

****

**Midterm Project Report**

**Advanced Computer Programming**

**Student Name : Ervando Vincen Wang**

**Student ID :113021129**

**Teacher : DINH-TRUNG VU**

**2024-04**

# Introduction

## Github

1. **Personal Github Account**: <https://github.com/SatelitKaca>
2. **Group Project Repository**: <https://github.com/113021134/acp-1132>

## Overview

The Scrapy framework, JSON handling, regular expressions, and GitHub REST API were used in my program. My program extracts information about repository name, URL, About, Last Updated date, Languages, and Number of Commits of repositories on the page <https://github.com/SatelitKaca>. I used CSS selectors to extract HTML elements, regular expressions to parse URLs, and meta fields to pass data between chained API requests. My spider combines web scraping and API data and exports the final results in structured format (e.g., XML or JSON).

# Implementation

#### **Class 1: GithubSpider**

**Description**:  
This class performs the full crawl and data extraction workflow. It starts from the GitHub profile and uses both web scraping and REST API calls to gather complete repository information.

### Fields

**Fields**:

* name: Defines the name of the spider ("github")
* allowed\_domains: Limits the spider to github.com and api.github.com
* start\_urls: Begins at <https://github.com/SatelitKaca?tab=repositories>

### Methods

**Methods**:

* parse(self, response): Extracts initial repository data and starts API requests
* parse\_commits(self, response): Parses commit history from the API
* parse\_languages(self, response): Parses languages used in the repository

**Method 1: parse(self, response)**

**Description**:  
This method scrapes the GitHub repositories on the user's profile page. It extracts each repo’s name, URL, description (or falls back to the repo name), and last updated date. Then it creates API requests for commit and language data, passing along metadata.

**Method 2: parse\_commits(self, response)**

**Description**:  
This method receives the commit data as a JSON array, counts the number of commits, and initiates a request to the GitHub languages API with this count included in the metadata.

**Method 3: parse\_languages(self, response)**

**Description**:  
This final method adds the language list to the previously collected data and yields the complete repository record.

# Results

**Result 1**

The program successfully scrapes all repositories from https://github.com/SatelitKaca, extracting:

* Repository name
* Repository URL
* Description
* Last updated date
* Languages used
* Number of commits  
  The data is structured and can be saved in XML or JSON format.

**Result 2**

The spider handles edge cases:

* If a repository lacks a description, the name is used as fallback.
* Empty repositories return None for languages and commit counts. This makes the spider resilient and suitable for real-world data collection.

.

# Conclusions

This project shows practical knowledge in using Python for data collection, focusing on real-time web data extraction. It integrates multiple technologies: HTML parsing, API handling, and asynchronous crawling. The final spider is modular, maintainable, and handles missing data gracefully, making it a strong example of advanced web scraping techniques.