def user\_recommendation():

    if request.method == 'POST':

        # Mendapatkan input dari formulir

        personality = request.form['personality']

        listbhspemrograman = request.form.getlist('bahasapemrograman')

        role = request.form['role']

        project\_name = request.form['projek']

        project\_description = request.form['deskripsi']

        # Gabungkan semua bahasa pemrograman menjadi satu teks

        bhspemrograman = ' '.join(listbhspemrograman)

        # Gabungkan semua kriteria menjadi satu teks

        input\_text = ' '.join([personality, bhspemrograman, role, project\_name, project\_description])

        # Dapatkan data mhs\_data dari database

        mhs\_data = get\_data\_cleaned()

        # Lakukan preprocessing pada teks input

        preprocessed\_text = preprocessing\_input(input\_text)

        # Hitung TF, IDF, dan W = TF \* (IDF + 1) menggunakan function dari file

        vectorizer = TfidfVectorizer()

        tfidf\_matrix = vectorizer.fit\_transform([preprocessed\_text, \*mhs\_data['metadata']])

        # Hitung cosine similarity

        cosine\_similarities = cosine\_similarity(tfidf\_matrix[0:1], tfidf\_matrix[1:])

        # Dapatkan indeks proyek yang memiliki nilai cosine similarity di atas 0.6

        similar\_projects\_indices = np.where(cosine\_similarities > 0.6)[1]

        # Ambil data yang memiliki nilai cosine similarity di atas 0.6

        recommended\_projects = mhs\_data.iloc[similar\_projects\_indices]

        # Menampilkan hasil rekomendasi

        return render\_template('user-recommendation.html', projects=recommended\_projects.to\_dict(orient='records'))

def get\_data\_cleaned():

        cur = mysql.connection.cursor()

        cur.execute("SELECT metadata FROM mhs\_data")  # Memilih hanya kolom metadata

        projects = cur.fetchall()

        cur.close()

        df = pd.DataFrame(projects, columns=['metadata'])  # Menentukan kolom 'metadata' untuk DataFrame

        print("Data fetched successfully")

        return df

TFIDF PERKATA  
import pandas as pd

import numpy as np

from sklearn.feature\_extraction.text import TfidfVectorizer

import streamlit as st

# Memuat data

data = pd.read\_csv('tfidf.csv', encoding='utf-8', delimiter=';', dtype={'NIM': str})

data = data[['Nama','NIM','personality', 'Bahasa pemrograman', 'Komunitas', 'Role', 'Proyek', 'Deskripsi', 'metadata']]

# Menghapus baris dengan nilai yang hilang di kolom 'metadata'

data = data.dropna(subset=['metadata'])

# Fungsi untuk menghitung TF, IDF, dan W

def calculate\_tfidf\_per\_word(data):

    # Inisialisasi DataFrame kosong untuk menyimpan hasil perhitungan

    result\_df = pd.DataFrame(columns=['Dokumen', 'Term', 'TF', 'IDF', 'W'])

    # Loop melalui setiap dokumen

    for i, doc in enumerate(data['metadata']):

        # Inisialisasi vectorizer

        vectorizer = TfidfVectorizer(tokenizer=lambda x: x.split())

        # Hitung TF-IDF untuk dokumen saat ini

        tfidf\_matrix = vectorizer.fit\_transform([doc])

        terms = vectorizer.get\_feature\_names\_out()

        tf\_values = tfidf\_matrix.toarray().flatten()

        # Hitung IDF

        idf\_values = vectorizer.idf\_

        # Hitung W

        w\_values = tf\_values \* idf\_values

        # Menyimpan hasil perhitungan TF, IDF, dan W per term

        doc\_result = pd.DataFrame({

            'Dokumen': [data.iloc[i]['Nama']] \* len(terms),

            'Term': terms,

            'TF': tf\_values,

            'IDF': idf\_values,

            'W': w\_values

        })

        # Gabungkan hasil perhitungan untuk dokumen saat ini ke dalam DataFrame utama

        result\_df = pd.concat([result\_df, doc\_result])

    return result\_df

# Menampilkan hasil perhitungan TF, IDF, dan W

st.write("Hasil perhitungan TF, IDF, dan W per kata")

st.write(calculate\_tfidf\_per\_word(data))

TFIDF