Midterm Project Report

Advanced Computer Programming

Student Name : Arif Sabdho Nugrahadi
Student ID : 113021204
Teacher : DINH-TRUNG VU

2025-04

Chapter 1 Introduction

1.1 Github

1) **Personal Github Account**: ArifSabdho
2) **Group Project Repository**: Team Duck (https://github.com/GH-Team-Duck/MidTerm_Project)

1.2 Overview

GitHub has become one of the most prominent platforms for developers to host, share, and collaborate on code repositories. With the increasing number of open-source projects available, gaining insights into repositories by automating data collection becomes highly valuable. In this project, I have developed a Python-based web scraper that can systematically extract key information from GitHub topic pages, focusing on repositories related to "Arduino", "Bitcoin", and "Chrome". The scraper collects details such as repository names, descriptions, star counts, programming languages, last updated timestamps, and total commits. The collected data is stored in a structured XML format for easy analysis and integration.

The goal of this midterm project is to develop an effective and maintainable scraping solution by integrating numerous external libraries and investigating a number of sophisticated Python programming capabilities. The core components and technologies used include:

a. **BeautifulSoup**: This library is used for parsing HTML content and navigating the DOM tree to extract specific tags and attributes.

b. **Requests**: Used to make HTTP GET requests to GitHub pages and APIs.

c. **Regular Expressions (re)**: Utilized to parse GitHub's pagination headers when determining the total number of commits for each repository.

d. **XML Writing**: Data is saved using the xml.etree.ElementTree and xml.dom.minidom libraries, which provide structured XML formatting.

e. **Rate-Limiting Consideration**: A polite delay of one second between requests is added using time.sleep(1) to avoid overwhelming GitHub's servers.

The script extracts all required metadata from the HTML content of GitHub topic pages and enriches it with additional information (such as commit counts) by accessing the GitHub API. The script is designed to be easily reusable and expandable for other topics or repository types.

Chapter 2 Implementation

2.1 Class 1

Description of the main scraping class behavior. Functions are modular and mimic the use of a class by separating responsibilities clearly.

2.1.1 Fields

`GITHUB_TOKEN` is a field to store optional GitHub API token used to bypass the rate-limit

2.1.2 Methods

- `get_headers()` : Returns HTTP headers for requests, including the authorization token if available.
- `parse_star_count()` : Converts abbreviated star counts (e.g. '1.2k') into integer values.
- `get_commit_count` : Fetches the number of commits from the GitHub API using pagination info.
- `save_to_xml`: Converts the scraped data in a DataFrame into a formatted XML file.

2.1.3 Functions

- `get_repo_info()` : Gathers repo name, description, languages, update time, and commit count.
- `get_topic_page()` : Downloads and parses a GitHub topic page.
- `get_topic_repos()` : Iterates over all found repos and compiles a structured list into a DataFrame.
- `scrape_topic()` : Combines all above steps to process a single topic.
- `scrape_arduino_and_bitcoin()` : Wrapper function to scrape multiple topics (Arduino, Bitcoin, Chrome).

2.2 Class 2

No separate object-oriented class was defined, but the modular function group collectively behaves like a class.

2.3 Function 1

`get_repo_info()` This function plays a central role by aggregating all relevant metadata per repository, such as repo name, star count, About section, update timestamp, languages used, and commit count (via API).

2.4 Function 2

`scrape_topic()` This is the high-level function that controls the flow of the scraping process for each topic. It handles checking if output already exists, collecting and parsing HTML, and saving data.

Chapter 3 Results

After the full execution of the program, the web scraper successfully collected and compiled metadata for repositories listed under three GitHub topic pages: Arduino, Bitcoin, and Chrome. The script ran without errors, respecting rate limits by introducing polite delays between requests and ensuring data integrity through structured extraction and validation.

A related XML file was generated for each topic. Twenty GitHub repositories, chosen from the first page of the corresponding subject listings, are included in each XML file. The structure of the output ensures uniformity and contains detailed information per repository, such as the GitHub username, repository name, direct URL, total number of stars, a brief "About" description, last updated date (formatted in ISO datetime), a comma-separated list of programming languages used in the repository, and the total number of commits retrieved via the GitHub API.

The collected data provides valuable insight into the open-source ecosystem for each topic. For example, the Arduino repositories primarily focused on IoT firmware, embedded systems, and hardware control libraries. Meanwhile, Bitcoin repositories include highly starred crypto trading bots, blockchain implementations, and Bitcoin Core itself—demonstrating deep community interest and high development activity. The Chrome category spans a wide range of automation tools, browser extensions, and JavaScript-based UI frameworks, showcasing the versatility of Chrome-related development.

Chapter 4 Conclusions

This project demonstrates Python's strength in data organization and web scraping. By integrating BeautifulSoup, Requests, and GitHub’s API, we built a solution that gathers comprehensive metadata from GitHub repositories. The data is stored in a structured XML format suitable for future analysis.

I gained knowledge about the value of modular code architecture, controlling API restrictions, and handling inconsistent data on public websites as a result of this experience. Furthermore, a practical grasp of intermediate-to-advanced programming abilities is demonstrated by the integration of numerous Python libraries.