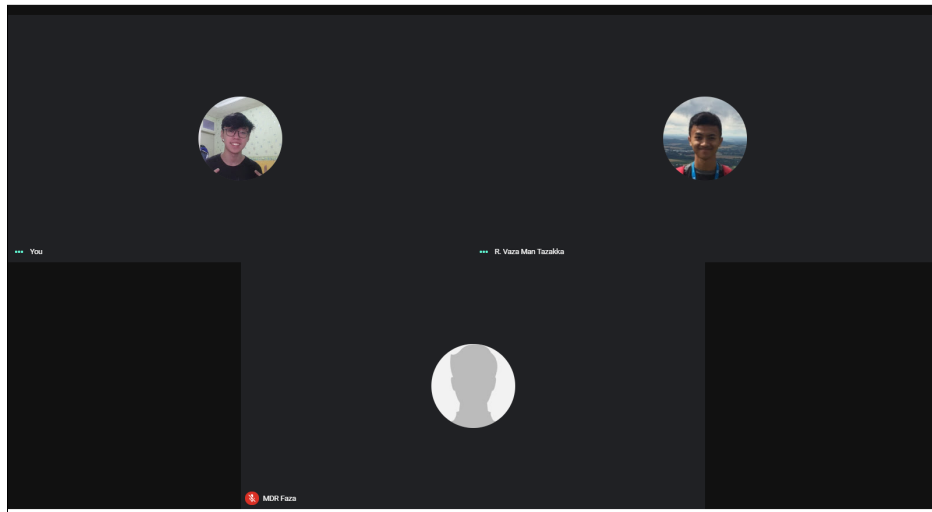


# **TUGAS BESAR III IF2211**

## **STRATEGI ALGORITMA**

### **2020/2021**



**Oleh**

**Muhammad Dzaki Razaan Faza 13519033**

**Rais Vaza Man Tazakka 13519060**

**James Chandra 13519078**

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**  
**INSTITUT TEKNOLOGI BANDUNG**  
**2020**

## BAB I

### Deskripsi Tugas

Bukan sesuatu yang janggal lagi jika semakin hari tugas-tugas di Teknik Informatika Semester 4 ITB semakin bertambah banyak. Hal ini tentunya berakibat pada bertambahnya kegiatan dan pekerjaan yang harus dilakukan mahasiswa. Tak jarang pula ada tugas yang terlupakan karena mahasiswa sulit untuk mengingat semua tugas dan deadline tersebut. Oleh karena itu, kami membuat suatu Google Assistant sederhana berupa *Deadline Reminder Assistant*, atau dalam bahasa Indonesia adalah Asisten Pengingat Deadline.

Di era digital ini, kita tentu sudah pernah mendengar teknologi atau aplikasi seperti *Chatbot*, *LINE Bot*, atau *Google Assistant*. Ketiganya merupakan agen cerdas yang meniru kemampuan manusia untuk melakukan percakapan dengan *user*. Kehadiran *Chatbot* ini tentu membantu kehidupan manusia, khususnya dalam membantu menyajikan informasi yang diperlukan *user* dan menjawab berbagai pertanyaan yang sering ditanyakan oleh *user*. Secara spesifik dalam konteks Asisten Pengingat Deadline ini, *Chatbot* tersebut akan menjawab pertanyaan-pertanyaan mahasiswa yang sering ditanyakan seperti *deadline* seminggu ke depan, *deadline* di bulan ini, dan *task-task* penting lainnya yang perlu dilakukan. *Chatbot* ini akan sangat membantu *user* agar tidak lagi melewatkan *deadline* tugas.

Dalam tugas besar ini, kami membuat 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*, chatbot dibuat interaktif dan sederhana layaknya *Google Assistant* yang akan menjawab segala pertanyaan *user* terkait informasi *deadline* tugas-tugas yang ada.

## BAB II

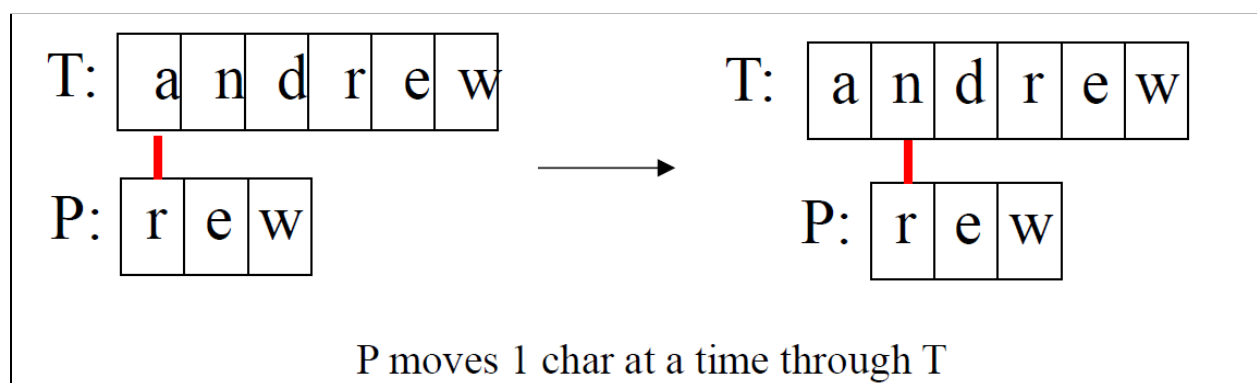
### Landasan Teori

#### 2.1 String/Pattern Matching

*String/pattern matching* adalah proses mencari lokasi pertama dalam teks yang bersesuaian dengan *pattern*. Teks merupakan *string* dengan panjang  $n$  karakter. *Pattern* adalah *string* dengan panjang  $m$  (dan  $m$  sangat kecil dibandingkan  $n$ ) yang akan dicari dalam teks. Penerapan *string/pattern matching* sering ditemukan dalam kehidupan sehari-hari, seperti pencarian dalam *text editor*, *web search engine*, analisis citra, bioinformatika, dan masih banyak lagi. Dalam melakukan *string/pattern matching* terdapat beberapa algoritma yang bisa digunakan. Diantaranya adalah algoritma *brute force*, algoritma Knuth-Morris-Pratt(KMP), dan algoritma Boyer-Moore(BM).

##### 2.1.1 Algoritma *Brute Force*

Algoritma *brute force* merupakan algoritma *string matching* yang paling sederhana dan menyelesaikan permasalahan secara gamblang. Proses pencarian *pattern* dalam teks dilakukan dengan cara mengecek setiap posisi dalam teks  $T$  apakah *pattern*  $P$  dimulai dari posisi tersebut.

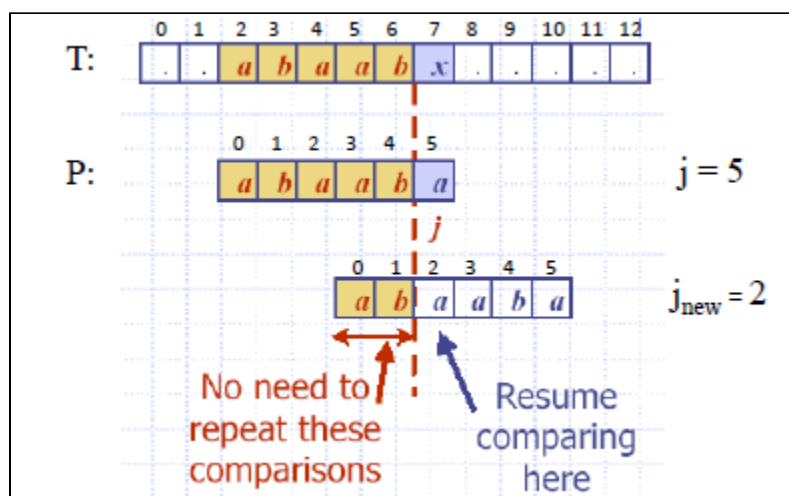


Gambar 2.1.1.1 Ilustrasi Algoritma *Brute Force*

Kompleksitas algoritma ini dapat dilihat dari jumlah perbandingan karakter yang dilakukan. Dalam berbagai kasus kompleksitasnya dapat mencapai  $O(mn)$ ,  $O(n)$ ,  $O(m+n)$  untuk kasus terburuk, terbaik, dan rata-rata. Algoritma *brute force* akan sangat cepat apabila alfabet dari teks besar (contoh: A..Z, a..z, 1..9) dan akan sangat lambat bila alfabetnya kecil (contoh: 0,1 dalam *binary file*).

### 2.1.2 Algoritma Knuth-Morris-Pratt

Algoritma KMP mencari *pattern* dalam teks dengan urutan kiri ke kanan seperti algoritma *brute force*, tetapi menggerakkan *pattern* dengan lebih “pintar”. Jika ada ketidakcocokan antara teks T dan *pattern* P di  $P[j]$  (karakter P ke- $j$ ), akan dilakukan pergeseran *pattern* sehingga sebisa mungkin menghindari perbandingan yang tidak berguna. KMP menjawab cara ini dengan mencari prefiks terbesar dari  $P[0..j-1]$  yang merupakan sufiks dari  $P[1..j-1]$  menggunakan fungsi pinggiran.



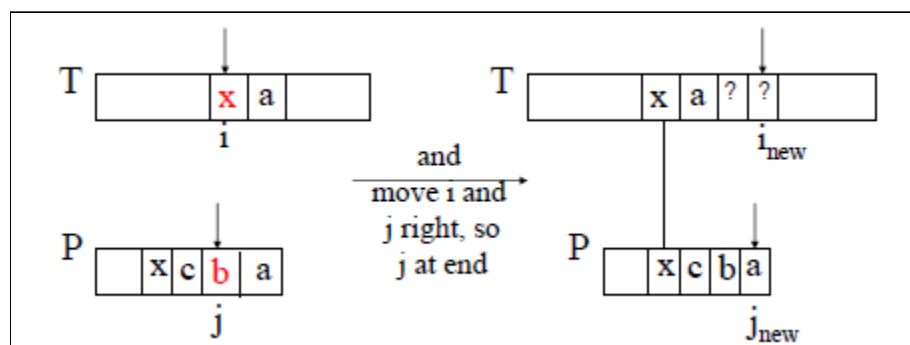
Gambar 2.1.2.1 Ilustrasi Algoritma KMP

Kompleksitas waktu dari algoritma KMP ditentukan dari perhitungan fungsi pinggiran yaitu  $O(m)$  dan pencarian string yaitu  $O(n)$ . Secara keseluruhan kompleksitas algoritma ini adalah  $O(m+n)$ . Hasil ini membuktikan bahwa algoritma KMP lebih cepat dari *brute force* dalam melakukan *string matching*. Algoritma KMP tidak akan pernah membutuhkan pergerakan mundur di teks masukan sehingga sangat cocok untuk memproses *file* sangat besar yang dibaca dari perangkat eksternal atau melalui arus

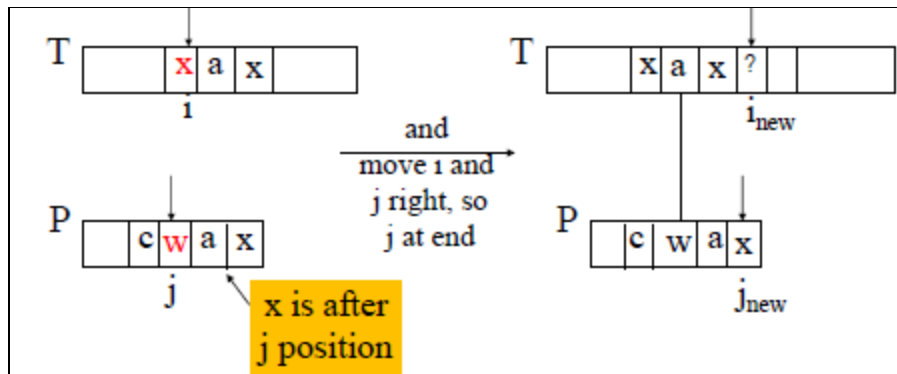
jaringan. Namun, algoritma KMP semakin tidak efektif dengan bertambahnya ukuran dari alfabet. Kemungkinan ketidakcocokan menjadi bertambah dan ketidakcocokan biasanya terjadi di awal *pattern* padahal algoritma KMP lebih cepat jika ketidakcocokan terjadi di belakang *pattern*.

### 2.1.3 Algoritma Boyer-Moore

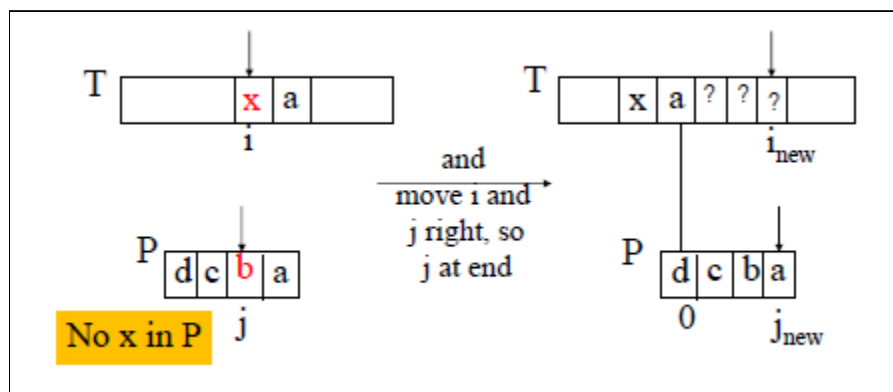
Algoritma BM menerapkan *string matching* dengan dua teknik dasar. Teknik pertama adalah dengan mencari *pattern* P dalam teks T dengan bergerak secara mundur dalam P, memulai perbandingan karakter dari akhir ke awal *pattern*. Teknik kedua adalah *character-jump* yaitu ketika terjadi ketidakcocokan antara  $P[j]$  dan  $T[i]$  pada  $T[i]=x$  maka akan terjadi tiga kasus yang mungkin. Kasus pertama adalah jika P mengandung x maka geser P ke kanan hingga kemunculan x terakhir di P sejajar dengan  $T[i]$  lalu mulai perbandingan kembali. Kasus kedua adalah jika dengan menggeser tidak bisa mensejajarkan x terakhir di P dengan  $T[i]$  maka geser P ke kanan sebanyak 1 karakter ke  $T[i+1]$ . Kasus ketiga adalah kemungkinan kejadian yang tidak ditangani kasus pertama dan kedua maka geser P sehingga  $P[0]$  sejajar dengan  $T[i+1]$ .



Gambar 2.1.3.1 Ilustrasi Algoritma MP Kasus 1



Gambar 2.1.3.2 Ilustrasi Algoritma MP Kasus 2



Gambar 2.1.3.3 Ilustrasi Algoritma MP Kasus 3

Algoritma BM mempunyai kompleksitas algoritma dalam kasus terburuk yaitu  $O(nm+A)$ . Namun, algoritma ini lebih cepat ketika ukuran alfabet( $A$ ) besar dan lambat jika ukurannya kecil. Algoritma BM akan sangat cepat dibandingkan *brute force* untuk pencarian *pattern* dalam teks bahasa Inggris.

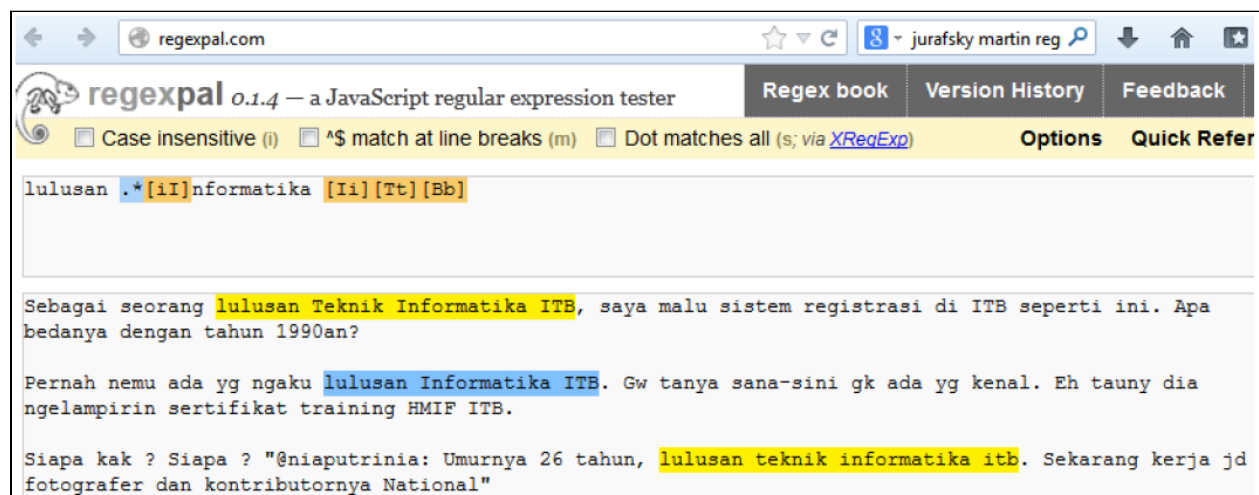
#### 2.1.4 Regular Expression (Regex)

Regex adalah serangkaian karakter yang mendefinisikan sebuah pola pencarian. Pola tersebut digunakan dalam *string matching* untuk melakukan operasi "cari" atau "cari dan ganti" pada *string*, atau untuk memeriksa *string* masukan. Ekspresi reguler merupakan teknik yang dikembangkan dalam bidang ilmu komputer teori dan teori bahasa formal.

.	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.
??,*?,+?	Same as above, but as few as possible.
{n}?, etc.	
(expr)	Capture expr for use with \1, etc.
(?:expr)	Non-capturing group.
(?=expr)	Followed by expr.
(?!expr)	Not followed by expr.

[Near-complete reference](#)

Gambar 2.1.1 Notasi Regex



Gambar 2.2.2 Ilustrasi String Matching menggunakan Regex

## 2.2 Penjelasan Singkat mengenai Chatbot

Chatbot adalah perangkat lunak yang menyediakan percakapan antara *user* dengan perangkat lunak sebagai pengganti percakapan dengan manusia asli. Kegunaan chatbot biasa digunakan untuk meringankan beban pengguna atau melakukan otomatisasi terhadap suatu proses yang tergolong repetitif atau *menial*.

## BAB III

### Analisis Pemecahan Masalah

#### 3.1 Langkah penyelesaian masalah setiap fitur

Setiap fitur pada program ini dijalankan dengan memeriksa kehadiran beberapa kata yang berkorespondensi dengan setiap fitur tersebut.

1. Fitur menambahkan *task* baru

Fitur ini dijalankan dengan memeriksa kehadiran kata tanggal, matkul, tugas, dan topik. Jika keempat kata tersebut ditemukan dalam *query* yang dikirim oleh *user*, program akan menambahkan *task* baru ke dalam database sesuai dengan kata-kata tersebut.

2. Fitur melihat daftar *task*

Fitur ini dieksekusi ketika terdapat kata “deadline” dan kata “semua” atau kata “deadline” dan kata “sejauh”. Fitur ini juga dijalankan ketika *user* mengirimkan kata “deadline” dan dua tanggal, rentang waktu, atau kata “hari ini”. Fitur ini juga dapat dijalankan dengan menambah spesifikasi jenis tugas dengan mengirimkan jenis tugas.

3. Fitur menampilkan *deadline* tugas

Fitur ini dijalankan ketika *user* mengirimkan kata “deadline” dan satu kode mata kuliah, kemudian mencari tugas yang dispesifikasikan (contoh “tubes” atau “tucil”, asumsikan bahwa hanya “tugas” tidak terhitung dan tidak berarti “tubes” dan “tucil”)

4. Fitur memperbaharui *task* tertentu



Fitur ini dieksekusi ketika *user* mengirimkan *task id*, tanggal terbaru untuk task yang ingin diperbaharui, dan kata “*update*”, “*undur*”, atau “*maju*”.

5. Fitur menandai selesai suatu *task*

Fitur ini akan jalan ketika *user* mengirimkan *task id* dan kata “selesai” atau “delete”.

6. Fitur opsi *help*

Fitur ini dijalankan ketika terdapat kata “lakukan” dan “apa”.

### 3.2 Fitur fungsional dan arsitektur Chatbot yang dibangun

Fitur fungsional:

1. Menambahkan task baru
2. Melihat daftar task
3. Menampilkan deadline tugas
4. Memperbaharui task tertentu
5. Menandai selesai suatu task
6. Opsi help

Arsitektur yang digunakan:

1. Frontend : React JS
2. Backend : Flask
3. Database : SQLite

## **BAB IV**

### **Implementasi dan Pengujian**

#### **4.1 Spesifikasi Teknis Program (struktur data, fungsi, prosedur yang dibangun)**

##### **Struktur data**

Berikut adalah bentuk struktur data/skema yang ada pada relasi yang digunakan untuk menyimpan daftar deadline/tugas sang pengguna, sebagai berikut.

```
CREATE TABLE tasks (  
    id integer PRIMARY KEY,  
    tanggal date NOT NULL,  
    matkul text NOT NULL,  
    tugas text NOT NULL,  
    topik text NOT NULL  
);
```

Karena ID sudah dibuat primary key dan merupakan integer, karena relasi diimplementasikan dengan sqlite3, maka sudah secara otomatis auto-inkremen dari angka 1, kemudian tanggal berupa format tanggal dan tidak boleh bernilai null, beserta dengan mata kuliah, jenis tugas dan topik tugas/deadline yang juga tidak boleh bernilai null.

##### **Fungsi dan Prosedur**

Berikut adalah fungsi dan prosedur yang diimplementasikan pada program ini.

`kmp_border_function(pattern, j)`: menerima suatu *string pattern* dan indeks *j* yang merupakan indeks tempat terjadinya *mismatch* antara *pattern* dengan *text*. Fungsi ini mengembalikan nilai ukuran *prefix* terbesar ketika *prefix* itu sama dengan *suffix*.

`make_kmp_table(kmp_table, pattern)`: menerima suatu array `kmp_table` dan *string pattern*. Prosedur ini membuat tabel kmp border function yang nanti digunakan untuk fungsionalitas *string-matching* dengan algoritma Knuth-Morris-Pratt.

`kmp(text, pattern, kmp_table)`: menerima *text*, *pattern*, dan tabel kmp border function. Fungsi ini melakukan *string-matching* terhadap *text* dengan pola *pattern*. Fungsi ini mengembalikan indeks tempat *matching* antara *text* dengan *pattern* dimulai. Jika tidak terdapat *match*, fungsi ini mengembalikan nilai -1.

`findWord(teks, pattern)`: mencari *pattern* dalam teks menggunakan algoritma Knuth-Morris-Pratt. Jika terdapat *pattern* dalam teks, fungsi ini mengembalikan `True`. Jika tidak, fungsi ini mengembalikan `False`.

`findNumberAfterWord(text, word)`: mencari satu atau lebih digit angka setelah teks *text* lalu hasilnya dimasukkan dalam *word*.

`findNumberBeforeWord(text, word)`: mencari satu atau lebih digit angka sebelum teks *text* lalu hasilnya dimasukkan dalam *word*.

`findMatkul(text)`: mencari kode mata kuliah pada *text* menggunakan regex. Fungsi ini mengembalikan kode mata kuliah yang ditemukan pada *text*.

`findTopik(text)`: mencari topik pada `text` yang terletak dalam tanda kutip (""). Fungsi ini mengembalikan topik yang ditemukan pada `text`.

`findTask(kalimat)`: mencari kata-kata tugas seperti "Tubes", "Tucil", dan sebagainya dalam kalimat dan mengembalikan kata-kata tersebut.

`lev(typo, benar)` : mencari *levenshtein distance* dari kedua string. *Levenshtein distance* merupakan jumlah minimum edit yang harus dilakukan agar kedua string membentuk string yang sama

`recommendWord(kalimat)` : mengganti kata-kata dalam kalimat menjadi kata-kata dalam kamus yang disediakan dengan menggunakan tingkat kemiripan dari *levenshtein distance*-nya.

`findTanggal(kalimat)` : mengembalikan *array* dari tanggal yang sesuai dengan format dalam kalimat

## 4.2 Tata Cara Penggunaan Program

Untuk menggunakan program ini, *user* perlu mengirimkan pesan dengan menulis pesan dalam kolom pesan dan menekan tombol enter pada keyboard. Terdapat beberapa fitur dalam program ini. Untuk menjalankan setiap fitur tersebut, *user* harus mengirimkan pesan yang mengandung kata-kata tertentu.

### 1. Fitur menambahkan *task* baru

Untuk menjalankan fitur ini, *user* perlu mengirim pesan yang mengandung kata tanggal, matkul, tugas, dan topik. Jika keempat kata tersebut ditemukan dalam pesan yang dikirim oleh *user*, program akan menambahkan *task* baru ke dalam

database sesuai dengan kata-kata tersebut.

## 2. Fitur melihat daftar *task*

Untuk menjalankan fitur ini, *user* perlu mengirim pesan yang mengandung kata “deadline” dan kata “semua” atau kata “deadline” dan kata “sejauh”. Fitur ini juga dapat dijalankan ketika *user* mengirimkan kata “deadline” dan dua tanggal, rentang waktu, atau kata “hari ini”. Fitur ini juga dapat dijalankan dengan menambah spesifikasi jenis tugas dengan mengirimkan jenis tugas.

## 3. Fitur menampilkan *deadline* tugas

Fitur ini dieksekusi ketika *user* mengirimkan kata “deadline” dan satu kode mata kuliah, kemudian mencari tugas yang dispesifikasikan (contohnya “tubes” atau “tucil”, asumsikan bahwa hanya “tugas” tidak terhitung dan tidak berarti “tubes” dan “tucil”)

## 4. Fitur memperbaharui *task* tertentu

Fitur ini dieksekusi ketika *user* mengirimkan *task id*, tanggal terbaru untuk task yang ingin diperbaharui, dan kata “*update*”, “undur”, atau “maju”.

## 5. Fitur menandai selesai suatu *task*

Fitur ini akan jalan ketika *user* mengirimkan *task id* yang disertai dengan kata “selesai” atau “delete”.

## 6. Fitur opsi *help*

Fitur ini dijalankan ketika terdapat kata “lakukan” dan “apa”.

## 4.3 Hasil Pengujian

Halo, bot tolong ingetin kalau ada kuis if3110 "Bab 2 sampai 3" pada 22/04/2021

[TASK BERHASIL DICATAT]  
(ID: 1) 22/04/2021 - IF3110 - Kuis - Bab 2 sampai 3

### Gambar 4.3.1 Fitur 1 Penambahan Tugas

bot, tambahin kuis if1130 ke jadwal

Maaf, pesan tidak dikenali  
Mungkin maksud Anda:  
bisa tambahin tugas if1130 ke jadwal

### Gambar 4.3.2 Fitur 1 Penambahan Tugas Tanpa Info Lengkap

bot, tambahin tubes if2123 "String Matching" tanggal 22-10-2021

[TASK BERHASIL DICATAT]  
(ID: 2) 22/10/2021 - IF2123 - Tubes - String Matching

### Gambar 4.3.3 Fitur 1 Penambahan Tugas Dengan Format Tanggal Berbeda

Apa saja deadline yang dimiliki sejauh ini?

[Daftar Deadline]  
1. (ID: 1) 22/04/2021 - IF3110 - Kuis - Bab 2 sampai 3  
2. (ID: 2) 22/10/2021 - IF2123 - Tubes - String Matching

### Gambar 4.3.4 Fitur 2 Melihat Daftar Task yang Harus Dikerjakan

deadline kuis hari ini

[Daftar Deadline]  
1. (ID: 3) 28/04/2021 - GH2345 - Kuis - Regex

### Gambar 4.3.5 Fitur 2 Melihat Daftar Task yang Harus Dikerjakan dengan Filter Waktu dan Kata Penting



**Gambar 4.3.6 Fitur 2 Melihat Daftar Task yang Harus Dikerjakan dengan Filter Waktu dan Kata Penting Kasus Tidak Ada Deadline yang Memenuhi**



**Gambar 4.3.7 Fitur 3 Menampilkan Deadline dari Task Tertentu**



**Gambar 4.3.8 Fitur 3 Menampilkan Deadline dari Task Tertentu yang Tidak Ada di Jadwal**



**Gambar 4.3.9 Fitur 4 Memperbarui Task Tertentu**



**Gambar 4.3.10 Fitur 4 Memperbarui Task Tertentu yang Tidak Ada di Jadwal**



**Gambar 4.3.11 Fitur 5 Menandai Suatu Task Sudah Dikerjakan**



**Gambar 4.3.12 Fitur 5 Menandai Suatu Task Sudah Dikerjakan yang Sudah Tidak Ada di Jadwal**



**Gambar 4.3.12 Fitur 6 Menampilkan Opsi Help yang Difasilitasi Oleh Asisstant**



**Gambar 4.3.12 Fitur 8 dan 9 Menampilkan Pesan Error Sekaligus Menyediakan Kata Rekomendasi**

## 4.4 Analisis Hasil Pengujian



Berdasarkan hasil percobaan yang telah dilakukan dapat dianalisis bahwa yeetBOT masih memiliki batasan. Contohnya, pada implementasi pencarian kata kunci agar mengenali topik pada kalimat masukan masih harus diberikan batasan yaitu berupa kalimat yang dilingkupi tanda kutip. Pada implementasi tanggal format yang dikenali hanya terdapat tiga jenis, yaitu dd-mm-yy, dd/mm/yyyy, dd MMMM yyyy. Namun, dengan adanya batasan tersebut program kami dapat menerima hampir seluruh input untuk kemudian diberikan respon yang sesuai. Seluruh fitur yang dibuat dapat dijalankan dengan baik dan memberikan hasil yang diharapkan.

## BAB V

### Keimpulan

#### 5.1 Kesimpulan

Tim penyusun laporan dapat menyimpulkan bahwa mempelajari pemrograman web sangatlah intuitif, dan banyak sekali seluk beluknya yang menarik dan mudah untuk dieksplor karena banyaknya sumber belajar yang ada di jejaring internet.

Tim penyusun juga dapat menyimpulkan bahwa topik *string matching* merupakan topik yang cukup menarik untuk dipelajari karena memiliki banyak repercusi dan aplikasi ke depannya, terutama di dunia berkarir nanti. Penulis juga dapat menyimpulkan bahwa tugas seperti ini dapat membangkitkan pula keingintahuan yang cenderung bisa dibilang tidak biasa, seperti saat ketiga penulis mencari-cari tentang perjalanan serta cerita kehidupan dari pasangan trio Knuth-Morris-Pratt.

#### 5.2 Saran

Kami menyadari bahwa dalam pengerjaan program masih banyak yang dapat dikembangkan, sebagai berikut:

1. Menyempurnakan *website* baik dari segi tampilan dan performanya.
2. Membuat kode lebih rapi dan memberikan komentar yang lebih detail serta mudah dipahami.
3. Membuat ekspresi reguler yang lebih baik yaitu mencakup alternatif input lebih banyak lagi

#### 5.3 Refleksi

Pada tugas besar 3 strategi algoritma ini kami belajar untuk membuat *website* baik dari segi *frontend* maupun *backend*. Komentar kami terhadap tugas besar ini adalah,

walaupun cukup menantang karena kami dipaksa untuk mempelajari hal baru, tetapi ilmu yang didapat sangat berguna untuk kedepannya.

## Daftar Pustaka

Facebook Open Source. (n.d.). *React Documentation*.

<https://reactjs.org/docs/getting-started.html>

Flask Documentation. (n.d.). *Flask Table*. <https://flask-table.readthedocs.io/en/stable/>

IF2211 Strategi Algoritma. (2021). *Pencocokan String (String/Pattern Matching)*.

Program Studi Teknik Informatika STEI ITB.

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

Khodra, M. L. (n.d.). *String Matching dengan Regular Expression*. Program Studi Teknik Informatika STEI ITB.

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