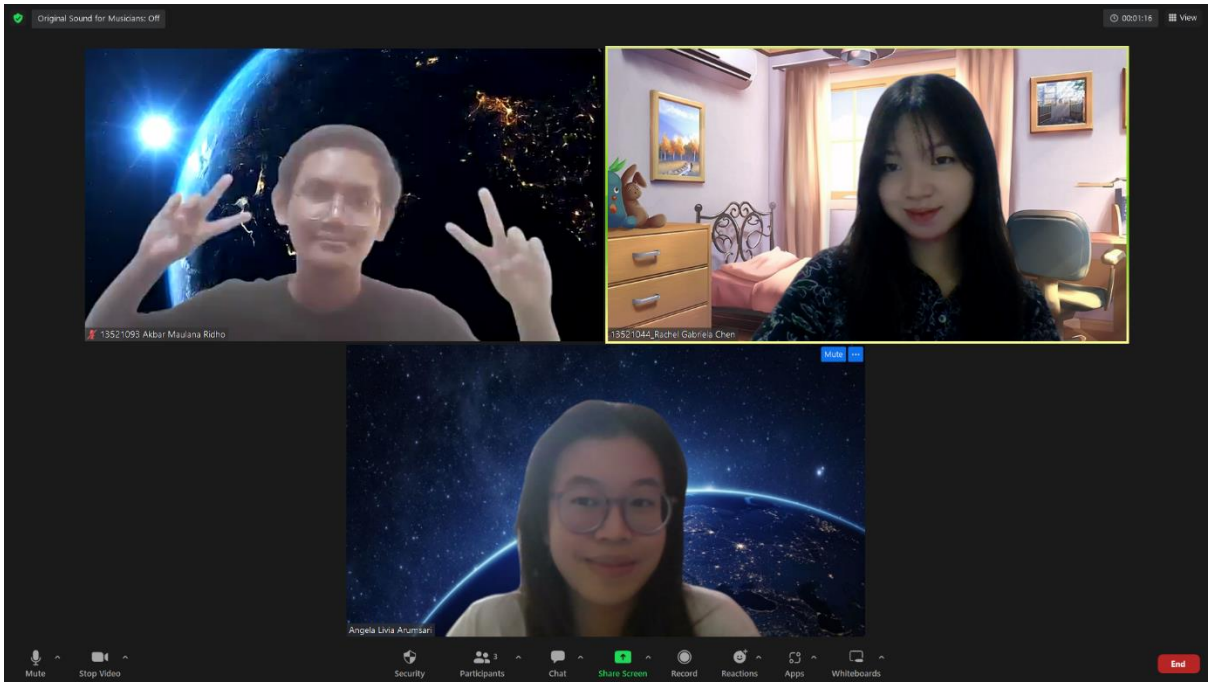
LAPORAN TUGAS BESAR 3
IF2211 STRATEGI ALGORITMA

PENERAPAN STRING MATCHING DAN REGULAR EXPRESSION DALAM PEMBUATAN CHATGPT SEDERHANA



Kelompok bangudahbang

Anggota :

1. **13521044 Rachel Gabriela Chen**
2. **13521093 Akbar Maulana Ridho**
3. **13521094 Angela Livia Arumsari**

PROGRAM STUDI TEKNIK INFORMATIKA

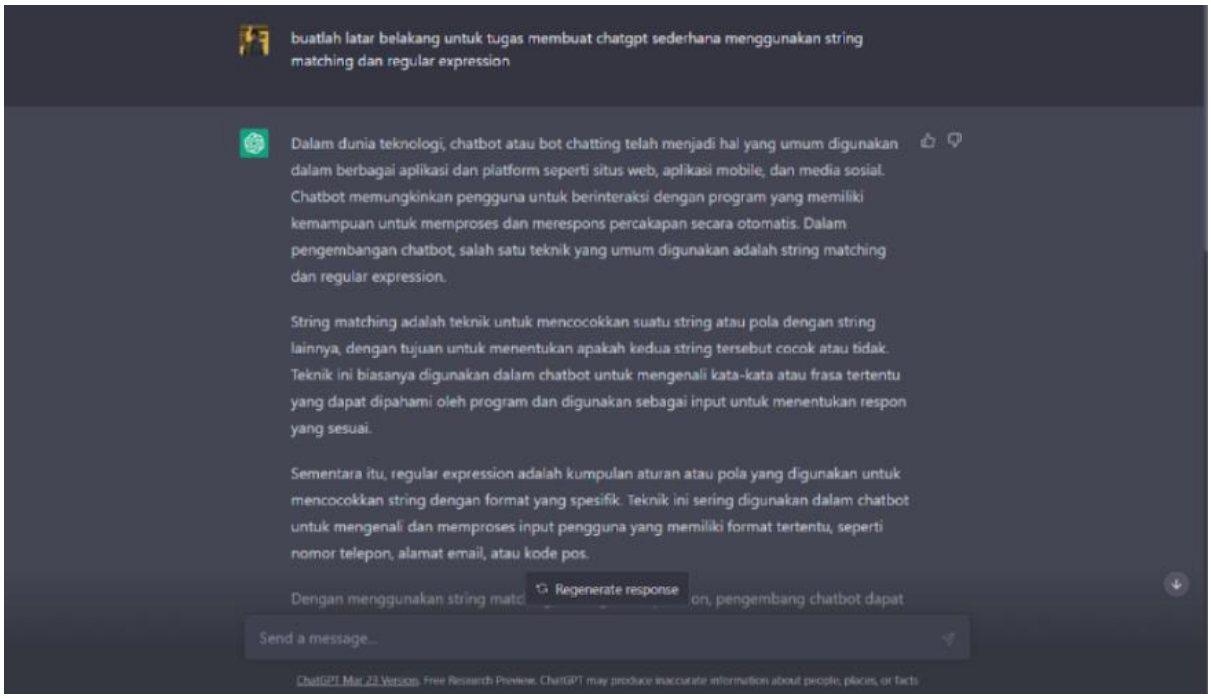
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

BAB I
DESKRIPSI MASALAH

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.



Gambar 1. Ilustrasi Chatbot ChatGPT (funfact latar belakang spek ini dari chatgpt)

Sumber: https://chat.openai.com/chat

Pembangunan chatbot dapat dilakukan dengan menggunakan berbagai pendekatan dari bidang Question Answering (QA). Pendekatan QA yang paling sederhana adalah menyimpan sejumlah pasangan pertanyaan dan jawaban, menentukan pertanyaan yang paling mirip dengan pertanyaan yang diberikan pengguna, dan memberikan jawabannya kepada pengguna. Untuk mencocokkan input pengguna dengan pertanyaan yang disimpan pada database, kalian bisa menggunakan string matching.

String matching adalah teknik untuk mencocokkan suatu string atau pola dengan string lainnya, dengan tujuan untuk menentukan apakah kedua string tersebut cocok atau tidak. Teknik ini biasanya digunakan dalam chatbot untuk mengenali kata-kata atau frasa tertentu yang dapat dipahami oleh program dan digunakan sebagai input untuk menentukan respon

yang sesuai. Sementara itu, regular expression adalah kumpulan aturan atau pola yang digunakan untuk pencocokan string dengan format yang spesifik. Teknik ini sering digunakan dalam chatbot untuk mengenali dan memproses input pengguna yang memiliki format tertentu, seperti nomor telepon, alamat email, atau kode pos.

BAB II
LANDASAN TEORI

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

Algoritma Knuth-Morris-Pratt (KMP) adalah sebuah algoritma pencocokan pola yang digunakan untuk mencari sebuah pola dalam sebuah teks/string. Algoritma ini melakukan pencocokan pola dengan urutan kiri ke kanan seperti algoritma Brute Force, tetapi KMP menggeser pola dengan lebih efisien dibanding algoritma Brute Force.

Langkah algoritma KMP secara umum adalah sebagai berikut:

1. KMP melakukan *pre-process* pola untuk mencari kecocokan antara prefiks dari pola tersebut dengan pola itu sendiri. Hal ini dilakukan dengan menggunakan tabel prefiks dari pola yang akan dicocokkan. Tabel dibuat dengan mencari panjang prefiks dari setiap substring yang cocok dengan suffix dari substring tersebut.
2. Mulai pencocokan pola dari kiri. Misalkan, i = indeks yang sedang dicocokkan = 0.
3. Cocokkan karakter pada posisi i dari teks dengan karakter pada posisi j pada pola.
4. Jika karakter pada teks dan pola sama, lanjutkan pencocokan dengan memeriksa karakter berikutnya pada teks dan pola.
5. Jika karakter pada teks dan pola berbeda, maka gunakan tabel prefix untuk menentukan posisi selanjutnya pada pola yang akan dibandingkan dengan teks. Cari nilai p pada tabel prefix dengan indeks j-1. i diset menjadi p+1.
6. Jika posisi pencocokan pada pola sudah mencapai akhir pola j = panjang pola – 1, maka pola telah ditemukan. Catat posisi i – panjang pola sebagai hasil pencocokan.
7. Ulangi langkah 3-6 untuk setiap karakter pada teks sampai mencapai akhir teks.

2.2 Algoritma Boyer-Moore (BM)

Algoritma Boyer-Moore adalah algoritma lain yang dapat digunakan untuk mencari kecocokan pola dalam teks/string T. Algoritma in memanfaatkan dua teknik, yaitu:

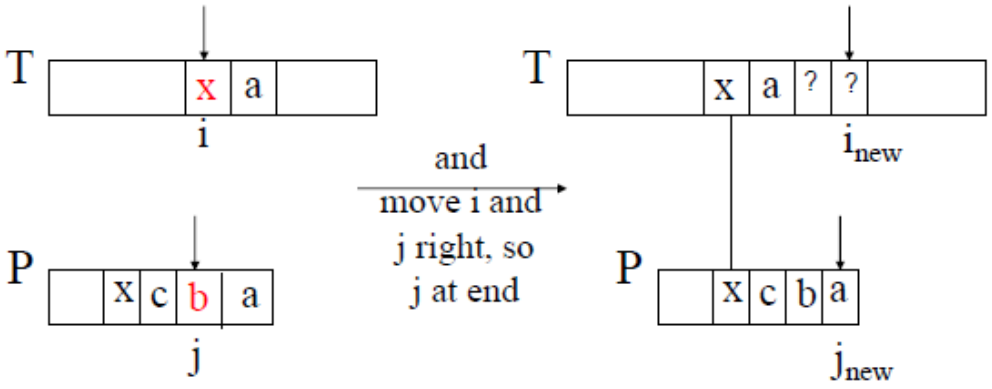
1. Teknik *looking-glass*

   Pencarian dilakukan dengan bergerak mundur (dari kanan ke kiri) pada pola P yang dicocokkan, dimulai dengan akhirnya.
2. Teknik *character-jump*

   Ketika ditemukan karakter x yang tidak cocok pada indeks ke i di T, maka terdapat 3 kemungkinan kasus.

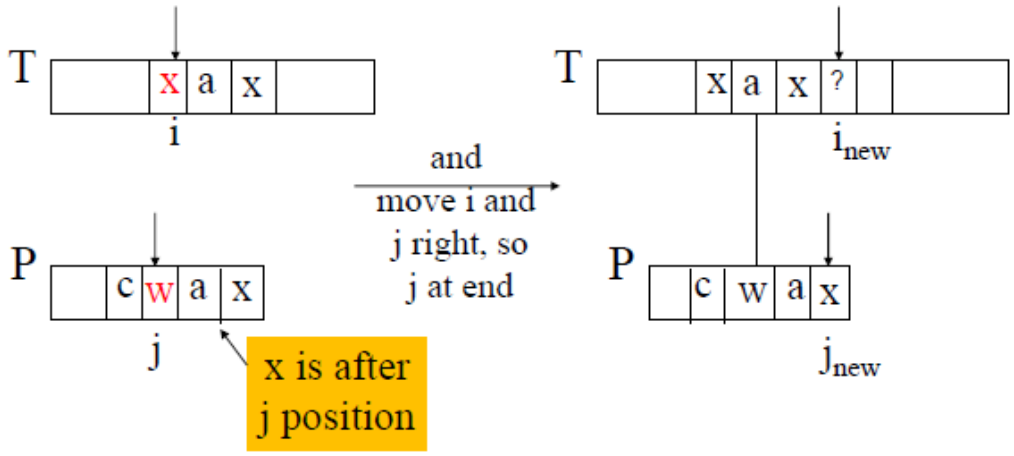
Kasus 1:

Jika pola P memiliki x, maka geser P ke kanan agar x pada P sejajar dengan x pada teks.



Kasus 2:

Jika pola P memiliki x, tetapi P tidak bisa digeser seperti pada kasus 1, maka geser P ke kiri sebanyak 1 karakter.



45

Kasus 3:

Jika x tidak ada dalam P, maka geser P sehingga awal P sejajar dengan karakter selanjutnya pada teks.

2.3 Levenshtein Distance

Levenshtein distance merupakan salah satu metode menghitung kemiripan dari dua string. Levenshtein distance dari dua string adalah jumlah minimal operasi yang dibutuhkan untuk mengubah suatu string ke string yang lain, di mana operasi-operasi tersebut adalah operasi penyisipan, penghapusan, atau penyubstitusian sebuah karakter. Algoritma ini dinamakan berdasarkan Vladimir Levenshtein yang ditemukannya pada tahun 1965. Algoritma levenshtein distance didefinisikan sebagai berikut.

$$\operatorname{lev}(a,b) = \begin{cases} |a| & \text{if } |b| = 0, \\ |b| & \text{if } |a| = 0, \\ \operatorname{lev}\big(\operatorname{tail}(a), \operatorname{tail}(b)\big) & \text{if } a[0] = b[0], \\ 1 + \min \begin{cases} \operatorname{lev}\big(\operatorname{tail}(a), b\big) \\ \operatorname{lev}\big(a, \operatorname{tail}(b)\big) \\ \operatorname{lev}\big(\operatorname{tail}(a), \operatorname{tail}(b)\big) \end{cases} & \text{otherwise} \end{cases}$$

2.4 Regex

Regex atau regular expression adalah sebuah sequence atau pola karakter yang digunakan untuk mencocokkan atau memvalidasi suatu string atau teks dalam pemrograman atau pengolahan teks. Dengan menggunakan regex, kita dapat mencari, mengganti, dan memanipulasi teks dengan sangat efektif.

Regex terdiri dari berbagai karakter khusus yang memiliki arti dan fungsi tertentu dalam pencarian dan pengolahan teks. Beberapa karakter khusus yang sering digunakan dalam regex antara lain:

- Karakter "." yang merepresentasikan satu karakter apa saja.
- Karakter "^" yang merepresentasikan awal baris atau teks.
- Karakter "$" yang merepresentasikan akhir baris atau teks.
- Karakter "+" yang merepresentasikan satu atau lebih dari karakter sebelumnya.
- Karakter "*" yang merepresentasikan nol atau lebih dari karakter sebelumnya.
- Karakter "?" yang merepresentasikan nol atau satu karakter sebelumnya.
- Karakter "[" dan "]" yang merepresentasikan sebuah kelas karakter.
- Karakter "(" dan ")" yang merepresentasikan grup.

Dalam penggunaannya, regex dapat digunakan dalam berbagai bahasa pemrograman seperti Java, Python, C++, dan lainnya. Dalam pemrograman, regex sering digunakan untuk mencari atau memvalidasi input pengguna, melakukan parsing atau analisis teks, dan dalam pengolahan data teks lainnya.

2.5 Web Application CHAT-AR;

Web Appplication CHAT-ARL adalah sebuah *web application* yang dikembangkan dengan Golang (*backend*) dan React.JS (*frontend*). Database yang digunakan adalah PostgreSQL. CHAT-ARL dikembangkan sebagai solusi dari permasalahan yang telah dijelaskan pada Bab 1.

Fitur-fitur aplikasi ini antara lain:

1. **Fitur pernyataan teks**

   Fitur ini menerima pertanyaan dari input pengguna dan mencocokkan dengan pertanyaan yang telah disimpan pada database dengan memanfaatkan algoritma KMP, BM, dan Levenshstein Distance.

2. **Fitur kalkulator**

   Fitur ini menerima input pengguna yang berupa persamaan matematika. Operasi yang didukung adalah operasi bilangan bulat dengan operator +, -, *, /, ^, (,).

3. **Fitur tanggal**

   Fitur ini menerima input pengguna yang berupa tanggal dengan format DD-MM-YYYY dan YYYY-MM-DD dan memberikan hari apa di tanggal tersebut.

4. **Fitur tambah pertanyaan dan jawaban ke database**

   Pengguna dapat menambahkan pertanyaan dan jawaban ke database dengan query "Tambahkan pertanyaan xxx dengan jawaban yyy". Jika pertanyaan sudah terdapat pada database, maka jawaban dari pertanyaan akan di-*update*.

5. **Fitur hapus pertanyaan dari database**

   Pengguna dapat menghapus sebuah pertanyaan dari database dengan query contoh "Hapus pertanyaan xxx".

Klasifikasi dilakukan menggunakan regex dan terklasifikasi layaknya bahasa sehari - hari. Algoritma string matching KMP dan BM digunakan untuk klasifikasi query teks. Tersedia toggle pada website bagi pengguna untuk memilih algoritma KMP atau BM.

BAB III
ANALISIS PEMECAHAN MASALAH

1.1 Langkah Penyelesaian Fitur Pertanyaan Teks

Regular expression digunakan untuk melakukan klasifikasi masukan pengguna, untuk nantinya diproses oleh berbagai fitur. Hasil klasifikasinya adalah masukan perhitungan, masukan tanggal, masukan penambahan query, masukan menghapus query, dan masukan pertanyaan.

Prioritas matchnya adalah sebagai berikut: tanggal, ekspresi matematika, penambahan query, menghapus query, dan pertanyaan. Proses match ekspresi matematika dilakukan dengan regular expression yang mencari match berisi whitespace, angka, dan simbol matematis. Lalu, proses match date dilakukan dengan mencari substring yang memenuhi format DD-MM-YYYY atau YYYY-MM-DD. Proses match penambahan query dilakukan dengan match regex string yang memiliki format Tambahkan Pertanyaan (...) dengan jawaban (...). Proses match menghapus query dilakukan dengan match regex string yang memiliki format Hapus pertanyaan (...). Bila sebuah masukan pengguna tidak masuk ke ketiganya, maka akan diklasifikasikan sebagai pertanyaan.

Pemrosesan fitur pertanyaan teks terdiri atas beberapa tahap. Tahap pertama adalah tahap pencarian exact match dengan algoritma KMP/BM. Bila ditemukan, responsenya akan diberikan kepada pengguna. Bila tidak ada, proses selanjutnya adalah pencarian dengan string distance. Bila terdapat query dengan similarity lebih dari 90%, response query tersebut akan diberikan kepada pengguna. Bila tidak ada, akan diambil tiga query dengan similarity tertinggi untuk dijadikan response kepada pengguna.

1.2 Langkah Penyelesaian Fitur Kalkulator

Fitur kalkulator adalah fitur yang menerima input pengguna berupa *string* yang merepresentasikan ekspresi matematika infix. Fitur ini mengimplementasikan algoritma yang memanfaatkan dua *stack*, satu untuk *operands*(bilangan yang dioperasikan) dan yang satu untuk operator. Langkah-langkah dalam algoritma ini antara lain:

1. Selama masih ada karakter pada *string* yang belum diproses;
    1.1 Proses karakter selanjutnya
    1.2 Jika karakter selanjutnya adalah:

1.2.1 Sebuah angka: simpan angka tersebut dan proses karakter selanjutnya hingga ditemukan karakter yang bukan angka. Setiap angka yang telah dibaca diproses digit-digitnya menjadi sebuah nilai.

1.2.2 Sebuah kurung buka: masukkan ke *stack operator*

1.2.3 Sebuah kurung tutup:

a. Selama elemen teratas dari *stack operator* bukan kurung buka,
   1. *Pop* operator dari *stack* operator
   2. *Pop* nilai pada *stack operands* dua kali
   3. Lakukan kalkulasi kedua nilai tersebut dengan operator yang di-*pop* dengan urutan yang benar.
   4. Hasil kalkulasi dimasukkan ke *stack operands*

b. *Pop* kurung buka dari *stack operator*

1.2.4 Sebuah operator:

a. Selama *stack operator* tidak kosong, dan *top* dari *tack operator* memiliki prioritas yang lebih besar sama dengan *operator* yang sedang diperiksa:
   1. *Pop* operator dari *stack* operator
   2. *Pop* nilai pada *stack operands* dua kali
   3. Lakukan kalkulasi kedua nilai tersebut dengan operator yang di-*pop* dengan urutan yang benar.
   4. Hasil kalkulasi dimasukkan ke *stack operands*

b. Masukkan operator yang sedang diperiksa ke dalam *stack operands*.

2. Selama *stack operator* tidak kosong:
   1. *Pop* operator dari *stack* operator
   2. *Pop* nilai pada *stack operands* dua kali
   3. Lakukan kalkulasi kedua nilai tersebut dengan operator yang di-*pop* dengan urutan yang benar.
   4. Hasil kalkulasi dimasukkan ke *stack operands*
3. Nilai pada *top stack operands* adalah hasil kalkulasi

1.3 Langkah Penyelesaian Fitur Tanggal

Fitur tanggal menerima input *string* yang merepresentasikan tanggal dengan format YYYY-MM-DD dan DD-MM-YYYY. *String* tersebut kemudian di-*parse* sesuai *layout*

YYYY-MM-DD dan DD-MM-YYYY. Setelah *string diparse*, program memanfaatkan *library* untuk mendapatkan hari dari tanggal yang diinput.

1.4 Langkah Penyelesaian Tambah Pertanyaan dan Jawaban ke Database

Fitur tambah pertanyaan menerima dua parameter yaitu *string* pertanyaan dan *string* jawaban. Langkah-langkah penambahan pertanyaan ke *database* yaitu:

1. Dicari pertanyaan tersebut pada *database* dengan memanfaatkan *stringmatcher* dengan algoritma Boyer-Moore/ KMP.
2. Jika pertanyaan ditemukan pada *database*, maka jawaban dari pertanyaan tersebut di-*update* menjadi jawaban yang diberikan oleh pengguna.
3. Jika belum, maka pertanyaan tersebut ditambahkan ke *database*.

1.5 Langkah Penyelesaian Hapus Pertanyaan dari Database

Fitur hapus pertanyaan menerima satu parameter, yaitu string pertanyaan. Langkah-langkah penghapusan pertanyaan tersebut dari database yaitu:

1. Dicari pertanyaan tersaebut pada database dengan memanfaatkan stringmatcher dengan algoritma Boyer-Moore/ KMP
2. Jika pertanyaan ditemukan pada database, maka pertanyaan tersebut akan dihapus dari database.
3. Jika tidak ada, akan menampilkan pesan error.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1 Repositori Github

https://github.com/akbarmridho/Tubes3_bangudahbang/

4.2 Implementasi dalam *Pseudocode*

Berikut adalah *pseudocode* untuk algoritma-algoritma pada bab II dan III yang diimplementasikan:

a. KMP

```
function KMP(patternStr, toMatchStr){
        patternStr = toLower(patternStr)
        toMatchStr = toLower(toMatchStr)
        result = []
        patternLen = len(pattern)
        borders := []
        j, i = 0, 1
        for i < patternLen do
                if pattern[j] = pattern[i] then
                        // j+1 chars match
                        borders[i] ← j + 1
                        i++
                        j++
                else if j > 0 then // j follows matching prefix
                        j ← borders[j-1]
                else  // no match
                        borders[i] ← 0
                        i++

        // begin matching
        i, j = 0, 0

        for i < len(toMatch) do
                if pattern[j] = toMatch[i] then
                        if j = patternLen-1 then
                                result ← append(result, i-patternLen+1) // found match
                        i++
                        j++
                else if j > 0 then
                        j ← borders[j-1]
                else
```

```
                        i++
        return result
```

b. BM

```
procedure KMP(patternStr, toMatchStr){
        patternStr = toLower(patternStr)
        toMatchStr = toLower(toMatchStr)
        result = []
        patternLen = len(pattern)
        borders := []
        j, i = 0, 1
        for i < patternLen do
                if pattern[j] = pattern[i] then
                        // j+1 chars match



function BadCharHeuristic(pattern, patLen) {
        ret = []
        for i ← 0 to 256-1 do
                ret ← append(ret, -1)
        endfor
        for i ← 0 to patLen-1 do
                ret[int(pattern[i])] ← i // updates the last index of all the chars in the
pattern
        endfor
        return ret

function BM(patternStr, toMatchStr) []int {
        // make both the pattern and to match string into lowercase
        patternStr = toLower(patternStr)
        toMatchStr = toLower(toMatchStr)
        patLen = len(patternStr)
        textLen = len(toMatchStr)
        ret = []
        if patLen <= textLen then
                badChar ← BadCharHeuristic(patternStr, patLen) // calculate the bad char
heuristic
                i ← 0
                for i <= textLen-patLen
                        j ← patLen - 1
                        // check from backwords
                        for j >= 0 && patternStr[j] == toMatchStr[i+j] do
                                j--
```

```
                        endfor

                        if j < 0 then // if it is a matching string
                                ret ← append(ret, i)
                                // shift the pattern so the next char is alligned with its
last occurence in the pattern
                                if i+patLen < textLen then
                                        i ← i + patLen - badChar[toMatchStr[i+patLen]]
                                else
                                        i ← i + 1
                                endif
                        else
                                // shift the pattern so that the mismatched character aligns
with the last occurence of it in pattern
                                diff ← j - badChar[toMatchStr[i+j]]
                                if diff > 1 then
                                        i ← i + diff
                                else
                                        i ← i + 1
                                endif
                        endif
                endfor
        endif
        return ret
}
```

c. Levenshtein Distance

```
function LevenshteinDistance(input, toMatch) {
        str1 := toLower(input)
        str2 := toLower(toMatch)
        len1 := len(str1)
        len2 := len(str2)

        matrix = [][]
        for i ← 0 to len1 do
                matrix[i][0] ← i
        endfor
        for j ← 0 to len2 do
                matrix[0][j] = j
        endfor
        for j ← 1 to len2 do
                for i ← 1 to len1 do
                        if str1[i-1] = str2[j-1] then
```

```
                        matrix[i][j] ← matrix[i-1][j-1]
                else
                        minimum ← matrix[i-1][j]
                        if matrix[i][j-1] < minimum then
                                minimum ← matrix[i][j-1]
                        endif
                        if matrix[i-1][j-1] < minimum then
                                minimum ← matrix[i-1][j-1]
                        endif
                        matrix[i][j] ← minimum + 1
                endif
        endfor
    endfor
    return matrix[len1][len2]

// MeasureSimilarity Calculate similarity in percent using Levenshtein Distance
function MeasureSimilarity(input, toMatch) {
    distance ← LevenshteinDistance(input, toMatch)
    maxLen ← len(input)
    if len(toMatch) > maxLen then
        maxLen ← len(toMatch)
    endif
    similarity ← 1 - (float64(distance) / float64(maxLen))
    return similarity
}
```

d. Evaluasi Ekspresi Matematika

```
function Calculate(input)) {
    values = []
    op = []

    for i ← 0 to len(input) - 1 do
        c ← (input[i])
        if c = ' ' then
            continue
        else if c = '(' then
            ops = append(ops, c)
        else if c >= '0' && c <= '9' then
            val ← 0
            j ← i
            for j < len(input) && input[j] >= '0' && input[j] <= '9' do
                val ← val*10 + float64(input[j]-'0')
```

```
                        j++
                endfor
                values ← append(values, val)
                i ← j - 1
        else if c = ')'
                for len(ops) > 0 && ops[len(ops)-1] != '(' do
                        val2 ← PopVal(&values)
                        val1  ← PopVal(&values)
                        op ← PopOp(&ops)
                        if val2 = 0 && op = '/' then
                                return   "Undefined.   Can't   divide   with   0",
errors.New("invalid expression")
                        endif
                        values ←append(values, ApplyOp(val1, val2, op))
                endfor
                if len(ops) > 0 then
                        PopOp(&ops)
                endif
        else
                for len(ops) > 0 && Precedence(ops[len(ops)-1]) >= Precedence(c) do
                        val2 ← PopVal(&values)
                        val1  ← PopVal(&values)
                        op ← PopOp(&ops)
                        if val2 = 0 && op = '/' then
                                return "Undefined. Can't divide with 0"
                        endif
                        values ← append(values, ApplyOp(val1, val2, op))
                endfor
                ops ← append(ops, c)
        endif
    endfor
    for len(ops) > 0 do
        val2 ← PopVal(&values)
        val1  ← PopVal(&values)
        op ← PopOp(&ops)
        if val2 = 0 && op = '/' then
              return  "Undefined.  Can't  divide  with  0",  errors.New("invalid
expression")
        endif
        values ←append(values, ApplyOp(val1, val2, op))
    endfor
    if len(values) > 1 || len(ops) > 1 then
        return "There's a mistake in the math expression entered"
    endif
```

```
        return (values[len(values)-1]
```

e. Hari dari Tanggal

```
function GetDay(input) {
        found ← false
        layoutFormats ← {
                "2006-01-02 15:04:05",
                "2006-1-02 15:04:05",
                "2006-01-2 15:04:05",
                "2006-1-2 15:04:05",
                "02-01-2006 15:04:05",
                "2-01-2006 15:04:05",
                "02-1-2006 15:04:05",
                "2-1-2006 15:04:05",
        }

        for i ← 0 to len(layoutFormats) - 1 && !found do
                value ← input + " 08:04:00"
                parsed ← time.Parse(layoutFormats[i], value)
                if err = nil then
                        date ← parsed
                        found ← true
                endif
        endfor
        if !found then
                return "I'm sorry I can't get the day of the date, the date inputted is
invalid. Make sure it's YYYY-MM-DD or DD-MM-YYYY"
        endif
        return input + " is " + date.Weekday().String()
```

f. Tambah Pertanyaan

```
function AddQuery(question, answer) {
        // find first exact match
        match ← models.Query{
                Response: "",
        }
        for i ← 0 to i < len(queries) - 1 && match.Response “”  do
                query ← queries[i]
                matchIdxs ← BM(question, query.Query)
                if len(matchIdxs) != 0 then
                        match ← query
```

```
                endif
        endfor

        db ← configs.DB.GetConnection()

        lock.Lock()

        // if exists
        if match.Response != "" then
                // update the response
                db.Model(&models.Query{}).Where("id   =   ?",   match.ID).Update("response",
answer)
        else
                // create new query and ans
                newQuery ← &models.Query{
                        Query:    question,
                        Response: answer,
                }
                db.Create(newQuery)
        endif
        isDirty ← true
        lock.Unlock()

        return "Successfully added " + question + " to database with answer " + answer
}
```

g. Hapus Pertanyaan

```
function DeleteQuery(input) {
        // find first exact match
        match ← models.Query{
                Response: "",
        }
        i ← 0
        for i < len(queries) && match.Response = "" do
                query ← queries[i]
                matchIdxs = []
                matchIdxs = KMP(input, query.Query)
                if len(matchIdxs) != 0 then
                        match ← query
                endif
                i++
        endfor
```

```
        if match.Response = "" then
                return "Tidak ada pertanyaan " + input + " pada database!"
        endif

        db ← configs.DB.GetConnection()

        lock.Lock()

        db.Delete(&models.Query{}, match.ID)

        isDirty ← true
        lock.Unlock()
        return "Pertanyaan " + match.Query + " telah dihapus", nil
}
```

h. Regex Pemilihan Fitur

```
onlyMathRegex ← regexp.MustCompile(`[\s\d()+\-*/^]+`)
dateRegex  ←  regexp.MustCompile(`(\d{4})[-  -.](0*[1-9]|1[012])[-  -.](0*[1-9]|[12][0-
9]|3[01])|(0*[1-9]|[12][0-9]|3[01])[- -.](0*[1-9]|1[012])[- -.](\d{4})`)
addQueryRegex ← regexp.MustCompile(`^[Tt]ambahkan pertanyaan (.*) dengan jawaban (.*)$`)
deleteQueryRegex ← regexp.MustCompile(`^[Hh]apus pertanyaan (.*)$`)
        if dateRegex.MatchString(queryRequest.Input) then
                message, err = services.GetDay(queryRequest.Input)
        else if onlyMathRegex.MatchString(queryRequest.Input) then
                message, err = services.Calculate(queryRequest.Input)
        else if addQueryRegex.MatchString(queryRequest.Input) then
                matches := addQueryRegex.FindStringSubmatch(queryRequest.Input)
                message, err = services.AddQuery(matches[1], matches[2])
        else if deleteQueryRegex.MatchString(queryRequest.Input) then
                matches := deleteQueryRegex.FindStringSubmatch(queryRequest.Input)
                message, err = services.DeleteQuery(matches[1])
        else
                message, err = services.MatchQuery(queryRequest.Input, queryRequest.IsKMP)
        endif
```

4.3 Struktur Data Program dan Spesifikasi Program

Web Application CHAT-ARL dikembangkan menurut *tech-stack*:

- Backend – Golang
- Frontend – React.JS (styling dengan Tailwind CSS dan Daisy UI)

- Database – PostgreSQL

Struktur data yang digunakan dalam program terdapat pada models di *backend*. Models ini antara lain:

- history.go

  history.go mendefinisikan tipe *struct* History yang digunakan untuk menyimpan History dari sebuah *session*.

```
type History struct {
    ID        uint      `gorm:"primarykey" json:"id"`
    CreatedAt time.Time `json:"created_at"`
    UserQuery string    `json:"user_query"`
    Response  string    `json:"response"`
    SessionId uuid.UUID `json:"session_id"
gorm:"type:uuid;index"`
}
```

  ID menyatakan ID dari History, CreatedAt waktu dari History tersebut, UserQuery yaitu pertanyaan yang diinput user, Response yaitu jawaban dari pertanyaan, dan SessionId yaitu uuid dari *session* yang bersangkutan.

- query.go

  history.go mendefinisikan tipe *struct* Query yang merupakan pertanyaan dan jawaban yang disimpan di *database*.

```
type Query struct {
    ID        uint      `gorm:"primarykey" json:"id"`
    CreatedAt time.Time `json:"created_at"`
    Query     string    `json:"query"`
    Response  string    `json:"response"`
}
```

  ID menyatakan ID dari Query, CreatedAt waktu dibuatnya Query, Query yaitu pertanyaan, dan Response yaitu jawaban dari pertanyaan.

- response.go

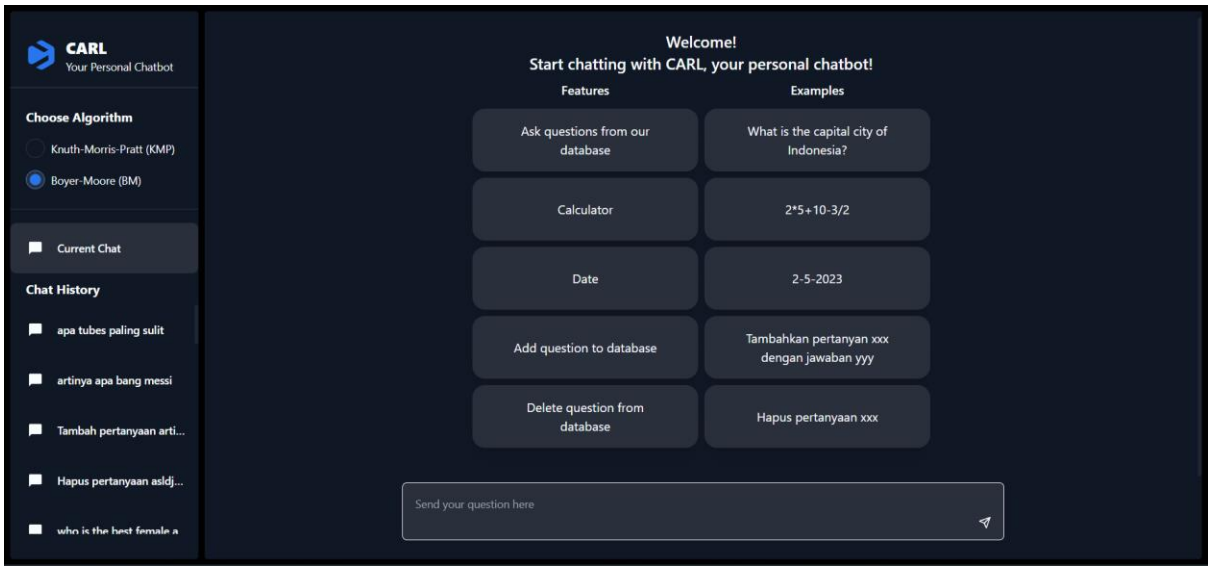
  response.go mendefinisikan tipe *struct* Response yaitu *response* untuk Query yang diinput oleh pengguna.

```
type Response[T interface{}] struct {
    Message string `json:"message,omitempty"`
    Data    T      `json:"data,omitempty"`
}
```

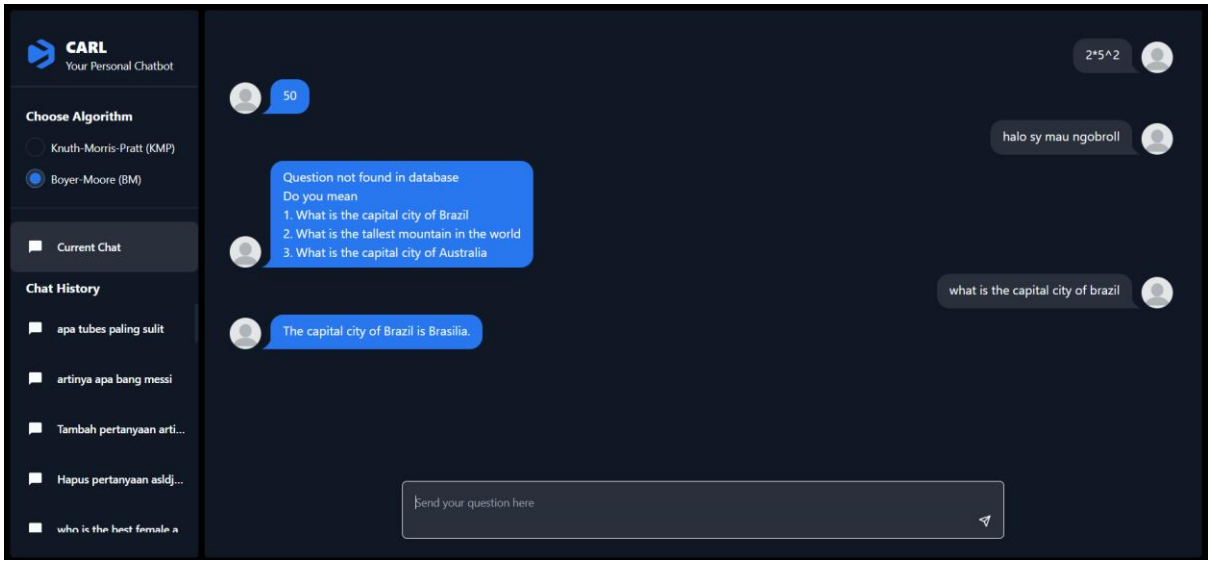
Message yaitu pesan sukses/gagal, sedangkan data memuat jawaban untuk query yang diinput.

4.4 Tata Cara Penggunaan Program

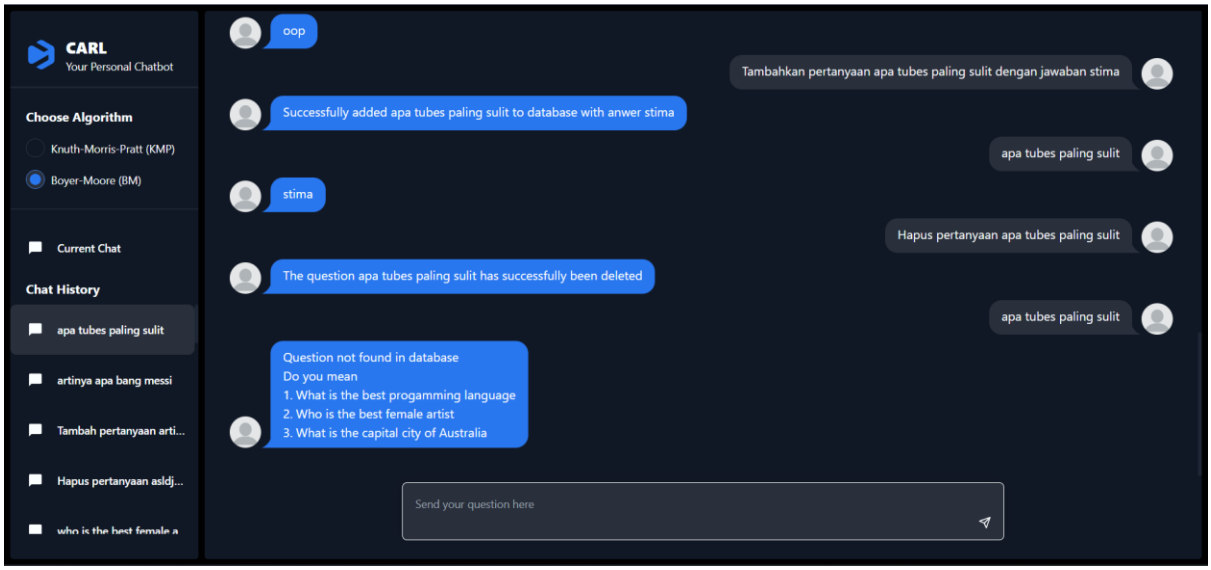
Website dapat diakses melalui link berikut https://tubes3-bangudahbang.vercel.app/. Berikut adalah tampilan awal website.



Ketika awal membuka website, pengguna akan otomatis masuk ke sesi *current chat* yang masih kosong. Pada chat yang masih kosong ini, pengguna dapat melihat panduan fitur yang dimiliki oleh website maupun contoh penggunaannya. Pada kolom yang bertuliskan "Send your question here", pengguna dapat memasukkan pertanyaan dan melakukan submit dengan menekan tombol enter atau klik icon send di pojok kanan bawah. Pada bagian kiri website, pengguna dapat mengatur algoritma pencocokan string yang ingin digunakan, KMP atau BM. Berikut merupakan tampilan website ketika pengguna sudah chat beberapa pertanyaan.



Pada bagian kiri website, terdapat juga kumpulan chat history dari seluruh riwayat chat pengguna. Tiap chat history dapat diklik untuk melihat seluruh pertanyaan dan jawaban yang ada. Selain itu, pengguna juga dapat melanjutkan chat pada chat history yang dipilih. Berikut merupakan tampilan saat pengguna membuka salah satu chat history.



4.5 Analisis dan Pengujian

Pada kasus-kasus exact match, program akan langsung menampilkan jawaban yang sesuai dengan response yang ada pada database. Hal ini berlaku untuk algoritma KMP maupun BM.

Kasus 1 : Pertanyaan *exact match* dengan *query* di Database (KMP)

Kasus 2 : Pertanyaan *exact match* dengan *query* di Database (BM)

Pada fitur kalkulator, program akan melakukan validasi terlebih dahulu terhadap ekspresi matematika yang diberikan. Pada ekspresi yang valid, program akan menjawab hasil dari ekspresi matematika. Pada ekspresi yang tidak valid, program akan memunculkan pesan bahwa ekspresi tidak valid. Sedangkan pada ekspresi matematika seperti 1/0, program akan memunculkan pesan undefined.

Kasus 3 : Kalkulator – Ekspresi Valid

Kasus 4 : Kalkulator – Ekspresi Tidak Valid

Kasus 5 : Kalkulator – Undefined

Pada fitur hari, program dapat menampilkan hari pada suatu tanggal dengan dua buah format, yaitu DD-MM-YYYY dan YYYY-MM-DD.

Kasus 6 : Fitur Hari (format: DD-MM-YYYY)

Kasus 7 : Fitur Hari – Format YYYY-MM-DD

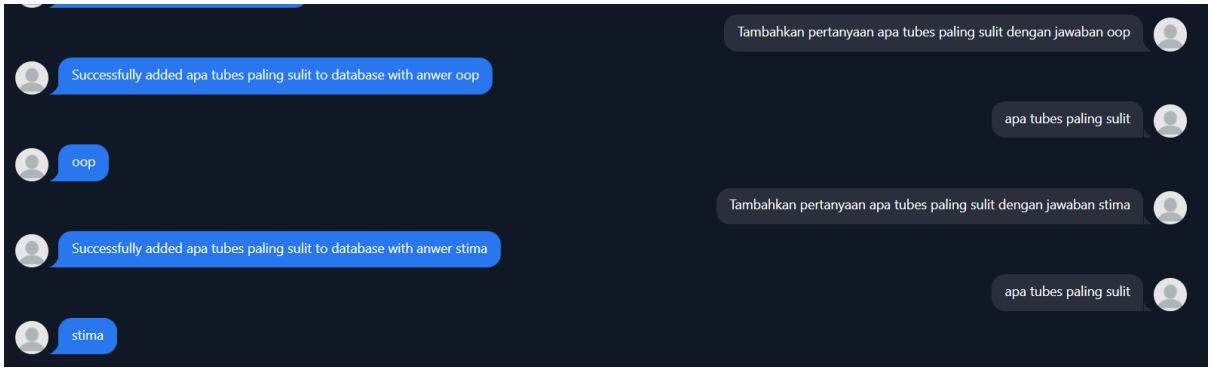
Pada kasus string pertanyaan tidak exact match dengan pertanyaan pada database, program akan tetap memunculkan jawaban jika kemiripan > 90%. Pada saat kemiripan < 90%, program akan menampilkan tiga pertanyaan yang memiliki kemiripan paling tinggi.

Kasus 8 : Tidak *exact match* dengan kemiripan > 90%

Kasus 9 : Tidak ada *match* pada database

Pengguna dapat menambahkan maupun menghapus pertanyaan dari database. Setelah menambahkan pertanyaan, pengguna dapat menanyakan pertanyaan yang ditambahkan tersebut. Jika pengguna menambahkan pertanyaan yang sudah ada di database, maka jawaban pertanyaan tersebut akan diupdate. Saat menghapus pertanyaan, akan divalidasi terlebih dahulu apakah pertanyaan tersebut ada pada database. Setelah pengguna menghapus, maka pertanyaan akan dihapus dari database.

Kasus 10 : Tambahkan pertanyaan xxx dengan jawaban yyy



Kasus 11 : Hapus pertanyaan

BAB V
KESIMPULAN DAN SARAN

Algoritma KMP dan BM adalah algoritma yang baik untuk pencocokan pola dalam string. Algoritma KMP dan BM menunjukkan efisiensi yang baik. Selain itu, RegEx dapat digunakan untuk memroses pola tertentu dalam sebuah string. Salah satu pemanfaatan KMP, BM, dan RegEx adalah untuk membuat ChatBot yang dapat menerima pertanyaan dan memberikan jawaban kepada pengguna berdasarkan pertanyaan di *database*.

Namun, dalam praktiknya, algoritma pencocokan pola sebenarnya bukan algoritma yang cocok untuk ChatBot yang diharapkan. ChatBot memiliki fitur yang sangat terbatas dan sangat bergantung pada pertanyaan di database. Selain itu, pengguna juga harus memberikan input yang diharapkan oleh ChatBot agar ChatBot dapat bekerja dengan baik.

DAFTAR PUSTAKA

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/String%20Matching.ppt, terakhir diakses 1 Mei 2023, 20.46

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/StringMatching-2.pdf, terakhir diakses 1 Mei 2023, 20.48

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/StringMatching-3.ppt, terakhir diakses 1 Mei 2023, 20.53

REPOSITORY

https://github.com/akbarmridho/Tubes3_bangudahbang/

YOUTUBE VIDEO

https://youtu.be/4Pp2ScLA4mk

WEBSITE

https://tubes3-bangudahbang.vercel.app/