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 11 - DedBot

Rizky Anggita S. Siregar 13519132

Wilson Tandya 13519209

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021

BAB I
DESKRIPSI TUGAS

Dalam tugas besar ini, kami 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 membangun sebuah *chatbot* interaktif sederhana layaknya *Google Assistant* yang akan menjawab segala pertanyaan kami 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
    ```

    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

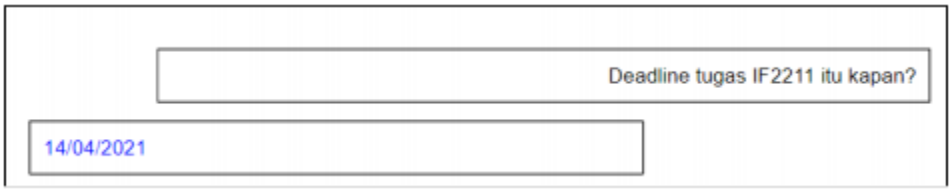
d. Contoh interaksi

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*)

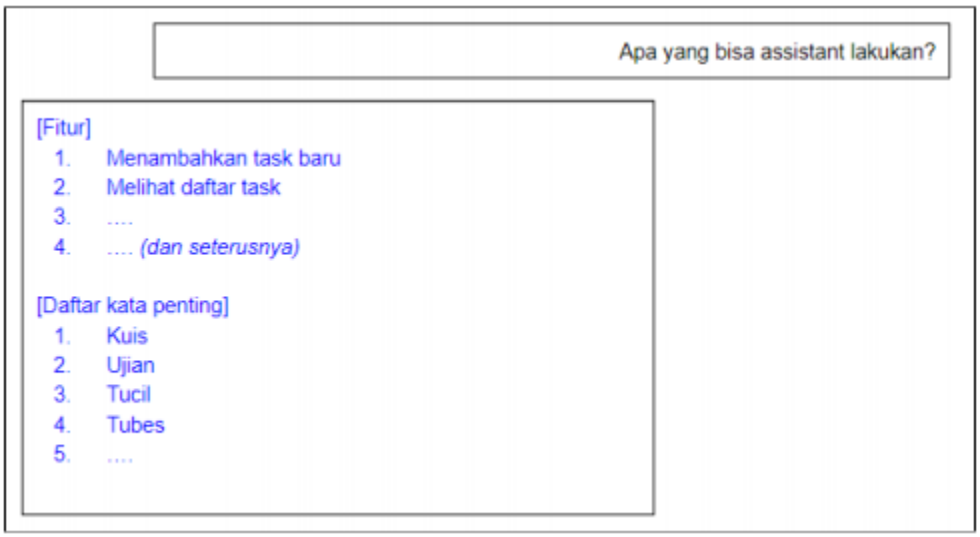
6. Menampilkan opsi *help* yang difasilitasi oleh *assistant*

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*.

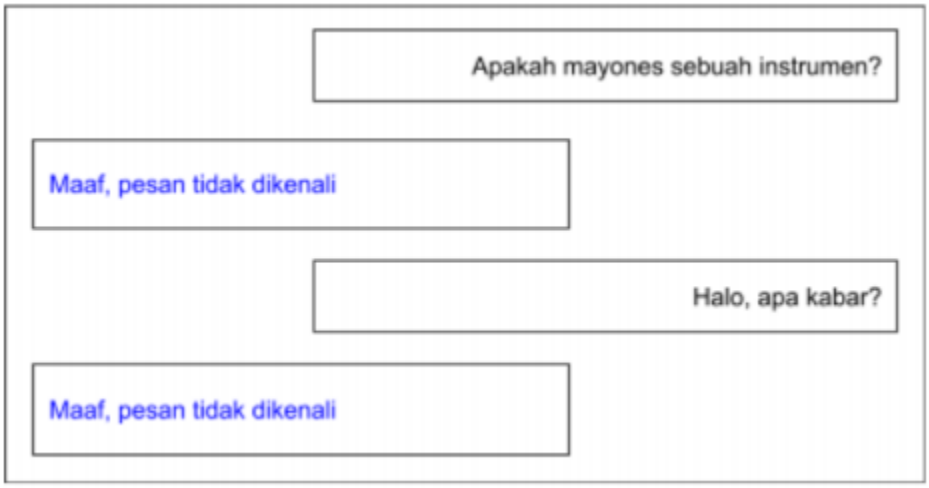
c. Daftar kata penting tidak perlu dibuat dinamis, cukup *static* saja atau *hardcoded*.

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

c. Contoh interaksi



BAB II
LANDASAN TEORI

2.1 Algoritma *KMP*, *BM*, dan *Regex*

Algoritma *Knuth-Morris-Pratt (KMP)* merupakan salah satu algoritma pencocokan *string* yang dikembangkan secara terpisah oleh Donald E. Knuth pada tahun 1967 dan James H. Morris bersama Vaughan R. Pratt pada tahun 1966. Algoritma ini mencari pola pada sebuah teks dengan urutan dari kiri ke kanan (sama dengan algoritma *brute force*), namun perbedaannya adalah, algoritma ini melakukan perpindahan yang lebih "pintar" dari *brute force*. Pada saat ketidaksamaan terjadi pada sebuah teks T dan pola P, perpindahan yang dilakukan adalah sebesar *prefix* terbesar dari P[0..j-1] yang merupakan *suffix* dari P[1..j-1]. Algoritma ini memiliki kompleksitas waktu *O(m+n)*. Kelebihan dari algoritma *KMP* ini adalah algoritma tidak pernah bergerak mundur dalam pencocokan, sehingga membuat pemrosesan *file* yang besar sangat efisien, sedangkan kekurangan dari *KMP* adalah, algoritma ini tidak bekerja baik bila jumlah alfabet besar (kemungkinan *mismatch* lebih tinggi).

Algoritma *Boyer-Moore (BM)* merupakan algoritma pencocokan *string* lain yang dipublikasikan oleh Robert S. Boyer dan J. Strother Moore pada tahun 1977. Algoritma ini dibuat berdasarkan 2 teknik, antara lain *looking-glass technique* dan *character-jump technique*. Pada teknik *looking-glass,* akan dicari pola dalam teks dengan melakukan pencarian mundur melewati pola pada bagian akhirnya, dan pada teknik *character-jump*, akan memindahkan indeks perbandingan pada saat *mismatch* terjadi. Kompleksitas algoritma *BM* adalah *O(nm +A)*. Kelebihan dari algoritma ini adalah, bekerja cepat untuk jumlah alfabet*(A)* yang besar, jadi sangat efisien untuk teks pada umumnya, dan kelemahannya adalah tidak efisien pada jumlah alfabet*(A)* seperti angka biner.

Regex atau yang juga dikenal dengan *Regular Expression* adalah salah satu metode lain dalam pencocokan *string* pula. Algoritma ini biasa digunakan dalam operasi *"find and replace"* pada sebuah *string*. Konsep ini dicetuskan sekitar tahun 1950 pada saat Stephen Cole Kleene membuat deskripsi *regular language* yang nantinya dapat ditemukan pada *Unix text-processing utilities*.

2.2 Penjelasan singkat mengenai *Chatbot*

Chatbot yang dibuat pada tugas besar 3 kami beri nama *Dedbot*, berfungsi untuk membantu mengingat berbagai *deadline*, tanggal penting, dan *task-task* tertentu pada penggunanya dengan memanfaatkan algoritma *string matching* dan *regular expression*. *Dedbot* memiliki berbagai fitur, antara lain, menambahkan *task* baru, melihat daftar *task* yang harus dikerjakan, menampilkan *deadline* dari suatu *task* tertentu, memperbaharui *task* tertentu, Menandai bahwa suatu *task* sudah selesai dikerjakan, menampilkan opsi bantuan, dan menampilkan pesan kesalahan bila *dedbot* tidak dapat mengenali masukkan pengguna.

BAB III
ANALISIS PEMECAHAN MASALAH

3.1 Langkah-Langkah Penyelesaian Masalah

3.1.1 Menambahkan *Task* Baru

Pada fitur menambahkan *task* baru, kami menyimpan *data* ID yang dibuat *auto increment* agar setiap *task* memiliki ID yang unik. Dengan menggunakan *regex* pada tanggal, kode mata kuliah, jenis tugas, dan topik tugas, kami dapat memisahkan keempat data tersebut dan menyimpannya ke *database*. Kami juga menampilkan pesan sukses bila terdapat keempat *data* tersebut.

3.1.2 Melihat Daftar *Task* yang Harus Dikerjakan

Dalam melihat daftar *task* pada *Dedbot*, filter yang pertama kali dicek apakah terdapat kata *"deadline"* kalimat. Jika tidak, maka fungsi melihat daftar task tidak dijalankan. Jika ya maka akan dilakukan pengecekan *keyword* untuk fitur melihat daftar *task* yang dimiliki sejauh ini, dengan *keyword "deadline", "semua", "dimiliki", "sejauh", "ini"*.

Dengan pengecekan menggunakan *KMP*, jika ditemukan setidaknya 4 buah kata yang sama antara *keyword* dan kata dalam kalimat inputan *user*, kemudian akan dilakukan pengecekan dengan *regex* apakah kalimat inputan *user* mengandung kata penting. Selanjutnya akan ditampilkan seluruh *task/deadline* yang dimiliki sejauh ini.

Jika tidak memenuhi kondisi pengecekan *keyword* tersebut, maka selanjutnya akan mengecek kemungkinan melihat *task* lain, yaitu *task* dengan periode tertentu, *task* N minggu kedepan, *task* N hari ke depan, dan *task* hari ini. Pengecekan keempat *pattern* tersebut dilakukan dengan *regular expression*, termasuk mengekstrak N dari kalimat inputan *user*. Jika fungsi *regex* yang kami buat mengembalikan nilai *pattern* yang sesuai, maka selanjutnya akan ditampilkan *task/deadline* yang bersesuaian dengan kalimat *input user*, termasuk jenis *task* apakah tucil, tubes, ujian, dll.

3.1.3 Menampilkan *Deadline* dari Suatu *Task* Tertentu

Dalam menampilkan *deadline* dari sebuah *task,* kami mengecek apakah terdapat kata *"tugas", "deadline"* dan *"kapan"* dengan menggunakan *regex*, selanjutnya bila semua kondisi telah terpenuhi maka kami akan menelusuri dari *database* apakah terdapat kode kuliah yang sama dengan masukkan pengguna, bila ada kami akan menampilkan tanggal tenggat dari tugas besar dan tugas kecil (karena dalam spesifikasi program hanya menampilkan tugas yang memiliki tenggat waktu) beserta ID tugas tersebut.

3.1.4 Memperbaharui *Deadline Task* Tertentu

Pada fitur memperbaharui *deadline* dari *task,* kami mengecek apakah terdapat *keyword* seperti *"deadline", "diundur", "diubah", "task", "menjadi"* dengan *minimum match* 4 buah

keywords, karena bila kurang dari 4, akan terdapat kekurangan informasi untuk menjalankan fitur ini. Setelah itu, kami melakukan penelusuran terhadap *input* ID *task* yang diberikan pengguna, dan bila ditemukan, akan diubah *deadline*-nya sesuai tanggal baru masukkan pengguna.

3.1.5 Menandai Bahwa Suatu *Task* Sudah Selesai Dikerjakan

Pada fitur menandai suatu *task* yang telah diselesaikan, sama dengan memperbaharui *deadline*, kami mengecek apakah terdapat *keyword* seperti *“sudah”, “selesai”, “mengerjakan”, “kelar”, “menyelesaikan”* dengan *minimum match* 3 buah *keywords*. Kemudian, kami menelusuri ID dari masukkan pengguna dengan bantuan *traversal* dan algoritma *KMP*, bila ditemukan, kami menghapus data *task* tersebut dari *database*.

3.1.6 Menampilkan Opsi *Help* yang Difasilitasi oleh *Assistant*

Pada fitur ini, *keyword* paling utama adalah kata *“help”*. Jika semua pengecekan *pattern* lain tidak berhasil, maka akan dilakukan pengecekan apakah terdapat kata *“help”* pada kalimat input *user*. Jika ya maka akan menampilkan daftar fitur dan list kata penting yang ada.

Jika tidak ditemukan *keyword “help”* maka akan dilakukan pengecekan dengan *keyword* berikutnya yaitu *"apa", "bisa", "dilakukan", "bot", "assistant"*. Dengan pengecekan *KMP*, jika ditemukan kesesuaian setidaknya 4 buah kata, maka *Dedbot* akan menampilkan daftar fitur dan list kata penting yang ada.

3.1.7 Mendefinisikan *List* Kata Penting

Penentuan kata penting yang kami buat bersifat *hard-coded*, yang terdiri dari 5 kata yaitu *"kuis”, “ujian”, “tucil”, “tubes”, “praktikum"* untuk menentukan apakah itu merupakan suatu *task* atau bukan.

3.1.8 Menampilkan Pesan *Error* Jika *Assistant* Tidak Dapat Mengenali Masukkan *User*.

Jika *DedBot* sudah menjalankan semua fitur yang sudah ada dan tidak menemukan pattern yang sesuai dengan kalimat *input user*, maka bot akan mengeluarkan pesan *“Bot tidak mengenali pesanmu!”*.

3.2 Fitur fungsional dan Arsitektur *Chatbot*

Fungsionalitas Chatbot yang kami buat sama seperti spesifikasi yang diberikan yaitu fitur menambah *task* baru, melihat *task* dengan berbagai parameter, menampilkan *deadline task* tertentu, memperbaharui deadline sebuah *task*, menandai sebuah *task* telah selesai dikerjakan, menampilkan opsi *help*, menampilkan pesan *error* jika bot tidak mengenali masukan, dan database *task* berupa *file .txt*. Semua fitur telah dilakukan pengujian.

Chatbot dibangun menggunakan *Python3* dengan *framework Flask* untuk bagian *backend* dan menggunakan Bootstrap untuk l*ibrary frontend*-nya. Arsitektur database aplikasi bot yang kami buat yaitu terdiri dari n buah baris, dimana baris pertama menyimpan nilai *auto-increment*

untuk atribut Id sebuah *task*, dan n-1 baris sisanya merupakan *task*. Tiap baris berisi data untuk sebuah *task*, dengan setiap atributnya dipisahkan spasi (*space-separated value*). *Database* disimpan dalam format *.txt*. Tiap kali *server*/aplikasi dijalankan, maka *database* ini akan diekstrak dan disimpan ke dalam struktur *data list* pada aplikasi. Ketika dilakukan perubahan, yaitu *insert, update, delete*, maka akan secara langsung perubahan tersebut ditulis ke *database.txt.*

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1 Spesifikasi Teknis Program

Pada *bot.py* kami menggunakan struktur data *list of list string* untuk menyimpan *database* yang tersedia dalam bentuk *file .txt*. Terdapat fungsi tambah_task dengan parameter kalimat dan AUTO_INCREMENT (untuk memberi ID pada sebuah *task* yang ditambahkan), kemudian, terdapat fungsi lihat_task yang memiliki 4 status yang membedakan lihat_task berdasarkan periode waktu, selanjutnya terdapat fungsi lihat_deadline, ubah_deadline, task_selesai yang ketiganya berparameter kalimat masukkan pengguna dan database yang telah di *load*, terakhir terdapat fungsi help_bot yang berparameter masukkan dari pengguna. Seluruh fungsi yang dibuat di *bot.py* ini menandakan satu fitur yang diberikan pada spesifikasi tugas besar.

Pada *function.py* terdapat 5 fungsi yang berguna untuk melakukan konversi agar data masukkan pengguna memiliki format yang sama dengan yang disimpan dalam *database*. Fungsi tersebut meliputi, konvert_tahun, konvert_bulan, konvert_hari, konvert_tanggal, data_db_to_String.

Pada *KMP.py* hanya terdapat satu fungsi yang mengembalikan *True* bila ditemukan *match* sebuah *pattern* pada sebuah *text* yang dimasukkan di parameternya.

Pada *regex.py* kami menggunakan *library re* dan membuat fungsi regex_tanggal, regex_kodekuliah, regex_katapenting, regex_topik, regex_lihattask, regex_get_nTask, dan regex_tugas, semuanya berfungsi untuk mencari kecocokan dari kata pada *database* sesuai dengan format yang kami tetapkan. Tanggal memiliki format XX Nama_Bulan YYYY, kode kuliah memiliki format IFXYAA, dimana X[1-9], Y[0-3], A[0-9]. Kata penting terdiri dari kuis, ujian, tucil, tubes, dan praktikum. Pada topik, format yang kami tetapkan adalah kata “topik” diikuti dengan 2 kata berikutnya akan menjadi topik dari *task*.

Pada *app.py* terdapat fungsi home yang digunakan untuk menampilkan *landing page* dari *DedBot* dan get_response digunakan untuk mendapatkan *input user* pada *halaman web*. *Dedbot.html, landing.html,* dan *layout.html* digunakan untuk keperluan *website* yang dibuat, terdapat *main.css* untuk *styling,* dan *database.txt* untuk menyimpan *data deadline*.

4.2 Tata Cara Penggunaan Program

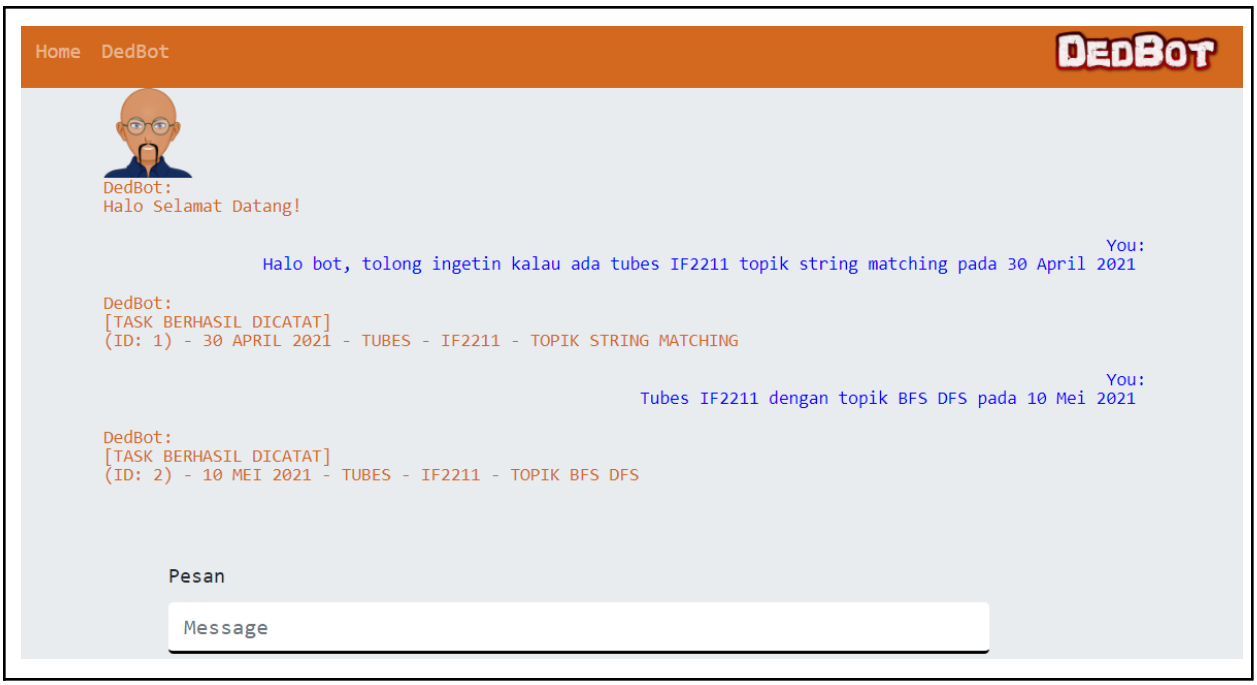
Untuk menjalankan program, dibutuhkan *Python 3* dan *Flask 1.1.2*, bila kedua hal tersebut telah dimiliki maka dengan menjalankan *app.py* dan membuka http://127.0.0.1:5000/ secara lokal pada *web browser* program sudah dapat dipakai. Pada saat pertama kali masuk, akan diarahkan ke *home* dari program dan dengan menekan tombol *DedBot* pada *navigation bar*, program langsung dapat digunakan.

Fitur yang disediakan oleh program *DedBot* antara lain, *DedBot* mampu menambahkan sebuah *deadline* baru, *DedBot* mampu menampilkan task yang dimiliki dengan berbagai parameter, seperti semua *deadline* saat ini, *deadline* N hari ke depan, *deadline* N minggu ke

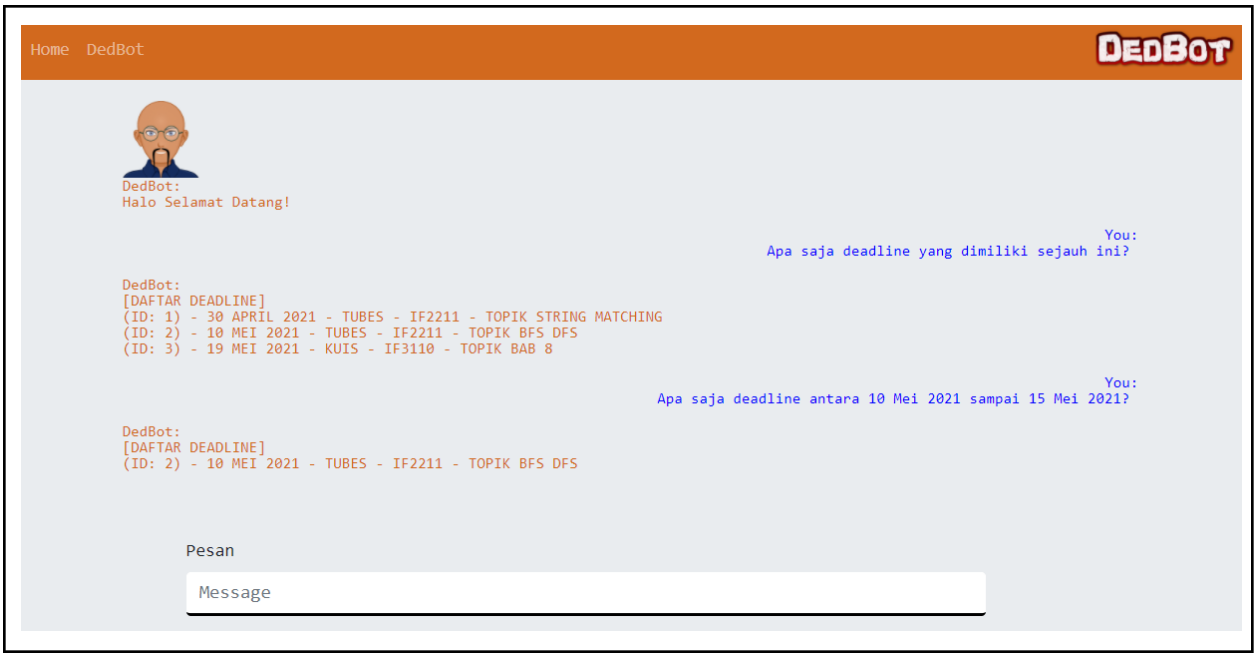
depan, dan *deadline* hari ini, *DedBot* mampu menampilkan *deadline task* tertentu dengan *keyword* kode kuliah, *DedBot* mampu memperbaharui *deadline* sebuah *task*, dan terakhir *DedBot* mampu menandai dan menghapus *task* yang sudah diselesaikan.

4.3 Hasil Pengujian

Gambar 4.3.1 Tampilan Awal *DedBot*



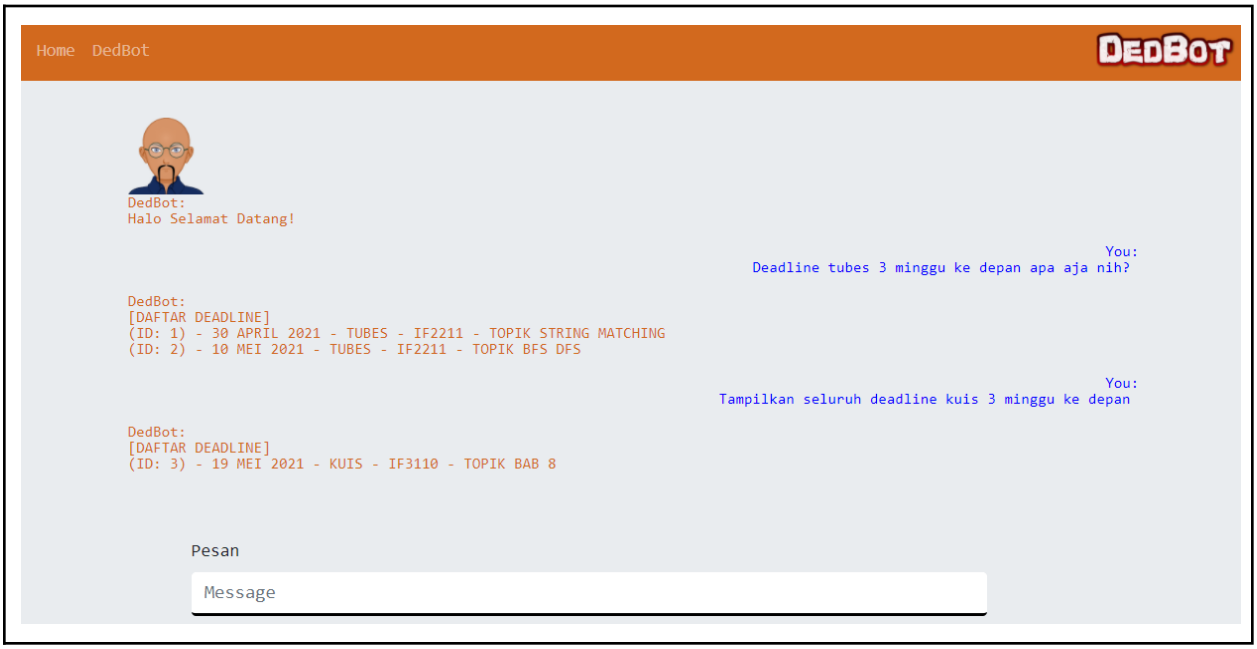
Gambar 4.3.2 Fitur menambahkan *task* baru



Gambar 4.3.3 Fitur melihat seluruh *task* yang sudah tercatat dan pada periode tertentu

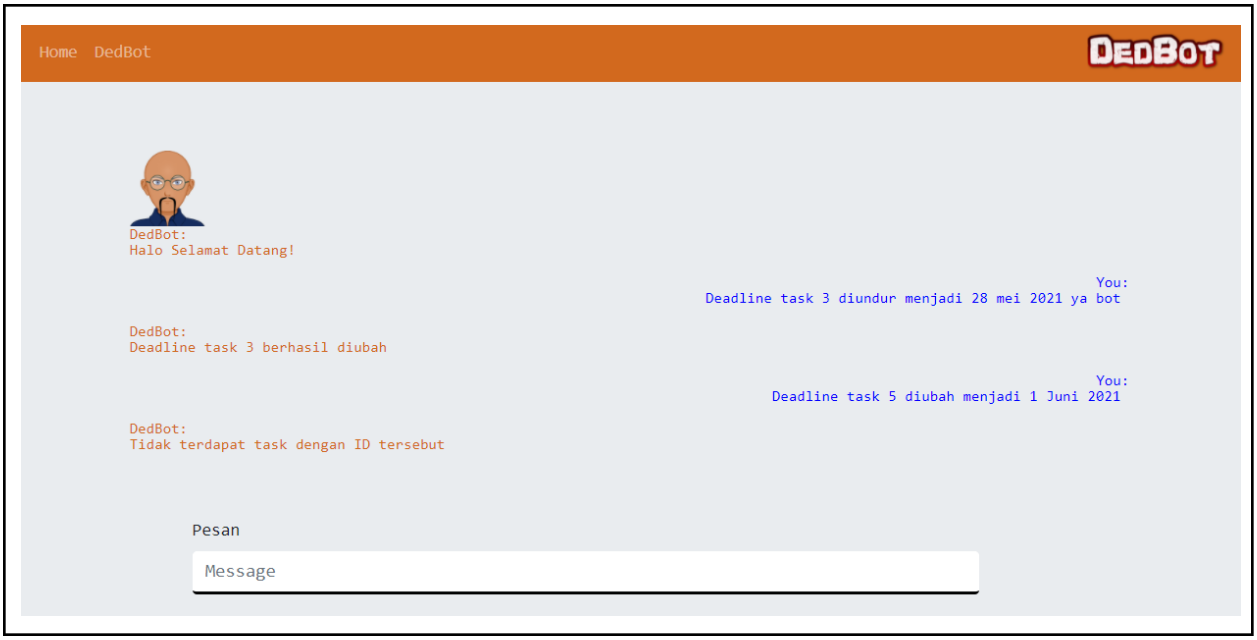
Gambar 4.3.4 Fitur melihat *task* N minggu ke depan dan N hari ke depan

Gambar 4.3.5 Fitur melihat *task* hari ini

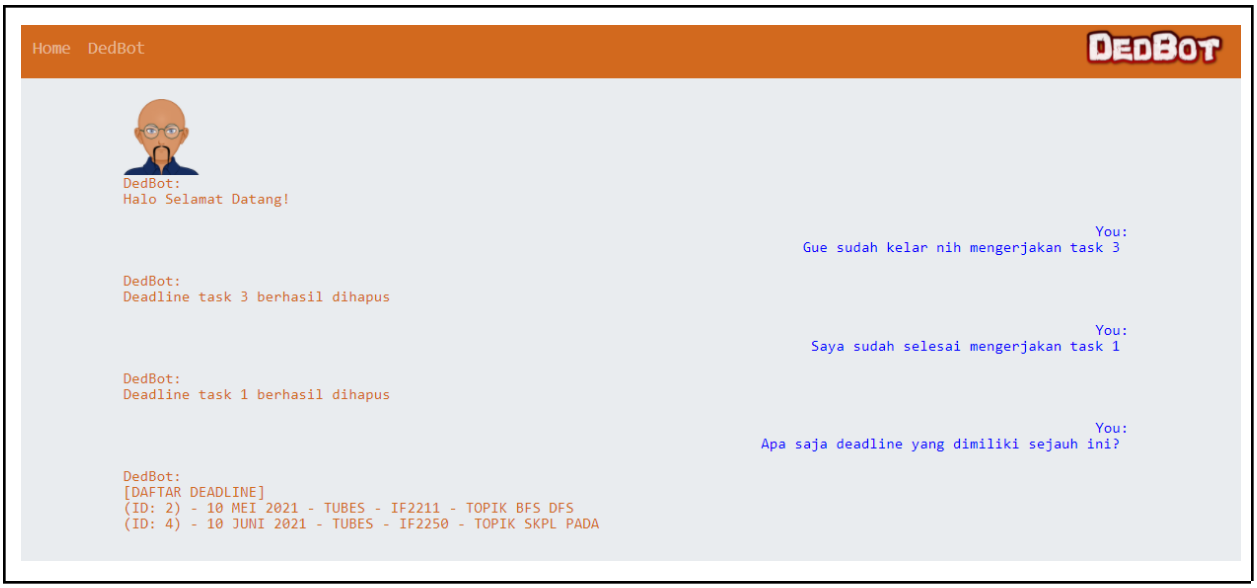


Gambar 4.3.6 Fitur melihat *task* dengan kata penting

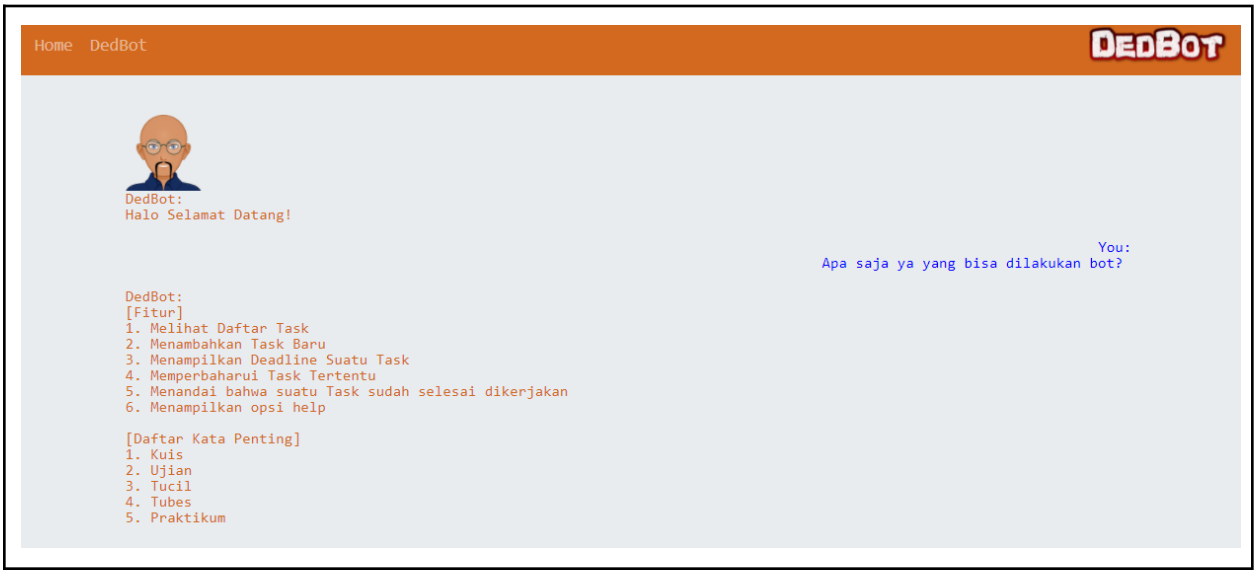
Gambar 4.3.7 Fitur menampilkan *deadline* dari suatu *task* tertentu



Gambar 4.3.8 Fitur memperbaharui *task* tertentu



Gambar 4.3.9 Fitur menandai bahwa suatu *task* sudah selesai dikerjakan



Gambar 4.3.10 Fitur menampilkan opsi *help* yang difasilitasi oleh *assistant*

Gambar 4.3.11 Fitur menampilkan opsi *help* yang difasilitasi oleh *assistant* (variasi *command*)

Gambar 4.3.12 Fitur menampilkan pesan *error* jika *assistant* tidak dapat mengenali *input user*

Gambar 4.3.13 Fitur setiap *task* memiliki ID unik

4.4 Analisis Hasil Pengujian

Pada Gambar 4.3.2 ditunjukan 2 dari banyak alternatif pengujian fitur menambahkan *task* baru dan disimpan data-data yang penting, seperti tanggal, kata penting, kode mata kuliah, dan topik ke dalam sebuah *list*. Untuk menampilkan seluruh daftar *deadline* yang disimpan dalam *database* berupa *file .txt*, dapat terlihat pada Gambar 4.3.3. Selain itu, *DedBot* juga memiliki fitur untuk melihat daftar *deadline* pada *interval* tanggal tertentu, N minggu ke depan, N hari ke depan, dan *deadline* pada hari tersebut yang tersimpan dalam *database*.

Pada Gambar 4.3.6 ditunjukan pula fitur untuk melihat *task* sesuai daftar kata penting yang kami *hard-coded*, antara lain, *"kuis", "ujian", "tucil", "tubes", "praktikum"*. Pada fitur ini, hanya tugas dalam kata penting dalam periode yang kita masukkan muncul. Selain itu, terdapat pula fitur menampilkan *deadline* dari suatu *task* yang spesifik, dapat dilihat pada Gambar 4.3.7, dilakukan pencarian *deadline* dari Tugas dengan kode mata kuliah tertentu, dan mengembalikan pesan kesalahan bila tugas tersebut tidak tersimpan di dalam *database.*

Dalam Gambar 4.3.8 ditampilkan fitur memperbaharui *task* tertentu, kami membuat agar dapat digunakan *keyword "diundur"* atau "*diubah"* karena *deadline* dari sebuah *task* dapat dipercepat juga. Bila terdapat *task* dengan ID yang tidak tercatat dalam *database,* akan ditampilkan pesan kesalahan. Fitur berikutnya yang dimiliki *Dedbot* adalah menandai bahwa suatu *task* telah selesai dikerjakan, pada *DedBot*, hal ini cukup dilakukan dengan menghapus data dari *database* sesuai dengan ID dari *task* yang dimasukkan oleh pengguna.

Dalam *DedBot* ini pula terdapat fitur *help* yang dapat diakses dengan menggunakan *matching keywords* seperti pada Gambar 4.3.10, atau cukup dengan mengetikan *"help"* seperti pada Gambar 4.3.11. Setiap pesan yang tidak dikenali oleh *DedBot* akan kami berikan pesan kesalahan *"Bot tidak mengenali pesanmu!"* seperti yang ditampilkan pada Gambar 4.3.12.

Pada Gambar 4.3.13, disimulasikan penambahan suatu *deadline* setelah melakukan penghapusan terhadap *task* ID terakhir, dan dapat dilihat bahwa pada *database, DedBot* menyimpan ID *task* yang terakhir, jadi tidak akan terbuat ID yang sama untuk *task* berbeda walaupun *task* telah dihapus.

BAB V
SIMPULAN

5.1 Kesimpulan

Kesimpulan dari tugas besar ini adalah sebuah *chatbot* sederhana *deadline reminder* dapat dibuat dengan menggunakan kombinasi dari algoritma *KMP* dan *Regular Expression* dan dapat bekerja tanpa melakukan *exact matching* dari masukkan pengguna.

5.2 Saran

Dalam pengerjaan tugas kedepannya dengan topik serupa, kami memberikan saran untuk meluangkan lebih banyak waktu dalam melakukan desain *website* dari *chatbot* agar terkesan lebih bagus dan bersih untuk dilihat.

5.3 Refleksi dan Komentar

Pada tugas besar 3 strategi algoritma ini kami membuat sebuah *bot* yang bertindak seperti *Google Assistant* dengan menggunakan bahasa pemrograman *Python* yang dibangun dalam sebuah *web*. Komentar kami terhadap tugas besar ini adalah, walaupun menggunakan kakas yang sudah familiar dari tugas-tugas sebelumnya, namun menambah pengalaman karena dapat membuat sesuatu yang dapat langsung dipakai oleh *user* manapun dan memiliki kegunaan secara langsung bagi penggunanya.

DAFTAR PUSTAKA

Munir, R. 2021. *Pencocokan string (String matching/pattern matching) (2021)*. Bandung: Institut Teknologi Bandung.

Munir, R. 2021. *Pencocokan string dengan Regular Expression (Regex) (2021)*. Bandung: Institut Teknologi Bandung.

Python3. *Dokumentasi Library re Python3*, https://docs.python.org/3/library/re.html