Tugas Besar 3 IF2211 Strategi Algoritma

Penerapan String Matching dan Regular Expression dalam Pembuatan ChatGPT Sederhana

Disusun oleh :

Kelompok ReadDoangBalasEnggak

Kartini Copa	13521026
Fajar Maulana Herawan	13521080
Vieri Fajar Firdaus	13521099

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2023

Daftar Isi

Bab I

Deskripsi Masalah

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

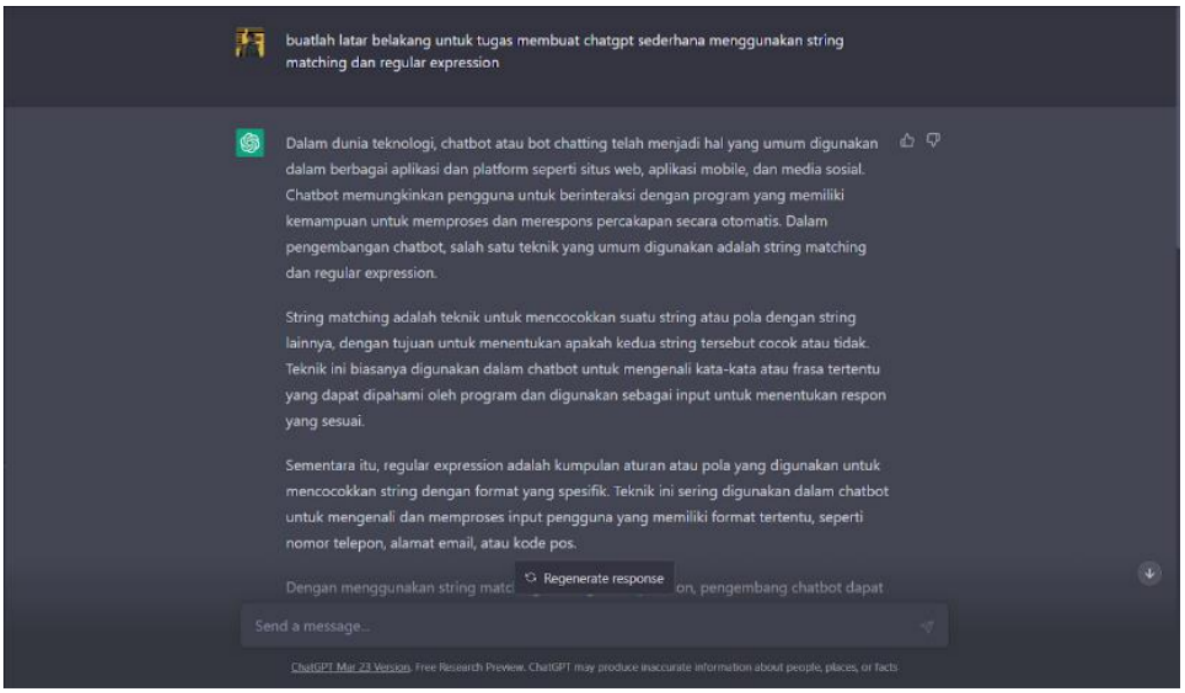
Pengguna memasukkan *input* berupa tanggal, lalu chatbot akan merespon dengan hari apa di tanggal tersebut. Contohnya adalah 25/08/2023 maka chatbot akan menjawab dengan hari senin.

4. Tambah pertanyaan dan jawaban ke database

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

5. Hapus pertanyaan dari database

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



Gambar 1. Ilustrasi Chatbot ChatGPT (funfact latar belakang spek ini dari chatgpt)
Sumber:
https://chat.openai.com/chat

Bab II

Landasan Teori

2.1 Algoritma

2.1.1 Knuth–Morris–Pratt (KMP)

Algoritma Knuth-Morris-Pratt (KMP) merupakan salah satu jenis algoritma pencarian pola/pencocokan string dengan efisien. Algoritma KMP mencari kemiripan atau kecocokan antara pola dan teks dengan membandingkan karakter-karakter pada setiap posisi pada kedua string tersebut, dimulai dari posisi awal atau kiri seperti algoritma *Brute Force*. Namun, algoritma KMP menggunakan sebuah tabel prefix yang mengandung informasi tentang kemiripan pola pada posisi-posisi sebelumnya, sehingga proses pencocokan dapat melompati beberapa karakter pada teks jika karakter tersebut tidak cocok dengan pola, dan kemudian dilanjutkan kembali pada posisi yang lebih optimal. Hal ini membuat algoritma KMP lebih efisien dibandingkan dengan algoritma *Brute Force* yang mencari kemiripan secara langsung pada setiap posisi pada teks.

Pada saat pencarian, algoritma KMP melacak posisi pencocokan dari pola yang sedang dicari dalam teks. Jika karakter dalam pola cocok dengan karakter dalam teks, maka posisi pencocokan akan dinaikkan. Namun, jika karakter dalam pola tidak cocok dengan karakter dalam teks, maka posisi pencocokan akan dikurangi menggunakan tabel prefix. Implementasi algoritma KMP pada program kami dimulai dengan mengecek apakah panjang pola atau teks adalah 0. Jika panjang pola adalah 0, maka pola pasti ada dalam teks (*return true*). Jika panjang teks adalah 0, maka pola tidak mungkin ada dalam teks (*return false*). Selanjutnya, algoritma menghitung tabel prefix dari pola dengan cara mencocokkan awalan pola dengan akhiran pola dan menyimpan jumlah karakter yang cocok dalam tabel prefix. Setelah tabel prefix selesai dibuat, algoritma mencocokkan pola dalam teks dengan menggunakan tabel prefix yang sudah dibuat.

Jika pola ditemukan dalam teks, maka algoritma akan mengembalikan nilai true. Jika tidak, maka algoritma akan mengembalikan nilai false. Algoritma ini efisien karena memiliki kompleksitas waktu O(n + m), di mana n adalah panjang teks dan m adalah panjang pola.

2.1.2 Boyer Moore

Algoritma Boyer Moore (BM) merupakan suatu solusi pencarian yang efisien dapat melakukan perbandingan pola mulai dari kanan ke kiri. Algoritma BM bekerja dengan cara membandingkan pola dari kanan ke kiri, dan pada saat terjadi ketidakcocokan, algoritma akan menggeser pola ke kanan sejauh mungkin berdasarkan karakter yang tidak cocok di teks. Algoritma ini juga menggunakan tabel karakter buruk (*bad character table*) untuk menghindari pencocokan yang sia-sia.

Cara kerja algoritma BM adalah sebagai berikut:

1. Membuat tabel karakter buruk (*bad character table*) dari pola yang akan dicari.
2. Memulai pencarian pola dari posisi terakhir di teks.
3. Melakukan pencocokan karakter pola dengan karakter teks, dimulai dari karakter terakhir pola hingga karakter pertama pola.
4. Jika karakter cocok, lanjutkan pencocokan ke kiri hingga seluruh pola cocok dengan teks atau terjadi ketidakcocokan karakter.
5. Jika terjadi ketidakcocokan karakter pada posisi ke-j dari kanan ke kiri dalam pola, geser pola sejauh m - j karakter ke kanan (m adalah panjang pola), sehingga karakter ke-j menjadi berada pada posisi terakhir pola yang cocok dengan karakter di teks.
6. Ulangi langkah 3-5 sampai seluruh teks telah dicocokkan atau tidak ada lagi pola yang cocok dengan teks.

Algoritma Boyer Moore pada dasarnya menghindari pencocokan karakter yang tidak perlu dengan cara menggeser pola sejauh mungkin berdasarkan tabel karakter buruk. Dalam implementasinya, algoritma ini terbukti efisien untuk mencari pola dalam teks, terutama ketika panjang pola lebih panjang dari panjang teks, atau ketika terdapat banyak karakter yang sama di akhir pola. Kompleksitas waktu dari algoritma BM adalah O(n/m), di mana n adalah panjang teks (text) dan m adalah panjang pola (pattern).

Namun, dalam praktiknya, algoritma Boyer Moore biasanya memiliki performa yang lebih cepat daripada algoritma pencocokan string lainnya seperti algoritma KMP. Hal ini disebabkan oleh penggunaan dua tabel heuristik, yaitu tabel karakter buruk (*bad character table*) dan tabel sufiks baik (*good suffix table*), yang memperbolehkan algoritma BM untuk melakukan "loncatan" yang lebih besar dalam pencarian string.Namun, dalam banyak kasus, algoritma BM dapat mencapai kecepatan yang sangat tinggi dalam pencarian string.

2.1.3 Levenshtein Distance

Algoritma *Levenshtein Distance* merupakan algoritma yang digunakan untuk menghitung jarak antara dua string atau kata. Jarak yang dihitung adalah jumlah minimum dari operasi yang dibutuhkan untuk mengubah satu string menjadi string lainnya. Pada algoritma ini, pertama-tama dilakukan inisialisasi matriks yang akan digunakan untuk menghitung jarak antara dua string. Kemudian, dilakukan perhitungan jarak dengan menggunakan konsep dynamic programming. Proses perhitungan jarak dimulai dari baris dan kolom pertama matriks yang sudah diinisialisasi. Pada setiap selanjutnya, nilai jarak dihitung dengan mempertimbangkan nilai pada tiga sel sebelumnya, yaitu di atas, di kiri, dan diagonal kiri atas. Operasi *insert*, *delete*, *atau substitute* dilakukan tergantung pada perbandingan karakter pada kedua string pada posisi yang sama. Setelah seluruh sel pada matriks terisi dengan nilai jarak yang sesuai, maka nilai jarak minimum antara kedua string dapat ditemukan pada sel terakhir di sudut kanan bawah matriks.

Kompleksitas waktu algoritma ini adalah O(mn), di mana m dan n adalah panjang dari kedua string yang dibandingkan. Hal ini dikarenakan terdapat dua buah perulangan untuk mengisi sel pada matriks dengan kompleksitas O(mn) dan O(1) operasi pada setiap sel. Kompleksitas ruangnya adalah O(mn) karena matriks berukuran mxn digunakan untuk menyimpan nilai jarak.

2.1.4 Regular Expression

Regular Expression merupakan sebuah pola atau susunan karakter yang digunakan untuk mencari atau mengganti sebuah teks tertentu dengan format tertentu. *Regular expression* didefinisikan berdasarkan aturan teori bahasa formal. *Regular expression* digunakan untuk memproses teks dengan cara yang lebih kompleks daripada hanya mencari dan mengganti teks yang tepat. Dalam tugas besar ini, *regular expression* digunakan untuk menghapus karakter *whitespace* yang ada di awal dan akhir teks dengan menggunakan fungsi strings.TrimSpace(). Kemudian, *regular expression* digunakan untuk mengonversi seluruh karakter pada teks menjadi *lowercase* dengan menggunakan fungsi strings.ToLower(). Selanjutnya, *regular expression* digunakan untuk mengganti semua karakter *whitespace* yang berulang menjadi satu karakter *whitespace* saja dengan menggunakan fungsi regexp.MustCompile(\s+). Terakhir, *regular expression* digunakan untuk menghapus semua karakter yang bukan *alphanumeric* atau *whitespace* dengan menggunakan fungsi regexp.MustCompile([[:alnum:]\s]).

Dalam proses penggunaannya, *regular expression* memiliki beberapa operator seperti karakter spesial seperti ".", "^", "$", "*", "+", "?", "()", "[]", "{}", "|" yang digunakan untuk menggambarkan pola pencarian atau penggantian. *regular expression* juga memiliki metakarakter seperti "\d", "\w", "\s" yang digunakan untuk mencocokan karakter tertentu seperti digit, karakter word, atau whitespace.

2.2 Aplikasi Web yang Dibangun

Aplikasi web yang dibangun merupakan aplikasi ChatGPT sederhana dengan mengaplikasikan pendekatan *question answering* (QA) sederhana. 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. Pencarian pertanyaan yang paling mirip dengan pertanyaan yang diberikan pengguna dilakukan dengan algoritma pencocokan string Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM). *Regular expression* digunakan untuk menentukan format dari pertanyaan. Jika tidak ada satupun pertanyaan pada database yang *exact match* dengan pertanyaan pengguna melalui algoritma KMP ataupun BM, maka pencarian pertanyaan termirip dengan kesamaan setidaknya 90% menggunakan *Levenshtein Distance*. Apabila tidak ada pertanyaan yang kemiripannya di atas 90%, maka akan memberikan maksimum 3 pilihan pertanyaan yang paling mirip untuk dipilih oleh pengguna.
Aplikasi ini memiliki beberapa fitur seperti fitur pertanyaan teks, fitur kalkulator, fitur tanggal, fitur tambah pertanyaan dari database, dan fitur hapus pertanyaan dari database. (didapat dari database)

Bab III

Analisis Pemecahan Masalah

3.1 Langkah Penyelesaian Masalah setiap Fitur

3.1.1 Fitur Pertanyaan Teks

Fitur pertanyaan teks menerima tiga parameter: question yang berisi pertanyaan yang ingin dicari jawabannya, questions yang berisi kumpulan pertanyaan yang sudah ada, dan answers yang berisi kumpulan jawaban yang sesuai dengan pertanyaan. Berikut merupakan langkah penyelesaian fitur pertanyaan teks.

1. Pertama-tama, memeriksa jenis *searching* yang digunakan (KMP atau BM) yang dipilih oleh pengguna pada saat penggunaan program.
2. Setelah jenis *searching* ditentukan, algoritma menggunakan fungsi findMatch untuk mencari kesamaan antara pertanyaan pengguna dengan setiap pertanyaan dalam pertanyaanList.
3. Dalam findMatch, pertanyaan pengguna dan pertanyaan dalam pertanyaanList diproses terlebih dahulu dengan fungsi processText yang merupakan implementasi *regular expression* untuk menghilangkan karakter-karakter khusus dan mengubah semua karakter menjadi huruf kecil agar dapat dicocokkan dengan tepat.
4. Kemudian, membuat fitur pilih algoritma pencocokan (KMP atau BM) sesuai dengan jenis *searching* yang dipilih oleh pengguna, dan menggunakan algoritma itu untuk mencari pertanyaan yang sama persis dalam pertanyaanList.
5. Jika ditemukan pertanyaan yang sama persis, algoritma mengembalikan jawaban yang sesuai dari jawabanList. Jika tidak, algoritma menggunakan fungsi levenshteinDistance untuk mencari pertanyaan yang mirip dalam pertanyaanList.
6. Fungsi levenshteinDistance digunakan untuk menghitung jarak Levenshtein (jarak edit) antara dua string. Jarak edit adalah jumlah minimum operasi yang diperlukan untuk mengubah satu string menjadi string yang lain, di mana setiap operasi dapat berupa penghapusan, penggantian, atau penambahan karakter.
7. Algoritma menggunakan jarak edit untuk membandingkan pertanyaan pengguna dengan setiap pertanyaan dalam pertanyaanList, dan mengembalikan pertanyaan dengan skor terbaik (terdekat) sebagai pertanyaan yang mirip.
8. Jika tidak ada pertanyaan yang mirip ditemukan, algoritma mengembalikan pesan "Maaf, saya tidak mengerti pertanyaan Anda." Jika ada, algoritma menyusun pesan yang memuat daftar pertanyaan yang mirip dalam urutan yang sesuai dengan skor kesamaannya dan mengembalikan pesan itu sebagai jawaban.

3.1.2 Fitur Kalkulator

Fitur kalkulator memungkinkan pengguna memasukkan *input query* berupa persamaan matematika. Berikut merupakan langkah penyelesaian fitur kalkulator.

1. Membuat fungsi allMath untuk memeriksa apakah *input* hanya terdiri dari angka dan operator matematika yang diizinkan. Jika *input* tidak valid, maka fungsi akan mengembalikan nilai false.
2. Membuat fungsi filterMath untuk memfilter karakter-karakter yang tidak valid dari *input* dan mengembalikan string yang hanya terdiri dari angka dan operator matematika yang diizinkan.
3. Membuat fungsi calculator yang menerima *input* sebagai string dan mengembalikan hasil perhitungan dalam bentuk float64 dan error. Fungsi calculator melakukan tokenisasi *input* menggunakan fungsi tokenize, mengubah notasi infix menjadi postfix menggunakan fungsi infixToPostfix, dan mengevaluasi postfix menggunakan fungsi evaluatePostfix.
4. Membuat fungsi tokenize untuk mengubah *input* dari string menjadi array token. Fungsi ini akan memisahkan angka dan operator matematika ke dalam token terpisah.
5. Membuat fungsi infixToPostfix untuk mengubah notasi infix menjadi postfix menggunakan algoritma shunting-yard. Fungsi ini akan mengecek operator dan operand dari *input*, kemudian memasukkannya ke dalam output array sesuai dengan urutan postfix yang diinginkan.
6. Membuat fungsi evaluatePostfix untuk mengevaluasi hasil postfix dari *input*. Fungsi ini akan memasukkan operand ke dalam stack nilai, kemudian melakukan operasi binary pada operator yang ditemukan pada stack operator.
7. Membuat beberapa fungsi pendukung seperti isNumber dan isOperator untuk membantu fungsi utama dalam mengevaluasi *input*. isNumber digunakan untuk memeriksa apakah token saat ini adalah angka atau bukan, sementara isOperator digunakan untuk memeriksa apakah token saat ini adalah operator matematika yang diizinkan.
8. Membuat fungsi hasHigherPrecedence untuk memeriksa apakah operator pertama memiliki precedensi yang lebih tinggi dari operator kedua.
9. Membuat fungsi evaluateBinaryOperator untuk mengevaluasi operasi binary yang ditemukan pada *input*.

3.1.3 Fitur Tanggal

Fitur tanggal memungkinkan respon nama hari sesuai *input* pengguna berupa tanggal. Berikut merupakan langkah penyelesaian fitur kalkulator.

1. Memeriksa apakah format tanggal yang dimasukkan sesuai dengan format dd/mm/yyyy menggunakan fungsi dateCheck. Fungsi ini menggunakan *regular expression* untuk melakukan validasi pada format tanggal yang dimasukkan.

2. Memastikan bahwa tanggal yang dimasukkan sesuai dengan format dd/mm/yyyy dengan fungsi parsingDate. Jika sesuai, fungsi akan mengembalikan string tanggal tersebut. Jika tidak sesuai, fungsi akan mengembalikan string kosong ("").
3. Memeriksa apakah tanggal yang dimasukkan valid atau tidak menggunakan fungsi isValidDate. Fungsi ini menggunakan package time pada Go untuk memeriksa validitas tanggal.
4. Parsing pada tanggal yang dimasukkan menjadi format dd/mm/yyyy menggunakan fungsi parsingValidDate. Fungsi ini akan mengembalikan string tanggal yang sudah terformat dengan benar.
5. Mendapatkan nama hari berdasarkan tanggal yang dimasukkan menggunakan fungsi getDay. Fungsi ini akan mengembalikan string nama hari dalam bahasa Indonesia.

3.1.4 Tambah pertanyaan dan jawaban ke database

Fitur tambah pertanyaan dan jawaban ke database memungkinkan pengguna dapat menambahkan pertanyaan dan jawabannya sendiri ke database. Fitur ini menggunakan algoritma string matching untuk mencari tahu apakah pertanyaan sudah ada. Apabila sudah, maka jawaban akan diperbaharui. Berikut langkah
Membuat struct Pertanyaan dan Respon sebagai model data untuk pertanyaan dan jawaban yang akan ditambahkan ke dalam database.

1. Membuat struct Pertanyaan dan Respon sebagai model data untuk pertanyaan dan jawaban yang akan ditambahkan ke dalam database.
2. Membuat fungsi addQuestionReq(quest Pertanyaan) string untuk melakukan validasi data pertanyaan yang akan ditambahkan. Fungsi ini akan mengembalikan string kosong jika data pertanyaan valid, dan mengembalikan pesan error jika data tidak valid.
3. Membuat endpoint addQuestion pada echo.Context yang akan menerima request dengan metode HTTP POST yang akan menambahkan data pertanyaan ke dalam database. Langkah-langkah yang dilakukan pada endpoint ini adalah menerima *request* dari *client* dengan parameter question yang berisi data pertanyaan. Setelah itu, validasi data pertanyaan dengan menggunakan fungsi addQuestionReq(quest Pertanyaan) string. Jika data pertanyaan valid, maka data akan ditambahkan ke dalam database menggunakan method db.Create(&quest) pada ORM GORM. Mengembalikan pesan sukses jika data berhasil ditambahkan, atau pesan error jika terjadi kegagalan.
4. Membuat fungsi addResponReq(respon Respon) string untuk melakukan validasi data respon yang akan ditambahkan. Fungsi ini akan mengembalikan string kosong jika data respon valid, dan mengembalikan pesan error jika data tidak valid.
5. Membuat endpoint addRespon pada echo.Context yang akan menerima request dengan metode HTTP POST yang akan menambahkan data respon ke dalam database.

3.1.5 Hapus pertanyaan dari database

Langkah penyelesaian untuk menghapus pertanyaan dari database dapat dilakukan dengan langkah-langkah sebagai berikut.

1. Membuat fungsi deleteQuestionReq dengan parameter input berupa string question, yang akan mengembalikan string sebagai output.
2. Pada fungsi deleteQuestionReq, menghubungkan ke database untuk mencari pertanyaan dengan menggunakan question sebagai kunci pencarian.
3. Jika pertanyaan ditemukan pada database, maka lakukan penghapusan data dengan menggunakan perintah SQL DELETE FROM dan menjalankannya melalui eksekutor database.
4. Jika penghapusan berhasil, kembalikan pesan sukses yang sesuai.
5. Jika pertanyaan tidak ditemukan pada database, kembalikan pesan error yang sesuai.
6. Membuat fungsi deleteQuestion pada endpoint API, yang akan menerima permintaan HTTP DELETE dengan parameter question dan akan memanggil fungsi deleteQuestionReq.
7. Mengembalikan respons HTTP berdasarkan hasil dari fungsi deleteQuestionReq.

3.2 Fitur fungsional dan arsitektur aplikasi web yang dibangun

3.2.1 Fitur Fungsional

Aplikasi web CHATGPT yang telah dibuat memiliki berbagai fitur fungsional, antara lain sebagai berikut.

1. Pertanyaan Teks: Pengguna dapat mengetikkan pertanyaan atau masukan teks dan sistem akan memberikan jawaban berdasarkan informasi yang ada di database.
2. Kalkulator: Pengguna dapat melakukan perhitungan matematika sederhana dengan mengetikkan persamaan matematika pada kolom masukan. Sistem akan menghitung persamaan tersebut dan memberikan hasilnya.
3. Tanggal: Pengguna dapat mengetikkan pertanyaan yang berhubungan dengan tanggal dan waktu, seperti "hari ini tanggal berapa?", dan sistem akan memberikan jawaban yang sesuai.
4. Tambah Pertanyaan dan Jawaban ke Database: Pengguna dapat menambahkan pertanyaan dan jawaban baru ke dalam database dengan mengikuti format yang ditentukan.
5. Hapus Pertanyaan dari Database: Pengguna dapat menghapus pertanyaan yang tidak diperlukan lagi dari database dengan memasukkan pertanyaan yang ingin dihapus.

3.2.2 Arsitektur Aplikasi Web

Aplikasi web ChatGPT sederhana dibangun dan dikembangan dengan arsitektur sebagai berikut.

1. *Backend*: Golang
2. *Frontend*: JavaScript, HTML, CSS, React
3. Kakas basis data: MySQL

Sisi frontend terdapat pada folder "client". Aplikasi ini dibangun menggunakan *framework* React dan menggunakan bahasa pemrograman TypeScript. Pada folder "public" terdapat file-file yang dapat diakses publik seperti logo dan gambar icon. Pada folder "src" terdapat file-file sumber untuk membangun tampilan website seperti file untuk membuat komponen, halaman, dan file-file konfigurasi. Beberapa komponen yang digunakan dalam aplikasi ini antara lain:

- Chat: untuk menampilkan pesan-pesan pada chat.
- Message: untuk menampilkan pesan pada chat.
- SendMessage: untuk mengirim pesan pada chat.
- Sidebar: untuk menampilkan daftar chat.
- SideButton: untuk membuat tombol pada sidebar.
- TypingMessage: untuk menampilkan pesan yang sedang diketik.

Sisi backend terdapat pada folder "server". Struktur direktori pada folder server terdapat file-file berbahasa pemrograman Go yang digunakan untuk membentuk server dan beberapa fitur yang diimplementasikan pada server antara lain kalkulator, kalender, dan algoritma. Aplikasi ini memiliki API endpoint yang akan digunakan oleh client untuk mengakses fitur-fitur tersebut.

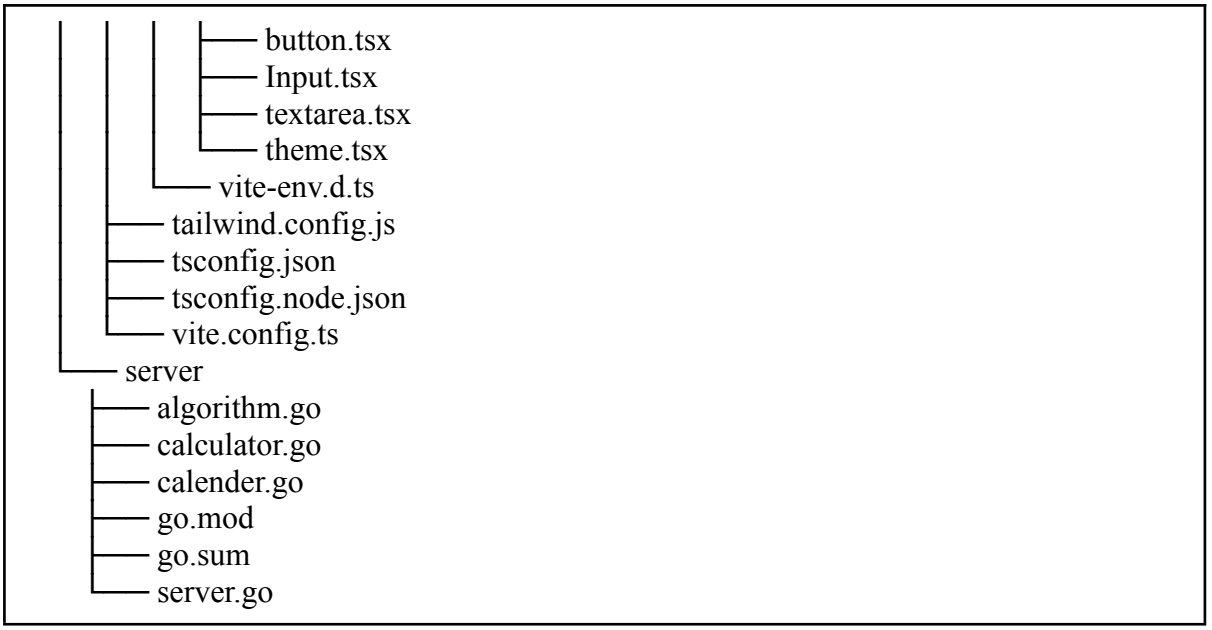
Bab IV

Implementasi dan Pengujian

4.1. Spesifikasi Teknis Program

4.1.1 Struktur data

```
.
├── .DS_Store
├── .github
│   └── workflows
│       └── go.yml
├── build
│   └── web.config
├── doc
├── README.md
└── src
    ├── client
    │   ├── .eslintrc.cjs
    │   ├── .gitignore
    │   ├── index.html
    │   ├── package-lock.json
    │   ├── package.json
    │   ├── postcss.config.js
    │   ├── public
    │   │   ├── logo.png
    │   │   ├── send.png
    │   │   ├── sendIcon.svg
    │   │   └── user.png
    │   ├── src
    │   │   ├── App.css
    │   │   ├── App.tsx
    │   │   ├── assets
    │   │   │   └── react.svg
    │   │   ├── components
    │   │   │   ├── chat.tsx
    │   │   │   ├── message.tsx
    │   │   │   ├── sendmessage.tsx
    │   │   │   ├── sidebar.tsx
    │   │   │   ├── sideButton.tsx
    │   │   │   └── typingMessage.tsx
    │   │   ├── index.css
    │   │   ├── interface
    │   │   │   └── index.tsx
    │   │   ├── main.tsx
    │   │   ├── pages
    │   │   │   └── Home.tsx
    │   │   └── theme
```



Program terdiri atas 2 folder utama, yaitu doc dan src, berikut uraian lengkap mengenai struktur data program.

1. **.DS_Store**: file ini adalah file metadata
2. **.github**: direktori ini berisi file konfigurasi GitHub Actions, yaitu **go.yml** yang digunakan untuk melakukan build dan test pada proyek Go.
3. **build**: direktori ini berisi file konfigurasi untuk aplikasi web, yaitu **web.config**.
4. **doc**: direktori ini mungkin berisi dokumentasi atau panduan penggunaan aplikasi.
5. **README.md**: file ini biasanya berisi informasi tentang proyek dan bagaimana cara menggunakannya.
6. **src**: direktori ini berisi kode sumber dari proyek perangkat lunak.
    - **client**: direktori ini berisi kode sumber untuk sisi klien (client-side) dari aplikasi web.
        - client/.eslintrc.cjs, client/.gitignore, client/index.html, client/package-lock.json, client/package.json, client/postcss.config.js, client/tailwind.config.js, client/tsconfig.json, client/tsconfig.node.json, client/vite.config.ts: file-file konfigurasi untuk proyek aplikasi web React.
        - client/public: direktori ini berisi file-file statis untuk aplikasi web, seperti gambar logo, ikon, dan lain-lain.
        - client/src: direktori ini berisi kode sumber untuk aplikasi web React.
    - **server**: direktori ini berisi kode sumber untuk sisi server (server-side) dari aplikasi web.
        - **algorithm.go**, **calculator.go**, dan **calender.go**: tiga file Go yang berisi fungsi-fungsi utilitas.
        - **go.mod** dan **go.sum**: dua file yang digunakan untuk mengelola dependensi Go.
        - **server.go**: file ini adalah file utama untuk menjalankan server web.

4.1.2 Fungsi dan Prosedur

Berikut merupakan fungsi utama dalam program yang kami buat.

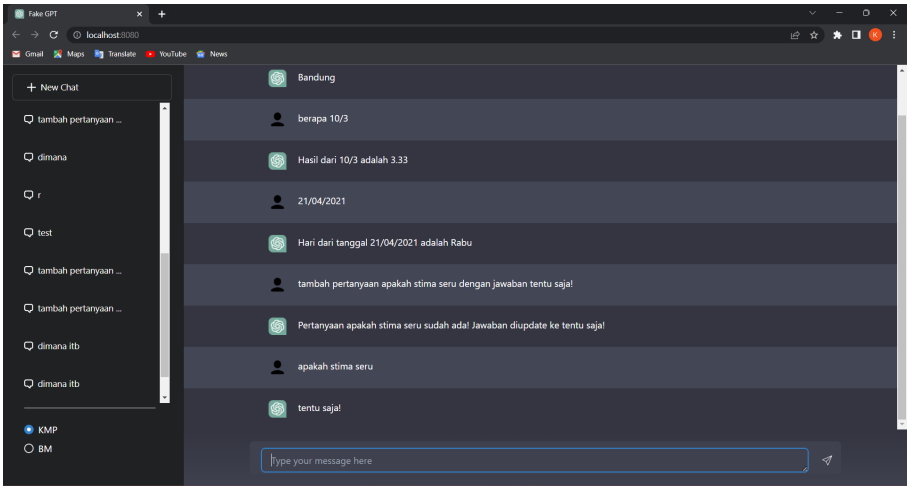
Fungsi atau prosedur	Penjelasan
max	Mengembalikan nilai maksimum dari dua bilangan bulat. Fungsi ini digunakan dalam algoritma pencocokan pola Boyer-Moore (BM) untuk menghitung jarak mundur (*backward shift*) maksimum dari indeks yang tidak cocok di antara karakter dalam pola dan teks.
min	Mendapatkan nilai minimum dari dua nilai integer. Fungsi ini digunakan dalam implementasi algoritma Levenshtein distance pada kodingan di atas.
KMP	Mengecek apakah sebuah pola tertentu ditemukan dalam teks dari kanan ke kiri menggunakan tabel prefix. Tabel prefix digunakan untuk mengetahui jumlah karakter yang harus di-skip ketika terjadi ketidakcocokan antara pola dan teks dalam proses pencocokan pola. Dalam proses pencarian, fungsi KMP akan membandingkan pola dengan string mulai dari karakter pertama hingga akhir. Jika ditemukan karakter yang tidak sesuai dengan pola, fungsi akan memanfaatkan tabel prefix untuk menentukan posisi karakter yang akan dibandingkan pada iterasi berikutnya. Fungsi ini mengembalikan nilai *true* jika pola ditemukan pada string, dan *false* jika tidak ditemukan.
BM	Melakukan pencarian pola dalam sebuah teks dengan memanfaatkan informasi dari kemunculan karakter terakhir dari pola yang dicari untuk mempercepat pencarian pola dalam teks. Pencocokan pola dilakukan dengan menggeser pola ke arah kanan hingga pola cocok atau tidak cocok dengan teks yang ada. Pada saat pola tidak cocok, geserannya dilakukan berdasarkan aturan bad character, yaitu dengan mencari kemunculan terakhir karakter yang tidak cocok pada pola dalam teks dan menggeser pola ke arah kanan hingga karakter pada posisi yang sama dengan karakter tidak cocok pada pola. Kemudian, jika pola tidak cocok pada karakter tertentu di teks, geseran pola dilakukan berdasarkan aturan good suffix. Fungsi max() digunakan untuk mencari nilai maksimum antara dua bilangan.
levenshteinDistance	Mengukur berapa banyak operasi yang diperlukan untuk mengubah satu string menjadi string lainnya. Operasi yang diizinkan adalah *insert*, *delete*, atau *replace* karakter pada suatu posisi dalam string. Dalam hal ini, fungsi ini akan menghitung jarak antara string pertanyaan pengguna dengan string pertanyaan yang ada di database dan memberikan hasil jarak

	sebagai acuan untuk menentukan pertanyaan mana yang paling mirip dengan pertanyaan pengguna.
processText (*regular expression*)	Memproses teks dengan menghilangkan karakter non-alphanumeric dan non-whitespace, mengkonversi teks ke huruf kecil, menghilangkan spasi berlebih, dan menghapus spasi awal dan akhir teks.
calculator	Melakukan kalkulasi aritmatika pada sebuah string *input* sehingga menghasilkan perhitungan matematika dengan mudah dan cepat. Fungsi-fungsi di dalamnya, seperti allMath, filterMath, calculator, tokenize, infixToPostfix, evaluatePostfix, isNumber, isOperator, hasHigherPrecedence, evaluateBinaryOperator, dan pow berguna untuk melakukan parsing dan evaluasi ekspresi matematika dalam bentuk string menjadi hasil yang diharapkan. Fungsi allMath dan filterMath digunakan untuk memastikan input hanya berisi karakter yang valid untuk operasi matematika. Fungsi calculator adalah fungsi utama yang memanggil fungsi-fungsi lainnya untuk melakukan evaluasi ekspresi matematika. Fungsi tokenize digunakan untuk memecah ekspresi matematika menjadi token-token yang dapat diproses lebih lanjut. Fungsi infixToPostfix melakukan konversi dari notasi infix menjadi notasi postfix. Fungsi evaluatePostfix melakukan evaluasi ekspresi matematika dalam notasi postfix. Fungsi-fungsi lainnya seperti isNumber, isOperator, hasHigherPrecedence, evaluateBinaryOperator, dan pow digunakan untuk melakukan operasi matematika pada bilangan.
calender	Memeriksa dan memproses tanggal yang dimasukkan oleh pengguna.Fungsi dateCheck digunakan untuk memeriksa apakah format tanggal yang diberikan sesuai dengan format yang diharapkan, yaitu DD/MM/YYYY. Fungsi parsingDate digunakan untuk memastikan bahwa tanggal yang diberikan sesuai dengan format yang diharapkan dan mengembalikan tanggal tersebut dalam format yang diharapkan. Fungsi isValidDate memeriksa apakah tanggal yang diberikan benar-benar valid dengan memeriksa apakah tanggal tersebut dapat di-parse menggunakan fungsi time.Parse. Fungsi parsingValidDate memastikan bahwa tanggal yang diberikan sesuai dengan format yang diharapkan dan mengembalikan tanggal tersebut dalam format DD/MM/YYYY. Fungsi getDay digunakan untuk menentukan nama hari dalam bahasa Indonesia berdasarkan tanggal yang diberikan menggunakan rumus Zeller's congruence.
updateQuestion	Menambah pertanyaan dari pengguna ke database
addRespon	Menambahkan jawaban dari pengguna ke database
deleteQuestion	Menghapus pertanyaan yang ada di database sesuai masukkan

	dari pengguna
getChatFromId	Mengambil data histori *chat* berdasarkan ID histori yang diberikan. Fungsi ini digunakan dalam sebuah program yang mungkin merupakan bagian dari aplikasi *chat* atau sistem pesan instan.
addHistori	Menambahkan data histori baru ke dalam database. Fungsi ini mengambil parameter dari HTTP request body yang berisi data yang ingin ditambahkan, lalu memasukkannya ke dalam database menggunakan objek db yang sudah terkoneksi dengan database.
deleteHistori	Menghapus suatu data histori beserta seluruh data respon yang terkait dengan ID tertentu dari database. Fungsi ini menggunakan parameter id_histori yang diterima melalui query parameter dari request yang masuk.
showHistori	Menampilkan isi dari histori berdasarkan Id_histori yang diambil dari parameter query pada URL. Fungsi ini melakukan query pada database untuk mendapatkan data histori dan respons yang terkait, kemudian mengembalikan hasil query dalam bentuk JSON dengan struktur yang telah ditentukan sebelumnya pada struct Histori. Selain itu, fungsi ini juga mengambil nama histori berdasarkan Id_histori dan dimasukkan ke dalam struct Histori.
updateHistoriName	Mengubah nama histori yang terdapat dalam database. Fungsi ini menerima *input* dari POST request dalam bentuk UpdateHistori yang memiliki dua atribut, yaitu ID_histori dan NewName. ID_histori digunakan untuk mengidentifikasi histori mana yang ingin diubah namanya, sedangkan NewName adalah nama baru yang akan digunakan.

4.2 Tata Cara Penggunaan Program

Interface aplikasi web ChatGPT sederhana terdiri dari sebuah kotak *input* untuk pengguna mengetik pertanyaan mereka dan sebuah kotak output untuk menampilkan jawaban yang dihasilkan oleh ChatGPT. Pengguna dapat mengetik pertanyaan apapun ke kotak *input* dan ChatGPT akan memberikan jawaban terbaik berdasarkan pertanyaan yang diberikan.

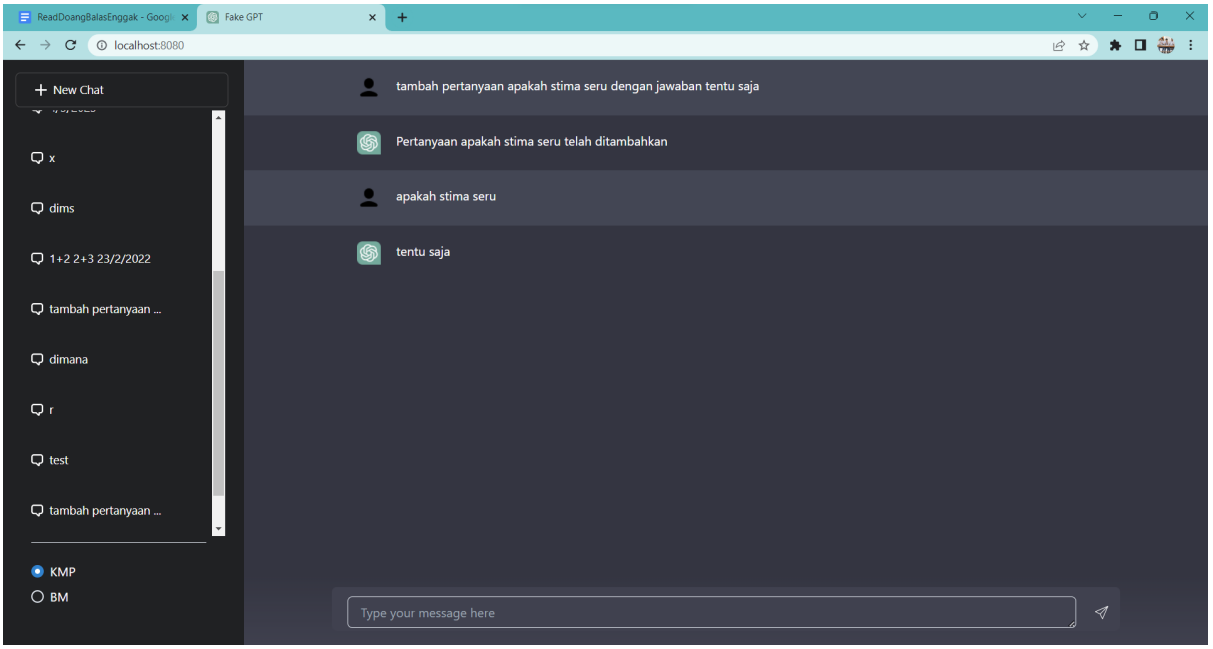


Program ini memiliki beberapa fitur yang tersedia untuk membantu pengguna dalam menemukan jawaban atas pertanyaan mereka. Fitur-fitur ini termasuk:

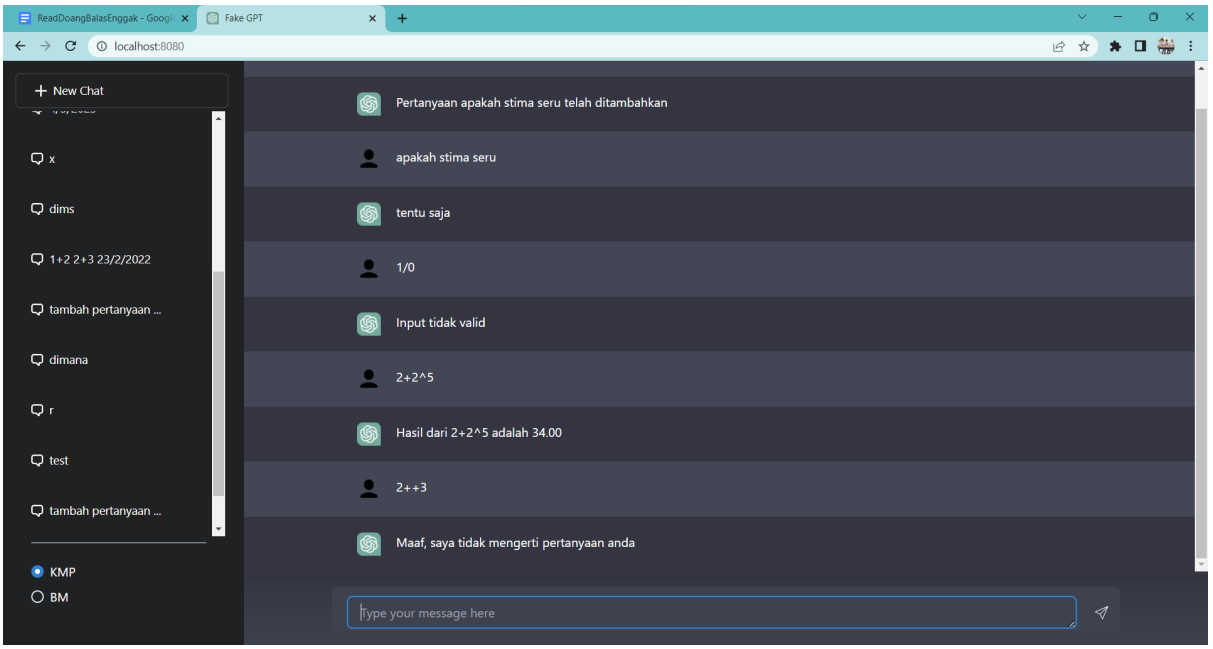
1. Fitur pertanyaan teks: pengguna dapat mengetik pertanyaan apapun yang terkait dengan topik tertentu dan chatbot akan mencoba memberikan jawaban yang sesuai.
2. Fitur tanggal: pengguna dapat mengetik pertanyaan terkait tanggal dan chatbot akan memberikan jawaban yang sesuai dengan pertanyaan tersebut.
3. Fitur kalkulator: pengguna dapat melakukan perhitungan matematika dengan mengetik pertanyaan ke kotak *input*.
4. Fitur tambah pertanyaan dan jawaban ke database: pengguna dapat menambahkan pertanyaan dan jawaban baru ke database dengan mengetik "tambah pertanyaan" dan mengikuti instruksi selanjutnya.
5. Fitur hapus pertanyaan: pengguna dapat menghapus pertanyaan tertentu dari database dengan mengetik "hapus pertanyaan" dan mengikuti instruksi selanjutnya.
6. Untuk memulai menggunakan program, pengguna hanya perlu membuka halaman web aplikasi dan mengetik pertanyaan mereka ke kotak *input*. Setelah pertanyaan dikirim, ChatGPT akan memberikan jawaban terbaik berdasarkan pertanyaan yang diberikan. Pengguna juga dapat menggunakan fitur-fitur tambahan seperti tambah pertanyaan dan hapus pertanyaan untuk membantu meningkatkan database dan mengoptimalkan kinerja ChatGPT.

4.3 Hasil Pengujian

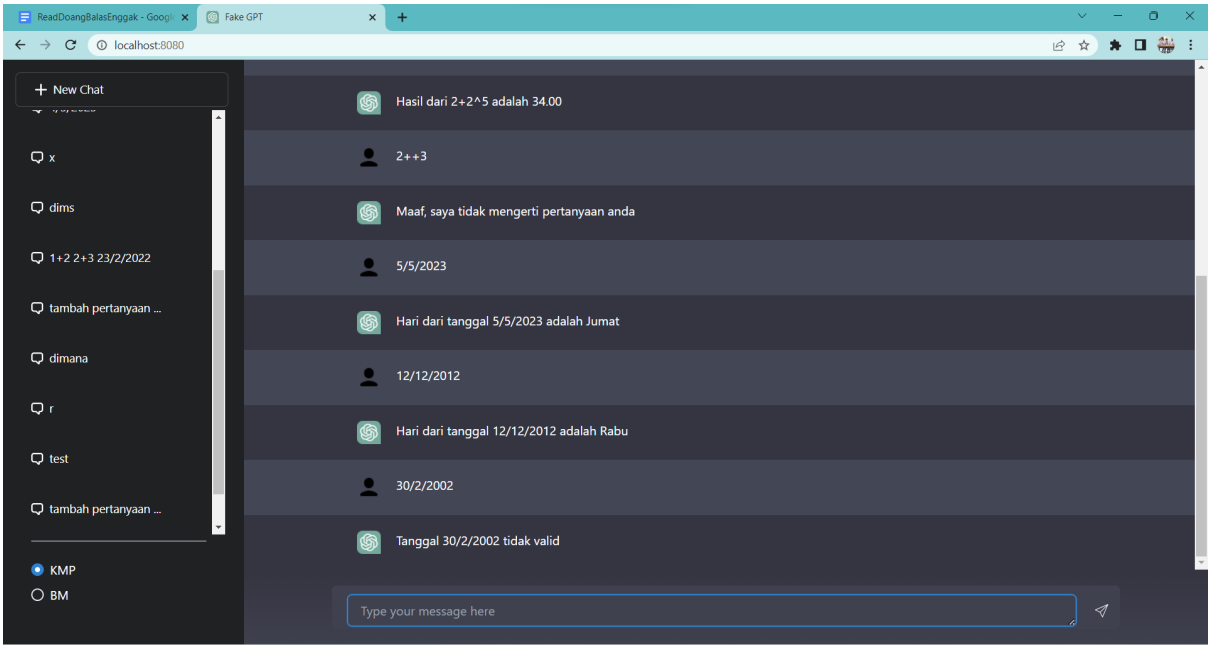
4.3.1 Pengujian Pertanyaan Teks



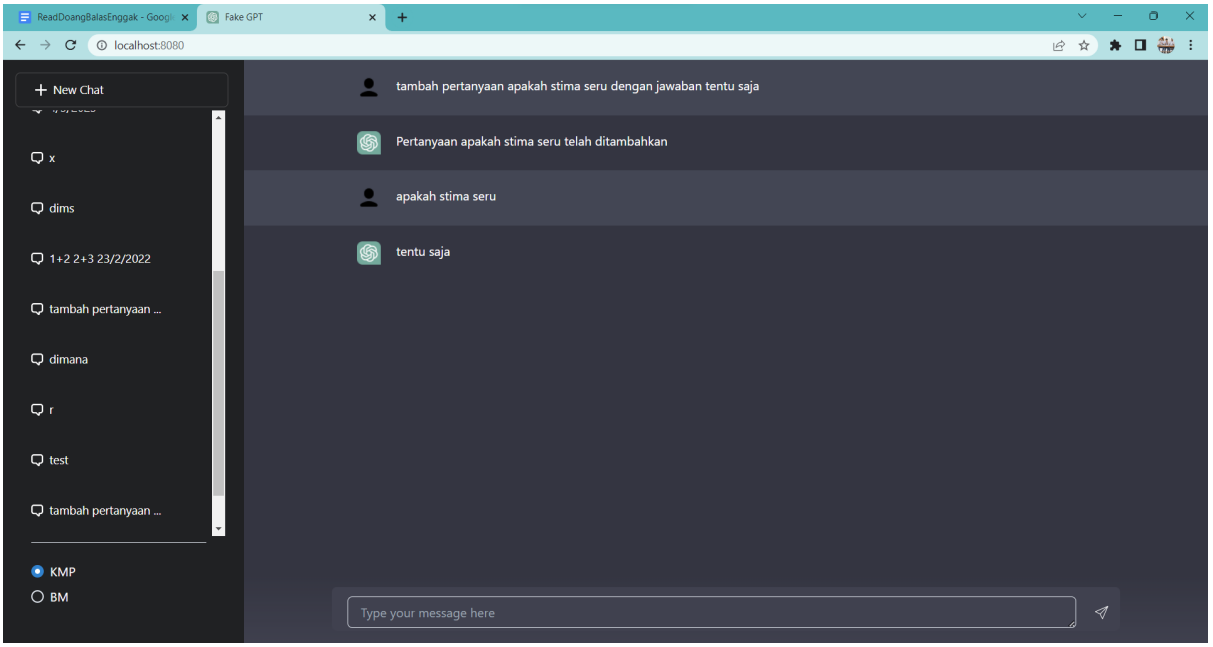
4.3.2 Pengujian Kalkulator



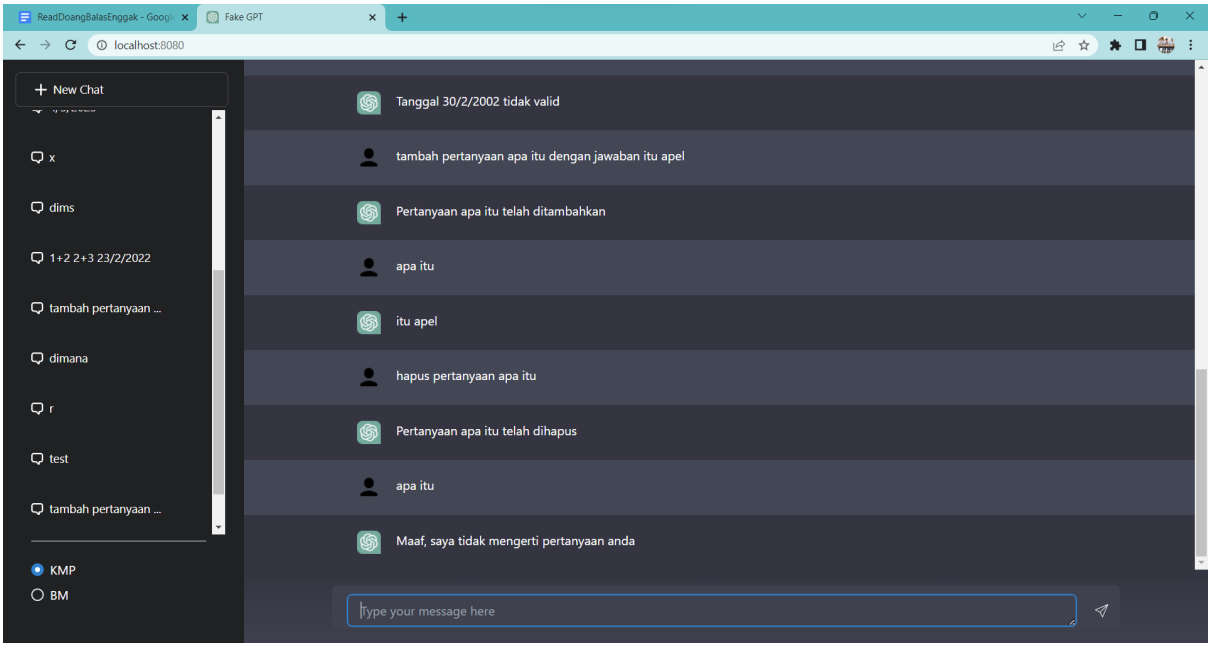
4.3.3 Pengujian Tanggal



4.3.4 Pengujian Tambah Pertanyaan dan Jawaban ke Database



4.3.5 Pengujian Hapus pertanyaan dari database



4.4 Analisis Hasil Pengujian

Penggunaan algoritma pencocokan string Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM) dalam melakukan pencarian pertanyaan pada database cukup efektif. Meskipun performa dari kedua algoritma ini tergantung pada panjang dari pola atau teks yang dicocokkan, namun secara keseluruhan, hasil pengujian menunjukkan bahwa aplikasi web ChatGPT mampu memberikan jawaban yang akurat dengan waktu respon yang cepat.

Aplikasi web ChatGPT mampu membantu pengguna dalam mencari jawaban atas pertanyaan yang mereka ajukan. Dengan menggunakan pendekatan *question answering* (QA) yang sederhana dengan memanfaatkan algoritma pencocokan string seperti Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM), aplikasi mampu menemukan pertanyaan yang paling mirip dengan pertanyaan yang diajukan oleh pengguna.

Selain itu, aplikasi ChatGPT dilengkapi dengan berbagai fitur seperti pertanyaan teks, fitur tanggal, fitur kalkulator, fitur tambah pertanyaan dan jawaban ke database, serta fitur hapus pertanyaan. Jika tidak ditemukan pertanyaan yang *exact match* dengan algoritma KMP atau BM akan dilakukan pencarian menggunakan metode Levenshtein Distance untuk menghitung tingkat kemiripan antara pertanyaan pengguna dan pertanyaan yang ada di database.

Penggunaan algoritma pencocokan string seperti KMP dan BM telah terbukti efektif dalam menemukan pertanyaan yang paling mirip dengan yang diberikan pengguna. Selain itu, fitur pencarian tanggal dan kalkulator juga berjalan dengan baik dan memberikan hasil yang akurat. Levenshtein Distance sangat membantu dalam menemukan jawaban yang sesuai dengan pertanyaan pengguna. Namun, pada beberapa kasus, meskipun pertanyaan pengguna

cukup mirip dengan pertanyaan di database, aplikasi masih belum dapat memberikan jawaban yang akurat.

Dengan fitur-fiturnya, aplikasi ini dapat membantu pengguna dengan cepat dan mudah menemukan jawaban atas pertanyaan mereka. Jika pertanyaan pengguna tidak ditemukan secara tepat, aplikasi akan mencari pertanyaan dengan tingkat kemiripan tertentu dan mengembalikan maksimal tiga jawaban terbaik yang mirip. Oleh karena itu, aplikasi ChatGPT dapat menjadi solusi yang baik bagi pengguna yang membutuhkan bantuan dalam mencari jawaban atas pertanyaan mereka secara cepat dan mudah.

Bab V

Kesimpulan

5.1 Kesimpulan

- *String Matching* dan *Regular Expression* dapat digunakan untuk memproses teks yang dimasukkan pengguna pada ChatGPT Sederhana.
- Algoritma Knuth-Morris-Pratt (KMP) dan Boyer Moore (BM) adalah dua metode *string matching* yang efektif digunakan untuk membandingkan teks *input* pengguna dengan teks yang tersimpan dalam database untuk menemukan jawaban yang tepat.
- Algoritma Levenshtein dapat digunakan untuk memperbaiki *input* pengguna yang salah ketik dan menemukan jawaban yang paling mirip dengan *input* pengguna yang sebenarnya.
- Dalam pengembangan ChatGPT Sederhana, penanganan *input* pengguna yang salah format juga penting dan dapat dilakukan dengan menggunakan *regular expression*.

5.2 Saran

- Dalam penggunaan algoritma Knuth-Morris-Pratt (KMP), Boyer Moore, dan Levenshtein, perlu dilakukan evaluasi performa pada skenario yang lebih kompleks untuk mengetahui keefektifan penggunaan algoritma tersebut.
- Pengembangan lebih lanjut pada ChatGPT sederhana dapat dilakukan dengan mengimplementasikan teknik *Machine Learning* seperti *Deep Learning* pada model untuk meningkatkan kemampuan dalam memproses bahasa alami pengguna.

5.3 Refleksi

- Dalam proses pembuatan ChatGPT sederhana, penggunaan teknik *String Matching* dan *Regular Expression* sangat membantu dalam mengolah *input* pengguna dan memberikan respons yang sesuai.
- Implementasi algoritmaKnuth-Morris-Pratt (KMP), Boyer Moore (BM), dan Levenshtein pada ChatGPT sederhana memberikan gambaran pada pentingnya penggunaan algoritma String Matching dalam pemrosesan teks.
- Dalam pengembangan selanjutnya, akan menjadi tantangan untuk meningkatkan kemampuan ChatGPT sederhana dalam memproses bahasa alami pengguna dengan hasil yang lebih akurat dan responsif.

Daftar Pustaka

[1] Munir, Rinaldi. 2021.Pencocokan String (String/Pattern Matching).
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf
[2] Munir, Rinaldi.2019. String Matching dengan Regular Expression.
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021

Lampiran

Link repository: https://github.com/vierifirdaus/TUBES3_STIMA.git
Link hasil *deploy* website: tubes3_stima.angkutin.my.id/
Link youtube: https://youtu.be/fxC5mxHHao0