LAPORAN TUGAS BESAR
TUGAS BESAR III TAHUN 2022/2023
PENERAPAN STRING MATCHING DAN REGULAR EXPRESSION DALAM PEMBUATAN CHATGPT SEDERHANA

Kelompok ChatTerUlung:

Wilson Tansil	(13521054)
Mutawally Nawwar	(13521065)
Ulung Adi Putra	(13521122)

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2023

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 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”.

BAB 2
LANDASAN TEORI

Algoritma Knuth-Morris-Pratt adalah salah satu algoritma pencarian string, dikembangkan secara terpisah oleh Donald E. Knuth pada tahun 1967 dan James H. Morris bersama Vaughan R. Pratt pada tahun 1966, tetapi keduanya mempublikasikannya secara bersamaan pada tahun 1977. Algoritma Knuth-Morris-Pratt (KMP) mencari pola dalam teks dengan urutan dari kiri ke kanan (seperti algoritma brute force).

Algoritma KMP dalam bahasa Java

```
public static int kmpMatch(String text, String pattern)
{
      int n = text.length();
      int m = pattern.length();
      int b[] = computeBorder(pattern);
      int i=0;
      int j=0;
      while (i < n) {
            if (pattern.charAt(j) == text.charAt(i)) {
                  if (j == m - 1) {
                        return i - m + 1; // match
                        i++;
                        j++;
                  }
            }
            else if (j > 0)
                  j = b[j-1];
            else
                  i++;
      }
      return -1; // no match
}// end of kmpMatch()

public static int[] computeBorder(String pattern)
{
      int b[] = new int[pattern.length()];
      fail[0] = 0;
      int m = pattern.length();
      int j = 0;
      int i = 1;
      while (i < m) {
      if (pattern.charAt(j) == pattern.charAt(i)) {
            //j+1 chars match
            b[i] = j + 1;
            i++;
            j++;
      }
      else if (j > 0) // j follows matching prefix
            j = b[j-1];
```

```
      else { // no match
            b[i] = 0;
            i++;
            }
      }
      return fail;
}// end of computeBorder()
```

Algoritma Boyer-Moore adalah salah satu algoritma pencarian string, dipublikasikan oleh Robert S. Boyer, dan J. Strother Moore pada tahun 1977. Algoritma ini dianggap sebagai algoritma yang paling efisien pada aplikasi umum. Tidak seperti algoritma pencarian string yang ditemukan sebelumnya, algoritma Boyer-Moore mulai mencocokkan karakter dari sebelah kanan pattern. Ide di balik algoritme ini adalah bahwa dengan memulai pencocokan karakter dari kanan, dan bukan dari kiri, maka akan lebih banyak informasi yang didapat.

Algoritma pencocokan pola Boyer-Moore didasarkan pada dua teknik.

1. The looking-glass technique
   - Temukan P di T dengan bergerak mundur melalui P, mulai dari ujungnya.
2. The character-jump technique
   - Ketika ketidaksesuaian terjadi pada T[i] == x.
   - Karakter pada pola P[j] tidak sama dengan T[i].

Ada 3 kemungkinan kasus (berurutan)

1. Kasus 1

Jika P berisi x di suatu tempat, maka coba geser P ke kanan untuk menyejajarkan kejadian terakhir x di P dengan T[i].

2. Kasus 2

Jika P berisi x di suatu tempat, tetapi pergeseran ke kanan ke kejadian terakhir tidak dimungkinkan, maka geser P ke kanan sebanyak 1 karakter ke T[i+1].

3. Kasus 3

Jika kasus 1 dan 2 tidak berlaku, geser P untuk menyejajarkan P[0] dengan T[i+1].

Algoritma BM dalam bahasa Java

```
public static int bmMatch(String text, String pattern)
{
      int last[] = buildLast(pattern);
      int n = text.length();
      int m = pattern.length();
      int i = m-1;
      if (i > n-1)
            return -1; // no match if pattern is
            // longer than text
      int j = m-1;
      do {
```

```
            if (pattern.charAt(j) == text.charAt(i))
                  if (j == 0)
                        return i; // match
                  else { // looking-glass technique
                        i--;
                        j--;
                  }
            else { // character jump technique
                  int lo = last[text.charAt(i)]; //last occ
                  i = i + m - Math.min(j, 1+lo);
                  j = m - 1;
            }
      } while (i <= n-1);
      return -1; // no match
} // end of bmMatch()

public static int[] buildLast(String pattern)
/* Return array storing index of last occurrence of each ASCII char
in pattern. */
{
      int last[] = new int[128]; // ASCII char set
      for(int i=0; i < 128; i++)
            last[i] = -1; // initialize array
      for (int i = 0; i < pattern.length(); i++)
            last[pattern.charAt(i)] = i;
      return last;
} // end of buildLast()
```

Regex, singkatan dari Regular Expression, adalah serangkaian pola atau aturan yang digunakan untuk mencocokkan, menemukan, dan mengganti teks berdasarkan pola tertentu. Dalam konteks pencocokan string, pola ini digunakan untuk membandingkan teks input dengan pola atau format yang diinginkan.

Regex umumnya digunakan untuk memanipulasi teks atau data string dalam berbagai aplikasi, seperti pengolahan teks, analisis log, pengecekan validitas input, dan lain-lain.

Berikut adalah *regex expression*

`.`	Any character except newline.
`\.`	A period (and so on for `\*`, `\(`, `\\`, etc.)
`^`	The start of the string.
`$`	The end of the string.
`\d,\w,\s`	A digit, word character `[A-Za-z0-9_]`, or whitespace.
`\D,\W,\S`	Anything except a digit, word character, or whitespace.
`[abc]`	Character a, b, or c.
`[a-z]`	a through z.
`[^abc]`	Any character except a, b, or c.
`aa|bb`	Either aa or bb.
`?`	Zero or one of the preceding element.
`*`	Zero or more of the preceding element.
`+`	One or more of the preceding element.
`{n}`	Exactly *n* of the preceding element.
`{n,}`	*n* or more of the preceding element.
`{m,n}`	Between *m* and *n* of the preceding element.
`??,*?,+?, {n}?`, etc.	Same as above, but as few as possible.
`(expr)`	Capture *expr* for use with `\1`, etc.
`(?:expr)`	Non-capturing group.
`(?=expr)`	Followed by *expr*.
`(?!expr)`	Not followed by *expr*.

Near-complete reference

2.2 PENJELASAN SINGKAT MENGENAI APLIKASI WEB YANG DIBANGUN

Aplikasi web adalah program komputer yang diakses melalui browser web seperti Google Chrome, Mozilla Firefox, atau Safari. Aplikasi web ini menggunakan teknologi web seperti HTML, CSS, dan JavaScript untuk membuat tampilan dan interaksi antara pengguna dan server. Biasanya, aplikasi web ini disimpan pada server dan diakses oleh pengguna melalui internet.

Contohnya, jika kamu menggunakan aplikasi web email seperti Gmail, kamu dapat mengakses email kamu dengan membuka browser web dan masuk ke halaman Gmail. Di sana kamu dapat membaca, menulis, dan mengirim email, serta melakukan berbagai tindakan lainnya. Begitu juga dengan aplikasi web lain seperti e-commerce, media sosial, dan lain-lain.

Aplikasi web memungkinkan pengguna untuk mengakses informasi dan melakukan tindakan dari mana saja dengan koneksi internet, tanpa harus menginstal program atau aplikasi khusus di komputer atau perangkat seluler mereka. Ini membuat aplikasi web lebih mudah diakses dan lebih mudah di-update secara berkala oleh pembuatnya.

Pada tugas besar kali ini, aplikasi web dibangun menggunakan javascript dengan react. Dengan menggunakan teknologi ini, pegembang dapat membuat aplikasi web yang terstruktur dan modular

BAB 3
ANALISIS PEMECAHAN MASALAH

3.1 LANGKAH PENYELESAIAN MASALAH SETIAP FITUR

Penyelesaian persoalan “Pembuatan chatGPT sederhana” diimplementasikan dengan beberapa pendekatan algoritma pattern matching yaitu:

a. Penyelesaian dengan KMP

Pada algoritma KMP yang diimpementasikan, terdapat fungsi computeBorderFunction yang berfungsi untuk menentukan indeks dimana patern akan mengulangi pencocokan jika terjadi mismatch. Pada saat fungsi pencocokan text dengan patern dengan algoritma KMP dimulai, akan dipanggil fungsi computeBorderFunction terlebih dahulu yang hasilnya akan disimpan dalam list yang bernama borders.

Setelah itu akan dilakukan iterasi dari sebanayak panjang dari text. Insialisasi variabel i yang merepresentasikan indeks dimana char pada text sedang dievaluasi dan j yang merepresentasikan indeks dimana char pada patern sedang dievaluasi. Pada tiap iterasi akan dicocokan char pada text dengan indeks i dan char pada patern dengan indeks j. Jika tidak terjadi mismatch atau char tersebut sama, maka indeks i dan j akan ditambah 1. Jika terjadi mismatch, maka indeks i tetap akan ditambah 1, namun indeks j akan diubah nilainya menjadi nilai borders[j-1]. Jika nilai j sama dengan panjang patern, maka patern dan text cocok.

b. Penyelesaian dengan BM

Pada algoritma BM yang diimplementasikan, terdapat dua marker yakni marker yang mencocokan string dengan pattern dan marker yang digunakan sebagai acuan. Dalam hal BM, akan dilakukan jump ke posisi yang sesuai dan dikategorikan menjadi 3 buah jenis lompatan dengan kasus tertentu.

Pada kasus pertama, apabila string dan pattern pada indeks tertentu dinyatakan tidak sama maka program akan mencari apakah terdapat pattern di bagian prefix yang sama dengan string pada indeks tersebut. Apabila iya maka marker acuan akan digeser sebesar perbedaan index pattern yang terhenti (karena tidak sama dengan string) dengan index char terakhir pada prefix sebelum index pattern yang terhenti.

Pada kasus kedua, apabila char (char pada string yang tidak sama dengan char pattern) terletak pada sebelah kanan indeks pattern yang tidak sama, maka maker acuan akan digeser sebanyak satu

Pada kasus ketiga, apabila tidak terdapat char (char pada string yang tidak sama dengan char pattern) tidak terdapat pada pattern, maka dapat dipastikan bahwa string pada blok tersebut tidak akan terdapat pattern. Maka dari itu, marker acuan akan digeser sebesar panjang pattern itu sendiri.

Pada pencocokan string dengan pattern algoritma BM akan mencocokannya dari index paling akhir pattern dan blok string.

c. Penyelesaian dengan LevenshteinDistance

Levenshtein distance adalah algoritma yang digunakan untuk mengetahui persentase kemiripan antara 2 string. Pada tubes ini, implementasi algoritma levenshtein distance menggunakan package dari "github.com/agnivade/levenshtein"

Fitur Kalkulator akan memberikan penyelesaian dari equation yang diberikan oleh user. Terdapat beberapa tipe dari respon program yaitu “Invalid Syntax” dan jawaban yang sebenarnya. Fitur ini akan terlebih dahulu memvalidasi equation yang diberikan. Hal ini mungkin saja bentrok dengan tanggal yang memiliki syntax “31/2/2022”, maka dari itu harus diperjelas dengan menambahkan kata Equation atau equation pada saat bertanya. Setelah melakukan validasi maka akan dikembalikan jawaban dari persamaan yang ditanya user.

Fitur Tanggal akan memberikan hari dari tanggal yang ditanyakan. Terdapat beberapa tipe dari respon program yaitu “Invalid Date” dan jawaban yang sebenarnya. Fitur ini akan terlebih dahulu memvalidasi tanggal yang diberikan. Hal ini mungkin saja bentrok dengan fitur kalkulator yang memiliki syntax “31/2/2022”, maka dari itu harus diperjelas dengan menambahkan kata Date, Day, date atau day pada saat bertanya. Setelah melakukan validasi maka akan dikembalikan jawaban dari tanggal yang ditanya oleh user.

Fitur Penambahan QnA adalah fitur yang digunakan untuk menambah pertanyaan dengan jawaban kedalam database. Fitur ini dapat diakses dengan cara mengetikan query “tambah pertanyaan x dengan jawaban y”, dimana x adalah pertanyaan yang ingin dimasukkan, dan y adalah jawaban dari pertanyaan x. Pada fitur ini bisa terdapat 2 respon yang diberikan oleh bot. Respon pertama adalah pertanyaan sukses dimasukkan kedalam database dan respon kedua adalah jika pertanyaan sudah ada didalam database.

Fitur Menghapus QnA adalah fitur yang digunakan untuk menaspu pertanyaan yang ada didatabase. Fitur ini dapat diakses dengan cara mengetikan query ”hapus pertanyaan x“, dimana x adalah pertanyaan yang ingin dihapus oleh pengguna. Fitur ini dapat memberikan 2 respon. Respon pertama adalah jika pertanyaan berhasil dihapus dalam database, dan respon kedua adalah jika pertanyaan memang tidak ada dalam database.

3.2 FITUR FUNGSIONAL DAN ARSITEKTUR APLIKASI WEB YANG DIBANGUN

a. Fitur Fungsional

- Program dapat menambahkan pertanyaan beserta jawaban ke basis data
- Program dapat menghapus pertanyaan dari basis data
- Program dapat menyimpan histori pertanyaan dari user dan dapat ditampilkan langsung dari web
- Program dapat menyimpan riwayat login dari user
- Program dapat menampilkan hari dari tanggal yang dipertanyakan
- Program dapat menampilkan jawaban hasil dari persamaan matematika
- Program akan mengembalikan pesan invalid apabila terdapat kesalahan pada pertanyaan

b. Arsitektur Aplikasi

- **FrontEnd**

Pada arsitektur frontend terdapat beberapa folder Pertama adalah folder component yang merupakan folder untuk menyimpan component yang dipakai seluruh feature. Kemudian folder feature dimana folder untuk menyimpan semua feature aplikasi. Pada frontend, main.tsx merupakan pintu utama dalam menjalankan aplikasi frontend.

- **Backend**
  Pada arsitektur backend program ini, terdapat beberapa folder yang dimana satu folder berisi satu package. Package yang dibuat pada bagian backend adalah regex, controller, models, feature, dan algorithm. Package regex berfungsi untuk mengklasifikasikan query berdasarkan regex yang telah dibuat. Package controller berfungsi untuk mengatur handler dari router router pada web. Package berisi tipe data struct yang dibutuhkan serta database. Package feature berisi fitur fitur untuk kalkulator dan menentukan hari. Dan package algorithm berisi algoritma KMP dan BM.

BAB 4

IMPLEMENTASI DAN PENGUJIAN

4.1 SPESIFIKASI TEKNIK PROGRAM

4.1.1 STRUKTUR DATA

Data pada tugas besar ini disimpan dalam bentuk dokumen menggunakan mongodb. Pada mongodb data disimpan dalam format dokumen yang dikenal sebagai BSON (Binary JSON). BSON adalah representasi biner dari JSON dan digunakan untuk menyimpan data dalam MongoDB. Setiap dokumen dalam MongoDB memiliki format yang sama dengan kunci dan nilai yang diperbolehkan untuk berbagai tipe data.

User

```
_id: ObjectId('64516f45721a2b7cd16aba94')
username: "paijem"
password: "12345"
```

QnA

```
_id: ObjectId('64481c378a5f0e16578d1c17')
question: "Tubes apa yang paling ga ngotak"
answer: "jelas stima dong"
```

History

```
_id: ObjectId('645269a2257a2a6781de0dac')
userid: ObjectId('6452677c257a2a6781de0d8b')
name: "history 1"
```

Chat

```
_id: ObjectId('645144138fa312ac62971b08')
historyid: ObjectId('64510e7323514c4e6b4f1691')
chat: "Apa Mata Kuliah paling seru di semester 4"
isbot: false
```

4.2.2 FUNGSI DAN PROSEDUR

No.	Nama Fungsi/Prosedur	Keterangan
1.	func checkNextJump(j int, currentChar string, pattern string) int	Mengembalikan berapa pelompatan marker pada algoritma BM
2.	func BoyerMoore(text, pattern string) int	Mengembalikan index pertama dari blok string yang sesuai dengan pattern
3.	func GetDay(dateString string) (string, error)	Mengembalikan day apabila date yang diberikan benar dan error untuk data yang salah
4.	func MathematicalOperationSolver(mathematicalExpression string) (float64, error)	Mengembalikan double apabila persamaan yang diberikan benar dan error untuk persamaan yang salah
5	func getDateQuery (query string) string	Mendapatkan date yang akan diproses
6	func getMathOperatorQuery(query string) string	Mendapatkan persamaan yang akan diproses
7	func isDateQuery(query string) (bool, bool)	Memvalidasi apakah terdapat command date yang exact atau terdapat date pada query
8	func isMathOprQuery(query string) (bool, bool)	Memvalidasi apakah terdapat command persamaan yang exact atau terdapat persamaan pada query
9	func QueryClassification (query string) string	Melakukan pemrosesan jawaban berdasarkan kategori masing-masing
10	func (p QueryProcessor) QuerySearch(method, query string) (int, float64)	Melakukan pencarian query pada database dan mereturn int yang merepresentasikan index qna ditemukan dan floatyang merepresentasikan similarity terbesar
11	func (p QueryProcessor) GetSimilarityList(query string) ([]float64)	Fungsi yang mereturn list of float yang merepresentasikan similarity query yang ada di database
12	func parseQuery(query string) (string, string)	Fungsi yang berguna untuk memisah query pada fitur tambah pertanyaan. Fungsi ini mereturn 2 string yang merupakan jawaban yang akan dimasukkan dan jawabannya
13	func getTop3Indexes(numbers []float64) []int	Mendapatkan index pada database dengan similarity tertinggi
14	func computeBorderFunction(pattern string) []int	Melakukan pencarian indeks dimulainya pencocokan

		patern jika terjadi mismatch
15	func KMP(pattern, text string) int	Mengembalikan index pertama dari blok string yang sesuai dengan pattern
16	func LevenshteinDistance (pattern string, text string) float64	Menghitung similarity antara string dan pattern

4.2 PENJELASAN TATA CARA PENGGUNAAN PROGRAM

a. Frontend

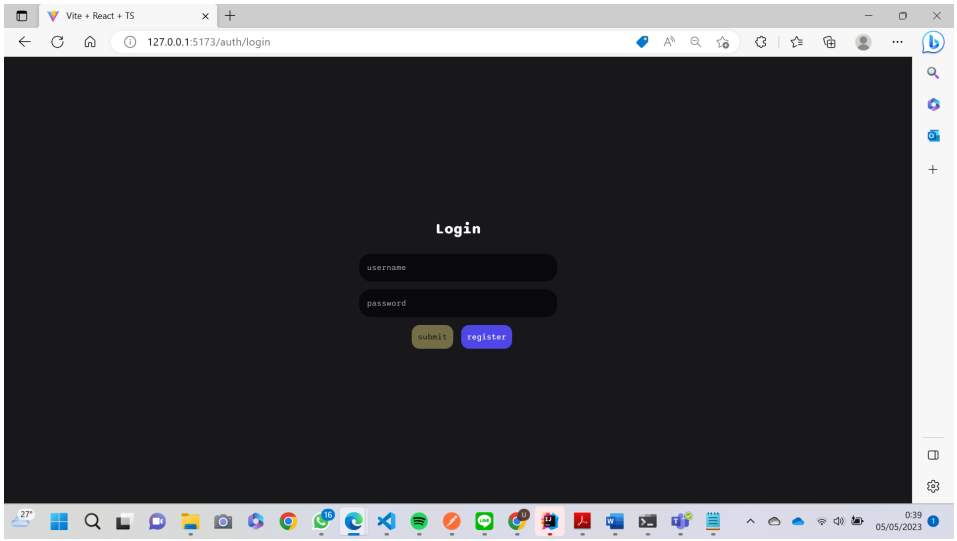
1. Install nodejs
2. Buka folder frontend
3. Jalankan “npm install”
4. Jalankan “npm run dev”

b. Backend

1. Install go
2. Buka folder backend
3. Jalankan “go run main.go”

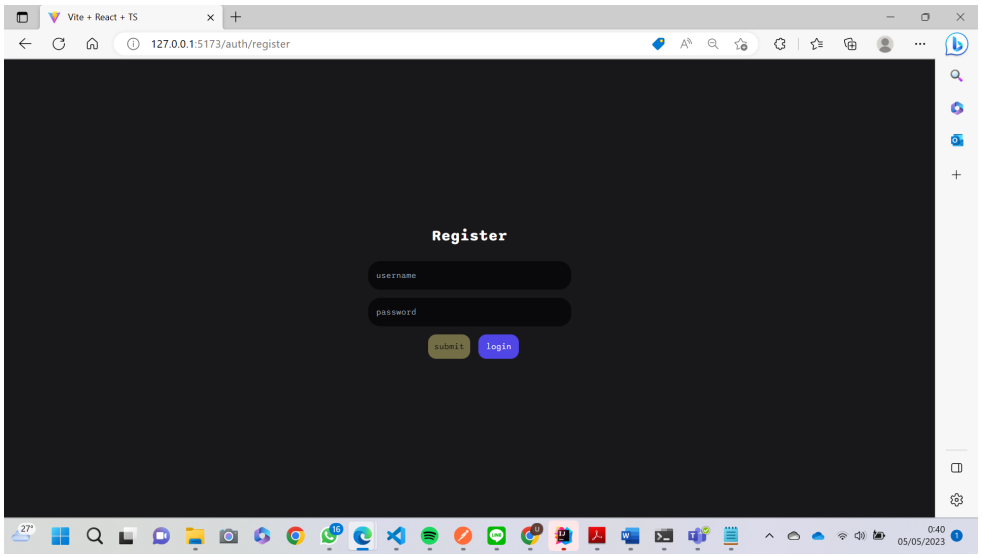
4.3 HASIL PENGUJIAN

1. Halaman Login



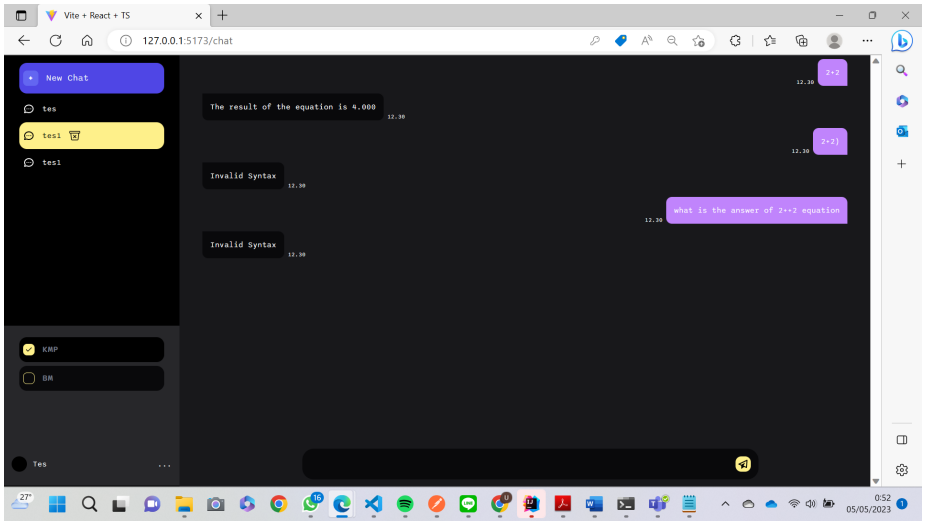
Gambar 4.3.1 Halaman login

2. Halaman Register



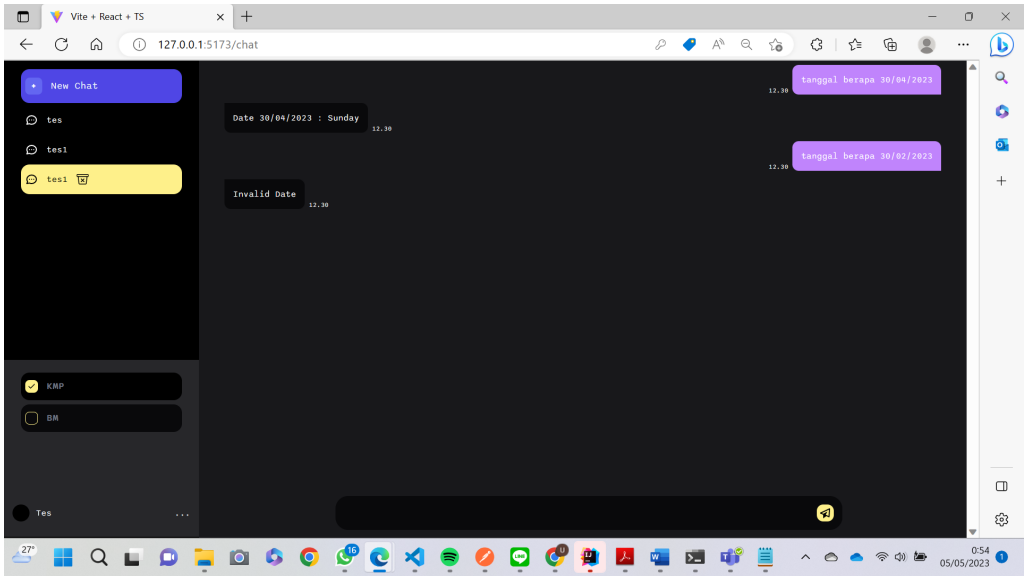
Gambar 4.3.2 Halaman Register

3. Fitur Kalkulator



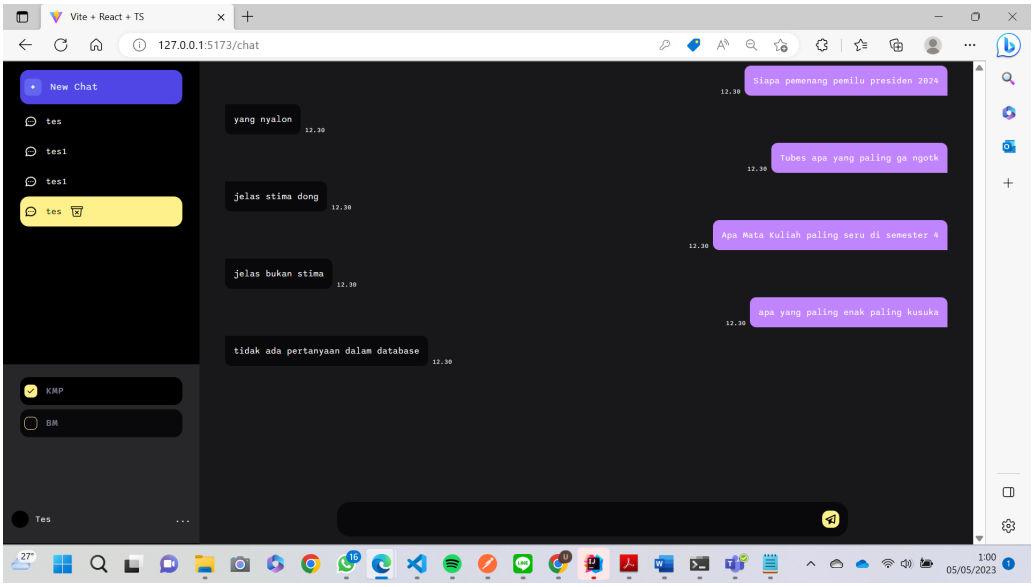
Gambar 4.3.3 Fitur Kalkulator

4. Fitur Tanggal



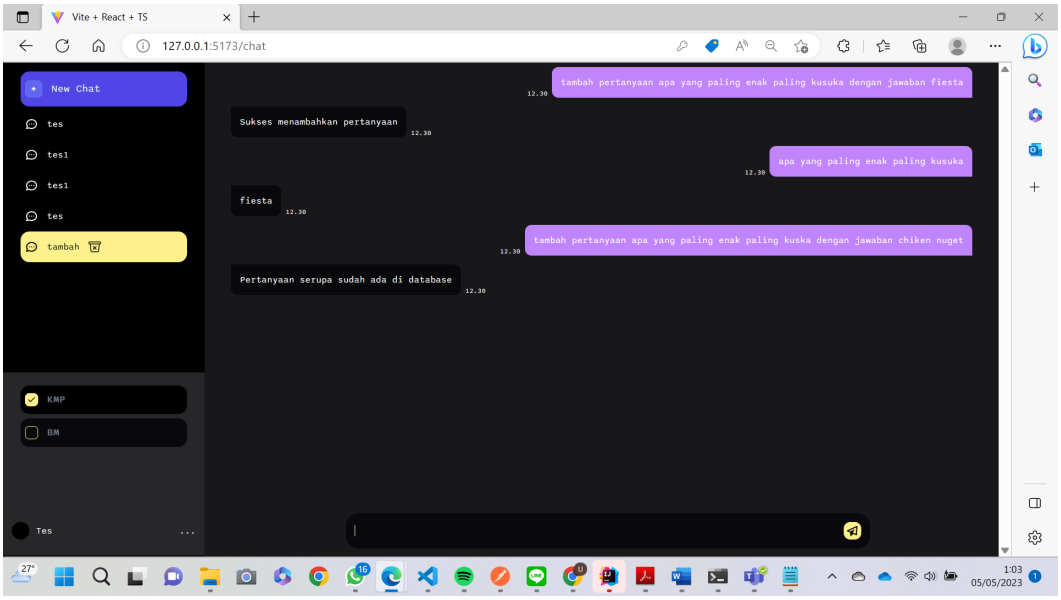
Gambar 4.3.4 fitur kalender

5. Fitur QnA



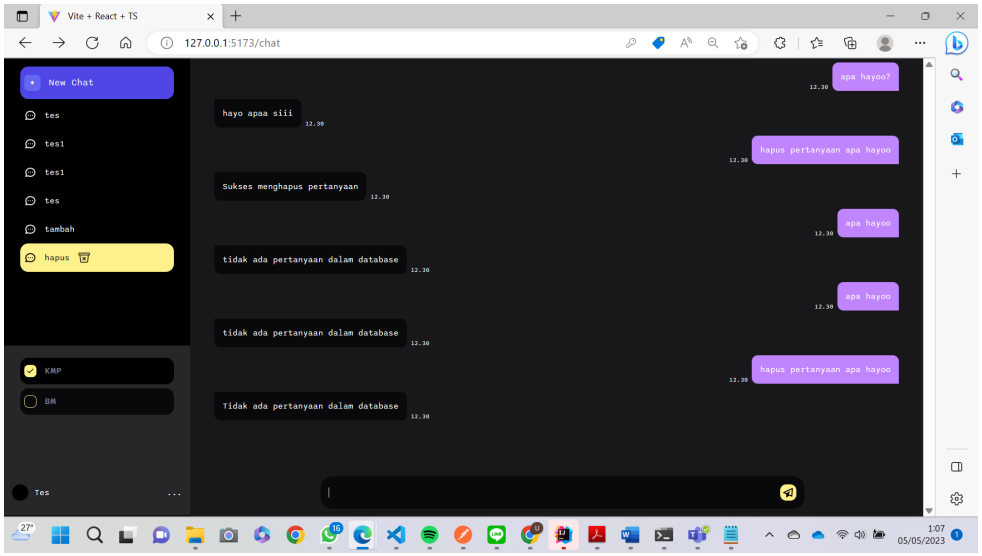
Gambar 4.3.5 Fitur QnA

6. Fitur Tambah Pertanyaan



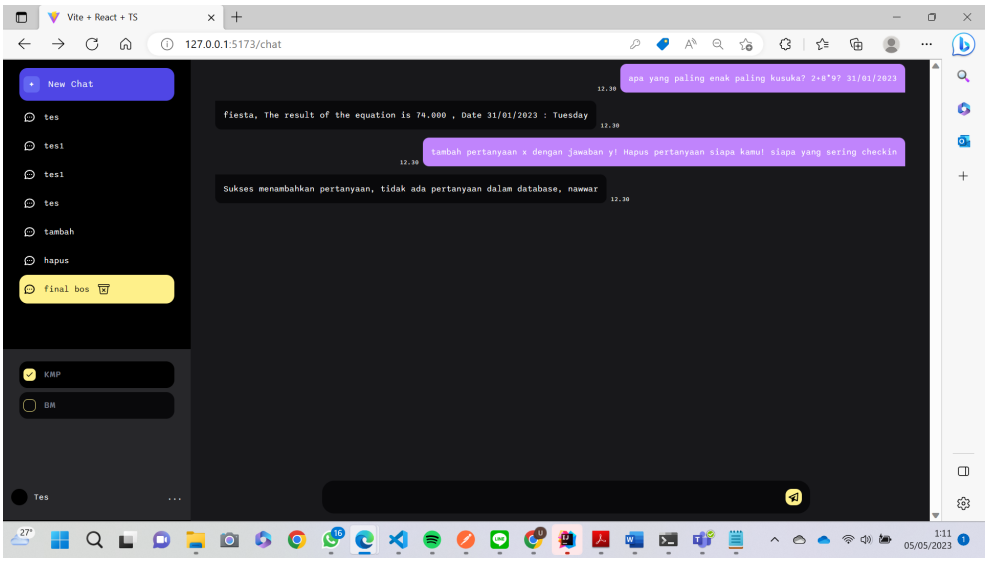
Gambar 4.3.6 Fitur Tambah Pertanyaan

7. Fitur Hapus Pertanyaan



Gambar 4.3.7 Fitur Hapus pertanyaan

8. Fitur Multiple Query



Gambar 4.3.8 Fitur Multiple Query

BAB 5
KESIMPULAN DAN SARAN

5.1 KESIMPULAN

"ChatGPT sederhana" ini berhasil dibangun menggunakan algoritma string matching dan regular expression, serta menambahkan algoritma levenshtein distance sebagai pelengkap. Semua fitur telah berhasil memenuhi spesifikasi yang diberikan dan telah dibuat bonus yang berupa deployment dan video.

5.2 SARAN

Apabila ingin menambahkan Tubes perhatikanlah deadline yang diberikan, jangan sangat mendekati ke UAS, tidak ada waktu buat belajar.

5.3 REFLEKSI

Penulis merasa bahwa tugas kali ini cukup menarik namun terdapat beberapa ketidakjelasan pada spesifikasi yang lumayan membingungkan.

DAFTAR PUSTAKA

1. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf
2. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf

LINK VIDEO DAN GITHUB

Github : https://github.com/Ulung32/Tubes3_13521054.git
Web : tubes3-13521054.vercel.app
Video : https://youtu.be/koKyP7ZVb_o