**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»**

Навчально-науковий інститут атомної та теплової енергетики

Кафедра інженерії програмного забезпечення в енергетиці

«На правах рукопису»
УДК 004.4

ДО ЗАХИСТУ ДОПУЩЕНО
В.о. завідувача кафедри
____ Олександр КОВАЛЬ

……. _....._2024 р.

# Магістерська дисертація

**на здобуття ступеня магістра**
**за освітньо-професійною програмою «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці»**
**зі спеціальності 121 Інженерія програмного забезпечення**
**на тему: «Збір та первинна обробка даних наукової діяльності з різнорідних джерел»**

Виконав: студент 2 курсу, групи ТВ-22мп
Василенко Микита Андрійович _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник: доцент, к.т.н., Кузьміних В.О. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент: с.н.с., к.т.н. Сенченко В.Р. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2024

**Національний технічний університет України**
**«Київський політехнічний інститут ім. Ігоря Сікорського»**

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці
Рівень вищої освіти другий, магістерський
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Інженерія програмного забезпечення інтелектуальних кібер - фізичних систем в енергетиці»

ЗАТВЕРДЖУЮ
В.о.завідувача кафедри
_____ Олександр КОВАЛЬ
(підпис)

____ _____ 2024р.

# З А В Д А Н Н Я
## НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

Василенко Микиті Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації:
Збір та первинна обробка даних наукової діяльності з різнорідних джерел
науковий керівник к.т.н., доцент, Кузьміних В.О.
( прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «06»листопада 2023 року № 5152-с
2. Строк подання студентом дисертації 10 січня 2024 року
3. Об'єктами дослідження є наукові твори, дисертації, тощо – які є результатами досліджень(діяльності) науковців.
4. Вихідні дані до роботи програмний продукт для збору роботи користувача з даними, необхідними для подальшого аналізу.
5. Перелік питань, які потрібно розробити провести пошук та аналіз літератури, джерел та матеріалів; виявити існуючі аналогічні системи з подібним функціоналом; обрати технічні засобіи розробки; розробити архітектуру та структуру програмного продукту та пропрацювати роботу користувача з системою; розробити програмний продукт, який буде відповідати поставленим вимогам; провести тестування програмного продукту та перевірити його на працездатність
6. Орієнтований перелік ілюстративного матеріалу огляд існуючих систем, планування архітектури системи, планування структури програмного застосунку, діаграма прецедентів, діаграми класів, тестування системи
7. Дата видачі завдання «01» листопада 2022 р.

# КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів виконання магістерської дисертації | строки виконання етапів магістерської дисертації | Примітка |
|---|---|---|---|
| 1 | Отримання завдання | 01.11.2022 | виконано |
| 2 | Дослідження предметної області | 01.11.2022 – 01.12.2022 | виконано |
| 3 | Постановка вимог до проєктування системи | 01.12.2022 – 15.12.2022 | виконано |
| 4 | Дослідження існуючих рішень | 15.12.2022 – 03.01.2023 | виконано |
| 5 | Розробка програмного продукту | 03.03.2023 – 20.09.2023 | виконано |
| 6 | Тестування | 20.09.2023 – 15.10.2023 | виконано |
| 7 | Захист програмного продукту | 16.10-2023 – 20.10.2023 | виконано |
| 8 | Підготовка магістерської дисертації | 21.10.2023 – 17.11.2023 | виконано |
| 9 | Передзахист | 18.12.2023 – 22.12.2023 | виконано |
| 10 | Захист | 15.01.2024 – 19.01.2024 | виконано |

Студент _____ Василенко М.А.

( підпис ) (прізвище та ініціали)

Науковий керівник _____ Кузьміних В.О.

( підпис ) (прізвище та ініціали)

# РЕФЕРАТ

**Структура і обсяг кваліфікаційної роботи.** Магістерська дисертація складається зі вступу, сьоми розділів, висновків та 2 додатків. Робота містить посилання на 18 джерел, 37 рисунків та 1 таблицю. Основна частина роботи викладена на 65 сторінках.

**Актуальність.** Актуальність розробки програмного забезпечення для узагальнення профілів користувача визначається необхідністю подолання низки викликів, пов’язаних із сучасними рішеннями. Існуючі програми, що вирішують це завдання, є недосконалими. Різноманіття джерел інформації та різні формати подання даних створюють труднощі у здійсненні повного узагальнення, що може призвести до недостатньої точності та повноти отриманої інформації.

Забезпечення ефективного аналізу та узагальнення інформації з різних джерел стає ключовим завданням у зв'язку із зростаючою складністю та обсягом даних. Практична важливість полягає в забезпеченні користувачам точних та релевантних рекомендацій, а також в оптимізації різноманітних інформаційних сервісів. Актуальність проблеми узагальнення профілів користувачів підсилюється зростаючою популярністю використання різноманітних профілів користувачів в різних сервісах та платформах.

**Мета роботи і завдання дослідження.** Метою є проведення аналізу наукової діяльності організацій та окремих науковців, задля подальшого ефективного розподілу ресурсів(зокрема фінансових) компаній. Для вирішення даного питання необхідна розробка методів та підходів проведення збору інформації про наукові статті, матеріали конференцій та препрінти, які зберігаються у спеціалізованих джерелах.

**Об’єктами дослідження** є наукові твори, дисертації, тощо – які є результатами досліджень(діяльності) науковців.

**Предмет дослідження** є методи обробки та вилучення інформації з джерел.

**Методи дослідження.** Для вирішення поставлених задач і досягнення поставленої мети використовувалися методи парсингу для збору та обробки даних, які необхідні для аналізу наукової діяльності.

**Практичне значення одержаних результатів** дослідження полягає в тому, що розроблена програмна система дає користувачу інструмент необхідних даних та можливості взаємодіяти з базою даних через зручний інтерфейс користувача, зберігати звіти та сценарії пошуку для подальшого розвитку програмного забезпечення та ефективного користування.

**Ключові слова:** парсинг, інтелектуальний аналіз інформації, бібліографічні джерела інформації, збір даних, статистичний аналіз

# ABSTRACT

**Structure and scope of qualification work.** The master's thesis consists of an introduction, five chapters, conclusions and 2 appendices. The work contains references to 18 sources, 37 figures and 1 table. The main part of the work is laid out on 84 pages.

**Topicality.** The relevance of software development for summarizing user profiles is determined by the need to overcome a number of challenges associated with modern solutions. Existing programs solving this task are imperfect. The variety of information sources and different data presentation formats create difficulties in making a full generalization, which can lead to insufficient accuracy and completeness of the information obtained.

Providing effective analysis and summarization of information from various sources becomes a key task in connection with the growing complexity and volume of data. The practical importance lies in providing users with accurate and relevant recommendations, as well as in the optimization of various information services. The urgency of the problem of summarizing user profiles is reinforced by the growing popularity of the use of various user profiles in various services and platforms.

**The purpose of the work and the tasks of the research.** The goal is to conduct an analysis of the scientific activity of organizations and individual scientists, for the further effective distribution of resources (in particular, financial) companies. To solve this issue, it is necessary to develop methods and approaches for collecting information about scientific articles, conference materials, and preprints stored in specialized sources.

**The objects of research** are scientific works, dissertations, etc., which are the results of research (activity) of scientists.

**The subject of research is** methods of processing and extracting information from sources.

**Research methods.** To solve the set tasks and achieve the set goal, parsing methods were used to collect and process data, which are necessary for the analysis of scientific activity.

**The practical significance of the obtained research** results is that the developed software system provides the user with the necessary data tool and the ability to interact with the database through a convenient user interface, save reports and search scripts for further software development and effective use.

**Keywords:** parsing, intellectual analysis of information, bibliographic sources of information, data collection, statistical analysis

# ЗМІСТ

# ВСТУП

Здавна людство випробовує нові шляхи полегшення свого існування, введення щось невідомого і раніше неспробуваного. Проте не завжди ці експерименти завершувалися, як очікувалося. Так народжується потреба у фіксації результатів попередніх спроб для подальшого аналізу і виправлення помилок. Так починається документування результатів досліджень, від базових "наскальних малюнків".

Висновки залежать від спостережень, однак різні ідеї та думки людей можуть призводити до відмінних тем досліджень та різних висновків навіть щодо одного й того ж дослідження. Тому з'являється потреба у спільному обміні досвідом для швидкого розвитку.

З розвитком технологій люди шукають більш стислі та безпечні методи збереження інформації, від наскальних малюнків до паперу, що сприяло розвитку науково-технічного прогресу. Збільшення обсягу записів призводить до вдосконалення методів зберігання та розподілу інформації, а також до сортування записів для врахування актуальності та нових результатів досліджень.

Розвиток комунікаційних систем дозволяє науковцям ефективніше поширювати результати своєї роботи, що сприяє вільній обміну інформацією між дослідниками, забезпечуючи доступ до робіт у будь-якому місці та часі.

Видатні дослідники, чиї відкриття стали проривом, завжди привертають увагу. Нові науковці часто використовують їхні роботи, цитуючи для підтримки своїх висновків, доповнення або спростування результатів інших вчених та висловлюючи власні думки, теорії та рішення.

Однак, завдяки відносно вільному доступу до інформації, виникає проблема, яка може дискредитувати трудові зусилля науковців, витрачені на десятиліття, місяці чи навіть ціле життя. Деякі люди можуть використовувати чужі наукові досягнення для власної слави, матеріального збагачення або просто з лінії.

Зі зростанням обсягу інформації, збереженої людством, аналіз усіх досягнень інших науковців стає складнішим, а виявлення проблем, пов'язаних із недобросовісністю в покращенні власної роботи, стає більш важким завданням. Це призводить до питань про академічну доброчесність.

Академічна доброчесність - це сукупність морально-етичних принципів та правил, які визначаються науковою спільнотою та законом, і якими керуються всі учасники наукової спільноти та освітнього процесу.

В цю спільноту входять:

1) науковці;
2) викладачі;
3) здобувачі освіти;
4) спеціалізовані установи;
5) видавці навчальної та наукової літератури;

Порушення основних принципів академічної чесності може викликати сумніви не лише у досягненнях автора роботи, але й у репутації установи, з якою він безпосередньо пов'язаний.

Такі інциденти можуть викликати суттєві негативні наслідки і спричинити скандал. Порушення академічної чесності розглядаються як серйозні, які можуть мати непередбачувані наслідки. Прикладами таких порушень є:

1) фабрикація;
2) обман;
3) списування;
4) академічний плагіат;
5) необ’єктивне оцінювання;
6) самоплагіат;
7) надання перешкод або допомоги, які непередбачені процесом;

З появою перших електронно-обчислювальних машин людство розпочало формування нового методу зберігання та отримання інформації, який перспективно обіцяє бути більш надійним і швидким у порівнянні з

традиційними аналогами. Поступово вчені почали вдосконалювати ці технології та впроваджувати їх масово. З появою світової системи взаємопов'язаних комп'ютерних мереж, таких як Інтернет, і розширенням доступу до неї, розпочалася нова ера обміну інформацією.

Наукові журнали, видання та бібліотеки, відзначивши всі переваги зберігання інформації у електронному форматі, почали переносити накопичену людством інформацію у цей електронний формат збереження даних. Цей процес дозволяє нам зберігати та відтворювати інформацію й досі.

# 1 ПОСТАНОВКА ЗАДАЧІ ТА ОСНОВНІ ЗАВДАННЯ

Для отримання точних результатів у дослідженні важливо чітко сформулювати завдання. Розглянемо мету цієї роботи та основні задачі, пов'язані з нею.

## 1.1 Опис проблеми

Для успішного продовження досліджень науковцям необхідно провести аналіз обширної кількості інформації, яка безпосередньо пов'язана з предметом їхніх досліджень, зокрема попередніх навчань видатних науковців. Важливо виокремити з цих досліджень всю необхідну інформацію та зберегти її для подальшого цитування. Аналітикам необхідно забезпечити доступ до ресурсів, ретельно розробивши графік, який дозволить якісно та з урахуванням усіх можливих перешкод виконати поставлені завдання щодо збору та обробки необхідної інформації.

Процес аналізу обсягу інформації аналітиками може розрізнятися в залежності від кількості матеріалів для аналізу, часу отримання доступу до інформації та компетентності аналітика у обраній сфері.

Важливо також пам'ятати, що надто поспішний аналіз, вимагаючи швидкого результату, може призвести до отримання неточної, хибної або непотрібної інформації, збільшуючи тим самим затримки у виконанні графіку або навіть може призвести до провалу дослідження.

Платформа для збору інформації повинна включати широкий спектр робіт у відповідній галузі науки, які будуть використані у майбутніх дослідженнях. Крім того, вона повинна зберігати базову інформацію про статті, авторів, роки публікацій та інші джерела інформації, використовувані при написанні цих статей.

Зокрема, платформа повинна вміщувати нові, ще не визнані науковою спільнотою роботи, з метою знаходження нових, перспективних ідей та підходів

для вирішення існуючих завдань. Цілеспрямованість цих робіт може визначатися як параметр, за допомогою якого можна визначити їхню актуальність серед інших дослідників та аналітиків.

Також важливо враховувати, що ця інформація може бути потрібна одночасно багатьом аналітикам, які можуть перебувати в різних місцях, де фактично зберігається ця інформація. І саме для вирішення цієї проблеми були створені бібліографічні бази даних, в яких вже зібрана значна кількість робіт та базова інформація про них. Це надає можливість аналітикам отримувати доступ до необхідної інформації незалежно від їхнього місцезнаходження, сприяючи ефективнішій роботі та спільному використанню ресурсів.

## 1.2 Джерела даних та їх проблематика.

Прикладом джерел є бібліографічні бази даних. Бібліографічні бази даних представляють собою систему, в якій зберігається значна кількість інформації, сприяючи обміну досягненнями між науковцями з усього світу в галузі науки. Зазвичай ці бази даних організовані у вигляді каталогу, що містить посилання на наукові роботи, які позначені для полегшення пошуку конкретних праць серед інших.

Отримання доступу до інформації, яка міститься в цих бібліографічних базах даних, можливе через міжнародну комп'ютерну мережу Інтернет, і зараз ці бази активно використовуються.

Бібліографічні дані широко використовуються в аналізі діяльності різних об'єднань членів наукової спільноти. Збираючи інформацію з бібліографічних джерел, аналітики можуть оцінити результати досліджень конкретних об'єднань членів наукової спільноти для їх подальшого порівняння з роботами інших вчених.

Отримуючи дані про дослідження цієї групи людей, аналітики можуть оцінювати перспективи об'єднань в певній галузі досліджень. Це дозволяє ефективно розподіляти бюджет організації та залучати зацікавлених спонсорів.

Бібліографічні бази даних дозволяють користувачам знаходити статті в будь-якому місці та практично у будь-який час. Інформацію зберігають на комп'ютерах з великими обчислювальними та дисковими ресурсами, зокрема, на серверах, які забезпечують одночасну роботу з базою даних для багатьох користувачів.

Пошук зазвичай здійснюється за попередньо визначеними сценаріями через фільтри пошуку. Якість класифікації інформації в базі даних та ефективність реалізованих фільтрів безпосередньо впливають на швидкість та успішність пошуку. Однак існують різні підходи до реалізації системи, включаючи питання збереження даних та параметрів пошуку, що може створювати невизначеність щодо необхідної інформації. Тому аналітикам важливо проводити аналіз діяльності на кількох ресурсах одночасно та консолідувати результати для узагальнення.

## Висновки до розділу 1

Успішне продовження наукових досліджень вимагає комплексного аналізу обширної інформації, пов'язаної з предметом досліджень. Важливо ретельно виокремити та зберегти необхідну інформацію для подальшого використання. Аналітикам слід ефективно керувати часом та ресурсами, уникати спішки у процесі аналізу, оскільки це може призвести до неточних результатів.

Створення платформи для збору інформації є ключовим етапом, що дозволяє забезпечити доступ до різноманітних ресурсів та зберігати базову інформацію для подальших досліджень. Бібліографічні бази даних грають важливу роль у забезпеченні доступу до наукових робіт та сприяють ефективній роботі аналітиків.

Джерела даних, такі як бібліографічні бази, дозволяють оцінювати результати досліджень, роблять можливим порівняння об'єднань наукової спільноти та сприяють розподілу бюджету та залученню спонсорів. Однак важливо враховувати проблематику якості класифікації та ефективності фільтрів для швидкого та успішного пошуку інформації.

# 2 АНАЛІЗ АКТУАЛЬНОСТІ ЗАДАЧІ ТА ОГЛЯД ІСНУЮЧИХ СИСТЕМ

Система призначена для використання аналітиками наукової діяльності, завдяки яким фінансуючі фірми можуть більш доцільно розподіляти бюджет на фінансування саме тих компаній, які мають більшу перспективу.

Завдяки автоматизованому збору необхідної інформації за обраною тематикою діяльність аналітиків стає більш продуктивною та з’являється можливість проаналізувати більшу кількість інформації.

На даний час не є доступні системи, які б охоплювали велику кількість джерел специфічних джерел та об`єднувати їх в загальній базі даних.

## 2.1 Дослідження інформаційно-пошукових систем

Інформаційно-пошукові системи (ІПС) є автоматизованими системами, призначеними для збирання, пошуку, оброблення, збереження та надання інформації за допомогою технічних засобів. Ці системи включають в себе інформаційні масиви, такі як бази і банки даних, інтернет-системи, носії інформації (магнітні, оптичні тощо), а також програмне та технічне забезпечення.

Інформаційно-пошукові системи спрямовані на впорядкування інформаційних потоків та полегшення доступу до різних джерел інформації. Термін "інформаційний пошук" почав використовуватися ще з 1950-х років. Спочатку ці системи використовувалися для організації інформації в архівах, картотеках, бібліотеках державних установ, наукових і навчальних закладах та інших місцях.

З розвитком комп'ютеризації суспільства і зокрема зі створенням Інтернету, інформаційно-пошукові системи значно розширили свою сферу застосування, набуваючи великого поширення та важливості у сучасному інформаційному середовищі.

### 2.1.1 ArnetMiner (AMiner)

ArnetMiner (AMiner) призначений для проведення пошуку та виконання операцій інтелектуального аналізу даних стосовно наукових публікацій в Інтернеті. Використовуючи аналіз соціальних мереж, система виявляє зв'язки між дослідниками, конференціями та публікаціями. Це надає можливість виконання різноманітних послуг, таких як пошук експертів, географічний пошук, аналіз тенденцій, рекомендації рецензентів, пошук асоціацій, пошук курсів, оцінка успішності та моделювання тем.

ArnetMiner почав свій шлях як дослідницький проект у галузі аналізу соціального впливу, ранжирування в соціальних мережах та вилучення даних з соціальних мереж. Результати розробки системи були опубліковані в ряді рецензованих статей. Протягом трьох років її існування ArnetMiner індексував понад 130 мільйонів дослідників та більше 200 мільйонів публікацій. Проект отримав фінансування від Китайської національної програми досліджень та розробок у галузі високих технологій та фонду.

ArnetMiner широко використовується в академічних колах для визначення взаємозв'язків та статистичних кореляцій між дослідженнями та дослідниками. Система залучила понад 10 мільйонів незалежних IP-доступів із 220 країн та регіонів. Продукт використовувався на платформі SciVerse Elsevier та в рамках наукових конференцій, таких як SIGKDD, ICDM, PKDD, WSDM.

ArnetMiner автоматично здійснює екстракцію профілів дослідників з Інтернету, ідентифікує відповідні сторінки та використовує єдиний метод для отримання даних із визначених документів. Крім того, система отримує публікації з електронних бібліотек за допомогою евристичних правил.

ArnetMiner об'єднує отримані профілі дослідників та екстраговані публікації, використовуючи ім'я дослідника як ідентифікатор. Для вирішення проблеми неоднозначності імені при інтеграції, запропоновано використовувати ймовірнісну

структуру. Інтегровані дані зберігаються у базі знань дослідницької мережі (RNKB).

Іншими ключовими продуктами у цій галузі є Google Scholar, Scirus від Elsevier та відкритий проект CiteSeer.

### 2.1.2 CiteSeer

CiteSeer замінив CiteSeer і всі запити CiteSeer автоматично перенаправляються. CiteSeer є відкритою пошуковою системою та цифровою бібліотекою, фокусуючись на наукових та академічних статтях, зокрема у галузі комп'ютерних наук та інформатики. Останнім часом CiteSeer розширює свою діяльність на інші галузі, такі як економіка та фізика. Запущений у 2008 році, він базується на попередній версії CiteSeer та використовує нову інфраструктуру з відкритим вихідним кодом, SeerSuite, а також нові алгоритми та їх реалізації. Розроблений дослідниками, такими як доктор Ісаак Кунсілл і доктор К. Лі Джайлс з Коледжу інформаційних наук та технологій, Державний університет Пенсільванії, проект підтримує цілі, визначені CiteSeer.

Система активно сканує та збирає наукові документи з відкритих веб-сайтів, використовуючи запити цитувань для ранжування та визначення впливу цитування. Зараз проект управляється командою, до якої входять Лі Джайлс, Прасенджит Мітра, Сьюзан Гауч, Мін-Єн Кан, Праділ Тереговда та інші. Можливість пошуку за таблицею є нещодавнім доповненням. Фінансування надходить від Національного наукового фонду, НАСА та Microsoft Research.

CiteSeer залишається одним із найкращих світових репозиторіїв і утримує лідерство в рейтингах. На даний момент в ньому зберігається понад 6 мільйонів документів від майже 6 мільйонів унікальних авторів, з 120 мільйонами посилань.

CiteSeer активно ділиться своїм програмним забезпеченням, даними, базами даних та метаданими з іншими дослідниками, використовуючи Amazon S3 та rsync. Його нова модульна архітектура з відкритим вихідним кодом побудована на Apache

Solr та інших інструментах з відкритим вихідним кодом, що дозволяє тестувати нові алгоритми збору, ранжування, індексації та вилучення інформації.

Також важливо відзначити, що CiteSeer кешує певні відскановані PDF-файли, і на кожній сторінці присутнє посилання DMCA для повідомлення про можливі порушення авторських прав.

### 2.1.3. WorldWideScience

WorldWideScience.org є глобальним науковим порталом, який надає єдиний пошук у понад 100 наукових і технічних базах даних із понад 70 країн. Цей портал заснований на стратегічному партнерстві, відомому як WorldWideScience Alliance, яке об'єднує національні та міжнародні бібліотеки та центри даних. Операційні функції Альянсу виконує OSTI (Office of Scientific and Technical Information), і члени Альянсу спрямовані на усунення перешкод у пошуку та обміні науковою та технічною інформацією.

Використовуючи технологію федеративного пошуку, WorldWideScience.org надає користувачам можливість проводити одночасний пошук у режимі реального часу в різних базах даних. Результати представлені у вигляді консолідованого списку, впорядкованого за релевантністю, і включають інформацію у текстовому, мультимедійному та науковому форматах. Підтримка багатомовних перекладів (десять мов) робить науковий матеріал більш доступним для глобальної аудиторії.

WorldWideScience.org спрощує пошук даних досліджень, особливо для користувачів, які не знайомі з конкретними центрами обробки даних або не впевнені в наявності певних наборів даних. Результати пошуку включають вкладку "Відкритий доступ", яка дозволяє безперешкодно переходити до безкоштовних журнальних статей та рецензованих рукописів.

WorldWideScience.org також враховує рух громадського доступу, включаючи портали загального доступу від федеральних наукових агенцій США та міжнародних партнерів. Включення дослідницьких даних і порталів загального

доступу допомагає розширити доступ до результатів досліджень і підтримує наукову співпрацю та прогрес. WorldWideScience.org значно підвищує видимість та використання результатів досліджень і розробок, сприяючи їхньому глобальному впровадженню.

## 2.2 Дослідження джерел наукової інформації

В даний час існує значна кількість рішень, спеціально створених для пошуку бібліографічної інформації, які використовуються науковцями з усього світу для зберігання своїх статей та досліджень. Деякі з найбільш відомих рішень на сьогодні включають:

1) Google Scholar: Загально відомий інструмент від Google, який надає доступ до різних наукових публікацій, включаючи журнали, конференції та патенти.
2) ArXiv (Cornell University): Репозиторій для наукових статей, особливо популярний в галузі фізики, математики та комп'ютерних наук.
3) Scopus: Бібліографічна база даних, яка охоплює різні наукові галузі, включаючи природничі, технічні та медичні науки.
4) Wolfram Alpha: Комп'ютерний пошуковик, який надає не лише бібліографічні дані, а й можливість аналізу та обчислень.
5) ScienceDirect: Платформа Elsevier, яка надає доступ до тисяч наукових журналів та книг.
6) Web of Science: Інструмент, який об'єднує багато різних наукових баз даних і надає можливість виконувати комплексний пошук.

Кожна з цих бібліографічних баз має свою структуру та покриття у різних галузях. Для аналізу можна обрати дві діаметрально протилежні, але широко використовувані бази даних, які представляють різні галузі наук та зведені до єдиної системи для отримання більш повного огляду наукових джерел.

### 2.2.1 Google Scholar

Google Академія є наукометричною базою даних з відкритим доступом, розробленою вченим індійського походження Анурагом Ачарья (Anurag Acharya) за допомогою пошукової системи Google. Створення цієї бази мало на меті допомагати аналітичній та академічній спільноті.

Цей інструмент дозволяє аналітикам проводити пошук серед широкого спектру наукової літератури, такої як:

1) тези;
2) рецензовані статті;
3) книги, дисертації;
4) презентації;
5) технічні звіти;
6) наукові журнали;
7) препринти;
8) реферати;

На даний момент Google Академія має найбільшу кількість наукової літератури, охоплюючи найбільшу кількість галузей досліджень. У 2016 році її база охоплювала 160 мільйонів документів з унікальним значенням, що в декілька разів більше, ніж у конкуруючих базах даних Scopus та Web of Science.

Google Академія включає в себе як відомості про документи з бібліографією або рефератами, так і повнотекстові матеріали, якщо вони доступні для вільного читання. Сервіс також надає додаткові дані, такі як індекс цитування документів, посилання на авторів статей та зовнішні посилання на матеріали, які використовувались у статті.

На основі вільно доступних профілів дослідників у Google Академії регулярно складаються рейтинги наукових груп за критерієм цитування науковців, які належать до наукових установ чи інших груп.

### 2.2.2 ArXiv

ArXiv це база даних, яка була створена на основі бібліотеки Корнелльського університету і вміщує значну кількість електронних публікацій, включаючи наукові статті, їхні препринти, реферати та дисертації.

Архів виник за ініціативою американського астрофізика Джона Кона. До серпня 1991 року Джон отримував наукові статті від 180 науковців на свою особисту поштову скриньку, основною тематикою яких була фізика. Це послужило відомчому запиту створити архів наукових статей, які б могли бути класифіковані за науковими спеціалізаціями.

Архів є повністю відкритим для публікації статей для будь-якого автора, і видавцям не доводиться турбуватися про питання логістики, а також немає обмежень на використання тексту та відсутність питань щодо плагіату. Також не має сумнівів у відомостях видавця серед наукової громади, що дає можливість будь-кому опублікувати свої роботи для огляду та оцінки іншими вченими з метою отримання їхнього відгуку на статтю.

На 2018 рік Архів нараховував близько двох мільйонів статей у різних категоріях. В Архіві можна знайти роботи у напрямках:

1) фізика;
2) математика;
3) комп’ютерні науки;
4) кількісна біологія;
5) кількісні фінанси;
6) статистика;
7) електротехніки;
8) економіки.

Незважаючи на доступність платформи, Архів визнаний більш надійним порівняно з багатьма науковими журналами, оскільки відбір статей для публікації у ньому ґрунтується на активній перевірці відповідності їхнього змісту обраній

категорії науковою спільнотою. Якщо публікація не відповідає вибраній категорії або не представляє значущого внеску для наукової громади, то її категорію переглядають із наступним відповідним визначенням або вона класифікується як "сміття".

### 2.2.3 Scopus

Scopus є базою даних, яка включає в себе обширні анотації та інформацію про цитування в науковій літературі та має системи для відстеження дій користувачів з метою подальшого аналізу популярності статей та візуалізації статистики.

База даних включає статті з більш ніж 23 тисяч міжнародних видань та понад п'ять тисяч видавців, які мають авторитет у всьому світі. Вона отримала визнання серед міжнародних університетів за якість зібраних даних та високу оцінку в науковій спільноті.

Scopus охоплює різні галузі, такі як природні науки, гуманітарні та громадські науки, медицина, мистецтва та техніка. Заснована в 2004 році компанією Elsevier, яка існує з 1880 року у Амстердамі. Ця база даних широко використовується рейтинговими агенціями для складання світових рейтингів університетів.

Для публікації на Scopus допускаються лише перевірені серед наукової спільноти видавці, а користувач повинен бути учасником наукової спільноти та пройти реєстрацію на платформі. Для доступу до інформації на цій платформі необхідно перебувати в науковій установі, яка має угоду з компанією Elsevier.

### 2.2.4 Wolfram Alpha

Wolfram Alpha - це об'єднана пошукова система, також відома як "Вольфрам Альфа", яка є "обчислювальною машиною знань". Вона поєднує в собі елементи

пошуку, обробники для результатів пошуку користувачем та алгоритми, що базуються на обробці природної мови (англійської). Wolfram Alpha є прямим конкурентом популярної пошукової системи Google.

Цей ресурс не пропонує користувачеві список посилань на літературу як результати запиту, оскільки він повністю спирається на свою власну базу даних, що складається з обширного обсягу знань у таких галузях, як:

1) фізика;
2) астрономія;
3) хімія;
4) медицина;
5) математика;
6) кінематограф;
7) музика;
8) політика;
9) географія;
10) історія;

Сервери пошуку Wolfram налічують близько 10 тисяч процесорів і забезпечують постійне обслуговування, щоб надати користувачам системи безперебійний доступ до об'єднаної бази знань. Користувачі, які бажають збирати ексклюзивні дані для подальшого аналізу, можуть оформити підписку на платній основі, що включає доступ до хмарного сховища та статистики пошукових запитів. Підписка також надає рекомендації для подальших наукових досліджень на основі аналізу користувацьких запитань.

### 2.2.5 ScienceDirect

ScienceDirect - це джерело медичної та науково-технічної інформації, яке містить близько 14 мільйонів публікацій з авторитетних наукових журналів та книг, видавництва Elsevier. Відоме своєю важливою роллю в дослідженнях у різних

областях, студенти, викладачі та аналітики використовують ScienceDirect для доступу до фундаментальних трудів, написаних видатними вченими, починаючи з 1823 року

На платформі ScienceDirect є багато інструментів для зручного пошуку та роботи з інформацією, таких як зручний інтерфейс, віддалений доступ до пошуку, можливість працювати на мобільних пристроях, рекомендації щодо пов'язаних статей, спеціальний пошук за зображеннями або відео, а також візуалізація даних про статті у режимі реального часу.

Платформа ScienceDirect доступна користувачам за підпискою на платній основі, що надає можливість отримувати доступ до широкого спектру наукової інформації.

### 2.2.6 Web of Science

Платформа Web of Science є власністю компанії Clarivate Analytics та є наслідком розвитку індексів наукового цитування, які були запропоновані Юджином Гарфілдом в 50-х роках минулого сторіччя. З 1964 року інститут для наукової інформації (Institute for Scientific Information) розпочав створення трьох спеціалізованих індексів, а з 1997 року ці індекси об'єдналися на веб-платформі, отримавши назву Web of Science.

На платформі Web of Science розташовано 15 баз даних, що створюються як компанією Clarivate Analytics, так і її партнерами. Загалом платформа індексує більше 33 тисяч видань з 1864 року, охоплюючи всі дисципліни.

Основною частиною колекції є наукометрична база даних Web of Science Core Collection (WoSCC). Ця база включає індекси наукового цитування періодичних видань у галузі природничих і технічних наук (Science Citation Index Expanded - SCIE), суспільних наук (Social Sciences Citation Index - SSCI) та гуманітарних наук (Art and Humanities Citation Index - AHCI). З 2015 року до

WoSCC також було додано мультидисциплінарний індекс Emerging Source Citation Index (ESCI), а архів розширено до 2005 року.

Крім того, WoSCC включає індекси цитувань кращих конференцій (Conference Proceedings Citation Index - CPCI), монографії (Book Citation Index - BkCI), а також два хімічні індекси: Current Chemical Reactions (CCR-EXPANDED) та Index Chemicus (IC).

## Висновки до розділу 2

У висновку слід відзначити, що розроблена система для аналітиків наукової діяльності вирішує важливу проблему фінансового розподілу бюджету, дозволяючи фінансуючим фірмам ефективніше визначати перспективні компанії для інвестування. Застосування автоматизованого збору інформації підвищує продуктивність аналітичної діяльності, дозволяючи обробляти більший обсяг даних. Однак слід відзначити, що на даний момент немає доступних систем, які охоплюють велику кількість специфічних джерел та об'єднують їх в єдиній базі даних, що може слугувати напрямком для подальших розвитків.

# 3 АНАЛІЗ МЕТОДІВ ВИРІШЕННЯ ПОСТАВЛЕННОЇ ЗАДАЧІ

## 3.1 Дослідження структури веб-сайтів

Для автоматичного збору бібліографічної інформації зі спеціалізованих веб-сайтів необхідно мати розуміння загальної структури таких ресурсів у всесвітній мережі та базові знання щодо Інтернет-протоколів. Структура сайту включає схему розташування його сторінок, категорій, підкатегорій та інших об'єктів. Іншими словами, це план, за яким встановлюється логічний зв'язок між різними частинами сайту.

Отже, для ефективного збору бібліографічної інформації необхідно розуміти структуру конкретного сайту та вибрати відповідний метод доступу, враховуючи його типову організацію.

Виділяють три основні типи структур сайтів: лінійний, довільний та ієрархічний. Лінійна структура передбачає послідовний перехід від однієї сторінки до іншої. Довільна структура характеризується вільним доступом до будь-якої сторінки з будь-якої іншої. Ієрархічна структура включає в себе систему категорій та підкатегорій, що дозволяє організувати інформацію в логічному порядку.

Більшість складних веб-сайтів складаються з різних компонентів, включаючи:

- Розмітку (HTML): Мова розмітки HTML відповідає за визначення структури даних та об'єктів на веб-сайті. Вона вказує розташування блоків, шрифти, кольори блоків та тексту, а також посилання.
- Каскадну таблиця стилів (CSS): CSS відповідає за оформлення та реагування веб-сайту на дії користувача. Вона використовується для створення зовнішнього вигляду та стилю веб-сайту.

- Скрипти користувача (JavaScript): JavaScript використовується для реалізації скриптів, які виконуються на стороні користувача під час взаємодії з веб-сайтом.
- Скрипти на сервері (наприклад, PHP): Серверні мови програмування, такі як PHP, використовуються для взаємодії між сервером і веб-сайтом, передачі даних з серверу на веб-сайт і виконання процедур.
- Базу даних: База даних організована для зберігання даних відповідно до певної концепції. Вона використовується для структурування та зберігання інформації, необхідної для правильної роботи веб-сайту та для зберігання даних користувачів. Дані можуть зберігатися на різних серверах для оптимізації навантаження.

### 3.1.1 Веб-сайт з лінійною структурою

Лінійна структура використовується, коли веб-сайт надає користувачеві інформацію за чіткою послідовністю. Зазвичай, перегляд інформації на таких веб-сайтах відбувається від початкової до останньої сторінки, і кожна сторінка містить лише посилання на попередню та наступну сторінку.

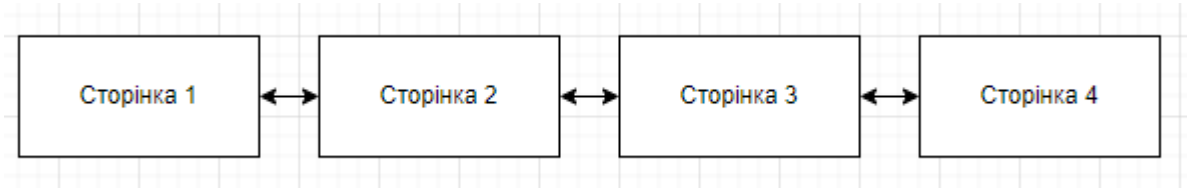



Рисунок. 3.1 — Приклад лінійної структури веб-сайту

Лінійну структуру веб-сайту можна проілюструвати за допомогою приблизної схеми, зображеної на рисунку 3.1.

### 3.1.2 Веб-сайт з довільною структурою

Вільна структура використовується для сайтів, де відсутні будь-які обмеження в розміщенні посилань в середині веб-сайту.

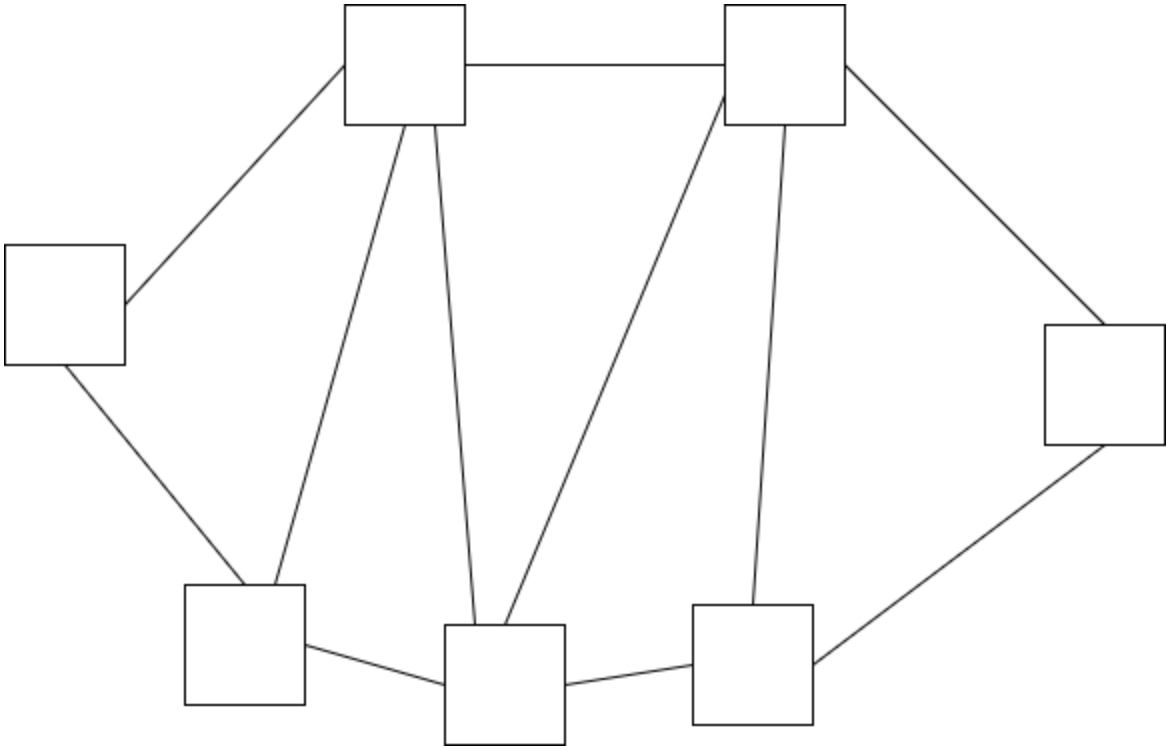


Рисунок 3.2 — Приклад довільної структури веб-сайту

Часом окремі частини цих веб-сайтів можуть містити елементи як з лінійною, так і з ієрархічною структурою. Приблизна схема веб-сайту з вільною структурою зображена на рисунку 3.2.

### 3.1.3 Веб-сайт з ієрархічною структурою

Використання ієрархічної структури вважається ефективним при використанні сторінок з різним рівнем доступу. Кожна сторінка на більш високому рівні має посилання на сторінку або сторінки на меншому рівні.

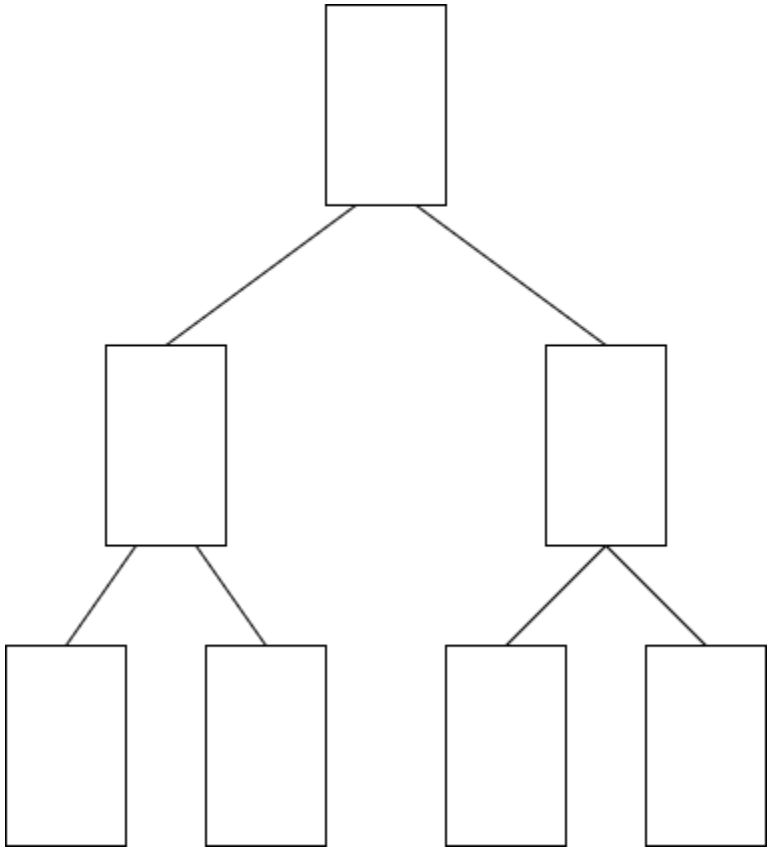


Рисунок 3.3 — Приклад Ієрархічної структури веб-сайту

Такий тип структури часто використовується в каталогах і бібліографічних базах даних. Приблизна схема веб-сайту з ієрархічною структурою представлена на рисунку 3.3.

## 3.2 Дослідження методів парсингу

Парсинг - це процес аналізу та видобування інформації зі структурованого джерела даних, такого як веб-сайт, текстовий файл чи база даних. Існує кілька методів парсингу, які можуть використовуватися для отримання даних з різних джерел:

1) HTML-парсинг;
2) API-парсинг;

3) парсинг регулярними виразами;
4) синтаксичний аналіз (Syntax Parsing);
5) веб-скрапінг;
6) оптичне розпізнавання символів (OCR - Optical Character Recognition);
7) машинним навчанням;

Важливо враховувати етичні та юридичні аспекти при використанні парсингу, оскільки незаконний або недозволений доступ до даних може призвести до порушення правил і порушень.

### 3.2.1 HTML-парсинг

HTML-парсинг - це процес аналізу HTML-коду з метою отримання інформації з веб-сторінки. Це корисна дія, коли вам потрібно автоматизувати збір даних із веб-сайту або обробити велику кількість інформації.

Процес HTML-парсингу зазвичай включає наступні кроки:

1) завантаження HTML-коду;
2) використання інструменту для парсингу ;
3) навігацію по елементам;
4) вилучення необхідних даних;

### 3.2.2. API-парсинг

API-парсинг, або робота із зовнішніми API (Application Programming Interface), є процесом взаємодії з веб-службами для отримання даних. На відміну від HTML-парсингу, де ви аналізуєте HTML-код веб-сторінки, під час роботи з API дані зазвичай надаються у структурованому форматі, такому як JSON або XML. Процес API-парсингу зазвичай включає наступні кроки:

1) отримання API-ключа (якщо необхідно);
2) використання бібліотеки або створення HTML запиту;

3) відправлення запросу до API;
4) обробка відповіді від API;
5) вилучення даних;

### 3.2.3 Парсинг регулярними виразами

Парсинг з використанням регулярних виразів (regex) - це метод обробки текстових даних, що ґрунтується на шаблонах, заданих з використанням регулярних виразів. Регулярні вирази – це послідовності символів, що визначають шаблон пошуку., заданих з використанням регулярних виразів. Регулярні вирази – це послідовності символів, що визначають шаблон пошуку.

Даний вид парсингу зазвичай включає наступні кроки:

1) імпорт необхідних бібліотек;
2) використання метасимволів та комбінацій;
3) вилучення груп;
4) обробка та заміна співпадінь

### 3.2.4 Парсинг методом синтаксичного аналізу

Парсинг методом синтаксичного аналізу буває двох видів: верхнього рівня (top-down) та нижнього рівня (bottom-up). Кожен із цих методів має свої підходи, алгоритми та інструменти.

Верхній рівень (Top-down) парсинг має 2 види аналізаторів:

1) Рекурсивний спуск (Recursive Descent): Це метод, у якому кожен нетермінал граматики зіставляється з функцією, що викликає інші функції обробки підлеглих символів. У мовах програмування, що підтримують рекурсію, це може бути природним та елегантним способом реалізації.
2) LL-аналізатори (Left-to-Right, Leftmost derivation): Це сімейство алгоритмів, що використовуються для верхнього рівня парсингу, де вхідний потік

символів обробляється ліворуч, а виводиться найбільш лівостороннє дерево розбору.

Нижній рівень (Bottom-up) парсинг має 2 види аналізаторів:

1) LR-аналізатори (Left-to-Right, Rightmost derivation): Ці алгоритми працюють праворуч наліво і будують дерево розбору від листя до кореня.
2) GLR-аналізатори (Generalized LR): Це узагальнені версії LR-аналізаторів, які можуть опрацьовувати граматики з неоднозначностями.

### 3.2.5 Веб-скрапінг

Веб-скрапінг (або веб-парсинг) – це процес автоматизованого вилучення даних із веб-сторінок. Він може бути використаний для різних цілей, таких як збирання інформації для аналізу, моніторинг цін на товари, оновлення даних на веб-сайтах та багато іншого. Ось детальніший опис кроків веб-скрапінгу:

1) відправка HTML-запросу;
2) обробка HTML коду;
3) вилучення даних;
4) обробка даних;
5) обробка динамічного контексту(за необхідністю);
6) робота з кінцевими даними;

### 3.2.6 Оптичне розпізнавання символів

Оптичне розпізнавання символів (OCR) – це технологія, яка дозволяє комп'ютеру читати текст із зображень або відсканованих документів. Це важливий компонент у сфері комп'ютерного зору та обробки зображень. Процес OCR включає кілька етапів:

Захоплення зображення: вихідним етапом є отримання зображення, що містить текст. Це може бути зроблено за допомогою сканера, фотокамери або інших пристроїв.

1) Попередня обробка: зображення можуть мати шум, розмиття або інші артефакти, які можуть вплинути на точність розпізнавання. Тому перед застосуванням OCR проводиться попередня обробка зображення, що включає корекцію контрасту, зменшення шуму, бінаризацію та інші техніки.
2) Сегментація зображення: у цьому кроці зображення розбивається окремі області, які мають символи чи слова. Це дозволяє алгоритму OCR точніше визначити межі кожного символу.
3) OCR-розпізнавання: на цьому етапі використовуються алгоритми машинного навчання та обробки зображень для розпізнавання символів у кожному сегменті. Існує безліч методів, включаючи традиційні методи, такі як метод опорних векторів (SVM), і сучасні глибокі нейронні мережі, такі як рекурентні нейронні мережі (RNN) або нейронні згорткові мережі (CNN).
4) Постобробка: результати OCR можуть вимагати постобробки, такої як виправлення друкарських помилок, вирівнювання тексту або обробка форматування.

### 3.2.7 Парсинг машинним навчанням

Парсинг машинним навчанням - це процес автоматизованого видобутку корисної інформації з невструктурованих даних за допомогою методів машинного навчання. Він використовується для аналізу тексту, зображень, аудіо та інших видів невструктурованих даних для отримання певного смислу чи структури.

Наприклад, у веб-скрапінгу (видобутку даних з веб-сайтів) парсинг машинним навчанням може використовувати алгоритми навчання, які розпізнають та витягають конкретні елементи сторінки, такі як заголовки, текст,

зображення або гіперпосилання. Основні кроки парсингу машинним навчанням включають:

1) Вибір даних: Визначення типу даних, які потрібно видобути.
2) Навчання моделі: Використання методів машинного навчання для створення моделі, яка може розпізнавати та видобувати ці дані. Можливі методи включають навчання з учителем або без учителя.
3) Тестування та налаштування: Перевірка ефективності моделі на тестових даних і вносити корективи для поліпшення її точності та продуктивності.
4) Використання моделі: Застосування навченої моделі для реального парсингу даних.

Парсинг машинним навчанням є потужним інструментом для автоматизації процесу аналізу великої кількості невструктурованих даних та отримання з них цінної інформації.

## 3.3 Дослідження методу зберігання даних

Існує безліч методів зберігання даних, і вибір конкретного методу залежить від великої кількості факторів, таких як обсяг даних, швидкість доступу, вартість, безпека та інші. Ось декілька загальних методів зберігання даних:

Жорсткі диски (Hard Disk Drives - HDD та Solid State Drives - SSD). HDD-використовуються для довгострокового зберігання великих обсягів даних. SSD-забезпечують швидкий доступ до даних, часто використовуються для додатків, яким потрібен високий рівень продуктивності.

Хмарне зберігання - дані зберігаються на серверах в інтернеті, а не на локальних пристроях. Забезпечує легкий доступ до даних з будь-якого місця та пристрою.

Оптичні носії. Використовуються CD, DVD, Blu-ray диски для зберігання обмеженої кількості даних. Зазвичай використовуються для архівування та довгострокового зберігання.

Резервне копіювання (Backup)- зберігання копій даних на інших пристроях або в інших локаціях для відновлення в разі втрати чи пошкодження основних даних.

Бази даних- використовуються для структурованого зберігання та організації даних, забезпечують можливість швидкого пошуку та звертання до інформації.

Флеш-накопичувачі- USB-флешки та інші флеш-пристрої для переносу та тимчасового зберігання даних.

Інтернет-сервіси- сервіси зберігання даних в Інтернеті, такі як Google Drive, Dropbox, OneDrive тощо.

Системи керування версіями (Version Control Systems): використовуються для зберігання та відстеження різних версій файлів та коду.

Хмарні сервіси зберігання даних: сховища в Інтернеті, які надають доступ до обчислювальних ресурсів та зберігання даних.

Холодне зберігання (Cold Storage)- зберігання даних на менш дорогих, але менш швидких пристроях для довгострокового архівування.

Сервери для обробки даних в реальному часі - використовуються для обчислень та зберігання даних, які вимагають негайної обробки та відповіді.

При виборі методу зберігання даних важливо враховувати потреби та вимоги конкретного проекту чи організації.

### 3.3.1 Локальна БД

Створення локальної бази даних є однією з найпоширеніших практик для зберігання подібної інформації, яка включає елементи з різними типами даних. Ця база даних представляє собою фіксовану модель, в рамках якої всі необхідні дані зберігаються у визначеному форматі. У випадку, коли інформація зберігається в одній локальній загальній базі даних, рекомендується використовувати реляційну модель. Реляційна модель бази даних використовує таблиці з рядками та стовпцями, де рядки представляють записи, а стовпці – поля, що містять дані.

Основні характеристики такої таблиці включають:

- відсутність повторюваних груп
- кожний елемент таблиці – один елемент даних
- кожний стовпець має унікальне ім’я
- у таблиці не має двох або більше однакових рядків

Сама база даних може зберігати інформацію, але для взаємодії з нею необхідно використовувати систему управління базами даних (СУБД).

### 3.3.2 Система управління базами даних

Для взаємодії програми з інформацією, збереженою у базі даних, широко використовується технологія ORM (Object Relational Mapping - об'єктно-реляційне відображення). Ця технологія в програмуванні призначена для створення віртуальної бази даних, використовуючи об'єктно-орієнтовані можливості мов програмування.

Об'єктно-орієнтований підхід розроблено з урахуванням основних принципів програмної інженерії, таких як зв'язок, агрегування та інкапсуляція, тоді як реляційна база даних ґрунтується на математичних теоріях. Щоб зробити їх взаємодію зручною, виникла технологія об'єктно-реляційного відображення.

ORM дозволяє розробникам використовувати свою улюблену мову програмування для взаємодії з базою даних без необхідності ручного написання SQL-операторів або запитів. Використання цієї технології надає гнучкий метод програмування взаємозв'язку програми та бази даних, оскільки вона абстрагує систему БД. Таким чином, розробник може змінювати базу даних у будь-який момент, а модель бази даних слабко пов'язана з іншою частиною програми, що дозволяє використовувати її в інших проектах.

Використання технології об'єктно-реляційного відображення ефективно зекономлює час розробника, оскільки:

- Розробник створює модель даних лише в одному місці, що спрощує оновлення, підтримку та повторне використання коду.

- Багато операцій виконуються автоматично, від обробки бази даних до взаємодії з елементами в ній.
- Не потрібно формувати складні SQL-запити.
- Виклик попередньо підготовлених операцій або транзакцій стає таким же простим, як і виклик звичайних функцій.

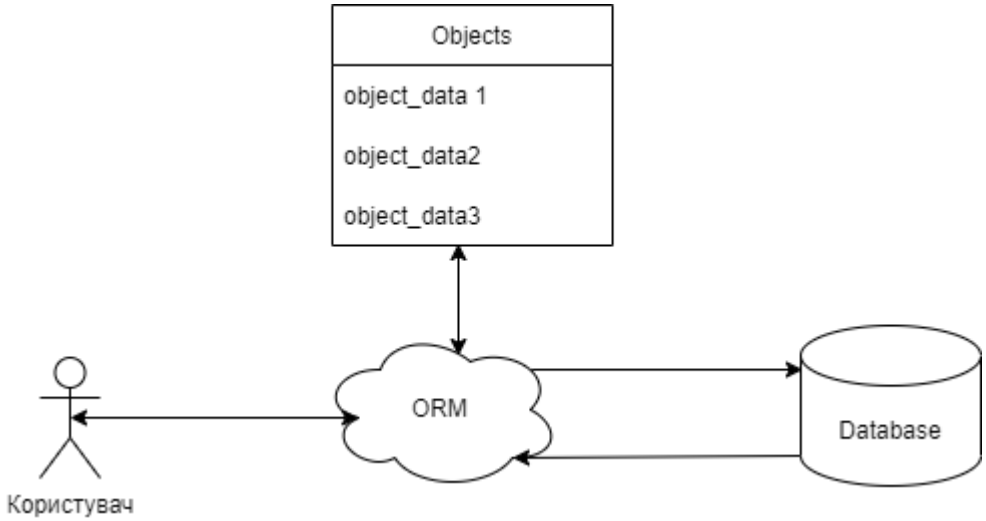



Рисунок 3.4 — Схема взаємодії користувача та бази даних через ORM

Недоліками використання даної технології є необхідність глибокого вивчення документації до цих інструментів і можливе зниження продуктивності програми при роботі з складними запитами через ці інструменти. Схема взаємодії між ORM та базою даних показана на рисунку 3.4.

## Висновки до розділу 3

Для автоматичного збору бібліографічної інформації зі спеціалізованих веб-сайтів важливо мати розуміння загальної структури таких ресурсів та базові знання щодо Інтернет-протоколів. Структура сайту включає схему розташування його

сторінок, категорій, підкатегорій та інших об'єктів, що формує логічний зв'язок між різними частинами сайту.

Для ефективного збору інформації важливо розуміти конкретну структуру сайту та вибирати відповідний метод доступу, враховуючи типову організацію сайту.

Парсинг є різноманітним інструментом для отримання інформації з різних джерел даних. Різні методи парсингу, такі як HTML-парсинг, API-парсинг, регулярні вирази та інші, надають можливість ефективно аналізувати та витягувати дані зі структурованих джерел. Важливо дотримуватися етичних та юридичних норм при використанні парсингу, оскільки нелегальний доступ до даних може призвести до неправомірних дій і порушень законодавства. Враховуючи різноманітність методів та потенційні ризики, користувачі повинні уважно вибирати і використовувати підходящий метод парсингу для своїх конкретних потреб і враховувати усі аспекти безпеки та конфіденційності даних.

Вибір методу зберігання даних є важливим завданням, яке визначається рядом факторів. Жорсткі диски, які включають HDD та SSD, використовуються залежно від обсягу даних і потреб у продуктивності. Хмарне зберігання надає зручний доступ до даних, тоді як оптичні носії та резервне копіювання забезпечують довгострокове зберігання та відновлення даних. Бази даних, флеш-накопичувачі, інтернет-сервіси, системи керування версіями та інші методи також використовуються відповідно до конкретних завдань. Важливо враховувати потреби та вимоги проекту чи організації при виборі оптимального методу зберігання даних.

# 4 ОПИС РЕАЛІЗАЦІЇ СИСТЕМИ

Вибір правильних інструментів розробки є ключовою складовою успішного процесу створення програмного забезпечення. Відповідні інструменти визначають ефективність розробки, якість та продуктивність команди. Коректно обрані інструменти дозволяють оптимізувати робочий процес, надають зручний інтерфейс для спілкування команди розробників та забезпечують ефективну систему.

Ключовими етапами розробки програмного забезпечення є планування, дизайн і безпосередня розробка. Планування визначає стратегію та мету проекту, установлює терміни та ресурси, необхідні для успішного завершення. Дизайн визначає архітектуру системи, вигляд та функціонал продукту. Цей етап є вирішальним для створення ефективної та інтуїтивно зрозумілої системи. Розробка є фазою активного програмування, де ідеї та концепції перетворюються в реальний код.

Успішна реалізація цих етапів залежить від правильного вибору інструментів, які сприяють комунікації та співпраці всіх учасників процесу розробки, а також дозволяють ефективно впоратися із завданнями кожного етапу.

## 4.1 Вибрані джерела для пошуку інформації

Перше джерело, Google Scholar (Google Академія), є однією з найбільш впливових інтернет-платформ для пошуку наукових публікацій. Ресурс забезпечує доступ до обширної бази даних наукових статей, тез, патентів та інших наукових джерел. Він автоматично сканує різні джерела, включаючи журнали, конференції та університетські репозиторії. Google Scholar також надає індекс цитування, що дозволяє вам визначити впливовість певної публікації в науковому середовищі.

Друге джерело, ArXiv, є безкоштовним архівом преддокторських досліджень в різних наукових областях. Ресурс дозволяє науковцям самостійно публікувати свої статті перед офіційним рецензуванням. Це сприяє швидкому поширенню

нових наукових відкриттів. ArXiv покриває широкий спектр дисциплін, від фізики та математики до комп'ютерних наук та соціальних наук.

Обидва ці джерела є важливими інструментами для наукового аналізу, надаючи дослідникам доступ до різноманітної та актуальної інформації.

## 4.2 Обрана мова програмування Python

Python - це високорівнева, інтерпретована мова програмування, яка визначається своєю простотою та читабельністю коду. Розроблена Гвідо ван Россумом у початку 1990-х років, Python вже давно завоював популярність серед програмістів, науковців, аналітиків даних та інших галузей.

Однією з основних особливостей Python є його синтаксис, який надає можливість виражати ідеї з меншою кількістю рядків коду порівняно з іншими мовами програмування. Це дозволяє розробникам писати програми швидше та зберігати їх легше в майбутньому.

Мова Python підтримує об'єктно-орієнтоване, функціональне та імперативне програмування, що робить її універсальною для великої кількості завдань. Це також дозволяє розробникам створювати ефективні та модульні програми, які можуть бути легко перевикористані та розширені.

Ще однією перевагою мови є велика кількість бібліотек та фреймворків, доступних для Python. Це робить його ідеальним вибором для розробки веб-додатків, наукових обчислень, штучного інтелекту, обробки даних та багатьох інших галузей.

Ще однією важливою особливістю Python є активне спільнота розробників, яка постійно внесенням удосконалень та розширенням функціоналу мови. Це гарантує, що Python залишатиметься актуальним та конкурентоспроможним інструментом у світі програмування.

В підсумку програмування Python має наступні плюси та недоліки:

- Простота та Читабельність: синтаксис Python легкий для вивчення і читання, що сприяє швидкому розвитку програм.
- Широка екосистема: велика кількість бібліотек і фреймворків дозволяє вирішувати різноманітні задачі.
- Крос-платформенність: розроблені програми можна запускати на різних операційних системах без змін вихідного коду.
- Спільнота розробників: активна та підтримуюча спільнота робить Python живою та зростаючою мовою.
- Швидкодія: у порівнянні з іншими мовами, Python може бути менш ефективним для виконання деяких завдань через інтерпретацію коду.
- Обмежені можливості мобільної розробки: Python не є основною мовою для розробки мобільних додатків, хоча існують фреймворки, такі як Kivy та BeeWare.

## 4.3 Опис обраного ORM Peewee

Peewee – це міні-ORM (Object-Relational Mapping) для мови програмування Python, яка надає простий та інтуїтивно зрозумілий спосіб взаємодії з базами даних, використовуючи об'єктно-орієнтований підхід. Він створений для полегшення роботи з реляційними базами даних, надаючи розробникам простий і зрозумілий спосіб працювати з даними, не вдаючись до деталей SQL-запитів. Ось кілька ключових аспектів Peewee ORM:

- Підтримка різних баз даних: Peewee підтримує кілька популярних реляційних баз даних, таких як SQLite, MySQL, PostgreSQL та інші. Це дозволяє легко переносити код між різними системами керування базами даних.
- Моделі даних: Peewee використовує поняття моделей даних, які є Python-класами, що становлять таблиці в базі даних. Кожен екземпляр класу представляє окремий запис у таблиці.

- Інтуїтивне визначення моделей.
- Відносини між таблицями: Peewee підтримує визначення відносин між таблицями, таких як один до одного, один до багатьох і багато до багатьох.
- Транзакції та безпека: Peewee надає можливості роботи з транзакціями для забезпечення цілісності даних та безпеки операцій.
- Підтримка міграцій: Peewee також має вбудовану підтримку міграцій, що полегшує зміну структури бази даних у процесі розвитку програми.

Peewee є простим у використанні ORM, який забезпечує зручний спосіб роботи з базами даних у додатках, написаних на Python.

## 4.4 Реляційна база даних SQLite

SQLite - це компактна, вбудована реляційна система управління базами даних (СУБД), яка забезпечує легкий доступ до баз даних за допомогою мови SQL. Основні характеристики SQLite:

1. Типи даних: SQLite підтримує стандартні типи даних, такі як INTEGER, TEXT, REAL (для чисел з плаваючою точкою), BLOB (для бінарних об'єктів) і NULL.
2. Вбудована база даних: SQLite не вимагає окремого сервера та працює вбудовано в програму. Це означає, що база даних представлена одним файлом, який можна легко переміщати та копіювати.
3. Динамічні таблиці: таблиці в SQLite можуть бути створені або змінені під час виконання програми, що робить його гнучким для розвитку та використання.
4. Транзакції: SQLite підтримує транзакції, що дозволяє вам виконувати групу операцій як єдину атомарну операцію.
5. Індексація: ви можете створювати індекси для полів таблиць, що поліпшує швидкість виконання запитів.

6. Безпека: SQLite підтримує механізм авторизації та ролей доступу, дозволяючи обмежувати доступ користувачів до різних об'єктів бази даних.
7. Сумісність з SQL: SQLite дотримується стандартів SQL, забезпечуючи можливість використовувати стандартні запити та команди.
8. Тригери та збережені процедури: SQLite підтримує тригери та збережені процедури, що дозволяє вам визначати власні дії при вставці, оновленні чи видаленні даних.
9. Крос-платформеність: SQLite підтримується на різних операційних системах, таких як Windows, Linux, і macOS.

SQLite - це відмінний вибір для невеликих та середніх проектів, де не потрібно великої серверної архітектури та де важлива простота використання та низький обсяг використання ресурсів.

## 4.5 Модуль PyQT5

PyQt - це обгортка над Qt для мови програмування Python. PyQt надає Python-розробникам можливість використовувати всю потужність Qt для створення високоякісних додатків з графічним інтерфейсом.

Основні характеристики PyQt:

- Багатомовність: PyQt підтримує багатомовність, що дозволяє створювати додатки з різними мовами і перекладами.
- Сполучення Python і Qt: Забезпечує взаємодію Python-коду з бібліотеками Qt, дозволяючи створювати потужні та ефективні додатки.
- Підтримка різних платформ: дозволяє розробляти додатки, які можуть працювати на різних операційних системах, таких як Windows, Linux, MacOS.
- Велика кількість документації та спільнота: PyQt має обширну документацію та активну спільноту, що полегшує вивчення та вирішення проблем.

## 4.6 Qt Designer

Qt Designer - це інтерактивний графічний інтерфейс для розробки візуальних елементів користувацького інтерфейсу (UI) для додатків, що використовують бібліотеку Qt. З його допомогою розробники можуть швидко та ефективно створювати та налаштовувати UI своїх програм.

Основні функції Qt Designer:

Візуальний редактор інтерфейсу: дозволяє вам створювати і редагувати елементи інтерфейсу за допомогою перетягування та розташування (drag-and-drop).

Надає засоби для налаштування властивостей об'єктів інтерфейсу.

Підтримка для віджетів Qt: дозволяє вбудовувати віджети Qt (такі як кнопки, тексти, таблиці) в інтерфейс без написання коду вручну.

Генерація коду: після створення UI, Qt Designer може згенерувати код (наприклад, на мові Python), який описує створений інтерфейс.

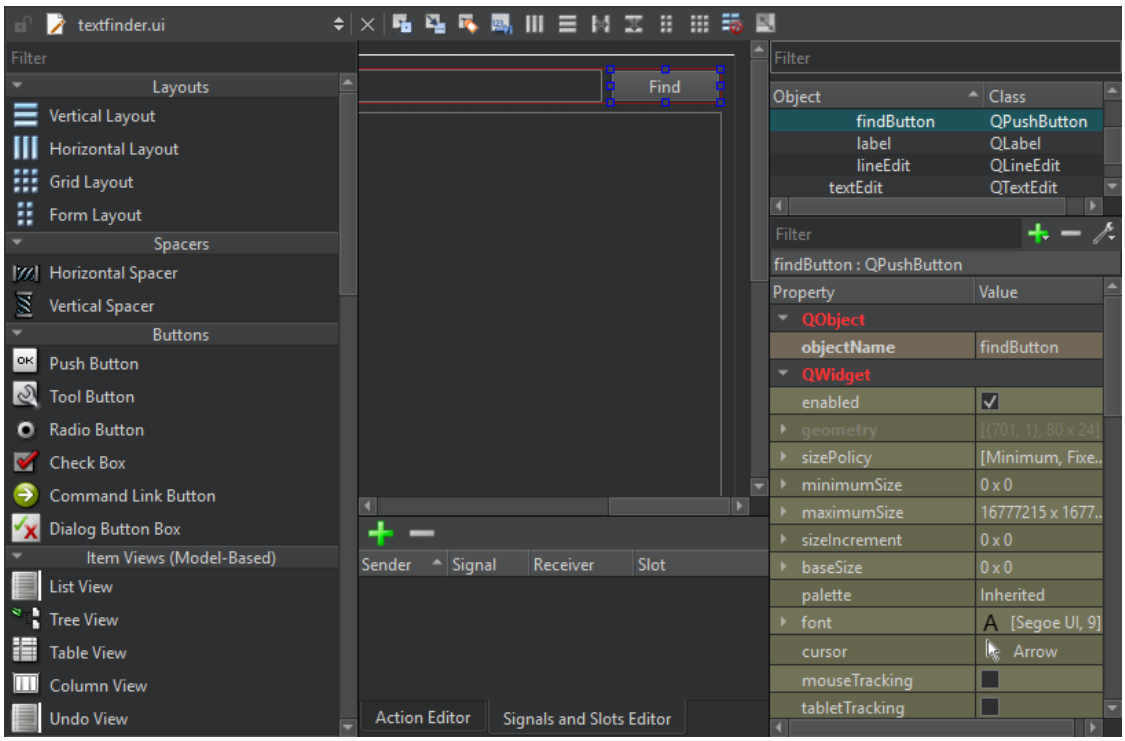



Рисунок 4 — Інтерфейс програми QtDesigner

Результати роботи в цьому інструменті можна зберігати у форматах XML або вже згенерованому автоматично Python-коді. Зовнішній вигляд інтерфейсу програми представлений на рисунку 4.

## 4.7 Середа розробки PyCharm

PyCharm – це інтегроване середовище розробки (IDE) для мови програмування Python, розроблене компанією JetBrains. Ця IDE призначена для полегшення процесу розробки Python-додатків, надаючи розробникам безліч зручних функцій та інструментів. Ось деякі ключові характеристики та можливості PyCharm:

Редактор коду: PyCharm надає потужний редактор коду з підсвічуванням синтаксису, автодоповненням, інтегрованим рефакторингом та іншими можливостями, які зроблять процес написання коду ефективнішим.

Автоматичне завершення коду (Code Completion): IDE пропонує підказки та автодоповнення коду, що дозволяє розробникам швидко та легко використовувати доступні функції та методи.

Інтегрований налагоджувач (Debugger): PyCharm включає потужний наладчик, який полегшує виявлення та усунення помилок у коді.

Система керування версіями: підтримка систем керування версіями, таких як Git, SVN та Mercurial, інтегрована в IDE, що спрощує відстеження змін та спільну роботу в команді.

Вбудована підтримка віртуальних середовищ розробки: PyCharm вміє працювати з віртуальними оточеннями Python, що дає змогу ізолювати залежності для кожного проекту.

Інструменти аналізу коду: IDE надає інструменти для аналізу коду на наявність потенційних помилок, а також попередження про стиль коду відповідно до PEP 8.

Підтримка веб-розробки: PyCharm включає інструменти для розробки веб-додатків з використанням фреймворків Django і Flask, а також підтримку HTML, CSS, JavaScript та інших технологій веб-розробки.

Зручності для роботи з базами даних: PyCharm надає можливості взаємодії з базами даних, включаючи підтримку SQL, візуальне редагування структури таблиць та виконання SQL-запитів.

Автоматичне тестування: IDE забезпечує інструменти для створення, запуску та відстеження результатів автоматичних тестів.

Підтримка різних фреймворків: PyCharm підтримує різні фреймворки, такі як Flask, Django, Pyramid та інші, надаючи інструменти для зручної розробки програм на їх основі.

Плагіни та розширення: PyCharm підтримує плагіни, що дозволяє розширювати функціональність IDE з урахуванням конкретних потреб розробника.

PyCharm є однією з найпопулярніших IDE для розробки на Python і надає великий набір функцій для полегшення процесу програмування.

## Висновки до розділу 4

Підсумовуючи можна зазначити, що обрані інструменти демонструють високу продуктивність, забезпечують зручний інтерфейс для команди розробників, а також дозволяють ефективно вирішувати завдання, пов'язані з розробкою, тестуванням і підтримкою програмного забезпечення.

Ключовими факторами при виборі інструментів є їхні можливості, спрощення робочого процесу, відповідність стандартам та можливість інтеграції з іншими інструментами чи системами.

Загалом, правильно обрані інструменти для розробки сприяють підвищенню ефективності роботи розробників, зменшенню часу на вирішення завдань та покращенню якості розроблюваного програмного забезпечення.

# 5 ТЕХНІЧНИЙ ОПИС СИСТЕМИ

## 5.1 Опис архітектури

Архітектура програмного забезпечення включає в себе всі процеси, що відбуваються всередині програми, а також взаємодію користувачів з програмою та наслідки цієї взаємодії, які можна зобразити графічно за допомогою діаграми прецедентів (англійською - Use Case diagram).

Діаграма прецедентів широко використовується при проектуванні системи для відображення поведінки системи у визначеному сценарії, який прямо пов'язаний із взаємодією користувача з системою. Іншими словами, вона надає схематичний опис процесів, які відбуватимуться у системі, а також прецедентів - учасників, які безпосередньо взаємодіють із системою.

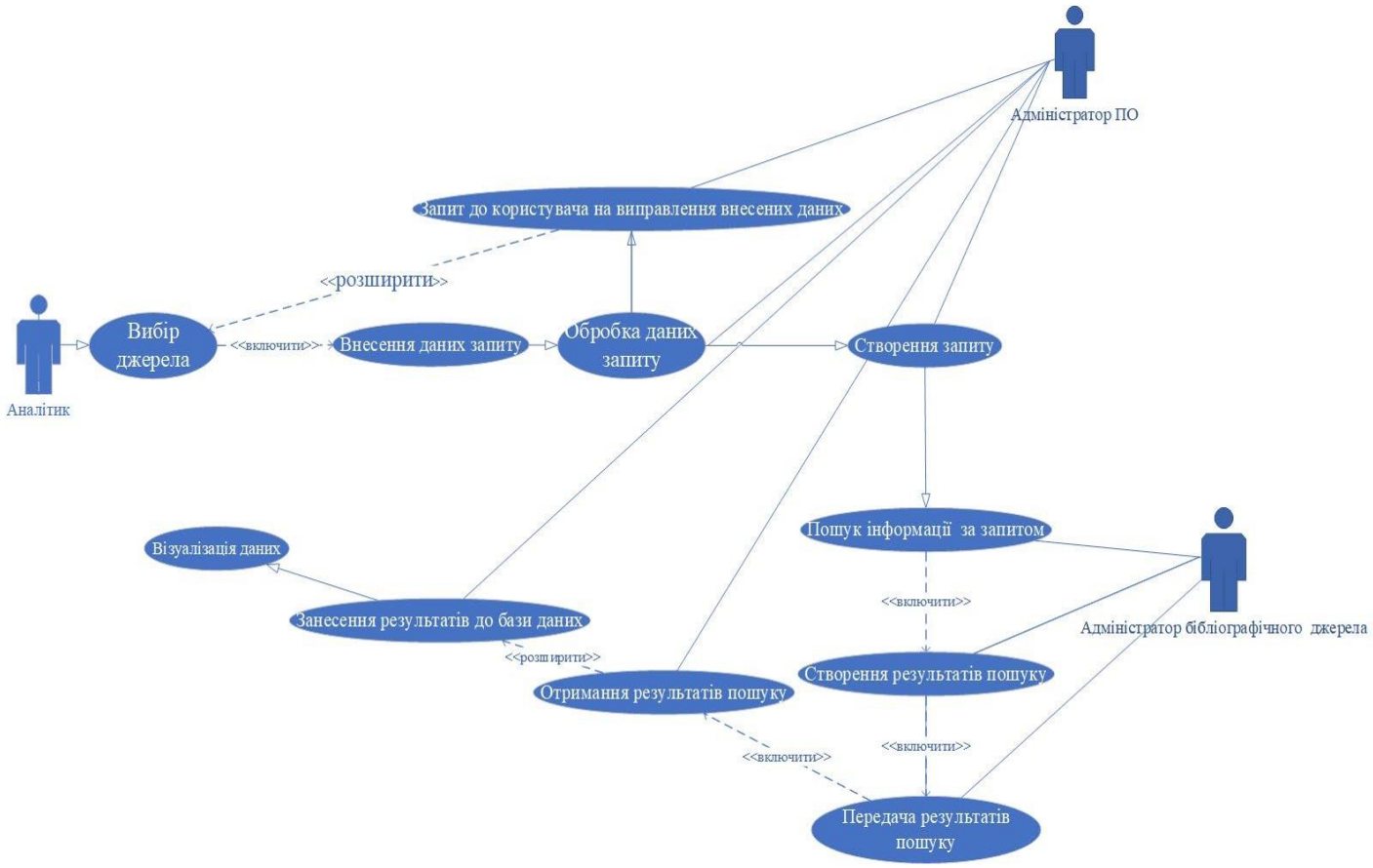



Рисунок 5.1 — Діаграма прецедентів

Таким чином, створення діаграми прецедентів дозволяє розробити концептуальну модель системи для подальшого огляду замовником з метою внесення зауважень, коригувань або затвердження системи у вигляді концепції.

Архітектура даного програмного забезпечення подана у вигляді схеми на рисунку 5.1. Під час проектування система враховує можливості кожного з її прецедентів, які будуть використовувати цю систему. Таким чином, може виникнути ієрархія, в рамках якої різні прецеденти мають різні сценарії використання системи.

## 5.2 Загальна структура проекту

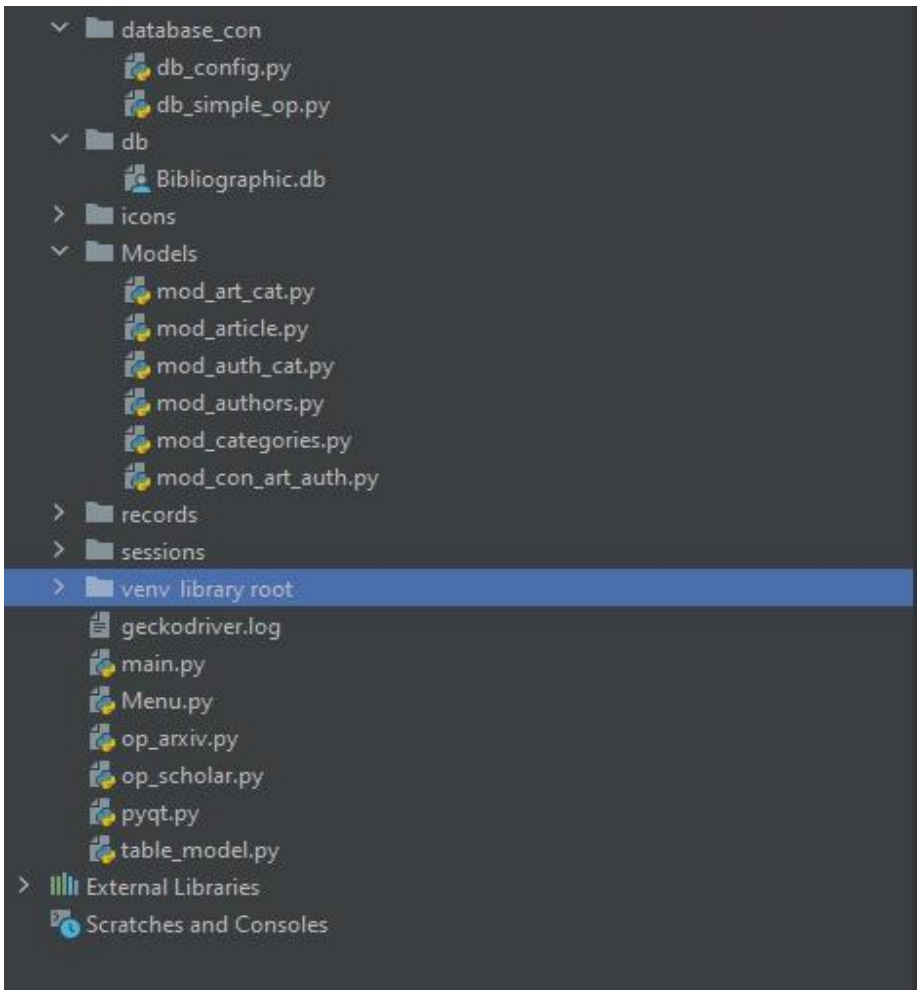



Рисунок 5.2 Загальна структура проекту

На рисунку 5.2 наведено повний перелік файлів, необхідних для стабільної роботи програми. Кожен з цих файлів відповідає за конкретний етап весь процесу, починаючи від запуску програми та з’єднання з базою даних і закінчуючи збереженням та встановленням зв'язку між збереженими даними.

Таблиця 5.1 — Перелік файлів та їх загального призначення

| Файл | Призначення у системі |
|---|---|
| db_config.py | Містить конфігураційні дані, такі як шлях для зберігання файлів, адреса для підключення до внутрішньої бази даних та ключ для з'єднання з генератором проксі, зберігаючи їх у відповідних файлах програми. |
| db_simple_op.py | Містить набір функцій для взаємодії з базою даних, таких як підключення до неї, збереження даних, відключення від бази та можливість створення нової бази даних. |
| Bibliographic.db | Файл у якому зберігається база даних |
| Mod_art_cat.py<br>Mod_article.py<br>Mod_auth_cat.py<br>Mod_authors.py<br>Mod_categories.py<br>Mod_con_art_auth.py | Зберігають моделі для взаємодії з відповідними таблицями в базі даних за їхніми іменами. Моделі включають в себе типи даних для кожного елементу в базі даних, а також імена таблиць та параметри сортування даних у базі. |
| main.py | Файл, в якому описані сценарії взаємодії інтерфейсу та функцій, які реалізовані в інших файлах програми. |

| | |
|---|---|
| Menu.py | Файл у якому прописаний прототип інтерфейсу користувача |
| op_arxiv.py | Файл, в якому визначені функції для виконання пошуку в ресурсі ArXiv, відповідно до сценаріїв, які описані у файлі main.py. |
| op_ scholar.py | Файл, де визначені функції для виконання пошуку в ресурсі Google Scholar, відповідно до сценаріїв, описаних у файлі main.py. |
| pyqt.py | Файл, що ініціює роботу інтерпретатора, включає в себе клас, який успадковує зазначений інтерфейс. У цьому класі визначені функції, які викликають відповідні сценарії при взаємодії користувача з інтерфейсом. |
| table_model.py | Файл, де знаходяться функції, відповідальні за відображення інформації в інтерфейсі користувача. |

## 5.3 Структура БД

Доречність схеми структури бази даних є важливим аспектом проектування баз даних. Вона визначає, наскільки ефективно та адекватно дана схема відображає потреби користувачів та вимоги системи.

Схема повинна точно відображати потреби та вимоги бізнесу. Вона повинна бути зорієнтована на предметну область, щоб забезпечити ефективну роботу

системи у відповідності з реальними потребами користувачів. Доречна схема повинна бути ефективною з точки зору використання ресурсів, таких як простір збереження даних, швидкодія бази даних і інші. Наприклад, оптимізація структури таблиць і використання індексів може покращити продуктивність системи.

Схема повинна бути гнучкою та легко розширюваною для врахування змін в бізнес-процесах або додаткових вимог. Це важливо для того, щоб система можна було легко адаптувати до змін в середовищі або вимогах користувачів. Схема повинна забезпечувати надійність та консистентність даних. Це включає в себе використання правильних типів даних, встановлення обмежень цілісності даних і механізмів забезпечення атомарності операцій. Схема повинна бути зрозумілою та зручною для використання. Це полегшує розробку, тестування, адміністрування та інші аспекти обслуговування бази даних. Необхідно також враховувати вимоги до безпеки даних. Це включає в себе правильне управління доступом, шифрування даних та інші заходи для захисту конфіденційності та цілісності інформації.

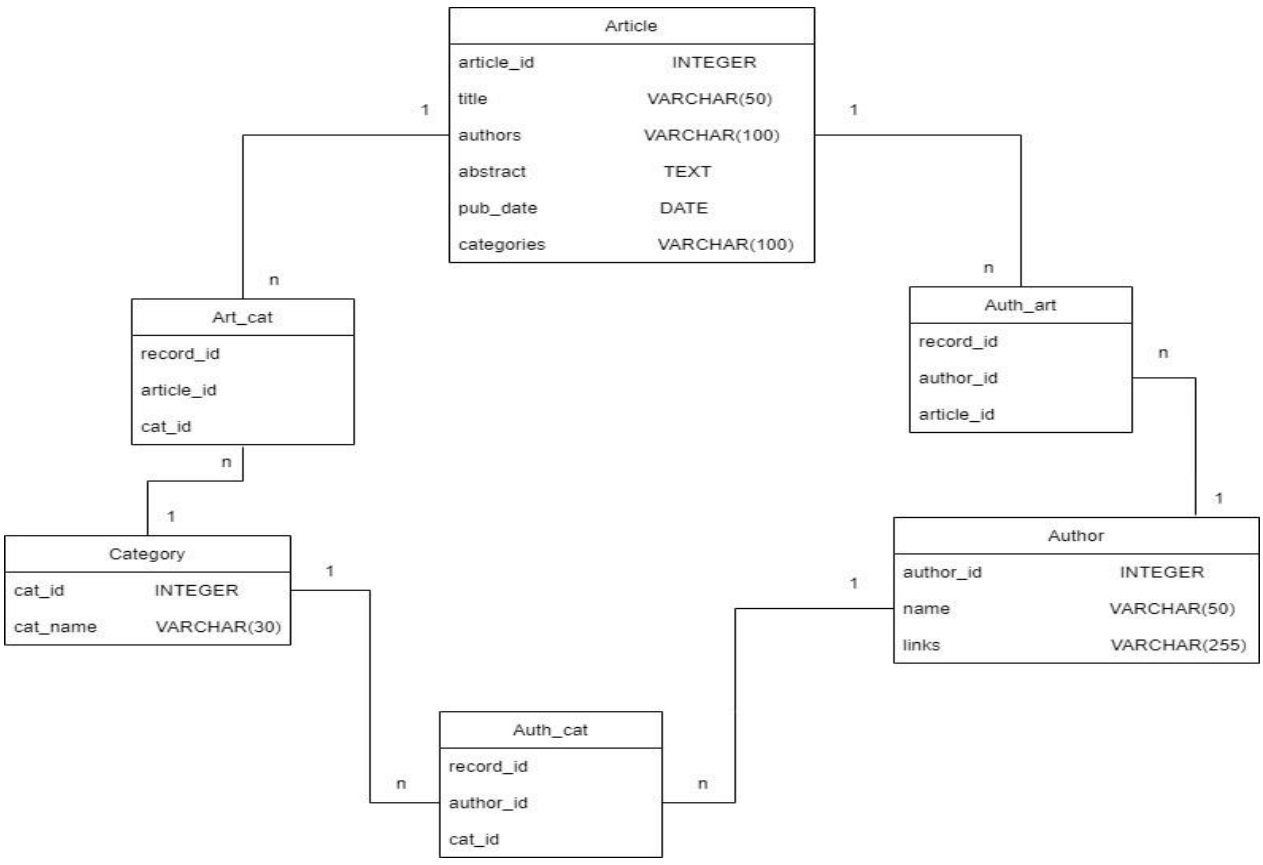



Рисунок 5.3 — Структура бази даних

Концептуальна модель бази даних дозволяє створити дизайн бази даних, враховуючи потреби програмного продукту, який буде використовувати цю базу. На рисунку 5.2 зображено саму концептуальну модель, а також взаємозв'язки між її елементами, які будуть використовуватися у проекті.

## Висновок до розділу 5

Архітектура програмного забезпечення визначає всі аспекти функціонування програми, включаючи внутрішні процеси, взаємодію з користувачами та наслідки цієї взаємодії. Діаграма прецедентів, яка графічно відображає сценарії взаємодії користувачів з системою, є важливим інструментом при проектуванні системи, дозволяючи створити концептуальну модель для подальшого огляду та затвердження.

Успішність схеми структури бази даних визначається її відповідністю потребам бізнесу, ефективністю використання ресурсів та гнучкістю для адаптації до змін. Важливою є також надійність та консистентність даних, що досягається за допомогою відповідних типів даних, обмежень цілісності та інших механізмів.

Основні аспекти врахування в процесі розробки схеми бази даних включають не лише технічні аспекти, такі як оптимізація та ефективність, але й гнучкість, зрозумілість та безпека даних. Це важливо для забезпечення успішної роботи системи відповідно до реальних потреб користувачів та забезпечення її легкості обслуговування.

# 6 ВЗАЄМОДІЯ КОРИСТУВАЧА З СИСТЕМОЮ

## 6.1 Вимоги програмного забезпечення

Для ефективної роботи програми користувач повинен відповідати наступним вимогам:

- Процесор з двома ядрами та тактовою частотою від 2.1 до 2.5 гігагерц, або вище.
- Вільне місце на накопичувачу не менше 2 гігабайт.
- Обсяг оперативної пам'яті не менше 2 гігабайт.
- Операційна система Windows 7 / Ubuntu 16.04 або більш нова.
- Інсталяція інтерпретатора мови програмування Python версії не нижче 3.9, а також необхідних бібліотек, які перераховані у вкладеному файлі.
- Наявність якісного Інтернет-з'єднання, оскільки програма взаємодіє з Інтернет-ресурсами.

Бажане встановлення системи управління базами даних (СУБД) з можливістю взаємодії з базами даних на основі SQLite для роботи з даними у базі даних.

## 6.2 Алгоритм роботи системи та його результати

Після запуску програми, перед тим як користувач побачить інтерфейс, система автоматично підключиться до існуючого файлу бази даних або створить новий файл з необхідним шаблоном для подальшого заповнення.

Після успішного підключення до бази даних користувач перегляне графічний інтерфейс, обладнаний наступними елементами:

- Два поля з прапорцями для вибору ресурсу, який буде задіяно при пошуку.
- Поле "Search by" для вибору сценарію (критерію) пошуку.

- Поле "Searching parameter" для введення користувачем даних, які будуть використані при запиті.
- Поле "Maximum result" для визначення максимальної кількості результатів пошуку для зберігання у базі даних.
- Поле "Category" для визначення наукових напрямків, за якими буде здійснено пошук відповідних статей.
- Кнопки: "Search" – для виконання сценарію пошуку, "Save current state" – для збереження копії бази даних під спеціальним ім'ям, "Exit" – для безпечного виходу з програми.

Процес виконання програми користувачем було проілюстровано на рисунках 6.1-6.3. Після запуску програми система автоматично встановить з'єднання з існуючим файлом бази даних або створить новий файл з необхідним шаблоном для подальшого заповнення, перш ніж користувач отримає доступ до інтерфейсу.


Рисунок 6.1 — Зображення початку процесу пошуку інформації

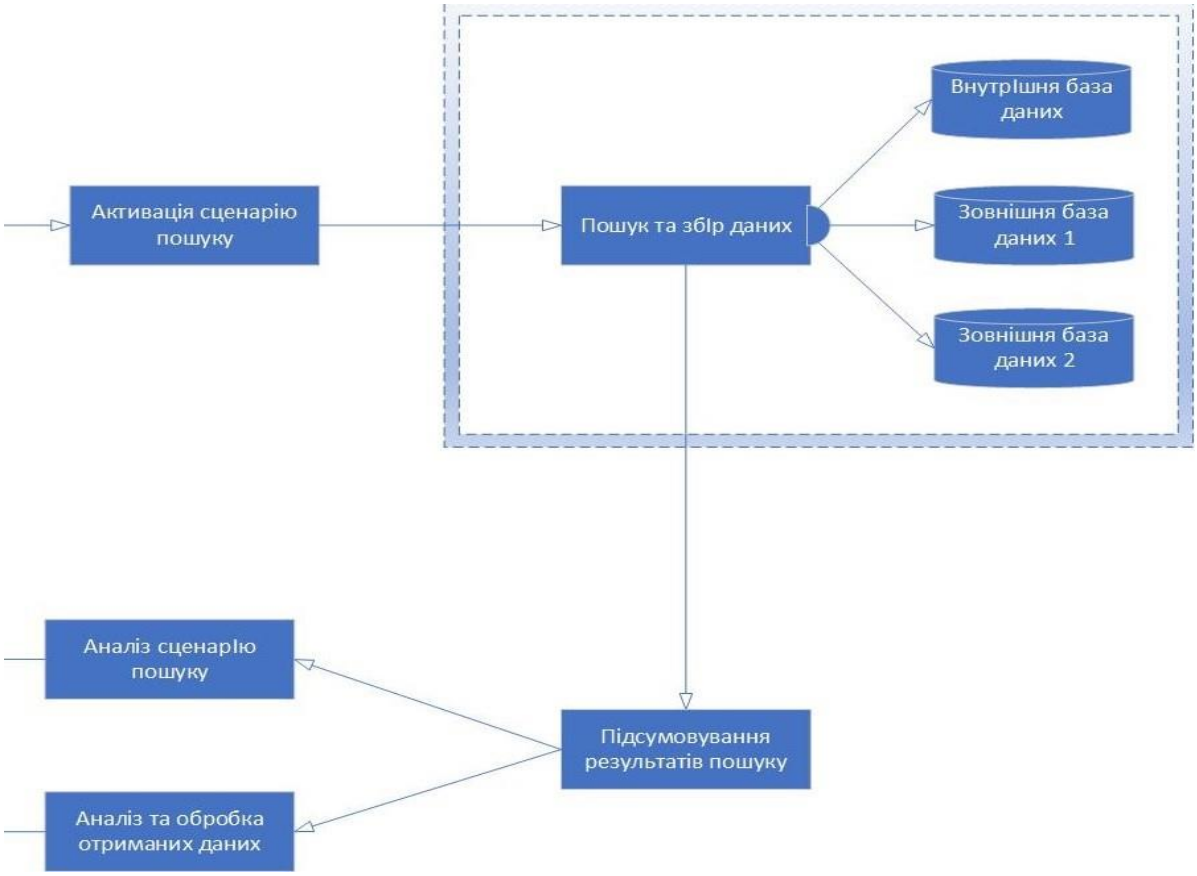



Рисунок 6.2 — Процес зберігання результатів пошуку

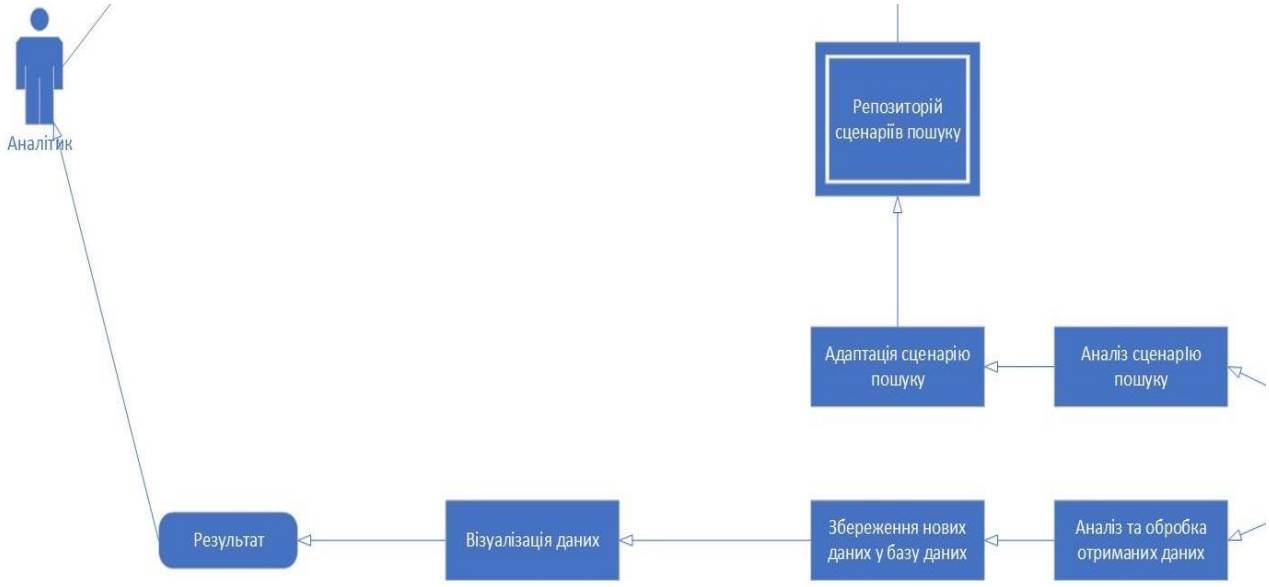



Рисунок 6.3 — Процес збереження необхідних даних та їх візуалізації

В даному розділі буде сформульована ідея стартапу, проаналізовано вже виконану роботу та ринок з урахуванням потреб користувачів, існуючих конкурентів та можливих проблем.

Інтерфейс містить поля для введення даних, які визначають сценарій пошуку на ресурсах, з яких користувач бажає отримати бібліографічну інформацію. Зовнішній вигляд інтерфейсу зображено на рисунку 6.4. Залежно від потрібних даних для кожного сценарію пошуку програма буде зчитувати введені користувачем дані у відповідних полях.

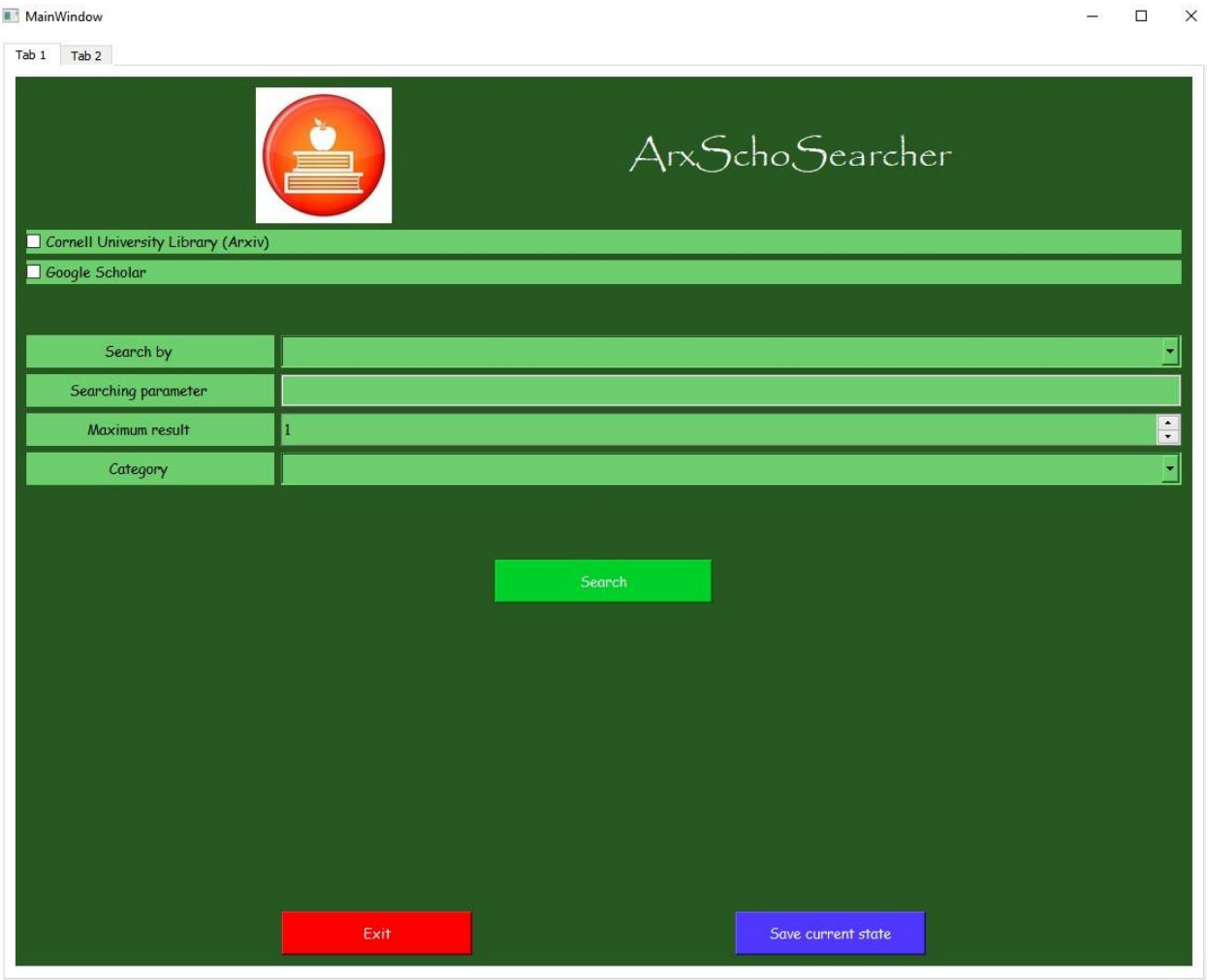



Рисунок 6.2.2— Інтерфейс головного меню пошуку

У випадку відсутності параметрів у обов'язковому полі для заповнення користувач отримає сповіщення від програми. Приклад сповіщення представлено на рисунку 6.5.

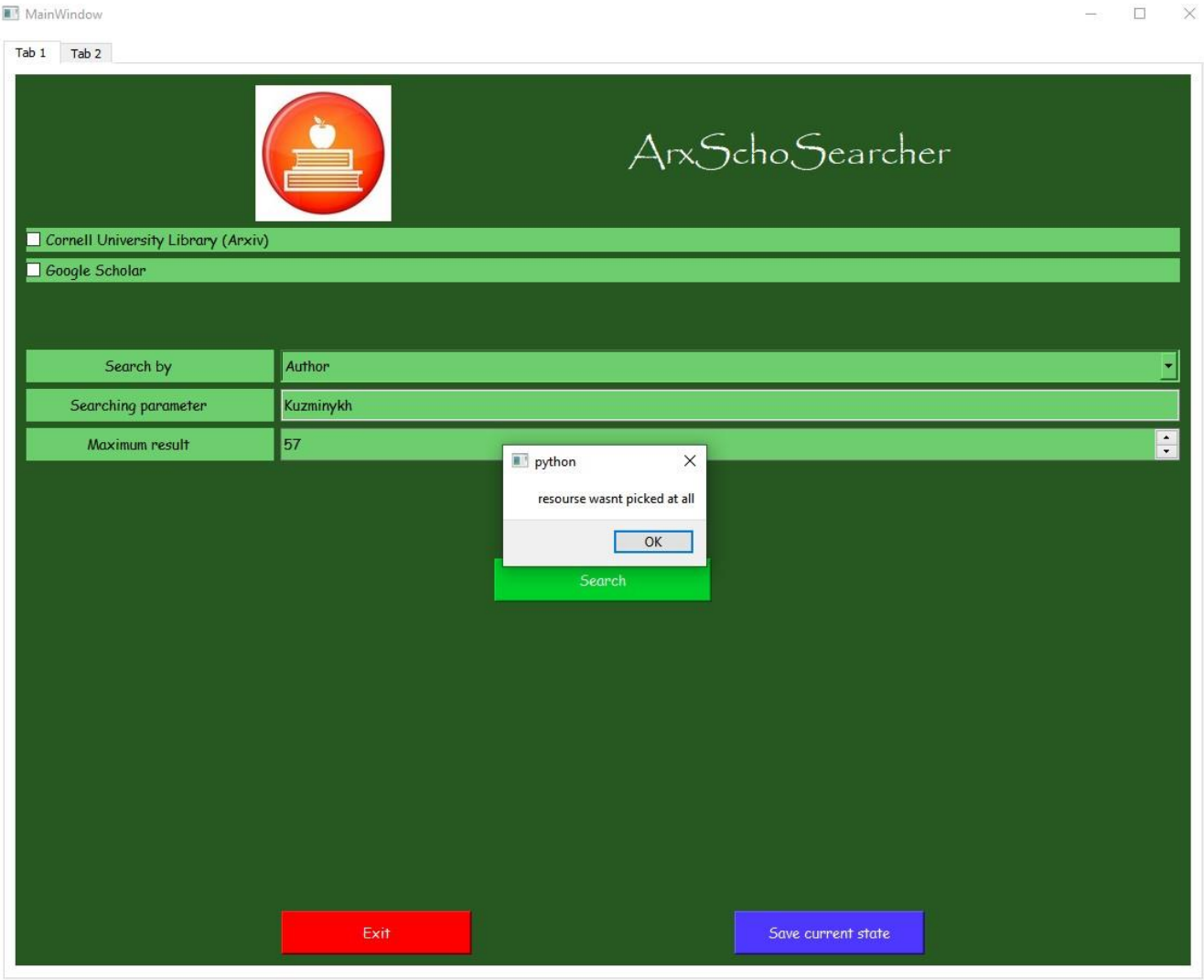



Рисунок 6.5 — Приклад сповіщення від програми. Сценарій коли користувач не обрав жодного ресурсу для пошуку інформації.

Користувач може проводити пошук бібліографічної інформації на двох ресурсах. На першому ресурсі, Cornell University Library (ArXiv), доступні параметри пошуку: Title (назва статті), Author (ім'я автора публікації) та Category (наукові напрямки), в яких статті були написані. На другому ресурсі користувач може шукати за параметрами Title та Author. Крім того, користувач може вказати бажану кількість перших знайдених результатів для їх збереження. Після завершення пошуку користувач отримає повідомлення про завершення процесу пошуку та занесення інформації до бази даних. Приклад такого сповіщення представлено на рисунку 6.6.

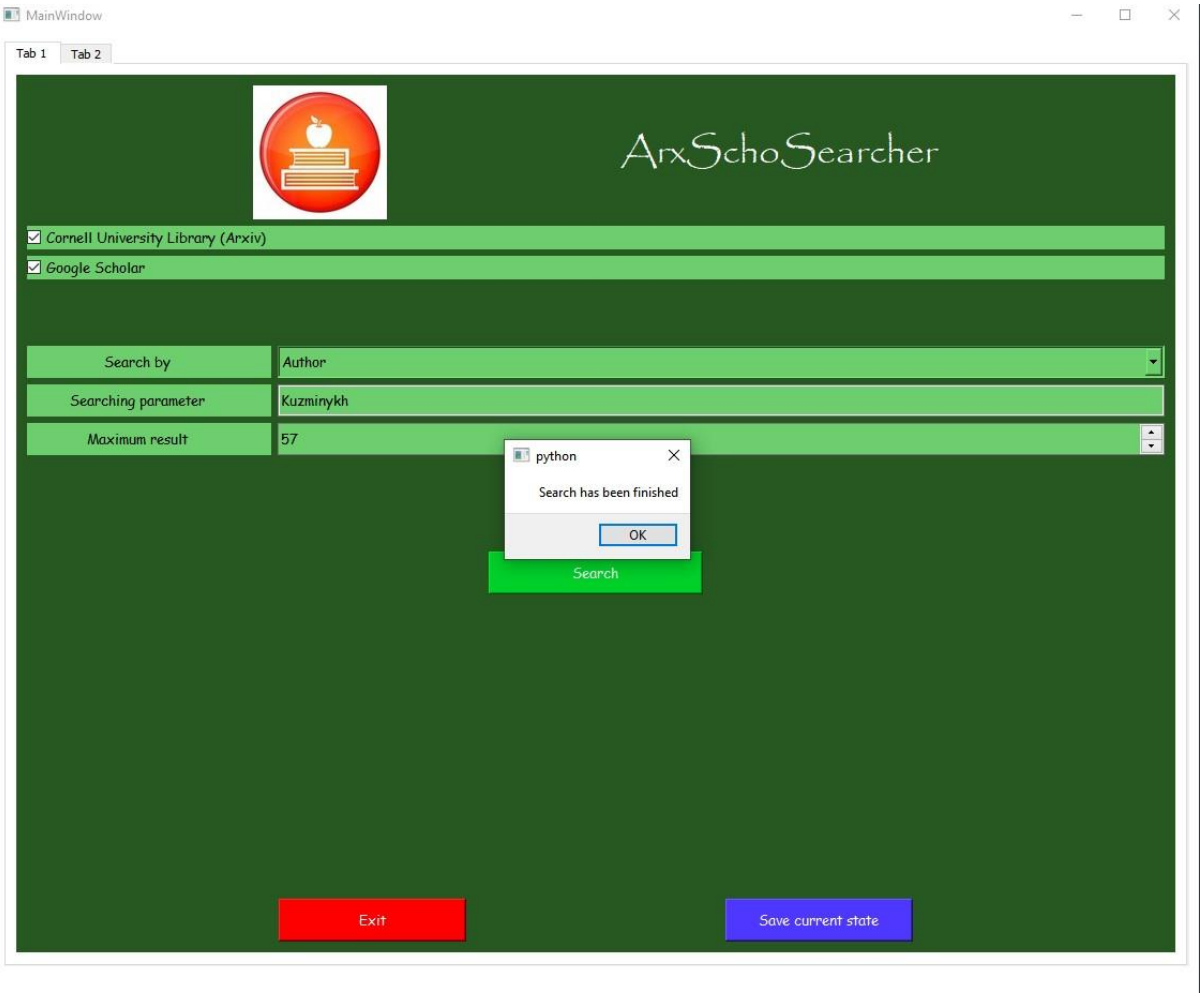



Рисунок 6.6 — Приклад сповіщення про закінчення пошуку інформації

Програмне забезпечення дозволяє користувачеві створити копію файлу бази даних. Файл бази даних може бути збережений користувачем у папці "records" під вказаним ним ім'ям або під стандартним ім'ям "record_" з точним вказанням часу, коли цей файл було збережено. Зовнішній вигляд діалогового вікна для цього процесу зображено на рисунку 6.7.

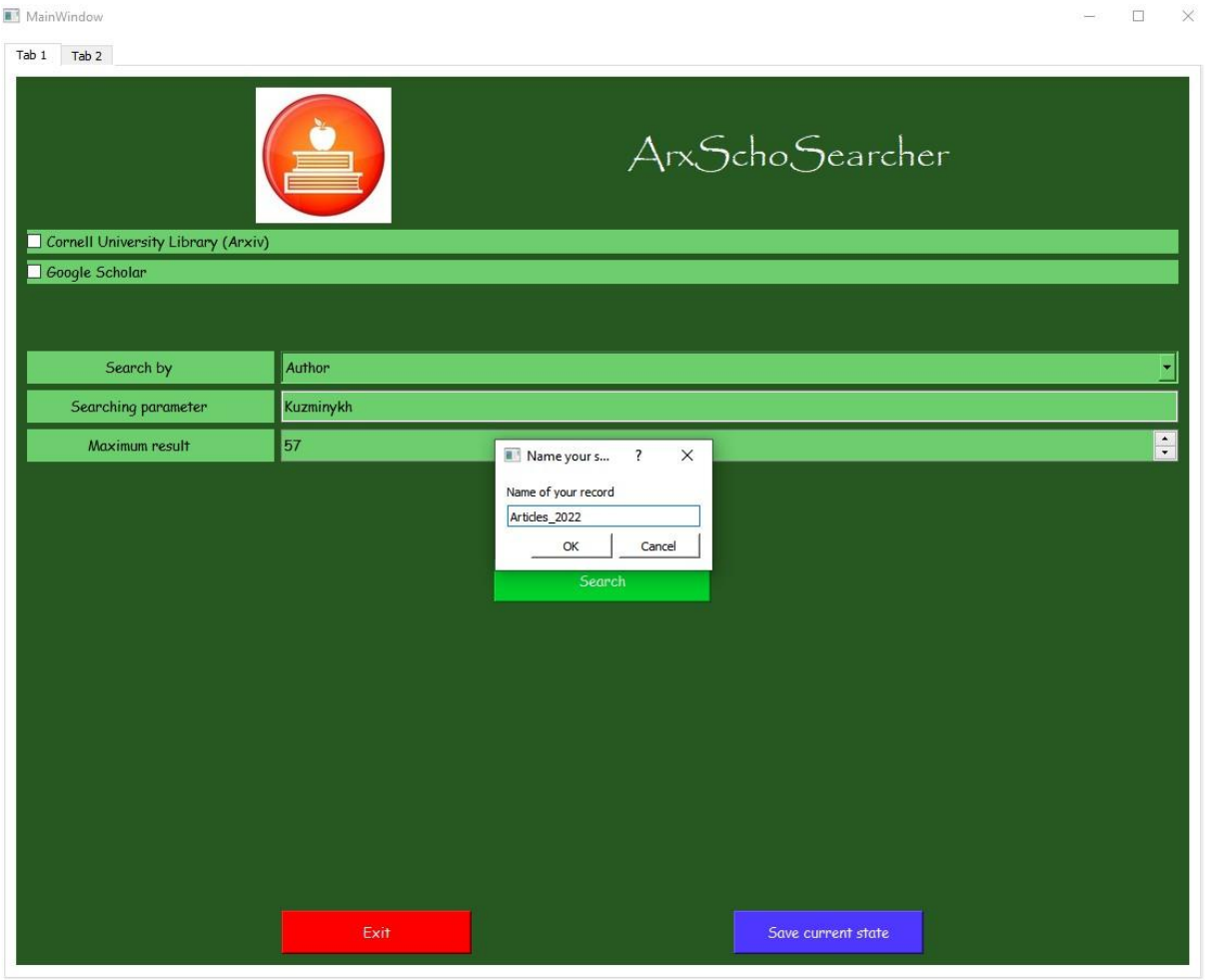



Рисунок 6.7. — Діалогове вікно для назви файлу користувачем

Результатами виконання програми є зібрані дані, які було зібрано парсером та розміщено до відповідних таблиць у базі даних програми. Користувач може переглядати інформацію, що знаходиться у базі даних, у режимі реального часу. Приклади збереженої інформації представлені на рисунках 6.8 та 6.9.


| | art_id | Title | Authors | Abstract | Publication date | Category |
|---|---|---|---|---|---|---|
| 0 | 1 | Category-Learning with Context-Augmented ... | Denis Kuzminykh, Laida Kushnareva, Timofey Grigoryev,... | Finding an interpretable non-redundant representation ... | 2020 | cs.LG |
| 1 | 2 | Impact of Network and Host Characteristics on the ... | Ievgeniia Kuzminykh, Bogdan Ghita, Alexandr Silonosov | Authentication based on keystroke dynamics is a ... | 2020 | cs.CR |
| 2 | 3 | The Challenges with Internet of Things for Business | Ievgeniia Kuzminykh, Bogdan Ghita, Jose M. Such | Many companies consider IoT as a central element for ... | 2020 | cs.CR |
| 3 | 4 | Comparative Analysis of Cryptographic Key Manageme... | Levgeniia Kuzminykh, Bogdan Ghita, Stavros Shiaeles | Managing cryptographic keys can be a complex task for... | 2021 | cs.CR |
| 4 | 5 | Audio Interval Retrieval using Convolutional Neural ... | Ievgeniia Kuzminykh, Dan Shevchuk, Stavros Shiaeles, ... | Modern streaming services are increasingly labeling ... | 2021 | cs.SD |
| 5 | 6 | Control of magnetic susceptibility of probiotic strain ... | Svitlana Gorobets, Oksana Gorobets, Liubov Kuzminykh | The paper investigates the increase in the natural ... | 2022 | physics.bio-ph |

Рисунок 6.8 — Приклад зібраної інформації користувачу системи у вікні системи

| | | |
|---|---|---|
| 17 | Jose M. Such | Cornell University Library (arXiv) article: http://... |
| 18 | Levgeniia Kuzminykh | Cornell University Library (arXiv) article: http://... |
| 19 | Stavros Shiaeles | Cornell University Library (arXiv) article: http://... |
| 20 | Dan Shevchuk | Cornell University Library (arXiv) article: http://... |
| 21 | Svitlana Gorobets | Cornell University Library (arXiv) article: http://... |
| 22 | Oksana Gorobets | Cornell University Library (arXiv) article: http://... |
| 23 | JG Brookshear | no Author Account link |
| 24 | PJ Denning | Google Scholar Author Account link: https://... |
| 25 | A McGettrick | no Author Account link |
| 26 | G Dodig-Crnkovic | Google Scholar Author Account link: https://... |
| 27 | A Ralston | no Author Account link |
| 28 | ED Reilly | no Author Account link |
| 29 | D Hemmendinger | no Author Account link |
| 30 | SM Ross | Google Scholar Author Account link: https://... |

Рисунок 6.9 – Приклад відображення зібраної інформації через стороннє програмне забезпечення

Зібрана під час виконання алгоритму інформація зберігається в трьох основних таблицях: Article, Author, Category. Крім того, існують три допоміжні таблиці, які містять унікальні ідентифікатори (ID) для кожного ключового елементу в системі. Ці допоміжні таблиці призначені для зберігання зв'язків між: статтями та їх категоріями, статтями та їх авторами, авторами та категоріями (науковими напрямками), в яких автор має напрацювання

## Висновок до розділу 6

Система розроблена таким чином, що не потрібно освоєння спеціальних навичок для повноцінного використання її функціоналу з метою виконання завдань зі збирання та первинної обробки даних. Вбудовані функції в системі працюють ефективно та точно, відповідаючи всім стандартам якості. Інтерфейс програми інтуїтивний й зрозумілий, що забезпечує комфортну взаємодію користувача з продуктом.

# 7 РОЗРОБКА СТАРТАП-ПРОЄКТУ

Цей стартап може знайти застосування у наукових організаціях для спрощення взаємодії з існуючими відкритими системами та автоматизації оцінки міжнародної науково-технічної діяльності. З достатнім залученням ресурсів і підтримкою системи ідея програмної платформи може успішно розвиватися та викликати великий інтерес серед наукових організацій.

Ця система призначена для використання аналітиками у сфері наукової діяльності, дозволяючи фінансуючим фірмам ефективніше розподіляти бюджет, спрямовуючи його на фінансування перспективних компаній. Завдяки автоматизованому збору необхідної інформації за обраною тематикою, робота аналітиків стає більш продуктивною, а також відкривається можливість аналізу більшого обсягу інформації.

Таблиця 6.1 – Опис ідеї стартап-проекту

| Зміст ідеї | Напрямки застосування | Вигоди для користувача |
|---|---|---|
| Створення системи для збору та первинної обробки різнорідних даних для визначення стану діяльності науковців. | Система призначена для використання : науковцями, аналітиками наукової діяльності, бібліографічними установами, архівними установами, окремими дослідниками, студентами та викладачами, фінансистами. | Швидкі й об’єктивні рішення, спрямовані на розподіл ресурсів |

## 7.1 Аналіз ідеї стартапу

Зараз, коли ми визначили можливі ідеї для стартапу, необхідно провести аналіз їх відповідно до розробленого програмного продукту, щоб порівняти вимоги до фінального продукту з тим, що вже було розроблено у контексті даної роботи. Розглянемо необхідний функціонал для кожної ідеї та рівень його поточної реалізації або важкість його впровадження у майбутньому.

Для замовника або користувача продукту важливим чинником є автоматизація процесів, що означає, що пошук профілів користувачів також повинен здійснюватися автоматично. Розробка цього функціоналу не є легкою задачею і може вимагати значних вкладень як з боку розробки, так і з точки зору фінансування.

Ще одним важливим аспектом для бізнесу є забезпечення вірності усіх даних, які надає розроблене програмне забезпечення. Виконання цієї вимоги є дещо простішим завданням і залежить від різних факторів, таких як джерела інформації та фінансовий успіх системи.

Для наукових та державних установ також важливою є повнота та вірність інформації. Щодо вірності інформації, яка вже обговорювалася вище, повноту інформації можна забезпечити, розширивши список джерел. Завдання розширення цього списку може не обов'язково входити в область розробки програмного забезпечення і може бути делеговане користувачеві системи.

## 7.2 Аналіз ринку

Аналіз попиту на розроблений продукт з боку бізнесу фактично є оцінкою попиту на нових кандидатів на поточний момент. Після вивчення статті з останніми новинами та статистикою вакансій та наймів в Україні, зокрема в інформаційно-технологічному секторі, можна зробити висновок, що наразі ринок переживає

смуту, що може призвести до зниження попиту на продукт. Проте, важливо відзначити, що попит не є повністю відсутнім.

Це свідчить про те, що незважаючи на труднощі на ринку праці, існує певний попит на нових кандидатів та, ймовірно, на інноваційні продукти, які можуть відповідати потребам ринку. У зв'язку з цим, можливо розглядати стратегії адаптації продукту до змінливого середовища та залучення цільової аудиторії в умовах зміни кон'юнктури.

Щодо конкурентних продуктів, вони не мають достатньої відкритості та частіше за все не об’єднують великий спектр напрямків.

Попит на розроблену систему від наукових та державних установ буде визначатися, переважно, наявністю самої системи, оскільки на даний момент подібних рішень не існує. Проте важливо врахувати, що відсутність конкуренції може зробити продукт більш привабливим для цільового сегменту.

Зазначені проблеми, які можуть виникнути у стартапу, включають недостатність функціоналу, розробку інших потенційних систем та відсутність попиту. Проте важливо зауважити, що ці проблеми вважаються мінорними, і більшість з них можна легко вирішити при достатньому зусиллі та стратегічному плануванні.

## Висновки до розділу 7

У цьому розділі були сформульовані та описані ідеї стартап-проекту, проведено аналіз готовності розробленої системи до впровадження та проаналізовано ситуацію на ринку, а також визначено конкурентоспроможність. З упевненістю можна заявити, що під час подальшого аналізу та впровадження певної ідеї можна досягти успіху для створеного стартап-проекту та розробленої системи в цілому.

# ВИСНОВКИ

В ході виконання поставленої задачі було створено систему та отримано наступні результати:

- Проведено аналіз існуючих аналогів для виконання поставленої задачі. Було вирішено створити власне програмне забезпечення з урахуванням усіх потреб та побажань можливих користувачів системи.
- Досліджено методи парсингу для ресурсів з аналогічною структурою веб-сайтів та з подібними даними, які зберігаються в середині цих ресурсів.
- Було обрано зручну та ефективну мову програмування, середу розробки програмного забезпечення та бібліотеки для розробки програмного забезпечення, яке буде виконувати поставлену задачу.

В ході розробки було досліджено предметну область та проаналізовано спеціалізовану літературу. Були здобуті навички праці зі спеціалізованими бібліотеками мови програмування, які були створенні для виконання подібних задач.

На основі отриманих навичок було побудовано відповідну систему, яка виконує поставлену задачу. Система створена без потреби опанування особливих навичок для користування повним функціоналом системи, з метою виконання задачі по збору та первинної обробки бібліографічних даних.

Спроектована система розроблена з урахуванням її подальшого можливого розвитку. Були проаналізовані можливі сценарії подальшого розвитку систему, які можуть покращити результати роботи.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Етапи створення веб-сайтів. URL: http://alextexnok.blogspot.com/p/blog-page_85.html
2. Поняття, структура та різновиди веб-сайтів. Автоматизоване розроблення веб-сайтів. URL: http://www.ndu.edu.ua/liceum/web.pdf
3. Python documentation. URL: https://docs.python.org/3/
4. Peewee documentation. URL: http://docs.peewee-orm.com/en/latest/
5. PyQt5 documentation. URL: https://doc.qt.io/qtforpython/
6. PyQt5 керівництво. URL: https://pythonist.eng/tutorial-popyqt5/
7. SQLite tutorial. URL: https://unetway.com/tutorial/sqlite
8. ORM. URL: https://simpleone.en/glossary/orm/
9. Google Scholar. Google Академія. URL: https://scholar.google.com/
10. Google Академія для науковців. URL: http://www.library.univ.kiev.ua/ukr/res/google_scholar.pdf
11. Wolfram Alpha. URL: https://www.wolfram.com/education/?source=nav
12. ScienceDirect. URL: https://kai.eng/documents/678086/
13. Highload. URL: https://highload.today/uk/parsing/
14. Geeksforgeeks. URL https://www.geeksforgeeks.org/html-parsing-and-processing/

# ДОДАТКИ

## ДОДАТОК А Лістинг розробленої програми

Файл **op_scholar.py** – виконання усіх пошукових операцій на платформі Google Scholar

```
# імпортування модулів
import scholarly
from scholarly import ProxyGenerator
from scholarly import scholarly
# Імпортування моделей головних таблиць
from Models.mod_authors import Author
from Models.mod_article import Article
from Models.mod_categories import Category
#Імпортування моделей допоміжних таблиць
from Models.mod_auth_cat import Auth_Cat
from Models.mod_art_cat import Art_Cat
from Models.mod_con_art_auth import Auth_Art
#Імпортування конфігураційних даних
from database_con.db_config import sch_key, db
# Підключення проксі генератора
pg = ProxyGenerator()
success = pg.ScraperAPI(str(sch_key))
scholarly.use_proxy(pg)
#Функція пошуку на платформі Google Scholar за ключовими словами або повною назвою у назві статті
def search_by_keyword(parametr,search_amount):
 res_count = 0
 pub_Links_separ = " "
```

```
 raw_cat = "Category unknown"
# Створення запиту та занесення його результатів у 1 змінну
 search_query = scholarly.search_pubs(parametr)
 search_number = 0
#Перебирання кожного результату пошуку циклом
for res in search_query:
 res_count += 1
 search_number += 1
 if search_number == search_amount:
 break
 else:
 authors_list = []
 authors_id = []
#Занесення даних у потрібному вигляді у змінні
 result_text = res['bib']
 raw_title = result_text['title']
 raw_abstract = result_text['abstract']
 raw_year = result_text['pub_year']
 raw_category = raw_cat
 result_author_id = res['author_id']
 result_author = result_text['author']
 for item in result_author:
 authors_list.append(item)
#Присвоєння кожному автору посилань на профіль(якщо існують)
 for item in result_author_id:
 link = "no Author Account link"
 if (item !=("")):
 lin = "Google Scholar Author Account link:
https://scholar.google.com/citations?hl=uk&user="
```

```
 link = lin + item
 authors_id.append(link)
 else:
 authors_id.append(link)
 i = 0
 while (i < len(authors_list)):
 author_data = authors_list[i]
 auth_acc_link = (authors_id[i])
 #Перевірка на існування записів про даного автора у базі даних

 author_check = None
 aut_check = (Author.select(Author.name).where(Author.name == author_data))
 for r in aut_check:
 author_check = r
#Занесення у базу даних , якщо запис відсутній
 if (author_check == None):
 links = auth_acc_link + pub_Links_separ
 record_author = Author.create(name=author_data, links=links)
 record_author.save()
 i += 1
 raw_authors = ', '.join(authors_list)
#Перевірка на існування записів про дану статтю у базі даних
 art_title_check = None
 art_check = (Article.select(Article.title).where((Article.title == raw_title) &
(Article.authors == raw_authors)))
 for r in art_check:
 art_title_check = r
#Занесення у базу даних , якщо запис відсутній
 if (art_title_check == None):
```

```
 record_articles = Article.create(title=raw_title,
 authors=raw_authors,
 abstract=raw_abstract,
 pub_date=raw_year,
 categories=raw_category)
 record_articles.save()
 db.commit()
#Перевірка на існування записів про дану категорію у базі даних
 cat_check = None
 aut_check = (Category.select(Category.cat_name).where(Category.cat_name ==
raw_category))
 for r in aut_check:
 cat_check = r
#Занесення у базу даних , якщо запис відсутній
 if (cat_check == None):
 record_cat = Category.create(cat_name=raw_category)
 record_cat.save()
#Перевірка на існування записів про зібрану та вже наявну інформацію в базі
даних
та присвоєння їм унікальних номерів(ID)
 for authors in authors_list:
 query_art_id_extr = (Article.select(Article.art_id).where(Article.title == raw_title))
 for id in query_art_id_extr:
 raw_art_id = id.art_id
 query_auth_id_extr = (Author.select(Author.author_id).where(Author.name ==
authors))
 for id in query_auth_id_extr:
 raw_auth_id = id.author_id
 query_auth_id_extr = (Category.select(Category.cat_id).where(Category.cat_name
```

```
== raw_category))
 for id in query_auth_id_extr:
 raw_cat_id = id.cat_id
 aut_cat_check = None
 aut_cat = (Auth_Cat.select(Auth_Cat.record_id).where(
 (Auth_Cat.author_id == raw_auth_id) & (Auth_Cat.cat_id == raw_cat_id)))
 for r in aut_cat:
 aut_cat_check = r
#Створення нових зв’язків, з урахуванням вже наявних у базі даних
 if (aut_cat_check == None):
 record_Auth_cat = Auth_Cat.create(author_id=raw_auth_id, cat_id=raw_cat_id)
 record_Auth_cat.save()
 art_cat_check = None
 art_cat = (Art_Cat.select(Art_Cat.record_id).where(
 (Art_Cat.art_id == raw_art_id) & (Art_Cat.cat_id == raw_cat_id)))
 for r in art_cat:
 art_cat_check = r
 if (art_cat_check == None):
 record_Art_cat = Art_Cat.create(art_id=raw_art_id, cat_id=raw_cat_id)
 record_Art_cat.save()
 auth_art_check = None
 auth_art = (Auth_Art.select(Auth_Art.record_id).where(
 (Auth_Art.art_id == raw_art_id) & (Auth_Art.author_id == raw_auth_id)))
 for r in auth_art:
 auth_art_check = r
 if (auth_art_check == None):
 record_Auth_Art = Auth_Art.create(author_id=raw_auth_id, art_id=raw_art_id)
 record_Auth_Art.save()
 return (res_count)
```

```
#Функція пошуку на платформі Google Scholar за авторами статей
def author_searcher(parameter, max_res):
 res_count = 0
# Створення запиту та занесення його результатів у 1 змінну
 search_query = scholarly.search_author(parameter)
#Перебирання кожного результату пошуку циклом
 for res in search_query:
 res_count += 1
 if(res_count == max_res):
 break
 else:
#Занесення даних у потрібному вигляді у змінні
 raw_category = "Category unknown"
 raw_authors = res["name"]
 link_id = res["scholar_id"]
 link_pre = "Google Scholar Author Account link:
https://scholar.google.com/citations?hl=uk&user="
 link = link_pre + link_id
 filled_author = scholarly.fill(res, sections=["publications"])
 pubs = filled_author["publications"]
#Перевірка на наявність записів про цього автора
 author_check = None
 aut_check = (Author.select(Author.name).where(Author.name == raw_authors))
 for r in aut_check:
 author_check = r
#Занесення у разі відсутності записів
 if (author_check == None):
 links = link
 record_author = Author.create(name=raw_authors, links=links)
```

```
 record_author.save()
#Перебирання статей, які написані цим автором та присвоєння зміним необхідних
даних
 for p in pubs:
 r = p["bib"]
 raw_title = (r["title"])
 try:
 raw_year = (r["pub_year"])
 except KeyError:
 raw_year = "1111"
#Перевірка на наявність даних про цю статтю у базі
 art_title_check = None
 art_check = (
 Article.select(Article.title).where((Article.title == raw_title) &
(Article.authors == raw_authors)))
 for r in art_check:
 art_title_check = r
#Занесення у разі відсутності записів
 if (art_title_check == None):
 record_articles = Article.create(title=raw_title,
 authors=raw_authors,
 pub_date=raw_year,
 categories=raw_category)
 record_articles.save()
 db.commit()
#Перевірка на наявність записів про категорію
 cat_check = None
 aut_check = (Category.select(Category.cat_name).where(Category.cat_name ==
raw_category))
```

```
 for r in aut_check:
 cat_check = r
#Занесення у разі відсутності записів
 if (cat_check == None):
 record_cat = Category.create(cat_name=raw_category)
 record_cat.save()
#Перевірка на існування записів про зібрану та вже наявну інформацію в базі даних
та присвоєння їм унікальних номерів(ID)
 query_art_id_extr = (Article.select(Article.art_id).where(Article.title == raw_title))
 for id in query_art_id_extr:
 raw_art_id = id.art_id
 query_auth_id_extr = (Author.select(Author.author_id).where(Author.name == raw_authors))
 for id in query_auth_id_extr:
 raw_auth_id = id.author_id
 query_auth_id_extr = (Category.select(Category.cat_id).where(Category.cat_name == raw_category))
 for id in query_auth_id_extr:
 raw_cat_id = id.cat_id
#Створення нових зв’язків, з урахуванням вже наявних у базі даних
 aut_cat_check = None
 aut_cat = (Auth_Cat.select(Auth_Cat.record_id).where(
 (Auth_Cat.author_id == raw_auth_id) & (Auth_Cat.cat_id == raw_cat_id)))
 for r in aut_cat:
 aut_cat_check = r
 if (aut_cat_check == None):
 record_Auth_cat = Auth_Cat.create(author_id=raw_auth_id,
```

```
cat_id=raw_cat_id)
 record_Auth_cat.save()
 art_cat_check = None
 art_cat = (Art_Cat.select(Art_Cat.record_id).where(
 (Art_Cat.art_id == raw_art_id) & (Art_Cat.cat_id == raw_cat_id)))
 for r in art_cat:
 art_cat_check = r
 if (art_cat_check == None):
 record_Art_cat = Art_Cat.create(art_id=raw_art_id, cat_id=raw_cat_id)
 record_Art_cat.save()
 auth_art_check = None
 auth_art = (Auth_Art.select(Auth_Art.record_id).where(
 (Auth_Art.art_id == raw_art_id) & (Auth_Art.author_id == raw_auth_id)))
 for r in auth_art:
 auth_art_check = r
 if (auth_art_check == None):
 record_Auth_Art = Auth_Art.create(author_id=raw_auth_id,
art_id=raw_art_id)
 record_Auth_Art.save()
```

Файл **op_arxiv.py** – файл відповідає за усі пошукові операції для платформи ArXiv

```
# імпортування модулів
import arxiv
# Імпортування моделей головних таблиць
from Models.mod_article import Article
from Models.mod_authors import Author
from Models.mod_categories import Category
#Імпортування моделей допоміжних таблиць
```

```
from Models.mod_auth_cat import Auth_Cat
from Models.mod_art_cat import Art_Cat
from Models.mod_con_art_auth import Auth_Art
#Імпортування конфігураційних даних
from database_con.db_config import db
#Загальна функція пошуку за категоріями та ключовими словами або назвою статті
def searcher(parametr,search_amount,search_category):
 res_count = 0
 platform = "Cornell University Library (arXiv) article: "
 searching = search_category + parametr
 search = arxiv.Search(
 query = searching,
 max_results = search_amount,
 sort_by=arxiv.SortCriterion.Relevance
 )
 for result in search.results():
 res_count += 1
 authors_list = []
 raw_title = result.title
 raw_year = result.published
 raw_year = raw_year.year
 raw_category = result.primary_category
 for author in result.authors:
 authors_list.append(str(author))
 raw_authors = ', '.join(authors_list)
 raw_abstract = result.summary
 art_title_check = None
 art_check = (Article.select(Article.title).where((Article.title == raw_title) &
```

```
(Article.authors == raw_authors)))
 for r in art_check:
 art_title_check = r
 if (art_title_check == None):
 record_articles = Article.create(title=raw_title,
 authors=raw_authors,
 abstract= raw_abstract,
 pub_date=raw_year,
 categories=raw_category)
 record_articles.save()
 db.commit()
 for authors in authors_list:
 info = platform + result.entry_id
 author_check = None
 aut_check = (Author.select(Author.name).where(Author.name == authors))
 for r in aut_check:
 author_check = r
 if (author_check == None):
 record_author = Author.create(name=authors, links=info)
 record_author.save()
 cat_check = None
 cat_recheck = (Category.select(Category.cat_name).where(Category.cat_name ==
raw_category))
 for r in cat_recheck:
 cat_check = r
 if (cat_check == None):
 record_cat = Category.create(cat_name=raw_category)
 record_cat.save()
 for authors in authors_list:
```

```
 query_art_id_extr = (Article.select(Article.art_id).where(Article.title == raw_title))
 for id in query_art_id_extr:
 raw_art_id = id.art_id
 query_auth_id_extr = (Author.select(Author.author_id).where(Author.name ==
authors))
 for id in query_auth_id_extr:
 raw_auth_id = id.author_id
 query_auth_id_extr = (Category.select(Category.cat_id).where(Category.cat_name
== raw_category))
 for id in query_auth_id_extr:
 raw_cat_id = id.cat_id
 aut_cat_check = None
 aut_cat = (Auth_Cat.select(Auth_Cat.record_id).where((Auth_Cat.author_id ==
raw_auth_id) & (Auth_Cat.cat_id == raw_cat_id)))
 for r in aut_cat:
 aut_cat_check = r
 if (aut_cat_check == None):
 record_Auth_cat = Auth_Cat.create(author_id=raw_auth_id, cat_id=raw_cat_id)
 record_Auth_cat.save()
 art_cat_check = None
 art_cat = (Art_Cat.select(Art_Cat.record_id).where((Art_Cat.art_id == raw_art_id)
& (Art_Cat.cat_id == raw_cat_id)))
 for r in art_cat:
 art_cat_check = r
 if (art_cat_check == None):
 record_Art_cat = Art_Cat.create(art_id=raw_art_id, cat_id=raw_cat_id)
 record_Art_cat.save()
 auth_art_check = None
 auth_art = (Auth_Art.select(Auth_Art.record_id).where((Auth_Art.art_id ==
```

```
raw_art_id) & (Auth_Art.author_id == raw_auth_id)))
 for r in auth_art:
 auth_art_check = r
 if (auth_art_check == None):
 record_Auth_Art = Auth_Art.create(author_id=raw_auth_id, art_id=raw_art_id)
 record_Auth_Art.save()
 return(res_count)
# функція яка визиває пошук та надає параметр (назва статті або ключові слова)
def searched_by_title_func(parametr,search_amount):
 search_category = "ti:"
 res_count =searcher(parametr,search_amount,search_category)
 return (res_count)
# функція яка визиває пошук та надає параметр (категорії)
def searched_by_category(parametr,search_amount):
 search_category = "cat:"
 res_count = searcher(parametr, search_amount, search_category)
 return (res_count)
#функція яка здійснює пошук за автором статті
def searcher_authors(parametr,search_amount):
 res_count = 0
 platform = "Cornell University Library (arXiv) article: "
 search_category = "au:"
 searching = search_category + parametr
 search = arxiv.Search(
 query = searching,
 max_results = search_amount,
 sort_by=arxiv.SortCriterion.Relevance
 )
 for result in search.results():
```

```
res_count += 1
authors_list = []
raw_title = result.title
raw_year = result.published
raw_year = raw_year.year
raw_category = result.primary_category
for author in result.authors:
authors_list.append(str(author))
raw_authors = ', '.join(authors_list)
raw_abstract = result.summary
art_title_check = None
art_check = (Article.select(Article.title).where((Article.title == raw_title) &
(Article.authors == raw_authors)))
for r in art_check:
art_title_check = r
if (art_title_check == None):
record_articles = Article.create(title=raw_title,
authors=raw_authors,
abstract= raw_abstract,
pub_date=raw_year,
categories=raw_category)
record_articles.save()
for authors in authors_list:
info = platform + result.entry_id
author_check = None
aut_check = (Author.select(Author.name).where(Author.name == authors))
for r in aut_check:
author_check = r
if (author_check == None):
```

```
 record_author = Author.create(name=authors, links=info)
 record_author.save()
 cat_check = None
 aut_check = (Category.select(Category.cat_name).where(Category.cat_name ==
raw_category))
 for r in aut_check:
 cat_check = r
 if (cat_check == None):
 record_cat = Category.create(cat_name=raw_category)
 record_cat.save()
 for authors in authors_list:
 query_art_id_extr = (Article.select(Article.art_id).where(Article.title == raw_title))
 for id in query_art_id_extr:
 raw_art_id = id.art_id
 query_auth_id_extr = (Author.select(Author.author_id).where(Author.name ==
authors))
 for id in query_auth_id_extr:
 raw_auth_id = id.author_id
 query_auth_id_extr = (Category.select(Category.cat_id).where(Category.cat_name
== raw_category))
 for id in query_auth_id_extr:
 raw_cat_id = id.cat_id
 aut_cat_check = None
 aut_cat = (Auth_Cat.select(Auth_Cat.record_id).where((Auth_Cat.author_id ==
raw_auth_id) & (Auth_Cat.cat_id == raw_cat_id)))
 for r in aut_cat:
 aut_cat_check = r
 if (aut_cat_check == None):
 record_Auth_cat = Auth_Cat.create(author_id=raw_auth_id, cat_id=raw_cat_id)
```

```
 record_Auth_cat.save()
 art_cat_check = None
 art_cat = (Art_Cat.select(Art_Cat.record_id).where((Art_Cat.art_id == raw_art_id)
& (Art_Cat.cat_id == raw_cat_id)))
 for r in art_cat:
 art_cat_check = r
 if (art_cat_check == None):
 record_Art_cat = Art_Cat.create(art_id=raw_art_id, cat_id=raw_cat_id)
 record_Art_cat.save()
 auth_art_check = None
 auth_art = (Auth_Art.select(Auth_Art.record_id).where((Auth_Art.art_id ==
raw_art_id) & (Auth_Art.author_id == raw_auth_id)))
 for r in auth_art:
 auth_art_check = r
 if (auth_art_check == None):
 record_Auth_Art = Auth_Art.create(author_id=raw_auth_id, art_id=raw_art_id)
 record_Auth_Art.save()
 return (res_count)
```

Файл **db_simple_op** – Зберігає функції для роботи з базою даних через ORM такі як:
підключення , зберігання даних, відключення від неї, створення нової бази даних

```
#Підключення моделей усіх таблиць
from Models.mod_auth_cat import Auth_Cat
from Models.mod_con_art_auth import Auth_Art
from Models.mod_article import Article
from Models.mod_authors import Author
from Models.mod_categories import Category
from Models.mod_art_cat import Art_Cat
```

```
from database_con.db_config import db
#Функція створення нової бази даних та підключення до неї
def create_new_database():
 db.connect()
 db.create_tables([Article, Author, Category, Auth_Art, Art_Cat, Auth_Cat])
 db.commit()
 db.close()
#Функція підключення до існуючої бази даних
def connect_to_database():
 db.connect()
 print('connected')
#Функція збереження змін у базі даних
def commit_database():
 db.commit()
 print('changes was saved')
#функція відключення бази даних
def disconect_database():
 print('exit')
 db.close()
```

Файл **main.py** – зберігає усі сценарії , які будуть виконуватись при виклику за відповідними кнопками у інтерфейсі

```
#Імпортування модулів та файлів
import os
import shutil
from datetime import datetime
import op_arxiv
import op_scholar
from database_con.db_simple_op import *
```

```
from database_con.db_config import db_standart_path , db_record_path
,db_session_path
#Функція , яка перевіряє чи існує база даних у відповідній папці, якщо ні то
створює
автоматично
def db_checker():
 path = db_standart_path
 if (os.path.exists(path)):
 pass
 else:
 if (os.path.exists('db')):
 create_new_database()
 else:
 os.mkdir('db')
 create_new_database()
#Функція, що зберігає копію бази даних під ім’ям що задав користувач, або
використовує шаблон: record_час зберігання копії.db
def record_saver(file_name):
 extension = ".db"
 if(len(file_name) < 3):
 record_time = datetime.now().strftime('%d_%m_%Y_%H_%M_%S')
 record = "Record-"
 timer = str(record_time)
 new_record_path = db_record_path + record + timer + extension
 else:
 record = file_name
 new_record_path = db_record_path + record + extension
 file_oldname = os.path.join(db_standart_path)
 file_newname_newfile = os.path.join(new_record_path)
```

```
 shutil.copy(file_oldname, file_newname_newfile)
#Функція яка зберігає копію бази даних при коректному виході з програми. Копія зберігається як session_часвиходу.db
def session_saver():
 record_time = datetime.now().strftime('%d_%m_%Y_%H_%M_%S')
 timer = str(record_time)
 extension = ".db"
 new_record_path = db_session_path + timer + extension
 file_oldname = os.path.join(db_standart_path)
 file_newname_newfile = os.path.join(new_record_path)
 shutil.copy(file_oldname, file_newname_newfile)
#Функція , що викликає пошук за назвою статті на платформі ArXiv
def searched_by_title_arx(user_parametr,user_max_res):
 parametr = user_parametr
 search_amount = user_max_res
 res_arx_count = op_arxiv.searched_by_title_func(parametr, search_amount)
 print(res_arx_count)
 commit_database()
#Функція , що викликає пошук за назвою статті на платформі Google Scholar
def searched_by_title_sch(user_parametr,user_max_res):
 parametr = user_parametr
 search_amount = user_max_res
 res_sch_count = op_scholar.search_by_keyword(parametr, search_amount)
 print(res_sch_count)
 commit_database()
#Функція , що викликає пошук за автором на платформі ArXiv
def searched_by_author_arx(user_parametr,user_max_res):
 parametr= user_parametr
 search_amount = user_max_res
```

```
 res_arx_count = op_arxiv.searcher_authors(parametr,search_amount)
 print(res_arx_count)
 commit_database()
#Функція , що викликає пошук за автором на платформі Google Scholar
def searched_by_author_sch(user_parametr,user_max_res):
 parametr = user_parametr
 op_scholar.author_searcher(parametr,user_max_res)
 #op_scholar.search_author(parametr)
 commit_database()
#Функція , що викликає пошук за категорією на платформі ArXiv
def searched_by_category_arx(user_parametr,user_max_res):
 parametr= user_parametr
 search_amount = user_max_res
 res_arx_count = op_arxiv.searched_by_category(parametr, search_amount)
 print(res_arx_count)
 commit_database()
```

Файл pyqt.py – Файл, який запускає інтерпретатор, має в собі клас, який наслідує прописаний інтерфейс. Має прописані функції, які будуть визивати відповідні сценарії при взаємодії користувача з інтерфейсом

```
#Імпорт модулів та класу, в якому знаходиться прототип інтерфейсу
import table_model
from main import *
import sys
from PyQt5 import QtWidgets
from PyQt5.QtWidgets import QMessageBox, qApp, QHeaderView, QInputDialog
import Menu
#Клас , який наслідує прототип інтерфейсу. Пов’язує виклик відповідних функцій при взаємодії користувача з інтерфейсом
class Form(QtWidgets.QMainWindow, Menu.Ui_MainWindow):
```

```
def __init__(self):
db_checker()
connect_to_database()
super(Form, self).__init__()
self.setupUi(self)
self.Search_button.clicked.connect(self.inputdata)
self.Save_session.clicked.connect(self.save_session)
self.Save_record.clicked.connect(self.save_record)
self.Refresh_info.clicked.connect(self.based)
self.based()
self.categories = {…}
self.Category_box.addItems(self.categories.keys())
self.Search_by_box.currentTextChanged.connect(self.categ_ch
#Функція , яка зчитує дані, що знаходяться у інтерфейсі під час її виклику.
Перевіряє внесені дані та викликає відповідний сценарій
def inputdata(self):
self.Save_record.setEnabled(False)
self.Save_session.setEnabled(False)
self.Search_button.setEnabled(False)
try:
if(self.Parametr_line.text() == ""):
parameter = "No"
else:
parameter = self.Parametr_line.text()
if (self.Category_box.currentText() == ""):
category = "Nothing"
else:
category = self.categories[str(self.Category_box.currentText())]
max_res = int(self.Max_results_box.value())
```

```
 search_by = self.Search_by_box.currentText()
 qApp.processEvents()
 if not self.CheckBox_Arxiv.isChecked() and not
self.Check_box_goo_sch.isChecked():
 error = QMessageBox()
 error.setText("resourse wasnt picked at all")
 error.setStandardButtons(QMessageBox.Ok)
 error.exec()
 elif self.CheckBox_Arxiv.isChecked() and not
self.Check_box_goo_sch.isChecked():
 if search_by == "Title":
 if (parameter == "No") or (len(parameter) < 3):
 error = QMessageBox()
 error.setText("Parameter has no data")
 error.setStandardButtons(QMessageBox.Ok)
 error.exec()
 else:
 error = QMessageBox()
 error.setText("Search has been finished")
 error.setStandardButtons(QMessageBox.Ok)
 searched_by_title_arx(parameter, max_res)
 error.exec()
 elif search_by == "Category":
 if (category == "Nothing"):
 error = QMessageBox()
 error.setText("Category wasnt picked")
 error.setStandardButtons(QMessageBox.Ok)
 error.exec()
 else:
```

```
error = QMessageBox()
error.setText("Search has been finished")
error.setStandardButtons(QMessageBox.Ok)
searched_by_category_arx(category, max_res)
error.exec()
elif search_by == "Author":
if (parameter == "No") or (len(parameter) < 3):
error = QMessageBox()
error.setText("Parameter has no data")
error.setStandardButtons(QMessageBox.Ok)
error.exec()
else:
error = QMessageBox()
error.setText("Search has been finished")
error.setStandardButtons(QMessageBox.Ok)
searched_by_author_arx(parameter, max_res)
error.exec()
else:
error = QMessageBox()
error.setText("Search_by wasn`t picked")
error.setStandardButtons(QMessageBox.Ok)
error.exec()
elif not self.CheckBox_Arxiv.isChecked() and
self.Check_box_goo_sch.isChecked():
if search_by == "Title":
if (parameter == "No") or (len(parameter) < 3):
error = QMessageBox()
error.setText("Parameter has no data")
error.setStandardButtons(QMessageBox.Ok)
```

```
error.exec()
else:
error = QMessageBox()
error.setText("Search has been finished")
error.setStandardButtons(QMessageBox.Ok)
searched_by_title_sch(parameter, max_res)
error.exec()
elif search_by == "Author":
if (parameter == "No") or (len(parameter) < 3):
error = QMessageBox()
error.setText("Parameter has no data")
error.setStandardButtons(QMessageBox.Ok)
error.exec()
else:
error = QMessageBox()
error.setText("Search has been finished")
error.setStandardButtons(QMessageBox.Ok)
searched_by_author_sch(parameter, max_res)
error.exec()
else:
error = QMessageBox()
error.setText("Search_by wasn`t picked")
error.setStandardButtons(QMessageBox.Ok)
error.exec()
elif self.CheckBox_Arxiv.isChecked() and self.Check_box_goo_sch.isChecked():
if search_by == "Title":
if (parameter == "No") or (len(parameter) < 3):
error = QMessageBox()
error.setText("Parameter has no data")
```

```
error.setStandardButtons(QMessageBox.Ok)
error.exec()
else:
error = QMessageBox()
error.setText("Search has been finished")
error.setStandardButtons(QMessageBox.Ok)
searched_by_title_arx(parameter, max_res)
searched_by_title_sch(parameter, max_res)
error.exec()
elif search_by == "Author":
if (parameter == "No") or (len(parameter) < 3):
error = QMessageBox()
error.setText("Parameter has no data")
error.setStandardButtons(QMessageBox.Ok)
error.exec()
else:
error = QMessageBox()
error.setText("Search has been finished")
error.setStandardButtons(QMessageBox.Ok)
searched_by_author_sch(parameter, max_res)
searched_by_author_arx(parameter, max_res)
error.exec()
elif search_by == "Category":
if (category == "Nothing"):
error = QMessageBox()
error.setText("Category wasnt picked")
error.setStandardButtons(QMessageBox.Ok)
error.exec()
else:
```

```
 error = QMessageBox()
 error.setText("Search has been finished")
 error.setStandardButtons(QMessageBox.Ok)
 searched_by_category_arx(category, max_res)
 error.exec()
 else:
 error = QMessageBox()
 error.setText("Search_by wasn`t picked")
 error.setStandardButtons(QMessageBox.Ok)
 error.exec()
 except Exception as err:
 print(err)
 self.Save_session.setEnabled(True)
 self.Save_record.setEnabled(True)
 self.Search_button.setEnabled(True)
#Функція, що викликає сценарій збереження сесії кориистувача
 def save_session(self):
 self.Save_record.setEnabled(False)
 self.Save_session.setEnabled(False)
 self.Search_button.setEnabled(False)
 session_saver()
 disconect_database()
 App.exit(0)
#Функція , що викликає сценарій збереження запису користувачем
 def save_record(self):
 self.Save_record.setEnabled(False)
 self.Save_session.setEnabled(False)
 self.Search_button.setEnabled(False)
 text, ok = QInputDialog.getText(self,'Name your saving file', 'Name of your record')
```

```
 if ok:
 file_name = str(text)
 record_saver(file_name)
 self.Save_session.setEnabled(True)
 self.Save_record.setEnabled(True)
 self.Search_button.setEnabled(True)
 def based(self):
self.tableView.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)
 articles = Article.select().tuples()
 model = table_model.MyTableModel(articles)
 self.tableView.setModel(model)
#Функція, яка проводить зміни у інтерфейсі, в залежності від обраних параметрів
 def categ_ch(self):
 if(self.Search_by_box.currentText() == "Title"):
 self.Label_Searching_parametr.show()
 self.Parametr_line.show()
 self.Label_category.hide()
 self.Category_box.hide()
 self.Check_box_goo_sch.show()
 elif(self.Search_by_box.currentText() == "Author"):
 self.Label_Searching_parametr.show()
 self.Parametr_line.show()
 self.Label_category.hide()
 self.Category_box.hide()
 self.Check_box_goo_sch.show()
 elif (self.Search_by_box.currentText() == "Category"):
 self.Category_box.show()
 self.Label_category.show()
 self.Label_Searching_parametr.hide()
```

```
 self.Parametr_line.hide()
 self.Check_box_goo_sch.hide()
 else:
 self.Label_Searching_parametr.show()
 self.Label_category.show()
 self.Label_Search_by.show()
 self.Check_box_goo_sch.show()
 self.Parametr_line.show()
 self.Category_box.show()
 self.Check_box_goo_sch.show()
#Функції , що відкривають інтерфейс при запуску програми
App = QtWidgets.QApplication(sys.argv)
window = Form()
window.show()
App.exec()
```

## ДОДАТОК Б Презентація


НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Кафедра Інженерії програмного забезпечення в енергетиці.

**ТЕМА: «**Збір та первинна обробка даних наукової діяльності з різнорідних джерел »

Виконав: студент магістерського рівня 2-го року навчання, групи ТВ-22мп

Василенко Микита Андрійович

Керівник: к.т.н., доцент, Кузьміних В.О.



## МЕТА РОБОТИ

- **Метою даної магістерської роботи** є проведення аналізу наукової діяльності організацій та окремих науковців, задля подальшого ефективного розподілу ресурсів(зокрема фінансових) компаній. Для вирішення даного питання необхідна розробка методів та підходів проведення збору інформації про наукові статті, матеріали конференцій та препринти, які зберігаються у спеціалізованих джерелах.

## АКТУАЛЬНІСТЬ ТЕМИ

Система призначена для використання аналітиками наукової діяльності, завдяки яким фінансуючі фірми можуть більш доцільно розподіляти бюджет на фінансування саме тих компаній, які мають більшу перспективу.
Завдяки автоматизованому збору необхідної інформації за обраною тематикою діяльність аналітиків стає більш продуктивною та з'являється можливість проаналізувати більшу кількість інформації.
На даний час не є доступні системи, які б охоплювали велику кількість джерел специфічних джерел та об`єднувати їх в загальній базі даних.



## ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Об'єктами дослідження** є наукові твори, дисертації, тощо – які є результатами досліджень(діяльності) науковців.
- **Предмет дослідження** є методи обробки та вилучення інформації з джерел.

## ЦІЛЬОВА АУДИТОРІЯ

Система призначена для використання:
- науковцями;
- аналітиками наукової діяльності;
- бібліографічними установами;
- архівними установами;
- окремими дослідниками;
- студентами та викладачами;
- фінансистами.



## ЗАДАЧІ ПО РОЗРОБЦІ ПЗ

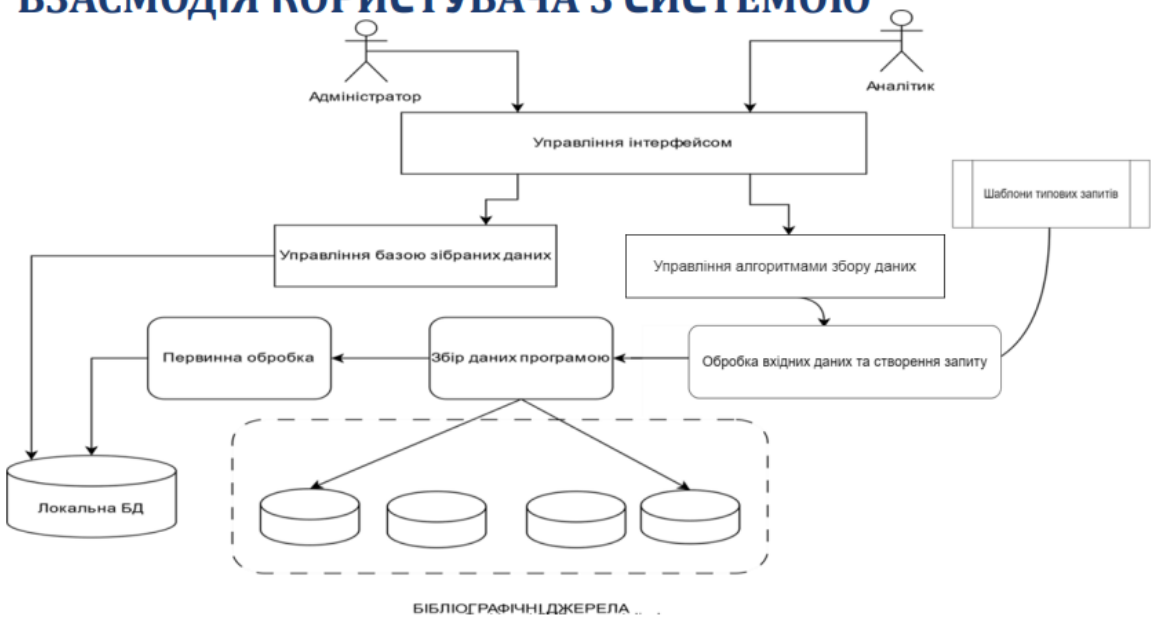
Для вирішення поставленої задачі розробнику необхідно:
1. Провести аналіз існуючих бази даних, які містять необхідну інформацію.
2. Провести аналіз та обрати алгоритм вилучення інформації з баз даних.
3. Реалізувати алгоритм вилучення інформації з обраних джерел.
4. Реалізувати перевірку коректності вводу даних, збереження отриманих даних, вилучення дублікатів.
5. Реалізувати алгоритм порівняння існуючої інформації з новою у базі даних та оновлення інформації за необхідністю.
6. Розробити програмне забезпечення, що здатне виконувати процес збору та збереження даних

## ПРАКТИЧНА ЗНАЧИМІСТЬ

- Розроблений програмний продукт може бути використаний науковими організаціями для спрощення взаємодії з існуючими відкритими системами та автоматизації оцінки міжнародної науково-технічної діяльності. За належного залучення ресурсів та за підтримки системи, ідея програмної платформи може розвиватися та отримати великий інтерес серед наукових організацій.



## ВЗАЄМОДІЯ КОРИСТУВАЧА З СИСТЕМОЮ



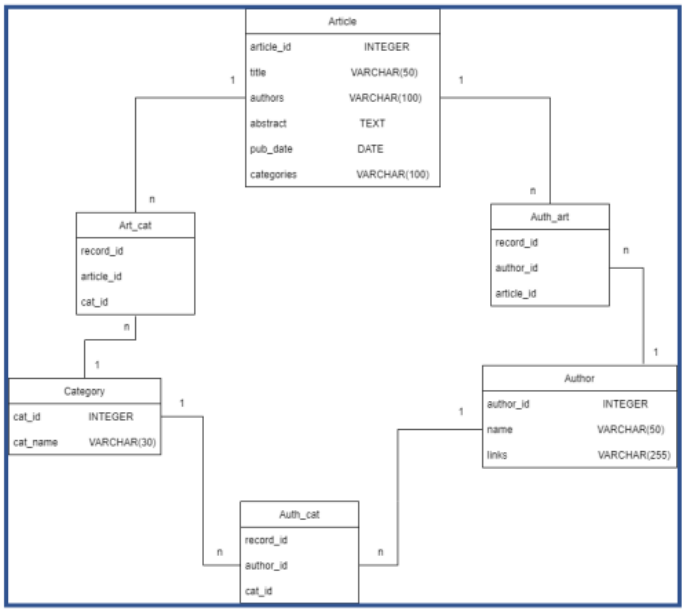

БІБЛІОГРАФІЧНІ ДЖЕРЕЛА

## ЗАГАЛЬНА СТРУКТУРА БД

Складається з трьох основних та трьох домоміжних таблиць

Основні таблиці зберігають усю інформацію про статті

Допоміжні таблиці зберігають унікальні ID для створення зв'язків між даними






## ОБРАНІ ПЛАТФОРМИ

ArXiv(Архів) – база даних створена на основі бібліотеки Корнелльського університету(Cornell University), де зберігається велика кількість електронних публікацій: наукових статей та їх препринтів, рефератів та дисертацій

Google Академія – є наукометричною база даних з відкритим доступом, яку було створено найпотужнішою пошуковою системою Google.

На даний момент Google Академія має найбільшу кількість наукової літератури, а звідси й охоплює найбільшу кількість галузей досліджень. Станом на 2016 рік її база охоплювала 160 мільйонів документів з унікальним значенням. Цей показник в декілька разів більший за конкуруючих з нею баз даних Scopus та Web of Science.

# СИСТЕМИ-АНАЛОГИ

AMiner.org (ArnetMiner) — онлайн-сервіс для академічного аналізу соціальних мереж і майнінгу. Система зібрала інформацію про понад 136 000 000 дослідників і 230 000 000 публікацій, 80 000 конференцій. Система працює в Інтернеті з 2006 року, її відвідали майже 7 320 000 незалежних IP-доступів, понад 300 000 переглядів сторінок на день і користувачі з більш ніж 220 країн/регіонів


WorldWideScience.org — це глобальна наукова пошукова система, призначена для прискорення наукових відкриттів і прогресу шляхом прискорення обміну науковими знаннями. Завдяки багатосторонньому партнерству WorldWideScience.org дозволяє будь-кому, хто має доступ до Інтернету, запустити пошук за одним запитом у національних наукових базах даних і порталах у понад 70 країнах, охоплюючи всі населені континенти світу та понад три чверті населення світу. Головною особливістю є доступ до "глибокої мережі", що дає змогу отримати інформацію, яка прихована від звичайного інтернет користувача.



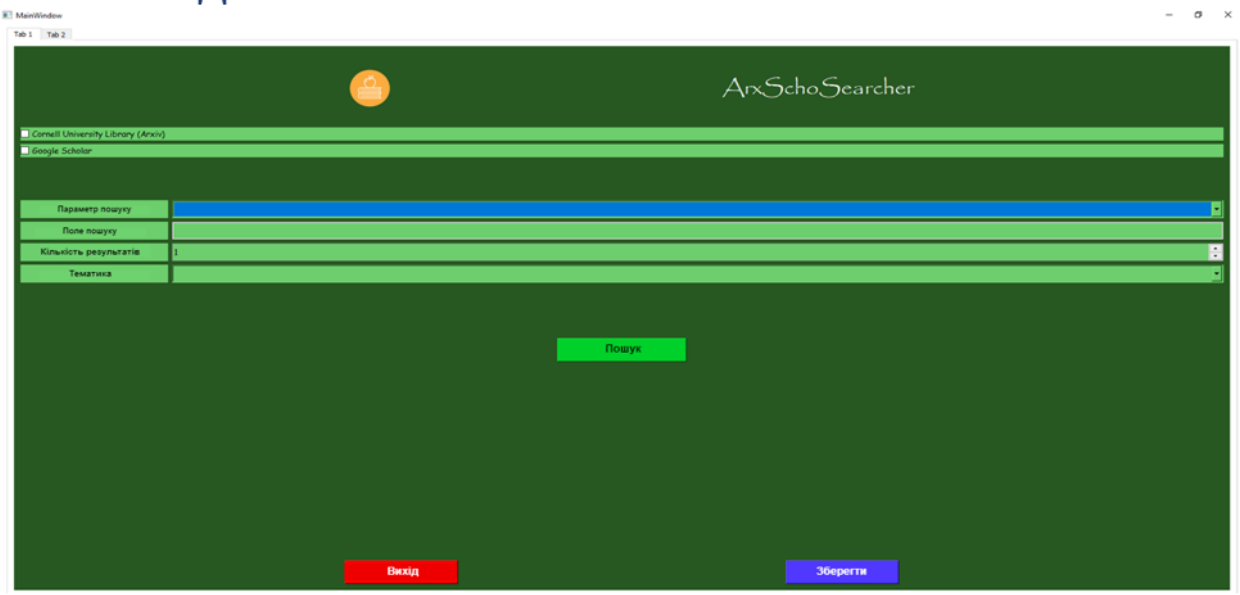
# РЕАЛІЗОВАНИЙ АЛГОРИТМ ПАРСИНГУ

Процес API-парсингу зазвичай включає наступні кроки:

1) отримання API-ключа (якщо необхідно);
2) використання бібліотеки або створення HTML запиту;
3) відправлення запросу до API;
4) обробка відповіді від API;
5) вилучення даних;

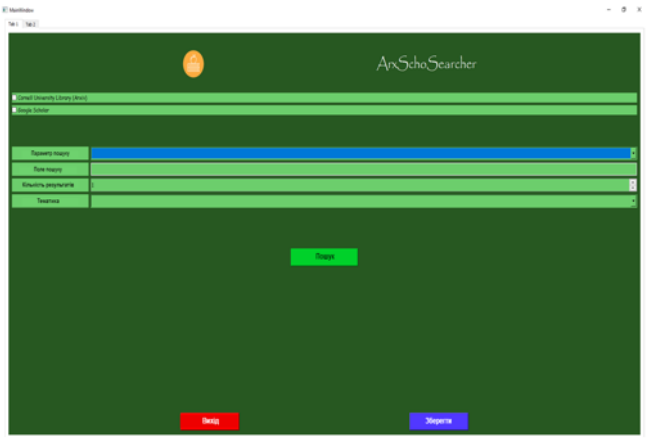

- API-парсинг, або робота із зовнішніми API (Application Programming Interface), є процесом взаємодії з веб-службами для отримання даних.

# ПРИКЛАД ІНТЕРФЕЙСУ КОРИСТУВАЧА





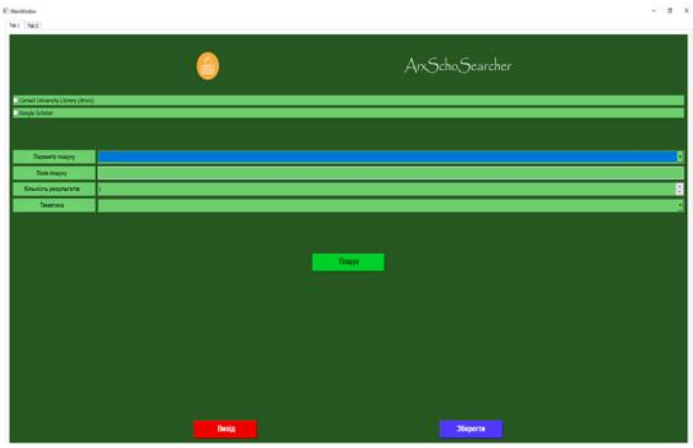

# ПОЛЯ ПОШУКУ





Головними елементами є поля для вхідної інформації від користувача, де є можливість обрати:
- Ресурси за якими буде здійснено пошук;
- Параметр(критерій) пошуку;
- Поле пошуку;
- Кількість результатів;
- Тематика;

# КРИТЕРІЇ ПОШУКУ

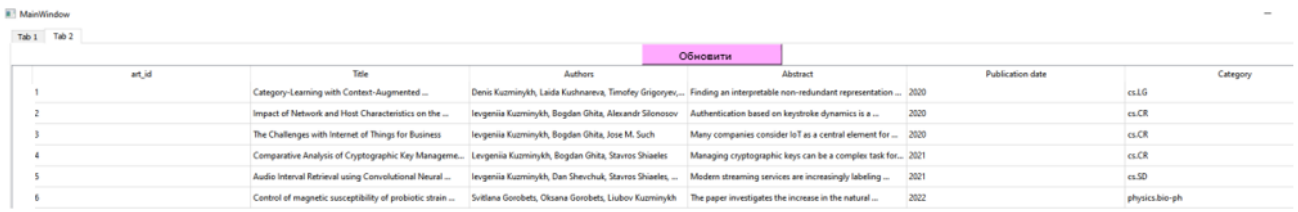



Назва статті
Автор
Напрямок

Система здійснює пошук за трьома критеріями:
- Пошук за назвою статті
- Пошук за прізвищем автора
- Пошук за тематикою



# ПРИКЛАД ВІДОБРАЖЕННЯ ЗІБРАНОЇ ІНФОРМАЦІЇ




| art_id | Title | Authors | Abstract | Publication date | Category |
|---|---|---|---|---|---|
| 1 | Category-Learning with Context-Augmented ... | Denis Kuzminykh, Laida Kushnareva, Timofey Grigoryev,... | Finding an interpretable non-redundant representation ... | 2020 | cs.LG |
| 2 | Impact of Network and Host Characteristics on the ... | Ievgeniia Kuzminykh, Bogdan Ghita, Alexandr Silonosov | Authentication based on keystroke dynamics is a ... | 2020 | cs.CR |
| 3 | The Challenges with Internet of Things for Business | Ievgeniia Kuzminykh, Bogdan Ghita, Jose M. Such | Many companies consider IoT as a central element for ... | 2020 | cs.CR |
| 4 | Comparative Analysis of Cryptographic Key Manageme... | Levgeniia Kuzminykh, Bogdan Ghita, Stavros Shiaeles | Managing cryptographic keys can be a complex task for... | 2021 | cs.CR |
| 5 | Audio Interval Retrieval using Convolutional Neural ... | Ievgeniia Kuzminykh, Dan Shevchuk, Stavros Shiaeles, ... | Modern streaming services are increasingly labeling ... | 2021 | cs.SD |
| 6 | Control of magnetic susceptibility of probiotic strain ... | Svitlana Gorobets, Oksana Gorobets, Liubov Kuzminykh | The paper investigates the increase in the natural ... | 2022 | physics.bio-ph |

# ОЧІКУВАНІ РЕЗУЛЬТАТИ

В ході розробки системи було:
1. Проаналізовано існуючі бази даних, які містять необхідну інформацію.
2. Проаналізовано та обрано алгоритм вилучення інформації з баз даних.
3. Реалізовано алгоритм вилучення інформації з обраних джерел.
4. Реалізовано перевірка коректності вводу даних, збереження отриманих даних, вилучення дублікатів.
5. Реалізовано алгоритм порівняння існуючої інформації з новою у базі даних та оновлення інформації за необхідністю.
6. Розроблено програмне забезпечення, що здатне виконувати процес збору та збереження даних



Національний технічний університет України
«Київський політехнічний інститутімені Ігоря Сікорського»

National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute"