

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ

(назва теми)

ПОШУКУ КОМП'ЮТЕРНОЇ ЛІТЕРАТУРИ. RESEARCH AND
SOFTWARE IMPLEMENTATION OF METHODS FOR SEARCHING OF
COMPUTER SCIENCE BOOKS

Виконав(ла): студент(ка) 2 курсу, групи КНТ-112м

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

КОТЛЯРОВ В.І.

(ПРІЗВИЩЕ та ініціали)

Керівник СКРУПСЬКИЙ С.Ю.

(ПРІЗВИЩЕ та ініціали)

Рецензент ГОЛУБ Т.В.

(ПРІЗВИЩЕ та ініціали)

2023_____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет _____ ФКНТ _____
Кафедра _____ програмних засобів _____
Ступінь вищої освіти _____ магістр _____
Спеціальність _____ 121 Інженерія програмного забезпечення _____
(код і найменування)
Освітня програма (спеціалізація) _____ Інженерія програмного забезпечення _____
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри _____ ПЗ, д.т.н, проф. _____
Сергій СУББОТІН
« _____ » _____ 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

КОТЛЯРОВА Владислава Ігоровича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів пошуку комп'ютерної літератури. Research and Software Implementation of Methods for Searching of Computer Science Books

керівник проєкту (роботи) к.т.н., доцент СКРУПСЬКИЙ Степан Юрійович _____,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від « 14 » листопада 2023 року № 437

2. Строк подання студентом проєкту (роботи) 01 грудня 2023 року
3. Вихідні дані до проєкту (роботи) _____ рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Розробка архітектури програми. 4. Реалізація програми. 5. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів, плакатів)
Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРІЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	СКРУПСЬКИЙ С.Ю., доцент		
Нормоконтроль	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання « 04 » вересня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Срок виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Дослідження проблеми	3-4 тижні	Розділ 2
4	Розробка архітектури програми	5 тиждень	Розділ 3
5	Розробка програми	6-7 тижні	Розділ 4
6	Тестування та експериментальне дослідження програми	8-9 тижні	Розділ 5
7	Оформлення пояснювальної записки та документів до неї	10 тиждень	Додатки
8	Нормоконтроль та рецензування	11 тиждень	
9	Захист роботи	12 тиждень	

Студент(ка)

_____ Владислав КОТЛЯРОВ
(підпис) (Ім'я ПРІЗВИЩЕ)

Керівник проєкту (роботи)

_____ Степан СКРУПСЬКИЙ
(підпис) (Ім'я ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
101 с., 26 рис., 4 табл., 3 дод., 12 джерел.

КОМП'ЮТЕРНА ЛІТЕРАТУРА, ПОШУК ДАНИХ,
КЛАСТЕРИЗАЦІЯ, PYTHON, DJANGO.

Об'єкт роботи – процес пошуку комп'ютерної літератури. Предмет роботи – методи пошуку комп'ютерної літератури. Мета роботи – розробити програмне забезпечення пошуку комп'ютерної літератури для знаходження релевантної літератури за проблемами, які цікавлять користувача.

Матеріали, методи та технічні засоби: мови Python, HTML, CSS, фреймворк Django, система для роботи з базами даних MySQL.

Результати. Програма призначена для пошуку комп'ютерної літератури, включаючи як книги з комп'ютерних наук загалом, так і окремі розділи цих книг, надаючи релевантну інформацію щодо тих проблем, які цікавлять користувача. Проблеми виражаються темами, підтемами та питаннями. Користувачі можуть виконувати оцінювання та аналіз книг і розділів стосовно зокрема підтем та питань, які в цих розділах розглядаються.

Висновки. Наукова новизна роботи полягає в запропонованому методі пошуку комп'ютерної літератури, який передбачає співвіднесення комп'ютерної літератури з проблемами, які в ній розглядаються, виділення літератури, якій за цією проблемою віддає перевагу користувач, розділення літератури за проблемою на кластери, визначення кластеру, до якого належить література, якій віддає перевагу користувач, та формування таким чином результатів пошуку.

Галузь використання. Платформа каталогізації комп'ютерної літератури, пошук комп'ютерної літератури для навчання, роботи.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 101 pages, 26 figures, 4 tables, 3 appendixes, 12 sources.

COMPUTER LITERATURE, DATA SEARCH, CLUSTERIZATION, PYTHON, DJANGO.

The object of the work is the process of searching computer literature. The subject of the work is methods for searching of computer science books. The purpose of the work is to develop software for searching of computer science books to find relevant literature on issues of interest to the user.

Materials, methods and technical means: Python, HTML, CSS languages, Django framework, MySQL system for working with databases.

The results. The program is designed to search computer literature, including both books on computer science in general and individual chapters of these books, providing relevant information on those problems of interest to the user. Problems are expressed by topics, subtopics and questions. Users can perform evaluations and analyzes of books and chapters with respect to particular subtopics and issues addressed in those chapters.

Conclusions. The scientific novelty of the work lies in the proposed method of searching for computer science books, which involves the correlation of computer literature with the problems that are considered in it, the selection of literature that is preferred by the user for this problem, the separation of literature by problem into clusters, the definition of a cluster, to which belongs the literature preferred by the user, and forming the search results in this way.

Field of use. Computer literature cataloging platform, search for computer literature for study, work.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	10
1.1 Програмні інструменти пошуку комп'ютерної літератури	10
1.2 Порівняння розроблюваної програми з існуючими	14
1.3 Висновки за розділом 1	15
2 Матеріали і методи.....	16
2.1 Пошук комп'ютерної літератури на основі використання кластеризації	16
2.2 Метод пошуку комп'ютерної літератури	18
2.3 Висновки за розділом 2	20
3 Розробка архітектури програми.....	22
3.1 Визначення сценаріїв використання програми користувачами	22
3.2 Рішення щодо вибору засобів розробки	25
3.3 Визначення структури даних у базі даних	26
3.4 Висновки за розділом 3	39
4 Реалізація програми	40
4.1 Визначення послідовності роботи з програмою	40
4.2 Особливості реалізації програми.....	41
4.3 Висновки за розділом 4	55
5 Експлуатація, тестування та експериментальне дослідження програми	56
5.1 Призначення програми	56
5.2 Виконання програми.....	56
5.3 Експериментальне дослідження	60
5.4 Висновки за розділом 5	63
Висновки	64
Перелік джерел посилання	65
Додаток А Технічне завдання	67

Додаток Б Текст програми	73
Додаток В Слайди презентації.....	94

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

ISBN – International Standard Book Number;

БД – база даних.

ВСТУП

Для пошуку літератури, створення і ознайомлення з відгуками, придбання літератури вже є готові програмні рішення. Однак, особливістю комп'ютерної літератури зокрема є те, що в ній можуть розглядатися різні теми, проблеми, а цікавити читача можуть не всі одразу, а тільки певні з них. При цьому традиційно пошук у всіх таких системах виконується для книги в цілому. Тобто якщо якесь питання є специфічним, невеликим, то знайти інформацію про нього стає достатньо складно. Якщо користувач знає, яка книга його цікавить, то подальші дії не є проблемою, а ось якщо потрібно виконувати такий пошук, то це проблема на даний момент.

Отримання інформації про різні складові комп'ютерної літератури, зокрема розділи книги (хоча такими складовими можуть бути не тільки розділи загалом, але і дрібніші складові – пункти, окремі питання), є важливим в розрізі комп'ютерної літератури. Наприклад, в художній літературі важливою є книга в цілому, з комп'ютерною літературою це не завжди так. Наприклад, при навчанні користувача можуть цікавити певні конкретні питання, і необхідно знайти релевантну літературу саме під таке питання.

З іншого боку потрібно розуміти і те, як певне питання розкрито в книзі, оцінювання зазвичай відбувається книги в цілому. Тобто щонайменше загальна оцінка стосується всієї книги, а в тексті вже можуть бути вказані певні деталі, проте акцент може бути і не зробленим або стосуватися різних питань. Тож на даний момент існує потреба в програмному забезпеченні, яке б дозволило виконувати пошук, отримання інформації про комп'ютерну літературу, надаючи можливості щодо окремих розділів книги, а не тільки книги загалом. Зважаючи на актуальність такої постановки, саме таке програмне забезпечення потрібно розробити в роботі.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Програмні інструменти пошуку комп'ютерної літератури

За результатами початкового аналізу було виділено два рішення для подальшого детального аналізу. Першим таким рішенням є Amazon (рис. 1.1) [1]. Це майданчик для продажу книг, при чому різних жанрів, в тому числі і комп'ютерної літератури. Проте на вебсайті представлено і достатньо багато даних про ці книги, окрім того інші рішення, які каталогізують книги, для придбання книг нерідко направляють саме на вебсайт Amazon, що є найбільшим подібним майданчиком.

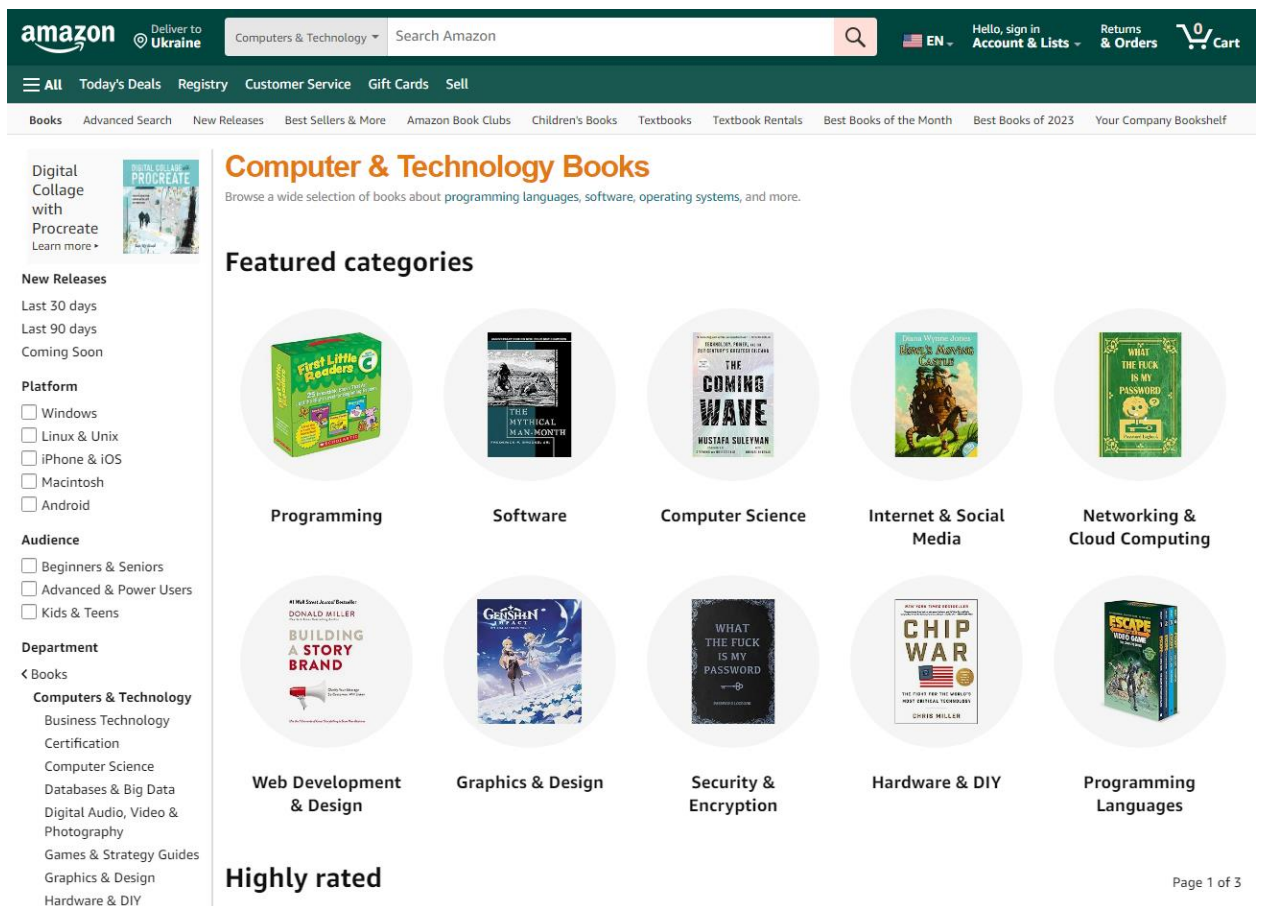


Рисунок 1.1 – Категорії комп'ютерної літератури на вебсайті Amazon

Серед всіх товарів на вебсайті Amazon можна вибрати книги, а серед жанрів книг вибрати комп'ютери і технології, що відповідає комп'ютерній літературі, а далі доступний перелік категорій, звідки можна обрати

програмування, вебпрограмування та дизайн, захист та шифрування, графіка та дизайн, мови програмування, комп'ютерні науки, програмне забезпечення тощо. Тож набір категорій достатньо великий і це дозволяє достатньо детально структурувати на найвищому рівні комп'ютерну літературу.

Пошук має велику кількість налаштувань, які при тому відносяться саме до літератури, тобто окрім назви можна обрати категорію, один з варіантів проміжку виконаних аналізів книги користувачами, серію видань, визначити період найновішої публікації, програмні платформи, які описуються в книзі (характеристика, що відноситься саме до комп'ютерної літератури), аудиторія книги (початківці, діти, професіонали), формат видання, мова видання, умови (нові, використані), доступність на момент продажу. Сторінка самої книги (рис. 1.2) містить багато інформації, що як раз і дозволяє розглядати вебсайт Amazon в контексті пошуку пошуку комп'ютерної літератури.

The screenshot displays the Amazon product page for the book "Automate the Boring Stuff with Python, 2nd Edition: Practical Programming for Total Beginners 2nd Edition, Kindle Edition" by Al Sweigart. The page includes the Amazon logo, navigation links, and a search bar. The book cover is shown on the left, featuring a robot character. The main content area on the right provides details about the book, including its title, author, ratings (4.7 stars), and a description. It also lists various purchase options: Kindle Edition (\$23.99), Paperback (\$15.87), and Spiral-bound (\$30.99). A "Buy now with 1-Click" button is prominently displayed. The page also features a "Follow the author" section with a profile picture of Al Sweigart and a "Follow" button. At the bottom, there is a section for "Buy for others" with a "Buy for others" button.

Рисунок 1.2 – Сторінка комп'ютерної літератури на вебсайті Amazon

Стосовно книги надається інформація про автора, обкладинка, фрагмент з книги, якщо він доступний, інформація про видавця, відгуки від редакторів, відгуки звичайних користувачів, мова видання, кількість сторінок, вік читачів, на який книга розрахована, вага, розміри відео з книгою.

Другим рішенням, яке далі детально проаналізовано, є Goodreads (рис. 1.3) [2].

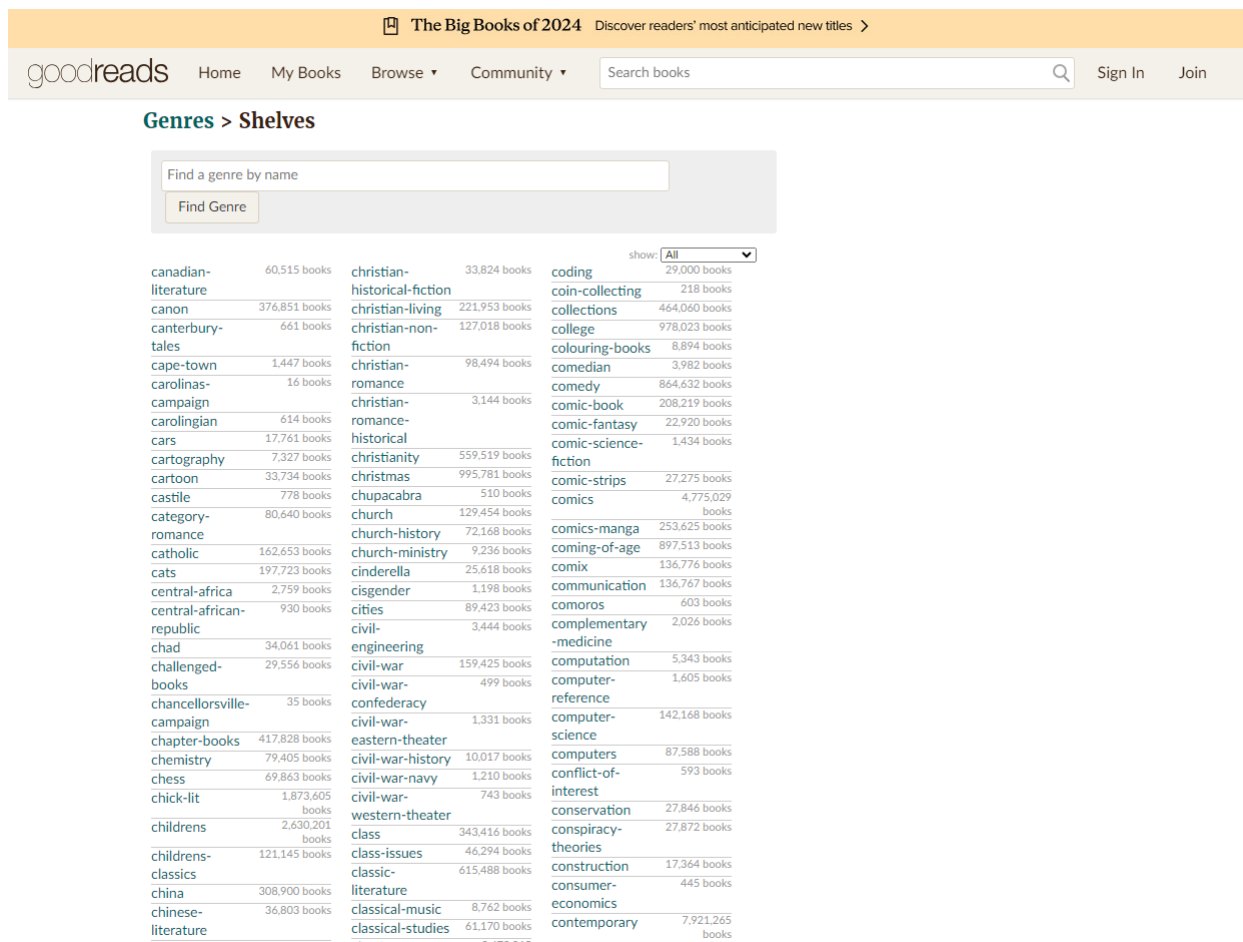


Рисунок 1.3 – Повний перелік жанрів на вебсайті Goodreads

Goodreads має каталогізацію книг як основну направленість. Він також не має окремої спеціалізації, але серед книг дозволяє отримувати інформацію і про комп'ютерну літературу. Однак загалом Goodreads дозволяє обрати потрібний жанр, декілька разів послідовно його розширюючи, а підсумковий перелік жанрів величезний і займає 14 сторінок.

Сама сторінка книги (рис. 1.4) містить також багато базової інформації, включаючи назву, автора, опис, інформацію про саме видання (більш обмежену ніж Amazon).

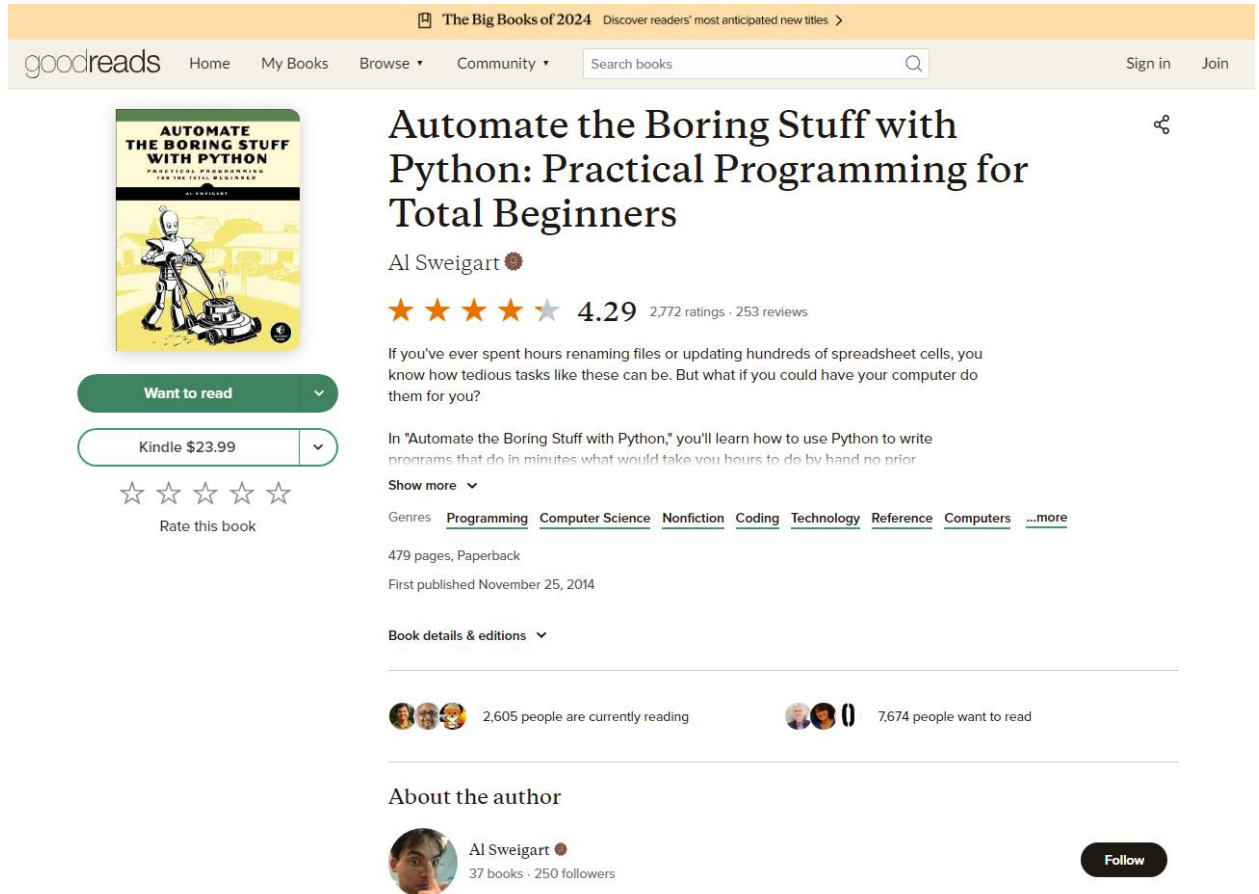


Рисунок 1.4 – Сторінка комп'ютерної літератури на вебсайті Goodreads

Проте на сторінці книги зроблений великий акцент на враженнях від користувачів. Передбачається формування окремої оцінки за результатами оцінювань, коли виставляється тільки оцінка користувачів, та за результатами рейтингування (рис. 1.5). Рейтингування при цьому несе розширений сенс, адже окрім оцінки передбачає ще й детальний опис книги, аналіз. У роботі вирішено використовувати для оцінювання книг подібне рішення, тобто також виділяти оцінки, а другий варіант названо аналізом. Тобто користувач може аналізувати книгу і наводити факти щодо того, як розкрито в ній певні питання.

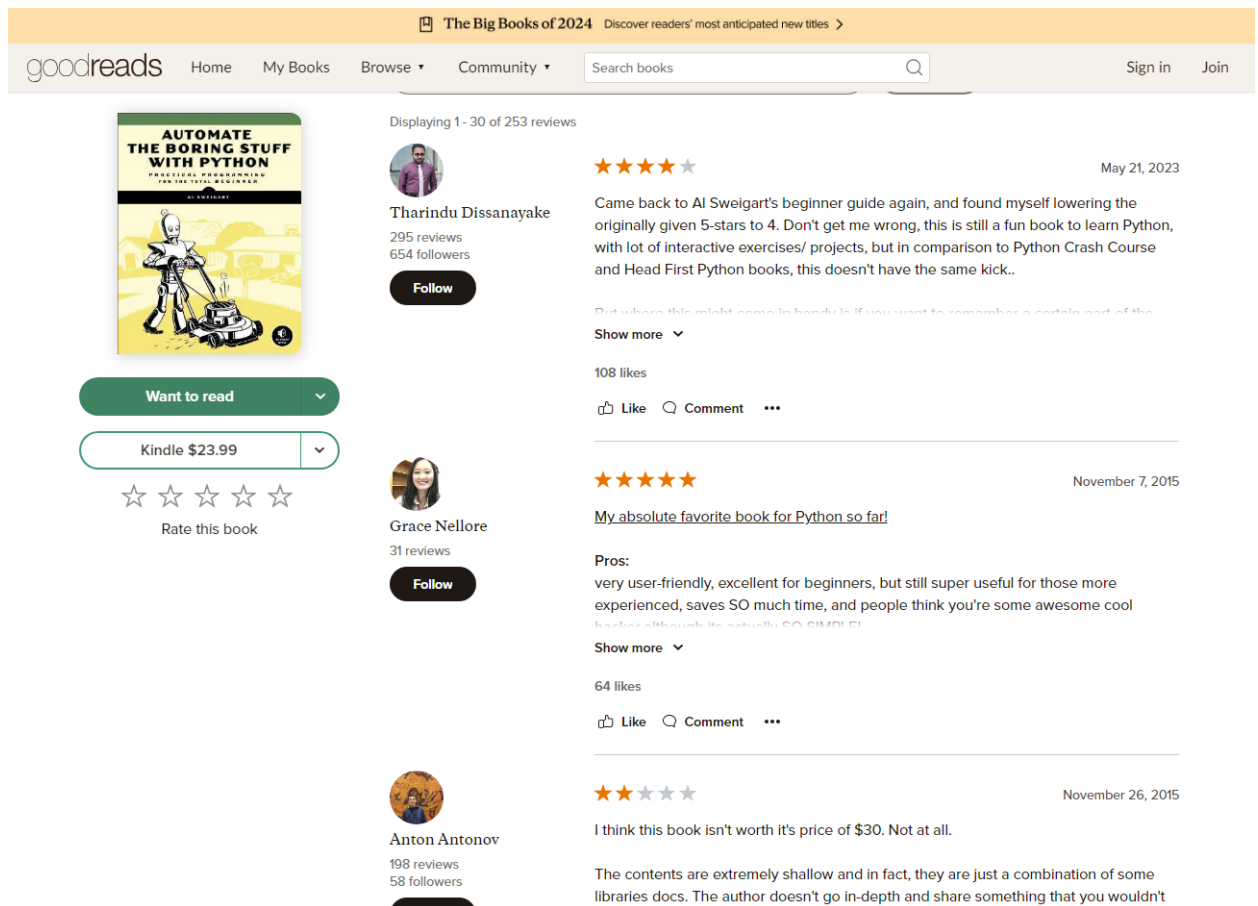


Рисунок 1.5 – Сторінка з виконаним аналізом користувачами комп'ютерної літератури на вебсайті Goodreads

1.2 Порівняння розроблюваної програми з існуючими

Виконуючи порівняння Amazon і Goodreads, було також порівняно їх з тією концепцією, яку вирішено закласти при створенні програми CompBooker, яка розробляється в роботі (табл. 1.1).

Основною відмінністю CompBooker має бути можливість пошуку за розділами книг, а також аналізування не тільки книг загалом, але і розділів. Така концепція дозволить виконувати відповідно оцінювання, аналізування за тими проблемами з комп'ютерних наук, які можуть цікавити користувача. Комп'ютерні науки в даному випадку розглядаються в широкому сенсі, включаючи і програмування, і науки про дані, і інші подібні сфери. Враховуючи можливість надання інформації про те, як певні проблеми

розкрито в розділі книги, можливим стає підбір книг для користувача саме за вказаною проблемою.

Таблиця 1.1 – Характеристика розроблюваної та існуючих програм

Показник	Amazon	Goodreads	CompBooker
Пошук книг з комп'ютерних наук	+	+	+
Пошук за розділами книг з комп'ютерних наук	—	—	+
Аналізування книг користувачами	—	+	+
Аналізування розділів книг стосовно проблем	—	—	+
Об'єкт підписування	Автори	Автори, користувачі	Автори, проблеми
Підбір книг для користувача	За книгою	За книгою, за аналізуваннями	За проблемою

1.3 Висновки за розділом 1

У підсумку було визначено завдання роботи:

- аналіз існуючих програм, які виконують пошук комп'ютерної літератури, що було забезпечено в даному розділі;
- дослідження методів і моделей пошуку комп'ютерної літератури;
- проєктування програми пошуку комп'ютерної літератури;
- реалізація програми пошуку комп'ютерної літератури.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Пошук комп'ютерної літератури на основі використання кластеризації

Пошук комп'ютерної літератури є важливою практичною задачею, адже від її розв'язання залежить те, чи отримає користувач релевантне джерело для навчання, розв'язання проблеми, яка перед ним виникла. Якщо пошук буде успішним і таке джерело буде знайдено, то проблеми будуть вирішені. Тож логічним є визначення для кожного джерела (книги, розділу книги) тієї проблеми, яка в ньому розглядається, а далі врахування цієї проблеми під час пошуку. Це дещо інший підхід порівняно з традиційним, коли пошук відбувається тільки за назвою, за характеристиками, що відносяться до книги (видавництво, рік видання тощо). Тоді можливим стає накопичення даних про те, які проблеми в книзі та окремих розділах книг розглядаються, а далі визначення того, на скільки якісно це зроблено. Для визначення якості можна долучити користувачів, які або шляхом встановлення тільки оцінок, або з визначенням певного аналізу (текстового опису та оцінки) в результаті можуть допомогти накопичити дані про те, на скільки ця книга відповідає конкретній проблемі.

Усі проблеми було вирішено розділити на декілька рівнів:

- теми – найвищий рівень, який відповідає всій книзі з комп'ютерної тематики (комп'ютерних наук);
- підтеми – рівень, що відповідає або може відповідати конкретному розділу книги;
- питання – конкретний метод, інструмент, ситуація, що розглядаються в розділі книги.

Тож користувач повинен визначити проблему, яка його цікавить, а задача пошуку комп'ютерної літератури тоді полягає в тому, щоб на основі визначеної користувачем проблеми та наявних оцінок і виконаних користувачем раніше аналізів за цією проблемою виконати підбір тієї

комп'ютерної літератури, що буде найбільше відповідати його зацікавленостям. Результатом розв'язання задачі пошуку комп'ютерної літератури має бути підмножина всієї комп'ютерної літератури, включаючи розділи книг (якщо пошук відповідає підтемі або питанню) або книги (якщо пошук відповідає темі), яка є релевантною параметрам пошуку та відповідає тому, яку літературу раніше вважав релевантною для себе користувач.

Така постановка задачі не направлена на початкове дослідження питання, яке виникає перед користувачем, але вона направлена на те, щоб в подальшому поглиблювати вивчення теми, проблеми на основі літератури, яку не розглядав раніше користувач. Тоді, коли література оновлюється, з'являються нові джерела, користувач може зрозуміти, які з них є такими, що можуть бути рекомендовані йому за певною проблемою.

Зважаючи на природу описаної задачі пошуку комп'ютерної літератури, одним з варіантів розв'язання задачі, який зокрема потребує найменшої участі користувача, а саме такий варіант слід вважати доцільним, є виконання кластеризації комп'ютерної літератури. Тоді якщо виконано кластеризацію, літературу розділено на певні кластери, то можна знайти той, якому віддає перевагу користувач. Особливістю кластеризації є те, що це є навчанням без учителя, тобто не потрібно розмічати дані, а значить не потрібно користувачу визначати цілий ряд книг, які його цікавлять чи ні. Враховуючи описані особливості задачі пошуку комп'ютерної літератури в тому вигляді, якому вона розглядається в роботі, достатньо виконати тільки декілька простих оцінок або деталізованих через аналіз оцінок, і тоді за ними можна дізнатися переваги користувача.

Одним з варіантів розв'язання задачі кластеризації є використання методу k-середніх. Як результат роботи методу, створені в результаті нові кластери допомагають користувачам зрозуміти закономірності, поведінку та іншу інформацію з даних завдяки меншій кількості точок даних і меншим відхиленням [3]-[4].

Загалом метод k-середніх передбачає виконання наступних етапів:

- визначення кількості кластерів, що є фактично вхідними даними для роботи методу, після структурування набору даних;
- виконання ініціалізації центроїдів: одним з варіантів розв’язання цієї проблеми може бути використання випадкової ініціалізації;
- вимірювання відстані усіх точок даних до кожного центроїда та призначення кластерів найближчих центроїдів;
- обчислення нових центроїдів;
- повторення тих самих кроків, доки не буде забезпечена збіжність і остаточні кластери не сформовані [3]-[5].

Метод спектральної кластеризації може допомогти зі складними (неопуклими) структурами та великими наборами даних, він може працювати, щоб розділити дані на частини, які складаються разом, як пазли, тобто форма кластерів є більш складною у підсумку [6].

Тож у підсумку задача пошуку комп’ютерної літератури полягає в тому, щоб визначити підмножину комп’ютерної літератури, яка відповідає потребам користувача, знаючи при цьому визначену проблему, яка цікавить користувача, літературу, якій раніше надавав перевагу користувач, що є вхідними даними для цієї задачі.

2.2 Метод пошуку комп’ютерної літератури

Метод пошуку комп’ютерної літератури використовує ту логіку, яка була закладена у відповідну задачу і описана. Тому для забезпечення роботи потрібно спочатку визначити вхідні дані для подальшого оброблення. Однак вхідні дані в даному випадку мають бути певним чином перетворені. Тобто дані потребують певного оброблення для того, щоб можна було виконати далі кластеризацію.

Першим етапом роботи методу пошуку комп’ютерної літератури є виділення всіх проблем, тобто тем, підтем, питань, які розглядаються.

Другим етапом є структуризація та оброблення даних. До цих даних мають бути віднесені дані тільки про ту комп'ютерну літературу, яка має надаватися користувачам у результаті. Якщо якась література не відповідає певним встановленим правилам, то вона має бути відфільтрована з цих даних. Далі дані мають бути структуровані за проблемами. Тобто для кожної проблеми має бути виділений набір даних, кожен екземпляр якого включає наступні властивості:

- основний автор джерела (книги або розділу книги);
- кількість оцінок, які були виставлені джерелу через оцінювання (без аналізу);
- кількість оцінок, які були виставлені джерелу через аналіз;
- середня оцінка джерела;
- рік публікації джерела;
- видавець джерела;
- входження книги до серії видань або відсутність такого входження.

Для всіх значень має бути виконано числову трансформацію:

- видавець джерела має бути переведений у відповідний код, представлений числом, для чого має бути сформований з усіх переданих даних перелік усіх видавців;
- автор джерела має бути представлений числом, для чого також має бути сформований перелік всіх авторів, якщо він початково не заданий, а кожному автору представлений код;
- якщо початково задано, в яку саме серію видань входить книга, то для всіх випадків, коли книга є частиною серії має бути внесено 1, а для тих, у яких серію не вказано, – 0.

Таким чином, в результаті виконання даного етапу має бути сформовано набір даних, представлених числами за кожною ознакою, для кожної проблеми, що може розглядатися.

Наступний етап полягає у виконанні безпосередньо кластеризації даних, сформованих на початкових етапах. Для цього запропоновано

використовувати метод спектральної кластеризації з встановленням потрібної кількості кластерів. При встановленні значення k потрібно враховувати кількість комп'ютерної літератури за проблемою, тому необхідно виконувати кластеризацію для кожної проблеми окремо. В результаті для кожної проблеми має бути визначено центри кластерів та розмічено наявні дані стосовно належності до кластерів.

На наступному етапі користувач визначає проблему, яка його цікавить, та визначається перелік джерел за цією проблемою, які оцінив або за якими провів аналіз користувач. З усіх таких оцінок відбираються тільки ті джерела, які стосуються тієї самої проблеми, яку визначив користувач. З усіх відібраних оцінок залишаються тільки ті, які перевищують встановлене значення. Саме такі джерела формують перелік на виході даного етапу.

На останньому етапі виконується обчислення відстані від центру кожного кластера до всіх джерел з переліку, які були оцінені або проаналізовані користувачем та відповідним чином попередньо відібрані. Кластер, до якого відстань виявляється найменшою, визначається як результат виконання даного етапу. Уся комп'ютерна література, що належить до даного кластеру, надається користувачу як результат роботи методу. Якщо користувачем встановлено додаткові параметри, то дані з виділених таким чином даних мають бути відкинуті ті книги або розділи книг, які цим значенням параметрів не відповідають.

Останні два етапи проводяться кожен раз за запитом користувача. Попередні ж до них етапи мають виконуватися з певним кроком тоді, коли з'являється нова література. Тоді створюються нові кластери, а робота останніх 2 етапів підтримується вже на їх основі.

2.3 Висновки за розділом 2

Розділ 2 в якості основного отриманого результату представляє запропонований метод пошуку комп'ютерної літератури, яким і визначається

наукова новизна роботи. Даний метод передбачає співвіднесення комп'ютерної літератури з проблемами, які в ній розглядаються, виділення літератури, якій за цією проблемою віддає перевагу користувач, розділення літератури за проблемою на кластери, визначення кластеру, до якого належить література, якій віддає перевагу користувач, та формування таким чином результатів пошуку.

3 РОЗРОБКА АРХІТЕКТУРИ ПРОГРАМИ

3.1 Визначення сценаріїв використання програми користувачами

Стандартний користувач, який може бути авторизованим або неавторизованим, але для виконання певних функцій потребує авторизації, може використовувати програму на основі наступних сценаріїв:

- пошук комп'ютерної літератури;
- авторизація;
- отримання інформації про книгу з комп'ютерних наук;
- отримання інформації про розділ книги з комп'ютерних наук;
- демонстрування переліку книг за видавництвом;
- демонстрування переліку книг за темою;
- демонстрування переліку книг за підтемою;
- демонстрування переліку книг за питанням;
- оцінювання книги з комп'ютерних наук;
- оцінювання викладення підтеми в розділі книги з комп'ютерних наук;
- оцінювання викладення питання в розділі книги з комп'ютерних наук;
- демонстрування історії оцінювань за розділом книги з комп'ютерних наук;
- демонстрування історії оцінювань за книгою з комп'ютерних наук;
- демонстрування історії власних оцінювань;
- публікування аналізу книги з комп'ютерних наук;
- публікування аналізу підтеми в розділі книги з комп'ютерних наук;
- публікування аналізу питання в розділі книги з комп'ютерних наук;
- редагування опублікованого аналізу комп'ютерної літератури;
- демонстрування історії аналізів за розділом книги з комп'ютерних наук;
- демонстрування історії аналізів за книгою з комп'ютерних наук;

- демонстрування історії власних аналізів комп'ютерної літератури;
- висловлення підтримки аналізу комп'ютерної літератури;
- коментування аналізу комп'ютерної літератури;
- демонстрування власних дискусій користувача;
- отримання інформації про автора;
- редагування інформації про автора;
- пошук авторів;
- підписування на книги автора;
- підписування на книги за темою;
- підписування на книги за підтемою;
- підписування на книги за питанням;
- скасування підписування на книги автора;
- скасування підписування на книги за темою;
- скасування підписування на книги за підтемою;
- скасування підписування на книги за питанням;
- демонстрування переліку книг за підпискою;
- отримання інформації про видавця;
- редагування інформації про видавця;
- пошук видавців;
- демонстрування комп'ютерної літератури видавця;
- демонстрування серій книг видавця;
- демонстрування переліку книг в серії;
- створення переліку комп'ютерної літератури;
- додавання книги до переліку комп'ютерної літератури;
- вилучення книги з переліку комп'ютерної літератури;
- додавання підрозділу книги до переліку комп'ютерної літератури;
- вилучення підрозділу книги з переліку комп'ютерної літератури;
- демонстрування власних переліків комп'ютерної літератури користувача;
- демонстрування комп'ютерної літератури з переліку;

– редагування облікових даних (рис. 3.1).

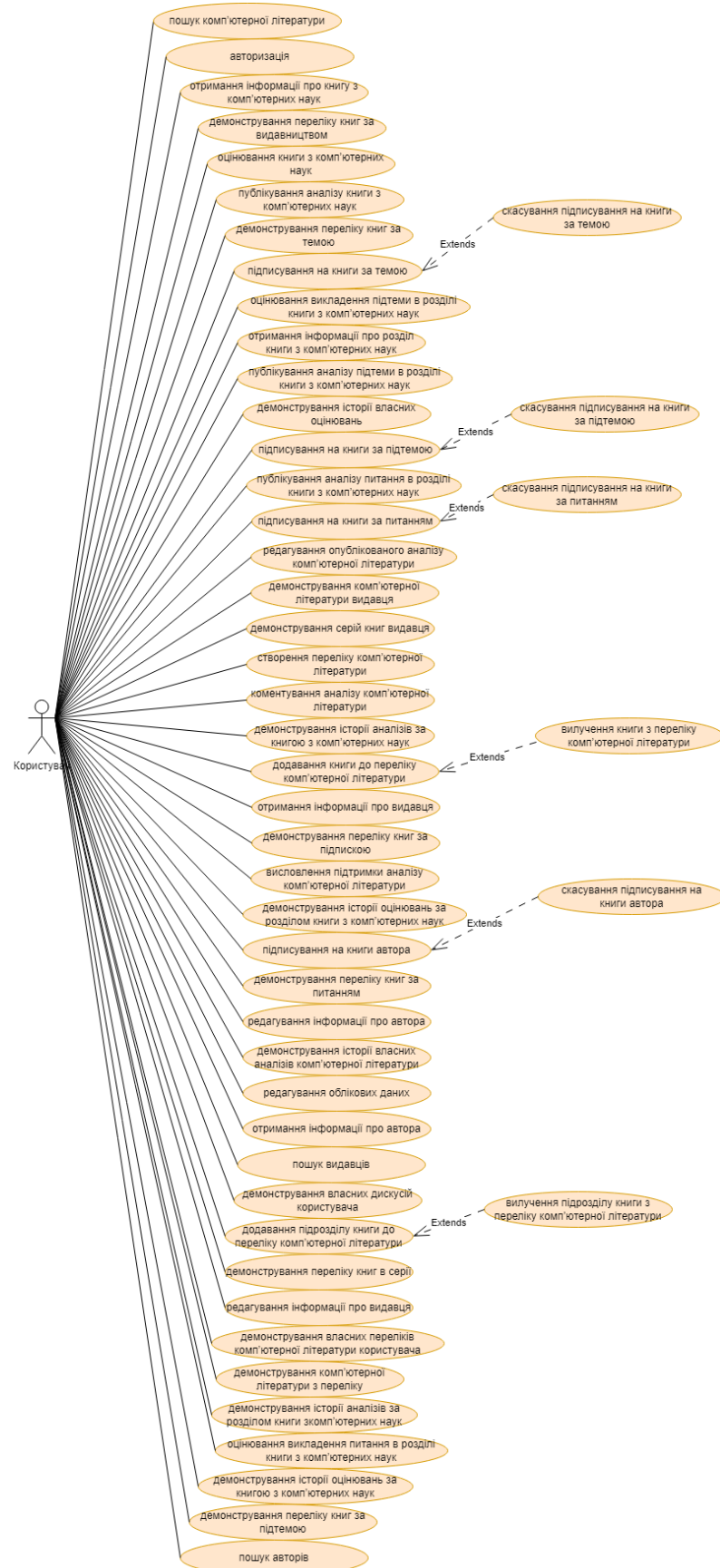


Рисунок 3.1 – Діаграма прецедентів користувача при роботі з програмою

Адміністратор повинен додатково до цих сценаріїв мати можливість внесення тих даних, які використовують всі користувачі, тому для цього необхідні сценарії:

- додавання книги з комп'ютерних наук;
- додавання розділу книги з комп'ютерних наук;
- редагування даних книги з комп'ютерних наук;
- редагування розділу книги з комп'ютерних наук;
- визначення інформації про автора;
- визначення інформації про видавця (рис. 3.2).

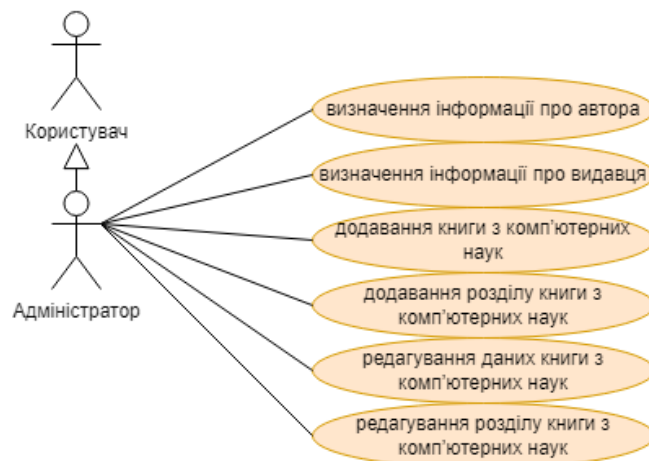


Рисунок 3.2 – Діаграма прецедентів адміністратора при роботі з програмою

3.2 Рішення щодо вибору засобів розробки

Мова Python обрана для розробки, оскільки має цілий ряд засобів для підтримки роботи з даними, для забезпечення машинного навчання, в тому числі кластеризації, надаючи для цього пакети pandas, numpy для роботи з даними, scikit-learn для реалізації кластеризації, виконання статичних обчислень і інших типів дій. Саму веброзробку вирішено виконувати за допомогою фреймворку, серед яких розглянуто Pyramid [7] і Django [8]-[9] (табл. 3.1), а обрано останній з них, оскільки Django має засоби керування для адміністратора, що одразу дозволяє змінювати потрібні дані, вносити їх і

почати роботу з програмою, має вбудований шаблонізатор, а окрім того і загалом вищий рівень підтримки, тобто більшу кількість статей, книг, розв'язаних проблем.

Таблиця 3.1 – Порівняння засобів для розробки програми пошуку комп'ютерної літератури

Критерій	Pyramid	Django
Засоби керування для адміністратора	Ні, але можливе розширення, потребує пошуку	Так
Рівень підтримки	Нижчий	Високий
Шаблонізатор сторінок	Потребує вибору і встановлення	Так
Підтримка авторизації	Так	Так

3.3 Визначення структури даних у базі даних

Зважаючи на концепцію програми, яка і повинна визначити її відмінності від існуючих, комп'ютерна література має бути представлена одразу двома базовими складовими:

- самою книгою, яка зазвичай і присутня в програмах;
- розділом, який може мати певний інтерес для користувачів стосовно питань з комп'ютерних наук, які в ньому розглянуті: розділом може бути або безпосередньо глава, або певна частина такої глави, якщо при розгляді комп'ютерної літератури буде виявлено, що цей блок книги варто вважати окремим блоком і він має окрему цінність стосовно розглянутих у ньому питань.

Основний контейнер комп'ютерної літератури (книги), що і містить загальний опис книги, представлений у створеній базі даних (БД) таблицею Computerbook з полями:

- id – ідентифікатор комп'ютерної літератури (книги з комп'ютерних наук);
- firsttitle – основна назва комп'ютерної літератури (книги з комп'ютерних наук);
- secondtitle – додаткова назва комп'ютерної літератури (або назва іншою мовою);
- slug – текстовий ідентифікатор комп'ютерної книги (книги з комп'ютерних наук);
- publisherseries – серія, до якої належить ця комп'ютерна книга (книга з комп'ютерних наук), що може бути не задана, якщо серії не існує, але якщо задана, то визначається зовнішнім ключем до таблиці Publisherseries;
- bookcover – фотографія титульної сторінки комп'ютерної книги (книги з комп'ютерних наук);
- examplepdf – витяг з комп'ютерної книги (книги з комп'ютерних наук), представлений pdf-файлом;
- analysisrating – середній рейтинг, виставлений в проведених користувачами аналізах цієї комп'ютерної книги;
- scorering – середній рейтинг, виставлений у оцінках, наданих комп'ютерній книзі користувачами (без аналізу, тільки оцінка);
- pagescount – кількість сторінок комп'ютерної книги (книги з комп'ютерних наук);
- publishdate – дата публікації комп'ютерної книги (книги з комп'ютерних наук);
- publisher – видавець комп'ютерної книги (книги з комп'ютерних наук), визначається зовнішнім ключем до таблиці Publisher;
- isbn – міжнародний номер комп'ютерної книги (книги з комп'ютерних наук), відомий як International Standard Book Number (ISBN): не може використовуватися як первинний ключ таблиці тому, що не всі книги можуть мати цей номер;

- language – мова, якою написана комп'ютерна книга (книга з комп'ютерних наук);
- original – комп'ютерна книга (книга з комп'ютерних наук), яка є оригіналом даної (може бути не вказане значення, якщо книга сама є оригіналом), визначається зовнішнім ключем до таблиці Computerbook;
- description – опис комп'ютерної книги (книги з комп'ютерних наук);
- sourcetype – тип основної публікації комп'ютерної книги (електронне видання, фізичне надруковане).

Розділ комп'ютерної книги представлений таблицею Computerbookpart з полями:

- id – ідентифікатор розділу комп'ютерної книги (книги з комп'ютерних наук);
- slug – текстовий ідентифікатор розділу комп'ютерної книги (книги з комп'ютерних наук);
- computerbook – комп'ютерна книга (книга з комп'ютерних наук), в якій є цей розділ, визначається зовнішнім ключем до таблиці Computerbook;
- partnumber – номер розділу за порядком, у якому він приведений у цій комп'ютерній книзі (книзі з комп'ютерних наук);
- title – назва комп'ютерної книги (книги з комп'ютерних наук);
- description – повний опис комп'ютерної книги (книги з комп'ютерних наук);
- examplepdf – витяг з розділу комп'ютерної книги (книги з комп'ютерних наук), представлений pdf-файлом;
- partcontent – зміст даного розділу комп'ютерної книги (книги з комп'ютерних наук), тобто формальне представлення того, як цей розділ структуровано саме в самій книзі;
- startpage – сторінка в самій комп'ютерній книзі (книзі з комп'ютерних наук), з якою починається цей розділ;
- endpage – сторінка в самій комп'ютерній книзі (книзі з комп'ютерних наук), якою закінчується цей розділ.

Усі книги з комп'ютерних наук розділені на певні теми. Це загальні теми сфери комп'ютерних наук. Наприклад, це може бути програмування, наука про дані тощо. Однак, для пошуку важливим є звичайно не такий високий рівень, а нажчий. Тому окрім тем було введено підтеми, а також окремі питання. Окреме питання – це може бути конкретний інструмент або, наприклад, метод, який може шукати користувач. Наприклад, метод кластеризації k-середніх – це окреме питання. У той же час темою може бути наука про дані, а підтемою – кластеризація.

Теми з комп'ютерних наук представлені таблицею Topic з полями:

- id – ідентифікатор теми з комп'ютерних наук;
- title – назва самої теми з комп'ютерних наук.

Відповідно підтеми з комп'ютерних наук представлені таблицею Subtopic з полями:

- id – ідентифікатор підтеми з комп'ютерних наук;
- topic – тема з комп'ютерних наук, до якої відноситься ця підтема, визначається зовнішнім ключем до таблиці Topic;
- title – назва самої підтеми з комп'ютерних наук.

Питання з комп'ютерних наук представлені таблицею Question з полями:

- id – ідентифікатор питання з комп'ютерних наук;
- topic – тема з комп'ютерних наук, до якої відноситься це питання, визначається зовнішнім ключем до таблиці Topic;
- title – назва самого питання з комп'ютерних наук.

Визначення того, які саме теми в яких саме комп'ютерних книгах представлені, відбувається в таблиці БД Computerbooktopic з полями:

- id – ідентифікатор співвідношення книги з комп'ютерних наук з темою;
- computerbook – книга з комп'ютерних наук, в якій ця тема розглядається, визначається зовнішнім ключем до таблиці Computerbook;

– topic – тема з комп'ютерних наук, яка розглядається в цій книзі, визначається зовнішнім ключем до таблиці Topic.

Теми співвідносяться з цілими книгами, а підтеми стосуються вже конкретних розділів, де вони розглядаються, тому співвідношення між розділами комп'ютерної книги та підтемами встановлено в таблиці БД Computerbookpartsubtopic з полями:

– id – ідентифікатор співвідношення книги з комп'ютерних наук з темою;

– computerbookpart – розділ книги з комп'ютерних наук, в якому ця підтема розглядається, визначається зовнішнім ключем до таблиці Computerbookpart;

– subtopic – підтема з комп'ютерних наук, яка розглядається в цьому розділі книги, визначається зовнішнім ключем до таблиці Subtopic;

– analysisrating – середній рейтинг, виставлений в проведених користувачами аналізах цього розділу комп'ютерної книги;

– scorering – середній рейтинг, виставлений у оцінках, наданих розділу комп'ютерної книги користувачами (без аналізу, тільки оцінка);

– description – опис стосовно того, як саме ця підтема розглядається в цьому розділі книги.

Також саме розділу комп'ютерної книги стосуються питання, тобто має бути зазначено не те, що загалом в книзі розглядається певне питання, а те, в якому саме розділі це виконано. Якщо питання розглядається у всій книзі, то це або можливо тема, а не питання, або вся книга має розглядатися цілісно, тобто представлятися як один розділ.

Питання співвідносяться з розділом комп'ютерної книги і знаходяться в таблиці Computerbookpartquestion з полями:

– id – ідентифікатор співвідношення розділу книги з комп'ютерних наук з питанням;

– computerbookpart – розділ книги з комп’ютерних наук, в якому це питання розглядається, визначається зовнішнім ключем до таблиці Computerbookpart;

– question – питання з комп’ютерних наук, яке розглядається в цьому розділі книги, визначається зовнішнім ключем до таблиці Question;

– analysirating – середній рейтинг, виставлений в проведених користувачами аналізах цього питання, представленого в розділі комп’ютерної книги;

– scorerating – середній рейтинг, виставлений у оцінках, наданих цьому питанню, представленому в розділі комп’ютерної книги (без аналізу, тільки оцінка);

– description – опис стосовно того, як саме це питання розглядається в цьому підрозділі книги.

Зважаючи на описану структуру, що фактично інформація в програмі стосується книг та розділів у розрізі підтем і питань, то саме оцінювання користувачами винесено в таблицю Searcherscore, щоби вже далі визначити в окремих таблицях, чого саме це оцінювання стосується (книги, підтеми в розділі книги, питання в розділі книги). Таблиця Searcherscore має поля:

– id – ідентифікатор оцінювання користувачем певного контенту (книги, підтеми в розділі книги або питання в розділі книги);

– searcher – користувач, який це оцінювання контенту виконав, визначається зовнішнім ключем до таблиці Searcher;

– scoretime – час, коли оцінювання користувачем контенту було виконано;

– score – оцінка, яку користувач виставив контенту.

Відповідно далі визначається зв’язок цього оцінювання з конкретним типом контенту. У таблиці Computerbooksearcherscore представлені оцінювання цілих комп’ютерних книг, має поля:

– id – ідентифікатор співвідношення оцінювання з комп’ютерною книгою;

– computerbook – комп’ютерна книга, яку було оцінено, визначається зовнішнім ключем до таблиці Computerbook;

– searcherscore – оцінювання контенту, в якому представлена оцінка книзі, визначається зовнішнім ключем до таблиці Searcherscore.

У таблиці Computerbookpartsubtopicsearcherscore представлені оцінювання того, як у конкретному розділі книги розглянуто певну підтему, має поля:

– id – ідентифікатор співвідношення оцінювання з розглядом підтеми у розділі комп’ютерної книги;

– computerbookpartsubtopic – розгляд підтеми у розділі комп’ютерної книги, що було оцінено, визначається зовнішнім ключем до таблиці Computerbookpartsubtopic;

– searcherscore – оцінювання контенту, в якому представлена оцінка розгляду підтеми у розділі комп’ютерної книги, визначається зовнішнім ключем до таблиці Searcherscore.

У таблиці Computerbookpartquestionsearcherscore представлені оцінювання того, як у конкретному розділі книги розглянуто певне питання, має поля:

– id – ідентифікатор співвідношення оцінювання з розглядом питання у розділі комп’ютерної книги;

– computerbookpartquestion – розгляд питання у розділі комп’ютерної книги, що було оцінено, визначається зовнішнім ключем до таблиці Computerbookpartquestion;

– searcherscore – оцінювання контенту, в якому представлена оцінка розгляду питання у розділі комп’ютерної книги, визначається зовнішнім ключем до таблиці Searcherscore.

Окрім просто виставлення оцінки користувач може ще й виконати повноцінний аналіз, представивши деталі щодо того, на скільки актуальними, важливими і які саме питання взагалі розглядаються в книзі. Для цього створена таблиця Searcheranalysis, яка має поля:

- id – ідентифікатор виконаного аналізу користувачем певного контенту (книги, підтеми в розділі книги або питання в розділі книги);
- searcher – користувач, який цей аналіз контенту виконав, визначається зовнішнім ключем до таблиці Searcher;
- basereview – текст самого проведеного аналізу;
- analysistime – час, коли аналіз користувачем контенту було виконано;
- score – оцінка, яку користувач виставив контенту в результаті аналізу.

Так само, як це було для оцінювання, аналіз також має бути пов'язаний з конкретним контентом (книгою, підтемою в розділі книги, питанням в розділі книги). За кожен такий зв'язок відповідає окрема таблиця в БД. У таблиці Computerbooksearcheranalysis представлено зв'язування виконаного аналізу з комп'ютерною книгою в якості контенту на основі полів:

- id – ідентифікатор співвідношення виконаного аналізу з комп'ютерною книгою;
- computerbook – комп'ютерна книга, яку було проаналізовано, визначається зовнішнім ключем до таблиці Computerbook;
- searcheranalysis – виконаний аналіз контенту, в якому представлена ця книга, визначається зовнішнім ключем до таблиці Searcheranalysis.

У таблиці Computerbookpartsubtopicsearcheranalysis представлені співвідношення виконаного аналізу з розглядом підтеми у розділі комп'ютерної книги, має поля:

- id – ідентифікатор співвідношення виконаного аналізу з розглядом підтеми у розділі комп'ютерної книги;
- computerbookpartsubtopic – розгляд підтеми у розділі комп'ютерної книги, для якого було виконано аналіз, визначається зовнішнім ключем до таблиці Computerbookpartsubtopic;

– `searcheranalysis` – виконаний аналіз контенту, в якому представлена оцінка розгляду підтеми у розділі комп'ютерної книги, визначається зовнішнім ключем до таблиці `Searcheranalysis`.

У таблиці `Computerbookpartquestionsearcheranalysis` представлені співвідношення виконаного аналізу з розглядом у конкретному розділі книги певного питання, має поля:

– `id` – ідентифікатор співвідношення оцінювання з розглядом питання у розділі комп'ютерної книги;

– `computerbookpartquestion` – розгляд питання у розділі комп'ютерної книги, стосовно якого виконано аналіз, визначається зовнішнім ключем до таблиці `Computerbookpartquestion`;

– `searcheranalysis` – виконаний аналіз контенту, в якому представлена оцінка розгляду питання у розділі комп'ютерної книги, визначається зовнішнім ключем до таблиці `Searcheranalysis`.

Користувачі можуть підтримати проведений аналіз, висловивши свою позитивну думку (фактично проголосувавши в підтримку результатів аналізу), дані щодо чого знаходяться в таблиці `Searcheranalysisupport` з полями:

– `id` – ідентифікатор підтримки, висловленої користувачем результатам аналізу певного контенту;

– `searcheranalysis` – виконаний аналіз контенту, підтримку якому висловив користувач, визначається зовнішнім ключем до таблиці `Searcheranalysis`;

– `searcher` – користувач, який висловив підтримку аналізу контенту, визначається зовнішнім ключем до таблиці `Searcher`.

Окрім підтримки користувачі можуть також коментувати результати аналізу, висловлюючи конкретні думки при цьому. Такі дані знаходяться в таблиці БД `Searcheranalysiscomment` з полями:

– `id` – ідентифікатор коментаря до проведеного аналізу контенту;

– `searcheranalysis` – виконаний аналіз контенту, до якого відноситься коментар, визначається зовнішнім ключем до таблиці `Searcheranalysis`;

– `searcher` – користувач, який створив коментар до аналізу контенту, визначається зовнішнім ключем до таблиці `Searcher`;

– `commenttext` – текст коментаря;

– `commenttime` – час створення коментаря.

Автори книжок мають базову інформацію про себе, представлену в програмі, а зберігається ця інформація в таблиці `Author` з полями:

– `id` – ідентифікатор автора;

– `surname` – прізвище автора;

– `name` – ім'я автора;

– `about` – інформація про автора;

– `photo` – фотографія автора;

– `borndate` – дата народження автора;

– `website` – вебсайт автора або з інформацією про автора.

Для того щоб бачити нові книги, можна підписатися на автора, тему, підтему або питання, яке цікавить користувача.

У таблиці `Authorfollow` знаходяться підписки на авторів на основі полів:

– `id` – ідентифікатор підписки на автора;

– `author` – автор, на якого підписався користувач, визначається зовнішнім ключем до таблиці `Author`;

– `searcher` – користувач, який підписався на автора, визначається зовнішнім ключем до таблиці `Searcher`.

У таблиці `Topicfollow` знаходяться підписки на теми на основі полів:

– `id` – ідентифікатор підписки на тему;

– `topic` – тема, на яку підписався користувач, визначається зовнішнім ключем до таблиці `Topic`;

– `searcher` – користувач, який підписався на тему, визначається зовнішнім ключем до таблиці `Searcher`.

У таблиці Subtopicfollow знаходяться підписки на підтеми на основі полів:

- id – ідентифікатор підписки на підтему;
- subtopic – підтема, на яку підписався користувач, визначається зовнішнім ключем до таблиці Subtopic;
- searcher – користувач, який підписався на підтему, визначається зовнішнім ключем до таблиці Searcher.

У таблиці Questionfollow знаходяться підписки на питання на основі полів:

- id – ідентифікатор підписки на питання;
- question – питання, на яке підписався користувач, визначається зовнішнім ключем до таблиці Question;
- searcher – користувач, який підписався на питання, визначається зовнішнім ключем до таблиці Searcher.

Комп'ютерна книга (книга з комп'ютерних наук) може мати декілька авторів, а тому для реалізації цього зв'язку створено таблицю Computerbookauthor з полями:

- id – ідентифікатор участі автора в створенні книги з комп'ютерних наук;
- computerbook – комп'ютерна книга (книга з комп'ютерних наук), в написанні якої приймав участь автор, визначається зовнішнім ключем до таблиці Computerbook;
- author – автор, який приймав участь у написанні книги, визначається зовнішнім ключем до таблиці Author;
- role – роль, яку відіграв цей автор у створенні книги (тобто опис його участі в створенні книги).

Автори можуть визначатися не тільки для всієї книги, але і для розділів. Окреме виділення авторів розділів книг разом з наявністю авторів книг обмовлено тим, що не завжди відомо, який саме розділ яким саме

автором виконувався. Якщо це відомо, то такі дані мають знаходитись в таблиці Computerbookauthorpart з полями:

- id – ідентифікатор участі автора в створенні розділу книги з комп'ютерних наук;
- computerbookpart – розділ комп'ютерної книги (книги з комп'ютерних наук), в написанні якого приймав участь автор, визначається зовнішнім ключем до таблиці Computerbookpart;
- computerbookauthor – автор всієї комп'ютерної книги, який приймав участь у створенні цього розділу книги, визначається зовнішнім ключем до таблиці Computerbookauthor;
- role – роль, яку відіграв цей автор у створенні розділу книги (тобто опис його участі в створенні розділу книги).

Видавці книг з комп'ютерних наук знаходяться в таблиці Publisher з полями:

- id – ідентифікатор видавця комп'ютерних книг;
- title – назва видавця комп'ютерних книг;
- officecity – місто, де знаходиться центральний офіс видавця;
- website – вебсайт видавця комп'ютерних книг.

Книги можуть публікуватися не тільки окремо або як перекладні версії, але і в певній серії книг. Серії комп'ютерних книг знаходяться в таблиці Publisherseries з полями:

- id – ідентифікатор серії комп'ютерних книг;
- publisher – видавець цієї серії комп'ютерних книг, визначається зовнішнім ключем до таблиці Publisher;
- seriestitle – назва серії комп'ютерних книг.

Користувачі програми характеризуються в основній частині обліковими записами в таблиці Searcher з полями:

- id – ідентифікатор користувача програми;
- username – логін облікового запису користувача програми;
- password – пароль облікового запису користувача програми;

- photo – фотографія користувача;
- description – інформація про користувача (наприклад, інтереси в сфері комп'ютерних наук).

Переліки комп'ютерної літератури можуть створюватися самим користувачем, щоб повертатися до тієї літератури, яка цікавить користувача. Такі переліки можуть містити як самі книги повністю, так і розділи комп'ютерних книг. Переліки комп'ютерної літератури знаходяться в таблиці Searcherbooklist з полями:

- id – ідентифікатор переліку комп'ютерної літератури;
- searcher – користувач, якому належить цей перелік комп'ютерної літератури, визначається зовнішнім ключем до таблиці Searcher;
- listname – назва переліку комп'ютерної літератури.

Комп'ютерні книги, які входять до переліку, знаходяться в таблиці Searcherbooklistbook з полями:

- id – ідентифікатор комп'ютерної книги з переліку;
- searcherbooklist – перелік комп'ютерної літератури, куди входить книга, визначається зовнішнім ключем до таблиці Searcherbooklist;
- computerbook – комп'ютерна книга, яка входить до цього переліку, визначається зовнішнім ключем до таблиці Computerbook.

Розділи комп'ютерних книг, які входять до переліку, знаходяться в таблиці Searcherbooklistbookpart з полями:

- id – ідентифікатор розділу комп'ютерної книги з переліку;
- searcherbooklist – перелік комп'ютерної літератури, куди входить цей розділ книги, визначається зовнішнім ключем до таблиці Searcherbooklist;
- computerbookpart – розділ комп'ютерної книги, який входить до цього переліку, визначається зовнішнім ключем до таблиці Computerbookpart (рис. 3.3).

Роботу з БД організовано через MySQL.

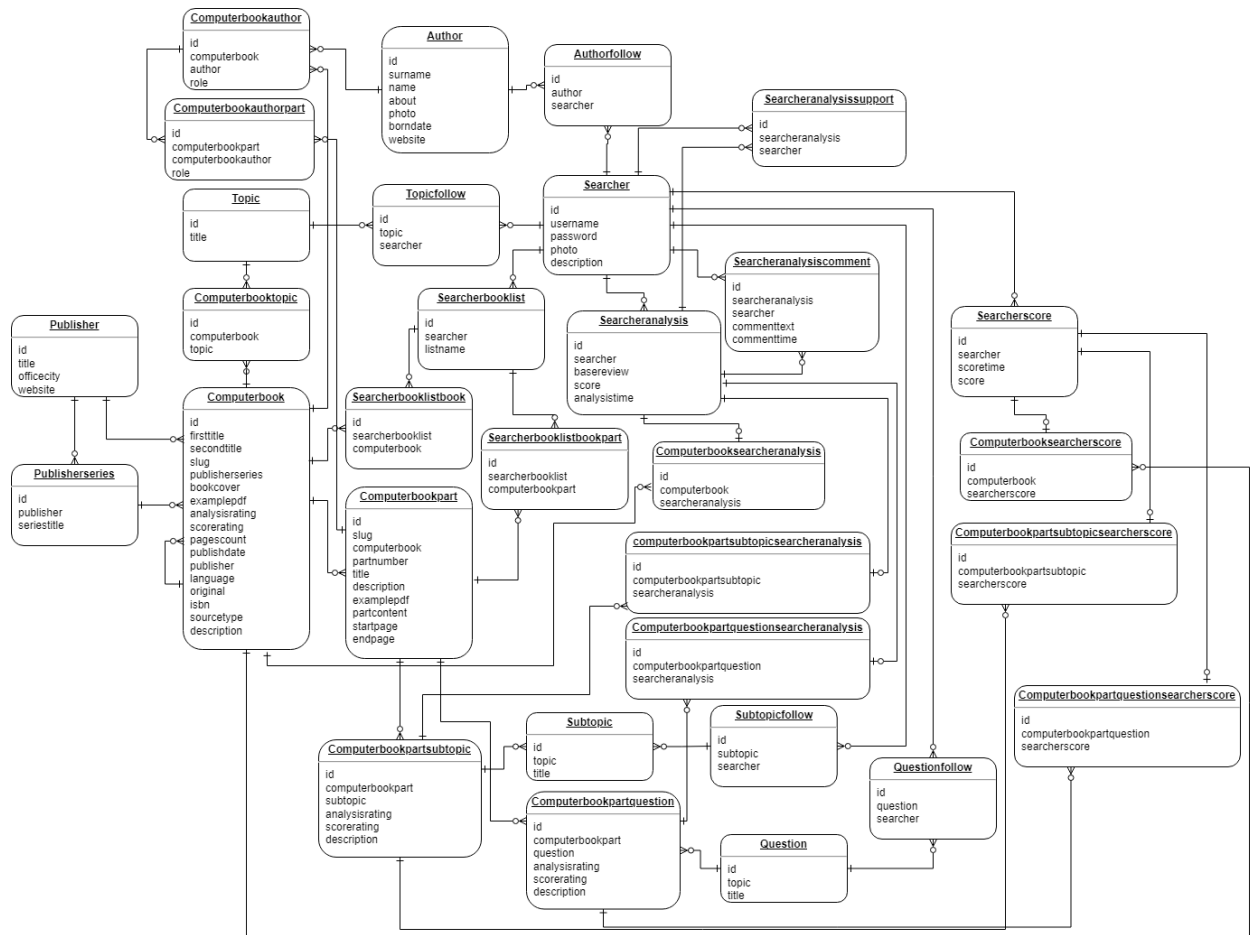


Рисунок 3.3 – Схема БД

3.4 Висновки за розділом 3

Розділ 3 містить структуру роботи з даними, діаграми прецедентів користувача і адміністратора, вибір Python і Django для веброзробки, що є результатом виконання завдання проектування програми пошуку комп'ютерної літератури.

4 РЕАЛІЗАЦІЯ ПРОГРАМИ

4.1 Визначення послідовності роботи з програмою

Послідовність роботи програми виражена порядком роботи з нею користувача (рис. 4.1).

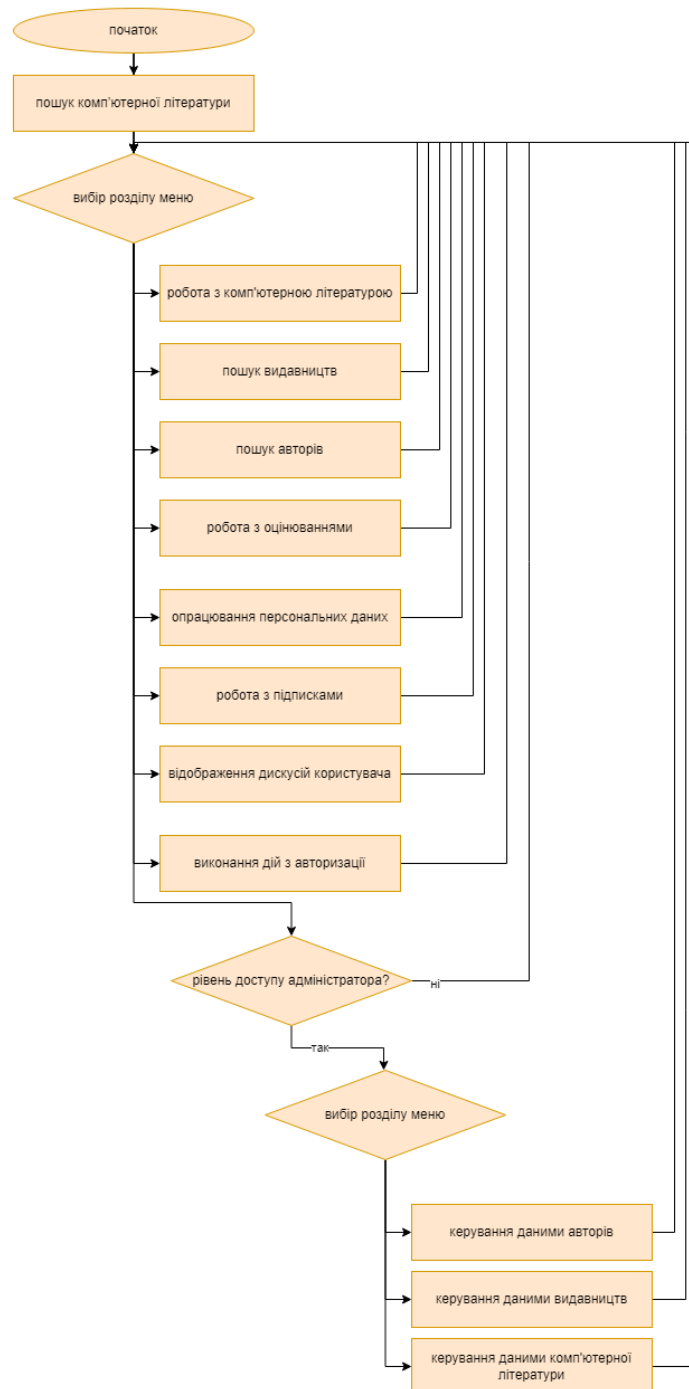


Рисунок 4.1 – Схема функціонування програми пошуку комп'ютерної літератури

4.2 Особливості реалізації програми

Django передбачає доволі сталу структуру застосунку, але вона деталізується конкретними функціями і класами, які залежать від тих сценаріїв, для яких програма використовується.

Функція демонстрування комп'ютерної літератури видавця реалізована як `publishercomputerbooks`. Вона має 3 параметри: параметри запиту вказані в першому – `request` (звідки витягаються дані про авторизацію користувача у функції), параметр `publisherid` є номером видавця, а `booksort` – напрямок сортування визначеної для цього видавця комп'ютерної літератури. Напрямок сортування може бути наступним:

- за зменшенням оцінки з проведених користувачами аналізів літератури (значення 0);
- за зменшенням оцінки з виставлених користувачами оцінювань літератури (значення 1);
- за віддаленням дати публікації літератури від поточної (значення 2);
- за наближенням дати публікації літератури від поточної (значення 3).

Така ж система сортування використовується і в інших функціях за наявності такого параметра. У функція, де застосовується запропонований метод, значення 4 відповідає результатам використання запропонованого методу.

У цій функції, а також у інших функціях, не тільки заповнюється шаблон вебсторінки даними книг, створений за вільним шаблоном [10], але і визначається те, як з такими книгами раніше поведився користувач, якщо він авторизований (чи додавав він цю книгу до своїх переліків та яких саме, чи підписався він на автора книги, які саме оцінки востаннє під час оцінювання та аналізу виставив він цій книзі).

Функція демонстрування переліку книг за темою створена реалізована як `topiccomputerbooks`, окрім перелічених вище параметрів для попередньої

функції сортування та запити функція також має параметр `topic`, представлений назвою теми. Саме у даній функції використовується підтримка методу пошуку комп'ютерної літератури, що було запропоновано, для отримання результатів та впорядкування результатів, якщо в якості значення параметра `booksort` задано 4.

Функція демонстрування переліку книг за підтемою реалізована як `subtopiccomputerbooks`. Серед параметрів має аналогічні параметри, а назва підтеми задається через параметр `subtopic`. Відповідно функція демонстрування переліку книг за питанням реалізована як `questioncomputerbooks`.

Функція демонстрування переліку книг в серії реалізована як `seriescomputerbooks`, де параметр `seriesid` визначає номер серії, за якою вибираються всі книжки. Параметр `booksort` задає напрямок сортування.

Функція демонстрування серій книг видавця реалізована як `publisherseries`, а номер видавця задається параметром `publisherid`. Сортування серій відбувається за часом додавання, тобто серії з більшим ідентифікатором виводяться першими.

Функція демонстрування комп'ютерної літератури з переліку реалізована як `searcherbooklist`. Параметр `booklistid` задає номер такого переліку. Звернення відбувається за отриманими даними облікового запису. Тому якщо цей перелік не належить користувачу, що авторизувався, то дані не будуть відображені. Також підтримується як і у попередніх методах сортування в напрямку, заданому параметром `booksort`.

Функція демонстрування переліку книг за підпискою реалізована як `followbooks`. До результатів включаються всі книги, які створені авторами, на яких підписався користувач, що авторизувався, які представляють тему, на яку підписався користувач, або розділи книг, які стосуються підтеми або питання, на яке підписався користувач. У подальшому перелік об'єднується та впорядковується, тобто і книги, і розділи розташовані послідовно, але без повторень.

Функція отримання інформації про книгу з комп'ютерних наук реалізована як `computerbookabout`. Текстовий унікальний ідентифікатор книги задається через параметр `compbook`. Відбираються всі основні дані про цю комп'ютерну книгу, включаючи серію видання, якщо книга належить до серії, видавця, початкову версію (якщо наприклад, переклад). Також до результатів додається перелік розділів, авторів книги, тем, що розглянуті в книзі, підтем, що розглянуті в розділах книги, питань, що розглянуті в розділах книги. Для авторів, тем, підтем та питань виконується перевірка того, чи підписаний авторизований користувач на них. Окрім того до результатів включаються 5 найновіших оцінювань книги та проведених аналізів книги.

Функція отримання інформації про розділ книги з комп'ютерних наук реалізована як `computerbookpartabout`. Текстовий унікальний ідентифікатор книги задається через параметр `compbookpart`. Відбираються всі основні дані про сам розділ книги, базова інформація про книгу, перелік авторів самого розділу книги, перелік тем книги, перелік підтем та питань, які розглянуті саме в цьому розділі. Для авторів, тем, підтем та питань виконується перевірка того, чи підписаний авторизований користувач на них. Окрім того до результатів включаються 5 найновіших оцінювань цього розділу книги та проведених аналізів розділу книги.

Функція демонстрування історії оцінювань за розділом книги з комп'ютерних наук реалізована як `computerbookpartscores`, а розділ книги задається своїм текстовим ідентифікатором `compbookpart`.

Функція демонстрування історії оцінювань за книгою з комп'ютерних наук реалізована як `computerbookscores`, а номер книги задається своїм текстовим ідентифікатором `compbook`.

Функція демонстрування історії власних оцінювань реалізована як `scoreshistory`. Параметрів окрім `request` функція більше не має, оскільки сам користувач визначається за авторизацією.

Функція демонстрування історії аналізів за розділом книги з комп'ютерних наук реалізована як `computerbookpartanalyses`, а розділ книги задається своїм текстовим ідентифікатором `compbookpart`.

Функція демонстрування історії аналізів за книгою з комп'ютерних наук реалізована як `computerbookanalyses`, а номер книги задається своїм текстовим ідентифікатором `compbook`.

Функція демонстрування історії власних аналізів комп'ютерної літератури реалізована як `analysishistory`. Параметрів окрім `request` функція більше не має, оскільки сам користувач визначається за авторизацією.

Функція пошуку комп'ютерної літератури реалізована як `literaturesearch`. Пошук відбувається одночасно і за книгами, і за розділами книг, об'єднуючи результати разом. Окрім напрямку сортування результатів, значення якого відповідає значенням, описаним вище, задаються також наступні параметри: назва книги або будь-якого розділу з книги, при чому може задаватися як повна назва, так і фрагмент назви, проміжок, в який має потрапляти середня оцінка за результатами аналізу, проміжок, в який має потрапляти середня оцінка за результатами оцінювання, проміжок, в який має потрапляти рік видання книги, мова книги, видавець, номер ISBN, тип видання (друковане або електронне), тему, яка має бути розглянута в книзі, підтему, яка має бути розглянута в розділі книги, питання, яке має бути розглянуто в розділі книги, автори книги. Не всі значення мають бути встановлені, можуть пропускатися ті, для яких значення не потрібно встановлювати.

Функція пошуку авторів реалізована як `authorsearch`. У параметрах пошуку задається ім'я автора.

Функція пошуку видавців реалізована як `publishersearch`. У параметрах пошуку задається назва видавництва та місто, де розташовано головний офіс видавництва.

Функція оцінювання книги з комп'ютерних наук реалізована функцією `bookscorecreate`, при чому параметри, що визначають унікальний

текстовий ідентифікатор книги `compbook` та оцінка `score` задаються без використання форми програмно, як параметри функції. Оскільки аналіз включає ще й текст аналізу, то для реалізації публікування аналізу книги з комп'ютерних наук створено функцію `bookanalysiscreate`, а для підтримки її роботи забезпечено форму `Searcheranalysisform`. На основі тієї ж форми реалізовано функцію `bookpartanalysiscreate` для публікування аналізу підтеми в розділі книги з комп'ютерних наук та функцію публікування аналізу питання в розділі книги з комп'ютерних наук `questionanalysiscreate`. Ця ж форма лежить в основі функції `bookpartanalysisedit` редагування опублікованого аналізу комп'ютерної літератури для внесення змін в описані вище результати.

Функція отримання інформації про видавця реалізовано як `publisherabout`, номер видавця при цьому передається через параметр `publisher`. Для визначення інформації про видавця реалізовано функцію `publishercreate` та форму `Publisherform`. Через неї ж працює функція `publishedit`.

Функція підписування на книги автора реалізована як `authorfollowcreate`, параметр якої `author` визначає номер автора. Підписування реалізується для авторизованого користувача. Функція підписування на книги за темою реалізована як `topicfollowcreate`, параметр якої `topic` визначає назву теми. Функція підписування на книги за підтемою реалізована як `subtopicfollowcreate`, параметр якої `subtopic` визначає назву підтеми. Функція підписування на книги за питанням реалізована як `questionfollowcreate`, параметр якої `question` визначає питання.

Функція скасування підписування на книги автора реалізована як `authorfollowdelete`, параметр якої `author` визначає номер автора, а користувач визначається даними авторизації. Функція скасування підписування на книги за темою реалізована як `topicfollowdelete`, параметр якої `topic` визначає назву теми. Функція скасування підписування на книги за підтемою реалізована як `subtopicfollowdelete`, параметр якої `subtopic` визначає назву підтеми. Функція

скасування підписування на книги за питанням реалізована як `questionfollowdelete`, параметр якої `question` визначає питання.

Аналогічно реалізовані всі функції представлень.

Модель книги з комп'ютерних наук реалізована класом `Computerbook`, який має набір реалізованих атрибутів.

Атрибут `firsttitle` реалізовано як тип `CharField`, представляє основну назву комп'ютерної літератури (книги з комп'ютерних наук).

Атрибут `secondtitle` реалізовано як тип `CharField`, представляє додаткову назву комп'ютерної літератури (або назву іншою мовою).

Атрибут `slug` реалізовано як тип `SlugField`, представляє текстовий ідентифікатор комп'ютерної книги (книги з комп'ютерних наук);

Атрибут `publisherseries` реалізовано як тип `ForeignKey`, представляє серію, до якої належить ця комп'ютерна книга, тобто визначає зовнішній ключ до моделі `Publisherseries`.

Атрибут `bookcover` реалізовано як тип `ImageField`, представляє фотографію титульної сторінки комп'ютерної книги (книги з комп'ютерних наук).

Атрибут `examplepdf` реалізовано як тип `FileField`, представляє витяг з комп'ютерної книги (книги з комп'ютерних наук), представлений `pdf`-файлом.

Атрибут `analysisrating` реалізовано як тип `FloatField`, представляє середній рейтинг, виставлений в проведених користувачами аналізах цієї комп'ютерної книги.

Атрибут `scorerating` реалізовано як тип `FloatField`, представляє середній рейтинг, виставлений у оцінках, наданих комп'ютерній книзі користувачами (без аналізу, тільки оцінка).

Атрибут `pagescount` реалізовано як тип `PositiveIntegerField`, представляє кількість сторінок комп'ютерної книги (книги з комп'ютерних наук).

Атрибут `publishdate` реалізовано як тип `DateField`, представляє дату публікації комп'ютерної книги (книги з комп'ютерних наук).

Атрибут `publisher` реалізовано як тип `ForeignKey`, представляє видавця комп'ютерної книги (книги з комп'ютерних наук), тобто визначає зовнішній ключ до моделі `Publisher`.

Атрибут `isbn` реалізовано як тип `CharField`, представляє ISBN книги.

Атрибут `language` реалізовано як тип `CharField`, представляє мову, якою написана комп'ютерна книга (книга з комп'ютерних наук).

Атрибут `original` реалізовано як тип `ForeignKey`, представляє комп'ютерну книгу (книгу з комп'ютерних наук), яка є оригіналом даної, визначається зовнішнім ключем до моделі `Computerbook`.

Атрибут `description` реалізовано як тип `CharField`, представляє опис комп'ютерної книги (книги з комп'ютерних наук).

Атрибут `sourcetype` реалізовано як тип `BooleanField`, представляє тип основної публікації комп'ютерної книги (електронне видання, фізичне надруковане).

Атрибут `topics` реалізовано як тип `ManyToManyField` на основі зв'язку багато до багатьох до моделі `Topic`, представляє собою перелік тем, які розглянуті в цій комп'ютерній книзі (книзі з комп'ютерних наук).

Модель розділу книги з комп'ютерних наук реалізована класом `Computerbookpart`, який має набір реалізованих атрибутів.

Атрибут `slug` реалізовано як тип `SlugField`, представляє текстовий ідентифікатор розділу комп'ютерної книги (книги з комп'ютерних наук).

Атрибут `computerbook` реалізовано як тип `ForeignKey`, представляє комп'ютерну книгу (книгу з комп'ютерних наук), в якій є цей розділ, визначається зовнішнім ключем до моделі `Computerbook`.

Атрибут `partnumber` реалізовано як тип `SmallIntegerField`, представляє номер розділу за порядком, у якому він приведений у цій комп'ютерній книзі (книзі з комп'ютерних наук).

Атрибут `title` реалізовано як тип `CharField`, представляє назву комп'ютерної книги (книги з комп'ютерних наук).

Атрибут `description` реалізовано як тип `CharField`, представляє повний опис комп'ютерної книги (книги з комп'ютерних наук).

Атрибут `examplepdf` реалізовано як тип `FileField`, представляє витяг з розділу комп'ютерної книги (книги з комп'ютерних наук), представлений pdf-файлом.

Атрибут `partcontent` реалізовано як тип `CharField`, представляє зміст даного розділу комп'ютерної книги (книги з комп'ютерних наук), тобто формальне представлення того, як цей розділ структуровано саме в самій книзі.

Атрибут `startpage` реалізовано як тип `PositiveIntegerField`, представляє сторінку в самій комп'ютерній книзі (книзі з комп'ютерних наук), з якою починається цей розділ.

Атрибут `endpage` реалізовано як тип `PositiveIntegerField`, представляє сторінку в самій комп'ютерній книзі (книзі з комп'ютерних наук), якою закінчується цей розділ.

Модель теми з комп'ютерних наук реалізована класом `Topic` з одним атрибутом з безпосереднього поля таблиці у БД `title`, адже автоматично створюється первинний ключ в Django, що визначає назву самої теми з комп'ютерних наук та атрибутом `followers`, який реалізовано як тип `ManyToManyField` на основі зв'язку багато до багатьох до моделі `Searcher`, представляє собою перелік користувачів, які підписалися на цього автора.

Модель підтеми з комп'ютерних наук реалізована класом `Subtopic`, який має набір реалізованих атрибутів.

Атрибут `topic` реалізовано як тип `ForeignKey`, представляє тему з комп'ютерних наук, до якої відноситься ця підтема, визначається зовнішнім ключем до моделі `Topic`.

Атрибут `title` реалізовано як тип `CharField`, представляє назву самої підтеми з комп'ютерних наук.

Атрибут `followers` реалізовано як тип `ManyToManyField` на основі зв'язку багато до багатьох до моделі `Searcher`, представляє собою перелік користувачів, які підписалися на цю підтему.

Моделювання питання з комп'ютерних наук реалізовано класом `Question`, який має набір реалізованих атрибутів.

Атрибут `topic` реалізовано як тип `ForeignKey`, представляє тему з комп'ютерних наук, до якої відноситься це питання, визначається зовнішнім ключем до моделі `Topic`.

Атрибут `title` реалізовано як тип `CharField`, представляє назву самого питання з комп'ютерних наук.

Атрибут `followers` реалізовано як тип `ManyToManyField` на основі зв'язку багато до багатьох до моделі `Searcher`, представляє собою перелік питань, які підписалися на цього автора.

Моделювання визначення підтем, які розглядаються в розділах комп'ютерної книги, реалізовано класом `Computerbookpartsubtopic`, який має набір реалізованих атрибутів.

Атрибут `computerbookpart` реалізовано як тип `ForeignKey`, представляє розділ книги з комп'ютерних наук, в якому ця підтема розглядається, визначається зовнішнім ключем до моделі `Computerbookpart`.

Атрибут `subtopic` реалізовано як тип `ForeignKey`, представляє підтему з комп'ютерних наук, яка розглядається в цьому розділі книги, визначається зовнішнім ключем до моделі `Subtopic`.

Атрибут `analysisrating` реалізовано як тип `FloatField`, представляє середній рейтинг, виставлений в проведених користувачами аналізах цього розділу комп'ютерної книги.

Атрибут `scorerating` реалізовано як тип `FloatField`, представляє середній рейтинг, виставлений у оцінках, наданих розділу комп'ютерної книги користувачами (без аналізу, тільки оцінка).

Атрибут `description` як тип `CharField`, представляє опис стосовно того, як саме ця підтема розглядається в цьому розділі книги.

Модель визначення питань, які розглядаються в розділах комп'ютерної книги, реалізована класом `Computerbookpartquestion`, який має набір реалізованих атрибутів.

Атрибут `computerbookpart` реалізовано як тип `ForeignKey`, представляє розділ книги з комп'ютерних наук, в якому це питання розглядається, визначається зовнішнім ключем до моделі `Computerbookpart`.

Атрибут `question` реалізовано як тип `ForeignKey`, представляє питання з комп'ютерних наук, яке розглядається в цьому розділі книги, визначається зовнішнім ключем до моделі `Question`.

Атрибут `analysisrating` реалізовано як тип `FloatField`, представляє середній рейтинг, виставлений в проведених користувачами аналізах цього питання, представленого в розділі комп'ютерної книги.

Атрибут `scorerating` реалізовано як тип `FloatField`, представляє середній рейтинг, виставлений у оцінках, наданих цьому питанню, представленому в розділі комп'ютерної книги (без аналізу, тільки оцінка).

Атрибут `description` реалізовано як тип `CharField`, представляє опис стосовно того, як саме це питання розглядається в цьому підрозділі книги.

Модель оцінювання користувачами контенту (книги, підтеми в розділі книги, питання в розділі книги) реалізована класом `Searcherscore`, який має набір реалізованих атрибутів.

Атрибут `searcher` реалізовано як тип `ForeignKey`, представляє користувача, який це оцінювання контенту виконав, визначається зовнішнім ключем до моделі `Searcher`.

Атрибут `scoretime` реалізовано як тип `DateTimeField`, представляє час, коли оцінювання користувачем контенту було виконано.

Атрибут `score` реалізовано як тип `PositiveSmallIntegerField`, представляє оцінку, яку користувач виставив контенту.

Оскільки `Searcherscore` може належати до різного контенту, якою може відноситися до моделей `Computerbook`, `Computerbookpartsubtopic`,

Computerbookpartquestion, то у складі цієї моделі було визначено також такі атрибути:

- content_type – тип контенту, що реалізовано як тип ForeignKey на основі зв'язування з класом ContentType з підпаketу django.contrib.contenttypes.models;
- object_id – фактично значення ключа, за яким зазвичай відбувається зв'язування між моделями, має тип PositiveIntegerField;
- content_object – вміст об'єкта, який відповідає значенню ключа, має тип GenericForeignKey і в ньому прив'язуються атрибути content_type та object_id.

Модель аналізу контенту користувачем реалізована класом Searcheranalysis, який має набір реалізованих атрибутів.

Атрибут searcher реалізовано як тип ForeignKey, представляє користувача, який цей аналіз контенту виконав, визначається зовнішнім ключем до моделі Searcher.

Атрибут basereview реалізовано як тип CharField, представляє текст самого проведеного аналізу.

Атрибут analysistime реалізовано як тип DateTimeField, представляє час, коли аналіз користувачем контенту було виконано.

Атрибут score реалізовано як тип PositiveSmallIntegerField, представляє оцінку, яку користувач виставив контенту в результаті аналізу.

Атрибут supports реалізовано як тип ManyToManyField на основі зв'язку багато до багатьох до моделі Searcher, представляє собою перелік користувачів, які висловили підтримку аналізу контенту, виконаному іншим користувачем.

Так само, як це було з оцінюванням, у даному випадку можливий зв'язок з різними типами контенту, що представляє собою моделі Computerbook, Computerbookpartsubtopic, Computerbookpartquestion.

Тому додатковими атрибутами моделі є:

– `content_type` – тип контенту, що реалізовано як тип `ForeignKey` на основі зв'язування з класом `ContentType` з підпаketу `django.contrib.contenttypes.models`;

– `object_id` – фактично значення ключа, за яким зазвичай відбувається зв'язування між моделями, має тип `PositiveIntegerField`;

– `content_object` – вміст об'єкта, який відповідає значенню ключа, має тип `GenericForeignKey` і в ньому прив'язуються атрибути `content_type` та `object_id`.

Модель коментування результатів аналізу реалізована класом `Searcheranalysiscomment`, який має набір реалізованих атрибутів.

Атрибут `searcheranalysis` реалізовано як тип `ForeignKey`, представляє виконаний аналіз контенту, до якого відноситься коментар, визначається зовнішнім ключем до моделі `Searcheranalysis`.

Атрибут `searcher` реалізовано як тип `ForeignKey`, представляє користувача, який створив коментар до аналізу контенту, визначається зовнішнім ключем до моделі `Searcher`.

Атрибут `commenttext` реалізовано як тип `CharField`, представляє текст коментаря.

Атрибут `commenttime` реалізовано як тип `DateTimeField`, представляє час створення коментаря.

Модель авторів книжок реалізована класом `Author`, який має набір реалізованих атрибутів.

Атрибут `surname` реалізовано як тип `CharField`, представляє прізвище автора.

Атрибут `name` реалізовано як тип `CharField`, представляє ім'я автора.

Атрибут `about` реалізовано як тип `CharField`, представляє інформація про автора.

Атрибут `photo` реалізовано як тип `ImageField`, представляє фотографію автора.

Атрибут `borndate` реалізовано як тип `DateField`, представляє дата народження автора.

Атрибут `website` реалізовано як тип `URLField`, представляє вебсайт автора або з інформацією про автора.

Атрибут `followers` реалізовано як тип `ManyToManyField` на основі зв'язку багато до багатьох до моделі `Searcher`, представляє собою перелік користувачів, які підписалися на цього автора.

Модель авторства у комп'ютерній книзі (книзі з комп'ютерних наук) реалізована класом `Computerbookauthor`, який має набір реалізованих атрибутів.

Атрибут `computerbook` реалізовано як тип `ForeignKey`, представляє комп'ютерну книгу (книгу з комп'ютерних наук), в написанні якої приймав участь автор, визначається зовнішнім ключем до таблиці `Computerbook`.

Атрибут `author` реалізовано як тип `ForeignKey`, представляє автора, який приймав участь у написанні книги, визначається зовнішнім ключем до моделі `Author`.

Атрибут `role` реалізовано як тип `CharField`, представляє роль, яку відіграв цей автор у створенні книги (тобто опис його участі в створенні книги).

Модель авторства у розділі комп'ютерної книги (книги з комп'ютерних наук) реалізована класом `Computerbookauthorpart`, який має набір реалізованих атрибутів.

Атрибут `computerbookpart` реалізовано як тип `ForeignKey`, представляє розділ комп'ютерної книги (книги з комп'ютерних наук), в написанні якого приймав участь автор, визначається зовнішнім ключем до моделі `Computerbookpart`.

Атрибут `computerbookauthor` реалізовано як тип `ForeignKey`, представляє автора всієї комп'ютерної книги, який приймав участь у створенні цього розділу книги, визначається зовнішнім ключем до моделі `Computerbookauthor`.

Атрибут `role` реалізовано як тип `CharField`, представляє роль, яку відіграв цей автор у створенні книги (тобто опис його участі в створенні книги).

Модель видавця книги з комп'ютерних наук реалізована класом `Publisher`, який має набір реалізованих атрибутів.

Атрибут `title` реалізовано як тип `CharField`, представляє назву видавця комп'ютерних книг.

Атрибут `officacity` реалізовано як тип `CharField`, представляє місто, де знаходиться центральний офіс видавця.

Атрибут `website` реалізовано як тип `URLField`, представляє вебсайт видавця комп'ютерних книг.

Модель серії комп'ютерних книг реалізована класом, який має набір реалізованих атрибутів.

Атрибут `publisher` реалізовано як тип `ForeignKey`, представляє видавця цієї серії комп'ютерних книг, визначається зовнішнім ключем до моделі `Publisher`.

Атрибут `seriestitle` реалізовано як тип `CharField`, представляє назву серії комп'ютерних книг.

Модель користувача програми `Searcher` реалізована наслідуванням моделі `User`, що використовується для надання засобів аутентифікації в Django, а тому `username`, `password` використовуються саме з цієї моделі, а додатково визначено наступні атрибути.

Атрибут `photo` реалізовано як тип `ImageField`, представляє фотографію користувача.

Атрибут `description` реалізовано як тип `CharField`, представляє інформацію про користувача.

Модель переліку комп'ютерної літератури реалізована класом `Searcherbooklist`, який має набір реалізованих атрибутів.

Атрибут `searcher` реалізовано як тип `ForeignKey`, представляє користувача, якому належить цей перелік комп'ютерної літератури, визначається зовнішнім ключем до моделі `Searcher`.

Атрибут `listname` реалізовано як тип `CharField`, представляє назву переліку комп'ютерної літератури.

Атрибут `computerbooks` реалізовано як тип `ManyToManyField` на основі зв'язку багато до багатьох до моделі `Computerbook`, представляє собою перелік комп'ютерних книг, які входять до переліку комп'ютерної літератури.

Атрибут `computerbookparts` реалізовано як тип `ManyToManyField` на основі зв'язку багато до багатьох до моделі `Computerbookpart`, представляє собою перелік розділів комп'ютерних книг, які входять до переліку комп'ютерної літератури.

4.3 Висновки за розділом 4

Розділ 4 містить результати виконання завдання реалізації програми пошуку пошуку комп'ютерної літератури стосовно створеного коду програми, включаючи опис функцій і класів, що розроблені, а також схему функціонування програми.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Призначення програми

Програма призначена для пошуку комп'ютерної літератури, включаючи як книги з комп'ютерних наук загалом, так і окремі розділи цих книг, надаючи релевантну інформацію щодо тих проблем, які цікавлять користувача.

5.2 Виконання програми

Виконання програми розпочинається з пошуку, який відбувається одночасно і за книгами, і за розділами книг, об'єднуючи результати разом. Для виконання пошуку можуть задаватися:

- назва книги або будь-якого розділу з книги;
- проміжок, в який має потрапляти середня оцінка за результатами аналізу;
- проміжок, в який має потрапляти середня оцінка за результатами оцінювання;
- проміжок, в який має потрапляти рік видання книги;
- мова книги;
- видавець;
- номер ISBN;
- тип видання (друковане або електронне);
- тема, яка має бути розглянута в книзі;
- підтема, яка має бути розглянута в розділі книги;
- питання, яке має бути розглянуто в розділі книги;
- автори книги.

В пошуку окрім цього одразу має бути задано напрям сортування:

- за зменшенням оцінки з проведених користувачами аналізів літератури;
- за зменшенням оцінки з виставлених користувачами оцінювань літератури;
- за віддаленням дати публікації літератури від поточної;
- за наближенням дати публікації літератури до поточної.

Пошук також доступний з меню «Література», де у випадяючому меню можна також обрати теми, питання. За кожним з них на сторінці можна обрати потрібну, наприклад, тему (або ввести потрібну), а тоді відобразиться результат – книги за темою (рис. 5.1) або розділи книг за питанням. Результати також можна сортувати, обираючи потрібну вкладку, але при цьому доступна і вкладка «Рекомендації», яка дозволяє переглянути перелік, сформований для користувача за запропонованим методом, якщо він виконував аналізування або оцінювання книг раніше.

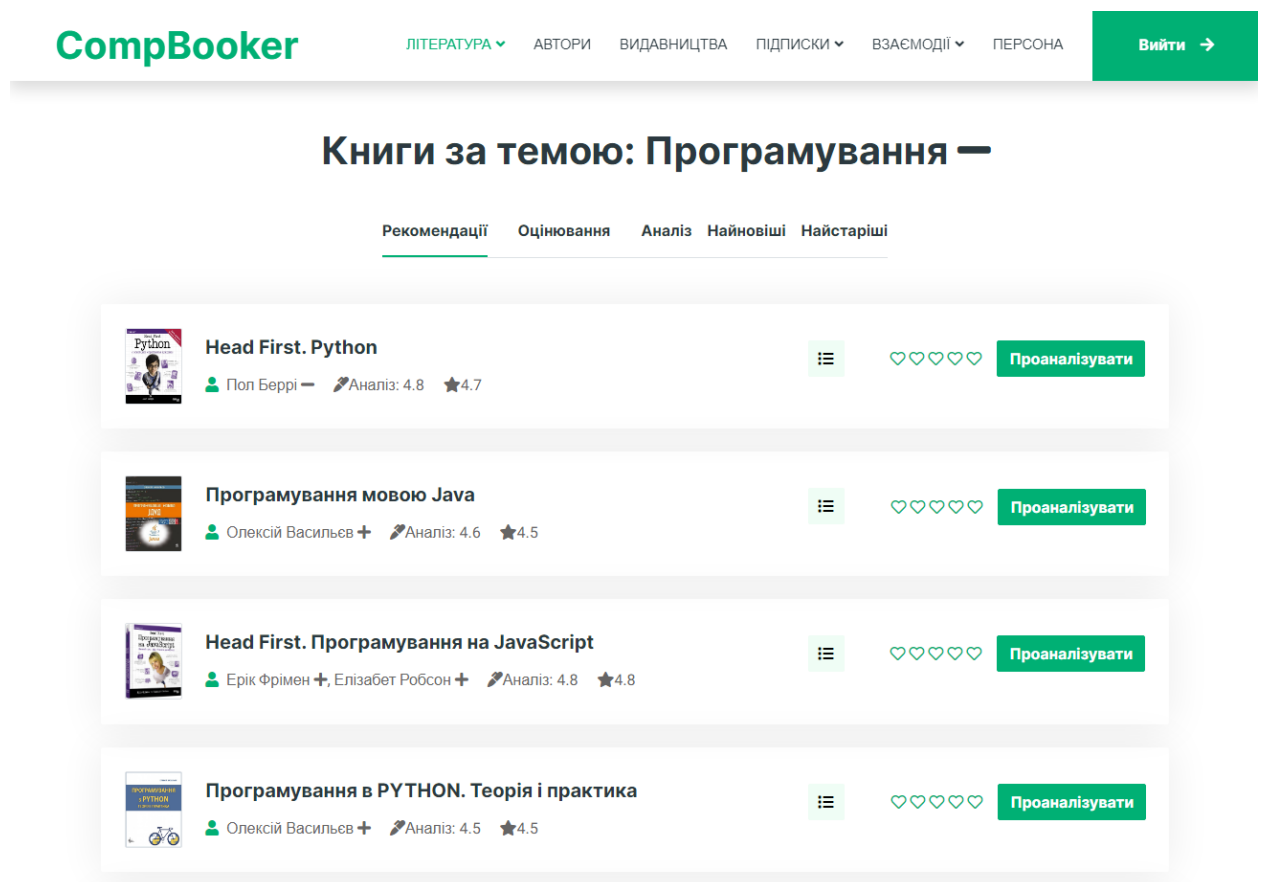


Рисунок 5.1 – Демонстрування переліку книг за темою

Зображення серцець біля книги дозволяє одразу виконати оцінювання, натискаючи на відповідну позицію, що позначає рівень оцінки. Якщо потрібно створити повноцінний аналіз, описавши своє рішення, то потрібно натиснути на кнопку «Проаналізувати». За кожною книгою відображаються її автори, а біля кожного автора іконка, яка визначає, що на автора оформлено підписку (тоді показується мінус) або що на автора підписка не оформлена (плюс). Натискаючи на це зображення, можна додати або вилучити підписку на автора. Також біля книги показується середня оцінка за результатами аналізу (поруч з маркером), за результатами оцінювань (біля зірочки). Якщо раніше було проведено аналіз, то в дужках поруч з середньою оцінкою вказується оцінка користувача. Також справа позначення переліка дозволяє у випадяючому переліку переглянути переліки, в які користувач додав цю книгу.

Користувач також може відкрити сторінку з підписками (рис. 5.2).

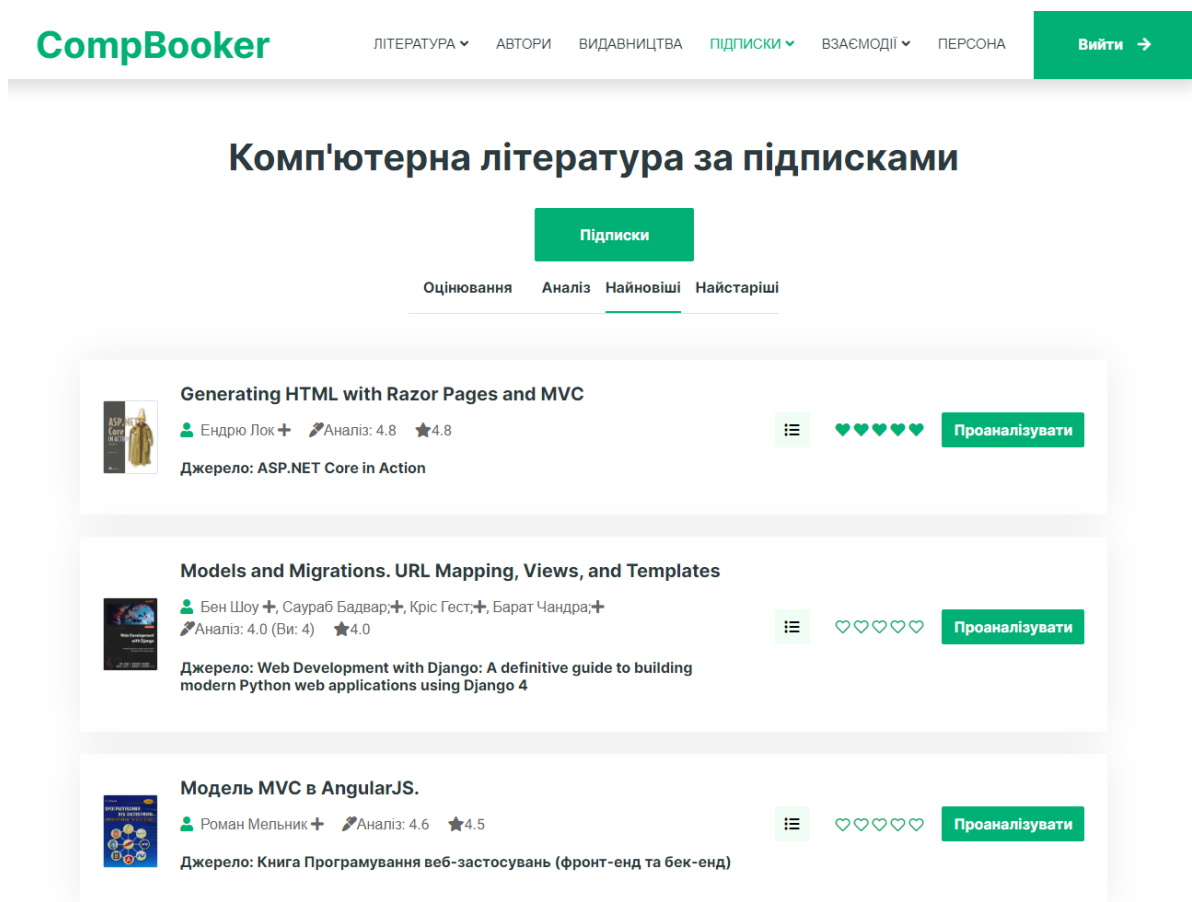


Рисунок 5.2 – Демонстрування переліку книг (розділів книг) за підпискою

На сторінці з підписками відображаються всі книги, розділи книг, які відповідають авторам, темам, підтемам, категоріям, на які підписався користувач. У даному випадку підписка була оформлена тільки на питання Model View Controller. Відповідно оцінки відображаються стосовно саме цього питання.

Окрім сторінки з результатами підписок користувачу з меню «Підписки» також доступний і перелік того, на що він підписався.

З меню «Автори» можна перейти до пошуку авторів книг, а звідки до відповідно книг автора. З меню «Видавництва» можна перейти до пошуку видавництв. У випадку авторизації адміністратора він може з цих меню також вибирати варіанти, які дозволяють додавати нових авторів, видавництва. Адміністратору також на відповідних сторінках доступні кнопки редагування.

З меню «Персона» користувач може переглянути інформацію власного облікового запису і одразу її змінити, якщо потрібно.

З меню «Взаємодії» доступні варіанти для переходу до власних оцінювань, власних опублікованих аналізів літератури, дискусій, у яких брав участь користувач, тобто коментарі до аналізів.

Користувач може перейти також на сторінку книги, на сторінку розділу, звідки доступна вся основна інформація з можливістю переходу до деталізації історій. Зокрема історії аналізування, наприклад, розділу. З тієї сторінки тоді можна обрати, яке питання або яка підтема цікавить користувача. Після того, як підтема або питання обрано, відображаються аналізування саме за цією підтемою або питанням (рис. 5.3). За кожним аналізуванням відображається автор, текст аналізу, оцінка, посилання, яке дозволяє відкрити перелік коментарів до аналізування, якщо коментарі були створені, посилання, яке відкриває форму створення коментаря, посилання, яке дозволяє підтримати аналізування (якщо вже підтримано, то посилання відсутнє), а також інформація про кількість підтримок цього аналізування.

Аналізування розділу Models and Migrations. URL Mapping, Views, and Templates за питанням MVC —



Іван Знаменний

15.12.2023 23:22

Якщо загалом книга достатньо непогана, то стосовно MVC потрібно сказати, що її не треба розглядати в цьому контексті. Так, в ній наявний в принципі цей розділ, однак по-перше, це ж не MVC, а MVT. Маю на увазі, що в Django рішення відрізняється. І в контексті цього рішення написано достатньо непогано, але якщо розглядати цей розділ як опис MVC, то він зіб'є, заплутає і швидше за все про MVC ви нічого не зрозумієте. Але якщо це не перші кроки, тоді в принципі може бути непогано, тому і оцінка така.



Коментарів: 0

Додати коментар

Підтримки: 3

Підтримати



Віктор Леонченко

16.12.2023 20:31

MVC тут доволі викривлений через Django. Тобто це MVC по-Django. І в такому варіанті описано непогано, але все ж для вивчення саме MVC не підійде.



Коментарів: 0

Додати коментар

Підтримки: 1

Підтримати



Мирослав Торначний

16.12.2023 20:45



Рисунок 5.3 — Отримання інформації про розділ книги з комп'ютерних наук

5.3 Експериментальне дослідження

Вибірка [11] використана для проведення експериментального дослідження. Вибірка складається з даних, які було витягнуто з вебсайту Goodreads [12]. Серед всіх наявних у вибірці характеристик екземплярів даних було витягнуто наступні:

- основний автор джерела, яким вважався автор, який вказаний у книзі першим;
- кількість оцінок, які були виставлені джерелу через оцінювання (без аналізу);
- кількість оцінок, які були виставлені джерелу через аналіз;
- середня оцінка джерела;
- рік публікації джерела, який був витягнутий з дати публікації, яка могла бути представлена числом у текстовому описі або місяцем, тому було

використано програмне оброблення, яке дозволило за шаблоном виділити рік з дати, а якщо рік був вказаний некоректно (наприклад, три цифри), то такі екземпляри відкидалися;

– видавець джерела;

– входження книги до серії видань або відсутність такого входження: якщо за відповідною ознакою були відсутні дані про серію видання, то вважалось, що книга не входить у серію.

Відкидаючи екземпляри з порожніми значеннями, якщо вони були отримані після такого оброблення даних, вилучаючи дублікати, було використано частину вибірку для проведення дослідження. Оскільки метод передбачає пошук за фактично конкретним напрямком, то варіантів вибору фактично мають бути не десятки тисяч, тому простір пошуку був звужений. Тобто комп'ютерної літератури може бути і дуже багато, але за конкретним питанням, темою, підтемою вже пошук буде звужений.

Під час експериментів було порівняно два методи кластеризації: метод k-середніх (табл. 5.1) та спектральна кластеризація (табл. 5.2).

Таблиця 5.1 – Використання методу k-середніх

Номер кластеру	Середня відстань всередині кластера	Мінімальна середня відстань між кластерами	Середня відстань між кластерами
0	0,02833938	0,04964135	0,051930566
1	0,009819105	0,012310325	0,023537541
2	0,016702048	0,014083161	0,023576244
3	0,012434121	0,012310325	0,023128767
4	0,013663798	0,012805369	0,023163394

Порівняння виконувалось на основі обчислення відстані між екземплярами. Відстань між екземплярами обчислювалась як сума квадратів різниць між значеннями відповідних ознак екземплярів. Відстань

обчислювалась окремо для категорій (авторів, видавництв), оскільки віднімання при цьому не є коректним, тому якщо значення у екземплярів за цими ознаками не співпадали, то додавалась 1, а в протилежному випадку нічого не додавалось.

Таблиця 5.2 – Використання спектральної кластеризації

Номер кластеру	Середня відстань всередині кластера	Мінімальна середня відстань між кластерами	Середня відстань між кластерами
0	0,013356447	0,013484824	0,026068068
1	0,025813239	0,052423492	0,055881299
2	0,009332151	0,012184824	0,02367195
3	0,016149931	0,017526867	0,024500129
4	0,006521888	0,013747065	0,027542732

Порівняння відбувалось окремо всередині кластера, окремо між різними кластерами. Для кожного кластера було визначено інший кластер, з яким у нього була мінімальна середня відстань у екземплярів та обчислено середнє значення стосовно всіх кластерів.

За використання методу k-середніх (табл. 5.1) не завжди мінімальна середня відстань між різними кластерами є меншою порівняно з середньою відстанню всередині кластера. Фактично у 3 з 5 випадків за використання методу k-середніх мінімальна середня відстань між різними кластерами виявилась меншою ніж відстань всередині такого кластера.

У той же час за використання спектральної кластеризації розділення на кластери виявилось кращим, адже у всіх 5 випадках значення мінімальної відстані між кластерами виявилось більшим за середню відстань всередині кластера. Тому для виконання кращого розділення даних на кластери варто використовувати спектральну кластеризацію.

5.4 Висновки за розділом 5

Розділ 5 містить опис роботи користувача з програмою, основні інтерфейси та логіку такої роботи, результати експериментального дослідження, які засвідчили, що метод спектральної кластеризації порівняно з методом k-середніх дозволив розбити дані на кластери таким чином, що вони ближче розташовані всередині кластерів ніж відносно екземплярів з інших кластерів.

ВИСНОВКИ

У роботі необхідно було розробити програмне забезпечення пошуку комп'ютерної літератури для знаходження релевантної літератури за проблемами, які цікавлять користувача, що і було в результаті виконано.

Програма, яку було створено, призначена для пошуку комп'ютерної літератури, включаючи як книги з комп'ютерних наук загалом, так і окремі розділи цих книг, надаючи релевантну інформацію щодо тих проблем, які цікавлять користувача. Проблеми виражаються темами, підтемами та питаннями. Користувачі можуть виконувати оцінювання та аналіз книг і розділів стосовно зокрема підтем та питань, які в цих розділах розглядаються. Це дозволяє одразу отримувати інформацію не про книгу загалом, а в розрізі тих проблем, які цікавлять користувача, а відповідно і виконувати пошук саме в такому розрізі, коли це потрібно.

Наукова новизна роботи полягає в запропонованому методі пошуку комп'ютерної літератури, який передбачає співвіднесення комп'ютерної літератури з проблемами, які в ній розглядаються, виділення літератури, якій за цією проблемою віддає перевагу користувач, розділення літератури за проблемою на кластери, визначення кластеру, до якого належить література, якій віддає перевагу користувач, та формування таким чином результатів пошуку. В основі виконання кластеризації запропоновано використовувати спектральну кластеризацію. У роботі проведено експериментальне дослідження, в якому було отримано результати стосовно розподілу літератури на кластери та якості такого розподілу.

Усі початково поставлені для виконання в роботі завдання було в підсумку виконано: виконано аналіз існуючих програм, які виконують пошук комп'ютерної літератури, досліджено методи і моделі пошуку комп'ютерної літератури, виконано проєктування і реалізацію програми пошуку комп'ютерної літератури.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Amazon.com [Electronic resource] : Books. – Access mode : <https://www.amazon.com/books-used-books-textbooks/b?ie=UTF8&node=283155>.
2. Recent updates. Goodreads [Electronic resource]. – Access mode : <https://www.goodreads.com/>.
3. Understanding K-Means Clustering Algorithm [Electronic resource] : Steps and Techniques for Accurate Cluster Analysis. Medium. – Access mode : <https://denizpekcanli.medium.com/what-is-k-means-clusterization-16c2c38beae1>.
4. ML Clustering [Electronic resource] : When To Use Cluster Analysis, When To Avoid It. – Access mode : <https://www.explorium.ai/blog/machine-learning/clustering-when-you-should-use-it-and-avoid-it/>.
5. Clustering. Introduction, Different Methods, and Applications [Electronic resource]. – Access mode : <https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/>.
6. Spectral Clustering [Electronic resource]. – Access mode : <https://www.youngwonks.com/blog/spectral-clustering>.
7. Welcome to Pyramid, a Python Web Framework [Electronic resource]. – Access mode : <https://trypyramid.com/>.
8. Django introduction – Learn web development. MDN [Electronic resource]. – Access mode : <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction>.
9. Why We Use Django Framework & What Is Django Used For. Django Stars [Electronic resource]. – Access mode : <https://djangostars.com/blog/why-we-use-django-framework/>.
10. JobEntry – Job Portal Website Template [Electronic resource]. – Access mode : <https://htmlcodex.com/job-portal-website-template/>.
11. Reese, A. Goodreads Books – 31 Features [Electronic resource] /

A. Reese. – Access mode : <https://www.kaggle.com/datasets/austinreese/goodreads-books>.

12. Best Books Ever (120091 books). Goodreads [Electronic resource]. – Access mode : https://www.goodreads.com/list/show/1.Best_Books_Ever.

ДОДАТОК А
Технічне завдання

Вступ

Для пошуку літератури, створення і ознайомлення з відгуками, придбання літератури вже є готові програмні рішення. Однак, особливістю комп'ютерної літератури зокрема є те, що в ній можуть розглядатися різні теми, проблеми, а цікавити читача можуть не всі одразу, а тільки певні з них. При цьому традиційно пошук у всіх таких системах виконується для книги в цілому. Тобто якщо якесь питання є специфічним, невеликим, то знайти інформацію про нього стає достатньо складно. Якщо користувач знає, яка книга його цікавить, то подальші дії не є проблемою, а ось якщо потрібно виконувати такий пошук, то це проблема на даний момент.

Отримання інформації про різні складові комп'ютерної літератури, зокрема розділи книги (хоча такими складовими можуть бути не тільки розділи загалом, але і дрібніші складові – пункти, окремі питання), є важливим в розрізі комп'ютерної літератури. Наприклад, в художній літературі важливою є книга в цілому, з комп'ютерною літературою це не завжди так. Наприклад, при навчанні користувача можуть цікавити певні конкретні питання, і необхідно знайти релевантну літературу саме під таке питання.

З іншого боку потрібно розуміти і те, як певне питання розкрито в книзі, оцінювання зазвичай відбувається книги в цілому. Тобто щонайменше загальна оцінка стосується всієї книги, а в тексті вже можуть бути вказані певні деталі, проте акцент може бути і не зробленим або стосуватися різних питань. Тож на даний момент існує потреба в програмному забезпеченні, яке б дозволило виконувати пошук, отримання інформації про комп'ютерну літературу, надаючи можливості щодо окремих розділів книги, а не тільки книги загалом.

A.1 Підстави для розробки

Завдання на роботу визначено самим виданим завданням від 4 вересня 2023 року, що має входити до складу пояснювальної записки, та наказом № 437 від 14 листопада 2023 р. за Національним університетом «Запорізька політехніка», де затверджено тему роботи: «Дослідження та програмна реалізація методів пошуку комп'ютерної літератури».

A.2 Призначення розробки

Програма призначена для пошуку комп'ютерної літератури, включаючи як книги з комп'ютерних наук загалом, так і окремі розділи цих книг, надаючи релевантну інформацію щодо тих проблем, які цікавлять користувача.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Для того щоб досягти сформульовано призначення розробки в підсумку, потрібно виконати такі функціональні вимоги:

- пошук комп'ютерної літератури;
- авторизація;
- отримання інформації про книгу з комп'ютерних наук;
- отримання інформації про розділ книги з комп'ютерних наук;
- додавання книги з комп'ютерних наук;
- додавання розділу книги з комп'ютерних наук;
- редагування даних книги з комп'ютерних наук;
- редагування розділу книги з комп'ютерних наук;
- демонстрування переліку книг за видавництвом;
- демонстрування переліку книг за темою;

- демонстрування переліку книг за підтемою;
- демонстрування переліку книг за питанням;
- оцінювання книги з комп'ютерних наук;
- оцінювання викладення підтеми в розділі книги з комп'ютерних наук;
- оцінювання викладення питання в розділі книги з комп'ютерних наук;
- демонстрування історії оцінювань за розділом книги з комп'ютерних наук;
- демонстрування історії оцінювань за книгою з комп'ютерних наук;
- демонстрування історії власних оцінювань;
- публікування аналізу книги з комп'ютерних наук;
- публікування аналізу підтеми в розділі книги з комп'ютерних наук;
- публікування аналізу питання в розділі книги з комп'ютерних наук;
- редагування опублікованого аналізу комп'ютерної літератури;
- демонстрування історії аналізів за розділом книги з комп'ютерних наук;
- демонстрування історії аналізів за книгою з комп'ютерних наук;
- демонстрування історії власних аналізів комп'ютерної літератури;
- висловлення підтримки аналізу комп'ютерної літератури;
- коментування аналізу комп'ютерної літератури;
- демонстрування власних дискусій користувача;
- отримання інформації про автора;
- визначення інформації про автора;
- редагування інформації про автора;
- пошук авторів;
- підписування на книги автора;
- підписування на книги за темою;
- підписування на книги за підтемою;
- підписування на книги за питанням;

- скасування підписування на книги автора;
- скасування підписування на книги за темою;
- скасування підписування на книги за підтемою;
- скасування підписування на книги за питанням;
- демонстрування переліку книг за підпискою;
- отримання інформації про видавця;
- визначення інформації про видавця;
- редагування інформації про видавця;
- пошук видавців;
- демонстрування комп'ютерної літератури видавця;
- демонстрування серій книг видавця;
- демонстрування переліку книг в серії;
- створення переліку комп'ютерної літератури;
- додавання книги до переліку комп'ютерної літератури;
- вилучення книги до переліку комп'ютерної літератури;
- додавання підрозділу книги до переліку комп'ютерної літератури;
- вилучення підрозділу книги до переліку комп'ютерної літератури;
- демонстрування власних переліків комп'ютерної літератури користувача;
- демонстрування комп'ютерної літератури з переліку;
- редагування облікових даних.

A.3.2 Умови експлуатації та вимоги

Умови експлуатації програми визначаються тими засобами, які мають бути обрані під час проектування програми, включаючи зокрема підтримку обраної мови програмування, організацію роботи серверу визначеним чином, забезпечення на ньому роботи системи керування базами даних, яку має бути обрано, а також будь-яких додаткових засобів, які обрані та використані під час розробки програми та потребують інсталяції.

A.4 Порядок контролю та приймання

Контроль виконання заявлених складових роботи має відбуватися згідно з завданням, виданим на роботу, що є однією з підстав для розробки. Виконання роботи має бути завершено протягом 12 тижнів з проведенням захисту перед екзаменаційною комісією.

ДОДАТОК Б
Текст програми

```

from compbooker.searchmethod import searchbooks
from django.shortcuts import render
from .models import Publisher
from .models import Computerbook
from .models import Searcher
from .models import Computerbookauthor
from .models import Searcherbooklistbook
from .models import Searcherscore
from .models import Searcheranalysis

def publishercomputerbooks(request, publisherid, booksort =
0):
    try:
        computerbookpublisher = Publisher.objects.get(id =
publisherid)
    except Publisher.DoesNotExist:
        return render(request, "compbooker/publisher.html",
{})

    publishercomputerbooks =
Computerbook.objects.filter(publisher = computerbookpublisher)
    if booksort == 0:
        publishercomputerbooks =
publishercomputerbooks.order_by("-analysisrating")
    elif booksort == 1:
        publishercomputerbooks =
publishercomputerbooks.order_by("-scorerating")
    elif booksort == 2:
        publishercomputerbooks =
publishercomputerbooks.order_by("-publishdate")
    elif booksort == 3:
        publishercomputerbooks =
publishercomputerbooks.order_by("publishdate")
    if request.user.is_authenticated:
        searcher = Searcher.objects.filter(username =
request.user.username)
        for compbook in publishercomputerbooks:
            compbooklists =
Searcherbooklistbook.objects.filter(computerbook = compbook)
            compbooklists =
Searcherbooklistbook.objects.filter(searcherbooklist__searcher =
searcher)
            compbooklists =
compbooklists.select_related("searcherbooklist")
            compbooklists =
compbooklists.order_by("listname")
            compbook.lists = compbooklists
            authors =
Computerbookauthor.objects.filter(computerbook = compbook)
            authors = authors.select_related("author")
            authors =
authors.prefetch_related("author__followers")

```

```

        if searcher in authors.author.followers:
            compbook.authorfollowed = True
        else:
            compbook.authorfollowed = False
        scores =
Searcherscore.objects.filter(content_object = compbook, searcher
= searcher)
        scores = scores.order_by("-scoretime")
        scores = scores.values()
        compbook.searcherscore = scores[0]
        scores =
Searcheranalysis.objects.filter(content_object = compbook,
searcher = searcher)
        scores = scores.order_by("-analysistime")
        scores = scores.values()
        compbook.searcheranalysisisscore = scores[0]
        return render(request, "compbooker/publisher.html", {
"books": publishercomputerbooks, "publisher":
computerbookpublisher })

def topiccomputerbooks(request, topic, booksort = 4):
    try:
        computerbooktopic = Topic.objects.get(title =
topic)
        computerbooktopic =
computerbooktopic.prefetch_related("followers")
        except Topic.DoesNotExist:
            return render(request, "compbooker/topic.html", {})
        topiccomputerbooks = Computerbook.objects.filter(topics
= computerbooktopic)
        if booksort == 0:
            topiccomputerbooks = topiccomputerbooks.order_by("-
analysisrating")
        elif booksort == 1:
            topiccomputerbooks = topiccomputerbooks.order_by("-
scorerating")
        elif booksort == 2:
            topiccomputerbooks = topiccomputerbooks.order_by("-
publishdate")
        elif booksort == 3:
            topiccomputerbooks =
topiccomputerbooks.order_by("publishdate")
            computerbooktopic.followed = False
            if request.user.is_authenticated:
                searcher = Searcher.objects.filter(username =
request.user.username)
                if booksort == 4:
                    analysedbooks =
Searcheranalysis.objects.filter(searcher = searcher,
content_object__topics = computerbooktopic)

```

```

        publishercomputerbooks =
searchbooks(topiccomputerbooks, mode = 0, analysedbooks)
        if searcher in computerbooktopic.followers:
            computerbooktopic.followed = True
        for compbook in topiccomputerbooks:
            compbooklists =
Searcherbooklistbook.objects.filter(computerbook = compbook)
            compbooklists =
compbooklists.filter(searcherbooklist__searcher = searcher)
            compbooklists =
compbooklists.select_related("searcherbooklist")
            compbooklists =
compbooklists.order_by("listname")
            compbook.lists = compbooklists
            authors =
Computerbookauthor.objects.filter(computerbook = compbook)
            authors = authors.select_related("author")
            authors =
authors.prefetch_related("author__followers")
            if searcher in authors.author.followers:
                compbook.authorfollowed = True
            else:
                compbook.authorfollowed = False
            scores =
Searcherscore.objects.filter(content_object = compbook, searcher
= searcher)
            scores = scores.order_by("-scoretime")
            scores = scores.values()
            compbook.searcherscore = scores[0]
            scores =
Searcheranalysis.objects.filter(content_object = compbook,
searcher = searcher)
            scores = scores.order_by("-analysistime")
            scores = scores.values()
            compbook.searcheranalysisisscore = scores[0]
        elif booksort == 4:
            publishercomputerbooks =
searchbooks(publishercomputerbooks, mode = 0)
            return render(request, "compbooker/topic.html", {
"books": topiccomputerbooks, "topic": computerbooktopic })

from .models import Subtopic
from .models import Computerbookpart
from .models import Computerbookpartsubtopic

def subtopiccomputerbooks(request, subtopic, booksort = 0):
    try:
        computerbooksubtopic = Subtopic.objects.get(title
= subtopic)
        computerbooksubtopic =
computerbooksubtopic.prefetch_related("followers")

```

```

        except Subtopic.DoesNotExist:
            return render(request, "compbooker/subtopic.html",
                {})

        subtopiccomputerbooks =
Computerbookpartsubtopic.objects.filter(subtopic =
computerbooksubtopic)
        subtopiccomputerbooks =
subtopiccomputerbooks.select_related("computerbookpart")
        subtopiccomputerbooks =
subtopiccomputerbooks.select_related("computerbookpart__computer
book")

        if booksort == 0:
            subtopiccomputerbooks =
subtopiccomputerbooks.order_by("-analysisrating")
        elif booksort == 1:
            subtopiccomputerbooks =
subtopiccomputerbooks.order_by("-scorerating")
        elif booksort == 2:
            subtopiccomputerbooks =
subtopiccomputerbooks.order_by("-
computerbookpart__computerbook__publishdate")
        elif booksort == 3:
            subtopiccomputerbooks =
subtopiccomputerbooks.order_by("computerbookpart__computerbook__
publishdate")
            computerbooksubtopic.followed = False
            if request.user.is_authenticated:
                searcher = Searcher.objects.filter(username =
request.user.username)
                if booksort == 4:
                    compparts =
Computerbookpartsubtopic.objects.filter(subtopic =
computerbooksubtopic)
                    compparts =
compparts.select_related("computerbookpart")
                    compparts =
compparts.values_list("computerbookpart")
                    analysedbooks =
Searcheranalysis.objects.filter(searcher = searcher,
content_object__in = compparts)
                    subtopiccomputerbooks =
searchbooks(subtopiccomputerbooks, mode = 0, analysedbooks)
                    if searcher in computerbooksubtopic.followers:
                        computerbooksubtopic.followed = True
                    for compbookpart in subtopiccomputerbooks:
                        compbook = compbookpart.computerbook
                        compbooklists =
Searcherbooklistbook.objects.filter(computerbook = compbook)
                        compbooklists =
compbooklists.filter(searcherbooklist__searcher = searcher)

```

```

        compbooklists =
compbooklists.select_related("searcherbooklist")
        compbooklists =
compbooklists.order_by("listname")
        compbook.lists = compbooklists
        authors =
Computerbookauthor.objects.filter(computerbook = compbook)
        authors = authors.select_related("author")
        authors =
authors.prefetch_related("author__followers")
        if searcher in authors.author.followers:
            compbook.authorfollowed = True
        else:
            compbook.authorfollowed = False
        scores =
Searcherscore.objects.filter(content_object = compbook, searcher
= searcher)
        scores = scores.order_by("-scoretime")
        scores = scores.values()
        compbook.searcherscore = scores[0]
        scores =
Searcheranalysis.objects.filter(content_object = compbook,
searcher = searcher)
        scores = scores.order_by("-analysistime")
        scores = scores.values()
        compbook.searcheranalysisisscore = scores[0]
    elif booksort == 4:
        subtopiccomputerbooks =
searchbooks(subtopiccomputerbooks, mode = 1)
        return render(request, "compbooker/subtopic.html", {
"books": subtopiccomputerbooks, "subtopic":
computerbooksubtopic})

from .models import Computerbookpartquestion

def questioncomputerbooks(request, question, booksort = 0):
    try:
        computerbookquestion = Question.objects.get(title
= question)
        computerbookquestion =
computerbookquestion.select_related("topic")
        computerbookquestion =
computerbookquestion.prefetch_related("followers")
        except Subtopic Question DoesNotExist:
            return render(request, "compbooker/question.html",
{})
        questioncomputerbooks =
Computerbookpartquestion.objects.filter(question =
computerbookquestion)
        questioncomputerbooks =
questioncomputerbooks.select_related("computerbookpart")

```

```

        questioncomputerbooks =
questioncomputerbooks.select_related("computerbookpart__computer
book")
        if booksort == 0:
            questioncomputerbooks =
questioncomputerbooks.order_by("-analysisrating")
        elif booksort == 1:
            questioncomputerbooks =
questioncomputerbooks.order_by("-scorerating")
        elif booksort == 2:
            questioncomputerbooks =
questioncomputerbooks.order_by("-
computerbookpart__computerbook__publishdate")
        elif booksort == 3:
            questioncomputerbooks =
questioncomputerbooks.order_by("computerbookpart__computerbook__
publishdate")
        questioncomputerbooks.followed = False
        if request.user.is_authenticated:
            searcher = Searcher.objects.filter(username =
request.user.username)
            if booksort == 4:
                compparts =
questioncomputerbooks.objects.filter(question =
computerbookquestion)
                compparts =
compparts.select_related("computerbookpart")
                compparts =
compparts.values_list("computerbookpart")
                analysedbooks =
Searcheranalysis.objects.filter(searcher = searcher,
content_object__in = compparts)
                questioncomputerbooks =
searchbooks(questioncomputerbooks, mode = 0, analysedbooks)
                if searcher in computerbookquestion.followers:
                    computerbookquestion.followed = True
                for compbookpart in questioncomputerbooks:
                    compbook = compbookpart.computerbook
                    compbooklists =
Searcherbooklistbook.objects.filter(computerbook = compbook)
                    compbooklists =
compbooklists.filter(searcherbooklist__searcher = searcher)
                    compbooklists =
compbooklists.select_related("searcherbooklist")
                    compbooklists =
compbooklists.order_by("listname")
                    compbook.lists = compbooklists
                    authors =
Computerbookauthor.objects.filter(computerbook = compbook)
                    authors = authors.select_related("author")

```

```

        authors =
authors.prefetch_related("author__followers")
        if searcher in authors.author.followers:
            compbook.authorfollowed = True
        else:
            compbook.authorfollowed = False
        scores =
Searcherscore.objects.filter(content_object = compbook, searcher
= searcher)
        scores = scores.order_by("-scoretime")
        scores = scores.values()
        compbook.searcherscore = scores[0]
        scores =
Searcheranalysis.objects.filter(content_object = compbook,
searcher = searcher)
        scores = scores.order_by("-analysistime")
        scores = scores.values()
        compbook.searcheranalysisisscore = scores[0]
    elif booksort == 4:
        questioncomputerbooks =
searchbooks(questioncomputerbooks, mode = 1)
        return render(request, "compbooker/question.html", {
"books": questioncomputerbooks, "question":
computerbookquestion})

from .models import Publisherseries

def seriescomputerbooks(request, seriesid, booksort = 0):
    try:
        publisherseries = Publisherseries.objects.get(id =
seriesid)
    except Publisherseries.DoesNotExist:
        return render(request,
"compbooker/publisherserie.html", {})
        seriescomputerbooks =
Computerbook.objects.filter(publisherseries = publisherseries)
        if booksort == 0:
            seriescomputerbooks =
seriescomputerbooks.order_by("-analysisrating")
        elif booksort == 1:
            seriescomputerbooks =
seriescomputerbooks.order_by("-scorerating")
        elif booksort == 2:
            seriescomputerbooks =
seriescomputerbooks.order_by("-publishdate")
        elif booksort == 3:
            seriescomputerbooks =
seriescomputerbooks.order_by("publishdate")
        if request.user.is_authenticated:
            searcher = Searcher.objects.filter(username =
request.user.username)

```

```

        for compbook in seriescomputerbooks:
            compbooklists =
Searcherbooklistbook.objects.filter(computerbook = compbook)
            compbooklists =
compbooklists.filter(searcherbooklist__searcher = searcher)
            compbooklists =
compbooklists.select_related("searcherbooklist")
            compbooklists =
compbooklists.order_by("listname")
            compbook.lists = compbooklists
            authors =
Computerbookauthor.objects.filter(computerbook = compbook)
            authors = authors.select_related("author")
            authors =
authors.prefetch_related("author__followers")
            if searcher in authors.author.followers:
                compbook.authorfollowed = True
            else:
                compbook.authorfollowed = False
            scores =
Searcherscore.objects.filter(content_object = compbook, searcher
= searcher)
            scores = scores.order_by("-scoretime")
            scores = scores.values()
            compbook.searcherscore = scores[0]
            scores =
Searcheranalysis.objects.filter(content_object = compbook,
searcher = searcher)
            scores = scores.order_by("-analysistime")
            scores = scores.values()
            compbook.searcheranalysisisscore = scores[0]
        return render(request,
"compbooker/publisherserie.html", { "books":
seriescomputerbooks, "series": publisherseries })

from .models import Searcherbooklistbookpart

def publisherseries(request, publisherid):
    try:
        computerbookpublisher = Publisher.objects.get(id =
publisherid)
    except Publisher.DoesNotExist:
        return render(request,
"compbooker/publisherseries.html", {})
    publisherseries =
Publisherseries.objects.filter(publisher =
computerbookpublisher)
    publisherseries = publisherseries.order_by("-id")
    return render(request,
"compbooker/publisherseries.html", { "series": publisherseries,
"publisher": computerbookpublisher })

```

```

def searcherbooklist(request, booklistid, booksort = 0):
    if request.user.is_authenticated:
        searcher = Searcher.objects.filter(username =
request.user.username)
        try:
            boobooklist = Searcherbooklist.objects.get(id =
booklistid, searcher = searcher)
            except Searcherbooklist.DoesNotExist:
                return render(request,
"compbooker/booklist.html", {})
            listbooks =
Searcherbooklistbook.objects.filter(searcherbooklist =
boobooklist)
            listbookparts =
Searcherbooklistbookpart.objects.filter(searcherbooklist =
boobooklist)
            listbookparts =
listbookparts.select_related("computerbook")
            for listbookpart in listbookparts:
                listbookpart.publishdate =
listbookpart.computerbook.publishdate
                listbooksadd = listbooks.values()
                listbooksadd = listbooksadd +
listbookparts.values()
                if booksort == 0:
                    listbooksadd.sort(key = lambda book:
book.analysisrating, reverse = True)
                elif booksort == 1:
                    listbooksadd.sort(key = lambda book:
book.scorerating, reverse = True)
                elif booksort == 2:
                    listbooksadd.sort(key = lambda book:
book.publishdate, reverse = True)
                elif booksort == 3:
                    listbooksadd.sort(key = lambda book:
book.publishdate)
                for compbook in listbooksadd:
                    if not hasattr(compbook, 'startpage'):
                        compbooklists =
Searcherbooklistbook.objects.filter(computerbook = compbook)
                        compbooklists =
compbooklists.filter(searcherbooklist__searcher = searcher)
                        compbooklists =
compbooklists.select_related("searcherbooklist")
                        compbooklists =
compbooklists.order_by("listname")
                        compbook.lists = compbooklists
                    else:
                        compbooklists =
Searcherbooklistbookpart.objects.filter(computerbookpart=
compbook)

```

```

        compbooklists =
compbooklists.filter(searcherbooklist__searcher = searcher)
        compbooklists =
compbooklists.select_related("searcherbooklist")
        compbooklists =
compbooklists.order_by("listname")
        compbook.lists = compbooklists
        if not hasattr(compbook, 'startpage'):
            authors =
Computerbookauthor.objects.filter(computerbook = compbook)
            authors = authors.select_related("author")
            authors =
authors.prefetch_related("author__followers")
        else:
            authors =
Computerbookauthorpart.objects.filter(computerbookpart=
compbook)
            authors = authors.select_related("author")
            authors =
authors.prefetch_related("author__followers")
            if searcher in authors.author.followers:
                compbook.authorfollowed = True
            else:
                compbook.authorfollowed = False
            scores =
Searcherscore.objects.filter(content_object = compbook, searcher
= searcher)
            scores = scores.order_by("-scoretime")
            scores = scores.values()
            compbook.searcherscore = scores[0]
            scores =
Searcheranalysis.objects.filter(content_object = compbook,
searcher = searcher)
            scores = scores.order_by("-analysistime")
            scores = scores.values()
            compbook.searcheranalysisisscore = scores[0]
        return render(request,
"compbooker/searcherbooklist.html", { "books": listbooksadd })

from .models import Authorfollow
from .models import Topicfollow
from .models import Subtopicfollow
from .models import Questionfollow

def followbooks(request, booksort = 0):
    if request.user.is_authenticated:
        searcher = Searcher.objects.filter(username =
request.user.username)
        authorfollows = Authorfollow.objects.get(searcher =
searcher)

```

```

        authorfollows =
authorfollows.select_related("author")
        authorfollows = authorfollows.values_list("author")
        books_on_author =
Computerbookauthor.objects.filter(author__in = authorfollows)
        books_on_author =
books_on_author.select_related("computerbook")
        books_on_author_id =
books_on_author.values_list("computerbook__id")
        bookparts_on_author =
Computerbookauthorpart.objects.filter(computerbookauthor__author
__in = authorfollows)
        bookparts_on_author =
bookparts_on_author.select_related("computerbookauthor")
        bookparts_on_author =
bookparts_on_author.select_related("computerbookpart")
        bookparts_on_author =
bookparts_on_author.exclude(computerbookpart__id__in =
books_on_author_id)
        bookparts_on_author =
bookparts_on_author.select_related("computerbook")
        for listbookpart in bookparts_on_author:
            listbookpart.publishdate =
listbookpart.computerbook.publishdate
        topicfollows = Topicfollow.objects.get(searcher =
searcher)
        topicfollows = topicfollows.select_related("topic")
        topicfollows = topicfollows.values_list("topic")
        books_on_topic =
Computerbook.objects.filter(topics__in = topicfollows)
        subtopicfollows =
Subtopicfollow.objects.get(searcher = searcher)
        subtopicfollows =
subtopicfollows.select_related("subtopic")
        subtopicfollows =
subtopicfollows.values_list("subtopic")
        bookparts_on_subtopic =
Computerbookpartsubtopic.objects.filter(subtopic__in =
subtopicfollows)
        bookparts_on_subtopic =
bookparts_on_subtopic.select_related("computerbook")
        for listbookpart in bookparts_on_subtopic:
            listbookpart.publishdate =
listbookpart.computerbook.publishdate
        questionfollows =
Questionfollow.objects.get(searcher = searcher)
        questionfollows =
questionfollows.select_related("question")
        questionfollows =
questionfollows.values_list("question")

```

```

        bookparts_on_question =
Computerbookpartquestion.objects.filter(question__in =
subtopicfollows)
        bookparts_on_question =
bookparts_on_question.select_related("computerbook")
        for listbookpart in bookparts_on_question:
            listbookpart.publishdate =
listbookpart.computerbook.publishdate
            listbooks =
Searcherbooklistbook.objects.filter(searcherbooklist =
booklist)
            listbooksadd = bookparts_on_author.values()
            listbooksadd = listbooksadd +
bookparts_on_question.values()
            listbooksadd = listbooksadd +
bookparts_on_subtopic.values()
            listbooksadd = listbooksadd + listbooks.values()
            if booksort == 0:
                listbooksadd.sort(key = lambda book:
book.analysisrating, reverse = True)
            elif booksort == 1:
                listbooksadd.sort(key = lambda book:
book.scorerating, reverse = True)
            elif booksort == 2:
                listbooksadd.sort(key = lambda book:
book.publishdate, reverse = True)
            elif booksort == 3:
                listbooksadd.sort(key = lambda book:
book.publishdate)
            for compbook in listbooksadd:
                if not hasattr(compbook, 'startpage'):
                    compbooklists =
Searcherbooklistbook.objects.filter(computerbook = compbook)
                    compbooklists =
compbooklists.filter(searcherbooklist__searcher = searcher)
                    compbooklists =
compbooklists.select_related("searcherbooklist")
                    compbooklists =
compbooklists.order_by("listname")
                    compbook.lists = compbooklists
                else:
                    compbooklists =
Searcherbooklistbookpart.objects.filter(computerbookpart=
compbook)
                    compbooklists =
compbooklists.filter(searcherbooklist__searcher = searcher)
                    compbooklists =
compbooklists.select_related("searcherbooklist")
                    compbooklists =
compbooklists.order_by("listname")
                    compbook.lists = compbooklists

```

```

        if not hasattr(compbook, 'startpage'):
            authors =
Computerbookauthor.objects.filter(computerbook = compbook)
            authors = authors.select_related("author")
            authors =
authors.prefetch_related("author__followers")
        else:
            authors =
Computerbookauthorpart.objects.filter(computerbookpart=
compbook)
            authors = authors.select_related("author")
            authors =
authors.prefetch_related("author__followers")
            if searcher in authors.author.followers:
                compbook.authorfollowed = True
            else:
                compbook.authorfollowed = False
            scores =
Searcherscore.objects.filter(content_object = compbook, searcher
= searcher)
            scores = scores.order_by("-scoretime")
            scores = scores.values()
            compbook.searcherscore = scores[0]
            scores =
Searcheranalysis.objects.filter(content_object = compbook,
searcher = searcher)
            scores = scores.order_by("-analysistime")
            scores = scores.values()
            compbook.searcheranalysisisscore = scores[0]
        return render(request, "compbooker/followbooks.html", {
"books": listbooksadd})

def computerbookabout(request, compbook):
    try:
        computerbook =
Computerbook.objects.select_related("publisherseries")
        computerbook =
computerbook.select_related("publisher")
        computerbook =
computerbook.select_related("original")
        computerbook = computerbook.get(slug = compbook)
    except Computerbook.DoesNotExist:
        return render(request, "compbooker/compbook.html",
{})

    computerbookparts =
Computerbookpart.objects.filter(computerbook = computerbook)
    computerbookparts =
computerbookparts.order_by("partnumber")
    authors =
Computerbookauthor.objects.filter(computerbook = computerbook)
    aauthors = authors.select_related("author")

```

```

        topics = Computerbooktopic.objects.filter(computerbook
= computerbook)
        topics = topics.select_related("topic")
        subtopics =
Computerbookpartsubtopic.objects.filter(computerbookpart__comput
erbook = computerbook)
        subtopics = subtopics.select_related("subtopic")
        questions =
Computerbooktopic.objects.filter(computerbookpart__computerbook
= computerbook)
        questions = questions.select_related("question")
        allscores = Searcherscore.objects.filter(content_object
= computerbook)
        allscores = allscores.select_related("searcher")
        allscores = allscores.order_by("-scoretime")
        allscores = allscores.values()[:5]
        allanalyses =
Searcheranalysis.objects.filter(content_object = computerbook)
        allanalyses = allanalyses.select_related("searcher")
        allanalyses = allanalyses.order_by("-analysistime")
        allanalyses = allanalyses.values()[:5]
        if request.user.is_authenticated:
            searcher = Searcher.objects.filter(username =
request.user.username)
            for compbook in computerbookparts:
                compbooklists =
Searcherbooklistbookpart.objects.filter(computerbookpart =
compbook)
                compbooklists =
compbooklists.filter(searcherbooklist__searcher = searcher)
                compbooklists =
compbooklists.select_related("searcherbooklist")
                compbooklists =
compbooklists.order_by("listname")
                compbook.lists = compbooklists
                authors =
Computerbookauthorpart.objects.filter(computerbookpart=
compbook)
                authors = authors.select_related("author")
                authors =
authors.prefetch_related("author__followers")
                if searcher in authors.author.followers:
                    compbook.authorfollowed = True
                else:
                    compbook.authorfollowed = False
                scores =
Searcherscore.objects.filter(content_object = compbook, searcher
= searcher)
                scores = scores.order_by("-scoretime")
                scores = scores.values()
                compbook.searcherscore = scores[0]

```

```

        scores =
Searcheranalysis.objects.filter(content_object = compbook,
searcher = searcher)
        scores = scores.order_by("-analysistime")
        scores = scores.values()
        compbook.searcheranalysiscore = scores[0]
    compbooklists =
Searcherbooklistbook.objects.filter(computerbook = computerbook)
    compbooklists =
compbooklists.filter(searcherbooklist__searcher = searcher)
    compbooklists =
compbooklists.select_related("searcherbooklist")
    compbooklists = compbooklists.order_by("listname")
    computerbook.lists = compbooklists
    authors =
Computerbookauthor.objects.filter(computerbook = computerbook)
    authors = authors.select_related("author")
    authors =
authors.prefetch_related("author__followers")
    if searcher in authors.author.followers:
        computerbook.authorfollowed = True
    else:
        computerbook.authorfollowed = False
    scores =
Searcherscore.objects.filter(content_object = computerbook,
searcher = searcher)
    scores = scores.order_by("-scoretime")
    scores = scores.values()
    computerbook.searcherscore = scores[0]
    scores =
Searcheranalysis.objects.filter(content_object = computerbook,
searcher = searcher)
    scores = scores.order_by("-analysistime")
    scores = scores.values()
    computerbook.searcheranalysiscore = scores[0]
    for tp in topics:
        folls = Topicfollow.objects.filter(searcher =
searcher)

        folls = folls.select_related("topic")
        tps = folls.values_list("topic")
        if tp.topic in tps:
            tp.followed = True
        else:
            tp.followed = False
    for tp in subtopics:
        folls = Subtopicfollow.objects.filter(searcher
= searcher)

        folls = folls.select_related("subtopic")
        tps = folls.values_list("subtopic")
        if tp.topic in tps:
            tp.followed = True

```

```

        else:
            tp.followed = False
    for tp in questions:
        folls = Questionfollow.objects.filter(searcher
= searcher)

        folls = folls.select_related("question")
        tps = folls.values_list("question")
        if tp.topic in tps:
            tp.followed = True
        else:
            tp.followed = False
    return render(request, "compbooker/compbook.html", {
"book": computerbook, "parts": computerbookparts, "authors":
aauthors, "topics": topics, "subtopics": subtopics,
"questions": questions, "analyses": allanalyses, "scores":
allscores })

from .models import Searcheranalysissupport

def computerbookpartabout(request, compbookpart):
    try:
        computerbookpart =
Computerbookpart.objects.select_related("computerbook__publisher
series")
        computerbookpart =
computerbookpart.select_related("computerbook__publisher")
        computerbookpart =
computerbookpart.select_related("computerbook__original")
        computerbookpart =
computerbookpart.select_related("computerbook")
        computerbookpart = computerbookpart.get(slug =
compbookpart)
    except Computerbookpart.DoesNotExist:
        return render(request,
"compbooker/compbookpart.html", {})
    authors =
Computerbookauthorpart.objects.filter(computerbookpart =
computerbookpart)
    aauthors =
authors.select_related("computerbookauthor__author")
    topics = Computerbooktopic.objects.filter(computerbook
= computerbookpart.computerbook)
    topics = topics.select_related("topic")
    subtopics =
Computerbookpartsubtopic.objects.filter(computerbookpart =
computerbookpart)
    subtopics = subtopics.select_related("subtopic")
    questions =
Computerbooktopic.objects.filter(computerbookpart =
computerbookpart)
    questions = questions.select_related("question")

```

```

        allscores = Searcherscore.objects.filter(content_object
= computerbookpart)
        allscores = allscores.select_related("searcher")
        allscores = allscores.order_by("-scoretime")
        allscores = allscores.values()[:5]
        allanalyses =
Searcheranalysis.objects.filter(content_object =
computerbookpart)
        allanalyses = allanalyses.select_related("searcher")
        allanalyses = allanalyses.order_by("-analysistime")
        allanalyses = allanalyses.values()[:5]
        if request.user.is_authenticated:
            searcher = Searcher.objects.filter(username =
request.user.username)
            compbooklists =
Searcherbooklistbookpart.objects.filter(computerbookpart =
computerbookpart)
            compbooklists =
compbooklists.filter(searcherbooklist__searcher = searcher)
            compbooklists =
compbooklists.select_related("searcherbooklist")
            compbooklists = compbooklists.order_by("listname")
            computerbookpart.lists = compbooklists
            authors =
Computerbookauthorpart.objects.filter(computerbookpart=
computerbookpart)
            authors = authors.select_related("author")
            authors =
authors.prefetch_related("author__followers")
            if searcher in authors.author.followers:
                computerbookpart.authorfollowed = True
            else:
                computerbookpart.authorfollowed = False
            scores =
Searcherscore.objects.filter(content_object = computerbookpart,
searcher = searcher)
            scores = scores.order_by("-scoretime")
            scores = scores.values()
            computerbookpart.searcherscore = scores[0]
            scores =
Searcheranalysis.objects.filter(content_object =
computerbookpart, searcher = searcher)
            scores = scores.order_by("-analysistime")
            scores = scores.values()
            computerbookpart.searcheranalysisscore = scores[0]
            compbooklists =
Searcherbooklistbook.objects.filter(computerbook =
computerbookpart.computerbook)
            compbooklists =
compbooklists.filter(searcherbooklist__searcher = searcher)

```

```

        compbooklists =
compbooklists.select_related("searcherbooklist")
        compbooklists = compbooklists.order_by("listname")
        computerbookpart.booklists = compbooklists
        scores =
Searcherscore.objects.filter(content_object =
computerbookpart.computerbook, searcher = searcher)
        scores = scores.order_by("-scoretime")
        scores = scores.values()
        computerbookpart.searcherscore = scores[0]
        scores =
Searcheranalysis.objects.filter(content_object =
computerbookpart.computerbook, searcher = searcher)
        scores = scores.order_by("-analysistime")
        scores = scores.values()
        computerbookpart.searcheranalysisisscore = scores[0]
for tp in topics:
    folls = Topicfollow.objects.filter(searcher =
searcher)

    folls = folls.select_related("topic")
    tps = folls.values_list("topic")
    if tp.topic in tps:
        tp.followed = True
    else:
        tp.followed = False
for tp in subtopics:
    folls = Subtopicfollow.objects.filter(searcher
= searcher)

    folls = folls.select_related("subtopic")
    tps = folls.values_list("subtopic")
    if tp.topic in tps:
        tp.followed = True
    else:
        tp.followed = False
for tp in questions:
    folls = Questionfollow.objects.filter(searcher
= searcher)

    folls = folls.select_related("question")
    tps = folls.values_list("question")
    if tp.topic in tps:
        tp.followed = True
    else:
        tp.followed = False
for analysis in allanalyses:
    supports =
Searcheranalysisissupport.objects.filter(searcheranalysis =
analysis).count()

    analysis.supports = supports
    comments =
Searcheranalysiscomment.objects.filter(searcheranalysis =
analysis)

```

```

        comments = comments.select_related("searcher")
        comments = comments.order_by("commenttime")
        analysis.comments = comments
    return render(request, "compbooker/compbookpart.html",
{ "part": computerbookpart, "parts": computerbookparts,
"authors": aauthors, "topics": topics, "subtopics": subtopics,
"questions": questions, "analyses": allanalyses, "scores":
allscores })

def computerbookpartscores(request, compbookpart):
    try:
        computerbookpart =
Computerbookpart.objects.select_related("computerbook__publisher
series")
        computerbook =
computerbook.select_related("computerbook__publisher")
        computerbook =
computerbook.select_related("computerbook__original")
        computerbook =
computerbook.select_related("computerbook")
        computerbookpart = computerbook.get(slug =
compbookpart)
    except Computerbookpart.DoesNotExist:
        return render(request,
"compbooker/compbookpartscores.html", {})
        allscores = Searcherscore.objects.filter(content_object
= computerbookpart)
        allscores = allscores.select_related("searcher")
        allscores = allscores.order_by("-scoretime")
        return render(request,
"compbooker/compbookpartscores.html", { "bookpart":
computerbookpart, "scores": allscores })

def computerbookscores(request, compbook):
    try:
        computerbook =
Computerbook.objects.select_related("publisherseries")
        computerbook =
computerbook.select_related("publisher")
        computerbook =
computerbook.select_related("original")
        computerbook = computerbook.get(slug = compbook)
    except Computerbook.DoesNotExist:
        return render(request,
"compbooker/compbookscores.html", {})
        allscores = Searcherscore.objects.filter(content_object
= computerbook)
        allscores = allscores.select_related("searcher")
        allscores = allscores.order_by("-scoretime")

```

```

        return render(request,
"compbooker/compbookscores.html", { "book": computerbook,
"scores": allscores })

    def scoreshistory(request):
        if request.user.is_authenticated:
            searcher = Searcher.objects.filter(username =
request.user.username)
            allscores = Searcherscore.objects.filter(searcher =
searcher)
            allscores = allscores.select_related("searcher")
            allscores =
allscores.select_related("content_object")
            allscores = allscores.order_by("-scoretime")
            return render(request,
"compbooker/scoreshistory.html", { "scores": allscores })
        else:
            return render(request,
"compbooker/scoreshistory.html", { "scores": [] })

```

ДОДАТОК В
Слайди презентації

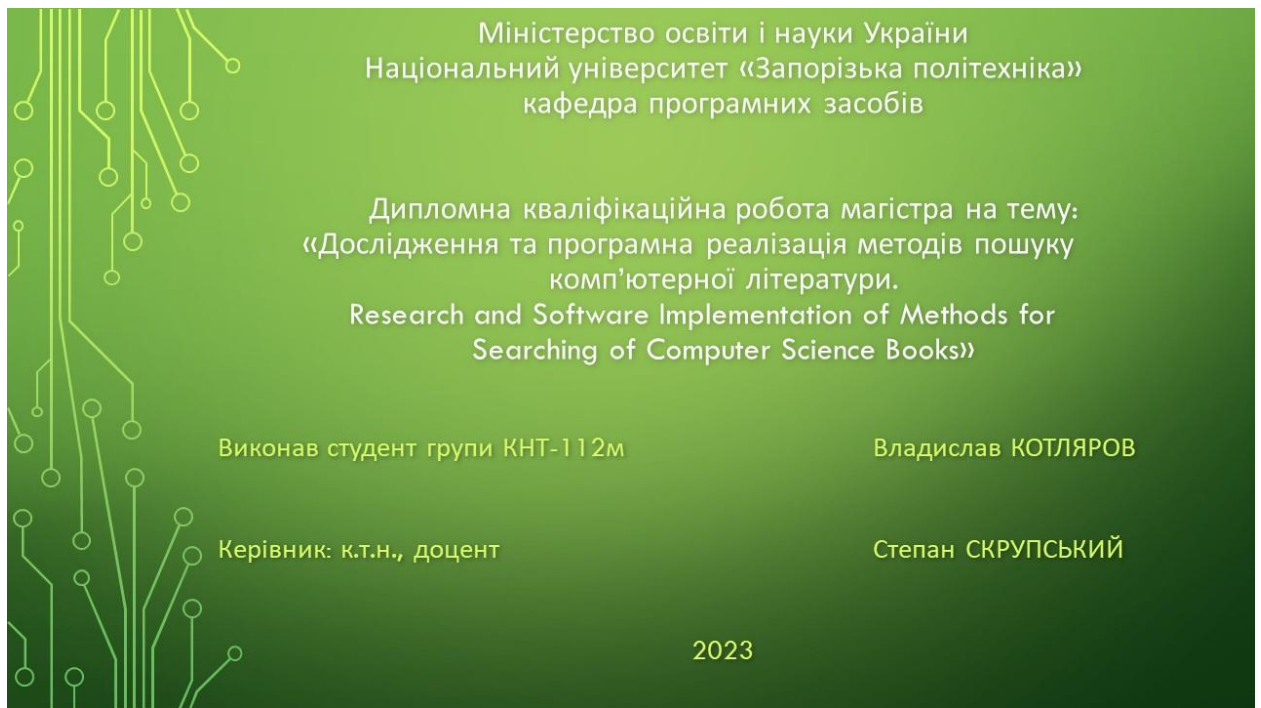


Рисунок В.1 – Титульний слайд

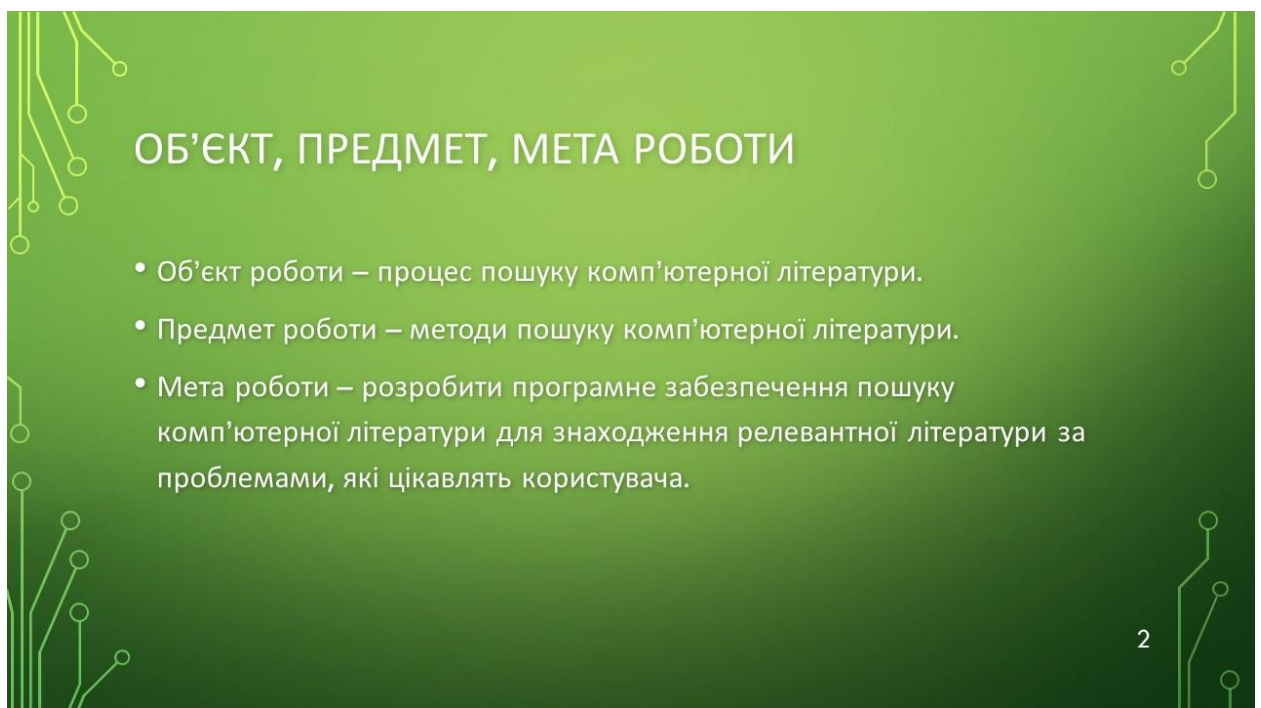


Рисунок В.2 – Об'єкт і предмет, мета роботи

ЗАВДАННЯ РОБОТИ

- аналіз існуючих програм, які виконують пошук комп'ютерної літератури, що було забезпечено в даному розділі;
- дослідження методів і моделей пошуку комп'ютерної літератури;
- проєктування програми пошуку комп'ютерної літератури;
- реалізація програми пошуку комп'ютерної літератури.

3

Рисунок В.3 – Завдання роботи

АНАЛІЗ РОЗРОБЛЮВАНОЇ ТА ІСНУЮЧИХ ПРОГРАМ

Показник	Amazon	Goodreads	CompBooker
Пошук книг з комп'ютерних наук	+	+	+
Пошук за розділами книг з комп'ютерних наук	–	–	+
Аналізування книг користувачами	–	+	+
Аналізування розділів книг стосовно проблем	–	–	+
Об'єкт підписування	Автори	Автори, користувачі	Автори, проблеми
Підбір книг для користувача	За книгою	За книгою, за аналізуваннями	За проблемою

4

Рисунок В.4 – Аналіз розроблюваної та існуючих програм

ДІАГРАМИ ПРЕЦЕДЕНТІВ КОРИСТУВАЧІВ ПРИ РОБОТІ З ПРОГРАМОЮ

Користувач

Адміністратор

визначення інформації про автора

визначення інформації про видавця

додання книги з комп'ютерних наук

додання розділу книги з комп'ютерних наук

редагування даних книги з комп'ютерних наук

редагування розділу книги з комп'ютерних наук

6

Рисунок В.6 – Діаграми прецедентів користувачів при роботі з програмою

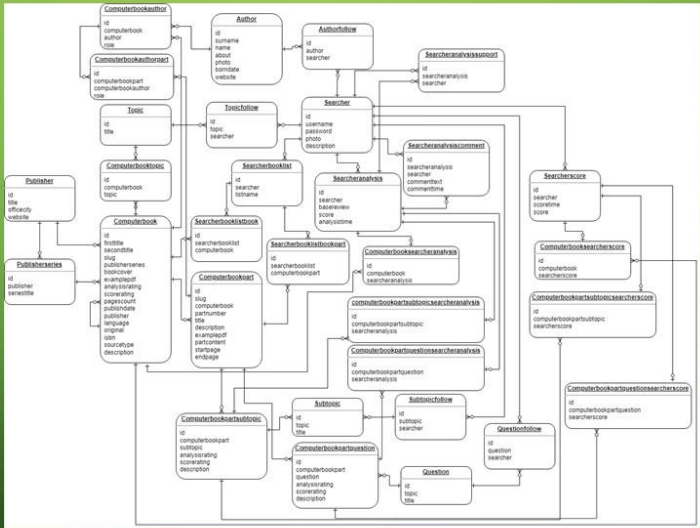
ПОРІВНЯННЯ ЗАСОБІВ ДЛЯ РОЗРОБКИ ПРОГРАМИ ПОШУКУ КОМП'ЮТЕРНОЇ ЛІТЕРАТУРИ

Критерій	Pyramid	Django
Засоби керування для адміністратора	Ні, але можливе розширення, потребує пошуку	Так
Рівень підтримки	Нижчий	Високий
Шаблонізатор сторінок	Потребує вибору і встановлення	Так
Підтримка авторизації	Так	Так

7

Рисунок В.7 – Порівняння засобів для розробки програми пошуку комп'ютерної літератури

СХЕМА БАЗИ ДАНИХ



8

Рисунок В.8 – Схема бази даних

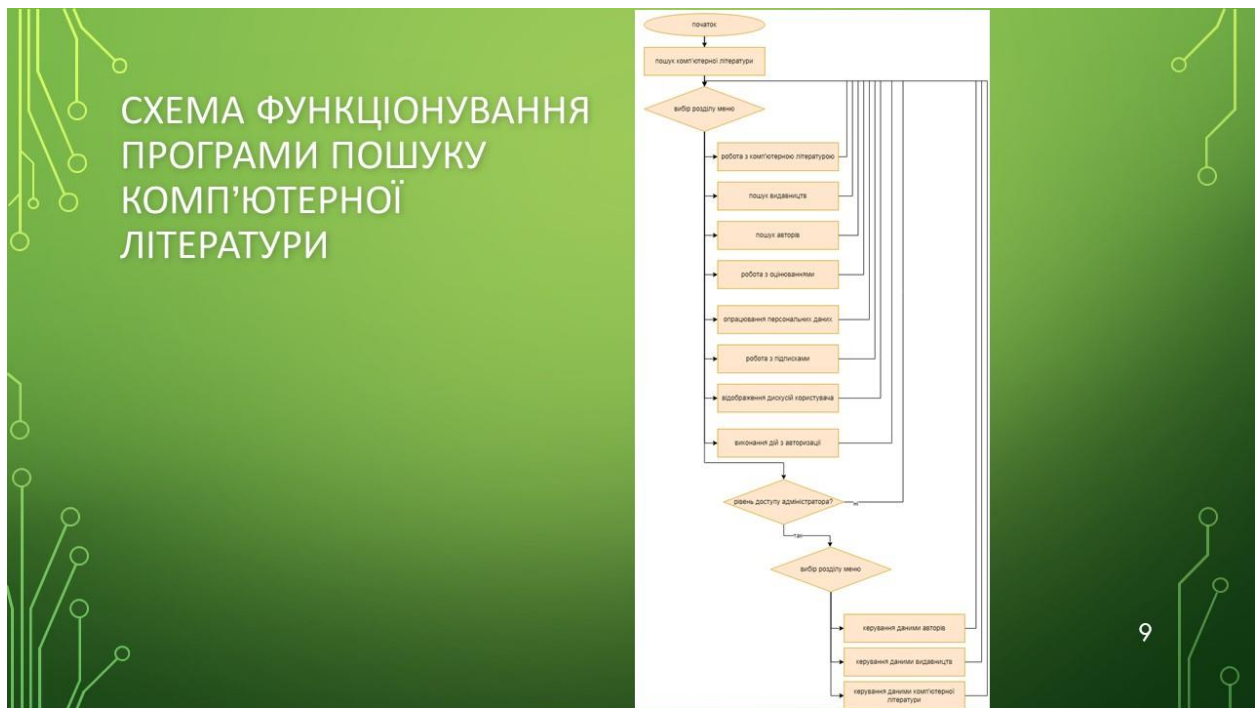


Рисунок В.9 – Схема функціонування програми пошуку комп'ютерної літератури

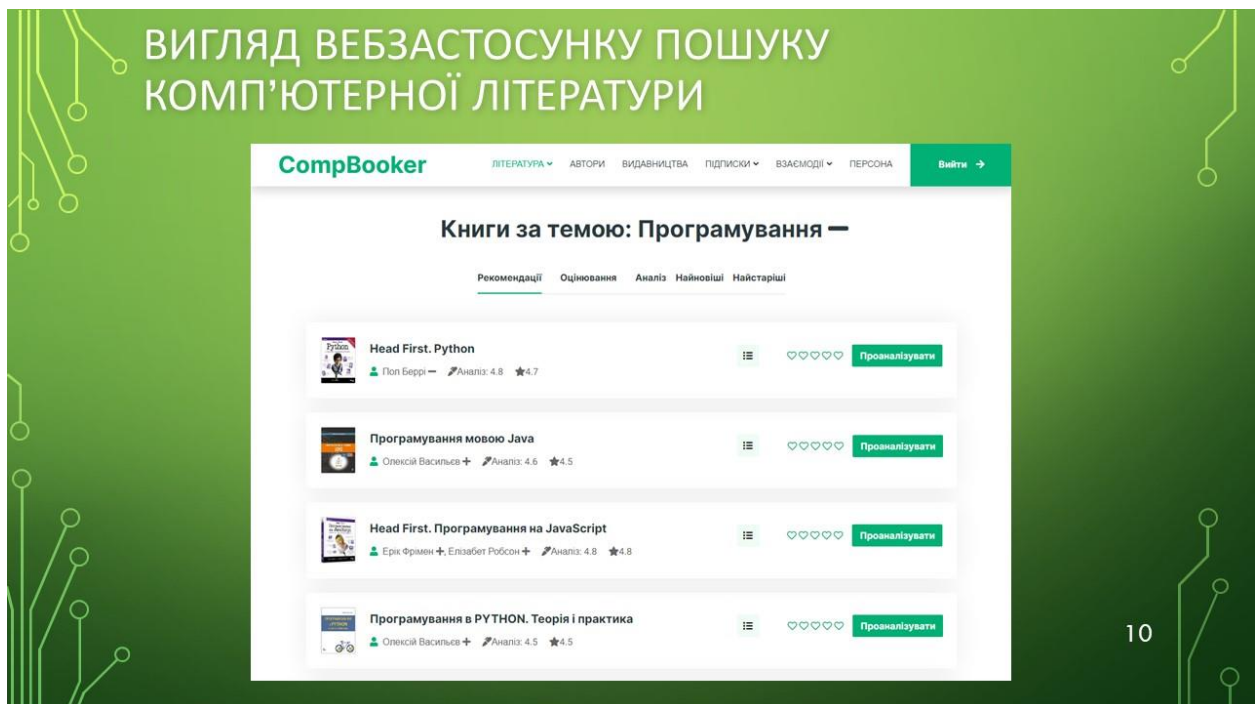


Рисунок В.10 – Вигляд вебзастосунку пошуку комп'ютерної літератури

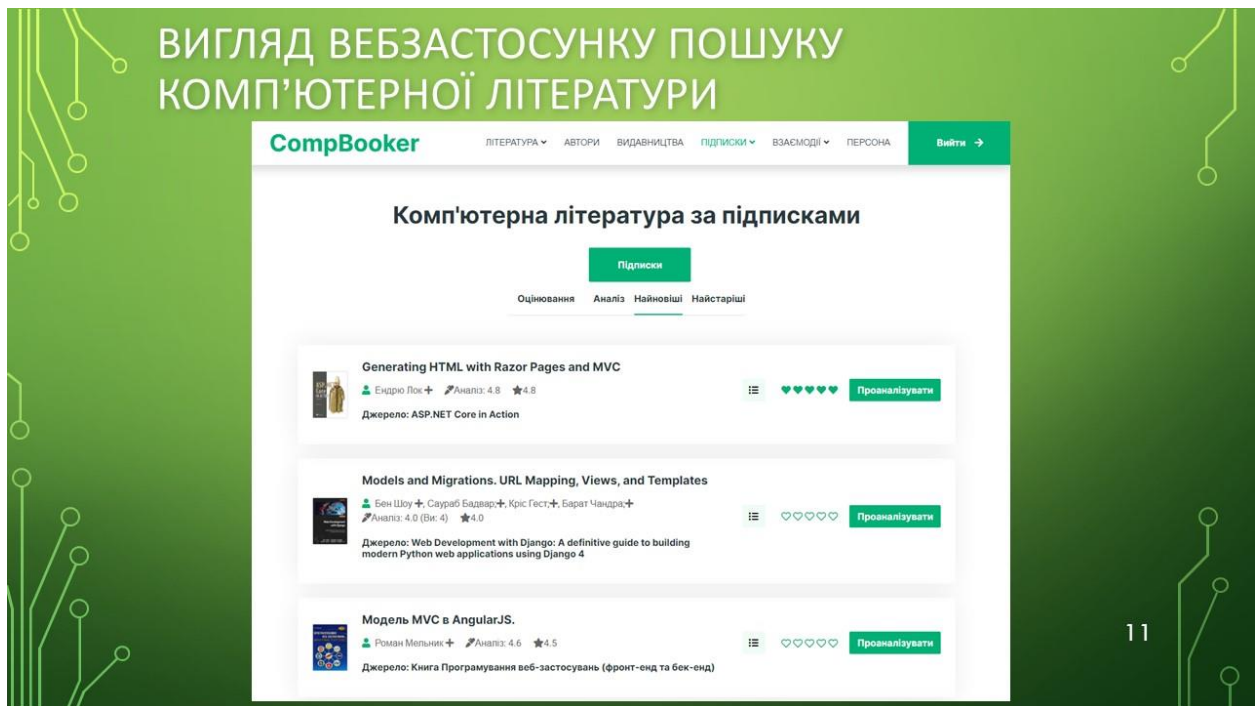


Рисунок В.11 – Вигляд вебзастосунку пошуку комп'ютерної літератури

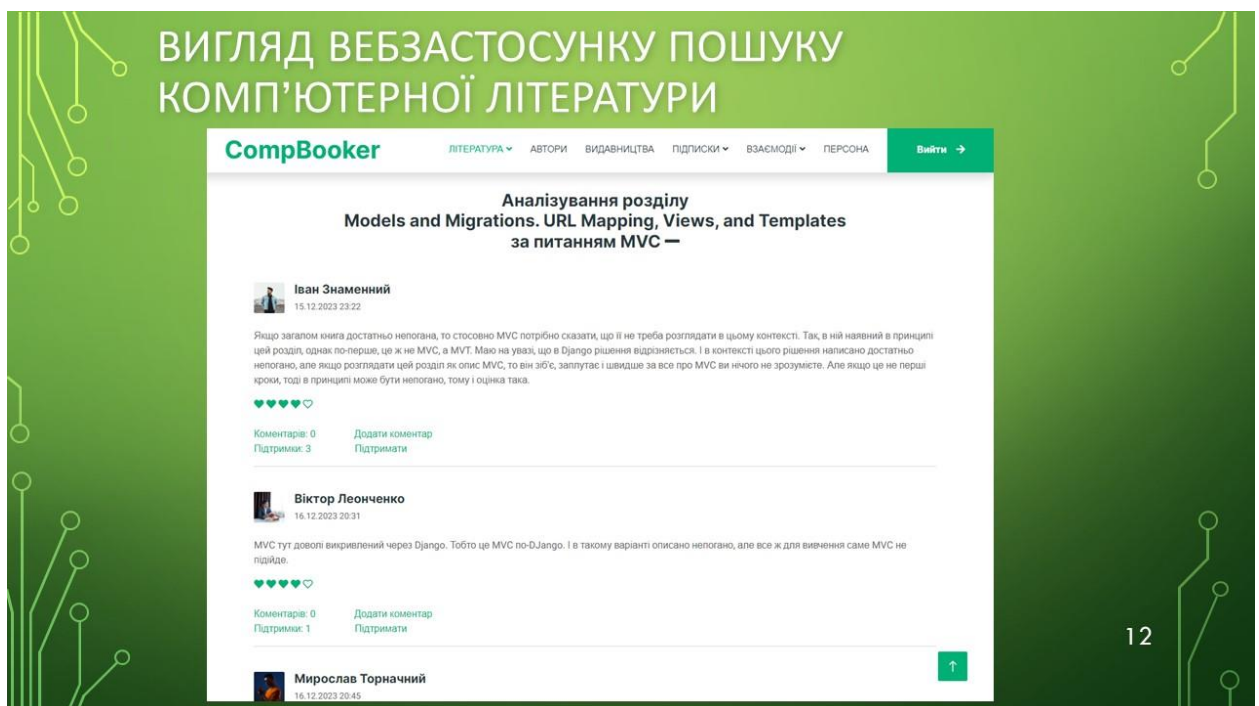


Рисунок В.12 – Вигляд вебзастосунку пошуку комп'ютерної літератури

ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ							
Таблиця 1 – Використання методу k-середніх				Таблиця 2 – Використання спектральної кластеризації			
Номер кластеру	Середня відстань всередині кластера	Мінімальна середня відстань між кластерами	Середня відстань між кластерами	Номер кластеру	Середня відстань всередині кластера	Мінімальна середня відстань між кластерами	Середня відстань між кластерами
0	0,02833938	0,04964135	0,051930566	0	0,013356447	0,013484824	0,026068068
1	0,009819105	0,012310325	0,023537541	1	0,025813239	0,052423492	0,055881299
2	0,016702048	0,014083161	0,023576244	2	0,009332151	0,012184824	0,02367195
3	0,012434121	0,012310325	0,023128767	3	0,016149931	0,017526867	0,024500129
4	0,013663798	0,012805369	0,023163394	4	0,006521888	0,013747065	0,027542732

13

Рисунок В.13 – Експериментальне дослідження

ВИСНОВКИ
- Усі початково поставлені для виконання в роботі завдання було в підсумку виконано: виконано аналіз існуючих програм, які виконують пошук комп'ютерної літератури, досліджено методи і моделі пошуку комп'ютерної літератури, виконано проектування і реалізацію програми пошуку комп'ютерної літератури. - Наукова новизна роботи полягає в запропонованому методі пошуку комп'ютерної літератури, який передбачає співвіднесення комп'ютерної літератури з проблемами, які в ній розглядаються, виділення літератури, якій за цією проблемою віддає перевагу користувач, розділення літератури за проблемою на кластери, визначення кластеру, до якого належить література, якій віддає перевагу користувач, та формування таким чином результатів пошуку.

14

Рисунок В.14 – Висновки