
    var i int = 1
    for i < len(pat) {
        if pat[j] == pat[i] {
            border[i] = j + 1
            j++
            i++
        } else if j > 0 {
            j = border[j-1]
        } else {
            border[i] = 0
            i++
        }
    }
    return border
}

```

4.1.8 question.go

```

package lib

import "regexp"

func FindPrefixQ(text string) string {
    knowledge_base := map[string]string{
        "[A!a]akah (.*)":      "X",
        "[A|a]pa (.*)":       "X",
        "[S|s]iapakah (.*)":  "X",
        "[S|s]iaha (.*)":     "X",
        "(.*) ?":             "X",
    }
}

```

```

notFound := "notFound"
for key, value := range knowledge_base {
    m := regexp.MustCompile(key)
    if m.MatchString(text) {
        answer := value
        len_groups := len(m.FindString(text))
        if len_groups == 0 {
            return answer
        } else {
            x := m.FindStringSubmatch(text)[1]
            answer =
        }
    }
    regexp.MustCompile("X").ReplaceAllString(answer, x)
}
return answer
}
}
return notFound
}

```

4.1.9 remove.go

```

package lib

import (
    "regexp"
)

func ValidateRemoveDatabase(text string) string {
    knowledge_base := map[string]string{
        "[H|h]apus pertanyaan (.*)": "Pertanyaan X berhasil dihapus",
    }
    notFound := "notFound"
    for key, value := range knowledge_base {
        m := regexp.MustCompile(key)
        if m.MatchString(text) {
            answer := value
            len_groups := len(m.FindString(text))
        }
    }
    return answer
}

```

```
        if len_groups == 0 {
            return answer
        } else {
            x := m.FindStringSubmatch(text)[1]
            answer =
        regexp.MustCompile("X").ReplaceAllString(answer, x)
            return answer
        }
    }
    return notFound
}

func RemoveDatabase(text string, pertanyaan []string) string {
    knowledge_base := map[string]string{
        "[H|h]apus pertanyaan (.*)": "Pertanyaan X berhasil dihapus",
    }
    notFound := "notFound"
    for key, value := range knowledge_base {
        m := regexp.MustCompile(key)
        if m.MatchString(text) {
            answer := value
            len_groups := len(m.FindString(text))
            if len_groups == 0 {
                return answer
            } else {
                x := m.FindStringSubmatch(text)[1]
                answer =
            regexp.MustCompile("X").ReplaceAllString(answer, x)
                return RemoveFromDatabase(x, pertanyaan)
            }
        }
    }
    return notFound
}
```

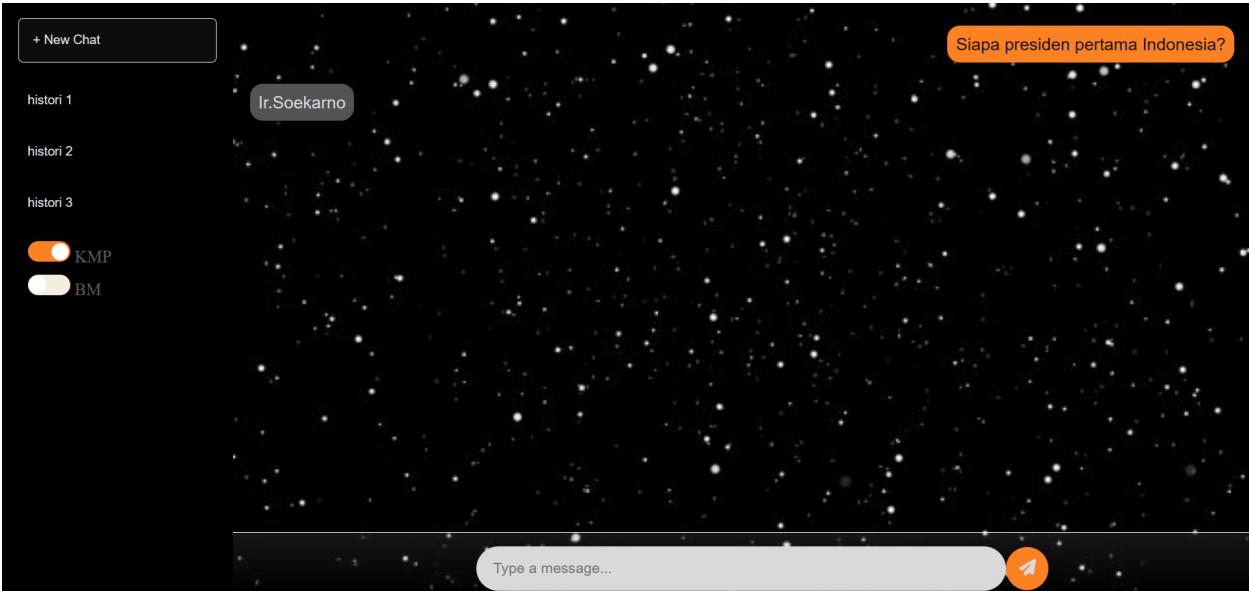
4.2 Tata Cara Penggunaan Program

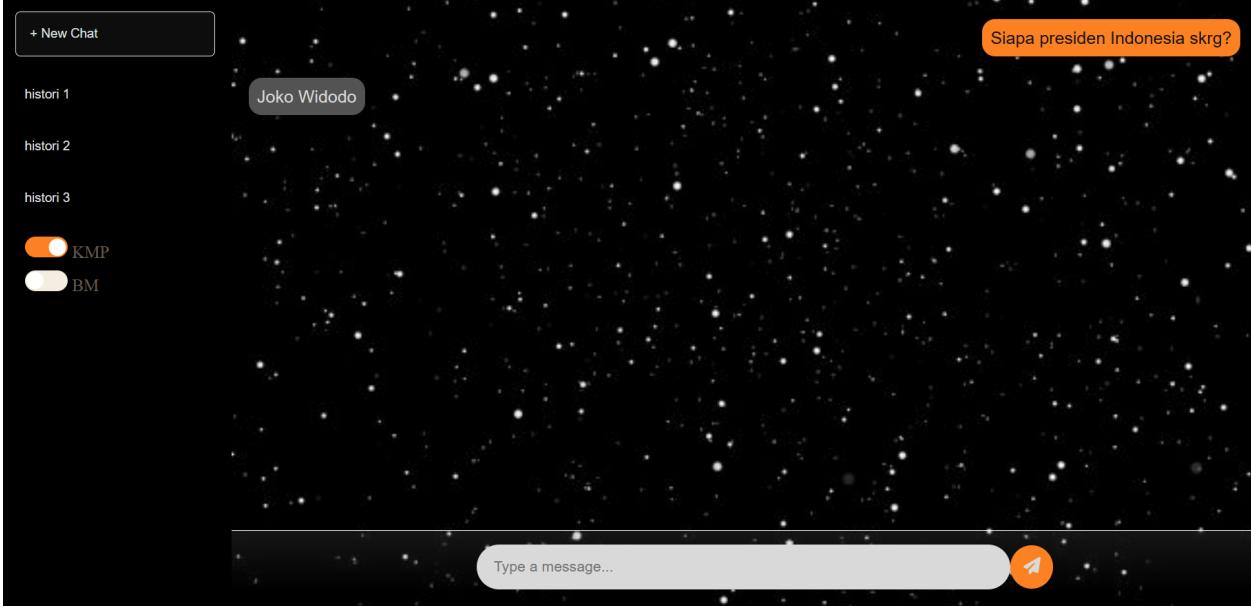
Program dapat digunakan dengan mengikuti langkah-langkah berikut ini.

1. Clone repository / download zip dari repository dari link yang ada pada lampiran
2. Pastikan anda sudah menginstall Golang
3. Masukkan file dump sql ke dalam database lokal yang ada pada device Anda dan atur file .env yang ada pada folder src/backend/models sehingga sesuai dengan database lokal Anda
4. Jalankan Backend command berikut
cd src/backend
go run main.go
5. Jalankan Frontend command berikut
cd src/frontend
npm install
npm run dev

4.3 Hasil Pengujian dan Analisis

4.3.1 Fitur Pertanyaan Teks

No	User Interface dan Analisis
1.	 <p>The screenshot shows a dark-themed chat application. At the top left is a button labeled '+ New Chat'. On the right, a message from 'Ir. Soekarno' asks, 'Siapa presiden pertama Indonesia?'. Below the messages, there are three history items: 'histori 1', 'histori 2', and 'histori 3'. At the bottom left are two toggle switches: one for 'KMP' (which is turned on) and one for 'BM'. A message input field at the bottom center contains the text 'Type a message...'. An orange send button with a white arrow icon is located at the bottom right.</p> <p>Siapa presiden pertama Indonesia? Connected! ini pertanyaan Masuk KMP 1 langsung Ir. Soekarno</p>

	Pada kasus ini, pertanyaan “presiden pertama Indonesia” ada di dalam database sehingga langsung mengeluarkan jawaban ketika masuk algoritma KMP yang dipilih.
2.	 <pre> Siapa presiden Indonesia skrg Connected! ini pertanyaan Masuk KMP -1 Masuk Levenshtein 85.18518518518519 85.18518518518519 0 Joko Widodo </pre> <p>Pada kasus ini, pertanyaan “presiden Indonesia skrg” tidak ada di dalam database dan pertanyaan dalam database yang paling mendekati ialah “presiden Indonesia sekarang” sehingga setelah masuk KMP dan tidak memenuhi maka akan masuk ke dalam Levenshtein. Dengan persentase kemiripan 85.185%, jawaban dari pertanyaan “presiden Indonesia sekarang” akan dikeluarkan.</p>

4.3.2 Fitur Kalkulator

No	User Interface dan Analisis
----	-----------------------------

1.

The screenshot shows a mobile application interface with a dark background featuring a starry pattern. At the top, there is a header bar with a button labeled '+ New Chat'. Below the header, there is a list of messages:

- histori 1: Hasilnya adalah 5.00
- histori 2
- histori 3

Below the messages are two toggle switches: one labeled 'KMP' and another labeled 'BM'. In the bottom right corner of the screen, there is a text input field with the placeholder 'Type a message...' and a send icon.

At the bottom of the screen, there is a large black text area containing the internal log of the calculator's processing of the expression $2+4+((5-6)^2/2)$. The log details the step-by-step evaluation of the expression, showing the state of the stack and the calculation of intermediate results.

```
Berapa 2+4+((5-6)^2/2)
Connected!
ini kalkulator
2+4+((5-6)^2/2)
[+ ( + ( ( - ) * / ) )
[2 4 5 6 2 2]
idxopen: 4, idxclose: 6
operandIn: [-], angkaIn: [5 6]
subResult: -1
operand: [+ ( + ( * / ) )], angka: [2 4 -1 2 2], subResult: -1
idxopen: 3, idxclose: 6
operandin: [* /], angkaIn: [-1 2 2]
subResult: -1
operand: [+ ( + )], angka: [2 4 -1], subResult: -1
idxopen: 1, idxclose: 3
operandin: [+], angkaIn: [4 -1]
subResult: 3
operand: [+], angka: [2 3], subResult: 3
Hasilnya adalah 5
Hasilnya adalah 5.00
operandIn: [-], angkaIn: [5 6]
subResult: -1
operand: [+ ( + ( * / ) )], angka: [2 4 -1 2 2], subResult: -1
idxopen: 3, idxclose: 6
operandin: [* /], angkaIn: [-1 2 2]
subResult: -1
operand: [+ ( + )], angka: [2 4 -1], subResult: -1
idxopen: 1, idxclose: 3
operandin: [+], angkaIn: [4 -1]
subResult: 3
operand: [+], angka: [2 3], subResult: 3
Hasilnya adalah 5
```

Pada kasus ini, sintaks pertanyaan memenuhi regex Calculator sehingga jawaban ekspresi matematika dikeluarkan dalam bubble chat halaman web

2.

The screenshot shows a mobile application interface with a dark background featuring a starry sky pattern. At the top left is a button labeled '+ New Chat'. On the right, a message bubble contains the text 'Berapa (2+(4+5)-6)*2)/2?'. Below this, there is a list of messages: 'histori 1' followed by 'Sintaks tidak sesuai', 'histori 2', and 'histori 3'. On the left side, there are two toggle switches: one orange switch labeled 'KMP' and one white switch labeled 'BM'. At the bottom, there is a text input field with the placeholder 'Type a message...' and a blue send button with a white arrow icon. A large text area at the bottom displays the following text:
Berapa (2+(4+5)-6)*2)/2?
Connected!
ini kalkulator
(2+(4+5)-6)*2)/2)
[((+ () -) *) /)]
[2 4 5 6 2 2]
Sintaks tidak sesuai

Pada kasus ini, sintaks pertanyaan tidak memenuhi regex Calculator sehingga string “Sintaks tidak sesuai” dikeluarkan dalam bubble chat halaman web

4.3.3 Fitur Tanggal

No	User Interface dan Analisis
----	-----------------------------

1.

The screenshot shows a messaging application interface with a dark background. On the left, there's a sidebar with a '+ New Chat' button, followed by three messages labeled 'histori 1', 'histori 2', and 'histori 3'. Below these are two toggle switches: 'KMP' (on) and 'BM' (off). On the right, a message bubble from 'hari apa' says 'Hari Jumat' and 'hari apa 5/5/2023?'. At the bottom, there's a text input field with 'Type a message...' placeholder and a send icon. Below the app, a terminal window displays the following text:

```
[GIN-debug] Listening and serving HTTP on :8080
/hari apa 5/5/2023
hari apa 5/5/2023
Connected!
ini kalender
```

Pada kasus ini pertanyaan yang dilempar oleh pengguna sesuai dengan regex yang sudah ada, sehingga pertanyaan akan langsung diproses sebagai kalender, lalu fungsi kalender akan mengecek apakah tanggal yang diberikan valid, jika akan mengembalikan hari seperti pada contoh diatas.

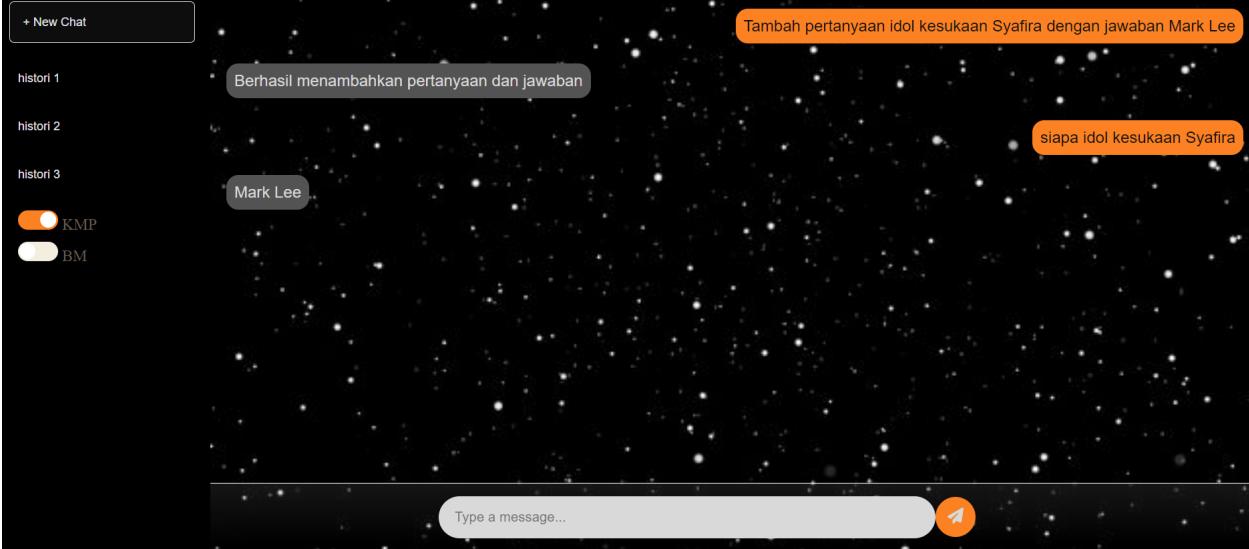
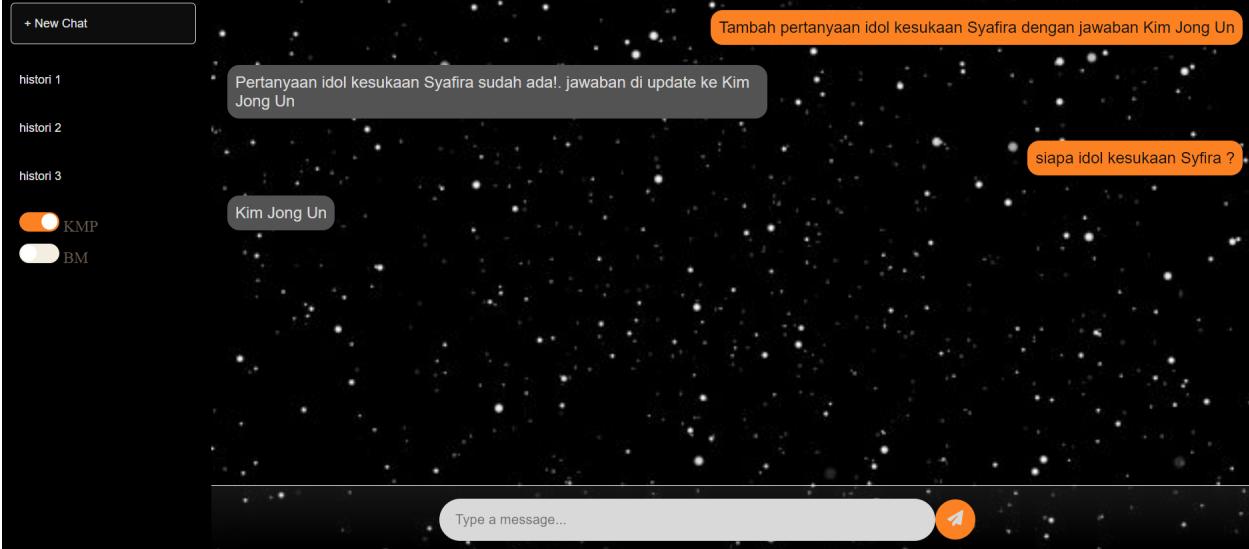
2.

The screenshot shows a messaging application interface with a dark background. On the left, there's a sidebar with a '+ New Chat' button, followed by three messages labeled 'histori 1', 'histori 2', and 'histori 3'. Below these are two toggle switches: 'KMP' (on) and 'BM' (off). On the right, a message bubble from 'hari apa' says 'Invalid Date' and 'hari apa 30/2/2029?'. At the bottom, there's a text input field with 'Type a message...' placeholder and a send icon. Below the app, a terminal window displays the following text:

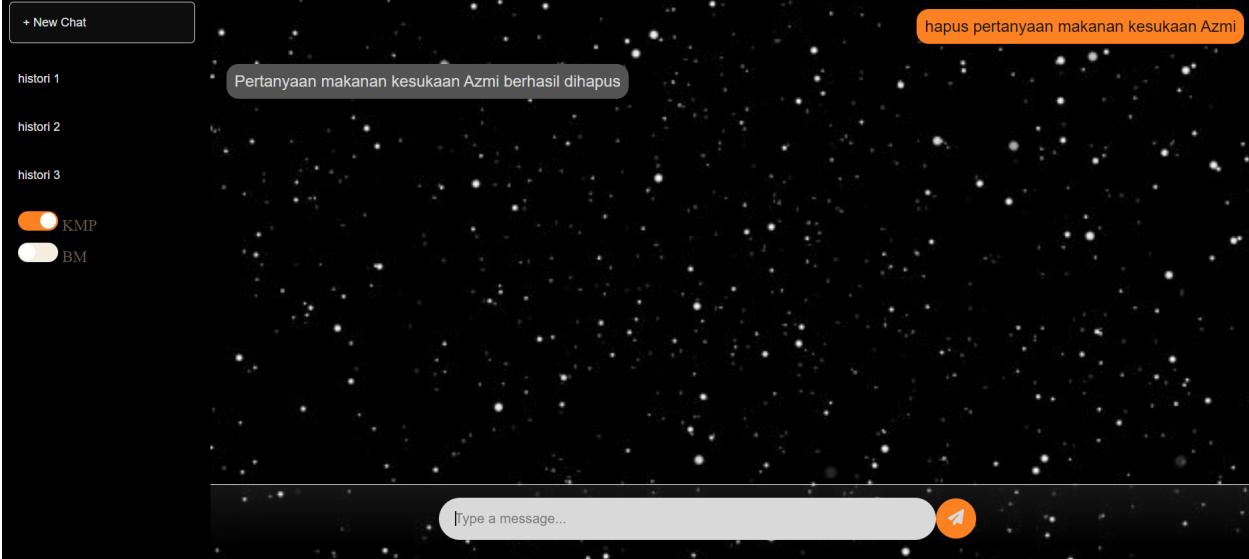
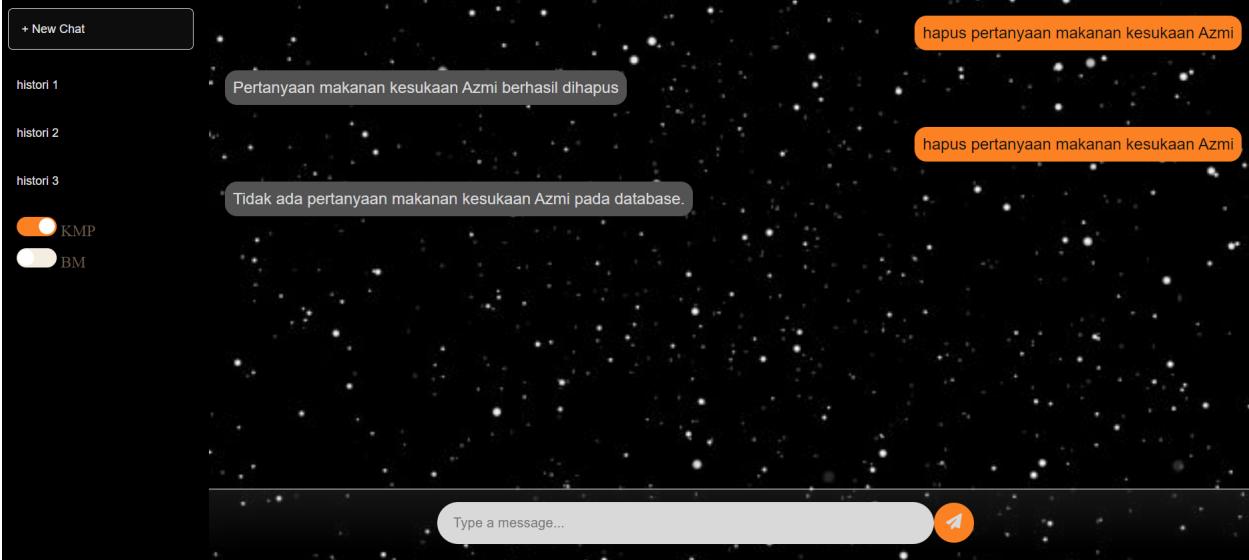
```
Type a message...
```

Pada kasus ini, tanggal yang diberikan invalid karena tidak ada tanggal 30 Februari, sehingga program mengembalikan Invalid Date.

4.3.4 Fitur Tambah Pertanyaan dan Jawaban

No	User Interface dan Analisis
1.	 <p>Pada kasus ini pertanyaan “idol kesukaan Syafira” tidak ada pada database, lalu program menambahkan jawaban dan pertanyaan kedalam database, sehingga bisa diakses setelahnya.</p>
2.	 <p>Pada kasus ini pertanyaan “idol kesukaan Syafira” sudah ada pada database, sehingga program akan mengupdate jawaban dengan yang baru.</p>

4.3.5 Fitur Hapus Pertanyaan

No	User Interface dan Analisis
1.	 <p>Pada kasus ini query pengguna yaitu “hapus pertanyaan makanan kesukaan Azmi”, query tersebut ada pada database, sehingga query berhasil dihapus</p>
2.	 <p>Pada kasus ini, query tidak ada pada database, sehingga mengembalikan pesan tidak ada pertanyaan.</p>

BAB V

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 pelaksanaan 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 pelaksanaan 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 pelaksanaan 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

<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>

<https://github.com/labstack/echo>

<https://reactjs.org/docs/getting-started.html#react-for-beginners>

<https://sequelize.org/docs/v6/getting-started/>

LAMPIRAN

Tautan Github: https://github.com/afnanramadhan/Tubes3_13521011.git