TUGAS BESAR II IF2123
ALJABAR LINEAR DAN GEOMETRI
2020/2021

Oleh

Muhammad Dzaki Razaan Faza	13519033
Joel Triwira	13519073
Rizky Anggita Syarbaini Siregar	13519132

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2020

BAB I
Deskripsi Masalah

Hampir semua orang pernah menggunakan *search engine*, seperti *google*, *bing* dan *yahoo! search*. Setiap hari, bahkan untuk sesuatu yang sederhana orang-orang menggunakan mesin pencarian. Pada tugas besar Aljabar Linear dan Geometri kali ini penulis akan menjelaskan bagaimana cara search engine tersebut mendapatkan semua dokumen kita berdasarkan apa yang ingin pengguna cari.

Sebagaimana yang telah diajarkan di dalam kuliah pada materi vektor di ruang Euclidean, temu-balik informasi (*information retrieval*) merupakan proses menemukan kembali (*retrieval*) informasi yang relevan terhadap kebutuhan pengguna dari suatu kumpulan informasi secara otomatis. Biasanya, sistem temu balik informasi ini digunakan untuk mencari informasi pada informasi yang tidak terstruktur, seperti laman *web* atau dokumen.

Ide utama dari sistem temu balik informasi adalah mengubah *search query* menjadi ruang vektor. Setiap dokumen maupun *query* dinyatakan sebagai vektor $w = (w_1, w_2,..., w_n)$ di dalam R^n, dimana nilai w_i dapat menyatakan jumlah kemunculan kata tersebut dalam dokumen (*term frequency*). Penentuan dokumen mana yang relevan dengan *search query* dipandang sebagai pengukuran kesamaan (*similarity measure*) antara *query* dengan dokumen. Semakin sama suatu vektor dokumen dengan vektor *query*, semakin relevan dokumen tersebut dengan *query*. Kesamaan tersebut dapat diukur dengan *cosine similarity* dengan rumus:

$$sim(\mathbf{Q}, \mathbf{D}) = \cos\theta = \frac{\mathbf{Q} \cdot \mathbf{D}}{\|\mathbf{Q}\|\|\mathbf{D}\|}$$

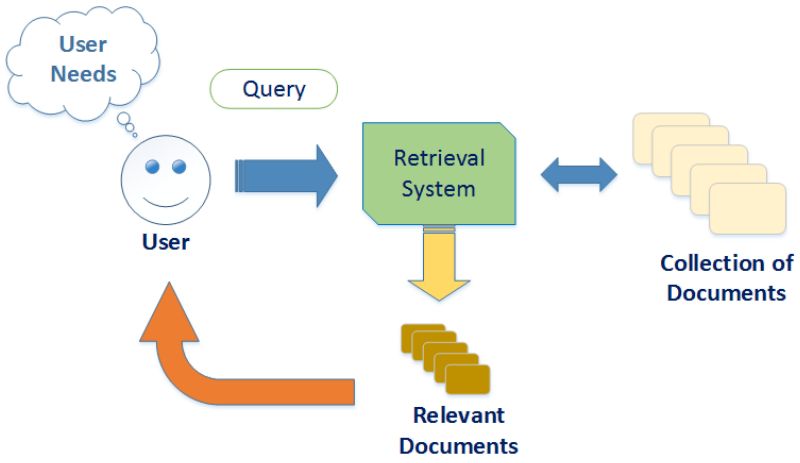
Pada tugas besar kali ini dengan menggunakan model ruang vektor dan *cosine similarity* penulis membuat sebuah *search engine* sederhana.

BAB II
Teori Singkat

2.1 *Information Retrieval*

Sistem *information retrieval* (IR) system adalah sistem yang digunakan untuk menemukan kembali (*retrieve*) informasi-informasi yang relevan terhadap kebutuhan pengguna dari suatu kumpulan informasi secara otomatis [Bunyamin, 2005].

Sistem IR terutama berhubungan dengan pencarian informasi yang isinya tidak memiliki struktur. Demikian pula ekspresi kebutuhan pengguna yang disebut *query*, juga tidak memiliki struktur. Hal ini yang membedakan sistem IR dengan sistem basis data. Dokumen adalah contoh informasi yang tidak terstruktur. Isi dari suatu dokumen sangat tergantung pada pembuat dokumen tersebut.



Gambar 2.1 Gambaran Information Retrieval
Sumber: https://ir.cs.ui.ac.id/new/images/ir.png

Sebagai suatu sistem, sistem IR memiliki beberapa bagian yang membangun sistem secara keseluruhan.Berikut adalah bagian-bagian dari sistem IR:

1. *Text Operations* (operasi terhadap teks) yang meliputi pemilihan kata-kata dalam *query* maupun dokumen (*term selection*) dalam pentransformasian dokumen atau *query* menjadi term *index* (indeks dari kata-kata).

2. *Query formulation* (formulasi terhadap *query*) yaitu memberi bobot pada indeks kata-kata *query*.

3. *Ranking* (perangkingan), mencari dokumen-dokumen yang relevan terhadap *query* dan mengurutkan dokumen tersebut berdasarkan kesesuaiannya dengan *query*.

Sistem IR menerima *query* dari pengguna, kemudian melakukan perangkingan terhadap dokumen pada koleksi berdasarkan kesesuaiannya dengan query. Hasil perangkingan yang diberikan kepada pengguna merupakan dokumen yang menurut sistem relevan dengan query.

2.2 Vektor

Vektor adalah kuantitas fisik yang memiliki besar dan arah. Salah satu model IR adalah model menggunakan ruang vektor. Model ini menggunakan teori di dalam aljabar vektor. Misalkan terdapat n kata berbeda sebagai kamus kata (*vocabulary*) atau indeks kata (*term index*). Kata-kata tersebut membentuk ruang vektor berdimensi n. Setiap dokumen maupun query dinyatakan sebagai vektor $w = (w_1, w_2, ..., w_n)$ di dalam R^n. w_i = bobot setiap kata i di dalam query atau dokumen. Nilai w_i dapat menyatakan jumlah kemunculan kata tersebut dalam dokumen (*term frequency*).

Ada beberapa langkah atau proses untuk mendapatkan hasil dari *query* yang dimasukkan dengan menggunakan model ruang vektor:

1. Membuang kata depan dan kata penghubung.
2. Menggunakan *stemmer* pada kumpulan dokumen dan query, yaitu aplikasi yang digunakan untuk menghilangkan imbuhan (awalan, akhiran).
3. Menghitung banyaknya frekuensi atau kemunculan kata dalam kumpulan dokumen yang sesuai dengan query.
4. Mengubah dokumen dan query menjadi vektor
5. Mengurutkan dokumen berdasarkan similaritas, dengan menghitung perkalian titik vektor (*cosine similarity*).

2.3 *Cosine similarity*

Penentuan dokumen mana yang relevan dengan *query* dipandang sebagai pengukuran kesamaan (*similarity measure*) antara *query* dengan dokumen. Semakin sama suatu vektor dokumen dengan vektor *query*, semakin relevan dokumen tersebut dengan

query. Kesamaan (*sim*) antara dua vektor Q = (q_1, q_2, …, q_n) dan D = (d_1, d_2, …, d_n) diukur dengan rumus *cosine similarit*y yang merupakan bagian dari rumus perkalian titik (*dot product*) dua buah vektor:

$$sim(\mathbf{Q}, \mathbf{D}) = \cos\theta = \frac{\mathbf{Q} \cdot \mathbf{D}}{\|\mathbf{Q}\| \|\mathbf{D}\|}$$

dengan Q•D adalah perkalian titik yang didefinisikan sebagai:

$$\mathbf{Q} \cdot \mathbf{D} = q_1 d_1 + q_2 d_2 + ... + q_n d_n$$

Jika cos q = 1, berarti q = 0, vektor Q dan D berimpit, yang berarti dokumen D sesuai dengan *query* Q. Jadi, nilai cosinus yang besar (mendekati 1) mengindikasikan bahwa dokumen cenderung sesuai dengan *query*. Setiap dokumen di dalam koleksi dokumen dihitung kesamaannya dengan query dengan rumus cosinus di atas.

Selanjutnya hasil perhitungan diurutkan berdasarkan nilai cosinus dari besar ke kecil sebagai proses pemilihan dokumen yang yang “dekat” dengan *query*. Pengurutan tersebut menyatakan dokumen yang paling relevan hingga yang kurang relevan dengan *query*. Nilai cosinus yang besar menyatakan dokumen yang relevan, nilai cosinus yang kecil menyatakan dokumen yang kurang relevan dengan *query*.

BAB III
Implementasi Program

Program yang dibuat menggunakan bahasa *Python* dan menggunakan framework *Flask* untuk *backend* nya. Pada bagian *frontend*, kami menggunakan *HTML*, *CSS*, dan *Bootstrap 4*.

Terdapat 4 buah program modular (*backend*), 3 buah file *HTML*, dan 1 buah file *CSS* yang kami buat pada Tugas Besar II ini.

1. *App.py*

   Kami menggunakan *Flask* sebagai *framework* untuk membuat *web* aplikasi ini. *Library* yang digunakan pada *app.py* adalah *flask*, *werkzeug*, *os*, dan *shutil*. Berikut adalah penjelasan mengenai fungsi yang terdapat pada *app.py.*

Tabel 3.1 *App.py*

Fungsi/Prosedur	Keterangan
home	Menampilkan *home.html.*
show	Menampilkan *file txt* yang di-*input.*
search	Menerima *file .txt* dan *query* yang ingin dicari, kemudian menampilkan hasil pencarian. *Method = GET, POST*
allowed_files	Mengecek apakah *file* yang diinput memiliki format yang benar (.*txt*).
request_txt	Menerima *multiple files input* dari *user.*

2. *Text.py*

   Kumpulan fungsi yang melakukan *text-processing* pada dokumen yang di-input oleh pengguna. Dokumen yang diinput adalah berbahasa Indonesia. Kami menggunakan *library Sastrawi* untuk melakukan beberapa proses yaitu *filtering stopword* dan *stemming*. Berikut beberapa fungsi yang terdapat di *text.py*.

Tabel 3.2 *Text.py*

Fungsi/Prosedur	Keterangan
stemming_doc	Menyiapkan objek *Stemmer* dan melakukan *stemming* pada dokumen/*query.*
filtering_stopword	Menyiapkan objek *StopWordRemover* dan melakukan *filtering stopword* pada dokumen/*query.*
stemming_filtering_doc	Melakukan *filtering stopword* dan *stemming* pada kumpulan dokumen dengan menggunakan fungsi *stemming_doc* dan *filtering_stopword* di atas secara sekaligus.
stemming_filtering_query	Melakukan *filtering stopword* dan *stemming* pada *query* dengan menggunakan fungsi *stemming_doc* dan *filtering_stopword* di atas secara sekaligus.
first_sentence	Melakukan pengambilan kalimat pertama dari setiap dokumen yang di-*input user.*

3. *Docs.py*

   Kumpulan fungsi yang melakukan proses lebih detail terhadap dokumen yang di-*input* oleh pengguna. Berikut fungsi-fungsi yang terdapat pada *docs.py.*

Tabel 3.3 *Docs.py*

Fungsi	Keterangan
open_doc	Melakukan *opening file* dan mengembalikan *txt* dari *file* tersebut.
count_kata_doc	Menghitung jumlah kata pada kumpulan dokumen sebelum dilakukan *text-processing* apapun pada kumpulan dokumen tersebut.
count_word	Menghitung jumlah kata per dokumen

remove_duplicate	Menghilangkan duplikasi pada kumpulan kata (*term*) dari setiap dokumen dan *query.*
makeTerm	Membentuk *term* dari kumpulan kata yang berasal dari dokumen dan *query.*

4. Similarity.py

   Kumpulan fungsi yang melakukan proses terhadap perhitungan kemiripan (similarity) antara query dan dokumen. Terdapat pustaka(library) *math* yang digunakan untuk menghitung akar pangkat 2 (square root). Berikut fungsi-fungsi yang terdapat pada similarity.py

Fungsi	Keterangan
dot_product	Menghitung hasil kali titik (dot product) dari dua buah vektor.
besar	Menghitung norm(besar/panjang) dari sebuah vektor.
sim	Menghitung cosine similarity pada query dan dokumen dengan menggunakan fungsi *dot_product* dan *besar.*
count_found_term	Menghitung kemunculan term pada pada dokumen dan query dan mengembalikan sebuah array yaitu *arrayHasil*.
sortHasil	Melakukan pengurutan(sorting) pada beberapa array. Pengurutan berdasarkan *arrayHasil* yang merupakan hasil perhitungan dari *count_found_term*.

5. *Layout.html*

   Merupakan *template* yang dibuat untuk di-*load* pada file *html* lainnya.

6. *Index.html*

   Merupakan *file html* yang digunakan pada saat pengguna mengklik *Search.*

7. *Search.html*

   Merupakan *file html* yang digunakan pada saat hasil pencarian ditampilkan.

8. *View.html*

   Merupakan *file html* yang digunakan pada saat menampilkan isi dari *file txt* yang di-*upload* oleh pengguna.

9. *Home.html*

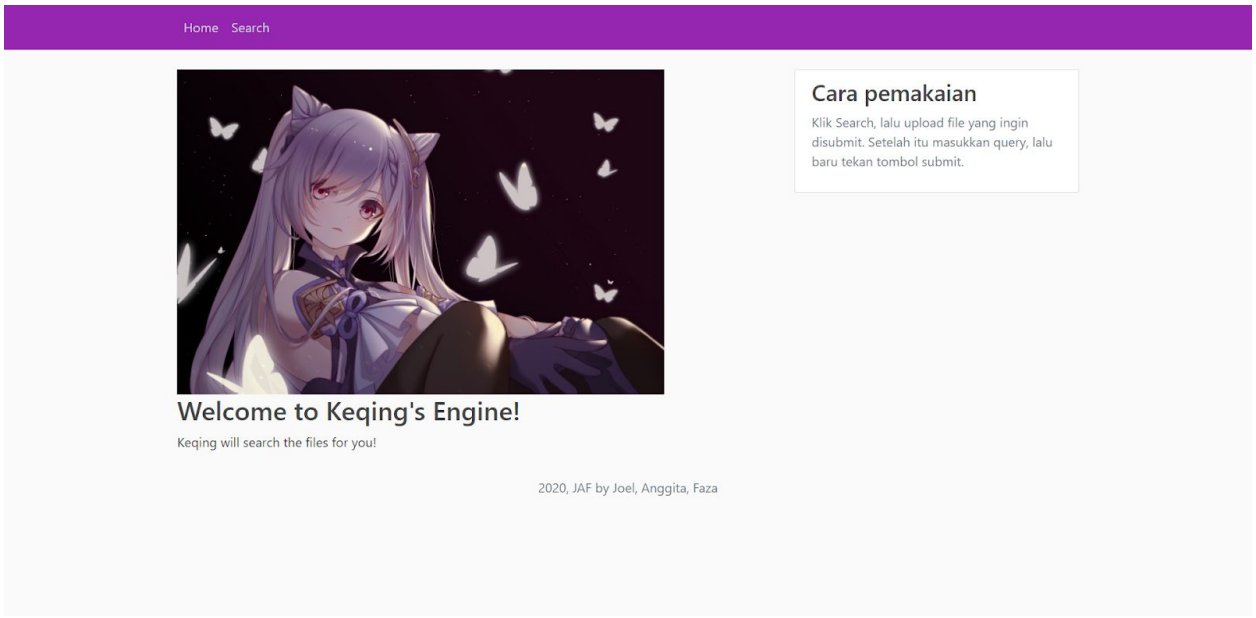
   Merupakan *file html* yang digunakan untuk *landing page*.

10. *Main.css*

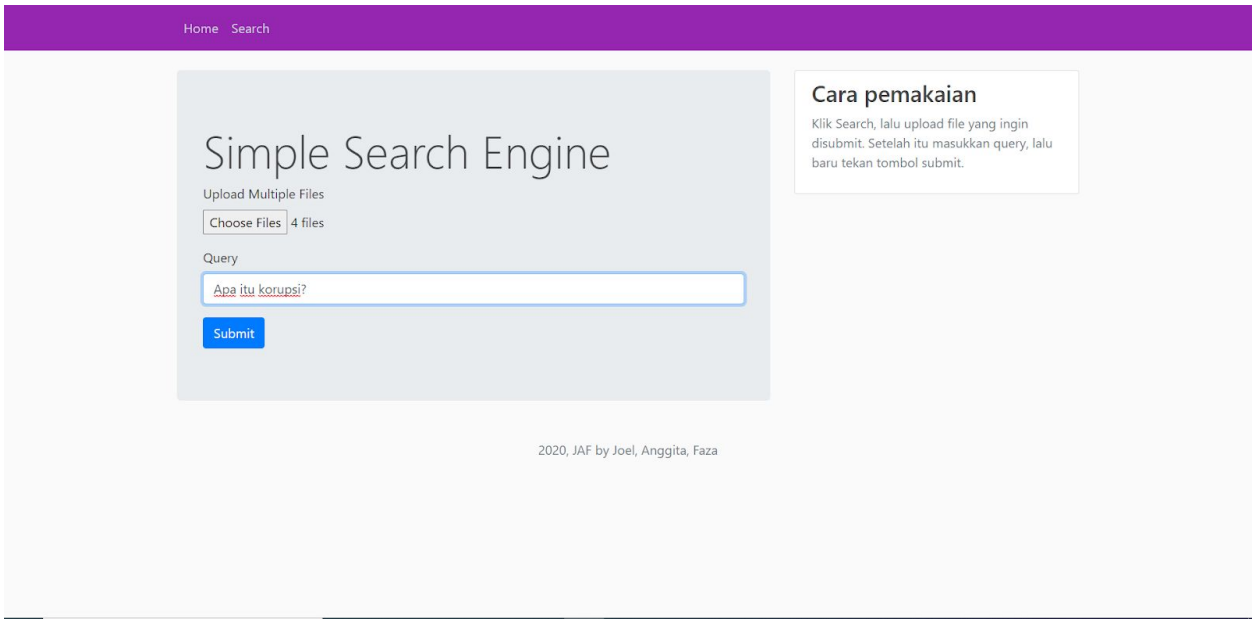
    Merupakan *file CSS* yang digunakan untuk *styling* pada tampilan *web* ini.

BAB IV

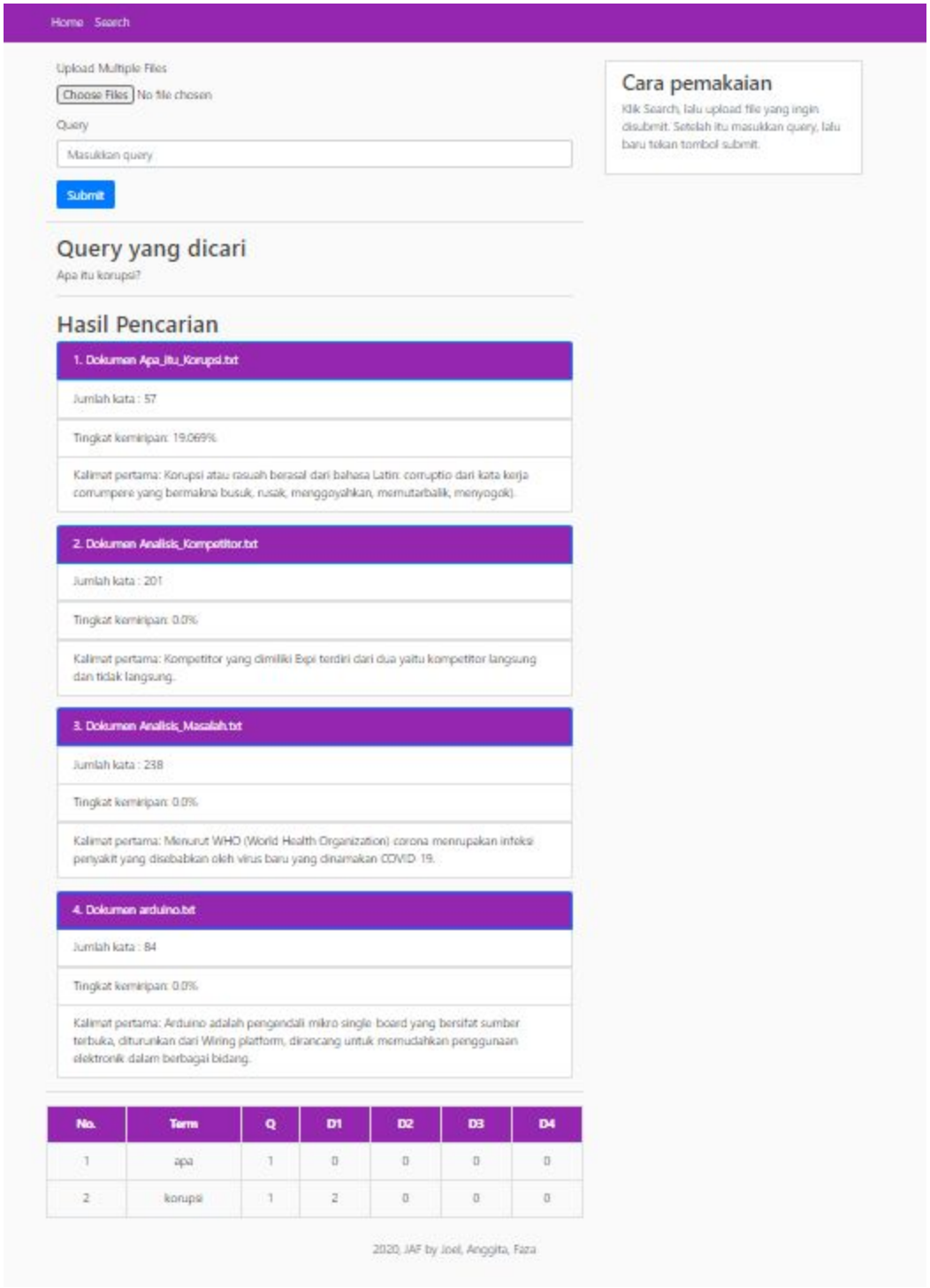
Eksperimen



Gambar 4.1 Halaman Depan *Website*



Gambar 4.2 Tampilan Input Dokumen dan Query dari Pengguna



Gambar 4.3 Hasil Pencarian *Terms Query* pada Dokumen

BAB V
Simpulan

5.1 Simpulan

Search engine sederhana dapat dibuat dengan menerapkan *cosine similarity* yang memanfaatkan model ruang vektor. *Query* dan dokumen yang didapat dari pengguna dapat diolah menggunakan *stemming* dan penghapusan *stopword* lalu dimasukkan ke dalam sebuah array yang sudah menampung *term* yang bersih. Selanjutnya, kemunculan *term* di *query* dalam dokumen dapat dihitung sehingga didapat vektor yang dapat dimanfaatkan untuk mencari *cosine similarity* dari *query* dan dokumen. Lalu, dokumen diurutkan berdasarkan nilai kemiripannya dengan *query.* Dengan bantuan *framework flask* dan sedikit *html* serta *css, search engine* dapat diaplikasikan dalam bentuk *website* lokal sederhana.

5.2 Saran

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

1. Memperindah desain *website* supaya lebih menarik bagi pengguna. Beberapa diantaranya adalah desain tombol, *typography, color pallete,* dan membuat tampilan yang responsif.
2. Membuat kode lebih rapi dan memberikan komentar yang lebih detail serta mudah dipahami.
3. Memberikan konvensi dalam menentukan standar nama fungsi atau variabel.
4. Menerapkan *web scraping* agar bisa menerima input *html.*

5.3 Refleksi

Melalui tugas ini, penulis belajar untuk bekerja sama dalam kelompok, tidak menunda - nunda pekerjaan, berbagi tugas, dan juga membuat sebuah *website* baik dari segi *frontend* maupun *backend* menggunakan *flask.*

Daftar Pustaka

Bunyamin, H., & Negara, C. P. (2008). Aplikasi Information Retrieval (IR) CATA Dengan Metode Generalized Vector Space Model. *Jurnal Informatika*, *4*(1), 29-38. https://media.neliti.com/media/publications/219301-none.pdf

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

Munir, R. (2020). Aplikasi Dot Product pada Sistem Temu-balik Informasi (Information Retrieval System). In *Bahan kuliah IF2123 Aljabar Linier dan Geometri* (pp. 8-14). Program Studi Teknik Informatika, STEI-ITB.

Schafer, C. (2018, May Friday). *Python Flask Tutorial: Full-Featured Web App*. Retrieved November, 2020, from https://www.youtube.com/watch?v=QnDWIZuWYW0&list=PL-osiE80TeTs4UjLw5MM6OjgkjFeUxCYH&index=2&ab_channel=CoreySchafer

Visual Studio Code. (n.d.). *Flask Tutorial in Visual Studio Code*. https://code.visualstudio.com/docs/python/tutorial-flask