CHAT GPT SEDERHANA

LAPORAN TUGAS BESAR

Diajukan sebagai salah satu tugas mata kuliah IF2211 Strategi Algoritma pada Semester II Tahun Akademik 2022/2023

oleh

Henry Anand Septian Radityo	**13521004**
Matthew Mahendra	**13521007**
Ahmad Nadil	**13521024**

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2023

DAFTAR ISI

BAB I
DESKRIPSI TUGAS

I.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.

BAB II
LANDASAN TEORI

II.I Algoritma Knuth-Morris-Pratt

Algoritma Knuth-Morris-Pratt (KMP) merupakan algoritma pencocokan string. Pencocokan string yang dilakukan oleh algoritma ini adalah pencocokan eksak, artinya tidak didesain untuk menemukan kemiripan. Algoritma KMP dikemukakan oleh Donald Knuth, James Morris, dan Vaughan Pratt pada tahun 1977. Algoritma ini melakukan pencarian suatu pola dalam suatu teks, yang bergerak dari kiri ke kanan.

Algoritma KMP pada dasarnya menyerupai algoritma pencocokan string dengan metode *brute force*, akan tetapi pergeseran yang dilakukan oleh algoritma ini lebih cerdas dibandingkan dengan *brute force*. Pergeseran jika ditemukan ketidakcocokan pada karakter ke-j dilakukan sebesar jumlah terpanjang dari prefix pola P[0- j-1] yang juga merupakan suffix polaP[1-j-1]. Penentuan jumlah pergeseran tersebut ditentukan melalui fungsi kegagalan atau *border function* KMP. Hasil dari *border function* KMP akan mengeluarkan jumlah pergeseran *b(k)* yang diperlukan jika kegagalan berada pada indeks ke-*j* pada pola masukan, dengan k = j-1.

Kompleksitas dari algoritma KMP dapat dilihat dari kompleksitas *border function* dan proses pencariannya. Untuk *border function*, kompleksitas algoritmanya adalah O(m) dan untuk pencarian string adalah O(n). Dengan demikian kompleksitasnya menjadi O(m+n).

II.II Algoritma Boyer-Moore

Algoritma Boyer–Moore (BM) merupakan algoritma pencocokan string yang lainnya. Pencocokan string dilakukan berdasarkan *looking glass technique* yang artinya pencocokan pada pattern dilakukan dari belakang (dari indeks paling besar) –walaupun pencocokan terhadap teks tetap dilakukan dari depan– dan berdasarkan *character jumping* yaitu metode pergeseran karakter pada pola.

Algoritma BM memanfaatkan tabel *last occurrence*, yang merupakan tabel berisi informasi terakhir kemunculan karakter-karakter yang terdapat pada teks pada pola yang akan dicari. Informasi dari tabel ini akan digunakan untuk melakukan pergeseran pada algoritma.

Pencarian string pada dasarnya dilakukan dengan memeriksa apakah indeks yang sedang diperiksa pada teks sama atau tidak dengan pattern. Jika sama maka dilakukan penelusuran ke

indeks yang lebih kecil pada pattern. Sebaliknya, melakukan pergeseran untuk karakter di kesalahan di teks pada pola, agar disesuaikan dengan lokasi kesalahan tersebut. Jika lokasi karakter pada teks lebih kecil dari lokasi saat ini, maka dilakukan pergeseran sebanyak satu. Jika pada teks ada karakter yang tidak muncul pada pola, maka dilakukan pergeseran secara menyeluruh pola tersebut. Kompleksitas algoritma ini adalah O(mn).

II.III Regular Expression

Pencocokan string menggunakan *regular expression* pada dasarnya adalah melakukan pencarian pola pada teks berdasarkan regular expression yang dibentuk. Pencarian pola dapat digunakan untuk melakukan ekstraksi informasi atau melakukan validasi teks. Notasi umum dari *regular expression* adalah sebagai berikut,

Notasi	**Arti**
.	Semua karakter kecuali *newline*
^	Mencocokkan awal string
$	Mencocokkan akhir string
\d, \w, \s	Digit, karakter [A-Za-z0-9], spasi
\D, \W, \S	Kecuali digit, karakter, dan spasi
[abc]	a atau b atau c
[a-z]	Dari karakter a sampai z
\., \\, *	Karakter titik, backslash, dan bintang

aa | bb	aa atau bb
*	Pengulangan karakter sebelumnya sebanyak 0 atau lebih kali
+	Mencocokkan karakter sebelumnya setidaknya satu kali atau lebih.
?	Mencocokkan karakter sebelumnya setidaknya satu kali atau tidak sama sekali

Dalam bahasa pemrograman, dari sebuah pola *regular expression* yang diberikan, dapat diambil subteks dari karakter yang digunakan melalui *built-in function* yang telah diberikan oleh suatu bahasa pemrograman.

II.IV Penjelasan Singkat Situs yang Dibangun

Situs yang dibangun, dibuat menggunakan bahasa pemrograman Go sebagai bahasa pemrosesan *backend* termasuk algoritma KMP, BM, dan regular expression dan menggunakan React Typescript sebagai bahasa pemrograman *frontend* dengan kerangka desain menggunakan Tailwind. Basis data yang dikembangkan menggunakan DBMS PostgreSQL berikut informasi yang disimpan adalah informasi pertanyaan dan jawaban, informasi *chat* masukan dan jawaban, dan informasi perbincangan (*bubble chat*).

Situs memiliki beberapa halaman utama seperti login/register page, home page, conversation page. Login page bersamaan dengan register merupakan suatu fitur yang digunakan untuk melakukan pendaftaran atau masuk ke dalam suatu akun. Fitur ini akan memanfaatkan tabel dari user di dalam database. Setelah melakukan login, maka pengguna akan diarahkan ke

halaman home dan dapat membuka suatu chat untuk menuju ke halaman conversation. Pada kedua halaman tersebut, tersedia sidebar dan juga history untuk mendukung fitur dari chat bot.

Pada halaman conversation, pengguna dapat memasukkan suatu pesan dan juga memilih metode pencarian dari string matching yang digunakan. Metode pencarian akan bekerja pada bagian backend yang telah dibuat dalam bahasa go. Pencarian ini akan melibatkan database di tabel chat dan question. Tabel chat digunakan untuk menyimpan percakapan sedangkan question digunakan untuk mendapatkan data mengenai pertanyaan dan jawaban. Selain itu pengguna juga dapat menambahkan, memperbaharui, dan menghapus pertanyaan.

BAB III
ANALISIS PEMECAHAN MASALAH

III.I Langkah Penyelesaian Fitur ChatBot

ChatBot dibagi menjadi fitur yang dapat menerima suatu perintah lalu melakukan perintah yang bersesuaian seperti melakukan kalkulasi operasi matematika atau mencari nama hari pada suatu tanggal tertentu. Selanjutnya, ChatBot juga dapat melakukan tanya jawab berdasarkan pertanyaan yang telah disimpan pada basis data. Matriks penggunaan algoritma dan fitur berikut keterangan penggunaannya adalah sebagai berikut,

Algoritma / Fungsi	Kegunaan	Keterangan
Regular Expression	Fitur Command	Regular expression akan memeriksa apakah ada format dari command-command yang didefinisikan dalam masukan user. Jika ada, lakukan sesuai dengan definisi command tersebut.
KMP	Fitur Tanya Jawab	Jika Regular Expression tidak ditemukan, asumsikan bahwa pengguna meminta jawaban dari basis data. Lakukan string matching dengan pola yaitu masukan dan teks dari basis data. Jika ditemukan, keluarkan jawaban yang bersesuaian. Penggunaan algoritma KMP atau BM didefinisikan pengguna.
BM		
Fungsi Levenshtein Distance		Jika Algoritma KMP atau BM gagal, maka carilah dari basis data pertanyaan yang memiliki kemiripan >=90% dengan masukan. Jika ada, keluarkan jawaban yang bersesuaian. Jika tidak ada,

		keluarkan tiga pertanyaan dengan kemiripan tertinggi.

III.II Fitur Fungsional dan Arsitektur Situs

III.II.I Fitur Memasukkan Login

Untuk melakukan segregasi/pemisahan terhadap chat-chat yang dibuat, maka dibuat halaman Login untuk membagi menjadi pengguna menjadi beberapa akun.

III.II.II Fitur Menambah Chat

Layaknya sebuah percakapan baru, dapat dibuat percakapan baru yang memungkinkan untuk dibuat beberapa percakapan. Dengan demikian tidak akan terjadi tumpang tindih data dalam visibilitas masing-masing percakapan, tidak disimpan dalam satu percakapan yang sama. Tiap juga akan menyimpan pertanyaan dan jawaban yang telah dihasilkan sebelumnya.

III.II.III. Fitur Command

Memasukkan masukan perintah seperti menanyakan hari dari tanggal dalam format dd/mm/yyyy, menambah pertanyaan X dengan jawaban Y, menghapus pertanyaan X, dan melakukan operasi matematika.

III.II.IV. Fitur Tanya dan Jawab

Memasukkan pertanyaan yang ada dalam database dan mengeluarkan jawaban yang sesuai. Algoritma pencarian dapat dipilih oleh pengguna melalui toggle button pada laman situs yang telah dibuat.

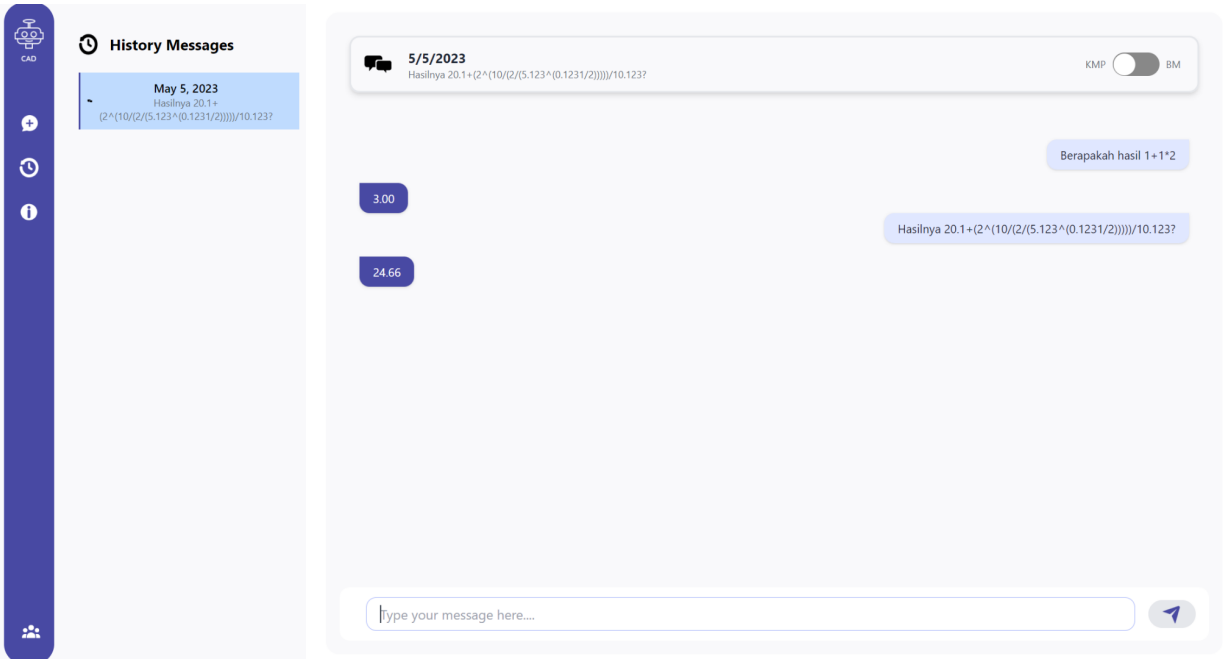
III.II.V. Arsitektur Situs

Situs memiliki beberapa halaman dan berikut merupakan arsitektur dari situs chatbot ChatAkuDong

1. Halaman Login dan Register
2. Halaman Utama
   2.1 Sidebar
   2.2 History
   2.3 Welcome Frame
3. Halaman Conversation
   2.1 Sidebar
   2.2 History
   2.3 Chat Frame
4. Halaman About Us

2.1 Sidebar
2.2 History
2.3 About Us Frame

5. Halaman Information
2.1 Sidebar
2.2 History
2.3 Information Frame



Gambar 3.2.1 Tampilan Situs ChatBot

BAB IV
IMPLEMENTASI DAN PENGUJIAN

IV.I Spesifikasi Teknis Program

IV.I Struktur Data Program

Struktur data terdiri dari *frontend* dan *backend*. *Frontend* bertanggung jawab terhadap tampilan dari situs berikut layout, input, button, dll. *Backend* bertanggung jawab terhadap pemrosesan masukan berikut input dan mengambil data dari dan ke basis data, pemrosesan pencocokan kata, dll.

Dibuat struktur data pula untuk melakukan query ke dalam database. Model tersebut dapat dilihat sebagai berikut,

```
type User struct {
      IDUser   int    `gorm:"primaryKey"`
      Username string `gorm:"not null"`
      Password string `gorm:"not null"`
}

type Conversation struct {
      IDConversation int       `gorm:"primaryKey"`
      IDUser         int       `gorm:"not null"`
      Topic          string    `gorm:"not null"`
      Chats          []Chat
`gorm:"foreignKey:IDConversation;constraint:OnDelete:CASCADE"`
      Date           time.Time `gorm:"not null"`
}

type Chat struct {
      IDChat         int    `gorm:"type:serial;primaryKey;not null"`
      IDConversation int    `gorm:"not null"`
      IDUser         int    `gorm:"not null"`
      Question       string `gorm:"not null"`
      Answer         string `gorm:"not null"`
      SearchMethod   int    `gorm:"not null"`
}

type Question struct {
      IDQuestion int    `gorm:"primaryKey"`
      Question   string `gorm:"not null"`
      Answer     string `gorm:"not null"`
}
```

IV.II Fungsi

Fungsi yang disimpan dalam *package* algo dan dibuat dalam bahasa pemrograman Go. Fungsi yang dibuat antara lain adalah, algoritma KMP, BM, Levenshtein Distance Ratio, Regular Expression, Tanggal, dan Kalkulator.

Berikut *snippet* dari masing-masing fungsi tersebut

```
/* Algoritma KMP */
package Algo

func border(pattern string) []int {
	b := make([]int, len(pattern))
	b[0] = 0

	var m int = len(pattern)
	var j, i int = 0, 1

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

	return b
}

func KMP(pattern string, text string) int {
	var n int = len(text)
	var m int = len(pattern)

	var b []int = border(pattern)

	var i, j int = 0, 0

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

```
            } else {
                  i++
            }
      }
      return -1 // no match
}
```

```
/* Algoritma Boyer-Moore */
package Algo

import "math"

/**
 * Tabel Last Occurence dari ASCII (128 karakter)
 */
func lastOccurence(text string) [128]int {
      var last [128]int

      for i := 0; i < 128; i++ {
            last[i] = -1
      }

      for i := 0; i < len(text); i++ {
            last[text[i]] = i
      }

      return last
}

func BMMatch(pattern string, text string) int {
      var last [128]int = lastOccurence(text)
      n := len(text)
      m := len(pattern)
      i := m - 1
      if i > n-1 {
            return -1
      } else {
            j := m - 1
            for {
                  if pattern[j] == text[i] {
                        if j == 0 {
                              return i
                        } else {
                              i--
                              j--
                        }
                  } else {
                        var lo int = last[text[i]]
                        i = i + m - int(math.Min(float64(j), float64(1+lo)))
                        j = m - 1
                  }
```

```
                    if i > n-1 {
                          break
                    }
              }
        }

        return -1
}
```

```
/* Algoritma Levenshtein */
package Algo

func MatchRatio(s1, s2 string) float64 {
    if len(s1) > len(s2) {
        s1, s2 = s2, s1
    }
    len1, len2 := len(s1), len(s2)
    if len2 == 0 {
        return 1.0
    }
    row1 := make([]int, len2+1)
    for i := 0; i <= len2; i++ {
        row1[i] = i
    }
    for i := 0; i < len1; i++ {
        row2 := make([]int, len2+1)
        row2[0] = i + 1
        for j := 0; j < len2; j++ {
            cost := 1
            if s1[i] == s2[j] {
                cost = 0
            }
            row2[j+1] = min(row2[j]+1, row1[j+1]+1, row1[j]+cost)
        }
        row1 = row2
    }
    return 1.0 - float64(row1[len2])/float64(max(len1, len2))
}

func min(a, b, c int) int {
    if a < b {
        if a < c {
            return a
        }
        return c
    }
    if b < c {
        return b
    }
    return c
}

func max(a, b int) int {
```

```
    if a > b {
        return a
    }
    return b
}
```

```
/* Algoritma Day */
/* Menggunakan Algoritma Zeller untuk penentuan hari berdasarkan kalender
gregorian */
package Algo

import (
    "strconv"
)

func Day(dateStr string) int{
    /* Input dateStr dalam bentuk dd/mm/yyyy */
    day := dateStr[:2]
    d, _ := strconv.Atoi(day)
    month := dateStr[3:5]
    m, _ := strconv.Atoi(month)
    year := dateStr[6:]
    y, _ := strconv.Atoi(year)

    return getDay(y, m, d)
}

func getDay(year, month, day int) int {
    t := []int{0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4}
    if month < 3 {
        year -= 1
    }
    return (year + year/4 - year/100 + year/400 + t[month-1] + day) % 7
}
```

```
/* Algoritma Kalkulator String */
package Algo

import (
    // "strconv"
      "strings"
    "unicode"
    "math"
)

func applyOperation(a, b float64, op rune) float64 {
      /*
            I.S. Menerima 2 buah bilangan real dan 1 buah operator
            F.S. Mengembalikan hasil operasi dari kedua bilangan tersebut
      */
    switch op {
```

```
    case '+':
        return a + b
    case '-':
        return a - b
    case '*':
        return a * b
    case '/':
        return a / b
    case '^':
        return math.Pow(a, b)
    }
    return 0
}

func Calculate(expression string) (float64, error) {
      // Inisialisai variabel
      var numStack []float64
      var opStack []rune

      // Inisialisasi map untuk menentukan prioritas operator (precedence)
      precedence := map[rune]int{
            '(': 0,
            '+': 1,
            '-': 1,
            '*': 2,
            '/': 2,
            '^': 3,
      }

      // Loop sepajang ekspresi
      for len(expression) > 0 {
            // Ambil token pertama dari ekspresi dan hapus token tersebut
dari ekspresi
            token := expression[0]
            expression = expression[1:]

            // Percabangan untuk menentukan token
            // Operasi di dalam kurung akan dijalankan terlebih dahulu
            if token == '(' {
                  // Menambahkan '(' ke stack
                  opStack = append(opStack, rune(token))

            } else if token == ')' {
                  for opStack[len(opStack)-1] != '(' {
                        // Mengambil 2 buah angka dan 1 buah operator dari
stack
                        num2 := numStack[len(numStack)-1]
                        numStack = numStack[:len(numStack)-1]
                        num1 := numStack[len(numStack)-1]
                        numStack = numStack[:len(numStack)-1]
                        op := opStack[len(opStack)-1]
                        opStack = opStack[:len(opStack)-1]
                        numStack = append(numStack, applyOperation(num1,
num2, op))
```

```
                  }

                  // Menghapus '(' dari stack
                  opStack = opStack[:len(opStack)-1]

            } else if strings.ContainsRune("+-*/^", rune(token)) { // Jika
token merupakan operator
                  // Operasi akan dijalankan jika operator pada stack
memiliki precedence yang lebih besar atau sama dengan operator token
                  for len(opStack) > 0 && precedence[opStack[len(opStack)-1]]
>= precedence[rune(token)] {
                        // Mengambil 2 buah angka dan 1 buah operator dari
stack
                        num2 := numStack[len(numStack)-1]
                        numStack = numStack[:len(numStack)-1]
                        num1 := numStack[len(numStack)-1]
                        numStack = numStack[:len(numStack)-1]
                        op := opStack[len(opStack)-1]
                        opStack = opStack[:len(opStack)-1]

                        // Menghitung hasil operasi dan memasukkan hasil
operasi ke stack angka
                        numStack = append(numStack, applyOperation(num1,
num2, op))
                  }
                  opStack = append(opStack, rune(token)) // Memasukkan
operator token ke stack operator

            } else if unicode.IsDigit(rune(token)) { // Jika token merupakan
angka akan di push ke stack angka
                  num := int(token - '0') // Konversi dari rune ke int

                  // Loop untuk menghandle angka dengan lebih dari 1 digit
                  for len(expression) > 0 &&
unicode.IsDigit(rune(expression[0])) {
                        num = num*10 + int(expression[0]-'0')
                        expression = expression[1:]
                  }

                  // Push angka ke stack angka
                  numStack = append(numStack, float64(num))
            }
      }

      // Loop untuk menghitung hasil operasi yang tersisa di stack
      for len(opStack) > 0 {
            // Mengambil 2 buah angka dan 1 buah operator dari stack
            num2 := numStack[len(numStack)-1]
            numStack = numStack[:len(numStack)-1]
            num1 := numStack[len(numStack)-1]
            numStack = numStack[:len(numStack)-1]
            op := opStack[len(opStack)-1]
            opStack = opStack[:len(opStack)-1]
```

```
            // Menghitung hasil operasi dan memasukkan hasil operasi ke stack
angka
            numStack = append(numStack, applyOperation(num1, num2, op))
      }

      return numStack[0], nil
}
```

```
func Regex(text string, questions []model.Question, newQuestion
*model.Question, searchMethod int) (string, int) {
      /* retcode :
      -1    : regex gagal
      1     : regex berhasil
      2     : tambah pertanyaan
      3     : hapus pertanyaan
      */

      /* Periksa apakah terdapat format tanggal dalam teks */
      regex :=
regexp.MustCompile(`([0][1-9]|[1-2][0-9]|[3][0-1])\/(0[1-9]|1[0-2])\/[0-9]{4}`
)
      return_bool, _ :=
regexp.MatchString(`([0][1-9]|[1-2][0-9]|[3][0-1])\/(0[1-9]|1[0-2])\/[0-9]{4}`
, text)
      if return_bool {
            matches := regex.FindAllString(text, -1)
            d := Day(matches[0])
            switch d {
            case 0:
                  return ("Hari Minggu"), 1
            case 1:
                  return ("Hari Senin"), 1
            case 2:
                  return ("Hari Selasa"), 1
            case 3:
                  return ("Hari Rabu"), 1
            case 4:
                  return ("Hari Kamis"), 1
            case 5:
                  return ("Hari Jumat"), 1
            case 6:
                  return ("Hari Sabtu"), 1
            }
            return "day not found", 1
      }

      /* Periksa apakah ekspresi matematika yang valid terkandung dalam teks*/
      regex = regexp.MustCompile(`(?:\s|^)([\-\+\*\^\/\(\)\s?0-9.?]+)`)
      return_bool, _ =
regexp.MatchString(`(?:\s|^)([\-\+\*\^\/\(\)\s?0-9.?]+)`, text)
      if return_bool {
            matches := regex.FindAllString(text, -1)
            res, _ := Calculate(matches[0])
```

```
            return strconv.FormatFloat(res, 'f', 2, 64), 1
      }

      /* Periksa apakah terkandung kata Hapus Pertanyaan xxx */
      regex = regexp.MustCompile(`(?i)^Hapus Pertanyaan\s+(.*)$`)
      return_bool, _ =
regexp.MatchString(`[Hh][Aa][Pp][Uu][Ss]\s[Pp][Ee][Rr][Tt][Aa][Nn][Yy][Aa][Aa]
[Nn]\s`, text)
      if return_bool {
            matches := regex.FindStringSubmatch(text)
            newQuestion.Question = matches[1]
            return "", 3
      }

      /* Periksa apakah terkandung kata Tambah Pertanyaan XXX dengan jawaban
YYY */
      regex = regexp.MustCompile(`(?i)^tambah pertanyaan\s+(.*?)\s+dengan
jawaban\s+(.*?)$`)
      matches := regex.FindStringSubmatch(text)
      if len(matches) > 0 {
            // fmt.Println(matches[1])
            // fmt.Println(matches[2])
            newQuestion.Question = matches[1]
            newQuestion.Answer = matches[2]
            return "", 2
      }

      /* Periksa dengan KMP atau BM */
      for _, question := range questions {
            // make text and question to lowercase
            if searchMethod == 1 {
                  if KMP(strings.ToLower(text),
strings.ToLower(question.Question)) != -1 {
                        return parseEndLine(string(question.Answer)), 1
                  } else if KMP(strings.ToLower(question.Question),
strings.ToLower(text)) != -1 {
                        return parseEndLine(string(question.Answer)), 1
                  }
            } else {
                  if BMMatch(strings.ToLower(text),
strings.ToLower(question.Question)) != -1 {
                        return parseEndLine(question.Answer), 1
                  }else if BMMatch(strings.ToLower(question.Question),
strings.ToLower(text)) != -1 {
                        return parseEndLine(question.Answer), 1
                  }
            }
      }

      /* Jika tidak ada yang exact match, coba levenstein */
      ans := ""
      num := 1

      /* sort questions berdasarkan match ratio-nya */
```

```
	sortedQuestions := sortByMatchRatio(questions, text)

	/* masukkan top 3 questions yang memiliki match ratio < 0.9 */
	for i, question := range sortedQuestions {
		if i >= 3 || MatchRatio(strings.ToLower(text),
strings.ToLower(question.Question)) >= 0.9 {
			break
		}
		ans += strconv.Itoa(num) + ". " + question.Question + " (" +
strconv.FormatFloat(MatchRatio(strings.ToLower(text),
strings.ToLower(question.Question))*100, 'f', 1, 64) + "%)\n"
		num++
	}

	/* jika ada yang match ratio-nya >= 0.9, return jawaban pertama */
	if num == 1 {
		return sortedQuestions[0].Answer, 1
	}

	/* jika tidak, return pertanyaan yang memiliki match ratio tertinggi */
	if ans != "" {
		return "Mungkin pertanyaan yang anda maksud adalah :\n" + ans, 1
	}

	return "Error", -1
}
```

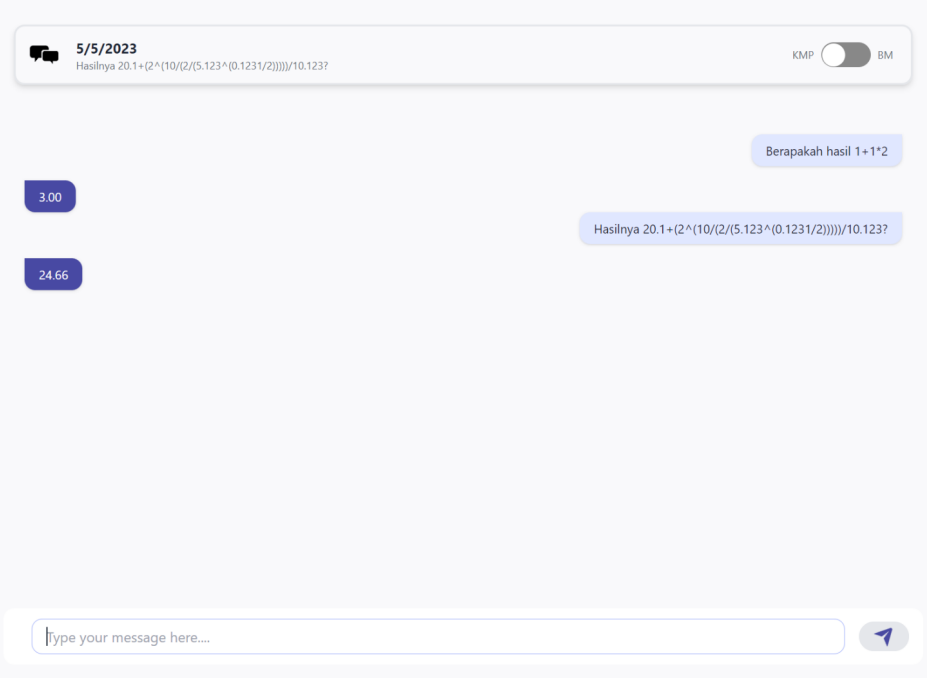
IV.II Cara Penggunaan Program

Penggunaan umum program adalah sebagai berikut,

1. Login atau lakukan registrasi terlebih dahulu
2. Buatlah percakapan baru atau melanjutkan percakapan yang sebelumnya
3. Pada masukan percakapan, berikut adalah beberapa command yang dapat digunakan
    a. Untuk menambahkan pertanyaan: “Tambah pertanyaan <XXX> dengan jawaban <YYY>” dengan <XXX> pertanyaan dan <YYY> jawaban
    b. Untuk menghapus pertanyaan: “Hapus pertanyaan <XXX>” dengan <XXX> pertanyaan yang akan dihapus
    c. Untuk mencari hari pada suatu tanggal: Masuk harus mengandung suatu teks dengan format “dd/mm/yyyy” dengan dd tanggal, mm bulan, yyyy tahun
    d. Untuk melakukan operasi matematika: Masukkan operasi matematika yang valid
    e. Untuk melakukan tanya jawab: Masukkan pertanyaan Anda
        i. Jika ada pertanyaan yang sesuai atau kemiripan mendekati 90%, maka jawaban akan langsung diberikan
        ii. Jika tidak ada, maka dikeluarkan tiga pertanyaan dengan kemiripan tertinggi

IV.III Hasil Pengujian

IV.III.I Pengujian Perintah Perhitungan Operasi Matematika

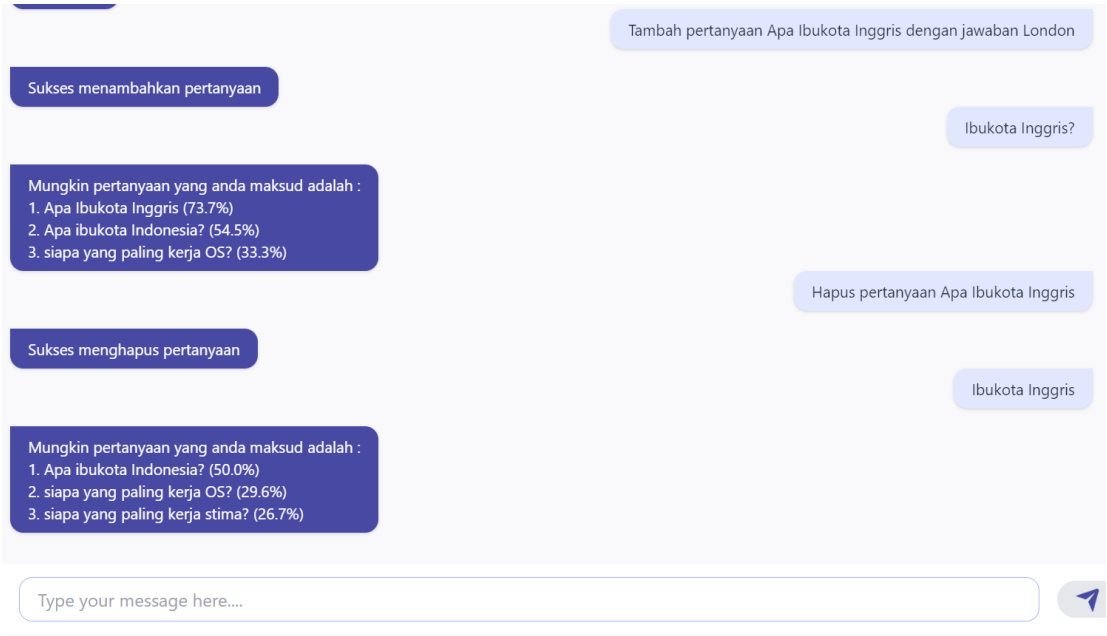


Gambar 4.3.1 Pengujian Perintah Perhitungan Operasi Matematika

IV.III.II Pengujian Penentuan Hari pada Suatu Tanggal

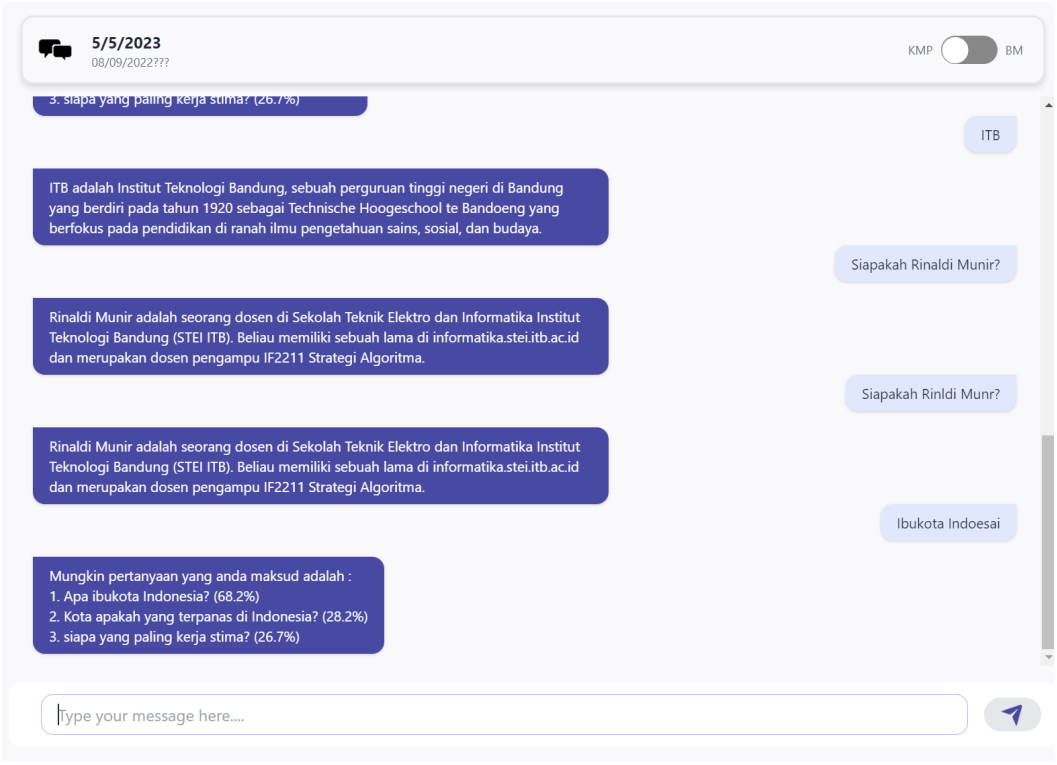
Gambar 4.3.2 Pengujian Penentuan Hari pada Suatu Tanggal

IV.III.III Pengujian Penambahan Pertanyaan dan Penghapusan Pertanyaan



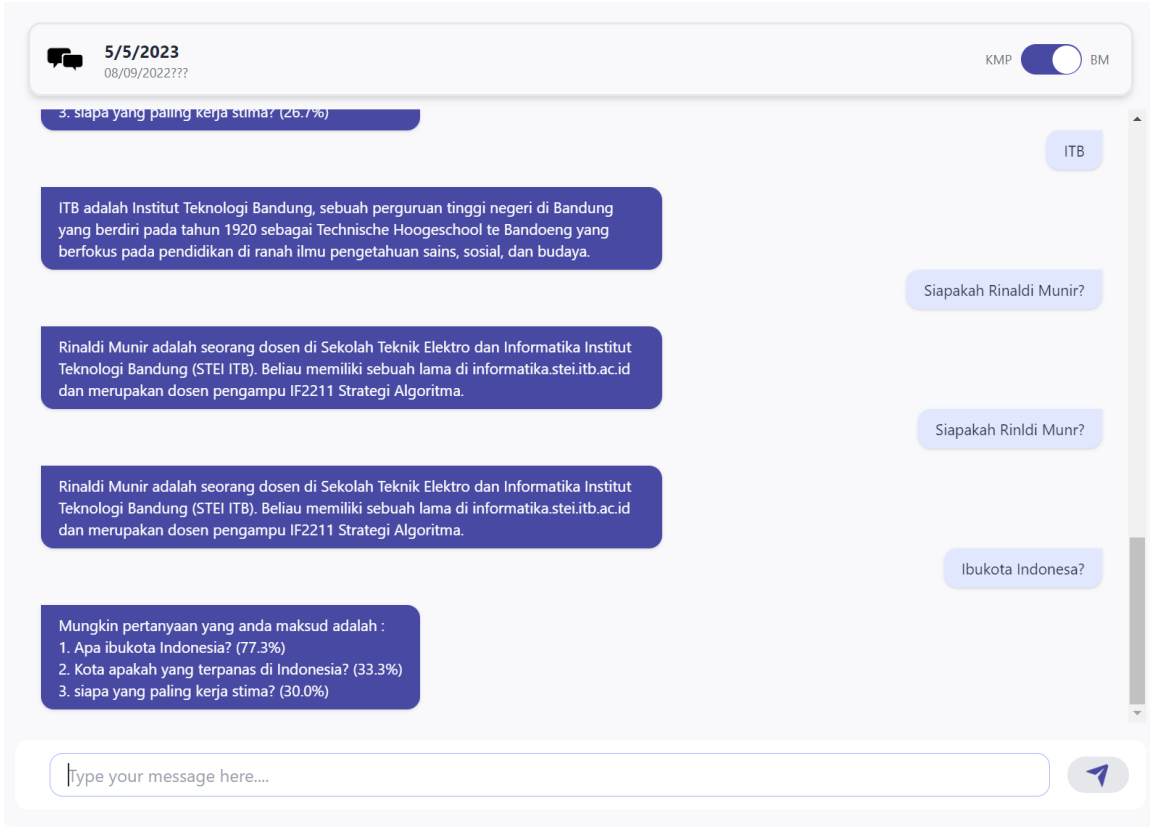
Gambar 4.3.3 Pengujian Penambahan Pertanyaan dan Penghapusan Pertanyaan

IV.III.IV Percobaan QnA dengan Algoritma KMP



Gambar 4.3.4 Percobaan QnA dengan Algoritma KMP

IV.III.V Percobaan QnA dengan Algoritma BM



Gambar 4.3.5 Percobaan QnA dengan Algoritma BM

IV.III.VI Percobaan *Multi Input*

Gambar 4.3.6 Percobaan *Multi Input*

IV.IV Analisis Hasil Pengujian

Dari percobaan yang dilakukan, semua fungsionalitas dari program telah dibuat dengan baik. *Regular Expression* berfungsi dengan baik yaitu mampu menentukan perintah-perintah default seperti pemisahan beberapa pertanyaan *multi input* dengan tanda semikolon, penentuan hari, dan melakukan kalkulasi. Fungsi Levenshtein juga berfungsi dengan baik sehingga dapat menentukan kemiripan antar dua buah fungsi.

Untuk algoritma KMP dan BM, dapat dirasakan bahwa algoritma BM menemukan kesamaan dengan lebih jelas karena pada dasarnya algoritma ini memiliki kompleksitas yang lebih baik daripada algoritma KMP dengan pergeseran berdasarkan tabel *last occurrence* yang dibuat untuk menentukan pergeseran dari pola yang diberikan.

Namun, karena keterbatasan dari *hosting*, maka terasa adanya suatu penundaan antara masukan dan respon dari aplikasi yang dibuat.

BAB V
SIMPULAN

V.I Simpulan

Dari tugas besar ini disimpulkan hal-hal sebagai berikut,

1. Penggunaan algoritma KMP dan BM dapat digunakan untuk melakukan *exact string matching*
2. Penggunaan *regular expression* dapat digunakan untuk melakukan pencarian pola berdasarkan format *regular expression* yang diberikan dan juga untuk melakukan validasi terhadap suatu teks
3. Penggunaan levenshtein distance dapat digunakan untuk menghitung kemiripan antar kedua
4. Pengembangan situs memerlukan bagian *frontend* dan *backend* dengan fungsionalitas masing-masing adalah tampilan dan pemrosesan algoritma berikut penyimpanan komponen-komponen yang diperlukan

V.II Saran

Saran kami untuk pengembangan aplikasi serupa maupun pembuatan adalah sebagai berikut,

1. Sebaiknya menggunakan websocket agar proses *backend* dapat berjalan lebih baik dan lancar
2. Lebih memperhatikan penggunaan useEffect dalam React agar tidak memakan terlalu banyak *memory*
3. Untuk keperluan komersial, sebaiknya penggunaan hosting lebih dimatangkan agar dapat mencakup lebih banyak pengguna dalam satu waktu tertentu

V.III Komentar

- Henry Anand Septian Radityo: Semangat kaka
- Matthew Mahendra: Haduhhhh
- Ahmad Nadil: Stuck skksksksskksksksk

LAMPIRAN

Tautan Repositori Github:

- Frontend: https://github.com/IceTeaXXD/Tubes3_ChatAkuDong_FE
- Backend: https://github.com/IceTeaXXD/Tubes3_ChatAkuDong_BE

BONUS:
Tautan *Deployment* Situs: https://chatakudong.vercel.app/
Tautan Video Youtube: https://youtu.be/OcAE0r-t8uI