

LAPORAN PRAKTIKUM

DATA SCRAPING

Disusun untuk Memenuhi Mata Kuliah Praktikum Data Scraping
Dibimbing oleh Arif Hadi Sumitro, M.Kom



Oleh:

Nama: Annisa Maharani

Nim: 1123102127

PROGRAM STUDI S1 TEKNIK INFORMATIKA
SEKOLAH TINGGI ILMU KOMPUTER PGRI BANYUWANGI
2025

KATA PENGANTAR

Dengan mengucapkan syukur ke hadirat Allah SWT atas limpahan rahmat dan karunia-Nya, laporan praktikum mata kuliah Data Scraping ini dapat saya selesaikan dengan baik. Laporan ini disusun sebagai bagian dari tugas akhir praktikum yang bertujuan untuk mendokumentasikan hasil pembelajaran serta implementasi konsep-konsep yang telah dipelajari selama perkuliahan. Mata kuliah Data Scraping merupakan komponen penting dalam bidang teknologi informasi karena memberikan pemahaman tentang bagaimana mengambil, mengolah, dan menyajikan data secara otomatis dari berbagai sumber web. Selama proses praktikum, saya memperoleh pengalaman berharga dalam mengaplikasikan teori-teori seperti parsing HTML, penggunaan pustaka Python seperti BeautifulSoup dan Requests, serta penerapan data scraping untuk berbagai kebutuhan informasi.

Melalui tantangan-tantangan teknis yang dihadapi, saya belajar untuk memecahkan masalah secara logis, teliti, dan kreatif, serta memahami pentingnya etika dalam pengambilan data dari internet. Praktikum ini juga melatih saya dalam berpikir analitis dan menyusun hasil scraping ke dalam format yang bermanfaat.

Saya mengucapkan terima kasih yang sebesar-besarnya kepada dosen pembimbing mata kuliah ini atas bimbingan, dukungan, dan ilmu yang telah diberikan. Terima kasih juga saya sampaikan kepada teman-teman praktikum atas kerja sama dan diskusinya yang sangat membantu selama proses pembelajaran berlangsung.

Saya menyadari bahwa laporan ini masih memiliki kekurangan, baik dalam isi maupun penyajian. Oleh karena itu, saya terbuka terhadap kritik dan saran yang membangun demi perbaikan di masa mendatang. Semoga laporan ini dapat memberikan manfaat dan menjadi referensi bagi pembaca yang ingin memahami lebih dalam mengenai praktik data scraping.

Dengan segala kerendahan hati, saya ucapkan terimakasih dan semoga laporan ini dapat memenuhi tujuan yang diharapkan.

Banyuwangi, 21 Juli 2025

Annisa Maharani

DAFTAR ISI

Daftar Isi

KATA PENGANTAR	ii
DAFTAR GAMBAR	vii
BAB 1	1
1.1 Capaian Praktikum Pertemuan 1	1
1.2 Indikator Capaian	1
1.3 Landasan Teori	1
1.4 Pelaksanaan Praktikum	2
1.4.1 Percobaan Pertama	3
1.5 Kesimpulan	5
1.5.1 Kesimpulan Percobaan 1	5
DAFTAR PUSTAKA	6
BAB 2	7
2.1 Capaian Praktikum Pertemuan 2	7
2.2 Indikator Capaian	7
2.3 Landasan Teori	7
2.4 Pelaksanaan Praktikum	8
2.4.1 Percobaan Pertama	8
2.5 Kesimpulan	12
2.5.1 Kesimpulan Percobaan 1	12
DAFTAR PUSTAKA	13
BAB 3	14
3.1 Capaian Praktikum Pertemuan 3	14

3.2 Indikator Capaian	14
3.3 Landasan Teori.....	14
3.4 Pelaksanaan Praktikum.....	15
3.4.1 Percobaan Pertama.....	15
3.4.2 Percobaan Kedua	20
3.5 Kesimpulan.....	24
3.5.1 Kesimpulan Percobaan 1.....	24
3.5.2 Kesimpulan Percobaan 2.....	24
DAFTAR PUSTAKA	26
BAB 4.....	27
4.1 Capaian Praktikum Pertemuan 4.....	27
4.2 Indikator Capaian	27
4.3 Landasan Teori.....	27
4.4 Pelaksanaan Praktikum.....	28
4.4.1 Percobaan Pertama	28
4.4.2 Percobaan Kedua	32
4.5 Kesimpulan.....	35
4.5.1 Kesimpulan Percobaan 1.....	35
4.5.2 Kesimpulan Percobaan 2.....	35
DAFTAR PUSTAKA	37
BAB 5.....	38
5.1 Capaian Praktikum Pertemuan 5.....	38
5.2 Indikator Capaian	38

5.3 Landasan Teori.....	38
5.4 Pelaksanaan Praktikum	39
5.4.1 Percobaan Pertama.....	39
5.4.2 Percobaan Kedua	43
5.5 Kesimpulan.....	46
5.5.1 Kesimpulan Percobaan 1.....	46
5.5.2 Kesimpulan Percobaan 2.....	47
DAFTAR PUSTAKA	48
BAB 6.....	49
6.1 Capaian Praktikum Pertemuan 6.....	49
6.2 Indikator Capaian	49
6.3 Landasan Teori.....	49
6.4 Pelaksanaan Praktikum.....	50
6.4.1 Percobaan Pertama.....	50
6.4.2 Percobaan Kedua	53
6.5 Kesimpulan.....	57
6.5.1 Kesimpulan Percobaan 1.....	57
6.5.2 Kesimpulan Percobaan 2.....	57
DAFTAR PUSTAKA	59
BAB 7	60
7.1 Capaian Praktikum Pertemuan 5.....	60
7.2 Indikator Capaian	60
7.3 Landasan Teori.....	60

7.4 Pelaksanaan Praktikum	61
7.4.1 Percobaan Pertama	61
7.5 Kesimpulan.....	66
7.5.1 Kesimpulan Percobaan 1.....	66
DAFTAR PUSTAKA	68
BAB 8.....	69
8.1 Capaian Praktikum Pertemuan 8.....	69
8.2 Indikator Capaian	69
8.3 Landasan Teori.....	69
8.4 Pelaksanaan Praktikum	70
8.4.1 Percobaan Pertama	70
8.5 Kesimpulan.....	97
8.5.1 Kesimpulan Percobaan 1.....	97
DAFTAR PUSTAKA	98

DAFTAR GAMBAR

Daftar Gambar

1.1 Gambar Hasil Uji Coba Bab 1	4
2.1 Gambar Hasil Uji Coba Bab 2	11
3.1 Gambar Hasil Scraping Bab 3	19
3.2 Gambar Hasil Scraping Bab 3	23
4.1 Gambar Hasil Scraping Bab 4	31
4.2 Gambar Hasil Scraping Bab 4	34
5.1 Gambar Hasil Scraping Bab 5	42
5.2 Gambar Hasil Scraping Bab 5	46
6.1 Gambar Hasil Scraping Bab 6	52
6.2 Gambar Hasil Scraping Bab 6	56
7.1 Gambar Hasil Uji Coba Bab 7	66
8.1 Gambar Tampilan Halaman Index Bab 8	93
8.2 Gambar Tampilan Halaman Bola Sport Bab 8	93
8.3 Gambar Tampilan Halaman Detik Jatim Bab 8	94
8.4 Gambar Tampilan Halaman Detik Jateng Bab 8	94
8.5 Gambar Tampilan Halaman Detik Jabar Bab 8	94
8.6 Gambar Tampilan Halaman Berita Gabungan Liputan6 Bab 8	95
8.7 Gambar Tampilan Halaman Berita Gabungan Bab 8	95
8.8 Gambar Tampilan Halaman Detail Berita Bab 8	95
8.9 Gambar Tampilan Halaman Detail Berita Bab 8	96

BAB 1

1.1 Capaian Praktikum Pertemuan 1

Mahasiswa mampu mengelola struktur direktori dan file menggunakan Python. Mahasiswa mampu menggunakan modul os untuk membuat folder dan file secara otomatis. Mahasiswa mampu melakukan pengecekan eksistensi file dan folder dengan os.path.exists(). Mahasiswa juga mampu menulis data ke dalam file teks menggunakan Python. Mahasiswa mampu membuat program otomatisasi sederhana dengan perulangan dan logika pemrograman dasar.

1.2 Indikator Capaian

Mahasiswa mampu membuat folder utama menggunakan Python sebagai bukti pemahaman dasar terhadap manipulasi sistem file menggunakan modul os. Mahasiswa mampu membuat subfolder secara dinamis menggunakan perulangan (looping). Mahasiswa mampu membuat lima subfolder secara otomatis dengan menggabungkan path dengan benar menggunakan os.path.join. Mahasiswa mampu membuat dan menuliskan isi ke dalam file teks secara otomatis. Mahasiswa menguasai kemampuan dasar dalam otomatisasi pengelolaan file, direktori, dan penerapan logika pemrograman Python.

1.3 Landasan Teori

Menurut Guttag (2016), otomatisasi dalam pemrograman Python, termasuk dalam pengelolaan file dan direktori, merupakan salah satu kekuatan utama Python yang menjadikannya sangat cocok untuk scripting dan pemrosesan data skala kecil hingga menengah. Dengan logika pemrograman dasar, seperti percabangan dan perulangan, pengguna dapat mengembangkan alat bantu sederhana yang sangat berguna untuk tugas-tugas berulang.

Menurut Summerfield (2010) menyatakan bahwa penggunaan struktur kontrol seperti perulangan for dalam Python sangat efisien untuk menciptakan objek secara

berulang, baik untuk data maupun struktur file. Ini sangat berguna saat membuat banyak folder atau file dengan pola nama berurutan, seperti folder1, folder2, dan seterusnya, karena menyederhanakan kode program dan mengurangi duplikasi

1.4 Pelaksanaan Praktikum

1.4.1 Percobaan Pertama

Mahasiswa diperintahkan untuk membuat struktur folder dan file secara otomatis menggunakan Python, dengan tujuan memahami penggunaan modul os dan dasar-dasar manipulasi file dan direktori.

a. Script / Setting Program

```
import os

main_folder = 'data_scraping'

subfolders = [f'dir{i}' for i in range(1, 6)]

if not os.path.exists(main_folder):
    os.makedirs(main_folder)
    print(f"Folder '{main_folder}' berhasil dibuat.")
else:
    print(f"Folder '{main_folder}' sudah tersedia.")

for folder in subfolders:
    folder_path = os.path.join(main_folder, folder)

    if not os.path.exists(folder_path):
        os.makedirs(folder_path)
        print(f"Subfolder '{folder}' dibuat.")
    else:
        print(f"Subfolder '{folder}' sudah tersedia.")

    for i in range(1, 6):
        file_name = f'file{i}.txt'
        file_path = os.path.join(folder_path, file_name)

        if not os.path.exists(file_path):
            with open(file_path, 'w') as f:
                f.write(f"Ini isi file ke-{i} di folder {folder}.")
            print(f"File '{file_name}' dibuat di folder '{folder}'.")
        else:
            print(f"File '{file_name}' sudah tersedia di folder '{folder}' .")
```

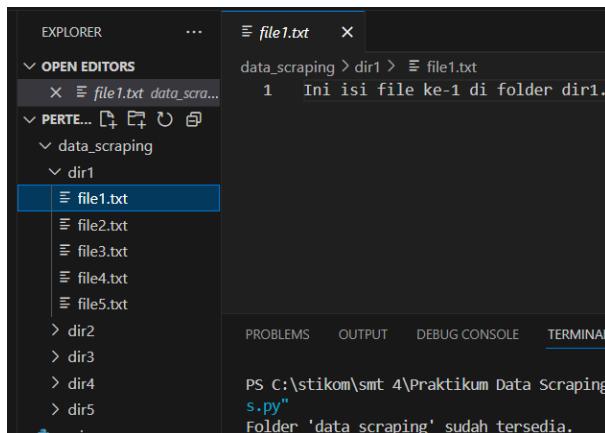
b. Langkah Uji Coba

- **import os** Mengimpor modul os, yang digunakan untuk berinteraksi dengan sistem operasi, seperti membuat folder dan file, mengecek apakah file/folder sudah ada, menggabungkan path, dll.
- **main_folder = 'data_scraping'** Mendefinisikan nama folder utama (data_scraping).
- **subfolders = [f'dir{i}' for i in range(1, 6)]** Membuat daftar 5 subfolder (dir1 sampai dir5) secara otomatis menggunakan list comprehension.
- **if not os.path.exists(main_folder):** Mengecek apakah folder data_scraping sudah ada.
- **os.makedirs(main_folder)** Jika belum ada, maka folder tersebut akan dibuat menggunakan os.makedirs().
- **print(f"Folder '{main_folder}' berhasil dibuat.")** mencetak bahwa folder berhasil dibuat.
- **else: print(f"Folder '{main_folder}' sudah tersedia.")** Jika sudah ada, maka mencetak bahwa folder sudah tersedia.
- **for folder in subfolders:** Melakukan iterasi untuk setiap nama subfolder.
- **folder_path = os.path.join(main_folder, folder)** digunakan untuk menyusun path lengkap subfolder (data_scraping/dir1, dst.).
- **if not os.path.exists(folder_path):** Mengecek apakah subfolder (dir1 sampai dir5) sudah ada di dalam folder utama.
- **os.makedirs(folder_path)** Mencetak informasi keberhasilan atau jika subfolder sudah ada.
- **print(f"Subfolder {folder} dibuat.")** else:
print(f"Subfolder {folder} sudah tersedia.") Jika sudah, maka akan mencetak bahwa subfolder sudah tersedia.
- **for i in range(1, 6):** Melakukan perulangan sebanyak 5 kali untuk membuat file file1.txt sampai file5.txt di setiap subfolder.

- `file_name = f'file{i}.txt'` file_name menyimpan nama filenya.
- `file_path = os.path.join(folder_path, file_name)` file_path menyimpan path lengkap ke file tersebut.
- `if not os.path.exists(file_path):` Mengecek apakah file tersebut sudah ada.
- `with open(file_path, 'w') as f:` Jika belum ada, file dibuat menggunakan with open(..., 'w'), artinya file dibuka untuk ditulis.
- `f.write(f"Ini isi file ke-{i} di folder {folder}.)")` Jika sudah ada, program mencetak bahwa file tersebut sudah tersedia.
- `print(f"File '{file_name}' dibuat di folder '{folder}'")` Jika file sudah ada, hanya mencetak bahwa file sudah tersedia.

c. Hasil Uji Coba

```
PS C:\stikom\smt 4\Praktikum Data Scraping\pertemuan1>
s.py"
Folder 'data_scraping' sudah tersedia.
Subfolder 'dir1' sudah tersedia.
File 'file1.txt' sudah tersedia di folder 'dir1'.
File 'file2.txt' sudah tersedia di folder 'dir1'.
File 'file3.txt' sudah tersedia di folder 'dir1'.
File 'file4.txt' sudah tersedia di folder 'dir1'.
File 'file5.txt' sudah tersedia di folder 'dir1'.
Subfolder 'dir2' sudah tersedia.
File 'file1.txt' sudah tersedia di folder 'dir2'.
File 'file2.txt' sudah tersedia di folder 'dir2'.
File 'file3.txt' sudah tersedia di folder 'dir2'.
File 'file4.txt' sudah tersedia di folder 'dir2'.
File 'file5.txt' sudah tersedia di folder 'dir2'.
Subfolder 'dir3' sudah tersedia.
File 'file1.txt' sudah tersedia di folder 'dir3'.
File 'file2.txt' sudah tersedia di folder 'dir3'.
File 'file3.txt' sudah tersedia di folder 'dir3'.
File 'file4.txt' sudah tersedia di folder 'dir3'.
File 'file5.txt' sudah tersedia di folder 'dir3'.
Subfolder 'dir4' sudah tersedia.
```



Gambar 1.1 Hasil Uji Coba Membuat 5 Directory

d. Analisa Hasil

Kode ini berhasil membuat folder utama dengan nama data_scraping, kemudian di dalamnya menghasilkan lima subfolder bernama dir1 hingga dir5. Setiap subfolder tersebut berisi lima file teks (file1.txt hingga file5.txt) yang masing-masing telah diisi dengan kalimat yang menunjukkan urutan file dan nama foldernya. Selain itu, program ini juga dilengkapi dengan pengecekan eksistensi sebelum membuat folder atau file, sehingga saat dijalankan ulang tidak terjadi duplikasi atau error.

1.5 Kesimpulan

1.5.1 Kesimpulan Percobaan 1

Pada percobaan ini, mahasiswa berhasil mengimplementasikan fungsionalitas untuk membuat struktur folder dan file secara otomatis menggunakan modul os pada Python. Program ini menunjukkan bahwa mahasiswa memahami cara dalam membuat folder, subfolder, serta file, lengkap dengan penulisan isi dan pengecekan eksistensi sebelum proses pembuatan dilakukan. Hal ini membuktikan bahwa mahasiswa dapat memahami konsep dasar pengelolaan file dan direktori secara terstruktur, efisien, dan aman. Program juga dapat dijalankan berulang kali tanpa menghasilkan error atau duplikasi, yang mengindikasikan penggunaan logika kontrol yang tepat dan efektif.

Mengetahui:

Dosen Pengampu Mata Kuliah

Arif Hadi Sumitro , M.Kom

NIKP. xxx

DAFTAR PUSTAKA

1. Guttag, J. V. (2016). *Introduction to Computation and Programming Using Python* (2nd ed.). MIT Press.
2. Zelle, J. M. (2017). Python Programming: An Introduction to Computer Science (3rd ed.). Franklin, Beedle & Associates Inc.
3. Python Software Foundation. (2024). Reading and Writing Files. Diakses dari <https://docs.python.org/3/tutorial/inputoutput.html#reading-and-writing-files>

BAB 2

2.1 Capaian Praktikum Pertemuan 2

- Mahasiswa mampu melakukan instalasi BeautifulSoup.
- Mahasiswa mengimplementasikan BeautifulSoup untuk melakukan scraping.
- Mahasiswa mengimplementasikan request untuk melakukan scraping.

2.2 Indikator Capaian

Mahasiswa mampu melakukan instalasi dan konfigurasi modul eksternal seperti BeautifulSoup dan requests pada Python. Selain itu, mahasiswa mampu mengimplementasikan BeautifulSoup untuk melakukan web scraping dengan mengekstrak data dari halaman HTML secara terstruktur. Mahasiswa juga menunjukkan pemahaman dalam penggunaan modul requests untuk mengambil konten dari halaman web secara otomatis. Melalui praktikum ini, mahasiswa dapat memproses dan mengambil informasi spesifik dari struktur HTML berdasarkan tag dan atribut yang sesuai.

2.3 Landasan Teori

Web scraping merupakan teknik otomatisasi untuk mengambil data dari situs web secara terstruktur. Menurut Mitchell (2018), web scraping adalah metode untuk mengekstrak informasi dari halaman web dengan cara memanfaatkan program yang dapat mengakses, membaca, dan memproses konten HTML. Teknik ini sangat bermanfaat dalam pengumpulan data secara cepat dan efisien, terutama ketika data tidak disediakan dalam bentuk API atau file unduhan resmi.

Menurut Sweigart (2019), BeautifulSoup memungkinkan pengembang untuk menavigasi dan memanipulasi elemen HTML secara sederhana menggunakan pemahaman dasar struktur dokumen HTML dan CSS. Sementara itu, modul requests digunakan untuk mengirim permintaan HTTP dan mendapatkan respon halaman web dalam bentuk teks atau data lainnya.

Menurut Yulianto (2020), kemampuan mahasiswa dalam memahami dan mengimplementasikan modul-modul tersebut mencerminkan penguasaan dasar pemrograman Python serta kemampuan untuk memecahkan masalah berbasis data secara otomatis. Web scraping juga memberikan pemahaman praktis tentang bagaimana data dapat diakses, diambil, dan digunakan kembali secara sah untuk tujuan analisis, riset, maupun pengembangan aplikasi.

2.4 Pelaksanaan Praktikum

2.4.1 Percobaan Pertama

Mahasiswa diminta untuk melakukan instalasi dan implementasi modul BeautifulSoup dan requests pada Python dengan tujuan memahami dasar-dasar teknik web scraping, cara mengambil data dari halaman web secara otomatis, serta mengolah struktur HTML untuk mengekstrak informasi yang relevan.

a. Script / Setting Program

```
import requests
from bs4 import BeautifulSoup

url = 'https://www.antaranews.com/'

response = requests.get(url)

if response.status_code == 200:

    soup = BeautifulSoup(response.content, 'html.parser')

    links = soup.find_all('a', href=True)
    for link in links:

        if 'berita' in link['href']:
            print(f'Tautan Berita: https://www.antaranews.com{link["href"]}')

    categories = soup.find_all('li', class_='list-inline-item')
    for category in categories:
        tag = category.find('a')
        if tag:
            print(f'Kategori Berita: {tag.get_text(strip=True)}')

    times = soup.find_all('li', class_='list-inline-item')
    for time in times:
        span = time.find('span', class_='text-secondary')
        if span:
            print(f'Waktu: {span.get_text(strip=True)})')

    footer_links = soup.find_all('div', class_='widget_footer')
    for footer in footer_links:

        footer_items = footer.find_all('a', href=True)
```

```

        for item in footer_items:
            print(f"URL Footer: {item['href']}")

section_headers = soup.find_all('h4', class_='border_section')
for section_header in section_headers:

    links = section_header.find_all('a')
    for link in links:
        title = link.get('title') # Judul dari atribut title
        if title:
            print(f"Title: {title}")
else:
    print('Gagal mengakses website.')

```

b. Langkah Uji Coba

- **import requests**, untuk mengambil konten dari website.
- **BeautifulSoup**, dari **bs4** untuk parsing HTML.
- Menentukan URL target (<https://www.antaranews.com/>) dan mengirim permintaan HTTP GET menggunakan **requests.get()**.
- **if response.status_code == 200**, mengecek apakah permintaan berhasil dengan status kode 200 (artinya halaman berhasil diakses).
- **soup = BeautifulSoup(response.content, 'html.parser')**, mengubah konten HTML dari respon menjadi objek BeautifulSoup agar mudah ditelusuri.
- **links = soup.find_all('a', href=True)**, mengambil semua <tag> yang memiliki atribut href.
- **for link in links: if 'berita' in link['href']**, menyaring kata yang mengandung 'berita' dalam tautannya.
- **categories = soup.find_all('li', class_='list-inline-item')**, mencari elemen dengan class 'list-inline-item'. Menampilkan teks dari link <a> di dalamnya sebagai kategori berita.
- **times = soup.find_all('li', class_='list-inline-item')**, masih dari elemen yang sama, kali ini dicari dengan class 'text-secondary' untuk mengambil informasi waktu publikasi.

- `footer_links = soup.find_all('div', class_='widget__footer')`, mencari semua elemen div dengan class ‘**widget__footer**’. Mengambil semua link `<a>` di dalamnya dan mencetak URL-nya.
- `section_headers = soup.find_all('h4', class_='border_section')`, menelusuri elemen h4 dengan class ‘**border_section**’. Menampilkan isi atribut title dari setiap link `<a>` yang ditemukan, yang menggambarkan judul atau topik berita.
- Jika `response.status_code` bukan 200, maka mencetak: `print('Gagal mengakses website.')`

c. Hasil Uji Coba

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

tm_campaign=category_home
Tautan Berita: https://www.antaranews.comhttps://www.antaranews.comberita/474471
tm_campaign=category_home
Tautan Berita: https://www.antaranews.comhttps://www.antaranews.comberita/474462
sktop&utm_campaign=category_home
Tautan Berita: https://www.antaranews.comhttps://www.antaranews.comberita/474462
sktop&utm_campaign=category_home
Tautan Berita: https://www.antaranews.comhttps://www.antaranews.comberita/474441
s&utm_medium=desktop&utm_campaign=category_home
Tautan Berita: https://www.antaranews.comhttps://www.antaranews.comberita/474441
s&utm_medium=desktop&utm_campaign=category_home
Tautan Berita: https://www.antaranews.comhttps://www.antaranews.comberita/474414
rce=antaranews&utm_medium=desktop&utm_campaign=category_home
Tautan Berita: https://www.antaranews.comhttps://www.antaranews.comberita/474414
rce=antaranews&utm_medium=desktop&utm_campaign=category_home
Tautan Berita: https://www.antaranews.comhttps://www.antaranews.comberita/474289
desktop&utm_campaign=category_home

Tautan Berita: https://www.antaranews.comhttps://www.antaranews.comberita/474289
Kategori Berita: Haji 2025
Kategori Berita: Judi Online
Kategori Berita: Konflik Iran Israel
Kategori Berita: Liga Champions
Kategori Berita: MBG
Kategori Berita: IKN Nusantara
Kategori Berita: Palestina
Kategori Berita: Konflik Rusia - Ukraina
Kategori Berita: Arus Balik
Kategori Berita: Arus Mudik
Kategori Berita: Piala AFF U-19
Kategori Berita: Kebakaran Los Angeles

```

```

Waktu: 1 jam lalu
Waktu: 2 jam lalu
Waktu: 2 jam lalu
Waktu: 2 jam lalu
Waktu: 5 jam lalu
Waktu: 5 jam lalu
Waktu: 6 jam lalu
Waktu: 9 jam lalu
Waktu: 10 jam lalu
Waktu: 11 jam lalu
Waktu: 5 menit lalu
Waktu: 11 menit lalu
Waktu: 11 menit lalu
Waktu: 13 menit lalu

WAKTU: 17 Mei 2023
URL Footer: https://www.antaranews.com/terkini
URL Footer: https://www.antaranews.com/top-news
URL Footer: https://www.antaranews.com/terpopuler
URL Footer: https://www.antaranews.com/pilihan-editor
URL Footer: https://www.antaranews.com/foto
URL Footer: https://www.antaranews.com/video
URL Footer: https://www.antaranews.com/infografik
URL Footer: https://www.antaranews.com/politik
URL Footer: https://www.antaranews.com/hukum
URL Footer: https://www.antaranews.com/ekonomi
URL Footer: https://www.antaranews.com.metro
URL Footer: https://www.antaranews.com/sepakbola
URL Footer: https://www.antaranews.com/olahraga

SKE PAPER: https://paperkit.net
Title: Info Haji 2025
Title: Pilihan Editor
Title: Indeks Pilihan Editor
Title: Berita Terkini
Title: Indeks Berita Terkini
Title: Berita Terpopuler
Title: Telaah
Title: Indeks Telaah
Title: Politik
Title: Indeks Politik
Title: Hukum
Title: Indeks Hukum

```

Gambar 2.1 Hasil Uji Coba Scraping dari web antaresnews.com

d. Analisa Hasil

Program berhasil melakukan scraping pada berbagai elemen penting seperti tautan berita, kategori, waktu publikasi, link footer, dan judul. Beberapa data seperti tautan berita mengalami pengulangan, perlu difilter agar tidak mencetak data yang sama. Secara umum, struktur scraping berjalan sesuai dengan target dan informasi yang ditampilkan cukup lengkap untuk membentuk berita ringkas berbasis data dari situs.

2.5 Kesimpulan

2.5.1 Kesimpulan Percobaan 1

Pada percobaan ini menunjukkan bahwa mahasiswa telah berhasil mengimplementasikan teknik dasar web scraping menggunakan Python dengan memanfaatkan modul requests dan BeautifulSoup. Mahasiswa mampu mengakses halaman situs berita antaranews.com, mengekstrak berbagai elemen penting seperti tautan berita, kategori, waktu publikasi, tautan pada bagian footer, dan judul.

Scrapper berhasil menangkap informasi secara otomatis dari struktur HTML yang kompleks, membuktikan pemahaman mahasiswa terhadap teknik navigasi elemen web serta pengolahan data teks. Walaupun terdapat sedikit duplikasi pada hasil tautan berita, hal tersebut menjadi bagian dari pembelajaran dalam penyaringan dan pembersihan data (data cleaning).

Secara keseluruhan, percobaan ini mencerminkan keberhasilan mahasiswa dalam memahami konsep dasar *web scraping*, parsing HTML, serta penerapan logika pemrograman Python untuk otomasi pengambilan data dari web.

Mengetahui:

Dosen Pengampu Mata Kuliah

Arif Hadi Sumitro , M.Kom

NIKP. xxx

DAFTAR PUSTAKA

- Mitchell, R. (2018). *Web Scraping with Python: Collecting More Data from the Modern Web* (2nd ed.). O'Reilly Media.
- Sweigart, A. (2019). *Automate the Boring Stuff with Python: Practical Programming for Total Beginners* (2nd ed.). No Starch Press.
- Yulianto, B. (2020). Pemrograman Python untuk Data Sains dan Otomatisasi. Deepublish.

BAB 3

3.1 Capaian Praktikum Pertemuan 3

Pada praktikum pertemuan ke-3 ini mahasiswa diharapkan mampu mengintegrasikan fungsi pembuatan folder dan file dengan request URL, menggunakan BeautifulSoup dan requests untuk scraping data, serta menganalisis dan menyimpan hasil scraping tersebut ke dalam file secara terstruktur.

3.2 Indikator Capaian

- Mahasiswa melakukan integrasi fungsi folder dan request.
- Mahasiswa melakukan integrasi fungsi pada beautifulsoup dan request.
- Mahasiswa melakukan scraping pada website dengan menggunakan request & beautifulsoap serta menyimpannya dalam file.

3.3 Landasan Teori

Menurut Dr. Charles Severance, Web scraping adalah teknik dasar untuk mengambil data dari website ketika API tidak tersedia. Menggunakan tools seperti BeautifulSoup dan requests memungkinkan mahasiswa untuk berinteraksi langsung dengan konten web secara nyata dan meningkatkan logika pemrograman mereka.

Menurut Ryan Mitchell, Web scraping melatih mahasiswa untuk memahami struktur HTML, permintaan HTTP, dan proses parsing data. Ini bukan hanya tentang menulis kode, tapi juga tentang pemecahan masalah dan pemahaman etika pengambilan data.

Menurut Eric Matthes, dengan menggunakan BeautifulSoup dan requests, mahasiswa mendapatkan pengalaman nyata dalam berinteraksi dengan internet dan

mengekstrak data yang berguna. Ini adalah salah satu cara terbaik untuk membuat Python terasa praktis dan memberdayakan.

3.4 Pelaksanaan Praktikum

3.4.1 Percobaan Pertama

Pada percobaan pertama ini, mahasiswa diperintahkan untuk membuat pengaturan pada sisi program utama (main.py) agar dapat membaca data dari suatu URL secara otomatis, serta memungkinkan pengguna untuk menentukan URL dan direktori penyimpanan secara manual atau dinamis. Dalam proses scraping pertama kali, program diatur agar mengambil dan menampilkan struktur data dari website (tidak dilakukan berulang-ulang), serta menampilkan waktu dan tanggal saat data diambil. Selain itu, program juga diatur agar dapat menyimpan hasil scraping ke dalam file dengan format yang memuat informasi tanggal dan waktu pengambilan data secara lengkap.

a. Script / Setting Program

Main.py

```
from bs4 import BeautifulSoup
import requests
import fungsi
import os

def main_scraper(url, directory):
    fungsi.create_directory(directory)

    response = requests.get(url)
    if response.status_code != 200:
        print(f"Gagal mengakses halaman: {url}")
        return

    soup = BeautifulSoup(response.text, "html.parser")

    # Ambil judul utama halaman
    title_tag = soup.select_one("#firstHeading")
    title = title_tag.text.strip() if title_tag else "Tidak ada judul"

    # Ambil konten utama
    content_div = soup.select_one("div.mw-parser-output")
    if not content_div:
```

```

print("Konten tidak ditemukan.")
return

# Ambil paragraf pertama
first_paragraph = ""
for element in content_div.find_all(["p", "div"], recursive=False):
    if element.name == "p" and element.text.strip():
        first_paragraph = element.text.strip()
        break

# Ambil semua subjudul h2 dan h3 (lebih toleran)
subheadings = []
headings = content_div.find_all(["h2", "h3"])
for heading in headings:
    span = heading.find("span", class_="mw-headline")
    if span and span.text.strip():
        subheadings.append(span.text.strip())

# Jika tidak ditemukan, beri catatan
if not subheadings:
    subheadings.append("(Tidak ada subjudul ditemukan)")

# Format hasil scraping
article_format = (
    f"Judul Halaman: {title}\n\n"
    f"Ringkasan Awal:\n{first_paragraph}\n\n"
    f"Daftar Subjudul:\n"
)
for i, sub in enumerate(subheadings, 1):
    article_format += f"{i}. {sub}\n"

article_format += f"\n{'-'*40}\nSumber: {url}\n"

# Simpan ke file
file_path = os.path.join(directory, "artikel.txt")
if not fungsi.does_file_exist(file_path):
    fungsi.create_new_file(file_path)

fungsi.write_to_file(file_path, article_format)
print("Scraping selesai. Data disimpan di folder:", directory)

# Jalankan
main_scraper("https://id.wikipedia.org/wiki/Sejarah_Islam", "Hasil")

```

Fungsi.py

```
import os

def create_directory(path):
    if not os.path.exists(path):
        os.mkdir(path)

def create_new_file(path):
    f = open(path, 'w', encoding='utf-8')
    f.write("")
    f.close()

def does_file_exist(path):
    return os.path.exists(path)

def write_to_file(path, text):
    with open(path, 'a', encoding='utf-8') as f:
        f.write(text + "\n")
```

b. Langkah Uji Coba

Menjalankan main.py

- **BeautifulSoup**: Untuk mem-parsing dan mengekstrak elemen HTML.
- **requests**: Untuk melakukan permintaan HTTP (GET) ke URL target.
- **main_scraper(url, directory)**, bertugas melakukan seluruh proses scraping dan penyimpanan hasil.
- **fungsi.create_directory(directory)**, mengecek dan membuat folder (misalnya Hasil/) jika belum ada.
- **response = requests.get(url)**, **if response.status_code != 200: print(f"Gagal mengakses halaman: {url}")**, mengirim permintaan ke URL yang diberikan. Jika gagal (status_code ≠ 200), fungsi akan dihentikan.
- **soup = BeautifulSoup(response.text, "html.parser")**, parsing HTML dari halaman menjadi objek yang bisa ditelusuri dengan BeautifulSoup.
- **title_tag = soup.select_one("#firstHeading"), title = title_tag.text.strip() if title_tag else "Tidak ada judul"**, mengambil elemen id="firstHeading" yang berisi judul utama artikel Wikipedia.

- `div.mw-parser-output`, kontainer utama isi artikel Wikipedia.
- `for heading in content_div.find_all(["h2", "h3"]): span = heading.find("span", class_="mw-headline")`, mencari semua elemen heading (h2 dan h3) dalam isi artikel yang memiliki `span.mw-headline`, yang merupakan subjudul artikel.
- `file_path = f"{directory}/artikel.txt"`, mengecek apakah file artikel.txt sudah ada. Jika belum, dibuat file kosong.
- `print("Scraping selesai. Data disimpan di folder:", directory)`, menampilkan notifikasi ke terminal bahwa scraping berhasil dilakukan.
- `main_scraper("https://id.wikipedia.org/wiki/Sejarah_Islam", "Hasil")`, menjalankan fungsi scraping dengan: URL Artikel Wikipedia tentang *Sejarah Islam*.

Menjalankan fungsi.py

- `def create_directory(path): if not os.path.exists(path): os.mkdir(path)`, mengecek apakah folder pada path sudah ada dengan `os.path.exists(path)`. Jika belum ada, akan dibuat folder baru dengan `os.mkdir(path)`.
- `def create_new_file(path): f = open(path, 'w', encoding='utf-8') f.write("") f.close()`, membuka file dengan mode 'w' (write), artinya akan menulis dari awal dan menghapus isi lama jika file sudah ada. Menuliskan string kosong "" ke dalam file (untuk memastikan file kosong saat dibuat). Ditutup dengan `f.close()`.
- `def write_to_file(path, text): with open(path, 'a', encoding='utf-8') as f: f.write(text + "\n")`, membuka file dalam mode 'a' (append) menambahkan teks di akhir file tanpa menghapus isi lama. Menulis teks yang diberikan ke dalam file, diikuti newline (`\n`) agar teks baru dimulai di baris berikutnya.

c. Hasil Uji Coba

The screenshot shows a terminal window with the following content:

```
Hasil > artikel.txt
1  |judul Halaman: Sejarah Islam
2
3  Ringkasan Awal:
4  Sejarah Islam meliputi perkembangan politik, sosial, ekonomi, militer, dan budaya peradaban Islam.
5
6  Daftar Subjedul:
7  1. (Tidak ada subjedul ditemukan)
8
9  Sumber: https://id.wikipedia.org/wiki/Sejarah\_Islam
10

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\stikom\smt 4\Praktikum Data Scraping\pertemuan3> & "C:/Program Files/Python312/python.exe" "c:/stikom/smt
raktikum Data Scraping/pertemuan3/main.py"
Scraping selesai. Data disimpan di folder: Hasil
PS C:\stikom\smt 4\Praktikum Data Scraping\pertemuan3>
```

Gambar 3.1 Hasil Scraping dari halaman Wikipedia Sejarah Islam

- Hasil uji coba diatas menunjukkan bahwa program berjalan dengan baik, hanya saja subjedul tidak ditemukan karena kemungkinan struktur halaman tidak memiliki tag `` yang biasa digunakan pada Wikipedia.

d. Analisa Hasil

- Program sudah berfungsi dengan baik untuk scraping judul dan paragraf pertama artikel Wikipedia.
- Hanya saja terdapat kekurangan pada bagian pencarian subjedul karena struktur HTML yang tidak konsisten.
- Perlu dilakukan penyesuaian selector agar lebih fleksibel jika digunakan di berbagai halaman Wikipedia dengan struktur berbeda.

3.4.2 Percobaan Kedua

Pada percobaan kedua ini, mahasiswa diminta mengembangkan program untuk mengekstrak data terstruktur dari halaman web, seperti judul, tanggal posting, ringkasan cerita, dan URL lanjutan. Program ini bertujuan agar hasil scraping lebih informatif dan sesuai dengan format artikel yang umum digunakan. Data yang diperoleh kemudian disimpan dalam file teks dengan format yang rapi dan sistematis, serta disertai sumber URL sebagai dokumentasi.

a. Script / Setting Program

Tugas.py

```
import requests
from bs4 import BeautifulSoup
import fungsi
import os
from datetime import datetime

def scrape_article(url):
    headers = {
        "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)"
    }
    response = requests.get(url, headers=headers)
    if response.status_code != 200:
        print("Gagal mengakses:", url)
        return None

    soup = BeautifulSoup(response.text, "html.parser")
    title_tag = soup.select_one("#firstHeading")
    title = title_tag.text.strip() if title_tag else "-"

    waktu = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

    content_div = soup.select_one("div.mw-parser-output")
    first_para = ""
    if content_div:
        for p in content_div.find_all("p", recursive=False):
            if p.text.strip():
                first_para = p.text.strip()
                break

    subs = []
    if content_div:
        for heading in content_div.find_all(["h2", "h3"]):
            span = heading.find("span", class_="mw-headline")
            if span and span.text.strip():
```

```

        subs.append(span.text.strip())

data = (
    f"Judul           : {title}\n"
    f"Tanggal Scraping: {waktu}\n"
    f"Ringkasan Cerita: {first_para}\n"
    f"URL Lanjutan   : {url}\n"
    + ("Daftar Subjudul:\n" + "\n".join([f"- {s}" for s in subs]) + "\n" if
subs else "") +
    "-"*40 + "\n"
)
return data

def main_scraper_wiki_all(directory):
    fungsi.create_directory(directory)
    kategori_url = "https://en.wikipedia.org/wiki/Category:Indonesian_folklore"

    headers = {
        "User-Agent": "Mozilla/5.0"
    }

    response = requests.get(kategori_url, headers=headers)
    if response.status_code != 200:
        print("Gagal mengakses halaman kategori.")
        return

    soup = BeautifulSoup(response.text, "html.parser")

    # Ambil semua link artikel dari daftar kategori
    links = soup.select("div.mw-category a")
    print(f"Ditemukan {len(links)} artikel dalam kategori.")

    for link in links:
        relative_url = link.get("href")
        if relative_url is None or not relative_url.startswith("/wiki/"):
            continue

        full_url = "https://en.wikipedia.org" + relative_url
        data = scrape_article(full_url)
        if data:
            print(f"Menyimpan: {link.text}")
            file_path = os.path.join(directory, "artikel_wiki.txt")
            if not fungsi.does_file_exist(file_path):
                fungsi.create_new_file(file_path)
            fungsi.write_to_file(file_path, data)

    print("\n✓ Scraping semua artikel selesai. Cek folder:", directory)

if __name__ == "__main__":
    main_scraper_wiki_all("WikiScraping")

```

b. Langkah Uji Coba

- **kategori_url= https://en.wikipedia.org/wiki/Category:Indonesian_folklore,** mengakses halaman kategori Wikipedia Indonesian folklore, Mengambil semua link artikel dalam kategori tersebut, Membuka setiap artikel satu per satu.
- Semua hasil disimpan ke dalam file teks **artikel_wiki.txt** dalam folder **WikiScraping**.
- **scrape_article(url)**, mengambil satu artikel Wikipedia berdasarkan URL, format data disusun menjadi teks rapi.
- **(div.mw-parser-output p)**, memproses paragraph pertama.
- **main_scraper_wiki_all(directory)**, membuat folder WikiScraping jika belum ada.
- Mengambil semua **elemen <a>** dari bagian **div.mw-category**.
- Melakukan perulangan: mengambil atribut **href** dari link, menggabungkan menjadi URL lengkap, memanggil **scrape_article()** untuk diambil datanya, menyimpan ke file **article_wiki.txt**.

c. Hasil Uji Coba

```
WikiScraping > artikel_wiki.txt
1 Judul : UFO sightings in Indonesia
2 Tanggal Scraping: 2025-06-27 20:19:46
3 Ringkasan Cerita: The following is the list of UFO sightings reported in Indonesia.
4 URL Lanjutan : https://en.wikipedia.org/wiki/UFO\_sightings\_in\_Indonesia
5 -----
6
7 Judul : Category:Balinese folklore
8 Tanggal Scraping: 2025-06-27 20:26:04
9 Ringkasan Cerita: Balinese folklore is folklore created by Balinese people and influenced by Balinese culture.
10 URL Lanjutan : https://en.wikipedia.org/wiki/Category:Balinese\_folklore
11 -----
12
13 Judul : Category:Indonesian fairy tales
14 Tanggal Scraping: 2025-06-27 20:26:04
15 Ringkasan Cerita:
16 URL Lanjutan : https://en.wikipedia.org/wiki/Category:Indonesian\_fairy\_tales
17 -----
18
19 Judul : Category:Indonesian folklorists
20 Tanggal Scraping: 2025-06-27 20:26:05
21 Ringkasan Cerita:
22 URL Lanjutan : https://en.wikipedia.org/wiki/Category:Indonesian\_folklorists
23 -----
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Menyimpan: Satrio Piningit
Menyimpan: Satu Suro
Menyimpan: Siamang putih
Menyimpan: Singo Ulung
Menyimpan: Totok Kerot
Menyimpan: UFO sightings in Indonesia

Scraping semua artikel selesai. cek folder: WikiScraping
PS C:\stikom\smt 4\Praktikum Data Scraping\pertemuan3>

Gambar 3.2 Hasil Scraping dari halaman Wikipedia

- Program berhasil scraping banyak artikel dari halaman Wikipedia.
- Namun, Wikipedia menyimpan halaman kategori halaman khusus, bukan artikel biasa. Karena tidak cocok untuk di scrape seperti artikel biasa.
- Halaman kategori umumnya tidak memiliki paragraph utama, sehingga beberapa ringkasan cerita kosong.

d. Analisa Hasil

- Program berhasil scraping artikel dari kategori Indonesia folklore di Wikipedia.
- Data yang diambil meliputi judul, tanggal, ringkasan, subjudul, dan URL.
- Looping artikel dan proses penyimpanan berjalan lancar.
- Namun, ada beberapa ringkasan kosong pada halaman kategori karena tidak memiliki paragraf utama.

3.5 Kesimpulan

3.5.1 Kesimpulan Percobaan 1

Mahasiswa telah berhasil melakukan praktikum pada artikel Sejarah Islam, menunjukkan bahwa program scraping berhasil dijalankan dengan baik untuk mengambil data dari halaman Wikipedia. Informasi yang diperoleh meliputi judul halaman, ringkasan paragraf pertama, daftar subjudul, serta URL sumber artikel, program juga mencatat waktu scraping sebagai bagian dari dokumentasi data. Percobaan ini membuktikan bahwa integrasi antara modul requests, BeautifulSoup, dan fungsi bantu dari file fungsi.py telah berjalan efektif untuk melakukan ekstraksi data dari situs web berbasis HTML secara otomatis.

3.5.2 Kesimpulan Percobaan 2

Mahasiswa telah berhasil melakukan praktikum pada artikel Wikipedia kategori Indonesian folklore yang menunjukkan bahwa program mampu melakukan scraping terhadap banyak artikel dalam satu halaman kategori. Program berhasil mengambil data berupa judul, ringkasan paragraf pertama, subjudul, tanggal scraping, dan URL artikel secara otomatis, lalu menyimpannya ke dalam file teks. Hasil yang diperoleh cukup lengkap dan sesuai format, meskipun masih ditemukan beberapa halaman kategori yang ikut terbaca sebagai artikel, sehingga bagian ringkasannya kosong. Hal ini menunjukkan bahwa program sudah berjalan dengan baik dalam melakukan scraping massal, namun masih perlu penyempurnaan dalam penyaringan jenis halaman agar hanya artikel yang relevan saja yang diambil.

Mengetahui:

Dosen Pengampu Mata Kuliah

Arif Hadi Sumitro , M.Kom

NIKP. xxx

DAFTAR PUSTAKA

1. Python for Everybody – www.py4e.com
2. *Mitchell, R. (2018). Web Scraping with Python (Edisi ke-2). O'Reilly Media.*
3. Matthes, E. (2019). Python Crash Course. No Starch Press.

BAB 4

4.1 Capaian Praktikum Pertemuan 4

Pada praktikum pertemuan ke-4 ini mahasiswa diharapkan mampu memahami ketentuan dan etika dalam melakukan scraping, khususnya terkait pengambilan gambar dari internet. Mahasiswa juga diharapkan dapat mengimplementasikan penggunaan library BeautifulSoup dan requests untuk mengambil URL gambar dari sebuah website, serta mampu mengambil file gambar tersebut dan menyimpannya ke dalam direktori yang telah ditentukan secara tepat.

4.2 Indikator Capaian

- Mahasiswa tidak melanggar aturan norma dan etika dalam pengambilan gambar.
- Mahasiswa melakukan scraping URL gambar dengan menggunakan BeautifulSoup dan Request.
- Mahasiswa melakukan penyimpanan gambar dengan menggunakan BeautifulSoup dan Request.
- Mahasiswa melakukan penyimpanan file gambar pada direktori yang ditentukan.
- Mahasiswa melakukan analisa website yang akan di scraping.

4.3 Landasan Teori

Menurut Mitchell (2018) dalam bukunya “*Web Scraping with Python*”, web scraping adalah teknik otomatisasi untuk mengekstrak informasi dari situs web menggunakan program. Proses ini mencakup pengambilan data melalui permintaan

HTTP dan parsing HTML untuk mengambil elemen spesifik seperti teks, gambar, atau link.

Menurut Richardson, L. (2023), BeautifulSoup adalah library Python yang digunakan untuk memarsing dokumen HTML dan XML. Menurut dokumentasi resmi, BeautifulSoup memungkinkan navigasi struktur HTML secara mudah dan fleksibel, serta sering digunakan untuk mencari elemen dengan tag tertentu.

Menurut Reitz, K. (2021), Requests merupakan library HTTP untuk Python yang digunakan untuk mengirim permintaan HTTP ke server. Menurut Reitz (2021), requests adalah library paling sederhana dan elegan untuk melakukan komunikasi client-server dalam Python.

Menurut Auerbach (2012), scraping harus memperhatikan aspek legalitas dan etika, terutama ketika mengambil data atau konten dari situs web yang dilindungi hak cipta. Scraper harus membaca robots.txt dan memperhatikan izin penggunaan data.

4.4 Pelaksanaan Praktikum

4.4.1 Percobaan Pertama

Pada percobaan pertama ini, dilakukan untuk menguji kemampuan program dalam mengambil gambar dari situs dongeng ceritarakyat.com dengan memanfaatkan atribut data-lazy-src, data-src, dan src pada tag . Program berhasil mendeteksi dan mengumpulkan URL gambar yang valid, lalu menyimpannya ke dalam folder bernama gambar_dongeng yang dibuat secara otomatis di direktori kerja. Hasil percobaan menunjukkan bahwa program dapat berjalan dengan baik dalam mengenali elemen gambar dari berbagai atribut dan menyimpan file secara terstruktur tanpa error.

a. Script / Setting Program

Main.py

```
import os

import requests

from bs4 import BeautifulSoup


# URL target

url = 'https://dongengceritarakyat.com/'

page = requests.get(url)

soup = BeautifulSoup(page.content, 'html.parser')

# Ambil semua gambar dari elemen <img>

images = []

for img in soup.find_all('img'):

    img_url = img.get('data-lazy-src') or img.get('data-src') or img.get('src')

    if img_url and img_url.endswith(('.jpg', '.png', '.gif')):

        images.append(img_url)

print(f"Jumlah gambar ditemukan: {len(images)}")

print("Contoh URL gambar:", images[:3])

#  Ganti folder penyimpanan di sini

save_path = os.path.join(os.getcwd(), 'gambar_dongeng')

if not os.path.exists(save_path):

    os.makedirs(save_path)

# Download dan simpan gambar

for img in images:
```

```

try:

    response = requests.get(img)

    filename = os.path.basename(img)

    filepath = os.path.join(save_path, filename)

    with open(filepath, 'wb') as f:

        f.write(response.content)

    print(f"{filename} berhasil disimpan di {save_path}")

except Exception as e:

    print(f"Gagal menyimpan {img}: {e}")

```

b. Langkah Uji Coba

- **BeautifulSoup**: Untuk mem-parsing dan mengekstrak elemen HTML.
- **requests**: Untuk mengirim permintaan HTTP (GET) ke situs web.
- **url = 'https://dongengceritarakyat.com/'**, menyimpan alamat situs target.
- **requests.get(url)**: mengambil konten halaman web.
- **soup = BeautifulSoup(page.content, 'html.parser')**, membaca isi HTML dari halaman yang diambil agar bisa diolah.
- **for img in soup.find_all('img')**: menemukan semua tag dalam halaman.
- **img.get(...)**: mengambil nilai atribut gambar (**data-lazy-src**, **data-src**, atau **src**).
- **endswith(...)**: memastikan hanya gambar dengan format .jpg, .png, atau .gif yang diambil.
- **images.append(full_url)**: Menyimpan URL gambar ke dalam list images.
- **print(f"Jumlah gambar ditemukan: {len(images)}")**, menampilkan jumlah gambar yang ditemukan.

- `print("Contoh URL gambar:", images[:3])`, menampilkan 3 URL gambar pertama sebagai contoh hasil scraping.
- `os.getcwd()`: mengambil direktori saat ini.
- `os.path.join(...)`: menggabungkan path direktori dengan nama folder **gambar_dongeng**.
- `requests.get(img)`: mengambil konten gambar dari URL.
- `os.path.basename(img)`: mengambil nama file dari URL gambar.
- `open(..., 'wb')`: membuka file dalam mode tulis biner (wb) untuk menyimpan data gambar.
- `f.write(...)`: menulis isi gambar ke file.
- `print(...)`: menampilkan notifikasi bahwa gambar telah berhasil disimpan.
- `print(f"Gagal menyimpan {img}: {e}")`, menampilkan pesan error agar pengguna tahu gambar mana yang gagal diproses.

c. Hasil Uji Coba

```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
PS C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4 & "C:/...
Jumlah gambar ditemukan: 0
Contoh URL gambar: []
PS C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4
  
```

Gambar 4.1 Hasil Scraping dari halaman dongeng cerita rakyat

- Program berhasil dijalankan, namun tidak menemukan gambar karena struktur HTML halaman dongengceritarakyat.com tidak sesuai dengan metode scraping yang digunakan. Diperlukan analisa lebih lanjut terhadap struktur `` atau beralih ke situs yang lebih terbuka untuk scraping.

d. Analisa Hasil

Hasil uji coba menunjukkan bahwa meskipun program berhasil mengakses situs dongengceritarakyat.com, tidak ada gambar yang berhasil ditemukan atau disimpan. Hal ini disebabkan karena elemen pada halaman tersebut kemungkinan tidak menggunakan atribut standar seperti src, data-src, atau datalazy-src, atau gambar dimuat secara dinamis menggunakan JavaScript yang tidak dapat ditangani oleh requests dan BeautifulSoup.

4.4.2 Percobaan Kedua

Pada percobaan kedua ini, mahasiswa diperintahkan untuk melakukan percobaan scraping gambar dari website menggunakan library requests dan BeautifulSoup, di mana program mengambil URL gambar dari elemen dan menyimpannya secara otomatis ke dalam folder yang telah ditentukan.

a. Script / Setting Program

Tugas.py

```
import os
import requests
from bs4 import BeautifulSoup
from urllib.parse import urljoin

# Ganti dengan URL baru
url = 'https://books.toscrape.com/'
page = requests.get(url)
soup = BeautifulSoup(page.content, 'html.parser')

images = []
for img in soup.find_all('img'):
    img_url = img.get('src')
    if img_url and img_url.endswith('.jpg', '.png', '.gif'):
        # Gabungkan URL relatif jadi absolut
        full_url = urljoin(url, img_url)
        images.append(full_url)

print(f"Jumlah gambar ditemukan: {len(images)}")
print("Contoh URL gambar:", images[:3])
```

```

# Simpan ke folder hasil_gambar
save_path = os.path.join(os.getcwd(), 'hasil_gambar')
if not os.path.exists(save_path):
    os.makedirs(save_path)

for img in images:
    try:
        response = requests.get(img)
        filename = os.path.basename(img)
        filepath = os.path.join(save_path, filename)

        with open(filepath, 'wb') as f:
            f.write(response.content)

        print(f"{filename} berhasil disimpan di {save_path}")
    except Exception as e:
        print(f"Gagal menyimpan {img}: {e}")

```

a. Langkah Uji Coba

- **BeautifulSoup**: Untuk mem-parsing dan mengekstrak elemen HTML.
- **requests**: Untuk mengirim permintaan HTTP (GET) ke situs web.
- **from urllib.parse import urljoin**, untuk menggabungkan URL relatif menjadi URL absolut.
- **url = 'https://books.toscrape.com/'**, situs web yang dituju.
- **BeautifulSoup(page.content, 'html.parser')**: Mem-parsing HTML dari halaman untuk diolah.
- **soup.find_all('img')**: Menemukan semua elemen di halaman.
- **img.get('src')**: Mengambil atribut src dari tag , yaitu URL gambar.
- **endswith(...)**: Memastikan hanya file gambar yang diambil.
- **urljoin(url, img_url)**: Menggabungkan URL relatif (img_url) dengan URL utama (url) agar menjadi URL lengkap.
- **images.append(full_url)**: Menyimpan URL gambar ke dalam list images.

- `print(f"Jumlah gambar ditemukan: {len(images)}")`, menampilkan jumlah gambar yang ditemukan.
 - `print("Contoh URL gambar:", images[:3])`, menampilkan 3 URL gambar pertama sebagai contoh hasil scraping.
 - `os.getcwd()`: mengambil direktori saat ini.
 - `os.path.join(...)`: menggabungkan path folder hasil gambar.
 - `requests.get(img)`: mengambil konten gambar dari URL.

b. Hasil Uji Coba

```
verte... PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
hasil_gambar
0bbcd0a6f4bcd81cc...
2cdad67c44b002e7...
3d54940e57e662c4...
3ee9f99c9d9aef346...
5b88c52633f53acf...
09a3aeaf557576e1...
27a53d0b55bd88...
81ca4973364e17d...
94b1b8b244bce967...
260c6ae16bce31c8f...
1048f63d3b5061cd...
3251cf3a3412f53f3...
54607fe8945897cd...
66883b91f6804b23...
68339b4c9b03426...
922749a5b7c251fea...
553310a7162dfbc2...
5846057fe28022268...
bea5697f2534a2f86...
bef44da28c98f905a...
PS C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4> & "C:/Program Files/Python312/python.exe" "c:/stikom/smt 4\Praktikum Data Scraping\pertemuan4\hasil_gambar.py"
Jumlah gambar ditemukan: 20
Contoh URL gambar: [ 'https://books.toscrape.com/media/cache/2c/da/2cdad67c44b002e7ead0cc35693c0e8b.jpg', 'https://books.toscrape.com/media/cache/f4/1c/f41c.jpg', 'https://books.toscrape.com/media/cache/3e/ef/3eeff99c9d9aef34639f510662022830.jpg' ]
2cdad67c44b002e7ead0cc35693c0e8b.jpg berhasil disimpan di C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4\hasil_gambar
260c6ae16bce31c8f895dadd9f4a1c.jpg berhasil disimpan di C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4\hasil_gambar
3ee9f99c9d9aef34639f510662022830.jpg berhasil disimpan di C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4\hasil_gambar
3251cf3a3412f53f39e42c2134093.jpg berhasil disimpan di C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4\hasil_gambar
bea5697f2534a2f86aef27b5a8c12a6.jpg berhasil disimpan di C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4\hasil_gambar
68339b4c9b034267e1da611ab3f48.jpg berhasil disimpan di C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4\hasil_gambar
922749a5b7c251fea59a2b8a78275ab4.jpg berhasil disimpan di C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4\hasil_gambar
3d54940e57e662c4dd1f3f0c78c64.jpg berhasil disimpan di C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4\hasil_gambar
66883b91f6884f2323c8369331cb7dd1.jpg berhasil disimpan di C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4\hasil_gambar
5846057fe82022268153beff6d352b06c.jpg berhasil disimpan di C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4\hasil_gambar
bef44da28c98f905a3ebec087b8e8530.jpg berhasil disimpan di C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4\hasil_gambar
94b1b8b244bce9677c2f29ccc890d4d2.jpg berhasil disimpan di C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4\hasil_gambar
81ca4973364e17d01f217e1188253d5e.jpg berhasil disimpan di C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4\hasil_gambar
54607fe8945897cdcce00441b3b10b6.jpg berhasil disimpan di C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4\hasil_gambar
553310a7162dfbc26d19a84da0df9e1.jpg berhasil disimpan di C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4\hasil_gambar
09a3aeaf557576e1a85b7afea8c7b.jpg berhasil disimpan di C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4\hasil_gambar
94b1b8b244bce9677c2f29ccc890d4d2.jpg berhasil disimpan di C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4\hasil_gambar
81ca4973364e17d01f217e1188253d5e.jpg berhasil disimpan di C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4\hasil_gambar
54607fe8945897cdcce00441b3b10b6.jpg berhasil disimpan di C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4\hasil_gambar
553310a7162dfbc26d19a84da0df9e1.jpg berhasil disimpan di C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4\hasil_gambar
94b1b8b244bce9677c2f29ccc890d4d2.jpg berhasil disimpan di C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4\hasil_gambar
81ca4973364e17d01f217e1188253d5e.jpg berhasil disimpan di C:\stikom\smt 4\Praktikum Data Scraping\pertemuan4\hasil_gambar
```

Gambar 4.2 Hasil Scraping dari halaman books.toscrape

- Hasil scraping berjalan dengan sukses, di mana website target memberikan respon yang sesuai sehingga seluruh gambar berhasil dikumpulkan dan disimpan ke dalam folder yang telah ditentukan. Program ini menggunakan metode urljoin() untuk menggabungkan URL relatif menjadi URL absolut, sehingga gambar dapat diakses dan diunduh dengan benar. Selain itu, program juga menyaring file yang diambil berdasarkan ekstensi gambar seperti .jpg, .png, dan .gif, sehingga hanya file yang valid yang diproses dan disimpan.

d. Analisa Hasil

Program Uji coba scraping gambar berjalan dengan sangat baik. Semua komponen mulai dari akses web, parsing HTML, pengambilan URL, hingga penyimpanan file telah diimplementasikan dengan benar. Folder hasil_gambar terisi dengan file gambar yang sesuai, menandakan bahwa program dapat digunakan sebagai dasar scraping gambar dari situs lain yang serupa.

4.5 Kesimpulan

4.5.1 Kesimpulan Percobaan 1

Mahasiswa telah melakukan percobaan, bahwa proses scraping gambar pada situs dongengceritarakyat.com belum berhasil mendeteksi dan menyimpan gambar karena tidak ditemukan URL gambar yang sesuai dalam struktur HTML-nya. Hal ini menunjukkan bahwa tidak semua situs dapat di-scrape dengan metode HTML statis menggunakan requests dan BeautifulSoup, terutama jika elemen gambar dimuat secara dinamis melalui JavaScript atau menggunakan atribut khusus yang tidak umum. Oleh karena itu, diperlukan analisa lebih dalam terhadap struktur website atau penggunaan tools scraping yang mendukung pemrosesan dinamis seperti Selenium.

4.5.2 Kesimpulan Percobaan 2

Mahasiswa telah berhasil melakukan percobaan scraping menggunakan situs books.toscrape.com yang menunjukkan bahwa program berhasil dijalankan dengan baik dan sesuai harapan. Program mampu mengakses situs, mengambil elemen , mengekstrak URL gambar menggunakan atribut src, dan menggabungkannya menjadi URL absolut dengan urljoin. Seluruh gambar yang ditemukan kemudian berhasil disimpan ke dalam folder hasil_gambar tanpa error. Percobaan ini

membuktikan bahwa teknik scraping berbasis HTML statis dengan requests dan BeautifulSoup sangat efektif jika digunakan pada situs yang struktur HTML-nya sederhana dan tidak memiliki pembatasan scraping.

Mengetahui:

Dosen Pengampu Mata Kuliah

Arif Hadi Sumitro , M.Kom

NIKP. Xxx

DAFTAR PUSTAKA

1. Mitchell, R. (2018). *Web Scraping with Python: Collecting Data from the Modern Web* (2nd ed.). O'Reilly Media.
2. Richardson, L. (2023). *Beautiful Soup Documentation*.
<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>
3. Reitz, K. (2021). *Requests: HTTP for Humans*. <https://docs.python-requests.org>
4. Auerbach, D. (2012). *Scraping By: How Web Scrapers Help and Hurt*. Slate Technology. <https://slate.com/technology/2012/03/web-scraping-legal-issues.html>

BAB 5

5.1 Capaian Praktikum Pertemuan 5

Pada praktikum pertemuan ke-5 ini mahasiswa diharapkan mampu memahami penggunaan function dalam Python yang digunakan untuk melakukan proses scraping data dari web. Selain itu, mahasiswa juga diharapkan dapat memahami cara kerja modul requests dan BeautifulSoup dalam menentukan serta mengekstrak sub link dari sebuah halaman utama. Dengan pemahaman tersebut, mahasiswa mampu melakukan proses scraping secara menyeluruh, termasuk mengambil data dari halaman utama maupun dari sub link yang menjadi bagian dari struktur data web tersebut.

5.2 Indikator Capaian

- Mahasiswa melakukan pemanggilan function yang sudah dibuat.
- Mahasiswa membuat function baru untuk melakukan sub scraping.
- Mahasiswa melakukan scraping secara menyeluruh sari sub URL yang ditemukan.

5.3 Landasan Teori

Menurut Mitchell (2018) dalam bukunya “*Web Scraping with Python*”, web scraping adalah teknik otomatisasi untuk mengekstrak informasi dari situs web menggunakan program. Proses ini mencakup pengambilan data melalui permintaan HTTP dan parsing HTML untuk mengambil elemen spesifik seperti teks, gambar, atau link.

Menurut Richardson, L. (2023), BeautifulSoup adalah library Python yang digunakan untuk memarsing dokumen HTML dan XML. Menurut dokumentasi resmi, BeautifulSoup memungkinkan navigasi struktur HTML secara mudah dan fleksibel, serta sering digunakan untuk mencari elemen dengan tag tertentu.

Menurut Reitz (2015), Library requests pada Python digunakan untuk melakukan permintaan HTTP ke suatu halaman web, requests menyederhanakan proses pengambilan data dari web karena API-nya bersifat human-friendly dan fleksibel.

Dalam proses web scraping, terkadang informasi penting tersebar di halaman-halaman sub-link. Menurut Prasad et al. (2018), pengambilan data dari sub-link merupakan bagian penting dari teknik scraping yang mendalam (deep scraping), di mana program menavigasi dari halaman utama ke halaman-halaman detail untuk mendapatkan informasi yang lebih kaya.

5.4 Pelaksanaan Praktikum

5.4.1 Percobaan Pertama

Pada percobaan pertama mahasiswa melakukan scraping dari situs quotes.toscrape.com, di mana program mengekstrak kutipan beserta nama penulis dari halaman utama, lalu secara otomatis mengikuti tautan detail ke halaman profil masing-masing penulis. Dari halaman detail tersebut, scraper mengambil informasi tambahan seperti nama lengkap, tanggal lahir, tempat lahir, dan deskripsi singkat penulis, yang kemudian disimpan dalam bentuk file teks per penulis di dalam folder Hasil.

a. Script / Setting Program

scraper.py

```
from bs4 import BeautifulSoup
import requests
import fungsi
BASE_URL = "https://quotes.toscrape.com"
def get_details(relative_url, directory):
    url = BASE_URL + relative_url
    response = requests.get(url)
    soup = BeautifulSoup(response.text, "html.parser")
    nama = soup.find("h3", class_="author-title").text.strip()
    tanggal_lahir = soup.find("span", class_="author-born-date").text.strip()
    tempat_lahir = soup.find("span", class_="author-born-location").text.strip()
    deskripsi = soup.find("div", class_="author-description").text.strip()
    isi = f"Nama: {nama}\nTanggal Lahir: {tanggal_lahir}\nTempat Lahir: {tempat_lahir}\n\nDeskripsi:\n{deskripsi}\n\n"
    file_path = f"{directory}/{nama.replace(' ', '_')}.txt"
    with open(file_path, 'w', encoding='utf-8') as f:
        f.write(isi)
def main_scraper(url, directory):
    fungsi.create_directory(directory)
    response = requests.get(url)
    soup = BeautifulSoup(response.text, "html.parser")
    quotes = soup.find_all("div", class_="quote")
    for quote in quotes:
        teks = quote.find("span", class_="text").text.strip()
        author = quote.find("small", class_="author").text.strip()
        detail_link = quote.find("a")["href"]
        print(f"Quote: {teks}")
        print(f"Author: {author}")
```

```

print(f"Detail URL: {BASE_URL + detail_link}\n")

fungsi.write_to_file(f"{directory}/quotes.txt", f"{teks} - {author}\n")

get_details(detail_link, directory)

# Jalankan scraper

main_scraper("https://quotes.toscrape.com", "Hasil")

```

b. Langkah Uji Coba

- **BeautifulSoup:** Untuk mem-parsing dan mengekstrak elemen HTML.
- **requests:** Untuk mengirim permintaan HTTP (GET) ke situs web.
- **BASE_URL = https://quotes.toscrape.com,** menyimpan alamat situs target.
- **def get_details(relative_url, directory),** mengambil detail penulis dari halaman profil mereka.
- **url = BASE_URL + relative_url,** pastikan hasil gabungan benar.
- **response = requests.get(url),** pastikan respon HTTP = 200.
- **soup = BeautifulSoup(response.text, "html.parser"),** pastikan HTML berhasil diparsing.
- **isi = f"Nama: {nama}\nTanggal Lahir: {tanggal_lahir}\nTempat Lahir: {tempat_lahir}\n\nDeskripsi:\n{deskripsi}\n\n",** pastikan format teks benar sebelum ditulis ke file.
- **file_path = f"{directory}/{nama.replace(' ', '_)}.txt",** pastikan penamaan file benar.
- **with open(file_path, 'w', encoding='utf-8') as f: f.write(isi),** cek folder Hasil/ apakah file berhasil ditulis dan isinya sesuai.
- **def main_scraper(url, directory),** pastikan fungsi dipanggil di bagian bawah program.
- **fungsi.create_directory(directory),** pastikan folder Hasil/ dibuat jika belum ada.

- `quotes = soup.find_all("div", class_="quote")`, pastikan elemen quote ditemukan.
- `for quote in quotes`, looping akan dilakukan pada setiap kutipan di halaman.
- `fungsi.write_to_file(f"{directory}/quotes.txt", f"{teks} - {author}\n")`, buka file quotes.txt di folder Hasil/ dan pastikan kutipan tersimpan dengan benar.
- `get_details(detail_link, directory)`, pastikan detail author diambil dan disimpan ke file masing-masing nama author.
- `main_scraper("https://quotes.toscrape.com", "Hasil")`, jalankan program dengan python nama_file.py dan pastikan folder Hasil/ berisi: quotes.txt → berisi daftar kutipan, file per author → Albert_Einstein.txt, Jane_Austen.txt, dll.

c. Hasil Uji Coba

```

Albert_Einstein.txt  André_Gide.txt
Hasil >  Albert_Einstein.txt
1 Nama: Albert Einstein
2 Tanggal Lahir: March 14, 1879
3 Tempat Lahir: in Ulm, Germany
4
5 Deskripsi:
6 In 1879, Albert Einstein was born in Ulm, Germany. He completed
7
8

Hasil >  André_Gide.txt
1 Nama: André Gide
2 Tanggal Lahir: November 22, 1869
3 Tempat Lahir: in Paris, France
4
5 Deskripsi:
6 André Paul Guillaume Gide was a French author and winner of the Nobel Prize in Literature in 1947. He is best known for his novels "La Révolte des éléphants" (1891) and "Le Tragique" (1892). He also wrote plays, essays, and travelogues. His work often explored themes of sexuality, spirituality, and social inequality.
7
8

```

Gambar 5.1 Hasil Scraping dari halaman Quotes to Scrape

- Program berhasil dijalankan, fungsi `get_details()` dan `main_scraper()` berjalan sesuai harapan. Data dari halaman detail (sub-link author) berhasil diambil dan disimpan. Program berhasil membuat file .txt per penulis dengan konten terstruktur. Tidak ditemukan error parsing, encoding, maupun file write.

d. Analisa Hasil

Hasil uji coba menunjukkan bahwa Struktur Data tersimpan rapi. Setiap informasi penulis seperti nama, tanggal lahir, tempat lahir, dan deskripsi berhasil

diambil dan disimpan ke dalam file .txt secara terstruktur dan sesuai format yang diinginkan.

5.4.2 Percobaan Kedua

Pada percobaan kedua dilakukan dengan mengubah URL target menjadi halaman khusus tag tertentu, yaitu "https://quotes.toscrape.com/tag/inspirational/page/1/", serta mengganti folder penyimpanan hasil menjadi "DataScraping". Pada percobaan ini, program tidak hanya mengambil kutipan dan informasi penulis seperti sebelumnya, tetapi juga melakukan penelusuran berdasarkan tag dari halaman detail penulis untuk mendapatkan kutipan tambahan yang relevan. Dengan demikian, percobaan ini telah berhasil menampilkan scraping hingga dua tingkat kedalaman sub-URL dan menyimpan data secara otomatis dalam folder baru yang telah ditentukan.

a. Script / Setting Program

Main.py

```
from bs4 import BeautifulSoup
import requests
import fungsi

BASE_URL = "https://quotes.toscrape.com"

def scrape_by_tag(tag_url, directory, author_name):
    """Level 3: scrape kutipan dari halaman tag penulis"""
    url = BASE_URL + tag_url
    response = requests.get(url)
    soup = BeautifulSoup(response.text, "html.parser")
    quotes = soup.find_all("div", class_="quote")

    fungsi.write_to_file(f"{directory}/{author_name.replace(' ', '_')}.txt",
    "\nKutipan lain berdasarkan tag:\n")

    for q in quotes:
        text = q.find("span", class_="text").text.strip()
        fungsi.write_to_file(f"{directory}/{author_name.replace(' ', '_')}.txt",
        f"- {text}")

def get_details(relative_url, directory):
    """Level 2: scrape data author dan tag"""
    url = BASE_URL + relative_url
```

```

response = requests.get(url)
soup = BeautifulSoup(response.text, "html.parser")

nama = soup.find("h3", class_="author-title").text.strip()
tanggal_lahir = soup.find("span", class_="author-born-date").text.strip()
tempat_lahir = soup.find("span", class_="author-born-location").text.strip()
deskripsi = soup.find("div", class_="author-description").text.strip()

isi = f"Nama: {nama}\nTanggal Lahir: {tanggal_lahir}\nTempat Lahir: {tempat_lahir}\n\nDeskripsi:\n{deskripsi}\n"
file_path = f"{directory}/{nama.replace(' ', '_)}.txt"

with open(file_path, 'w', encoding='utf-8') as f:
    f.write(isi)

tag_section = soup.find_all("a", class_="tag")
if tag_section:
    tag_href = tag_section[0]["href"]
    scrape_by_tag(tag_href, directory, nama)

def main_scraper(url, directory):
    """Level 1: scrape kutipan dan link penulis"""
    fungsi.create_directory(directory)
    response = requests.get(url)
    soup = BeautifulSoup(response.text, "html.parser")
    quotes = soup.find_all("div", class_="quote")

    for quote in quotes:
        teks = quote.find("span", class_="text").text.strip()
        author = quote.find("small", class_="author").text.strip()
        detail_link = quote.find("a")["href"]

        print(f"Quote: {teks}")
        print(f"Author: {author}")
        print(f"Detail URL: {BASE_URL + detail_link}\n")

        fungsi.write_to_file(f"{directory}/quotes.txt", f"{teks} - {author}\n")

        get_details(detail_link, directory)

# Jalankan scraper dengan URL baru dan folder penyimpanan baru
main_scraper("https://quotes.toscrape.com/tag/inspirational/page/1/",
            "DataScraping")

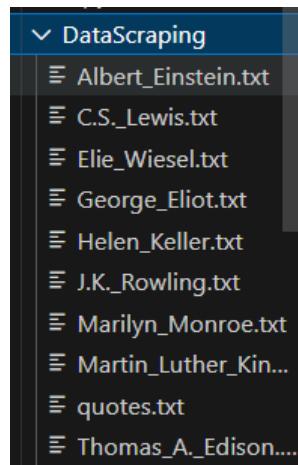
```

b. Langkah Uji Coba

- **BeautifulSoup:** Untuk mem-parsing dan mengekstrak elemen HTML.
- **requests:** Untuk mengirim permintaan HTTP (GET) ke situs web.
- **url = 'https://books.toscrape.com/'**, situs web yang dituju.
- **def scrape_by_tag(tag_url, directory, author_name)**, fungsi dipanggil dari `get_details()`, pastikan parameter valid.

- `url = BASE_URL + tag_url`, pastikan hasil gabungan URL benar.
- `quotes = soup.find_all("div", class_="quote")`, cek `len(quotes)` untuk memastikan kutipan ditemukan di halaman tag.
- `fungsi.write_to_file(f"{directory}/{author_name.replace(' ', '_)}.txt", "\nKutipan lain berdasarkan tag:\n")`, lihat file output apakah bagian heading kutipan dari tag ditambahkan.
- `def get_details(relative_url, directory)`, fungsi dipanggil dari `main_scraper()`, parameter `relative_url` valid.
- `isi = f"Nama: {nama}\nTanggal Lahir: {tanggal_lahir}\nTempat Lahir: {tempat_lahir}\n\nDeskripsi:\n{deskripsi}\n"`, cetak `isi` untuk memastikan format teks sesuai.
- `file_path = f"{directory}/{nama.replace(' ', '_)}.txt"`, pastikan nama file dihasilkan dengan benar tanpa spasi.
- `with open(file_path, 'w', encoding='utf-8') as f: f.write(isi)`, cek apakah file penulis berhasil dibuat dan isinya lengkap.
- `def main_scraper(url, directory)`, fungsi dipanggil di akhir kode, url dan directory sudah ditentukan.
- `fungsi.create_directory(directory)`, periksa apakah folder DataScraping/ otomatis dibuat jika belum ada.

c. Hasil Uji Coba



```

PS c:\stikom\smt 4\Praktikum Data Scraping> & "c:/Program Files/Python312/python.exe" "c:/stikom/smt 4/Praktikum Data Scraping/pertemuan5.py"
Quote: "There are only two ways to live your life. One is as though nothing is a miracle. The other is as though everything is a miracle."
Author: Albert Einstein
Detail URL: https://quotes.toscrape.com/author/Albert-Einstein

Quote: "Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous than absolutely boring."
Author: Marilyn Monroe
Detail URL: https://quotes.toscrape.com/author/Marilyn-Monroe

Quote: "I have not failed. I've just found 10,000 ways that won't work."
Author: Thomas A. Edison
Detail URL: https://quotes.toscrape.com/author/Thomas-A-Edison

Quote: "This life is what you make it. No matter what, you're going to mess up sometimes, it's a universal truth. But the good part is you fix it up. Girls will be your friends - they'll act like it anyway. But just remember, some come, some go. The ones that stay with you through thick and thin. Don't let go of them. Also remember, sisters make the best friends in the world. As for lovers, well, they'll come and go too. And basically pretty much all of them are going to break your heart, but you can't give up because if you give up, you'll never find your soulmate. You whole and that goes for everything. Just because you fail once, doesn't mean you're gonna fail at everything. Keep trying, hold on, and most importantly, keep smiling, because if you don't, then who will, sweetie? So keep your head high, keep your chin up, and most importantly, keep smiling, because o much to smile about."
Author: Marilyn Monroe
Detail URL: https://quotes.toscrape.com/author/Marilyn-Monroe

Quote: "The opposite of love is not hate, it's indifference. The opposite of art is not ugliness, it's indifference. The opposite of faith is not doubt, it's apathy. The opposite of life is not death, it's indifference."
Author: Elie Wiesel
Detail URL: https://quotes.toscrape.com/author/Elie-Wiesel

```

Gambar 5.2 Hasil Scraping dari halaman books.toscrape

- Hasil scraping menunjukkan bahwa program berhasil menampilkan kutipan, nama penulis, dan link detail dari halaman tag inspirational. Data ditampilkan rapi di terminal dan menunjukkan bahwa proses scraping level pertama berjalan dengan baik tanpa error.

d. Analisa Hasil

Program uji coba scraping gambar berjalan dengan sangat baik. Semua komponen mulai dari akses web, parsing HTML, pengambilan URL, hingga penyimpanan file telah diimplementasikan dengan benar. Folder hasil_gambar terisi dengan file gambar yang sesuai, menandakan bahwa program dapat digunakan sebagai dasar scraping gambar dari situs lain yang serupa.

5.5 Kesimpulan

5.5.1 Kesimpulan Percobaan 1

Mahasiswa telah melakukan percobaan 1, program berhasil melakukan proses web scraping pada halaman utama situs Quotes to Scrape. Program mampu mengekstrak kutipan, nama penulis, dan tautan ke halaman detail penulis dengan tepat. Seluruh data yang diperoleh ditampilkan secara rapi di terminal dan disimpan dalam file teks tanpa error. Hal ini menunjukkan bahwa fungsi main_scrap() telah

berjalan sesuai harapan, serta proses parsing dan penyimpanan data telah berhasil dilakukan dengan benar.

5.5.2 Kesimpulan Percobaan 2

Mahasiswa telah berhasil melakukan percobaan 3, program berhasil melakukan web scraping pada halaman tag *inspirational* dari situs quotes.toscrape dengan dua tingkat kedalaman URL. Program tidak hanya menampilkan kutipan dan penulis dari halaman utama, tetapi juga mengambil informasi detail penulis serta kutipan tambahan berdasarkan tag yang dimiliki penulis tersebut. Semua data berhasil disimpan dalam file teks dengan struktur yang rapi, tanpa mengalami error saat pengambilan atau penulisan data. Hal ini menunjukkan bahwa program mampu menelusuri dan mengambil data dari sub-link secara menyeluruh sesuai dengan tujuan scraping tingkat lanjut.

Mengetahui:

Dosen Pengampu Mata Kuliah

Arif Hadi Sumitro , M.Kom

NIKP. Xxx

DAFTAR PUSTAKA

1. Mitchell, R. (2018). Web Scraping with Python: Collecting Data from the Modern Web (2nd ed.). O'Reilly Media.
2. Richardson, L. (2023). Beautiful Soup Documentation.
<https://www.crummy.com/software/BeautifulSoup/bs4/doc/>
3. Kenneth Reitz, Requests Documentation, 2015. <https://docs.python-requests.org>
4. Prasad, K. et al. (2018). "Web Scraping Techniques and Applications", International Journal of Computer Applications, Vol. 182, No. 10.

BAB 6

6.1 Capaian Praktikum Pertemuan 6

Pada praktikum pertemuan ke-6 ini mahasiswa diharapkan mampu memahami perbedaan antara penggunaan BeautifulSoup dan Pandas dalam proses web scraping, di mana BeautifulSoup digunakan untuk mengekstrak data secara detail dari struktur HTML atau XML, sedangkan Pandas lebih difokuskan pada pembacaan dan manipulasi data dalam bentuk tabel, seperti yang terdapat pada elemen HTML <table>. Selain itu, mahasiswa juga ditargetkan mampu mengimplementasikan teknik scraping menggunakan Pandas, termasuk membaca tabel dari situs web, mengolah data yang diambil, serta menyimpannya dalam format yang sesuai untuk analisis lebih lanjut.

6.2 Indikator Capaian

- Mahasiswa tidak melanggar aturan norma dan etika dalam pengambilan gambar.
- Mahasiswa melakukan scraping URL Gambar dengan menggunakan BeautifulSoup dan Request.

6.3 Landasan Teori

Menurut Mitchell (2018) dalam bukunya “*Web Scraping with Python*”, web scraping adalah teknik otomatisasi untuk mengambil data dari situs web, dengan cara mengekstrak konten HTML dan mengolahnya sesuai kebutuhan pengguna. Ia menjelaskan bahwa BeautifulSoup merupakan salah satu library Python yang populer dan dirancang untuk navigasi serta pencarian elemen

dalam dokumen HTML secara efisien dan fleksibel, sehingga cocok digunakan untuk scraping data yang tidak berbentuk tabel secara langsung.

Sementara itu, menurut McKinney (2017) dalam buku "*Python for Data Analysis*", Pandas adalah pustaka analisis data yang sangat kuat dan banyak digunakan untuk membaca, memproses, dan menyimpan data dalam berbagai format, termasuk tabel HTML. Fungsi `read_html()` pada Pandas memungkinkan pengguna mengambil langsung tabel dari halaman web tanpa perlu parsing manual, sehingga lebih efisien untuk data yang sudah terstruktur dalam format tabel.

6.4 Pelaksanaan Praktikum

6.4.1 Percobaan Pertama

Pada percobaan pertama mahasiswa melakukan scraping dari situs *Wikipedia*, di mana program menunjukkan proses web scraping menggunakan library Pandas untuk mengambil data tabel dari situs Wikipedia yang berisi daftar populasi negaranegara di dunia.

a. Script / Setting Program

`pandas.py`

```

import pandas as pd

# URL berisi tabel populasi negara-negara
url
'https://en.wikipedia.org/wiki/List_of_countries_and_dependencies_by_population'
# Ambil semua tabel dengan class 'wikitable'
df_list = pd.read_html(url, header=0, attrs={'class': 'wikitable'})
# Cek jumlah tabel dan pilih yang sesuai print(f"Jumlah
tabel ditemukan: {len(df_list)}") print("Tampilkan
kolom-kolom dari tabel pertama:")
print(df_list[0].columns)

# Gunakan tabel pertama (data populasi negara-negara)
df = df_list[0]

# Tampilkan beberapa baris awal
print("Data Populasi Negara-Negara:")
print(df.head())

# Simpan ke Excel
df.to_excel("population_by_country.xlsx", index=False)

print("\nData berhasil disimpan ke population_by_country.xlsx")

```

b. Langkah Uji Coba

- **url='https://en.wikipedia.org/wiki/List_of_countries_and_dependencies_by_population'**, menyimpan URL halaman Wikipedia yang berisi data populasi negara-negara dalam sebuah variabel url.
- **df_list = pd.read_html(url, header=0, attrs={'class': 'wikitable'})**, mengambil semua tabel dengan class "wikitable" dari halaman web.
- **print(f"Jumlah tabel ditemukan: {len(df_list)}")**, menampilkan jumlah tabel yang berhasil di-scrape dari halaman tersebut.
- **print("Tampilkan kolom-kolom dari tabel pertama:")**
print(df_list[0].columns), menampilkan nama kolom dari tabel pertama dalam df_list.
- **df = df_list[0]**, menyimpan tabel pertama (index ke-0) ke dalam variabel df.

- `print("Data Populasi Negara-Negara:") print(df.head())`, menampilkan 5 baris pertama dari tabel populasi.
- `df.to_excel("population_by_country.xlsx", index=False)`, menyimpan DataFrame df ke file Excel bernama population_by_country.xlsx tanpa menyertakan indeks.
- `print("\nData berhasil disimpan ke population_by_country.xlsx")`, menampilkan pesan konfirmasi ke terminal.

c. Hasil Uji Coba

Location	Population	% of world	Date	Source (official or from the United Nations)	Note
World	8232000000	100%	13 Jun 2025	UN projection[1][3]	NaN
India	1417492000	17.3%	1 Jul 2025	Official projection[4]	[b]
China	1408280000	17.2%	31 Dec 2024	Official estimate[5]	[c]
United States	340110988	4.2%	1 Jul 2024	Official estimate[6]	[d]
Indonesia	284438782	3.5%	30 Jun 2025	National annual projection[7]	NaN

Gambar 6.1 Hasil Scraping dari halaman Wikipedia

- Program berhasil dijalankan, proses scraping berhasil mengambil tabel dari halaman Wikipedia secara utuh. Struktur dan format tabel tetap terjaga dengan baik setelah dikonversi ke Excel. Data dapat digunakan langsung untuk analisis lebih lanjut, misalnya sorting, filtering, atau visualisasi.

d. Analisa Hasil

Hasil scraping menunjukkan bahwa data populasi berhasil diambil dengan baik dari Wikipedia, menampilkan 10 negara dengan populasi terbesar di dunia secara rapi. India dan China mendominasi dengan lebih dari sepertiga populasi global, sementara Indonesia berada di posisi kelima dengan 3,5%. Data disertai tanggal dan sumber resmi, sehingga dapat dipercaya dan layak untuk analisis lebih lanjut.

6.4.2 Percobaan Kedua

Pada percobaan kedua dilakukan dengan proses scraping data COVID-19 dari Wikipedia menggunakan Pandas dengan fokus pada kolom "Cases" dan "Deaths". Data dibersihkan dari karakter non-numerik seperti koma dan tanda tambah, lalu difilter agar hanya menyisakan nilai numerik yang valid. Setelah itu, dilakukan konversi tipe data ke float dan ditambahkan kolom baru bernama "Jumlah" yang merupakan hasil perkalian antara jumlah kasus dan jumlah kematian. Hasil akhir disimpan dalam file Excel covid_tambah_kolom.xlsx untuk analisis lanjutan.

a. Script / Setting Program

Tugas.py

```

import pandas as pd

# URL sumber data
url = 'https://en.wikipedia.org/wiki/Template:COVID-19_pandemic_data#covid-19pandemic-data'

# Atribut HTML tabel
attrs = {'class': 'wikitable'}

# Membaca tabel
df_list = pd.read_html(url, header=0, index_col=0, attrs=attrs) df = df_list[0]

# Tampilkan kolom-kolom awal untuk analisa print("Kolom tersedia:", df.columns)

# Bersihkan nama kolom jika perlu
df.columns = df.columns.str.replace(r"\[.*\]", "", regex=True).str.strip()
# Ganti koma & tanda + di angka agar bisa dikonversi ke float
df["Cases"] = df["Cases"].astype(str).str.replace(",", "").str.replace("+", "").str.strip()
df["Deaths"] = df["Deaths"].astype(str).str.replace(",", "").str.replace("+", "").str.strip()

# Hapus baris dengan nilai kosong atau bukan angka
df = df[df["Cases"].str.isnumeric() & df["Deaths"].str.isnumeric()]
# Konversi ke float
df["Cases"] = df["Cases"].astype(float) df["Deaths"] = df["Deaths"].astype(float)

#Tambahkan kolom baru "Jumlah"
df["Jumlah"] = df["Cases"] * df["Deaths"]

# Simpan ke Excel
df.to_excel("covid_tambah_kolom.xlsx", index=True)

# Tampilkan hasil akhir
print(df[["Cases", "Deaths", "Jumlah"]].head())

```

b. Langkah Uji Coba

- `url='https://en.wikipedia.org/wiki/Template:COVID19_pandemic_data#covid-19-pandemic-data'`, menyimpan URL halaman Wikipedia berisi data COVID-19 ke variabel url.
- `attrs = {'class': 'wikitable'}`, menentukan atribut HTML (`class="wikitable"`) sebagai filter untuk mengambil tabel yang diinginkan.

- `df_list = pd.read_html(url, header=0, index_col=0, attrs=attrs),`
membaca semua tabel dari halaman yang memiliki class wikitble.
- `df = df_list[0]`, mengambil tabel pertama dari df_list untuk diproses.
- `print("Kolom tersedia:", df.columns)`, menampilkan semua nama kolom dari tabel.
- `df.columns = df.columns.str.replace(r"\.*\"", "", regex=True).str.strip()`,
membersihkan nama kolom dari catatan kaki (misalnya [1], [2]) dan spasi.
- `df["Cases"] = df["Cases"].astype(str).str.replace(",","").str.replace("+", "")`.
`.str.strip()`, menghapus koma, tanda plus, dan
spasi dari kolom "Cases" lalu ubah ke tipe string.
- `df["Deaths"] = df["Deaths"].astype(str).str.replace(",","").str.replace("+", "")`.
`.str.strip()`, menghapus koma, tanda plus, dan
spasi dari kolom "Deaths" lalu ubah ke tipe string.
- `df = df[df["Cases"].str.isnumeric() & df["Deaths"].str.isnumeric()]`,
memfilter hanya baris yang berisi angka valid pada kolom "Cases" dan
"Deaths".
- `df["Cases"] = df["Cases"].astype(float) df["Deaths"] = df["Deaths"].astype(float)`, mengubah tipe data "Cases" dan "Deaths"
dari string ke float.
- `df["Jumlah"] = df["Cases"] * df["Deaths"]`, membuat kolom baru
"Jumlah" hasil dari "Cases" dikali "Deaths".
- `df.to_excel("covid_tambah_kolom.xlsx", index=True)`, menyimpan hasil
akhir ke file Excel.
- `print(df[["Cases", "Deaths", "Jumlah"]].head())`, menampilkan 5 baris
pertama dari kolom hasil akhir.

c. Hasil Uji Coba

```
PS C:\stikom\smt 4\Praktikum Data Scraping\pertemuan6> & "C:/Program Files/Python312/python.exe" "c:/stikom/smt 4\Praktikum Data Scraping\pertemuan6\tugas.py"
Kolom tersedia: Index(['Location', 'Cases', 'Deaths'], dtype='object')
c:\stikom\smt 4\Praktikum Data Scraping\pertemuan6\tugas.py:27: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
  df["Cases"] = df["Cases"].astype(float)
c:\stikom\smt 4\Praktikum Data Scraping\pertemuan6\tugas.py:28: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
  df["Deaths"] = df["Deaths"].astype(float)
c:\stikom\smt 4\Praktikum Data Scraping\pertemuan6\tugas.py:31: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
  df["Jumlah"] = df["Cases"] * df["Deaths"]
      Cases      Deaths      Jumlah
NaN  775866783.0  7057132.0  5.475394e+15
NaN  185822587.0  1262988.0  2.346917e+14
NaN  103436829.0  1193165.0  1.234172e+14
NaN  99373219.0   122304.0  1.215374e+13
NaN  45041748.0   533623.0  2.403531e+13
PS C:\stikom\smt 4\Praktikum Data Scraping\pertemuan6>
```

Location	Case	Deat	Jumlah
World[a]	7,76E+08	7057132	5,47539E+15
European Union[b]	1,86E+08	1262988	2,34692E+14
United States	1,03E+08	1193165	1,23417E+14
China[c]	99373219	122304	1,21537E+13
India	45041748	533623	2,40353E+13
France	38997490	168091	6,55513E+12
Germany	38437756	174979	6,7258E+12
Brazil	37511921	702116	2,63377E+13
South Korea	34571873	35934	1,24231E+12

Gambar 6.2 Hasil Scraping COVID-19 pada halaman Wikipedia

- Hasil scraping dari Wikipedia yang berisi data jumlah kasus (Case) dan kematian (Deat) akibat COVID-19 di berbagai wilayah, seperti World, European Union, United States, India, dan lainnya. Selain itu, kolom baru bernama Jumlah berhasil ditambahkan sebagai hasil perkalian antara jumlah kasus dan jumlah kematian pada masing-masing wilayah.
- Nilai-nilai dalam kolom Case, Deat, dan Jumlah ditampilkan dalam notasi ilmiah (exponential) karena besarnya angka, contoh:

- World[a] memiliki 776 juta kasus dan 7 juta kematian, menghasilkan nilai Jumlah sebesar 5.47×10^{15} .
- Brazil memiliki lebih dari 37 juta kasus dan 702 ribu kematian, dengan hasil perkalian sekitar 2.63×10^{13} .

d. Analisa Hasil

Program uji coba scraping menunjukkan bahwa data kasus dan kematian COVID-19 berhasil di-scrape dan diolah dengan baik, termasuk penambahan kolom baru "Jumlah" yang merupakan hasil perkalian antara jumlah kasus dan kematian. Nilai-nilai ditampilkan dalam notasi ilmiah karena ukuran datanya sangat besar, mencerminkan dampak global pandemi. Data ini siap digunakan untuk analisis lebih lanjut seperti perbandingan antar negara atau visualisasi.

6.5 Kesimpulan

6.5.1 Kesimpulan Percobaan 1

Mahasiswa telah melakukan percobaan 1, program berhasil melakukan proses scraping tabel populasi negara dari Wikipedia menggunakan Pandas berhasil dilakukan dengan baik, mulai dari pembacaan tabel HTML, pemilihan data, hingga penyimpanan ke file Excel. Data yang diperoleh memiliki struktur rapi dan mencakup informasi penting seperti jumlah populasi, persentase terhadap populasi dunia, dan sumber data. Hasil ini membuktikan bahwa Pandas sangat efektif untuk mengambil dan mengelola data tabular dari situs web secara otomatis dan efisien.

6.5.2 Kesimpulan Percobaan 2

Mahasiswa telah berhasil melakukan percobaan 2, program berhasil melakukan proses scraping data COVID-19 dari Wikipedia menggunakan Pandas,

termasuk pembersihan data, konversi tipe numerik, dan penambahan kolom baru "Jumlah" sebagai hasil perkalian antara jumlah kasus dan kematian. Data yang awalnya tidak terstruktur berhasil diolah menjadi tabel rapi dan informatif, lalu disimpan dalam file Excel untuk kebutuhan analisis lebih lanjut. Percobaan ini menunjukkan bahwa Pandas sangat andal dalam mengelola data web berskala besar secara efisien.

Mengetahui:

Dosen Pengampu Mata Kuliah

Arif Hadi Sumitro , M.Kom

NIKP. xxx

DAFTAR PUSTAKA

1. Mitchell, R. (2018). *Web Scraping with Python: Collecting Data from the Modern Web* (2nd ed.). O'Reilly Media.
2. McKinney, W. (2017). *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython* (2nd ed.). O'Reilly Media.

BAB 7

7.1 Capaian Praktikum Pertemuan 7

- Mahasiswa mampu memahami setup flask pada environment Python
- Mahasiswa mampu mengimplementasikan flask dengan menggunakan pycharm
- Mahasiswa meampu memahami alur menampilkan data hasil scrap kedalam flask.

7.2 Indikator Capaian

Mahasiswa mampu melakukan instalasi Flask dan menjalankan aplikasinya dengan baik sebagai dasar pengembangan web menggunakan Python. Mahasiswa juga mampu melakukan proses scraping data dari sumber tertentu dan menampilkannya ke dalam aplikasi Flask. Selain itu, mahasiswa dapat membuat rute (route) dan fungsi yang sesuai untuk masing-masing data yang ditampilkan, sehingga aplikasi dapat menyajikan informasi secara terstruktur dan dinamis.

7.3 Landasan Teori

Menurut Grinbergm (2018), Flask adalah micro web framework yang ringan dan fleksibel, dirancang untuk memudahkan pengembangan aplikasi web dengan Python. Flask memberikan kebebasan kepada pengembang untuk mengatur struktur proyek sesuai kebutuhan, serta menyediakan fitur dasar seperti routing, templating, dan pengelolaan request/response.

Menurut Mitchell (2018), web scraping adalah proses otomatisasi pengambilan data dari situs web menggunakan kode program. Dalam konteks Python, scraping umumnya dilakukan dengan pustaka seperti requests dan

BeautifulSoup, untuk mengakses dan memproses data HTML secara efisien sebelum ditampilkan dalam aplikasi.

Routing adalah konsep dasar dalam pengembangan aplikasi web yang menghubungkan URL dengan fungsi atau aksi tertentu. Menurut Raganathan (2020), dalam Flask setiap route didefinisikan dengan menggunakan decorator @app.route, yang memungkinkan pemetaan permintaan HTTP ke fungsi Python tertentu, sehingga memungkinkan pembuatan aplikasi yang responsif dan terstruktur.

7.4 Pelaksanaan Praktikum

7.4.1 Percobaan Pertama

Mahasiswa diperintahkan untuk membuat struktur folder dan file secara otomatis menggunakan Python, dengan tujuan memahami penggunaan modul os dan dasar-dasar manipulasi file dan direktori.

a. Script / Setting Program

app.py

```
from bs4 import BeautifulSoup
import requests
from flask import Flask, render_template

app = Flask(__name__)
def main_scraper(url,
directory):
    fungsi.create_directory(directory)
headers = {
    'User-Agent': 'Mozilla/5.0'
}
response = requests.get(url, headers=headers)
soup = BeautifulSoup(response.text, "html.parser")

articles = soup.select("article a[href]")
data = []      for
article in articles:
    title = article.get_text(strip=True)      link
= article.get("href")      if title and link and
link.startswith("https://"):
    data.append({
"title": title,
"url": link
})
return data

@app.route("/")
def index():
    url = "https://www.detik.com/"
    hasil = main_scraper(url, "hasil_scrape")
return render_template("index.html", data=hasil)
if __name__ ==
 "__main__":
    app.run(debug=True)
```

fungsi.py

```
import os

def create_directory(directory_name):
    if not os.path.exists(directory_name):
        os.makedirs(directory_name)
```

index.html

```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Berita detik.com</title>
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
<style>
    body { background-color: #f9f9f9; }
    .card:hover { transform: scale(1.02); transition: 0.2s; }
</style>
</head>
<body>
    <div class="container py-5">
        <h1 class="text-center mb-4">detik.com</h1>
        {% if data %}
            <div class="row row-cols-1 row-cols-md-2 row-cols-lg-3 g-4">
                {% for item in data %}
                    <div class="col">
                        <div class="card h-100 shadow-sm">
                            <div class="card-body">
                                <h5 class="card-title">{{ item.title }}</h5>
                                <a href="{{ item.url }}" class="btn btn-sm btn-success mt-3" target="_blank">Baca Berita</a>
                            </div>
                        </div>
                    {% endfor %}
                </div>
            {% else %}
                <div class="alert alert-warning text-center">Tidak ada berita ditemukan.</div>
            {% endif %}
        </div>
    </body>
</html>
```

b. Langkah Uji Coba

app.py

- **import requests**, untuk mengambil konten dari website.
- **BeautifulSoup**, dari **bs4** untuk parsing HTML.
- **Flask, render_template**, untuk membuat dan menampilkan web secara lokal.
- **Fungsi** untuk memanggil **create_directory()** yang akan membuat folder hasil scraping.

- `app = flask(name)` membuat instance flask sebagai aplikasi utama, `__name__` digunakan untuk menentukan apakah file ini dijalankan langsung atau diimpor sebagai modul.
- `fungsi.create_directory(directory)`, memanggil fungsi dari file `fungsi.py` untuk membuat folder dengan nama `hasil_scrape`, yang berfungsi menyimpan hasil atau sebagai indikasi scraping berhasil dilakukan.
- `header = {'User-Agent':'Mozilla/5.0'}`, menentukan user-agent untuk menghindari pemblokiran oleh situs target saat melakukan permintaan (request).
- `response = requests.get(url), headers=headers`, mengirim request HTTP GET ke URL target (<https://www.detik.com/>) dan menyimpan respons HTMLnya.
- `soup = BeautifulSoup(response.text, "html.parser")`, memarsing HTML dari halaman website agar dapat diekstrasi informasinya.
- `articles = soup.select("article a[href]"),` mengambil elemen `<a>` di dalam tag `<article>` yang memiliki atribut href, sebagai target utama scraping.
- `for article in articles,` melakukan iterasi untuk setiap link artikel. Mengambil teks (judul) dan link (URL) untuk dimasukkan ke list data.
- `Data.append({...})`, menyimpan hasil scraping dalam bentuk list of dictionary `{“title”: ..., “url”: ...}` yang nantinya dikirim ke template.
- `@app.route("/")`, mendefinisikan route utama `(/)` pada Flask. Saat dibuka di browser, akan menampilkan hasil scraping.
- `Return render_template("index.html", data=hasil),` mengirimkan data hasil scraping ke file HTML `(index.html)` untuk ditampilkan sebagai halaman web.

- **If name == “main”:**`app.run(debug=True)`, menjalankan server lokal Flask jika file dijalankan langsung. Mode debug aktif agar perubahan kode otomatis ter-update saat dijalankan.

fungsi.py

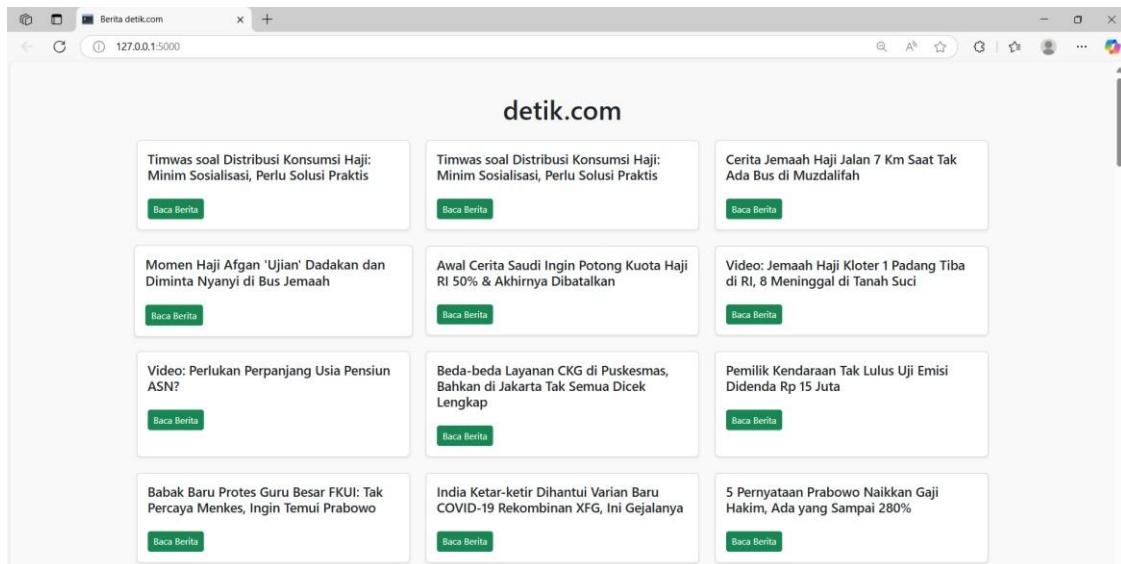
- **import os**, mengimpor modul `os`, yang digunakan untuk berinteraksi dengan sistem operasi, seperti membuat folder dan file, mengecek apakah sudah ada, serta menggabungkan path.
- **def create_directory(directory_name)**, mendefinisikan fungsi dengan nama `create_directory` yang menerima satu parameter `directory_name` sebagai nama folder yang akan dibuat.
- **if not os.path.exists(directory_name)**, mengecek apakah folder dengan nama `directory_name` sudah ada. Jika belum ada, maka lanjut ke Langkah berikutnya.
- **os.makedirs(directory_name)**, membuat folder baru dengan nama sesuai `directory_name`. Fungsi ini juga akan membuat folder di dalam folder lain jika path yang diberikan bersifat bertingkat.

index.html

- Menggunakan template HTML standar dengan Bootstrap 5 untuk tampilan yang responsif.
- **Percabangan {%** if data %**}**, mengecek apakah data hasil scraping tersedia. Jika ada, akan ditampilkan dalam bentuk card.
- **Perulangan {%** for item in data %**}**, melakukan iterasi untuk setiap berita yang di scrape. Menampilkan judul berita dan tombol.
- Mengarahkan pengguna ke URL asli berita di `detik.com` dengan `target=_blank`.

- Jika tidak ada data, ditampilkan alert Bootstrap: “**Tidak ada berita ditemukan.**”

c. Hasil Uji Coba



Gambar 7.1 Hasil Uji Coba Scraping dari web detik.com

d. Analisa Hasil

Aplikasi berhasil menampilkan data berita terkini dari situs detik.com dalam bentuk card yang rapi dan responsif. Setiap card memuat judul berita dan tombol “Baca Berita” yang mengarah langsung ke URL berita asli. Data berhasil ditampilkan secara dinamis menggunakan `render_template()` dari Flask, hal ini membuktikan bahwa scraping, pemrosesan data, dan penyajian data secara otomatis berhasil diimplementasikan dengan benar.

7.5 Kesimpulan

7.5.1 Kesimpulan Percobaan 1

Pada percobaan ini, dapat disimpulkan bahwa proses scraping data dari situs web, pengelolaan file dan folder, serta menampilkan data secara dinamis melalui framework Flask dapat berjalan dengan baik. Mahasiswa berhasil mengintegrasikan

modul requests, BeautifulSoup, dan os untuk mengambil dan mengelola data. Percobaan ini menunjukkan bahwa mahasiswa telah memahami dasar-dasar otomatisasi pemrograman Python, penerapan logika program, serta pengembangan aplikasi web sederhana yang dapat digunakan untuk menyajikan informasi secara real-time.

Mengetahui:

Dosen Pengampu Mata Kuliah

Arif Hadi Sumitro , M.Kom

NIKP. xxx

DAFTAR PUSTAKA

1. Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python* (2nd ed.). O'Reilly Media.
2. Mitchell, R. (2018). *Web Scraping with Python: Collecting More Data from the Modern Web* (2nd ed.). O'Reilly Media.
3. Raganathan, S. (2020). *Mastering Flask Web Development*. Packt Publishing.

BAB 8

8.1 Capaian Praktikum Pertemuan 8

Pada praktikum pertemuan ke-8 ini mahasiswa diharapkan mampu memahami konsep dasar penggunaan routing pada Flask sebagai salah satu framework web berbasis Python. Selain itu, mahasiswa juga dituntut untuk dapat menerapkan hasil scraping data dari berbagai sumber web ke dalam sistem routing Flask. Dengan demikian, mahasiswa tidak hanya memahami alur pengambilan data melalui scraping, tetapi juga mampu mengintegrasikan setiap elemen hasil scraping ke dalam tampilan web menggunakan routing yang tepat pada aplikasi Flask.

8.2 Indikator Capaian

- Mahasiswa membuat konsep routing pada flask dengan ketentuannya.
- Mahasiswa membuat website dengan template jinja untuk mendukung data yang ditampilkan dari scrape.
- Mahasiswa membuat routing dengan serta menerapkan disetiap routingnya hasil dari scrape.

8.3 Landasan Teori

Menurut Grinberg (2018) dalam bukunya "*Flask Web Development*", routing adalah mekanisme yang digunakan untuk menghubungkan URL tertentu dengan fungsi (view function) dalam aplikasi Flask. Routing bertindak sebagai penghubung antara permintaan dari pengguna (request) dengan respons yang akan diberikan oleh server. Setiap route ditentukan menggunakan dekorator @app.route, yang mengarahkan URL ke fungsi Python tertentu.

Menurut Mitchell (2018) dalam bukunya "*Web Scraping with Python*", web scraping adalah teknik untuk mengekstrak data dari situs web secara otomatis menggunakan program. Dalam Python, proses ini biasanya dilakukan dengan

bantuan library seperti requests untuk mengambil halaman web dan BeautifulSoup untuk mem-parsing struktur HTML. Web scraping banyak digunakan dalam pengumpulan data untuk analisis, riset, dan aplikasi web dinamis.

Sementara itu, menurut Sonmez & Dede (2020), integrasi data hasil scraping ke dalam aplikasi web dapat meningkatkan fungsionalitas dan dinamika konten situs. Dengan framework seperti Flask, hasil scraping dapat diolah dan ditampilkan secara real-time melalui template HTML menggunakan Jinja2.

8.4 Pelaksanaan Praktikum

8.4.1 Percobaan Pertama

Pada percobaan pertama mahasiswa menampilkan hasil scraping dari situs BolaSport melalui routing Flask. Data diambil menggunakan requests dan BeautifulSoup, lalu ditampilkan ke halaman web menggunakan template Jinja2.

a. Script / Setting Program

app.py

```
from flask import Flask, render_template, request
from main import (
    scrape_bolasport,
    scrape_detik_jatim,
    scrape_detik_jateng,
    scrape_detik_jabar,
    scrape_liputan6,
    scrape_cnnindonesia,
    scrape_detail_berita
)

app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/bola-sport')
def bola_sport():
    data = scrape_bolasport()
    return render_template('bola-sport.html', data=data)

@app.route('/detik-jatim')
def detik_jatim():
    data = scrape_detik_jatim()
    return render_template('detik-jatim.html', data=data)

@app.route('/detik-jateng')
```

```

def detik_jateng():
    data = scrape_detik_jateng()
    return render_template('detik-jateng.html', data=data)

@app.route('/detik-jabar')
def detik_jabar():
    data = scrape_detik_jabar()
    return render_template('detik-jabar.html', data=data)

@app.route('/berita-gabungan')
def berita_gabungan():
    liputan6_data = scrape_liputan6()
    cnn_data = scrape_cnnindonesia()
    return render_template('berita_gabungan.html', liputan6=liputan6_data,
cnn=cnn_data)

@app.route('/berita')
def detail_berita():
    url = request.args.get('url')
    if not url:
        return 'URL berita tidak ditemukan', 400
    detail = scrape_detail_berita(url)
    return render_template('detail_berita.html', berita=detail)

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

```

main.py

```

import requests

from bs4 import BeautifulSoup

def scrape_bolasport():
    url = 'https://www.bolasport.com/'

    res = requests.get(url)

    soup = BeautifulSoup(res.text, 'html.parser')

    return soup.find_all('div', class_='news-list__item clearfix')

def scrape_detik_jatim():

    url = 'https://www.detik.com/jatim'

    headers = {'User-Agent': 'Mozilla/5.0'}

    res = requests.get(url, headers=headers)

    soup = BeautifulSoup(res.text, 'html.parser')

    return soup.find_all('article')

def scrape_detik_jateng():

    url = 'https://www.detik.com/jateng'

```

```

headers = {'User-Agent': 'Mozilla/5.0'}

res = requests.get(url, headers=headers)

soup = BeautifulSoup(res.text, 'html.parser')

return soup.find_all('article')

def scrape_detik_jabar():

    url = 'https://www.detik.com/jabar'

    headers = {'User-Agent': 'Mozilla/5.0'}

    res = requests.get(url, headers=headers)

    soup = BeautifulSoup(res.text, 'html.parser')

    return soup.find_all('article')

def scrape_liputan6():

    url = 'https://www.liputan6.com/citizen6'

    headers = {'User-Agent': 'Mozilla/5.0'}

    res = requests.get(url, headers=headers)

    soup = BeautifulSoup(res.text, 'html.parser')

    articles = soup.find_all('article', class_='articles--iridescent-list--text-item')

    data = []

    seen = set()

    for article in articles:

        link_tag = article.find('a', class_='articles--iridescent-list--text-item__title-link')

        title_span = article.find('span', class_='articles--iridescent-list--text-item__title-link-text')

        img_tag = article.find('img')

        if not link_tag or not title_span:

            continue

        title = title_span.get_text(strip=True)

        link = link_tag['href']

        image = ''

        if img_tag:

            image = (
                img_tag.get('data-src') or

```

```

        img_tag.get('src') or
        img_tag.get('data-original') or
        ''
    )
    if title not in seen:
        data.append({'title': title, 'link': link, 'image': image})
        seen.add(title)
    return data
def scrape_cnnindonesia():
    url = 'https://www.cnnindonesia.com/gaya-hidup'
    headers = {'User-Agent': 'Mozilla/5.0'}
    res = requests.get(url, headers=headers)
    soup = BeautifulSoup(res.text, 'html.parser')
    data = []
    anchors = soup.find_all('a', class_='flex group items-center gap-4')
    for anchor in anchors:
        link = anchor.get('href')
        img_tag = anchor.find('img')
        h2_tag = anchor.find('h2')
        if not link or not img_tag or not h2_tag:
            continue
        image = img_tag.get('src', '')
        title = h2_tag.get_text(strip=True)
        data.append({
            'title': title,
            'link': link,
            'image': image
        })
    return data
def scrape_detail_berita(url):
    headers = {'User-Agent': 'Mozilla/5.0'}
    res = requests.get(url, headers=headers)

```

```

soup = BeautifulSoup(res.text, 'html.parser')

# Liputan6

title = soup.find('h1', class_='read-page--header--title')

if title:

    title = title.get_text(strip=True)

    img_url = ''

    gallery_div = soup.find('div', class_='read-page--photo-gallery--item__content')

    if gallery_div:

        img_tag = gallery_div.find('img')

        if img_tag:

            img_url = img_tag.get('data-src') or img_tag.get('src') or ''

    content_div = soup.find('div', class_='article-content-body__item-page')

    content_html = str(content_div) if content_div else ''

    return {'title': title, 'image': img_url, 'content': content_html}

# CNN Indonesia

title = soup.find('h1', class_='mb-2 text-[32px] text-cnn_black font-merriweather')

if title:

    title = title.get_text(strip=True)

    img_tag = soup.find('img', class_='w-full')

    img_url = img_tag.get('src', '') if img_tag else ''

    content_div = soup.find('div', class_='detail-text text-cnn_black text-sm grow min-w-0')

    content_html = str(content_div) if content_div else ''

    return {'title': title, 'image': img_url, 'content': content_html}

return {'title': 'Berita tidak ditemukan', 'image': '', 'content': ''}

```

bola-sport.html

```

{% extends 'index.html' %}

{% block body %}

```

```

<h2>Bola Sport</h2>

<div class="row">

    {% for item in data %}

        <div class="col-3">

            {{ item.find('img')|safe }}

        </div>

        <div class="col-9">

            <a href="{{ item.find('a', {'class': 'news-list__link'})['href'] }}>

                <p class="fs-6">{{ item.find('img')['alt']|safe }}</p>

            </a>

        </div>

    {% endfor %}

</div>

{% endblock %}

```

detik-jabar.html

```

{% extends 'index.html' %}

{% block body %}

<h2>Detik Jabar</h2>

<div class="row">

    {% for item in data %}

        <div class="col-3">

            {% if item.find('img') and item.find('img').has_attr('src') %}

            {% endif %}

        </div>

        <div class="col-9">

            {% if item.find('a') and item.find('a').has_attr('href') %}

                <a href="{{ item.find('a')['href'] }}>

                    <p>{{ item.find('h2').text.strip() if item.find('h2') else 'Tanpa
Judul' }}</p>

                </a>

            {% endif %}

        </div>

    {% endfor %}

</div>

```

```

    {% endif %}

  </div>

  {% endfor %}

</div>

{% endblock %}

```

detik-jateng.html

```

{% extends 'index.html' %}

{% block body %}

<h2>Detik Jateng</h2>

<div class="row">

  {% for item in data %}

    <div class="col-3">

      {% if item.find('img') and item.find('img').has_attr('src') %}

      {% endif %}

    </div>

    <div class="col-9">

      {% if item.find('a') and item.find('a').has_attr('href') %}

        <a href="{{ item.find('a')['href'] }}>

          <p>{{ item.find('h2').text.strip() if item.find('h2') else 'Tanpa
Judul' }}</p>

        </a>

      {% endif %}

    </div>

  {% endfor %}

</div>

{% endblock %}

```

detik-jatim.html

```

{% extends 'index.html' %}

{% block body %}

```

```

<h2>Detik Jatim</h2>

<div class="row">

    {% for item in data %}

        <div class="col-3">

            {% if item.find('img') %}

            {% endif %}

        </div>

        <div class="col-9">

            <a href="{{ item.find('a')['href'] }}>

                <p>{{ item.find('h2').text.strip() if item.find('h2') else 'Tanpa
Judul' }}</p>

            </a>

        </div>

    {% endfor %}

</div>

{% endblock %}

```

berita_gabungan.html

```

<!DOCTYPE html>

<html lang="id">

<head>

    <meta charset="UTF-8">

    <title>Berita Gabungan</title>

    <style>

        body {

            font-family: 'Segoe UI', sans-serif;
            background-color: #f9f9f9;
            padding: 20px;
        }

        h2 {
            color: #2c3e50;
        }
    
```

```
}

.berita-container {
    display: flex;
    flex-wrap: wrap;
    gap: 20px;
}

.berita-card {
    background-color: #fff;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
    width: 300px;
    overflow: hidden;
    transition: 0.3s;
}

.berita-card:hover {
    transform: scale(1.03);
}

.berita-card img {
    width: 100%;
    height: 180px;
    object-fit: cover;
}

.berita-card .content {
    padding: 15px;
}

.berita-card h4 {
    font-size: 18px;
    margin: 10px 0;
}

.berita-card a {
    text-decoration: none;
    color: #3498db;
```

```

        }

    </style>

</head>

<body>

    <h2>Berita dari Liputan6</h2>

    <div class="berita-container">

        {% for item in liputan6 %}

            <div class="berita-card">

                <div class="content">

                    <h4>{{ item.title }}</h4>

                    <a href="/berita?url={{ item.link }}>Baca Selengkapnya</a>

                </div>

            </div>

        {% endfor %}

    </div>

    <h2>Berita dari CNN Indonesia</h2>

    <div class="berita-container">

        {% for item in cnn %}

            <div class="berita-card">

                <div class="content">

                    <h4>{{ item.title }}</h4>

                    <a href="/berita?url={{ item.link }}>Baca Selengkapnya</a>

                </div>

            </div>

        {% endfor %}

    </div>

</body>

</html>

```

detail_berita.html

```
<!DOCTYPE html>

<html lang="id">
<head>

    <meta charset="UTF-8">

    <title>{{ berita.title }}</title>

    <style>

        body {
            font-family: 'Segoe UI', sans-serif;
            background-color: #f2f2f2;
            margin: 0;
            padding: 30px;
        }

        .container {
            max-width: 800px;
            margin: auto;
            background-color: #ffffff;
            border-radius: 10px;
            box-shadow: 0 4px 12px rgba(0,0,0,0.1);
            overflow: hidden;
        }

        .header-image {
            width: 100%;
            max-height: 400px;
            object-fit: cover;
        }

        .content {
            padding: 25px 30px;
        }

        .content h1 {
            font-size: 26px;
            color: #2c3e50;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="header-image" style="background: url('https://www.w3schools.com/html/pic_mountain.jpg') no-repeat center / cover;">
```

```

        margin-bottom: 20px;
    }

    .content p {
        font-size: 16px;
        line-height: 1.8;
        color: #333;
        margin-bottom: 15px;
    }

    .back-button {
        display: inline-block;
        margin: 20px 30px;
        background-color: #3498db;
        color: white;
        padding: 10px 18px;
        text-decoration: none;
        border-radius: 6px;
        transition: background 0.3s;
    }

    .back-button:hover {
        background-color: #2a80b9;
    }

```

</style>

</head>

<body>

<div class="container">

{
% if berita.image %}

{
% endif %}

<div class="content">

<h1>{{ berita.title }}</h1>

{{ berita.content|safe }}

```

        </div>
    </div>
    <a class="back-button" href="/">← Kembali ke Halaman index</a>
</body>
</html>

```

index.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Berita Terkini</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <div class="container">
            <a class="navbar-brand" href="/">Portal Berita</a>
            <div class="collapse navbar-collapse">
                <ul class="navbar-nav ms-auto">
                    <li class="nav-item"><a class="nav-link" href="/bola-sport">Bola Sport</a></li>
                    <li class="nav-item"><a class="nav-link" href="/detik-jatim">Detik Jatim</a></li>
                    <li class="nav-item"><a class="nav-link" href="/detik-jateng">Detik Jateng</a></li>
                    <li class="nav-item"><a class="nav-link" href="/detik-jabar">Detik Jabar</a></li>
                    <li class="nav-item"><a class="nav-link" href="/berita-gabungan">Berita Gabungan</a></li>
                </ul>
            </div>
        </div>
    </nav>

```

```
</nav>

<div class="container mt-4">
    {% block body %}{% endblock %}
</div>
</body>
</html>
```

b. Langkah Uji Coba

- app.py

- **from flask import Flask**, mengimpor Flask (membuat instance aplikasi), **render_template** (untuk merender HTML dari folder templates), dan **request** (mengakses parameter request).
- **from main import ()**, mengimpor semua fungsi scraping dari file main.py. Masing-masing fungsi bertugas mengambil data berita dari situs tertentu.
- **app = Flask(__name__)**, membuat objek Flask utama bernama app. Ini adalah inti dari aplikasi web Flask.
- **@app.route('/') def home()**, ketika pengguna mengakses URL root (/), fungsi home() akan dipanggil dan merender index.html.
- **@app.route('/bola-sport') def bola_sport()**, mengakses URL /bola-sport akan memicu fungsi scrape_bolasport(), dan hasilnya dikirim ke template bola-sport.html.
- **@app.route('/detik-jatim')def detik_jatim()**, menyediakan data dari Detik Jatim untuk dirender di detik-jateng.html.
- **@app.route('/detik-jateng) def detik_jateng()**, menyediakan data dari Detik Jateng untuk dirender di detik-jateng.html.
- **@app.route('/detik-jabar)def detik_jabar()**, menyediakan data dari Detik Jabar untuk dirender di detik-jateng.html.

- `@app.route('/berita-gabungan') def berita_gabungan(),` menampilkan berita gabungan dari dua sumber (Liputan6 & CNN). Masing-masing hasil scraping dikirim ke template berita_gabungan.html.
- `@app.route('/berita') def detail_berita(),` Endpoint /berita menerima parameter url melalui query string. Jika url ditemukan, maka fungsi `scrape_detail_berita(url)` dipanggil dan hasilnya ditampilkan di detail_berita.html.

- main.py

- `import requests,` untuk mengambil data HTML dari website. `from bs4 import BeautifulSoup,` untuk mem-parsing HTML dan mengekstrak informasi.
- `def scrape_bolasport(),` mengambil daftar berita dari BolaSport berdasarkan elemen div dengan class tertentu.
- `def scrape_detik_jatim(), def scrape_detik_jateng(), def scrape_detik_jabar(),` mengambil semua tag `<article>` dari halaman Detik wilayah (Jatim, Jateng, Jabar).
- `def scrape_liputan6(),` mengambil berita dari Liputan6. `for article in articles,` menghindari duplikat judul. `data.append({'title': ..., 'link': ..., 'image': ...})`, mengembalikan list dict: title, link, image.
- `def scrape_cnnindonesia(),` mengambil berita dari CNN Indonesia, bagian gaya hidup. `data.append({'title': ..., 'link': ..., 'image': ...})`, ambil data dari tag `<a>` dengan struktur gambar dan judul.
- `def scrape_detail_berita(url), if title: return [...],` menampilkan detail berita: Jika url dari Liputan6, ambil judul, gambar, dan isi. Jika url dari CNN, lakukan hal serupa. Jika tak cocok, kembalikan "berita tidak ditemukan".

- templates (bola-sport.html)

- `{% extends 'index.html' %}`, template ini mewarisi struktur dari index.html (biasanya ada kerangka HTML dasar seperti `<html>`, `<head>`, `<body>`, dll).
- `{% block body %}`, memulai blok konten bernama body, yang akan menggantikan blok body di index.html.
- `<h2>Bola Sport</h2>`, menampilkan judul “Bola Sport” di halaman.
- `<div class="row">`, membuat baris (row) Bootstrap sebagai kontainer untuk grid kolom.
- `{% for item in data %}`, melakukan perulangan untuk setiap elemen dalam variabel data (berisi hasil scraping `scrape_bolasport()`).
- `<div class="col-3">`, membuat kolom selebar 3 (dari 12) untuk gambar berita.
- `{{ item.find('img')|safe }}`, menampilkan elemen `` dari objek item. Gunakan `|safe` agar HTML gambar dirender langsung, bukan sebagai teks.
- `<div class="col-9">`, kolom lebar 9 untuk judul berita.
- ``, tautan ke berita lengkap, diambil dari elemen `<a>` dengan class `news-list__link`.
- `<p class="fs-6">{{ item.find('img')['alt']|safe }}</p>`, menampilkan teks alternatif dari gambar sebagai judul berita dalam paragraf ukuran kecil (fs-6 dari Bootstrap).
- `{% endfor %}`, menutup blok body yang dimulai sebelumnya.

- templates (detik_jabar.html)

- `{% extends 'index.html' %}`, template ini mewarisi dari index.html, artinya hanya mengisi bagian tertentu dari kerangka besar (`<html>`, `<head>`, dll).
- `{% block body %}`, memulai blok bernama body yang akan menggantikan isi blok body di index.html.
- `<h2>Detik Jabar</h2>`, menampilkan judul halaman “Detik Jabar”.
- `<div class="row">`, memulai **baris (row)** layout Bootstrap untuk grid 12 kolom.

- `{% for item in data %}`, perulangan untuk setiap elemen item dalam data (hasil dari `scrape_detik_jabar()`).
- `<div class="col-3">`, membuat kolom selebar 3 (untuk gambar berita).
- `{% if item.find('img') and item.find('img').has_attr('src') %}`, cek apakah elemen `` ada dan punya atribut src sebelum menampilkan gambar.
- ``, menampilkan gambar berita dengan Bootstrap img-fluid agar responsive.
- `{% endif %}`, akhir kondisi if untuk gambar.
- `<div class="col-9">`, kolom sebesar 9 untuk menampilkan judul/link berita.
- `{% if item.find('a') and item.find('a').has_attr('href') %}`, cek apakah ada tag `<a>` dan memiliki atribut href.
- ``, membuat tautan ke halaman berita.
- `<p>{{ item.find('h2').text.strip() if item.find('h2') else 'Tanpa Judul' }}</p>`, menampilkan judul berita dari elemen `<h2>`. Jika tidak ada tag `<h2>`, tampilkan teks “Tanpa Judul”.
- `{% endfor %}`, akhir dari perulangan for.
- `{% endblock %}`, menutup blok body.

- templates (detik_jateng.html)

- `{% extends 'index.html' %}`, template ini mewarisi dari index.html, artinya hanya mengisi bagian tertentu dari kerangka besar (`<html>`, `<head>`, dll).
- `{% block body %}`, memulai blok bernama body yang akan menggantikan isi blok body di index.html.
- `<h2>Detik Jateng</h2>`, menampilkan judul halaman “Detik Jateng”.
- `<div class="row">`, membuat baris grid Bootstrap untuk mengatur tata letak berita dalam 2 kolom (gambar + teks).
- `{% for item in data %}`, melakukan perulangan terhadap setiap elemen item dalam variabel data (hasil scraping dari `scrape_detik_jateng()`).

- <div class="col-3">, membuat kolom berukuran 3 (dari total 12) untuk menampilkan gambar berita.
- {% if item.find('img') and item.find('img').has_attr('src') %}, memastikan elemen ada dan memiliki atribut src sebelum ditampilkan.
- , menampilkan gambar berita dari tag dengan class Bootstrap img-fluid agar gambar responsif.
- {% endif %}, akhir dari kondisi untuk memeriksa dan menampilkan gambar.
- <div class="col-9">, membuat kolom berukuran 9 untuk teks (judul berita).
- {% if item.find('a') and item.find('a').has_attr('href') %}, memastikan ada tag <a> dan memiliki atribut href untuk membuat tautan yang valid.
- , membuat tautan ke halaman berita berdasarkan atribut href dari tag <a>.
- <p>{{ item.find('h2').text.strip() if item.find('h2') else 'Tanpa Judul' }}</p>, menampilkan judul berita dari tag <h2>, jika ada. Jika tidak ada, tampilkan “Tanpa Judul”.
- {% endfor %}, menutup perulangan berita (for).
- {% endblock %}, menutup blok body.

- templates (detik_jatim.html)

- {% extends 'index.html' %}, template ini mewarisi dari index.html, artinya hanya mengisi bagian tertentu dari kerangka besar (<html>, <head>, dll).
- {% block body %}, memulai blok bernama body yang akan menggantikan isi blok body di index.html.
- <h2>**Detik Jatim**</h2>, menampilkan judul halaman “Detik Jatim”.
- <div class="row">, membuat baris (row) Bootstrap grid system agar berita bisa dibagi ke dalam kolom.
- {% for item in data %}, melakukan perulangan untuk setiap elemen item dalam variabel data (yang dikirim dari fungsi scrape_detik_jatim() di app.py).

- <div class="col-3">, membuat kolom berukuran 3 (dari total 12) untuk menampilkan gambar berita.
- {% if item.find('img') %}, mengecek apakah item memiliki tag . Tidak memeriksa apakah ada src, jadi bisa error jika img tanpa atribut src.
- , membuat gambar responsif (menyesuaikan ukuran layar).
- {% endif %}, menutup kondisi pengecekan gambar.
- <div class="col-9">, kolom sebesar 9 digunakan untuk menampilkan judul dan link berita.
- , membuat tautan (link) ke halaman berita berdasarkan tag <a> di dalam item.
- <p>{{ item.find('h2').text.strip() if item.find('h2') else 'Tanpa Judul' }}</p>, menampilkan judul berita dari tag <h2>, jika ada. Jika tidak ada, tampilkan "Tanpa Judul".
- {% endfor %}, menutup perulangan berita (for).
- {% endblock %}, menutup blok body.

- templates (berita_gabungan.html)

- <!DOCTYPE html> <html lang="id">, menandakan dokumen ini adalah HTML5 dan menggunakan bahasa Indonesia (lang="id").
- <meta charset="UTF-8">, menentukan encoding karakter menjadi UTF-8 (standar karakter internasional).
- <title>Berita Gabungan</title>, judul halaman yang akan muncul di tab browser: "Berita Gabungan".
- <style> body { font-family, background-color, padding }, mengatur font default, warna latar belakang (#f9f9f9), dan padding pada seluruh halaman.
- h2 { color }, warna teks pada elemen <h2> (judul kategori berita).

- `.berita-container { display, flex-wrap, gap }`, mengatur layout container berita agar berderet fleksibel dengan jarak antar elemen (gap).
- `.berita-card { background-color, border-radius, box-shadow, width, overflow, transition }`, styling untuk tiap kartu berita, seperti latar putih, sudut membulat, bayangan lembut, dan efek transisi saat hover.
- `.berita-card:hover { transform }`, saat hover, kartu berita akan membesar sedikit (efek interaktif).
- `.berita-card img { width, height, object-fit }`, gambar dalam kartu berita akan menyesuaikan lebar penuh dan dipotong rapi sesuai rasio 180px.
- `.berita-card .content { padding }`, konten teks di dalam kartu akan diberi jarak dari pinggir.
- `.berita-card h4 { font-size, margin }`, margin berita dalam kartu akan berukuran sedang dan diberi margin atas-bawah.
- `.berita-card a { text-decoration, color }`, tautan tidak bergaris bawah dan berwarna biru terang.
- `<body> <h2>Berita dari Liputan6</h2>`, judul bagian pertama: berita yang berasal dari **Liputan6**.
- `<div class="berita-container"> {% for item in liputan6 %}`, membuat container grid, lalu perulangan Jinja2 untuk setiap berita dari liputan6 (data hasil scraping).
- `<div class="berita-card">`, menampilkan gambar berita (`item.image`) dengan deskripsi alternatif "Gambar Berita".
- `<div class="content"> <h4>{{ item.title }}</h4>`, menampilkan judul berita (`item.title`).
- `Baca Selengkapnya`, menampilkan link ke halaman detail berita, dengan url sebagai parameter query string.
- `{% endfor %}`, menutup blok kartu dan blok perulangan berita dari Liputan6.

- <h2>Berita dari CNN Indonesia</h2> <div class="berita-container"> {% for item in cnn %}, judul bagian kedua: berita dari **CNN Indonesia**. Memulai loop untuk data dari cnn.
- <div class="berita-card"> , menampilkan masing-masing berita CNN dalam format kartu yang sama seperti Liputan6.
- <div class="content"> <h4>{{ item.title }}</h4>, menampilkan judul berita (item.title).
- Baca Selengkapnya, menampilkan link ke halaman detail berita, dengan url sebagai parameter query string.
- {% endfor %}, menutup blok kartu dan blok perulangan berita dari CNN Indonesia.

- templates detail_berita.html

- <!DOCTYPE html> <html lang="id">, mendeklarasikan dokumen HTML5 dan bahasa halaman sebagai Bahasa Indonesia.
- <meta charset="UTF-8">, mengatur encoding karakter ke UTF-8 agar mendukung karakter internasional.
- <title>{{ berita.title }}</title>, menampilkan judul berita sebagai judul halaman (dinamis dari variabel berita).
- <style> body { font-family, background-color, margin, padding }, mengatur font, warna latar belakang, dan padding global halaman.
- .container { max-width, margin, background-color, border-radius, box-shadow, overflow }, gaya utama kontainer berita: batas lebar, tengah, warna putih, border-radius, dan bayangan.
- .header-image { width, max-height, object-fit }, mengatur gambar header agar memenuhi lebar penuh, tinggi maksimum 400px, dan tidak terdistorsi.
- .content { padding }, menambahkan ruang di dalam konten utama.

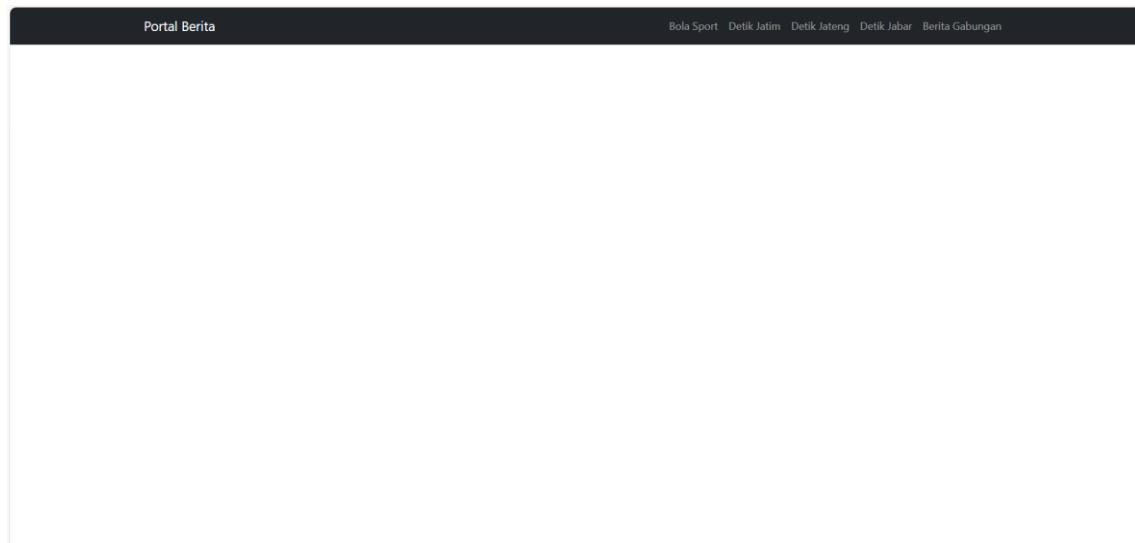
- `.content h1 { font-size, color, margin-bottom } .content p { font-size, line-height, color, margin-bottom }`, style untuk judul berita dan paragraf konten: ukuran, warna, dan spasi antar elemen.
- `.back-button { display, margin, background-color, color, padding, text-decoration, border-radius, transition }` `.back-button:hover { background-color }`, gaya untuk tombol kembali: warna biru, teks putih, padding, dan efek hover.
- `<body> <div class="container">`, elemen pembungkus utama untuk konten berita.
- `{% if berita.image %} {% endif %}`, jika ada gambar (berita.image tidak kosong), tampilkan gambar di atas konten.
- `<div class="content"> <h1>{{ berita.title }}</h1> {{ berita.content|safe }}`, menampilkan judul berita dan isi konten (berita.content) secara aman (boleh mengandung HTML).
- `← Kembali ke Halaman index`, menutup div kontainer, dan membuat tombol kembali ke halaman utama (/).

- templates (index.html)

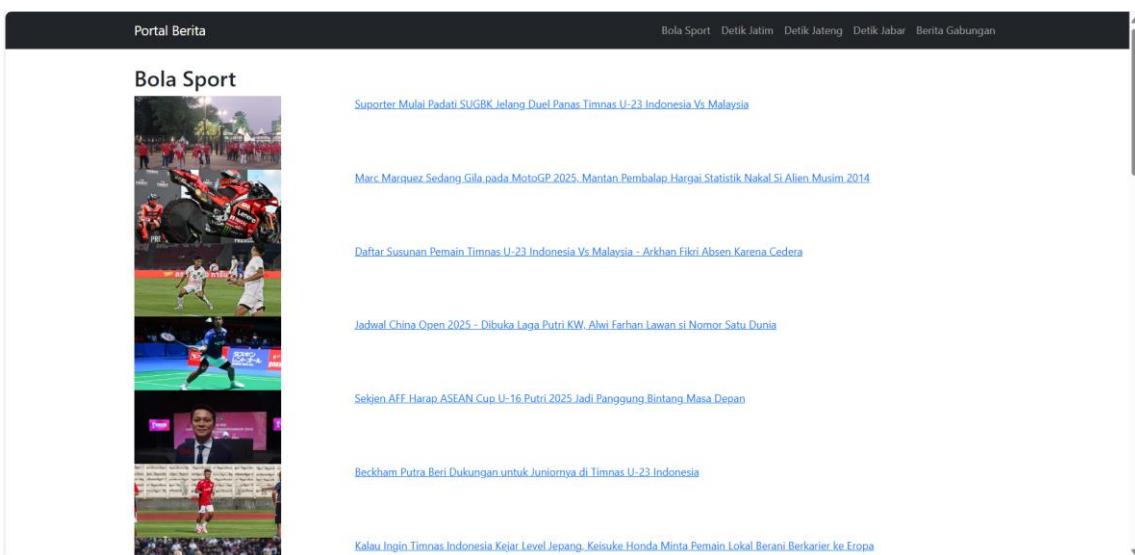
- `<!DOCTYPE html> <html lang="en">`, mendeklarasikan dokumen HTML5 dan menetapkan bahasa halaman ke **Bahasa Inggris** (en).
- `<meta charset="UTF-8">`, menetapkan encoding karakter menjadi UTF-8 agar mendukung karakter internasional dan simbol.
- `<title>Berita Terkini</title>`, menampilkan judul halaman browser sebagai "Berita Terkini".
- `<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">`, menyisipkan CDN Bootstrap 5.3 untuk mendukung komponen dan gaya CSS Bootstrap.

- <body> <nav class="navbar navbar-expand-lg navbar-dark bg-dark">, membuat navbar Bootstrap yang gelap (bg-dark) dan akan **melebar** di layar besar (navbar-expand-lg).
- <div class="container">, membungkus isi navbar dengan kelas container agar rata tengah dan tidak full-width.
- Portal Berita, link judul navbar yang akan membawa pengguna ke halaman utama (/), dengan gaya navbar-brand.
- <div class="collapse navbar-collapse">, container untuk elemen navbar yang bisa "collapse" (menyusut di layar kecil).
- <ul class="navbar-nav ms-auto">, daftar menu navigasi, ms-auto membuat menu berada di kanan (margin-start auto).
- <li class="nav-item">Bola Sport, item menu yang mengarah ke halaman /bola-sport.
- <li class="nav-item">Detik Jatim, link ke halaman kategori berita "Detik Jatim".
- <li class="nav-item">Detik Jateng, link ke halaman kategori berita "Detik Jateng".
- <li class="nav-item">Detik Jabar, link ke halaman kategori berita "Detik Jabar".
- <li class="nav-item">Berita Gabungan, link ke halaman kategori berita "Gabungan Berita".
- <div class="container mt-4">, container untuk isi utama halaman dengan margin atas (mt-4) agar tidak menempel ke navbar.
- {% block body %}{% endblock %}, placeholder Jinja2 untuk konten dinamis. Template lain akan mengisi blok ini (misalnya: detik_jatim.html).

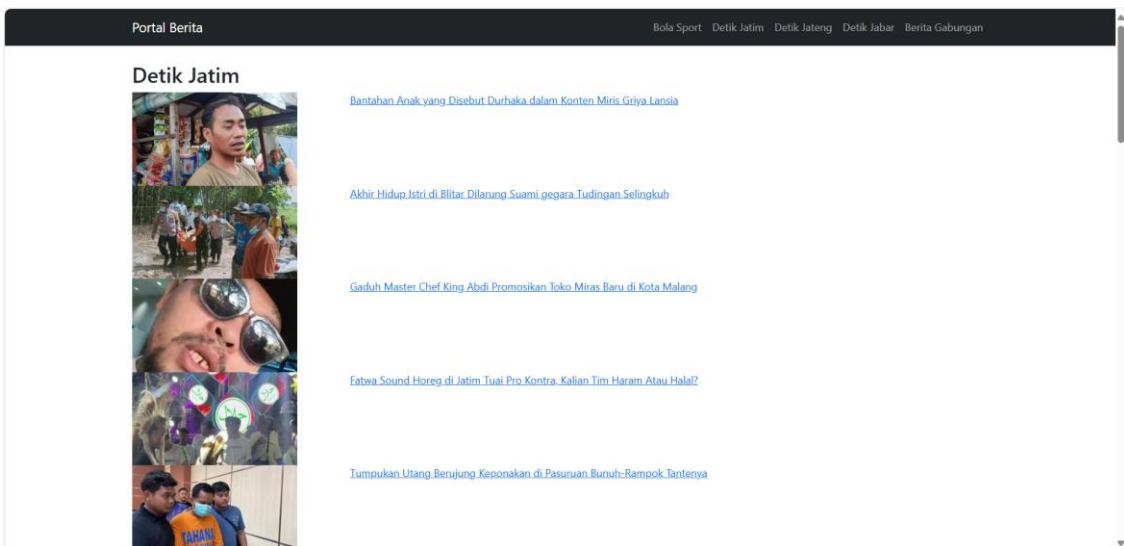
c. Hasil Uji Coba



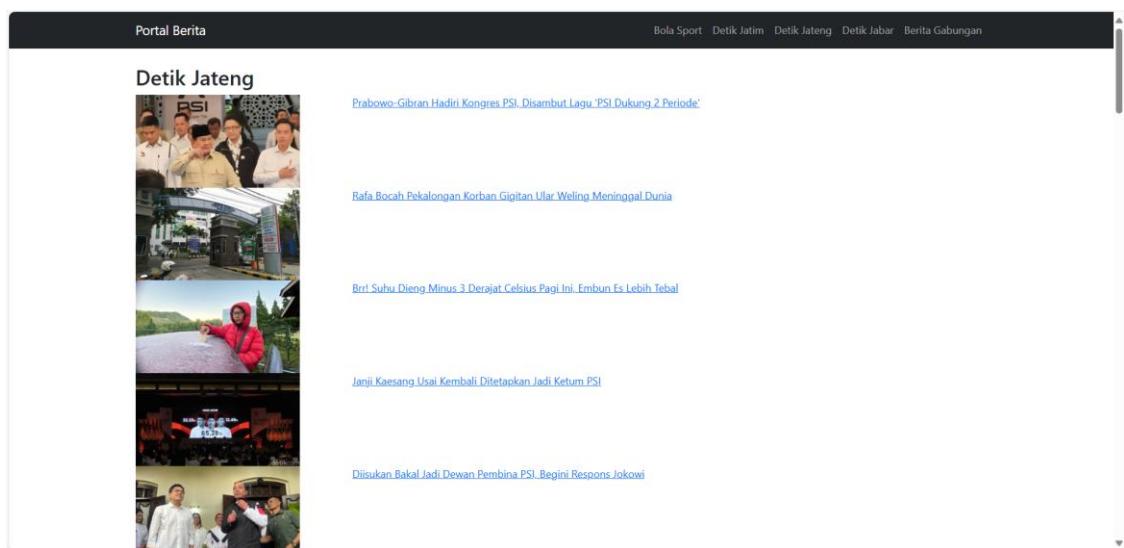
Gambar 8.1 Tampilan Halaman Index (index.html)



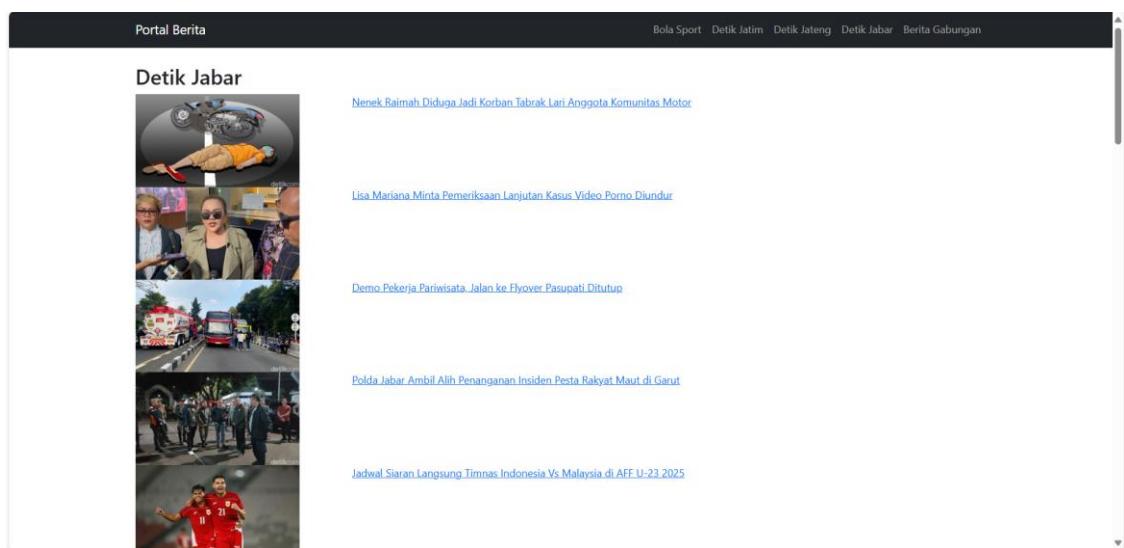
Gambar 8.2 Tampilan Halaman Bola Sport (bola-sport.html)



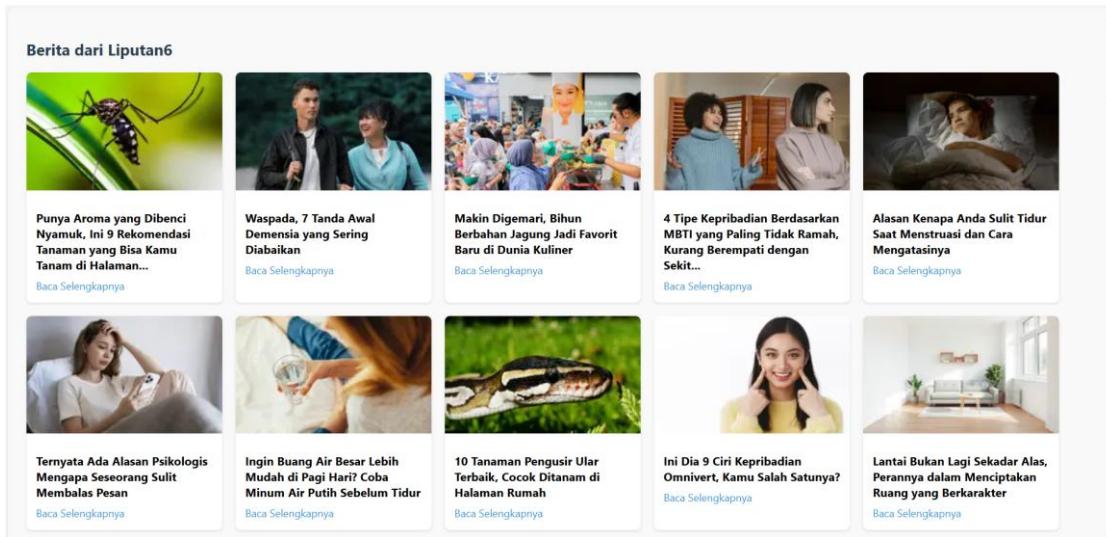
Gambar 8.3 Tampilan Halaman Detik Jatim (detik-jatim.html)



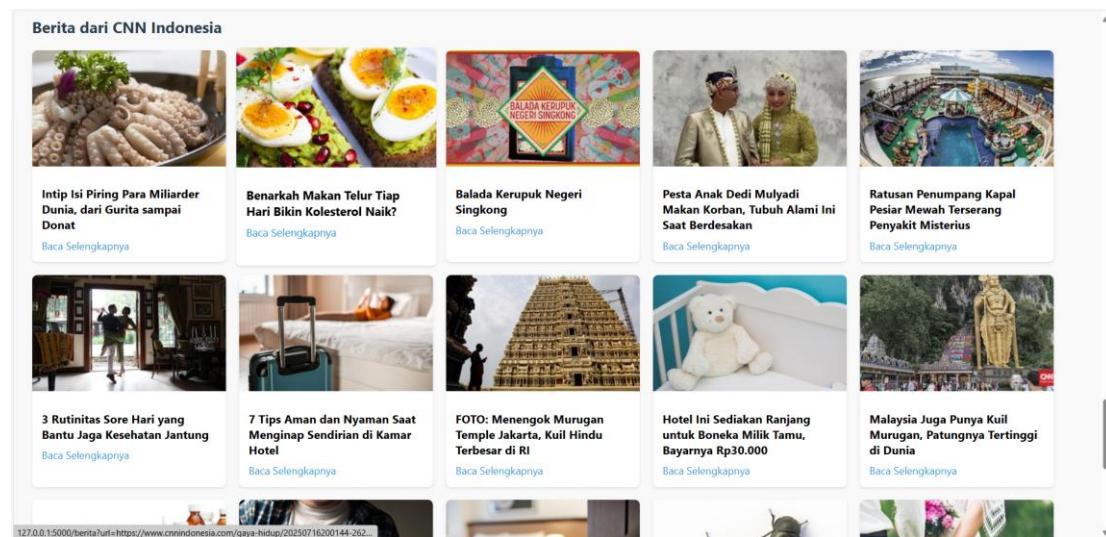
Gambar 8.4 Tampilan Halaman Detik Jateng (detik-jateng.html)



Gambar 8.5 Tampilan Halaman Detik Jabar (detik-jabar.html)



Gambar 8.6 Tampilan Halaman Berita Gabungan Liputan6 (berita_gabungan.html)



Gambar 8.7 Tampilan Halaman Berita Gabungan CNN Indonesia (berita_gabungan.html)



Gambar 8.8 Tampilan Halaman Detail Berita Liputan6 (detail_berita.html)



Gambar 8.9 Tampilan Halaman Detail Berita CNN Indonesia (detail_berita.html)

d. Analisa Hasil

Secara umum, sistem yang dibangun menunjukkan integrasi data scraping berita dari berbagai sumber. Template index.html berperan sebagai layout utama dengan navigasi navbar Bootstrap yang mengarahkan ke berbagai sumber berita seperti Bola Sport, Detik Jatim, Detik Jateng, Detik Jabar dan Berita Gabungan.

Tiap halaman turunan seperti detik_jatim.html mewarisi struktur dasar dari index.html dan menampilkan daftar berita dari sumber terkait dengan tata letak berbasis grid Bootstrap, memisahkan gambar dan judul secara proporsional. berita_gabungan.html menggabungkan dua sumber berita (Liputan6 dan CNN) dalam dua blok berbeda, menggunakan card dengan bayangan dan transisi hover yang meningkatkan tampilan profesional dan estetika.

Sementara itu, detail_berita.html berfokus pada tampilan satu berita secara lengkap, menyajikan gambar header, isi konten, dan tombol kembali yang fungsional. Penyaringan konten dengan |safe juga menunjukkan bahwa konten HTML dari hasil scraping ditampilkan langsung tanpa di-escape, yang berguna untuk mempertahankan format artikel asli. Struktur layout dan hasil scraping tampak berhasil diterapkan dengan baik tanpa error tampilan atau pemformatan,

mencerminkan bahwa template Jinja dan struktur data yang digunakan sudah sesuai dengan konteks scraping masing-masing situs berita.

8.5 Kesimpulan

8.5.1 Kesimpulan Percobaan 1

Percobaan ini berhasil menunjukkan bahwa integrasi antara scraping data berita dan penyajian antarmuka berbasis Flask serta template Jinja2 dapat berjalan dengan baik. Setiap halaman seperti bola-sport.html, detik_jatim.html, detik_jateng.html, detik_jabar.html, dan berita_gabungan.html mampu menampilkan hasil scraping dalam bentuk yang informatif dan terstruktur, baik gambar maupun judul berita. Template index.html berfungsi efektif sebagai kerangka dasar yang dapat diwarisi oleh halaman lain, sementara detail_berita.html memberikan tampilan konten berita secara lengkap. Antarmuka pengguna menggunakan Bootstrap membuat tampilan lebih rapi dan responsif. Dari sisi fungsionalitas, sistem mampu menavigasi antarhalaman dan menampilkan konten sesuai sumber masing-masing. Hal ini membuktikan bahwa alur pengambilan data, pemrosesan, dan rendering di sisi frontend telah berjalan dengan baik tanpa error atau bug tampilan yang berarti.

Mengetahui:

Dosen Pengampu Mata Kuliah

Arif Hadi Sumitro , M.Kom

NIKP. xxx

DAFTAR PUSTAKA

1. Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python*. O'Reilly Media.
2. Mitchell, R. (2018). *Web Scraping with Python: Collecting Data from the Modern Web* (2nd ed.). O'Reilly Media.
3. Sonmez, E., & Dede, E. (2020). Web Scraping and Data Integration in Python Using Flask Framework. *International Journal of Engineering Research and Technology (IJERT)*, 9(5).