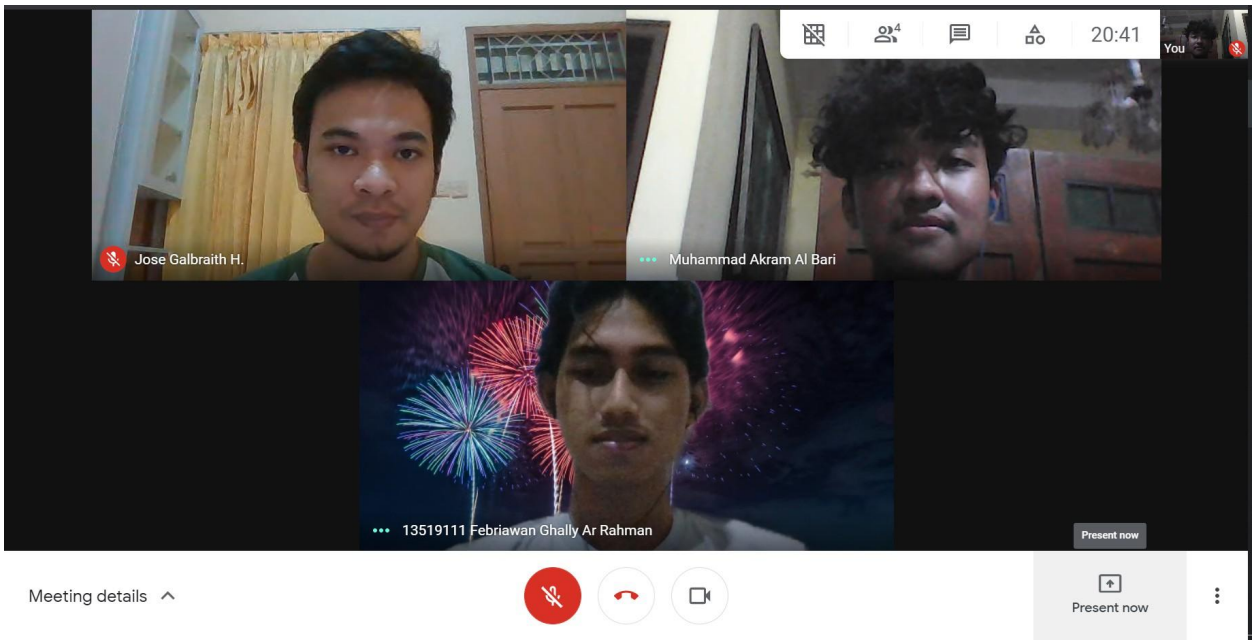
LAPORAN TUGAS BESAR 3 IF2211

Penerapan String Matching dan Regular Expression dalam Pembangunan Deadline Reminder Assistant



DuaBelibis

Disusun Oleh:

Jose Galbraith Hasintongan	13519022
Febriawan Ghally Ar Rahman	13519111
Muhammad Akram Al Bari	13519142

BAB I

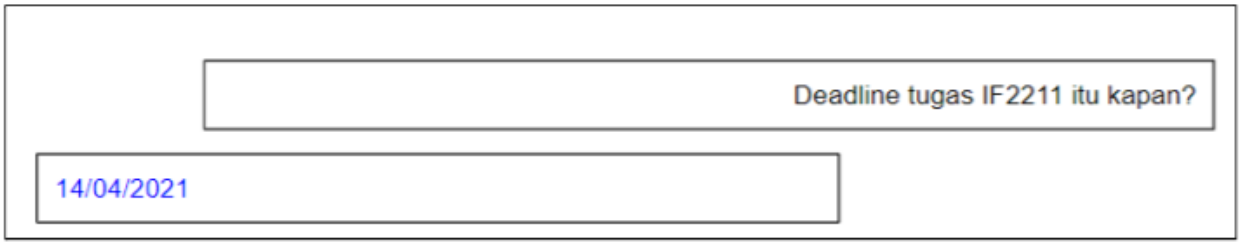
Deskripsi Tugas

Dalam tugas besar ini, Anda 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, Anda dapat membangun sebuah chatbot interaktif sederhana layaknya Google Assistant yang akan menjawab segala pertanyaan Anda terkait informasi deadline tugas-tugas yang ada.

Fitur-Fitur Aplikasi: 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.
    d. Contoh interaksi

2. Melihat daftar task yang harus dikerjakan
    a. Seluruh task yang sudah tercatat oleh assistant
        ● Contoh perintah yang dapat digunakan: “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
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?”
    c. Contoh interaksi



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)

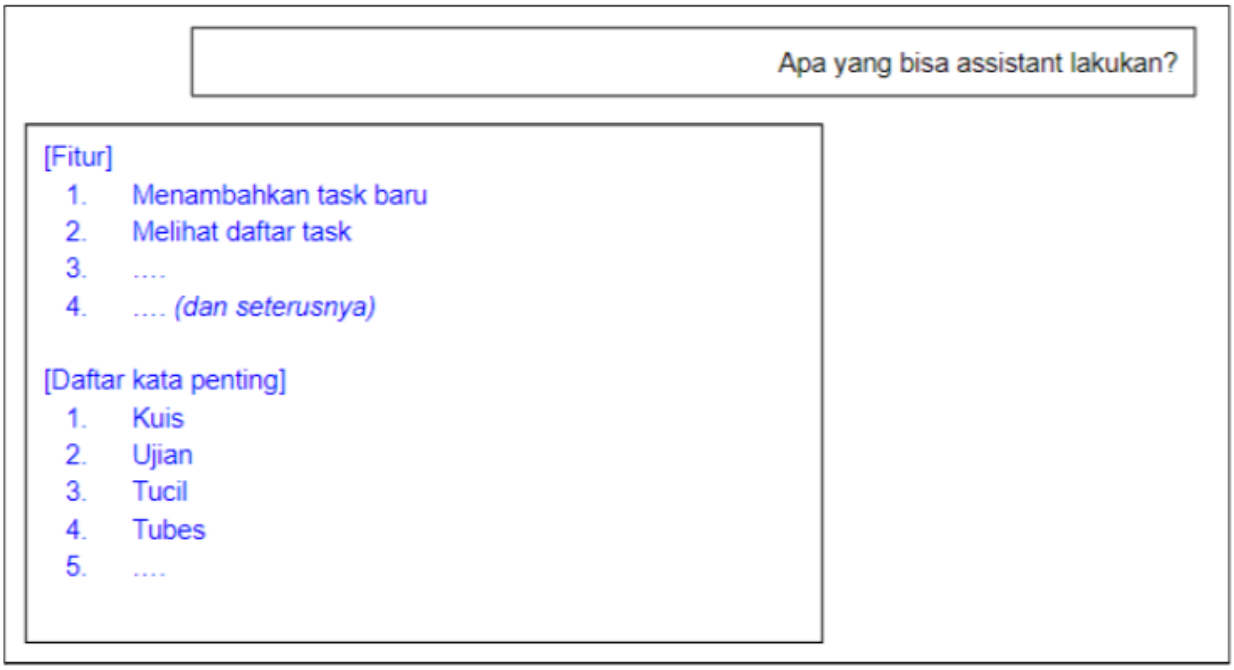
6. Menampilkan opsi help yang difasilitasi oleh assistant

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)

d. Contoh interaksi



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.

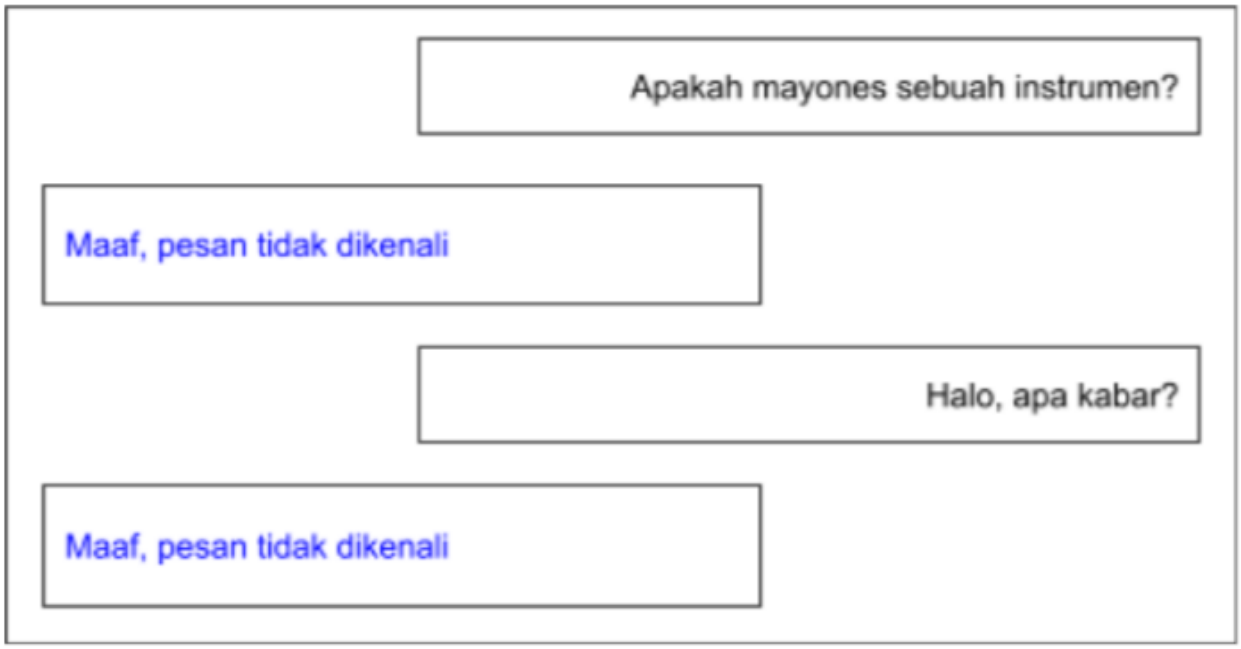
c. Daftar kata penting tidak perlu dibuat dinamis, cukup static saja atau hardcoded.

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

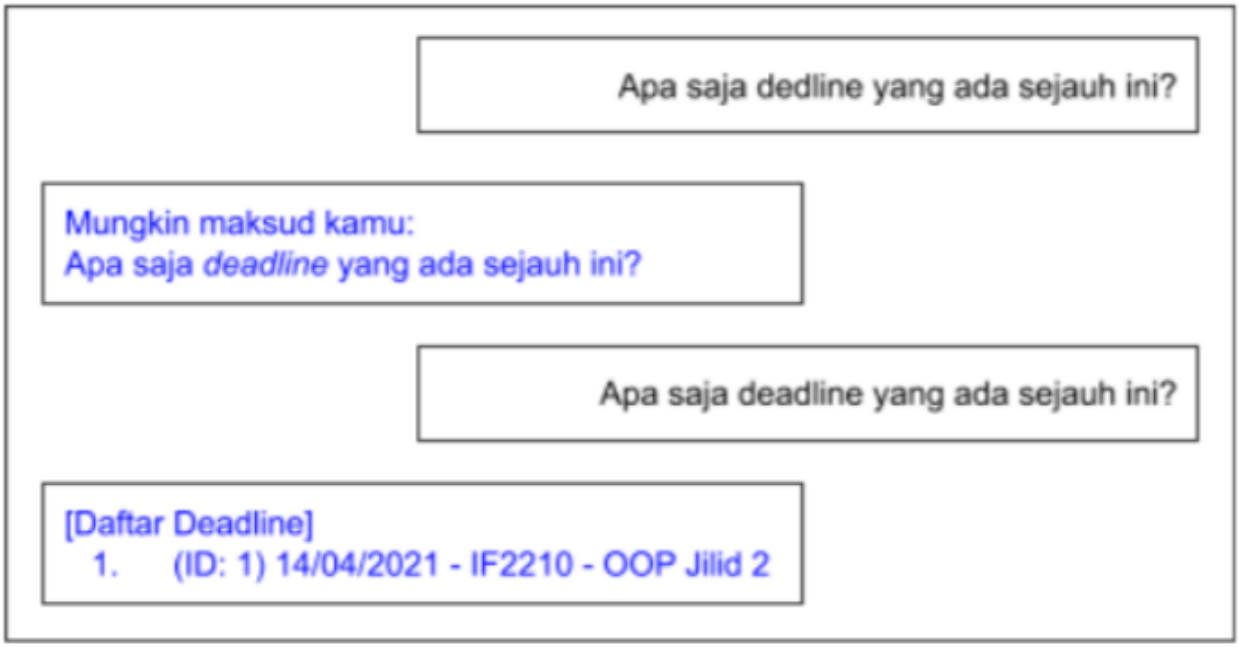
c. Contoh interaksi



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. Contoh interaksi



c. Ada berbagai metriks yang dapat dimanfaatkan untuk mencari kemiripan kata, salah satunya adalah Levenshtein distance yang diukur melalui pendekatan dynamic programming

BAB II

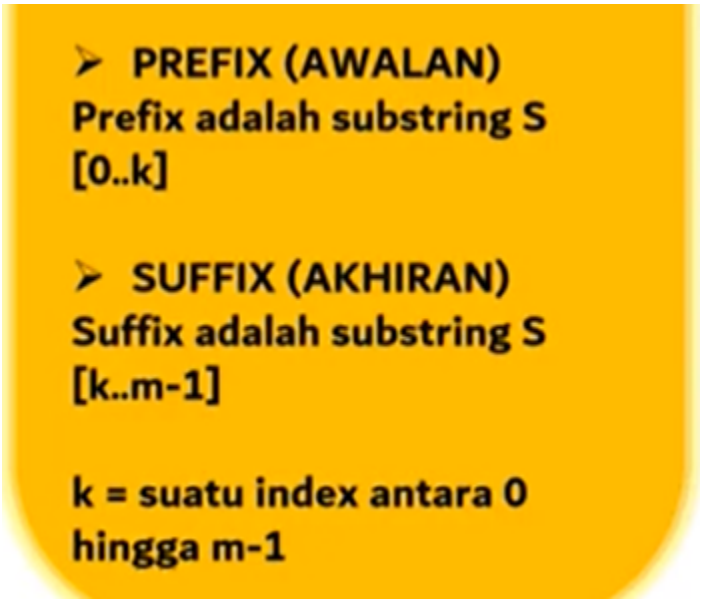
Landasan Teori

Dasar Teori

Algoritma Pencocokan String

String Matching atau pencocokan string adalah algoritma untuk mencari suatu kemunculan sebuah string yang “lebih pendek”, biasa disebut sebagai *pattern*, terhadap string lain yang “lebih panjang” biasa disebut sebagai teks. Terdapat berbagai macam pendekatan dalam algoritma pencocokan string. Diantaranya ialah *brute force*, algoritma *Knuth-Morris-Pratt* (KMP), dan juga algoritma *Boyer Moore* (BM).

Dalam algoritma pencocokan string KMP, kita mengenal istilah prefiks dan sufiks. Di mana, dalam konteks ini, prefiks didefinisikan sebagai suatu substring yang dimulai dari indeks ke 0 sampai dengan suatu bilangan k, di mana k adalah indeks terjadinya ketidakcocokan dikurangi satu, dari suatu pattern ketika dilakukan pencocokan dengan teks(suffiks = teks[0...k]). Sedangkan sebaliknya, sufiks adalah suatu substring yang dimulai dari indeks ke k sampai dengan indeks ke 1 suatu pattern. Secara formal dapat kita definisikan sebagai berikut:



sumber: https://www.youtube.com/watch?v=HWlSF8f-uWU&t=781s

Secara garis besar, algoritma KMP akan melakukan pencocokan dengan cara mencocokan pattern dengan teks mulai dari kiri, dari indeks terkecil. Apabila di tengah-tengah pencocokan terjadi suatu *mismatch* karakter yang ada pada keduanya, maka algoritma KMP akan melakukan kalkulasi untuk menemukan suatu substring

sufiks dan prefiks yang mungkin dibentuk, dan masih sama (tidak terjadi *mismatch*). kemudian akan dilakukan pergeseran secara cerdas dengan memanfaatkan kalkulasi terhadap sufiks dan prefiks tersebut.

Algoritma BM melakukan pencocokan string dengan cara yang terbalik dibandingkan dengan KMP. Apabila KMP melakukan pencocokan string dan pattern dari kiri, algoritma BM akan melakukan pencocokan string dan pattern dimulai dari karakter terakhir dari pattern. Dalam algoritma BM, kita mengenal istilah *character jump*, di mana algoritma BM ini akan melakukan pertimbangan-pertimbangan tertentu untuk mengoptimalkan pergeseran karakter seminimal mungkin. Apabila terjadi suatu *mismatch*, maka algoritma BM akan melakukan pergeseran dengan mempertimbangkan tiga macam kasus yang mungkin terjadi, yakni:

1. Jika *mismatch* pada suatu karakter x, dan x ada pada pattern namun belum dilakukan pengecekan
2. Jika *mismatch* terjadi pada suatu karakter x, namun x sudah muncul pada pattern dan sudah dicek sebelum *mismatch* terjadi
3. Jika *mismatch* terjadi pada suatu karakter x, namun x tidak ada pada pattern.

Secara garis besar, algoritma BM akan melakukan pengecekan karakter dengan teknik yang disebut sebagai *looking-glass*, di mana pengecekan karakter dilakukan dari bagian belakang pattern. Apabila terjadi *mismatch*, maka algoritma BM akan melakukan *character jump* sesuai dengan kasus yang ditemukan.

Selain kedua algoritma pencocokan string di atas, pencocokan string juga dapat dilakukan dengan menggunakan Regular Expression atau Regex. Regex merupakan sebuah string yang mendefinisikan suatu pola pencarian sehingga kita dapat menemukan substring-substring yang sesuai dengan pola yang telah kita definisikan pada suatu teks. Secara umum, regex sangat memudahkan kita dalam melakukan pencarian atau pencocokan suatu pattern tertentu pada sebuah teks.

Chatbot

Secara definisi, Chatbot merupakan program yang dirancang untuk dapat mensimulasikan proses percakapan secara pintar antara pengguna dengan program. Pada tugas besar kali ini, chatbot dirancang agar dapat memahami masukan-masukan dari pengguna yang berhubungan dengan penyimpanan suatu tugas, melihat deadline tugas tertentu, serta melakukan update terhadap tugas yang telah didefinisikan.

Pada chatbot yang kami rancang, kami memanfaatkan database SQL untuk menyimpan data-data user yang ada. Kemudian, kami menuliskan kode program Chatbot dalam bahasa Python, serta Chatbot ini merupakan aplikasi yang berbasis

web. Untuk menghubungkan kode dengan web, kami menggunakan *framework* bernama *Flask*.

BAB III

Analisis Pemecahan Masalah

Langkah penyelesaian masalah setiap fitur

Tugas Besar kali ini merupakan tugas yang mengharuskan kami untuk mengimplementasikan algoritma pattern matching (string matching) untuk melakukan pengenalan pola-pola tertentu pada masukan yang diberikan oleh user. Pola-pola tersebut utamanya adalah berupa kalimat-kalimat “penting” yang mendefinisikan suatu perintah yang valid yang dapat dimasukan user ke dalam query chatbot.

Dengan menggunakan algoritma pencocokan string, kami berusaha untuk menemukan ada kata penting apa saja yang terkandung di dalam query yang diberikan oleh user. Secara garis besar, kami mengimplementasikan algoritma pencocokan string Boyer-Moore (BM) dan algoritma Knuth-Morris-Pratt (KMP). Dengan beberapa pertimbangan, utamanya mempertimbangkan dari efisiensi eksekusi algoritma untuk menangani kasus yang ada, kami memilih algoritma BM sebagai algoritma utama untuk melakukan pencocokan string.

Selain menggunakan algoritma BM, kami juga menggunakan Regular Expression (regex) untuk membantu menemukan pola pola yang lebih general dalam masukan yang diberikan oleh user. Regex utamanya kami gunakan dalam menentukan input yang berupa pola-pola tertentu, namun masukannya bisa sangat berbeda dari satu masukan ke masukan lain.

Untuk fitur pertama, yakni fitur menambahkan task ke dalam program, kami memanfaatkan regex untuk mengidentifikasikan, ada dan lengkap tidaknya komponen komponen yang mendefinisikan suatu task pada query yang diberikan oleh user. Komponen task yang kami maksud diantaranya ialah komponen nama matkul, topik tugas, jenis tugas, dan deadline suatu tugas. Apabila keempat komponen ini terdefinisi, dan program tidak menemukan adanya kalimat perintah lain di dalam query, maka program akan mendefinisikan query tersebut sebagai perintah untuk menambahkan suatu task baru ke dalam program

Fitur berikutnya adalah melihat daftar task. Pada fitur kali ini, kami diminta untuk mengimplementasikan kemampuan melihat task-task tertentu, sesuai dengan kombinasi query yang diberikan oleh user. Pada dasarnya, perintah menampilkan task pada program ini akan terdefinisi apabila program membaca kata “deadline” pada query yang dimasukan oleh user. Kata “deadline” ini bisa saja diikuti oleh kombinasi-kombinasi lain, semisal jumlah hari, jumlah minggu, ataupun kata hari ini. Sesuai dengan kombinasi tambahan yang terdeteksi pada query user, program akan melakukan query-query yang bersesuain agar hasil output dari program sesuai dengan query yang diberikan. Apabila program menemukan kombinasi yang tak lazim, misal terdapat kata minggu dan hari sekaligus, maka program akan mengartikan perintah tersebut sesuai dengan kata yang pertama kali muncul dalam query.

Selain itu, program juga dapat menampilkan deadline suatu task tertentu dengan memasukan nama matakuliah suatu task. Proses pengenalan query pada bagian ini, lagi-lagi kami memanfaatkan regex, dengan terlebih dahulu melakukan pengecekan, apakah ada kata “matkul/mata kuliah” yang diberikan pada query yang dimasukan oleh user. Kemudian, program akan mengambil seluruh kata yang mengikuti kata “matkul/mata kuliah”, apabila kata-kata tersebut diawali oleh huruf kapital. Kemudian, program juga akan melakukan pengenalan kata “deadline” dengan menggunakan algoritma BM.

Program kami kali ini juga memanfaatkan library SQL agar kami bisa menggunakan dan mengintegrasikan database SQL pada program. Program juga harus bisa melakukan fitur update data suatu task. Mula-mula, setiap task yang telah terdefinisi akan disimpan pada suatu file SQL (.db), kemudian program akan memuat file .db ini diawal keberjalanannya. Selanjutnya, apabila user memasukan perintah “diundur”, dan diikuti dengan kata “task” dan ID suatu task lalu harus ada juga satu buah tanggal, maka program akan mendefinisikan query tersebut sebagai perintah untuk melakukan update deadline suatu task. Setiap update dan penyimpanan data dibuat terpusat pada SQL.

Selain itu, terdapat pula fitur untuk menandai bahwa suatu task telah selesai dikerjakan. Task yang sudah dikerjakan akan tidak ditampilkan apabila pengguna memasukan perintah deadline pada program. Sama dengan update deadline, perintah ini juga memanfaatkan SQL untuk melakukan update dan menyimpan data.

Terakhir, kami juga memanfaatkan metriks *Levensteihn Distance* untuk mengukur suatu kemiripan kata dengan kata lainnya. metriks ini kami gunakan untuk mengimplementasikan fitur rekomendasi perintah, apabila nantinya program menemukan terjadinya typo, namun masih mirip dengan kata-kata perintah yang terdefinisi pada program.

Fitur fungsional dan arsitektur Chatbot yang dibangun

Fitur fungsional yang terdapat pada program ini meliputi fitur input berupa textbox yang dapat diisi oleh pengguna untuk memasukan query program. Tampilan utama program ini merupakan tampilan yang berbasiskan web, dan pengguna dapat memasukan query/pesan yang diinginkan ke dalam tampilan tersebut. Nantinya, bot akan memberikan respons sesuai dengan query yang dikenali oleh bot. Tampilan yang kami buat, menyerupai tampilan chat pada kebanyakan media sosial yang ada saat ini.

Kemudian, pada program juga terdapat fitur-fitur untuk menambahkan suatu task, untuk melihat deadline dari seluruh task yang sudah terdaftar, melihat task berdasarkan kriteria tertentu: rentang tanggal, rentang hari, rentang minggu, dan hari ini. Selain itu, program juga dapat menampilkan task sesuai dengan jenis task yang dipilih oleh pengguna. Program juga dapat menandai suatu task, apabila sudah selesai dikerjakan, dan juga dapat memperbaharui data deadline suatu task apabila terjadi perubahan deadline.

Program juga dapat memberikan pesan kesalahan, apabila query yang dimasukan pengguna sama sekali tidak dikenali oleh program. Selain itu, program juga dapat memberikan rekomendasi kata kunci yang tepat, apabila pengguna mengalami typo yang tidak terlalu jauh. Rekomendasi kata kunci ini dihitung dengan menggunakan matriks *Levenshtein Distance*.

Backend:
Pada bagian backend, kami menggunakan bahasa pemrograman Python sebagai bahasa yang digunakan dalam mengimplementasikan keseluruhan program. Dalam realisasinya, kami menggunakan beberapa library semisal library re(regular expression) untuk melakukan pemrosesan terhadap operasi operasi regex. Selain itu kami juga menggunakan library datetime untuk mendapatkan tanggal saat ini, karena hal tersebut penting dan akan digunakan juga pada beberapa perintah di dalam program. Kami memisahkan fungsi dan prosedur pada beberapa buat file .py untuk memudahkan dalam hal pengelompokkan.

Frontend:
Pada bagian Frontend. Kami menggunakan framework Flask dan bahasa pemrograman Python. Pada file app.py terdapat fungsi get_bot_response yang dapat mengambil

respons dari user yang berupa *string*. Pada file index.html, kami menyusun body untuk tampilan chatbox. Pada bagian style.css, kami menentukan font, color, padding, ukuran untuk tampilan keseluruhan di *website*.

Database:
Pada bagian Database, kami menggunakan fungsi bawaan dari Flask. Kemudian kami mendefinisikan relasi beserta atribut yang dibutuhkan, serta beberapa fungsi yang digunakan untuk menambahkan dan juga mengupdate data..

BAB IV

Implementasi dan Pengujian

Spesifikasi teknis program

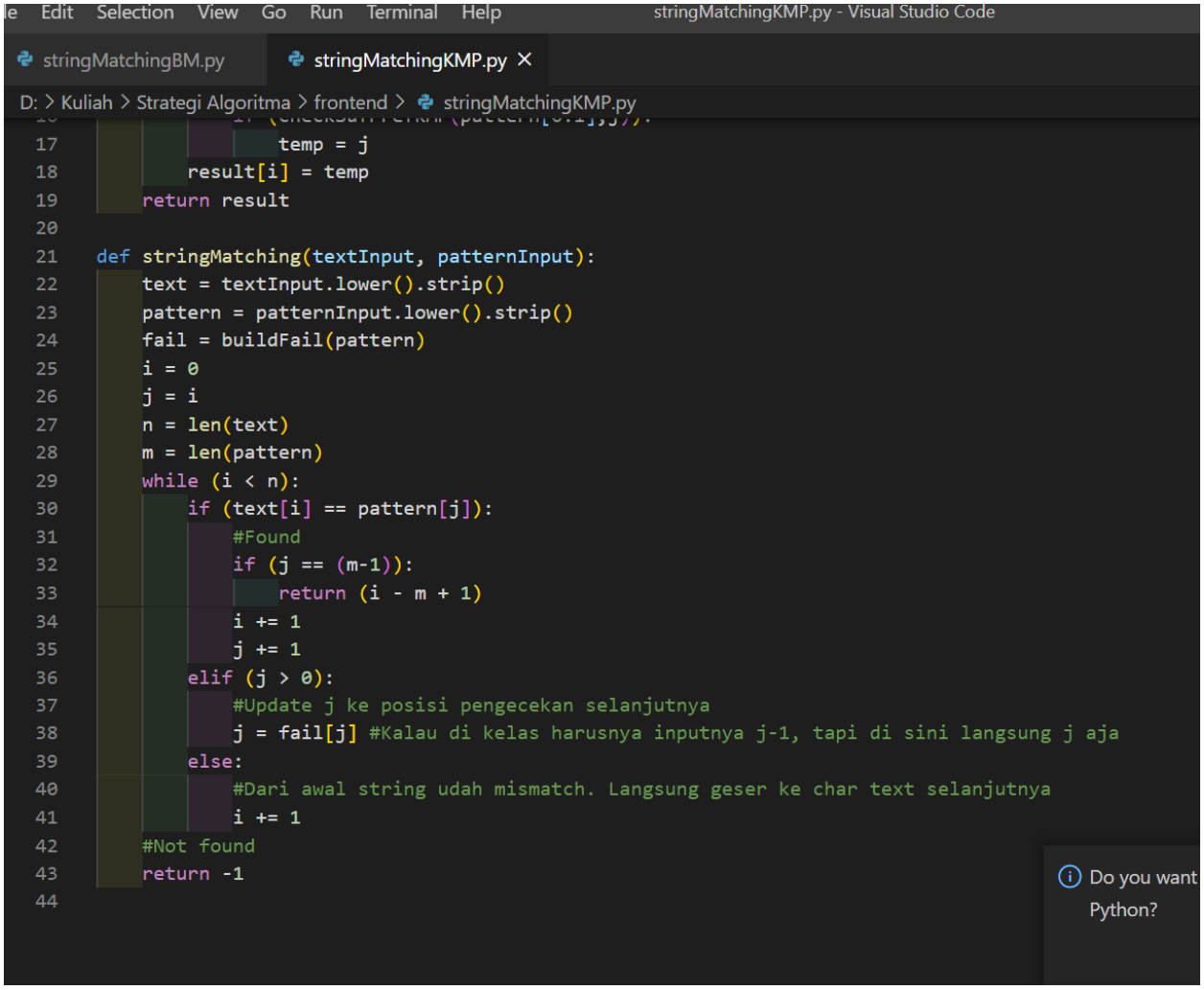
Kami menggunakan database relational, atau SQL, dengan satu buah table yang memiliki attribute sebagai berikut: id, matkul, jenis, deadline, dan status. Selain itu, dalam program ini terdefinisi macam-macam fungsi/prosedur yang digunakan untuk memanipulasi hasil dari suatu task agar bisa menyesuaikan dengan tampilan yang diminta oleh query

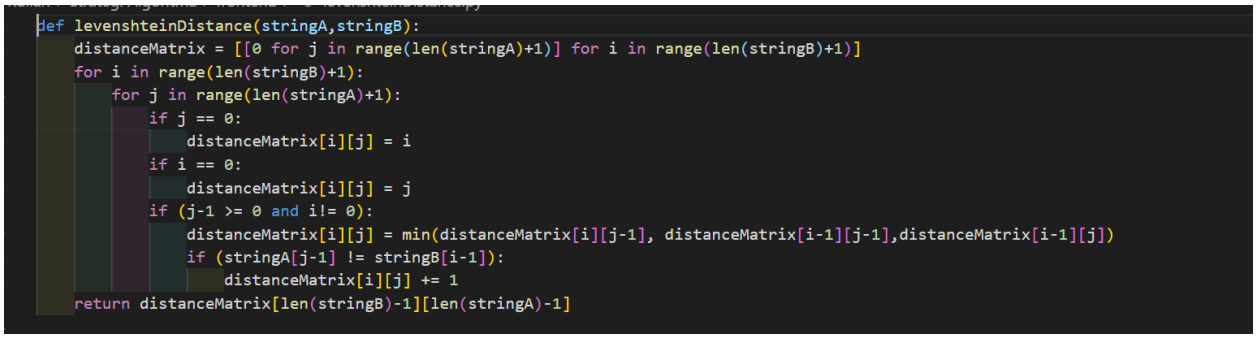
Sebagian besar fungsi/prosedur digunakan untuk memanipulasi string query dari pengguna agar dapat dikenali macam-macam pola tertentu yang ada didalamnya sehingga program bisa menentukan aksi yang tepat untuk menangani suatu query tertentu.

stringMatchingBM.py

D: > Kuliah > Strategi Algoritma > frontend > stringMatchingBM.py

```
        # ord = fungsi buat dapetin ASCII dari suatu char
        last[ord(char)] = index
    return last

def stringMatching(textInput, patternInput):
    text = textInput.lower().strip()
    pattern = patternInput.lower().strip()

    j = len(pattern)-1
    i = j
    n = len(text)
    m = len(pattern)
    lastO = buildLast(pattern)
    while (i < n):
        if (text[i] == pattern[j]):
            if (j == 0):
                #if found matching pattern
                return i
            #looking-glass
            i -= 1
            j -= 1
        else:
            #Geser i sesuai kasus (1, 2, atau 3)
            i = i + m - min(j, lastO[ord(text[i])]+1)

            #Balikin lagi j ke indeks akhir pattern
            j = m - 1
    #not found
    return -1
```


le Edit Selection View Go Run Terminal Help
stringMatchingKMP.py - Visual Studio Code
stringMatchingBM.py
stringMatchingKMP.py
D: > Kuliah > Strategi Algoritma > frontend > stringMatchingKMP.py
temp = j
result[i] = temp
return result
def stringMatching(textInput, patternInput):
text = textInput.lower().strip()
pattern = patternInput.lower().strip()
fail = buildFail(pattern)
i = 0
j = i
n = len(text)
m = len(pattern)
while (i < n):
if (text[i] == pattern[j]):
#Found
if (j == (m-1)):
return (i - m + 1)
i += 1
j += 1
elif (j > 0):
#Update j ke posisi pengecekan selanjutnya
j = fail[j] #Kalau di kelas harusnya inputnya j-1, tapi di sini langsung j aja
else:
#Dari awal string udah mismatch. Langsung geser ke char text selanjutnya
i += 1
#Not found
return -1
Do you want
Python?


def levenshteinDistance(stringA,stringB):
distanceMatrix = [[0 for j in range(len(stringA)+1)] for i in range(len(stringB)+1)]
for i in range(len(stringB)+1):
for j in range(len(stringA)+1):
if j == 0:
distanceMatrix[i][j] = i
if i == 0:
distanceMatrix[i][j] = j
if (j-1 >= 0 and i!= 0):
distanceMatrix[i][j] = min(distanceMatrix[i][j-1], distanceMatrix[i-1][j-1],distanceMatrix[i-1][j])
if (stringA[j-1] != stringB[i-1]):
distanceMatrix[i][j] += 1
return distanceMatrix[len(stringB)-1][len(stringA)-1]

```python
def missWordRecc(text, kataPenting, missThreshold, result):
    # result = [[0],[]]

    for count, mismatch in enumerate((text.split())):
        foundMismatch = False
        for keyword in kataPenting:
            editDistance = levenshteinDistance(mismatch.lower(),keyword.lower())
            mismatchPercentage = float(editDistance/max(len(keyword),len(mismatch)))
            if (mismatchPercentage < missThreshold and mismatchPercentage > 0):
                print("mismatch percentage",mismatchPercentage)
                print(keyword)
                print("editdistance: ", editDistance)
                result[1].append(keyword)
                foundMismatch = True
                result[0] = 1
                break
        if (not foundMismatch and mismatch not in result[1]):
            result[1].append(mismatch)
    if (result):
        return result
    return -1
```

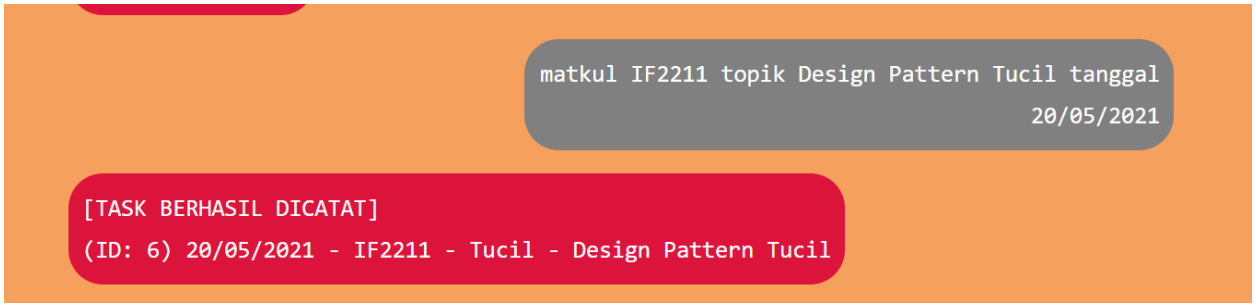
```python
def extractTask(query, deadlineTask,normalTask, stringMatching):
    task = {"id": [],
            "matkul":[],
            "jenis":[],
            "topik":[],
            "deadline":[],
            "status":[]}
    # kodeMatkulPattern = re.compile(r'[a-zA-Z]{2}\d{4}')

    #Referensi regex buat ambil consecutive capitalized word: https://stackoverflow.com/questions/31570699/reg
    #matkulPattern = re.compile(r'((?:[Mm]atkul|[Mm]ata [Kk]uliah)|[a-zA-Z]{2}\d{4})\s?([A-Za-z\s]*)?\s?([Tt]o
    idPattern = re.compile(r'[Tt]ask\s\d+')
    matkulPattern = re.compile(r'(\b(?:[A-Z][a-z]*\b\s*\d?)+|[A-Z]{2}\d{4})')
    datePattern = re.compile(r'(\d{2}.\d{2}.\d{4}|\d{2}.(?:[Jj]anuari|[Ff]ebruari|[Mm]aret|[Aa]pril|[Mm]ei|[Jj
    # topikPattern = re.compile(r'\b(?:[A-Z][a-z]*\b\s*\d)+')
    topikPattern = re.compile(r'(\b(?:[A-Z][a-z]*\b\s*\d?)+|[A-Z]{2}\d{4})')
    # topikPattern = re.compile(r'^topik\s')


    #Add task from query

    #Add ID
```

Penjelasan tata cara penggunaan program

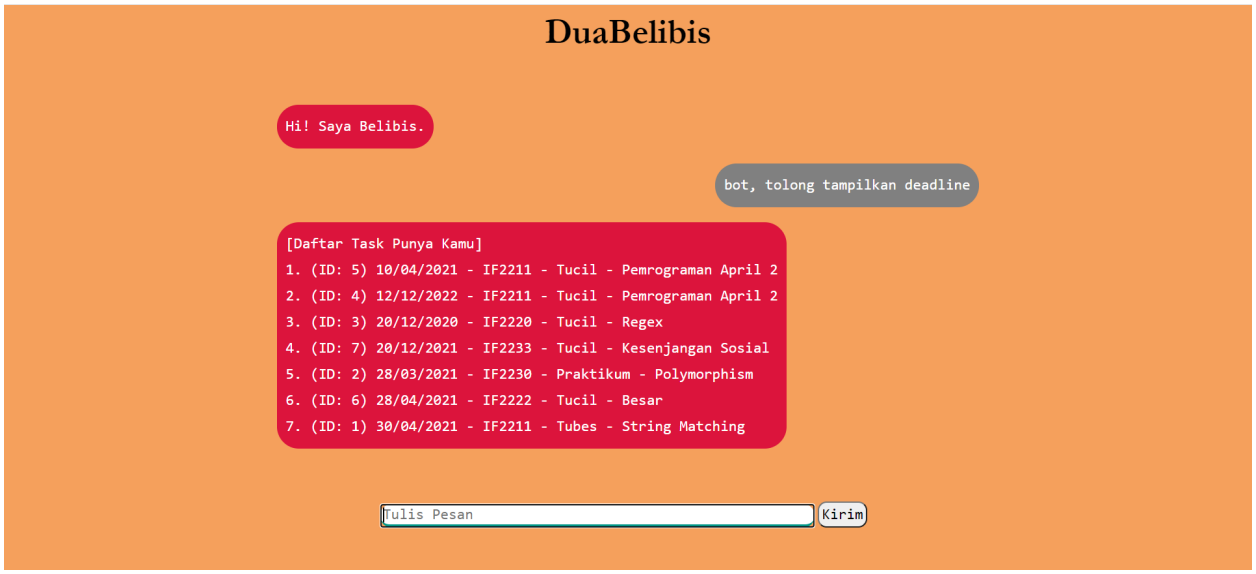
Untuk menjalankan program, disarankan menggunakan *virtual environment* dan menginstall library Flask dan SQLAlchemy. Jalankan program dengan menuliskan command flask run. Fitur yang disediakan program antara lain adalah : Menampilkan deadline, menambahkan task baru, melihat daftar task yang harus dikerjakan,

memperbaharui task, menandai task yang sudah dikerjakan, menampilkan opsi help, menampilkan pesan error, dan rekomendasi kata.

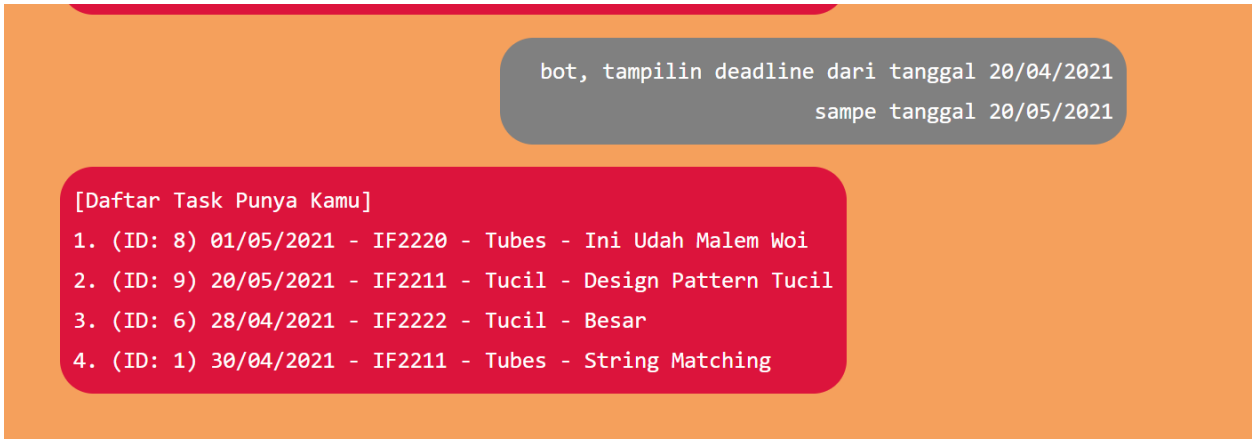
Hasil pengujian



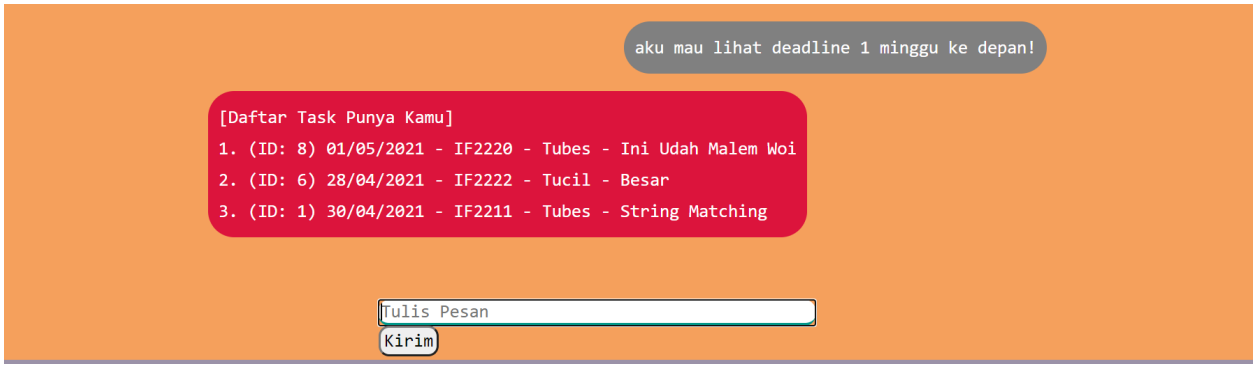
Menambahkan task baru



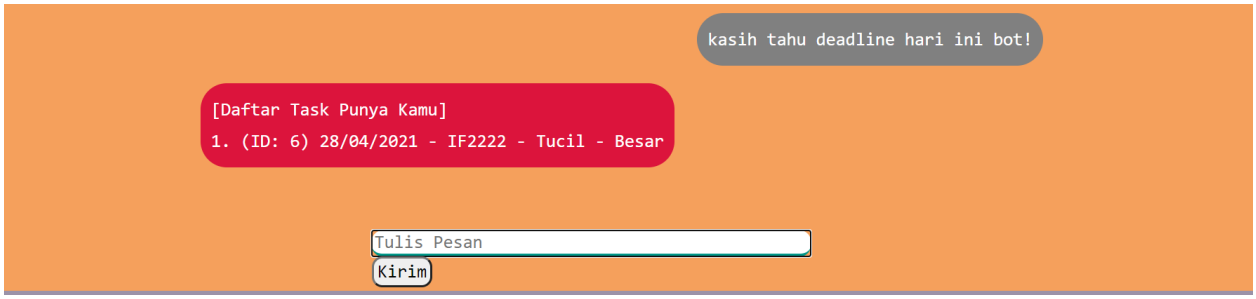
Menampilkan deadline



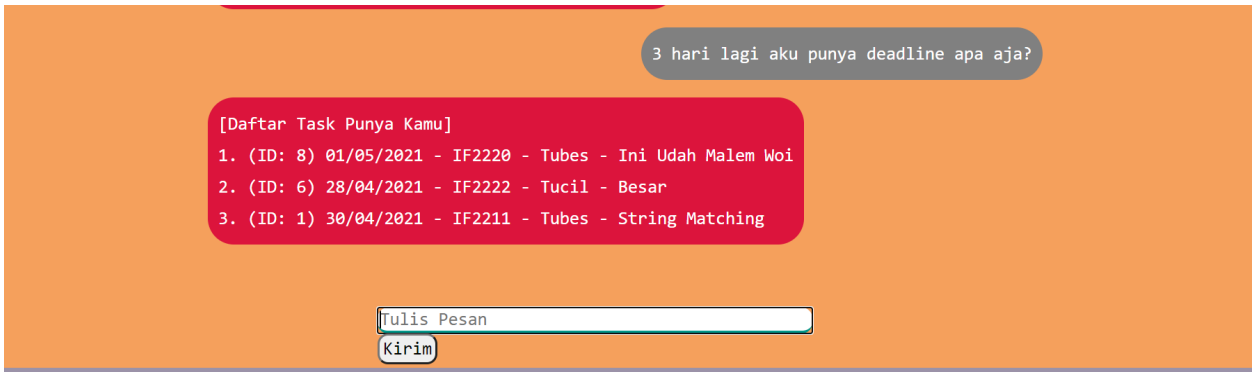
Menampilkan deadline diantara dua tanggal



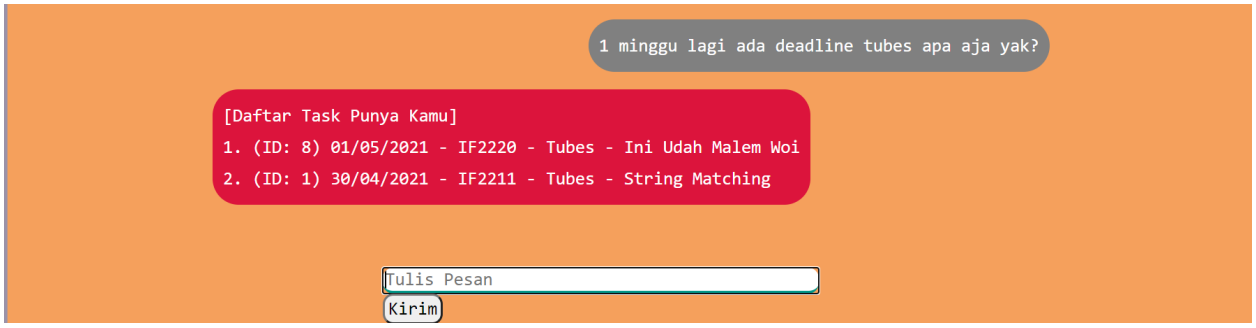
Menampilkan deadline 1 minggu ke depan



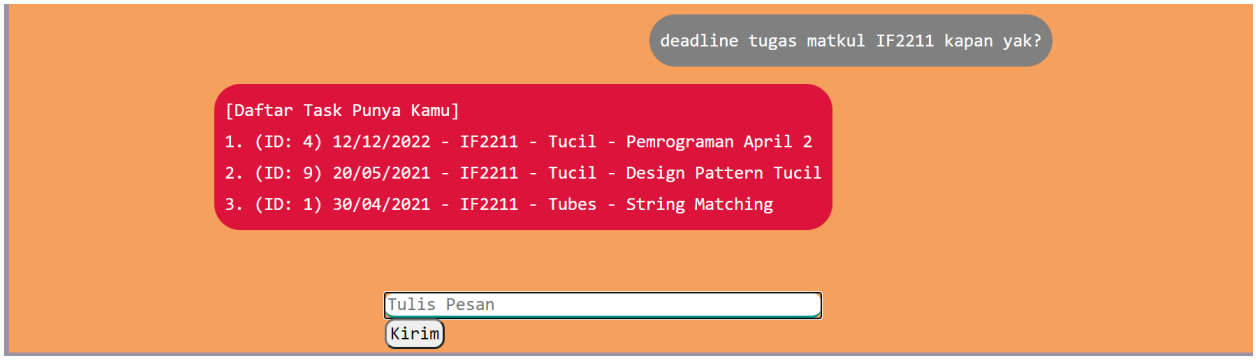
Menampilkan deadline hari ini



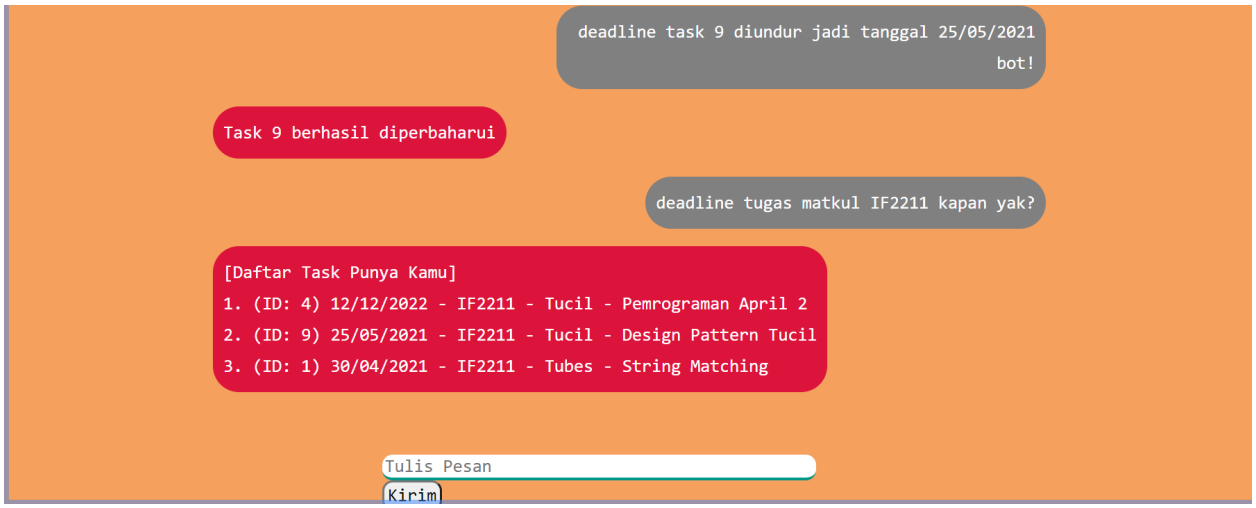
Menampilkan deadline 3 hari yang akan datang



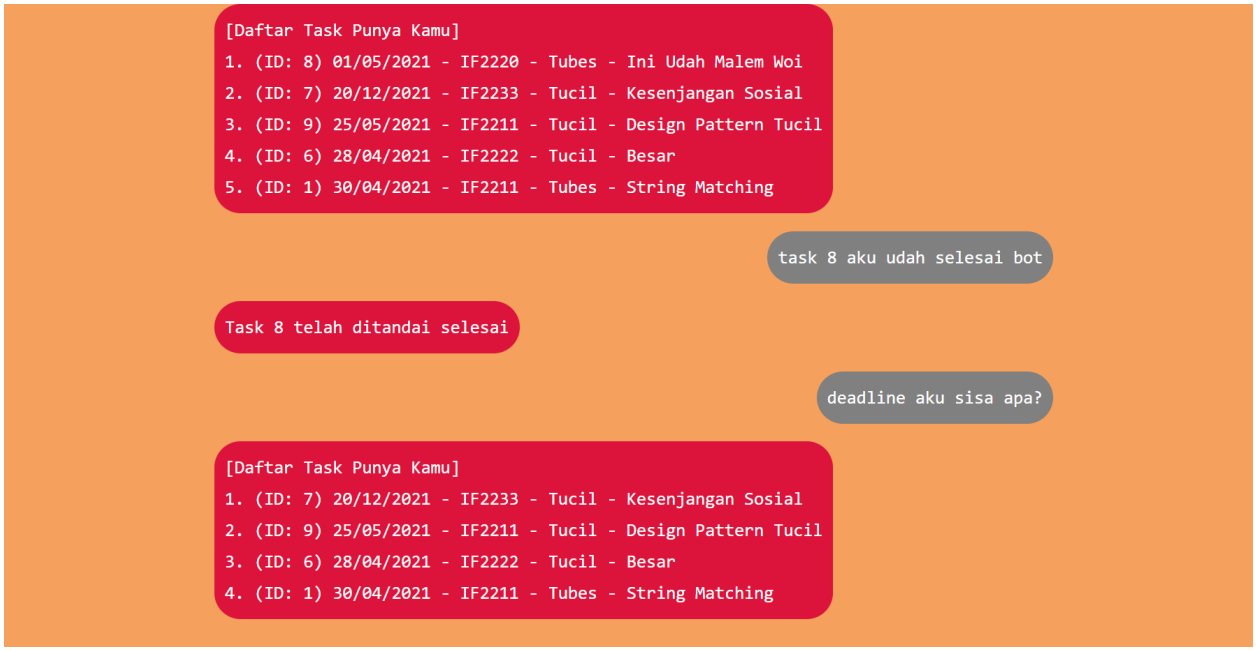
Menampilkan deadline tubes 1 minggu yang akan datang



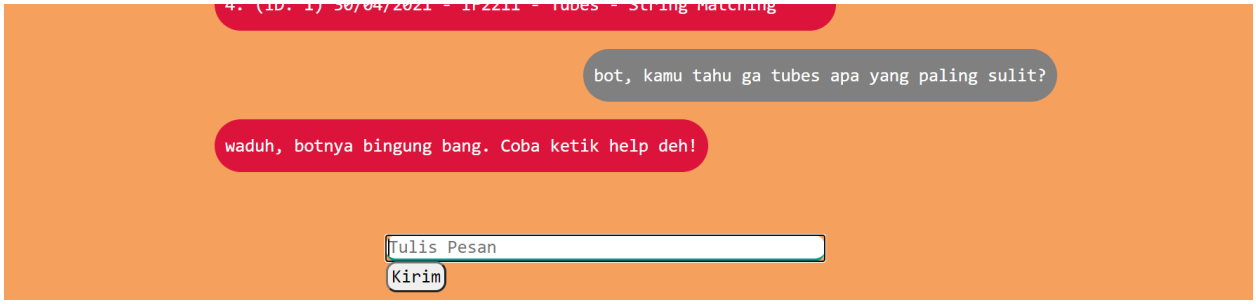
Menampilkan deadline matkul IF2211



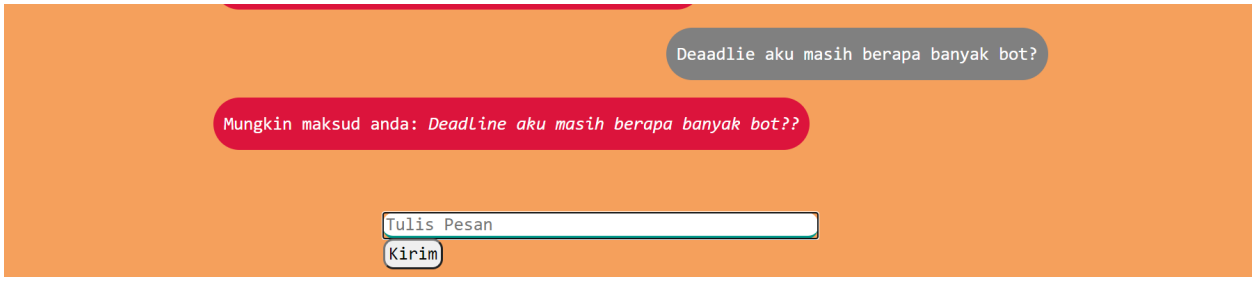
Mengupdate deadline suatu task



Menandai task sudah selesai



Pesan kesalahan apabila query tidak dikenali



Rekomendasi apabila terjadi typo namun masih mirip

Analisis Hasil Pengujian

Setelah diimplementasikan, kami merasakan bahwa program yang telah jadi berjalan dengan cukup cepat karena tidak dirasa ada proses komputasi yang terlalu berat sehingga output dari program dapat dengan cepat terlihat langsung setelah proses kalkulasi terjadi.

Karena program ini menggunakan algoritma BM sebagai salah satu kerangka utama dalam melakukan pengenalan string, kompleksitas algoritma BM secara umum adalah O(mn), sehingga ini sudah cukup baik karena bukan merupakan kompleksitas yang eksponensial ataupun yang diatasnya.

Alasan kami tidak menggunakan algoritma KMP dalam program ini adalah, karena algoritma BM bekerja dengan lebih baik apabila menangani kasus query yang memiliki ragam variasi huruf yang banyak, dan karena query kami asumsikan memiliki ragam karakter yang banyak, kami memilih untuk menggunakan algoritma KMP.

Selain itu, matriks Levenshtein Distance juga memberikan analisis yang cukup akurat untuk mengetahui hubungan jarak antara suatu kata dengan kata lainnya. Metriks ini yang kami jadikan sebagai acuan utama untuk menentukan apakah suatu query user dapat diberikan rekomendasi kata atau tidak apabila ternyata metriks ini berkata demikian.

BAB V

Kesimpulan dan Saran

Kesimpulan

Pada tugas besar IF2211 Strategi Algoritma ini kami belajar mengenai Penerapan String Matching dan Regular Expression dalam Pembangunan Deadline Reminder Assistant pada bahasa pemrograman python dengan framework yang digunakan ialah Flask. Kami menggunakan beberapa library semisal library re(regular expression) untuk melakukan pemrosesan terhadap operasi operasi regex. Selain itu kami juga menggunakan library datetime untuk mendapatkan tanggal saat ini, karena hal tersebut penting dan akan digunakan juga pada beberapa perintah di dalam program. Kami memisahkan fungsi dan prosedur pada beberapa buat file .py untuk memudahkan dalam hal pengelompokkan.

Refleksi

Sangat bermanfaat karena bisa bereksplorasi banyak hal baru mengenai pemrograman Web.

DAFTAR PUSTAKA

1. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Tugas-Besar-3-IF2211-Strategi-Algoritma-2021.pdf diakses tanggal 27 April 2021
2. https://stackoverflow.com/. diakses tanggal 27 April 2021