МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«До захисту допущено»

В.о. завідувача кафедри

Ігор ШЕЛЕХОВ

(підпис)

червня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 - Комп'ютерних наук,

освітньо-професійної програми «Інформатика»

на тему: **«Інформаційне та програмне забезпечення сервісу збору та аналізу даних наукової діяльності співробітників закладу вищої освіти з використанням інструментарію парсингу»**

здобувача групи ІНЗ -91с Жукова Олександра Андрійовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Олександр Жуков

(підпис)

Керівник, доцент,

кандидат фізико-математичних наук Сергій ШАПОВАЛОВ

(підпис)

Суми – 2023Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«Затверджую»

В.о. завідувача кафедри

Ігор ШЕЛЕХОВ

(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»

здобувача групи ІНЗ-91с Жукова Олександра Андрійовича

1. Тема роботи: «Інформаційне та програмне забезпечення сервісу збору та аналізу даних наукової діяльності співробітників закладу вищої освіти з використанням інструментарію парсингу»

затверджую наказом по СумДУ від *«01» червня 2023 р. № 0475-VI*

2. Термін здачі здобувачем кваліфікаційної роботи *до 09 червня 2023 року*

3. Вхідні дані до кваліфікаційної роботи _____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Огляд технологій, що використовуються для вирішення завдань дослідження. 3) Розробка інформаційного та програмного забезпечення. 4) Тестування та аналіз результатів.*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «27» Жовтня 2022р.

Завдання прийняв до виконання  Керівник 

(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*	20.10.2023-27.10.2022	
2	*Огляд технологій, що використовуються для вирішення завдань дослідження*	27.10.2022-15.12.2022	
3	*Розробка інформаційного та програмного забезпечення*	15.12.2022-14.02.2023	
4	*Аналіз отриманих результатів*	14.02.2023-17.02.2023	
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*	17.02.2023-20.02.2023	

Здобувач вищої освіти  Керівник 

(підпис) (підпис)

АНОТАЦІЯ

Записка:
55стор.,16рис.,1додаток.,7використаних джерел.

У діяльності вузів наукова діяльність є одним із головних напрямків роботи. Аналіз наукової діяльності є важливою та складною задачею, яка потребує багато часу та зусиль.Відповідно виникає потреба в інструментах, які допоможуть автоматизувати процес збору та аналізу даних наукової діяльності співробітників вузу. Одним з інструментів є програмне забезпечення збору та аналізу даних наукової діяльності співробітників вузу з використанням інструментарію парсінга

Об’єктом дослідження є процес збору та аналізу даних

Метою кваліфікацією роботи є розробка інформаційного й програмного забезпечення для збору та аналізу даних наукової діяльності співробітників вузу з використанням інструментарію парсінгу.
Програма розроблялася з метою збору та аналізу даних про наукову діяльність співробітників вузу.

Для створення програми парсера необхідно реалізувати алгоритм, що дозволяє мати можливість при виборі дії (операції) виводити її на екран і отримувати результат обчислень. Для підвищення зручності користування програмою розроблено простий графічний інтерфейс, тобто мінімальна кількість операцій, які користувач може виконувати у програмі, виведені безпосередньо на екран користувача.

У кваліфікаційній роботі розроблено програмне забезпечення для збору та аналізу даних наукової діяльності співробітників вузу з використанням

інструментарію парсингу. Програма має збирати інформацію про наукові події з сайту https://ceur-ws.org/, аналізувати їх та відображати результати в графічному інтерфейсі.. Основною функцією програми має бути визначення присутності наукових працівників вузу серед авторів наукових статей, які були опубліковані в рамках наукових подій . Розробка програмного забезпечення такого роду може бути досить корисним інструментом для управління науковою діяльністю співробітників вузу, що дозволить збільшити ефективність та якість наукових досліджень

ЗМІСТ

ВСТУП

В сучасному світі наукова діяльність є важливою складовою розвитку суспільства. Для вузів наукова діяльність є одним із головних напрямків роботи, адже вона не тільки забезпечує підвищення престижу навчального закладу, а й є основою для вивчення нових знань та технологій, що дозволяє відповідати вимогам сучасного ринку праці. Проте, аналіз наукової діяльності є важливою та складною задачею, яка потребує багато часу та зусиль. У зв'язку з цим, виникає потреба в інструментах, які допоможуть автоматизувати процес збору та аналізу даних наукової діяльності співробітників вузу.

Одним з таких інструментів є програмне забезпечення збору та аналізу даних наукової діяльності співробітників вузу з використанням інструментарію парсінга. Це програмне забезпечення дозволяє автоматизувати процес збору та аналізу даних наукової діяльності співробітників вузу з сайту організації https://ceur-ws.org/ та проводити їх аналіз в контексті участі викладачів університету. У наступних пунктах будуть розглянуті етапи проектування, та загальна інформація щодо даного програмного забезпечення.

Реферат

Даний дипломний проект присвячений розробці програмного забезпечення для збору та аналізу даних наукової діяльності співробітників вузу з використанням інструментарію парсінгу. Метою проекту є створення програмного забезпечення, яке допоможе співробітникам вузу знаходити наукові події, участь в яких беруть працівники цього вузу, та оцінювати внесок кожного співробітника у наукові дослідження.

Програмне забезпечення працює з сайтом організації https://ceur-ws.org/ та здійснює пошук наукових подій, у яких беруть участь співробітники вузу. Для кожної наукової статті, яка бере участь у науковій події, програма шукає потрібних авторів серед співробітників вузу та аналізує їх внесок у дослідження.

Для оцінки внеску кожного співробітника у наукові дослідження була розроблена формула, яка базується на кількості авторів статті та кількості посилань на неї. Крім того, програмне забезпечення має графічний інтерфейс, який дозволяє настроювати список іскомих працівників та вагу авторства та посилань.

Програмне забезпечення працює в окремому потоці, що дозволяє йому не блокувати графічний інтерфейс. Після завершення роботи програма зберігає результати у файлі .csv, який можна відкрити у будь-якому редакторі таблиць.

Отже, дане програмне забезпечення допоможе співробітникам вузу знаходити та аналізувати наукові події, участь в яких беруть працівники вузу. Програма буде автоматично сканувати бази даних наукових конференцій, семінарів та інших подій, які відбуваються в галузях, пов'язаних з освітою та наукою. Далі, вона буде збирати дані про дати, місце та теми цих подій.

Крім того, програма буде мати можливість аналізувати наукові статті, які були опубліковані в наукових журналах. Вона буде збирати дані про авторів статей, їхні інституції та тематику досліджень. Також, програма буде

мати можливість автоматично визначати цитованість та імпакт-фактор наукових статей, що дозволить працівникам вузу відслідковувати тенденції та розвиток певної галузі.

За допомогою цих функцій, працівники вузу зможуть знаходити потенційні наукові події, які можуть бути корисними для їхньої роботи та досліджень. Вони також зможуть бути в курсі останніх наукових досягнень в своїй галузі та взаємодіяти зі співробітниками з інших університетів, що працюють у схожих напрямках досліджень.

Пояснювальна записка складається із вступу, шести розділів, висновку, списку використаних джерел; містить 40 сторінок, 18 рисунків та 1 додаток. Список використаних джерел включає 4 бібліографічних найменувань та 3 електронних ресурсів.

Ключові слова: інформаційне забезпечення; програмне забезпечення; збір даних; аналіз даних; наукова діяльність; співробітники вузу; інструментарій парсінга; статистичні методи; Python

Abstract

This thesis project is devoted to the development of software for collecting and analyzing data on the scientific activities of university employees using parsing tools. The aim of the project is to create software that will help university employees find scientific events in which the employees of this university participate and evaluate the contribution of each employee to scientific research.

The software works with the organization's website https://ceur-ws.org/ and searches for scientific events in which university staff members participate. For each scientific article that participates in a scientific event, the program searches for the necessary authors among the university staff and analyzes their contribution to the research.

A formula based on the number of authors of the article and the number of references to it has been developed to assess the contribution of each employee to scientific research. In addition, the software has a graphical interface that allows you to customize the list of employees you are looking for and the weight of authorship and references.

The software runs in a separate thread, which allows it not to block the graphical interface. When the program is finished, it saves the results in a .csv file that can be opened in any spreadsheet editor.

Thus, this software will help university staff to find and analyze scientific events in which university employees participate. The program will automatically scan databases of scientific conferences, seminars and other events that take place in the fields related to education and science. Further, it will collect data on the dates, locations, and topics of these events.

In addition, the program will be able to analyze scientific articles that have been published in scientific journals. It will collect data about the authors of the articles, their institutions, and the topics of research. Also, the program will be able to automatically determine the citation and impact factor of scientific articles, which will allow university staff to track trends and developments in a particular field.

With the help of these features, university staff will be able to find potential scientific events that may be useful for their work and research. They will also be able to keep abreast of the latest scientific advances in their field and interact with staff from other universities working in similar research areas.

The explanatory note consists of an introduction, six chapters, a conclusion, and a list of references; it contains 40 pages, 18 figures, and 1 append. The list of references includes 4 bibliographic titles and 3 electronic resources.

Keywords: information software; software; data collection; data analysis; scientific activity; university staff; parsing tools; statistical methods; Python

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

1. Парсінг - автоматичне збирання та обробка інформації з веб-сторінок.
2. Наукові події - конференції, семінари, дискусії тощо, що проходять в науковому середовищі.
3. Статті - наукові матеріали, що публікуються в спеціалізованих журналах.
4. Автори - науковці, що беруть участь у написанні статті або приймають участь у науковій події.
5. Графічний інтерфейс - частина програми, яка дозволяє користувачеві взаємодіяти з програмою за допомогою графічних елементів.
6. Вікно довідки - окреме вікно, яке містить інформацію про програму та її функції.
7. Налаштування - параметри програми, які можна змінювати, щоб налаштувати її на власний розсуд.
8. CSV - Comma Separated Values, формат збереження даних у вигляді таблиці, де значення окремих комірок розділяються комами.
9. Потік - окремий процес, що працює паралельно з іншими процесами програми.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд предметної області

Збір та аналіз даних наукової діяльності є важливим етапом у розвитку науки та вищої освіти. Щоб забезпечити ефективний збір та аналіз даних, використовуються різноманітні інструменти та програмне забезпечення, які дозволяють автоматизувати процеси збору та аналізу даних.

Одним із таких інструментів є парсери, які дозволяють витягувати дані з різних джерел та перетворювати їх у структуровані формати. У контексті наукової діяльності співробітників вузів, аналізатори можуть використовуватися для збору даних про публікації, наукові заходи, наукові дослідження та інші аспекти наукової діяльності.

Розробка програмного забезпечення для збору та аналізу даних наукової діяльності співробітників вузу з використанням інструментарію розбіру є актуальною задачею для вузів та дослідницьких організацій. Таке програмне забезпечення може допомогти у підвищенні ефективності збору та аналізу даних, зниженні часових та фінансових витрат на ці процеси та поліпшенні якості отриманих даних.

Однак, розробка програмного забезпечення для збору та аналізу даних наукової діяльності співробітників вузу з використанням інструментарію аналізу також має свої виклики та обмеження, зокрема, пов'язані з технічними та етичними аспектами використання парсерів та збору персональних даних.

1.2 Поняття парсінгу та його використання в програмуванні

Загалом, парсінг - це лінійне зіставлення послідовності слів із правилами мови. Текст інтернет-сторінок є ієрархічний набір даних, структурований за допомогою людських і комп'ютерних мов. На людській мові надано інформацію, знання, заради яких, власне, люди користуються Інтернетом. Комп'ютерні мови (html, JavaScript, CSS) визначають, як інформація виглядає на моніторі.

Парсінг сайтів є ефективним рішенням для автоматизації збору та зміни інформації.

Поняття «мова» розглядається у найширшому контексті. Це може бути людська мова, яка використовується для комунікації людей. А може й формалізована мова, зокрема будь-яка мова програмування.

Парсінг (Parsing) – це прийняте в інформатиці визначення синтаксичного аналізу. Для цього створюється математична модель порівняння лексем із формальною граматикою, описана однією з мов програмування. Наприклад, PHP, Perl, Ruby, Python.

Коли людина читає, то, з погляду науки філології, він здійснює синтаксичний аналіз, порівнюючи побачені на папері слова (лексеми) з тими, що є у його словниковому запасі (формальною граматикою).

Незалежно від того якою формальною мовою програмування написано парсер, алгоритм його дії залишається однаковим:

1) вихід в інтернет, отримання доступу до коду веб-ресурсу та його скачування;

2) читання, вилучення та обробка даних;

3) представлення вилучених даних у зручному вигляді - файли.txt,.sql,.xml,.html та інших форматах.

В інтернеті часто зустрічаються вирази, з яких випливає, ніби парсер (пошуковий робот, бот) мандрує Всесвітньою мережею. Але найчастіше ця програма ніколи не залишає комп'ютера, на якому вона інстальована.

Збір інформації в інтернеті - трудомістка, рутинна робота, що забирає багато часу. Парсери, здатні протягом доби перебрати більшість веб-ресурсів у пошуках необхідної інформації, автоматизують її.

Найактивніше «парсять» всесвітню мережу роботи пошукових систем. А так інформація збирається через лексичний аналіз у приватних інтересах. На її основі, наприклад, можна написати дисертацію.

Парсінг використовують програми автоматичної перевірки унікальності текстової інформації, швидко порівнюючи вміст сотень веб-сторінок із

запропонованим текстом. Без програм парсінгу власникам інтернет-магазинів, яким потрібні сотні однотипних описів товарів, технічних характеристик та іншого контенту, що не є інтелектуальною власністю, важко було б вручну заповнювати характеристики товарів.

Можливістю «списати» чужий контент для наповнення свого сайту користуються багато веб-майстрів та адміністраторів сайтів. Це виправдано, якщо потрібно часто змінювати контент для представлення поточних новин або іншої інформації, що швидко змінюється.

Парсінг дозволяє працювати з даними будь-якої тематики. Серед основних сфер застосування такої технології можна виділити:

- пошук та наповнення ресурсів текстовим та мультимедійним контентом;
- товари та ціни в інтернет-магазинах;
- Дані з оголошень, розміщених на спеціальних ресурсах;
- пошук та збирання контактних даних користувачів;
- у рамках соціальних мереж (наприклад, відгуки та коментарі);
- сайти, що спеціалізуються на публікації спортивних результатів.

Звичайно ж, парсери не читають тексту, вони лише порівнюють запропонований набір слів з тим, що виявили в інтернеті і діють за заданою програмою. Те, як пошуковий робот повинен вчинити зі знайденим контентом, написано в командному рядку, що містить набір букв, слів, виразів та знаків програмного синтаксису. Такий командний рядок називається "регулярний вираз". Російські програмісти використовують жаргонні слова "маска" та "шаблон".

Щоб парсер розумів регулярні висловлювання, він має бути написаний мовою, яка підтримує їх у роботі з рядками. Така можливість є у PHP, Perl. Регулярні вирази описуються синтаксисом Unix, який хоч і вважається застарілим, але широко застосовується завдяки властивості зворотної сумісності.

Синтаксис Unix дозволяє регулювати активність парсінгу, роблячи його «лінивим», «жадібним» і навіть «наджадібним». Від цього параметра залежить довжина рядка, який парсер копіює з веб-ресурсу. Надто жадібний аналіз отримує весь контент сторінки, її HTML-код і зовнішню таблицю CSS.

Python - це одна з найбільш популярних мов програмування для розробки парсерів. Це пояснюється тим, що Python є мовою з простим і зрозумілим синтаксисом, а також має велику кількість різноманітних бібліотек та інструментів для парсінгу даних.

У Python є багато бібліотек для парсінгу веб-сторінок, таких як BeautifulSoup, lxml, Scrapy та інші. З їх допомогою можна отримувати інформацію з різних сайтів та обробляти її залежно від потреб користувача.

Окрім розбору веб-сторінок, Python може бути використаний для парсінгу інших типів даних, таких як текстові файли, PDF-документи, таблиці Excel тощо. Для цього можуть використовуватися різноманітні бібліотеки, такі як PyPDF2, xlrd, pandas тощо.

Крім того, Python має велику спільноту розробників, що забезпечує наявність великої кількості різноманітних прикладів коду та допомогу при вирішенні проблем.

Питання, чи є лексичний аналіз крадіжкою контенту, активно обговорюється у Всесвітній мережі. Більшість опонентів вважають, що запозичення частини контенту, що не є інтелектуальною власністю, наприклад технічних описів, припустимо. Посилання на джерело контенту розглядається як спосіб часткової легітимації. Водночас нахабне копіювання, включаючи граматичні помилки, засуджується інтернет-спільнотою, а пошуковими системами розглядається як привід для блокування ресурсу.

Парсінг сайтів – послідовний синтаксичний аналіз інформації, розміщеної на веб-сторінках. Парсінг сайтів є найбільш ефективним рішенням для автоматизації збору та зміни інформації. Для написання парсерів підходять будь-які мови програмування, якими створюються програми для роботи зі Всесвітнім Павутинням. Веб-програми для парсінгу зазвичай пишуть

на C++, Delphi, Perl, Ruby, Python, PHP. [4] У порівнянні з людиною, комп'ютерна програма-парсер:

• Здатна швидко проаналізувати понад тисячу веб-сторінок (за кілька секунд);

• Безпомилково відібрати потрібну інформацію;

• Надає кінцеві дані в необхідному вигляді (база даних або електронна таблиця).

У порівнянні з пошуковою системою, комп'ютерна програма-парсер:

• Має більш високу швидкість підбору необхідної інформації;

• Відрізняється підвищеною точністю: містить конкретні дані за заданими критеріями пошуку;

Може бути універсально налаштована під вимоги та запити будь-якого користувача. У цьому роботі було поставлено завдання створення парсера мовою Python.

1.3 Огляд аналогів

Огляд аналогів на тему розробки програмного забезпечення для збору та аналізу даних наукової діяльності співробітників вузу з використанням інструментарію парсінгу показав наявність декількох подібних програмних засобів на ринку. Однак, не всі вони мають такі ж переваги, як моя програма.

Один з аналогів - Scopus, який забезпечує збір та аналіз наукових даних. Однак, ця програма є платною, тому її використання може бути обмеженим для окремих користувачів. Крім того, Scopus не має можливості розбору, що обмежує можливості користувачів щодо створення власних запитів та вибірки даних.

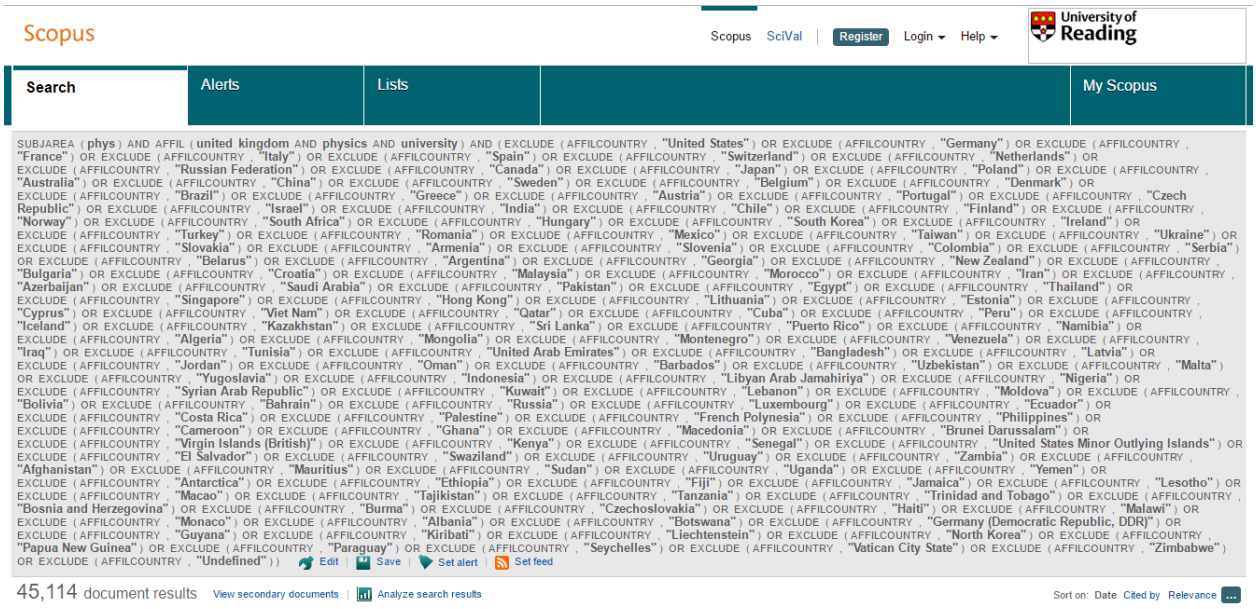

Рисунок 1.1 - Scopus

Інший аналог - Web of Science, також є платним програмним засобом зі збору та аналізу наукових даних. Він має більшу кількість наукових джерел, ніж Scopus, але також не надає можливість парсінгу даних, що обмежує користувачів у виборці необхідних даних та створенні власних аналітичних звітів.

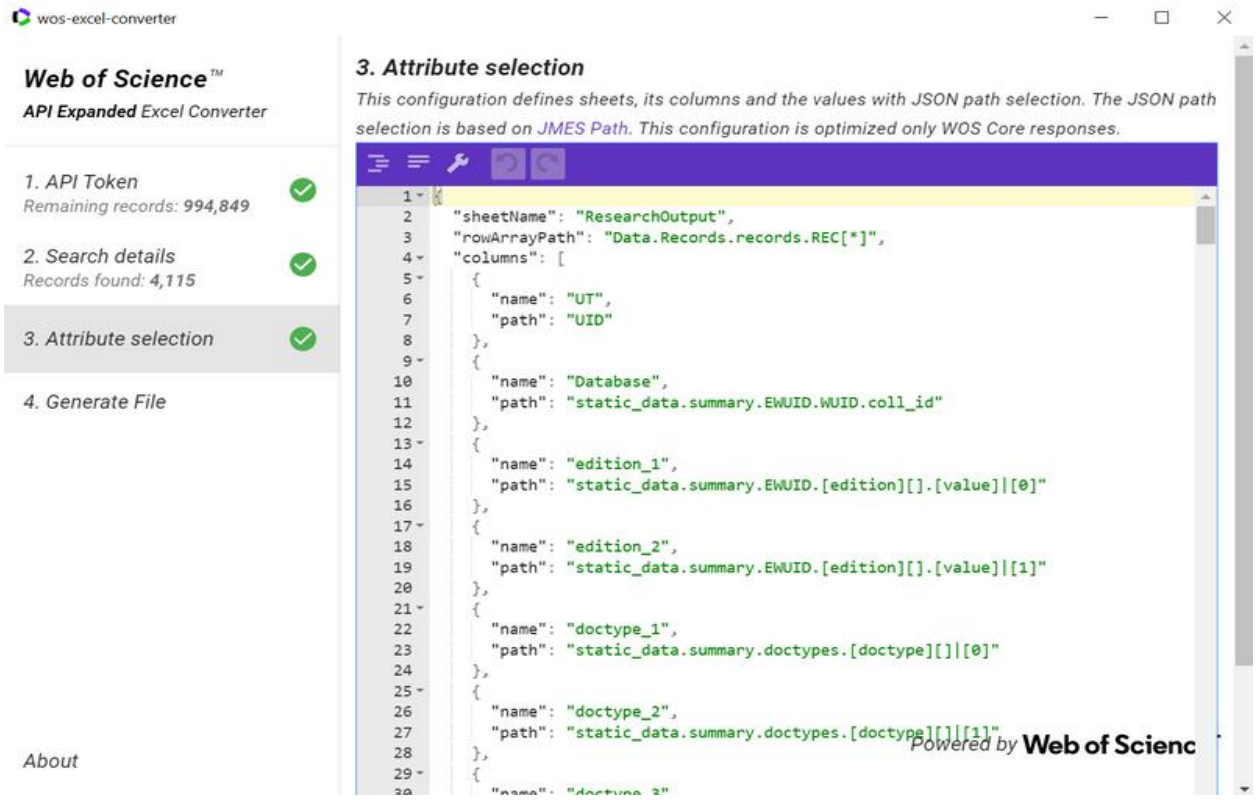

Рисунок 1.2 - Web of Science

Моя програма має кілька переваг в порівнянні з цими аналогами. Перш за все, вона пропонує безкоштовне використання та можливість самостійно вибирати та парсіти необхідні дані. Крім того, вона працює з великою кількістю наукових джерел, що дозволяє користувачам отримати більш повне уявлення про наукову діяльність співробітників вузу. Моя програма також забезпечує зручний інтерфейс та можливість створення власних звітів та аналізів на основі зібраних даних. Усі ці переваги роблять мою Моя програма також забезпечує зручний інтерфейс та можливість створення власних звітів та аналізів на основі зібраних даних. Усі ці переваги роблять мою програму більш зручною та корисною у порівнянні з існуючими аналогами.

1.4 Постановка задачі

Основною метою дипломної роботи є розробка інформаційного та програмного забезпечення збору та аналізу даних наукової діяльності співробітників вузу з використанням інструментарію парсінгу. Для досягнення цієї мети необхідно вирішити такі задачі:

1. Розробити програмний продукт, що здійснює розбір наукових статей та подій з сайту https://ceur-ws.org/.
2. Розробити механізм пошуку та вибору необхідних викладачів вузу серед авторів статей та відповідних посилань.
3. Розробити алгоритм оцінки наукових подій за участю викладачів вузу на основі вагової функції.
4. Розробити графічний інтерфейс програмного продукту, що дозволяє користувачам змінювати налаштування та відслідковувати результати парсінгу у реальному часі.
5. Розробити механізм збереження результатів роботи програмного продукту у форматі .csv для подальшого використання.
6. Здійснити тестування програмного продукту та описати результати тестування.

7. Написати документацію до програмного продукту, включаючи опис функцій та інструкцію з використання.

Для створення програмного продукту використовуватимуться сучасні технології та інструменти, такі як мова програмування Python. Програмний продукт буде містити модулі для збору даних про наукову діяльність співробітників вузу, аналізу цих даних та візуалізації результатів аналізу.

У результаті розробки програмного продукту буде забезпечена можливість збору та обробки даних про наукову діяльність співробітників вузу в автоматичному режимі, що значно зменшить час та зусилля, необхідні для збору та аналізу цих даних вручну. Крім того, програмний продукт дозволить отримувати звіти про стан наукових проектів та науково-дослідної роботи в цілому, що дозволить ефективніше управляти цими проектами та підвищити їх результативність.

Отже, результатом дипломної роботи буде програмний продукт, який дозволить збирати та аналізувати дані наукової діяльності співробітників вузу, що покращить управління науковими проектами та підвищить рівень науково-дослідної роботи.

2 ВИБІР МЕТОДІВ РІШЕННЯ ЗАДАЧІ

2.1 Python як вибір мови програмування

Загальне призначення програмного засобу - виконання операцій, що конвертуються для використання в навчальному процесі та повсякденному житті.

Реалізоване завдання полягає в тому, щоб при виборі дії виконувалася певна операція.

Python - це високорівнева мова програмування, яка використовується в різних сферах IT, таких як машинне навчання, розробка програм, web, парсінг та інші.

У 2019 році Python став найпопулярнішою мовою програмування, обігнавши Java на 10%. Це зумовлено багатьма причинами, одна з яких – висока оплата праці кваліфікованих фахівців (близько 100 тисяч доларів на рік).

Програма на Python, як і в інших мовах програмування реалізує алгоритм вирішення завдання. Вона об'єднує послідовність дій, що виконуються над певними типами даних за допомогою операцій, що визначаються можливостями мови. Мова Python є універсальним мовою, тобто. на ньому можна писати обчислювальні, графічні та системні програми.

Python є однією з найбільш популярних мов програмування у світі з відкритим вихідним кодом. Вона має декілька переваг, які роблять її привабливою для програмістів різних рівнів досвіду і для різних проектів:

1. Простота: Python має простий і зрозумілий синтаксис, що дозволяє швидко створювати програми без зайвих зусиль.
2. Розширюваність: Python має велику кількість бібліотек та модулів, які дозволяють розширювати функціональність програми, зменшуючи час розробки та підвищуючи ефективність.
3. Кросплатформенність: Python працює на різних операційних системах, включаючи Windows, macOS і Linux.

4. Підтримка спільноти: Python має велику та дружню спільноту розробників, що забезпечує швидке розроблення нових інструментів та рішень, а також допомогу у вирішенні проблем.

5. Великий вибір задач: Python може бути використаний для багатьох різних задач, від веб-розробки та наукових обчислень до розробки ігор та штучного інтелекту.

6. Широке застосування: Python є популярною мовою програмування у багатьох галузях, включаючи веб-розробку, наукові дослідження, обробку даних, машинне навчання, аналіз тексту та багато іншого.

Таким чином, Python є хорошим вибором мови програмування для різних проектів та рівнів досвіду, завдяки своїй простоті, розширюваності, кросплатформенність, підтримці спільноти, великому вибору задач та широкому застосуванню.

2.2 Огляд бібліотек котрі будуть використовуватись

В програмі будуть використовуватись дуже багато бібліотек, так як є поставлене завдання вже можна накидати список бібліотек котрі будуть використовуватись в програмному застосунку , а саме csv, dataclass, time, tkinter, typing, pypdf2, requests, bs4, datetime, re, io, threading, urlib та інші

1. csv: ця бібліотека дозволяє роботу з CSV-файлами, тобто з файлами, що містять табличні дані, розділені комами (або іншим символом). З її допомогою можна зчитувати та записувати дані у форматі CSV, а також виконувати різноманітні маніпуляції з цими даними.

2. datetime: ця бібліотека надає засоби для роботи з датами та часом у Python. З її допомогою можна створювати, зчитувати та форматувати дати та час, а також виконувати різноманітні операції з ними.

3. io: ця бібліотека надає інтерфейс для роботи з різними типами введення-виведення даних, такими як файли, рядки, байти тощо. З її

допомогою можна створювати об'єкти, що представляють різноманітні потоки даних, та виконувати з ними різноманітні операції.

4. re: ця бібліотека надає засоби для роботи з регулярними виразами (або regex) в Python. З її допомогою можна здійснювати пошук та заміну тексту, використовуючи складні шаблони з використанням різних символів та конструкцій.

5. threading: ця бібліотека дозволяє створювати та управляти потоками виконання у Python. З її допомогою можна виконувати декілька операцій паралельно та контролювати взаємодію між ними.

6. urllib: це модуль вбудованої бібліотеки Python, який надає функції для роботи з URL-адресами, зокрема для створення запитів і отримання відповідей з веб-сторінок. Цей модуль зазвичай використовується в парсінгу веб-сторінок, де необхідно отримати дані з веб-сайту або взаємодіяти з API.

7. PyPDF2: це бібліотека Python для роботи з PDF-файлами. З її допомогою можна створювати, читати та редагувати PDF-документи. Наприклад, вона може бути використана для витягування даних з PDF-файлів, що є важливим для парсінгу і аналізу даних.

8. requests: це бібліотека Python, яка надає можливість здійснювати HTTP-запити до веб-сторінок та інших ресурсів. Вона дозволяє отримувати відповіді з сервера, відправляти дані на сервер, керувати параметрами запиту та інші операції, пов'язані з HTTP-запитами.

9. bs4: це бібліотека Python, яка надає інтерфейс для парсінгу HTML- та XML-документів. Вона дозволяє витягувати дані з HTML-файлів, що є важливим для отримання даних з веб-сторінок для подальшого аналізу та обробки.

10. Tkiner – це графічна бібліотека, що дозволяє створювати програми з віконним інтерфейсом. Ця бібліотека є інтерфейсом до популярної мови програмування та інструменту створення графічних програм tcl/tk.

Для нанесення написів на кнопки, які використовуються в інтерфейсі програми, ми використовуємо text='Якусь назву для кнопки'.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ

3.1 План розробки програмного забезпечення

Для створення програми парсера необхідно реалізувати алгоритм, що дозволяє мати можливість при виборі дії (операції) виводити її на екран і отримувати результат обчислень. Також необхідно організувати можливість скидання одержаних результатів.

Для підвищення зручності користування програмою розроблено простий графічний інтерфейс, тобто мінімальна кількість операцій, які користувач може виконувати у програмі, виведені безпосередньо на екран користувача. Для початку розробки програмного застосування потрібно побудувати майбутню схему з усіх функцій та класів. Ця програма є графічним інтерфейсом для парсінгу та аналізу наукових подій та статей. Основними функціями програми є:

1. Клас Parser, який містить вкладений клас Event для представлення наукових подій. Клас Event містить метод calculate_score() для обчислення фінального балу події та методи gt() та lt() для сортування подій.

2. Функція analyze_article() для аналізу PDF-файлу статті за посиланням.

3. Функція analyze_event() для аналізу наукової події, що містить виклик функції analyze_article() для аналізу кожної статті, пов'язаної з подією.

4. Функція get_latest_event() для оновлення даних щодо останньої події на сайті.

5. Функція settings() для налаштування програми, яка містить вкладену функцію save() для збереження введених даних.

6. Функція edit_professors() для внесення змін до списку викладачів, яка містить вкладену функцію save() для збереження змін.

7. Функція about() для відображення текстової інформації про програму.

8. Функція parse() для парсінгу веб-сторінки, яка містить список наукових подій.

9. Функція start() для старту процесу парсінгу в окремому потоці.

10.Функція update_table() для оновлення таблиці подій.

11.Функція update_counter() для оновлення інформаційного ярлику.

12.Функція write_to_file() для запису результатів в файл.

Ця програма використовує графічний інтерфейс бібліотеки tkinter, а також многопотоковість з бібліотеки threading для паралельного виконання процесу парсінгу. Також використовуючи функції складу діаграму класу рис. 3.1

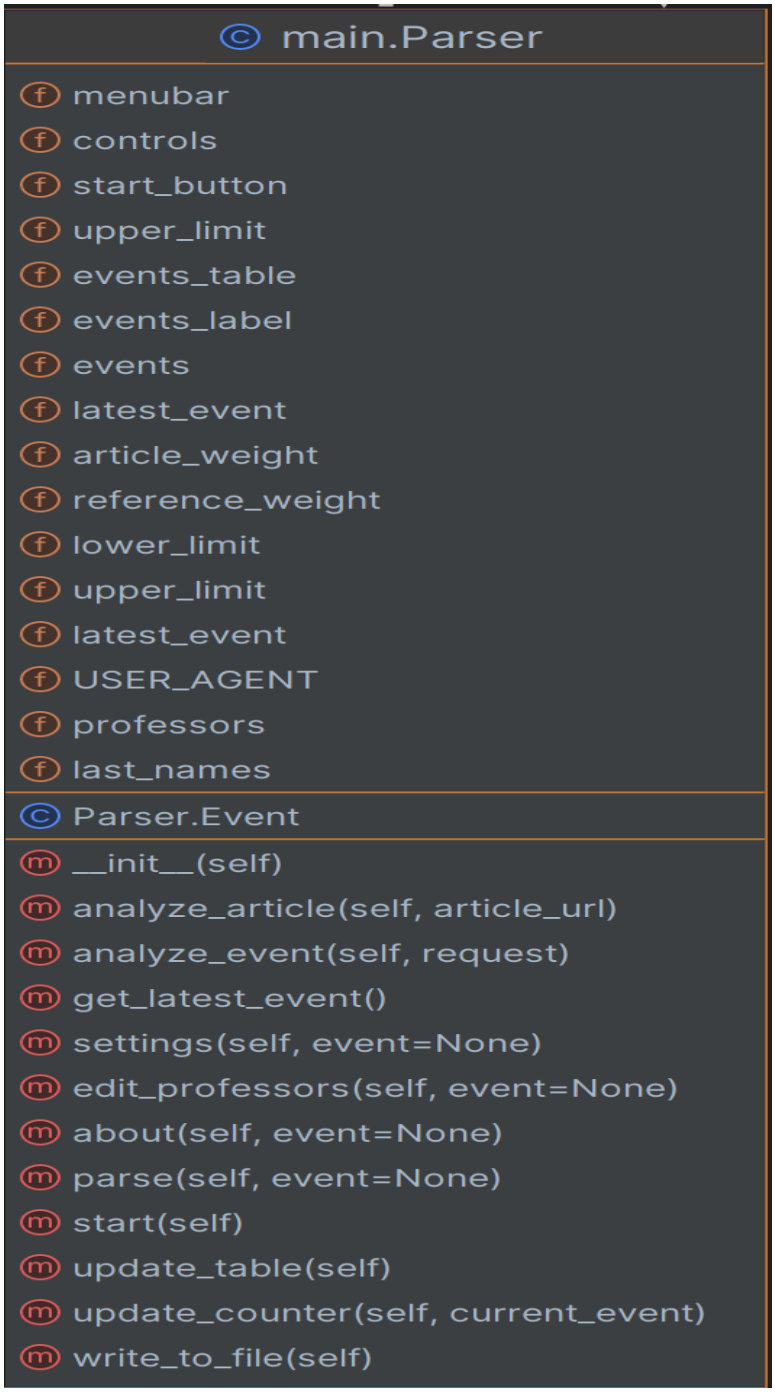

Рисунок 3.1 – діаграма класу

Визначення технічних вимог до сайту

Мінімальні вимоги до програмно-апаратної частини для розгортання застосунку на основі Python зазвичай визначаються рекомендаціями розробників.

Зазвичай ці вимоги пов'язані з версією Python, та операційної системи, що використовується. Загалом, мінімальні вимоги для розгортання програмного-застосунку на Python можуть бути наступними:

Мінімальні вимоги до програмного забезпечення:

- Python версії 3.10 або вище

Мінімальні вимоги до апаратного забезпечення:

- Процесор з частотою не менше 1 ГГц
- Оперативна пам'ять не менше 512 МБ
- Мінімум 100 МБ вільного місця на жорсткому диску

Щодо сервера для розташування програми, то він також повинен відповідати певним вимогам, зокрема:

- Наявність підтримки віртуалізації, якщо використовується віртуальний сервер
- Мінімум 1 ГБ оперативної пам'яті для стабільної роботи

Загалом, вимоги до програмно-апаратної частини можуть змінюватись в залежності від конкретної реалізації програми, її розміру та специфіки використовуваних функцій.

3.2 Розробка програмного забезпечення

Почати потрібно зі створення нового проекту, та додавання усіх необхідних бібліотек, маючи усі необхідні дані з функціоналом, потрібно розпочати роботу зі створення класу Parser який є підкласом класу Tk з бібліотеки tkinter.

В середині класу Parser є вкладений клас Event, який представляє наукову подію та використовує декоратор dataclass для створення класу з анотаціями типів. Об'єкти класу Event містять наступні поля:

- url: str - URL-адреса події;
- title: str - назва події;
- articles_count: int - кількість статей, написаних викладачами зі списку;
- references_count: int - кількість посилань на викладачів всередині статей;
- authors: Set - набір авторів статей;
- references: Set - набір викладачів, на яких були знайдені посилання;
- score: float - фінальний бал події, який обчислюється за допомогою методу calculate_score().

У класі Event також є метод post_init(), який викликає метод calculate_score() для обчислення фінальної ваги події після ініціалізації об'єкта.

Клас Parser містить також деякі змінні, що відображають значення вагових коефіцієнтів, нижній та верхній ліміти, а також номер останньої події. Клас також містить два набори імен викладачів - professors та last_names - для використання при пошуку посилань в PDF-файлах.

У класі Parser немає конструктора (init()), але він успадковується від класу Tk. Вміст класу Parser складається з опису вкладеного класу Event та змінних, що містять значення вагових коефіцієнтів та набори імен викладачів.

Далі потрібно створити Конструктор класу __init__ визначається, щоб ініціалізувати об'єкт класу при його створенні

```
def __init__(self):
    super(Parser, self).__init__()  # виклик батьківського конструктору

    self.events = []  # сюди будемо записувати усі оцінені події

    #  починаємо розташовування єлементів інтерфейсу

    self.title("CEUR Parser")  # назва вікна
    self.resizable(False, False)
    style = ttk.Style(self)
    style.configure("Treeview", rowheight=40)  # призначаємо таблиці висоту стрічки

    #  інформаційний ярлик
    self.events_label = ttk.Label(
        self,
        text='Перейдіть до налаштувань або одразу натисніть "Старт" щоб розпочати парсінг.',
    )
    self.events_label.grid(row=0, column=0, padx=4, pady=4)

    #  кнопка старту
    self.start_button = ttk.Button(self, text="Старт", command=self.start)
    self.start_button.grid(row=2, column=0, padx=4, pady=4)
```

```
# таблиця подій
columns = ("Назва", "URL", "Бали", "Статей", "Цитувань", "Автори", "Посилання")
self.events_table = ttk.Treeview(
   self, columns=columns, show="headings", height=10
)
for column in columns:
   self.events_table.column(
      column, stretch=True, width=len(column) * 30, anchor=tk.CENTER
   )
for column in columns:
   self.events_table.heading(column, text=column)
self.events_table.grid(row=1, column=0, padx=10, pady=10)

#  контекстне меню
self.menubar = tk.Menu(self)
self.controls = tk.Menu(self.menubar, tearoff=0)
self.controls.add_command(
   label="Допомога", command=self.about, accelerator="F1"
)
self.controls.add_command(
   label="Налаштування", command=self.settings, accelerator="Ctrl+S"
)
self.controls.add_command(
   label="Список викладачів",
   command=self.edit_professors,
   accelerator="Ctrl+P",
)

self.menubar.add_cascade(label="Меню", menu=self.controls)
self.config(menu=self.menubar)
self.bind_all("<Control-s>", self.settings)
self.bind_all("<Control-p>", self.edit_professors)
self.bind_all("<F1>", self.about)

self.about()
```

Першим рядком в конструкторі є виклик конструктора батьківського класу за допомогою функції super(). Це забезпечує успадкування властивостей та методів від батьківського класу до дочірнього класу Parser.

Наступний рядок створює порожній список self.events, до якого будуть додаватися оцінені події.

Після цього код визначає вигляд і розміщення елементів інтерфейсу, які використовуються для взаємодії з користувачем.

Стрічки визначають заголовок вікна, а також призначають його незмінність у відношенні до розмірів.

На рядку style.configure визначається стиль Treeview, який використовується для таблиці подій. У цьому рядку встановлюється висота рядка для кожного рядка таблиці.

Далі код створює два елементи інтерфейсу: events_label та start_button. events_label - це елемент типу Label, який містить текстову інформацію для користувача. start_button - це кнопка, яка запускає парсер.

Далі стрічки визначають таблицю подій, яка містить інформацію про кожну подію, що була оцінена. Для створення цієї таблиці використовується клас Treeview з бібліотеки tkinter. В цих рядках визначаються назви та ширини колонок, а також відображення заголовків.

Після цього визначається контекстне меню, яке містить допоміжні команди для користувача. Встановлюється команда "Допомога" (Help), яка викликає метод about() об'єкту Parser при натисканні на клавішу F1. Метод about() відповідає за відображення інформації про додаток у вікні з допомогою.

Команда "Налаштування" (Settings) викликає метод settings() об'єкту Parser при натисканні на комбінацію клавіш Ctrl+S. Метод settings() відповідає за відображення вікна з налаштуваннями додатку.

Команда "Список викладачів" (Professor list) викликає метод edit_professors() об'єкту Parser при натисканні на комбінацію клавіш Ctrl+P. Метод edit_professors() відповідає за відображення списку викладачів у вікні з налаштуваннями додатку.

Також у наступних рядках створюється головне меню програми, що містить три пункти: "Меню", "Допомога" та "Налаштування". Пункти "Допомога" та "Налаштування" містять підпункти, які відповідають вищезгаданим командам.

Нарешті, на рядках 55-57 встановлюються клавіатурні шорти для команд "Налаштування", "Список викладачів" та "Допомога". Коли користувач натискає відповідну комбінацію клавіш, виконується відповідна команда, яка була встановлена раніше.

Наступною буде функція аналізу PDF-файлу статті за посилання, ознайомитись з якою можна у додатку. Там прописаний проста функція по якій відбувається підключення до серверу, зчитування файлу, далі зчитування

файлу сторінка за сторінкою, та пошук посилань. На яких відбувається пошук викладачів.

Після функцію аналізу PDF файлів буде функція аналізу наукових події у якій , використовуючи бібліотекebs4 , будуть відбуватись обробки статей, маючи 2 лічильники

articles_count = 0

references_count = 0

Цей фрагмент коду Python містить статичний метод get_latest_event(), який оновлює дані щодо останньої події на сайті CEUR-WS (http://ceur-ws.org/). Для цього використовується бібліотека requests для отримання HTML-сторінки з сайту та бібліотека BeautifulSoup для парсінгу сторінки та витягування необхідної інформації.

Код починається зі створення запиту на головну сторінку сайту за допомогою методу requests.get(). У заголовках запиту вказується користувацький агент (User-Agent), що дозволяє ідентифікувати запит як запит з браузера. Отриману відповідь на запит зберігається в змінну r.

Далі використовується об'єкт BeautifulSoup, який створюється з використанням отриманого HTML-коду сторінки в змінній r.content та розбирає його за допомогою парсера "html.parser". Отриманий об'єкт soup містить дерево HTML-елементів сторінки. Наступна функція налаштування програми .

```
#  функція налаштування програми
   def settings(self, event=None):
      # викликаємо нове вікно та додаємо елементи інтерфейсу
      settings_window = tk.Toplevel(self)

      #  підказки юзеру
      ttk.Label(settings_window, text="Вага статті").grid(
         row=0, column=0, padx=4, pady=4
      )
      ttk.Label(settings_window, text="Вага посилання").grid(
         row=1, column=0, padx=4, pady=4
      )
      ttk.Label(settings_window, text="Номер першої події").grid(
         row=2, column=0, padx=4, pady=4
      )
      ttk.Label(
         settings_window, text="Номер останньої події (0 - визначати автоматично)"
      ).grid(row=3, column=0, padx=4, pady=4)
```

```
#  поле вводу ваги статті
article_weight_entry = ttk.Entry(settings_window)
article_weight_entry.grid(row=0, column=1, padx=4, pady=4)
article_weight_entry.insert(tk.END, self.article_weight)

# поле вводу ваги посилання
reference_weight_entry = ttk.Entry(settings_window)
reference_weight_entry.grid(row=1, column=1, padx=4, pady=4)
reference_weight_entry.insert(tk.END, self.reference_weight)

# поле вводу номеру події, від якої буде проводитися парсінг
lower_limit_entry = ttk.Entry(settings_window)
lower_limit_entry.grid(row=2, column=1, padx=4, pady=4)
lower_limit_entry.insert(tk.END, self.lower_limit)

# поле вводу номеру події, до якої буде проводитися парсінг
upper_limit_entry = ttk.Entry(settings_window)
upper_limit_entry.grid(row=3, column=1, padx=4, pady=4)
upper_limit_entry.insert(tk.END, self.upper_limit)
```

Ця частина коду Python створює нове вікно з елементами інтерфейсу для налаштування програми. Функція settings має один аргумент event, який за замовчуванням має значення None.

Починається виконання функції зі створення нового вікна settings_window за допомогою методу Toplevel класу tk.

Далі, за допомогою методу ttk.Label класу ttk, на вікні створюються 4 підписи (Label), які містять пояснення до полів вводу налаштувань. Кожен з цих Label розміщується відповідно до рядка та колонки вікна за допомогою методів grid(row, column).

Далі, за допомогою методу ttk.Entry, на вікні створюються 4 поля вводу (Entry), де користувач може вводити значення налаштувань. Кожен з цих Entry також розміщується відповідно до рядка та колонки вікна за допомогою методів grid(row, column).

Для кожного поля вводу, значення за замовчуванням заповнюється змінними класу, які містять поточні значення налаштувань.

На цьому створення вікна з полями вводу для налаштування програми завершується.

Наступною потрібно написати функцію внесення змін до списку викладачів код буде наступним:

```
def edit_professors(self, event=None):
    # функція збереження змін
    def save():
        try:
            user_input = textarea.get("1.0", tk.END).split(
                ","
            )  # считуємо дані та сплітимо їх за комою
            #  записуємо считані дані до нового набору викладачів
            new_professors = set(
                [
                    professor.strip()
                    for professor in user_input
                    if len(professor.strip()) > 3
                ]
            )
            self.professors = (
                new_professors  # зберігаємо набор як основний набор викладачів
            )
            self.last_names = {
                name.split()[-1] for name in self.professors
            }  # оновлюємо набор прізвищ
            edit_window.destroy()  # закриваємо вікно
        except:
            label[
                "text"
            ] = "Виникла помилка! Перевірте, чи правильно ви ввели імена"

    #  нове вікно та елементи інтерфейсу на ньому
    edit_window = tk.Toplevel(self)
    label = ttk.Label(
        edit_window, text="Введіть імена викладачів через кому"
    )  # лейбл для вказівок
    label.grid(row=0, column=0)
    textarea = tk.Text(edit_window, width=40, height=10, wrap=tk.WORD)
    textarea.grid(row=1, column=0)
    textarea.insert(
        tk.END, ", ".join(self.professors)
    )  # вставляємо поточні дані у текстбокс
    ttk.Button(edit_window, text="Зберегти", command=save).grid(
        row=2, column=0, pady=4, padx=4
    )  # кнопка для зберігання змін
```

Цей фрагмент коду містить функцію edit_professors(), яка відповідає за редагування списку викладачів.

Спочатку визначається функція save(), яка буде виконуватися при натисканні кнопки "Зберегти". Функція save() зчитує дані, введені користувачем у текстовому полі, розділені комами, і записує їх у новий набір викладачів, виключаючи порожні рядки та імена, довжина яких менше 3 символів. Затім оновлений набір викладачів стає основним набором, а також оновлюється набір прізвищ, зберігається ім'я та закривається вікно редагування.

Сама функція edit_professors() створює нове вікно, яке містить текстове поле для введення даних, лейбл з вказівками та кнопку для збереження введених даних. Текстове поле вже заповнене поточним списком викладачів.

Функція edit_professors() може бути викликана як через подію (event), так і без неї.

Ще не мало мажливою буде функція парсінгу. Код представлено нижче:

```
# функція парсінгу
    def parse(self, event=None):
        self.events = []  # очищення списку подій
        self.update_table()  # очищення таблиці
        self.events_label["text"] = "З'єднання з сервером..."
        #  оновлення номеру останньої події
        if self.upper_limit <= 0 or self.upper_limit > self.latest_event:
            self.upper_limit = self.latest_event = self.get_latest_event()

        # парсінг подій
        current_event = self.lower_limit
        while current_event <= self.upper_limit:
            url = f"http://ceur-ws.org/Vol-{current_event}"  # URL поточної події
            r = requests.get(
                url, headers={"User-Agent": self.USER_AGENT}
            )  # робимо запрос на сервер
            if r:  # якщо запрос вдалий
                self.update_counter(current_event)
                event = self.analyze_event(r)  # аналізуємо наукову подію подію
                if (
                    event.score > 0
                ):  # якщо бали більше нуля, подія містить викладачів, які нас цікавлять
                    self.events.append(event)  # додаємо подію до списку
                    self.update_table()  # оновлюємо таблицю
                current_event += 1
            elif r.status_code == 404 or r.status_code == 403:  # якщо ресурсу не існує, пропускаємо його
                current_event += 1
            else:  # якщо сервер відповів відмовою, чекаємо 5 секунд та питаємо ще раз
                sleep(5)

        #  запис до файлу
        self.write_to_file()

        #  вмикаємо елементи управління
        self.start_button["state"] = "normal"
        self.controls.entryconfig("Налаштування", state="normal")
        self.controls.entryconfig("Список викладачів", state="normal")
```

Ця частина коду представляє функцію parse, яка відповідає за парсінг наукових подій з веб-сайту ceur-ws.org. Основні етапи функції можна описати наступним чином:

1. Очистка списку подій і таблиці.
2. Оновлення номеру останньої події, якщо він менший або дорівнює 0 або більший, ніж номер останньої наукової події, доступної на веб-сайті.

3. Парсінг подій від нижнього до верхнього ліміту, заданого в налаштуваннях програми.

4. Робить запит на сервер з використанням бібліотеки requests та обробляє результат.

5. Аналізує наукову подію та, якщо вона містить викладачів, які цікавлять користувача, додає її до списку подій та оновлює таблицю.

6. Записує отримані дані до файлу.

7. Вмикає елементи управління (кнопку "Почати", меню "Налаштування", меню "Список викладачів").

Ця функція викликається при натисканні кнопки "Почати" або по натисненні клавіші Enter, коли фокус знаходиться на полі вводу верхнього ліміту. Останньою із функцій я хотів би описати функцію старту процесу парсінгу в окремому потоці.

```
# функія старту процесу парсінгу в окремому потоці
    def start(self):
        parse_thread = threading.Thread(
            target=self.parse
        )  # починаємо парсінг в окремому треді
        parse_thread.start()
        # вимикаємо елементи управління
        self.start_button["state"] = "disabled"
        self.controls.entryconfig("Налаштування", state="disabled")
        self.controls.entryconfig("Список викладачів", state="disabled")
    # функція оновлювання таблиці подій
    def update_table(self):
        self.events_table.delete(
            *self.events_table.get_children()
        )  # видаляємо дані з таблиці
        #  додаємо кожну подію з відсортованого списку подій
        for event in sorted(self.events, reverse=True):
            self.events_table.insert(
                "",
                tk.END,
                values=(
                    event.title,
                    event.url,
                    event.score,
                    f"{event.articles_count} (+{(event.articles_count * Parser.article_weight):.1f} до суми балів)",
                    f"{event.references_count} (+{(event.references_count * Parser.reference_weight):.1f} до суми
балів)",
                    ", ".join(event.authors),
                    ", ".join(event.references),
                ),          )
```

Цей фрагмент коду містить дві функції. Функція start() запускає процес парсінгу в окремому потоці. Вона створює об'єкт parse_thread, що представляє

потік, в якому буде запущена функція parse(). Після цього викликається метод start() об'єкта parse_thread, що запускає потік. Після початку парсінгу елементи управління, такі як кнопка запуску парсінгу та меню, вимикаються, щоб користувач не міг взаємодіяти з програмою під час парсінгу даних.

Функція update_table() відповідає за оновлення таблиці подій у графічному інтерфейсі програми. Спочатку вона видаляє всі рядки з таблиці викликом методу delete() об'єкта events_table. Потім вона додає нові рядки, що відповідають кожній події зі списку подій, відсортованих у зворотньому порядку за датою. Кожен рядок містить інформацію про заголовок події, посилання на неї, кількість балів, набраних за кількість статей та посилань на неї, список авторів та список посилань на неї. Ці дані отримуються з відповідного об'єкта події. Додавання нових рядків здійснюється викликом методу insert() об'єкта events_table.

Прописавши усі необхідні функції потрібно перейти до тестування. Ознайомитись з усім кодом функціоналу можна у додатку А.

3.3 Тестування програмного забезпечення

Тестування потрібно розпочати с запуску IDE PyCharm після чого потрібно запустити проект.

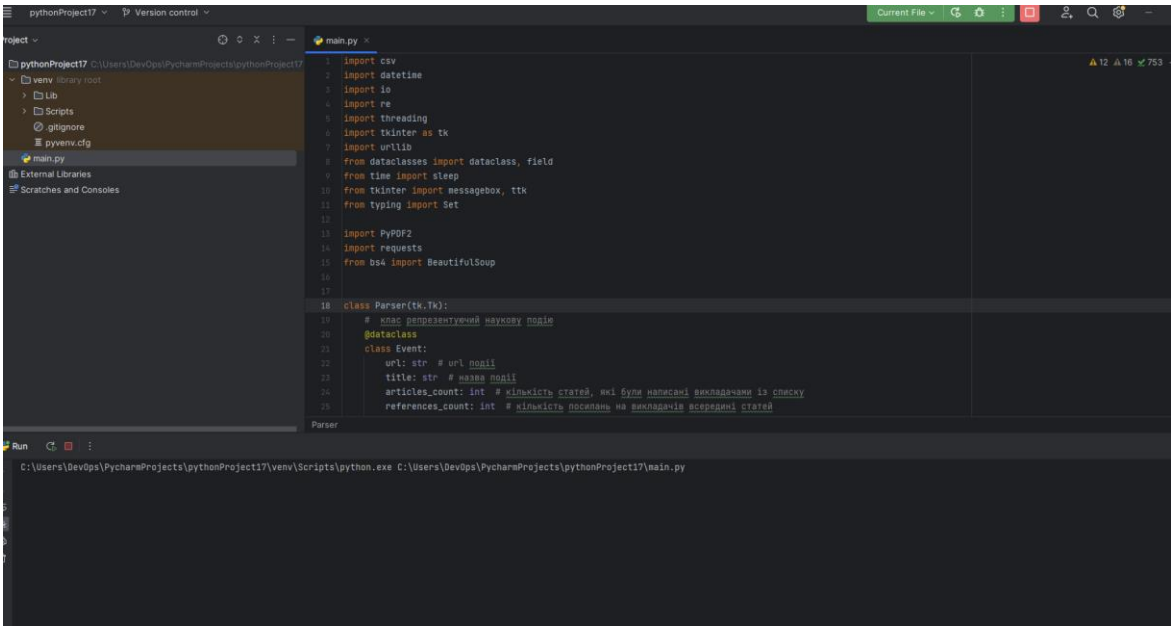

Рисунок 3.2 - IDE PyCharm

При запуску нас зустріне програмним застосунком.

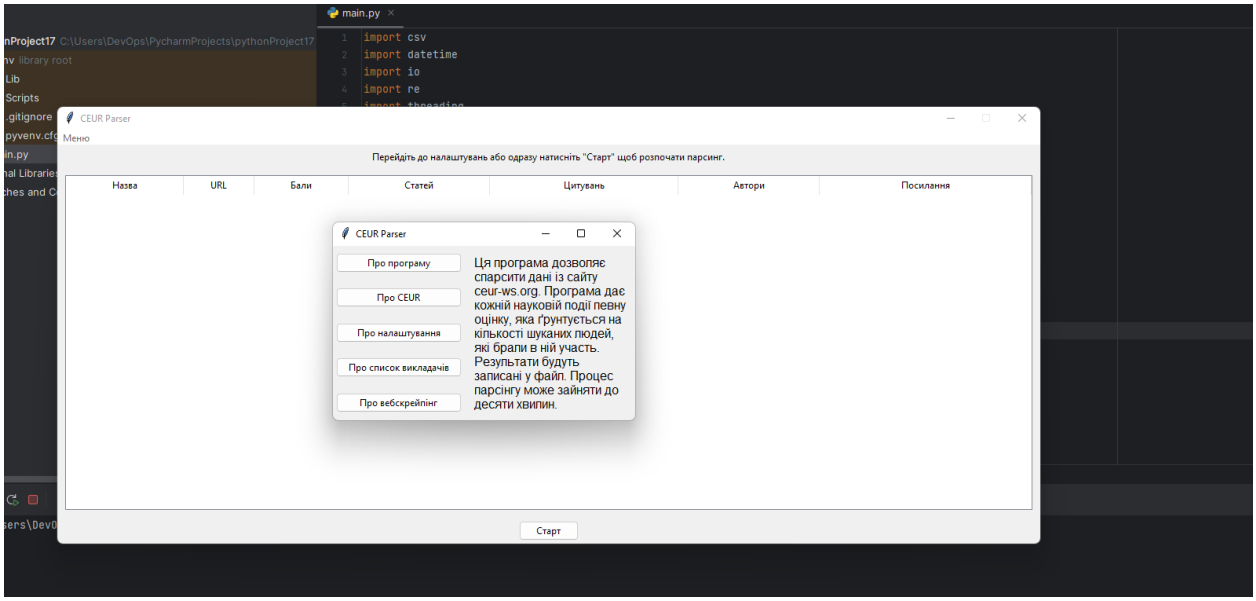

Рисунок 3.3 - Головне меню програми

При відкритті програмного застосунку для парсінгу, можна побачити меню допомоги, в якому буде представлена уся необхідна інформація до програми. Натискаючи на необхідні кнопки можна ознайомитись з усією документацію, усі дії представлені на (рис. 3.3-3.7).

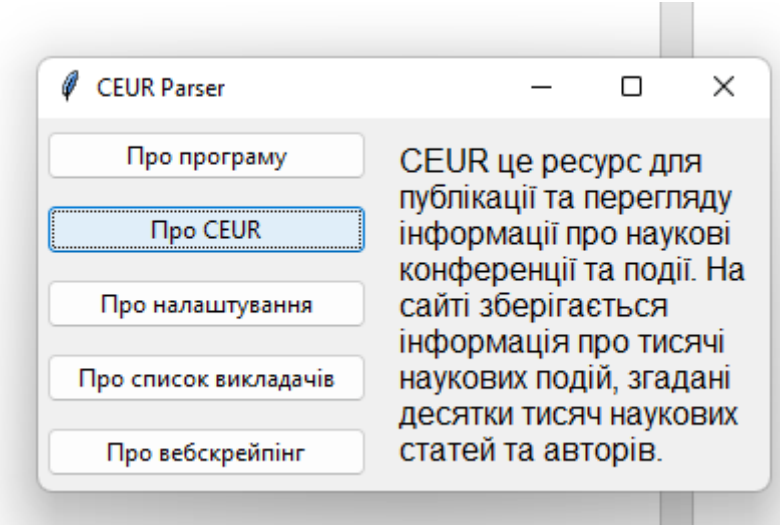

Рисунок 3.4 - Інформація про CEUR

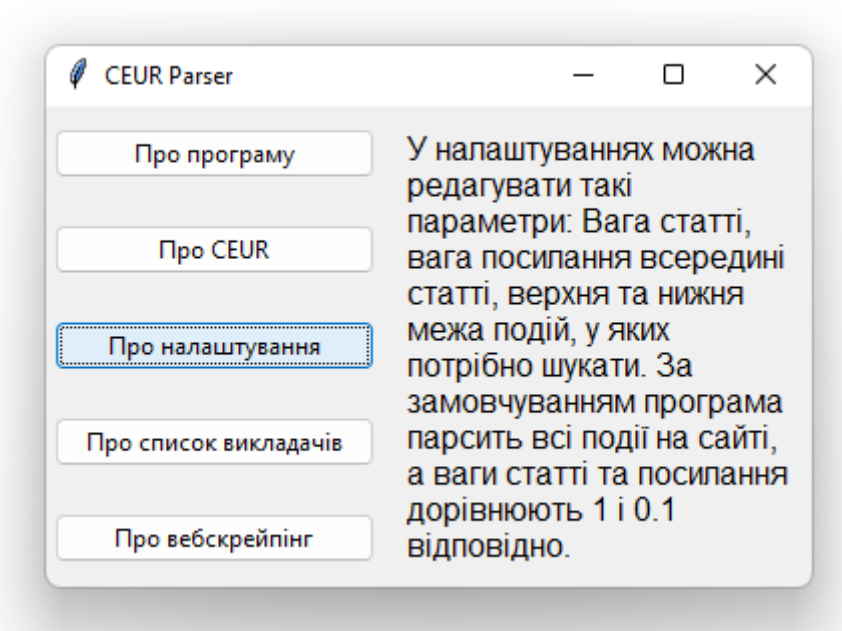

Рисунок 3.5 - Про налаштування

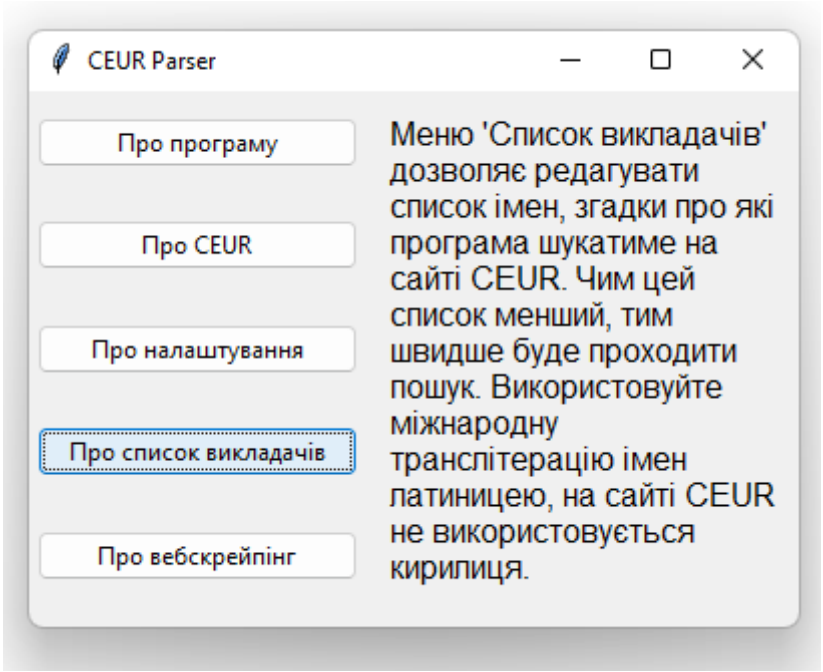

Рисунок 3.6 - Про список викладачів

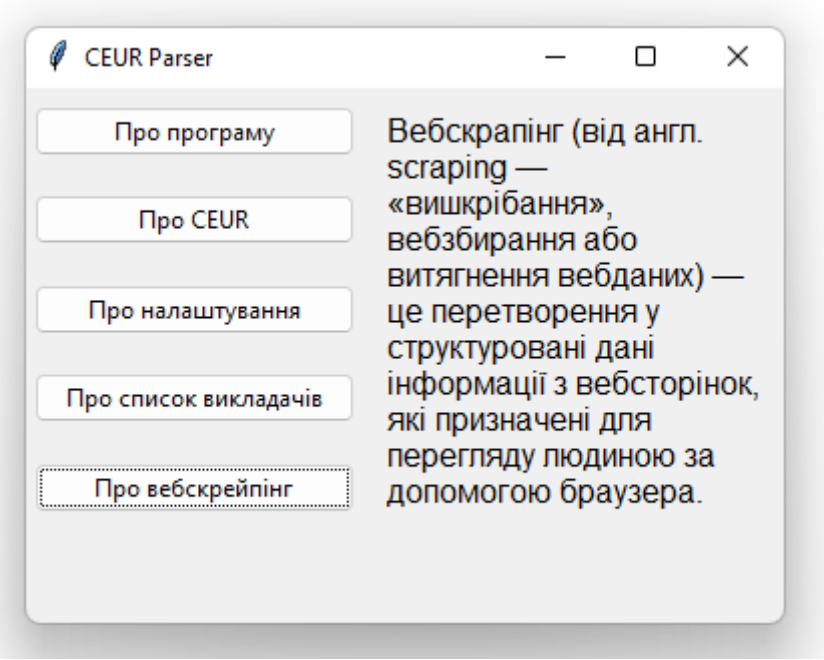

Рисунок 3.7 - Про вебскрейпінг

Також через меню можна перейти, у вікно де через кому задають імена викладачів.

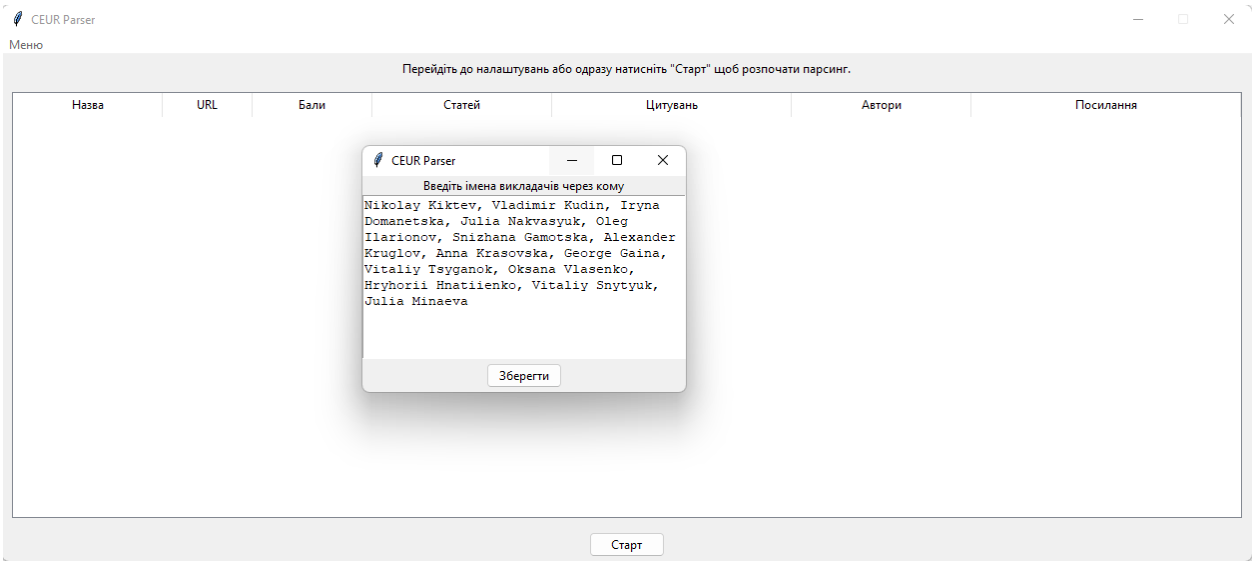

Рисунок 3.8 - IDE PyCharm

Також у наступному меню є налаштування до парсінгу. У якому можна більш детальніше усе налаштувати рис (3.9).

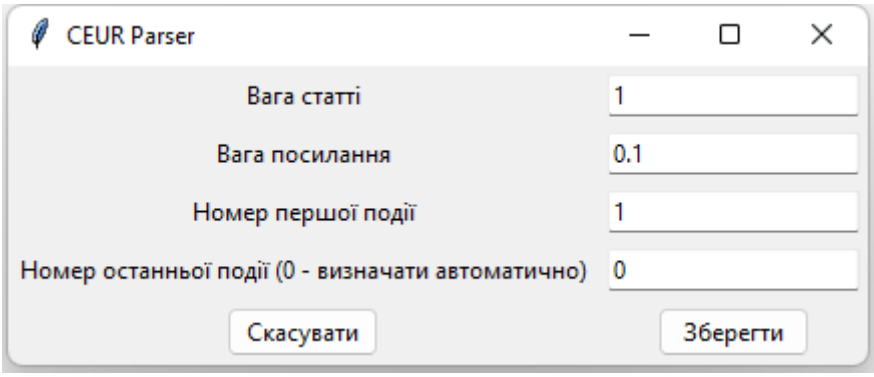

Рисунок 3.9 - IDE PyCharm

Після того як ми зробили усі необхідні налаштування потрібно натиснути «Старт» після підключення до серверу прослідує парсування усіх статей, це може зайняти певний час. Прогрес поточних подій вказано на рис.3.10

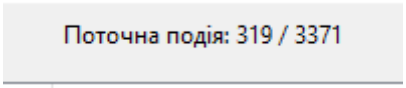

Рисунок 3.10 - IDE PyCharm

Коли розрахунок поточних подій дійде до кінця, з’явиться список відпарсованих необхідних статей, увесь результат буде відразу відправлено у файл формату .csv у тому самому каталозі що й проект рис.1.1.

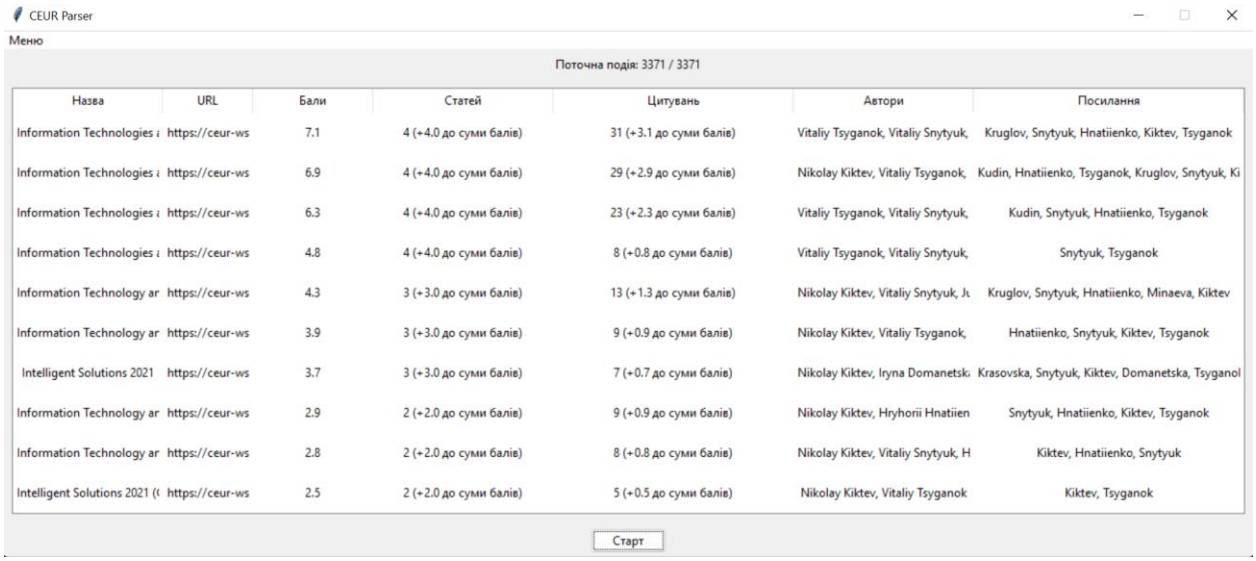

Рисунок 3.11 - Результат відпрацювання програми

В програмі відбувається сортування за балами.

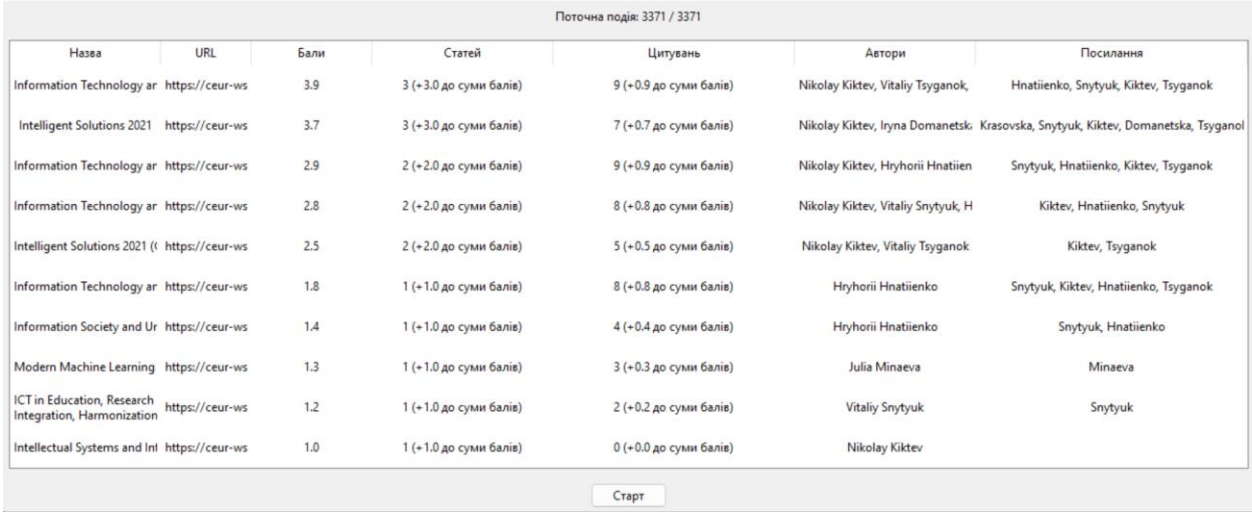

Рисунок 3.12 - Результат відпрацювань програми

Увесь даний результат буде збережено у файл формату .csv. Який розташовано у тій самій папці що й проект.

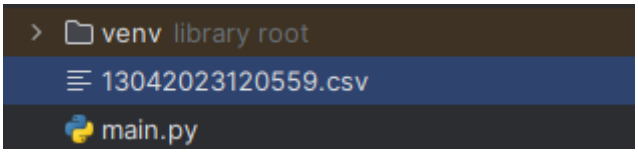

Рисунок 3.13 - CSV файл

При відкритті цього файлу, можна побачити увесь необхідний текст, з вказаним статей їх url , ваги статей та інше.

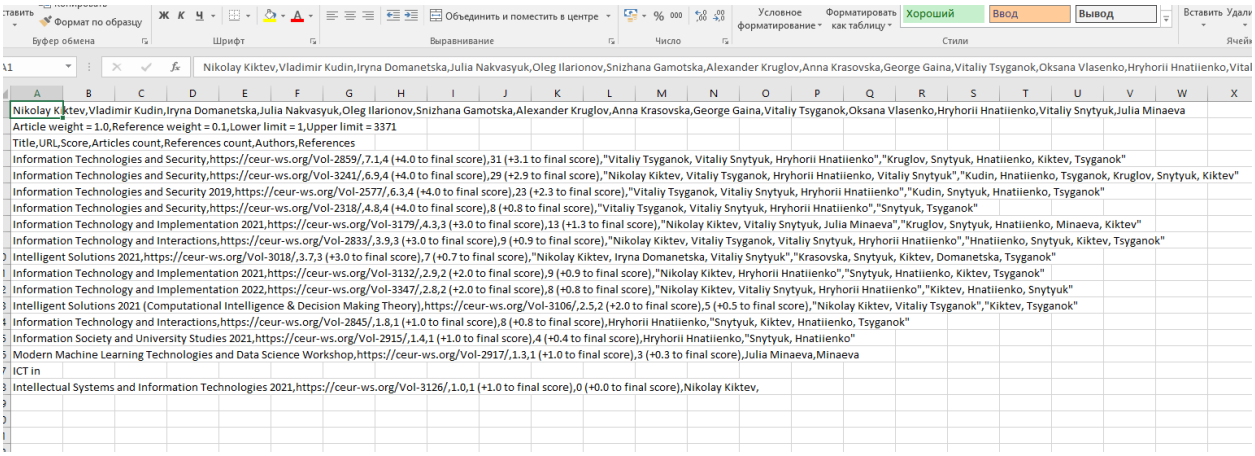

Рисунок 3.14 - Файл csv

ВИСНОВКИ

В представленій кваліфікаційній роботі створено інформаційне й програмне забезпечення для збору та аналізу даних наукової діяльності співробітників вузу з використанням інструментарію парсінгу. У цьому висновку будуть описані основні функції та можливості програми, а також її потенційні переваги для управління науковою діяльністю вузу.

Програма розроблена з метою збору та аналізу даних про наукову діяльність співробітників вузу.

У процесі дослідження було розроблено програмне забезпечення для збору та аналізу даних наукової діяльності співробітників вузу з використанням інструментарію парсінгу. Програма здатна збирати інформацію про наукові події з сайту https://ceur-ws.org/, аналізувати їх та відображати результати в графічному інтерфейсі.. В результаті роботи було виявлено, що розроблене програмне забезпечення дозволяє ефективно збирати та аналізувати великі обсяги даних про наукову діяльність співробітників вузу.

Основною функцією програми є визначення присутності наукових працівників вузу серед авторів наукових статей, які були опубліковані в рамках наукових подій. Для цього використовується список працівників, який може бути налаштований користувачем, а також пошук по посиланнях в кінці статей. На основі кількості присутніх працівників зі списку в наукових подіях, присвоюється фінальна оцінка події за формулою (кількість авторів * вага автора + кількість згадок * вага згадки).

Програма має графічний інтерфейс з вікном налаштувань, головним вікном та вікном довідки. Головне вікно містить таблицю з даними, які програма зібрала та проаналізувала, включаючи назву наукової події, посилання на неї, фінальну оцінку та її пояснення, знайдених авторів та згадок. Таблиця оновлюється в режимі реального часу. В вікні налаштувань можна змінювати список шуканих працівників, вагу авторства та згадки, а також

встановлювати межі парсінгу на сайті. В вікні довідки міститься інформація про програму та її функції.

Завдяки проведеним тестування можна впевнено сказати що даний програмний застосунок має увесь необхідний та працюючий функціонал що може стабільно та швидко працювати. Розроблене програмне забезпечення може бути корисним інструментом для управління науковою діяльністю співробітників вузу, що дозволить збільшити ефективність та якість наукових досліджень, а також підвищити рівень конкурентоспроможності вузу в науковому середовищі.

СПИСОК ВИКОРОСТАНИХ ДЖЕРЕЛ

1. Requests: HTTP for Humans [Електронний ресурс]. – Режим доступу: https://requests.readthedocs.io/en/latest/, вільний (дата звернення: 13.04.2023).

2. NumPy [Електронний ресурс]. – Режим доступу: https://numpy.org/, вільний (дата звернення: 13.04.2023).

3. Beautiful Soup Documentation [Електронний ресурс]. – Режим доступу: https://www.crummy.com/software/BeautifulSoup/bs4/doc/, вільний (дата звернення: 13.04.2023).

4. Python Software Foundation. Python 3.9.9 documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/, вільний (дата звернення: 13.04.2023).

5. Scrapy. An open source and collaborative web crawling framework for Python [Електронний ресурс]. – Режим доступу: https://scrapy.org/, вільний (дата звернення: 13.04.2023).

6. Федоров, А. В. Основи парсінгу веб-сторінок [Текст]/А. В. Федоров, А. А. Полухін // Інформаційні технології в освіті. - 2016. - № 26. - С. 93-104.

7. Графська, О. А. Аналіз методів парсінгу веб-сторінок [Текст] / О. А. Графська // Матеріали міжнародної науково-практичної конференції «Сучасні технології, економіка та освіта». - 2018. - С. 68-71.

ДОДАТОК А (ceur_parser.py)

```
import csv
import datetime
import io
import re
import threading
import tkinter as tk
import urllib
from dataclasses import dataclass, field
from time import sleep
from tkinter import messagebox, ttk
from typing import Set

import PyPDF2
import requests
from bs4 import BeautifulSoup


class Parser(tk.Tk):
    #  клас репрезентуючий наукову подію
    @dataclass
    class Event:
        url: str  # url події
        title: str  # назва події
        articles_count: int  # кількість статей, які були написані викладачами із списку
        references_count: int  # кількість посилань на викладачів всередині статей
        authors: Set  # набор авторів статей
        references: Set  # набор викладачів, на яких були знайдені посилання
        score: float = field(init=False)  # фінальний бал події

        #  функція для обчислення фінального балу події
        def calculate_score(self):
            self.score = round(
                self.articles_count * Parser.article_weight
                + self.references_count * Parser.reference_weight,
                1,
            )

        # цей метод буде викликан після __init__(). До конструктора не передається фінальна вага та її потрібно обчислити
        def __post_init__(self):
            self.calculate_score()  # обчисляємо фінальну вагу

        #  методи для сортування подій
        def __gt__(self, other):
            return self.score > other.score

        def __lt__(self, other):
            return self.score < other.score

    #  Значення вагових коефіцієнтів за замовчуванням, які юзер може редагувати
```

```
    article_weight = 1
    reference_weight = 0.1
    lower_limit = 1
    upper_limit = 0
    latest_event = 3139

    USER_AGENT = "Mozilla/5.0"
    #  Імена викладачів факультету
    professors = {
        "Vitaliy Snytyuk",
        "Vitaliy Tsyganok",
        "Hryhorii Hnatiienko",
        "Nikolay Kiktev",
        "Oleg Ilarionov",
        "Oksana Vlasenko",
        "George Gaina",
        "Snizhana Gamotska",
        "Iryna Domanetska",
        "Alexander Kruglov",
        "Anna Krasovska",
        "Vladimir Kudin",
        "Julia Minaeva",
        "Julia Nakvasyuk",
    }
    #  Прізвища викладачів для пошуку посилань в PDF-файлах. Частіш за все там зазначені
тільки прізвище та ініціали
    #  тому пошук повного імені не дасть результату
    last_names = {name.split()[-1] for name in professors}

    def __init__(self):
        super(Parser, self).__init__()  # виклик батьківського конструктору

        self.events = []  # сюди будемо записувати усі оцінені події

        #  починаємо розташовування єлементів інтерфейсу

        self.title("CEUR Parser")  # назва вікна
        self.resizable(False, False)
        style = ttk.Style(self)
        style.configure("Treeview", rowheight=40)  # призначаємо таблиці висоту стрічки

        #  інформаційний ярлик
        self.events_label = ttk.Label(
            self,
            text='Перейдіть до налаштувань або одразу натисніть "Старт" щоб розпочати
парсінг.',
        )
        self.events_label.grid(row=0, column=0, padx=4, pady=4)

        #  кнопка старту
        self.start_button = ttk.Button(self, text="Старт", command=self.start)
        self.start_button.grid(row=2, column=0, padx=4, pady=4)
```

```
    # таблиця подій
    columns = ("Назва", "URL", "Бали", "Статей", "Цитувань", "Автори", "Посилання")
    self.events_table = ttk.Treeview(
        self, columns=columns, show="headings", height=10
    )
    for column in columns:
        self.events_table.column(
            column, stretch=True, width=len(column) * 30, anchor=tk.CENTER
        )
    for column in columns:
        self.events_table.heading(column, text=column)
    self.events_table.grid(row=1, column=0, padx=10, pady=10)

    #  контекстне меню
    self.menubar = tk.Menu(self)
    self.controls = tk.Menu(self.menubar, tearoff=0)
    self.controls.add_command(
        label="Допомога", command=self.about, accelerator="F1"
    )
    self.controls.add_command(
        label="Налаштування", command=self.settings, accelerator="Ctrl+S"
    )
    self.controls.add_command(
        label="Список викладачів",
        command=self.edit_professors,
        accelerator="Ctrl+P",
    )

    self.menubar.add_cascade(label="Меню", menu=self.controls)
    self.config(menu=self.menubar)
    self.bind_all("<Control-s>", self.settings)
    self.bind_all("<Control-p>", self.edit_professors)
    self.bind_all("<F1>", self.about)

    self.about()

#  функція аналізу PDF-файлу статті за посилання
def analyze_article(self, article_url):
    #  підключення до серверу та считування файлу
    r = urllib.request.Request(article_url, headers={"User-Agent": self.USER_AGENT})
    try:
        article_pdf = urllib.request.urlopen(r).read()  # спроба відкрити url
    except urllib.error.HTTPError as e:
        if e.code == 404:  # якщо ресурса не існує, пропускаємо його
            return 0, set()
        elif (
            e.code == 503
        ):  # якщо сервер тимчасово не відповідає, чекаємо 3 секунди та пробуємо ще раз
            sleep(3)
            return self.analyze_article(article_url)
```

```
        #  считуємо файл
        article_bytes = io.BytesIO(article_pdf)
        article = PyPDF2.PdfReader(article_bytes)

        # ініціалізація лічильнику статей та пустого набору викладачів, який буде
заповнюватися знайденими викладачами із
        # списку
        references_count = 0
        found_professors = set()

        #  Препроцессінг даних із PDF-файлу
        pages = []
        #  Проходимо усі сторінки файлу з кінця, тому що список посилань зазвичай
знаходиться наприкінці статті
        for page in article.pages[::-1]:
            # прибираємо усі мусорні символи із кожної сторінки
            page = re.sub(r"[\W]", "", page.extract_text())

            #  якщо дойшли до сторінки, де починаютється список посилань, виходимо з циклу
            if "references" in page.lower():
                page = re.sub(r".+references", "", page, 1, re.IGNORECASE)
                pages.append(page)  # додаємо сторінку до нового масиву
                break
            else:
                pages.append(page)  # додаємо сторінку до нового масиву

        #  Пошук викладачів на оброблених сторінках списку посилань
        for page in pages:
            for professor in self.last_names:
                professor_mentions = page.count(professor)

                #  якщо знайшли викладача на сторінці, додаємо його до набору знайдених
викладачів та збільшуємо лічильник
                if professor_mentions:
                    references_count += professor_mentions
                    found_professors.add(professor)

        return references_count, found_professors

    #  функція аналізу наукової події
    def analyze_event(self, request):
        #  використовуємо BS4 задля обробки відповіді з сервера
        soup = BeautifulSoup(request.content, "html.parser")

        #  знаходимо назву події двома способами
        title = soup.find("span", {"class": "CEURVOLTITLE"})
        if not title:  # якщо перший не дає результатів
            title = soup.find("h1")  # пробуємо інший
        title = title.text if title else "Не знайдено"

        #  лічильники
        articles_count = 0
```

```
        references_count = 0

        #  сюди будемо додавати знайдених викладачів
        participated_professors = set()  # якщо автор
        referenced_professors = set()  # якщо посилання

        #  знаходимо на сторінці усі статті
        articles = soup.find_all("li")
        if articles:
            #  оброблюємо кожну статтю
            for article in articles:
                #  знаходимо авторів статті першим способом
                authors = {
                    span.text
                    for span in article.findChildren(
                        "span", {"class": "CEURAUTHOR"}, recursive=False
                    )
                }
                #  якщо не знайшли, шукаємо другим способом
                if not authors:
                    span = article.findChild("span", {"class": "CEURAUTHORS"})
                    if span:
                        authors = set(span.text.split(", "))
                #  якщо не знайшли, шукаємо третім способом
                if not authors:
                    i = article.findChild("i")
                    if i:
                        authors = set(i.text.split(", "))

                #  знаходимо перетин набору викладачів та набору авторів статті
                matched_authors = self.professors.intersection(authors)

                #  якщо набор викладачів та набор авторів статті перетинаються
                if matched_authors:
                    articles_count += 1  # збільшуємо лічильник на 1
                    pdf_link = article.findChild("a")  # знаходимо посилання на PDF
                    article_references, found_professors = self.analyze_article(
                        request.url + pdf_link["href"]
                    )  # аналізуємо статтю

                    references_count += article_references  # додаємо до лічильнику події значення
лічильнику статті
                    participated_professors.update(
                        matched_authors
                    )  # додаємо нових викладачів до набору викладачів-авторів
                    referenced_professors.update(
                        found_professors
                    )  # додаємо нових викладачів до набору посилань, якщо знайдені

        #  створюємо екземпляр класу події та повертаємо його
        return self.Event(
            request.url,
```

```
            title,
            articles_count,
            references_count,
            participated_professors,
            referenced_professors,
        )

    #  функція оновлення даних щодо останньої події на сайті
    @staticmethod
    def get_latest_event():
        r = requests.get(
            "http://ceur-ws.org/", headers={"User-Agent": Parser.USER_AGENT}
        )  # питаємо головну сторінку сайту
        soup = BeautifulSoup(r.content, "html.parser")  # парсимо її
        main_table = soup.find("table", {"id": "MAINTABLE"})  # шукаємо основну таблицю
        latest_event = main_table.findChild(
            "a", {"name": re.compile(r"Vol-\d+")}
        )  # беремо останню подію

        #  якщо вдалося
        if latest_event:
            return int(
                latest_event["name"].split("-")[-1]
            )  # повертаємо номер знайденої події
        else:
            return 3139  # інакше повертаємо номер 3139 - номер останньої події на момент
23.05.2022

    #  функція налаштування програми
    def settings(self, event=None):
        # викликаємо нове вікно та додаємо елементи інтерфейсу
        settings_window = tk.Toplevel(self)

        #  підказки юзеру
        ttk.Label(settings_window, text="Вага статті").grid(
            row=0, column=0, padx=4, pady=4
        )
        ttk.Label(settings_window, text="Вага посилання").grid(
            row=1, column=0, padx=4, pady=4
        )
        ttk.Label(settings_window, text="Номер першої події").grid(
            row=2, column=0, padx=4, pady=4
        )
        ttk.Label(
            settings_window, text="Номер останньої події (0 - визначати автоматично)"
        ).grid(row=3, column=0, padx=4, pady=4)
        #  поле вводу ваги статті
        article_weight_entry = ttk.Entry(settings_window)
        article_weight_entry.grid(row=0, column=1, padx=4, pady=4)
        article_weight_entry.insert(tk.END, self.article_weight)

        # поле вводу ваги посилання
```

```
        reference_weight_entry = ttk.Entry(settings_window)
        reference_weight_entry.grid(row=1, column=1, padx=4, pady=4)
        reference_weight_entry.insert(tk.END, self.reference_weight)

        # поле вводу номеру події, від якої буде проводитися парсінг
        lower_limit_entry = ttk.Entry(settings_window)
        lower_limit_entry.grid(row=2, column=1, padx=4, pady=4)
        lower_limit_entry.insert(tk.END, self.lower_limit)

        # поле вводу номеру події, до якої буде проводитися парсінг
        upper_limit_entry = ttk.Entry(settings_window)
        upper_limit_entry.grid(row=3, column=1, padx=4, pady=4)
        upper_limit_entry.insert(tk.END, self.upper_limit)

        #  функція зберігання введених даних
        def save():
            errors = []

            try:
                #  зчитування даних з полей
                entered_article_weight = float(article_weight_entry.get())
                entered_reference_weight = float(reference_weight_entry.get())
                entered_lower_limit = int(lower_limit_entry.get())
                entered_upper_limit = int(upper_limit_entry.get())

                #  валідація
                if 0 >= entered_article_weight or entered_article_weight > 1:
                    errors.append("Вага статті повинна бути в межах від 0 до 1")

                if 0 >= entered_reference_weight or entered_reference_weight > 1:
                    errors.append("Вага посилання повинна бути в межах від 0 до 1")

                if entered_upper_limit < 0:
                    errors.append("Остання подія не може бути негативним числом. ")

                if entered_upper_limit > 0:
                    if (
                        1 > entered_lower_limit
                        or entered_lower_limit > entered_upper_limit
                        or entered_upper_limit > self.latest_event
                    ):
                        errors.append(
                            "Вводіть числа за наступним законом: 1 <= Перша подія <= Остання подія
<= Остання подія на сайті"
                        )
                else:
                    if (
                        1 > entered_lower_limit
                        or entered_lower_limit > self.latest_event
                    ):
                        errors.append(
```

```
                    "Номер першої події повинен бути більше 0 і менше номера останньої
події."
                )

            except ValueError:
                errors.append(
                    "Будь ласка, використовуйте лише цифри. Десятковий роздільник - крапка. "
                )

            if errors:
                messagebox.showerror(
                    "Помилка!", ";\n".join(errors)
                )  # показ помилок, якщо вони присутні
            else:
                #  зберігання даних
                self.article_weight = entered_article_weight
                self.reference_weight = entered_reference_weight
                self.lower_limit = entered_lower_limit
                self.upper_limit = entered_upper_limit

                #  закриття вікна
                settings_window.destroy()

        #  дві кнопки для скасування або збереження змін
        ttk.Button(
            settings_window, text="Скасувати", command=settings_window.destroy
        ).grid(row=4, column=0, pady=4, padx=4)
        ttk.Button(settings_window, text="Зберегти", command=save).grid(
            row=4, column=1, pady=4, padx=4
        )

    # функція внесення змін до списку викладачів
    def edit_professors(self, event=None):
        # функція збереження змін
        def save():
            try:
                user_input = textarea.get("1.0", tk.END).split(
                    ","
                )  # считуємо дані та сплітимо їх за комою
                #  записуємо считані дані до нового набору викладачів
                new_professors = set(
                    [
                        professor.strip()
                        for professor in user_input
                        if len(professor.strip()) > 3
                    ]
                )
                self.professors = (
                    new_professors  # зберігаємо набор як основний набор викладачів
                )
                self.last_names = {
                    name.split()[-1] for name in self.professors
```

```
            }  # оновлюємо набор прізвищ
            edit_window.destroy()  # закриваємо вікно
        except:
            label[
                "text"
            ] = "Виникла помилка! Перевірте, чи правильно ви ввели імена"

    #  нове вікно та елементи інтерфейсу на ньому
    edit_window = tk.Toplevel(self)
    label = ttk.Label(
        edit_window, text="Введіть імена викладачів через кому"
    )  # лейбл для вказівок
    label.grid(row=0, column=0)
    textarea = tk.Text(edit_window, width=40, height=10, wrap=tk.WORD)
    textarea.grid(row=1, column=0)
    textarea.insert(
        tk.END, ", ".join(self.professors)
    )  # вставляємо поточні дані у текстбокс
    ttk.Button(edit_window, text="Зберегти", command=save).grid(
        row=2, column=0, pady=4, padx=4
    )  # кнопка для зберігання змін

def about(self, event=None):
    def display_text(text):
        help_label["text"] = text

    greet_window = tk.Toplevel(self)
    greet_window.attributes("-topmost", "true")
    about = (
        "Ця програма дозволяє спарсити дані із сайту ceur-ws.org. Програма дає кожній науковій події певну "
        "оцінку, яка ґрунтується на кількості шуканих людей, які брали в ній участь. Результати будуть "
        "записані у файл. Процес парсінгу може зайняти до десяти хвилин. "
    )
    ceur = (
        "CEUR це ресурс для публікації та перегляду інформації про наукові конференції та події. На сайті "
        "зберігається інформація про тисячі наукових подій, згадані десятки тисяч наукових статей та авторів. "
    )
    settings = (
        "У налаштуваннях можна редагувати такі параметри: Вага статті, вага посилання всередині статті, "
        "верхня та нижня межа подій, у яких потрібно шукати. За замовчуванням програма парсить всі події "
        "на сайті, а ваги статті та посилання дорівнюють 1 і 0.1 відповідно. "
    )
    professors = (
        "Меню 'Список викладачів' дозволяє редагувати список імен, згадки про які програма шукатиме на "
```

```
            "сайті CEUR. Чим цей список менший, тим швидше буде проходити пошук. Використовуйте міжнародну "
            "транслітерацію імен латиницею, на сайті CEUR не використовується кирилиця."
        )
        parsing = (
            "Вебскрапінг (від англ. scraping — «вишкрібання», вебзбирання або витягнення вебданих) — "
            "це перетворення у структуровані дані інформації з вебсторінок, "
            "які призначені для перегляду людиною за допомогою браузера."
        )
        ttk.Button(
            greet_window,
            text="Про програму",
            command=lambda: display_text(about),
            width=25,
        ).grid(row=0, column=0, padx=4, pady=4)
        ttk.Button(
            greet_window, text="Про CEUR", command=lambda: display_text(ceur), width=25
        ).grid(row=1, column=0, padx=4, pady=4)
        ttk.Button(
            greet_window,
            text="Про налаштування",
            command=lambda: display_text(settings),
            width=25,
        ).grid(row=2, column=0, padx=4, pady=4)
        ttk.Button(
            greet_window,
            text="Про список викладачів",
            command=lambda: display_text(professors),
            width=25,
        ).grid(row=3, column=0, padx=4, pady=4)
        ttk.Button(
            greet_window,
            text="Про вебскрейпінг",
            command=lambda: display_text(parsing),
            width=25,
        ).grid(row=4, column=0, padx=4, pady=4)

        help_label = ttk.Label(
            greet_window, text=about, wraplength=200, justify=tk.LEFT, font="Arial 12"
        )
        help_label.grid(row=0, column=1, rowspan=5, padx=10, pady=10)

    # функція парсінгу
    def parse(self, event=None):
        self.events = []  # очищення списку подій
        self.update_table()  # очищення таблиці
        self.events_label["text"] = "З'єднання з сервером..."
        #  оновлення номеру останньої події
        if self.upper_limit <= 0 or self.upper_limit > self.latest_event:
            self.upper_limit = self.latest_event = self.get_latest_event()
```

```
        # парсінг подій
        current_event = self.lower_limit
        while current_event <= self.upper_limit:
            url = f"http://ceur-ws.org/Vol-{current_event}"  # URL поточної події
            r = requests.get(
                url, headers={"User-Agent": self.USER_AGENT}
            )  # робимо запрос на сервер
            if r:  # якщо запрос вдалий
                self.update_counter(current_event)
                event = self.analyze_event(r)  # аналізуємо наукову подію подію
                if (
                    event.score > 0
                ):  # якщо бали більше нуля, подія містить викладачів, які нас цікавлять
                    self.events.append(event)  # додаємо подію до списку
                    self.update_table()  # оновлюємо таблицю
                current_event += 1
            elif r.status_code == 404 or r.status_code == 403:  # якщо ресурсу не існує,
пропускаємо його
                current_event += 1
            else:  # якщо сервер відповів відмовою, чекаємо 5 секунд та питаємо ще раз
                sleep(5)

        #  запис до файлу
        self.write_to_file()

        #  вмикаємо елементи управління
        self.start_button["state"] = "normal"
        self.controls.entryconfig("Налаштування", state="normal")
        self.controls.entryconfig("Список викладачів", state="normal")

    # функія старту процесу парсінгу в окремому потоці
    def start(self):
        parse_thread = threading.Thread(
            target=self.parse
        )  # починаємо парсінг в окремому треді
        parse_thread.start()

        # вимикаємо елементи управління
        self.start_button["state"] = "disabled"
        self.controls.entryconfig("Налаштування", state="disabled")
        self.controls.entryconfig("Список викладачів", state="disabled")

    # функція оновлювання таблиці подій
    def update_table(self):
        self.events_table.delete(
            *self.events_table.get_children()
        )  # видаляємо дані з таблиці

        #  додаємо кожну подію з відсортованого списку подій
        for event in sorted(self.events, reverse=True):
            self.events_table.insert(
                "",
```

```
            tk.END,
            values=(
                event.title,
                event.url,
                event.score,
                f"{event.articles_count} (+{(event.articles_count * Parser.article_weight):.1f} до
суми балів)",
                f"{event.references_count} (+{(event.references_count *
Parser.reference_weight):.1f} до суми балів)",
                ", ".join(event.authors),
                ", ".join(event.references),
            ),
        )

    # функція для оновлення інформаційного ярлику
    def update_counter(self, current_event):
        if self.events_label:
            self.events_label[
                "text"
            ] = f"Поточна подія: {current_event} / {self.upper_limit}"

    def write_to_file(self):
        with open(
            f"{datetime.datetime.now().strftime('%d%m%Y%H%M%S')}.csv",
            "w",
            encoding="utf8",
            newline="",
        ) as csvfile:
            writer = csv.writer(csvfile, delimiter=",")
            writer.writerow(self.professors)
            writer.writerow(
                [
                    f"Article weight = {self.article_weight}",
                    f"Reference weight = {self.reference_weight}",
                    f"Lower limit = {self.lower_limit}",
                    f"Upper limit = {self.upper_limit}",
                ]
            )
            writer.writerow(
                [
                    "Title",
                    "URL",
                    "Score",
                    "Articles count",
                    "References count",
                    "Authors",
                    "References",
                ]
            )
            for event in sorted(self.events, reverse=True):
                writer.writerow(
                    [
```

```
                    event.title,
                    event.url,
                    event.score,
                    f"{event.articles_count} (+{(event.articles_count * Parser.article_weight):.1f} to
final score)",
                    f"{event.references_count} (+{(event.references_count *
Parser.reference_weight):.1f} to final score)",
                    ", ".join(event.authors),
                    ", ".join(event.references),
                ]
            )


app = Parser()
app.mainloop()
```