



**Midterm Project Report**

**Advanced Computer Programming**

**Student Name : Nomin Javkhlanbayar**

**Student ID : 113021190**

**Teacher : DINH-TRUNG VU**

**2024-04**

# Introduction

## Github

1. **Personal Github Account**: <https://github.com/jnomin>
2. **Group Project Repository**: <https://github.com/jnomin/GroupAdvancedComputerProgramming>

## Overview

In this project, I have developed a web scraping solution to extract detailed information about repositories from a GitHub user's profile page. The primary goal was to gather repository data such as the URL, description, languages used, and number of commits for each repository in an efficient and organized manner.

In my program, I used several advanced Python features and libraries to effectively scrape data from GitHub. The core of my program is the **Scrapy** framework, a powerful web scraping tool that handles requests, parsing, and data extraction efficiently, allowing me to navigate between pages and extract structured data from GitHub. I relied on **XPath** and **CSS selectors** to target specific elements on the GitHub pages, with XPath being used to extract dynamic data like commit counts, and CSS selectors helping to gather repository URLs, descriptions, and last updated times. To ensure smooth data flow and organization, I utilized Scrapy's **meta feature** to pass extracted data between different scraping methods. Additionally, I applied **regular expressions** to handle variations in data formatting, such as extracting commit counts and filtering programming languages used in each repository.

# Implementation

## Class 1 - GithubScraperItem

RepositoryScraper is responsible for fetching repository metadata, such as the URL, description, languages used, last updated time, and number of commits from a given GitHub user's public profile. It then serializes this data into a well-formatted XML file.

### Fields

The GithubScraperItem class is a data container used to hold the information about each repository. It contains the following fields:

* **url**: A string field to store the URL of the repository.
* **about**: A string field to store the description of the repository.
* **last\_updated**: A string field to store the last updated timestamp of the repository.
* **languages**: A list of strings to store the programming languages used in the repository.
* **commits**: A string field to store the number of commits for the repository.

### Methods

The **GithubScraperItem** class is a data class and does not contain complex methods. It primarily serves as a container for the data extracted during the scraping process.

### Function 1

The class itself does not contain specific functions but relies on Scrapy's framework for interaction with the data. The **GithubScraperItem** class is used to hold data temporarily while it's being processed and passed through the Scrapy spider.

## Class 2 - GithubReposSpider

The **GithubReposSpider** class is responsible for scraping the repository data from the GitHub profile page. This class is defined using Scrapy's spider class and manages how requests are made, responses are handled, and data is extracted.

## Method/Function 1 - parse

The **parse** method is the main function of the spider that is executed when the initial request is made to the GitHub profile page. This method uses CSS selectors to extract the repository URLs, descriptions, and other relevant data. It also initiates further requests to each individual repository's page for more detailed information, passing along the current data using Scrapy's **meta** feature.

## Method/Function 2 - parse\_repo\_details

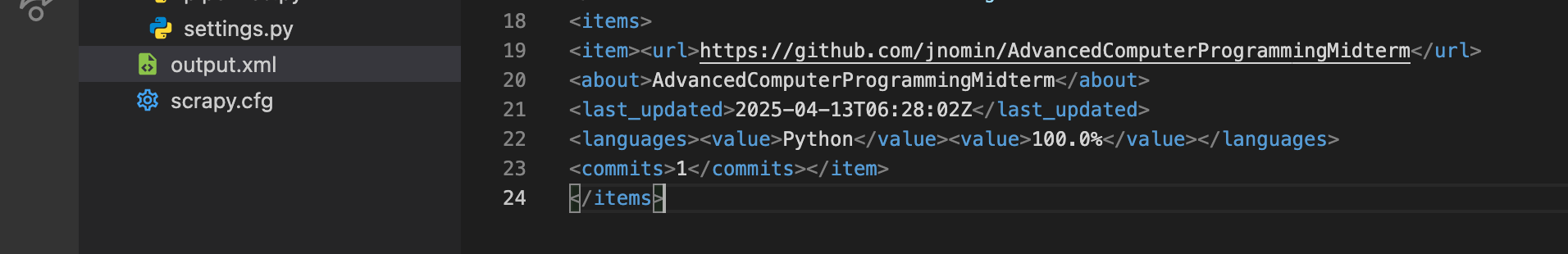
The **parse\_repo\_details** method is called when Scrapy returns the response from a repository's page. It extracts additional details such as the programming languages used and the number of commits. If this information is available, it is added to the item object. If not, default values are used. The method then yields the completed item containing all the repository data.

.

# Results

The output of my web scraping program is structured in XML format, with each <item> tag representing a repository scraped from the specified GitHub user profile. The program successfully extracted the relevant information from the repository, demonstrating the accuracy and functionality of the implemented spider.

In the current result, the scraper collected data from one repository:



The result confirms that the scraper functions as expected, accurately navigating to the user's GitHub profile, parsing the necessary data from both the main repositories page and the individual repository page, and outputting it in a clean, structured XML format. This format is suitable for further processing, analysis, or integration with other systems.

.

# Conclusions

In this project, I successfully developed a GitHub repository scraper using the Scrapy framework in Python. The goal was to automate the extraction of important information such as repository URLs, descriptions, last updated dates, programming languages, and the number of commits from a GitHub user's profile. Through the use of advanced Python features including CSS selectors, XPath, meta data handling, and regular expressions, the program was able to navigate through pages, extract structured data, and output it in a clean XML format.

The results demonstrate that the scraper performs accurately and efficiently. It was able to gather and organize all the targeted repository information from the GitHub user jnomin, producing output that can be reused for further data analysis or integration into larger applications.

Overall, this project highlights the power of Scrapy for web scraping tasks and showcases how advanced programming techniques can be combined to solve real-world problems such as data extraction and automation. The implementation is also flexible and can be easily extended to scrape more fields or adapted to work with different GitHub profiles.