亞洲大學
ASIA UNIVERSITY

Midterm Project Report

Advanced Computer Programming

Student Name	:	Vija Wildan Gita Prabawa
Student ID	:	112021221
Teacher	:	DINH-TRUNG VU

2024-04

Chapter 1 Introduction

1.1 Github

1) **Personal Github Account**: https://github.com/vijawildan
2) **Group Project Repository**: https://github.com/HanifiSetiawan/ACP-Group-06/tree/main

1.2 Overview

This project uses 'Scrapy' which is a python framework that is used for web scraping. This project is focused on scraping a github repository where for the example will be using my github repository.

The goal of this project is to extract the metadata of a github profile, specifically:

- Repository URL
- About section or fallback name
- Last updated timestamp
- Programming languages used
- Number of commits

All of these are going to be extracted into an XML file where we can see all the scraped data in the file.

Since we're using a feature from the scrapy framework, we need to use the following command in terminal to create the project:

```
scrapy startproject githubscraper
cd githubscraper
```

Implementation

1.1 Class 1: scrapy.Spider

This class is taken from the scrapy framework that needed to use some of the functions.

1.1.1 Methods and Functions

- `start_requests()`: To start the scraping from the url
- `response.follow(url, callback=...)`: To follow the links and it will handle all the HTTP requests

1.2 Class 2: GithubSpider

```
class GithubSpider(scrapy.Spider):
    name = "github_spider"
    allowed_domains = ["github.com"]
    start_urls = ["https://github.com/vijawildan?tab=repositories"]

    def parse(self, response):
        repo_links = response.css('div.d-inline-block h3
a::attr(href)').getall()
        for link in repo_links:
            yield response.follow(url=link, callback=self.parse_repo)

    def parse_repo(self, response):
        repo_url = response.url
        name = repo_url.split('/')[-1]

        about = response.css('p.f4.my-3::text').get()
        about = about.strip() if about else None

        is_empty = response.css('div.Box-body p::text').re_first(r"This
repository is empty")

        if not about and not is_empty:
            about = name

        updated = response.css('relative-time::attr(datetime)').get()

        if not is_empty:
            languages = response.css('ul[data-testid="repo-language-list"] li
span::text').getall()
            commits = response.css('li.commits a
span::text').re_first(r'\d+')
        else:
            languages = None
            commits = None

        yield {
            'url': repo_url,
            'about': about,
            'last_updated': updated,
            'languages': languages,
            'commits': commits
        }
```

This is the class that needed to scrap the github profile using the scrapy framework.

1.2.1 Fields

- `name` : Unique spider name used to run the scraper (scrapy crawl github_spider)
- `allowed_domains` : To limit the domain only for github
- `start_urls` : Initial URL for the spider to start scraping

1.2.2 Methods and Functions

- `def parse(self, response):`

  ```
          repo_links = response.css('div.d-inline-block h3
  a::attr(href)').getall()
          for link in repo_links:
              yield response.follow(url=link,
  callback=self.parse_repo)
  ```

  - Select all the repositories using CSS selector
  - Entry point for crawling, gets list of repos
  - To pass the detailed scraping into the next function

- `def parse_repo(self, response):`

  ```
          repo_url = response.url
          name = repo_url.split('/')[-1]

          about = response.css('p.f4.my-3::text').get()
          about = about.strip() if about else None

          is_empty = response.css('div.Box-body
  p::text').re_first(r"This repository is empty")

          if not about and not is_empty:
  ```

```
                about = name

            updated =
response.css('relative-time::attr(datetime)').get()

            if not is_empty:
                languages =
response.css('ul[data-testid="repo-language-list"] li
span::text').getall()
                commits = response.css('li.commits a
span::text').re_first(r'\d+')
            else:
                languages = None
                commits = None

            yield {
                'url': repo_url,
                'about': about,
                'last_updated': updated,
                'languages': languages,
                'commits': commits
            }
```

- This function handles each individual repository page and extracts detailed info
- Parses each repo page for details

.

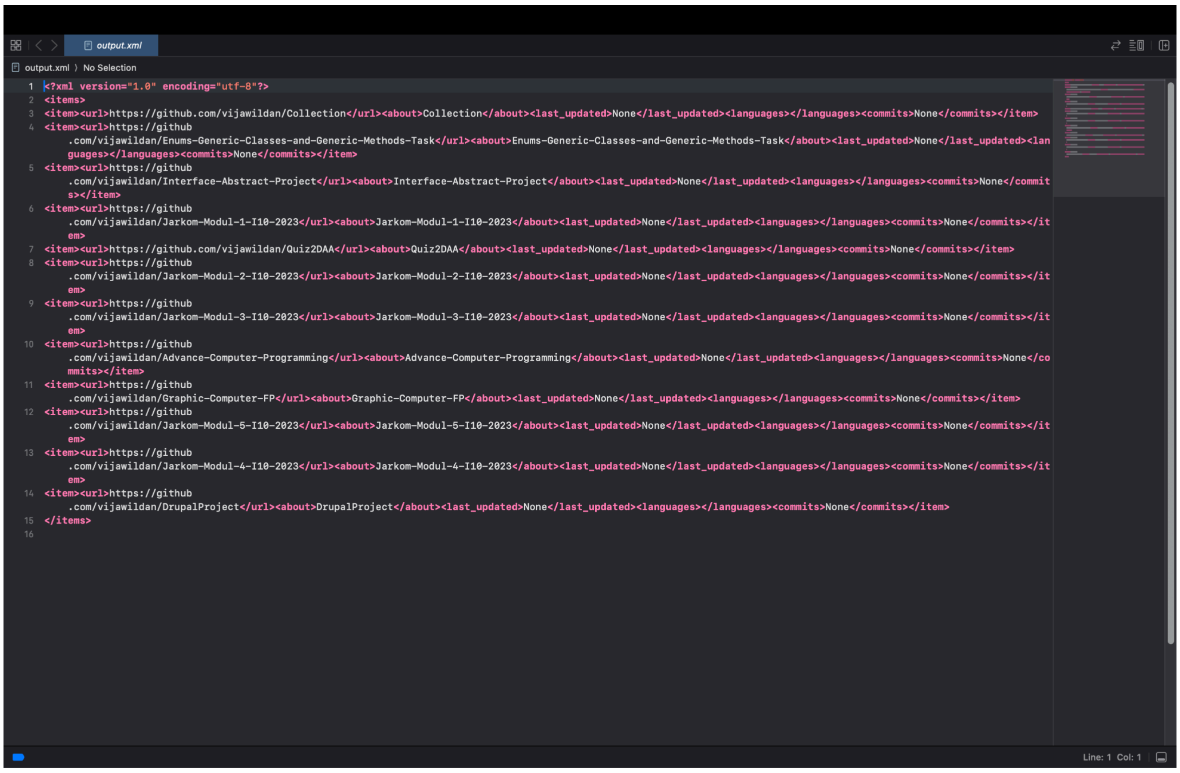
Chapter 2 Results

1.1 Result

To run the code and get the result, the following command line is used on terminal:

```
scrapy crawl github_spider -o output.xml
```

After we run the command, we will get the result from our code. The example of the output will be as following:



```
<?xml version="1.0" encoding="utf-8"?>
<items>
<item><url>https://github.com/vijawildan/Collection</url><about>Collection</about><last_updated>None</last_updated><languages></languages><commits>None</commits></item>
<item><url>https://github.com/vijawildan/Enums-Generic-Classes-and-Generic-Methods-Task</url><about>Enums-Generic-Classes-and-Generic-Methods-Task</about><last_updated>None</last_updated><languages></languages><commits>None</commits></item>
<item><url>https://github.com/vijawildan/Interface-Abstract-Project</url><about>Interface-Abstract-Project</about><last_updated>None</last_updated><languages></languages><commits>None</commits></item>
<item><url>https://github.com/vijawildan/Jarkom-Modul-1-I10-2023</url><about>Jarkom-Modul-1-I10-2023</about><last_updated>None</last_updated><languages></languages><commits>None</commits></item>
<item><url>https://github.com/vijawildan/Quiz2DAA</url><about>Quiz2DAA</about><last_updated>None</last_updated><languages></languages><commits>None</commits></item>
<item><url>https://github.com/vijawildan/Jarkom-Modul-2-I10-2023</url><about>Jarkom-Modul-2-I10-2023</about><last_updated>None</last_updated><languages></languages><commits>None</commits></item>
<item><url>https://github.com/vijawildan/Jarkom-Modul-3-I10-2023</url><about>Jarkom-Modul-3-I10-2023</about><last_updated>None</last_updated><languages></languages><commits>None</commits></item>
<item><url>https://github.com/vijawildan/Advance-Computer-Programming</url><about>Advance-Computer-Programming</about><last_updated>None</last_updated><languages></languages><commits>None</commits></item>
<item><url>https://github.com/vijawildan/Graphic-Computer-FP</url><about>Graphic-Computer-FP</about><last_updated>None</last_updated><languages></languages><commits>None</commits></item>
<item><url>https://github.com/vijawildan/Jarkom-Modul-5-I10-2023</url><about>Jarkom-Modul-5-I10-2023</about><last_updated>None</last_updated><languages></languages><commits>None</commits></item>
<item><url>https://github.com/vijawildan/Jarkom-Modul-4-I10-2023</url><about>Jarkom-Modul-4-I10-2023</about><last_updated>None</last_updated><languages></languages><commits>None</commits></item>
<item><url>https://github.com/vijawildan/DrupalProject</url><about>DrupalProject</about><last_updated>None</last_updated><languages></languages><commits>None</commits></item>
</items>
```

.

Chapter 3 Conclusions

This project efficiently utilizes a Scrapy-based web crawler to crawl the data of a user GitHub repository. By navigating from a central GitHub profile page, the spider efficiently collects repository URLs, descriptions (About), last updated timestamps, languages used, and the number of commits. It handles edge cases such as empty repositories by assigning default or none values where it applicable, ensuring reliability in the data extraction process. With using scrapy, the base of it also include the function of parse and parse_repo that makes it maintainable and scalable.