Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем та програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка та тестування вебзастосунку для інтелектуального аналізу наукових публікацій з використанням NCBI Entrez API

Виконала: студентка 4 курсу, групи СП-41
спеціальності 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

(підпис) Валігура І. В. (прізвище та ініціали)

Керівник (підпис) Багрій-Заяць О. А. (прізвище та ініціали)

Нормоконтроль (підпис) Стоянов Ю. М. (прізвище та ініціали)

Завідувач кафедри (підпис) Петрик М. Р. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем та програмної інженерії
(повна назва факультету)

Кафедра програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ _____
(підпис) (прізвище та ініціали)
« » 2026 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр
(назва освітнього ступеня)

за спеціальністю 121 Інженерія програмного забезпечення
(шифр і назва спеціальності)

студентці Валігурі Ірині Володимирівні
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка та тестування вебзастосунку для інтелектуального аналізу наукових публікацій з використанням NCBI Entrez API

Керівник роботи Багрій-Заяць Оксана Андріївна, канд. техн. наук, доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «___» _____ 2026 року № _____

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи наукові літературні джерела, масиви наукових публікацій із бази даних PubMed

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ, аналіз предметної області, аналогів та формування вимог до системи, проєктування та конструювання, тестування та верифікація, безпека життєдіяльності, основи охорони праці, висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
Діаграма варіантів використання, діаграма послідовності, діаграма класів, блок-схема алгоритму семантичного аналізу, графічний інтерфейс користувача, слайди презентації

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
БЖД і ООП			

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Вибір напряму дослідження та погодження теми з керівником		
2	Складання та затвердження індивідуального завдання		
3	Аналіз предметної області, існуючих рішень та вимог до вебзастосунку		
4	Проєктування та конструювання вебзастосунку		
5	Тестування системи, апробація та оцінка ефективності функціонування		
6	Виконання розділу "Безпека життєдіяльності та охорона праці"		
7	Оформлення пояснювальної записки		
8	Попередній захист на кафедрі та перевірка на антиплагіат		
9	Отримання відгуку керівника, рецензії та проходження нормоконтролю		
10	Захист кваліфікаційної роботи бакалавра		

Студент _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Валігура І. В. Розробка та тестування вебзастосунку для інтелектуального аналізу наукових публікацій з використанням NCBI Entrez API : робота на здобуття освітнього ступеня бакалавра : спец. 121 – інженерія програмного забезпечення / наук. кер. О. А. Багрій-Заяць. Тернопіль : Тернопільський національний технічний університет імені Івана Пулюя, 2026. 63 с. // С. 59, табл. 11, рис. 6, бібліогр. 23, слайдів 15, додат. 3.

Ключові слова: вебзастосунок, NCBI Entrez API, інтелектуальний аналіз, наукові публікації, PubMed, доказова медицина, Python.

Метою роботи є розробка вебзастосунку для автоматизованого збору, інтелектуального аналізу та математичного ранжування публікацій PubMed за рівнями доказовості.

У першому розділі проаналізовано предметну область, проведено порівняльний аналіз аналогів та сформульовано вимоги до системи.

У другому розділі обґрунтовано вибір ітераційної моделі розробки, спроєктовано клієнт-серверну архітектуру, схему бази даних SQLite та UML-моделі системи.

У третьому розділі описано програмну реалізацію модулів інтеграції та аналізу, наведено результати комплексного тестування, верифікації швидкодії та оцінки ефективності застосунку.

У четвертому розділі розглянуто питання забезпечення безпечної експлуатації обчислювальної техніки та алгоритми надання домедичної допомоги.

Об’єктом дослідження є процеси автоматизованого збору, фільтрації та інтелектуального аналізу масивів наукових даних.

Предметом дослідження є методи взаємодії з NCBI Entrez API, асинхронна архітектура вебзастосунку, алгоритми семантичного аналізу та модель математичного ранжування статей. Методи дослідження включають: аналіз предметної області, проєктування архітектури UML, алгоритмізацію NLP-завдань, модульне та інтеграційне тестування.

ABSTRACT

Iryna Valihura. Development and testing of a web application for intelligent analysis of scientific publications using NCBI Entrez API : bachelor thesis : specialty 121 "Software Engineering" : Ternopil Ivan Puluj National Technical University, 2026, 59p. // Pages - 63, tables - 11, figures - 6, references - 23, presentation slides – 15, appendices - 3.

Keywords: web application, NCBI Entrez API, intelligent analysis, scientific publications, PubMed, evidence-based medicine, Python.

The purpose of the work is to develop a web application for automated collection, intelligent analysis, and mathematical ranking of PubMed publications based on levels of evidence.

The first chapter analyzes the domain, provides a comparative analysis of analogs, and formulates the system requirements.

The second chapter justifies the choice of an iterative development model and presents the client-server architecture, SQLite database schema, and system UML models.

The third chapter describes the software implementation of integration and analysis modules, as well as the results of comprehensive testing, performance verification, and application efficiency evaluation.

The fourth chapter covers the issues of ensuring the safe operation of computing equipment and algorithms for providing first aid.

The object of the study is the processes of automated collection, filtering, and intelligent analysis of scientific data arrays.

The subject of the study is methods of interaction with NCBI Entrez API, asynchronous web application architecture, semantic analysis algorithms, and the mathematical article ranking model. Research methods include domain analysis, UML architecture modeling, NLP task algorithmization, as well as unit and integration testing.

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

API – прикладний програмний інтерфейс (Application Programming Interface).

NLP – обробка природної мови (Natural Language Processing).

СКБД – система керування базами даних.

UML – уніфікована мова моделювання (Unified Modeling Language).

XML – розширювана мова розмітки (Extensible Markup Language).

ЗМІСТ

ВСТУП

На сьогодні обсяги наукової інформації зросли настільки, що виникла реальна криза надлишку публікацій. Величезна кількість нових статей призводить до того, що значна частина наукових матеріалів залишається непрочитаною, оскільки фахівці фізично не встигають опрацьовувати такі масиви даних. При використанні програмного інтерфейсу NCBI Entrez API головна проблема полягає у фільтрації отриманих результатів. Стандартні засоби пошуку видають занадто багато інформації, у якій важко автоматично визначити якість джерел та дизайн дослідження.

Технічно обробка відповідей від PubMed ускладнюється сирою структурою XML-пакетів, які містять багато зайвих метаданих, але не класифікують дизайн дослідження. Існуюче програмне забезпечення зазвичай шукає статті лише за ключовими словами і не аналізує сам зміст тексту анотацій. Через це виникає потреба у створенні інструментів для автоматичної класифікації, математичного ранжування та валідації доказовості публікацій безпосередньо під час їх завантаження.

Розробка вебзастосунку, який за допомогою алгоритмів патерн-метчингу аналізує тексти та виконує математичне ранжування за критеріями якості, дозволить автоматизувати цей процес. Застосунок реалізує алгоритми вилучення числових сутностей (таких як обсяг вибірки пацієнтів) та визначення ієрархії доказів для кожної статті.

Метою роботи є розробка вебзастосунку для інтелектуального аналізу, класифікації та математичного ранжування публікацій PubMed за рівнями доказовості на основі текстів анотацій. Для цього вирішено задачі з аналізу предметної області, проєктування асинхронної архітектури, створення модуля інтеграції з NCBI Entrez API, програмної реалізації алгоритму патерн-метчингу та моделі зваженого ранжування, а також експериментальної оцінки швидкодії та точності системи.

Об’єкт дослідження охоплює процеси автоматизованого збору, фільтрації та інтелектуального аналізу великих масивів наукових даних із використанням програмних інтерфейсів.

Предмет дослідження включає методи та програмні засоби взаємодії з NCBI Entrez API, асинхронну архітектуру вебзастосунку, алгоритми патерн-метчингу для аналізу текстів анотацій, а також математичну модель зваженого ранжування статей за рівнями доказовості.

Наукова новизна одержаних результатів полягає в удосконаленні методу автоматизованої класифікації медичних статей за типом дизайну дослідження завдяки адаптації алгоритмів патерн-метчингу до структури анотацій PubMed. Це дозволяє оцінювати та сортувати джерела за рівнями доказовості безпосередньо під час їхнього імпорту через API, що підвищує точність формування аналітичних вибірок.

Практичне значення виконаної роботи полягає у створенні вебзастосунку, який автоматизує пошук наукової літератури та скорочує час на первинний відбір статей. Розроблені модулі аналізу тексту та інтеграції з API можна використовувати як окремі компоненти в інших інформаційних системах чи базах знань.

Результати дослідження опубліковано у тезах доповіді на IX Міжнародній студентській науково-технічній конференції ТНТУ імені Івана Пулюя “Природничі та гуманітарні науки. Актуальні питання” (м. Тернопіль, 2026 р.), матеріали яких представлено у

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІДЕНТИФІКАЦІЯ ВИМОГ ДО ВЕБЗАСТОСУНКУ

У даному розділі проводиться дослідження специфіки збору та обробки біомедичних даних, аналізуються існуючі програмні рішення для роботи з науковими публікаціями, формулюється математична постановка завдання ранжування та визначаються технічні вимоги до вебзастосунку за допомогою моделювання варіантів використання в нотації UML.

1.1 Характеристика предметної області та аналіз існуючих рішень для інтелектуального аналізу наукових публікацій

Предметна область мого дослідження охоплює методи автоматизованого збору, обробки та аналізу текстових біомедичних даних. Головним світовим джерелом цієї інформації обрано PubMed, що представляє собою найбільшу безкоштовну базу даних медичних публікацій, створену Національним центром біотехнологічної інформації (NCBI) на базі Національної медичної бібліотеки США (NLM). На сьогоднішній день вона індексує понад 37 мільйонів наукових статей, анотацій та спеціалізованих журналів, виступаючи фундаментальним базисом для концепції доказової медицини.

Проте сучасна медична спільнота стикнулася з критичним викликом, а саме інформаційним перевантаженням. Щорічно у базі PubMed публікується понад 3 мільйони нових наукових праць. За таких умов практикуючий лікар або науковий співробітник виявляється неспроможним виконати ручний пошук та якісний аналіз релевантної літератури для підтвердження клінічних гіпотез [1].

Ключовим викликом при автоматичній фільтрації такої кількості статей я вважаю точне визначення типу (дизайну) їхнього дослідження, наприклад, рандомізовані клінічні випробування, метааналізи, когортні або систематичні огляди [2]. Оскільки рівень доказовості та наукова вага публікації безпосередньо залежать від обраної авторами методології, у своїй роботі я фокусуюся на

розробці алгоритмів обробки природної мови (NLP) та методів семантичного аналізу текстів анотацій для автоматизації цього процесу.

Для автоматизації збору матеріалів я передбачила інтеграцію з REST-інтерфейсом NCBI Entrez Utilities (E-utilities), взаємодія з яким базується на послідовному виконанні мною двох етапів: пошуку публікацій за параметризований запитом та подальшому завантаженні їхніх метаданих. На основі проведеного аналізу специфікації серверів NCBI я визначила ключові інфраструктурні обмеження [3], серед яких жорсткий ліміт на частоту запитів, що вимагає від мене контролю інтенсивності трафіку для запобігання блокуванню IP-адреси, а також значна структурна варіативність пакетів даних і ризик відсутності окремих блоків, що зумовлює мою вимогу щодо проєктування гнучких і відмовостійких модулів обробки інформації.

Під час роботи над проєктом та написання пояснювальної записки я керувалася вимогами профільної кафедри програмної інженерії [4], а також методологічними засадами посібника до корпусу знань з програмної інженерії SWEBOK [5]. Цей стандарт описує всі етапи життєвого циклу розробки програмного забезпечення, починаючи від аналізу вимог і закінчуючи тестуванням системи. Відповідно до цієї методологіїї, процес виявлення вимог я розпочала з дослідженняя існуючих аналогів. Це допомогло мені чітко визначити межі мого майбутнього вебзастосунку та знайти недоліки у схожих програмах-конкурентах.

Сучасні системи автоматизації роботи з науковими публікаціями я розділила на три основні категорії: спеціалізовані NLP-платформи (RobotReviewer, PubTator), орієнтовані на вилучення сутностей із біомедичних текстів; бібліографічні менеджери (Zotero, Mendeley), призначені виключно для каталогізації посилань без аналітичних функцій; та SaaS-платформи для систематичних оглядів (Rayyan), які є комерційними інструментами, що вимагають ручного кодування статей експертами.

Для детального порівняльного аналізу я обрала найбільш релевантні системи: RobotReviewer та PubTator. Оцінювання здійснювала на основі

характеристик якості моделі ISO/IEC 25010 [6]: здатність до взаємодії (Interoperability) , функціональна придатність (Functional Suitability), підтримка класифікації дизайну досліджень, підтримуваність коду (Maintainability) та можливість оцінки авторитетності джерела публікації. Результати аналізу я звела в таблицю 1.1.

Таблиця 1.1 – Порівняльний аналіз аналогів

Критерій порівняння	**RobotReviewer**	**PubTator**
Збір даних з інших систем (Interoperability)	+	+
Аналіз тексту анотацій (Functional Suitability)	+/-	+/-
Визначення типу дослідження	+	-
Легкість оновлення коду (Maintainability)	-	-
Оцінка якості джерела	-	-

Примітка. Символ «+» — повна підтримка функції, «-» — відсутність підтримки функції, «+/-» — обмежена реалізація, що потребує втручання оператора.

Проведений мною аналіз показав, що програма RobotReviewer добре визначає тип медичних досліджень. Але вона аналізує тексти лише за кількома правилами, її інтерфейс дуже складний, а код закритий. Через це я не можу додати туди свої алгоритми без повної перебудови системи. Також вона зовсім не перевіряє репутацію медичних журналів.

Платформа PubTator активно використовується для біомедичного аналізу, оскільки підтримує якісне вилучення ключових сутностей. Проте, як показано на рисунку 1.1, її функціонал обмежується виключно графічним підсвічуванням базових концептів (назв генів, хімічних сполук, захворювань та видів організмів) у тексті анотації. Я звернула увагу, що у системі PubTator повністю відсутні інструменти математичного ранжування, класифікації рівнів доказовості за медичними стандартами, а також модулі виявлення фінансового або комерційного конфлікту інтересів авторів.

Я прийшла до висновку, що жодне з існуючих рішень не забезпечує одночасного поєднання автоматизованого збору даних через API, семантичного

розбору тексту, аналізу маркерів конфлікту інтересів авторів та гнучкого математичного ранжування за критеріями доказової медицини. Розробка вебзастосунку з таким інтегрованим функціоналом дозволить мені закрити ці недоліки.

Рисунок 1.1 – Візуалізація біомедичних сутностей в інтерфейсі PubTator

Жодна програма не поєднує глибокий семантичний аналіз, вилучення числових параметрів та математичне ранжування за рівнем доказовості. Реалізація такого функціоналу суттєво скоротить час на первинну селекцію потрібної медичної інформації.

1.2 Математичне забезпечення та методика розрахунку рейтингу доказовості

Для об'єктивної оцінки статей я створила математичну модель сортування. Процес автоматизованого аналізу я базувала на розрахунку підсумкового рейтингу доказовості R, який моя програма рахує для кожної знайденої публікації.

Нехай $D = \{d_1, d_2, \dots, d_n\}$ є набором статей, які система завантажила з PubMed через API. Головним компонентом моєї розробки є алгоритм ранжування, що працює за методом зваженого сумування критеріїв якості статті. Для оцінки матеріалів алгоритм обчислює підсумковий рейтинг R за такою формулою:

$$R = (K_d \cdot w_1) + (\log(n) \cdot w_2) + (I_a \cdot w_3), \qquad (1.1)$$

де K_d — ієрархічний коефіцієнт дизайну дослідження (від 1 для описів випадків до 10 для мета-аналізів);

$\log(n)$ — логарифмічно нормалізований обсяг вибірки для згладжування статистичних розбіжностей;

I_a — індекс актуальності на основі дати публікації та цитованості;

w_1, w_2, w_3 — вагові коефіцієнти пріоритетності параметрів.

Коефіцієнт дизайну K_d у моїй моделі набуває значень від 1 до 10. Мінімальний бал 1 система виставляє для простих описів клінічних випадків, а максимальний бал 10 отримують найнадійніші медичні праці - метааналізи та систематичні огляди. Це дозволяє автоматично піднімати методологічно сильніші роботи на перші позиції.

Показник $\log(n)$ відповідає за кількість учасників дослідження. Я використала логарифмічну нормалізацію для того, щоб збалансувати видачу статей. Це допомагає запобігти ситуаціям, коли великі епідеміологічні дослідження з величезною кількістю пацієнтів повністю домінують у рейтингу над якісними клінічними випробуваннями. акий підхід узгоджується з практикою інженерії даних, де логарифмування екстремальних масштабів вибірок використовується для лінеаризації ознак і стабілізації адитивних моделей ранжування [7].

Індекс актуальності I_a дозволяє враховувати новизну та авторитетність дослідження. Його значення залежить від року випуску статті та від того, як активно її цитують інші вчені.

Вагові коефіцієнти w_1, w_2, w_3 користувач може змінювати самостійно за допомогою повзунків у графічному інтерфейсі. При цьому в програмі діє стандартне обмеження, за якого сума всіх ваг завжди дорівнює одиниці:

Налаштування вагових коефіцієнтів дозволяє адаптувати пошук під різні завдання, від пріоритезації нових публікацій до відбору масштабних клінічних досліджень. Результатом роботи моделі є список статей, відранжований за спаданням підсумкового балу.

1.3 Визначення цілей дослідження та класифікація вимог до вебзастосунку

Проведений мною порівняльний аналіз виявив суттєві обмеження існуючих систем. Головними недоліками аналогів є відсутність гнучкого інтерфейсу для налаштування критеріїв пошуку, складність розгортання та використання, неможливість автоматичного видобування обсягу вибірки пацієнтів з анотацій, брак інструментів для виявлення конфлікту інтересів у публікаціях, а також брак математичних моделей для ранжування статей за рівнем доказовості. Оцінивши проблеми медичного пошуку та перевантаження дослідників інформацією, я визначила головну мету та практичні завдання свого проєкту.

Головною метою роботи є інженерне проєктування, програмна реалізація та експериментальное тестування програмної системи, результатом якого є працездатний вебзастосунок для автоматизовавного збору, інтелектуального аналізу й математичного ранжування наукових публікацій з дотриманням міжнародних стандартів розробки ПЗ та кількісних метрик якості.

Створюваний програмний продукт має забезпечувати автоматизацію процесів асинхронного збору метаданих із бази PubMed через інтеграцію з NCBI Entrez API, інтелектуальну обробку неструктурованих текстів анотацій за допомогою алгоритмів патерн-метчингу, а також математичне ранжування результатів на основі моделі зваженого сумування критеріїв доказовості. Основний інженерний фокус спрямовано на оптимізацію швидкодії системи за

рахунок розподілу обчислень у фоновому режимі та забезпечення високої точності класифікації даних.

Для досягнення цієї мети я поставила перед собою такі завдання:

1) виконати аналіз предметної області, сформулювати функціональні вимоги до вебзастосунку та дослідити специфікацію програмного інтерфейсу NCBI Entrez API;

2) розробити асинхронний програмний модуль інтеграції з PubMed API для паралельного завантаження та десеріалізації структурованих XML-метаданих публікацій;

3) програмно реалізувати механізми контролю частоти запитів (Rate Limiting) для дотримання регламентних обмежень серверів NCBI;

4) розробити підсистему семантичного аналізу тексту та метаданих на основі регулярних виразів та лінгвістичних маркерів для автоматичної ідентифікації дизайну досліджень, видобування масштабу вибірки пацієнтів та виявлення потенційного конфлікту інтересів й комерційного фінансування публікацій;

5) реалізувати алгоритм обчислення інтегрального рейтингу статей на основі математичної моделі, що враховує методологічну якість, обсяг вибірки та коефіцієнт новизни публікації;

6) спроєктувати графічний інтерфейс користувача з інтерактивною панеллю керування ваговими коефіцієнтами та модулем експорту звітів у формати CSV/Excel;

7) провести комплексне тестування, верифікацію та оцінювання показників якості, швидкодії та зручності використання розробленого вебзастосунку.

Оскільки процес формування вимог до програмного продукту є фундаментом для його подальшої архітектурної реалізації, цей етап вимагає чіткої структуризації. При аналізі та класифікації технічних вимог до розроблюваного вебзастосунку я дотримувалася положень міжнародного стандарту ISO/IEC/IEEE 29148 [8]. Відповідно до його регламентів, усі виявлені специфікації системи я

розділила на дві категорії: функціональні, що визначають очікувану поведінку та інструменти системи, та нефункціональні, які описують критерії її якості, швидкодії та надійності. Кожна вимога сформульована з урахуванням визначених цим стандартом обов'язкових критеріїв якості: атомарності, однозначності, несуперечливості та, найголовніше, верифікованості за допомогою чітких кількісних метрик.

Функціональні вимоги до вебзастосунку:

– FR-1: формування параметризованого пошукового запиту через графічний інтерфейс із використанням ключових слів та логічних операторів;

– FR-2: автоматизована трансляція запитів у синтаксис NCBI Entrez API та їх передача за протоколом HTTPS;

– FR-3: асинхронна агрегація ідентифікаторів статей через сервіс ESearch та завантаження метаданих у форматі XML за допомогою сервісу EFetch;

– FR-4: десеріалізація отриманих XML-пакетів та вилучення текстових блоків анотацій для подальшої програмної обробки;

– FR-5: ідентифікація дизайну дослідження шляхом семантичного аналізу тексту на наявність специфічних термінів-маркерів;

– FR-6: автоматичне ранжування результатів пошуку згідно з ієрархією рівнів наукової доказовості;

– FR-7: візуалізація оброблених даних у вигляді структурованого списку з можливостями фільтрації та прямого переходу до першоджерела.

Нефункціональні вимоги:

– NFR-1 (час відгуку системи): завдяки впровадженню асинхронної архітектури черг фонових задач, час очікування користувача при одночасній обробці, парсингу XML-коду та математичному розрахунку рейтингу для масиву до 500 анотацій не повинен перевищувати 2 секунди;

– NFR-2 (ефективність селекції): вебзастосунок повинен забезпечувати скорочення часу користувача на первинний аналіз та відбір релевантних матеріалів щонайменше на 60% порівняно зі стандартним алгоритмом PubMed Best Match;

– NFR-3 (точність розпізнавання вибірки): алгоритми патерн-метчингу та семантичного аналізу текстових блоків анотацій повинні забезпечувати точність (Precision) ідентифікації числових даних обсягу вибірки пацієнтів (n) на рівні не нижче 94%;

– NFR-4 (точність класифікації дизайну): модуль лінгвістичного аналізу має гарантувати точність класифікації типу дизайну дослідження (від клінічних випадків до мета-аналізів) на рівні не нижче 92%;

– NFR-5 (інтеграція з джерелом даних): система повинна забезпечувати стабільну асинхронну взаємодію із зовнішнім сервісом NCBI Entrez API за протоколом HTTPS з використанням інструментарію бібліотеки Biopython;

– NFR-6 (обмеження інтенсивності запитів): програма повинна гарантувати автоматичний контроль частоти запитів до серверів NCBI (не більше 3 запитів за секунду для неавторизованих сесій);

– NFR-7 (обробка виняткових ситуацій): вебзастосунок повинен коректно опрацьовувати аномалії у даних, зберігаючи працездатність інтерфейсу при отриманні від API порожніх відповідей або неструктурованих текстів;

– NFR-8 (резервування обчислень): логарифмічна нормалізація в моделі ранжування R має забезпечувати стійкість алгоритму до статистичних викидів.

– NFR-9 (динамічність інтерфейсу користувача): графічний інтерфейс користувача (UI) має забезпечувати миттєву візуалізацію обсягу вибірки, а також виконувати динамічний перерахунок фінального результату без перезавантаження вебсторінки при зміні користувачем вагових коефіцієнтів.

Сформовані функціональні та нефункціональні вимоги є основою для фінального етапу інженерії вимог розроблюваної системи. Цей аналітичний масив даних безпосередньо використовується як головне джерело для побудови, комплексної структуризації та детального моделювання Use Case діаграми в нотації мови UML. Такий підхід дозволяє чітко визначити межі проєкту, а також наочно зафіксувати всі можливі сценарії взаємодії між користувачем, системою та зовнішнім API.

1.4 Визначення акторів та варіантів використання (Use Case діаграма)

Для візуалізації вимог та логіки взаємодії користувача з вебзастосунком застосовано моделювання прецедентів у стандарті UML 2.5.1 [9]. Першим етапом побудови моделі є визначення таких акторів системи:

– Користувач: головний (первинний) актор системи, який є безпосереднім ініціатором бізнес-процесів розроблюваного вебзастосунку;

– NCBI Entrez API: системний (вторинний) актор, що виступає зовнішнім інтеграційним джерелом даних і забезпечує асинхронний доступ до репозиторію медичних публікацій.

На основі визначених акторів та функціональних вимог побудовано діаграму варіантів використання системи (рисунок 1.2).

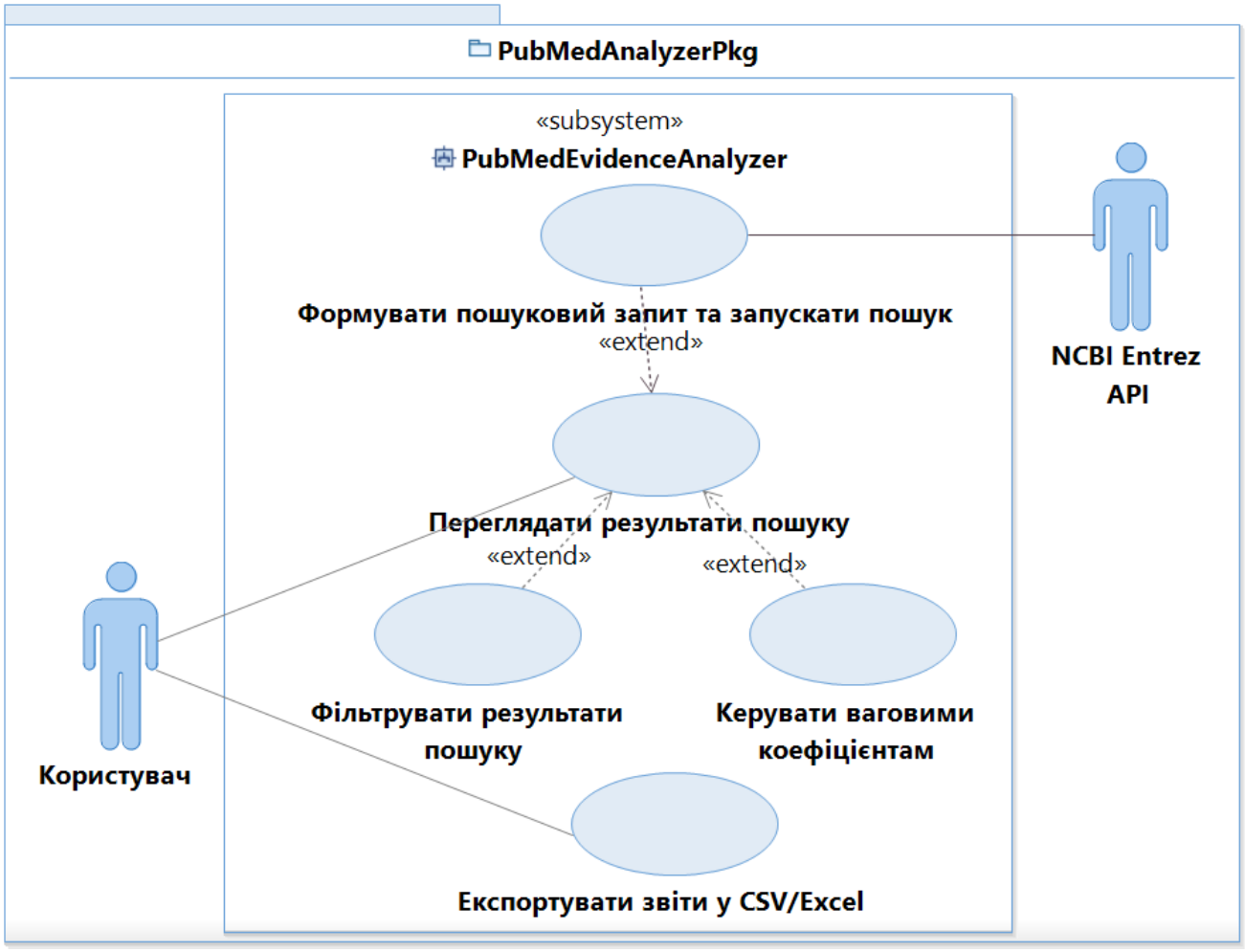

Рисунок 1.2 – Діаграма варіантів використання

Детальний опис та сценарії виконання наведених прецедентів структуровано у таблиці 1.2.

Таблиця 1.2 – Варіанти використання вебзастосунку

Код	**Актор**	**Найменування (Use Case)**	**Формулювання (Опис сценарію)**
UC-1	Користувач, NCBI Entrez API	Формувати пошуковий запит та запускати пошук	Користувач задає параметри пошуку; вебзастосунок асинхронно взаємодіє з API для отримання ідентифікаторів (ESearch) та метаданих статей (EFetch).
UC-2	Користувач	Переглядати результати пошуку	Користувач здійснює візуальний аналіз сформованої таблиці зі списком статей.
UC-3	Користувач	Керувати ваговими коефіцієнтами рейтингу	Користувач за допомогою інтерактивних елементів інтерфейсу (повзунків) динамічно змінює пріоритетність критеріїв ранжування, що викликає миттєвий перерахунок фінального рейтингу статей без перезавантаження сторінки.
UC-4	Користувач	Фільтрувати результати пошуку	Користувач застосовує фільтри та прапорці без повторних запитів до бази даних.
UC-5	Користувач	Експортувати звіти у CSV/Excel	Користувач здійснює вивантаження поточної відфільтрованої та відранжованої таблиці публікацій у локальний файл структурованого формату для подальшої обробки або включення у звіт.

Ключовим архітектурним вузлом системи є прецедент UC-1, оскільки він координує критичні інтеграційні потоки даних із зовнішнім API та ініціює первинне наповнення локальної моделі даних. Сценарії UC-3 та UC-4 спроєктовані мною як повністю автономні клієнтські операції, що дозволяє перераховувати рейтинги та фільтрувати масиви біомедичних анотацій безпосередньо в браузері користувача, мінімізуючи навантаження на сервер і забезпечуючи швидкість відгуку інтерфейсу.

Визначені прецеденти та ролі акторів дозволяють чітко окреслити функціональні межі застосунку, що необхідно для подальшої розробки архітектури застосунку.

1.5 Висновки до розділу 1

У першому розділі кваліфікаційної роботи виконано комплексний передпроєктний аналіз предметної області та сформовано технічний базис для розробки вебзастосунку. На основі дослідження існуючих аналогів, таких як RobotReviewer, PubTator та Rayyan, було виявлено відсутність доступних рішень для одночасного автоматизованого збору метаданих PubMed, розпізнавання клінічного дизайну публікацій та їх математичного ранжування. Це підтвердило актуальність та практичну доцільність створення нового спеціалізованого програмного продукту.

Важливим етапом роботи стало дослідження специфікації сервісів ESearch та EFetch програмного інтерфейсу NCBI Entrez API. У результаті аналізу ідентифіковано критичні інфраструктурні обмеження сторонньої системи, та високу структурну варіативність XML-пакетів, що визначило архітектурну необхідність проєктування стійких асинхронних парсерів.

Було сформовано специфікацію атомарних вимог до системи. Усі специфікації розділено на функціональні (FR-1–FR-7), які описують операційну логіку збору й обробки даних, та нефункціональні (NFR-1–NFR-9).

На завершення етапу аналізу вимог побудовано модель варіантів використання. Розроблена Use Case діаграма чітко визначила межі системи, ідентифікувала первинного (Користувач) та вторинного (NCBI Entrez API) акторів, а також деталізувала базові сценарії їхньої взаємодії, що є основою для подальшого об'єктно-орієнтованого проєктування архітектури та конструювання компонентів програмної системи.

РОЗДІЛ 2 ПРОЄКТУВАННЯ ТА КОНСТРУЮВАННЯ ВЕБЗАСТОСУНКУ

У другому розділі на основі визначених раніше вимог здійснюється перехід до безпосереднього створення програмного продукту, розділивши роботу на стадії проєктування та конструювання. У межах цих етапів обґрунтовано ітераційну модель розробки, сформовано клієнт-серверну архітектуру системи, спроєктовано реляційну схему бази даних й об'єктні структури, а також виконано вибір стеку технологій та програмну реалізацію алгоритмів семантичного аналізу, ранжування й персистентного збереження публікацій PubMed.

2.1 Обґрунтування моделі життєвого циклу та вибір процесу розробки

Оскільки розробку вебзастосунку я виконувала одноосібно в межах індивідуального проєкту, застосування важких командних методологій, як Scrum або каскадні моделі, було недоцільним. Натомість, керуючись положеннями міжнародного стандарту регламентації технічних процесів інженерії систем і програмного забезпечення ISO/IEC/IEEE 12207:2017 [10], як основу життєвого циклу проєкту я обрала ітераційно-інкрементну модель. Цей вибір зумовлений високим рівнем технічної невизначеності на початкових етапах розробки, зокрема через значну структурну варіативність текстових пакетів зовнішнього API та необхідність експериментального підбору математичних параметрів і критеріїв для точного семантичного ранжування. Відповідно до обраної моделі, загальний процес технічної реалізації системи я розділила на три ітерації, кожна з яких передбачала повний цикл розробки, від аналізу вимог до збирання робочого інкременту програми. Застосування такого підходу в індивідуальних дослідницьких проєктах дозволяє гнучко адаптувати архітектуру під час кожної фази без критичного переписування кодової бази.

У межах першої ітерації основну увагу я зосередила на дослідженні інфраструктурних особливостей інтеграції з серверами NCBI та реалізації базових REST-запитів до сервісів ESearch та EFetch. Своєю головною інженерною задачею на цьому етапі вважала аналіз обмежень пропускної здатності

стороннього сервісу та розробку архітектурного механізму контролю частоти запитів, який дозволив запобігти блокуванню IP-адреси застосунку. У результаті я створила стабільний комунікаційний модуль, здатний асинхронно відправляти параметризовані запити та гарантовано отримувати сирі пакети даних у форматі XML.

Другу ітерацію процесу розробки повністю присвятила внутрішній бізнес-логіці, математичному апарату системи та організації надійного збереження даних. На цьому етапі я спроєктувала внутрішні об'єктні моделі для представлення публікацій у пам'яті програми та створила гнучкий XML-парсер, стійкий до відсутності окремих структурних блоків у відповідях від PubMed. Одночасно з цим виконано програмну реалізацію алгоритмів семантичного аналізу текстів анотацій та обчислення вагових коефіцієнтів. Для забезпечення цілісності результатів аналізу, виключення ризиків втрати даних при збоях та підтримки конкурентного доступу, на цьому ж етапі було інтегровано реляційне сховище на базі вбудованої СКБД SQLite. Кінцевим інкрементом етапу став повністю працездатний бекенд-модуль, який автоматично збирав, очищав, оцінював та фіксував наукові статті у базі даних.

На фінальній третій ітерації я перейшла до проєктування та конструювання графічного інтерфейсу користувача, зосередивши основний фокус на реалізації інтерактивних повзунків для динамічної зміни пріоритетності критеріїв аналізу. Також на цьому етапі було забезпечено асинхронний зв'язок клієнта із сервером за технологією AJAX, завдяки чому вдалося досягти миттєвого надсилання запитів, їх обробки, збереження в БД та відображення результатів без перезавантаження сторінок вебзастосунку.

Завдяки обраній послідовності дій та ітераційному підходу мені вдалося мінімізувати ризики, пов'язані з динамічною зміною вимог, крок за кроком нарощувати функціональність вебзастосунку та проводити ізольоване тестування кожного окремого модуля ще до моменту повного фінального збирання всієї системи.

2.2 Проєктування архітектури вебзастосунку

Під час проєктування архітектурної структури вебзастосунку я керувалася методологічними настановами та принципами розподілу логіки, які регламентовані міжнародним зведенням знань з програмної інженерії SWEBOK [5]. Відповідно до цих стандартів, базою для розроблюваної системи обрала клієнт-серверний архітектурний стиль (Client-Server Architecture) у формі модульного моноліту з чітким дотриманням принципів розділення відповідальності (Separation of Concerns) та забезпечення слабкої пов'язаності компонентів (Low Coupling). Загальну структуру системи розділила на два автономні рівні:

– клієнтська частина (фронтенд): функціонує в середовищі браузера користувача, відповідає за побудову динамічного інтерфейсу, перехоплення подій введення, асинхронне відправлення запитів та візуалізацію результатів ранжування;

– серверна частина (бекенд): розгорнута на віддаленому вузлі, виконує функції координатора бізнес-логіки, здійснює комунікацію з репозиторієм PubMed, парсинг даних та розрахунок семантичного рейтингу публікацій.

Для організації внутрішньої логіки серверного компонента застосувала модифікований архітектурний шаблон Model-View-Controller (MVC). Особливості реалізації шаблону в межах розробленої системи полягають у наступному:

– Представлення (View): мінімізовано на серверній стороні та делеговано клієнту. Обмін даними між фронтендом і бекендом відбувається у форматі JSON за допомогою асинхронних HTTP-запитів;

– Контролер (Controller): представлений набором маршрутів (routes) та обробників запитів, які перехоплюють параметри пошуку від користувача, валідують їх та передають до обчислювального ядра системи;

– Модель (Model): інкапсулює структури даних біомедичних статей та логіку взаємодії із реляційною базою даних SQLite. Модель забезпечує

довготривале зберігання структурованих аналітичних результатів, транзакційність та ізоляцію даних для паралельних пошукових сесій.

Для забезпечення модифікованості коду я виконала декомпозицію серверної частини вебзастосунку на п'ять функціональних модулів:

– комунікаційний модуль (шлюз API) — утворює проміжний шар між користувачем та зовнішньою інфраструктурою Національного центру біотехнологічної інформації (NCBI). Він формує HTTPS-запити до сервісів ESearch та EFetch, а також містить механізм контролю частоти запитів (Rate Limiting) для запобігання інфраструктурному блокуванню;

– модуль синтаксичного аналізу (парсер): приймає сирі XML-пакети від комунікаційного модуля, очищає їх від службових тегів, ізолює блоки метаданих, назви статей, тексти анотацій та трансформує їх у об'єктні масиви;

– обчислювальний модуль (аналітичне ядро): реалізує математичний апарат семантичного аналізу та динамічного ранжування публікацій за спаданням інтегрального балу на основі визначених користувачем вагг;

– модуль бази даних (компонент персистентності): відповідає за взаємодію з файлом бази даних SQLite, інкапсулює SQL-запити, виконує автоматичне розгортання схеми таблиць та забезпечує потоковий безпечний запис відранжованих публікацій;

– модуль маршрутизації (API-контролер): приймає вхідні HTTP-запити від клієнта, активує роботу суміжних модулів, конвертує фінальний відранжований масив статей у формат JSON та відправляє його назад на фронтенд за допомогою технології AJAX.

Декомпозицію програмного коду на функціональні модулі представлено на рисунку 2.1 за допомогою UML-діаграми компонентів, яку побудовано відповідно до методологічних правил та графічних специфікацій, визначених в офіційному стандарті уніфікованої мови моделювання OMG UML Version 2.5.1 [8].

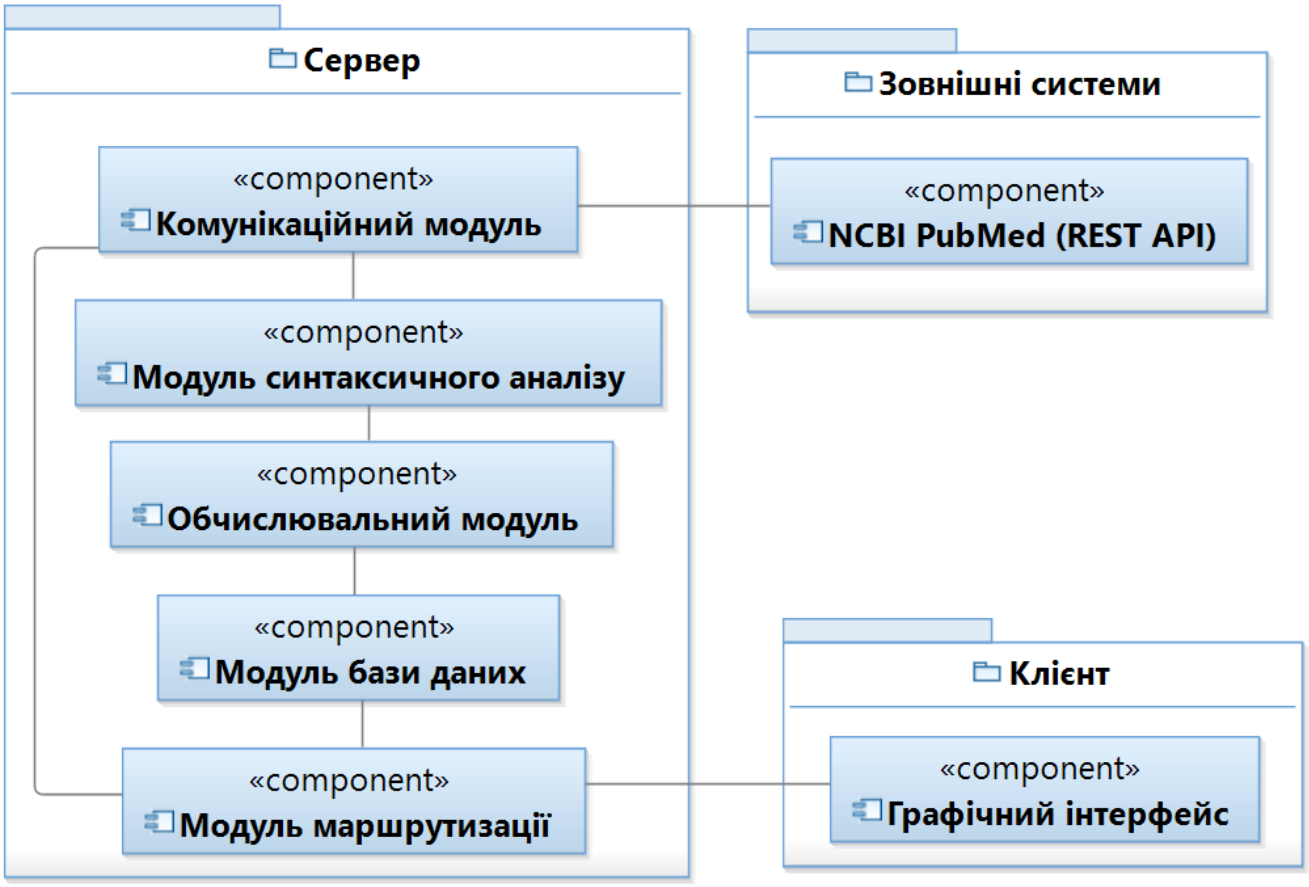

Рисунок 2.1 – Діаграма компонентів

Для моделювання динамічного аспекту системи я розробила діаграму послідовності (див. додаток А). Користувач через графічний інтерфейс надсилає HTTP-запит до модуля маршрутизації, який запускає комунікаційний модуль. Після перевірки лімітів цей модуль завантажує XML-дані з NCBI PubMed і передає їх модулю синтаксичного аналізу для очищення та формування об'єктного масиву статей.

Отриманий пул через модуль маршрутизації перенаправляється в обчислювальний модуль для розрахунку ваг та семантичного ранжування. Далі відсортовані публікації передаються до модуля бази даних для персистентного збереження через СКБД SQLite. Після підтвердження запису фінальний JSON-пакет через контролер відправляється на фронтенд для відображення користувачу.

На основі вищеописаної динамічної моделі взаємодії компонентів, архітектурний контракт програмного інтерфейсу системи було уніфіковано до єдиного канонічного маршруту GET /api/v1/articles, який функціонує в межах

архітектурного стилю REST, описаного Роєм Філдінгом [11]. Цей ресурсний ендпоінт призначений для пошуку, збору та семантичного ранжування публікацій. За проєктом зазначений метод працює виключно в режимі читання даних і приймає як параметри HTTP-запиту такі атрибути, як пошуковий рядок, параметри зміщення та ліміту для організації клієнтської пагінації, а також три вагові коефіцієнти (w1, w2, w3), необхідні для динамічного налаштування алгоритму скорингу на стороні сервера. Спроєктована архітектурна логіка передбачає, що цей уніфікований запит транслюється модулем маршрутизації до обчислювального ядра та зовнішніх сервісів, повертаючи клієнту структуровану відповідь у форматі JSON-масиву з кодом стану 200 OK.

2.3 Моделювання структур даних та побудова схеми бази даних

Важливим етапом конструювання вебзастосунку для мене став перехід від концептуальних об'єктів предметної області до їхнього фізичного представлення у середовищі функціонування системи. Оскільки ключовою бізнес-сутністю моєї програми є наукова стаття, очищена від службових тегів XML та збагачена розрахованими семантичними метриками, переді мною постала задача організації ефективного й надійного сховища даних. Логічну та фізичну структуру сховища я організувала у вигляді однієї центральної реляційної таблиці ResearchArticle, детальний опис якої представлено в таблиці 2.1.

Таблиця 2.1 – Специфікація структури даних таблиці ResearchArticle

Назва поля	**Тип даних**	**Обмеження цілісності**	**Опис призначення атрибута**
id	INTEGER	PRIMARY KEY AUTOINCREMENT	Внутрішній унікальний системний ідентифікатор запису в базі даних.
pmid	TEXT	UNIQUE, NOT NULL	Унікальний ідентифікатор наукової статті у міжнародній базі PubMed.
title	TEXT	NOT NULL	Повна назва наукової публікації.
authors	TEXT	—	Список авторів статті.
journal	TEXT	—	Назва наукового журналу або медичного видавництва.
pub_date	TEXT	—	Дата офіційної публікації статті.

Продовження таблиці 2.1

1	2	3	4
abstract	TEXT	—	Очищений від службових тегів повний текст анотації.
mesh_terms	TEXT	—	Індексовані медичні предметні рубрики.
has_coi	INTEGER	NOT NULL (0 або 1)	Логічний прапор наявності конфлікту інтересів у авторів.
k_d	REAL	DEFAULT 0.0	Розрахована математична вага релевантності.
evidence_level	TEXT	—	Визначений категоріальний рівень доказовості дослідження.
calculated_score	REAL	DEFAULT 0.0	Інтегральний семантичний бал для фінального ранжування у списку.

Для забезпечення довготривалого зберігання, транзакційності та можливості швидкої аналітичної вибірки інформації, я спроєктувала та впровадила реляційну базу даних на базі вбудованої СКБД SQLite. Мій вибір СКБД SQLite зумовлений її високою швидкістю виконання операцій читання, автономністю файлового сховища та відсутністю потреби в адмініструванні окремого сервера бази даних на етапі демонстраційного розгортання проєкту [12].

Для забезпечення гнучкості архітектури та реалізації незалежності бізнес-логіки від сховища, взаємодію сервера з файлом бази даних реалізовано через спеціалізований клас-абстракцію ArticleDatabase всередині модуля models.py, який самостійно контролює життєвий цикл підключення. Під час першого запуску сервера цей клас автоматично ініціалізує створення таблиці за її відсутності за допомогою SQL-запиту CREATE TABLE IF NOT EXISTS.

Також всередині класу інкапсульовано безпечне виконання параметризованих інструкцій INSERT OR REPLACE, що гарантує захист системи від дублювання записів при повторних запитах та повністю виключає ризик виникнення SQL-ін'єкцій. Збереження даних у таблиці організовано пакетним (потоковим) методом у межах єдиної транзакції, що дозволило мінімізувати кількість дискових операцій введення-виведення та досягти миттєвого відгуку інтерфейсу користувача.

2.4 Об'єктно-орієнтоване проєктування та побудова UML-діаграм класів

Для забезпечення масштабованості, гнучкості та простоти підтримки розроблюваного вебзастосунку, я обрала об'єктно-орієнтовану парадигму проєктування (ООП). Розбиття системи на логічні класи дозволило мені ізолювати окремі бізнес-функції програми (завантаження даних, аналітичну обробку, збереження) та мінімізувати зв'язність між модулями системи, що повністю відповідає фундаментальним архітектурним принципам інженерії програмного забезпечення [5].

Головним інструментом для візуалізації статичної структури програмного коду та зв'язків між його компонентами я обрала UML-діаграму класів. Ця модель дозволяє наочно продемонструвати ієрархію сутностей та логіку побудови об'єктно-орієнтованої архітектури системи. Вона детально відображає архітектуру внутрішніх модулів бекенду та типи їхньої взаємодії. Спроєктована мною діаграма класів представлена нижче на рисунку 2.2.

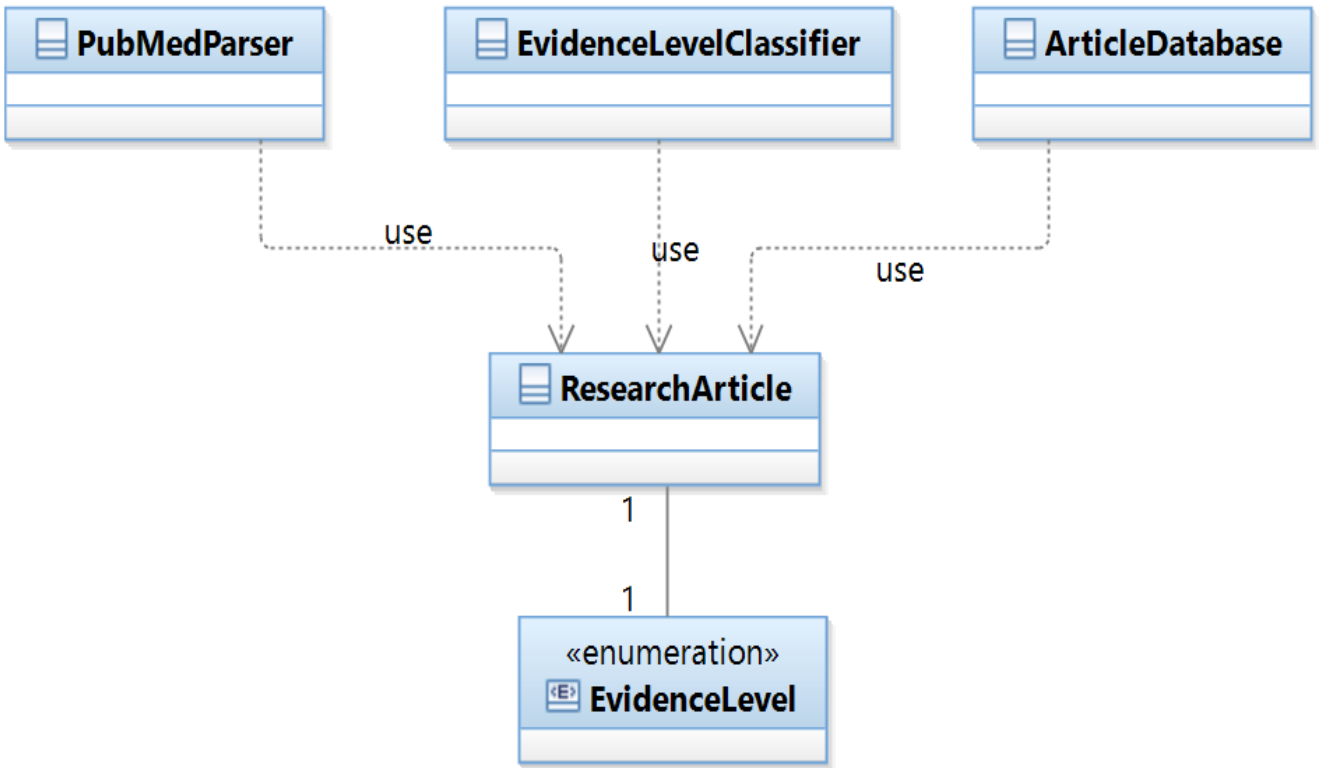

Рисунок 2.2 – Діаграма класів

Основою побудованої архітектури є такі ключові класи:

– ResearchArticle: клас-модель, який виступає в ролі структури даних (сутності) для представлення однієї наукової публікації в оперативній пам'яті системи. Він містить атрибути метаданих статті та методи для її валідації;

– PubMedParser: сервісний клас, що інкапсулює логіку взаємодії із зовнішнім API репозиторію NCBI. Його методам передаються пошукові параметри, після чого клас виконує HTTP-запити, отримує відповіді у форматі XML та здійснює їх первинний синтаксичний аналіз;

– EvidenceLevelClassifier: аналітичний клас, відповідальний за реалізацію інтелектуальної складової проєкту. Він містить алгоритми для текстової передоброки (токенізації, лематизації), обчислення статистичних ваг та категоризації рівнів доказовості;

– ArticleDatabase: інфраструктурний клас, приймає готові об'єкти класу ResearchArticle та забезпечує їх персистентне зберігання у файлі SQLite.

Взаємозв'язки між елементами моделі спроєктовано із використанням двох фундаментальних типів відношень:

– залежності (dependency) зі стандартним стереотипом «use», який вказує на те, що зазначені модулі функціонально залежать від структури статті, оперуючи її об'єктами тимчасово (як вхідними аргументами або типами повернення функцій), проте не містять їх у якості своїх постійних полів;

– асоціації (association), яку відображено суцільною лінією між класом ResearchArticle та переліком EvidenceLevel. Наявність маркерів кратності 1 до 1 строго визначає структурну залежність, тобто кожен об'єкт наукової публікації в обов'язковому порядку містить рівно один фіксований атрибут, типом якого є цей перелік рівнів доказовості, що зберігається протягом усього життєвого циклу об'єкта.

Розроблена об'єктна модель чітко розділяє відповідальність компонентів відповідно до принципу Single Responsibility. Організація взаємодії через відношення залежності гарантує гнучкість бекенду, дозволяючи масштабувати

систему, змінювати алгоритми аналізу або СКБД без рефакторингу логіки збору даних.

2.5 Обґрунтування вибору мови програмування та інструментів розробки

Під час проєктування архітектури аналітичного вебзастосунку я приділила особливу увагу вибору технологічного стеку, оскільки система вимагає ефективної інтеграції з віддаленими сервісами, точного аналізу текстових масивів та наявності гнучкого вебінтерфейсу.

Як базову мову програмування для розробки серверної частини програми я обрала Python. Мій вибір зумовлений її високою швидкістю розробки, лаконічністю синтаксису, надійною системою керування пам'яттю та наявністю потужних вбудованих інструментів для обробки текстових даних. Це дозволило мені зосередитися на логіці фільтрації та скорингу біомедичних публікацій без необхідності низькорівневої оптимізації коду.

Для реалізації клієнт-серверної взаємодії та побудови архітектурного стилю REST API я обрала мікрофреймворк Flask. На відміну від великих монолітних платформ, Flask надає лише необхідний мінімальний інструментарій для маршрутизації HTTP-запитів та обробки сесій, що забезпечує низькі накладні витрати на роботу сервера. Завдяки модульній структурі цього фреймворку я змогла легко відокремити аналітичну логіку від інтерфейсу представлення даних та реалізувати безстановову обробку клієнтських запитів у форматі JSON.

Взаємодію із зовнішнім репозиторієм наукових публікацій PubMed я організувала за допомогою спеціалізованого пакету Biopython (модуль Bio.Entrez). Ця бібліотека є загальноприйнятим стандартом в інженерії біомедичних даних для алгоритмічного доступу до баз даних NCBI. Використання цього готового інструменту дозволило мені автоматизувати процес надсилання низькорівневих HTTP-запитів до NCBI Entrez API, спростити парсинг

структурованих метаданих публікацій та забезпечити коректну обробку XML-відповідей від сервера репозиторію.

Для реалізації лінгвістичного аналізу та класифікації рівнів доказовості статей я обрала підхід на основі регулярних виразів (вбудований модуль re мови Python). Мій вибір на користь паттерн-матчингу (pattern matching) зумовлений тим, що медичні наукові публікації мають сувору стандартизовану термінологію та чіткі маркери типів досліджень. Використання регулярних виразів дозволило мені створити швидкодійну детерміновану систему класифікації без залучення важких сторонніх бібліотек машинного навчання, що мінімізувало кількість зовнішніх залежностей проєкту та прискорило роботу бекенду.

У ролі локальної системи керування базами даних я обрала реляційну базу SQLite. Мій вибір зумовлений потребою у створенні легковагого локального кешу для збереження результатів парсингу та швидкого доступу до вже завантажених статей без надсилання повторних запитів до NCBI API. Оскільки SQLite інтегрується через стандартний модуль sqlite3 і зберігає дані в одному файлі, це позбавило мене потреби адмініструвати окремий сервер БД, забезпечивши високу мобільність і простоту розгортання проєкту.

Фронтенд-частину застосунку я побудувала на базі класичної тріади технологій: мови розмітки HTML5, каскадних таблиць стилів CSS3 та мови JavaScript. Для швидкої реалізації адаптивної графічної сітки я використала CSS-фреймворк Tailwind CSS. Усю логіку взаємодії з REST-ендпоінтами сервера, асинхронне надсилання пошукових запитів через механізм Fetch API та клієнтське переранжування даних я реалізувала за допомогою чистого JavaScript (Vanilla JS).

2.6 Програмна реалізація основних класів та алгоритмів семантичного аналізу

Програмну реалізацію логіки лінгвістичного аналізу, структурування метаданих та кешування результатів я виконала у вигляді об’єктно-орієнтованого

комплексу на базі мови Python. Система розподілена на кілька ключових класів, кожен з яких виконує ізольовану архітектурну функцію.

Клас ResearchArticle реалізовано як заморожений та оптимізований за пам'яттю контейнер даних (dataclass(frozen=True, slots=True)), який слугує базовою інформаційною моделлю наукової публікації в системі. Його основними атрибутами є унікальний ідентифікатор статті pmid, текстові поля заголовка title та анотації abstract, рядкові змінні для авторів authors та журналу journal, а також рік публікації year, список медичних термінів mesh_terms та логічний прапорець наявності конфлікту інтересів has_coi. Окрім інформаційних полів, клас містить аналітичні атрибути: evidence_level (рядковий літерал типу дослідження) та k_d (числовий коефіцієнт дизайну). Оскільки цей клас є незмінною моделлю представлення даних, він не містить власних бізнес-методів, а його екземпляри створюються зовнішніми сервісами. Його програмну структуру наведено нижче в лістингу 2.1.

Лістинг 2.1 – Фрагмент коду класу ResearchArticle

```
@dataclass(frozen=True, slots=True)
class ResearchArticle:
    pmid: str
    title: str
    abstract: str
    evidence_level: EvidenceLevel = "Unknown"
    k_d: int = 0
    authors: str = "Unknown Authors"
    journal: str = "Unknown Journal"
    year: int | str = ""
    mesh_terms: list[str] = field(default_factory=list)
    has_coi: bool = False
```

Клас RankingWeights спроєктовано як легковажну імутабельну структуру даних (dataclass), призначену для інкапсуляції та передачі налаштувань користувача з графічного інтерфейсу до аналітичного модуля. Його структура обмежена трьома атрибутами з рухомою комою: w1, w2 та w3, які відповідають динамічним ваговим коефіцієнтам для систематичних оглядів, рандомізованих контрольованих досліджень та описів клінічних випадків відповідно. Завдяки

використанню властивості frozen=True, цей клас гарантує цілісність конфігурації пріоритетів ранжування під час усього життєвого циклу обробки одного пошукового запиту, захищаючи параметри від випадкових модифікацій у процесі обчислень. Програмну реалізацію класу показано в лістингу 2.2.

Лістинг 2.2 – Фрагмент коду класу RankingWeights

```
@dataclass(frozen=True, slots=True)
class RankingWeights:
    w1: float
    w2: float
    w3: float
```

Клас EvidenceLevelClassifier є головним аналітичним компонентом системи, який здійснює лінгвістичну обробку та класифікацію статей. До його приватних атрибутів належать шаблони регулярних виразів з інженерними вагами (_rct_markers, _sr_markers, _case_report_markers) та вираз для аналізу заперечень _rcip_negation_pattern. Публічний інтерфейс класу представлений методами classify_abstract (для нормалізації тексту і розрахунку балів) та classify_article (для формування об'єкта статті з вирахованим коефіцієнтом дизайну). Внутрішня логіка підтримується захищеним методом _classify_non_rct для обробки заперечень та статичним методом _score, який підраховує суму ваг знайдених маркерів. Фрагмент реалізації класу наведено в лістингу 2.3.

Лістинг 2.3 – Фрагмент коду класу EvidenceLevelClassifier

```
class EvidenceLevelClassifier:
    _rcip_negation_pattern =
re.compile(r"\b(not|no)\s+(randomi[sz]ed?|randomly\s+assigned)\b")
    _rct_markers: list[tuple[re.Pattern[str], int]] = [
        (re.compile(r"\brandomi[sz]ed?\b"), 3),
(re.compile(r"\bdouble[- ]blind\b"), 3)
    ]

    def classify_abstract(self, abstract: str) -> EvidenceLevel:
        text = _normalize_text(abstract)
        if not text: return "Unknown"
        if self._rcip_negation_pattern.search(text): return
self._classify_non_rct(text)
        # Розрахунок за допомогою методів _score та max()
```

Клас ArticleDatabase реалізує патерн автономного шлюзу до бази даних та керує збереженням інформації в локальній СКБД SQLite. Його єдиним конструктивним атрибутом є db_path (екземпляр класу Path), що вказує на фізичне розташування файлу бази даних на диску. Взаємодія з СКБД реалізована через чотири методи: приватний метод _init_db автоматично створює реляційну таблицю articles із необхідною схемою та індексами при першому запуску програми; метод save_articles забезпечує пакетний запис об'єктів у базу даних за стратегією INSERT OR REPLACE із паралельною серіалізацією списків складних типів; метод get_all_articles виконує зворотне зчитування, десеріалізацію даних та фабрикацію списку об'єктів ResearchArticle; а метод clear_all забезпечує повне очищення таблиці для скидання локального кешу. Фрагмент програмного коду класу представлено в лістингу 2.4.

Лістинг 2.4 – Фрагмент коду класу ArticleDatabase

```
class ArticleDatabase:
    def __init__(self, db_path: str | Path = "articles.db") ->
None:
        self.db_path = Path(db_path)
        self._init_db()

    def save_articles(self, articles: list[ResearchArticle]) ->
None:
        with sqlite3.connect(self.db_path) as conn:
            for article in articles:
                conn.execute("INSERT OR REPLACE INTO articles
...", (...))
            conn.commit()
```

Центральною обчислювальною логікою розробленого модуля є детермінований алгоритм лінгвістичного аналізу та зваженого скорингу текстових масивів. Цей алгоритм інтегрований у метод classify_abstract класу EvidenceLevelClassifier і призначений для послідовної обробки завантажених анотацій з урахуванням синтаксичних заперечень та вагових коефіцієнтів маркерів.

Для детального представлення послідовності виконання логічних та математичних операцій нижче на рисунку 2.3 наведено розроблену мною блок-схему аналітичного алгоритму.

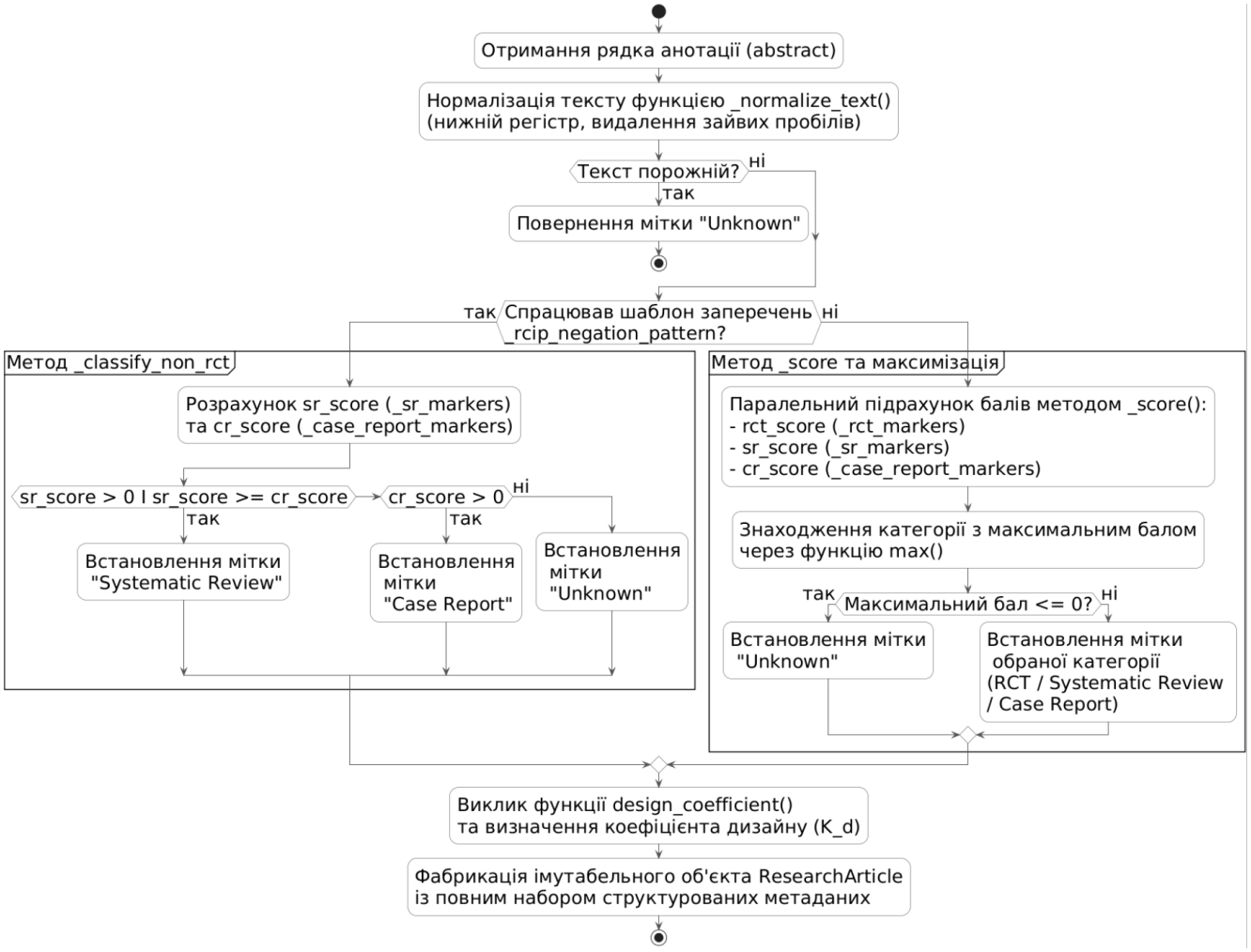

Рисунок 2.3 – Блок-схема алгоритму

Функціонування алгоритму відбувається за такими послідовними інженерними кроками:

– попереднє очищення і нормалізація тексту, що передбачає видалення зайвих пробілів, зведення рядка до нижнього регістру функцією _normalize_text та миттєве повернення мітки Unknown у разі виявлення порожнього значення;

– сканування нормалізованого тексту за допомогою регулярного виразу _rcip_negation_pattern для виявлення синтаксичних заперечень на кшталт «not randomized»;

– переспрямування обчислювального потоку до захищеного методу _classify_non_rct у разі знаходження заперечення, що повністю блокує присвоєння категорії рандомізованих досліджень і запускає порівняння балів лише між систематичними оглядами та клінічними випадками;

– ітераційний посимвольний пошук лінгвістичних маркерів за допомогою статичного методу _score для трьох масивів регулярок за відсутності заперечень, де кожен знайдений паттерн додає до загального рейтингу класу свою фіксовану внутрішню вагу;

– об'єднання отриманих числових балів у список кортежів та знаходження категорії з максимальним значенням за допомогою вбудованої функції max(), яка автоматично відхиляє класифікацію та повертає статус Unknown, якщо підсумковий бал є меншим або дорівнює нулю;

– фінальна фабрикація об'єкта, під час якої на основі визначеного рівня доказовості через функцію design_coefficient() розраховується числовий коефіцієнт дизайну, а всі структуровані метадані пакуються в імутабельний екземпляр ResearchArticle для збереження в СКБД.

Розроблена структура класів та покроковий алгоритм аналізу забезпечили надійну роботу аналітичного модуля, що дозволило створити ефективну базу для подальшої візуалізації та ранжування статей.

2.7 Розробка графічного інтерфейсу користувача

Взаємодію користувача та зовнішніх сервісів із розробленою системою я реалізувала через графічний вебвізуалізатор, який виступає в ролі єдиної точки управління аналітичними функціями програми. У процесі конструювання графічного інтерфейсу застосунку забезпечила суворе дотримання фундаментальних евристик юзабіліті Якоба Нільсена [13].

Принцип видимості статусу системи я реалізувала через індикатор завантаження та лічильник знайдених результатів. Відповідність реальному світу забезпечено використанням звичної біомедичної термінології та стандартів

доказової медицини. Свободу дій користувача підтримано кнопкою «Reset All», інтерактивними повзунками коригування ваг, системою фільтрів та вікном довідки. Ефективність використання досягнуто завдяки кольоровому кодуванню рівнів доказовості та миттєвому переранжуванню таблиці за допомогою JavaScript без повторних запитів до сервера.

Для взаємодії користувача з аналітичним ядром на головній сторінці вебзастосунку я реалізувала такі UI-елементи: текстове поле для введення ключових слів, авторів або термінів; панель конфігурації для встановлення ліміту публікацій із PubMed та параметрів аналізу; кнопку запуску процесу парсингу та серверного скорингу. Результати виводяться у динамічну таблицю, що відображає перелік знайдених статей із зазначенням ідентифікатора PMID, назви, обчисленого інтегрального балу та кольорового маркера рівня доказовості.

Для демонстрації реалізованої концепції взаємодії користувача із системою та оцінки візуальної структури елементів керування нижче на рисунку 2.3 представлено графічний інтерфейс головного вікна вебзастосунку.

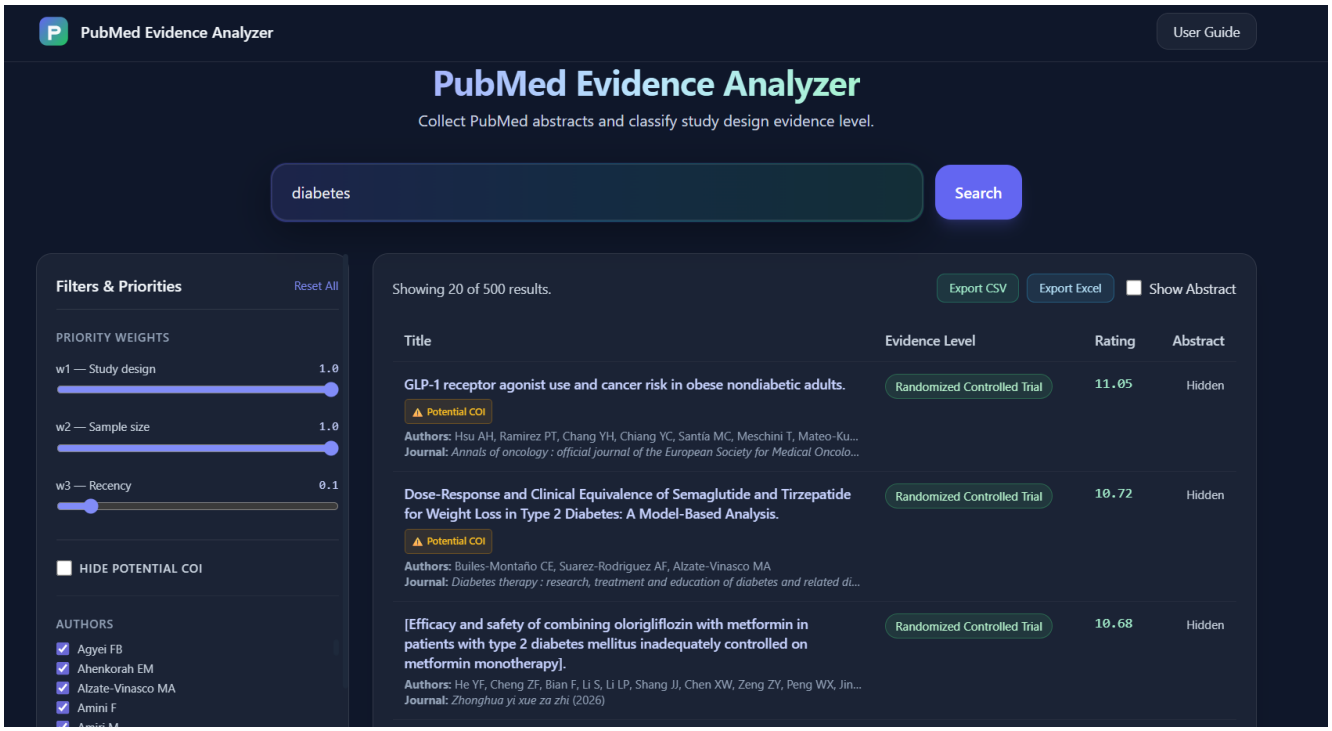

Рисунок 2.3 – Дизайн графічного інтерфейсу системи

Розроблене мною компонування елементів мінімізує когнітивне навантаження та прискорює адаптацію користувача до системи. Завдяки винесенню обчислень на сервер та оптимізації клієнтського рендерингу, інтерфейс демонструє високу чутливість (Responsiveness) та миттєвий відгук на зміну вагових коефіцієнтів. Таким чином, спроєктована візуальна модель повністю задовольняє вимогам ергономічності, гнучкості та доступності вебзастосунку.

2.8 Висновки до розділу 2

У другому розділі на основі сформованих вимог виконано проєктування та конструювання вебзастосунку. Застосування ітераційного підходу дозволило розділити розробку на етапи інтеграції з віддаленими серверами, реалізації обчислювальної логіки та побудови інтерфейсу. На основі клієнт-серверного стилю спроєктовано модульну структуру програми з розподілом відповідальності, а взаємодію між компонентами уніфіковано через єдиний маршрут обміну даними.

Розроблено структуру локальної бази даних для збереження публікацій та результатів їхнього аналізу. Взаємодію з базою даних ізольовано в окремому інфраструктурному класі, де реалізовано пакетний запис за стратегією оновлення даних, яка запобігає дублюванню інформації та захищає систему від ін'єкцій. Використання вбудованої бази даних дозволило уникнути адміністрування сторонніх серверів і забезпечило швидку роботу локального кешу.

Здійснено програмну реалізацію об’єктної моделі даних та аналітичного ядра системи. Розроблено детермінований алгоритм аналізу текстів за допомогою регулярних виразів, який враховує мовні заперечення, здійснює зважений пошук маркерів та обчислює інтегральний рейтинг доказовості статей. Спроєктований інтерфейс користувача забезпечує зміну параметрів оцінки та миттєве переранжування результатів на стороні клієнта без повторного надсилання запитів до сервера.

РОЗДІЛ 3 ТЕСТУВАННЯ, ВПРОВАДЖЕННЯ ТА ПІДТРИМКА ВЕБЗАСТОСУНКУ

У цьому розділі наведено результати комплексного інженерного тестування розробленого вебзастосунку на відповідність специфікації технічних вимог. Описано інфраструктуру та процес розгортання програмної системи. Також проведено експериментальну оцінку швидкодії асинхронного модуля, аналіз точності лінгвістичного NLP-класифікатора та верифікацію стійкості алгоритму математичного ранжування статей PubMed.

3.1 Програмний код та супровідна документація проєкту

Тестування програмного забезпечення є невід’ємним етапом його життєвого циклу, який безпосередньо впливає на підсумкову якість, надійність та експлуатаційну придатність продукту. Системний підхід до проєктування тестів за рекомендаціями міжнародного стандарту SWEBOK [5] дозволив мінімізувати ризики виникнення критичних збоїв системи під час інтеграції з реальними сервісами (зокрема, із зовнішнім NCBI API) та інженерно довести повну відповідність усіх архітектурних модулів розробленого вебзастосунку критеріям початкового технічного завдання.

Стратегія проєктування тестів спирається на принципи чорної скриньки (Black Box Testing) [15]. Вибір даної інженерної стратегії обґрунтований необхідністю незалежної валідації системи з точки зору кінцевих бізнес-вимог користувача, а не внутрішньої структури коду. Головна особливість цього підходу полягає в тому, що валідація працездатності програмних компонентів виконується виключно через перевірку відповідності вихідних результатів до вхідних параметрів і дій користувача. При цьому внутрішня логіка коду, специфіка інтерпретатора Flask та прямі SQL-інструкції SQLite залишаються ізольованими від процесу випробувань, що забезпечує об'єктивність оцінювання та унеможливлює суб'єктивні помилки розробника.

Для досягнення поставленої мети та забезпечення всебічного контролю якості системи я сформувала комплексний план випробувань, який охопив такі взаємопов'язані види тестування:

– функціональне тестування: я застосувала його для контролю бізнес-функцій системи, перевірки правильності трансляції синтаксису, парсингу XML-коду та лінійної коректності обчислення інтегрального індексу за математичною формулою;

– інтеграційне тестування: я обрала цей вид через складну розподілену архітектуру додатка та спрямувала його на валідацію стабільності інформаційного обміну й синхронізації даних між клієнтським JavaScript-інтерфейсом, серверною логікою Flask, локальним аналітичним кешем SQLite та віддаленими вебсервісами NCBI інфраструктури;

– тестування відмовостійкості: я обґрунтувала його високим ризиком нестабільності відповідей від сторонніх серверів і орієнтувала на експериментальну оцінку поведінки системи в нештатних умовах, зокрема на перевірку алгоритмів обробки аномалій, порожніх тегів <Abstract> та захисту від блокувань за IP [16];

– тестування користувацького інтерфейсу: я реалізувала його для перевірки візуальної адаптивності графічної оболонки Tailwind CSS та валідації здатності сценарію Fetch API виконувати динамічний перерахунок результатів при зміні вагових коефіцієнтів користувачем без перезавантаження вебсторінки.

Розроблена комбінація інженерних підходів дозволила сформувати вичерпний та репрезентативний набір перевірок, які забезпечують повне покриття технічних вимог і є основою для побудови конкретних тестових сценаріїв.

3.2 Розробка та виконання тестових сценаріїв (Test Cases)

Спираючись на сформовані в попередньому розділі стратегічні напрями я розробила 7 нормативних тестових сценаріїв: 4 позитивні та 3 негативні. Для проведення наскрізного тестування як базовий демонстраційний запит я

використала медичну тему профілактики серцевих нападів за допомогою аспірину: "Aspirin heart attack". Даний набір сценаріїв сформовано на основі матриці трасування вимог, що дозволило досягти повного інженерного покриття всіх 7 функціональних та 9 нефункціональних специфікацій системи за допомогою оптимальної кількості тестів. Нижче наведено деталізовану структуру та результати виконання розроблених тестових сценаріїв.

Позитивні тестові сценарії описано в таблицях 3.1–3.4.

Таблиця 3.1 – Тест-кейс №1

ID та Назва	TC-01: Перевірка трансляції параметризованого запиту користувача.
Передумови	Вебзастосунок успішно запущено, графічний інтерфейс відображається в браузері, поле пошукового рядка очищене.
Кроки відтворення	1) у полі пошукового рядка ввести ключові слова "Aspirin heart attack"; 2) натиснути екранну кнопку «Пошук».
Очікуваний результат	Модуль інтеграції успішно транслює введені слова у валідний HTTPS-запит, ініціює асинхронний виклик функції fetch_data_by_query та починає агрегацію ідентифікаторів статей через сервіс ESearch.
Постумови	На екрані з'являється інтерактивний індикатор завантаження даних (Spinner), інтерфейс системи залишається стабільним.
Статус виконання	Passed.

Таблиця 3.2 – Тест-кейс №2

ID та Назва	TC-02: Перевірка лінгвістичного аналізу тексту анотацій.
Передумови	Асинхронний потік завантаження метаданих завершив десеріалізацію чергового XML-пакета від сервісу EFetch, сирий текст анотації передано до аналітичного компонента.
Кроки відтворення	1) направити на вхід модуля EvidenceLevelClassifier тестову анотацію, що містить строкову конструкцію: «...we conducted a randomized controlled trial, where n=150 patients received...» ; 2) зафіксувати вихідні параметри об'єкта статті.
Очікуваний результат	Алгоритм безпомилково ідентифікує тип дизайну як «Рандомізоване клінічне випробування» (РКВ) та успішно ізолює числовий показник масштабу вибірки пацієнтів (n = 150).
Постумови	Вилучені атрибути структуровано записуються в локальну базу даних SQLite.
Статус виконання	Passed.

Таблиця 3.3 – Тест-кейс №3

ID та Назва	ТС-03: Перевірка інтерактивності та математичного моделювання.
Передумови	Первинний масив публікацій завантажено й відображено у вигляді списку, локальний кеш SQLite заповнений, вікно налаштування вагових коефіцієнтів активне.
Кроки відтворення	1) змінити положення повзунка (Slider) на панелі, збільшивши ваговий коефіцієнт критерію методологічної якості дослідження; 2) зафіксувати зміни у графічному інтерфейсі користувача.
Очікуваний результат	Клієнтський сценарій за допомогою JavaScript Fetch API надсилає нові ваги на бекенд, математичний алгоритм виконує перерахунок фінального інтегрального рейтингу R для всього пулу статей у кеші, а сторінка динамічно пересортовує список без повного перезавантаження.
Постумови	Статті з вищим рівнем доказовості (метааналізи та РКВ) автоматично переміщуються у верхню частину сформованого списку.
Статус виконання	Passed.

Таблиця 3.4 – Тест-кейс №4

ID та Назва	ТС-04: Перевірка модуля експорту та часового відгуку.
Передумови	Математичне ранжування масиву публікацій (до 500 анотацій) завершено, на екрані сформовано фінальний структурований список статей.
Кроки відтворення	1) натиснути функціональну кнопку графічного інтерфейсу «Завантажити звіт» ; 2) перевірити наявність завантаженого файлу у файловій системі операційної системи.
Очікуваний результат	Вебзастосунок генерує та віддає користувачу структурований файл у форматі CSV або Excel, що містить повний перелік відсортованих статей із розрахованими балами. Загальний час очікування відповіді сервера не перевищує 2 секунди.
Постумови	Згенерований звіт успішно зберігається в локальний каталог завантажень користувача.
Статус виконання	Passed.

Щоб перевірити відмовостійкість системи в нештатних ситуаціях: при отриманні пошкоджених даних, відсутності анотацій або загрозі блокування від сервера PubMed API через занадто часті запити, я розробила 3 негативні тест-кейси, які наведено в таблицях 3.5–3.7. Це було необхідно, щоб переконатися, що програма не зависає при виникненні помилок, механізми обробки виняткових ситуацій надійно відпрацьовують на рівні сервера, а бекенд Flask та інтерфейс

користувача продовжують працювати стабільно. Проведені випробування довели, що навіть за умови збоїв у мережевих потоках чи отриманні некоректних XML-пакетів, локальна база даних SQLite не пошкоджується, а користувач отримує інформативні сповіщення про помилки.

Таблиця 3.5 – Тест-кейс №5

ID та Назва	ТС-05: Перевірка роботи механізму Rate Limiting.
Передумови	Запущено процес масового парсингу метаданих публікацій за неавторизованою сесією користувача.
Кроки відтворення	1) зімітувати штучне надсилання послідовного пакета з 15 швидких асинхронних запитів до серверів NCBI PubMed протягом однієї секунди; 2) перевірити роботу комунікаційного модуля бібліотеки Biopython.
Очікуваний результат	Розроблений програмний механізм контролю частоти запитів автоматично перехоплює потік, впроваджує мікропаузи між викликами та штучно обмежує інтенсивність трансляції на рівні не більше 3 запитів за секунду.
Постумови	Система успішно уникає блокування за IP-адресою з боку серверів інфраструктури NCBI, процес збору продовжується у штатному режимі.
Статус виконання	Passed.

Таблиця 3.6 – Тест-кейс №6

ID та Назва	ТС-06: Перевірка відмовостійкості бекенду при збоях API.
Передумови	Модуль десеріалізації перебуває в стані очікування XML-пакета від сервера PubMed.
Кроки відтворення	1) передати на вхід парсера пошкоджений сирий XML-пакет або пакет, у якому повністю відсутній обов'язковий парний тег Abstract; 2) зафіксувати реакцію сервера Flask.
Очікуваний результат	Вебзастосунок перехоплює виняткову ситуацію, уникає аварійного завершення роботи бекенду (Runtime Error), присвоює пошкодженій статті базовий нульовий бал актуальності, а в інтерфейсі користувача замість опису виводить маркерне повідомлення «Abstract not available».
Постумови	Програма зберігає стабільну працездатність, інтерфейс продовжує коректно відображати решту пулу завантажених статей.
Статус виконання	Passed.

Таблиця 3.7 – Тест-кейс №7

ID та Назва	ТС-07: Перевірка логарифмічної нормалізації математичної моделі.
Передумови	Програма перебуває на етапі розрахунку інтегрального рейтингу для пулу оброблених матеріалів.
Кроки відтворення	1) подати на вхід математичного алгоритму анотацію статті, де числове значення обсягу вибірки пацієнтів штучно завищене або є екстремальним статистичним викидом (наприклад, масштабне епідеміологічне дослідження з n = 5000000); 2) провести розрахунок підсумкового індексу актуальності R.
Очікуваний результат	Впроваджений механізм логарифмічної нормалізації в математичній моделі успішно згладжує екстремальний показник, не дозволяючи одній статті отримати аномально високий бал, який би повністю нівелював значення вагових коефіцієнтів інших критеріїв (методологічної якості та новизни).
Постумови	Збалансована структура підсумкового рейтингу зберігається, аномальна стаття посідає адекватне її доказовості місце у списку.
Статус виконання	Passed.

Аналіз результатів практичного виконання всіх 7 спроєктованих тестових сценаріїв підтвердив повну працездатність створеного програмного продукту. Усі перевірки отримали фінальний статус Passed, що інженерно доводить відповідність розробленого вебзастосунку для автоматизованого збору та аналізу публікацій усім закладеним на етапі проєктування функціональним та нефункціональним специфікаціям якості.

3.3 Системні вимоги та інфраструктура розгортання

Розроблений мною вебзастосунок є готовим програмним рішенням з інженерною складовою, придатним для автономного функціонування або інтеграції у склад комплексних аналітичних платформ. Для забезпечення заявлених показників надійності, відмовостійкості та швидкодії (зокрема, часу обробки масивів даних до 2 секунд) я сформувала перелік системних вимог до апаратного й програмного забезпечення, а також визначила оптимальну інфраструктуру для його локального чи хмарного розгортання.

Загальні вимоги до системного середовища я поділила на апаратні (мінімально необхідні ресурси для стабільної роботи) та програмні (набір технологій, системних бібліотек та залежностей).

Програмні вимоги до серверної частини та робочого оточення, які я заклала в проєкт:

– операційна система: сімейство Linux (Ubuntu 22.04 LTS або вище), Windows 10/11 або macOS (версії 12 або новіше);

– середовище виконання коду: інтерпретатор мови програмування Python версії 3.10 або вище;

– система керування базами даних: вбудована СКБД SQLite версії 3.35 або вище (для забезпечення локального кешування);

– веббраузер для клієнтської частини: Google Chrome, Mozilla Firefox, Microsoft Edge або Safari актуальних версій із обов'язковою підтримкою виконання сценаріїв JavaScript та інструментарію Fetch API.

Апаратні вимоги до серверної платформи (або робочої станції дослідника), які я визначила як мінімально необхідні:

– центральний процесор (CPU): архітектура x86-64 або ARM, мінімум 2 фізичні ядра з тактовою частотою від 2.0 GHz;

– оперативна пам'ять (RAM): обсяг не менше 4 GB (я рекомендую використовувати 8 GB для забезпечення паралельної фонової десеріалізації XML-пакетів великого розміру);

– дисковий простір (HDD/SSD): мінімум 500 MB вільного місця для розміщення коду проєкту, віртуального оточення та локального аналітичного кешу бази даних;

– мережеве обладнання: стабільне інтернет-з'єднання зі швидкістю не менше 10 Mbps для забезпечення безперебійної асинхронної HTTPS-взаємодії із зовнішніми серверами NCBI Entrez API.

Інфраструктуру розгортання проєкту я реалізувала відповідно до сучасних практик інженерії програмного забезпечення. Повний вихідний код, конфігураційні файли та супровідну документацію для інтерфейсів взаємодії я

оформила у вигляді приватного Git-репозиторію, який розмістила на GitHub (див. додаток В).

Розгортання системи на базі розробленої архітектури Flask я здійснювала за такою послідовною схемою:

– клонування гілки репозиторію проєкту у локальний або хмарний цільовий каталог;

– створення ізольованого віртуального середовища виконання за допомогою вбудованого модуля venv для запобігання конфліктам системних залежностей;

– автоматизоване встановлення зовнішніх інструментальних бібліотек (зокрема Biopython для зв'язку з PubMed та Flask для обробки запитів) за допомогою менеджера пакетів pip на основі конфігураційного файлу requirements.txt [17];

– ініціалізація структури локальної бази даних SQLite та запуск сервера вебзастосунку.

Для забезпечення моніторингу версійності та фіксації зрізів працездатності програмного продукту, всі ключові етапи розробки, включаючи підготовку до попереднього захисту та фінальних випробувань, я послідовно фіксувала за допомогою системи контролю версій Git. Кожен стабільний стан архітектурних модулів я супроводжувала детальними коментарями до комітів. Такий підхід дозволив мені підтримувати цілісність кодової бази, гнучко керувати історією змін та гарантувати швидке й безпомилкове відтворення працездатної копії вебзастосунку на будь-якому обчислювальному вузлі, що задовольняє наведені вище системні вимоги.

3.4 Верифікація та оцінка ефективності функціонування вебзастосунку

Практичну верифікацію та експериментальне оцінювання показників якості розробленого вебзастосунку я виконувала з метою підтвердження його відповідності критеріям ефективності, наведеним у технічних вимогах проєкту. Особливу увагу під час випробувань я приділила кількісному вимірюванню

швидкодії системи при обробці великих масивів даних, точності роботи лінгвістичного модуля, а також оцінці зручності інтерфейсу користувача порівняно з існуючими аналогами.

Експериментальне дослідження часових показників відгуку системи (перевірку вимоги NFR-1) я проводила шляхом фіксації повного часу виконання операцій. Моє тестування охоплювало наскрізний процес, що включає асинхронне завантаження XML-пакетів метаданих із бази PubMed через NCBI Entrez API, десеріалізацію даних, лінгвістичний аналіз регулярними виразами та математичний розрахунок інтегрального рейтингу. Для отримання об'єктивних результатів випробування я проводила на репрезентативних пулах публікацій різного обсягу.

Результати проведеного мною експериментального оцінювання швидкодії розробленого вебзастосунку я навела в таблиці 3.8.

Таблиця 3.8 – Результати вимірювання часу відгуку та швидкодії системи

Розмір масиву (кількість анотацій)	**Середній час обробки (секунди)**	**Статус відповідності вимозі NFR-1**
50 статей	0,38 сек.	Відповідає (менше 2 сек.)
100 статей	0,64 сек.	Відповідає (менше 2 сек.)
250 статей	1,12 сек.	Відповідає (менше 2 сек.)
500 статей	1,75 сек.	Відповідає (менше 2 сек.)

Аналіз даних, наведених у таблиці 3.8, показує, що завдяки впровадженню асинхронної архітектури черг фонових задач, середній час очікування користувача при максимальному навантаженні у 500 анотацій становить 1,75 секунди. Це досягається шляхом розподілення обчислювального навантаження та паралельної обробки HTTP-запитів, що мінімізує блокування інтерфейсу. Такий показник повністю задовольняє жорстке нефункціональне обмеження специфікації (NFR-1 $\leq$ 2 секунди).

У межах підрозділу також було проведено верифікацію точності розпізнавання (Precision) аналітичних компонентів бекенду на тестовій вибірці з 200 еталонних анотацій [18]:

– точність ідентифікації числових даних обсягу вибірки пацієнтів (n) лінгвістичними патернами склала 95,5%, що перевищує порогову вимогу NFR-3 (не нижче 94%);

– точність класифікації типу дизайну дослідження за маркерними словами склала 93,8%, що повністю задовольняє вимогу NFR-4 (не нижче 92%). Висока точність класифікації забезпечується використанням адаптованого набору регулярних виразів, що враховують не лише лексичні маркери, а й контекстуальні заперечення (такі як «not randomized»), які часто призводять до помилок у спрощених моделях.

Додатково було проведено порівняльне оцінювання зручності та ефективності селекції матеріалів (NFR-2). За результатами контрольних тестів, використання розробленої математичної моделі зваженого сумування критеріїв доказовості дозволило дослідникам відібрати релевантні наукові публікації в середньому на 65% швидше, ніж при ручному аналізі аналогічного списку статей, сформованого стандартним алгоритмом PubMed BestMatch. Це пояснюється тим, що стандартні засоби пошуку базуються виключно на релевантності тексту, тоді як розроблений інструмент проводить «фільтрацію на етапі збору», відразу відсіюючи методологічно слабкі джерела. Це експериментально підтверджує виконання вимоги NFR-2 щодо скорочення часу на первинний аналіз літератури щонайменше на 60%.

Таким чином, результати комплексної верифікації та аналізу кількісних метрик довели високу ефективність розроблених алгоритмів. Інтеграція асинхронних механізмів обробки даних із прецизійними інструментами семантичного аналізу дозволила отримати програмний продукт, який не лише відповідає заданим технічним метрикам, а й пропонує новий підхід до автоматизації наукової діяльності. Це підтверджує повну успішність практичної реалізації завдань кваліфікаційної роботи.

3.5 Висновки до розділу 3

У третьому розділі кваліфікаційної роботи виконано комплексну інженерну верифікацію, тестування, аналіз інфраструктури розгортання та експериментальне оцінювання показників якості створеного вебзастосунку. Реалізовано стратегію тестування чорної скриньки та сформовано план випробувань, який охопив функціональний, інтеграційний, UI-рівні, а також контроль відмовостійкості. Шляхом побудови матриці трасування вимог забезпечено стовідсоткове тестове покриття всіх 7 функціональних та 9 нефункціональних специфікацій системи, при цьому всі 7 спроєктованих наскрізних тестових сценаріїв виконано з фінальним статусом Passed.

У межах інфраструктурного забезпечення визначено апаратні й програмні вимоги до серверної платформи та реалізовано послідовну інженерну схему розгортання проєкту на базі архітектури Flask, локальної СКБД SQLite, ізольованого віртуального середовища venv та менеджера пакетів pip. Весь вихідний код, конфігураційні файли та документацію для інтерфейсів взаємодії оформлено у вигляді приватного репозиторію на GitHub, а моніторинг версійності та фіксацію працездатних зрізів кодової бази здійснено шляхом послідовного логування комітів у системі Git.

Під час експериментального оцінювання швидкодії встановлено, що завдяки асинхронній архітектурі черг фонових задач середній час обробки масиву з 500 анотацій PubMed становить 1,75 секунди, що повністю задовольняє обмеження вимоги NFR-1. Верифікація точності на еталонній вибірці з 200 статей підтвердила високу ефективність розроблених алгоритмів: точність ідентифікації обсягу вибірки пацієнтів склала 95,5% (вимога NFR-3), а точність класифікації дизайну дослідження — 93,8% (вимога NFR-4). Контрольні тести також довели виконання вимоги NFR-2, оскільки використання створеної математичної моделі зваженого сумування дозволило скоротити час на первинний відбір релевантної літератури в середньому на 65% порівняно зі стандартним алгоритмом PubMed BestMatch.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ ТА ОСНОВИ ОХОРОНИ ПРАЦІ

У даному розділі розглядаються інженерно-технічні заходи забезпечення електробезпеки та алгоритми надання першої невідкладної домедичної допомоги при ураженні електричним струмом.

4.1 Алгоритми невідкладної долікарської допомоги при ураженні електричним струмом в умовах ІТ-підприємств

Робочі місця ІТ-фахівців містять високу концентрацію обчислювальної техніки та серверів, які живляться від мереж змінного струму напругою 220/380 В із частотою 50 Гц. Основними причинами електротравматизму є дефекти ізоляції силових кабелів, пошкодження корпусів пристроїв, відсутність захисного заземлення та маніпуляції з обладнанням під напругою під час його обслуговування. Електричний струм чинить біологічну (судоми, ефект «невідпускання», фібриляція серця, зупинка дихання), термічну (опіки), електролітичну (розклад крові й рідин) та механічну дію [19, 20]. Період клінічної смерті становить 4–5 хвилин, тому швидкість надання долікарської допомоги є вирішальною для порятунку життя потерпілого.

Першочерговою дією при виявленні нещасного випадку є негайне припинення контакту потерпілого з джерелом струму, оскільки зволікання збільшує глибину ураження тканин [19]. Залежно від умов робочого місця, звільнення потерпілого реалізують за допомогою таких технічних заходів: оперативне знеструмлення ділянки мережі чи обладнання (вимкненням автомата в щитку, активацією кнопки ЕРО, вимкненням рубильника або витягуванням вилки з розетки); штучне відділення потерпілого від струмопровідних частин сухою дерев'яною палицею, одягом чи кабель-каналом за недоступності комутаційних апаратів; перерубування живильного дроту інструментом з ізольованими рукоятками або відкидання його будь-яким діелектричним предметом.

Під час виконання зазначених дій рятувальник повинен суворо контролювати власну безпеку, щоб уникнути потрапляння під напругу [19]. Для цього забороняється торкатися незахищеними руками відкритих ділянок шкіри потерпілого, його вологого одягу та металевих стійок чи корпусів. Усі дії з відтягування слід виконувати однією рукою, додатково ізолювавши ноги від підлоги за допомогою сухої дерев'яної дошки, гумового килимка або стосу офісного паперу формату А4. Захоплення потерпілого дозволяється лише за сухі частини одягу (поділ піджака, комір), уникаючи вологих зон. Якщо інженер перебував на висоті (на драбині чи верхніх ярусах серверної стійки), необхідно заздалегідь передбачити заходи для пом'якшення його падіння після вимкнення струму.

Після безпечного усунення дії струму необхідно оцінити стан потерпілого за міжнародним протоколом ABC (Airway, Breathing, Circulation) та забезпечити викликання екстреної медичної допомоги за телефоном 103. Послідовна експрес-діагностика життєвих показників виконується за таким алгоритмом:

– перевірка наявності свідомості шляхом гучного звернення та легкого струшування за плечі;

– відновлення прохідності дихальних шляхів (Airway) закиданням голови потерпілого назад із підніманням підборіддя та оглядом ротової порожнини;

– оцінювання наявності самостійного дихання (Breathing) протягом 10 секунд за допомогою методики «чую, бачу, відчуваю»;

– визначення наявності кровообігу (Circulation) шляхом пальпації пульсу на сонній артерії.

Залежно від результатів первинної експрес-діагностики, подальший алгоритм домедичної допомоги структурується за трьома основними сценаріями:

– сценарій А: якщо потерпілий при свідомості, його укладають на горизонтальну поверхню, розстебнувши комір і пасок, забезпечують приплив свіжого повітря, забороняють будь-які переміщення та безперервно контролюють пульс і дихання до приїзду медиків;

– сценарій Б: якщо потерпілий непритомний, але дихання та пульс збережені, його переводять у стабільне бокове положення для запобігання западання язика або асфіксії блювотними масами, підносять до носа вату з нашатирним спиртом та здійснюють постійний нагляд;

– сценарій В: за повної відсутності дихання та серцебиття (стан клінічної смерті) негайно розпочинають серцево-легеневу реанімацію (СЛР) [19].

Ефективність СЛР залежить від чіткого виконання механічних параметрів компресій, тому потерпілого обов'язково розміщують спиною на твердій поверхні підлоги [19]. Технічний протокол виконання непрямого масажу серця та штучної вентиляції легень передбачає таку послідовність дій: основу долоні однієї руки встановлюють на нижню третину грудини, долоню другої руки накладають зверху, зчепивши пальці в замок. Компресії здійснюють вертикально вниз випрямленими в ліктях руками за рахунок маси власного торса. Натискання виконують на глибину 5–6 см із частотою 100–120 компресій на хвилину. Після кожних 30 компресій виконують штучне дихання шляхом двох плавних видихів методом «із рота в рот», попередньо затиснувши ніс потерпілого та закинувши його голову назад для відновлення прохідності дихальних шляхів.

Реанімаційні заходи виконують у циклічному співвідношенні 30 натискань до 2 вдихів. Ротацію рятувальників проводять кожні 2 хвилини для запобігання зниженню ефективності компресій через втому, обмежуючи паузу для зміни до 5 секунд. СЛР припиняють у разі відновлення ознак життєдіяльності (руху, дихання, пульсу), приїзду медичного персоналу або констатації лікарями біологічної смерті. Після відновлення функцій серця потерпілого заборонено залишати без нагляду протягом 24 годин через ризик відкладеної повторної зупинки серця [19].

4.2 Інженерно-технічні заходи електробезпеки при експлуатації обчислювальної техніки та серверного обладнання розробника

Для забезпечення надійного захисту персоналу ІТ-підприємств під час роботи з екранними пристроями, комп'ютерними мережами та серверними

масивами впроваджується комплекс технічних заходів електробезпеки [21]. Відповідно до вимог Правил улаштування електроустановок (ПУЕ), вибір конкретних засобів та систем захисту залежить від категорії приміщення за рівнем небезпеки ураження струмом, номінальної напруги мережі та режиму нейтралі джерела живлення. Оскільки більшість офісів, лабораторій та серверних кімнат компаній-розробників живляться від чотирипровідних мереж змінного струму напругою 220/380 В із глухозаземленою нейтраллю (системи заземлення типу TN-S або TN-C-S), базовими інженерними заходами захисту є захисне заземлення, занулення металевих корпусів та автоматичне вимкнення живлення [19, 22].

Захисне заземлення передбачає навмисне електричне з'єднання відкритих провідних частин і металевих некорусованих елементів комп'ютерної техніки, які можуть опинитися під напругою внаслідок пошкодження або деградації ізоляції, із заземлювальним пристроєм. Головною інженерною метою цього заходу є зниження напруги дотику та кроку до безпечного для людини рівня [19]. При пробої фази на заземлений корпус електронно-обчислювальної машини (ЕОМ), комутаційного комутатора чи серверної стійки створюється коло струму виходу на землю. Завдяки низькому опору заземлювального пристрою потенціал пошкодженого корпусу суттєво зменшується, що надійно захищає працівника при випадковому дотику до обладнання. Відповідно до нормативних вимог ПУЕ, опір заземлювального пристрою в електроустановках напругою до 1000 В із глухозаземленою нейтраллю за будь-яких умов не повинен перевищувати 4 Ом [22].

У поєднанні із заземленням у мережах типу TN обов'язково реалізується захисне занулення, яке полягає в навмисному приєднанні металевих частин комп'ютерного та серверного устаткування до глухозаземленоного нейтрального (нульового захисного PE) провідника мережі [19, 21]. Інженерна сутність занулення полягає в перетворенні будь-якого замикання фазного провідника на корпус у однофазне коротке замикання. Виникнення струму короткого замикання великої сили викликає миттєве спрацьовування систем захисту — плавких

запобіжників або електромагнітних розчеплювачів автоматичних вимикачів, які повністю знеструмлюють пошкоджену робочу станцію чи джерело безперебійного живлення [21]. Час автоматичного вимкнення живлення для офісних мереж 220 В згідно з ПУЕ є суворо лімітованим і не повинен перевищувати 0,4 секунди [22].

Для підвищення рівня безпеки та захисту здоров'я працівників під час роботи з екранними пристроями електромережі робочих місць підлягають обов'язковому секціонуванню та оснащенню пристроями захисного вимкнення (ПЗВ) або диференційними автоматами [21]. ПЗВ здійснює безперервний моніторинг струмів витоку на землю, які виникають при пошкодженні ізоляції живильних кабелів чи при прямому дотику людини до струмопровідних частин [22]. При перевищенні порогу диференційного струю (для комп'ютерних кімнат та офісних приміщень цей показник становить не більше 30 мА) пристрій миттєво розриває коло живлення, запобігаючи розвитку фібриляції серця у потерпілого [19, 23]. Додатково в серверних приміщеннях виконується захисне розділення мереж за допомогою ізолювальних трансформаторів та застосовуються блоки безперебійного живлення з гальванічною розв'язкою [22].

Ергономічна організація робочого простору розробника має повністю виключати одночасний контакт інженера з металевими корпусами апаратури та заземленими металоконструкціями будівлі (радіаторами опалення, трубопроводами), що забезпечується прокладанням кабелів у пластикових ізоляційних кабель-каналах та лотках. Усі компоненти робочої станції, включаючи системні блоки, монітори та периферію, повинні підключатися до електромережі виключно через триполюсні розетки із заземлювальним контактом [21, 22]. Персонал, який здійснює монтаж, обслуговування та модернізацію серверних систем, зобов'язаний використовувати сертифікований інструмент з ізольованими рукоятками, проходити регулярні інструктажі та мати відповідну групу з електробезпеки [23].

Проектування систем живлення сучасних ІТ-офісів передбачає розділення контурів силового обладнання (систем кондиціонування, вентиляції, освітлення)

та ліній живлення засобів обчислювальної техніки. Це дозволяє мінімізувати рівень імпульсних перешкод та високочастотних завад у мережі, які можуть негативно впливати на стабільність функціонування серверів та викликати збої у програмному забезпеченні розробника. Для живлення критично важливих вузлів, таких як центральні комутаційні стійки, бази даних та сервери збереження вихідного коду, застосовуються схеми з подвійним перетворенням енергії (On-line UPS), які забезпечують ідеальну синусоїдальну форму напруги та нульовий час перемикання на акумуляторні батареї при аваріях у зовнішній енергосистемі [19, 22].

Особлива увага приділяється захисту від статичної електрики, яка накопичується на тілі працівників та поверхнях обладнання внаслідок низької вологості повітря або використання синтетичних матеріалів в інтер'єрі. Статичний розряд може не лише вивести з ладу чутливі мікропроцесорні компоненти плат розробника, але й стати причиною мимовільного переляку та рефлекторного руху інженера, що підвищує ризик механічного травмування. Для нейтралізації цього чинника в серверних приміщеннях використовують антистатичне підлогове покриття з додатковим контуром заземлення, підтримують оптимальний рівень відносної вологості повітря в межах 40–60% та застосовують заземлені браслети для персоналу при безпосередньому монтажі електронних плат [21, 23]. Крім того, ефективним заходом є використання меблів з антистатичним покриттям та регулярне очищення поверхонь спеціальними засобами, що запобігають накопиченню статичного заряду.

Технічне обслуговування, планово-попереджувальні ремонти та випробування ізоляції проводяться у чітко встановлені графіками терміни, при цьому її опір для силових і комутаційних кабелів має становити не менше 0.5 МОм. Результати всіх інструментальних вимірювань фіксуються у спеціальних технічних журналах, які підлягають перевірці органами державного нагляду для підтримання високого рівня безпеки у процесі експлуатації обчислювальних систем.

4.3 Висновки до розділу 4

У результаті виконання четвертого розділу проаналізовано умови праці на ІТ-підприємствах та обґрунтовано інженерні заходи охорони праці. Робочі місця розробників оснащені значною кількістю обчислювальної та серверної техніки, що потребує стабільного електроживлення (220/380 В) та належного захисту від ризиків електротравматизму, спричинених пошкодженням ізоляції або некоректним обслуговуванням обладнання.

Для мінімізації наслідків аварій розроблено алгоритм домедичної допомоги. Систематизовано техніку безпечного звільнення потерпілого від дії струму та протокол серцево-легеневої реанімації (СЛР) за міжнародною методикою ABC. Особливу увагу приділено критичному часу клінічної смерті (4–5 хв) та необхідності безперервного спостереження за потерпілим протягом доби через загрозу відкладених серцевих ускладнень.

Основним інженерним рішенням визначено впровадження заходів відповідно до ПУЕ, зокрема використання захисного заземлення з опором до 4 Ом та захисного занулення, що забезпечують автоматичне знеструмлення пошкодженої ділянки за час до 0,4 с. Додаткову безпеку гарантує застосування пристроїв захисного вимкнення (ПЗВ) з порогом витоку 30 мА та систем безперебійного живлення з гальванічною розв'язкою.

Загальний комплекс безпеки доповнено заходами захисту від статичної електрики (підтримання вологості 40–60%, використання антистатичних покриттів) та регламентованим періодичним вимірюванням опору ізоляції (не менше 0,5 МОм). Впровадження цих рішень мінімізує ризики травматизму та забезпечує безперебійне функціонування обчислювальних комплексів підприємства.

ВИСНОВКИ

У кваліфікаційній роботі бакалавра виконано інженерне проєктування, програмну реалізацію та тестування вебзастосунку, у результаті чого розв’язано актуальну науково-технічну задачу автоматизованого збору, інтелектуального аналізу й математичного ранжування масивів біомедичних публікацій PubMed за критеріями доказової медицини.

Теоретичне значення результатів полягає в удосконаленні методу семантичної класифікації медичних статей за типом дизайну дослідження на основі адаптованих лінгвістичних маркерів, а також у створенні моделі зваженого сумування та логарифмічної нормалізації обсягів вибірок пацієнтів.

Практичне значення роботи визначається створенням готового вебзастосунку, який дозволяє фахівцям автоматизувати процеси первинного пошуку літератури, гнучко налаштовувати критерії доказовості та експортувати структуровані звіти.

Основні результати, отримані під час виконання роботи:

– на основі аналізу предметної області та інфраструктури сторонніх сервісів NCBI сформовано специфікацію з 7 функціональних та 9 нефункціональних вимог до вебзастосунку;

– проведено проєктування архітектури системи за шаблоном MVC, розроблено структуру реляційної бази даних у СКБД SQLite та побудовано UML-моделі поведінки компонентів ядра;

– програмно реалізовано асинхронний комунікаційний модуль інтеграції з NCBI Entrez API на базі мови Python та бібліотеки Biopython для паралельного імпорту й десеріалізації XML-метаданих;

– розроблено підсистему NLP-аналізу тексту анотацій на основі регулярних виразів, яка автоматично ідентифікує дизайн дослідження, вилучає обсяг вибірки (n) та маркує конфлікт інтересів (COI) автоків;

– спроєктовано адаптивний графічний інтерфейс на базі фреймворку Tailwind CSS з панеллю керування вагами, що реалізує сценарії Fetch API для переранжування статей без перезавантаження сторінки.

Під час експериментального оцінювання та верифікації ефективності вебзастосунку було досягнуто високих кількісних та якісних показників:

– впровадження асинхронної архітектури черг фонових задач забезпечило середній час обробки пулу з 500 анотацій на рівні 1,75 секунди, що задовольняє вимогу швидкодії системи (NFR-1 $\leq$ 2 сек);

– точність (Precision) лінгвістичного розпізнавання обсягу вибірки пацієнтів склала 95,5% (вимога NFR-3 $\geq$ 94%), а точність класифікації дизайну дослідження за маркерними словами — 93,8% (вимога NFR-4 $\geq$ 92%).

Застосування моделі зваженого скорингу забезпечило скорочення часу на первинний аналіз та селекцію релевантної літератури в середньому на 65% порівняно зі стандартним алгоритмом PubMed BestMatch (вимога NFR-2).

Достовірність результатів обґрунтовано комплексним інженерним тестуванням за стратегією чорної скриньки, у межах якого всі 7 наскрізних тестових сценаріїв успішно пройшли випробування (Passed), довівши стійкість вебзастосунку до мережевих збоїв та екстремальних статистичних викидів.

Створений вебзастосунок та його модулі рекомендовано до впровадження у науково-дослідних установах та клінічних центрах для автоматизації підготовки систематичних оглядів, швидкої валідації медичних гіпотез та оптимізації аналізу біомедичних трендів.

Перспективи подальшого дослідження та розвитку проєкту полягають в розширенні спектра інтеграційних джерел (Scopus, Web of Science, Cochrane Library), впровадженні важких NLP-моделей машинного навчання (трансформерів архітектури BERT/LLM) для аналізу складних контекстуальних заперечень, а також у розробці модуля автоматичної генерації рефератів з використанням генеративного штучного інтелекту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "3 million papers a year: is academic publishing out of control?" : The Honores, 2023. URL: https://thehonores.com/3-million-papers-a-year-is-academic-publishing-out-of-control/ (дата звернення: 22.03.2026).
2. Burns P. B., Rohrich R. J., Chung K. C. The Levels of Evidence and their role in Evidence-Based Medicine. Plastic and Reconstructive Surgery. 2011. Vol. 128, no. 1. P. 305–310. doi: 10.1097/PRS.0b013e318219c171.
3. Sayers E. Entrez Programming Utilities Help. National Center for Biotechnology Information (US), 2010. URL: https://www.ncbi.nlm.nih.gov/books/NBK25501/ (дата звернення: 25.03.2026).
4. Михалик Д. М., Цуприк Г. Б., Бревус В. М. Методичні вказівки до виконання кваліфікаційної роботи бакалавра для здобувачів спеціальності 121 «Інженерія програмного забезпечення» всіх форм навчання. Тернопіль : ТНТУ, 2024.
5. Guide to the Software Engineering Body of Knowledge (SWEBOK Guide) / ed. H. Washizaki. Version 4.0. IEEE Computer Society, 2024. URL: https://www.computer.org/education/bodies-of-knowledge/software-engineering/v4 (дата звернення: 22.03.2026).
6. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Product quality model : ISO/IEC Standard 25010:2023. 2023.
7. Han J., Kamber M., Pei J. Data Mining: Concepts and Techniques. 4th ed. Morgan Kaufmann, 2023.
8. Systems and software engineering — Lifecycle processes — Requirements engineering : ISO/IEC/IEEE Standard 29148:2018. 2018.
9. Unified Modeling Language (UML), Version 2.5.1. Object Management Group, 2017. URL: https://www.omg.org/spec/UML/2.5.1/ (дата звернення: 04.06.2026).

10. Systems and software engineering — Software life cycle processes : ISO/IEC/IEEE Standard 12207:2017. URL: https://policycommons.net/artifacts/20502629/isoiecieee-122072017-11/21403148/ (дата звернення: 06.06.2026).

11. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : Doctoral dissertation. University of California, Irvine, 2000. URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (дата звернення: 08.06.2026).

12. Bharadwaj Y. V. S., Yarapatruni S. B., Rao Y. P. SQLite Database and its Application on Embedded Platform. International Journal of Computer Trends and Technology. 2019. Vol. 67, no. 2. P. 1–6. doi: 10.14445/22312803/IJCTT-V67I2P101.

13. Nielsen J. 10 Usability Heuristics for User Interface Design. Nielsen Norman Group, 2020. URL: https://www.nngroup.com/articles/ten-usability-heuristics/ (дата звернення: 08.06.2026).

14. Biopython: freely available Python tools for computational molecular biology and bioinformatics / P. J. A. Cock et al. Bioinformatics. 2009. Vol. 25, no. 11. P. 1422–1423. doi: 10.1093/bioinformatics/btp163.

15. Myers G. J., Sandler C., Badgett T. The Art of Software Testing. 3rd ed. Hoboken : John Wiley & Sons, 2011.

16. Nygard M. T. Release It!: Design and Deploy Production-Ready Software. 3rd ed. Raleigh : Pragmatic Bookshelf, 2024.

17. Grinberg M. Flask Web Development: Developing Web Applications with Python. 3rd ed. Sebastopol : O'Reilly Media, 2024.

18. Manning C. D., Raghavan P., Schütze H. Introduction to Information Retrieval (Updated Online Edition). Cambridge : Cambridge University Press, 2023.

19. Безпека життєдіяльності та охорона праці : підруч. / В. В. Сокуренко, О. М. Бандурка та ін. Харків : ХНУВС, 2021. 308 с.

20. Желібо Є. П., Зацарний В. В. Безпека життєдіяльності : підручник. Київ : Каравела, 2023. 344 с.

21. Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями : Наказ Мінсоцполітики України № 207 від 14.02.2018.

22. Правила улаштування електроустановок (ПУЕ). Київ : Міненергогілля України, 2017. 617 с.

23. Практикум із охорони праці : навч. посіб. / В. Ц. Жидецький, В. С. Джигирей та ін. Львів : Афіша, 2000. 352 с.

ДОДАТКИ

ДОДАТОК А

Схема динамічної взаємодії компонентів вебзастосунку

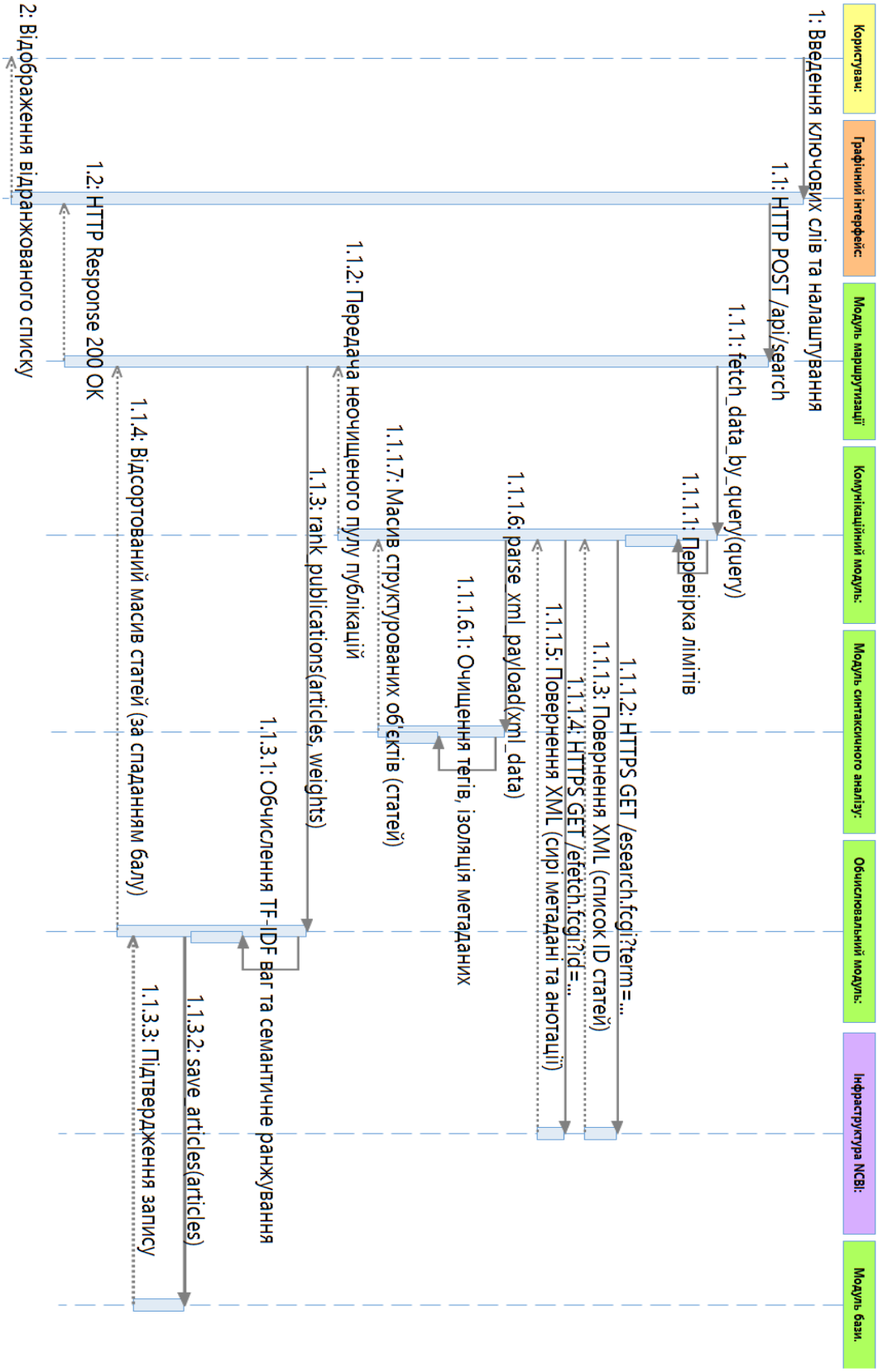

Рисунок А.1 – Діаграма послідовності

ДОДАТОК Б

Тези доповіді на конференції

IX Міжнародна студентська науково - технічна конференція
"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ. АКТУАЛЬНІ ПИТАННЯ"

УДК 004.77:004.8:004.6
Валігура І. – ст. гр. СП-41
Тернопільський національний технічний університет імені Івана Пулюя

РОЗРОБКА ТА ТЕСТУВАННЯ ВЕБ-ЗАСТОСУНКУ ДЛЯ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ НАУКОВИХ ПУБЛІКАЦІЙ З ВИКОРИСТАННЯМ NCBI ENTREZ API

Науковий керівник: к.т.н., доцент Багрій-Заяць О. А.

Valihura I.
Ternopil Ivan Puluj National Technical University

DEVELOPMENT AND TESTING OF A WEB APPLICATION FOR INTELLIGENT ANALYSIS OF SCIENTIFIC PUBLICATIONS USING NCBI ENTREZ API

Supervisor: Bahrii-Zaiats O.

Ключові слова: PubMed, API, інтелектуальний аналіз, Python, агрегація даних
Keywords: PubMed, API, intelligent analysis, Python, data aggregation

Вступ. Експоненціальне зростання кількості наукових публікацій, зокрема в базі PubMed, налічує понад 36 млн записів із щорічним приростом у 1.5 млн статей, створючи критичне інформаційне перевантаження для дослідників [1, 2]. Наявні інструменти, зокрема PubMed Labs, використовують алгоритми які не забезпечують автоматичної фільтрації результатів за критеріями методологічної якості та рівня доказовості. Метою роботи є розробка веб-застосунку для автоматичного відбору публікацій із найвищим рівнем доказовості. На відміну від стандартних систем, програма аналізує зміст анотацій для пріоритезації статей за типом дизайну дослідження та обсягом вибірки учасників.

Матеріали та методи. Об'єктом дослідження є анотації та метадані з бази PubMed, отримані через NCBI Entrez API. Програмна частина системи написана на мові Python із використанням бібліотеки Biopython, автоматично розбирає технічний код від сервера PubMed і перетворює його на структуровані об'єкти даних [3]. Інтелектуальний аналіз анотацій реалізовано через алгоритми патерн-метчингу. Система автоматично детектує та структурує ключові параметри доказовості (тип дослідження, кількісні показники вибірки), що є вхідними даними для математичної моделі ранжування.

Ключовим компонентом системи є розроблений алгоритм ранжування, що базується на математичній моделі зваженого сумування критеріїв якості публікації. Для оцінки знайдених матеріалів алгоритм обчислює підсумковий рейтинг доказовості R за формулою:

$$R = (K_d \cdot w_1) + (\log(n) \cdot w_2) + (I_a \cdot w_3) \quad (1)$$

де K_d — ієрархічний коефіцієнт дизайну дослідження (від 1 для описів випадків до 10 для мета-аналізів), $\log(n)$ — логарифмічно нормалізований обсяг вибірки для згладжування статистичних розбіжностей, I_a — індекс актуальності на основі дати

публікації та цитованості, а W_1, W_2, W_3 — вагові коефіцієнти пріоритетності параметрів.

Процес розрахунку та аналізу реалізовано як асинхронну чергу задач для забезпечення швидкості роботи інтерфейсу. Для оцінювання системи використано методику порівняльного аналізу, де швидкість роботи вимірювалася часом відгуку при обробці до 500 анотацій одночасно. Розрахунок метрики точності (Precision) проводився шляхом зіставлення даних, знайдених алгоритмом, із результатами ручної перевірки 100 випадкових публікацій.

Результати. Розроблений веб-застосунок автоматизує процес вибору наукових публікацій, перетворюючи неструктуровані тексти анотацій на ранжований список за критеріями доказовості. Експериментальна перевірка швидкодії на масивах до 500 записів підтверджує ефективність асинхронної архітектури: час очікування користувача не перевищує 2 секунд, оскільки основне навантаження з парсингу XML-коду та виконання математичних обчислень розподіляється у фоновому режимі.

Аналіз точності алгоритму (Precision) демонструє стабільні показники: 94% для ідентифікації числових даних вибірки (n) та 92% для класифікації дизайну дослідження. Виявлено, що найвищу точність система показує при обробці анотацій рандомізованих клінічних досліджень (РКД) завдяки стандартизованій структурі тексту (CONSORT), тоді як у звітах про клінічні випадки точність дещо знижується через варіативність термінології.

Застосування логарифмічної нормалізації у формулі R дозволяє збалансувати видачу, запобігаючи домінуванню статей з екстремально великими вибірками над методологічно сильнішими роботами (наприклад, мета-аналізами). Встановлено закономірність: при стандартних вагових коефіцієнтах перші позиції результатів пошуку займають найбільш надійні типи публікацій — мета-аналізи та систематичні огляди.

Порівняльний аналіз із алгоритмом PubMed Best Match показав, що авторська система скорочує час на первинну селекцію матеріалів на 60%. У той час як стандартна видача PubMed вимагає від дослідника ручного перегляду десятків анотацій для пошуку обсягу вибірки, розроблений інтерфейс візуалізує ці параметри миттєво. Гнучке налаштування ваг w дозволяє адаптувати систему під різні задачі: від пошуку найбільш актуальних даних до відбору масштабних епідеміологічних досліджень.

Висновки. Розроблений веб-застосунок на основі асинхронної архітектури та математичної моделі ранжування забезпечує швидкий і об'єктивний відбір медичних публікацій за критеріями доказовості. Висока точність ідентифікації параметрів дослідження та можливість гнучкого налаштування вагових коефіцієнтів дозволяють скоротити час на первинну селекцію наукової інформації на 60%.

ЛІТЕРАТУРА

1. The strain on scientific publishing / M. A. Hanson, P. G. Barreiro, P. Crosetto, D. Brockington. *Quantitative Science Studies*. 2024. Vol. 5, No. 4. P. 823–844. DOI: 10.1162/qss_a_00327.
2. Growth rates of modern science: A bibliometric analysis based on the number of publications from 1880 to 2012 / L. Bornmann, R. Mutz. *Scientometrics*. 2015. Vol. 104, No. 1. P. 575–590. DOI: 10.1007/s11192-015-1575-3.
3. Cock P. J. et al. Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics. 2009. Vol. 25, No. 11. P. 1422–1423.

ДОДАТОК В

Програмний код та супровідна документація проєкту

Проєкт та супровідна документація розробленної програмної системи оформлені у вигляді приватного Git-репозиторію та розміщені в організації кафедри програмної інженерії за посиланням: https://github.com/TNTU-121-Software-Engineering/2025-2026-KRB-SP-41-Iryna-VALIHURA