

Penerapan String Matching dan Regular Expression dalam Pembuatan ChatGPT Sederhana

Diajukan sebagai pemenuhan Tugas Besar 3 Mata Kuliah IF2211 Strategi Algoritma



Oleh:

Kelompok Chatty-Loli

1. 13521131 Jeremya Dharmawan Raharjo
2. 13521155 Kandida Edgina Gunawan
3. 13521156 Brigita Tri Carolina

Dosen Pengampu: Dr. Ir. Rinaldi Munir, MT.

IF2211 - Strategi Algoritma

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

2023

DAFTAR ISI

DAFTAR ISI	1
BAB I	2
BAB II	3
2.1. Algoritma Program	3
2.1.1 Algoritma Knuth-Morris-Pratt (KMP)	3
2.1.2 Algoritma Boyer-Moore (BM)	4
2.1.3 Regular Expression	5
2.2. Aplikasi Berbasis Web	5
BAB III	7
3.1 Analisis Pemecahan Masalah	7
3.2 Langkah Penyelesaian Masalah Tiap Fitur	7
3.3 Fitur Fungsional dan Arsitektur Aplikasi yang Dibangun	8
BAB IV	9
4.1 Spesifikasi Teknis Program	9
4.1.1 Struktur Data	9
4.1.2 Fungsi dan Prosedur	13
4.2 Tata Cara Penggunaan Program	16
4.3 Hasil Pengujian	20
4.4 Analisis Hasil Pengujian	23
BAB V	25
KESIMPULAN, SARAN, DAN REFLEKSI	25
DAFTAR REFERENSI	26
LAMPIRAN	27

BAB I

DESKRIPSI TUGAS

Dalam dunia teknologi, chatbot telah menjadi hal yang umum digunakan dalam berbagai aplikasi dan platform seperti situs web, aplikasi mobile, dan media sosial. Chatbot memungkinkan pengguna untuk berinteraksi dengan program yang memiliki kemampuan untuk memproses dan merespons percakapan secara otomatis. Salah satu contoh chatbot yang sedang booming saat ini adalah ChatGPT.

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.

BAB II

DASAR TEORI

2.1. Algoritma Program

2.1.1 Algoritma Knuth-Morris-Pratt (KMP)

Algoritma Knuth-Morris-Pratt (KMP) adalah sebuah algoritma pencocokan string yang digunakan untuk mencari kemunculan sebuah pola atau pattern dalam sebuah teks atau string. Algoritma ini ditemukan oleh Donald Knuth, Vaughan Pratt, dan James H. Morris pada tahun 1977.

Pada dasarnya, algoritma KMP bekerja dengan memanfaatkan informasi tentang pola yang ingin dicocokkan dalam proses pencarian. Dalam algoritma ini, dibangun sebuah tabel yang dikenal sebagai tabel prefix-fungsi. Tabel ini digunakan untuk mengetahui kemungkinan kemunculan pola yang cocok pada setiap posisi dalam teks, sehingga tidak perlu memulai pencarian dari awal setiap kali tidak cocok. Implementasi tabel ini, yang kadang disebut sebagai *border function*, membuat kompleksitas algoritma ini lebih mangkus dibandingkan algoritma *brute-force*.

Langkah-langkah algoritma KMP secara umum adalah sebagai berikut:

1. Buat tabel prefix dari pola yang akan dicari. Tabel ini akan menentukan kemungkinan posisi awal pola(p) pada string dengan *border function*.
2. Inisialisasi indeks i dan j ke 0.
3. Pada setiap iterasi, bandingkan karakter ke-i pada teks dengan karakter ke-j pada pola.
4. Jika karakter pada kedua posisi sama, maka naikan kedua indeks i dan j.
5. Jika j mencapai panjang pola, berarti pola ditemukan pada posisi i-j dan j di-set kembali ke p[j-1]. Kembalilah ke langkah 3 untuk mencari kemungkinan pola berikutnya pada string.
6. Jika karakter pada kedua posisi tidak sama, maka naikan nilai j dengan nilai p[j-1] (*border function* dari karakter ke-j pada pola).

7. Jika j masih 0, artinya tidak ada kemungkinan untuk mencocokkan pola lagi pada teks. Naikkan nilai i ke posisi berikutnya dan kembali ke langkah 3.
8. Ulangi langkah 3 hingga i mencapai akhir dari teks atau pola telah ditemukan.
9. Jika pola ditemukan, kembalikan indeks $i-j$. Jika tidak, kembalikan -1.
10. Selesai.

2.1.2 Algoritma Boyer-Moore (BM)

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

Pada dasarnya, algoritma BM bekerja dengan memanfaatkan dua strategi: strategi karakter skip dan strategi good suffix skip. Strategi karakter skip mengambil langkah mundur dalam pola yang cocok dengan karakter yang tidak cocok dalam teks, sedangkan strategi good suffix skip mengambil langkah maju dalam pola yang cocok dengan awalan yang cocok di dalam pola.

Langkah-langkah algoritma BM secara umum adalah sebagai berikut:

1. Pada tahap persiapan, buat dua tabel yang diperlukan:
 - Tabel karakter terakhir (*last occurrence table*): Untuk setiap karakter dalam alfabet, tabel ini menyimpan posisi kemunculan terakhir karakter tersebut dalam pola yang ingin dicari. Jika karakter tersebut tidak muncul dalam pola, tabel ini menyimpan nilai -1.
 - Tabel geser (*bad character shift table*): Tabel ini menyimpan jumlah pergeseran yang harus dilakukan jika ada ketidakcocokan antara karakter pada teks dengan pola yang sedang dicocokkan. Pergeseran dilakukan berdasarkan karakter yang tidak cocok terakhir dalam pola.
2. Mulai pencarian pola dalam teks. Pertama, inisialisasi indeks awal pencocokan di teks dengan nilai 0.

3. Lakukan pencocokan pola dengan teks mulai dari akhir pola (dari kanan ke kiri) dan maju ke kiri selama masih ada kemungkinan pencocokan. Pada setiap tahap pencocokan, lakukan hal berikut:
 - Bandingkan karakter terakhir pola dengan karakter pada teks di indeks pencocokan saat ini.
 - Jika kedua karakter cocok, periksa karakter sebelumnya pada pola dan teks secara berurutan hingga seluruh pola cocok atau terdapat ketidakcocokan.
 - Jika pola cocok dengan teks, pencarian berhasil dan indeks awal pencocokan ditemukan. Geser indeks awal pencocokan sejauh panjang pola dan mulai pencarian kembali dari indeks baru tersebut.
 - Jika terdapat ketidakcocokan, geser indeks awal pencocokan sejauh maksimum dari dua nilai berikut:
 - Nilai pergeseran karakter pada tabel geser untuk karakter yang tidak cocok terakhir dalam pola.
 - Jumlah karakter yang sudah dicocokkan dalam pola (tanpa memperhitungkan karakter yang tidak cocok terakhir).
4. Jika pencarian selesai dan pola tidak ditemukan di dalam teks, pencarian dihentikan.

2.1.3 Regular Expression

Regular expression, juga dikenal sebagai regex atau regexp, adalah sebuah urutan karakter yang digunakan untuk mencari dan mencocokkan pola teks tertentu dalam sebuah string. Regular expression dapat digunakan untuk melakukan operasi pencarian dan penggantian pada teks secara efisien dan fleksibel.

Regular expression terdiri dari serangkaian karakter khusus yang menentukan pola tertentu. Karakter tersebut dapat mencakup huruf, angka, karakter khusus, dan operator.

2.2. Aplikasi Berbasis Web

Aplikasi berbasis web adalah sebuah program yang disimpan pada suatu server yang berada di jarak yang jauh, dan kemudian diteruskan melalui jaringan internet dan ditampilkan

melalui antarmuka browser. Aplikasi Web diciptakan untuk berbagai keperluan dan dapat diakses oleh siapa saja. Beberapa contoh Aplikasi Web yang populer adalah webmail, kalkulator online, dan toko e-commerce. Dalam penggunaannya, Aplikasi Web tidak perlu diunduh dan dijalankan secara lokal, sehingga membutuhkan server web, server aplikasi, dan server database untuk beroperasi secara efektif. Server web bertanggung jawab untuk menangani permintaan dari klien, server aplikasi mengelola tugas yang diminta, dan database digunakan untuk menyimpan informasi yang dibutuhkan.

BAB III

ANALISIS DAN PEMECAHAN MASALAH

3.1 Analisis Pemecahan Masalah

Terdapat lima sub-masalah yang bisa diidentifikasi dari latar belakang permasalahan, di antaranya melakukan kalkulasi operasi matematika, melakukan kalkulasi hari dari tanggal yang di-*input*, menambahkan *record* pertanyaan dengan jawaban tertentu pada *database* atau meng-*update* jawaban jika pertanyaan sudah terdapat pada *database*, menghapus *record* pertanyaan pada *database*, dan menjawab pertanyaan berdasarkan data pada *database* berikut dengan algoritma *pattern matching* pada program.

3.2 Langkah Penyelesaian Masalah Tiap Fitur

1. Input operasi matematika dan kalkulasi operasi matematika
Pertama-tama dengan menggunakan regex, aplikasi menentukan apakah pertanyaan yang di-*input user* termasuk ke operasi matematika. Kemudian, digunakan sebuah stack untuk memvalidasi tanda kurung dari operasi matematika yang di-*input*. Setelah operasi matematika sudah tervalidasi merupakan operasi matematika yang valid maka dipanggil fungsi *calculateMathOperation* untuk menghitung operasi matematika dengan tingkat prioritas kurung, pangkat, perkalian/pembagian, dan pertambahan/pengurangan.
2. Input tanggal dan kalkulasi hari
Dengan menggunakan regex, maka pertanyaan diklasifikasi dan tervalidasi merupakan pertanyaan tanggal. Kemudian dipanggil fungsi *calculateDate* dengan masukan day, month, dan year. Fungsi kemudian meng-*convert* year dan month ke hari juga menentukan apakah tahun yang dimasukkan merupakan tahun kabisat dan jika iya dan bulan melewati Februari maka day ditambah 1 dan pada akhirnya ditambah dengan hari. Kemudian total hari yang telah dihitung akan dibagi dengan 7. Sisa dari pembagian tersebut kemudian dimasukkan pada fungsi *mapRemainderOfTheWeek* untuk menghasilkan string hari.
3. Menambahkan pertanyaan pada *database*
Dengan menggunakan regex, pertanyaan diklasifikasi dan tervalidasi merupakan kalimat yang berisi perintah untuk menambahkan pertanyaan pada *database*. Kemudian memanggil fungsi *Create* untuk menambahkan pertanyaan dengan jawaban tertera pada *database*. Pada prosesnya dilakukan pemanggilan algoritma KMP atau BM untuk menentukan apakah pertanyaan sudah ada pada *database*, jika sudah ada maka program hanya akan meng-*update* jawaban yang telah ada.
4. Menghapus pertanyaan pada *database*
Dengan menggunakan regex, pertanyaan diklasifikasi dan tervalidasi merupakan kalimat yang berisi perintah untuk menghapus pertanyaan pada *database*. Setelah itu dilakukan

fungsi *delete* untuk menghapus *record* pertanyaan pada *database*. Pada prosesnya dilakukan pemanggilan algoritma KMP atau BM untuk menentukan pertanyaan mana yang *matching* dan dapat dihapus. Jika tidak terdapat pertanyaan yang *match*, maka aplikasi akan mengirimkan pesan berupa pertanyaan tidak dapat dihapus.

5. Menjawab pertanyaan berdasarkan data pada *database* dan algoritma *pattern matching*
Dengan menggunakan regex, pertanyaan diklasifikasi dan tervalidasi merupakan kalimat yang berisi pertanyaan acak dari *user*. Program kemudian memanggil fungsi KMPMatch atau BMMatch untuk menentukan pertanyaan mana yang cocok yang ada pada *database*. Jika tidak terdapat pertanyaan yang cocok berdasarkan ketentuan *similarity* yang telah di-*set* maka aplikasi akan mengirimkan pesan berupa 3 saran pertanyaan yang paling mirip dengan pertanyaan masukan *user*.

3.3 Fitur Fungsional dan Arsitektur Aplikasi yang Dibangun

3.3.1 Fitur Fungsional

Beberapa fitur fungsional yang dapat dijalankan pada aplikasi web.

1. Melakukan perhitungan operasi matematika berupa perpangkatan, perkalian, pembagian, penambahan, pengurangan dan kurung.
2. Melakukan kalkulasi hari dari tanggal yang dimasukkan *user*.
3. Menambahkan pertanyaan pada *database*.
4. Meng-*update* jawaban pada *database* jika pertanyaan sudah ada pada *database*.
5. Menghapus pertanyaan pada *database* berdasarkan pesan kiriman *user*.
6. Menjawab pertanyaan acak yang dikirimkan oleh *user*.

3.3.2 Arsitektur Aplikasi

Aplikasi berbasis web ini menggunakan *mySQL* sebagai basis data, aplikasi *back-end* menggunakan bahasa pemrograman *Go* yang di-deploy dengan platform *railway*, dan aplikasi *front-end* dengan berupa *next.js*. yang di-deploy dengan platform *vercell app*

BAB IV

IMPLEMENTASI DAN PENGUJIAN PROGRAM

4.1 Spesifikasi Teknis Program

4.1.1 Struktur Data

Pada pembuatan aplikasi chatbot *Chatty Loli* berbasis web, terdapat beberapa struktur data yang digunakan, di antaranya sebagai berikut.

1. *Struct*

Struct merupakan salah satu tipe data dalam Bahasa Go. *Struct* merupakan tipe data yang digunakan untuk beberapa variabel dengan tipe data berbeda menjadi satu kesatuan. *Struct* mirip dengan *record* dalam bahasa pemrograman lain. *Struct* dalam bahasa Go dideklarasikan dengan menggunakan kata kunci 'type' diikuti dengan nama *struct* dan definisi *field*-nya. Berikut adalah beberapa implementasi *struct* yang digunakan pada program kami yang terdapat pada model.go.

```
package model

type QnA struct {
    Id      uint   `gorm:"primaryKey" json:"Id"`
    Question string `gorm:"type: LONGTEXT" json:"Question"`
    Answer  string `gorm:"type: LONGTEXT" json:"Answer"`
}

type InputUser struct {
    Id      uint   `json:"Id"`
    Session int64  `json:"Session" gorm:"foreignKey:Sessions"`
    InputText string `json:"InputText"`
    Algorithm bool   `json:"Algorithm" //true : kmp, false : bm`
    Answer  string `json:"Answer"`
}

type Sessions struct {
    Id      uint   `json:"Id"`
    Session int64  `json:"Session"`
}

type Request struct {
    Question string
    Answer   string
    Algorithm bool
}
```

2. *Array*

Struktur data lain yang digunakan pada program ini adalah *array*. Salah satunya adalah *array* dengan tipe *string*. *Array of string* digunakan untuk menyimpan operasi matematika pada sebuah buffer. Kemudian buffer diiterasi untuk menyelesaikan operasi matematika tersebut. Berikut adalah implementasi *array of string* yang digunakan pada mathop.go.

```
func calculateMathOperation(input string) string {  
    matharray := regexp.MustCompile(`(\-?\d+(\.\d+)?|\/|\(|\)|\^|\*|\/|\+|\-)`)  
    matcharray := matharray.FindAllStringSubmatch(input, -1)  
    var buffer []string  
    counter := 0  
  
    for _, match := range matcharray {  
        num, err := strconv.ParseFloat(match[1], 64)  
        if err == nil { // kalo operand  
            if num < 0 {  
                if counter >= 1 && !isOperator(buffer[counter - 1]) {  
                    buffer = append(buffer, "-")  
                    buffer = append(buffer, strconv.FormatFloat(-1*num, 'f', 2, 64))  
                    counter += 2  
                } else {  
                    buffer = append(buffer, strconv.FormatFloat(num, 'f', 2, 64))  
                    counter++  
                }  
            } else {  
                buffer = append(buffer, strconv.FormatFloat(num, 'f', 2, 64))  
                counter++  
            }  
        } else {  
            buffer = append(buffer, match[1])  
            counter++  
        }  
    }  
}
```

Implementasi *array of int* pada program.

```

func calculateBorder(pattern string) []int {
    var b [1000]int
    m := len(pattern)
    j := 0
    i := 1
    k := m-1

    for i < m {
        if pattern[j] == pattern[i] { // chars match
            b[i] = j + 1
            j++
            i++
        } else if j > 0 {
            j = b[j-1]
        } else {
            b[i] = 0
            i++
        }
    }

    return b[:k]
}

```

3. *Multidimensional Array*

Multidimensional Array pada Bahasa Golang tipe *array of array*. Pada program ini multidimensional program digunakan pada pencarian minimumDist pada algoritma Levenshtein. Berikut adalah implementasi multidimensional array pada program ini.

```

func MinimumDist(s, t string) int {
    m := len(s)
    n := len(t)

    // initialize the distance matrix
    d := make([][]int, m+1)
    for i := range d {
        d[i] = make([]int, n+1)
        d[i][0] = i
    }
    for j := range d[0] {
        d[0][j] = j
    }

    // fill the distance matrix
    for j := 1; j <= n; j++ {
        for i := 1; i <= m; i++ {
            if s[i-1] == t[j-1] {
                d[i][j] = d[i-1][j-1]
            } else {
                d[i][j] = min3Int(d[i-1][j]+1, d[i][j-1]+1, d[i-1][j-1]+1)
            }
        }
    }

    return d[m][n]
}

```

4. Kelas

Pada pengembangan *front-end*, kami menggunakan pustaka ReactJS dari bahasa pemrograman JavaScript. Bahasa JavaScript sendiri mendukung paradigma berorientasi objek untuk mengembangkan aplikasi. Berikut merupakan implementasi kelas dari program kami.

```

16 // import pattern from .../pages/pattern/pattern.js
17
18
19
20 class Sidebar extends React.Component {
21   constructor(props){
22     super(props);
23     this.state = {
24       isKMP : false,
25       isBM : false,
26       showHistory: true,
27       history : [],
28     };
29   }
30
31   setKMP = (e) =>{
32     this.setState({
33       isKMP : !this.state.isKMP
34     })
35     if(!this.state.isKMP)this.props.handleAlgoChange({target: {value: true}});
36     else this.props.handleAlgoChange({target: {value: this.state.isKMP}});
37   }
38
39   setBM = (e) =>{
40     this.setState({
41       isBM : !this.state.isBM
42     })
43     if(!this.state.isBM)this.props.handleAlgoChange({target: {value: false}});
44     else this.props.handleAlgoChange({target: {value: this.state.isKMP}});
45   }

```

4.1.2 Fungsi dan Prosedur

Program ini terbagi menjadi dua bagian yaitu *front-end* dan *back-end*. Beberapa fungsi dan prosedur dikelompokkan menjadi fungsi dan prosedur pada *front-end* dan *back-end*.

1. Back-end

Berikut adalah beberapa fungsi dan prosedur yang digunakan pada *back-end*.

Fungsi/Prosedur	Penjelasan
BMMatch(text string, pattern string)	Fungsi untuk memproses <i>pattern</i> dan <i>text</i> untuk dicocokkan dengan algoritma Boyer More yang akan mengembalikan nilai apakah patter sama dan kesamaan string(dihitung dengan jarak Levensthein)
buildLast(pattern string)	Fungsi mengembalikan 128 slice pertama dari last.
calculateDate(day int, month int, year int)	Melakukan kalkulasi hari berdasarkan masukan <i>day</i> , <i>month</i> , <i>year</i>

mapRemainderOfWeek(remainder int)	Mengembalikan string hari berdasarkan masukan sisa hari.
leap(year int)	Mengembalikan nilai 1 jika tahun masukan adalah <i>leap year</i> dan 0 jika tahun masukan bukan <i>leap year</i> .
leap_bool(year int)	Mengembalikan <i>true</i> jika tahun masukan adalah <i>leap year</i> .
KMPMatch(text string, pattern string)	Fungsi untuk memproses <i>pattern</i> dan <i>text</i> untuk dicocokkan dengan algoritma Knuth-Morris-Pratt yang akan mengambil index pertama <i>match</i> dan index -1 jika tidak terdapat <i>pattern</i> yang tidak <i>match</i> .
calculateBorder(pattern string)	Menghitung batas prefix dan postfix pada atau <i>border</i> pada <i>pattern</i> .
min3Int(x int, y int, z int)	Mengembalikan nilai terkecil dari ketiga integer.
MinimumDist(s, t string)	Mengembalikan nilai Levenshtein distance antara string s dan t.
similarityPercentage(text string, pattern string)	Menghasilkan persen kemiripan dari <i>pattern</i> ke text.
isOperator(input string)	Menghasilkan <i>true</i> apabila input adalah operator.
calculateMathOperation(input string)	Mengembalikan nilai hasil perhitungan dari string operasi matematika.
removeElement(arr []string, index int)	Menghapus sebuah element pada index index dari array arr.
insertElement(arr [] string, element string, index int)	Menyisipkan sebuah string element pada index index di array arr.
MathOperation(pattern string)	Berisi regex yang akan menghasilkan <i>true</i> jika <i>pattern matching</i> dengan regex.
IsMathOperationValid(pattern string)	Berisi regex yang akan menghasilkan <i>true</i> jika <i>pattern matching</i> dengan regex.

IsDate(pattern string)	Berisi regex yang akan menghasilkan <i>true</i> jika <i>pattern matching</i> dengan regex.
IsAddingQNAToDatabase(pattern string)	Berisi regex yang akan menghasilkan <i>true</i> jika <i>pattern matching</i> dengan regex.
IsErasingQuestion(pattern string)	Berisi regex yang akan menghasilkan <i>true</i> jika <i>pattern matching</i> dengan regex.
IsDayOutput(pattern string)	Menghasilkan <i>true</i> apabila string <i>pattern</i> merupakan suatu hari.
CheckQuestion(pattern string)	Melakukan cek pada string pertanyaan dari input <i>user</i> dan melakukan eksekusi pertanyaan sesuai dengan klasifikasi pertanyaan.
Index(c *fiber.Ctx) (inputController)	Mengembalikan <i>records input</i> yang terdapat pada <i>database</i> .
Show(c *fiber.Ctx) (inputController)	Fungsi mengembalikan <i>records</i> pada <i>database</i> yang terdapat pada <i>session</i> sekarang.
Create(c *fiber.Ctx) (inputController)	Fungsi merangkai jawaban dari beberapa pertanyaan yang di- <i>input</i> oleh <i>user</i> untuk ditampilkan kepada <i>user</i> .
Index(c* fiber.Ctx) (qnaController)	Fungsi mengembalikan <i>records</i> qna yang terdapat pada <i>database</i> .
Create(c* fiber.Ctx) (qnaController)	Fungsi mengembalikan jawaban atas pertanyaan yang diberikan oleh <i>user</i> .
Delete(c* fiber.Ctx) (qnaController)	Fungsi menghapus pertanyaan dari <i>database</i> kemudian mengembalikan pesan sukses atau gagal.
Update(c* fiber.Ctx) (qnaController)	Fungsi meng- <i>update</i> jawaban dari pertanyaan yang terdapat pada <i>database</i> .
Index(c* fiber.Ctx) (sessionController)	Fungsi mengembalikan <i>records</i> session yang terdapat pada <i>database</i> .
Create(c* fiber.Ctx) (sessionController)	Fungsi menambahkan <i>records session</i> pada <i>database</i> .

Connect()	Fungsi ini menyambungkan program dengan <i>database</i> .
-----------	---

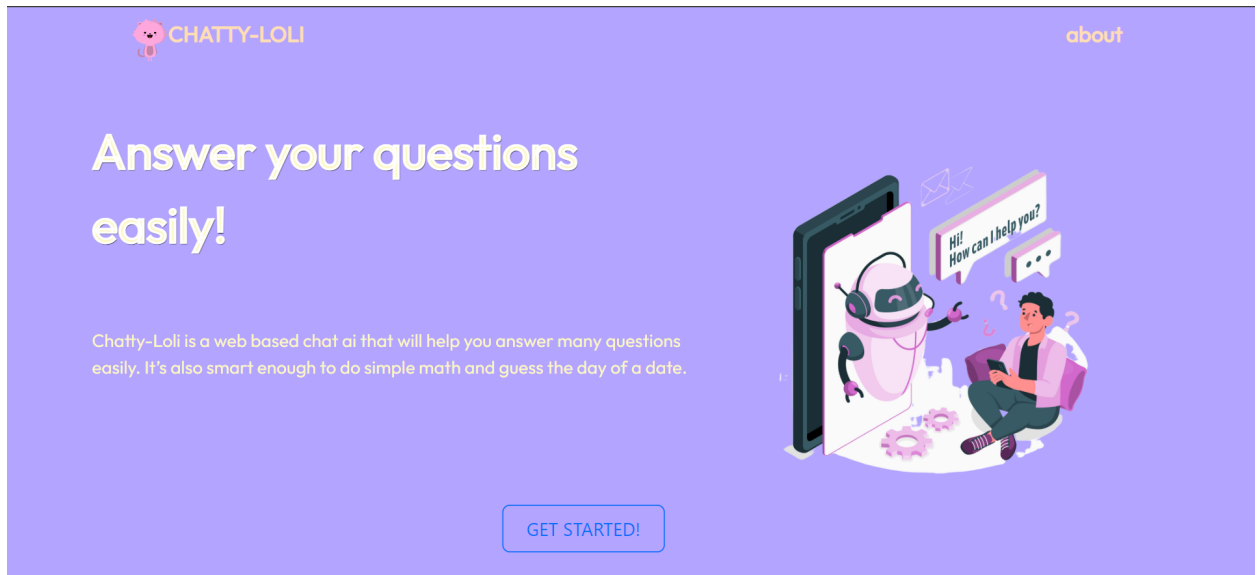
2. *Front-end*

Kelas	Penjelasan
App	Basis dari pengembangan <i>front-end</i>
FirstPage	Halaman utama antarmuka
About	Halaman mengenai pembuat dan link repository github
MainGPT	Halaman ChatBot
SideBar	Navigation Bar disamping kiri halaman ChatBot

4.2 Tata Cara Penggunaan Program

4.2.1 Antarmuka Program

1. Halaman Utama



Untuk memasuki halaman *chat*, pengguna cukup menekan tombol *get started*

2. *About Page*



Gambar di atas merupakan *display* dari *about page*.

3. Input Tanggal

Setelah memasuki halaman *chat* pengguna cukup memasukkan query berupa tanggal untuk mendapatkan prediksi hari pada tanggal tersebut.



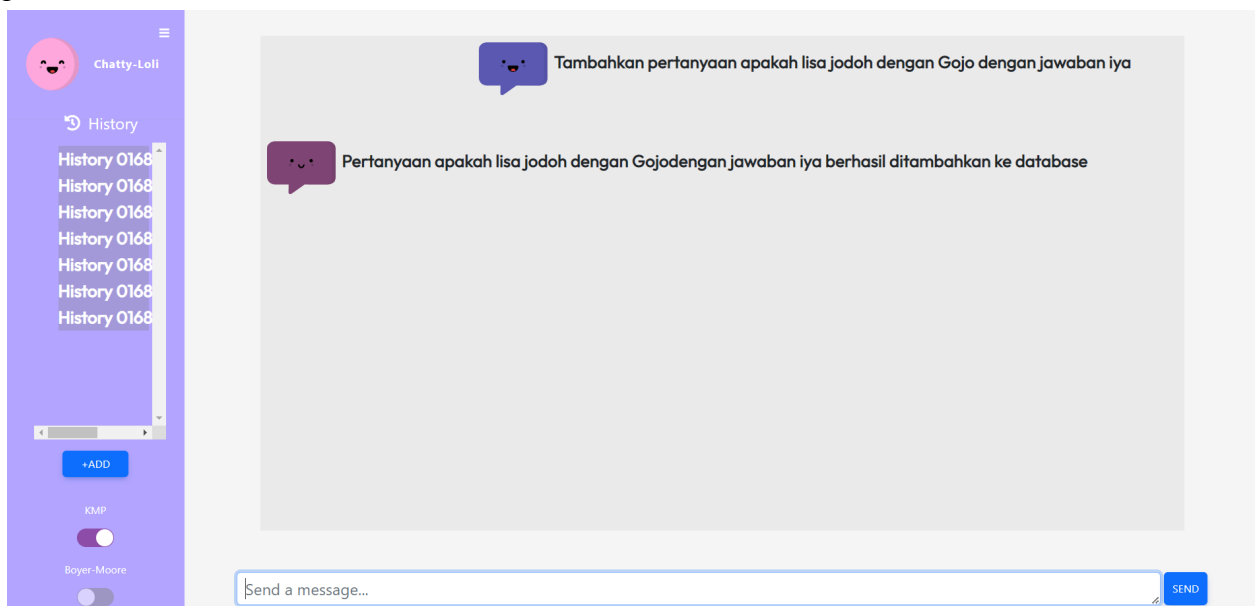
4. Input Operasi Matematika

Pengguna cukup memasukkan operasi matematika sebagai berikut pada halaman *chat*.

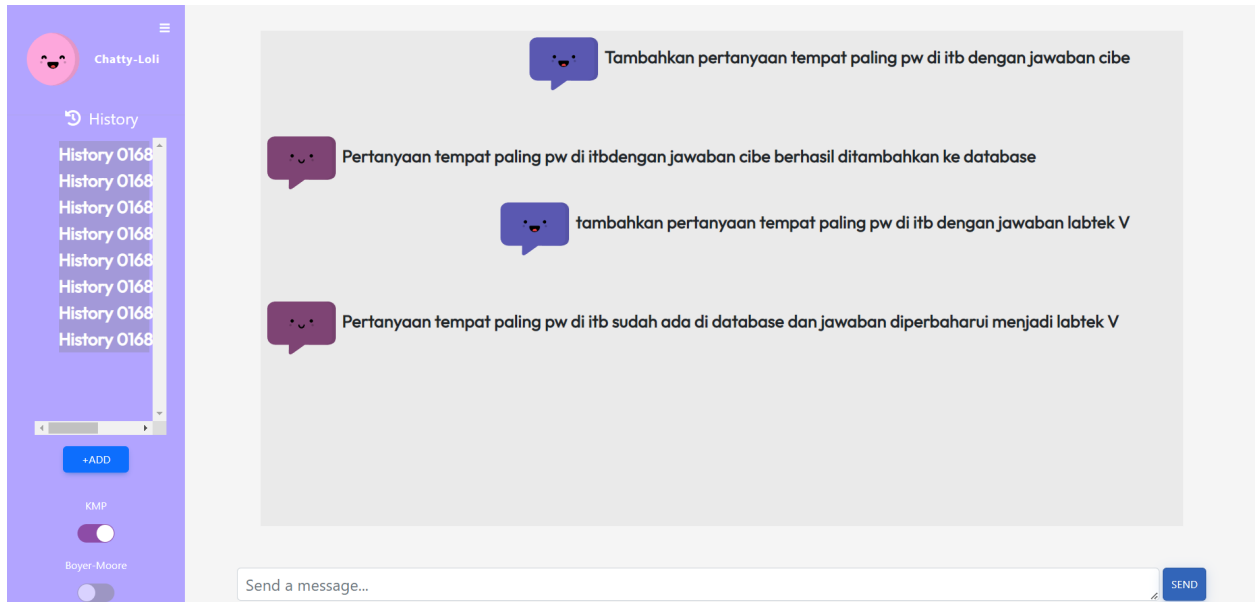


5. Menambah Pertanyaan

Pengguna cukup mengirimkan pesan sebagai berikut untuk menambahkan pertanyaan pada *database*.

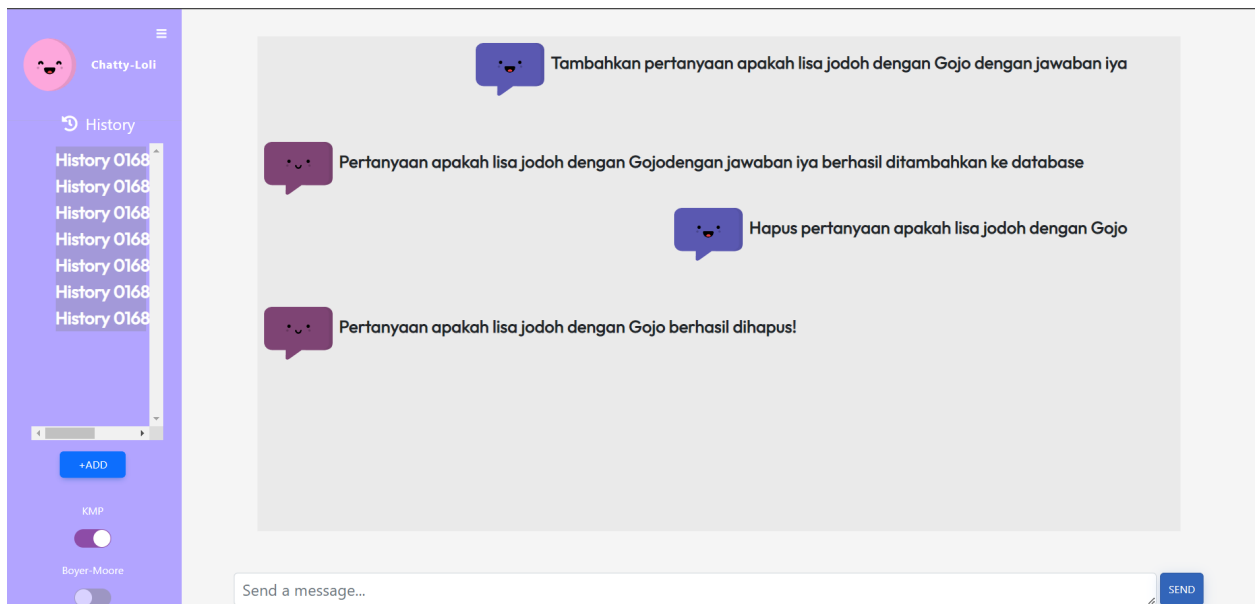


Jika pertanyaan telah ada pada *database* maka bot akan meng-*update* hanya jawaban saja.

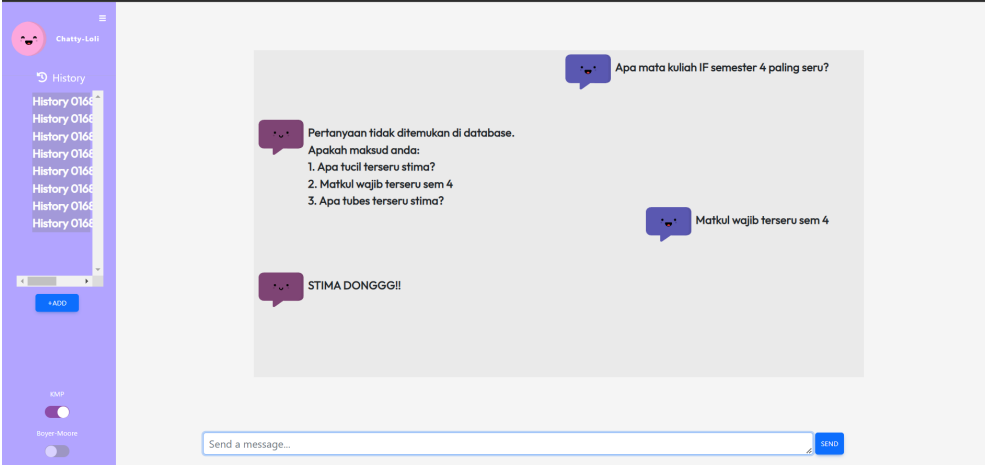
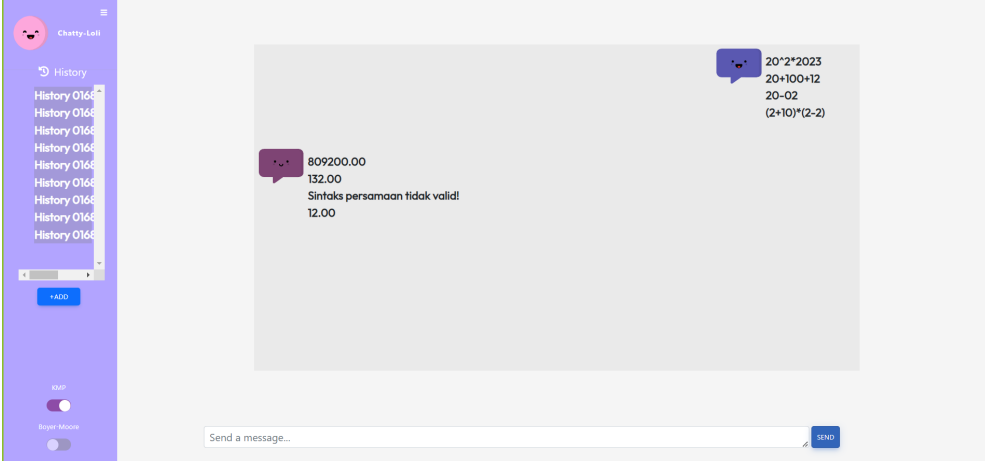




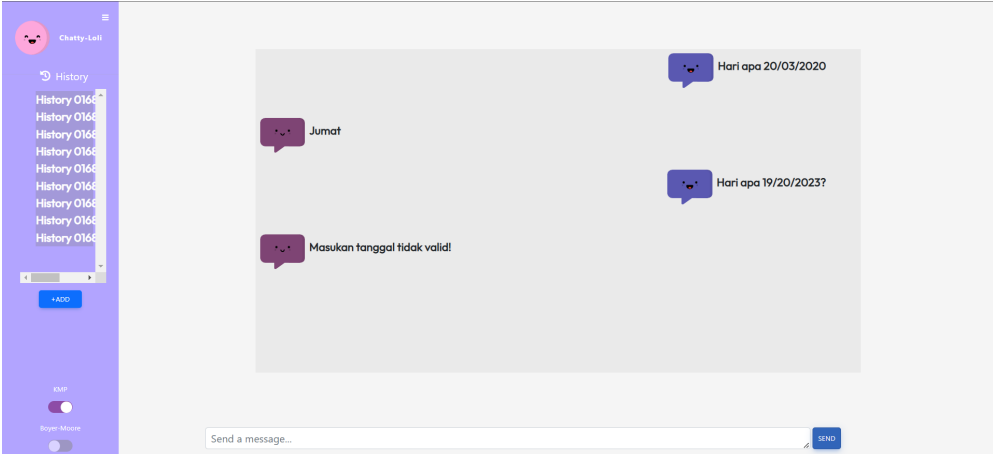
6. Menghapus pertanyaan pada *database*

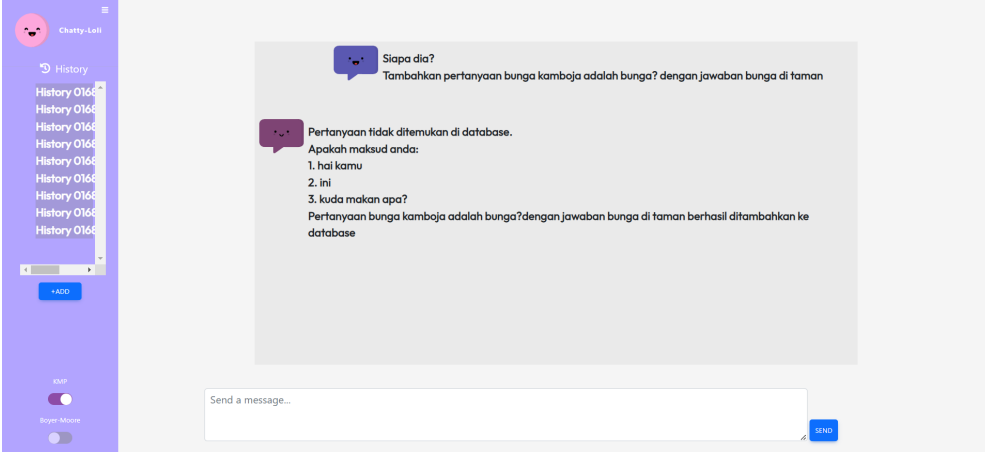


Pengguna cukup mengirimkan pesan sebagai berikut untuk menghapus pertanyaan pada *database*.




4.3 Hasil Pengujian

Tes 1	
Tes 2	
Tes 3	

<p>Tes 4</p>	
<p>Tes 5</p>	
<p>Tes 6</p>	

<p>Tes 7</p>	
<p>Tes 8</p>	
<p>Tes 9</p>	

<p>Tes 10</p>	
<p>Tes 11</p>	

4.4 Analisis Hasil Pengujian

Berdasarkan pengujian yang telah dilakukan, aplikasi dapat melakukan *input* operasi matematika, validasi operasi matematika, dan *output* hasil perhitungan matematika, *input* tanggal, validasi tanggal, dan *output* hari berdasarkan tanggal yang di-*input*, menambah *record* pertanyaan berdasarkan *input user*, menghapus *record* pertanyaan berdasarkan *input user*, dan menjawab pertanyaan acak dari *input user*.

Untuk *input* operasi matematika dikirimkan oleh *user*, regex melakukan klasifikasi pertanyaan kemudian memvalidasi operasi matematika. Aplikasi akan mengirimkan pesan “Sintaks persamaan tidak valid!” jika operasi matematika yang di-*input* oleh *user* tidak valid.

Untuk *input* tanggal yang dikirimkan oleh *user*, regex akan melakukan klasifikasi pertanyaan kemudian memvalidasi input tanggal yang dikirimkan oleh *user*. Aplikasi akan mengirimkan pesan “Masukan tanggal tidak valid!” jika tanggal yang dimasukkan oleh *user* tidak valid.

Pada tipe input selanjutnya yaitu menambahkan pertanyaan pada *database*, maka aplikasi akan *me-record* pertanyaan dan jawaban yang dimasukkan oleh *user*. Aplikasi akan *meng-update* jawaban dari pertanyaan jika pertanyaan sudah ada pada *database*.

Pada tipe input selanjutnya yaitu menghapus pertanyaan pada *database*, aplikasi akan mengirimkan pesan berhasil dihapus jika pertanyaan terdapat pada *database* jika pertanyaan terdapat pada *database* dan mengirimkan pesan tidak berhasil jika pertanyaan tidak ada pada *database*.

Pada tipe input selanjutnya yaitu pertanyaan acak yang *di-input user*, aplikasi akan menjawab pertanyaan berdasarkan *record* pertanyaan dan jawaban yang telah disimpan pada *database*. Aplikasi akan mengirimkan pesan berupa pertanyaan yang mirip dengan *input user* jika tidak terdapat pertanyaan yang *match* atau menjawab pertanyaan acak *user* jika terdapat pertanyaan yang *match* pada *database*.

BAB V

KESIMPULAN, SARAN, DAN REFLEKSI

Dari Tugas Besar III IF2211 Strategi Algoritma Semester II 2022/2023 berjudul *Penerapan String Matching dan Regular Expression dalam Pembuatan ChatGPT Sederhana*, kami mendapati bahwa untuk membuat suatu chatbot yang sederhana, kita dapat memanfaatkan algoritma string matching berupa Knuth-Morris-Pratt(KMP) dan Boyer-Moore (BMP) beserta regular expression(regex).

Tugas besar ini memaparkan dan mengenalkan kami secara lanjut terhadap bahasa pemrograman lain seperti Go dan Javascript beserta kaskas untuk pengembangan aplikasi berbasis web. Selain itu, kami juga lebih memahami dan terbantu dengan penerapan algoritma *string matching* beserta pengembangan perangkat lunak berbasis web.

Algoritma KMP dan BM yang kami terapkan telah berhasil mengidentifikasi pola-pola pada *string* pertanyaan yang diajukan. Selain itu, pembuatan regular expression juga membantu kami untuk lebih detail mengenali pola-pola untuk mngekseskusi operasi matematika dan operasi tanya-jawab yang berbasiskan pada sistem basis data.

Akhir kata, kami menyadari tugas besar yang kami masih jauh dari kata sempurna. Maka dari itu, kami masih perlu banyak belajar, baik dari sisi algoritma yang kami buat untuk lebih adaptif, maupun dari sisi rekayasa perangkat lunak yang memanfaatkan bahasa-bahasa dan paradigma baru. Tak lupa, kami juga berterima kasih terhadap Bapak Rinaldi Munir dan Ibu Nur Ulfa Maulidevi sebagai dosen pengampu kami yang memberikan kami kuliah mengenai Strategi Algoritma beserta contoh penerapannya.

DAFTAR REFERENSI

[Munir, Rinaldi \(2021\). Pencocokan String \(String/Pattern Matching\). Diakses pada tanggal 30 April 2023.](#)

[Munir, Rinaldi \(2021\). String Matching dengan Regular Expression. Diakses pada tanggal 1 Mei 2023.](#)

LAMPIRAN

Tautan repository tugas besar:

https://github.com/jejejery/Tubes3_13521131

Tautan Video:

<https://youtu.be/P0ocY3Xpk8E>