LAPORAN TUGAS BESAR

IF2211 - STRATEGI ALGORITMA

Penerapan String Matching dan Regular Expression dalam Pembuatan ChatGPT Sederhana

Kelas Mahasiswa K03

Dosen Pengampu: Ir. Rila Mandala, M.Eng., Ph.D.

Disusun Oleh:

Kelompok 02 - ChatHaidar

Haikal Ardzi Shofiyyurrohman	**13521012**
Hidayatullah Wildan Ghaly Buchary	**13521015**
Muhammad Haidar Akita Tresnadi	**13521025**

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2023

DAFTAR ISI

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.

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.

Dalam tugas besar ini, anda diminta untuk membangun sebuah aplikasi ChatGPT sederhana dengan mengaplikasikan pendekatan QA yang paling sederhana tersebut. Pencarian pertanyaan yang paling mirip dengan pertanyaan yang diberikan pengguna dilakukan dengan algoritma pencocokan string Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM). Regex digunakan untuk menentukan format dari pertanyaan (akan dijelaskan lebih lanjut pada bagian fitur aplikasi). Jika tidak ada satupun pertanyaan pada database yang exact match dengan pertanyaan pengguna melalui algoritma KMP ataupun BM, maka gunakan pertanyaan termirip dengan kesamaan setidaknya 90% Apabila tidak ada pertanyaan yang kemiripannya di atas 90%, maka chatbot akan memberikan maksimum 3 pilihan pertanyaan yang paling mirip untuk dipilih oleh pengguna. Perhitungan tingkat kemiripan dibebaskan kepada anda asalkan dijelaskan di laporan, namun disarankan menggunakan salah satu dari algoritma Hamming Distance, Levenshtein Distance, ataupun Longest Common Subsequence.

BAB II
LANDASAN TEORI

2.1 KMP (Knuth-Morris-Pratt) Algorithm

Algoritma KMP adalah sebuah algoritma pencocokan pola yang digunakan untuk mencari kemunculan suatu pola tertentu dalam sebuah teks. Dalam algoritma KMP, sebuah tabel prefix dibuat terlebih dahulu dari pola yang akan dicari. Tabel prefix ini digunakan untuk mempercepat pencocokan pola pada tahap pencarian.

Pada tahap pencarian, algoritma KMP memindai teks secara bertahap dan membandingkan karakter-karakter yang ada pada teks dengan karakter-karakter pada pola. Pencocokan dimulai dari karakter pertama pada teks dan karakter pertama pada pola. Jika kedua karakter tersebut sama, maka lanjutkan pencocokan ke karakter berikutnya.

Jika terdapat perbedaan antara karakter pada teks dan karakter pada pola pada posisi tertentu, maka tabel prefix akan digunakan untuk menentukan posisi selanjutnya pada pola yang harus dibandingkan dengan karakter pada teks.

Proses pencarian diulang hingga seluruh teks telah dipindai. Jika selama proses pencarian terdapat satu atau lebih kemunculan pola pada teks, maka algoritma KMP akan mengembalikan posisi-posisi pada teks di mana pola ditemukan.

2.2 BM (Boyer-Moore) Algorithm

Algoritma BM adalah salah satu algoritma pencocokan string yang digunakan untuk mencari sebuah pola dalam sebuah teks. Algoritma ini menggunakan dua tabel yaitu tabel bad-character dan tabel good-suffix untuk mempercepat proses pencarian. Tabel bad-character berisi pergeseran yang harus dilakukan pada pola jika ditemukan sebuah karakter yang tidak cocok pada teks. Sedangkan tabel good-suffix berisi pergeseran yang harus dilakukan pada pola jika sebuah substring dari pola cocok dengan sebuah substring di dalam teks, tetapi karakter di sebelah kiri substring tersebut tidak cocok.

Algoritma BM dimulai dengan membandingkan pola dengan teks dari kanan ke kiri, dan menggunakan tabel bad-character dan good-suffix untuk mengurangi jumlah perbandingan karakter antara pola dan teks. Jika ada karakter pada pola yang tidak cocok dengan karakter pada teks, maka pola akan dipindahkan ke kanan sesuai dengan pergeseran maksimum dari tabel bad-character atau good-suffix. Jika substring dari pola cocok dengan sebuah substring di dalam teks, tetapi karakter di sebelah kiri substring tersebut tidak cocok, maka pola akan dipindahkan ke kanan sesuai dengan pergeseran dari tabel good-suffix.

Pencarian dilakukan hingga pola cocok dengan sebuah substring di dalam teks atau sampai pola tidak dapat dipindahkan lagi. Jika pola cocok, algoritma akan mengembalikan indeks awal pola di dalam teks. Namun, jika pola tidak ditemukan, algoritma akan memberikan hasil pencarian negatif.

Algoritma BM sangat efisien dalam pencarian string dan sering digunakan dalam aplikasi yang memerlukan pencarian string, seperti pencarian teks dalam editor teks atau dalam aplikasi web. Dengan menggunakan tabel bad-character dan good-suffix, algoritma BM dapat mempercepat proses pencarian dan mengurangi jumlah perbandingan karakter antara pola dan teks.

2.3 Penjelasan Aplikasi Web yang Dibangun

Aplikasi web yang dibangun menggunakan ReactJS dan NodeJS. Aplikasi yang kami bangun merupakan aplikasi dengan fitur *string matching* (mirip seperti chatGPT) yang mampu merespon masukkan *user* dengan jawaban yang diharapkan oleh *user* berdasarkan *database* yang dimiliki oleh aplikasi. Aplikasi ini juga mampu tepat merespon jawaban dari masukkan *user* jika masukkan user memiliki kemiripan 90% ke atas dari yang ada di database serta memberikan sugesti pertanyaan sebenarnya jika kemiripan pertanyaan/masukkan di bawah 90%. Selain itu, aplikasi ini juga memiliki fitur kalkulator, fitur menanyakan hari pada tanggal kapanpun, serta fitur menambah-menghapus pertanyaan-jawaban pada database.

BAB III
ANALISIS PEMECAHAN MASALAH

3.1 Langkah Penyelesaian Masalah Setiap Fitur

3.1.1 Fitur pertanyaan teks

Pada fitur pertanyaan teks langkah pertama ialah memastikan bahwa teks yang dimasukkan oleh user bukan *template* teks untuk fitur lain dengan menggunakan *regular expression*. Lalu mengecek teks tersebut dengan algoritma *string matching* yang dipilih (Knuth-Morris-Pratt/Boyer-Moore) dengan cara mencari teks pada database yang sama persis dengan teks yang dimasukkan oleh user. Jika ada maka akan menampilkan jawaban dari pertanyaan tersebut, namun jika tidak ditemukan maka akan dilakukan pengecekan kemiripan dengan algoritma *Levenshtein Distance* untuk menampilkan jawaban dengan pertanyaan mirip di atas 90%, menampilkan sugesti 3 pertanyaan paling mirip yang dimaksud oleh *user*, atau memberikan keluaran bahwa pertanyaan tidak ada pada *database* aplikasi.

3.1.2 Fitur kalkulator

Pada fitur kalkulator digunakan *regular expression* "`/([\+\-\*\/\^\\%\|\&\(\)])/`" untuk mengecek teks yang diterima oleh *user*. Jika terdapat pola regEx tersebut maka dilakukan algoritma *calculateEquation(text)* untuk melakukan perhitungan matematika dan menampilkan jawaban berupa hasil dari pertanyaan matematik tersebut.

3.1.3 Fitur tanggal

Pada fitur tanggal, kami menggunakan perhitungan serta validasi secara manual terkait bulan, tanggal, serta tahun kabisat.

3.1.4 Tambah pertanyaan dan jawaban ke database

Pada fitur menambahkan pertanyaan, kami menggunakan format *regular expression* "`/^tambahkan pertanyaan\s*(.+)\s*dengan jawaban\s*(.+)\s*$/`". Setelah pertanyaan divalidasi, akan dilakukan pencarian di database mengenai pertanyaan terkait dan jika ada akan direplace jawabannya. Jika tidak ada maka pertanyaan baru akan ditambahkan.

3.1.5 Hapus pertanyaan dari database

Pada fitur menghapus pertanyaan, kami menggunakan format *regular expression* "`/^hapus pertanyaan\s*(.+)\s*$/`".

3.2 Fitur Fungsional dan Arsitektur Aplikasi Web yang dibangun

3.2.1 Fitur Fungsional

Aplikasi web yang dibangun menggunakan ReactJS dan NodeJS memiliki beberapa fitur fungsional, di antaranya adalah:

3.2.1.1 Login dan Autentikasi

Merupakan fitur yang memungkinkan pengguna untuk membuat akun, masuk, dan keluar dari aplikasi. Pengguna diperlukan untuk login sebelum memasuki aplikasi karena database history akan terkait pada akun pengguna.

3.2.1.2 MainPage

Merupakan fitur yang berfungsi untuk mengkoordinasi setiap perubahan yang terjadi pada contents dan sidebar. Fitur ini merupakan fitur yang bisa menghubungkan setiap kejadian yang sedang atau telah dialami oleh fitur Contents, fitur Sidebar, dan fitur lainnya. Dengan memanfaatkan fungsi useEffect yang telah ada pada framework ReactJS, bagian ini mampu untuk memberikan perubahan dan signal kepada komponen-komponen lain yang bersangkutan untuk memberikan pengalaman yang baik kepada pengguna. Fitur ini juga melakukan request ke database untuk melakukan load terhadap apa yang telah terjadi di akun pengguna.

3.2.1.3 Contents

Merupakan fitur yang berfungsi untuk mengkoordinasi setiap perubahan yang berkaitan dengan pesan oleh pengguna. Fitur ini merupakan salah satu fitur yang paling penting karena fitur inilah yang melakukan request pemanggilan algoritma BM atau KMP kepada backend. Dalam fitur ini, pesan dari hasil respon program dan juga input pengguna akan ditampilkan.

3.2.1.4 Sidebar

Merupakan fitur yang berfungsi untuk membentuk tombol yang nantinya akan digunakan untuk melakukan load riwayat pesan pengguna. Sidebar akan membentuk tombol baru setiap pengguna mengirim pesan baru pada fitur NewChat dan tombol ini berfungsi untuk mendapatkan pesan yang telah dimasukkan ke dalam database.

3.2.1.5 TextInput

Merupakan fitur yang bertanggung jawab atas masukan pengguna. Hasil dari masukan pengguna akan diolah dan diserahkan ke fitur Contents untuk nantinya dilakukan pemrosesan dengan backend.

3.2.1.6 Radio

Merupakan fitur yang berfungsi untuk memberikan pilihan kepada pengguna untuk memilih antara KMP atau BM untuk melakukan pencocokan string. Fitur ini memiliki default yaitu KMP. Setelah TextInput mengirimkan sinyal kepada fitur Contents, fitur Contents akan memanggil fitur Radio untuk mendapatkan informasi pilihan pengguna.

3.2.1.7 Delete

Merupakan fitur yang berfungsi untuk melakukan penghapusan terhadap seluruh percakapan yang telah dilakukan oleh pengguna dengan program. Fitur ini akan melakukan request pemanggilan fungsi yang ada di backend yang berfungsi untuk mengosongkan riwayat percakapan pengguna dengan program. Penghapusan ini dilakukan agar saat fitur MainPage melakukan load, akan hilang semua tombol pada fitur Sidebar dan pesan pada fitur Contents.

3.2.1.8 NewChat

Merupakan fitur yang berfungsi untuk memberikan informasi sederhana tentang tata cara penggunaan aplikasi ini. Selain itu, fitur ini juga berguna jika pengguna ingin membuat percakapan baru dengan program tanpa harus menghapus percakapan yang sudah ada.

3.2.2 Arsitektur Aplikasi Web

Aplikasi web yang dibangun menggunakan ReactJS dan NodeJS memiliki arsitektur yang terdiri dari beberapa komponen, di antaranya adalah:

3.2.2.1 Frontend

Merupakan komponen aplikasi yang terlihat oleh pengguna, dalam hal ini menggunakan ReactJS. ReactJS adalah library JavaScript yang populer dan sangat fleksibel untuk membangun antarmuka pengguna yang dinamis dan responsif. ReactJS memiliki kemampuan untuk mengubah tampilan halaman web secara real-time ketika ada perubahan data yang terjadi pada aplikasi, tanpa perlu me-refresh halaman secara keseluruhan. ReactJS juga dilengkapi dengan banyak komponen dan alat untuk memudahkan pengembangan aplikasi web, seperti routing, state management, dan styling.

3.2.2.2 Backend

Merupakan komponen aplikasi yang tidak terlihat oleh pengguna, dan digunakan untuk menjalakan proses yang diperlukan di sisi server, dalam hal ini menggunakan NodeJS. NodeJS adalah lingkungan runtime JavaScript yang memungkinkan developer untuk

menjalankan JavaScript di sisi server. NodeJS dikembangkan dengan tujuan untuk memudahkan pengembangan aplikasi web dengan arsitektur server-side yang efisien, scalable, dan dapat di-deploy dengan mudah. NodeJS juga memiliki banyak modul dan framework yang populer, seperti ExpressJS, yang membantu mempercepat proses pengembangan aplikasi web.

3.2.2.3 Database

Merupakan komponen yang digunakan untuk menyimpan data dari aplikasi, dalam hal ini menggunakan MySQL2. MySQL2 adalah modul NodeJS yang digunakan untuk menghubungkan aplikasi dengan database MySQL. MySQL adalah salah satu database management system yang paling populer di dunia, dan digunakan oleh banyak perusahaan dan aplikasi web. MySQL2 menyediakan API yang mudah digunakan dan efisien untuk mengakses dan memanipulasi data pada database MySQL.

3.2.2.4 Server

Merupakan komponen yang menghubungkan frontend, backend, dan database sehingga aplikasi dapat berjalan dengan semestinya. Dalam hal ini, aplikasi menggunakan Vercel untuk frontend, Railway untuk backend, dan PlanetScale untuk database.

Vercel adalah platform hosting untuk aplikasi web yang menggunakan ReactJS, NextJS, atau VueJS. Vercel menyediakan infrastruktur hosting yang scalable dan cepat, dengan dukungan untuk deployment otomatis dari repository Git. Dalam hal ini, Vercel digunakan untuk hosting frontend aplikasi web ini.

Railway adalah platform hosting untuk aplikasi web yang menggunakan NodeJS dan PostgreSQL atau MySQL. Railway menyediakan infrastruktur hosting yang scalable dan mudah digunakan, dengan dukungan untuk deployment otomatis dari repository Git. Dalam hal ini, Railway digunakan untuk hosting backend aplikasi web ini.

PlanetScale adalah platform cloud database yang menyediakan layanan database MySQL yang scalable dan highly available. PlanetScale menyediakan infrastruktur database yang handal dan efisien, dengan dukungan untuk replikasi dan failover otomatis. Dalam hal ini, PlanetScale digunakan untuk menyimpan data aplikasi web

BAB VI
IMPLEMENTASI DAN PENGUJIAN

4.1 Spesifikasi Teknis Program

Program menggunakan server.js sebagai file entry point dari backend. Pada server.js, terdapat kode yang menjalankan server HTTP menggunakan framework ExpressJS dan melakukan koneksi dengan database menggunakan library mysql2.

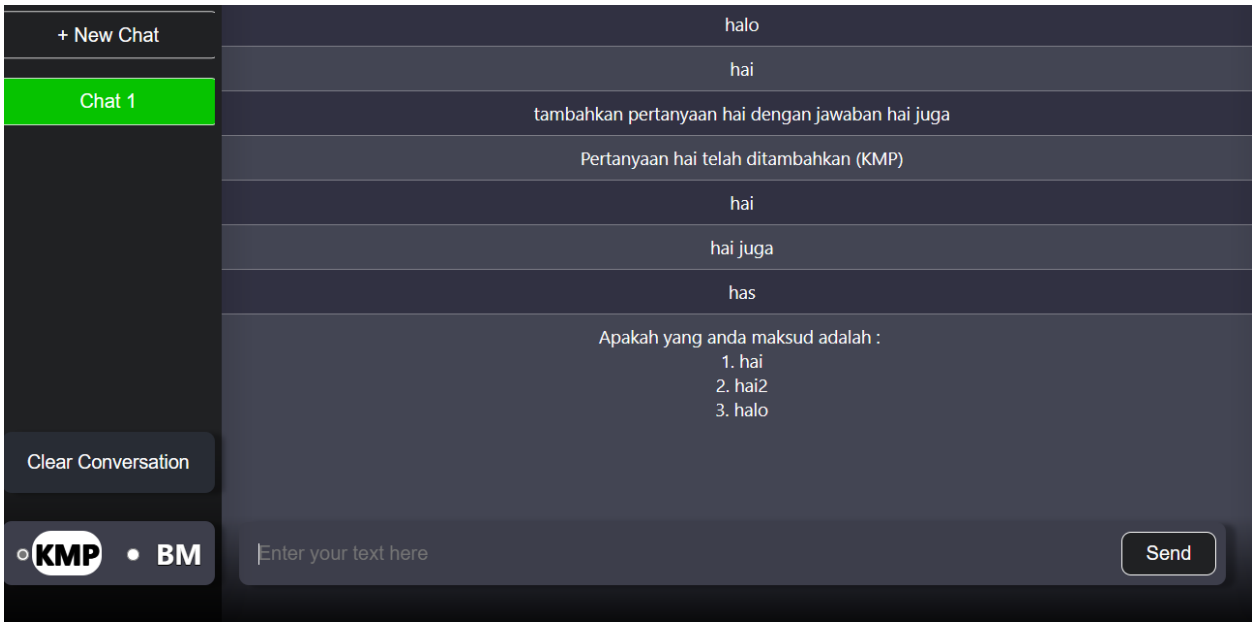
4.2 Tata Cara Penggunaan Program

Program telah di-deploy pada website https://chat-haidar.vercel.app/ sehingga dapat langsung diuji pada website tersebut. Jika ingin menjalankan program di localhost, langkah-langkah yang dapat diikuti adalah sebagai berikut:

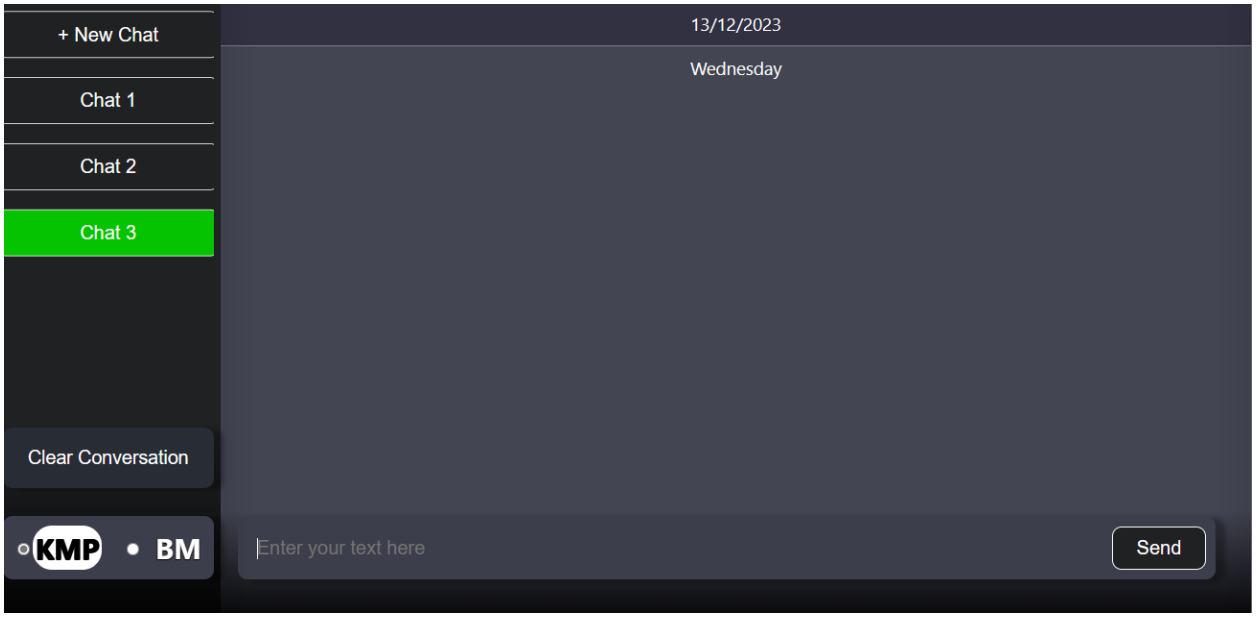
- Lakukan clone dari https://github.com/WildanGhaly/Tubes3_13521012.git
- Buka file Backend/server/db.js dan lakukan konfigurasi koneksi dengan server SQL yang diinginkan. Pastikan untuk mengisi informasi host, port, username, password, dan nama database dengan benar.
- Buka file Frontend/src/BackendCaller.js dan ubah konstanta menjadi https://localhost:8000
- Jalankan perintah npm install di folder Backend/server dan Frontend untuk menginstal semua dependensi yang dibutuhkan.
- Buka 2 terminal
- Pada terminal pertama, lakukan cd Backend/server dan jalankan perintah npm run dev untuk menjalankan server backend di localhost:8000.
- Pada terminal kedua, lakukan cd Frontend dan jalankan perintah npm start untuk menjalankan aplikasi frontend di localhost:3000.
- Program akan berjalan dan dapat diakses melalui browser dengan membuka alamat http://localhost:3000

4.3 Hasil Pengujian

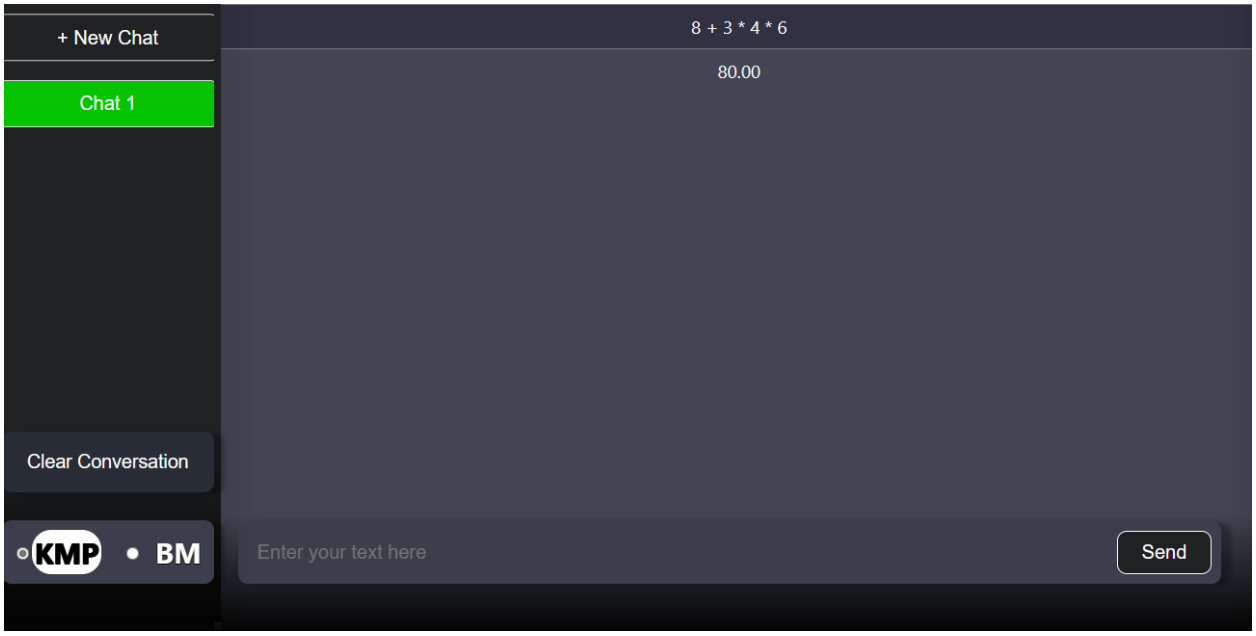
4.3.1 Fitur Pertanyaan Teks



4.3.2 Fitur tanggal



4.3.3 Fitur kalkulator



4.3.4 Fitur Menambahkan pertanyaan dan menghapus pertanyaan

4.4 Analisis Hasil Pengujian

Berdasarkan hasil uji yang kami lakukan, algoritma KMP maupun BM dalam kasus mencari match string pada sekumpulan pertanyaan di database memiliki hasil yang sama. Meskipun memiliki hasil yang sama, jika dianalisis lebih lanjut pada aspek efisiensi waktu,

algoritma KMP biasanya lebih cepat daripada algoritma BM pada teks dengan panjang yang pendek dan pola yang panjang. Namun, algortima BM biasanya lebih cepat daripada algoritma KMP pada teks dengan panjang yang panjang dan pola yang pendek atau tidak sama sekali. Selain itu, algoritma BM juga memiliki keunggulan pada kasus terburuk, karena pada kasus terburuk, algoritma KMP memerlukan O(mn) waktu untuk memproses teks, sedangkan algoritma BM hanya memerlukan O(nm) waktu. Meskipun memiliki kekurangan, Algoritma KMP juga memiliki kelebihan dikarenakan mudahnya implementasi yang diperlukan. KMP juga memiliki kelebihan dalam alokasi memori karena algoritma KMP memerlukan memori yang lebih sedikit dibandingkan BM algorithm.

BAB V
KESIMPULAN, SARAN DAN REFLEKSI

5.1 Kesimpulan

Tugas besar program Penerapan String Matching dan Regular Expression dalam Pembuatan ChatGPT Sederhana yang mengaplikasikan algoritma KMP dan BM dalam mata kuliah IF2211 Strategi Algoritma telah berhasil dibuat. Melalui penerapan algoritma tersebut, program ini berhasil diselesaikan sesuai dengan ketentuan yang tertera pada spesifikasi. Dalam proses pembuatan program, kami memperoleh pemahaman yang lebih baik tentang kedua algoritma dan bagaimana cara mengaplikasikannya dalam konteks pengolahan string dan pemrograman.

Melalui penerapan KMP dan BM, program ini dapat mencari kemunculan suatu pola dalam teks yang dimasukkan oleh pengguna. Selain itu, program ini juga dapat memvalidasi regular expression yang dimasukkan oleh pengguna. Dengan demikian, program ini dapat membantu pengguna dalam memproses dan memanipulasi string dengan lebih mudah dan efisien.

5.2 Saran

Kami menyarankan agar pengembangan aplikasi ini fokus pada peningkatan tampilan dan kreativitas desain antarmuka pengguna (GUI). Hal ini diharapkan dapat meningkatkan pengalaman pengguna dalam menggunakan aplikasi.

5.3 Refleksi

Tugas Besar Strategi Algoritma kali ini menjadi pembelajaran yang sangat berharga bagi penulis, banyak hal baru yang dipelajari dalam pembuatan tugas besar ini. Meskipun terdapat beberapa kendala dalam pembuatan algoritma KMP dan BM, hal itu malah menjadi pacuan agar penulis lebih semangat dalam mengeksplorasi untuk menemukan solusi yang terbaik.

DAFTAR PUSTAKA

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex2019.pdf

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

Pranala

Github utama: https://github.com/WildanGhaly/Tubes3_13521012

Github backend: https://github.com/WildanGhaly/Tubes3_ChatHaidar_BE

Github frontend: https://github.com/WildanGhaly/Tubes3_ChatHaidar_FE

Website: https://chat-haidar.vercel.app/

Video: https://youtube.com/playlist?list=PLBZGPqArhCSZDREzgD7xFQRLYLRtJIp2Q