MODUL 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
import fungsi
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>
        </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 berhasill 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 HTML-nya.
- **soup = BeautifulSoup(response.text, "html.parser"),** memparsing 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 kef file HTML **(index.html)** untuk ditampilkan sebagai halaman web.
- **If name == "main":app.run(debug=True),** menjalankan server lokal Flask jika dile dijalankan langsung. Mode debug aktif agar perubahan kode otomatis ter-update saat dijalankan.

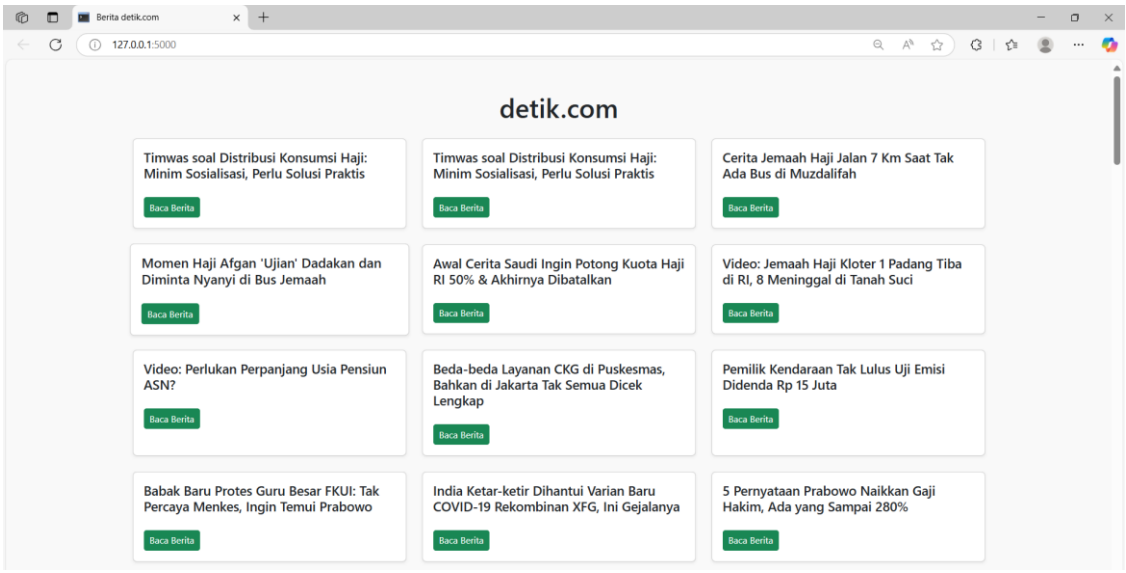
fungsi.py

- **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(directoy_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.