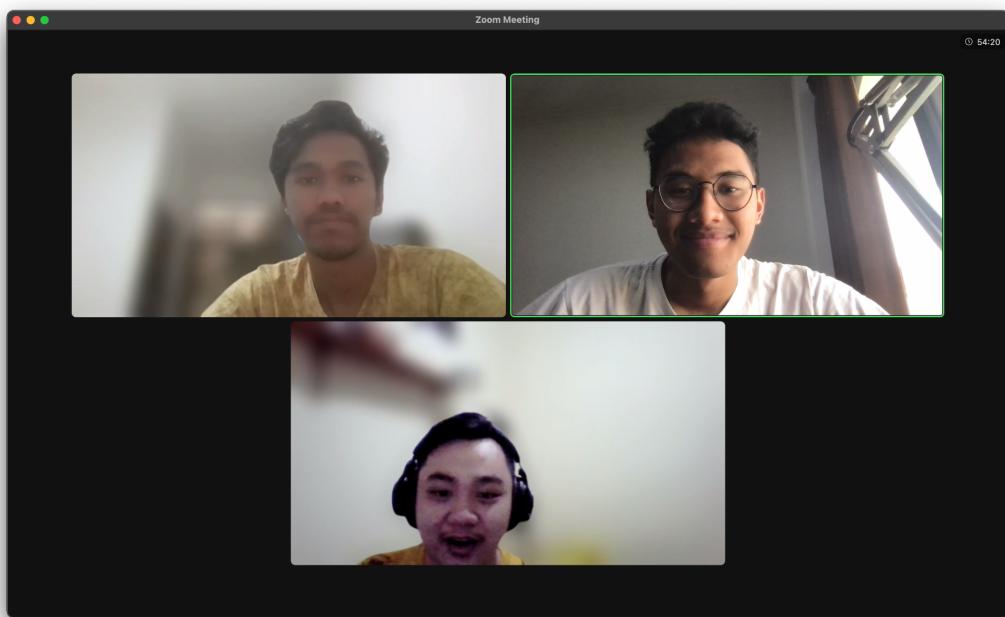


IF2211 Strategi Algoritma

Penerapan String Matching dan Regular Expression dalam Pembuatan ChatGPT Sederhana



Laporan Tugas Besar III

Disusun untuk memenuhi tugas mata kuliah IF2211 Strategi Algoritma pada
Semester 2 (dua) Tahun Akademik 2022/2023

Disusun oleh:

13521063 Salomo Reinhart Gregory Manalu

13521069 Louis Caesa Kesuma

13521072 Irsyad Nurwidianto Basuki

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

**BANDUNG
2023**

DAFTAR ISI

BAB I.....	1
BAB II.....	3
2.1 Dasar Teori.....	3
2.2 Penjelasan Singkat Aplikasi Web.....	4
BAB III.....	6
3.1 Langkah - Langkah Pemecahan Masalah.....	6
3.2 Fitur Fungsional dan Arsitektur Aplikasi Web.....	6
BAB IV.....	8
4.1 Spesifikasi Teknis Program.....	8
4.2 Tata Cara Penggunaan Program.....	30
4.3 Hasil Pengujian.....	32
4.4 Analisis.....	38
BAB V.....	39
5.1 Kesimpulan.....	39
5.2 Saran.....	39
5.3 Refleksi.....	39
5.4 Tanggapan.....	39
DAFTAR PUSTAKA.....	40
LAMPIRAN.....	41

BAB I

DESKRIPSI MASALAH

Dalam tugas besar ini, kami 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 asalkan dijelaskan di laporan, namun disarankan menggunakan salah satu dari algoritma Hamming Distance, Levenshtein Distance, ataupun Longest Common Subsequence.

Fitur-Fitur Aplikasi:

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.

Layaknya ChatGPT, di sebelah kiri disediakan history dari hasil pertanyaan. 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.

BAB II

LANDASAN TEORI

2.1 Dasar Teori

1. Regex

Regex (Regular Expression) adalah salah satu algoritma yang dapat digunakan dalam *string matching*. Secara garis besar, *regex* menggunakan *pattern matching* pada *string* yang ingin dicek. *Regex* juga dapat digunakan untuk mengganti *string* yang memenuhi *pattern* yang kita inginkan. Pada program ini, *regex* digunakan untuk menentukan jenis pertanyaan dari masukan *string* serta memformat *string* tersebut agar lebih mudah dicek dengan menggunakan algoritma lainnya.

2. KMP

KMP (Knuth-Morris-Pratt) adalah salah satu algoritma *string matching* yang umumnya digunakan untuk menentukan apakah suatu *string* merupakan *substring* dari *string* lain. Secara garis besar KMP melakukan pencocokan dengan cara menggeser *index* pengecekannya, namun KMP juga mengatur seberapa jauh *index* tersebut akan digeser berdasarkan panjang akhiran *string* terbesar yang juga merupakan awalan dari *string* tersebut. KMP umumnya akan lebih cocok jika *string* yang akan di-*match* berukuran lebih kecil, dan akan tidak terlalu efektif jika *string*-nya berukuran besar. KMP memiliki kompleksitas $O(m+n)$, sehingga KMP bisa dibilang sangat cepat jika dibandingkan dengan *brute force*.

3. BM

BM (Boyer-Moore) adalah salah satu algoritma *string matching* yang umumnya digunakan untuk menentukan apakah suatu *string* merupakan *substring* dari *string* lain.

Secara garis besar BM diimplementasikan dengan menggunakan 2 teknik, yaitu *looking-glass* dan *character-jump*. *Looking-glass* berupa teknik untuk menelusuri *string* dari belakang, dan *character-jump* merupakan teknik untuk menggeser *index* pencarian pada *string*. Pada BM, terdapat sebuah tabel yang berisikan *index* dimana karakter tersebut terakhir kali muncul, hal tersebut dapat dicari dengan memanfaatkan teknik *looking-glass*. Untuk pergeseran *index* umumnya ada 3 kasus, yaitu menggeser *pattern* jika karakter pada *index* sekarang pada teks ada pada *pattern*, menggeser *index* pada teks ke kanan jika ada karakter yang sama namun tidak memungkinkan untuk melakukan pergeseran kasus 1, dan jika tidak memenuhi keduanya maka geser *pattern* sehingga sejajar dengan *index* pada teks+1. BM umumnya akan lebih efektif jika digunakan untuk *string* yang panjang, namun akan tidak efektif jika *string*-nya pendek. BM memiliki kompleksitas $O(nm + A)$.

2.2 Penjelasan Singkat Aplikasi Web

Aplikasi web yang kami buat, dibuat menggunakan React untuk *framework front-end* serta Express JS dan Node JS untuk *framework backend*. Dalam implementasinya, kami menggunakan React sebagai framework front-end karena kemampuannya dalam membangun antarmuka pengguna yang responsif dan cepat. Sementara itu, Express JS dan Node JS digunakan sebagai framework backend untuk mengatur koneksi ke database dan melakukan operasi CRUD pada produk. Dalam hal styling, kami menggunakan CSS untuk mendesain tampilan antarmuka pengguna agar lebih menarik dan mudah dipahami.

Secara keseluruhan, aplikasi web yang kami buat merupakan sebuah platform pengelolaan produk yang dirancang dengan menggunakan teknologi React, Express JS, dan Node JS. Aplikasi ini memiliki

fitur-fitur yang berguna bagi pengguna dan didesain dengan tampilan antarmuka pengguna yang menarik dan mudah digunakan.

BAB III

PEMECAHAN MASALAH

3.1 Langkah - Langkah Pemecahan Masalah

Dalam menyelesaikan permasalahan ini, telah didekomposisi beberapa permasalahan sebagai langkah-langkah berikut ini.

1. Memahami dan mempelajari algoritma-algoritma *string matching* seperti KMP, BM, *Levenshtein Distance*, dan *Regex*.
2. Mempelajari *syntax-syntax* yang akan digunakan dalam bahasa Javascript, mongoDB, dan React.js.
3. Mengimplementasikan algoritma KMP, BM, *Levenshtein Distance*, dan *Regex* yang disesuaikan dengan implementasi GUI.
4. Membuat GUI yang menggunakan React.js yang disesuaikan dengan implementasi algoritma.
5. Mengimplementasikan *database* yang disesuaikan dengan implementasi algoritma serta GUI.
6. Menghubungkan seluruh program agar bisa dijalankan.

3.2 Fitur Fungsional dan Arsitektur Aplikasi Web

1. Fitur *Chat* / Percakapan

Program yang kami buat memiliki fitur percakapan yang memungkinkan pengguna untuk berinteraksi dengan sistem. Untuk memulai percakapan, pengguna diminta untuk memasukkan pesan melalui kolom input yang telah disediakan. Input pengguna akan dikelola oleh program ketika ada perubahan atau saat tombol submit ditekan. Setelah pengguna mengirimkan pesan, pesan tersebut akan dikirim ke API dan

disimpan dalam database. Pesan yang telah dikirimkan juga akan diolah menggunakan algoritma yang telah disediakan dan dipilih sebelumnya. Hasil dari percakapan akan diproses ke dalam komponen yang nantinya dapat ditampilkan ke pengguna. Percakapan juga akan ditampilkan secara berurutan sesuai dengan urutan waktu pengiriman.

2. Fitur Riwayat Kencan

Untuk menyimpan riwayat kencan, program kami menyimpan percakapan di lokal *front-end* React JS, yaitu di ‘previousChats’ pada file App.js. Dengan adanya fitur ini, pengguna dapat menambah percakapan baru dan dapat kembali ke percakapan yang telah dilakukan sebelumnya.

3. Fitur Pemilihan Algoritma

Untuk memilih algoritma yang ingin digunakan untuk melakukan pemrosesan *String Matching*, disediakan tombol untuk memilih algoritma Knuth-Morris-Pratt (KMP) dan Boyer-Moore (BM). Secara *default*, apabila tidak menekan tombol manapun, program akan menggunakan algoritma Boyer-Moore(BM).

BAB IV

ANALISIS PEMECAHAN MASALAH

4.1 Spesifikasi Teknis Program

1. Struktur data

a. List

List merupakan struktur data yang berupa daftar yang dapat berubah. *List* digunakan sebagai tempat penyimpanan data agar bisa diproses oleh program. Contoh penggunaan *list* adalah pada penyimpanan pertanyaan dari pengguna, penyimpanan data respons yang menurut program paling tepat, dan sebagainya.

b. Matriks

Matriks merupakan struktur data yang berupa *list of list* yang dapat berukuran mxn atau nxn. Contoh penggunaan matriks adalah pada algoritma *levenshtein distance*.

2. Backend

i. Algorithm.js

```

Salomo309 · 43 minutes ago | 2 authors (You and others)
1 const knowQuery = require('../query/knowQuery');
2 const chatsQuery = require('../query/chatsQuery');
3 const Chat = require('../models/chats');
4 const Knowledge = require('../models/knowledge');
5
6 /* ----- REGEX SECTION ----- */
7 const pattern = {
8     MATH : "mathExpr",
9     DATE : "dateExpr",
10    ADD : "addExpr",
11    DEL : "deleteExpr",
12    UNKNOWN : "unknownExp"
13 }
14
15 const regex = {
16     mathExprRegex : /(^\d{0-9}/*(\^+-)*)(?=(:\d*\d){2})([+-*/^()0-9\s]*)([^0-9/*(\^+-)*]*/,
17     dateRegex : /(^\s*\w{1,2}\s*\|\s*\d{1,2}\s*\|\s*\w*\d{1,4}\s*/),
18     addRegex : /(tambah\s*pertanyaan)\s*(.*?)*dengan\s*jawaban\s*(.*)/,
19     delRegex : /(hapus\s*pertanyaan)\s*(.*)/
20 }
21
22 function standarizeQuestions(input) { // standarisasi input
23     /* STANDARISASI INPUT */
24     // lowercase input
25     input = input.toLowerCase();
26
27     // hapus space berlebihan di input
28     input = removeUselessSpaces(input);
29
30     // ganti tanda-tanda spesial
31     input = input.replace(/[\^+*/-`^()?"|\n\w\s]/g, ""); // tanda-tanda yang diperbolehkan: +, -, /, *, ^, ', ", ?, |
32
33     /* MASUKKAN DAFTAR PERTANYAAN */
34     let listOfQuestions = [];
35
36     /* PISAHKAN PERTANYAAN */
37     const arrayOfQuestions = input.split("?\").map((element) => removeUselessSpaces(element)); // daftar pertanyaan
38

```

```

src > backend > JS Algorithm.js > ...
40     if (arrayOfQuestions[arrayOfQuestions.length - 1] == "" && arrayOfQuestions.length != 1) {
41         arrayOfQuestions.pop();
42     }
43
44     for (let i = 0; i < arrayOfQuestions.length; i++) {
45         listOfQuestions.push(findPattern(arrayOfQuestions[i])); // cari pattern dari setiap pertanyaan
46     }
47
48     return listOfQuestions;
49 }
50
51 function removeUselessSpaces(input) {
52     return input.trim().replace(/\s+/g, ' ');
53 }
54
55 function findPattern(question) { // mencari pattern yang cocok untuk pertanyaan
56     if (isItUnknown(question)) {
57         return [removeUselessSpaces(question), pattern.UNKNOWN, null];
58     } else if (isItDate(question)) {
59         return [removeUselessSpaces(question.replace(regex.dateRegex, " (date) ")), pattern.DATE, question.match(regex.dateRegex)[0]];
60     } else if (isItMath(question)) {
61         return [removeUselessSpaces(question.replace(question.match(regex.mathExprRegex)[2], " (math) ")), pattern.MATH, question.match(regex.mathExprRegex)[2]];
62     } else if (isItAdd(question)) {
63         for (let i = 0; i < question.match(regex.addRegex).length; i++) {
64             console.log(question.match(regex.addRegex)[i]);
65         }
66         let addElements = question.match(regex.addRegex)[0].split(/\sdengan\s(.*)\s/).map((element) => removeUselessSpaces(element));
67         let pElements = addElements[0].split(" ").map((element) => element.trim());
68         let jElements = addElements[1].split(" ").map((element) => element.trim());
69
70         let pertanyaan = "";
71         // masukkan pertanyaan
72         for (let i = 2; i < pElements.length; i++) {
73             if (i != 2) {
74                 pertanyaan = pertanyaan.concat(" ", pElements[i]);
75             } else {

```

```

src > backend > JS Algorithm.js > ...
76     |         pertanyaan = pertanyaan.concat(pElements[i]);
77     |
78   }
79
80   let jawaban = "";
81   // masukkan jawaban
82   for (let i = 1; i < jElements.length; i++) {
83     if (i != 1) {
84       jawaban = jawaban.concat(" ", jElements[i]);
85     } else {
86       jawaban = jawaban.concat(jElements[i]);
87     }
88   }
89
90   return [removeUselessSpaces(question.replace(regex.addRegex, " (add) ")), pattern.ADD, [removeUselessSpaces(pertanyaan), removeUselessSpaces(jawaban)]];
91 } else if (isItDeleted(question)) {
92   // handle jika terdapat input seperti "hapus pertanyaan apakah hapus pertanyaan termasuk perintah?"
93   // batasan: seluruh string setelah hapus pertanyaan pertama adalah pertanyaan
94   let delElements = question.match(regex.delRegex)[0].split(/hapus\s*pertanyaan(s.*)/s);
95   let pertanyaan = delElements[1];
96
97   return [removeUselessSpaces(question.replace(regex.delRegex, " (del) ")), pattern.DEL, removeUselessSpaces(pertanyaan)];
98 }
99
100
101 function isItMath(question) {
102   return regex.mathExprRegex.test(question);
103 }
104
105 function isItDate(question) {
106   return regex.dateRegex.test(question);
107 }
108
109 function isItAdd(question) {
110   return regex.addRegex.test(question);
111 }
112

```

```

110   function isItDelete(question) {
111     return regex.delRegex.test(question);
112   }
113
114   function isItUnknown(question) {
115     let ctr = 0;
116     if (isItDate(question)) {
117       ctr++;
118     }
119
120     if (isItMath(question) && ctr != 1) {
121       ctr++;
122     }
123
124     if (isItAdd(question)) {
125       ctr++;
126     }
127
128     if (isItDelete(question)) {
129       ctr++;
130     }
131
132     return ctr != 1;
133   }
134   /* ----- END OF REGEX SECTION ----- */
135

```

```
src > backend > js Algorithm.js > ...
140  /* ----- KMP SECTION ----- */
141  function kmp(input, data) {
142      let inputLength = input.length;
143
144      let k = 0;
145      let found = false;
146      let foundidx = null;
147      while (k < data.length && !found) {
148          let dataLength = data[k][0].length;
149
150          // cek apakah panjangnya sama
151          if (inputLength != dataLength) {
152              k++;
153          } else {
154              let b = computeBorder(input);
155
156              let i = 0;
157              let j = 0;
158
159              while (i < dataLength) {
160                  if (input[j] == data[k][0][i]) {
161                      if (j == inputLength - 1) {
162                          found = true;
163                          foundidx = k;
164                      }
165
166                      i++;
167                      j++;
168                  } else if (j > 0) {
169                      j = b[j-1];
170                  } else {
171                      i++;
172                  }
173              }
174          }
175      }
176  }
```

```
src > backend > JS Algorithm.js > ...
177     return [found, foundidx];
178 }
179
180 function computeBorder(pattern) {
181     let b = [pattern.length];
182     b[0] = 0;
183
184     let patternLength = pattern.length;
185     let j = 0;
186     let i = 1;
187
188     while (i < patternLength) {
189         if (pattern[j] == pattern[i]) {
190             b[i] = j + 1;
191             i++;
192             j++;
193         } else if (j > 0) {
194             j = b[j-1];
195         } else {
196             b[i] = 0;
197             i++;
198         }
199     }
200
201     return b;
202 }
203 /* ----- END OF KMP SECTION ----- */
```

```
src > backend > JS Algorithm.js > ...
206  /* ----- BM SECTION ----- */
207  function bm(input, data) {
208      let last = buildLast(input);
209      let inputLength = input.length;
210
211      let k = 0;
212      let found = false;
213      let foundidx = null;
214      while (k < data.length && !found) {
215          let dataLength = data[k][0].length;
216          let i = inputLength-1;
217          if (i > dataLength-1) {
218              k++;
219          } else {
220              let j = inputLength-1;
221              do {
222                  if (input[j] == data[k][0][i]){
223                      if (j == 0) {
224                          found = true;
225                          foundidx = k;
226                      }
227                      else {
228                          i--;
229                          j--;
230                      }
231                  } else {
232                      let lo = last[data[k][0][i]];
233                      i = i + inputLength - Math.min(j, 1+lo);
234                      j = inputLength - 1;
235                  }
236              } while (i <= dataLength-1 && !found);
237              k++;
238          }
239      }
240      return [found, foundidx];
241  }
242
```

```

src > backend > JS Algorithm.js > ...
243   function buildLast(pattern) {
244     let last = [];
245     for (let i = 0; i < 128; i++) {
246       last[i] = -1;
247     }
248
249     for (let i = 0; i < pattern.length; i++) {
250       last[pattern[i]] = i;
251     }
252
253     return last;
254   }
255   /* ----- END OF BM SECTION ----- */
256
257

```

```

src > backend > JS Algorithm.js > ...
258   /* ----- LEVENSHTEIN SECTION ----- */
259   function levenshtein(input, data) {
260     const m = input.length;
261     const n = data.length;
262
263     const dp = new Array(m + 1).fill(null).map(() => new Array(n + 1).fill(null));
264
265     for (let i = 0; i <= m; i++) {
266       dp[i][0] = i;
267     }
268
269     for (let j = 0; j <= n; j++) {
270       dp[0][j] = j;
271     }
272
273     for (let i = 1; i <= m; i++) {
274       for (let j = 1; j <= n; j++) {
275         if (input[i - 1] === data[j - 1][0]) {
276           dp[i][j] = dp[i - 1][j - 1];
277         } else {
278           dp[i][j] = 1 + Math.min(dp[i - 1][j], dp[i][j - 1], dp[i - 1][j - 1]);
279         }
280       }
281     }
282
283     return dp[m][n];
284   }
285   /* ----- END OF LEVENSHTEIN SECTION ----- */
286
287
288   /* ----- CALCULATOR & DATE SECTION ----- */
289   const calculationError = {
290     mathError : "Sintaks tidak valid",
291     dateError : "Tanggal tidak valid"
292   }
293
294   // fungsi untuk melakukan perhitungan matematika

```

ii. server.js

```

src > backend > JS server.js > ...
1  const express = require('express');
2  const mongoose = require('mongoose');
3  const axios = require('axios');
4  const chatsRoutes = require('../routes/chatsRoutes');
5  const chatsQuery = require('../query/chatsQuery');
6  const knowRoutes = require('../routes/knowRoutes');
7  const knowQuery = require('../query/knowQuery');
8
9  // express app
10 const app = express();
11 app.use(function(req, res, next) {
12   res.header("Access-Control-Allow-Origin", "http://localhost:3000"); // update to match the domain you will make the request from
13   res.header("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
14   next();
15 });
16
17 // middleware to handle json request body
18 app.use(express.json());
19
20 // add the Routes to the app
21 app.use('/chats', chatsRoutes);
22 app.use('/knowledge', knowRoutes);
23
24 // connect to database (MongoDB)
25 const dbURI = 'mongodb+srv://13521063:ngechatgpt@chatbot.ynjyvpn.mongodb.net/chatbotweb';
26 mongoose.connect(dbURI, { useNewUrlParser: true, useUnifiedTopology: true })
27   .then((result) => app.listen(4000))
28   .catch((err) => console.log(err));
29

```

```

src > backend > JS Algorithm.js > ...
295  function calculate(number) {
296    try {
297      const result = eval(number);
298      return result;
299    } catch {
300      return calculationError.mathError;
301    }
302  }
303
304 // fungsi untuk melakukan perhitungan hari dari input tanggal
305 function getDayFromDate(datestring) {
306   // input berupa DD/MM/YYYY
307   const [day, month, year] = datestring.split("/");
308   const daysList = ["Minggu", "Senin", "Selasa", "Rabu", "Kamis", "Jumat", "Sabtu"];
309   const standardizedDate = new Date(year, month-1, day);
310
311   if (standardizedDate.getDay() !== undefined) {
312     return daysList[standardizedDate.getDay()];
313   } else {
314     return calculationError.dateError;
315   }
316 }
317 /* ----- END OF CALCULATOR & DATE SECTION ----- */

```

```

src > backend > JS Algorithm.js > ...
320  /* ----- FINDING PROPER RESPONSES SECTION ----- */
321  // fungsi untuk mencari pertanyaan yang exact match menggunakan kmp/bm
322  async function findResponses(input, KMP) {
323      // nanti masukin proses ngambil daftar pertanyaan dan response dari query terus masukin ke data
324      /* INSERT HERE */
325      let data = await knowQuery.getQuestionAndAnswer();
326
327      const listOfQuestions = standarizeQuestions(input);
328
329      let listOfResponses = [];
330
331      for (let i = 0; i < listOfQuestions.length; i++) {
332          let [exact, index] = [null, null];
333          if (KMP) {
334              [exact, index] = kmp(listOfQuestions[i][0], data);
335          } else {
336              [exact, index] = bm(listOfQuestions[i][0], data);
337          }
338
339          if (exact) {
340              listOfResponses[i] = await generateResponse(listOfQuestions[i], data, index); // ambil response
341          } else {
342              listOfResponses[i] = null;
343          }
344      }
345
346      // berikan solusi
347      for (let i = 0; i < listOfResponses.length; i++) {
348          if (listOfResponses[i] == null) {
349              listOfResponses[i] = await findClosestSolution(listOfQuestions[i], data);
350          }
351      }
352
353      const result = listOfResponses.flat().join(' ');
354      return result;
355  }

```

```

src > backend > JS Algorithm.js > ...
357  // fungsi untuk mencari pertanyaan ketika tidak ada yang exact match
358  async function findClosestSolution(question, data) {
359      let rank = [];
360
361      for (let i = 0; i < data.length; i++) {
362          rank.push([i, (data[i][0].length - levenshtein(question[0], data[i][0])) / data[i][0].length]);
363      }
364
365      rank.sort((a, b) => (b[1] - a[1]));
366
367      // cek apakah ada yang mirip > 90%
368      if (rank[0][1] >= 0.9) {
369          return await generateResponse(question, data, rank[0][0]); // generate response untuk pertanyaan yang >= 90% mirip
370      } else {
371          // generate 3 pertanyaan termirip
372          let solutions = "Pertanyaan tidak ditemukan di database.\nBerikut pertanyaan yang mirip:\n";
373          for (let i = 0; i < rank.length && i < 3; i++) {
374              solutions = solutions.concat(i+1);
375              solutions = solutions.concat(", ");
376              solutions = solutions.concat(data[rank[i][0]][0]);
377              solutions = solutions.concat("\n");
378          }
379          return solutions;
380      }
381  }
382
383  // fungsi untuk melakukan kalkulasi untuk pertanyaan seperti kalkulator, tanggal, dll
384  async function generateResponse(question, data, idx) {
385      let solutions = "";
386      if (question[1] == pattern.MATH) {
387          if (calculate(question[2]) == calculationError.mathError) {
388              solutions = solutions.concat(calculationError.mathError);
389          } else {
390              solutions = solutions.concat(data[idx][1]);
391              solutions = solutions.concat(" ", calculate(question[2]));
392          }
393      } else if (question[1] == pattern.DATE) {

```

```

src > backend > js Algorithm.js > ...
394     if (getDayFromDate(question[2]) == calculationError.dateError) {
395         solutions = solutions.concat(calculationError.dateError);
396     } else {
397         solutions = solutions.concat(data[idx][1]);
398         solutions = solutions.concat(" ", getDayFromDate(question[2]));
399     }
400 } else if (question[1] == pattern.ADD) {
401     if (question[2][0].trim() == "" || question[2][1].trim() == "") {
402         solutions = solutions.concat("pertanyaan atau jawaban tidak boleh kosong");
403     } else if (knowQuery.isQuestionExist(question[2][0])) {
404         await knowQuery.updateKnowledgeByQuestion(question[2][0], question[2][1]);
405         solutions = solutions.concat("pertanyaan ", question[2][0], " sudah ada! jawaban diupdate menjadi ", question[2][1]);
406     } else {
407         const newKnowledge = new Knowledge({
408             question: question[1],
409             answer: solutions
410         });
411         await knowQuery.addKnowledge(newKnowledge);
412     }
413     solutions = solutions.concat("pertanyaan ", question[2][0], " telah ditambah");
414 }
415 } else if (question[1] == pattern.DEL) {
416     // // TODO: query delete dari database
417     if (knowQuery.isQuestionExist(question[1])) {
418         knowQuery.deleteByQuestion(question[1]);
419     } else {
420         solutions = solutions.concat("Tidak ada pertanyaan ", question[2], " di database");
421     }
422     solutions = 2; // sementara
423 } else {
424     solutions = solutions.concat(data[idx][1]);
425 }
426
427 }
428 /* ----- END OF FINDING PROPER RESPONSES SECTION ----- */

```

```

432     module.exports = {
433         standarizeQuestions,
434         removeUselessSpaces,
435         findPattern,
436         isItMath,
437         isItDate,
438         isItAdd,
439         isItDelete,
440         isItUnknown,
441         kmp,
442         computeBorder,
443         bm,
444         levenshtein,
445         calculate,
446         getDayFromDate,
447         findResponses,
448         findClosestSolution,
449         generateResponse
450     }
451

```

3. Frontend

i. App.js

```
function App() {
  const [messages, setMessages] = useState([]);
  const [currentMessage, setCurrentMessage] = useState("");
  const [previousChat, setPreviousChat] = useState([]);
  const [currentChatTitle, setCurrentChatTitle] = useState(null);
  const [algoButton, setAlgoButton] = useState(null);

  const emptyForm = useRef(null);

  useEffect(() => {
    // Empty the form input
    const empty = emptyForm.current;
    empty.value = "";
  }, [messages]);

  useEffect(() => {
    if (!currentChatTitle && currentMessage && messages) {
      setCurrentChatTitle(currentMessage);
    }
    if (currentChatTitle && currentMessage && messages) {
      setPreviousChat((prevChat) => [
        ...prevChat,
        {
          title: currentChatTitle,
          role: "user",
          value: currentMessage,
        },
        {
          title: currentChatTitle,
          role: messages.role,
          value: messages,
        },
      ]);
    }
  }, [messages, currentChatTitle]);
```

```
// SideBar
const newChat = () => {
  setCurrentMessage("");
  setMessages([]);
  setCurrentChatTitle(null);
};

const uniqueTitles = Array.from(
  new Set(previousChat.map((prevChat) => prevChat.title))
);

const handleClick = (uniqueTitle) => {
  setCurrentChatTitle(uniqueTitle);
  setMessages(null);
  setCurrentMessage("");
};
```

```

// MainBox
function handleInputChange(event) {
  setCurrentMessage(event.target.value);
}

async function handleFormSubmit(event) {
  event.preventDefault();
  // if (currentMessage !== "") {
  //   setMessages([...messages, currentMessage]);
  // }
  console.log(currentMessage);
  console.log("BUTTON:", algoButton);
  axios
    .post("http://localhost:4000/chats/answer", {
      message: currentMessage,
      useKMP: algoButton,
    })
    .then((response) => {
      console.log(response.data);
      setMessages(response.data);
      // Lakukan sesuatu dengan data response
    })
    .catch((error) => {
      console.error(error);
      // Lakukan sesuatu dengan error
    });
}

const currentChat = previousChat.filter(
  (previousChat) => previousChat.title === currentChatTitle
);

const ref = useRef(null);

useEffect(() => {
  // Scroll to bottom when messages is added
  const element = ref.current;
  setTimeout(() => {
    element.scrollTop = element.scrollHeight;
  }, 0);
  // element.scrollTop = element.scrollHeight;
}, [messages]);

```

```
function kmpClick() {
|  setAlgoButton(true);
}

function bmClick() {
|  setAlgoButton(false);
}
```

```

    return (
      <div className="App">
        <section className="sidebar">
          <button className="newchat" onClick={newChat}>
            + New Chat
          </button>
          <ul className="historylist">
            {uniqueTitles?.map((uniqueTitle, index) => (
              <li key={index} onClick={() => handleClick(uniqueTitle)}>
                <div>{uniqueTitle}</div>
              </li>
            ))}
          </ul>
          <div className="algorithm-button">
            <button className="kmpbutton" onClick={kmpClick}>
              KMP
            </button>
            <button className="bmbutton" onClick={bmClick}>
              BM
            </button>
          </div>
        </section>
        <section className="main">
          <ul className="chat" ref={ref}>
            {currentChat.map((message, index) => (
              <li key={index} className={message.role}>
                <div className="img-container">
                  <img
                    src={message.role === "user" ? saul : gpt}
                    className="imageLogo"
                  />
                </div>
                <div className="chat-container">
                  <p>{message.value}</p>
                </div>
              </li>
            ))}
          </ul>
          <div className="input">
            <div className="input-container">
              <form className="prompt" onSubmit={handleFormSubmit}>
                <input
                  ref={emptyForm}
                  type="text"
                  onChange={handleInputChange}
                  placeholder="Type your message"
                ></input>
              </form>
            </div>
          </div>
        </section>
      </div>
    );
}

export default App;

```

4. Models

i. chats.js

```
src > models > JS chats.js > [x] chats
  1  const mongoose = require('mongoose');
  2  const Schema = mongoose.Schema;
  3  // const AutoIncrement = require('mongoose-sequence')(mongoose);
  4
  5  const chats = new Schema({
  6    message: {
  7      type: String,
  8      required: true
  9    },
 10    is_bot_message: {
 11      type: Boolean,
 12      required: true
 13    }
 14  },
 15  { timestamps: true}
 16 );
 17
 18 // chats.plugin(AutoIncrement, {inc_field: 'seq'});
 19 const Chat = mongoose.model('Chat', chats);
 20 module.exports = Chat;
 21
```

ii. knowledge.js

```

src > models > JS knowledge.js > [e] knowledges > ↗ answer
  1  const mongoose = require('mongoose');
  2  const Schema = mongoose.Schema;
  3  // const AutoIncrement = require('mongoose-sequence')(mongoose);
  4
  5  const knowledges = new Schema({
  6    question: {
  7      type: String,
  8      required: true
  9    },
 10   answer: {
 11     type: String,
 12     required: true
 13   }
 14 });
 15
 16 const Knowledge = mongoose.model('Knowledge', knowledges);
 17 module.exports = Knowledge;
 18

```

5. Query

i. chatsQuery.js

```

src > query > JS chatsQuery.js > ↗ giveRespond > [e] respond
  1  const { MongoClient, ObjectId } = require('mongodb');
  2  const algo = require('../backend/Algorithm');
  3
  4  // definisikan konfigurasi koneksi ke database
  5  const uri = 'mongodb+srv://13521063:ngechatgpt@chatbot.ynjyvpn.mongodb.net/chatbotweb';
  6  const dbName = 'chatbotweb';
  7
  8  // buat fungsi untuk mendapatkan data chats
  9  async function getChats() {
 10    const client = new MongoClient(uri, { useUnifiedTopology: true });
 11    await client.connect();
 12    const db = client.db(dbName);
 13    const collection = db.collection('chats');
 14    const chats = await collection.find().toArray();
 15    await client.close();
 16    return chats;
 17  };
 18
 19  // buat fungsi untuk mendapatkan data chat berdasarkan ID
 20  async function getChatById(id) {
 21    const client = new MongoClient(uri, { useUnifiedTopology: true });
 22    await client.connect();
 23    const db = client.db(dbName);
 24    const collection = db.collection('chats');
 25    const chat = await collection.findOne({ _id: new ObjectId(id) });
 26    await client.close();
 27    return chat;
 28  };

```

```

src > query > JS chatsQuery.js > ⊕ giveRespond > ✉ respond
29
30  // Mendapatkan semua chat yang bukan merupakan bot message
31  async function getUserMessages() {
32    const db = await connect();
33    const chats = await db.collection('chats').find({ is_bot_message: false }).toArray();
34    return chats;
35  };
36
37 // Mendapatkan semua bot message
38 async function getBotMessages() {
39    const db = await connect();
40    const chats = await db.collection('chats').find({ is_bot_message: true }).toArray();
41    return chats;
42  };
43
44 // buat fungsi untuk menambah data chat baru
45 async function addChat(chat) {
46   const client = new MongoClient(uri, { useUnifiedTopology: true });
47   await client.connect();
48   const db = client.db(dbName);
49   const collection = db.collection('chats');
50   const result = await collection.insertOne(chat);
51   await client.close();
52   return result;
53 };
54

```

```

src > query > JS chatsQuery.js > ⊕ giveRespond > ✉ respond
55 // buat fungsi untuk menghapus data chat berdasarkan ID
56 async function deleteChatById(id) {
57   const client = new MongoClient(uri, { useUnifiedTopology: true });
58   await client.connect();
59   const db = client.db(dbName);
60   const collection = db.collection('chats');
61   const result = await collection.deleteOne({ _id: new ObjectId(id) });
62   await client.close();
63   return result;
64 };
65
66 async function giveRespond(question, KMP) {
67   const client = new MongoClient(uri, { useUnifiedTopology: true });
68   await client.connect();
69   const db = client.db(dbName);
70   const collection = db.collection('chats');
71   const respond = await algo.findResponses([question, KMP]);
72   await client.close();
73   return respond;
74 };
75
76 // ekspor semua fungsi yang telah dibuat
77 module.exports = {
78   getChats,
79   getChatById,
80   getUserMessages,
81   getBotMessages,
82   addChat,
83   deleteChatById,
84   giveRespond
85 };
86

```

ii. knowQuery.js

```

src > query > JS knowQuery.js > >DeleteByQuestion > [x] result > question
  1 const { MongoClient, ObjectId } = require('mongodb');
  2 const Knowledge = require('../models/knowledge');
  3 const mongoose = require('mongoose');
  4
  5 const uri = 'mongodb+srv://13521063:ngechatgpt@chatbot.ynjyvpn.mongodb.net/chatbotweb';
  6 const dbName = 'chatbotweb';
  7
  8 async function connect() {
  9   const client = new MongoClient(uri, { useNewUrlParser: true });
 10   await client.connect();
 11   const db = client.db(dbName);
 12   return db;
 13 };
 14
 15 // fungsi untuk mengambil semua data knowledge
 16 async function getAllKnowledge() {
 17   const db = await connect();
 18   const knowledge = await db.collection('knowledges').find().toArray();
 19   return knowledge;
 20 };
 21
 22 // fungsi untuk mengambil satu data knowledge berdasarkan id
 23 async function getKnowledgeById(id) {
 24   const db = await connect();
 25   const knowledge = await Knowledge.findOne({ _id: new mongoose.Types.ObjectId(id) });
 26   return knowledge;
 27 };
 28
 29 // fungsi untuk mengambil satu data knowledge berdasarkan question
 30 async function getKnowledgeByQuestion(question) {
 31   const db = await connect();
 32   const knowledge = await db.collection('knowledges').findOne({ question: question });
 33   return knowledge;
 34 };

```

```

src > query > JS knowQuery.js > DeleteByQuestion > [x] result > question
 36 // fungsi untuk menambahkan data knowledge
 37 async function addKnowledge(newKnowledge) {
 38   const db = await connect();
 39   const result = await db.collection('knowledges').insertOne(newKnowledge);
 40   return result.insertedId;
 41 };
 42
 43 // fungsi untuk mengupdate data knowledge berdasarkan question
 44 async function updateKnowledgeByQuestion(question, updatedAnswer) {
 45   const db = await connect();
 46   const result = await db.collection('knowledges').updateOne({ question: question }, { $set: { answer: updatedAnswer } });
 47   return result.modifiedCount;
 48 };
 49
 50
 51 // fungsi untuk menghapus data knowledge berdasarkan id
 52 async function deleteKnowledgeById(id) {
 53   const db = await connect();
 54   const result = await db.collection('knowledges').deleteOne({ _id: new ObjectId(id) });
 55   return result.deletedCount;
 56 };
 57
 58 async function deleteByQuestion(question) {
 59   try {
 60     const result = await Knowledge.deleteOne({ question: question });
 61     console.log(`Deleted ${result.deletedCount} knowledge(s) with question "${question}"`);
 62     return result.deletedCount;
 63   } catch (error) {
 64     console.error(error);
 65   }
 66 };
 67
 68 async function isQuestionExist(question) {
 69   const db = await connect();
 70   const knowledge = await db.collection('knowledges').findOne({ question: question });
 71   return knowledge !== null;
 72 };

```

```

src > query > JS knowQuery.js > ...
74 // fungsi untuk mendapatkan semua question dari collection knowledge
75 async function getAllQuestions() {
76   const db = await connect();
77   const questions = await db.collection('knowledges').distinct('question');
78   return questions;
79 };
80
81 // fungsi untuk mendapatkan semua answer dari collection knowledge
82 async function getAllAnswers() {
83   const db = await connect();
84   const answers = await db.collection('knowledges').distinct('answer');
85   return answers;
86 };
87
88 async function getQuestionAndAnswer() {
89   const db = await connect();
90   const result = await db.collection('knowledges').find({}, { projection: { _id: 0, question: 1, answer: 1 } }).toArray();
91   return result.map(({ question, answer }) => [question, answer]);
92 }
93
94 module.exports = {
95   getAllKnowledge,
96   getKnowledgeById,
97   getKnowledgeByQuestion,
98   addKnowledge,
99   updateKnowledgeByQuestion,
100  deleteKnowledgeById,
101  deleteByQuestion,
102  isQuestionExist,
103  getAllQuestions,
104  getAllAnswers,
105  getQuestionAndAnswer
106 };

```

6. Routes

i. chatsRoutes.js

```

src > routes > JS chatsRoutes.js > ⊗ router.post('/answer') callback
1  const express = require('express');
2  const Chat = require('../models/chats');
3  const { getChats, getChatById, addChat, deleteChatById, giveRespond } = require('../query/chatsQuery');
4  const router = express.Router();
5
6  // GET all chats
7  router.get('/', async (req, res) => {
8    try {
9      const chats = await getChats();
10     res.json(chats);
11   } catch (err) {
12     res.status(500).json({ message: err.message });
13   }
14 });
15
16 // GET a single chat by ID
17 router.get('/:id', async (req, res) => {
18   try {
19     const chat = await getChatById(req.params.id);
20     if (chat == null) {
21       return res.status(404).json({ message: 'Cannot find chat' });
22     }
23     res.json(chat);
24   } catch (err) {
25     res.status(500).json({ message: err.message });
26   }
27 });
28

```

```

src > routes > JS chatsRoutes.js >  router.post('/answer') callback
  29 // POST a new chat
  30 router.post('/', async (req, res) => {
  31   const chat = new Chat({
  32     message: req.body.message,
  33     is_bot_message: req.body.is_bot_message
  34   });
  35   try {
  36     const newChat = await addChat(chat);
  37     res.status(201).json(newChat);
  38   } catch (err) {
  39     res.status(400).json({ message: err.message });
  40   }
  41 });
  42
  43 // DELETE an existing chat by ID
  44 router.delete('/:id', async (req, res) => {
  45   try {
  46     const chat = await deleteChatById(req.params.id);
  47     if (chat == null) {
  48       return res.status(404).json({ message: 'Cannot find chat' });
  49     }
  50     res.json({ message: 'Deleted chat' });
  51   } catch (err) {
  52     res.status(500).json({ message: err.message });
  53   }
  54 });
  55

```

```

src > routes > JS chatsRoutes.js > ...
  55
  56 // POST a new chat and get response from bot
  57 router.post('/answer', async (req, res, next) => {
  58   try {
  59     const KMP = req.body.useKMP || false;
  60     const response = await giveRespond(req.body.message, KMP);
  61
  62     const newChat = new Chat({
  63       message: req.body.message,
  64       is_bot_message: false
  65     });
  66     await addChat(newChat);
  67
  68     const botResponse = new Chat({
  69       message: response,
  70       is_bot_message: true
  71     });
  72     await addChat(botResponse);
  73
  74     res.status(200).json(response);
  75   } catch (err) {
  76     res.status(500).json({ message: err.message });
  77   }
  78 });
  79
  80 module.exports = router;

```

ii. knowRoutes.js

```
src > routes > JS knowRoutes.js > ...
1  const express = require('express');
2  const Knowledge = require('../models/knowledge');
3  const { getAllKnowledge, getKnowledgeById, getKnowledgeByQuestion, addKnowledge, updateKnowledgeByQuestion,
4  | getAllAnswers,
5  | getQuestionAndAnswer } = require('../query/knowQuery');
6
7  const router = express.Router();
8
9  // GET all knowledge entries
10 router.get('/', async (req, res) => {
11   try {
12     const knowledge = await getAllKnowledge();
13     res.json(knowledge);
14   } catch (err) {
15     res.status(500).json({ message: err.message });
16   }
17 });
18
19 // GET a single knowledge entry by ID
20 router.get('/:id', async (req, res) => {
21   try {
22     const knowledge = await getKnowledgeById(req.params.id);
23     if (knowledge == null) {
24       return res.status(404).json({ message: 'Cannot find knowledge entry' });
25     }
26     res.send(knowledge);
27   } catch (err) {
28     res.status(500).json({ message: err.message });
29   }
30 });
31
```

```
src > routes > JS knowRoutes.js > Q router.patch('/:question') callback
32 // GET a single knowledge entry by Question
33 router.get('/question/:question', async (req, res) => {
34   try {
35     const knowledge = await getKnowledgeByQuestion(req.params.question);
36     if (knowledge == null) {
37       return res.status(404).json({ message: 'Cannot find knowledge entry' });
38     }
39     res.send(knowledge);
40   } catch (err) {
41     res.status(500).json({ message: err.message });
42   }
43 });
44
45 // CREATE a new knowledge entry
46 router.post('/', async (req, res) => {
47   const knowledge = new Knowledge({
48     question: req.body.question,
49     answer: req.body.answer
50   });
51
52   try {
53     const newKnowledge = await addKnowledge(knowledge);
54     res.status(201).json(newKnowledge);
55   } catch (err) {
56     res.status(400).json({ message: err.message });
57   }
58 });
```

```

src > routes > JS knowRoutes.js > ...
60  // UPDATE a knowledge entry
61  router.patch('/:question', async (req, res) => {
62    try {
63      const updatedKnowledge = await updateKnowledgeByQuestion(req.params.question, req.body.answer);
64      res.json(updatedKnowledge);
65    } catch (err) {
66      res.status(400).json({ message: err.message });
67    }
68  });
69
70 // DELETE a knowledge entry by question
71 router.delete('/:question', async (req, res) => {
72   try {
73     const result = await deleteByQuestion(req.params.question);
74     if (result.deletedCount === 0) {
75       return res.status(404).json({ message: 'Cannot find knowledge entry' });
76     }
77     res.json({ message: `Deleted ${result.deletedCount} knowledge(s) with question "${req.params.question}"` });
78   } catch (err) {
79     res.status(500).json({ message: err.message });
80   }
81 });
82
83 module.exports = router;

```

4.2 Tata Cara Penggunaan Program

Untuk menggunakan program ini diperlukan beberapa *requirement* sebagai berikut.

1. npm
2. nodemon
3. express
4. mongoose

Pada saat program dimulai, pengguna akan dapat langsung memasukan inputnya pada kolom input. Kemudian pengguna dapat mengirimkan inputnya tersebut agar program dapat memprosesnya serta memberikan jawaban dari input tersebut.

Fitur-fitur yang ditawarkan oleh program adalah sebagai berikut:

1. Fitur pertanyaan

Untuk menggunakannya cukup dengan memasukkan pertanyaan seperti biasa pada kolom masukan. Pengguna juga dapat menanyakan beberapa hal

sekaligus dengan memisahkan pertanyaannya dengan ‘?’ . Contoh: “Siapakah presiden pertama indonesia? Siapa presiden kedua indonesia?”

2. Fitur kalkulator

Untuk menggunakan fitur ini cukup dengan memasukkan operasi matematika seperti “1+2?” atau bisa juga dengan “Hasil 1+2?”. Hasil dari pertanyaan tersebut dapat berupa hasil kalkulasinya, dan jika operasinya tidak memenuhi kaidah operasi matematika maka hasilnya berupa “Sintaks tidak valid”.

3. Fitur tanggal

Untuk menggunakan fitur ini cukup dengan memasukkan tanggal yang ingin diketahui harinya seperti “1|1|2022?”. Perlu diperhatikan bahwa program dapat menerima masukan tanggal dengan format “hari|bulan|tahun”. Hasil dari pertanyaan tersebut berupa hari tanggal tersebut atau jika tanggal tersebut tidak *valid* maka *output*-nya adalah “Tanggal tidak valid”.

4. Fitur tambah pertanyaan dan jawaban ke *database*

Untuk menggunakan fitur ini cukup dengan memasukkan perintah dengan format “tambah pertanyaan {pertanyaan} dengan jawaban {jawaban}”, dimana {pertanyaan} dan {jawaban} diganti sesuai dengan namanya. Jika pertanyaan tersebut tidak ada di *database* maka pertanyaan dan jawaban tersebut akan ditambahkan ke *database*, namun jika pertanyaan tersebut sudah ada di *database* maka pertanyaan tersebut akan diubah jawabannya dengan jawaban baru.

5. Fitur hapus pertanyaan dari *database*

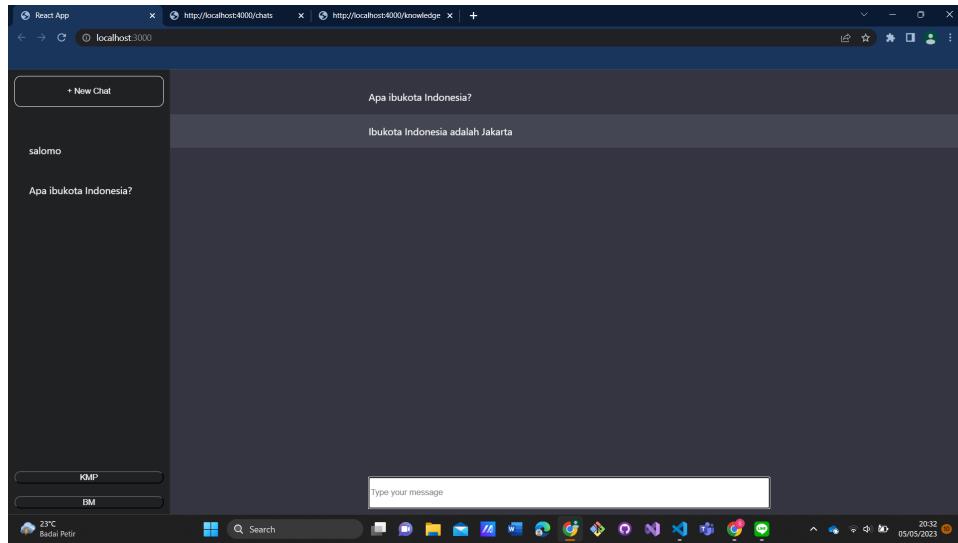
Untuk menggunakan fitur ini cukup dengan memasukkan perintah dengan format “hapus pertanyaan {pertanyaan}”. Jika pertanyaan tersebut tidak ada di

database maka hasil yang akan diberikan oleh program adalah “Tidak ada pertanyaan {pertanyaan} di *database*”. Namun jika pertanyaan tersebut ada di *database*, maka pertanyaan serta jawabannya akan dihapus dari *database*.

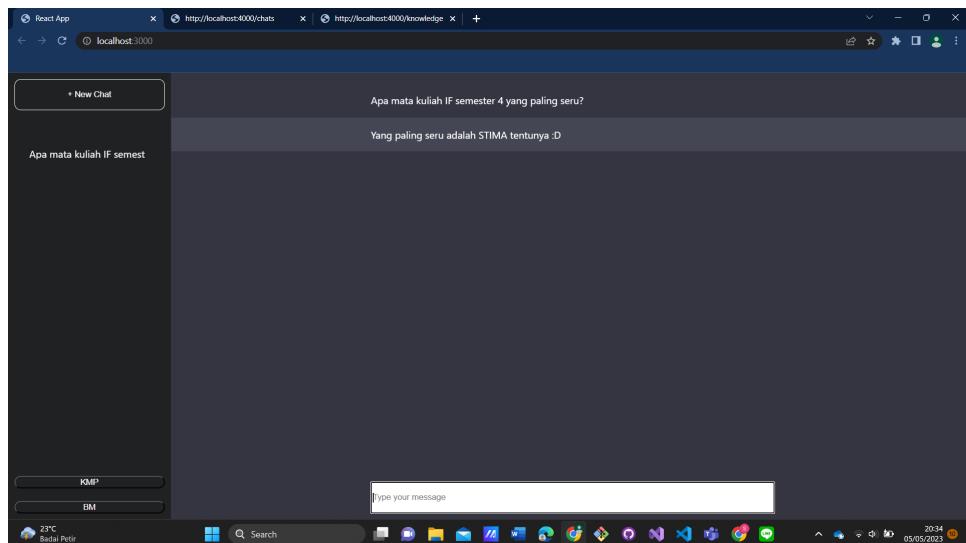
4.3 Hasil Pengujian

a. Pertanyaan Exact

i. Test 1

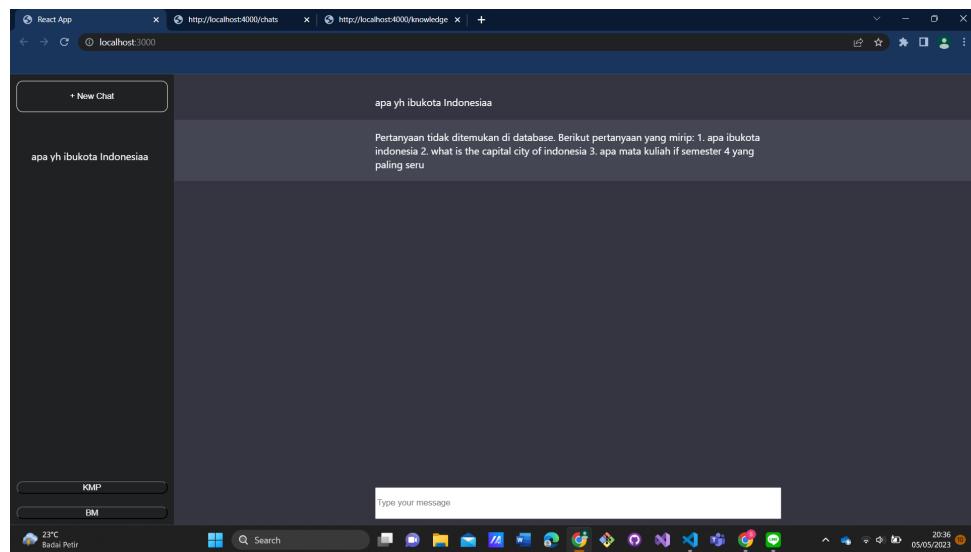


ii. Test 2

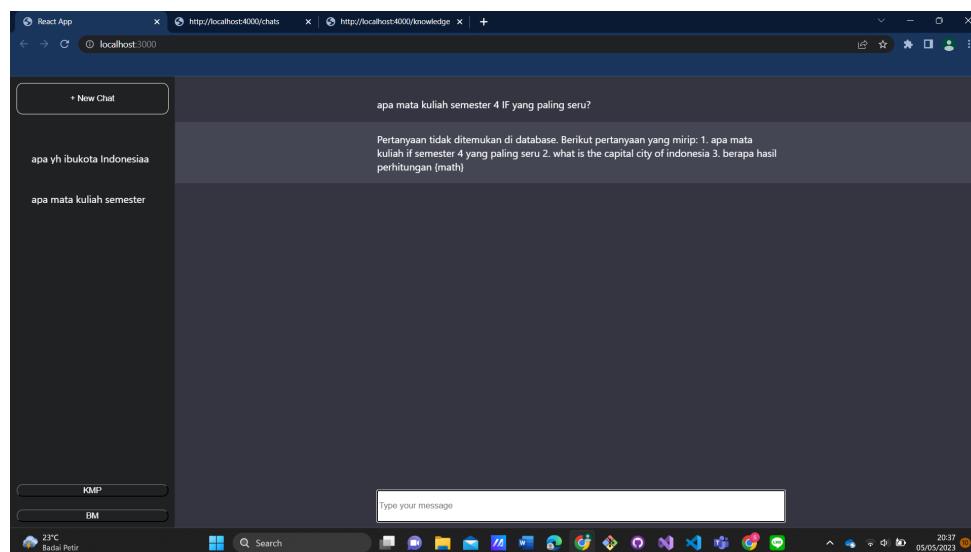


b. Pertanyaan Mirip

i. Test 1

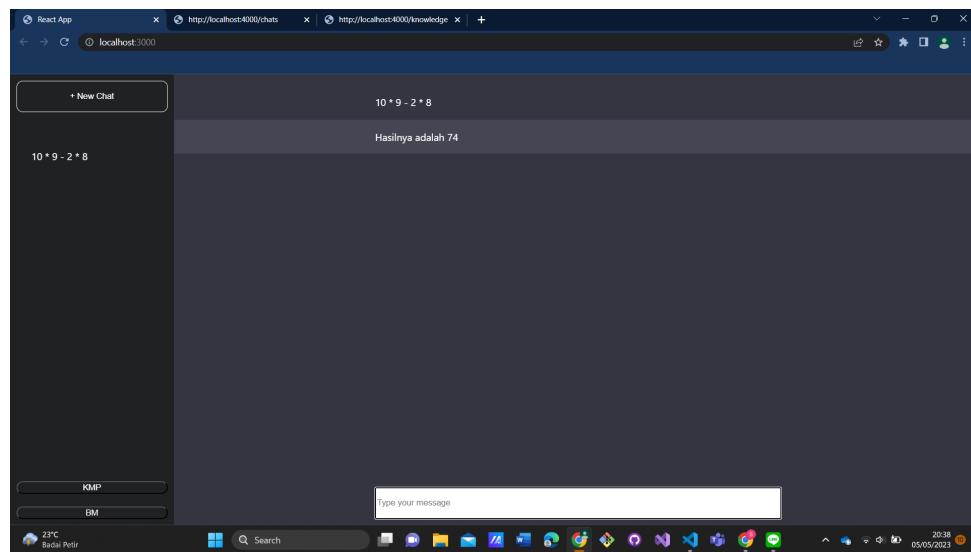


ii. Test 2

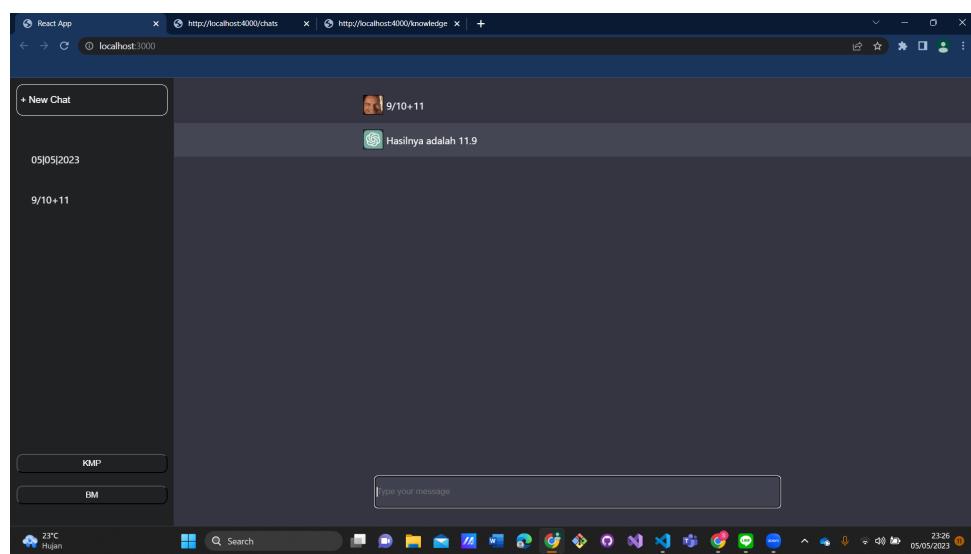


c. Pertanyaan Matematika

i. Test 1

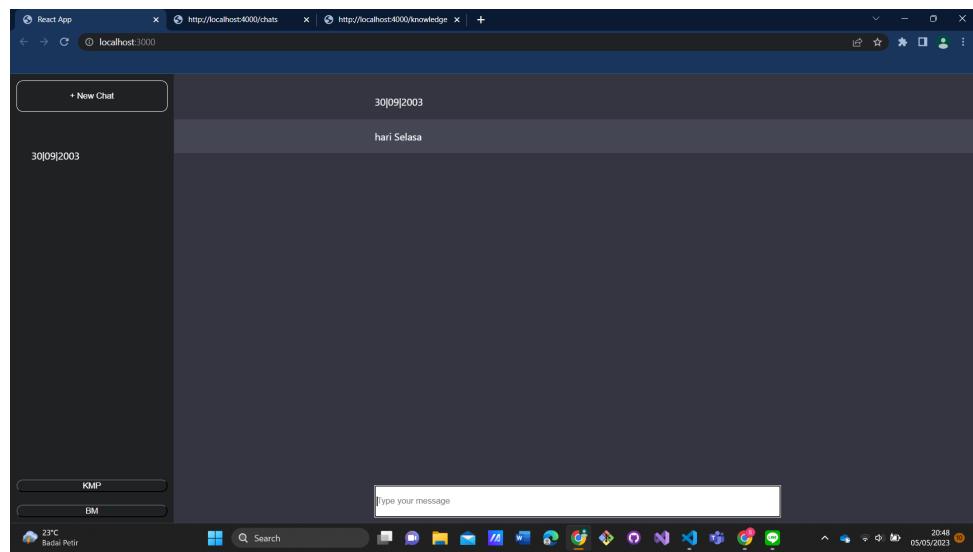


ii. Test 2

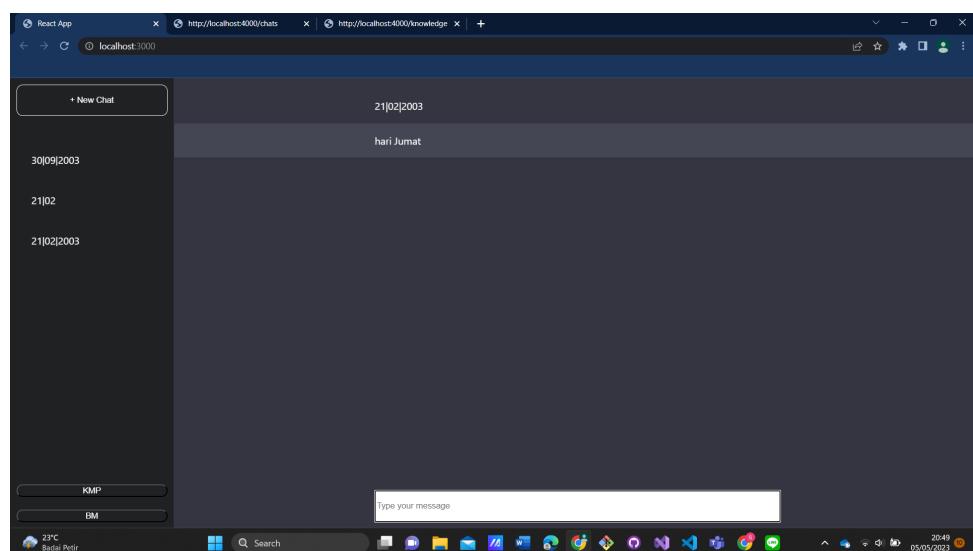


d. Pertanyaan Tanggal

i. Test 1

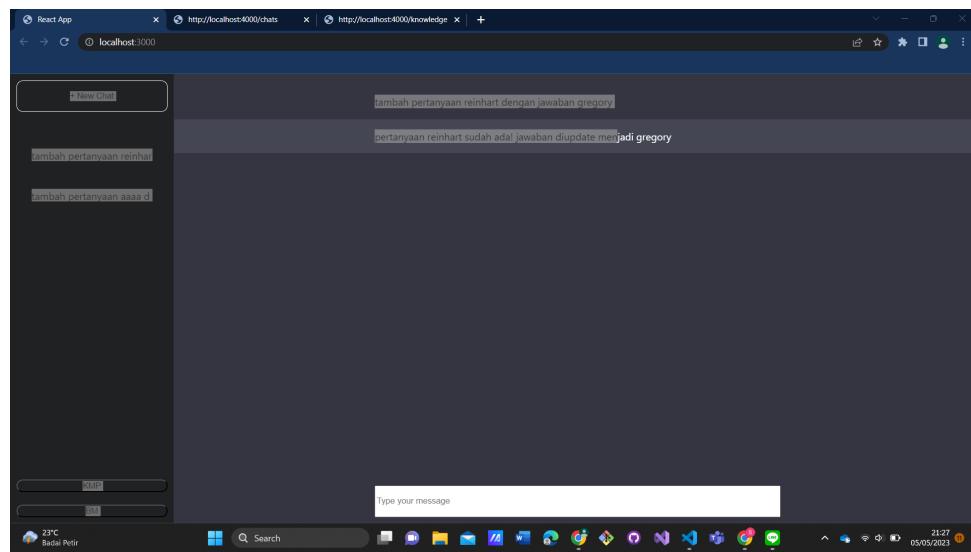


ii. Test 2

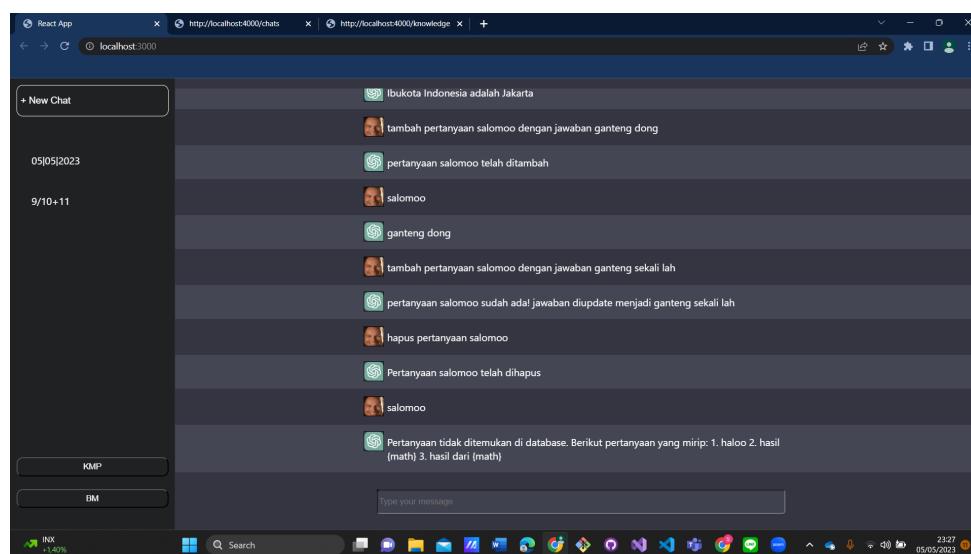


e. Hapus Pertanyaan

i. Test 1

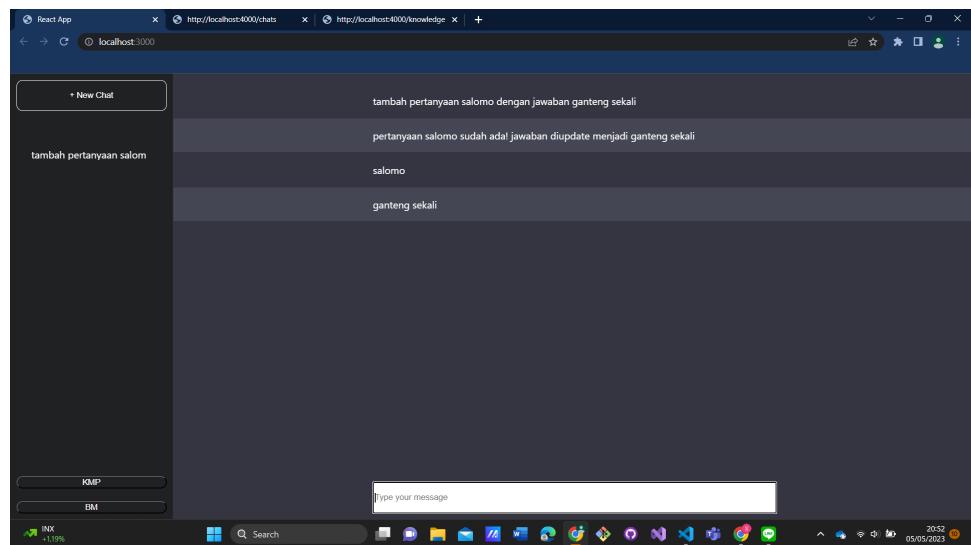


ii. Test 2

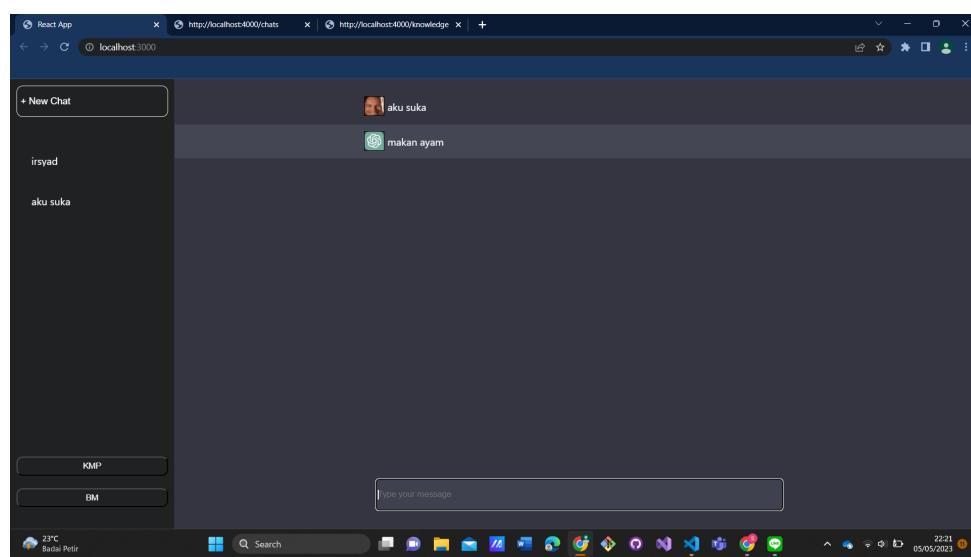
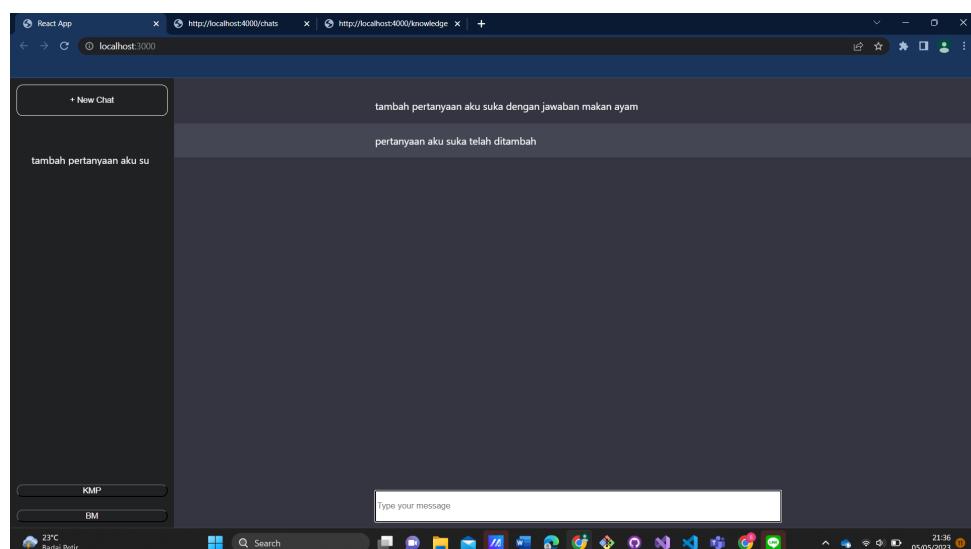


f. Tambah Pertanyaan

i. Test 1



ii. Test 2



4.4 Analisis

Dari hasil pengujian yang sudah dilakukan, sulit untuk mengetahui secara pasti lebih efektif penggunaan KMP/BM yang lebih efektif karena waktu yang diperlukan untuk mengakses *database* bisa dibilang lama. Namun secara teori, penggunaan algoritma KMP akan lebih cocok pada proses *string matching* pada program ini. Hal tersebut dikarenakan KMP lebih cocok untuk melakukan *string matching* jika *string* yang dicocokkan pendek. Karena program juga mengstandarisasi *input* dari pengguna, dan *input* dari pengguna juga bisa dibilang pendek, maka KMP akan lebih cocok dibandingkan BM.

Namun penggunaan KMP/BM pada program ini bisa dibilang kurang cocok, karena kedua algoritma tersebut digunakan untuk melakukan *exact matching* dan bukan *substring matching*. Sehingga untuk *input* yang tidak *exact* harus langsung menggunakan algoritma lain seperti *levenshtein distance*.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Regex dalam *string matching* dapat digunakan untuk menstandarisasi *string* yang ingin disamakan dengan data. Standarisasi tersebut dapat mempermudah program untuk menggeneralisasikan pertanyaan serta memberikan hasil yang sesuai. Pengeneralisasi tersebut dapat mempermudah pekerjaan algoritma seperti KMP, BM, dan Levenshtein.

5.2 Saran

Dalam pengimplementasian algoritma *string matching* akan lebih baik jika terdapat *format* yang dapat diikuti oleh seluruh *string*-nya, dengan begitu program dapat mengenal *string* dengan lebih mudah.

5.3 Refleksi

Akan lebih baik jika pembuatannya direncanakan terlebih dahulu agar semua anggota dapat mengerjakan bagiannya sesuai rencana, dan jika terdapat perubahan yang harus dilakukan maka tidak terlalu banyak perubahan yang harus dikonsiderasi oleh anggota lainnya.

5.4 Tanggapan

Mungkin penggunaan KMP dan BM bisa dibilang kurang cocok untuk program yang seperti ini, dimana kita harus melakukan *exact matching* untuk setiap *string* dan kemudian jika tidak ada yang tepat kita harus mencari *string* termirip.

DAFTAR PUSTAKA

edunex.itb.ac.id (Diakses pada tanggal 30 April 2023)

Munir, Rinaldi. 2021. “Pencocokan String (*String/Pattern Matching*)”.

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf>,
diakses 30 April 2023.

Munir, Rinaldi. 2021. “String Matching dengan Regular Expression”.

<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf>, diakses 30 April 2023.

LAMPIRAN

Pranala *repository* dapat diakses pada tautan: github.com/Ainzw0rth/Tubes3_13521063

Pranala YouTube dapat diakses pada tautan: <https://youtu.be/0xAPU9yl6I0>