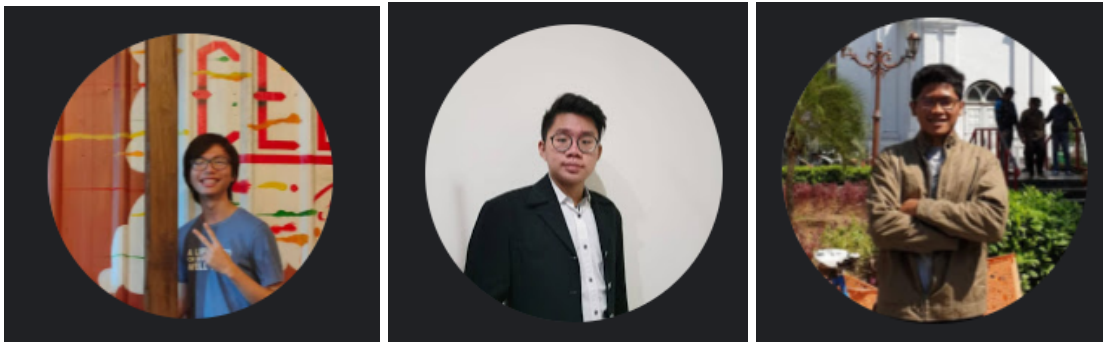


**PENERAPAN *STRING MATCHING* DAN *REGULAR EXPRESSION*
DALAM PEMBANGUNAN *DEADLINE REMINDER ASSISTANT***

Laporan Tugas Besar 3 IF2211 Strategi Algoritma
Semester II Tahun Akademik 2020/2021



Oleh:

Kelompok 17 - GhemBOT

Dionisius Darryl Hermansyah	13519058
Nathaniel Jason	13519108
Gregorius Dimas Baskara	13519190

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

2021

BAB I

DESKRIPSI TUGAS

1.1. Deskripsi tugas

Dalam tugas besar ini, kami akan diminta untuk membangun sebuah *chatbot* sederhana yang berfungsi untuk membantu mengingat berbagai deadline, tanggal penting, dan *task-task* tertentu kepada user yang menggunakannya. Dengan memanfaatkan algoritma *String Matching* dan *Regular Expression*, kami diharapkan dapat membangun sebuah *chatbot* interaktif sederhana layaknya *Google Assistant* yang akan menjawab segala pertanyaan Anda terkait informasi deadline tugas-tugas yang ada.

Deadline Reminder Assistant. akan dibangun dengan sistem *Question and Answer* dimana pengembang diharapkan sudah menyediakan kumpulan formula tertentu untuk melakukan pendeteksian setiap perbedaan command atau perintah pada aplikasi *Chatbot*. Berikut ini adalah runtutan fitur yang dimiliki oleh *Deadline Reminder Assistant* tersebut.

1. Menambahkan *task* baru

- a. Suatu kalimat diklasifikasikan sebagai suatu *task* apabila mengandung semua komponen berikut ini:
 - i. Tanggal (format dibebaskan)
 - ii. Kode Mata Kuliah / Nama Mata Kuliah (dibebaskan)
 - iii. Jenis Tugas (berdasarkan daftar kata penting yang sudah disediakan)
 - iv. Topik Tugas (tidak ada batasan)
- b. Point i sampai dengan iv diklasifikasikan menggunakan regular expression sehingga masukan kalimat benar-benar layaknya kalimat sehari-hari
- c. Jika pesan berhasil dikenali oleh assistant, maka *assistant* akan mengirim pesan balasan yang berisi ID (sesuai urutan task diinput), tanggal, kode mata kuliah, jenis tugas, dan topik tugas. Contoh pesan balasan dari bot sebagai berikut.

[TASK BERHASIL DICATAT]
(ID: 1) 14/04/2021 - IF2211 - Tubes - String matching

2. Melihat daftar *task* yang harus dikerjakan

a. Seluruh *task* yang sudah tercatat oleh assistant

- Contoh perintah: “Apa saja deadline yang dimiliki sejauh ini?”

b. Berdasarkan periode waktu

i. Pada periode tertentu (DATE_1 until DATE_2)

Contoh perintah yang dapat digunakan: “Apa saja deadline antara DATE_1 sampai DATE_2?”

ii. N minggu ke depan

Contoh perintah yang dapat digunakan: “Deadline N minggu ke depan apa saja?”

iii. N hari ke depan

Contoh perintah yang dapat digunakan: “Deadline N hari ke depan apa saja?”

iv. Hari ini

Contoh perintah yang dapat digunakan: “Apa saja deadline hari ini?”

c. Berdasarkan jenis task (kata penting)

i. Sesuai dengan daftar task yang didefinisikan

ii. User dapat melihat daftar task dengan jenis *task* tertentu

iii. Misalnya: “3 minggu ke depan ada kuis apa saja?”, maka *chatbot* akan menampilkan daftar kuis selama 3 minggu kedepan

Catatan: Eksekusi perintah pengguna bisa mencakup ketiga poin sekaligus sehingga formula pengenalan command sebaiknya dibuat sebagai satu kesatuan utuh

3. Menampilkan *deadline* dari suatu task tertentu

a. Hanya berlaku untuk task yang bersifat Tugas atau memiliki tenggat waktu

b. Misalnya: “Deadline tugas IF2211 itu kapan?”

4. Memperbaharui *task* tertentu

a. Memperbarui tanggal dari suatu *task* (dalam kehidupan nyata, tentu ada kejadian dimana deadline dari suatu task diundur)

b. Perintah yang dimasukkan meliputi 1 keyword untuk memperbaharui suatu *task* dan nomor task tertentu.

c. Misalnya:

- “Deadline task X diundur menjadi 28/04/2021” dimana X merupakan nomor ID dari suatu task.

d. Apabila *task* berhasil diperbaharui, *chatbot* akan menampilkan pesan sukses memperbaharui suatu task. Sebaliknya, *chatbot* akan menampilkan pesan *error* apabila task yang dimaksud tidak dikenali oleh *chatbot* (belum masuk ke dalam Daftar Task)

5. Menandai bahwa suatu *task* sudah selesai dikerjakan

a. Apabila *user* sudah menyelesaikan suatu *task*, maka task tersebut bisa ditandai bahwa *task* tersebut sudah selesai dan tidak perlu lagi ditampilkan pada Daftar Task selanjutnya.

b. Misalnya:

- “Saya sudah selesai mengerjakan task X” dimana X merupakan nomor ID dari suatu task.

c. Apabila perintah yang dimasukkan user bisa dieksekusi, *chatbot* akan menampilkan pesan sukses. Sebaliknya, *chatbot* akan menampilkan pesan *error* apabila *task* yang dimaksud tidak dikenali oleh *chatbot* (belum masuk ke dalam Daftar Task)

6. Menampilkan opsi *help* yang difasilitasi oleh assistant

a. Berisikan command-command yang dapat digunakan oleh *user*

b. Misalnya: “Apa yang bisa assistant lakukan?”

c. Bot akan memberikan hasil berupa daftar kata-kata yang bisa digunakan untuk menambahkan dan melihat daftar task (setiap kelompok bebas membentuknya seperti apa)

7. Mendefinisikan list kata penting terkait apakah itu merupakan suatu task atau tidak

a. Minimal terdapat 5 kata penting berbeda, contohnya adalah:

[“Kuis”, “Ujian”, “Tucil”, “Tubes”, “Praktikum”]

b. Kata penting akan digunakan pada penentuan jenis tugas dari suatu *task*.

c. Daftar kata penting tidak perlu dibuat dinamis, cukup *static* saja atau *hardcoded*.

8. Menampilkan pesan *error* jika *assistant* tidak dapat mengenali masukan user.
 - a. Masukan yang tidak termasuk ke dalam jenis pesan di poin 1 sampai 4 dapat dikategorikan sebagai masukan tak dikenali.
 - b. *Error message* dibebaskan sesuai kreativitas mahasiswa
9. (Bonus) *Chatbot* dapat memberikan rekomendasi kata jika terdapat kesalahan kata (*typo*) pada perintah yang ditulis pengguna
 - a. Berikan rekomendasi kata jika perintah masukan pengguna mismatch dengan daftar kata yang diterima *chatbot*, namun masih memiliki tingkat kemiripan di atas 75%
 - b. Ada berbagai metrik yang dapat dimanfaatkan untuk mencari kemiripan kata, salah satunya adalah *Levenshtein distance* yang diukur melalui pendekatan *dynamic programming*.

1.2. Spesifikasi program

Berikut ini merupakan spesifikasi dari program yang harus dibangun:

1. Aplikasi yang dibuat berbasis *web* (wajib) dan anda dapat menggunakan salah satu kakas *website*: PHP, Flask, Django, JavaScript.
2. Aplikasi (backend) harus menggunakan algoritma pencocokan string KMP, Boyer-Moore, dan Regex dengan menggunakan bahasa yang menunjang *regular expression*: Java, Javascript, PHP, Python.
3. Penyimpanan data-data dan pengetahuan yang diperlukan oleh Chatbot bisa didefinisikan melalui 2 cara (pilih salah satu), yaitu:
 - a. Membuat suatu *database* sederhana (penerapan Basis Data dalam Strategi Algoritma). Implementasi skema database (relasi, atribut) dibebaskan. Skema basis data tidak perlu dinormalisasi.

b. Menyimpannya dalam bentuk struktur data sendiri, pengambilan data dilakukan dengan menggunakan mekanisme *load / save* dari suatu file .txt. Struktur penyimpanan data dibebaskan.

4. Data-data yang diperlukan dan akan disimpan dalam suatu *chatbot* adalah sebagai berikut.

- a. List kata-kata penting
- b. Daftar task yang tercatat oleh *chatbot*
- c. Data-data pendukung lainnya (kreativitas kelompok)

5. Pencocokan *string* dapat anda implementasikan sesuai kriteria berikut.

Apa saja dedline yang ada sejauh ini?

Mungkin maksud kamu: Apa saja deadline yang ada sejauh

Apa saja deadline yang ada sejauh

[Daftar Deadline]

(ID: 1) 14/04/2021 - IF2211 Strategi Algoritma - Tugas Besar 3 8

a. Deteksi perintah (contoh: “Apa saja deadline yang ada sejauh ini?”) tidak dilakukan secara *exact matching* (input dibebaskan ke user, bukan programmer, selama mengandung kata kunci tertentu), anda dapat memanfaatkan regular expression dan string matching untuk mencari kata kunci dan melakukan pencocokan.

b. Rekomendasi kata: pencocokan *exact matching* (KMP, Boyer-Moore) dimanfaatkan untuk menentukan tingkat kemiripan suatu kata di perintah. Anda dapat mengembangkan algoritma yang telah diajarkan untuk menentukan kemiripan string.

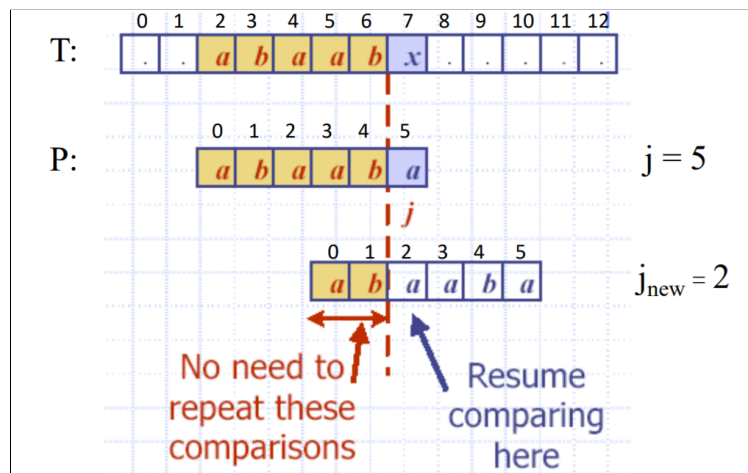
c. Pengekstrakan nilai-nilai berjenis numerik dan tanggal dilakukan dengan memanfaatkan Regular Expression.

BAB II

LANDASAN TEORI

2.1. Algoritma Knuth–Morris–Pratt

Algoritma Knuth-Morris-Pratt atau KMP ditemukan oleh Donald E. Knuth, James H. Morris, dan Vaughan R. Pratt pada tahun 1967. KMP merupakan sebuah algoritma pencocokan *string* yang melakukan iterasi secara *left-to-right*. Algoritma ini sebenarnya mirip dengan algoritma *brute force* namun memiliki cara yang lebih ‘cerdas’ dalam pencariannya. Algoritma KMP didasari oleh pertanyaan, jika terdapat *mismatch* pada teks dan *pattern* P di $P[j]$ misalnya $T[i] \neq P[j]$, berapa banyak kita dapat menggeser *pattern* sedemikian rupa sehingga perbandingan yang tidak mangkus dapat dihilangkan? Jawaban dari pertanyaan tersebut adalah prefiks terbesar dari $P[0 \dots j-1]$ yang merupakan sufiks dari $P[1 \dots j-1]$.



Gambar 2.1. Fungsi pencocokan string pada KMP yang ditingkatkan dari algoritma *brute force*

Secara umum, berikut merupakan langkah-langkah penerapan algoritma KMP dalam mencocokkan sebuah *pattern* dalam teks:

1. Algoritma ini mencocokkan karakter per karakter dari *pattern* ke karakter di teks yang bersesuaian dari kiri ke kanan.
2. Pencocokan ini akan dilakukan hingga:
 - a. Karakter di *pattern* dan di teks yang dibandingkan *mismatch* (tidak cocok).
 - b. Karakter di *pattern* dan di teks cocok seluruhnya.

- i. Jika hal ini terjadi, maka algoritma akan meng-*output* hasil.
3. Jika *pattern* belum ditemukan. Algoritma akan menggeser *pattern* berdasarkan fungsi pinggiran atau *border function*, dan mengulangi langkah-langkahnya.

Dalam menghitung fungsi pinggiran, kompleksitas algoritma KMP adalah $O(m)$, sedangkan dalam mencari *string* kompleksitasnya $O(n)$. Kompleksitas waktu algoritma KMP adalah $O(m+n)$ yang relatif sangat cepat dibandingkan *brute force* dalam pencocokan *string*.

2.2. Algoritma Boyer-Moore

Algoritma Boyer-Moore merupakan algoritma pencocokan *string* yang didasari oleh dua teknik utama. Teknik-teknik tersebut adalah:

1. Teknik *looking-glass*

Cari *pattern* P pada teks T dengan berjalan mundur pada P mulai dari akhir P.

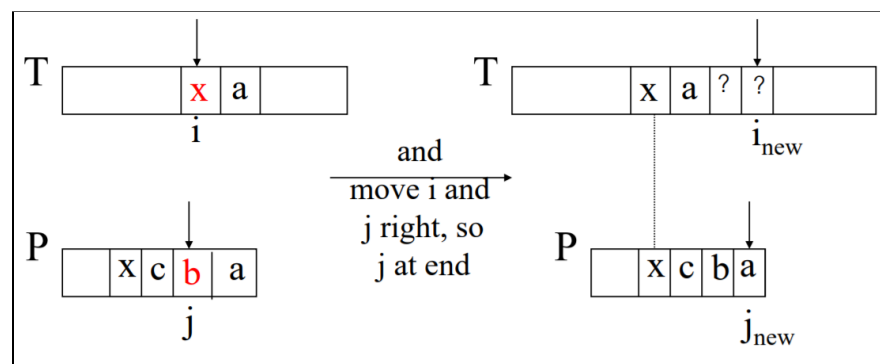
2. Teknik *character jump*

Dilakukan ketika:

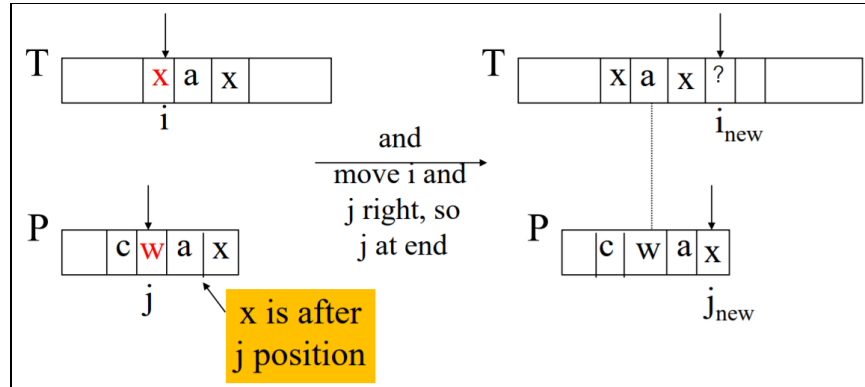
- Saat terjadi *mismatch* pada $T[i] \neq x$
- Saat $P[j]$ tidak sama dengan $T[i]$

Ada beberapa kemungkinan ‘lompatan’ yang dapat dilakukan:

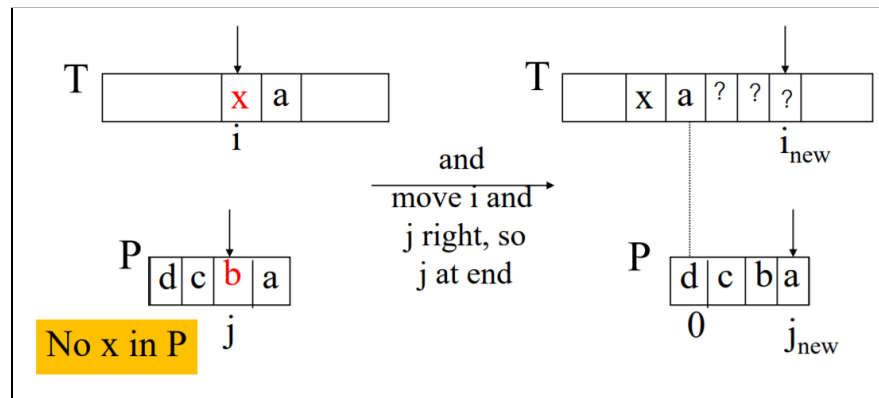
- a. Jika P mengandung karakter x, maka geser P ke kanan agar sesuai dengan *last occurrence* dari x di P dengan $T[i]$



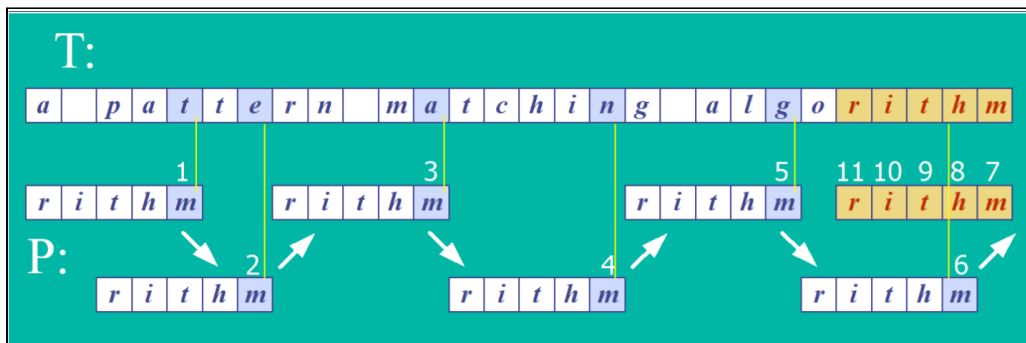
- b. Jika P mengandung karakter x, namun pergeseran P ke kanan sesuai dengan *last occurrence* tidak dimungkinkan, maka geser P ke kanan sebanyak 1 karakter dari $T[i+1]$



- c. Jika kasus (a) dan (b) tidak dihadapi maka geser P agar sesuai dengan $P[0]$ pada $T[i+1]$



Berikut merupakan contoh lengkap dari cara kerja algoritma Boyer-Moore:



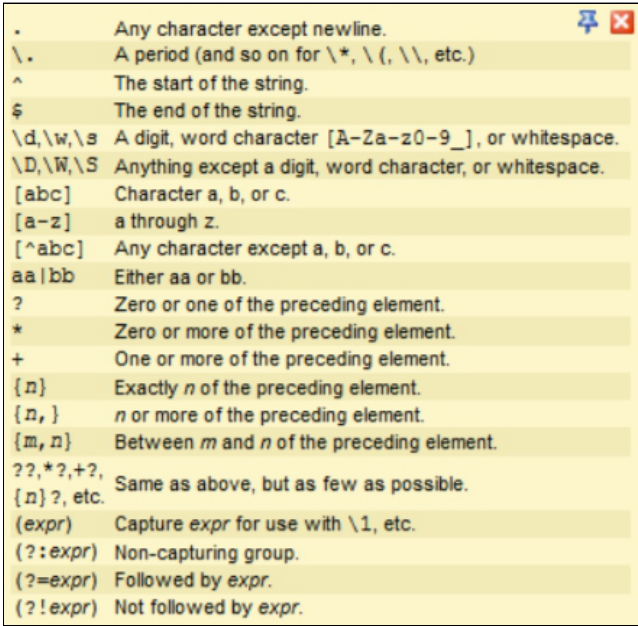
Gambar 2.2. Cara kerja algoritma Boyer-Moore

Kompleksitas algoritma Boyer-Moore memiliki *worst case* sebesar $O(nm + A)$ yang jauh lebih cepat jika dibandingkan dengan algoritma *brute force* dalam pencocokan *string* dalam

kasus jumlah alphabet yang besar seperti teks berbahasa inggris pada umumnya. Namun, algoritma ini kurang bekerja dengan baik jika jumlah alphabet sedikit seperti pada teks binari.

2.3. Regular expression

Ekspresi reguler atau *regular expression* yang biasanya disingkat sebagai regex merupakan serangkaian karakter yang mendefinisikan sebuah pola pencarian dan pencocokan *string*. Regex sangat berkaitan dengan ilmu teori komputer dan bahasa formal. Ada banyak ketentuan umum pola dalam penulisan regex, beberapa contohnya diperlihatkan oleh gambar 2.3.



.	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.

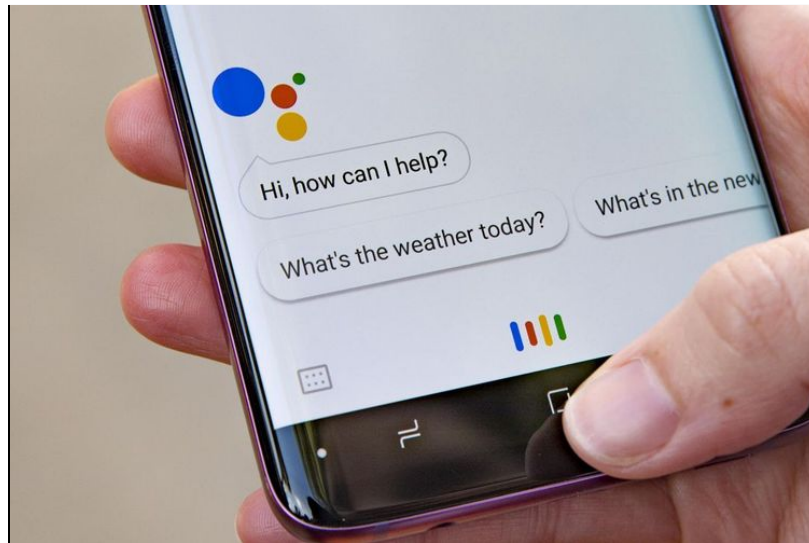
Gambar 2.3. Beberapa pola *regular expression*

Regex dapat digunakan misalnya pada pencarian kata kunci dalam berita, pengecekan kesalahan ketikan (*typo*) dan masih banyak lagi. Dalam implementasinya, banyak bahasa yang dapat digunakan untuk membuat ekspresi regex maupun menggunakannya secara langsung, misalnya pada Python, Java, JavaScript, dan PHP.

2.4. Chatbot

Chatbot merupakan sebuah program komputer yang dirancang sedemikian rupa untuk melakukan simulasi percakapan intelektual seperti sedang berbicara dengan manusia secara audio maupun teks. Tidak sedikit *chatbot* yang tampaknya dapat mengartikan dan menanggapi

input manusia yang sedang berinteraksi dengannya. Nyatanya, beberapa *chatbot* tersebut hanya memindai kata kunci dalam *input*, yang dapat berupa pesan teks, dan membalasnya dengan kata kunci yang paling cocok, atau pola kata-kata yang paling mirip dari basis data yang telah disediakan oleh pembuat *chatbot*.



Gambar 2.4. *Google assistant* yang merupakan *personal assistant chatbot* ternama di dunia

Di zaman modern ini, *chatbot* sudah berkembang dari yang awalnya hanya dapat bercakap-cakap dengan manusia, menjadi dapat membantu kehidupan manusia sehari-hari. Beberapa contoh kasusnya adalah *chatbot* yang digunakan untuk belanja *online*, *chatbot* pembantu layanan aplikasi, dan *personal assistant chatbot*. *Personal assistant chatbot* dapat membantu penggunanya dalam berbagai hal seperti mengingatkan sebuah acara, mengatur *alarm* pada *smartphone*, dan lain-lan. Salah satu *chatbot* yang berperan sebagai *personal assistant* misalnya adalah *Google Assistant* dan *Amazon Alexa*.

BAB III

ANALISIS PEMECAHAN MASALAH

3.1. Langkah penyelesaian masalah

3.1.1. Menambah task baru

Pertama Chatbot akan menerima input dari pengguna berupa kalimat sehari-hari. Setelah input dimasukkan oleh pengguna, maka Chatbot akan menggunakan regular expression untuk mencari komponen-komponen:

1. Tanggal dengan format DD/MM/YYYY (DD: hari, MM: bulan, YYYY: tahun). Dicari menggunakan **regular expression**.
2. Kode Mata Kuliah dengan format AABBBB (AA: kode prodi, BBBB: kode mata kuliah). Dicari menggunakan **regular expression**.
3. Jenis tugas, yang terdiri dari kuis, ujian, tubes, tucil, tugas, dan PR (sesuai daftar task yang ada).
4. Topik tugas

Apabila ditemukan, maka Chatbot akan menyimpan *task* tersebut di dalam database dan mengembalikan respon:

[TASK BERHASIL DICATAT]
(ID: X) DD/MM/YYYY - AABBBB - <Jenis tugas> - <Topik tugas>

3.1.2. Melihat daftar task yang harus dikerjakan

Terdapat tiga cara untuk melihat daftar task berdasarkan tiga macam input pengguna:

1. Melihat seluruh task tanpa filter
Chatbot akan menerima input berupa kalimat sehari-hari dan mencari apakah ada frasa “**Apa saja deadline yang dimiliki sejauh ini?**” Apabila ditemukan frasa tersebut, maka seluruh task akan ditampilkan kembali ke layar
2. Melihat task berdasarkan periode waktu
Terdapat beberapa cara untuk menampilkan task berdasarkan periode waktu berdasarkan input pengguna, yaitu:
 - a. Pada periode tertentu

Input mengandung komponen:

- Frasa “**deadline**”
- Frasa **DATE_1** dengan format DD/MM/YYYY
- Frasa **DATE_2** dengan format DD/MM/YYYY

b. N minggu ke depan

Input mengandung komponen:

- Frasa “**deadline**”
- Frasa “**N minggu**”. Dengan N sebuah integer

c. N hari ke depan

Input mengandung komponen:

- Frasa “**deadline**”
- Frasa “**N hari**”. Dengan N sebuah integer

d. Hari ini

Input mengandung komponen:

- Frasa “**deadline**”
- Frasa “**hari ini**”.

Apabila terdapat salah satu dari empat frasa tersebut, maka seluruh task dengan deadline pada periode yang berkesesuaian akan ditampilkan ke layar

3. Melihat task berdasarkan jenis task (kata penting)

Apabila di dalam kalimat yang di input terkandung frasa jenis task yang terdapat pada daftar task yang ada, maka hasil yang ditampilkan ke layar adalah hasil yang sudah di-filter berdasarkan jenis task tersebut

Ketiga jenis di atas dapat digabungkan satu dengan lainnya, maka ketika menerima input dari pengguna, Chatbot harus dapat menangkap seluruh permintaan secara utuh dengan menggunakan **algoritma String Matching dan regular expression**.

Apabila ditemukan, maka Chatbot akan mengembalikan respon:

```
[DAFTAR DEADLINE]
(ID: 1) DD/MM/YYYY - AABBBB - <Jenis tugas 1> - <Topik tugas 1>
(ID: 2) DD/MM/YYYY - AABBBB - <Jenis tugas 2> - <Topik tugas 2>
...
```

3.1.3. Menampilkan deadline

Pada fitur ini, pengguna dapat memberikan input kalimat yang memiliki komponen berikut ini:

1. Frasa “**Deadline**” atau “**Kapan**”
2. Frasa “**Tugas**” (atau jenis task lain yang memiliki tenggat) diikuti dengan kode prodi (AABBBB) atau topik tugas (**menggunakan regex**)

Pencarian frasa tersebut akan menggunakan **algoritma String Matching** dan **regular expression**. Apabila ditemukan, maka Chatbot akan mengembalikan respon:

DD/MM/YYYY
(yang merupakan deadline dari task terkait)

3.1.4. Memperbaharui task tertentu

Dimungkinkan bagi pengguna untuk mengubah tanggal deadline bagi task tertentu yang memiliki tenggat/deadline. Untuk menjalankan fitur ini, Chatbot akan menerima input dari pengguna dan menggunakan **KMP dan regular expression** akan mencari komponen-komponen berikut:

1. Frasa “**Task**” (dapat juga berupa frasa “tugas”) yang diikuti dengan ID dari task tersebut.
2. Frasa “**Diundur/Dimajukan/Diubah**”
3. Frasa DD/MM/YYYY yang merupakan tanggal setelah diubah (menggunakan regex)

Apabila berhasil diperbaharui, maka Chatbot akan memperbaharui deadline tersebut di dalam database dan mengembalikan respon:

Deadline Task X berhasil diubah

Apabila tidak ditemukan task dengan ID terkait, maka Chatbot akan mengembalikan respon:

Task dengan ID X tidak ditemukan

3.1.5. Menandai bahwa suatu task sudah selesai dikerjakan

Dimungkinkan bagi pengguna untuk menandai suatu task yang telah terselesaikan. Untuk menjalankan fitur ini, Chatbot akan menerima input dari pengguna dan menggunakan **regular expression** akan mencari komponen-komponen berikut:

1. Frasa **“Task”** (dapat juga berupa frasa “tugas”) yang diikuti dengan ID dari task tersebut.
2. Frasa **“Selesai”**

Apabila berhasil dicatat, maka Chatbot akan memperbaharui status task tersebut di dalam database dan mengembalikan respon:

```
Task X telah terselesaikan
```

Apabila tidak ditemukan task dengan ID terkait, maka Chatbot akan mengembalikan respon:

```
Task dengan ID X tidak ditemukan
```

3.1.6. Menampilkan opsi help yang difasilitasi oleh assistant

Pada fitur ini, Chatbot akan menerima input dari pengguna dan mencari frasa **“Apa yang bisa assistant lakukan?”** Setelah itu, Chatbot akan mengembalikan respon berupa:

```
[Fitur]
  1. Fitur 1
  2. Fitur 2
  3. Fitur 3
  ...
[Daftar kata penting]
  1. Kuis
  2. Ujian
  3. Tugas
  4. Tucil
  5. Tubes
```

...

3.1.7. Mendefinisikan list kata penting

Mendefinisikan kata-kata penting secara statis yang terkait suatu task atau tidak. Kata-kata penting yang terkait suatu task akan menentukan jenis task yang tercatat pada Chatbot. List kata penting yang dicatat adalah:

1. Kuis
2. Ujian
3. Tupil
4. Tubes
5. Tugas
6. Ulangan
7. Praktikum

3.1.8. Menampilkan pesan error jika assistant tidak dapat mengenali masukan user

Apabila frasa tidak dikenali, maka Chatbot akan mengembalikan respon, namun tidak terbatas pada:

Maaf, Pesan tidak dikenali

3.1.9. Memberikan rekomendasi kata jika terdapat kesalahan kata pada perintah yang ditulis pengguna

Dengan menggunakan Levenshtein distance yang diukur melalui pendekatan dynamic programming, maka untuk setiap input frasa yang setidaknya memiliki kemiripan 75% namun tidak sesuai 100%, Chatbot akan memberikan rekomendasi frasa terdekat sesuai dengan frasa yang diketahui Chatbot.

Apabila frasa tidak diketahui, namun rekomendasi mungkin dilakukan, maka Chatbot akan mengembalikan respon:

Mungkin maksud kamu:
<Frasa Rekomendasi>

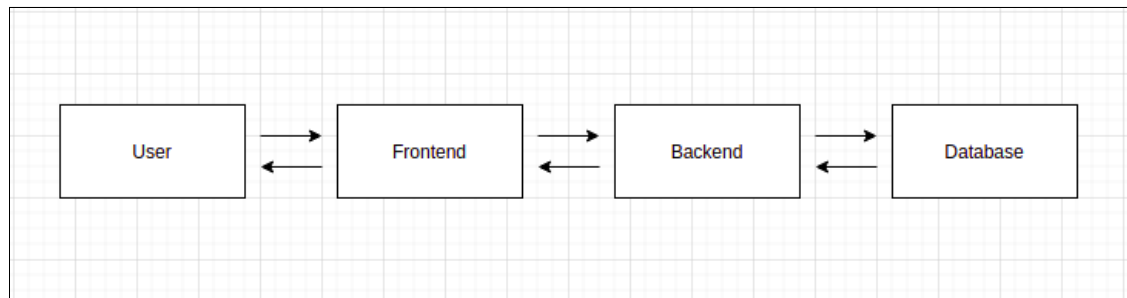
3.2. Fitur fungsional dan arsitektur chatbot

3.2.1. Fitur fungsional

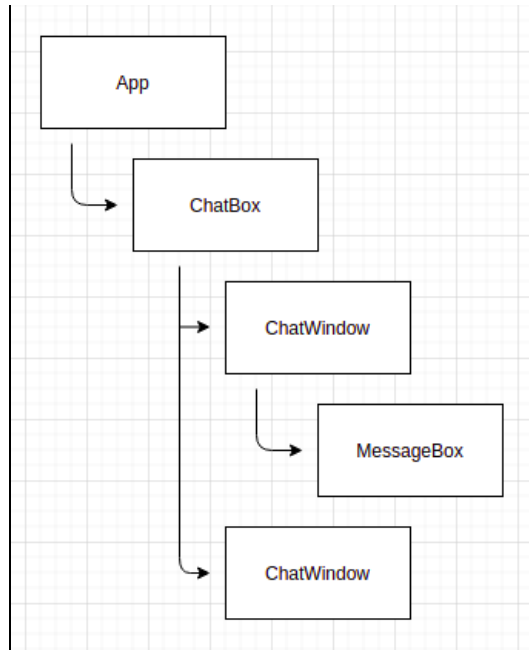
1. Menambahkan task baru
Menyimpan atau menambahkan *task* baru ke database.
2. Melihat daftar task yang harus dikerjakan
Menampilkan seluruh *task* yang sudah tercatat oleh assistant berdasarkan periode waktu atau / dan berdasarkan jenis *task*.
3. Menampilkan deadline
Menampilkan *list deadline* dari *task* yang memiliki tenggat waktu seperti *tubes*, *tucil*, *tugas*, dan *task*.
4. Memperbarui task tertentu
Memperbarui tanggal dari suatu *task* tertentu, diundur atau dimajukan tanggalnya. Chatbot akan menampilkan pesan sukses jika berhasil dieksekusi. Jika tidak dikenali *task* yang dimaksud maka akan menampilkan pesan error.
5. Menandai bahwa suatu task sudah selesai dikerjakan
Apabila *user* sudah menyelesaikan suatu *task*, maka *task* tersebut bisa ditandai bahwa *task* tersebut sudah selesai dan tidak perlu lagi ditampilkan pada daftar *task* selanjutnya. Chatbot akan menampilkan pesan sukses jika berhasil dieksekusi. Jika tidak dikenali *task* yang dimaksud maka akan menampilkan pesan error.
6. Menampilkan opsi help yang difasilitasi oleh assistant
Menampilkan *command* atau fitur apa saja yang dapat digunakan oleh user serta kata penting yang dikenali.
7. Menampilkan pesan error jika assistant terdapat kesalahan kata pada perintah yang ditulis pengguna
Mengembalikan pesan error jika frasa tidak dikenali oleh Chatbot.
8. Memberikan rekomendasi kata
Chatbot akan memberikan rekomendasi kata jika terdapat kata yang *typo* atau salah ditulis, jika kemiripannya diatas 75%.

3.2.2. Arsitektur

Arsitektur dari Chatbot yang dibangun terdiri dari 3 komponen utama, yaitu Frontend, Backend, dan Database. Frontend berfungsi untuk menampilkan data atau informasi yang tersedia dalam bentuk yang indah kepada pengguna, menerima masukan dari pengguna, dan meneruskan data dari masukan ke Backend agar diproses. Frontend berhubungan langsung dengan pengguna dan Backend. Backend berfungsi untuk memproses masukan dari pengguna yang diterima oleh Frontend dan melakukan modifikasi pada Database. Backend berhubungan langsung dengan Frontend dan Database. Database berfungsi untuk menyimpan data yang dimiliki. Database berhubungan langsung dengan Backend.



Framework Frontend yang digunakan adalah ReactJS. ReactJS bekerja sangat baik untuk User Interface atau tampilan yang dapat dibagi menjadi komponen - komponen. Pada Chatbot yang dibuat, ada satu komponen utama bernama App dan juga ada komponen buatan bantuan lainnya seperti ChatBox, ChatInput, ChatWindow, dan MessageBox. Suatu komponen dapat terdiri dari komponen lainnya. Arsitektur dari komponen yang ada di Frontend dapat dilihat pada diagram berikut.



Framework Backend yang digunakan adalah Flask. Program yang menggunakan Framework Flask ditulis dalam bahasa Python. Salah satu alasan kami menggunakan Flask adalah penulisan yang cukup mudah dan fleksibel, terutama untuk melakukan perhitungan. Backend akan menerima data dari Frontend berupa POST *request* ke endpoint “/bot”. Data yang diperlukan oleh Backend akan diambil dari *request body* yang diberikan Frontend dalam bentuk data JSON. Setelah melakukan komputasi, endpoint “/bot” akan memberikan *response* berupa jawaban dari Chatbot dalam bentuk data JSON. Jika diperlukan untuk melakukan modifikasi pada Database, maka Backend yang akan melakukan modifikasi itu.

Database yang digunakan untuk Chatbot ini adalah Firebase, atau lebih spesifik, Database yang digunakan adalah fitur Firestore dari Firebase. Firestore adalah basis data NoSQL. Berbeda dari basis data SQL pada umumnya, bentuk data yang disimpan pada suatu tabel dapat berbeda. Secara singkat, tabel pada basis data SQL mirip dengan Collection pada Firestore. Pada basis data SQL, suatu tabel dapat memiliki *row*. *Row* pada basis data SQL mirip dengan Document atau Doc pada Firestore.

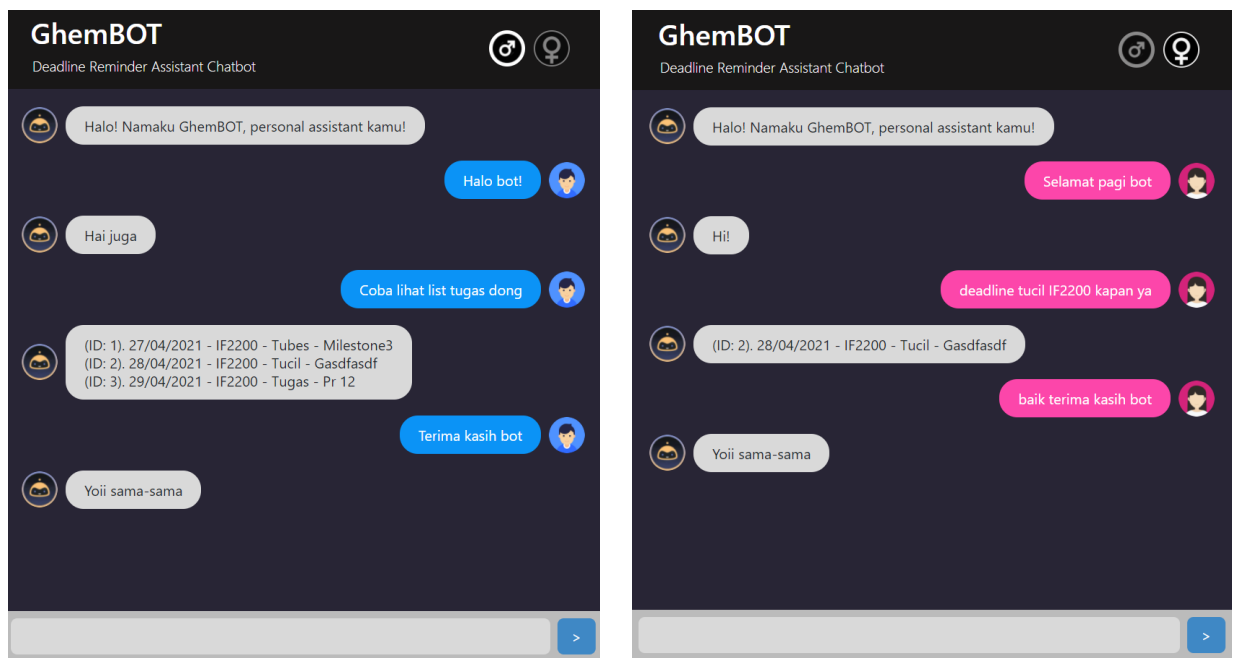
BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1. Spesifikasi teknis program

Program *chatbot* yang dibangun dalam bentuk *website* ini diberi nama GhemBOT. GhemBOT dibangun menggunakan teknologi Flask dari Python untuk *back-end* dan React dari JavaScript untuk *front-end*. GhemBOT juga menggunakan sebuah basis data NoSQL *online* yaitu Firebase. Gambar 4.1. menunjukkan tampilan utama dari GhemBOT. GhemBOT di-*deploy* menggunakan platform Netlify (*front-end*) dan Heroku (*back-end*) pada link berikut:

<https://ghembot.netlify.app/>



Gambar 4.1. Tampilan utama GhemBOT

Dalam implementasinya, GhemBOT menggunakan berbagai macam struktur data, fungsi, dan prosedur dalam bahasa pemrograman terkait. Tabel 4.1. menjelaskan spesifikasi teknis tersebut secara detail.

Tabel 4.1. Spesifikasi teknis program

Struktur Data, Fungsi, dan Prosedur	Keterangan
Backend	
algorithm.py	Modul algoritma utama
KMP	Algoritma KMP untuk mencari <i>pattern</i> pada string. Memiliki mode <i>case sensitive</i> dan tidak.
findLPS	Mencari <i>longest prefix suffix</i> (lps) dari <i>pattern</i>
isPatternExistKMP	Mengembalikan <i>true</i> jika salah satu dari <i>list</i> pattern terdapat pada string
date.py	Modul tanggal utama
date_to_str	Me-return string berformat DD/MM/YYYY dari sebuah datetime
str_to_date	Mereturn sebuah objek <i>date</i> dari string x. x dapat berformat DD/MM/YYYY atau DD-MM-YYYY atau DD MM YYYY
parse.py	Modul <i>parser</i> utama
case_add_task	Me-return sebuah objek result dengan message jika string x merupakan command untuk menambah sebuah task baru
case_mark_task_done	Me-return sebuah objek result dengan message jika <i>string</i> x merupakan <i>command</i> untuk menandai sebuah <i>task</i> telah selesai dikerjakan
case_show_all_task	Me-return sebuah objek result dengan message jika <i>string</i> x merupakan <i>command</i> untuk menampilkan seluruh <i>task</i> atau berdasarkan kasus tertentu
case_update_task	Me-return sebuah objek result dengan message jika <i>string</i> x merupakan <i>command</i> untuk mengupdate sebuah <i>task</i>

case_get_deadline_task	Me-return sebuah objek result dengan message jika <i>string</i> x merupakan <i>command</i> untuk menampilkan deadline dari task tertentu
case_error	Me-return sebuah objek result dengan message jika <i>string</i> x merupakan <i>command</i> invalid
case_help	Me-return sebuah objek result dengan message jika <i>string</i> x merupakan <i>command</i> yang meminta bantuan
case_other	Me-return sebuah objek result dengan message jika <i>string</i> x merupakan <i>command</i> lain-lain seperti mengucapkan salam dan terima kasih
parse	Fungsi utama untuk melakukan <i>parsing</i> terhadap sebuah <i>string</i> berdasarkan kasus yang telah didefinisikan
regex.py	Modul pemrosesan <i>regular expression</i> utama
get_date	Mengambil tanggal yang ada dari <i>string</i> . Asumsikan tanggal selalu valid. Dapat berformat DD/MM/YYYY atau D/MM/YYYY atau D/M/YYYY. Dapat menggunakan / atau - atau spasi
get_matkul	Mengambil kode mata kuliah yang ada dari <i>string</i> . Asumsikan tanggal selalu valid berformat XXYYYY seperti IF1212
get_number	Mengambil angka-angka yang ada dalam <i>string</i>
get_all_same_pattern	Mengambil seluruh pattern dari listPatt yang terdapat pada <i>string</i>
util.py	Modul utama utilitas lain
firestoreQueryResultsToDictArray	Mengkonversi hasil <i>query</i> dari basis data Firestore ke dalam <i>array</i>
getNewId	Menghasilkan sebuah ID baru untuk data baru yang ingin dimasukkan ke dalam basis data
Frontend	

App.js	Komponen utama aplikasi <i>frontend</i>
ChatBox.js	Komponen kotak tampilan chat
ChatInput.js	Komponen untuk meminta input pesan pengguna
ChatWindow.js	Komponen jendela <i>chat</i>
MessageBox.js	Komponen kotak tampilan <i>message</i> dari pengguna ataupun bot

4.2. Tata cara penggunaan

Berikut ini merupakan daftar dari fitur-fitur yang disediakan oleh GhemBOT:

1. Live pada pranala berikut <https://ghembot.netlify.app/>
2. Memilih tampilan *message box* berdasarkan gender pengguna
3. Melihat seluruh daftar tugas yang harus dikerjakan dengan mode:
 - a. Periode
 - b. N minggu ke depan
 - c. N hari ke depan
 - d. Hari ini
 - e. Seluruh tugas
4. Menambahkan tugas baru
5. Memperbarui task tertentu
6. Menampilkan deadline dari tugas tertentu
7. Menandai sebuah tugas telah selesai dikerjakan
8. Menampilkan opsi *help* yang difasilitasi oleh *assistant*
9. Menampilkan pesan *error* jika *assistant* tidak dapat mengenali masukan pengguna
10. Menyapa pengguna
 - a. Mengucapkan salam
 - b. Mengucapkan terima kasih
11. Menampilkan rekomendasi kata

Untuk menjalankan GhemBOT *personal assistant* pada komputer lokal anda, ikuti langkah-langkah berikut:

Live App

Kunjungi pranala berikut

<https://ghembot.netlify.app/>

Requirements

1. Python dan Flask
2. Node dan NPM

Backend

1. Pada *source code* yang tersedia, masuk ke dalam `./src/backend`
2. Anda dapat menjalankan *virtual environment* sesuai instruksi pada readme yang ada namun hal ini opsional
3. *Install dependencies* yang ada dengan

```
pip install -r requirements.txt
```

4. Untuk menjalankan server backend pada localhost, jalankan

```
$env:FLASK_APP = "main.py"
flask run
```

Frontend

1. Pada *source code* yang tersedia, masuk ke dalam `./src/frontend`
2. *Install dependencies* yang ada dengan

```
npm install
```

3. Untuk menjalankan frontend pada localhost, jalankan

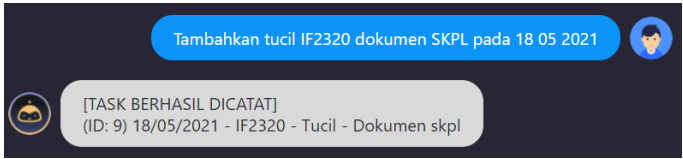
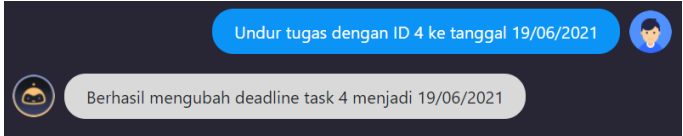
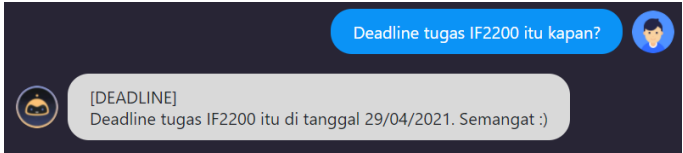
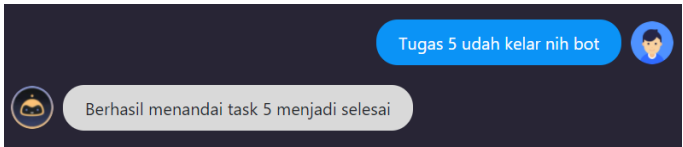
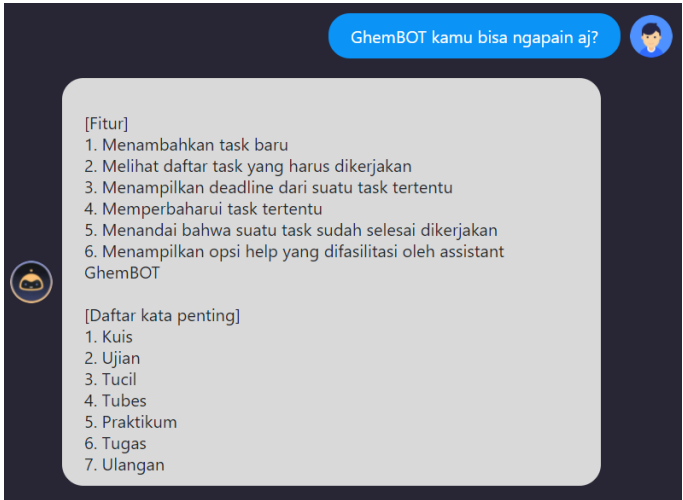

```
npm start
```


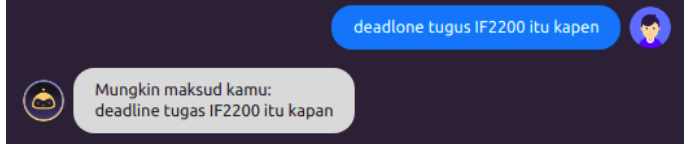

4.3. Hasil pengujian

Berikut ini merupakan hasil pengujian dari beberapa *test case* yang diberikan kepada GhemBOT.

Tabel 4.2. Hasil pengujian *chatbot* GhemBOT

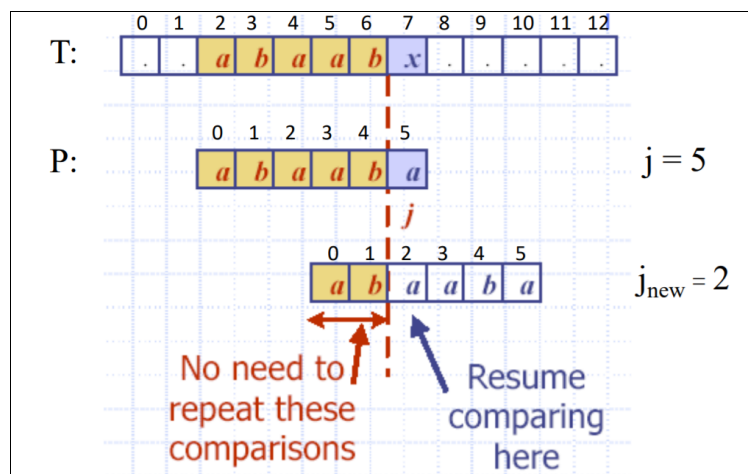
<i>Test case</i> pesan pengguna	Output pesan
Halo bot :)	
Bot tolong lihat ada tugas apa aj makasihh	
Tanggal 05-05-2021 sampai 15 05 2021 ada tugas apa aja bot	
10 minggu ke depan ada tubes apa aja nih	
Tugas buat 1 hari ke depan	
Hari ini ada kuis apa saja	

Tambahkan tucil IF2320 dokumen SKPL pada 18 05 2021	 <p>Tambahkan tucil IF2320 dokumen SKPL pada 18 05 2021</p> <p>[TASK BERHASIL DICATAT] (ID: 9) 18/05/2021 - IF2320 - Tucil - Dokumen skpl</p>
Undur tugas dengan ID 4 ke tanggal 19/06/2021	 <p>Undur tugas dengan ID 4 ke tanggal 19/06/2021</p> <p>Berhasil mengubah deadline task 4 menjadi 19/06/2021</p>
Deadline tugas IF2200 itu kapan?	 <p>Deadline tugas IF2200 itu kapan?</p> <p>[DEADLINE] Deadline tugas IF2200 itu di tanggal 29/04/2021. Semangat :)</p>
Tugas 5 udah kelar nih bot	 <p>Tugas 5 udah kelar nih bot</p> <p>Berhasil menandai task 5 menjadi selesai</p>
GhemBOT kamu bisa ngapain aj?	 <p>GhemBOT kamu bisa ngapain aj?</p> <p>[Fitur] 1. Menambahkan task baru 2. Melihat daftar task yang harus dikerjakan 3. Menampilkan deadline dari suatu task tertentu 4. Memperbaharui task tertentu 5. Menandai bahwa suatu task sudah selesai dikerjakan 6. Menampilkan opsi help yang difasilitasi oleh assistant GhemBOT</p> <p>[Daftar kata penting] 1. Kuis 2. Ujian 3. Tucil 4. Tubes 5. Praktikum 6. Tugas 7. Ulangan</p>
Asdfghjkl	 <p>Asdfghjkl</p> <p>Aku nggak ngerti :(</p>

Terima kasih bot!	
deadlone tugas IF2200 itu kapan	

4.4. Analisis hasil pengujian

Berdasarkan hasil pengujian, GhemBOT dapat menjalankan seluruh fitur yang dimilikinya dengan baik dan benar. GhemBOT dapat memproses *message* yang dibuat sebebas mungkin oleh pengguna dan berhasil menginterpretasikan seluruh *message* yang ada berdasarkan tujuan dari *message* tersebut.



Gambar 4.2. Cara kerja algoritma KMP

Proses *string* dan *pattern matching* yang menggunakan algoritma KMP dan *regular expression* dapat dianalisis kompleksitas algoritmanya dari masing-masing metode. Dalam menghitung fungsi pinggiran, kompleksitas algoritma KMP adalah $O(m)$ dimana m adalah panjang pattern, sedangkan dalam mencari *string* kompleksitasnya $O(n)$ dimana n adalah panjang string.

Kompleksitas waktu algoritma KMP adalah $O(m+n)$ yang relatif sangat cepat dibandingkan *brute force* dalam pencocokan *string*. Untuk *regular expression*, dapat diasumsikan kompleksitas algoritma adalah $O(n)$ dimana n adalah panjang *string*. Melalui analisis ini, dapat ditarik kesimpulan bahwa secara keseluruhan, untuk pengecekan sebuah *string* dalam semua kasus menggunakan algoritma KMP dan *regex* memiliki kompleksitas algoritma berturut-turut $O(m+n)$ dan $O(n)$ yang relatif lebih cepat jika dibandingkan dengan algoritma *string matching* lain seperti *brute force*.

Berdasarkan analisis tersebut, maka telah dibuktikan bahwa *personal deadline reminder assistant* dalam bentuk *chatbot* dapat diimplementasikan menggunakan algoritma *string matching* yaitu algoritma Knuth-Morris-Pratt dan *regular expression*.

BAB V

KESIMPULAN, SARAN, DAN KOMENTAR

5.1. Kesimpulan

Kesimpulan dari tugas besar ini adalah algoritma *string matching* yaitu algoritma Knuth-Morris-Pratt dan *regular expression* dapat dimanfaatkan secara nyata dalam wujud sebuah *deadline reminder assistant* berbasis *web* yang dibangun menggunakan kakas Python (Flask) dan JavaScript (React.js) serta basis data Firebase.

5.2. Saran

Dalam pengerjaan proyek yang bertopik serupa kedepannya, penulis memberikan beberapa saran yaitu:

- a. Membuat pengembangan lebih lanjut dari *chatbot* dengan algoritma yang lebih kompleks dan mangkus sehingga dapat meningkatkan interaktivitas *chatbot* .
- b. Mengembangkan *chatbot* agar dapat digunakan pada *platform* lain seperti pada LINE.

5.3. Refleksi dan komentar

Penulis berterima kasih kepada dosen dan asisten IF2211 Strategi Algoritma karena telah merancang tugas besar 3 ini secara detail dan aplikatif. Melalui pengerjaan tugas besar 3 IF2211 Strategi Algoritma ini, penulis memperoleh banyak hal baik dari segi akademik maupun non-akademik. Penulis dapat memahami materi tentang algoritma pencocokan *string* serta aplikasinya secara lebih mendalam dan jelas. Melalui implementasi algoritma-algoritma secara nyata dalam program, hal ini membantu penulis dalam memahami materi yang ada. Selain itu, penulis juga belajar untuk bekerja sama dalam tim terutama dalam mengerjakan sebuah proyek serta dapat melatih *skill* manajemen waktu. Seluruh hasil yang telah direfleksikan ini, diharapkan dapat membantu penulis untuk berkembang ke arah yang lebih baik lagi.

DAFTAR PUSTAKA

- Knuth, D.E., Morris, J.H., dan Pratt, V.R. 1977. *Fast Pattern Matching in Strings*. SIAM Journal of Computing Vol 6 No.2.
- N.N. 2021. *Penerapan String Matching dan Regular Expression dalam Pembangunan Deadline Reminder Assistant* . Bandung: Institut Teknologi Bandung.
- Munir, R. 2021. *Pencocokan String (2021)*. Bandung: Institut Teknologi Bandung.
- Munir, R. 2021. *String Matching dengan Regex (2021)*. Bandung: Institut Teknologi Bandung.
- Ningtyas, S. 2020. *Mengenal Chatbot*. Niagahoster. Dilansir dari www.niagahoster.co.id pada 12 April 2021.