ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ТЕХНОЛОГІЙ ЦИФРОВОГО РОЗВИТКУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Web-додаток «Онлайн бібліотека» для перегляду, пошуку книжок з використанням JavaScript та Python»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Технології цифрового розвитку»

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

_____ Олександр АНТОНОВ
(підпис)

Виконав: здобувач вищої освіти групи ТЦР-44
Олександр АНТОНОВ

Керівник: Андрій АРОНОВ
к.т.н.

Рецензент:

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Технологій цифрового розвитку

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Технології цифрового розвитку»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Технологій цифрового розвитку
_____ Вікторія ЖЕБКА
«_____» _____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Антонову Олександру Лаврентійовичу

1. Тема кваліфікаційної роботи: «Web-додаток «Онлайн бібліотека» для перегляду, пошуку книжок з використанням мов програмування JavaScript та Python»

керівник кваліфікаційної роботи кандидат технічних наук Андрій АРОНОВ

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» лютого 2026 р. № 51.

2. Строк подання кваліфікаційної роботи «_» ___ 2026 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література з веб-розробки та інформаційних систем; матеріали з проєктування клієнт-серверної архітектури; документація фреймворку Flask; специфікації мов програмування Python та JavaScript; технології HTML і CSS; принципи побудови реляційних баз даних; структура бази даних веб-додатку «Онлайн бібліотека»; SQL-запити для обробки та вибірки даних; підходи до реалізації пошуку, фільтрації та сортування; формати обміну даними (JSON); технічні засоби роботи з PDF-файлами.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Дослідження предметної області (аналіз особливостей функціонування онлайн бібліотек, огляд існуючих веб-рішень, визначення їх переваг і недоліків, обґрунтування вибору технологій для створення власного додатку).
2. Розробка структури та реалізація веб-додатку (побудова архітектури системи, проєктування бази даних, реалізація серверної частини на Python із використанням Flask, створення клієнтського інтерфейсу, впровадження механізмів авторизації, пошуку, фільтрації та перегляду книг).
3. Перевірка працездатності та оцінка ефективності системи (тестування основних функцій додатку, аналіз стабільності роботи, оцінка швидкості обробки запитів і зручності взаємодії користувача з інтерфейсом).

5. Перелік ілюстративного матеріалу: *презентація*

1. Актуальність створення веб-додатку «Онлайн бібліотека».
2. Архітектура клієнт-серверної системи.
3. ER-діаграма бази даних.
4. Структура бази даних (основні таблиці).
5. Схема системи авторизації користувача.
6. Інтерфейс головної сторінки веб-додатку.
7. Інтерфейс пошуку, фільтрації та сортування книг.
8. Сторінка детального перегляду книги.
9. Модуль завантаження PDF-файлів.
10. Результати тестування системи.

6. Дата видачі завдання «20» лютого 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз науково-технічної літератури з веб-розробки та інформаційних систем	21.02 –25.02.2026	
2	Аналіз існуючих онлайн бібліотек та визначення вимог до системи	25.02 –28.02.2026	
3	Проєктування архітектури веб-додатку та структури бази даних	28.02 –02.03.2026	
4	Реалізація серверної частини (Flask, маршрути, логіка)	02.03 –05.03.2026	
5	Розробка клієнтського інтерфейсу (HTML, CSS, JavaScript)	05.03 –10.03.2026	
6	Реалізація функціоналу (пошук, фільтрація, авторизація, PDF)	10.03 –14.03.2026	
7	Тестування системи та оформлення роботи (вступ, висновки, реферат)	14.03 –17.03.2026	
8	Підготовка презентації та демонстраційних матеріалів	17.03 –20.03.2026	

Здобувач вищої освіти _____ Олександр АНТОНОВ

Керівник кваліфікаційної роботи _____ Андрій АРОНОВ

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 79 стор., 15 табл., 6 рис., 39 джерел.

Мета роботи – розробка веб-додатку «Онлайн бібліотека», який забезпечує зручний доступ до електронних книг, а також реалізацію пошуку, перегляду та завантаження інформаційних ресурсів.

Об'єкт дослідження – процес функціонування веб-додатків, призначених для роботи з інформаційними ресурсами.

Предмет дослідження – методи та підходи до розробки клієнт-серверних веб-додатків і організація роботи з базами даних.

Короткий зміст роботи: У цьому дослідженні здійснено аналіз сучасних підходів до розробки веб-додатків, а також розглянуто існуючі онлайн бібліотеки з акцентом на їх функціональні можливості та виявлені обмеження. Запропоновано архітектуру веб-додатку «Онлайн бібліотека», що базується на клієнт-серверній моделі, та реалізовано ключові функції, включно з переглядом каталогу книг, пошуком за назвою і автором, а також фільтрацією і сортуванням результатів.

Забезпечено відображення докладної інформації про книги, можливість завантаження PDF-файлів, аутентифікацію користувачів і формування персоналізованого списку обраних ресурсів. Проведено тестування розробленого додатку з метою верифікації коректності основних функцій системи. Для серверної частини застосовано мову програмування Python у поєднанні з фреймворком Flask, для збереження даних – систему управління базами даних PostgreSQL, а у реалізації клієнтського інтерфейсу використано HTML, CSS та JavaScript.

КЛЮЧОВІ СЛОВА: ВЕБ-ДОДАТОК, ОНЛАЙН БІБЛІОТЕКА, FLASK, PYTHON, JAVASCRIPT, БАЗА ДАНИХ, ПОШУК, ФІЛЬТРАЦІЯ

ABSTRACT

Text part of the master's qualification work: 79 pages, 6 pictures, 15 table, 39 sources.

The purpose of the work is to develop a web application "Online Library" that provides convenient access to digital books and supports search, viewing, and downloading of resources.

Subject of research is the process of functioning of web-based information systems.

Summary of the work: tIn this study, modern approaches to web application development were analyzed, and existing online libraries were examined with a focus on their functional capabilities and identified limitations. The architecture of the "Online Library" web application based on the client–server model was proposed, and key functionalities were implemented, including browsing the book catalog, searching by title and author, as well as filtering and sorting results. The system provides detailed book information, supports PDF file downloading, user authentication, and the creation of a personalized list of favorite resources. The developed application was tested to verify the correctness of its core functions. The server side was implemented using the Python programming language in combination with the Flask framework, PostgreSQL was used for data storage, and HTML, CSS, and JavaScript were applied to develop the client interface.

KEYWORDS: WEB APPLICATION, ONLINE LIBRARY, FLASK, PYTHON, JAVASCRIPT, DATABASE, SEARCH, FILTERING

ЗМІСТ

ВСТУП

У контексті динамічного розвитку інформаційних технологій цифровізація процесів зберігання, обробки та забезпечення доступу до інформації набуває значної ваги. Однією зі сфер, що зазнають суттєвих трансформацій під впливом цих процесів, є бібліотечна діяльність. Спостерігається поступова трансформація традиційних бібліотечних установ у цифрові інформаційні системи, які надають користувачам оперативний доступ до значних масивів даних, не обмежуючись їх географічним розташуванням. Така трансформація сприяє спрощенню доступу до інформації, підвищенню ефективності її використання, а також формуванню якісно нових моделей взаємодії користувачів з інформаційними ресурсами.

У зв’язку з цим виникає необхідність створення ефективних веб-додатків, які дозволяють здійснювати пошук, перегляд та отримання інформаційних ресурсів у зручному та доступному форматі. Особливої актуальності набуває розробка веб-додатків для онлайн бібліотек, що поєднують у собі функціональність, зручність інтерфейсу та високу швидкість обробки запитів користувачів.

Використання сучасних мов програмування, таких як Python та JavaScript, дозволяє створювати надійні та ефективні системи. Python широко застосовується для реалізації серверної логіки, обробки даних та взаємодії з базами даних. У даній роботі він використовується разом із фреймворком Flask для реалізації серверної частини веб-додатку. JavaScript, у свою чергу, використовується для обробки дій користувача на сторінці, зокрема для роботи з формами пошуку та покращення взаємодії з інтерфейсом.

Актуальність даної тематики зумовлена необхідністю розробки сучасних веб-рішень, які б ефективно організовували доступ до інформаційних ресурсів, забезпечуючи при цьому оперативний пошук, швидку обробку даних та інтуїтивно зрозумілу взаємодію користувача з системою. Динамічне зростання обсягів цифрової інформації та потреба у її швидкій обробці вимагають постійного вдосконалення існуючих підходів до створення веб-додатків для онлайн-бібліотек.

Метою роботи є розробка веб-додатку «Онлайн бібліотека» для спрощення доступу користувачів до електронних книжкових ресурсів, забезпечення швидкого пошуку та зручного перегляду і завантаження книг у форматі PDF із використанням мови програмування Python та допоміжного використання JavaScript. Розроблений додаток повинен забезпечувати ефективну обробку запитів користувачів, зручний інтерфейс, можливість авторизації, збереження обраних книг та надійне зберігання даних.

Об'єктом дослідження є процеси створення та функціонування веб-додатків для роботи з інформаційними ресурсами.

Предметом дослідження є методи та засоби розробки веб-додатку онлайн бібліотеки із застосуванням сучасних технологій програмування.

Завдання роботи:

– проаналізувати сучасні підходи до створення веб-додатків для онлайн бібліотек

– дослідити можливості мов програмування Python та JavaScript у веб-розробці

– виконати огляд існуючих аналогів та визначити їх переваги і недоліки

– спроєктувати архітектуру веб-додатку

– розробити базу даних для зберігання інформації про книги

– реалізувати серверну частину системи та клієнтський інтерфейс

– забезпечити функціонування механізмів пошуку, фільтрації та завантаження книг

– реалізувати систему авторизації користувачів та роботу з обраними книгами

– провести тестування та оцінити ефективність роботи системи

Методи дослідження базуються на комплексному підході до аналізу, проєктування та реалізації інформаційних систем. У процесі виконання роботи використовувалися теоретичні методи, що передбачають аналіз наукової літератури та існуючих рішень у сфері веб-розробки, що дозволило визначити основні тенденції та підходи до створення онлайн бібліотек. Також

застосовувалися методи системного аналізу для формування структури веб-додатку та визначення взаємозв’язків між його компонентами. Практична частина дослідження включає методи програмування та моделювання, які використовувалися для створення функціональних модулів системи, розробки бази даних та реалізації інтерфейсу користувача. Крім того, використовувалися методи тестування програмного забезпечення, що дозволили оцінити коректність роботи системи, її продуктивність та зручність використання.

Практична значущість результатів полягає у створенні веб-додатку «Онлайн бібліотека», який дозволяє переглядати, шукати та завантажувати книги у форматі PDF. Розроблена система може використовуватися як основа для електронних бібліотек або навчальних інформаційних систем. Також результати роботи можуть бути використані під час вивчення веб-розробки та принципів побудови клієнт-серверних додатків.

Робота пройшла апробацію. За її результатами було опубліковано наступні тези доповідей:

1. Антонов О.Л., Аронов А.О. Реалізація системи авторизації та взаємодії користувача у веб-додатку «Онлайн бібліотека». // Матеріали Всеукраїнської науково-технічної конференції «Технології цифрового розвитку: програмні системи та децентралізація». – ДУІКТ, Київ, Україна, 2026. – (подано до друку).

2. Антонов О.Л., Аронов А.О. Аналіз реалізації пошуку книжок у веб-додатку онлайн бібліотека. // Матеріали VII Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». – ДУІКТ, Київ, Україна, 2026,С. 48-49

1 ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ВЕБ-ДОДАТКІВ ДЛЯ ОНЛАЙН-БІБЛІОТЕК

1.1 Сутність, призначення та роль онлайн-бібліотек у сучасних інформаційних системах

Сутність онлайн-бібліотек у сучасних інформаційних системах полягає в трансформації традиційних підходів до зберігання, організації та доступу до інформаційних ресурсів у цифрове середовище [1]. Це забезпечує швидкість, зручність і доступність інформаційних процесів. Онлайн-бібліотека функціонує як комплексна інформаційна система, що інтегрує програмні засоби, бази даних та інтерфейси взаємодії з користувачем [2], надаючи доступ до електронних ресурсів незалежно від географічного розташування та часу. Вона не є лише цифровою копією традиційної бібліотеки, а представляє собою інтегровану веб-платформу, яка реалізує функції пошуку, фільтрації, сортування та перегляду інформації. У цьому контексті онлайн-бібліотека розглядається як динамічна система, здатна адаптуватися до змін інформаційного середовища, обсягів даних та потреб користувачів.

З позиції інформаційних технологій, сутність онлайн-бібліотек визначається їхньою здатністю ефективно обробляти інформацію, що зберігається у структурованому вигляді в базах даних [4]. Їх функціонування базується на клієнт-серверній архітектурі [6], де серверна частина відповідає за обробку запитів, взаємодію з базою даних та реалізацію бізнес-логіки, тоді як клієнтська частина забезпечує користувацький інтерфейс та взаємодію із системою. Така архітектура дозволяє забезпечити стабільну роботу системи, гнучкість розширення та ефективну обробку запитів користувачів. Крім того, сучасні онлайн-бібліотеки можуть інтегруватися з іншими інформаційними системами, що

розширює їх функціональні можливості та підвищує ефективність використання інформаційних ресурсів [7].

Призначення онлайн-бібліотек полягає у забезпеченні зручного, швидкого та надійного доступу до інформації для широкого кола користувачів [1]. Вони виконують функцію посередника між користувачем та інформаційними ресурсами, забезпечуючи інтуїтивно зрозумілий інтерфейс для пошуку та перегляду матеріалів. Однією з ключових функцій є організація ефективного пошуку інформації, що реалізується за допомогою механізмів фільтрації, сортування та відбору результатів [3]. Це дозволяє користувачам швидко знаходити необхідні ресурси навіть у відносно великих обсягах даних.

Важливим аспектом є підтримка електронних текстових ресурсів, зокрема книг у форматі PDF, що дозволяє користувачам не лише переглядати інформацію, але й завантажувати її для подальшого використання [22]. Такий підхід значно розширює функціональність системи та підвищує її практичну цінність для користувачів.

Окрім базових функцій доступу до інформації, онлайн бібліотеки виконують важливу роль у забезпеченні впорядкованого зберігання та структуризації знань [4]. Вони виступають як електронні сховища, що дозволяють систематизувати інформаційні ресурси відповідно до певних категорій, авторів або інших характеристик. Це забезпечує швидкий доступ до необхідної інформації та зручність її використання.

Роль онлайн бібліотек у сучасних інформаційних системах є надзвичайно важливою та багатогранною. Вони виступають важливим елементом інформаційної інфраструктури, забезпечуючи доступ до знань у різних сферах діяльності, включаючи освіту, науку та професійну діяльність [2]. У сфері освіти онлайн бібліотеки сприяють розвитку дистанційного навчання, надаючи студентам і викладачам доступ до навчальних матеріалів у будь-який час [38]. У науковій діяльності вони забезпечують доступ до інформаційних ресурсів, що сприяє підвищенню ефективності навчання та досліджень [6].

Особливого значення набуває роль онлайн бібліотек у контексті розвитку інформаційного суспільства, де інформація виступає одним із ключових ресурсів [1]. Вони сприяють підвищенню доступності знань, зменшуючи бар'єри між користувачами та інформаційними ресурсами. Завдяки цьому забезпечується можливість отримання інформації незалежно від місця перебування користувача, що є важливим фактором розвитку сучасного суспільства.

Таблиця 1.1

Основні характеристики онлайн бібліотек як інформаційних систем

Характеристика	Зміст характеристики	Значення для користувача
Сутність	Цифрова інформаційна система для зберігання, обробки та доступу до електронних ресурсів	Забезпечує доступ до великої кількості інформації в одному місці
Призначення	Надання швидкого та зручного доступу до книг та інших інформаційних матеріалів	Економія часу та підвищення ефективності пошуку
Архітектура	Клієнт серверна структура з використанням веб-технологій	Забезпечує стабільну роботу та доступ через браузер
Функціональність	Пошук, фільтрація, перегляд, збереження та обробка даних	Полегшує роботу з інформацією та підвищує зручність використання
Масштабованість	Можливість розширення обсягів даних та кількості користувачів	Дозволяє системі працювати при великому навантаженні
Інтерактивність	Використання динамічних інтерфейсів (JavaScript)	Покращує користувацький досвід
Інтеграція	Взаємодія з іншими інформаційними системами та базами даних	Розширює функціональні можливості системи
Безпека	Захист даних та контроль доступу	Забезпечує конфіденційність та цілісність інформації
Роль у суспільстві	Джерело знань та інформаційної підтримки освіти і науки	Сприяє розвитку інформаційного суспільства

Аналіз наведених у таблиці характеристик дозволяє зробити висновок про комплексний характер онлайн бібліотек як сучасних інформаційних систем, що

поєднують у собі функції зберігання, пошуку та надання доступу до інформації. Особливістю таких систем є їх орієнтованість на користувача, що проявляється у створенні інтуїтивно зрозумілого інтерфейсу, зручній навігації та можливості швидкого отримання необхідних даних.

З технічної точки зору онлайн бібліотеки реалізуються на основі клієнт-серверної архітектури, що дозволяє забезпечити стабільність та ефективність роботи системи. Використання такого підходу забезпечує чіткий розподіл функцій між компонентами системи: серверна частина відповідає за обробку запитів та взаємодію з базою даних, а клієнтська — за відображення інформації та взаємодію з користувачем. Це дозволяє забезпечити гнучкість системи та можливість її подальшого розвитку.

Важливим аспектом функціонування онлайн бібліотек є організація процесів пошуку та відбору інформації. У розробленому веб-додатку ці процеси реалізовані за допомогою механізмів фільтрації та сортування даних, що дозволяє користувачам швидко знаходити необхідні книги за назвою, автором або категорією. Це підвищує ефективність використання системи та спрощує роботу з інформацією.

Не менш важливою є роль безпеки, оскільки система передбачає роботу з обліковими записами користувачів. У додатку реалізовано базові механізми аутентифікації та захисту даних, зокрема використання хешування паролів та обмеження доступу до функціоналу для неавторизованих користувачів. Це забезпечує базовий рівень захисту персональних даних.

З точки зору соціального значення, онлайн бібліотеки відіграють важливу роль у формуванні інформаційного суспільства, забезпечуючи доступ до знань та інформаційних ресурсів. Вони створюють умови для самостійного навчання та отримання інформації у будь-який час та з будь-якого місця, що сприяє розвитку освіти та підвищенню рівня обізнаності користувачів..

1.2 Аналіз сучасних веб-технологій для розробки інформаційних систем

Сучасний етап розвитку інформаційних технологій позначений інтенсивним застосуванням веб-технологій, що сформували фундаментальну основу для розробки інформаційних систем різного профілю [3]. Ці технології надають універсальне середовище для створення програмних продуктів, функціонування яких здійснюється безпосередньо у браузері, виключаючи необхідність локальної інсталяції [19]. Така архітектура дозволяє використовувати їх з будь-якого місця, де існує доступ до глобальної мережі Інтернет. Ця особливість підкреслює їхню критичну роль у конструюванні сучасних інформаційних систем, зокрема електронних бібліотек, які мають відповідати вимогам доступності, зручності інтерфейсу та оперативності обробки запитів користувачів [13].

Фундаментом будь-якої веб-системи слугує клієнт-серверна архітектура, що передбачає розподіл функціональних обов'язків між клієнтською та серверною частинами [6]. Клієнтська компонента реалізується як користувацький інтерфейс, що виконується у веб-браузері, тоді як серверна частина несе відповідальність за обробку запитів, імплементацію логіки функціонування додатка та взаємодію з базою даних. Зазначений підхід забезпечує структурну ясність системи та сприяє ефективній обробці інформації.

Клієнтські веб-технології базуються насамперед на використанні мови розмітки HTML, яка визначає структуру веб-сторінки, каскадних таблиць стилів CSS, що відповідають за її зовнішній вигляд, та мови програмування JavaScript, яка забезпечує додаткову інтерактивність. У розробленому веб-додатку JavaScript використовується для обробки подій користувача, формування параметрів пошуку та покращення взаємодії з інтерфейсом без ускладнення загальної архітектури системи.

Серверна частина веб-додатків може реалізовуватися за допомогою різних мов програмування, серед яких Python займає важливе місце завдяки своїй простоті та ефективності [5]. У даному проєкті використовується фреймворк Flask [11], який дозволяє швидко створювати веб-додатки з чіткою структурою. Сервер

відповідає за обробку HTTP-запитів, виконання логіки роботи додатку, авторизацію користувачів та взаємодію з базою даних. Формування веб-сторінок здійснюється за допомогою шаблонізатора Jinja, що дозволяє динамічно генерувати HTML-контент.

Одним із ключових компонентів інформаційної системи є база даних, яка використовується для зберігання інформації про книги, користувачів та їх дії. У розробленому веб-додатку використовується реляційна база даних PostgreSQL, яка забезпечує надійне зберігання структурованих даних та підтримку зв’язків між таблицями. Це дозволяє ефективно організувати роботу з інформацією та забезпечити цілісність даних.

Важливим аспектом сучасних веб-технологій є організація взаємодії між клієнтом і сервером за допомогою HTTP-запитів. У даному веб-додатку використовується класичний підхід, при якому клієнт надсилає запит, сервер обробляє його та повертає готову HTML-сторінку. Такий підхід є простим у реалізації та забезпечує надійну роботу системи.

Не менш важливою складовою є забезпечення базового рівня безпеки інформаційної системи. У веб-додатку реалізовано механізми автентифікації користувачів, обмеження доступу до окремих функцій та використання хешування паролів [27]. Це дозволяє захистити облікові записи користувачів та забезпечити контроль доступу до функціоналу системи.

Таким чином, сучасні веб-технології забезпечують ефективні засоби для створення інформаційних систем, а їх правильне використання дозволяє реалізувати зручний, функціональний та надійний веб-додаток онлайн бібліотеки.

Таблиця 1.2

Характеристика сучасних веб-технологій для розробки інформаційних систем

Технологія	Призначення	Основні можливості	Переваги використання
HTML	Створення структури веб сторінок	Формування елементів інтерфейсу	Простота, універсальність
CSS	Оформлення веб сторінок	Дизайн, адаптивність, стилізація	Гнучкість, покращення UX
JavaScript	Динаміка та інтерактивність	Обробка подій, асинхронні запити	Швидкість, інтерактивність
React / Vue / Angular	Розробка клієнтської частини	SPA додатки, компонентна структура	Висока продуктивність, масштабованість
Python (Django, FastAPI)	Серверна логіка	Обробка запитів, API, робота з БД	Надійність, швидкість розробки
REST API	Обмін даними між клієнтом і сервером	Передача даних у форматі JSON	Гнучкість, сумісність
SQL (MySQL, PostgreSQL)	Зберігання структурованих даних	Запити, зв’язки між даними	Надійність, цілісність даних
Технологія	Призначення	Основні можливості	Переваги використання
NoSQL (MongoDB)	Робота з неструктурованими даними	Гнучке зберігання інформації	Масштабованість, швидкість
HTTPS	Захист передачі даних	Шифрування інформації	Безпека
CDN	Прискорення доставки контенту	Кешування ресурсів	Швидкість завантаження

Аналіз представлених у таблиці веб-технологій дозволяє зробити висновок про їх комплексний характер та взаємозалежність у процесі створення сучасних інформаційних систем. Кожна з розглянутих технологій виконує окрему функцію, проте лише їх поєднання забезпечує повноцінне функціонування веб-додатку. Такий підхід дозволяє досягти достатнього рівня продуктивності, зручності використання та стабільності роботи системи, що є важливими вимогами до сучасних інформаційних рішень.

Особливу роль у побудові інформаційних систем відіграє взаємодія між клієнтською та серверною частинами. У розробленому веб-додатку клієнтська частина реалізована за допомогою HTML, CSS та JavaScript, що забезпечує відображення інформації та обробку дій користувача. JavaScript використовується для формування параметрів пошуку, обробки форм та покращення взаємодії з інтерфейсом. У свою чергу серверна частина, реалізована на Python із використанням фреймворку Flask, відповідає за обробку запитів, роботу з базою даних та формування сторінок для відображення.

Взаємодія між клієнтом і сервером у даному веб-додатку здійснюється за допомогою HTTP-запитів. Користувач вводить параметри пошуку у форму, після чого сервер обробляє запит, виконує SQL-запит до бази даних та повертає сформовану HTML-сторінку. Такий підхід забезпечує простоту реалізації та надійність роботи системи.

Важливим аспектом є використання бази даних для зберігання інформації. У даному проєкті застосовується реляційна база даних PostgreSQL, яка забезпечує структуроване зберігання даних про книги, користувачів, категорії та історію дій. Використання зв’язків між таблицями дозволяє реалізувати цілісну структуру даних та ефективно виконувати запити.

Не менш важливим є питання забезпечення базової безпеки веб-додатку. У системі реалізовано механізми автентифікації користувачів, зберігання паролів у зашифрованому вигляді та обмеження доступу до окремих функцій. Це дозволяє забезпечити захист облікових записів користувачів та контроль доступу до функціоналу системи.

Оптимізація роботи системи у даному випадку досягається за рахунок ефективного формування запитів до бази даних та використання серверного рендерингу сторінок. Це дозволяє зменшити навантаження на клієнтську частину та забезпечити стабільну роботу веб-додатку навіть при збільшенні обсягів даних.

Таким чином, поєднання клієнтських та серверних веб-технологій дозволяє створити функціональний та зручний веб-додаток онлайн бібліотеки, який відповідає основним вимогам сучасних інформаційних систем.

1.3 Порівняльний аналіз мов програмування Python та JavaScript у веб-розробці

У сучасній веб-розробці мови програмування Python та JavaScript займають важливе місце завдяки своїй універсальності, широким можливостям та активному розвитку. Їх використання охоплює різні рівні створення веб-додатків, що робить доцільним проведення порівняльного аналізу з метою визначення оптимальних підходів до їх застосування у процесі розробки інформаційних систем, зокрема онлайн бібліотек.

Python є високорівневою мовою програмування, яка відзначається простотою синтаксису, зрозумілістю та зручністю у використанні. Вона широко застосовується для реалізації серверної частини веб-додатків, де виконує функції обробки запитів, реалізації логіки роботи системи та взаємодії з базами даних. У розробленому проєкті використовується фреймворк Flask, який дозволяє створювати веб-додатки з чіткою структурою та забезпечує обробку HTTP-запитів і генерацію сторінок за допомогою шаблонів. Python також забезпечує зручну роботу з базами даних, що є важливим для систем, які працюють з каталогами інформації.

У свою чергу JavaScript є мовою програмування, яка використовується для реалізації клієнтської частини веб-додатків [10]. Його основною перевагою є можливість виконання безпосередньо у браузері, що забезпечує інтерактивність та швидку реакцію інтерфейсу на дії користувача [19]. У даному веб-додатку JavaScript застосовується для обробки подій, роботи з формами пошуку та формування параметрів запиту, що покращує взаємодію користувача з системою.

Порівнюючи ці дві мови, слід зазначити, що вони виконують різні, але взаємодоповнюючі функції у веб-розробці. Python орієнтований на серверну обробку даних, забезпечуючи стабільність та коректність роботи системи. JavaScript, у свою чергу, відповідає за взаємодію з користувачем та забезпечує зручність використання інтерфейсу. Такий розподіл ролей дозволяє ефективно організувати роботу веб-додатку.

З точки зору продуктивності, JavaScript забезпечує швидке виконання операцій, пов’язаних із взаємодією користувача з інтерфейсом, оскільки обробка відбувається на стороні клієнта. Python демонструє ефективність при обробці запитів на сервері та роботі з базами даних, що є важливим для інформаційних систем, які зберігають структуровані дані.

Важливим аспектом є також підтримка та масштабованість системи. Python дозволяє створювати зрозумілий та структурований код, що полегшує його подальше супроводження. JavaScript, у свою чергу, забезпечує гнучкість у реалізації інтерфейсу та можливість покращення взаємодії з користувачем без значних змін у серверній частині.

З точки зору безпеки, основна відповідальність покладається на серверну частину, реалізовану на Python. Саме на сервері здійснюється перевірка даних, автентифікація користувачів та обмеження доступу до функціоналу системи. JavaScript використовується лише як допоміжний інструмент для взаємодії з інтерфейсом і не виконує критичних функцій безпеки.

Таким чином, поєднання Python та JavaScript дозволяє створити ефективний веб-додаток, у якому серверна частина забезпечує обробку даних та логіку роботи системи, а клієнтська — зручну взаємодію користувача з інтерфейсом.

Таблиця 1.3

Порівняльна характеристика мов програмування Python та JavaScript у веб-розробці

Критерій	Python	JavaScript
Основне призначення	Серверна частина веб додатків	Клієнтська частина веб додатків
Сфера використання	Backend, обробка даних, робота з БД	Інтерфейс користувача, обробка подій
Синтаксис	Простий, зрозумілий, читабельний	Гнучкий, динамічний
Виконання	На стороні сервера	У браузері користувача
Продуктивність	Ефективний при обробці запитів і даних	Висока швидкість взаємодії з користувачем
Фреймворки	Flask	Використовується без фреймворків
Масштабованість	Висока при серверній обробці	Обмежена клієнтською логікою

Критерій	Python	JavaScript
Робота з базами даних	Пряма робота через SQL (PostgreSQL)	Відсутня, через сервер
Безпека	Контроль доступу, обробка даних на сервері	Залежить від серверної реалізації
Додаткові можливості	Робота з даними, серверна логіка	Інтерактивність, інтерфейсу
Використання на сервері	Так	Ні
Роль у системі	Логіка, обробка, зберігання	Взаємодія з користувачем

Python та JavaScript, незважаючи на різні функціональні призначення, взаємодоповнюють один одного у розробці сучасних веб-додатків. Python забезпечує реалізацію серверної логіки та управління даними, гарантуючи стабільність системи, тоді як JavaScript відповідає за інтерактивний користувацький інтерфейс.

Таке поєднання дозволяє ефективно розподілити функції: JavaScript обробляє клієнтські дії, а Python виконує основну серверну обробку даних. Це спрощує архітектуру та забезпечує стабільну роботу.

З точки зору розробки та підтримки програмного забезпечення, Python має значні переваги завдяки простоті синтаксису та високій читабельності коду, що дозволяє швидко розробляти та змінювати функціональність системи. JavaScript, у свою чергу, забезпечує гнучкість у реалізації інтерфейсу та можливість покращення користувацького досвіду без змін серверної частини.

Хоча JavaScript може застосовуватися на сервері, у даному проєкті серверна частина реалізована винятково на Python з використанням Flask. Цей підхід забезпечує чітке розмежування відповідальності та спрощує реалізацію.

Таким чином, використання Python та JavaScript у поєднанні дозволяє створити ефективний веб-додаток онлайн-бібліотеки, у якому серверна частина забезпечує обробку даних та логіку роботи системи, а клієнтська — зручну взаємодію користувача з інтерфейсом.

1.4 Аналіз існуючих онлайн-бібліотек та їх функціональних можливостей і недоліків

У сучасному інформаційному середовищі онлайн-бібліотеки функціонують у вигляді численних платформ, що розрізняються за функціональними можливостями, рівнем доступу до інформації та технічною реалізацією. Доречно розглянути такі системи, як Project Gutenberg, JSTOR, Wikisource та Open Library, кожна з яких застосовує власний підхід до організації доступу до цифрових ресурсів.

Аналіз функціональних аспектів цих систем демонструє орієнтацію сучасних онлайн-бібліотек насамперед на забезпечення оперативного доступу до значних обсягів інформації [2]. Наприклад, Project Gutenberg надає відкритий доступ до десятків тисяч електронних книг, переважно у вільному користуванні, що підкреслює його статус одного з найдавніших та найбільш доступних цифрових архівів. Натомість JSTOR орієнтований на академічну аудиторію та забезпечує доступ до наукових журналів, книг і першоджерел, використовуючи механізми повнотекстового пошуку та систематизації інформації. Діяльність платформи Wikisource полягає у формуванні відкритого багатомовного архіву текстів, наповнення якого здійснюється користувачами, що сприяє розширенню обсягу доступних матеріалів. Тоді як Open Library має на меті формування універсального каталогу усіх опублікованих книг, забезпечуючи інтеграцію бібліографічних даних та доступ до цифрових копій.

Функціональні можливості сучасних онлайн бібліотек включають розвинені механізми пошуку, які дозволяють здійснювати пошук за різними параметрами, такими як автор, назва, рік видання або ключові слова. Використання індексації та алгоритмів ранжування забезпечує швидке отримання релевантних результатів, що значно підвищує ефективність роботи користувача [4]. Крім того, сучасні системи підтримують персоналізацію, що дозволяє формувати рекомендації на основі попередніх запитів або інтересів користувача, що відповідає загальним тенденціям розвитку веб технологій [6].

Важливою характеристикою онлайн бібліотек є підтримка різних форматів інформації, включаючи текстові документи, електронні книги, наукові статті та мультимедійні матеріали. Це робить їх універсальними інструментами для роботи з інформацією у різних сферах діяльності. Крім того, сучасні онлайн бібліотеки інтегруються з іншими інформаційними системами, що дозволяє розширити їх функціональність та забезпечити обмін даними між різними платформами.

Попри виражені переваги, онлайн-бібліотеки також демонструють певні недоліки, що потенційно обмежують їхню ефективність. Серед ключових проблем варто виділити обмежений доступ до повнотекстових ресурсів, оскільки багато систем, зокрема JSTOR, функціонують на основі передплати, що може створювати перепони для користувачів. Додатковою проблемою є нерівномірність наповнення баз даних, оскільки не всі ресурси доступні у цифровому форматі або для вільного використання. Ця ситуація зумовлена як технічними, так і правовими обмеженнями, зокрема авторським правом.

Також слід відзначити проблеми, пов’язані з якістю індексації та пошуку інформації. Деякі дослідження показують, що не всі наукові ресурси коректно відображаються у пошукових системах, що може призводити до втрати частини інформації або ускладнювати її пошук [7]. Крім того, складність інтерфейсу деяких систем може знижувати ефективність їх використання, особливо для користувачів без достатнього досвіду роботи з інформаційними системами [13].

Доступність онлайн-бібліотек критично залежить від стабільної технічної інфраструктури та безперебійного інтернет-з'єднання. Відсутність такого підключення обмежує доступ до ресурсів, відрізняючи їх від традиційних аналогів. Ефективність функціонування систем також обумовлена продуктивністю серверного обладнання та оптимізацією програмного забезпечення.

Таблиця 1.4

Порівняльний аналіз існуючих онлайн бібліотек

Онлайн бібліотека	Основні функціональні можливості	Переваги	Недоліки
Project Gutenberg	Доступ до електронних книг, пошук, завантаження у різних форматах	Безкоштовний доступ, великий архів	Обмежений сучасний контент, простий інтерфейс
JSTOR	Доступ до наукових статей, архівів журналів, пошук за категоріями	Висока якість матеріалів, наукова спрямованість	Платний доступ, обмеження для користувачів
Open Library	Каталог книг, цифрові копії, система обліку книг	Велика база даних, інтеграція ресурсів	Не всі книги доступні у повному обсязі
Wikisource	Відкритий доступ до текстів, редагування користувачами	Безкоштовність, багатомовність	Нерівномірна якість контенту
Google Books	Пошук книг, перегляд фрагментів, бібліографія	Потужний пошук, велика база	Обмежений доступ до повних текстів
Internet Archive	Архів книг, відео, аудіо, документів	Великий обсяг даних, безкоштовність	Складний інтерфейс, повільна робота

Питання безпеки та захисту даних є критичним, адже інформаційні системи вразливі до кіберзагроз, особливо при обробці персональних даних. Отже, розробка веб-додатків вимагає імплементації базових механізмів, таких як автентифікація, контроль доступу та безпечне зберігання даних.

2. ПРОЕКТУВАННЯ ВЕБ-ДОДАТКУ «ОНЛАЙН-БІБЛІОТЕКА»

2.1 Постановка задачі та формування функціональних і нефункціональних вимог системи

Постановлення цілі представляє собою базовий етап у розробці сучасних інформаційних систем, оскільки воно визначає цільові установки, функціональні вимоги та експлуатаційні межі створюваного програмного продукту. У межах даного дослідження, основне завдання полягає у розробці веб-додатку, названого «Онлайн бібліотека». Передбачається, що ця система забезпечить ефективний доступ до електронних ресурсів, запропонує дієві механізми пошуку інформації та забезпечить інтуїтивну взаємодію з користувачем. Концептуальний підхід ґрунтується на створенні веб-орієнтованого інструментарію, мета якого — систематизація, архівація та презентація даних про літературні твори, а також оперативне отримання необхідних відомостей за допомогою веб-браузера.

З огляду на сучасні вимоги до інформаційних систем, розроблювана онлайн бібліотека повинна відповідати принципам доступності, надійності та масштабованості [17]. Це означає, що система має бути доступною для користувачів у будь-який час через веб-браузер, забезпечувати стабільну роботу при різному навантаженні та мати можливість розширення функціональності без суттєвих змін у її архітектурі. Крім того, важливим аспектом є орієнтація на користувача, що передбачає створення інтуїтивно зрозумілого інтерфейсу та забезпечення швидкої реакції системи на дії користувача.

Формування функціональних вимог системи базується на визначенні основних можливостей, які повинна забезпечувати онлайн бібліотека. До таких можливостей належить організація доступу до інформаційних ресурсів, що передбачає перегляд каталогу книг, отримання детальної інформації про кожен ресурс та можливість виконання пошуку за різними параметрами (назва, автор, ключові слова, категорія) [3]. Система забезпечує перегляд сторінки окремої книги, завантаження PDF-файлу з локального сховища, збереження книг до

обраного, а також реалізацію механізмів фільтрації та сортування результатів пошуку.

Крім того, функціональні вимоги передбачають можливість роботи з користувачами, що включає процеси реєстрації, автентифікації та управління обліковими записами. Реалізована система дозволяє користувачам входити в обліковий запис, додавати книги до списку обраного та переглядати історію взаємодії з системою. Збереження історії дій (перегляд, пошук, завантаження) забезпечує базовий рівень персоналізації роботи користувача.

Взаємодія між клієнтською та серверною частинами реалізована за допомогою HTTP-запитів у межах серверного рендерингу, без використання окремого REST API [9]. Обробка запитів виконується на сервері, а результати передаються у HTML-шаблони за допомогою механізму render_template.

Нефункціональні вимоги визначають якісні характеристики системи, які не пов’язані безпосередньо з її функціональністю, але мають значний вплив на ефективність роботи. Однією з ключових вимог є продуктивність системи, яка передбачає швидку обробку запитів до бази даних та мінімальний час відгуку інтерфейсу. Це досягається за рахунок використання ефективних SQL-запитів та оптимізації структури бази даних.

Не менш важливою є вимога надійності, яка передбачає стабільну роботу системи без збоїв та коректну обробку помилок при взаємодії з базою даних. У реалізації передбачено обробку виключень, що дозволяє уникнути критичних помилок під час виконання запитів.

Важливим аспектом нефункціональних вимог є безпека, яка включає захист даних від несанкціонованого доступу. У системі реалізовано механізми автентифікації користувачів, а також використовується хешування паролів перед їх збереженням у базі даних. Це забезпечує базовий рівень захисту персональних даних.

Також слід враховувати вимоги до масштабованості системи, що передбачає можливість її подальшого розширення. Використання модульної структури

додатку та розділення логіки на окремі компоненти дозволяє додавати новий функціонал без значних змін у вже реалізованих частинах системи.

Окрему увагу необхідно приділити зручності використання системи, яка визначається якістю інтерфейсу користувача, логічністю структури навігації та швидкістю виконання основних операцій. Інтерфейс реалізований із використанням HTML, CSS та базового JavaScript, що забезпечує простоту використання навіть для користувачів без технічної підготовки. Крім того, система коректно працює у сучасних веб-браузерах, що забезпечує її доступність для широкого кола користувачів.

Таблиця 2.1

Функціональні та нефункціональні вимоги до системи «Онлайн бібліотека»

Тип вимог	Вимога	Опис вимоги	Значення для системи
Функціональні	Пошук книг	Можливість пошуку за назвою, автором, ключовими словами	Швидкий доступ до інформації
Функціональні	Перегляд каталогу	Відображення списку книг із короткою інформацією	Зручність навігації
Функціональні	Детальна інформація	Відображення повної інформації про книгу	Повнота даних
Функціональні	Перегляд і завантаження книг	Відкриття сторінки книги та завантаження PDF-файлу	Повний доступ до ресурсу
Функціональні	Реєстрація користувачів	Створення облікового запису	Персоналізація
Функціональні	Авторизація	Вхід у систему	Контроль доступу
Функціональні	Обране (Favorites)	Додавання та видалення книг зі списку обраного	Персоналізація
Функціональні	Історія дій	Збереження переглядів, пошуків та завантажень	Аналіз активності користувача
Нефункціональні	Продуктивність	Швидкий відгук системи	Комфорт роботи

Тип вимог	Вимога	Опис вимоги	Значення для системи
Функціональні	Фільтрація	Відбір даних за назвою, автором, категорією та ключовими словами	Зручність пошуку
Нефункціональні	Надійність	Стабільна робота без збоїв	Безперервність роботи
Нефункціональні	Безпека	Захист даних користувачів	Конфіденційність
Нефункціональні	Масштабованість	Можливість розширення системи	Розвиток системи
Нефункціональні	Зручність використання	Інтуїтивний інтерфейс	Доступність для користувача
Нефункціональні	Сумісність	Робота на різних пристроях і браузерах	Універсальність

Проведений аналіз функціональних та нефункціональних вимог дозволяє сформувати цілісне уявлення про майбутню інформаційну систему та визначити ключові напрями її реалізації. Встановлені вимоги є основою для подальшого проєктування архітектури веб додатку, оскільки саме вони визначають структуру системи, взаємодію її компонентів та принципи обробки даних.

Функціональні вимоги окреслюють ключові можливості системи, як-от пошук та управління даними, реалізація яких потребує оптимізації для забезпечення швидкого доступу та ефективності операцій.

Нефункціональні вимоги, у свою чергу, визначають якість функціонування системи та впливають на її ефективність у довгостроковій перспективі. Високий рівень продуктивності забезпечує комфортну роботу користувачів навіть при значному навантаженні на систему. Надійність гарантує стабільність функціонування та збереження даних, що є критично важливим для інформаційних систем. Безпека виступає одним із ключових факторів, оскільки захист інформації та персональних даних користувачів є обов’язковою умовою сучасних веб додатків.

Особливе значення має масштабованість системи, яка дозволяє адаптувати її до зростання кількості користувачів та обсягів інформації. Це передбачає

використання сучасних технологічних рішень, які дозволяють розширювати функціональність без необхідності повної перебудови системи. Зручність використання також відіграє важливу роль, оскільки інтуїтивно зрозумілий інтерфейс сприяє підвищенню ефективності роботи користувачів та зменшує час на освоєння системи.

2.2 Розробка структури та архітектури веб-додатку

Розробка структури та архітектури веб-додатку є одним із важливих етапів створення інформаційної системи, оскільки на цьому етапі визначається організація її компонентів, принципи їх взаємодії та логіка обробки даних. У рамках даного дослідження архітектура веб-додатку «Онлайн бібліотека» формується з урахуванням сучасних методів побудови веб-орієнтованих інформаційних систем, що сприяють досягненню заданих показників продуктивності, надійності та зручності користування.

Основою архітектури слугує клієнт-серверна модель, яка передбачає розподіл функціональності між двома ключовими компонентами системи. Клієнтська частина відповідає за взаємодію з користувачем та реалізацію інтерфейсу, тоді як серверна частина забезпечує обробку запитів, виконання бізнес-логіки та управління даними. У даному проєкті серверна частина реалізована за допомогою фреймворку Flask [11], який обробляє HTTP-запити та формує HTML-відповіді.

Структура веб-додатку включає декілька взаємопов’язаних рівнів, кожен з яких виконує свою функцію у процесі обробки інформації. Рівень представлення реалізується за допомогою HTML, CSS та базового JavaScript [22] і відповідає за відображення даних та взаємодію з користувачем. Інтерфейс користувача формується за допомогою шаблонів Jinja, які динамічно генеруються на сервері залежно від запитів користувача.

Рівень логіки обробки даних реалізується на серверній стороні та відповідає за виконання основних операцій системи. Він включає обробку запитів користувача,

перевірку введених даних, реалізацію пошуку, фільтрації та сортування книг, а також управління доступом до функціоналу системи. Усі ці операції виконуються у межах одного серверного застосунку без використання окремого API, що відповідає архітектурі монолітного веб-додатку [15].

Рівень зберігання даних представлений реляційною базою даних PostgreSQL, яка використовується для зберігання інформації про книги, користувачів, категорії, обрані книги та історію дій. Структура бази даних організована з урахуванням зв’язків між сутностями, що забезпечує цілісність даних та ефективність їх обробки [25].

Взаємодія між клієнтською та серверною частинами системи здійснюється через стандартні HTTP-запити, які обробляються маршрутами Flask. Сервер отримує параметри запиту, виконує відповідні SQL-запити до бази даних та передає результати у шаблони для подальшого відображення. Такий підхід дозволяє забезпечити просту та зрозумілу архітектуру без необхідності використання складних механізмів обміну даними.

Особливу увагу приділено організації внутрішньої структури додатку. Код розділений на логічні блоки, що відповідають за роботу з базою даних, обробку запитів та взаємодію з користувачем. Використання окремих функцій для виконання SQL-запитів дозволяє зменшити дублювання коду та підвищити його читабельність [8].

Важливим аспектом архітектури є забезпечення безпеки системи, що передбачає використання механізмів автентифікації користувачів, хешування паролів перед збереженням у базі даних та контроль доступу до окремих функцій. Це дозволяє забезпечити базовий рівень захисту інформації та обмежити доступ до персональних даних.

З точки зору продуктивності, архітектура системи забезпечує достатньо швидку обробку запитів за рахунок ефективної роботи з базою даних та оптимізації SQL-запитів. Використання серверного рендерингу дозволяє зменшити навантаження на клієнтську частину та забезпечити стабільну роботу системи.

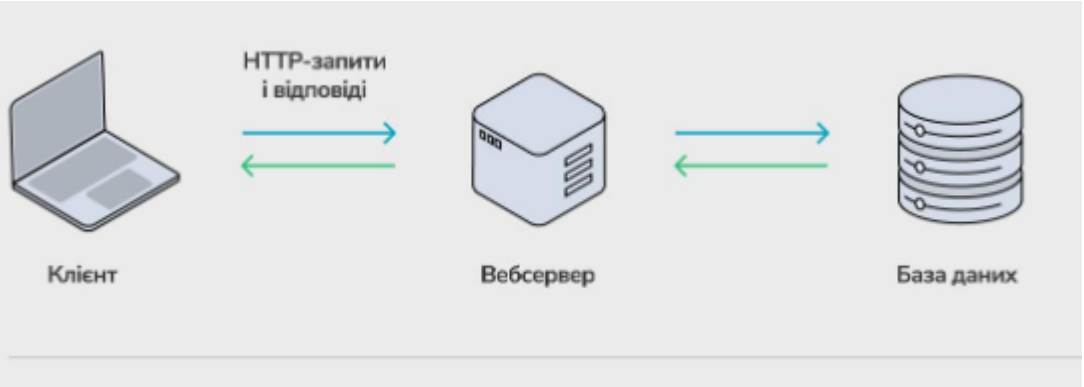


Рис. 2.1 Архітектура веб-додатку «Онлайн бібліотека»

Представлена схема ілюструє клієнт-серверну архітектуру веб-додатку, яка включає рівень клієнта, рівень сервера та рівень зберігання даних. У межах даного проєкту використовується спрощена архітектура без виділення окремого сервера додатків, що відповідає моделі монолітного веб-додатку.

На верхньому рівні розташовано клієнтську частину, яка представлена веб-інтерфейсом користувача. Саме через браузер користувач здійснює взаємодію із системою, виконує пошук книг, переглядає інформацію та працює з власним обліковим записом.

Середній рівень представляє серверну частину системи, яка реалізована за допомогою фреймворку Flask. У цьому компоненті поєднано обробку HTTP-запитів, виконання бізнес-логіки та формування HTML-відповідей. Сервер приймає запити від клієнта, виконує необхідні SQL-запити до бази даних та передає результати у шаблони для подальшого відображення.

Нижній рівень становить база даних PostgreSQL, у якій зберігається вся інформація системи, включаючи дані про книги, користувачів, категорії, обрані книги та історію дій. Серверна частина здійснює взаємодію з базою даних для отримання або збереження інформації.

Взаємодія між клієнтом і сервером відбувається через HTTP-запити, після чого сервер формує готову HTML-сторінку за допомогою шаблонів Jinja та повертає її користувачу. Таким чином, у системі використовується підхід серверного рендерингу без застосування окремого API.

Стрілки на схемі відображають напрямок передачі даних між компонентами системи, що забезпечує повний цикл обробки запиту користувача — від введення даних до отримання результату. Така архітектура є простою у реалізації, зручною у підтримці та достатньо ефективною для задач даного веб-додатку.

2.3 Проєктування бази даних (книги, користувачі, категорії, історія дій)

Проєктування бази даних є одним із найважливіших етапів створення веб-додатку «Онлайн бібліотека», оскільки саме база даних забезпечує зберігання, обробку та структуризацію всієї інформації, що використовується в системі. Якість проєктування бази даних безпосередньо впливає на продуктивність, надійність та ефективність усього програмного продукту. У межах даного проєкту база даних реалізована з використанням реляційної системи управління PostgreSQL, що дозволяє забезпечити цілісність даних та зручність їх обробки.

Основу структури бази даних становлять взаємопов’язані сутності, які відображають ключові об’єкти системи, а саме книги, користувачі, категорії, обрані книги (favorites) та історія дій. Кожна з цих сутностей представлена окремою таблицею, що містить набір атрибутів, необхідних для зберігання відповідної інформації. Для забезпечення цілісності даних використовуються первинні та зовнішні ключі.

Таблиця книг є центральним елементом бази даних, оскільки вона містить основну інформацію про літературні ресурси. У цій таблиці зберігаються такі дані, як унікальний ідентифікатор книги, назва, автор, рік видання, опис, рейтинг, назва обкладинки (cover_title), стиль відображення (cover_class), ім’я PDF-файлу та посилання на категорію. Це дозволяє забезпечити повноцінне відображення інформації у веб-інтерфейсі.

Таблиця користувачів призначена для зберігання інформації про зареєстрованих користувачів системи. Вона містить унікальний ідентифікатор користувача, ім’я користувача, електронну адресу та хеш пароля. Застосування хешування забезпечує базовий рівень безпеки персональних даних [27].

Таблиця категорій використовується для структуризації книг за тематичними напрямами. Вона містить назву категорії, а також допоміжні поля для відображення у інтерфейсі (іконка та стилі оформлення). Зв’язок між книгами та категоріями реалізується через зовнішній ключ.

Таблиця обраних книг (favorites) забезпечує можливість збереження користувачем улюблених книг. Вона містить посилання на користувача та книгу, що дозволяє реалізувати зв’язок багато-до-багатьох між цими сутностями.

Таблиця історії дій використовується для збереження інформації про взаємодію користувача з системою. У ній фіксуються такі дії, як перегляд книг, пошук та завантаження PDF-файлів. Це дозволяє реалізувати базову аналітику та персоналізацію.

Особливу увагу при проєктуванні бази даних приділено нормалізації, що дозволяє уникнути дублювання інформації та забезпечити логічну узгодженість даних. Структура таблиць організована таким чином, щоб кожна сутність відповідала окремій таблиці, а зв’язки між ними реалізовувалися через зовнішні ключі.

З точки зору продуктивності, база даних оптимізується шляхом використання індексів для часто використовуваних полів, таких як user_id та book_id. Це дозволяє прискорити виконання запитів та підвищити швидкість роботи системи.

Таким чином, розроблена структура бази даних забезпечує ефективне зберігання інформації, підтримує всі функціональні можливості системи та відповідає вимогам сучасних веб-додатків.

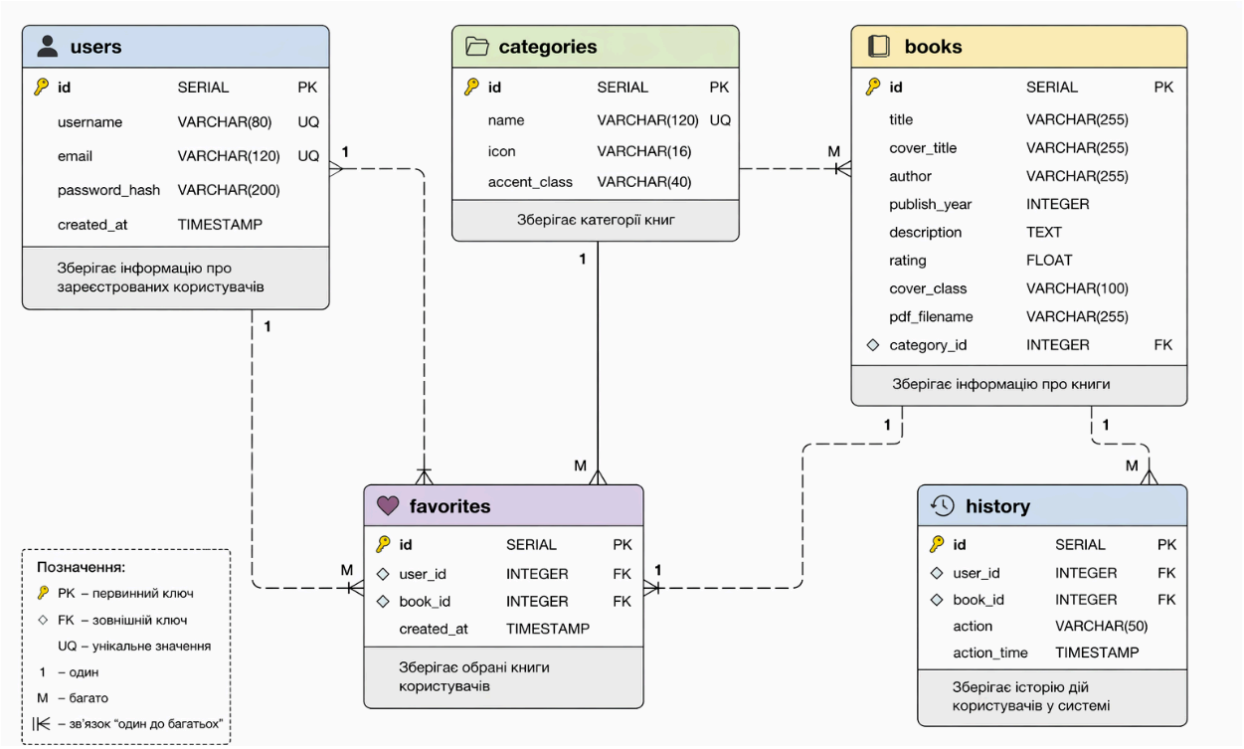


Рис. 2.2 ER-діаграма бази даних веб-додатку “Онлайн Бібліотека”

На наведеній ER-діаграмі відображено структуру бази даних, що використовується у веб-додатку. Основними сутностями системи є користувачі (users), книги (books), категорії (categories), обрані книги (favorites) та історія дій (history). Сутність books пов’язана з categories за принципом «багато до одного», оскільки кожна книга належить до однієї категорії, а кожна категорія може містити декілька книг. Сутність favorites реалізує зв’язок між користувачами та книгами, що дозволяє користувачам зберігати книги у списку обраного. Таким чином, між сутностями users і books реалізується зв’язок типу «багато до багатьох» через проміжну таблицю favorites. Сутність history використовується для збереження інформації про дії користувачів у системі, зокрема перегляд книг, пошук, додавання до обраного та завантаження PDF-файлів. Це дозволяє фіксувати взаємодію користувача з системою та забезпечує можливість подальшого аналізу активності. Така структура бази даних забезпечує впорядковане зберігання інформації, підтримує основний функціонал системи та відповідає реалізації, представленій у програмному коді веб-додатку.

Таблиця 2.2

Users

Назва поля	Тип даних	Опис
id	INT (PK)	Унікальний ідентифікатор користувача
username	VARCHAR(80)	Ім’я користувача
email	VARCHAR(100)	Електронна адреса
password_hash	VARCHAR(200)	Хеш пароля
created_at	TIMESTAMP	Дата створення

Таблиця 2.3

Categories

Назва поля	Тип даних	Опис
id	INT (PK)	Унікальний ідентифікатор категорії
name	VARCHAR(100)	Назва категорії
icon	VARCHAR(16)	Іконка категорії
accent_class	VARCHAR(40)	Стиль оформлення категорії

Таблиця 2.4

Books

Назва поля	Тип даних	Опис
id	INT (PK)	Унікальний ідентифікатор книги
title	VARCHAR(255)	Назва книги
cover_title	VARCHAR(255)	Текст для обкладинки
author	VARCHAR(255)	Автор книги
publish_year	INTEGER	Рік видання
description	TEXT	Опис книги
rating	FLOAT	Рейтинг книги
cover_class	VARCHAR(100)	CSS-клас оформлення обкладинки
pdf_filename	VARCHAR(255)	Назва PDF-файлу
category_id	INTEGER (FK)	Посилання на категорію

Таблиця 2.5

History

Назва поля	Тип даних	Опис
id	INT (PK)	Унікальний ідентифікатор запису
user_id	INT (FK)	Посилання на користувача
book_id	INT (FK)	Посилання на книгу
action	VARCHAR(50)	Тип дії
action_time	TIMESTAMP	Час дії

Таблиця 2.6

Favorites

Назва поля	Тип даних	Опис
id	INT (PK)	Унікальний ідентифікатор запису
user_id	INT (FK)	Посилання на користувача
book_id	INT (FK)	Посилання на книгу
created_at	TIMESTAMP	Дата додавання до обраного

Представлені таблиці відображають фізичну структуру бази даних веб-додатку «Онлайн бібліотека» та деталізують атрибути кожної сутності. Використання первинних ключів забезпечує унікальність записів у таблицях, тоді як зовнішні ключі дозволяють реалізувати зв’язки між сутностями та забезпечити цілісність даних.

Розроблена структура таблиць дозволяє ефективно організувати зберігання інформації, забезпечити швидкий доступ до даних та реалізувати основні функціональні можливості системи. Такий підхід відповідає сучасним вимогам до проєктування реляційних баз даних та створює надійну основу для подальшої реалізації веб-додатку.

2.4 Проєктування системи авторизації та особистого кабінету користувача

Проєктування системи авторизації та особистого кабінету користувача є важливим етапом розробки веб-додатку «Онлайн бібліотека», оскільки саме ці компоненти забезпечують персоналізацію взаємодії користувача із системою та контроль доступу до функціональних можливостей. У даному проєкті система авторизації реалізована з використанням бібліотеки Flask-Login, яка забезпечує керування сесіями користувачів та їх ідентифікацію під час роботи з додатком.

Система авторизації базується на перевірці облікових даних користувача, зокрема імені користувача (username) та пароля. Під час реєстрації користувач вводить необхідні дані, які після перевірки коректності зберігаються у базі даних. Паролі не зберігаються у відкритому вигляді, а проходять процедуру хешування за допомогою вбудованих засобів бібліотеки Werkzeug, що забезпечує базовий рівень захисту персональних даних.

Процес авторизації передбачає введення користувачем імені користувача та пароля з подальшою перевіркою цих даних серверною частиною. У разі успішної перевірки створюється сесія користувача, що дозволяє зберігати його стан авторизації між запитами без необхідності повторного введення облікових даних. Для керування сесіями використовується механізм cookies, що підтримується бібліотекою Flask-Login [21].

Особистий кабінет користувача реалізований у вигляді окремої сторінки веб-додатку, доступ до якої можливий лише після авторизації. У межах цього розділу користувач має можливість переглядати список обраних книг, а також історію власних дій, включаючи перегляд книг та інші взаємодії із системою. Це забезпечує персоналізацію використання та зручний доступ до важливої інформації.

Важливим елементом є інтеграція системи авторизації з іншими компонентами веб-додатку. Зокрема, виконання таких дій, як додавання книг до

обраного або перегляд персональної історії, доступне лише для авторизованих користувачів. Для цього використовуються спеціальні механізми обмеження доступу до маршрутів, що реалізуються за допомогою декоратора login_required.

З точки зору безпеки, система авторизації забезпечує базовий рівень захисту за рахунок хешування паролів та контролю доступу до захищених сторінок. Передача даних здійснюється через стандартні HTTP-запити, а коректність введених даних перевіряється на стороні сервера [9].

Проєктування особистого кабінету також враховує принципи зручності користування. Інтерфейс реалізований з використанням шаблонів Jinja, що дозволяє динамічно відображати дані користувача та забезпечує просту і зрозумілу навігацію в межах системи [13].

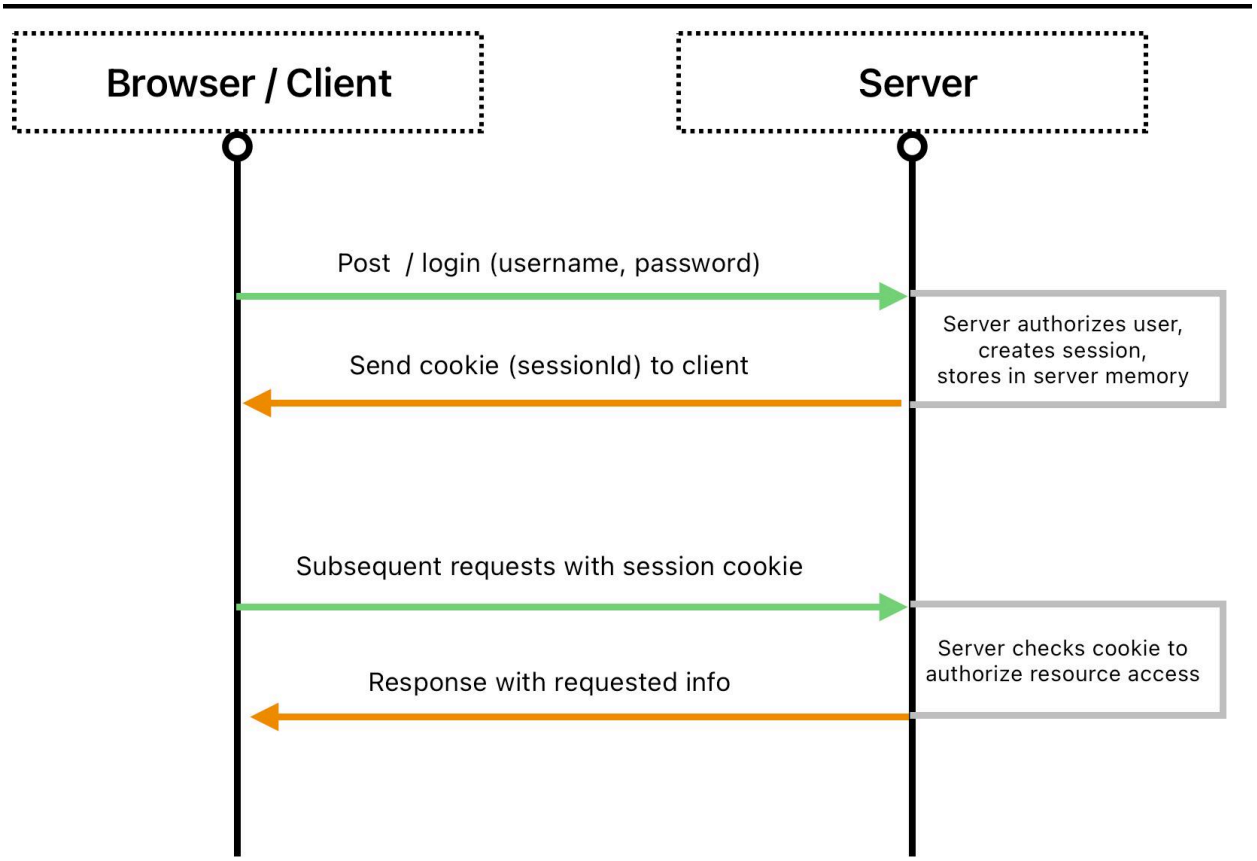


Рис. 2.4 Схема роботи системи авторизації користувача

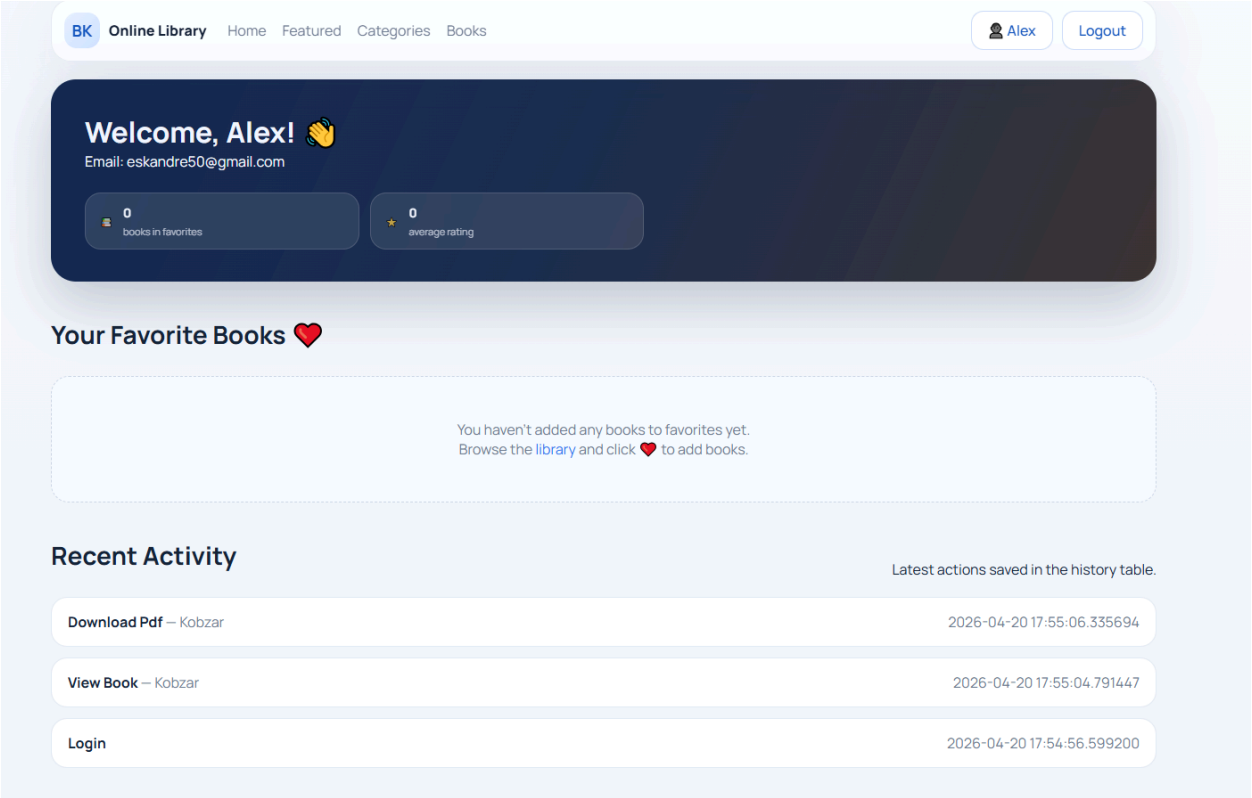


Рис. 2.5 Структура особистого кабінету користувача

Схема відображає структуру особистого кабінету користувача, який є важливим елементом взаємодії з веб-додатком після авторизації. Основними компонентами кабінету є інформація про користувача, історія дій та список обраних книг. Така структура відповідає реалізованому функціоналу системи та забезпечує персоналізовану взаємодію користувача з бібліотекою.

Інформація у кабінеті організована таким чином, щоб забезпечити швидкий доступ до основних функцій та зручність навігації. Користувач має можливість переглядати свої обрані книги, а також аналізувати власну активність у системі через історію дій, яка включає перегляд книг, пошук та інші взаємодії.

Така структура дозволяє реалізувати персоналізований підхід до взаємодії та підвищує ефективність використання веб-додатку, забезпечуючи зручний доступ до найбільш важливих для користувача даних.

2.5 Проєктування модуля перегляду книги та завантаження PDF-файлів

Проєктування модуля перегляду книги та завантаження PDF-файлів є важливою складовою веб-додатку «Онлайн бібліотека», оскільки саме він забезпечує перехід від загального списку результатів до детального ознайомлення з конкретним ресурсом [6]. Даний модуль дозволяє відобразити повну інформацію про книгу та надати користувачу доступ до електронного файлу у форматі PDF.

Основою проєктування цього модуля є створення окремої сторінки книги, яка формується серверною частиною на підставі унікального ідентифікатора запису у базі даних [4]. На сторінці відображаються назва книги, автор, рік видання, категорія, рейтинг, опис, а також кнопка для завантаження PDF-файлу. Такий підхід дозволяє розділити коротке представлення книги у каталозі та її повний інформаційний опис.

Інтерфейс сторінки книги організований таким чином, щоб користувач міг швидко отримати необхідну інформацію та виконати основні дії. Центральним елементом сторінки є інформаційний блок із характеристиками книги, а також елементи керування, що дозволяють перейти до інших розділів або виконати завантаження файлу.

Однією з ключових функцій модуля є завантаження PDF-файлу книги. Для цього у базі даних передбачено поле pdf_filename, яке містить назву відповідного файлу. Після натискання кнопки завантаження клієнт надсилає запит на сервер, де виконується перевірка наявності файлу у відповідній директорії. У разі успішної перевірки файл передається користувачу для завантаження.

Додатковою складовою модуля є фіксація дій користувача у таблиці history. У системі зберігаються записи про перегляд сторінки книги, пошук, додавання до обраного та завантаження PDF-файлів. Це дозволяє відображати активність користувача в особистому кабінеті та забезпечує базовий рівень аналітики.

Інтеграція сторінки книги з іншими компонентами веб-додатку забезпечує цілісність користувацького сценарію. Користувач переходить до детальної

інформації з каталогу, після чого може завантажити файл або повернутися до списку результатів. Така логіка підвищує зручність використання системи.

З технічної точки зору модуль реалізований у вигляді окремих маршрутів Flask. Один маршрут відповідає за відображення сторінки книги, інший — за обробку запиту на завантаження PDF-файлу. Дані передаються до шаблонів за допомогою функції render_template, що забезпечує динамічне формування HTML-сторінки.

Безпека модуля забезпечується перевіркою існування книги у базі даних, а також наявності відповідного PDF-файлу на сервері. У разі відсутності файлу система повертає повідомлення про помилку або перенаправляє користувача на сторінку книги.

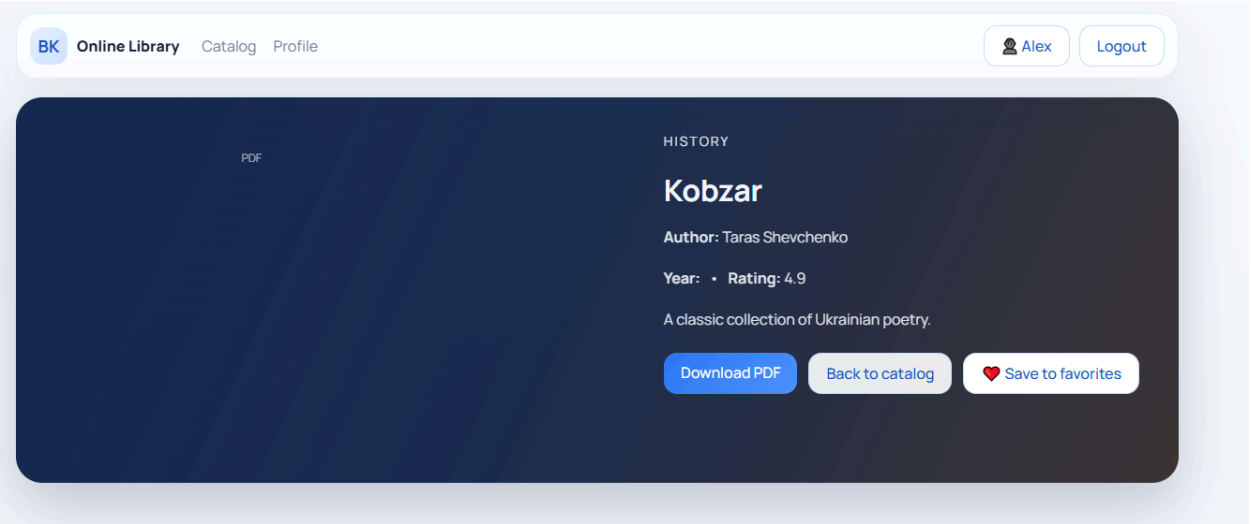


Рис 2.6 Структура сторінки книги веб-додатку

На представленому рисунку відображено загальну структуру сторінки окремої книги веб-додатку «Онлайн бібліотека». Центральним елементом є інформаційний блок, у якому відображаються основні характеристики книги, зокрема назва, автор, рік видання, категорія, рейтинг та опис.

Додатковими компонентами сторінки є блок дій користувача, що включає кнопку завантаження PDF-файлу, а також елементи навігації для повернення до каталогу або переходу до інших розділів системи. За наявності авторизації користувача можуть бути доступні додаткові дії, пов’язані із взаємодією з книгою.

Зв’язки між компонентами забезпечують повний цикл взаємодії користувача із системою — від переходу зі сторінки результатів пошуку до перегляду детальної інформації та завантаження електронного файлу книги.

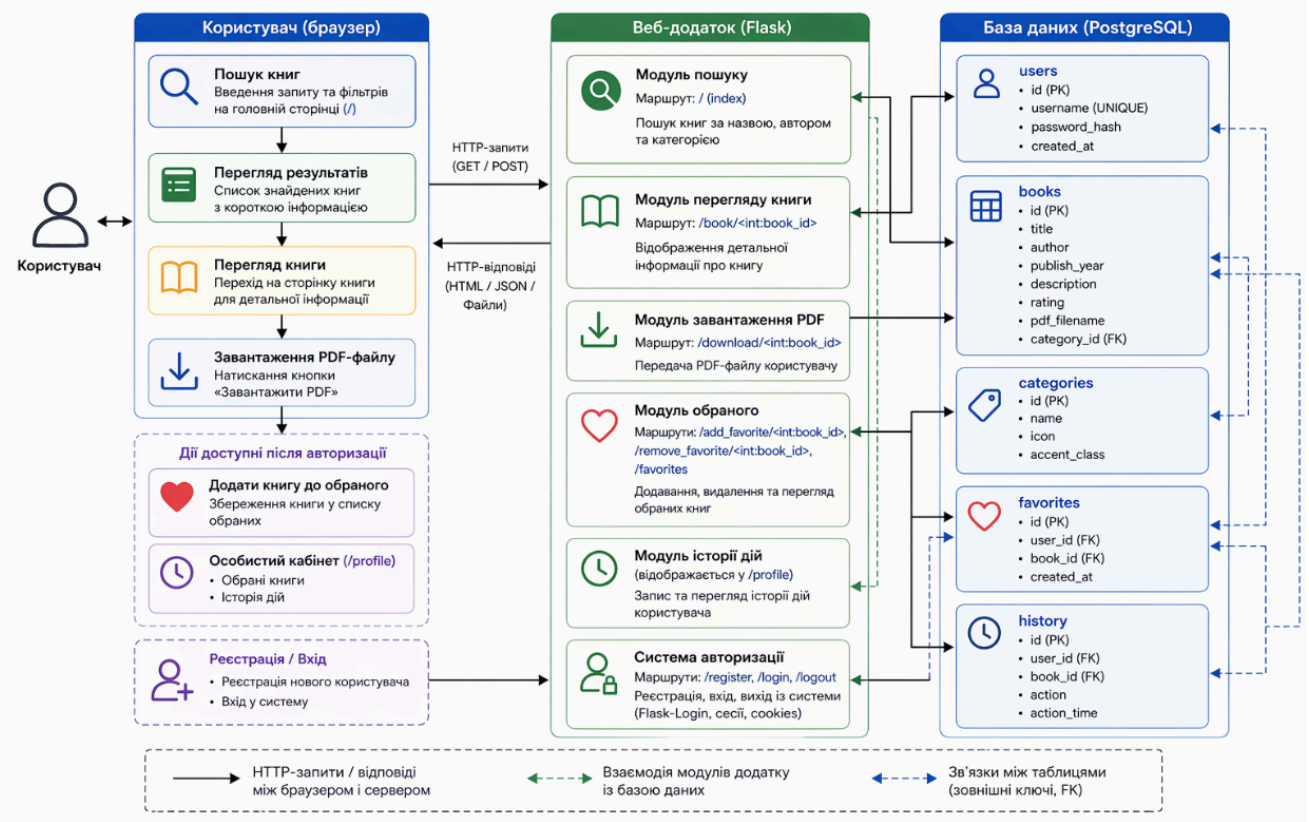


Рис. 2.7 Схема взаємодії користувача з веб-додатком “Онлайн бібліотека”

Схема відображає процес взаємодії користувача з веб-додатком під час перегляду окремої книги. Після переходу з каталогу до сторінки книги сервер виконує запит до бази даних, формує сторінку з повною інформацією та передає її клієнту. У разі натискання кнопки завантаження система виконує перевірку наявності PDF-файлу та надсилає його користувачу.

Усі операції супроводжуються перевіркою коректності даних та, за потреби, фіксуються у таблиці історії дій, що забезпечує надійне функціонування системи та можливість подальшого аналізу взаємодії користувачів із бібліотекою.

2.6 Розробка інтерфейсу користувача та UX-дизайну

Розробка інтерфейсу користувача та UX-дизайну є важливим етапом створення веб-додатку «Онлайн бібліотека», оскільки саме від зручності взаємодії залежить ефективність використання системи та загальне враження користувача. У межах даного проєкту інтерфейс орієнтований на простоту, зрозумілість та швидкий доступ до основних функцій.

Процес розробки інтерфейсу базується на визначенні ключових сценаріїв взаємодії користувача із системою, серед яких пошук книг, перегляд результатів, відкриття сторінки книги, завантаження PDF-файлів, а також робота з особистим кабінетом. Кожен із цих сценаріїв реалізований таким чином, щоб мінімізувати кількість дій користувача та забезпечити швидкий доступ до необхідної інформації.

Інтерфейс користувача реалізований за допомогою HTML та CSS, а динамічне відображення даних забезпечується за допомогою шаблонізатора Jinja, який використовується у Flask [23]. Такий підхід дозволяє формувати сторінки на сервері та передавати користувачу готовий HTML, що спрощує структуру клієнтської частини.

Структура інтерфейсу побудована за принципом логічного розділення на функціональні блоки. Головна сторінка містить форму пошуку, список категорій та результати пошуку. Сторінка книги відображає детальну інформацію про обраний ресурс, а також кнопку для завантаження PDF-файлу. Особистий кабінет надає доступ до обраних книг та історії дій користувача.

Особливу увагу приділено навігації, яка реалізована через меню, кнопки переходу та посилання між сторінками. Користувач має можливість легко переміщуватися між основними розділами системи без втрати контексту. Для покращення взаємодії використовується базовий JavaScript, який забезпечує обробку пошукової форми та формування запитів без зайвих дій користувача.

UX-дизайн орієнтований на створення зрозумілого та передбачуваного інтерфейсу. Використання візуальної ієрархії, кольорових акцентів та

структурованого розташування елементів дозволяє користувачу швидко сприймати інформацію та виконувати необхідні дії [14].

Таким чином, розроблений інтерфейс забезпечує зручність використання, логічну організацію даних та ефективну взаємодію користувача з веб-додатком.

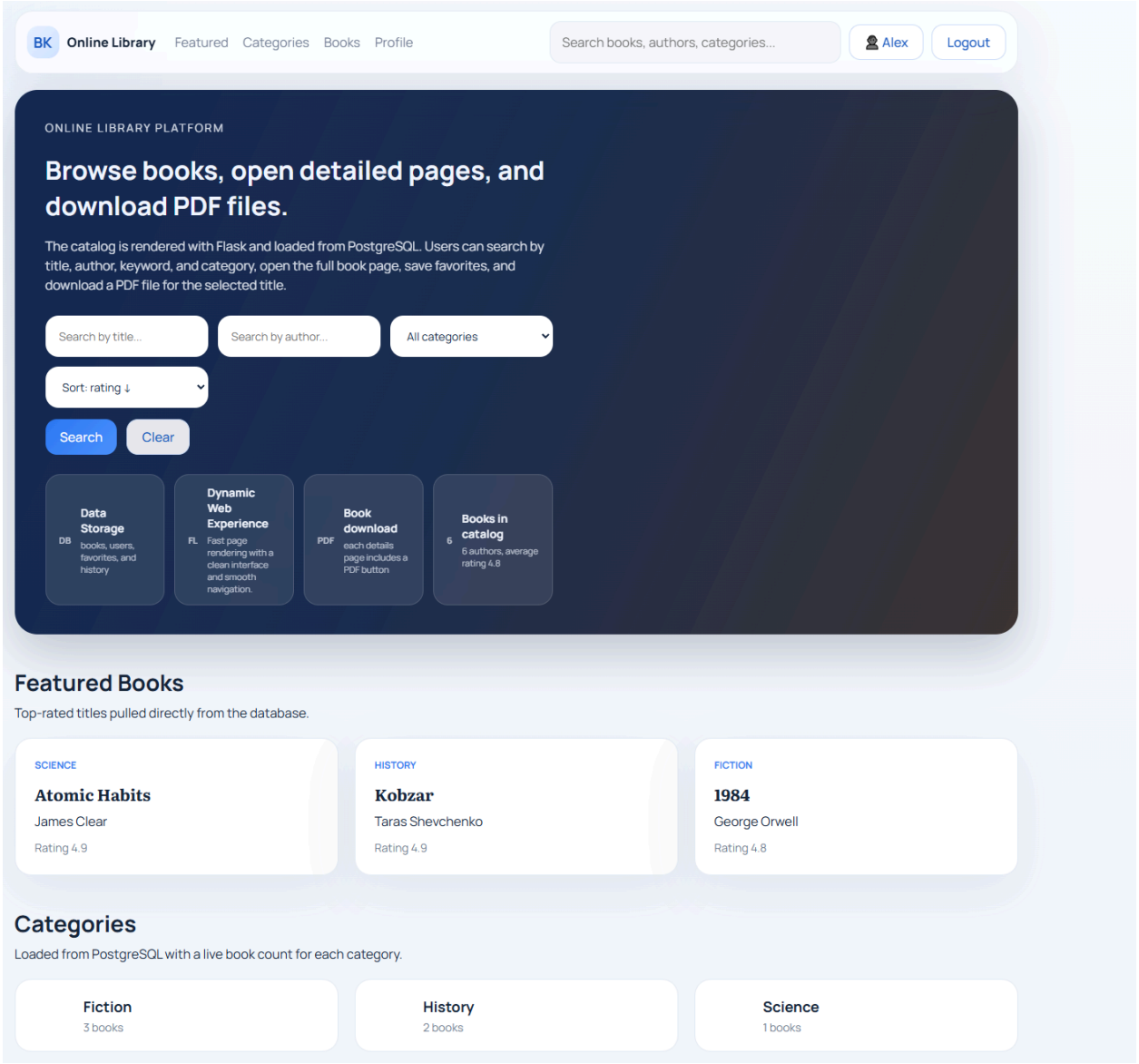


Рис. 2.8 Головна сторінка веб-додатку

На представленому рисунку відображено головну сторінку веб-додатку «Онлайн бібліотека», яка є основною точкою входу користувача в систему та забезпечує доступ до ключових функціональних можливостей. Інтерфейс сторінки побудований з урахуванням принципів зручності використання та швидкого доступу до інформації.

У верхній частині сторінки розташовано навігаційну панель, яка містить назву системи, основні розділи (Featured, Categories, Books, Profile), а також інформацію про авторизованого користувача і кнопку виходу з системи (Logout). Це забезпечує зручну навігацію та швидкий доступ до основних функцій додатку.

Центральним елементом сторінки є пошуковий блок, який дозволяє здійснювати пошук книг за назвою, автором та категорією. Додатково передбачено можливість сортування результатів (наприклад, за рейтингом), а також кнопки виконання пошуку та очищення введених параметрів. Така організація інтерфейсу дозволяє користувачу швидко сформувати запит та отримати релевантні результати.

Нижче розташовано блок Featured Books, який відображає популярні або рекомендовані книги, отримані безпосередньо з бази даних. Кожна книга представлена у вигляді картки з основною інформацією, зокрема назвою, автором та рейтингом, що дозволяє швидко оцінити доступні ресурси.

Окремим елементом сторінки є блок категорій, який забезпечує навігацію за тематичними напрямами. Користувач може обрати потрібну категорію та переглянути відповідні книги, що спрощує процес пошуку інформації.

Інтерфейс головної сторінки реалізований із використанням HTML, CSS та шаблонів Jinja, що дозволяє формувати сторінки на сервері та відображати актуальні дані з бази даних. Всі елементи інтерфейсу розташовані логічно та забезпечують швидкий доступ до основних функцій системи.

2.7 Моделювання алгоритмів пошуку, фільтрації та сортування книг

Моделювання алгоритмів пошуку, фільтрації та сортування є важливим етапом розробки веб-додатку «Онлайн бібліотека», оскільки саме ці процеси забезпечують ефективну взаємодію користувача з інформаційними ресурсами системи. У межах даного проєкту ці алгоритми реалізовані на основі обробки параметрів HTTP-запиту та формування відповідних SQL-запитів до бази даних.

Алгоритм пошуку книг базується на введенні користувачем параметрів у пошукову форму, зокрема назви книги, автора або ключового слова. Отримані значення передаються на сервер у вигляді параметрів запиту, після чого формується SQL-запит із використанням оператора LIKE, що дозволяє знаходити часткові збіги у текстових полях. Такий підхід забезпечує гнучкий пошук та дозволяє користувачу знаходити книги навіть за неповними даними.

Фільтрація результатів реалізується шляхом додавання додаткових умов до SQL-запиту. У даному проєкті передбачено фільтрацію за категорією, що дозволяє обмежити результати відповідно до обраного тематичного напряму. Фільтрація може застосовуватися одночасно з пошуком, що дозволяє уточнювати результати та підвищує їх релевантність.

Сортування результатів здійснюється за допомогою параметра sort, який передається разом із запитом користувача. На основі цього параметра формується відповідний вираз ORDER BY у SQL-запиті, що дозволяє впорядковувати результати, зокрема за рейтингом у порядку спадання або зростання [25]. Це забезпечує зручність перегляду знайдених книг.

Усі зазначені алгоритми працюють у межах єдиного процесу обробки запиту. Спочатку формується базовий SQL-запит, після чого до нього додаються умови пошуку та фільтрації, а на завершальному етапі застосовується сортування. Отримані результати передаються до шаблону та відображаються користувачу у вигляді списку книг.

Такий підхід забезпечує простоту реалізації, достатню продуктивність для невеликих обсягів даних та повністю відповідає архітектурі веб-додатку, побудованого на основі Flask та реляційної бази даних.

Таблиця 2.5

Характеристика алгоритмів пошуку, фільтрації та сортування книг

Алгоритм	Призначення	Принцип роботи	Результат
Пошук	Знаходження книг за запитом користувача	Формування SQL-запиту з використанням оператора LIKE (за назвою та автором)	Список релевантних книг
Фільтрація	Відбір книг за параметрами	Додавання умов до SQL-запиту (категорія, автор)	Зменшення результатів
Сортування	Упорядкування результатів	Використання ORDER BY (переважно за рейтингом)	Впорядкований список
Комбінований	Поєднання алгоритмів	Пошук → фільтрація → сортування	Оптимальний результат
Перегляд сторінки	Отримання повної інформації	Виклик маршруту /book/<id> та запит до БД	Сторінка з описом книги
Завантаження PDF	Отримання електронного файлу	Виклик маршруту /download/<id> та передача файлу	Завантажений PDF-файл

Проведений аналіз алгоритмів пошуку, фільтрації та сортування дозволяє визначити їх ключову роль у забезпеченні ефективного функціонування веб-додатку «Онлайн бібліотека». Кожен із розглянутих алгоритмів виконує окрему функцію, проте їх поєднання забезпечує комплексну обробку запитів користувача та формування релевантних результатів.

Алгоритм пошуку є базовим елементом системи, оскільки саме він забезпечує первинний відбір даних відповідно до запиту користувача. У межах реалізації пошук виконується шляхом формування SQL-запиту з використанням оператора LIKE, що дозволяє знаходити часткові збіги за назвою книги або автором. Це забезпечує достатню гнучкість пошуку та дозволяє працювати з неповними запитами.

Фільтрація результатів дозволяє уточнити результати пошуку та зменшити кількість нерелевантних записів. У даному проєкті вона реалізується шляхом додавання умов до SQL-запиту, зокрема за категорією або автором. Це підвищує точність результатів та зручність взаємодії користувача із системою.

Сортування результатів забезпечує впорядкування отриманих даних відповідно до заданого критерію. У системі реалізовано сортування за допомогою конструкції ORDER BY, зокрема за рейтингом книг, що дозволяє користувачу отримувати найбільш релевантні результати у зручному вигляді.

Особливе значення має комбіноване використання алгоритмів, яке передбачає їх спільну реалізацію в одному SQL-запиті. Такий підхід дозволяє одночасно виконувати пошук, фільтрацію та сортування, що забезпечує ефективну обробку запитів та швидке формування результатів.

Таким чином, реалізовані алгоритми забезпечують просту, але ефективну обробку даних, відповідають архітектурі веб-додатку та дозволяють досягти необхідного рівня зручності та продуктивності системи.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБ-ДОДАТКУ

3.1 Реалізація серверної частини на Python

Реалізація серверної частини веб-додатку «Онлайн бібліотека» є ключовим етапом розробки системи, оскільки саме сервер забезпечує обробку запитів користувачів, виконання логіки роботи системи, взаємодію з базою даних та передачу результатів на клієнтську частину. У межах даної роботи для створення серверної частини було обрано мову програмування Python, яка завдяки своїй простоті, гнучкості та широким можливостям дозволяє ефективно реалізовувати веб-додатки.

Серверна частина системи реалізована із використанням фреймворку Flask, який забезпечує маршрутизацію, обробку HTTP-запитів та формування HTML-відповідей. Усі запити, що надходять від клієнта, обробляються відповідними маршрутами, які виконують необхідні дії та повертають результат у вигляді згенерованої веб-сторінки.

Одним із ключових елементів серверної частини є система маршрутизації, яка визначає відповідність між запитами користувача та функціями обробки на сервері. У додатку реалізовано маршрути для відображення головної сторінки, перегляду окремої книги, завантаження PDF-файлів, реєстрації, входу, виходу, перегляду профілю, додавання книг до обраного та видалення їх зі списку обраного.

Реалізація логіки роботи системи включає пошук книг, фільтрацію, сортування, обробку авторизації користувачів та збереження історії дій. Усі ці операції виконуються на сервері, що дозволяє централізовано контролювати доступ до даних та забезпечувати їх цілісність. Для взаємодії з базою даних PostgreSQL використовується бібліотека psycopg, яка забезпечує виконання SQL-запитів та отримання результатів у зручному для подальшої обробки форматі.

Передача даних між серверною та клієнтською частинами у межах даного проєкту реалізується без використання окремого API. Сервер обробляє запити, отримує дані з бази даних і передає їх до шаблонів Jinja за допомогою функції render_template. Такий підхід відповідає архітектурі серверного рендерингу та дозволяє зберегти простоту структури застосунку.

Система авторизації та автентифікації реалізується на серверній стороні з використанням бібліотеки Flask-Login. Під час входу виконується перевірка облікових даних користувача, після чого створюється сесія, що дозволяє підтримувати авторизований стан між запитами. Для захисту паролів використовується хешування за допомогою засобів Werkzeug.

Особливу увагу приділено обробці помилок, яка дозволяє забезпечити стабільність роботи системи навіть у випадку некоректних запитів або проблем із доступом до бази даних. У коді застосовується обробка виключень, що дозволяє уникати критичних збоїв та коректно повідомляти користувача про помилки.

Таким чином, серверна частина, реалізована на Python із використанням Flask, забезпечує виконання всіх основних функцій веб-додатку, підтримує взаємодію з базою даних та створює основу для стабільної роботи системи.

Таблиця 3.1

Опис маршрутів серверної частини веб-додатку

Метод	URL	Опис
GET	/	Отримати каталог книг
GET	/book/<id>	Відкрити сторінку книги
GET	/download/<id>	Завантажити PDF-файл книги
GET	/logout	Вихід із системи
GET	/profile	Перегляд особистого кабінету
POST	/login	Виконати авторизацію користувача
POST	/add_favorite/<id>	Додати книгу до обраного
POST	/remove_favorite/<id>	Видалення книги з обраного

Представлена таблиця відображає основні серверні маршрути веб-додатку «Онлайн бібліотека», які забезпечують взаємодію користувача з каталогом книг, сторінкою окремої книги, системою авторизації та завантаженням PDF-файлів. Кожен із зазначених маршрутів реалізує окрему функціональну можливість системи та відповідає структурі серверної частини, створеної з використанням фреймворку Flask.

Маршрути типу GET використовуються для отримання інформації з сервера, зокрема для відображення каталогу книг, перегляду сторінки конкретної книги, переходу до особистого кабінету користувача, а також для завантаження PDF-файлів. Вони не змінюють стан системи та забезпечують безпечний доступ до даних.

Маршрути типу POST застосовуються для виконання дій, що змінюють стан системи або дані користувача. До таких дій належать авторизація користувача, додавання книги до списку обраних та її видалення. Після обробки таких запитів сервер, як правило, виконує перенаправлення користувача на відповідну сторінку.

Кожен запит обробляється серверною частиною, яка взаємодіє з базою даних PostgreSQL та формує HTML-відповідь за допомогою шаблонів Jinja. Такий підхід відповідає архітектурі серверного рендерингу та забезпечує простоту реалізації.

Використання стандартних HTTP-методів та чітко визначених URL-маршрутів дозволяє забезпечити зрозумілу структуру додатку, спростити його підтримку та підвищити ефективність взаємодії між клієнтом і сервером.

3.2 Реалізація клієнтської частини на JavaScript

Реалізація клієнтської частини веб-додатку «Онлайн бібліотека» забезпечує взаємодію користувача із системою та відображення інформації, отриманої з серверної частини. У межах даного проєкту клієнтська частина реалізована з використанням HTML, CSS та JavaScript .

Основна структура інтерфейсу формується за допомогою HTML-шаблонів, які генеруються сервером із використанням шаблонізатора Jinja. Це дозволяє передавати дані з бази даних безпосередньо у вигляді готових сторінок, що спрощує реалізацію клієнтської логіки.

Візуальне оформлення інтерфейсу забезпечується за допомогою CSS, який відповідає за розташування елементів, кольорову гаму та загальний стиль додатку. Інтерфейс побудований таким чином, щоб забезпечити зручну навігацію та швидкий доступ до основних функцій системи.

JavaScript у проєкті використовується для обробки подій користувача та покращення взаємодії з інтерфейсом. Зокрема, реалізовано обробку пошукових форм, формування параметрів запиту та перенаправлення користувача на відповідну сторінку з результатами пошуку. Використання JavaScript дозволяє спростити введення даних та підвищити зручність користування системою.

При виконанні пошуку або фільтрації дані передаються на сервер у вигляді параметрів URL (query string), після чого сервер формує відповідь та повертає користувачу оновлену сторінку. Такий підхід відповідає моделі серверного рендерингу та не потребує використання асинхронних запитів або окремого API.

Особлива увага приділяється зручності використання інтерфейсу, яка забезпечується логічною структурою сторінок, зрозумілою навігацією та швидкою реакцією на дії користувача. Усі основні дії, такі як пошук, перегляд книг або перехід між сторінками, реалізовані у вигляді простих і зрозумілих сценаріїв взаємодії.

Таким чином, клієнтська частина веб-додатку забезпечує ефективне відображення інформації, зручну взаємодію користувача із системою та повністю відповідає архітектурі серверного рендерингу, реалізованій у проєкті.

Таблиця 3.2

Основні елементи клієнтської частини

Елемент	Призначення
Поле пошуку	Введення запиту
Список книг	Відображення результатів
Кнопка	Виконання дії

Наведена таблиця відображає основні компоненти клієнтської частини веб-додатку «Онлайн бібліотека», які забезпечують взаємодію користувача із системою. Кожен з елементів реалізує специфічну функцію та входить до складу інтерфейсу, котрий формує загальний досвід користувача.

Поле пошуку служить основним каналом введення запитів, формуючи параметри, які передаються на сервер через URL. Сервер обробляє запит і повертає оновлену сторінку з результатами.

Список книг виконує функцію відображення результатів обробки запитів користувача. Він формується на серверній стороні на основі даних із бази даних та передається у вигляді готової HTML-сторінки. Кожен елемент списку містить основні відомості про книгу, що дозволяє користувачу швидко оцінити її зміст та перейти до детальної сторінки.

Кнопки інтерфейсу забезпечують виконання різних дій у системі, таких як запуск пошуку, очищення параметрів або перехід між сторінками. Вони взаємодіють із JavaScript-обробниками, які формують запит та виконують перенаправлення користувача.

Взаємодія зазначених елементів забезпечує цілісність роботи клієнтської частини та формує єдине середовище для роботи користувача. Усі дії користувача призводять до формування запиту до серверної частини, яка обробляє дані та повертає оновлений вміст сторінки. Такий підхід відповідає архітектурі серверного рендерингу та забезпечує стабільну роботу веб-додатку.

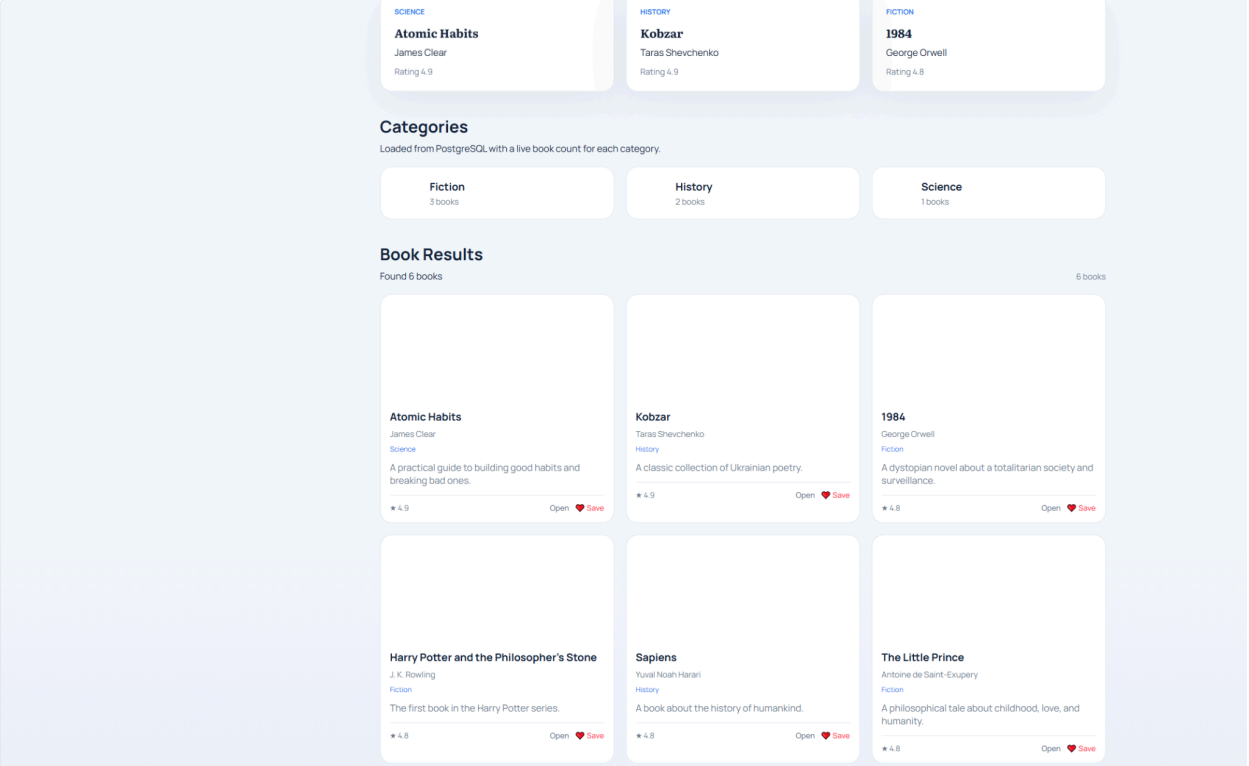


Рис. 3.1. Інтерфейс

На представленому рисунку зображено реалізований інтерфейс клієнтської частини веб-додатку «Онлайн бібліотека», який забезпечує взаємодію користувача із системою. Інтерфейс розроблений із урахуванням сучасних принципів UX-дизайну, що дозволяє забезпечити зручність використання та ефективність виконання основних операцій.

У верхній частині інтерфейсу розташовано навігаційну панель, яка містить назву додатку, основні розділи та елементи керування користувачем. Поруч із нею знаходиться пошуковий рядок, що дозволяє здійснювати пошук книг за ключовими параметрами. Його розташування забезпечує швидкий доступ та підкреслює важливість пошуку як основної функції системи.

Центральну частину сторінки займає пошуковий блок, який включає поля для введення назви книги, автора, вибору категорії та параметрів сортування. Така структура дозволяє користувачу гнучко формувати запит та отримувати релевантні результати.

Нижче розташовано блок відображення книг, який представлений у вигляді карток. Кожна картка містить основну інформацію про книгу, зокрема назву,

автора та рейтинг. Це дозволяє користувачу швидко оцінити доступні варіанти та перейти до детальної сторінки.

Додатково на сторінці присутні блоки популярних книг та категорій, які забезпечують швидкий доступ до найбільш затребуваних ресурсів та спрощують навігацію системою.

Інтерфейс побудований із використанням мінімалістичного дизайну, що сприяє зосередженню уваги користувача на основному контенті. Чітка структура, логічне розташування елементів та відсутність зайвих компонентів дозволяють підвищити зручність використання та ефективність роботи з веб-додатком.

3.3 Реалізація системи авторизації та особистого кабінету користувача

Реалізація системи авторизації та особистого кабінету користувача у веб-додатку «Онлайн бібліотека» забезпечує ідентифікацію користувачів, контроль доступу до функціональних можливостей системи та персоналізацію взаємодії з ресурсами.

Система авторизації реалізована на серверній стороні із використанням фреймворку Flask. Користувач вводить свої облікові дані (ім’я користувача та пароль), після чого вони передаються на сервер у межах HTTP-запиту [9]. На сервері виконується перевірка введених даних шляхом пошуку користувача у базі даних та порівняння хешу пароля.

Паролі у системі зберігаються у вигляді хешів, що забезпечує захист облікових даних навіть у випадку несанкціонованого доступу до бази даних.

Після успішної автентифікації створюється сесія користувача за допомогою механізму session, який дозволяє зберігати інформацію про авторизованого користувача між запитами. Це забезпечує можливість виконання подальших дій без повторного введення облікових даних.

Клієнтська частина отримує результат авторизації у вигляді перенаправлення на відповідну сторінку, після чого користувач отримує доступ до функціоналу системи, що потребує авторизації.

Особистий кабінет користувача реалізовано як окремий розділ інтерфейсу, який надає доступ до персоналізованих даних. У межах кабінету користувач може переглядати історію своїх дій та список обраних книг, що зберігаються у відповідних таблицях бази даних.

Дані для особистого кабінету формуються на серверній стороні та передаються у вигляді HTML-сторінки, що відповідає архітектурі серверного рендерингу, використаній у проєкті [15].

Таким чином, реалізована система авторизації забезпечує базовий рівень безпеки, контроль доступу до функціоналу та підтримку персоналізованої взаємодії користувача з веб-додатком.

Таблиця 3.3

Функціональні можливості системи авторизації та особистого кабінету

Компонент	Функція	Опис
Реєстрація	Створення облікового запису	Введення даних та збереження у базі
Авторизація	Вхід у систему	Перевірка логіну та пароля
Сесія	Підтримка входу	Збереження стану користувача (Flask session)
Особистий кабінет	Відображення даних	Перегляд профілю
Обрані книги	Збереження ресурсів	Додавання книг до списку favorites
Історія дій	Перегляд активності	Відображення дій користувача
Захист доступу	Контроль безпеки	Перевірка прав доступу

Важливим аспектом реалізації є забезпечення захисту даних користувача. Усі запити, що пов’язані з авторизованим доступом, супроводжуються перевіркою стану сесії користувача, що дозволяє обмежити виконання операцій лише для авторизованих користувачів. Це забезпечує базовий рівень захисту інформації та запобігає несанкціонованому доступу до персональних даних.

Крім того, система передбачає обробку помилок, що можуть виникати під час авторизації. У разі введення некоректних даних користувач перенаправляється на відповідну сторінку, де має можливість повторити спробу входу. Це забезпечує коректну взаємодію із системою та підвищує зручність використання.

Реалізація особистого кабінету також враховує принципи зручності інтерфейсу. Сторінка кабінету має логічну структуру та забезпечує швидкий доступ до основних функцій, що дозволяє користувачу ефективно взаємодіяти із системою.

Оновлення даних відбувається шляхом повторного формування сторінки на серверній стороні після виконання запиту, що відповідає архітектурі серверного рендерингу, використаній у веб-додатку.

3.4 Реалізація сторінки книги та завантаження PDF-файлів

Реалізація сторінки книги та завантаження PDF-файлів є важливим етапом розробки веб-додатку «Онлайн бібліотека», оскільки саме цей компонент забезпечує користувачу доступ до повної інформації про ресурс і можливість отримати електронний файл. На відміну від картки книги у загальному каталозі, окрема сторінка містить розширені відомості та засоби виконання дій із конкретним записом.

Розробка сторінки книги здійснюється на серверній стороні за допомогою окремого маршруту Flask, який отримує ідентифікатор книги, звертається до бази даних та передає отримані дані до HTML-шаблону. У результаті користувач отримує окрему сторінку з назвою книги, автором, категорією, роком видання, рейтингом, описом та кнопкою завантаження PDF-файлу.

Інтерфейс сторінки книги реалізований з урахуванням принципів зручності та логічної побудови даних. Основний інформаційний блок містить метадані книги, а поруч розміщується блок дій, який дозволяє завантажити PDF, повернутися до каталогу та, для авторизованого користувача, додати книгу до обраного. Така структура забезпечує цілісний користувацький сценарій роботи з ресурсом.

Функціональна модель системи орієнтована на перегляд та пошук книг, а не на редагування каталогу через інтерфейс. Основні операції користувача

включають відкриття сторінки книги, перегляд детальної інформації, застосування фільтрів і сортування, а також завантаження пов’язаного PDF-файлу.

Функція завантаження реалізована як окремий серверний маршрут, який перевіряє наявність PDF-файлу, пов’язаного з обраною книгою, та надсилає цей файл користувачу. Для збереження інформації про файл у таблиці books передбачено поле pdf_filename, що дозволяє пов’язати бібліографічний запис із відповідним електронним ресурсом.

З метою забезпечення коректної роботи системи реалізовано перевірку існування книги та відповідного PDF-файлу перед передачею ресурсу користувачу. Це дозволяє уникнути помилкових запитів і забезпечує стабільність роботи функції завантаження.

Використання окремих маршрутів для перегляду книги та завантаження файлу забезпечує чітке розмежування логіки системи та відповідає архітектурі веб-додатку.

Фрагмент реалізації перегляду сторінки книги та завантаження PDF-файлу наведено у лістингу 3.1.

Лістинг 3.1. Реалізація перегляду сторінки книги та завантаження PDF-файлу

```
@app.route('/book/<int:book_id>')
def book_detail(book_id):
    book = get_book(book_id)
    if not book:
        abort(404)
    log_action('view_book', book_id)
    return render_template('book_detail.html', book=book)
@app.route('/download/<int:book_id>')
def download_book(book_id):
    book = get_book(book_id)
    if not book or not book.get('pdf_filename'):
        abort(404)
    pdf_filename = book['pdf_filename']
```

```
pdf_path = os.path.join(PDF_DIR, pdf_filename)
if not os.path.exists(pdf_path):
    flash('PDF file is missing for this book.', 'error')
    return redirect(url_for('book_detail', book_id=book_id))
log_action('download_pdf', book_id)
return send_from_directory(PDF_DIR, pdf_filename, as_attachment=True)
```

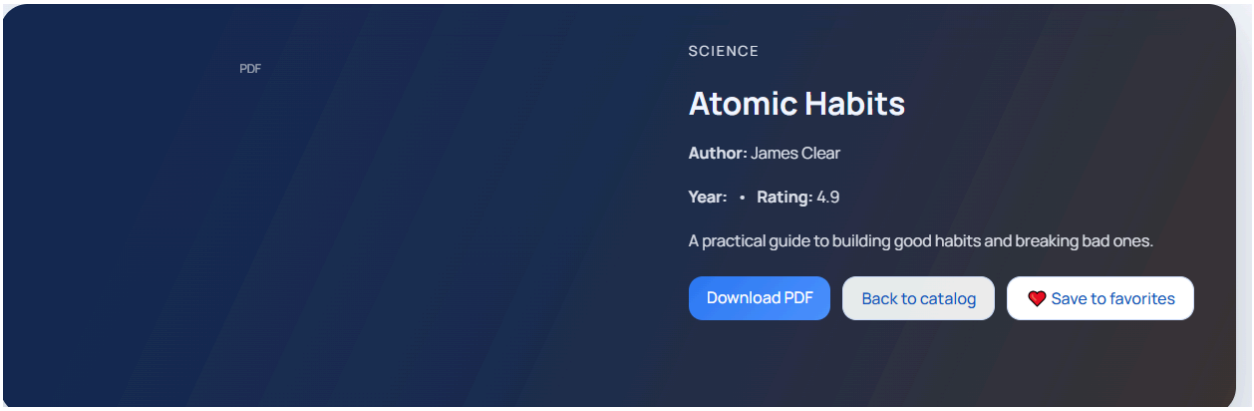


Рис. 3.4. Сторінка книги та модуль завантаження PDF-файлу

На представленому рисунку відображено інтерфейс сторінки окремої книги веб-додатку «Онлайн бібліотека», який забезпечує користувачу доступ до детальної інформації про видання та можливість завантаження PDF-файлу.

Сторінка виконана у вигляді інформаційного блоку, який поєднує візуальне оформлення та текстові дані. Ліва частина інтерфейсу використовується як декоративна зона, що імітує обкладинку або фон книги, тоді як основна інформація зосереджена у правій частині.

У правій частині сторінки відображаються ключові характеристики книги, зокрема категорія, назва, автор, рік видання та рейтинг. Нижче подано короткий опис, який дозволяє користувачу швидко ознайомитися зі змістом книги.

Особливу увагу приділено елементам керування, які забезпечують виконання основних дій користувача. Кнопка «Download PDF» дозволяє завантажити електронний файл книги, кнопка «Back to catalog» забезпечує

повернення до списку книг, а кнопка «Save to favorites» дає можливість додати книгу до обраного.

Таке розташування елементів забезпечує логічну структуру сторінки, швидкий доступ до основної інформації та зручність виконання дій.

3.5 Інтеграція клієнтської та серверної частин

Інтеграція клієнтської та серверної частин веб-додатку «Онлайн бібліотека» забезпечує узгоджену взаємодію між інтерфейсом користувача та серверною логікою. У межах даного проєкту інтеграція реалізована на основі моделі серверного рендерингу, що передбачає формування готових HTML-сторінок на сервері.

Клієнтська частина формує HTTP-запити у вигляді переходів за URL-адресами або надсилання параметрів через форму. Зокрема, під час виконання пошуку або фільтрації дані передаються у вигляді параметрів запиту (query string), після чого сервер отримує ці параметри та обробляє їх.

Серверна частина, реалізована на Flask, виконує обробку запитів, звертається до бази даних та формує відповідь у вигляді HTML-сторінки за допомогою шаблонів. Отримані дані передаються у шаблон, де відбувається їх відображення у вигляді списку книг або детальної сторінки.

Таким чином, кожна дія користувача призводить до формування нового запиту до серверної частини, яка обробляє дані та повертає оновлений вміст сторінки. Це забезпечує просту та надійну взаємодію між компонентами системи без використання додаткових механізмів обміну даними.

JavaScript у даному проєкті використовується для обробки подій інтерфейсу та формування запитів, зокрема для збору параметрів пошуку та перенаправлення користувача на відповідну сторінку.

Важливим аспектом інтеграції є перевірка стану сесії користувача, яка здійснюється на серверній стороні. Це дозволяє обмежити доступ до окремих функцій системи та забезпечити базовий рівень безпеки.

У процесі інтеграції також враховано обробку помилок, зокрема перевірку існування ресурсів та коректність запитів, що забезпечує стабільність роботи веб-додатку.

Таким чином, інтеграція клієнтської та серверної частин реалізована відповідно до принципів серверного рендерингу та забезпечує ефективну, зрозумілу та надійну взаємодію користувача із системою.

3.6 Реалізація пошуку, фільтрації та категоризації книг

Реалізація механізмів пошуку, фільтрації та категоризації книг у веб-додатку «Онлайн бібліотека» забезпечує ефективну роботу з інформаційними ресурсами та дозволяє користувачу швидко знаходити необхідні книги.

Пошук книг реалізовано як процес формування запиту користувачем через інтерфейс. Після введення даних (назви, автора або ключових слів) клієнтська частина формує параметри запиту та передає їх на сервер у вигляді параметрів URL. Серверна частина отримує ці дані, обробляє їх та формує відповідний запит до бази даних. Пошук виконується за основними полями, такими як назва книги та автор, що дозволяє отримати релевантні результати.

Фільтрація книг реалізується як додатковий механізм уточнення результатів. Користувач може обирати категорію та інші параметри, які передаються разом із пошуковим запитом. На сервері ці параметри використовуються для формування умов запиту до бази даних, що дозволяє обмежити вибірку та підвищити точність результатів.

Категоризація книг реалізована на рівні бази даних, де кожна книга пов’язана з певною категорією. Це дозволяє структурувати інформацію та забезпечити зручну навігацію. У інтерфейсі категорії відображаються у вигляді окремих елементів, що дозволяє користувачу швидко переходити до відповідних розділів.

Взаємодія пошуку, фільтрації та сортування реалізована як єдиний процес обробки запиту. Користувач може одночасно застосовувати декілька параметрів, що дозволяє отримувати більш точні результати.

Після обробки запиту сервер формує HTML-сторінку з результатами та передає її користувачу. Таким чином, оновлення даних відбувається шляхом повторного завантаження сторінки, що відповідає архітектурі серверного рендерингу.

З технічної точки зору, ефективність роботи забезпечується за рахунок оптимізованих запитів до бази даних та використання індексів для основних полів пошуку.

Особлива увага приділяється коректності роботи системи. У разі відсутності результатів користувач отримує відповідну сторінку зі списком, що може бути порожнім, що не порушує логіку роботи системи та забезпечує стабільність взаємодії.

Фрагмент реалізації SQL-запиту для пошуку, фільтрації та сортування книг наведено у лістингу 3.2.

Лістинг 3.2. Реалізація SQL-запиту для пошуку, фільтрації та сортування книг

```
def search_books(title, author, keyword, category, sort_by='rating_desc'):
    order_map = {
        'rating_desc': 'b.rating DESC, b.title ASC',
        'year_desc': 'b.publish_year DESC, b.title ASC',
        'year_asc': 'b.publish_year ASC, b.title ASC',
        'title_asc': 'b.title ASC',
        'title_desc': 'b.title DESC',
    }
    order_clause = order_map.get(sort_by, order_map['rating_desc'])
    query = f'''
        SELECT b.id, b.title, b.cover_title, b.author, b.publish_year, b.rating,
               b.cover_class, b.description, b.pdf_filename, c.name AS category_name
```

```
    FROM books b
    JOIN categories c ON c.id = b.category_id
    WHERE (%(title)s = '' OR b.title ILIKE '%%' || %(title)s || '%%')
      AND (%(author)s = '' OR b.author ILIKE '%%' || %(author)s || '%%')
      AND (%(category)s = '' OR c.name = %(category)s)
      AND (%(keyword)s = '' OR b.title ILIKE '%%' || %(keyword)s || '%%'
           OR b.author ILIKE '%%' || %(keyword)s || '%%'
           OR c.name ILIKE '%%' || %(keyword)s || '%%'
           OR COALESCE(b.description, '') ILIKE '%%' || %(keyword)s || '%%')
    ORDER BY {order_clause}
"""
```

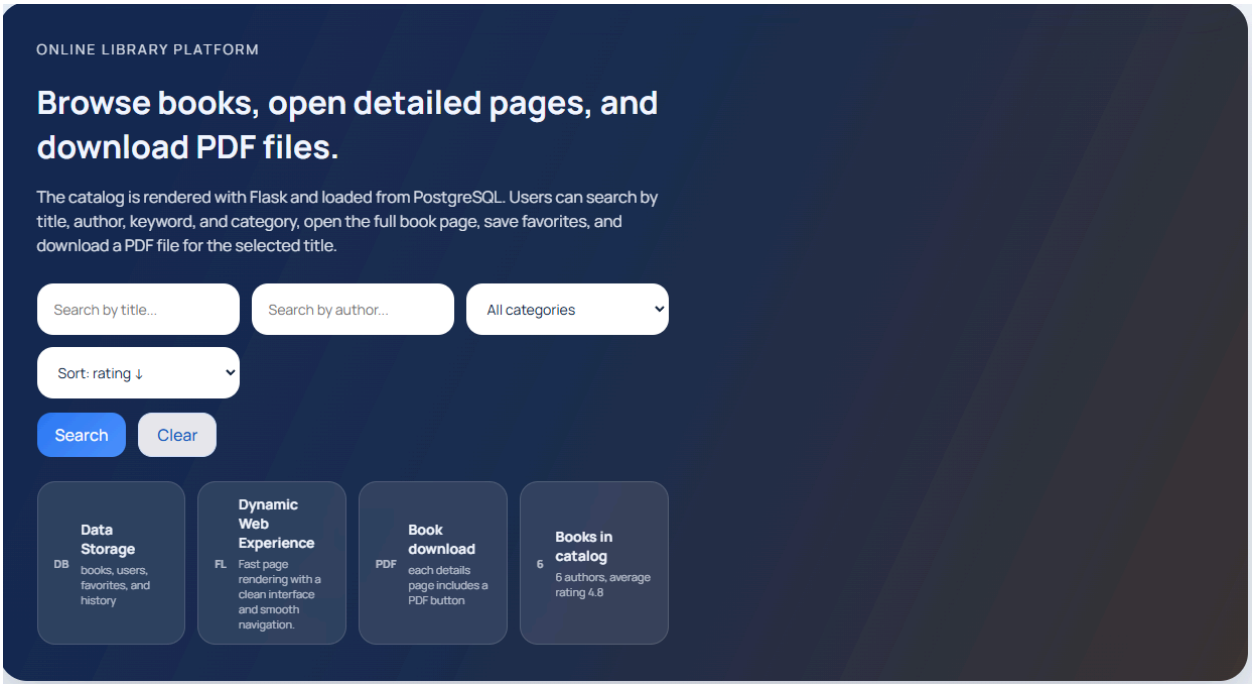


Рис. 3.4 Інтерфейс пошуку

На представленому рисунку відображено інтерфейс реалізованого механізму пошуку, фільтрації та категоризації книг у веб-додатку «Онлайн бібліотека». Даний інтерфейс є важливим елементом взаємодії користувача із системою, оскільки забезпечує швидкий доступ до інформації та зручну роботу з каталогом книг.

У верхній частині сторінки розташовано блок пошуку, який включає поля для введення назви книги та автора, а також елементи фільтрації за категоріями і сортування результатів. Така структура дозволяє користувачу одночасно

використовувати декілька параметрів для уточнення запиту, що підвищує точність отриманих результатів.

Після введення даних та натискання кнопки пошуку формується запит до серверної частини, який передається у вигляді параметрів URL. Сервер обробляє запит, виконує пошук у базі даних та формує HTML-сторінку з результатами, яка відображається користувачу.

Блок результатів пошуку представлений у вигляді списку книг, де кожен елемент містить основну інформацію про ресурс, зокрема назву, автора та рейтинг. Такий формат подання дозволяє швидко ознайомитися з результатами та перейти до сторінки детального опису книги.

Фільтрація та сортування реалізовані як частина єдиного механізму обробки запиту, що дозволяє користувачу змінювати параметри та отримувати уточнені результати. Інтерфейс також передбачає можливість очищення параметрів пошуку за допомогою відповідної кнопки.

Оновлення результатів відбувається шляхом повторного завантаження сторінки після обробки запиту сервером, що відповідає архітектурі серверного рендерингу, реалізованій у веб-додатку.

3.7 Тестування функціональності веб-додатку

Тестування функціональності веб-додатку «Онлайн бібліотека» є завершальним етапом розробки, який дозволяє перевірити коректність роботи реалізованих компонентів та підтвердити відповідність системи поставленим вимогам. Основною метою тестування є забезпечення стабільної роботи додатку та виявлення можливих помилок.

У процесі тестування було перевірено основні функціональні можливості системи. Зокрема, протестовано роботу пошуку книг за назвою та автором, фільтрацію за категоріями та сортування результатів. Перевірка показала, що система коректно формує запити до бази даних та відображає релевантні результати у вигляді списку книг.

Окрему увагу приділено тестуванню серверних маршрутів. Було перевірено роботу головної сторінки (GET /), сторінки окремої книги (GET /book/<id>), функції завантаження PDF-файлу (GET /download/<id>) та маршруту авторизації користувача (POST /login). У ході тестування встановлено, що всі маршрути обробляють запити коректно та повертають відповідні HTML-сторінки або файли.

Тестування сторінки книги включало перевірку відображення повної інформації про ресурс, а також коректність роботи кнопки завантаження PDF-файлу. У разі відсутності книги або файлу система коректно обробляє помилку та повертає відповідну відповідь (404).

Інтеграційне тестування дозволило перевірити взаємодію клієнтської та серверної частин. Було встановлено, що передача параметрів пошуку через URL працює коректно, а результати обробки запитів відображаються після повторного завантаження сторінки відповідно до архітектури серверного рендерингу.

Також проведено тестування системи авторизації. Перевірено процес реєстрації та входу користувача, правильність перевірки облікових даних, а також обмеження доступу до функцій, доступних лише авторизованим користувачам.

У процесі тестування було перевірено обробку некоректних даних, зокрема введення неправильного логіна або пароля, а також запити до неіснуючих ресурсів. Система коректно реагує на такі ситуації та забезпечує стабільну роботу.

Отримані результати тестування підтверджують, що веб-додаток функціонує відповідно до вимог, забезпечує коректну обробку запитів та може бути використаний у реальних умовах.

Таблиця 3.4

Результати тестування функціональності веб-додатку

Функція	Тестовий сценарій	Очікуваний результат	Результат
Авторизація	Введення правильних даних	Успішний вхід	Успішно
Авторизація	Невірний пароль	Помилка доступу	Успішно
Пошук	Введення назви книги	Відображення результатів	Успішно
Фільтрація	Вибір категорії	Відфільтрований список	Успішно
Завантаження PDF	Натискання кнопки download	Завантаження файлу	Успішно
Історія дій	Відкриття книги	Запис у history	Успішно
Фільтрація	Вибір категорії	Відфільтрований список	Успішно

Представлена таблиця відображає результати тестування основних функціональних можливостей веб-додатку «Онлайн бібліотека» та дозволяє оцінити рівень його працездатності. Проведене тестування охоплює ключові сценарії взаємодії користувача із системою та підтверджує коректність реалізації основного функціоналу.

Результати тестування свідчать про те, що система успішно виконує операції авторизації, забезпечуючи правильну обробку як коректних, так і некоректних облікових даних. Це підтверджує надійність механізмів контролю доступу та відповідність вимогам безпеки.

Перевірка функціоналу пошуку, фільтрації та сортування показала, що система коректно обробляє запити користувача та формує релевантні результати відповідно до заданих параметрів. Реалізовані механізми забезпечують швидкий доступ до необхідної інформації та зручну навігацію по каталогу.

Тестування переходу до сторінки книги, роботи з обраними книгами та завантаження PDF-файлів підтвердило коректність взаємодії між клієнтською та серверною частинами. Усі операції виконуються без помилок, а результати дій користувача відображаються відповідно до логіки системи.

Окрему увагу приділено перевірці обробки помилок, зокрема переходу до неіснуючих ресурсів та відсутності файлів. Система коректно реагує на такі ситуації, що свідчить про її стабільність та надійність.

Отримані результати тестування підтверджують, що веб-додаток функціонує відповідно до поставлених вимог, забезпечує коректну обробку запитів та може бути використаний у реальних умовах.

3.8 Аналіз результатів роботи та оцінка ефективності системи

Аналіз результатів роботи веб-додатку «Онлайн бібліотека» є завершальним етапом практичної частини дослідження та дозволяє оцінити рівень реалізації поставлених завдань і ефективність функціонування системи [17]. У межах даного етапу проведено комплексну оцінку роботи серверної та клієнтської частин, бази даних і механізмів їх взаємодії.

Результати тестування показали, що веб-додаток коректно реалізує основні функціональні можливості, передбачені на етапі проєктування. Система забезпечує стабільну роботу механізмів авторизації, що дозволяє надійно ідентифікувати користувачів і контролювати доступ до ресурсів. Реалізація особистого кабінету підтверджує можливість персоналізованої взаємодії користувача із системою.

Особливу увагу приділено оцінці роботи механізмів пошуку, фільтрації та сортування. Отримані результати свідчать, що система коректно обробляє запити користувача та формує релевантні результати. Використання оптимізованих запитів до бази даних дозволяє забезпечити достатню швидкість обробки даних у межах навчального проєкту.

Оцінка клієнтської частини показала, що інтерфейс системи є зручним та інтуїтивно зрозумілим. Взаємодія користувача із системою реалізована через послідовне оновлення сторінок після обробки запитів сервером, що відповідає обраній архітектурі серверного рендерингу. Інтерфейс є адаптивним та забезпечує коректне відображення на різних пристроях.

Аналіз продуктивності системи показав, що додаток забезпечує стабільну роботу при обробці користувацьких запитів без помітних затримок. Архітектура системи дозволяє у майбутньому масштабувати її функціональність та розширювати обсяг даних.

Важливим аспектом є забезпечення безпеки системи. Реалізовані механізми авторизації, контроль доступу до ресурсів і обробка помилок забезпечують захист даних та стабільну роботу додатку.

У процесі аналізу також враховано зручність супроводу системи. Використана архітектура дозволяє легко розширювати функціональність, додавати нові книги та вдосконалювати систему без суттєвих змін у її структурі.

Отримані результати дозволяють зробити висновок, що веб-додаток «Онлайн бібліотека» відповідає поставленим вимогам, демонструє стабільну роботу та може бути використаний як основа для подальшого розвитку та практичного застосування.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було досягнуто мету – розробку веб-додатку «Онлайн бібліотека», який забезпечує ефективне зберігання, обробку та пошук інформації про книги. Дослідження охопило теоретичні та практичні аспекти створення інформаційних систем, що дозволило сформувати цілісне розуміння принципів роботи онлайн-бібліотек.

1. У теоретичній частині проаналізовано сутність онлайн-бібліотек, сучасні веб-технології та проведено порівняльний аналіз мов програмування Python і JavaScript. Це дозволило обґрунтувати вибір технологічного стеку для реалізації системи.

2. Практична частина охоплює реалізацію серверної частини на Python із фреймворком Flask для обробки запитів і взаємодії з базою даних, а також клієнтської частини на базі HTML, CSS і JavaScript для створення інтерфейсу користувача. Інтеграція цих компонентів забезпечила функціональність системи.

3. Реалізовано систему авторизації, особистий кабінет, сторінку перегляду книги, механізм завантаження PDF-файлів, а також функції пошуку, фільтрації і сортування, що підвищує зручність користувацької взаємодії з каталогом.

4. Тестування підтвердило коректну роботу усіх компонентів, їх узгоджену взаємодію, стабільність і прийнятний рівень продуктивності в межах поставлених завдань.

У процесі роботи здобуто практичні навички у веб-розробці, зокрема у створенні клієнт-серверних додатків, проектуванні баз даних і реалізації зв’язків між компонентами, що підкреслює прикладну цінність дослідження. Отримані результати свідчать, що розроблений додаток відповідає сучасним вимогам інформаційних систем та може слугувати основою для подальшого вдосконалення. Перспективи розвитку пов’язані з розширенням функціональності, впровадженням рекомендаційних алгоритмів, інтеграцією зовнішніх сервісів і підвищенням безпеки та продуктивності системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Глушков В. М. Основи інформаційних систем. Київ : Наукова думка, 2018. 320 с.
2. Коваленко О. В. Інформаційні технології та системи. Київ : Центр учбової літератури, 2020. 384 с.
3. Бондаренко М. Ф. Веб-технології та веб-дизайн. Харків : ХНУРЕ, 2019. 276 с.
4. Пасічник В. В. Організація баз даних та знань. Львів : Новий світ, 2021. 400 с.
5. Іванченко І. С. Основи програмування. Київ : КНУ, 2017. 350 с.
6. Sommerville I. Software Engineering. 10th ed. Pearson, 2015. 810 p.
7. Pressman R. Software Engineering: A Practitioner’s Approach. McGraw-Hill, 2014. 970 p.
8. Martin R. Clean Code. Prentice Hall, 2008. 464 p.
9. Fielding R. REST Architectural Styles. 2000. 162 p.
10. Flanagan D. JavaScript: The Definitive Guide. O’Reilly, 2020. 706 p.
11. Grinberg M. Flask Web Development. O’Reilly, 2018. 258 p.
12. Ramalho L. Fluent Python. O’Reilly, 2022. 1014 p.
13. Nielsen J. Usability Engineering. Morgan Kaufmann, 1994. 362 p.
14. Krug S. Don’t Make Me Think. New Riders, 2014. 216 p.
15. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2002. 533 p.
16. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1994. 395 p.
17. ISO/IEC 25010:2011. Systems and software engineering — Systems and software quality requirements and evaluation (SQuaRE) — System and software quality models. 34 p.
18. IEEE Software Engineering Standards. IEEE, 2017. 120 p.
19. Mozilla Developer Network. Web Docs. URL: https://developer.mozilla.org/ (дата звернення: 10.04.2026).
20. Python Documentation. URL: https://docs.python.org/ (дата звернення: 10.04.2026).
21. Flask Documentation. URL: https://flask.palletsprojects.com/ (дата звернення: 10.04.2026).
22. HTML Standard. URL: https://html.spec.whatwg.org/ (дата звернення: 10.04.2026).

23. CSS Specification. URL: https://www.w3.org/Style/CSS/ (дата звернення: 10.04.2026).
24. ECMAScript Specification. URL: https://tc39.es/ecma262/ (дата звернення: 10.04.2026).
25. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/ (дата звернення: 10.04.2026).
26. MySQL Documentation. URL: https://dev.mysql.com/doc/ (дата звернення: 10.04.2026).
27. OWASP Foundation. Web Security Testing Guide. URL: https://owasp.org/ (дата звернення: 10.04.2026).
28. JSON Specification. URL: https://www.json.org/ (дата звернення: 10.04.2026).
29. REST API Tutorial. URL: https://restfulapi.net/ (дата звернення: 10.04.2026).
30. Git Documentation. URL: https://git-scm.com/docs/ (дата звернення: 10.04.2026).
31. Docker Documentation. URL: https://docs.docker.com/ (дата звернення: 10.04.2026).
32. Kubernetes Documentation. URL: https://kubernetes.io/docs/ (дата звернення: 10.04.2026).
33. FreeCodeCamp. Web Development Guide. URL: https://www.freecodecamp.org/ (дата звернення: 10.04.2026).
34. Smashing Magazine. Web Design Articles. URL: https://www.smashingmagazine.com/ (дата звернення: 10.04.2026).
35. Google Developers. Web.dev. URL: https://web.dev/ (дата звернення: 10.04.2026).
36. Stack Overflow Survey 2024. URL: https://survey.stackoverflow.co/ (дата звернення: 10.04.2026).
37. UX Design Institute. UX Principles. URL: https://uxdesigninstitute.com/ (дата звернення: 10.04.2026).
38. Khan Academy. Programming Courses. URL: https://www.khanacademy.org/ (дата звернення: 10.04.2026).
39. React Documentation. URL: https://react.dev/ (дата звернення: 10.04.2026).
40. Антонов О. Л., Аронов А. О. Реалізація системи авторизації та взаємодії користувача у веб-додатку «Онлайн бібліотека». Матеріали Всеукраїнської науково-технічної конференції «Технології цифрового розвитку: програмні системи та децентралізація». ДУІКТ, Київ, Україна, 2026. (подано до друку).

41. Антонов О. Л., Аронов А. О. Аналіз реалізації пошуку книжок у веб-додатку «Онлайн бібліотека». Матеріали VII Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». ДУІКТ, Київ, Україна, 2026.С. 48-49

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ТЕХНОЛОГІЇ ЦИФРОВОГО РОЗВИТКУ

КВАЛІФІКАЦІЙНА РОБОТА на тему: «Web-додаток «Онлайн бібліотека» для перегляду, пошуку книжок з використанням JavaScript та Python»

Виконав: здобувач вищої освіти
Гр. ТЦР-41 Антонов Олександр
Керівник: кандидат технічних наук Андрій Аронов

Київ-2026

Об’єкт мета і предмет дослідження

Об’єктом дослідження процес функціонування веб-додатків, призначених для роботи з інформаційними ресурсами.

Предметом дослідження методи та підходи до розробки клієнт-серверних веб-додатків і організація роботи з базами даних.

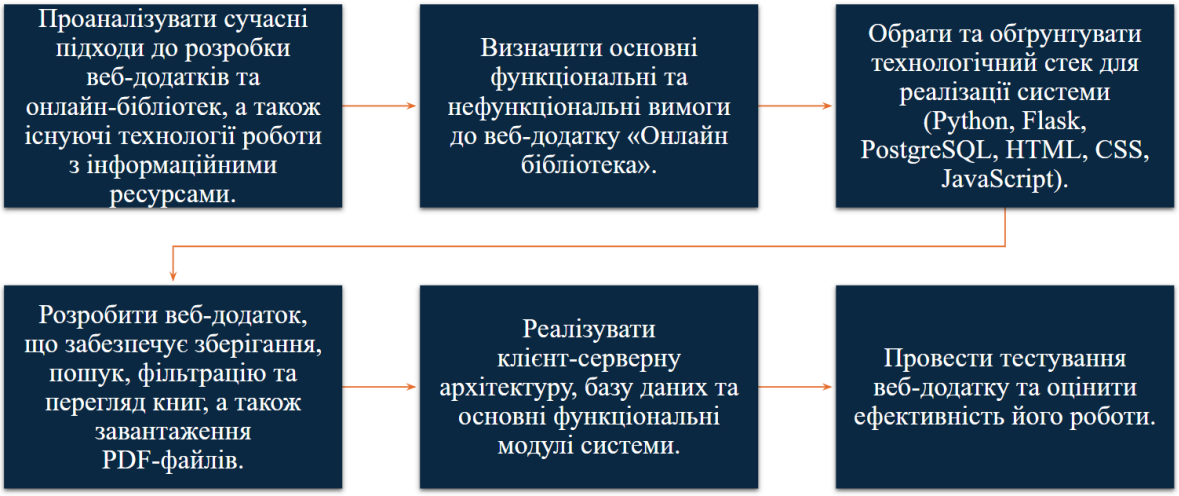
Мета дослідження розробка веб-додатку «Онлайн бібліотека», який забезпечує зручний доступ до електронних книг, а також реалізацію пошуку, перегляду та завантаження інформаційних ресурсів.

Актуальність теми

У сучасних умовах стрімкого розвитку інформаційних технологій зростає потреба у швидкому та зручному доступі до цифрових ресурсів. Онлайн-бібліотеки дозволяють користувачам переглядати, шукати та завантажувати книги незалежно від місця перебування, забезпечуючи ефективну організацію даних і доступність навчальних та літературних матеріалів.

Актуальним є створення веб-додатку «Онлайн бібліотека», який об’єднує каталог книг, пошук, фільтрацію, перегляд детальної інформації та завантаження PDF-файлів. Такий додаток має практичне значення для навчання, самоосвіти та зручної роботи з електронними ресурсами.

Завдання дослідження



Основні можливості системи

- Перегляд каталогу книг
- Пошук за назвою та автором
- Фільтрація та сортування
- Перегляд сторінки книги
- Завантаження PDF-файлів
- Реєстрація та авторизація
- Особистий кабінет користувача

ТЕХНОЛОГІЇ

Backend:

- Python – основна мова програмування
- Flask – обробка HTTP-запитів та серверна логіка
- Jinja2 – генерація HTML-сторінок

База Даних:

- PostgreSQL – зберігання книг, користувачів, категорій
- psycopg (psycopg2) – взаємодія з БД та SQL-запити

Frontend:

- HTML – структура сторінок
- CSS – оформлення інтерфейсу
- JAVASCRIPT – обробка подій та параметрів пошуку

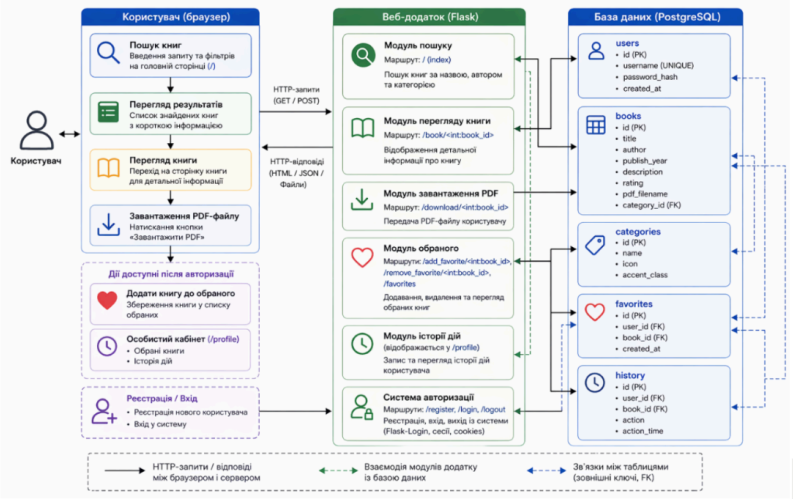
Авторизація та безпека

- Flask-Login – керування сесіями користувачів
- Werkzeug – хешування паролів

Реалізація функціоналу

- Пошук – SQL LIKE
- Фільтрація – WHERE
- Сортування – ORDER BY
- PDF – send_from_directory

АРХІТЕКТУРА



АРХІТЕКТУРА

-**Веб-додаток** «Онлайн бібліотека» побудований на основі клієнт-серверної архітектури з використанням серверного рендерингу.

-**Клієнтська частина** представлена веб-інтерфейсом, який працює у браузері та відповідає за відображення даних і взаємодію з користувачем. Усі дії користувача, такі як пошук, фільтрація або перехід між сторінками, формують HTTP-запити до серверної частини.

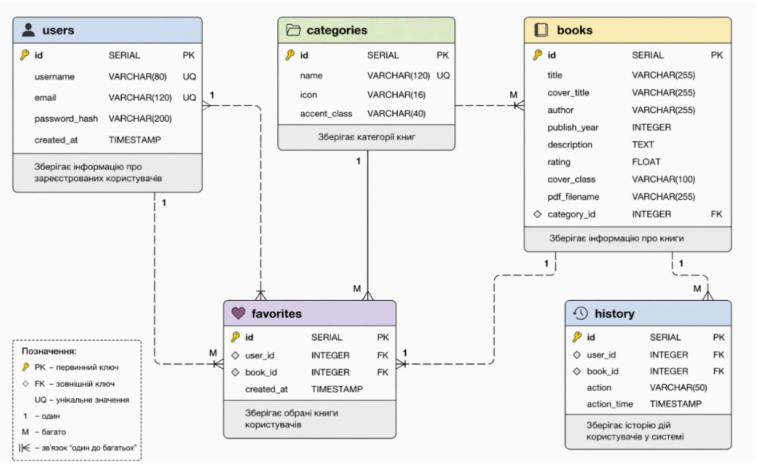
-**Серверна частина** реалізована за допомогою Flask і відповідає за обробку запитів, виконання бізнес-логіки та взаємодію з базою даних. Обробка даних включає формування SQL-запитів, отримання результатів і передачу їх у шаблони.

-**Формування веб-сторінок** здійснюється на сервері за допомогою шаблонізатора Jinja, що дозволяє генерувати HTML-контент динамічно залежно від запитів користувача.

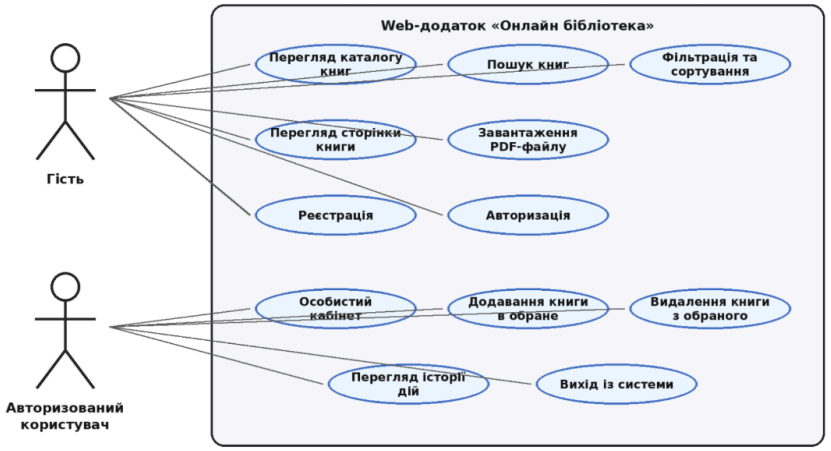
-**Зберігання** даних реалізовано за допомогою реляційної бази даних PostgreSQL, яка містить інформацію про книги, користувачів, категорії та історію дій.

-**Взаємодія між компонентами системи** здійснюється через HTTP-запити без використання окремого API, що спрощує структуру додатку та відповідає обраній архітектурі.

База даних тези прибрати



Діаграма варіантів використання



Діаграма варіантів використання

-**Діаграма варіантів використання** відображає основні сценарії взаємодії користувачів із веб-додатком «Онлайн бібліотека». У системі передбачено два типи користувачів: гість та авторизований користувач.

-**Гість** має можливість переглядати каталог книг, виконувати пошук, застосовувати фільтрацію і сортування, переглядати сторінки книг, завантажувати PDF-файли, а також проходити реєстрацію та авторизацію.

-**Авторизований користувач** отримує додатковий функціонал: доступ до особистого кабінету, перегляд історії дій, додавання та видалення книг з обраного, а також вихід із системи. Діаграма демонструє основні функціональні можливості веб-додатку та взаємодію користувачів із системою.

Перспективи подальших досліджень

-У межах подальшого розвитку веб-додатку «Онлайн бібліотека» можливе розширення його функціональних можливостей та підвищення ефективності роботи.

-Перспективними напрямами є впровадження рекомендаційної системи для підбору книг на основі інтересів користувача, оптимізація пошуку за допомогою більш складних алгоритмів, а також розширення бази даних інформаційних ресурсів.

-Окрім цього, можливим є створення веб-інтерфейсу адміністратора для управління контентом, інтеграція з зовнішніми сервісами та підвищення рівня безпеки системи.

Висновки

У ході виконання кваліфікаційної роботи було розроблено веб-додаток «Онлайн бібліотека», який забезпечує зберігання, обробку, пошук, фільтрацію та перегляд інформації про книги, а також можливість завантаження PDF-файлів.

Під час процесу дослідження було проаналізовано сучасні підходи до розробки веб-додатків та онлайн-бібліотек, а також існуючі технології роботи з інформаційними ресурсами. Це дозволило визначити основні принципи побудови систем такого типу та сформувати вимоги до майбутнього програмного продукту.

Було визначено функціональні та нефункціональні вимоги до веб-додатку «Онлайн бібліотека», зокрема вимоги щодо зручності використання, швидкодії, стабільності функціонування та організації пошуку й перегляду інформації.

Для реалізації системи було обрано та обґрунтовано технологічний стек, який включає Python, Flask, PostgreSQL, HTML, CSS та JavaScript. Використання зазначених технологій дозволило реалізувати клієнт-серверну архітектуру, забезпечити взаємодію між користувацьким інтерфейсом, серверною логікою та базою даних.

У практичній частині реалізовано структуру бази даних для зберігання інформації про книги, користувачів та їх дії, а також основні функціональні модулі системи, включаючи систему авторизації, особистий кабінет користувача, механізми пошуку, фільтрації, сортування та перегляду PDF-файлів.

Проведене тестування підтвердило коректність роботи веб-додатку, стабільність його функціонування та відповідність поставленим вимогам, а також дозволило оцінити ефективність реалізованого рішення.

Розроблений веб-додаток має практичну цінність у сфері інформаційних систем і може бути використаний як основа для подальшого розвитку, зокрема розширення функціональності, удосконалення механізмів пошуку та покращення взаємодії з користувачем.

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Робота пройшла апробацію. За її результатами було опубліковано наступні тези доповідей:

1. Антонов О.Л., Аронов А.О. Реалізація системи авторизації та взаємодії користувача у веб-додатку «Онлайн бібліотека». // Матеріали Всеукраїнської науково-технічної конференції «Технології цифрового розвитку: програмні системи та децентралізація». – ДУІКТ, Київ, Україна, 2026. – (подано до друку).

2. Антонов О.Л., Аронов А.О. Аналіз реалізації пошуку книжок у веб-додатку онлайн бібліотека. // Матеріали VII Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях». – ДУІКТ, Київ, Україна, 2026, С. 48-49

ДОДАТОК Б. ЛІСТИНГ ОСНОВНИХ МОДУЛІВ

```
import os
import time
from contextlib import closing
from flask import Flask, render_template, request, redirect, url_for, flash, send_from_directory, abort
from flask_login import LoginManager, UserMixin, login_user, logout_user, login_required, current_user
import psycopg
from psycopg.rows import dict_row
from werkzeug.security import generate_password_hash, check_password_hash

app = Flask(__name__)
app.secret_key = os.getenv("SECRET_KEY", "change-me-in-env")

BASE_DIR = os.path.dirname(os.path.abspath(__file__))
PDF_DIR = os.path.join(BASE_DIR, "static", "pdfs")

login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = "login"


@app.context_processor
def inject_user():
    return dict(current_user=current_user)


def get_connection():
    return psycopg.connect(
        host=os.getenv("PGHOST", "db"),
        port=os.getenv("PGPORT", "5432"),
        dbname=os.getenv("PGDATABASE", "online_library"),
        user=os.getenv("PGUSER", "postgres"),
        password=os.getenv("PGPASSWORD", "postgres"),
```

```
        row_factory=dict_row,
    )


class User(UserMixin):
    def __init__(self, id, username, email):
        self.id = id
        self.username = username
        self.email = email


@login_manager.user_loader
def load_user(user_id):
    try:
        with closing(get_connection()) as conn, conn.cursor() as cur:
            cur.execute("SELECT id, username, email FROM users WHERE id = %s", (user_id,))
            user = cur.fetchone()
            if user:
                return User(user["id"], user["username"], user["email"])
    except Exception:
        pass
    return None


def safe_execute(statement, params=None, fetch="all"):
    with closing(get_connection()) as conn, conn.cursor() as cur:
        cur.execute(statement, params or ())
        if fetch == "one":
            return cur.fetchone()
        if fetch == "none":
            conn.commit()
            return None
        return cur.fetchall()
```

```
def log_action(action, book_id=None):
    if not current_user.is_authenticated:
        return
    try:
        with closing(get_connection()) as conn, conn.cursor() as cur:
            cur.execute(
                """
                INSERT INTO history (user_id, book_id, action)
                VALUES (%s, %s, %s)
                """,
                (current_user.id, book_id, action),
            )
            conn.commit()
    except Exception:
        pass


def fetch_categories():
    return safe_execute(
        """
        SELECT c.id, c.name, c.description, COUNT(b.id) AS book_count
        FROM categories AS c
        LEFT JOIN books AS b ON b.category_id = c.id
        GROUP BY c.id, c.name, c.description
        ORDER BY c.name
        """
    )


def fetch_library_stats():
    return safe_execute(
        """
        SELECT COUNT(*) AS total_books,
               COUNT(DISTINCT author) AS total_authors,
               COALESCE(ROUND(AVG(rating), 1), 0) AS average_rating
```

```
        FROM books
        """,
        fetch="one",
    )


def fetch_featured_books():
    return safe_execute(
        """
        SELECT b.id, b.title, b.author, b.year, b.rating, c.name AS category_name
        FROM books b
        JOIN categories c ON c.id = b.category_id
        ORDER BY b.rating DESC, b.title ASC
        LIMIT 3
        """
    )


def search_books(title, author, keyword, category, sort_by="rating_desc"):
    order_map = {
        "rating_desc": "b.rating DESC, b.title ASC",
        "year_desc": "b.year DESC, b.title ASC",
        "year_asc": "b.year ASC, b.title ASC",
        "title_asc": "b.title ASC",
        "title_desc": "b.title DESC",
    }
    order_clause = order_map.get(sort_by, order_map["rating_desc"])

    query = f"""
        SELECT
            b.id,
            b.title,
            b.author,
            b.year,
            b.rating,
```

```
            b.description,
            b.pdf_filename,
            c.name AS category_name
        FROM books b
        JOIN categories c ON c.id = b.category_id
        WHERE (%(title)s = '' OR b.title ILIKE '%%' || %(title)s || '%%')
          AND (%(author)s = '' OR b.author ILIKE '%%' || %(author)s || '%%')
          AND (%(category)s = '' OR c.name = %(category)s)
          AND (%(keyword)s = '' OR b.title ILIKE '%%' || %(keyword)s || '%%'
               OR b.author ILIKE '%%' || %(keyword)s || '%%'
               OR c.name ILIKE '%%' || %(keyword)s || '%%'
               OR COALESCE(b.description, '') ILIKE '%%' || %(keyword)s || '%%')
        ORDER BY {order_clause}
    """

    params = {
        "title": title.strip(),
        "author": author.strip(),
        "keyword": keyword.strip(),
        "category": category.strip(),
    }

    with closing(get_connection()) as conn, conn.cursor() as cur:
        cur.execute(query, params)
        return cur.fetchall()


def get_book(book_id):
    return safe_execute(
        """
        SELECT
            b.id,
            b.title,
            b.author,
            b.year,
```

```
            b.rating,
            b.description,
            b.pdf_filename,
            c.name AS category_name
        FROM books b
        JOIN categories c ON c.id = b.category_id
        WHERE b.id = %s
        """,
        (book_id,),
        fetch="one",
    )


@app.route("/")
def index():
    title = request.args.get("title", "")
    author = request.args.get("author", "")
    keyword = request.args.get("q", "")
    category = request.args.get("category", "")
    sort_by = request.args.get("sort", "rating_desc")
    db_error = None

    try:
        books = search_books(title, author, keyword, category, sort_by)
        categories = fetch_categories()
        stats = fetch_library_stats()
        featured_books = fetch_featured_books()
        summary = f"Found {len(books)} books" if books else "No books found"

        if title or author or keyword or category:
            log_action("search")
    except Exception as exc:
        books = []
        categories = []
        stats = {"total_books": 0, "total_authors": 0, "average_rating": 0}
```

```
        featured_books = []
        summary = "Database connection error"
        db_error = str(exc)

    return render_template(
        "index.html",
        books=books,
        categories=categories,
        title_query=title,
        author_query=author,
        keyword_query=keyword,
        category_query=category,
        sort_query=sort_by,
        summary=summary,
        stats=stats,
        featured_books=featured_books,
        db_error=db_error,
    )


@app.route("/book/<int:book_id>")
def book_detail(book_id):
    book = get_book(book_id)
    if not book:
        abort(404)
    log_action("view_book", book_id)
    return render_template("book_detail.html", book=book)


@app.route("/download/<int:book_id>")
def download_book(book_id):
    book = get_book(book_id)
    if not book or not book.get("pdf_filename"):
        abort(404)
```

```
    pdf_filename = book["pdf_filename"]
    pdf_path = os.path.join(PDF_DIR, pdf_filename)
    if not os.path.exists(pdf_path):
        flash("PDF file is missing for this book.", "error")
        return redirect(url_for("book_detail", book_id=book_id))

    log_action("download_pdf", book_id)
    return send_from_directory(PDF_DIR, pdf_filename, as_attachment=True)


@app.route("/register", methods=["GET", "POST"])
def register():
    if current_user.is_authenticated:
        return redirect(url_for("index"))

    if request.method == "POST":
        username = request.form.get("username", "").strip()
        email = request.form.get("email", "").strip()
        password = request.form.get("password", "")
        confirm = request.form.get("confirm_password", "")

        if not username or not email or not password:
            flash("All fields are required.", "error")
        elif password != confirm:
            flash("Passwords do not match.", "error")
        elif len(password) < 6:
            flash("Password must be at least 6 characters long.", "error")
        else:
            try:
                hashed = generate_password_hash(password)
                with closing(get_connection()) as conn, conn.cursor() as cur:
                    cur.execute(
                        "INSERT INTO users (username, email, password_hash) VALUES (%s, %s, %s)",
                        (username, email, hashed),
                    )
```

```
                conn.commit()
            flash("Registration successful. Please sign in.", "success")
            return redirect(url_for("login"))
        except psycopg.Error as exc:
            if "unique" in str(exc).lower():
                flash("A user with this username or email already exists.", "error")
            else:
                flash(f"Database error: {exc}", "error")

    return render_template("register.html")


@app.route("/login", methods=["GET", "POST"])
def login():
    if current_user.is_authenticated:
        return redirect(url_for("index"))

    if request.method == "POST":
        username = request.form.get("username", "").strip()
        password = request.form.get("password", "")
        remember = bool(request.form.get("remember"))

        try:
            with closing(get_connection()) as conn, conn.cursor() as cur:
                cur.execute(
                    "SELECT id, username, email, password_hash FROM users WHERE username = %s",
                    (username,),
                )
                user = cur.fetchone()

                if user and check_password_hash(user["password_hash"], password):
                    login_user(User(user["id"], user["username"], user["email"]), remember=remember)
                    log_action("login")
                    flash(f"Welcome back, {user['username']}!", "success")
                    return redirect(url_for("index"))
```

```
            flash("Invalid username or password.", "error")
        except Exception as exc:
            flash(f"Database error: {exc}", "error")

    return render_template("login.html")


@app.route("/logout", methods=["POST"])
@login_required
def logout():
    logout_user()
    flash("You have logged out.", "info")
    return redirect(url_for("index") + "?t=" + str(int(time.time())))


@app.route("/profile")
@login_required
def profile():
    try:
        with closing(get_connection()) as conn, conn.cursor() as cur:
            cur.execute(
                """
                SELECT b.id, b.title, b.author, b.year, b.rating, b.description, b.pdf_filename,
                       c.name AS category_name
                FROM favorites f
                JOIN books b ON b.id = f.book_id
                JOIN categories c ON c.id = b.category_id
                WHERE f.user_id = %s
                ORDER BY f.created_at DESC
                """,
                (current_user.id,),
            )
            favorites = cur.fetchall()
```

```
            cur.execute(
                """
                SELECT COUNT(*) AS total_favorites,
                       COALESCE(ROUND(AVG(b.rating), 1), 0) AS avg_rating
                FROM favorites f
                JOIN books b ON b.id = f.book_id
                WHERE f.user_id = %s
                """,
                (current_user.id,),
            )
            stats = cur.fetchone()

            cur.execute(
                """
                SELECT h.action, h.action_time, b.title
                FROM history h
                LEFT JOIN books b ON b.id = h.book_id
                WHERE h.user_id = %s
                ORDER BY h.action_time DESC
                LIMIT 8
                """,
                (current_user.id,),
            )
            history = cur.fetchall()
    except Exception:
        favorites = []
        stats = {"total_favorites": 0, "avg_rating": 0}
        history = []

    return render_template("profile.html", favorites=favorites, stats=stats, history=history)


@app.route("/add_favorite/<int:book_id>", methods=["POST"])
@login_required
def add_favorite(book_id):
```

```
    try:
        with closing(get_connection()) as conn, conn.cursor() as cur:
            cur.execute(
                "INSERT INTO favorites (user_id, book_id) VALUES (%s, %s)",
                (current_user.id, book_id),
            )
            conn.commit()
        log_action("add_favorite", book_id)
        flash("Book added to favorites.", "success")
    except Exception:
        flash("This book is already in your favorites.", "info")
    return redirect(request.referrer or url_for("index"))


@app.route("/remove_favorite/<int:book_id>", methods=["POST"])
@login_required
def remove_favorite(book_id):
    with closing(get_connection()) as conn, conn.cursor() as cur:
        cur.execute(
            "DELETE FROM favorites WHERE user_id = %s AND book_id = %s",
            (current_user.id, book_id),
        )
        conn.commit()
    log_action("remove_favorite", book_id)
    flash("Book removed from favorites.", "info")
    return redirect(request.referrer or url_for("profile"))


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000, debug=True)


import os
import psycopg
```

```
DB_CONFIG = {
    "host": os.getenv("PGHOST", "db"),
    "port": os.getenv("PGPORT", "5432"),
    "dbname": os.getenv("PGDATABASE", "online_library"),
    "user": os.getenv("PGUSER", "postgres"),
    "password": os.getenv("PGPASSWORD", "postgres"),
}

try:
    conn = psycopg.connect(**DB_CONFIG)
    conn.autocommit = True
    cur = conn.cursor()

    print("Створення таблиці categories...")
    cur.execute(
        """
        CREATE TABLE IF NOT EXISTS categories (
            id SERIAL PRIMARY KEY,
            name VARCHAR(100) NOT NULL,
            description TEXT
        )
        """
    )

    print("Створення таблиці books...")
    cur.execute(
        """
        CREATE TABLE IF NOT EXISTS books (
            id SERIAL PRIMARY KEY,
            title VARCHAR(255) NOT NULL,
            author VARCHAR(255) NOT NULL,
            year INTEGER,
            description TEXT,
            rating NUMERIC(3,1) DEFAULT 0,
            category_id INTEGER REFERENCES categories(id) ON DELETE SET NULL,
```

```
            pdf_filename VARCHAR(255)
        )
        """
    )

    print("Створення таблиці history...")
    cur.execute(
        """
        CREATE TABLE IF NOT EXISTS history (
            id SERIAL PRIMARY KEY,
            user_id INTEGER,
            book_id INTEGER REFERENCES books(id) ON DELETE SET NULL,
            action VARCHAR(50) NOT NULL,
            action_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP
        )
        """
    )

    print("Створення індексів...")
    cur.execute("CREATE INDEX IF NOT EXISTS idx_books_category_id ON books(category_id)")
    cur.execute("CREATE INDEX IF NOT EXISTS idx_history_book_id ON history(book_id)")
      cur.execute("CREATE INDEX IF NOT EXISTS idx_history_action_time ON history(action_time
DESC)")

    print("Додавання початкових категорій...")
    cur.execute(
        """
        INSERT INTO categories (name, description)
        VALUES
            ('Fiction', 'Художня література'),
            ('Science', 'Наукова та освітня література'),
            ('History', 'Історична література')
        ON CONFLICT DO NOTHING
        """
    )
```

```
    print("Додавання тестових книг...")
    cur.execute(
        """
        INSERT INTO books (title, author, year, description, rating, category_id, pdf_filename)
            VALUES ('1984', 'George Orwell', 1949, 'Антиутопічний роман про тоталітарне суспільство та систему спостереження.', 4.8,
             1, '1984.pdf'),
            ('Atomic Habits', 'James Clear', 2018,
             'Практичний посібник із формування корисних звичок та позбавлення шкідливих.', 4.9,
2, 'atomic_habits.pdf'),
            ('Harry Potter and the Philosopher''s Stone', 'J. K. Rowling', 1997,
             'Перша книга серії про Гаррі Поттера.', 4.8, 1, 'harry_potter_1.pdf'),
                ('Kobzar', 'Taras Shevchenko', 1840, 'Класична збірка української поезії.', 4.9, 3,
'kobzar.pdf'),
            ('The Little Prince', 'Antoine de Saint-Exupery', 1943,
             'Філософська історія про дитинство, любов і людяність.', 4.8, 1, 'little_prince.pdf'),
            ('Sapiens', 'Yuval Noah Harari', 2011, 'Книга про історію людства.', 4.8, 3, 'sapiens.pdf')
        """
    )

    print("✅ Основні таблиці успішно створено!")
    cur.close()
    conn.close()

except Exception as e:
    print(f"❌ Помилка в init_db.py: {e}")

(function () {
  const heroForm = document.querySelector('.hero-search');
  const headerForm = document.querySelector('.header-search');
  const titleInput = document.querySelector('input[name="title"]');
  const authorInput = document.querySelector('input[name="author"]');
  const categoryInput = document.querySelector('select[name="category"]');
  const sortInput = document.querySelector('select[name="sort"]');
```

```
const keywordInput = document.querySelector('.header-search input[name="q"]');
const clearButton = document.querySelector('.secondary-button');

if (!heroForm || !headerForm || !titleInput || !authorInput || !categoryInput || !keywordInput) {
  return;
}

function buildUrl() {
  const params = new URLSearchParams();
  const title = titleInput.value.trim();
  const author = authorInput.value.trim();
  const category = categoryInput.value.trim();
  const keyword = keywordInput.value.trim();
  const sort = sortInput ? sortInput.value.trim() : '';

  if (keyword) params.set('q', keyword);
  if (title) params.set('title', title);
  if (author) params.set('author', author);
  if (category) params.set('category', category);
  if (sort) params.set('sort', sort);

  const query = params.toString();
  return query ? `/?${query}` : '/';
}

function submitSearch(event) {
  if (event) event.preventDefault();
  window.location.href = buildUrl();
}

heroForm.addEventListener('submit', submitSearch);
headerForm.addEventListener('submit', submitSearch);

if (clearButton) {
  clearButton.addEventListener('click', function (event) {
```

```
    event.preventDefault();
    titleInput.value = '';
    authorInput.value = '';
    categoryInput.value = '';
    if (sortInput) sortInput.value = 'rating_desc';
    keywordInput.value = '';
    window.location.href = '/';
  });
 }
})();
```