LAPORAN TUGAS RegEx
WEB APP 'TSD.KEPO'

Kelompok E

Fransiscus Ernest O.	162112133081
Farid Ardhan	162112133084
Najla Dhia Rusydi	164221043
Ramadhan Eko Saputra	164221088

MATA KULIAH NATURAL LANGUAGE PROCESSING

PROGRAM STUDI S1 TEKNOLOGI SAINS DATA

FAKULTAS TEKNOLOGI MAJU DAN MULTIDISIPLIN

UNIVERSITAS AIRLANGGA

2024

DAFTAR ISI

OUTLINE PROJECT

1. **Judul Proyek**:
   TSD.Kepo – Web Aplikasi Pencarian Data Mahasiswa dengan Autentikasi dan Validasi Email.
2. **Deskripsi Proyek**:
   Aplikasi web ini dirancang untuk memungkinkan pengguna melakukan pendaftaran (signup), login, dan pencarian data mahasiswa. Aplikasi ini mengimplementasikan validasi email menggunakan RegEx agar email yang digunakan mengikuti format khusus. Selain itu, web app ini menggunakan SQLite untuk menyimpan data pengguna dan data mahasiswa.
3. **Fitur Utama:**
   1) **Autentikasi Pengguna**:
      a) **Signup**: Pengguna bisa mendaftar dengan username, email, dan password yang akan diverifikasi.
      b) **Login**: Pengguna bisa masuk ke sistem setelah melakukan pendaftaran.
      c) **Logout**: Pengguna dapat keluar dari sesi aplikasi.
   2) **Validasi Email dengan RegEx**:
      a) Email yang digunakan untuk mendaftar harus mengikuti format tertentu:
         i) **Format yang didukung**: nama.nama.nama-angkatan@fakultas.unair.ac.id
         ii) Bagian fakultas harus terdiri dari 2 hingga 5 huruf.
         iii) Bagian nama bisa terdiri dari satu, dua, atau tiga bagian dipisahkan oleh titik.
   3) **Pencarian Data Mahasiswa**:
      a) Setelah login, pengguna dapat mencari informasi tentang mahasiswa berdasarkan nama atau NIM.
   4) **Keamanan Password**:
      a) Password harus memenuhi kriteria tertentu (menggunakan RegEx): minimal 8 karakter, harus mengandung huruf dan angka.
      b) Password yang disimpan dalam database dienkripsi menggunakan metode hashing pbkdf2:sha256.
   5) **Penyimpanan dan Manajemen Data**:
      a) SQLite digunakan sebagai database lokal untuk menyimpan data pengguna dan data mahasiswa.
4. **Penggunaan Regex dalam Proyek**:
   1) **Validasi Email**: Email harus mengikuti format khusus.

      ```
      ^[a-zA-Z]+(\.[a-zA-Z]+){0,2}-[0-9]{4}@[a-zA-Z]{2,5}\.unair\.ac\.id$
      ```

   2) **Validasi Password**: Password divalidasi menggunakan RegEx untuk memastikan memenuhi standar keamanan.

      ```
      ^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{8,}$
      ```

3) **Pencarian Data** : Memungkinkan pencarian data mahasiswa yang lebih fleksibel dan powerful dengan menggunakan pola pencarian berbasis RegEx.

```
pattern = re.compile(search_query, re.IGNORECASE)
```

ARSITEKTUR PROGRAM

1. **Frontend**
    1) **HTML Templates** : Web ini menggunakan template HTML (index.html, index1.html, signup.html, signin.html, login.html, dan search.html) untuk menampilkan antarmuka pengguna (UI).
    2) **Form Input** : User memasukkan data seperti username, email, password, dan query pencarian melalui form HTML lalu diolah oleh Flask menggunakan metode POST.
    3) **Flash Messages** : Menggunakan mekanisme flash dari Flask untuk memberikan umpan balik kepada user, seperti notifikasi keberhasilan registrasi, kesalahan validasi, atau pesan logout.

2. **Backend**
    1) **Flask Framework** : Mengani routing dan logika. Setiap route didefinisikan menggunakan decorator seperti **@app.route('/')** untuk mendefinisikan URL yang tersedia dalam aplikasi.
    2) **Autentikasi dan Registrasi Pengguna** :
        - SignUp : Pengguna baru mendaftarkan username, email, dan password. Email divalidasi menggunakan regex untuk memastikan format yang digunakan benar. Password kemudian di-hash menggunakan metode pbkdf2:sha256.
        - Login Pengguna : User yang terdaftar dapat login menggunakan email dan password mereka. Password yang digunakan dibandingkan dengan password hash yang telah disimpan di database menggunakan *check_password_hash*
    3) **Pencarian Data Mahasiswa** : Setelah proses login, pengguna bisa mencari data mahasiswa berdasarkan nama atau NIM. Hasil pencarian di filter menggunakan regex, memungkinkan pencarian yang fleksibel berdasarkan pola yang diinginkan.

3. **Database**
    1) **SQLite Database :** Aplikasi ini menggunakan SQLite sebagai database.
    2) **Inisialisasi Database** : Fungsi *init_sqlite-db()* membuat dua tabel, tabel *users* untuk menyimpan informasi pengguna (username, email, password hash) dan tabel *students* untuk menyimpan data mahasiswa (nama, NIM).

3) **CRUD** :
    - **Create** : Mendaftarkan pengguna baru (signup) dan menyimpan data pengguna ke tabel *users*.
    - **Read :** Mengambil data pengguna untuk login dan mengambil data mahasiswa untuk pencarian.

4. **Session Management**
    1) Aplikasi ini menggunakan sesi (session) untuk mengelola status login pengguna. Saat login berhasil, informasi login disimpan dalam session, sehingga pengguna tetap dapat mengakses halaman-halaman tertentu tanpa harus login kembali.
    2) Fungsi *session.clear()* digunakan untuk menghapus semua data sesi ketika logout
5. **Keamanan**
    1) **Hashing Password :** Password yang disimpan di database di-hash menggunakan *werkzeug.security.generate_password_hash* dengan algoritma pbkdf2:sha256.
    2) **Validasi Input** : Input dari pengguna, seperti email dan password divalidasi menggunakan regex untuk memastikan format sesuai sebelum disimpan ke dalam database.
6. **Routing dan Alur Kerja**
    1) **(Home)** : Menampilkan halaman utama yang berbeda antara yang sudah login dan belum.
    2) **(SignUp)** : Menangani registrasi pengguna baru.
    3) **(Login)** : Menangani autentikasi pengguna.
    4) **(Search)** : Mengizinkan pengguna yang sudah login untuk mencari data mahasiswa berdasarkan nama atau NIM menggunakan regex.
    5) **(Logout)** : Menghapus sesi pengguna dan mengembalikan mereka ke halaman utama.
7. **Error Handling dan Feedback**
    1) Penggunaan flash untuk menampilkan kesalahan dalam mengisi email yang tidak valid, password yang tidak sesuai kriteria, atau email sudah terdaftar.
    2) Menyediakan validasi di server-side untuk memastikan keamanan dan integritas data.
8. **Alur Kerja Pengguna**
    1) **Registrasi** : Pengguna mengunjungi halaman signup, mengisi form, dan melakukan validasi.
    2) **Login** : Setelah berhasil registrasi, pengguna login dan masuk ke halaman pencarian.
    3) **Pencarian Data** : Setelah login, pengguna bisa melakukan pencarian data mahasiswa menggunakan pola regex yang diinginkan.
    4) **Logout** : Pengguna logout dari aplikasi dan kembali ke halaman utama.

PENJELASAN ALGORITMA

```
from flask import Flask, render_template, request, redirect, url_for, flash, session
import re
import sqlite3
from werkzeug.security import generate_password_hash, check_password_hash

app = Flask(__name__)
app.secret_key = 'your_secret_key'

# Fungsi untuk menginisialisasi database
def init_sqlite_db():
    conn = sqlite3.connect('users.db')
    print("Opened database successfully")

    # Tabel pengguna untuk autentikasi
    conn.execute('CREATE TABLE IF NOT EXISTS users (id INTEGER PRIMARY KEY
AUTOINCREMENT, username TEXT, email TEXT, password TEXT)')

    # Tabel daftar nama anak, NIM, dan Angkatan
    conn.execute('CREATE TABLE IF NOT EXISTS students (id INTEGER PRIMARY KEY
AUTOINCREMENT, name TEXT, nim TEXT, batch TEXT)')

    print("Tables created successfully")
    conn.close()

# Inisialisasi database
init_sqlite_db()
```

Deskripsi Implementasi Aplikasi Web Menggunakan Flask dan SQLite

1. **Inisialisasi Aplikasi Flask**
   - ➢ **Flask App**: Flask adalah framework web yang ringan dan fleksibel untuk Python, yang memungkinkan pengembangan aplikasi web dengan mudah dan cepat. Pernyataan app = *Flask(__name__)* menciptakan instance aplikasi Flask, di mana __name__ adalah variabel yang menunjukkan nama modul Python. Hal ini penting untuk Flask mengetahui di mana harus mencari resource seperti template dan file statis.
2. **Konfigurasi Keamanan**
   - ➢ **Secret Key**: Key rahasia *app.secret_key* = '*your_secret_key*' digunakan oleh Flask untuk menandatangani data sesi, yang sangat penting untuk menjaga keamanan sesi dan pesan flash. Key ini harus dijaga kerahasiaannya dan harus cukup kompleks untuk mencegah peretasan.
3. **Database SQLite**
   - ➢ **Penggunaan SQLite:** Aplikasi ini menggunakan SQLite, sebuah sistem manajemen database yang ringkas dan yang tidak memerlukan konfigurasi server terpisah. SQLite

menyimpan seluruh database dalam satu file fisik, dalam kasus ini users.db, yang memudahkan pengelolaan dan distribusi database.

- ➢ **Fungsi Inisialisasi Database:** Fungsi *init_sqlite_db()* bertanggung jawab untuk membuat koneksi ke database SQLite dan inisialisasi tabel yang diperlukan.
  - Tabel users: Menyimpan data pengguna termasuk id, username, email, dan password yang di-hash. Ini mendukung proses autentikasi pengguna.
  - Tabel students: Menyimpan informasi mahasiswa seperti nama, NIM, dan angkatan. Hal ini memfasilitasi penyimpanan dan pengambilan informasi terkait mahasiswa.

4. **Keamanan Password**

- ➢ **Hashing Password:** Menggunakan fungsi *generate_password_hash* dan *check_password_hash* dari modul werkzeug.security untuk mengamankan password. Password di-hash sebelum disimpan ke dalam database dan verifikasi dilakukan melalui proses hashing selama login. Hal ini memastikan bahwa password asli tidak disimpan atau ditransmisikan, memberikan lapisan keamanan tambahan terhadap pencurian identitas.

Kesimpulan

Aplikasi ini memanfaatkan Flask untuk membangun interface yang responsif dan mudah digunakan, serta SQLite untuk manajemen data yang efisien dan ringkas. Kombinasi kedua teknologi ini menyediakan platform yang aman dan skalabel untuk aplikasi web, dengan penerapan praktik keamanan yang baik dalam pengelolaan sesi dan autentikasi pengguna.

```
# Route halaman home
@app.route('/')
def home():
    if session.get('logged_in'):
        return render_template('index1.html')
    else:
        return render_template('index.html')

# Route halaman signup
@app.route('/signup', methods=['GET', 'POST'])
def signup():
    if request.method == 'POST':
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']

        # Validasi email dan password menggunakan RegEx yang baru
        if not re.match(r'^[a-zA-Z]+(\.[a-zA-Z]+){0,2}-[0-9]{4}@[a-zA-Z]{2,5}\.unair\.ac\.id$', email):
            flash('Email tidak valid! Pastikan menggunakan email UNAIR', 'danger')
            return redirect(url_for('signup'))
```

```
        if not re.match(r'^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{8,}$', password):
            flash('Password harus minimal 8 karakter, mengandung huruf dan angka!', 'danger')
            return redirect(url_for('signup'))

        # Menggunakan metode hash yang benar 'pbkdf2:sha256'
        hashed_password = generate_password_hash(password, method='pbkdf2:sha256')

        with sqlite3.connect('users.db') as con:
            cur = con.cursor()
            # Memeriksa apakah email sudah terdaftar
            cur.execute("SELECT * FROM users WHERE email = ?", (email,))
            existing_user = cur.fetchone()
            if existing_user:
                flash('Email sudah terdaftar, silakan gunakan email lain atau login.', 'warning')
                return redirect(url_for('signup'))

            cur.execute("INSERT INTO users (username, email, password) VALUES (?, ?, ?)", (username,
email, hashed_password))
            con.commit()
            flash('Registrasi berhasil! Silakan login.', 'success')
            return redirect(url_for('login'))

    return render_template('signup.html')

# Route halaman login
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']

        with sqlite3.connect('users.db') as con:
            cur = con.cursor()
            cur.execute("SELECT * FROM users WHERE email = ?", (email,))
            user = cur.fetchone()
            if user and check_password_hash(user[3], password):
                session['logged_in'] = True
                session['username'] = user[1]
                flash('Login berhasil!', 'success')
                return redirect(url_for('search'))  # Redirect to the search page
            else:
                flash('Email atau password salah!', 'danger')

    return render_template('login.html')
```

Implementasi Aplikasi Web Menggunakan Flask untuk Autentikasi Pengguna

1. **Route Halaman Beranda (/)**

- **Deskripsi Fungsi:**
  - ➢ Fungsi home bertugas menampilkan halaman utama aplikasi. Halaman yang ditampilkan berbeda tergantung pada status login pengguna.
  - ➢ Jika pengguna sudah masuk (logged_in dalam sesi adalah True), halaman index1.html akan ditampilkan.
  - ➢ Jika pengguna belum masuk, halaman index.html akan ditampilkan sebagai default.

2. **Route Halaman Pendaftaran (/signup)**
   - **Metode yang Didukung: GET dan POST**
   - **Proses**:
     - ➢ **GET:** Mengirimkan halaman signup.html yang berisi formulir pendaftaran kepada pengguna.
     - ➢ **POST:** Memproses data yang dikirimkan pengguna melalui formulir pendaftaran.
       - Ekstraksi Data: Data username, email, dan password diambil dari formulir.
       - Validasi Email: Menggunakan ekspresi reguler (RegEx) untuk memverifikasi bahwa email memenuhi format email UNAIR yang spesifik (nama-nama-nama-2022@fakultas.unair.ac.id). Jika tidak valid, pengguna diarahkan kembali ke halaman pendaftaran dengan pesan kesalahan.
       - Validasi Password: Menggunakan RegEx untuk memastikan password memiliki minimal 8 karakter dan mengandung kombinasi huruf dan angka.
       - Verifikasi Email: Melakukan query ke database untuk memastikan email belum terdaftar. Jika sudah, memberikan opsi kepada pengguna untuk menggunakan email lain atau login.
       - Hashing Password: Menggunakan *generate_password_hash* untuk mengamankan password sebelum disimpan ke database.
       - Penyimpanan: Jika semua validasi berhasil, data pengguna disimpan ke database dan diarahkan ke halaman login dengan pesan berhasil.

3. **Route Halaman Login (/login)**
   - **Metode yang Didukung: GET dan POST**
   - **Proses:**
     - ➢ **GET:** Menampilkan formulir login pada login.html.
     - ➢ **POST:** Mengolah data login yang dikirim oleh pengguna.
       - Pengambilan Data Pengguna: Melakukan query database untuk mencari pengguna dengan email yang diberikan.
       - Verifikasi Password: Membandingkan hash password yang disimpan dengan hash dari password yang dimasukkan menggunakan *check_password_hash*.
       - Manajemen Sesi: Jika kredensial valid, sesi diatur dengan status login dan nama pengguna. Pengguna kemudian diarahkan ke halaman pencarian dengan pesan sukses.
       - Pengelolaan Kegagalan: Jika kredensial tidak cocok, pengguna mendapatkan pesan kesalahan dan diminta untuk mengisi ulang formulir login.

Kesimpulan

Implementasi ini mengilustrasikan penerapan manajemen sesi yang aman dan efektif menggunakan Flask dalam konteks aplikasi web yang mengelola autentikasi pengguna. Penggunaan ekspresi reguler untuk validasi input dan metode hashing yang kuat untuk keamanan password menunjukkan penerapan praktik keamanan yang baik dalam pengembangan aplikasi web. Setiap langkah dijelaskan secara rinci untuk memastikan kejelasan dalam proses autentikasi dan registrasi, memberikan contoh yang baik dari penerapan praktik pengembangan perangkat lunak yang baik.

```
# Route untuk pencarian data siswa setelah login
@app.route('/search', methods=['GET', 'POST'])
def search():
    if 'logged_in' in session and session['logged_in']:
        if request.method == 'POST':
            search_query = request.form['search_query']

            with sqlite3.connect('users.db') as con:
                cur = con.cursor()
                # Mencari semua data siswa
                cur.execute("SELECT * FROM students")
                all_students = cur.fetchall()

            # Filter hasil menggunakan regex
            pattern = re.compile(search_query, re.IGNORECASE)
            results = [student for student in all_students if pattern.search(student[1]) or
pattern.search(student[2])]

            return render_template('search.html', results=results)

        return render_template('search.html', results=None)

    else:
        flash('Anda harus login terlebih dahulu.', 'danger')
        return redirect(url_for('login'))

# Route untuk logout
@app.route('/logout')
def logout():
    session.clear()
    flash('Anda telah logout!', 'success')
    return redirect(url_for('home'))

if __name__ == '__main__':
    app.run(debug=True)
```

Implementasi Fitur Pencarian dan Logout pada Aplikasi Web Flask

1. **Route Pencarian Data Siswa (/search)**
   - ➢ **Metode yang Didukung: GET dan POST**
   - ➢ **Deskripsi Fungsi:**
     - Fungsi search mengelola pencarian data siswa yang hanya dapat diakses oleh pengguna yang telah login.
     - **GET Request:** Menampilkan halaman pencarian (search.html) tanpa hasil pencarian.
     - **POST Request:** Proses permintaan pencarian dari pengguna.
       - Pengambilan dan Penyaringan Data:
       - Koneksi database dibuka dan semua data siswa diambil dari tabel students.
       - Data siswa di filter menggunakan ekspresi reguler yang disesuaikan dengan query pencarian (*search_query*). Regex dicompile dengan opsi *re.IGNORECASE* untuk mengabaikan perbedaan besar/kecil huruf.
       - Siswa yang namanya atau NIM cocok dengan pola pencarian ditampilkan pada hasil.
       - Pengembalian Hasil: Hasil pencarian disajikan kembali ke pengguna melalui *search.html* dengan data hasil yang relevan.
   - ➢ **Pengelolaan Akses:**
     - Fungsi memeriksa status login pengguna. Jika pengguna tidak login, mereka diarahkan ke halaman login dengan pesan peringatan.
2. **Route Logout (/logout)**
   - ➢ **Deskripsi Fungsi:**
     - Fungsi logout bertugas untuk menghapus semua informasi sesi pengguna, efektif mengeluarkan mereka dari aplikasi.
     - Setelah sesi dibersihkan, pengguna diberi notifikasi sukses tentang proses logout dan diarahkan kembali ke halaman beranda (home).
3. **Konfigurasi Server**
   - ➢ **Menjalankan Server:**
     - **if __name__ == '__main__':** app.run(debug=True) memastikan bahwa server hanya dijalankan jika skrip dijalankan sebagai program utama.
     - Mode debug=True memungkinkan server untuk menampilkan informasi debug dan melakukan restart otomatis ketika ada perubahan pada kode.

Kesimpulan

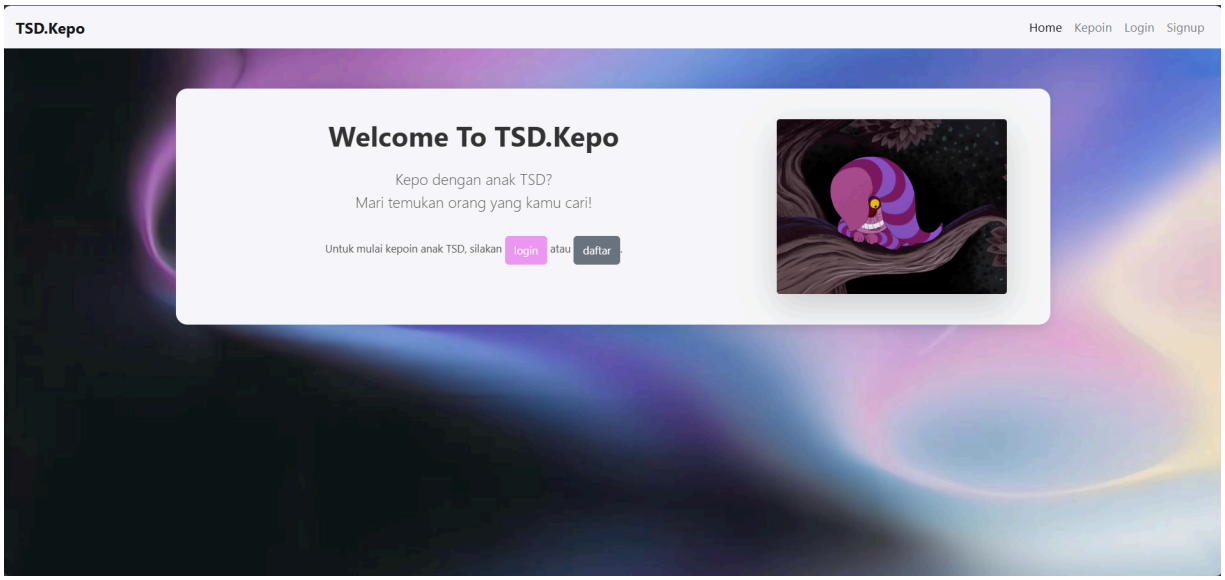
Fungsi pencarian dan logout yang diimplementasikan menggunakan Flask memungkinkan pengelolaan data siswa dan sesi pengguna secara efektif dan aman. Fitur pencarian memanfaatkan ekspresi reguler untuk menyediakan fleksibilitas dalam pencarian data, sedangkan prosedur logout memastikan bahwa sesi pengguna dibersihkan dengan aman, menghindari risiko keamanan. Implementasi ini menunjukkan praktik baik dalam pengembangan aplikasi web dengan menangani autentikasi dan manajemen data pengguna secara terpusat dan aman.

EVALUASI PERFORMA

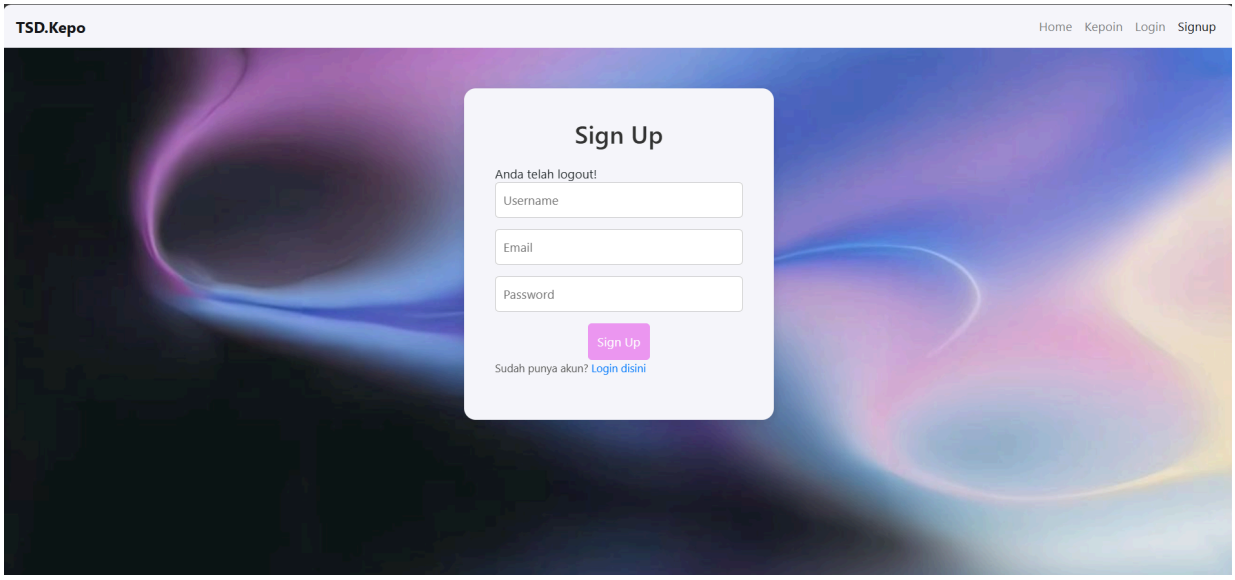
1. **Kecepatan Ekstraksi dan Validasi Input**
   - ➢ **Kecepatan Ekstraksi Data**
     - **Proses Cepat:** Ekstraksi data dalam sistem ini tergolong cepat. Pengguna hanya perlu memasukkan NIM yang diinginkan dan menekan tombol 'Cari'. Informasi terkait NIM tersebut kemudian ditampilkan segera setelahnya.
     - **Contoh Implementasi:** Dalam uji coba, menggunakan NIM 12 digit dari angkatan 2021, sistem berhasil menampilkan informasi yang relevan seperti nama, NIM, fakultas, program studi, dan angkatan dengan cepat. Ini menunjukkan bahwa sistem efisien dalam mengambil dan menampilkan data dari database.
     - **Pengujian dengan Berbagai Input:** Sistem menangani input NIM dengan 9 digit dari angkatan 2022 dan 2023 dengan efisiensi yang sama, memverifikasi bahwa implementasi regex (re.match()) berfungsi secara konsisten untuk berbagai pola input.
   - ➢ **Validasi dan Penanganan Input yang Tidak Valid**
     - **Penanganan Kesalahan Input:** Jika pengguna memasukkan NIM yang salah atau tidak sesuai format yang ditentukan, sistem secara otomatis memberikan feedback "NIM tidak valid". Ini membantu pengguna memahami bahwa input yang diberikan salah dan membutuhkan koreksi.
     - **Ketidaklengkapan Data:** Ada kasus ketika data yang dikembalikan tidak lengkap (misalnya, nama tidak tersedia meskipun NIM valid). Ini menandakan bahwa database tidak memiliki informasi lengkap untuk NIM tersebut. Dalam situasi ini, pengguna disarankan untuk memasukkan data yang diperlukan di halaman Service, yang menunjukkan baik sistem kegagalan toleransi dan responsivitas terhadap data yang tidak lengkap.
   - ➢ **Implikasi**
     - **Feedback Pengguna:** Sistem yang cepat dalam menampilkan hasil dan efektif dalam memberikan umpan balik tentang kesalahan membantu meningkatkan pengalaman pengguna dan kepuasan. Ini juga mengurangi risiko kesalahan data dan meningkatkan efisiensi dalam pemeliharaan data.

LAMPIRAN

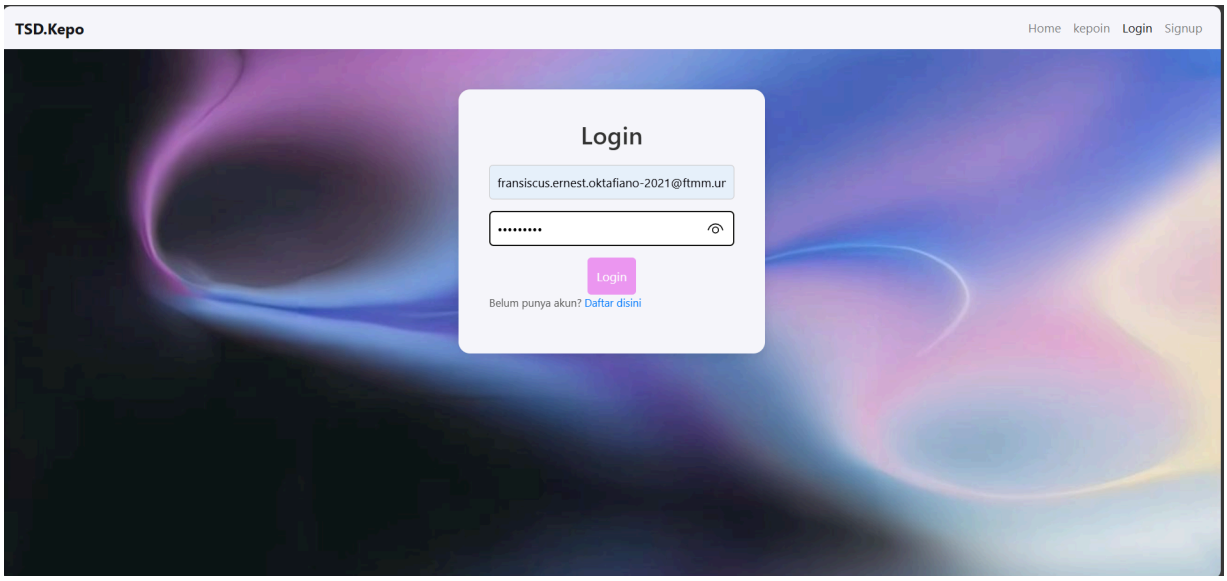
Tampilan *Home*



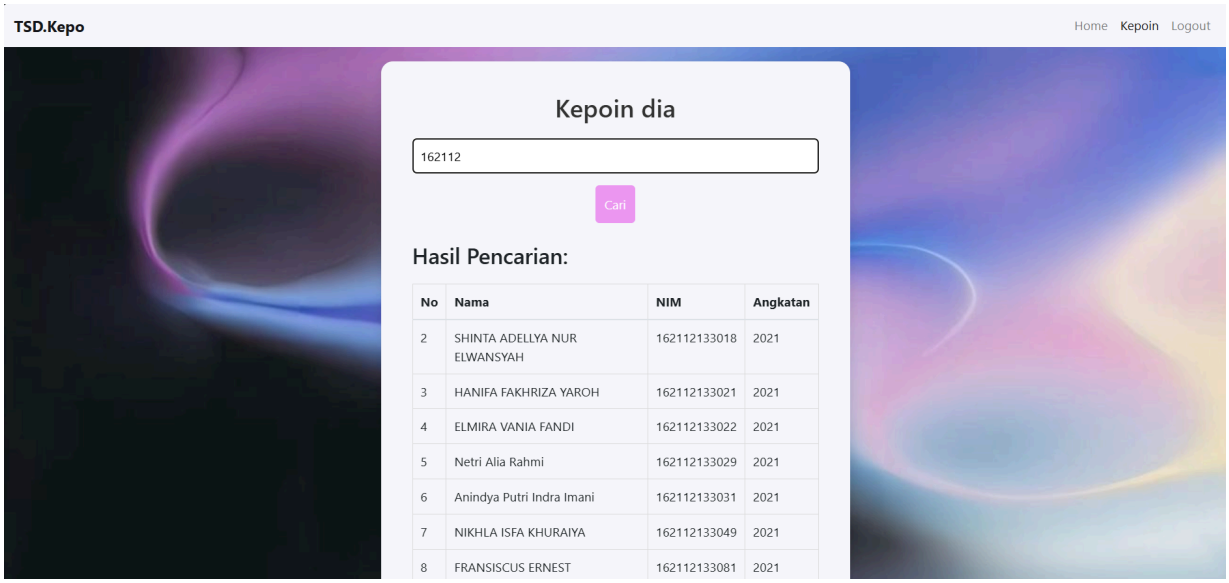
Tampilan *Sign-Up (penggunaan RegEx ddalam rute signup untuk validasi email dan password.)*



Tampilan *Login*



Tampilan *Filter hasil pencarian data mahasiswa menggunakan regex*



Codingan untuk Web app TSD.Kepo : Tugas 1