

# **LAPORAN TUGAS BESAR 3**

## **IF2211 STRATEGI ALGORITMA**

Semester II Tahun 2020/2021

**Penerapan String Matching dan Regular Expression dalam  
Pembangunan Deadline Reminder Asisten**



### **Disusun Oleh :**

Melita	(13519063)
Fransiskus Febryan Suryawan	(13519124)
David Owen Adiwiguna	(13519169)

**Program Studi Teknik Informatika  
Sekolah Teknik Elektro dan Informatika  
Institut Teknologi Bandung**

## 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.

## II. Landasan Teori

Algoritma KMP (Knuth-Morris-Pratt) merupakan algoritma pencocokan string hasil pengembangan solusi *brute-force* untuk pencocokan string. Algoritma ini mencocokkan string dengan berjalan dari awal ke akhir layaknya solusi *brute-force*, namun penggeseran dilakukan secara lebih efektif. Algoritma ini memunculkan fungsi pinggiran KMP (KMP *border function*) yang didefinisikan sebagai ukuran prefix terbesar dari pola  $P[0..k]$  yang juga merupakan *suffix* dari  $P[1..k]$ . Fungsi pinggiran (sering juga disebut *fail function*) digunakan untuk menggeser posisi pola, sehingga penggeseran tidak dilakukan satu per satu. Algoritma ini memiliki kompleksitas waktu  $O(m)$  untuk menghitung fungsi pinggiran dan  $O(n)$  untuk mencari pola dalam string.

Algoritma Boyer-Moore merupakan algoritma pencocokan string yang memberikan cara penyelesaian berbeda dengan algoritma KMP, yaitu mencocokkan string dengan berjalan dari belakang ke depan pola. Jika ditemukan karakter pada pola yang tidak sesuai dengan teks ( $P[j] \neq T[i]$ ), akan dilakukan *character-jump* dan pemeriksaan diulang mulai dari karakter paling belakang. Ada 3 kemungkinan *character-jump* yang dilakukan. Pertama, jika karakter yang tidak cocok pada teks tidak ditemukan pada pola, dilakukan pergeseran sehingga  $P[0]$  pola sejajar dengan  $T[i+1]$ . Kedua, jika karakter yang tidak cocok pada teks hanya ditemukan pada sebelah kanan indeks yang sedang diperiksa dalam pola, pola  $P$  akan digeser ke kanan satu langkah. Ketiga, jika karakter yang tidak cocok pada teks ditemukan pada sebelah kiri indeks yang sedang diperiksa dalam pola, akan diambil indeks kemunculan terakhir karakter tersebut sebelum indeks  $j$ , lalu  $P$  digeser sehingga kemunculan terakhir karakter tersebut sejajar dengan  $T[i]$ .

*Regular expression* adalah teknik pencarian pola dalam string dengan menggunakan urutan karakter tertentu. Teknik ini muncul dengan berkembangnya teori bahasa formal. Dengan memanfaatkan sifat-sifat dari *regular language*, suatu *regular expression* dapat mencari pola dalam string apapun. Pencocokan dengan teknik regular expression dilakukan dengan memberikan *string literal* yang dicocokkan langsung dengan karakter yang sama dan *metacharacter* yang merupakan karakter khusus yang dipetakan kepada karakter lain. Selain itu, terdapat *quantifier* yang memberikan berapa kali suatu *string literal* atau *metacharacter* dicocokkan.

Chatbot adalah program yang didesain sedemikian rupa sehingga program dapat berkomunikasi dengan pengguna melalui *chat* dengan bahasa alami. Chatbot biasanya menggunakan kumpulan data yang digunakan untuk dapat menyajikan informasi atau jasa. Terdapat dua jenis chatbot, yaitu *open chatbot* dan *closed chatbot*. *Open chatbot* mempelajari masukan dari pengguna untuk membuat respon yang lebih sesuai dengan AI, sedangkan *closed chatbot* tidak. Chatbot biasanya digunakan sebagai *customer service* atau *assistant* pada gawai.

### **III. Analisis Pemecahan Masalah**

#### **III.I. Langkah pemecahan masalah**

Permasalahan yang diberikan adalah bagaimana membuat suatu *chatbot* yang dapat merespon teks dari pengguna secara dinamis. Konsep kata penting melandasi solusi yang akan diberikan. Dengan mencari kata penting dalam teks dari pengguna, *chatbot* dapat menentukan bagaimana cara merespon yang paling tepat kepada pengguna. Dengan menambahkan kata penting pada *chatbot*, maka *chatbot* tidak perlu memahami keseluruhan kalimat.

Setelah menangkap kata penting dalam teks, *chatbot* selanjutnya harus melakukan aksi yang sesuai dengan yang diharapkan pengguna. Karena itu, kata penting atau kata kunci dikelompokkan menjadi beberapa kelas, untuk membedakan penggunaannya. Kemudian setelah dikelompokkan, *chatbot* memproses teks untuk mendapatkan informasi tambahan yang dibutuhkan, seperti tanggal dan jenis tugas.

Untuk menangkap kata penting dalam teks, dapat digunakan algoritma pencocokan string Boyer-Moore maupun Knuth-Morris-Pratt. Namun, karena algoritma tersebut bersifat *exact-matching*, maka untuk menangkap data lain dibutuhkan metode yang lebih dinamis. Maka, digunakan *Regular Expression* untuk menangkap berbagai bagian kalimat yang penting namun tidak dapat diketahui secara pasti.

#### **III.II. Fitur fungsional dan arsitektur**

Chatbot yang dibuat memiliki beberapa fitur sebagai berikut.

- 1) Menambah *task* baru
- 2) Melihat daftar *task* yang harus dikerjakan
- 3) Menampilkan *deadline* suatu *task* tertentu
- 4) Memperbarui *task* tertentu
- 5) Menandai bahwa suatu *task* telah selesai dikerjakan
- 6) Menampilkan opsi *help* yang difasilitasi oleh *assistant*
- 7) Menampilkan pesan *error* jika masukan *user* tidak dikenali
- 8) Memberikan rekomendasi jika terdapat kesalahan kata (*typo*)

## IV. Implementasi dan Pengujian

### IV.I. Spesifikasi teknis program (struktur data, fungsi dan prosedur)

Data *task* yang diterima dari pengguna disimpan dalam file `tasks.csv`. File ini dapat diakses oleh program kapan saja, untuk mencari *task* maupun untuk menambahkan *task* baru. File ini berisi 6 kolom, yaitu tipe, topik, matkul, deadline, sudah, dan id. Kolom tipe menjelaskan tipe dari *task*. Kolom topik menjelaskan topik dari *task*. Kolom matkul menjelaskan kode mata kuliah yang diasosiasikan dengan *task*. Kolom deadline menyimpan tanggal deadline dari *task*. Kolom sudah memberikan informasi mengenai apakah *task* sudah diselesaikan. Kolom id merupakan kolom yang mengidentifikasi setiap baris secara unik.

Program dibagi menjadi 2 bagian besar, yaitu *front-end* dan *back-end*. Bagian *front-end* hanya digunakan untuk menampilkan data secara cantik kepada pengguna. Bagian *back-end* menyimpan semua implementasi fungsi *pattern-matching* yang relevan, serta berbagai fungsi pembantu untuk membaca dan menulis ke file csv.

Bagian *back-end* dari program dipecah menjadi 4 file, dengan 3 file modul dan 1 file utama. File utama dari bagian ini adalah file `server.js`, yang menginstruksikan bagaimana cara *back-end* dijalankan, dan bagaimana merespon kepada pengguna. File tersebut menggunakan salah satu modul, yaitu modul `stringmatching.js` yang menyimpan berbagai fungsi untuk pencocokan pola dalam teks. Fungsi dalam `stringmatching.js` yang digunakan oleh file utama adalah fungsi `generateReply`, yang mencocokkan semua kemungkinan kata penting pada teks dari pengguna. Selain itu, ada juga implementasi algoritma KMP, Boyer-Moore, Levenshtein Distance, serta fungsi-fungsi pembantu kecil lainnya untuk mencari data dengan regex.

Modul lainnya berhubungan dengan file eksternal, yaitu modul `database.js` yang memberikan abstraksi “database”, serta modul `util.js` yang memberikan pembaca dan penulis untuk file CSV. File CSV digunakan untuk menyimpan *task* yang diterima dari pengguna dan juga untuk menyimpan kata-kata penting yang dikenali oleh program.

### IV.II. Tata cara penggunaan program

Program dapat dijalankan dengan terlebih dahulu menginstall node.js. Pastikan versi node.js yang dimiliki lebih atau sama dengan 15, karena ada beberapa fungsi yang baru ditambah pada versi tersebut dan digunakan dalam program. Kemudian, program dapat langsung dijalankan dengan menjalankan `npm run start`. Ketika server sudah menyala, antarmuka grafis dapat diakses melalui browser dengan membuka `localhost:6969`

Program dapat menambah *task* baru jika terdapat semua data yang dibutuhkan untuk membuat *task*, yaitu jenis tugas (tubes, tucil, pr, praktikum, ujian), tanggal *deadline* tugas, mata kuliah dalam format HHAAAA (H adalah huruf dan A adalah angka), dan topik dari tugas yang diapit oleh tanda petik dua (“”). Jika topik tidak ada, akan dikosongkan dalam data simpanan.

Daftar *task* yang disimpan dapat dilihat dengan mengetikkan kata tanya yang ada pada daftar kata tanya seperti “apa” dan “kapan” disertai dengan jenis tugas. Untuk jenis tugas yang dilihat, dapat dimasukkan kata kunci “semua” untuk semua jenis tugas, dan kata jenis tugas untuk menampilkan jenis tugas tertentu saja. Dapat juga digunakan kode mata kuliah

untuk mendapatkan semua tugas pada mata kuliah tersebut. Dapat digunakan kata kunci “hari ini” untuk melihat tugas dengan *deadline* pada tanggal *query* dibuat, atau dapat digunakan rentang antara dua tanggal untuk melihat *deadline* rentang tanggal tertentu. Selain itu, dapat digunakan perintah “... N hari ke depan” atau “...N minggu ke depan” untuk melihat *deadline* pada jangkauan waktu tersebut.

Suatu *task* dapat diubah tanggal *deadline*-nya dengan menggunakan perintah “...task N” dengan N berupa ID *task* dan kata kunci ubah seperti “ganti” dan “undur” yang ada pada daftar kata ubah disertai tanggal *deadline* yang baru. Suatu *task* dapat ditandai selesai dengan menggunakan perintah “...task N” dengan N berupa ID *task* dan kata kunci selesai seperti “sudah” dan “beres” sesuai daftar kata selesai. *Task* yang sudah ditandai selesai tidak akan muncul pada saat program menerima perintah untuk menampilkan daftar *deadline*.

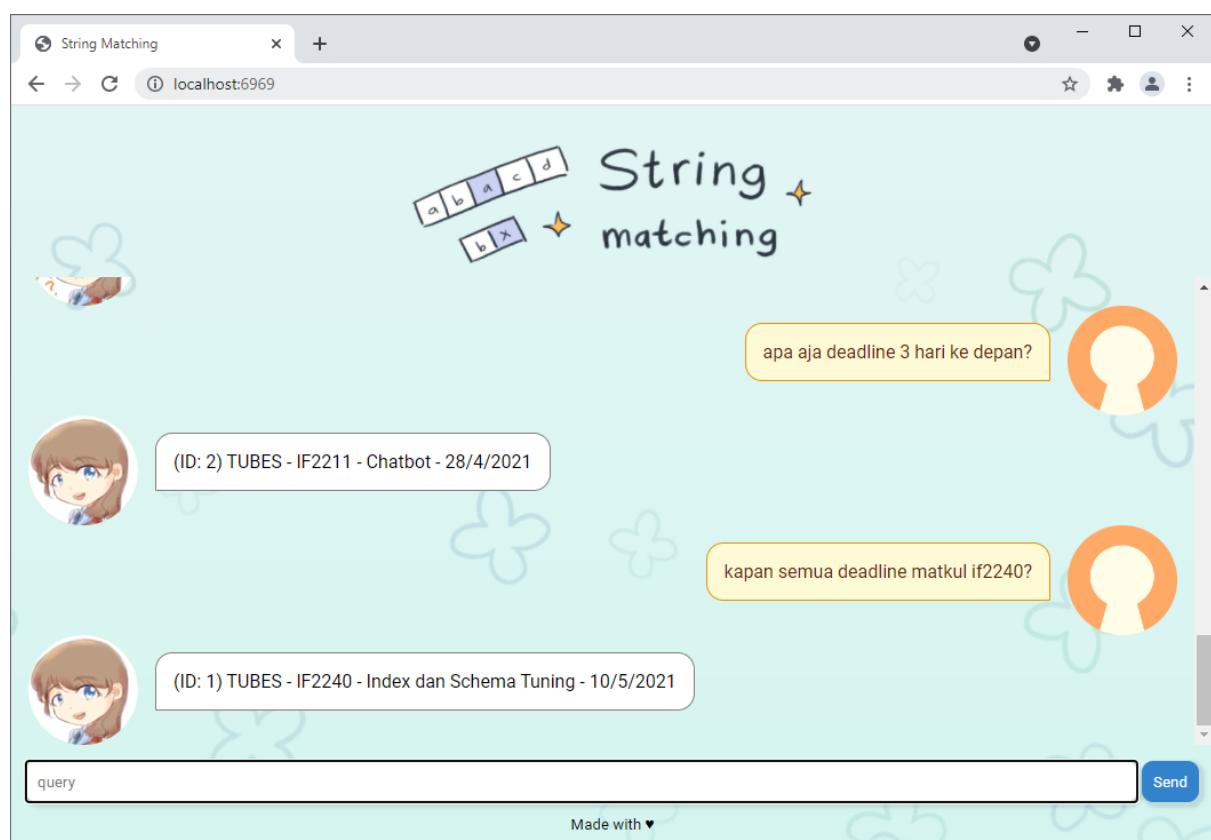
Program dapat menampilkan informasi berupa daftar kata penting yang ada dan fitur yang tersedia jika menerima kata kunci “help.” Selain itu, jika pesan tidak dikenali, akan dimunculkan sebuah pesan *error*. Jika ditemukan kata yang mirip dengan kata penting tetapi tidak sama persis karena kesalahan pengetikan, dapat ditampilkan rekomendasi kata yang benar tergantung dengan tingkat kemiripannya.

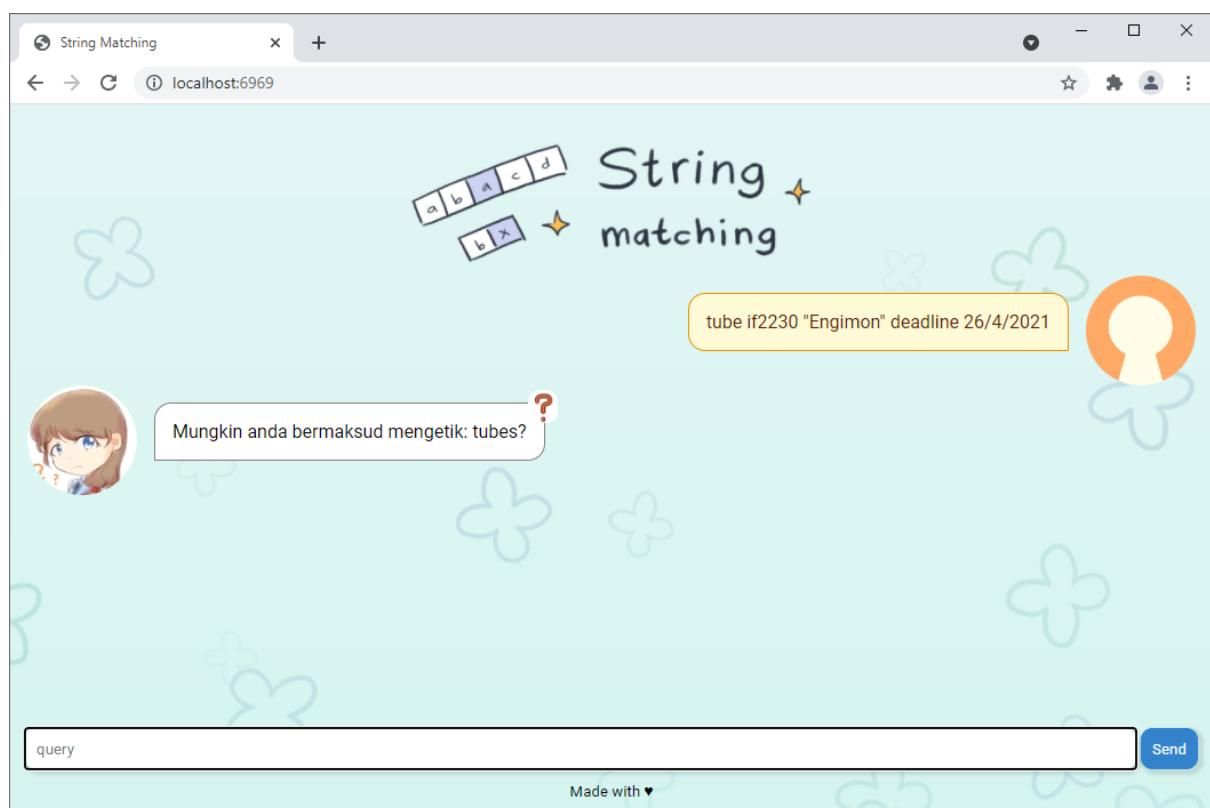
#### IV.III. Hasil pengujian

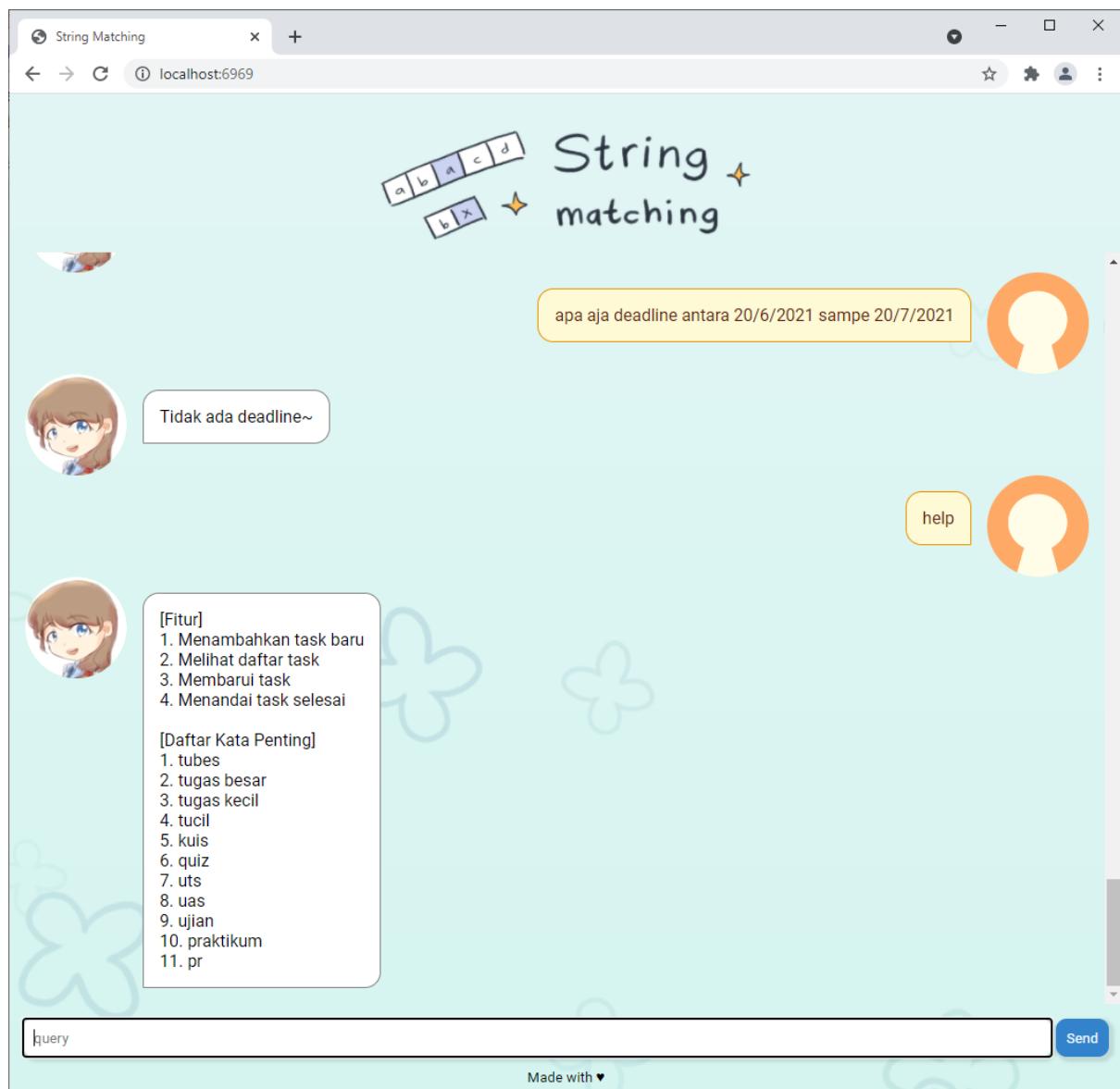
[tambah task]



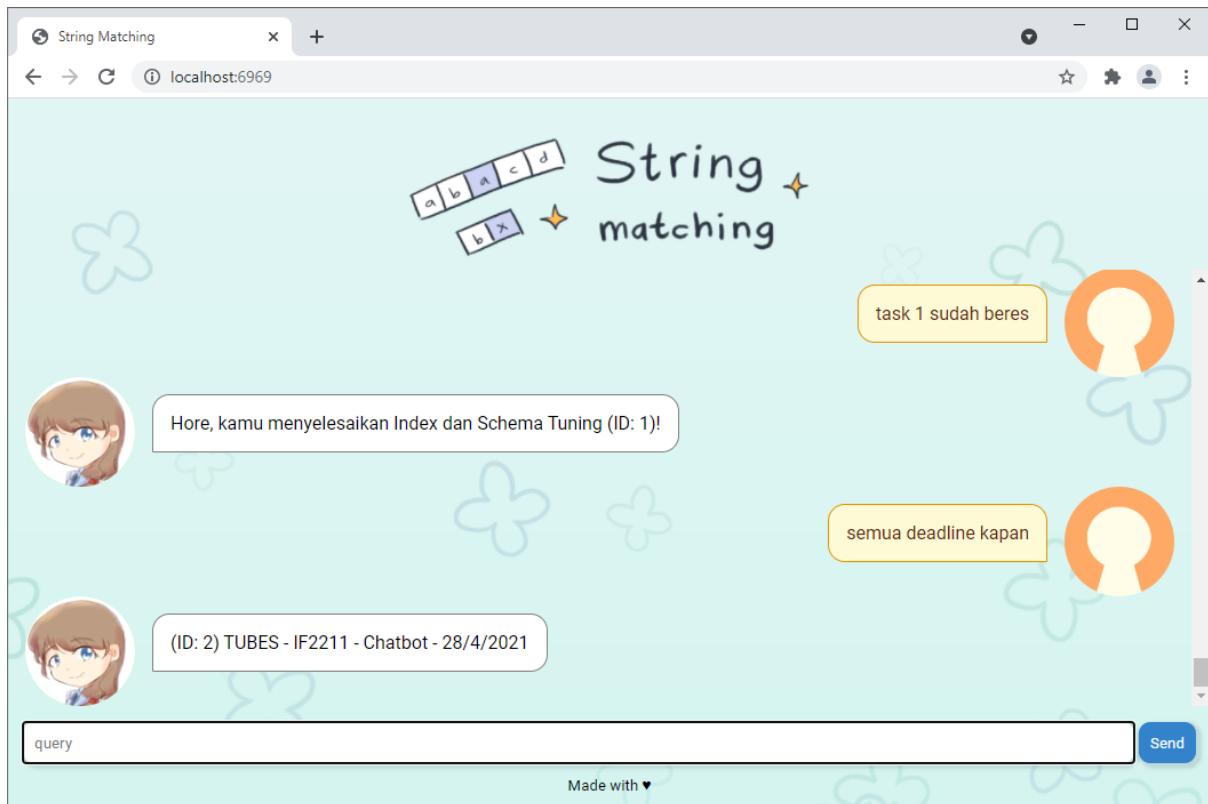
[lihat task]







[selesaikan task]



#### IV.IV. Analisis hasil pengujian

Algoritma *string matching* yang dibuat berhasil menemukan kata yang cocok pada *query*. Fungsi Levenshtein *distance* yang digunakan juga berhasil menghasilkan nilai kemiripan antarkata sehingga bot dapat mengembalikan saran kata penting yang tepat jika ada kesalahan pengetikan yang sedikit seperti kata yang kekurangan satu huruf. Semua data *task* disimpan dalam sebuah file .csv, sehingga data tetap tersimpan walaupun perangkat dimatikan dan dinyalakan kembali. Tugas yang ada dapat diperlihatkan sesuai dengan *filter* jenis tugas dan tanggal tugas, dan yang ditampilkan hanya tugas yang belum ditandai selesai. Terdapat juga opsi *help* dan pesan *error* saat pesan tidak sesuai dengan kata apa pun pada *database*. Jadi, program yang dibuat sudah dapat memenuhi semua kriteria fungsionalitasnya.

## **V. Kesimpulan, Saran, Refleksi, dan Komentar**

### **V.I. Kesimpulan**

*String matching* dapat dilakukan dengan menggunakan berbagai macam algoritma, di antaranya adalah KMP dan Boyer-Moore. Regex dapat digunakan untuk mencari kumpulan kata atau karakter yang memiliki pola, misalnya kode mata kuliah. Kemiripan antara dua kata dapat diukur dengan berbagai matriks, salah satunya adalah Levenshtein *distance*.

### **V.II. Saran**

Penggunaan *string matching* dengan KMP maupun Boyer-Moore dapat digunakan untuk pencocokan pola yang pasti, sedangkan regex dapat digunakan untuk mencari pola yang memiliki bentuk tertentu.

### **V.III. Refleksi**

Versi dari bahasa pemrograman yang digunakan (dalam hal ini NodeJS) harus diperhatikan, karena versi yang berbeda dapat memiliki fitur yang berbeda juga.

## **Daftar Pustaka**

Khodra, M. L. (2019). *String Matching dengan Regular Expression*. Homepage Rinaldi Munir.

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

Knuth, D. E., Morris, J. H., & Pratt, V. R. (1974). *FAST PATTERN MATCHING IN STRINGS\**.

Munir, R. (2021). *Pencocokan String*. Homepage Rinaldi Munir.

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

Stubblebine, T. (2003). *Regular Expression Pocket Reference*. O'Reilly Media, Inc.

[https://books.google.co.id/books?id=yWiAPqBKuqYC&dq=regular+expression&lr=&source=gbs\\_navlinks\\_s](https://books.google.co.id/books?id=yWiAPqBKuqYC&dq=regular+expression&lr=&source=gbs_navlinks_s)

Chatcompose. <https://www.chatcompose.com/what-are-chatbots.html>. Diakses 27 April 2021 pukul 21.21.