



**Midterm Project Report**

**Advanced Computer Programming**

**Student Name : Bayarmagnai Munkhbat**

**Student ID : 113021195**

**Teacher : DINH-TRUNG VU**

**2025-04**

# **1. Introduction**

## **1.1 Github**

1. **Personal Github Account**: [Jantsagdorj (113021195)](https://github.com/Jantsagdorj)
2. **Group Project Repository**: [Jantsagdorj/ACP-AU-1132: Project for Advanced Computer Programming Course - AU-1132 - Group](https://github.com/Jantsagdorj/ACP-AU-1132)

## **1.2 Overview**

The goal of this project was to build a **GitHub repository scraper** using the **Scrapy** framework in Python. The scraper was designed to automate the extraction of key repository information, such as:

* **URL** of the repository
* **Description** (or "about") of the repository
* **Last updated** timestamp
* **Programming languages** used  
  **Number of commits**

This data was collected from the GitHub user profile and organized into a clean **XML format** for easy analysis or integration with other systems.

## **2. Implementation**

### **Class 1 - GithubScraperItem**

**Purpose:** The GithubScraperItem class is used to store the repository data, including the repository's URL, description, last updated time, languages used, and number of commits. It serves as a container for data extracted during the scraping process.

**Fields:**

* **url:** Repository URL
* **about:** Description of the repository
* **last\_updated:** Last updated timestamp
* **languages:** List of programming languages used
* **commits:** Number of commits

### **Class 2 - GithubReposSpider**

**Purpose:** The GithubReposSpider class is responsible for scraping repository metadata from a given GitHub user's profile.

### **Method 1 - parse**

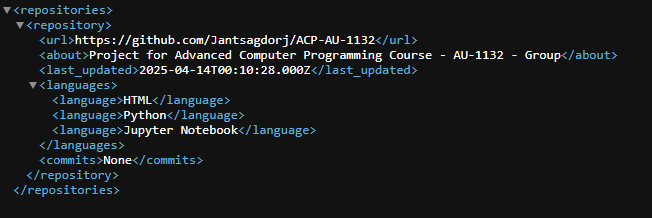
* This method is triggered when the initial request is made to the GitHub profile page.  
  It uses **CSS selectors** to extract repository URLs and initiates further requests to each repository's page for additional details.
* **Method 2 - parse\_repo\_details**
* This method extracts further information from each individual repository, including the description, last updated date, languages, and number of commits.  
  If any of the data is missing, default values is None
* The method yields the collected data as a GithubScraperItem.

### **XML Export**

The extracted data is saved in an **XML file** (github\_repos.xml) using the xml.etree.ElementTree module.

**3. Results**

The output of the web scraping program is an **XML file**, with each <repository> tag representing a scraped repository. The scraper successfully extracted key information from the GitHub profile, including the repository URL, description, last updated time, programming languages, and number of commits.



For the current test, the scraper successfully collected data from one repository, confirming the program's accuracy and functionality. The data was parsed from both the main GitHub profile page and the individual repository page, and then output in a structured XML format, making it suitable for further analysis or integration with other systems.

## **4. Conclusions**

In this project, I developed a GitHub repository scraper using Python's Scrapy framework to automate the extraction of essential information from a GitHub user's profile. The scraper retrieves repository URLs, descriptions, last updated dates, languages, and commit counts.

By leveraging CSS selectors and meta data handling, the program navigated pages, extracted data, and saved it in a clean XML format. The results demonstrate the scraper's effectiveness in gathering data accurately and organizing it for further use.

This project highlights the power of Scrapy for web scraping and automation. The implementation is flexible, scalable, and can be easily extended to scrape additional fields or adapted for different GitHub profiles.