

****

**Midterm Project Report**

**Advanced Computer Programming**

**Student Name : Duckens BELVAL**

**Student ID : 113021142**

**Teacher : DINH-TRUNG VU**

**2024-04-14**

# Introduction

## Github

1. **Personal Github Account**: https://github.com/Duckens03
2. **Group Project Repository**: https://github.com/ArifSabdho/Test

## Overview

Brief description of advanced language features and libraries used in your work as well as results you have archieved. For example:

The data class, pprint, pattern matching, regulation expresion, and Beautiful Soup have used in my program. My program have extracted information about URL, About, Language, Number of Commits of repositories on the page <https://github.com/trungvdhp>.

This project utilizes scrapy framework in python to systematically scrape repository information from my GitHub repository page. Key python features and libraries employed include:

Scrapy framework: for web crawling and scraping.

Scrapy.items: to define a structure data container (GitHubRepoItem) for the scrapped information.

Css selectors: used within the spider to locate and extract specific HTML elements containing the desired data.

Scrappy settings: configured for XML export (feeds), polite scraping (Download\_delay, Autothrottle\_Enabled) and spider identification.

Items exporter: (XmlItemsExporter): used via settings to automatically arrange scrapped items into an XML file.

# Implementation

## Class 1

GithubRepoItem: this class is a data structure that represents a single GitHub repository it’s used to store the scraped information for each repository.

### Fields

url: the URL of repository.

about: the description or “about” text of repository.

Last\_updated: the timestamp of the last update to the repository

Languages: a dictionary containing the language used in the repository and their percentages.

Num\_commits: the number of commits.

Repo\_name: a temporary filed used to store the repository name for the “about” logic.

## Class 2

GitHub Spider: this class define the scraping logic. It tells Scrapy how to navigate the website, extract data, and handle pagination.

## Methods:

Parse(self, response): this is the main scraping function that starts the process. It handles the initial page with the list repositories, extracts basic information, and create new requests to scrape repository pages.

Parse\_repo\_page(self, response): this function is called for each individual repository page. It extracts more detailed information like languages and commits, applies the “about” the logic, and yield the final scraped data for the repository.

## Fields:

Name: a unique name for the spider (github repositories in this case).

Start\\_urls: a list of URLs to start scrapping from (the Github’s repositoy page).

Custom\\_settings: a dictionary of settings to customize the spider’s behavior (like download delays, output format, atc.)

## Methods:

**1-Parse(self, response):**

Start by logging that it’s parsing the repository list page.

Defines CSS selectors to locate repository elements on the page.

Iterates through each repository element:

Extracts the repository name, URL, about text, and last updated time.

Creates new request to scrape the individual repository page using (parse\_repo\_page).

Handle pagination by looking for a next page link and creating a new request if found.

**Yields:**

Scrapy.request objects for individual repository pages.

scrapy.request objects for the next page of repositories (if pagination is present).

**2- parse\_repo\_page(self, response):**

Retrieve the GithubRepoItem objected created in the parse method.

Extracts the languages used and the number of commits using CSS selectors.

Applies logic to fill in the “about” field if it’s missing and the repository is not considered empty.

Removes the temporary repo\_name field.

Yields the final GithubRepoItem object with all scraped data.

Yields:

The completed GithubRepoItem object for the current repository.

# Results

## Result 1:

The scrapy spider successfully crawls the GitHub user’s repository pages, including handling pagination.

**Result 2:**

For each repository, the spider accurately extracts the URL, About description (applying the specified fallback logic), Last Updated timestamp Languages or (None) or number of commits (or None).

**Result 3:** All exctracted data is aggregated and saved into a well structured XML file name ‘repositories.XML in the scrapy’s project root directory.

Conclusion

# This project demonstrates the capability of scrapy framework to automate the extraction of specific, structured information from web pages, in this case, GitHub repository details. The spider correctly implements the required logic for handling potential missing descriptions and differentiating between active and potentially empty repositories based on available meta data. The outputted XML file provides a clean dataset suitable for further analysis.