Educational content aggregator with an open API*

Serhii Yevseiev[1†], Hanna Zavolodko[1†*], Kostiantyn Foksha[2†], Valerii Zavolodko[2†] and Iryna Aksonova[1†]

[1] *National Technical University "Kharkiv Polytechnic Institute", Kharkiv, Ukraine*
[2] *Individual Entrepreneur, Kharkiv, Ukraine*

Abstract

The article presents a complete architecture and implementation of a web-based educational content aggregator system that collects, structures, and updates information from open sources via an open API. The relevance of the topic is due to the rapid growth in the number of online courses and learning platforms, which creates a need for tools for easy navigation and personalized access to educational resources. The paper substantiates technical and methodological approaches to the creation of such a system, including the implementation of a Python/Django-based backend, the use of the MariaDB database, web crawling modules, and the creation of a mobile application in Flutter. The proposed architecture involves a modular approach using interfaces for connecting to different platforms, which allows the system to scale and easily adapt to new sources. At the center of user interaction is an aggregator that analyzes the request, checks the availability of relevant data in the database, and, if necessary, launches procedures for collecting and updating information. Special attention is paid to the logic of parsers that can adapt to different web page structures, which ensures the solution's versatility. Multilevel functionality testing was conducted on the example of four educational platforms: Coursera, Alison, Sololearn, and Edx. The analysis covers processing speed, amount of processed data, parsing accuracy, and error tolerance. The results demonstrated stability of the system, especially when working with platforms with clearly structured content. The developed aggregator can be integrated into broader EdTech ecosystems, including the state level, and used to build personalized educational trajectories. The system is able to serve both formal and informal educational requests of users, promoting the spread of digital skills, openness of education and improving its quality.

Keywords

content aggregator; web service; open API; educational platforms; web crawling; cloud technologies; mobile application; data parsing; Python/Django; digital education

1. Introduction

In today's digital environment, aggregator sites play an important role in providing access to a large amount of information from various sources in a unified manner. As noted by Matyash D. [1], Google actively indexes aggregator sites, as they provide users with aggregated data that simplifies search and decision-making.

The main technical tool for implementing aggregators is web crawling, an automated process of collecting data from web pages. A detailed explanation of the principles of crawling, as well as methods of robot management, is presented in [2]. Along with this, Henderson A. [3] provides an overview of modern free web crawling tools that allow you to effectively collect and update information in aggregator databases.

An important role in building aggregators is also played by the use of APIs - application programming interfaces that allow you to receive structured data directly from source servers. IBM

Proceedings of the Workshop on Scientific and Practical Issues of Cybersecurity and Information Technology at the V international scientific and practical conference Information security and information technology (ISecIT 2025), June 09–11, 2025, Lutsk, Ukraine
* Corresponding author.
† These authors contributed equally.
Serhii.Yevseiev@gmail.com (S. Yevseiev); anna.zavolodko@khpi.edu.ua (H. Zavolodko); kfoksha58@gmail.com (K. Foksha); vals2004@gmail.com (V. Zavolodko); ivaksyonova@gmail.com (I. Aksonova)
0000-0003-1647-6444 (S. Yevseiev); 0000-0003-0000-8910 (H. Zavolodko); 0000-0001-7119-0401 (K. Foksha); 0000-0002-7387-5663 (V. Zavolodko); 0000-0003-2605-0455 (I. Aksonova)

[4] clearly explains the essence of APIs as a standard for interaction between systems. An alternative approach to obtaining structured information is to use RSS feeds, described by Whitehead C. T. [5] as another convenient tool for content aggregation.

In practice, data collection tools are implemented using programming languages, in particular Python, which has a developed ecosystem of libraries for working with HTTP requests. Gorobtsov V. [6] demonstrates the use of web scraper for data collection, and the requests library [7-11] acts as a de facto standard for executing HTTP requests in Python.

2. The purpose of this study

The purpose of this study is to develop an architecture for a web service for aggregating educational content with support for an open API and a mobile application that provides convenient access to non-formal education.

This paper presents the development and functionality of a web service that acts as an aggregator of educational content from different platforms. The main tasks are: to implement the server infrastructure for course aggregation; to create a mobile client on Flutter; to provide automatic data updates via web crawling and API; to provide tools for forming a personal educational trajectory; to implement the system comprehensively.

3. The methodological basis

The methodological basis is the use of modern web technologies and principles of data integration: the Python/Django stack is used for the server side; the MariaDB database provides storage of user profiles, courses, and platforms; the web scraper system is developed in Python using the requests [8] and BeautifulSoup libraries; the architecture is modular, which allows adaptation to different platforms through interfaces.

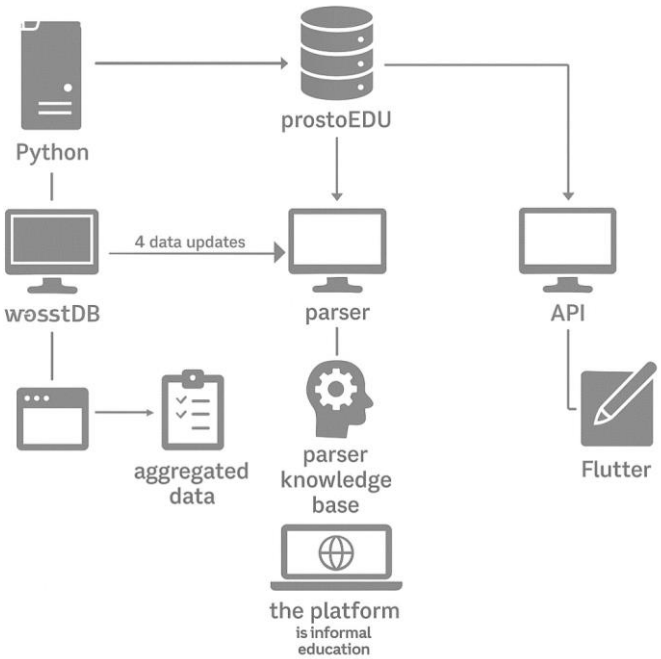

Figure 1: Scheme of the platform operation

The system also has a mobile application built on Flutter that interacts with the backend via an open API, providing cross-platform compatibility and convenient access to learning content.

The system can support both free and paid courses, allowing users to create their own educational trajectory according to their interests and professional goals. The project helps to increase access to quality education and develop digital skills in society.

The diagram in Figure 1 demonstrates the architecture and information flows of a web service that aggregates and provides access to training courses from various non-formal education platforms. The architecture includes both the server side and the mobile client.

The platform's architecture implements a full cycle of user interaction with the educational content aggregator - from the first request to receiving updated results.

The upper part of Figure 2 shows the target audience of the platform: all citizens who want to improve their digital competencies. This includes schoolchildren, students, teachers, entrepreneurs, IT professionals, opinion leaders, civil servants - in fact, anyone interested in modern non-formal education.

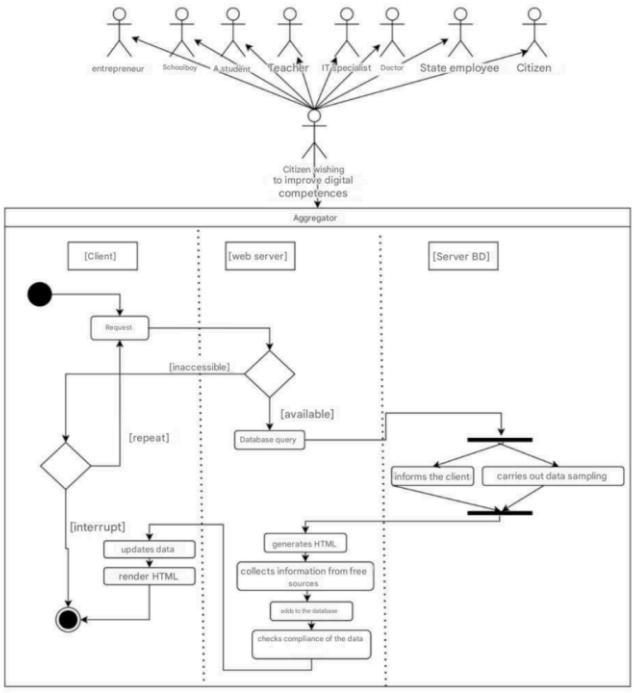

Figure 2: Scheme of the aggregator site operation with roles

The aggregator is at the center of the interaction. Its main function is to receive requests from the user (client), check the availability of data, and return relevant information. The whole process is divided into three conditional blocks: Client, Web server, and Database server.

During the development of the educational content aggregator, a comprehensive analysis of available web scraping tools was conducted to select the most suitable technologies for the task. The key evaluation criteria included the ability to handle dynamic content, session support, processing speed for large HTML/XML datasets, and capabilities for bypassing protection mechanisms or asynchronous data loading. Based on this analysis, a combination of Requests and BeautifulSoup4 was chosen as the primary toolset for parsing static web pages, offering ease of implementation and high code readability. For more complex scenarios involving JavaScript-rendered content, Selenium was employed to simulate full browser interactions. While LXML was considered for high-speed parsing of structured XML documents and Scrapy for scalable and rule-based crawling pipelines, their use was limited due to the specific structure and access policies of educational platforms. This hybrid approach enabled a flexible and effective aggregation mechanism tailored to heterogeneous data sources.

The process starts with a user request. If the database is unavailable, the system offers to retry or, if it fails, generates a page with available or updated data.

If the database is available, the web server makes a request to it. In response, the system either returns the information already stored or updates it: the aggregator collects new data from open educational platforms, checks its relevance, and stores it in the database. Then, an HTML page with relevant content is generated and delivered to the user.

This approach ensures the flexibility and reliability of the system: even in the event of temporary failures, it is able to update the information independently, check it for compliance, and provide the user with the result without the intervention of the administrator.

Thanks to this architecture, the aggregator can effectively serve different user groups and scale to the needs of educational initiatives, EdTech platforms, or government services.

To allow users to access courses from other learning platforms, the platform needs to store courses in the platform database. In order to receive courses from different platforms in prostoEDU, you need to use an information aggregator that will scan data from other platforms and update the relevant information. The general scheme of the aggregator is shown in Figure 3.

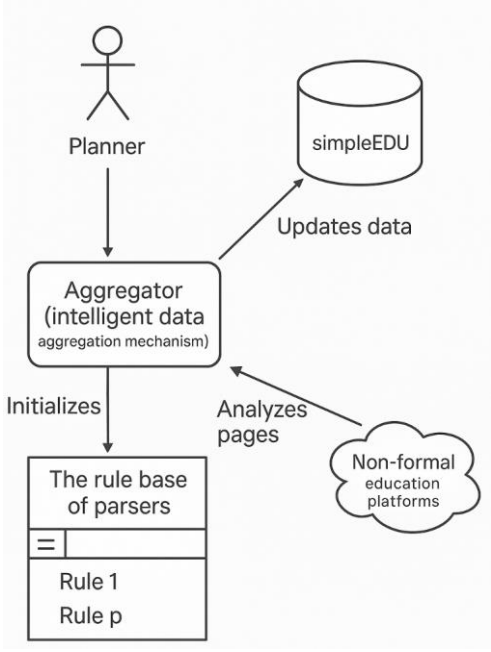

Figure 3: Data aggregator workflow

Among the requirements for the program are the following basic requirements: verification of data collection; definition of characteristics; error handling; speed and efficiency; and availability of process data.

The aggregator uses parser programs, each of which processes an individual website according to a given algorithm. Once the aggregator is launched, it launches programs that collect data. These programs are able to read and analyze the structure of web pages, extract the necessary information, such as headings, text, images, links, pictures, etc. After processing each site, the aggregator collects all the information obtained and enters the data into a database, for example, a list of new courses on the platform. Thus, the aggregator collects and compiles information from the platforms.

Since site structures differ, it becomes necessary to write an individual parser for each platform. However, each parser uses common parts of the code that should not be duplicated but transferred

to separate interfaces. Using these interfaces, it is convenient to change the code for all programs in one place at once, rather than changing the code for each one separately.

To ensure that modern educational platforms can promptly provide up-to-date courses, articles, or news from dozens of other sources, an aggregator works behind the scenes. It all starts with getting the URL of the site from which you want to collect data. The system then generates a list of pages that contain potentially useful information, such as course lists or individual training modules.

In order not to overload the database with duplicates, the aggregator performs an important check: it excludes links from which information has already been saved in the system. This allows you to store only new or updated data, saving resources and reducing the workload.

Next, the system checks whether all pages have been processed. If not, it proceeds to the stage of collecting information from a particular page. This is usually done by so-called parsers, which are mini-programs that extract text, headings, images, or metadata and prepare them for storage.

Once all the pages have been processed, the system proceeds to update the database, i.e. adds new information or updates the existing information. And finally, it shuts down, preparing for the next update cycle.

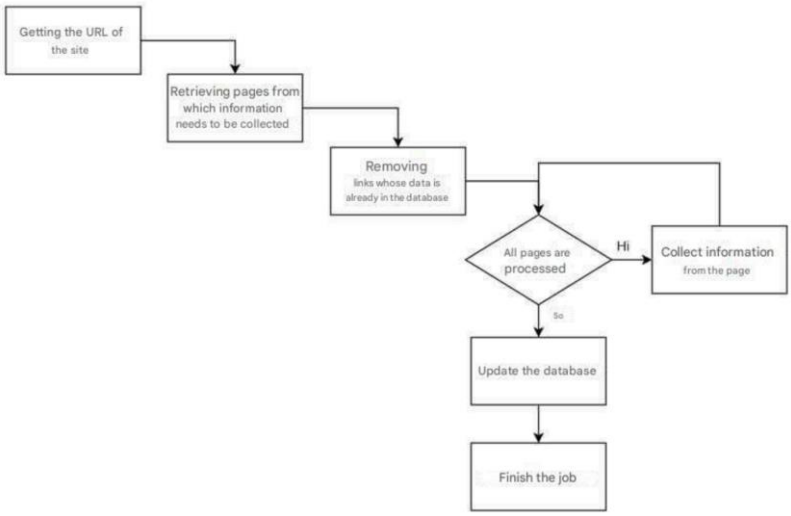

Figure 4: General scheme of the web parser operation

Thanks to this approach, aggregators can keep content up to date, work without human intervention, and most importantly, provide users with convenient and quick access to the most important things.

Data processing algorithms depend on page content, web page structure, data type and format, data volume and complexity, data availability, website limitations, and other factors, but the general steps of a web parser are shown in Figure 4

4. Research Results

The master's projects tested an educational content aggregation system that allows: receiving data from various sources (including paid and free courses); storing aggregated information in a unified structure; viewing content through a web interface or mobile application; supporting automatic data updates (through a task scheduler); scaling the solution for use by educational institutions or communities.

As part of the testing of the developed educational content aggregator system, three stages of testing were carried out with different amounts of input data. This allowed us to evaluate the stability of the system, the speed of information processing, and the accuracy of the results for four popular educational platforms: Coursera, Alison, Sololearn, and Edx. A table comparing the results of the program is shown in Table 1.

At the first stage of testing with a small amount of data, the system demonstrated high accuracy and fast response. For the Coursera and Sololearn platforms, the average processing time for one piece of information did not exceed 1 second. All data was collected without errors.

The second stage involved increasing the number of courses, which made it possible to identify the dependence of performance on the amount of input data. The Coursera and Sololearn platforms maintained low processing time per unit (less than 1 second), but in the case of Coursera, there were isolated errors related to network timeouts. At the same time, the Alison and Edx platforms showed significantly slower processing - over 3 and 7 seconds per unit, respectively.

At the third stage of testing, the aggregator processed the full data set. Despite the heavy load, the system coped with the task: Coursera processed more than 8,700 courses with an average time per unit of about half a second, while the Alison and Edx platforms took longer and recorded more errors. The least loaded platform was Sololearn, which consistently processed with the lowest time and without any errors.

Table 1
Comparison of the results

N	Characteristics	Platform. Coursera	Alison	Sololearn	Edx
1	Time (s)	17	35	8	204
	Amount of data (units)	60	10	5	32
	Time per unit (s)	0,28	3,5	1,6	6,37
	Errors (units)	0	0	0	0
2	Time (s)	422	115	14	506
	Amount of data (units)	1764	30	15	64
	Time per unit (s)	0,24	3,83	0,9	7,9
	Errors (units)	2	0	0	0
3	Time (s)	4655	13647	12	3056
	Amount of data (units)	8766	3791	27	456
	Time per unit (s)	0,53	3.59	0.44	6.7
	Errors (units)	15	10	0	1

A separate test of adding 100 courses to the database showed stable backend operation: all data was saved without errors, duplication, or loss. This indicates the effective integration of the modules for collecting, processing, and storing information.

The study has confirmed that the system has a high level of scalability, flexibility to the structure of different platforms, and the ability to keep data up-to-date in real time. Despite differences in processing speed for different sources, the developed aggregator demonstrates stable and reliable operation at all stages of testing.

The aggregator collects data by launching individual parsers for each platform. Thanks to the use of unified interfaces, the processing logic can be changed centrally. This increases reliability and simplifies system maintenance.

We also describe typical processing algorithms that depend on the structure of sites and data features (Fig. 4).

Acknowledgements

The proposed system of educational content aggregator has been successfully tested as an integrated solution that includes a server part, a data collection module and a mobile application. This approach: increases the accessibility of quality education; promotes the digital transformation of the educational process; provides integration with other EdTech solutions; has the potential to be scaled and used in formal and non-formal educational environments.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] Matyash D. Why do we need aggregator sites, why does Google love them so much? [Electronic resource] – Access mode: https://jam.in.ua/blog/navishcho-potribni-sajty-ahrehatory-chomu-google-ikh-tak-liubyt/#:~:text=Сайт-агрегатор - Title from the screen.

[2] What is crawling and how to manage robots [Electronic resource]. – Access mode: https://www.bizmaster.xyz/2019/04/scho-take-krauling-i-yak-keruvaty-robotamy.html - Title from the screen.

[3] Henderson A. 15 Best FREE Website Crawler Tools & Software (2023 Update) [Electronic resource]. – Access mode: https://www.guru99.com/web-crawling-tools.html - Title from the screen.

[4] Digital Commerce Intelligence [Electronic resource]. – Access mode: https://www.dexi.io - Title from the screen.

[5] Gorobtsov V. How to use web scraper to collect data from the Internet with Python [Electronic resource]. – Access mode: https://dou.ua/forums/topic/43070/ - Title from the screen.

[6] What is an API? - IBM. [Electronic resource]. - Access mode: https://www.ibm.com/topics/api - Title from the screen.

[7] Requests: HTTP for Humans™ [Electronic resource]. – Access mode: https://docs.python-requests.org/en/latest/index.html - Title from the screen.

[8] Zamkovyi M., et al. Algorithmic support for building a distributed iot system in a cloud service. In: 2023 IEEE 4th KhPI Week on Advanced Technology (KhPIWeek). IEEE, 2023. p. 1-6.

[9] Pihnastyi O., Usik V., Kozhevnikov G., Matiash O. Neural network-based approach for predicting the flow material in transport systems. CEUR Workshop Proceedings, 2024, 3790, pp. 76-86.

[10] Popovych B., Zavolodko G. Analysis of Methods for Classification and Aggregation of Textual Data From Images. Security of Infocommunication Systems and Internet of Things, 2024, 2.1: 01008-01008. pp. 1-5.

[11] Korolekh Y, Zavolodko G. Enhancing digital search: Synergizing the Levenshtein algorithm with NLP techniques, in IX International Scientific and Practical Conference "Scientific Problems and Options for Their Solution," Bucharest, Romania, Feb. 7-9, 2024, International Scientific Unity, pp. 60-64.