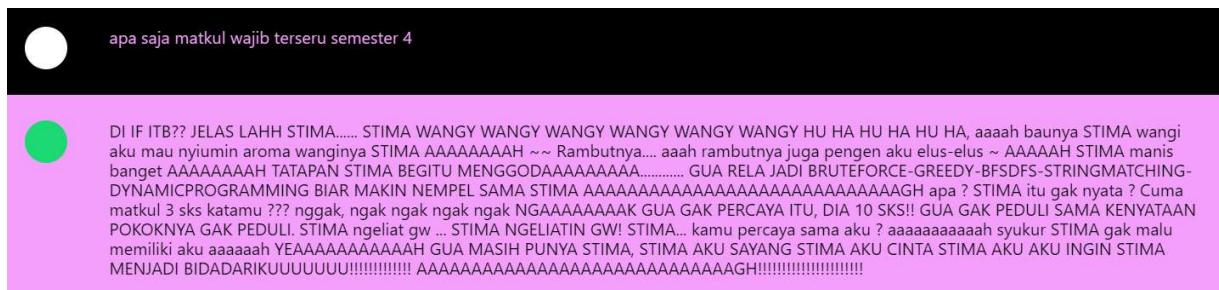


LAPORAN TUGAS BESAR III

Penerapan String Matching dan Regular Expression dalam Pembuatan ChatGPT Sederhana

**IF2211 Strategi Algoritma
Semester II Tahun 2022/2023**

Dosen Pengampu : Dr. Ir. Rinaldi Munir, M.T.



(maaf keos ga sempet fotbar)

Disusun oleh :

ChatPPT

Maggie Zeta RS 13521117
Muhammad Abdul Aziz Ghazali 13521128
Akhmad Setiawan 13521164

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

DAFTAR ISI

DAFTAR ISI.....	2
BAB 1.....	3
BAB 2.....	8
2.1 Knuth-Morris-Pratt (KMP).....	8
2.2 Boyer-Moore (BM).....	8
2.3 Regular Expression (Regex).....	8
BAB 3.....	9
3.1 Langkah penyelesaian masalah setiap fitur.....	9
3.1.1 Fitur Pertanyaan Teks (didapat dari database).....	9
3.1.2 Fitur Kalkulator dan tanggal.....	11
3.1.3 Fitur Menambah dan Mengupdate Pertanyaan.....	12
3.1.4 Fitur menghapus pertanyaan.....	13
BAB 4.....	14
4.1 Penjelasan tata cara penggunaan program.....	14
4.2 Hasil pengujian.....	16
4.3 Analisis hasil pengujian.....	22
BAB 5.....	23
5.1 Kesimpulan.....	23
5.2 Saran.....	23
5.3 Refleksi.....	23
LAMPIRAN.....	24
DAFTAR PUSTAKA.....	25

BAB 1

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.

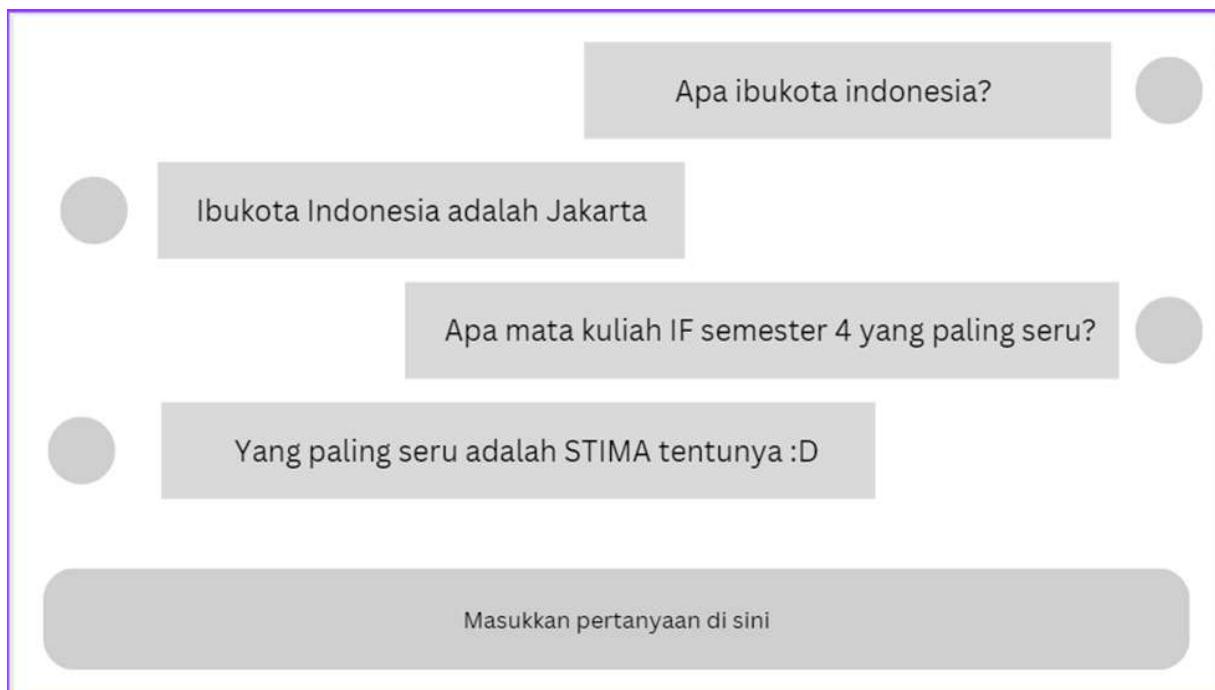
4. Tambah pertanyaan dan jawaban ke database

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 diperbarui.

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.

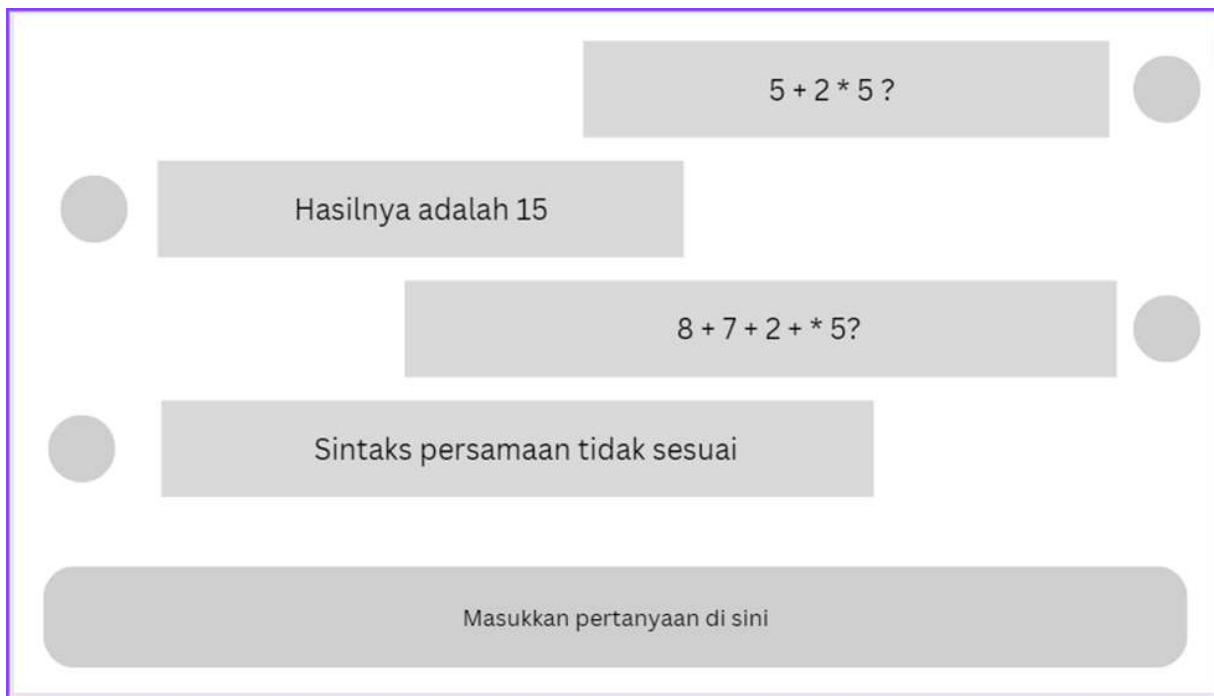
Klasifikasi dilakukan menggunakan **regex** dan terklasifikasi layaknya bahasa sehari - hari. Algoritma string matching KMP dan BM digunakan untuk klasifikasi query teks. Tersedia toggle untuk memilih algoritma KMP atau BM. Semua pemrosesan respons dilakukan pada sisi **backend**. Jika ada pertanyaan yang sesuai dengan fitur, maka tampilkan saja “Pertanyaan tidak dapat diproses”. Berikut adalah beberapa contoh ilustrasi sederhana untuk tiap pertanyaannya. (**Note:** Tidak wajib mengikuti ilustrasi ini, tampilan disamakan dengan chatGPT juga boleh)



Gambar 1. Ilustrasi Fitur Pertanyaan teks kasus exact



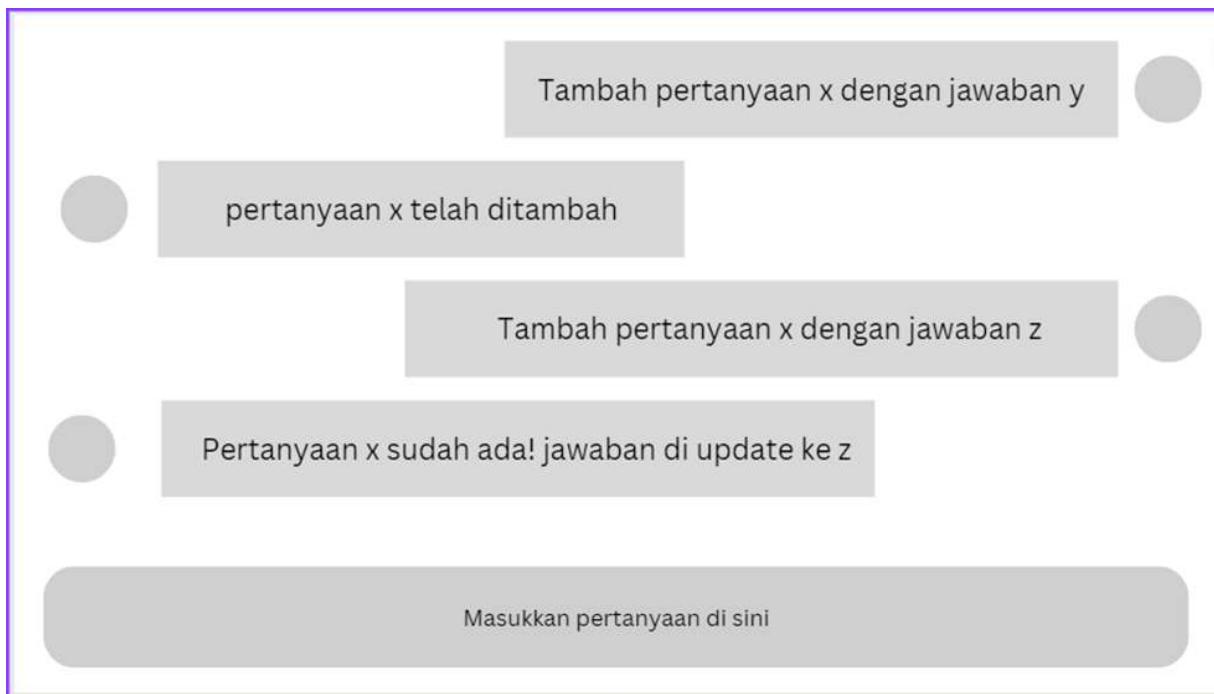
Gambar 2. Ilustrasi Fitur Pertanyaan teks kasus tidak exact



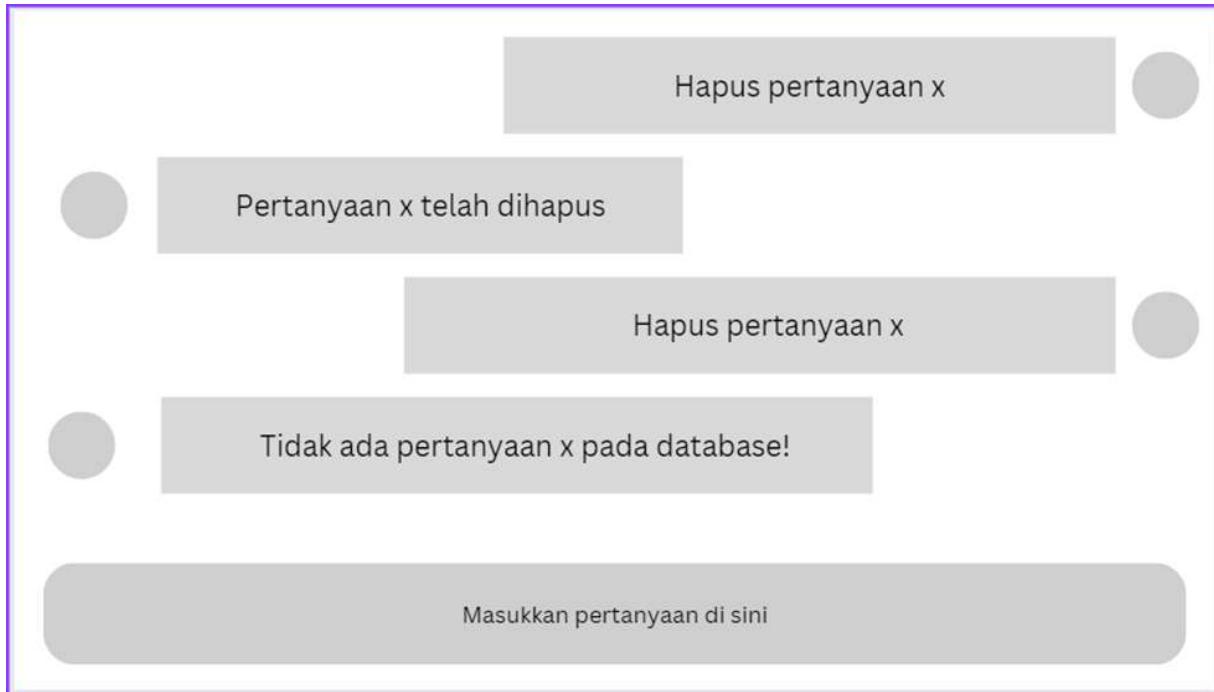
Gambar 3. Ilustrasi Fitur Kalkulator



Gambar 4. Ilustrasi Fitur Tanggal



Gambar 6. Ilustrasi Fitur Tambah Pertanyaan



Gambar 7. Ilustrasi Fitur Hapus Pertanyaan

Layaknya **ChatGPT**, di sebelah kiri disediakan **history** dari hasil pertanyaan anda. Cukup tampilkan 5-10 pertanyaan terbaru di toolbar kiri. Perhatikan bahwa sistem history disini disamakan dengan chatGPT, sehingga satu history yang diklik menyimpan **seluruh pertanyaan pada sesi itu**. Apabila history diclick, maka akan merestore seluruh pertanyaan dan jawaban di halaman utama. Contoh ilustrasi keseluruhan:

BAB 2

Landasan Teori

2.1 Knuth-Morris-Pratt (KMP)

Algoritma Knuth-Morris-Pratt (KMP) adalah algoritma pencocokan string yang mencocokkan string dari kiri ke kanan. Algoritma ini lebih efisien daripada algoritma brute force karena menggeser pattern yang dicari dengan lebih cerdas ketika terjadi mismatch. Saat terjadi mismatch, algoritma KMP menggeser pattern sejauh prefix terbesar dari pattern yang juga merupakan suffix dari pattern itu sendiri. Algoritma KMP membutuhkan border function atau failure function untuk mencari kesamaan antara prefix dari pattern dengan pattern itu sendiri.

2.2 Boyer-Moore (BM)

Algoritma Boyer-Moore adalah algoritma pencocokan string yang menggunakan teknik looking-glass dan teknik character-jump. Teknik looking-glass mencari pattern dari belakang ke depan, sedangkan teknik character-jump memindahkan pattern ketika terjadi mismatch. Algoritma ini menggunakan last occurrence function untuk menghitung indeks terbesar dari karakter-karakter pada pattern. Ketika terjadi mismatch, algoritma Boyer-Moore memindahkan pattern sesuai dengan aturan pada last occurrence function.

2.3 Regular Expression (Regex)

Regular expression atau regex adalah sebuah pattern string yang digunakan untuk mencocokkan dan mengatur text string. Regex memiliki aturan penulisan yang sudah didefinisikan dan harus diikuti. Contohnya, karakter "^" menandakan awalan string, karakter "\$" menandakan akhiran string, dan karakter "*" menandakan jumlah kemunculan sebuah karakter boleh nol atau banyak. Penggunaan regex meningkatkan fleksibilitas dalam pembentukan pattern untuk mencocokkan sebuah text string. Penggunaannya lebih efisien daripada string matching yang mencoba pattern untuk setiap teks yang berbeda.

BAB 3

Analisis Pemecahan Masalah

3.1 Langkah penyelesaian masalah setiap fitur

3.1.1 Fitur Pertanyaan Teks (didapat dari database)

Database yang digunakan adalah MongoDB. Dalam prakteknya, aplikasi pada Frontend akan memanggil endpoints pada backend sesuai dengan input usernya. Pada endpoints, selanjutnya akan memanggil method yang sesuai untuk mencocokkan input user dengan pertanyaan yang telah tersimpan di database, sesuai dengan metode yang dipilih oleh user (KMP atau BM). Jika tidak cocok dengan salah satu algoritma string matchingnya, akan digunakan fungsi LCS (Longest Common Substring) untuk mencari pertanyaan yang paling cocok dengan batas bawah 90%. Jika terdapat pertanyaan dengan kriteria string matching tersebut, akan dipanggil dan ditampilkan. Jika tidak ditemukan jawaban, akan memberikan maksimal 3 rekomendasi pertanyaan yang paling dekat dengan menggunakan LCS.

```
    } else {
      var encodedInput = encodeURIComponent(question);

      //if toggle KMP
      var url = `/response/KMP/${encodedInput}`;

      //if toggle BM
      var url = `/response/BM/${encodedInput}`;

      axios.get(url, {
        responseType: 'json'
      }).then(response => {
        if(response.status === 200) {
          if (response.data === null) {
            setChatLog(prevLog => [...prevLog, { user: "gpt", message: "Pertanyaan tidak ditemukan, silakan tambahkan pertanyaan"}]);
            setHistory(prevHistory => [...prevHistory, `Pertanyaan tidak ditemukan, silakan tambahkan pertanyaan`]);
            console.log("Data not found");
          }
        } else {
          // response.data is already json format
          if (response.data[0].answer != null){
```

Frontend mengakses endpoints dengan axios.get

IF2211 Strategi Algoritma

```
10
11 func main() {
12     r := gin.Default()
13     r.Use(static.Serve("/", static.LocalFile("./web", true)))
14     router := r.Group("/response")
15
16     router.Use(gin.Logger())
17     router.Use(cors.Default())
18
19     // Endpoints used
20     router.GET("/KMP/:question", controller.GetResponseKMP)
21     router.GET("/BM/:question", controller.GetResponseBM)
22
23     // Runs the server and allows it to listen to requests
24     // Runs in localhost 5000
25     r.Run()
26 }
27
```

Endpoints pada backend

```
for _, elmt := range questions {
    // Match using KMP, add into result
    // fmt.Println(Algorithm.KMPSearch(question, string(elmt["question"].(string))))
    if Algorithm.KMPSearch(question, string(elmt["question"].(string))) != -1 {
        result = append(result, elmt)
        break
    } else {
        // Not match KMP but LCS >= 90%, add into result
        // fmt.Println(Algorithm.LongestCommonSubstring(question, string(elmt["question"].(string))))
        if Algorithm.LongestCommonSubstring(question, string(elmt["question"].(string))) >= 90.0 {
            result = append(result, elmt)
            break
        }
    }
}
```

Mencocokkan input dengan String matching algorithm

IF2211 Strategi Algoritma

```
// Question not match KMP and LCS < 90%
if len(result) != 1 {
    flag := bson.M{"answer": "Pertanyaan tidak ditemukan, mungkin maksudnya:"}
    result = append(result, flag)
    rank := []float64{}

    // Get rank of the similiarity using LCS and sort them descending
    for i := 0; i < len(questions); i++ {
        rank = append(rank, Algorithm.LongestCommonSubstring(question, string(questions[i]["question"].(string))))
    }
    sort.Sort(sort.Reverse(sort.Float64Slice(rank)))

    numOfRecommendation := len(rank)
    if len(rank) >= 3 {
        numOfRecommendation = 3
    }
    for i := 0; i < numOfRecommendation; i++ {
        for j := 0; j < len(questions); j++ {

            // Add 3 biggest LCS to result based on rank
            if Algorithm.LongestCommonSubstring(question, string(questions[j]["question"].(string))) == rank[i] {
                result = append(result, questions[j])
            }
        }
    }
}
c.JSON(http.StatusOK, result)
```

Menghandle kasus tidak menemukan pertanyaan

3.1.2 Fitur Kalkulator dan tanggal

Fitur kalkulator dan tanggal tidak membutuhkan akses ke database sama sekali. Input user akan dicocokkan dengan masing-masing regex apakah sesuai dengan format kalkulator dan tanggal yang valid. Jika valid, input akan diolah dengan menggunakan library yang ada pada calculator.js dan date.js. Calculator dan date tidak dipanggil pada backend karena akan mengganggu query pada endpoints jika memiliki karakter ‘backslash’ (/) yang dianggap sebagai path baru pada endpoints, sehingga dihandle pada sisi .js.

```
async function getAnswer(question){
    const regexCalc = /^(\s*\d(\.\d*)?[*^+-]\s*)*\?\?$/;
    const regexDate = /(0?[1-9]|1[0-2])[0-9]{2}|(0?[1-9]|1[0-2])\d{4}/g;
```

Mencocokkan regex Calculator dan Date

IF2211 Strategi Algoritma

```
if (regexDate.test(question)) {
    if (date.isValidDate(question)) { // if valid input date dd/mm/yyyy or d/m/yyyy
        const answer = date.getDay(question)
        setChatLog(prevLog => [...prevLog, { user: "gpt", message: `Tanggal tersebut adalah hari ${answer}` }]);
        setHistory(prevHistory => [...prevHistory, `Tanggal tersebut adalah hari ${answer}`]);
    }
}

else if (regexCalc.test(question)) { // not valid as a date, but valid as calculator (cannot retrieve minus sign)
    const answer = calculator.calculate(question)
    setChatLog(prevLog => [...prevLog, { user: "gpt", message: `Perintah dianggap operasi matematika dengan hasil ${answer}` }]);
    setHistory(prevHistory => [...prevHistory, `Perintah dianggap operasi matematika dengan hasil ${answer}`]);
}

} else { // not valid as both
    setChatLog(prevLog => [...prevLog, { user: "gpt", message: `Input yang mengandung tanda '/' haruslah operasi matematika atau tanggal yang valid` }]);
    setHistory(prevHistory => [...prevHistory, `Input yang mengandung tanda '/' haruslah operasi matematika atau tanggal yang valid, bukan perintah`]);
}

} else if (regexCalc.test(question)) { // valid as calculator (cannot retrieve minus sign)
    const answer = calculator.calculate(question)
    setChatLog(prevLog => [...prevLog, { user: "gpt", message: `Perintah dianggap operasi matematika dengan hasil ${answer}` }]);
    setHistory(prevHistory => [...prevHistory, `Perintah dianggap operasi matematika dengan hasil ${answer}`]);
}
```

Memilih antara eksekusi penghitungan kalkulator atau pencarian hari sesuai tanggal

Implementasi kalkulasi dan pencarian tanggal terdapat pada file calculator.js dan date.js pada repository github terlampir.

3.1.3 Fitur Menambah dan Mengupdate Pertanyaan

Menambah dan mengupdate pertanyaan juga memerlukan akses ke database, sama seperti ketika ingin mencari jawaban sesuai dengan input user. Kecuali, terdapat regex untuk mengenali apakah sebuah input harus menambah atau mengupdate pertanyaan. Metode menambah dan mengupdate pertanyaan disimpan pada sisi backend dalam file questions.go.

```
// Add Question: "tambah pertanyaan .... jawaban ...." or "tambah pertanyaan .... dengan jawaban ...."
regexAdd := regexp.MustCompile(`^(?:(\s+)?tambah(?:\kan)?(?:\s+)?pertanyaan(?:(\s+)?(.+?)?(?:(\s+dengan)?\s+jawaban(?:nya)?))?(?:\s+)?(.+?)?(?:\s*|\b`))
matchAdd := regexAdd.MatchString(question)
parseAdd := regexAdd.FindStringSubmatch(question)

if matchAdd {
    questionAdded := ""
    answerAdded := ""

    // If user not adding the answer
    if parseAdd[1] == "" {
        questionAdded = parseAdd[2]
    } else {
        questionAdded = parseAdd[1]
        answerAdded = parseAdd[2]
    }
    AddQuestionKMP(c, questionAdded, answerAdded, questions)
    return
}
```

Regex dan metode menambah/mengupdate pertanyaan pada database

IF2211 Strategi Algoritma

```
// If matching with KMP or LCS >= 90%
// fmt.Println(Algorithm.LongestCommonSubstring(questionAdded, string(elmt["question"].(string))))
if Algorithm.KMPSearch(questionAdded, string(elmt["question"].(string))) != -1 ||
    Algorithm.LongestCommonSubstring(questionAdded, string(elmt["question"].(string))) >= 90.0 {

    if answerAdded == "" { // Case where user doesn't input the answer yet

        // Question exists but the answer doesn't
        if string(elmt["answer"].(string)) == "" {
            flag := bson.M{"answer": "Pertanyaan \"\" + elmt["question"].(string) + "\" sudah ada, namun belum tersimpan jawabannya. Silakan update
            result = append(result, flag)
            c.JSON(http.StatusOK, result)
            return
        } else {
            // Question exists and the answer as well
            flag := bson.M{"answer": "Pertanyaan \"\" + elmt["question"].(string) + "\" sudah ada dan telah tersimpan jawaban: \"\" + string(elmt["ar
            result = append(result, flag)
            c.JSON(http.StatusOK, result)
            return
        }

    } else { // Update question with new answer
        flag := bson.M{"answer": "Pertanyaan \"\" + elmt["question"].(string) + "\" sudah ada! Jawaban diupdate menjadi: \"\" + answerAdded + \"\"}
        result = append(result, flag)
        // fmt.Println(result[0]["answer"])
        c.JSON(http.StatusOK, result)
    }
}
```

Potongan kode menambahkan pertanyaan ke database

Awalnya input user akan dicocokkan dengan database. Jika ada yang cocok, akan dilakukan update jawaban. Jika tidak, akan ditambahkan jawaban baru. Kode lengkap dapat dilihat pada folder controller di backend pada repository terlampir.

3.1.4 Fitur menghapus pertanyaan

Sama seperti fitur menambah pertanyaan, fitur menghapus pertanyaan hanya memiliki perbedaan di implementasi regexnya saja. Jika input user memiliki pola yang cocok untuk instruksi menghapus pertanyaan, maka akan dipanggil metode untuk menghapus pertanyaan yang sesuai antara input user dan database. Jika tidak ditemukan pertanyaannya, maka akan mengembalikan pesan error “tidak menemukan pertanyaan di database”

```
// Delete Question prompt: "hapus pertanyaan ...." or "hapus ...."
regexDelete := regexp.MustCompile(`^^(?:\s+)?(?:meng)?hapus(?:lah)?(?:kan)?(?:\s+)?(?:pertanyaan(?:\s+)?)(?:.+?)(?:\s*|\b)$`)
matchDelete := regexDelete.MatchString(question)
parseDelete := regexDelete.FindStringSubmatch(question)

// Match with regex
if matchDelete {
    questionDeleted := parseDelete[1]
    DeleteQuestionKMP(c, questionDeleted, questions)
    return
}
```

Regex dan proses pemanggilan metode delete

IF2211 Strategi Algoritma

Saat dipanggil, metode delete akan kembali mencocokkan input user dengan semua pertanyaan pada database menggunakan algoritma string matching (KMP, BM, atau LCS). Jika tidak memenuhi salah satunya, akan gagal menghapus pertanyaan.

```
// Delete a question given the user input using KMP
func DeleteQuestionKMP(c *gin.Context, questionDeleted string, questions []bson.M) {
    var ctx, cancel = context.WithTimeout(context.Background(), 100*time.Second)
    defer cancel()
    var result []bson.M

    for _, elmt := range questions {
        // If matching with KMP algorithm or LCS >= 90.0
        if Algorithm.KMPSearch(questionDeleted, string(elmt["question"].(string))) != -1 || Algorithm.LongestCommonSubstring(questionDeleted, string(elmt["question"].(string))) >= 90.0 {
            // Delete in database
            _, err := questionCollection.DeleteOne(ctx, bson.M{"question": string(elmt["question"].(string))})
            if err != nil {
                c.JSON(http.StatusInternalServerError, gin.H{"error": err.Error()})
                fmt.Println(err)
                return
            }

            // Delete succeed response
            flag := bson.M{"answer": "Pertanyaan \"" + questionDeleted + "\" berhasil dihapus!"}
            result = append(result, flag)
            c.JSON(http.StatusOK, result)
            return
        }
    }

    // Cannot find the question in database
    flag := bson.M{"answer": "Pertanyaan \"" + questionDeleted + "\" tidak ditemukan sehingga tidak bisa dihapus!"}
    result = append(result, flag)
    c.JSON(http.StatusOK, result)
}
```

Metode menghapus pertanyaan dengan algoritma KMP dan LCS (metode BM terpisah)

Metode lengkap dapat dilihat pada folder controller di backend pada repository terlampir.

BAB 4

Implementasi dan Pengujian

4.1 Penjelasan tata cara penggunaan program

Awalnya, server backend harus dijalankan terlebih dahulu. Jangan lupa untuk menambahkan file .env di satu direktori main.go yang berisi seperti contoh berikut:

```
1 PORT=5000
2 MONGODB_URL=mongodb+srv://13521164:30XqnKFphHl4PIe9@cluster0.892igzh.mongodb.net/?retryWrites=true&w=majority
3 DATABASE_NAME=tubes3-stima
4 COLLECTION_NAME=questions
```

Isi dari .env

IF2211 Strategi Algoritma

Mongodb_url dapat diganti dengan url lainnya, namun pastikan anda memiliki collection_name bernama questions dan database_name berupa tubes3-stima. Jika ingin mengubahnya, anda harus mengubah beberapa nama variabel pada program utama.

Jangan lupa juga untuk menginstall package yang dibutuhkan. Setelah selesai, anda bisa menjalankan program backend dengan berada pada directory ./src/be dengan instruksi:

```
go run main.go
```

Ini akan menjalankan server pada localhost yang tersedia (defaultnya localhost:5000). Jika berhasil dijalankan akan muncul pesan seperti berikut:

```
PS C:\Users\Lenovo\Documents\Semester 4\Stima\Tubes3_13521117\src\be> go run main.go
Connected to MongoDB!
[GIN-debug] [WARNING] Creating an Engine instance with the Logger and Recovery middleware already attached.

[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
- using env: export GIN_MODE=release
- using code: gin.SetMode(gin.ReleaseMode)

[GIN-debug] GET    /response/KMP/:question  --> BE/controller.GetResponseKMP (6 handlers)
[GIN-debug] [WARNING] You trusted all proxies, this is NOT safe. We recommend you to set a value.
Please check https://pkg.go.dev/github.com/gin-gonic/gin#readme-don-t-trust-all-proxies for details.
[GIN-debug] Environment variable PORT="5000"
[GIN-debug] Listening and serving HTTP on :5000
```

Server terhubung ke server localhost:5000

Setelah terhubung ke server, selanjutnya dapat melanjutkan menjalankan fitur frontend, dengan masuk ke directory ‘./src/fe’ dan menjalankan perintah:

“npm start”

Jika berhasil, akan memunculkan pesan berikut:

```
Compiled successfully!

You can now view cuakgpt-client in the browser.

Local:          http://localhost:3000
On Your Network: http://192.168.56.1:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Berhasil terhubung ke server

Dan otomatis akan membuka localhost:3000 pada web lokal:

Kemudian dapat menjalankan perintah sesuai dengan fitur-fitur yang ada:

1. Fitur lihat pertanyaan : Masukkan perintah “lihat pertanyaan”
2. Fitur mencari jawaban : Masukkan pertanyaan yang sudah tersedia pada database
3. Fitur kalkulator : Masukkan perintah berupa pola operasi matematika (tidak dapat unary)
4. Fitur tanggal : Masukkan perintah berpola “DD/MM/YYYY”
5. Fitur tambah pertanyaan : Masukkan perintah dengan pola “tambah pertanyaan ... dengan jawaban ... ”
6. Fitur menghapus pertanyaan : Masukkan perintah dengan pola “hapus pertanyaan ... ”

4.2 Hasil pengujian



Test case 1

IF2211 Strategi Algoritma

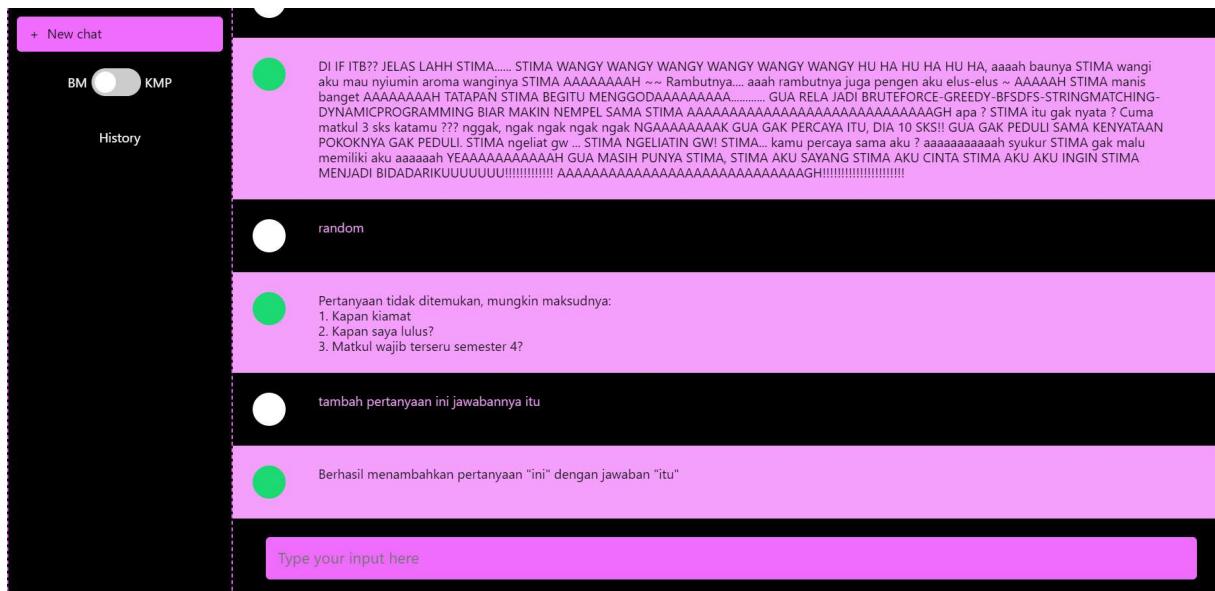


Test case 2



Test case 3

IF2211 Strategi Algoritma



Test case 4



Test case 5

IF2211 Strategi Algoritma



Test case 6



Test case 7

IF2211 Strategi Algoritma



Test case 8

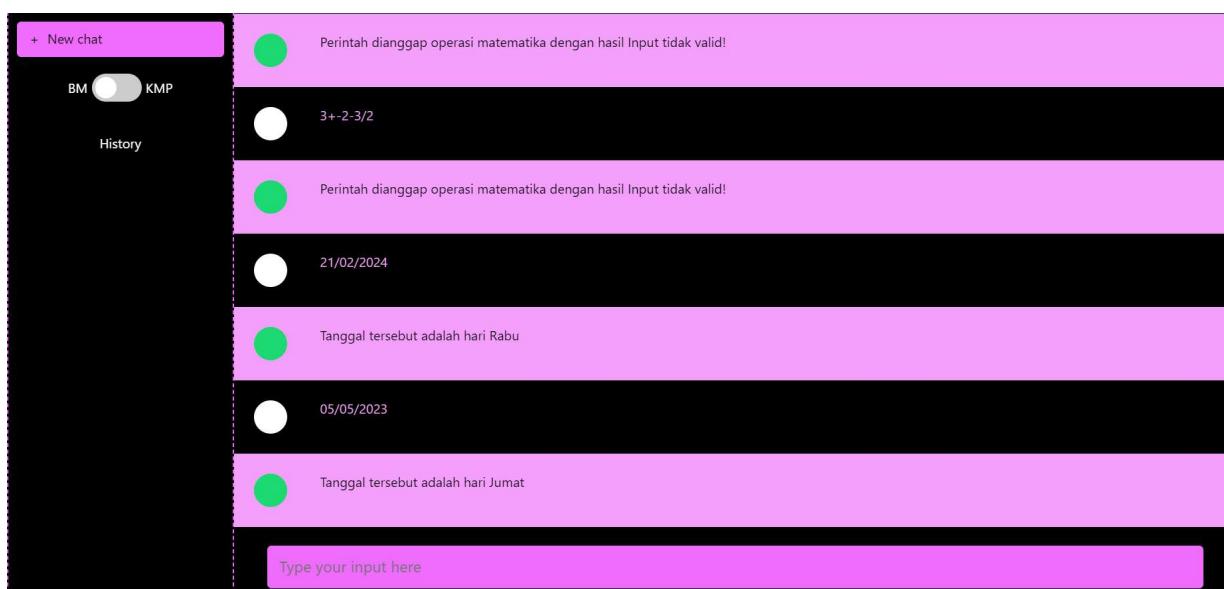


Test case 9 (tidak dapat menerima angka negatif)

IF2211 Strategi Algoritma



Test case 9



Test case 10



Test case 11

Jika input tanggal tidak valid, dianggap sebagai operasi pembagian matematika

4.3 Analisis hasil pengujian

1. Lihat daftar pertanyaan berhasil berjalan dengan baik. Program akan menampilkan semua pertanyaan yang telah terdaftar pada database.
2. Melihat jawaban dari pertanyaan yang sudah terdaftar berhasil dijalankan dan mengeluarkan jawaban yang paling sesuai dengan algoritma string matching.
3. Melihat jawaban dari pertanyaan yang belum tersimpan ke database, akan memunculkan 3 rekomendasi teratas dari pertanyaan yang mungkin dimaksud.
4. Tambah pertanyaan baru berhasil berjalan dengan baik. Pertanyaan dan jawabannya berhasil ditambahkan ke dalam database.
5. Tambah pertanyaan, namun sudah ada di database, berhasil dijalankan dengan mengupdate jawaban dari pertanyaan tersebut dan diletakkan di database kembali.
6. Hapus pertanyaan berhasil berjalan dengan baik. Pertanyaan pada database akan dihapus sesuai dengan input user.
7. Hapus pertanyaan yang tidak ada di database berhasil mengembalikan pesan bahwa tidak ada pertanyaan yang dapat dihapus.
8. Fitur kalkulator berhasil dijalankan dan mengembalikan hasil yang sesuai
9. Fitur kalkulator yang tidak valid berhasil mengembalikan pesan error yang sesuai
10. Fitur tanggal berhasil dijalankan dan memberikan hasil hari pada tanggal yang akurat
11. Fitur tanggal yang tidak valid berhasil nilai operasi matematikanya

BAB 5

Kesimpulan, Saran, dan Refleksi

5.1 Kesimpulan

Algoritma string matching Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM) serta Regular Expression (Regex) telah diimplementasikan dan digunakan dalam sebuah aplikasi ChatGPT yang sederhana. Regex digunakan untuk memvalidasi input dari pengguna. Front end dan back end telah berhasil digabungkan sehingga tercipta sebuah aplikasi ChatGPT sederhana yang memenuhi spesifikasi.

5.2 Saran

Kami dapat memberikan beberapa saran untuk Tugas Besar III IF2211 Strategi Algoritma Semester II 2022/2023. Pertama, algoritma Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM) yang digunakan dalam tugas ini masih memiliki potensi untuk ditingkatkan dalam hal efisiensi program karena kemungkinan masih terdapat kekurangan. Selain itu, untuk mempermudah pemeliharaan program oleh para programmer, disarankan untuk membuat kode program lebih modular dan tersegmentasi dengan baik. Terakhir, program ini dapat dibagikan ke publik setelah dikembangkan lebih lanjut agar bisa memberikan manfaat sebagai referensi publik.

5.3 Refleksi

Setelah menyelesaikan Tugas Besar III IF2211 Strategi Algoritma Semester II 2022/2023, kami menyadari bahwa komunikasi yang terjalin di antara anggota kelompok kami berjalan dengan lancar. Hal ini membuat kami dapat menyelesaikan tugas besar tanpa mengalami kesalahpahaman dan miskomunikasi yang berarti. Sebelum memulai pengerjaan tugas, kami juga melakukan diskusi untuk membahas pembagian tugas bagi setiap anggota kelompok.

LAMPIRAN

Link Repository : https://github.com/semitfinal-com/Tubes3_13521117

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>