

БАКАЛАВРСЬКА РОБОТА

БР. ІІ - 18.00.00.000 ІІЗ

Група ІІ-22-1

Чемна Оксана

2026

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Чемна Оксана Іванівна

(прізвище, ім'я, по батькові)

УДК 004.942
(індекс)

БАКАЛАВРСЬКА РОБОТА

Розробка АБІС для закладів загальної середньої освіти засобами React та
ASP.NET

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121– Інженерія програмного забезпечення

(шифр і назва спеціальності)

**Робота містить результати власних досліджень, використання ідей, результатів і
текстів інших авторів мають посилання на відповідне джерело:**

Здобувач освітнього ступеня **О. І. Чемна**

(підпис, ініціали та прізвище здобувача)

Науковий керівник **Мельник Віталій Дмитрович, к.т.н., доцент**

(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту

Завідувач кафедри

доц. Бандура В. В.

(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ – 2026

Івано-Франківський національний технічний університет нафти і газу
Інститут, факультет інформаційних технологій
Кафедра інженерії програмного забезпечення
Ступінь вищої освіти бакалавр
Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедрою ІПЗ

доц. В.В. Бандура

“ ” 2026 р.

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Чемній Оксані Іванівні

(прізвище, ім'я, по-батькові)

1. Тема проекту (роботи) “Розробка АБІС для закладів загальної середньої освіти засобами React та ASP.NET”

керівник проекту (роботи) Мельник Віталій Дмитрович, к.т.н., доцент

затверджені наказом закладу вищої освіти від “21” квітня 2026 р. № 179/7

2. Строк подання студентом проекту(роботи) червня 2026 р.

3. Вихідні дані до проекту (роботи) Результати і матеріали отримані під час проходження переддипломної практики

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

1. Огляд існуючих рішень та теоретичних основ проєктування автоматизованої бібліотечної інформаційної системи

2. Аналіз вимог та постановка задачі

3. Проєктування системи

4. Реалізація продукту

5. Тестування та впровадження

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

1. Зв’язок між бібліографічним записом і примірниками (рис. 1.1, ст. 18)

2. UML-діаграма розгортання системи BookCatalog (рис. 3.1, ст. 40)

3. UML-діаграма варіантів використання системи BookCatalog (рис. 3.2, ст. 42)

4. UML-діаграма діяльності процесу приєднання користувача (рис. 3.3, ст. 43)

5. UML-діаграма станів книги (рис. 3.4, ст. 44)

1. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

2. Дата видачі завдання _____

Керівник

(підпис) (розшифровка підпису)

Завдання прийняв до виконання

(підпис) (розшифровка підпису)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Визначення та обґрунтування теми роботи	15.01.2026	виконано
2	Огляд існуючих концепцій, рішень та сервісів в даній області	15.02.2026	виконано
3	Побудова моделі або алгоритму власного рішення	03.03.2026	виконано
4	Документування реалізації власного оригінального рішення вибраними засобами	12.05.2026	виконано
	Оформлення пояснювальної записки бакалаврської роботи	04.06.2026	виконано

Студент

(підпис) (розшифровка підпису)

Керівник роботи

(підпис) (розшифровка підпису)

АНОТАЦІЯ

Бакалаврська робота містить 81 сторінку, 48 рисунків, список використаних джерел із 39 найменування, 2 додатки.

Метою роботи є створення автоматизованої бібліотечної інформаційної системи (АБІС) для шкільних бібліотек.

Об'єктом дослідження є процеси організації та обліку книжкового фонду бібліотеки закладу загальної середньої освіти.

Предметом дослідження є методи, засоби та програмні компоненти розробки АБІС, зокрема її архітектура, модель даних, серверна логіка та клієнтський інтерфейс.

Результати дослідження: створено АБІС для цифровізації основних процесів роботи шкільної бібліотеки засобами React та ASP.NET Core.

У першому розділі проаналізовано сучасний стан автоматизації бібліотек, основні поняття предметної області та наявні програмні рішення.

У другому розділі сформовано вимоги та визначено задачу проектування.

У третьому розділі описано проектування системи.

У четвертому розділі подано реалізацію серверної та клієнтської частин.

У п'ятому розділі розглянуто тестування системи та впровадження.

Висновок: розроблена система забезпечує електронний облік книжкового фонду, підручників, читачів, документів і міжбібліотечний обмін, що сприяє зменшенню ручної роботи бібліотекаря та підвищенню точності бібліотечного обліку.

КЛЮЧОВІ СЛОВА: АВТОМАТИЗОВАНА БІБЛІОТЕЧНА ІНФОРМАЦІЙНА СИСТЕМА, БІБЛІОТЕКА, КНИЖКОВИЙ ФОНД, ЕЛЕКТРОННИЙ КАТАЛОГ, ОБЛІК КНИГ, ПІДРУЧНИКИ, БІБЛІОГРАФІЧНИЙ ОПИС, ЗВІТНІСТЬ, REST API, REACT, ASP.NET CORE, MICROSOFT SQL SERVER.

ANNOTATION

The bachelor's thesis contains 81 pages, 48 figures, a list of 39 references, and 2 appendices.

The purpose of the work is to create an automated library information system (ALIS) for school libraries.

The object of the research is the processes of organization and accounting of the book collection of a library in a general secondary education institution.

The subject of the research is the methods, tools, and software components for developing an ALIS, in particular its architecture, data model, server logic, and user interface.

Research results: an ALIS was created to digitalize the main processes of a school library using React and ASP.NET Core.

The first chapter analyzes the current state of library automation, the main concepts of the subject area, and existing software solutions.

The second chapter defines the requirements and formulates the design task.

The third chapter describes the system design.

The fourth chapter presents the implementation of the server and client parts.

The fifth chapter considers system testing and implementation.

Conclusion: the developed system provides electronic accounting of the book collection, textbooks, readers, documents, and interlibrary exchange, which helps reduce the librarian's manual work and improve the accuracy of library accounting.

KEYWORDS: AUTOMATED LIBRARY INFORMATION SYSTEM, LIBRARY, BOOK COLLECTION, ELECTRONIC CATALOG, BOOK ACCOUNTING, TEXTBOOKS, BIBLIOGRAPHIC DESCRIPTION, REPORTING, REST API, REACT, ASP.NET CORE, MICROSOFT SQL SERVER.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	8
ВСТУП	9
РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ТА ТЕОРЕТИЧНИХ ОСНОВ ПРОЄКТУВАННЯ АВТОМАТИЗОВАНОЇ БІБЛІОТЕЧНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ	12
1.1 Аналіз сучасного стану проблематики.....	12
1.2 Основні поняття та призначення інженерії ПЗ в даному домені	16
1.3 Сучасні технології та тенденції, що застосовують в даній області.....	200
1.4 Висновки по розділу.....	233
РОЗДІЛ 2. АНАЛІЗ ВИМОГ ТА ПОСТАНОВКА ЗАДАЧІ	25
2.1 Збір та аналіз вимог користувачів.....	25
2.2 Формування функціональних та нефункціональних вимог	28
2.3 Постановка задачі проєктування системи.....	32
2.4 Висновки по розділу.....	3434
РОЗДІЛ 3. ПРОЄКТУВАННЯ СИСТЕМИ	36
3.1 Розробка архітектури програмного забезпечення.....	36
3.2 Моделювання бізнес-процесів та системних компонентів	39
3.3 Вибір та обґрунтування технологій та інструментів розробки	51
3.4 Висновки по розділу.....	54
РОЗДІЛ 4. РЕАЛІЗАЦІЯ ПРОДУКТУ	55
4.1 Опис розробки програмного коду.....	55
4.2 Використані алгоритми та структури даних.....	59
4.3 Модульне тестування та налагодження	67
4.4 Висновки по розділу.....	70

					БР.ІІ – 18.00.00.000 ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Розробка АБІС для закладів загальної середньої освіти засобами React та ASP.NET Пояснювальна записка	Літ.	Арк.	Акрушів
Розроб.		Чемна О. І.						
Перевір.		Мельник В. Д.					6	101
Реценз.		Ваврик Т.О.				ІФНТУНГ ІІ-22-1		
Н. Контр.		Піх М.М.						
Затверд.		Бандура В. В.						

РОЗДІЛ 5. ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ	71
5.1 Стратегії та методи тестування.....	71
5.2 Результати тестування системи	73
5.3 Аналіз ефективності впровадження	76
5.4 Висновки по розділу	78
ВИСНОВКИ	80
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	Помилка! Закладку не визначено.
ДОДАТКИ	
БІБЛІОГРАФІЧНА ДОВІДКА	

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		7

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

АБІС — автоматизована бібліотечна інформаційна система

БД — база даних

ПЗ — програмне забезпечення

СУБД — система управління базами даних

УДК — універсальна десяткова класифікація

ISBN — International Standard Book Number

API — Application Programming Interface

REST — Representational State Transfer

HTTP — HyperText Transfer Protocol

JSON — JavaScript Object Notation

JWT — JSON Web Token

DTO — Data Transfer Object

CRUD — Create, Read, Update, Delete

SQL — Structured Query Language

ORM — Object-Relational Mapping

LINQ — Language Integrated Query

CORS — Cross-Origin Resource Sharing

SMTP — Simple Mail Transfer Protocol

SPA — Single Page Application

HTML — HyperText Markup Language

CSS — Cascading Style Sheets

JSX — JavaScript XML

DOM — Document Object Model

DI — Dependency Injection

E2E — End-to-End

XML — Extensible Markup Language

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

В умовах цифрової трансформації освітнього середовища значення набуває й автоматизація інформаційних процесів у закладах загальної середньої освіти.

Одним з таких процесів є організація роботи бібліотеки, яка виконує не лише функцію зберігання та видачі навчальної, художньої й довідкової літератури, а й забезпечує облік книжкового фонду, систематизацію видань, формування бібліографічних описів, підготовку звітної документації та підтримку інформаційних потреб учнів та педагогічних працівників.

Традиційне ведення бібліотечного обліку за допомогою паперових журналів або окремих електронних таблиць має низку недоліків. До них належать складність швидкого пошуку потрібної інформації, ризик дублювання записів, імовірність помилок під час ручного введення даних, відсутність централізованого контролю доступу, ускладнене формування звітів і недостатня гнучкість під час роботи з великим обсягом бібліотечних одиниць. Особливо відчутними ці проблеми стають у випадках, коли потрібно оперативно отримати інформацію про книгу, її інвентарний номер, авторів, редакторів, накладну, вартість, рік видання, УДК або чи списана вона.

Автоматизована бібліотечна інформаційна система дає змогу вирішити зазначені проблеми шляхом централізованого зберігання даних, структурного обліку фонду, автоматизації типових операцій, розмежування прав доступу користувачів і формування документів.

Актуальність теми зумовлена потребою закладів загальної середньої освіти у доступних, адаптованих і зручних програмних рішеннях для автоматизації бібліотечної діяльності. Багато наявних бібліотечних систем орієнтовані на великі бібліотеки або мають надмірно складний функціонал для невеликих освітніх установ. Тому доцільною є розробка вебзастосунку, який враховує специфіку шкільної бібліотеки та забезпечує основні процеси, з якими бібліотекар працює щодня.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

Метою роботи є розробка автоматизованої бібліотечної інформаційної системи для закладів загальної середньої освіти засобами React та ASP.NET Core, яка забезпечує електронний облік книжкового фонду, керування бібліотеками й користувачами, роботу з підручниками, формування документів і підтримку основних бібліотечних процесів.

Завданнями дослідження є аналіз предметної області шкільних бібліотек та існуючих програмних рішень, формування функціональних та нефункціональних вимог до автоматизованої бібліотечної інформаційної системи, проєктування архітектури вебзастосунку, структури бази даних і програмних інтерфейсів, реалізація серверної та клієнтської частини системи, забезпечення роботи з книжковим фондом, підручниками, читачами, бібліотеками, документами, а також проведення тестування та впровадження системи.

Об'єктом дослідження є процеси організації, обліку та управління книжковим фондом бібліотеки закладу загальної середньої освіти.

Предметом дослідження є методи, засоби та програмні компоненти розробки АБІС, зокрема її архітектура, модель даних, серверна логіка, REST API, клієнтський інтерфейс і механізми керування доступом.

Методами дослідження у роботі є аналіз предметної області та існуючих програмних рішень, порівняння підходів до автоматизації бібліотечних процесів, моделювання структури даних, об'єктно-орієнтоване проєктування, проєктування REST API, методи реляційного моделювання бази даних, клієнт-серверне проєктування, а також тестування функціональних можливостей розробленої системи.

Наукова новизна дослідження полягає у розробленні адаптованої АБІС, орієнтованої саме на потреби закладів загальної середньої освіти України. На відміну від універсальних бібліотечних платформ, запропоноване рішення зосереджене на практичних процесах шкільної бібліотеки. Тобто інвентарному обліку, роботі з накладними, списанні та відновленні книг, формуванні

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

бібліографічних описів, обліку підручників, створенні документів і керуванні доступом користувачів у межах конкретної бібліотеки.

Практичне значення роботи полягає у створенні програмного продукту, який може бути використаний для цифровізації основних процесів роботи бібліотеки закладу загальної середньої освіти. Розроблена система дасть змогу централізовано зберігати дані про книжковий фонд, швидше знаходити й редагувати інформацію про книги, автоматизувати облік надходжень, підручників і списаних видань, а також спростити формування бібліографічних описів і звітних документів. Використання системи зменшить обсяг ручної роботи бібліотекаря, знизить ймовірність помилок та створить єдине цифрове середовище для управління бібліотечними ресурсами.

Бакалаврська робота містить 81 сторінку, 48 рисунків, 5 розділів, список використаних джерел із 39 найменувань, 2 додатки.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ТА ТЕОРЕТИЧНИХ ОСНОВ ПРОЄКТУВАННЯ АВТОМАТИЗОВАНОЇ БІБЛІОТЕЧНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

1.1 Аналіз сучасного стану проблематики

Шкільна бібліотека є обов'язковим структурним підрозділом закладу загальної середньої освіти [1]. Вона забезпечує користувачів навчальною, художньою, довідковою та методичною літературою, веде облік фонду, оформлює надходження та вибуття документів, готує звітні матеріали і підтримує освітній процес. Відповідно до п. 1.6 Положення про бібліотеку загальноосвітнього навчального закладу, її користувачами є учні, педагоги, працівники закладу та батьки [1].

Бібліотека ліцею чи гімназії відрізняється від публічних, університетських чи наукових тим, що функціонує в межах одного закладу освіти та обслуговує чітко визначені групи користувачів. Вона не тільки надає доступ до художньої, довідкової та методичної літератури, а ще й займається постійним обліком підручників для забезпеченості класів та вчителів. Саме тому автоматизована інформаційна система для такої бібліотеки має враховувати специфіку освітнього процесу, а не лише зберігати перелік бібліографічних записів.

Дані щодо освітянської бібліотечної мережі підтверджують потребу в системному підході до автоматизації. Узагальнені кількісні показники наведено в таблиці 1.1 [2].

Таблиця 1.1

Кількісні показники освітянських бібліотек України

Показник	Значення
Освітянські бібліотеки	18 066
Бібліотеки загальноосвітніх навчальних закладів	близько 15 000
Сукупний фонд освітянських бібліотек	333,2 млн документів
Користувачі	7,7 млн
Відвідування	майже 202 млн
Річна книговидача	49 млн документів

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

Наведені показники демонструють значний обсяг бібліотечних фондів, користувачів і щорічних операцій з документами.

Сьогодні майже всі шкільні бібліотеки ведуть ручний облік у паперових інвентарних книгах, читацьких формулярах, картотеках або окремих електронних таблицях. І такий підхід має низку практичних недоліків. Зокрема, дублювання записів, помилки під час введення даних, складний пошук та відсутність контролю змін. Крім того, паперові й розрізнені електронні форми не гарантують збереження інформації у разі втрати документа або пошкодження файлу.

Трудомісткість ручного обліку підтверджують й Типові норми часу на основні процеси бібліотечної роботи. У документі визначено нормативну тривалість окремих операцій, пов'язаних з обліком фонду, каталогізацією, статистичною обробкою та довідково-бібліографічним обслуговуванням [3]. Найбільш показові для обґрунтування потреби в автоматизації процеси наведено в таблиці 1.2.

Таблиця 1.2

Норми часу на окремі процеси бібліотечної роботи

Процес	Одиниця виміру	Норма часу
Оформлення картки індивідуального обліку	Один документ	5 хв
Запис до інвентарної книги	Один запис	3 хв
Оформлення акта на вилучення документів	Один акт	1 год
Заповнити статистичні таблиці	Одна позиція	1 год
Складання таблиць складу та використання фонду	Одна таблиця	1 год 15 хв
Створення бібліографічного запису на документ, що надійшов уперше	Один документ	10 хв
Встановити відомості про ті або інші об'єкти, події, процеси та дати	Один документ	10-40 хв

Наведені норми часу свідчать, що навіть типові бібліотечні операції потребують значних витрат часу.

Для шкільної бібліотеки, де основні процеси часто виконує один працівник, ручне ведення обліку ускладнює швидке оновлення даних, пошук документів і підготовку звітності. Тому автоматизація таких операцій за допомогою АБІС є доцільною з точки зору раціонального використання робочого часу бібліотекаря та підвищення точності обліку.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						13
Змн.	Арк.	№ докум.	Підпис	Дата		

Згідно з п. 2.2 Положення про бібліотеку загальноосвітнього навчального закладу, її завданням є розширення інформаційних послуг через нові форми роботи та новітні технології [1].

Відкритої офіційної статистики щодо кількості шкільних бібліотек, які вже використовують АБІС, немає. У публікації Chytomo Оксана Бруй проаналізувала стан автоматизації бібліотек України станом на 2021 рік. Статистика показує, що цифрова трансформація відстає на 20-25 років. Також вказано, що автоматизація університетських бібліотек розвинена краще, 164 з 184 бібліотеки використовують АБІС, тоді як стан шкільних і спеціалізованих бібліотек залишається менш прозорим [4].

Державні документи також фіксують потребу в цифровізації бібліотек. Стратегія розвитку бібліотечної справи до 2025 року визначала проблемою недостатній рівень упровадження інформаційних технологій, тому очікуваним результатом була стовідсоткова комп'ютеризація бібліотек, підключення до інтернету та розвиток технологічної інфраструктури [2]. У Стратегії розвитку читання до 2032 року також планують інформатизацію публічних бібліотек і бібліотек закладів освіти шляхом впровадження програмного забезпечення для автоматизації бібліотечно-інформаційних процесів, централізовану каталогізацію та розвиток електронної бібліотеки [5].

Нормативна база не встановлює обов'язкової вимоги використовувати АБІС у кожній шкільній бібліотеці, однак вимагає повного, достовірного й своєчасного обліку фонду. Зокрема, Інструкція з обліку документів у бібліотечних фондах вимагає фіксацію всіх фактів надходження та вибуття документів, а також узгодженість облікових форм [6]. На практиці це означає, що система повинна зберігати дані про кожний примірник, його інвентарний номер, бібліографічний опис, поточний стан, видачу, повернення та можливе списання.

В Україні використовують різні АБІС. Перелік включає ІРБІС, УФД/Бібліотека, UniLib, ALEPH, Koha, MAPK-SQL та інші рішення [7]. Водночас більшість таких систем створювалася насамперед для великих бібліотек, бібліотечних мереж, університетів або наукових установ. Але для школи це радше

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		14

проблема. Тому що працює лише один бібліотекар та без постійної ІТ-підтримки. А надлишкова кількість модулів не спрощує роботу, а тільки, навпаки, ускладнює її впровадження.

Локальна модель розгортання теж є проблемою. Це притаманно частині АБІС. Виникає залежність від одного комп'ютера, потреба в резервному копіюванні, складність віддаленого доступу й необхідність технічного обслуговування. А у разі втрати локальної бази бібліотека може втратити облікову інформацію.

Вибір АБІС також пов'язують із витратами та підтримкою.

Загалом є три підходи для впровадження АБІС. Перший підхід — це придбання комерційної системи, другий — використання відкритого програмного забезпечення, і останній — розроблення власного рішення. Кожен із них потребує ресурсів. До них належать витрати на ліцензію, супровід, адаптація відкритого коду або залучення розробників [7]. А для шкільної бібліотеки важливо, щоб система була простою в розгортанні та зрозумілою для бібліотекаря.

Під час вибору АБІС також потрібно враховувати сумісність даних. У рекомендаціях згадуються стандарти й протоколи MARC 21, Dublin Core, RDA, Z39.50, OAI-PMH [7].

Для шкільної бібліотеки повна підтримка всіх стандартів може бути надмірною, але бібліографічні записи мають бути структурованими й придатними для подальшого експорту або розширення.

Ще одним аспектом є походження програмного забезпечення. Українська бібліотечна енциклопедія зазначає, що розробником ІРБІС є російська Асоціація ЕБНІТ [8]. У документах Національної наукової сільськогосподарської бібліотеки НААН однією з причин переходу на FOLIO названо уникнення залежності від АБІС «ІРБІС» і технічного супроводу з боку країни-агресора [9]. Для бібліотек, які працюють із даними про користувачів і фонд, це не лише питання функціональності, а й безпеки та довгострокової підтримки.

Окремий ризик становлять «псевдо-АБІС» — програми, які фактично є списком книжок, але не підтримують повний облік примірників, інвентарні

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

номери, акти списання, фонд підручників, ролі користувачів і звітність [7]. Для бібліотеки це не задовільнить більшість потреб.

Отже, основна проблема полягає в невідповідності між вимогами до обліку бібліотечного фонду та інструментами, які доступні більшості бібліотек закладів освіти. Паперові журнали й електронні таблиці не забезпечують зручного пошуку, відстеження змін, швидкого формування звітів і належного захисту даних. Професійні АБІС мають різноманітний функціонал, але потребують технічного супроводу, налаштування серверної частини або навчання персоналу. Тому доцільним є розроблення веборієнтованої АБІС, адаптованої до умов шкільної бібліотеки та зосередженої на ключових процесах: веденні електронного каталогу, обліку основного фонду й підручників, видачі та поверненні документів, списанні, формуванні звітності й розмежуванні прав доступу.

1.2 Основні поняття та призначення інженерії ПЗ в даному домені

Для проєктування шкільної АБІС потрібно описати не лише перелік книг, а й бібліотечні процеси: комплектування фонду, каталогізацію, видачу, повернення, списання, звітність і пошук.

Українська бібліотечна енциклопедія визначає АБІС як автоматизовану інформаційну систему для автоматизації бібліотечних процесів із використанням інформаційно-комунікаційних технологій, обчислювальної техніки та телекомунікаційних мереж [10].

До типових груп задач АБІС належать обробка документів, ведення електронного каталогу, обслуговування користувачів, книгообмін, підготовка бібліографічних матеріалів, облік, контроль і статистика. Для шкільної бібліотеки цей перелік слід звужувати до процесів, які реально підтримують роботу закладу освіти: облік книг і підручників, читацькі формуляри, видача, повернення, списання, пошук та звітність.

Бібліотечний фонд є основною сутністю предметної області. Закон України «Про бібліотеки і бібліотечну справу» визначає його як упорядковане зібрання

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

документів, що зберігається в бібліотеці та формується відповідно до її завдань, складу користувачів і їхніх інформаційних потреб [11].

У шкільній бібліотеці фонд пов'язаний з освітнім процесом, тому його варто поділяти на основний фонд, фонд підручників та окремі ресурси для дітей з особливими освітніми потребами. Такий поділ впливає на модель даних, оскільки художня й довідкова література потребує обліку окремих примірників, а підручники частіше описуються за назвою, класом, роком видання, кількістю, потребою учнів і фактичним розподілом.

Облік фонду має підтримувати різні рівні деталізації. Інструкція з обліку документів у бібліотечних фондах поділяє облік книг на сумарний та індивідуальний. Індивідуальний облік пов'язує з присвоєнням кожному примірнику інвентарного номера, тобто фіксує стан конкретного примірника, а сумарний — це надходження, вибуття й підсумки руху фонду [6].

Для підручників важливий груповий підхід, тобто кількість у фонді, видано, повернено, втрачено, пошкоджено, списано.

Інвентарний номер повинен бути унікальним і не дублюватися між примірниками. Якщо документ списано або втрачено, його запис не варто видаляти фізично, адже він потрібний для історії фонду, актів списання, статистики та звірки облікових даних. Коректнішою є модель статусів: «наявний», «виданий», «списаний», «втрачений», «пошкоджений» або «потребує ремонту». Така модель дає змогу відокремити поточний стан примірника від факту його існування в обліку.

Потрібно розрізняти бібліографічний запис і примірник.

Бібліографічний запис описує назву, а це включає автора, назву, видавництво, місто видання, рік, ISBN, кількість сторінок, УДК, ілюстрації та інші ознаки. Примірник відображає конкретну одиницю фонду, тобто інвентарний номер, ціну, стан, дату списання, місце зберігання або зв'язок із видачею.

Одна назва може мати кілька примірників, тому змішування цих понять призводить до дублювання даних і ускладнює пошук, списання та звітність. Зв'язок цих понять подано на рисунку 1.1.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

Порівняння бібліографічного запису і примірника

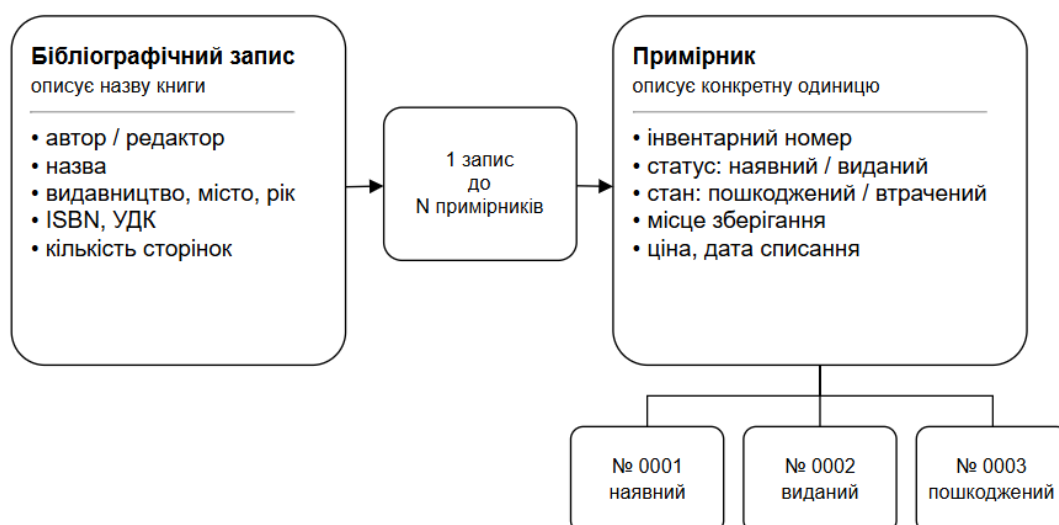


Рисунок 1.1 – Зв’язок між бібліографічним записом і примірниками

Як видно з рисунка 1.1, бібліографічний запис містить загальні відомості про видання, тоді як примірник фіксує дані про конкретну одиницю фонду.

Каталогізація перетворює фонд на структурований інформаційний ресурс. Для програмної системи це означає, що бібліографічний опис не слід зберігати як суцільний текст: автори, редактори, назва, рік, ISBN, УДК тощо мають бути окремими полями. Такий підхід потрібний для пошуку, фільтрації, сортування, формування бібліографічних списків і службових документів.

Для оформлення бібліографічних посилань в Україні використовується ДСТУ 8302:2015, а правила бібліографічного опису пов’язані з ДСТУ ГОСТ 7.1:2006 [12].

УДК використовується для систематизації документів за галузями знань. Перехід бібліотек закладів освіти на УДК закріплено наказом МОН України № 929 від 26.06.2017 [13]. Для шкільної АБІС це означає потребу в окремому полі індексу УДК, збереженні його у стандартизованій формі та підтримці пошуку або фільтрації за ним. УДК також може використовуватися під час формування статистики за тематичними групами фонду.

Електронний каталог є основою. Він має поєднувати бібліографічні записи, відомості про примірники та інформацію про доступність.

Користувачі АБІС мають різні ролі. Бібліотекар працює з фондом, читачами, видачею, поверненням і звітністю; читач переглядає каталог, власні книги та історію користування; адміністратор керує обліковими записами, ролями й службовими параметрами. Дані читачів, зокрема ПІБ, клас, дата народження, контакти та історія видачі, належать до персональних даних, тому їх обробка має відповідати Закону України «Про захист персональних даних» [14].

Абонементне обслуговування включає реєстрацію читача, пошук документа, оформлення видачі та визначення терміну користування, повернення, фіксацію втрати або пошкодження. У моделі даних цей процес пов'язує читача, примірник, дату видачі, очікувану дату повернення, фактичне повернення, статус і відповідального працівника.

Для підручників процес відрізняється. Видача може здійснюватися не одному читачеві, а класу або вчителю певною кількістю примірників. Тому облік видачі підручників має підтримувати часткове повернення, втрати, пошкодження та списання.

Звітність у шкільній бібліотеці базується на даних про склад фонду, надходження, вибуття, кількість читачів, книговидачу, забезпеченість підручниками, списання, втрати й пошкодження.

Наведені поняття не є лише теоретичними означеннями, оскільки кожне з них безпосередньо впливає на структуру майбутньої інформаційної системи.

Наприклад, поняття бібліотечного фонду визначає необхідність зберігати дані про книги та підручники, поняття примірника — потребу в інвентарному номері й статусі, а поняття читача — необхідність вести читацькі картки та історію користування літературою. Видача, повернення і списання формують окремі бізнес-процеси, які мають бути відображені в моделі даних і сценаріях роботи користувачів. Отже, термінологічний аналіз є основою для подальшого формування вимог до системи.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

1.3 Сучасні технології та тенденції, що застосовують в даній області

Автоматизовані бібліотечні інформаційні системи відрізняються не лише набором функцій, а й архітектурою, способом розгортання, вимогами до адміністрування, підтримкою бібліотечних стандартів і можливістю адаптації до локальних облікових процесів.

Для шкільної бібліотеки ці критерії мають практичне значення, оскільки заклади зазвичай не мають окремого системного адміністратора, власної серверної інфраструктури та бюджету на складне впровадження. Тому під час аналізу АБІС потрібно враховувати не тільки наявність певного функціоналу, а й те, наскільки система придатна для щоденної роботи бібліотекаря без постійної технічної підтримки.

За способом організації АБІС можна поділити на локальні, клієнт-серверні, веборієнтовані та хмарні. Локальні системи встановлюються на один комп'ютер або сервер бібліотеки, тому їхня робота залежить від стану конкретного обладнання. Клієнт-серверні рішення передбачають окрему базу даних і клієнтські робочі місця, що ускладнює налаштування, оновлення та резервне копіювання. Веборієнтовані системи працюють через браузер, не прив'язують користувача до одного пристрою та спрощують доступ до каталогу. А хмарні сервіси додатково зменшують потребу в локальному адмініструванні, але створюють залежність від постачальника, тарифів, політики зберігання даних і стабільності зовнішньої платформи.

У публікації Читомо серед поширених в українському бібліотечному сегменті названо ІРБІС, УФД/Бібліотека, UNILIB, ALEPH, КОНА та MAPK-SQL [4].

Більшість АБІС створювалися для університетських, наукових, публічних або відомчих бібліотек. Вони підтримують безліч модулів: від каталогізації, комплектування, електронного каталогу до обслуговування читачів, звітності та штрихкодуювання. Для великої бібліотеки така кількість модулів є виправдана, але для шкільної бібліотеки частина функцій не потрібна.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

Українські системи мають перевагу в тому, що вони ближчі до місцевої бібліотечної практики. Наприклад, УФД/Бібліотека підтримує електронний каталог, каталогізацію документів, класифікатори, комплектування фондів, обслуговування читачів та RFID [15]. З технічного погляду це комплексна модульна система з підтримкою вебдоступу, але для школи це недолік. Система потребує впровадження, налаштування та супроводу, а її можливості перевищують реальні потреби невеликої бібліотеки.

UniLib також орієнтована на українські бібліотечні процеси й позиціонується як комплекс програмних рішень [16]. Система має клієнт-серверну архітектуру, використовує Firebird 2.5, підтримує Unicode і може працювати на кількох робочих місцях. Це вже ближче до потрібного інструменту для шкільних бібліотек, проте така архітектура вимагає налаштування бази даних, клієнтських програм і резервного копіювання.

Окремо слід оцінювати системи російського походження, зокрема ІРБІС і МАРК-SQL. Для нового впровадження в закладах освіти такі рішення є недоцільними через походження та безпекові ризики.

Серед міжнародних відкритих систем найвідоміша Koha. Вона має відкритий код, вебінтерфейс, підтримує каталогізацію, надходження, обслуговування користувачів, OPAC, Z39.50/SRU та роботу з різними типами бібліотек [17]. Офіційна документація підкреслює, що Koha працює на GNU/Linux і не є desktop-додатком для Windows або macOS [18]. Для великих бібліотек це перевага, оскільки система підтримує стандарти й має активну спільноту. Для ліцею чи гімназії це стане проблемою. Адже виникає потреба в Linux-сервері, адмініструванні, налаштуванні MARC-фреймворків, імпорті даних, правах доступу, резервному копіюванні та адаптації звітності. ДНПБ також зазначає, що Koha не повністю пристосована до окремих процесів українських законодавчих реалій [19].

ALEPH і Evergreen демонструють інший тип систем — потужні рішення для великих бібліотек і бібліотечних мереж.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

ALEPH базується на Oracle, підтримує OPAC, комплектування, циркуляцію, періодику, Z39.50, MARC-сумісні формати, авторитетні файли, веб- і Windows-інтерфейси [19].

Evergreen створювалася для масштабної мережі публічних бібліотек і використовує PostgreSQL, OpenSRF, Apache, Perl, C, JavaScript та інші серверні компоненти [20, 21].

Обидві системи мають розвинену функціональність, але їхня архітектура орієнтована на професійне адміністрування. Для шкільної бібліотеки це зайві вимоги до технічного супроводу, навчання персоналу й ресурсів на впровадження.

Легші відкриті системи, зокрема OpenBiblio та PMB/PhpMyBibli, ближчі до потреб невеликих бібліотек. OpenBiblio написана мовою PHP і містить OPAC, модулі циркуляції, каталогізації та адміністрування. FAO описує її як просту веборієнтовану систему для малих бібліотек із фондами до 50 000 одиниць [22]. А ось PMB працює як вебзастосунок на базі PHP, Apache/IIS і MySQL або MariaDB, підтримує каталогізацію, циркуляцію, OPAC, Z39.50, UNIMARC, ISO 2709, XML та OAI [23]. Такі системи простіші за Koha або Evergreen, однак не орієнтовані на український облік підручників, інвентарні книги, акти списання тощо. Тому їх можна розглядати як технічну основу, але не як повністю готове рішення для школи.

Хмарні рішення, наприклад Librarika, зручні для малих бібліотек тим, що не потребують власного сервера або підтримки бази даних. Сервіс підтримує роботу з користувачами, матеріалами, резервуваннями, звітами та онлайн-доступом [24]. Однак така модель не усуває всіх ризиків. Крім того, типові хмарні сервіси не враховують українські освітні процеси.

Частина систем має переважно історичне або обмежене значення для сучасного аналізу. CDS/ISIS було розроблено ЮНЕСКО для введення, зберігання й пошуку структурованих нечислових даних; система мала версії для DOS, Windows і Unix та використовувалася в багатьох організаціях [25]. Liber Media у джерелах ДНПБ описується як система на постреляційній СУБД Pick [19]. Ці рішення важливі для розуміння еволюції АБІС, але їхня технологічна база не

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

відповідає очікуванням до сучасного вебзастосунку з мобільним доступом, ролями користувачів і простим адмініструванням.

Окремий напрям становлять модульні платформи бібліотечних сервісів. Прикладом є FOLIO — відкрита Library Services Platform, яку розвивають бібліотеки, розробники та постачальники [26]. Вона побудована на мікросервісній архітектурі, підтримує модульність, вебінтерфейс, багатокористувацьке середовище та роботу з друкованими, електронними й цифровими ресурсами [27]. Для шкільної бібліотеки корисною є сама ідея модульності, адже система не повинна містити зайві функції, які ускладнюють навчання та підтримку. Проте повноцінна мікросервісна платформа потребує розгортання, моніторингу, оновлень і супроводу, тому теж не для шкільних бібліотек.

Детальніше порівняння розглянутих систем за архітектурою, функціональністю, перевагами, обмеженнями та придатністю для бібліотеки закладу загальної середньої освіти наведено в додатку А. Узагальнення цього порівняння показує, що готові АБІС не повністю відповідають потребам ліцеїв та гімназій. Українські системи краще враховують локальний контекст, але часто є закритими, клієнт-серверними або потребують супроводу постачальника. Міжнародні відкриті системи мають сильну технічну базу й підтримують бібліотечні стандарти, однак потребують адміністрування, локалізації та адаптації до української звітності.

З технічного боку для шкільних бібліотек найбільш підходить веборієнтована архітектура. Такий підхід зменшує залежність від одного робочого місця, спрощує доступ до системи та краще відповідає умовам невеликої бібліотеки в закладі освіти.

1.4 Висновки по розділу

У першому розділі проаналізовано сучасний стан автоматизації шкільних бібліотек, основні поняття предметної області та технології, які використовують для створення АБІС.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

Шкільна бібліотека виконує не лише функцію зберігання книжок. Вона веде облік бібліотечного фонду, працює з підручниками, обслуговує читачів, оформлює видачу й повернення документів, готує звітність.

На практиці ці процеси часто виконуються вручну. Через це ускладнюється пошук потрібної інформації, зростає ризик помилок, дублювання записів і втрати важливих даних.

Основною проблемою є невідповідність між вимогами до обліку бібліотечного фонду та наявними інструментами.

Розглянуті теоретичні поняття предметної області показують, що майбутня система повинна враховувати специфіку бібліотек закладів загальної середньої освіти. Тобто мати наявність основного фонду, який включатиме художню та довідкову літературу, та фонд підручників і ресурсів для дітей з особливими освітніми потребами.

Також варто розрізняти бібліографічний запис і конкретний примірник, розуміти суть інвентарного обліку, статусів документів. Це все є вагомим для правильної побудови моделі даних і подальшого формування вимог до системи.

Також виконаний аналіз сучасних АБІС і підходи для їх реалізації. Існуючі професійні системи мають широкий функціонал, однак орієнтовані на великі бібліотеки, потребують технічного супроводу, складного налаштування або серверної інфраструктури.

Легші й хмарні рішення не враховують специфіку української шкільної бібліотеки, зокрема облік підручників, інвентарні номери, акти списання та локальну звітність.

Отже, результати аналізу підтверджують доцільність розроблення веборієнтованої автоматизованої бібліотечної інформаційної системи, адаптованої саме до потреб шкільної бібліотеки. Така система має бути простою у використанні, не вимагати складного адміністрування, забезпечувати електронний каталог, облік фонду й підручників, видачу та повернення документів, списання, пошук, формування звітів і захист персональних даних користувачів.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. АНАЛІЗ ВИМОГ ТА ПОСТАНОВКА ЗАДАЧІ

2.1 Збір та аналіз вимог користувачів

Збір вимог необхідний, щоб визначити реальні процеси, які існують в роботі шкільної бібліотеки. На цьому етапі аналізуються не окремі технічні функції, а щоденні дії бібліотекаря, наприклад, ведення фонду, пошук потрібного видання, облік видачі й повернення, робота з підручниками, підготовка документів, а також взаємодія з іншими бібліотеками.

У системній інженерії вимоги розглядають як формалізований опис потреб, очікувань і обмежень, які має задовольнити програмний продукт [28, с. 8]. Тому основна увага приділялася не тому, які сторінки має містити застосунок, а тому, які проблеми виникають у користувачів.

Аналіз вимог дає змогу уточнити реальні потреби користувача до початку програмування. Витрачений на це час у подальшому зменшує ризик помилок і сприяє створенню якіснішого коду [29, с. 16].

Основним джерелом стали консультації з працівниками і аналіз типових робочих ситуацій. Обраний підхід відповідає практиці збору вимог через інтерв'ю, аналіз документів, спостереження за процесами та сценарії використання [28, с. 215].

Під час консультацій уточнювалися правила обліку книг і підручників, порядок нумерації, особливості видачі та повернення, списання, а також проблеми, які виникають під час ручного ведення фонду. Наявні документи, правила обліку та чинні робочі процедури також можуть містити приховані вимоги до системи, тому їх аналіз теж був частиною збору вимог [28, с. 227].

Найпоширенішим процесом у роботі бібліотекаря є пошук і перевірка наявності видань. У паперовому або частково електронному варіанті така дія здебільшого займає більше часу, ніж сама видача. Працівнику потрібно швидко з'ясувати, чи є потрібна книга або підручник у фонді, скільки примірників доступно, а які з них уже видані.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

Заклад освіти має більше ніж один фонд. Для книжкового фонду важливими є інвентарний номер, автори, назва, рік видання та видавництво, УДК, ISBN і дані про надходження. Для підручників же вагомими стають клас, навчальна потреба, кількість у фонді та фактичний залишок. Отже, вимоги до системи мають враховувати різницю між звичайними книгами й підручниками, бо ці об'єкти використовуються по-різному.

Окремою проблемою є облік підручників для дітей з особливими освітніми потребами. Такі підручники не варто змішувати із загальним підручковим фондом, бо для них враховуються додаткові дані.

Тому надалі й будуть враховуватися ці три напрями щодо фонду. Такий поділ робить аналіз вимог більш керованим, оскільки різні групи об'єктів мають власні правила обліку, набори даних і сценарії використання [28, с. 83–84].

Також проаналізовано процес видачі та повернення. Потрібно не лише зафіксувати факт видачі, а й бачити, кому передано примірник, коли потрібно повернути та чи є протермінування.

У випадку підручників процес складніший, бо видача часто відбувається не одному читачу, а класу або вчителю. Повернення також може бути частковим. Частина примірників повертається в нормальному стані, частина втрачається або пошкоджується. Саме тому виникає потреба зберігати історію руху фонду, а не тільки поточний стан запису.

Наступною частиною аналізу став читацький облік. Бібліотекарю потрібна не просто картка читача, а швидкий доступ до його поточних видань. А для читача потрібно розуміти, які книги знаходяться зараз у нього на руках без звернення до бібліотекаря. Це зменшує кількість дрібних запитів і робить облік прозорішим для обох сторін.

Також врахована кількість працівників. У невеликому ліцеї чи гімназії всі операції часто виконує одна людина, а у більшому закладі фонд можуть вести двоє або троє працівників. Тому потрібна модель, за якої головний відповідальний за бібліотеку може приєднувати інших. Інші працівники повинні виконувати основні

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

бібліотечні операції, але не змінювати критичні налаштування, фінансові дані або службову інформацію закладу.

Один із найболючіших процесів — це обмін підручниками між закладами.

На практиці не кожна бібліотека має достатній резерв звичайних підручників або підручників для дітей з особливими освітніми потребами. Якщо певного видання бракує, бібліотекарю потрібно з'ясувати, у якому закладі є зайві примірники, чи готова бібліотека їх передати, як зв'язатися з відповідальною особою і на яких умовах можна домовитися про передачу. Без єдиної системи така комунікація відбувається повільно. Інформація про залишки розрізнена, контакти не завжди відкриті, а підтвердження домовленостей не зберігається в одному місці. Тому цей сценарій розглядався як окремий напрям, пов'язаний не тільки з каталогом, а й з контактними даними, статусом заявки та фактом завершення обміну.

Ще одним повторюваним процесом є підготовка документів. Бібліотекарі регулярно формують списки, звіти, бібліографічні матеріали та службові файли. Якщо ці матеріали готуються вручну, виникають помилки в назвах, номерах, датах або складі переліку. Тому варто передбачити генерування та збереження сформованих документів, а також повторне завантаження файлів.

Під час аналізу вимог використовувався сценарний підхід. Він описує не окремі функції, а послідовність дій користувача та очікуваний результат. У книзі «User Story Mapping» наголошується, що потрібно мати розуміння, для кого створюється продукт, яку проблему він розв'язує і який результат очікує користувач [30, с. 38]. Тому для цього проєкту сценарії будувалися навколо частих ситуацій, як от: знайти видання та видати книгу, сформувати документ або знайти заклад для обміну тощо.

Окремим джерелом вимог стали типові ускладнення та помилки, які виникають під час ручного ведення фонду. У роботі з вимогами такі відомості використовуються для виявлення процесів, що потребують покращення або автоматизації [28, с. 229–230].

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

Отже, результатом збору та аналізу вимог став опис частих бібліотечних процесів. Це основа для подальшого формування вимог до системи. Надалі такі сценарії деталізуються у форматі user story, де коротко визначаються роль користувача, потрібна дія та очікуваний результат. Такий підхід також допомагає пояснити причину реалізації функції та може бути використаний під час підготовки тестування [30, с. 16].

2.2 Формування функціональних та нефункціональних вимог

На основі сценаріїв, визначених у підрозділі 2.1, сформовано функціональні та нефункціональні вимоги до системи.

Функціональні вимоги описують дії, які має виконувати застосунок, а нефункціональні визначають умови для його роботи (безпека, продуктивність, цілісність даних, ергономічність і супроводжуваність) [28, с. 83-84]. Такий поділ потрібний для того, щоб окремо зафіксувати, які операції підтримує система, і окремо — якими мають бути якість, обмеження та поведінка цих операцій.

Щоб такі вимоги можна було використовувати на етапах проєктування, реалізації й тестування, вони мають бути точними, зрозумілими, узгодженими, однозначними, простежуваними та перевірюваними [28].

Функціональні вимоги поділено не лише за ролями користувачів, а й за напрямками роботи системи. Такий підхід узгоджується з принципом *scope definition*, за яким великий обсяг робіт або функціональний блок доцільно поділяти на менші й керованіші компоненти [31, с. 58]. Завдяки цьому кожен модуль можна окремо описувати, перевіряти та вдосконалювати. Наприклад, облік книги пов'язаний з інвентарним номером і бібліографічними даними, облік читача — з персональною карткою та історією видач, а облік підручників — із класами та кількістю примірників. Для кожного модуля можна визначити окремі сценарії використання, очікувані результати та критерії приймання.

У системі передбачено шість основних груп користувачів. Гість належить до неавторизованих користувачів і працює лише з відкритою частиною

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

застосунку. Зареєстрований користувач має обліковий запис у системі, однак його доступ до бібліотечних даних залежить від подальшої ролі або зв'язку з конкретною бібліотекою. Читач користується власним кабінетом, працівник бібліотеки виконує основні облікові операції, власник бібліотеки керує налаштуваннями своєї бібліотеки та працівниками, а адміністратор системи відповідає за службове керування платформою.

Гість працює лише з відкритою частиною застосунку. Для нього доступні інформаційні сторінки, статті, тарифи підписок, юридична документація, реєстрація, вхід та форма звернення. Доступ до фонду, читацьких даних чи службових документів не передбачено.

Читач має власний кабінет. Система має показувати його персональні дані, книги, які перебувають на руках, і історію користування бібліотекою. Для цієї ролі не передбачена зміна будь-яких даних.

Зареєстрований користувач після входу в систему може створити нову бібліотеку або приєднатися до вже наявної бібліотеки за кодом. Якщо користувач створює бібліотеку, він стає її власником і отримує доступ до керування основними параметрами закладу, налаштуваннями бібліотеки, підпискою та працівниками. Якщо користувач приєднується до бібліотеки за кодом, він отримує роль працівника бібліотеки та може виконувати основні бібліотечні операції в межах наданих прав.

Власник бібліотеки має розширені права в межах створеної бібліотеки. Він може редагувати дані закладу, змінювати налаштування бібліотеки та керувати підпискою, а також може виконувати бібліотечні операції.

Працівник бібліотеки виконує основні облікові операції в межах своєї бібліотеки. Він може працювати з фондами, читачами, видачею, поверненням, документами та заявками на обмін. Працівник також може покинути бібліотеку, до якої приєднався.

Першим функціональним напрямом є облік книжкового фонду. Для щоденної роботи потрібно пошук та фільтрація з посторінковим переглядом каталогу, створення запису про книгу, відкриття інформації про неї, редагування,

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

списання та відновлення списаного запису, а також видалення помилково створеної позиції. Інвентарний номер у межах однієї бібліотеки повинен бути унікальним, тому перед кожним ручним записом номеру варто перевіряти на дублювання.

Читацький модуль є другим напрямом. Потрібно, щоб була можливість створення читацької картки та акаунта, перегляд і редагування персональних даних, деактивацію та керування доступом до кабінету. Під час видачі потрібно фіксувати читача, примірник, дату видачі та очікувану дату повернення. Після повернення запис про операцію має зберігатися в історії.

Третій напрям пов'язаний з підручниками. Вони відокремлені від книжкового фонду, оскільки враховується клас, навчальна потреба та кількість у фонді. АБІС має мати функції створення, редагування, списання, відновлення та перегляд підручників. Повернення може бути частковим, тому потрібно окремо фіксувати повернуту, втрачену й пошкоджену кількість. Підручники для дітей з особливими освітніми потребами виділено в окремий модуль, бо для них додатково зберігаються код діагнозу.

Четвертим напрямом є обмін підручниками між бібліотеками. Має бути змога шукати доступні звичайні або спеціальні підручники в інших закладах, створювати заявку, переглядати вхідні й вихідні звернення, погоджувати, відхиляти, скасовувати або завершувати обмін. Бібліотека, яка виступатиме учасником, має вказати адресні та контактні дані.

П'ятий напрям охоплює документи та звітність. Формування службових документів має відбуватися на основі даних бібліотеки. Сформовані файли потрібно зберігати на сервері та давати змогу повторно їх завантажити. Генерація бібліографічного списку повинна відбуватися за вибраними книгами.

Відповідальність адміністратор системи стосуються підтримки платформи. Це включає функції керування користувачами, підписками, службовими полями, статтями, зверненнями та системною статистикою.

Кожна функціональна вимога повинна мати перевірюваний результат. Для створення запису це новий об'єкт, для списання — зміна стану, для видачі —

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

активний запис про книгу на руках, для повернення — перенесення в історію, для документа — сформований і доступний для завантаження файл. Це дає змогу пов'язати вимоги із тестуванням і прийманням реалізованих функцій [32, с. 68–69].

Нефункціональні вимоги визначають не окремі дії користувачів, а властивості системи під час виконання цих дій [32, с. 64]. Для АБІС розглядається безпека, продуктивність, цілісність даних, ергономічність і масштабованість.

Автентифікація користувачів, розмежування доступу за ролями, використання токенів та зберігання паролів у вигляді хешів — це все забезпечує безпеку. Перед кожною операцією потрібно перевіряти права користувача.

Бібліотека поступово накопичуватиме велику кількість записів, як книг і підручників, так і читачів, документів, історій змін. Тому сторінки з відображенням таких даних повинні відкриватися посторінково і враховувати фільтр та пошук. Це забезпечить власне продуктивність АБІС.

Цілісність даних означає, що зміна стану об'єкта не повинна руйнувати пов'язані записи. Деактивація читача не має видаляти історію його користування бібліотекою, списання книги не повинно знищувати дані про її попередні видачі, а завершення обміну має зберігати інформацію про заявку та її статуси. Усі об'єкти мають бути прив'язані до конкретної бібліотеки, щоб записи різних закладів не змішувалися між собою.

Ергономічність не визначається загальним словом «зручність», а конкретною поведінкою інтерфейсу в типових сценаріях. Зрозумілі назви дій, повідомлення про помилки, підтвердження небезпечних операцій і форми, які не дозволяють відправити некоректні дані, це все про користувацький досвід.

Масштабованість забезпечується поділом АБІС на модулі, які відповідають основним групам сутностей бази даних та бізнес-логіки. Така декомпозиція робить систему простішою для перевірки, зміни й подальшого розширення, оскільки кожен модуль має окрему зону відповідальності [31, с. 58].

Потрібно попідкуватися й про обробку помилкових ситуацій. Якщо запис не знайдено, користувач не має прав доступу, введені дані некоректні або бібліотека

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						31
Змн.	Арк.	№ докум.	Підпис	Дата		

не має активної оплати, застосунок повинен повідомити про причину відмови й не виконувати операцію.

Підсумовуючи, сформовані вимоги визначають межі застосунку, ролі користувачів, основні функціональні напрями та якісні характеристики системи. Поділ на модулі відповідає реальним процесам бібліотеки освітнього закладу й власне спрощує подальшу реалізацію, тестування та супровід програмного продукту.

2.3 Постановка задачі проєктування системи

Постановка задачі проєктування має спиратися на визначення меж системи, оскільки саме на етапі аналізу вимог потрібно встановити де починається і завершується відповідальність програмного продукту. Чітке окреслення меж дає змогу відокремити функції, які повинні бути реалізовані в програмному продукті, від зовнішніх процесів, що залишаються поза його відповідальністю [28].

На основі сформованих вимог спроектовано веборієнтовану бібліотечно-інформаційну систему для бібліотек закладів освіти. Вона об'єднує роботу з фондами (книжковим, фондом підручників і спецпідручників), читачами, документами, заявками на обмін і адміністративне керування сервісом.

Основна проєктна задача полягає не лише в автоматизації окремих бібліотечних операцій, а й у побудові багатокористувацької платформи. Кожна бібліотека повинна бути окремою логічною одиницею системи. Її книги, підручники, читачі, видачі, документи, заявки та фінансові дані мають зберігатися ізольовано від даних інших закладів. Такий підхід дає змогу використовувати одну систему для кількох бібліотек без змішування їхньої інформації.

У межах проєктування потрібно визначити структуру застосунку, склад основних модулів, зв'язки між сутностями бази даних, правила доступу до функцій і порядок взаємодії між клієнтською та серверною частинами. Окрему увагу слід приділити ролям користувачів. Гість має працювати лише з публічною частиною, зареєстрований користувач має власний обліковий запис і може

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						32
Змн.	Арк.	№ докум.	Підпис	Дата		

створити бібліотеку або приєднатися до неї, читач може працювати з власним кабінетом, працівник бібліотеки — з даними своєї бібліотеки, власник бібліотеки — з даними та налаштуваннями власної бібліотеки, а адміністратор системи — зі службовими й фінансовими параметрами платформи.

Визначення функціонального складу системи відповідає процесу планування обсягу проєкту, у межах якого поступово деталізується робота та формується письмовий опис цілей і основних результатів [31]. Тому межі системи мають бути сформульовані таким чином, щоб відокремити необхідний функціонал від другорядних можливостей, адже в програмних продуктах завжди існує більше потенційних ідей, ніж доступного часу та ресурсів для їх реалізації [30].

Межі системи охоплюють авторизацію, профіль користувача, керування бібліотеками, фондами, читачами, видачами, поверненнями, підручниками, документами, обміном підручниками, підписками, зверненнями, статистикою, звітністю та публічним контентом. Поза межами системи залишаються фізичне переміщення книг, фактичне проведення банківського платежу і юридичне оформлення договорів.

Постановка задачі також повинна враховувати цілісну картину майбутнього продукту. Розроблення лише окремих функцій без розуміння повного користувацького шляху може призвести до неузгодженого рішення. Саме тому під час проєктування доцільно аналізувати не лише перелік функцій, а й послідовність дій користувачів у межах основних сценаріїв роботи системи [30].

Під час проєктування потрібно врахувати, що частина функцій залежить від активного оплаченого статусу бібліотеки. Тому система має містити механізм перевірки доступу до можливостей згідно з тарифним планом. Працівник бібліотеки повинен бачити стан доступу лише своєї бібліотеки, тоді як адміністратор системи має керувати тарифами, підписками на ці тарифи, платежами та службовими реквізитами на рівні всієї платформи.

Важливою умовою є збереження історичних даних. Списання книги, деактивація читача, повернення виданого примірника або завершення заявки на

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

обмін не повинні призводити до втрати пов'язаних записів. Історія видач, повернень, списань, розподілів підручників і заявок має залишатися доступною для перегляду та формування звітних матеріалів.

Окремо потрібно передбачити роботу з великими списками даних. Каталоги книг і підручників, списки читачів, документи, звернення, заявки та історія операцій мають підтримувати пошук, фільтрацію і посторінкове виведення.

До складу системи також доцільно включити модуль статистики та звітності. Його варто розглядати не лише як допоміжну функцію, а як інструмент управління бібліотекою, оскільки кількісні та якісні показники можуть використовуватися для аналізу діяльності й прийняття управлінських рішень [33].

Результатом проєктування має стати структура вебзастосунку, у якій визначено основні модулі, ролі користувачів, межі відповідальності, зв'язки між даними та правила доступу. Така структура повинна забезпечити автоматизацію основних процесів шкільної бібліотеки, ізоляцію даних між закладами та адміністрування сервісу як окремого програмного продукту.

2.4 Висновки по розділу

У розділі проведено аналіз вимог до АБІС для бібліотек закладів освіти. На основі розгляду реальних робочих процесів бібліотеки визначено основні проблеми, які потребують автоматизації. Це облік книжкового фонду, робота з підручниками та спеціальними підручниками, ведення читацьких карток, формування документів, а також організація обміну підручниками між закладами.

У процесі аналізу встановлено, що бібліотечний фонд доцільно поділяти на кілька окремих напрямів: книжковий фонд, фонд підручників і фонд підручників для дітей з особливими освітніми потребами. Такий поділ обґрунтований тим, що ці групи об'єктів мають різні набори даних, правила обліку та сценарії використання.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						34
Змн.	Арк.	№ докум.	Підпис	Дата		

Також визначено потребу у збереженні історії руху фонду, оскільки для бібліотечної діяльності важливо фіксувати не тільки поточний стан примірника, а й попередні видачі, повернення, списання, втрати або пошкодження.

На основі зібраних вимог сформовано функціональні та нефункціональні вимоги.

Функціональні вимоги згруповано за ролями користувачів і напрямками роботи системи. Передбачено шість основних груп користувачів. Це гість, зареєстрований користувач, читач, працівник та власник бібліотеки та адміністратор системи. Для кожної ролі визначено доступні дії та межі відповідальності, що дає змогу забезпечити правильне розмежування доступу до даних і функцій.

Нефункціональні вимоги визначають якісні характеристики майбутньої системи. До них належать безпека, продуктивність, цілісність даних, ергономічність, масштабованість і коректна обробка помилкових ситуацій.

У межах постановки задачі проектування майбутня система має бути не лише засобом автоматизації окремих бібліотечних операцій, а багатокористувацькою платформою для кількох бібліотек. Тому кожна бібліотека повинна розглядатися як окрема логічна одиниця, а її дані мають зберігатися ізольовано від даних інших закладів. Такий підхід забезпечує можливість використання одного програмного продукту різними бібліотеками без змішування їхньої інформації.

Отже, у результаті виконаного аналізу сформовано основу для подальшого проектування вебзастосунку. Визначено ключові процеси бібліотеки, ролі користувачів, функціональні вимоги, нефункціональні, межі системи та основні проєктні задачі. Це дає змогу перейти до розроблення архітектури, структури бази даних, модулів застосунку та механізмів взаємодії між клієнтською і серверною частинами системи.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						35
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ПРОЄКТУВАННЯ СИСТЕМИ

3.1 Розробка архітектури програмного забезпечення

Майбутню АБІС спроектовано як веборієнтовану клієнт-серверну систему. Це передбачає поділ на клієнтську та серверну частину, а також сховище даних. Користувач працює із системою через вебінтерфейс, а обробка даних виконується на серверному рівні. Це дає змогу не прив'язувати роботу бібліотеки до одного комп'ютера та забезпечити доступ для бібліотекаря, адміністратора й читача з різних робочих місць.

На верхньому рівні АБІС складається з інтерфейсу користувача, серверної логіки, реляційної бази даних, файлового сховища та середовища розгортання.

Інтерфейс користувача призначений для відображення сторінок, форм, таблиць, фільтрів, повідомлень і результатів операцій. Він не повинен мати прямого доступу до бази даних або самостійно виконувати критичні бізнес-правила. Його завдання — приймати дії користувача, передавати запити на сервер і відображати отриману відповідь.

Клієнтська частина має підтримувати кілька логічних зон (публічну частину, кабінет працівника бібліотеки, читацький кабінет і адміністративний дашборд). Такий поділ не замінює серверну перевірку прав, але робить інтерфейс зрозумілим для різних груп користувачів.

Серверну частину доцільно організувати як модульний моноліт. Такий поділ відповідає підходу, за яким архітектурна структура системи може складатися з модулів із власною функціональною відповідальністю [32, с. 4–5]. До того ж, розділення відповідальності зменшує залежність між частинами системи [32, с. 20–21]. Для АБІС такий варіант простіший за мікросервісну модель, оскільки бібліотечні процеси мають спільну модель даних і не потребують окремої інфраструктури для взаємодії багатьох незалежних сервісів.

Внутрішня структура системи має будуватися за шаровим принципом. Перший шар відповідає за приймання запитів і передавання їх до потрібного

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						36
Змн.	Арк.	№ докум.	Підпис	Дата		

модуля. Другий шар містить прикладну логіку, наприклад, створення книги, реєстрацію читача або оформлення видачі. Третій шар відповідає за роботу зі сховищем даних. Шарова архітектура дає змогу відокремити інтерфейс, бізнес-правила та інфраструктурні деталі [34, с. 75].

База даних має бути централізованим реляційним сховищем. Реляційна модель підходить для бібліотечної системи, оскільки більшість даних пов'язані між собою. Книга може мати кількох авторів, бібліотека має багато читачів, читач має історію видач, а документи формуються на основі даних фонду. Реляційна модель подає дані у вигляді таблиць, пов'язаних спільними атрибутами. Правильне проектування бази даних є критичним, оскільки помилки структури складно виправити навіть за допомогою якісного програмування або потужної СУБД.

Окремим компонентом архітектури є файлове сховище. Воно буде використовуватися для сформованих документів, зображень користувачів, зображень інформаційних матеріалів та інших файлів, які не потребують зберігання безпосередньо в таблицях бази даних. У реляційному сховищі при цьому мають зберігатися метадані, такі як назва файлу, тип, власник, дата створення, зв'язок із бібліотекою або користувачем і шлях до файлу. Це відокремлює облікові дані від двійкових ресурсів і не перевантажує базу даних файлами.

Для розгортання передбачено ізольовані середовища виконання для клієнтської та серверної частини, і також для бази даних. Таким чином зменшується від налаштувань конкретного комп'ютера, спроститься перенесення між середовищами та буде змога запускати складові системи як окремі, але пов'язані компоненти.

Однак варто організувати так, щоб дані бази зберігалися у постійному сховищі, аби перезапуск середовища не призводив до втрати інформації.

Перевагою обраної архітектури є поєднання простоти розгортання та можливості подальшого розвитку.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

На початковому етапі немає потреби розділяти систему на багато незалежних сервісів, оскільки бібліотечні процеси мають спільну модель даних. Водночас модульний поділ дає змогу локалізувати зміни в межах окремих частин системи. Здатність підтримувати зміни з мінімальним впливом на інші компоненти є однією з ознак якісної архітектури [32, с. 27–28].

Для опису архітектури розгортання системи BookCatalog побудовано UML-діаграму розгортання, показану на рисунку 3.1. Вона відображає фізичні та логічні вузли, у межах яких працюють основні частини системи.

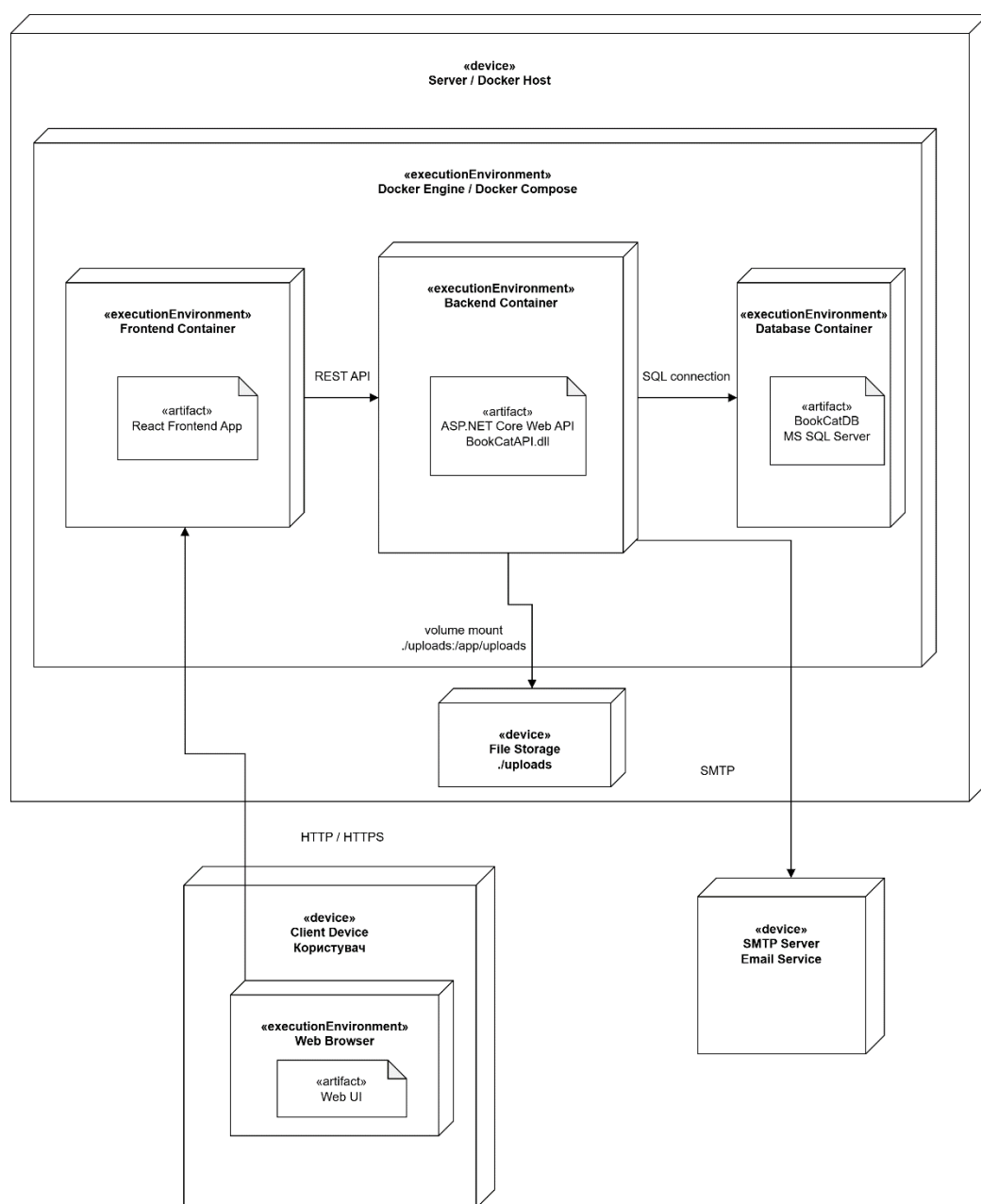


Рисунок 3.1 – UML-діаграма розгортання системи BookCatalog

У висновку архітектура майбутньої АБІС має забезпечити централізовану обробку даних, модульність, окреме зберігання файлів і можливість розгортання системи в ізольованому середовищі.

3.2 Моделювання бізнес-процесів та системних компонентів

Моделювання АБІС охоплює ролі користувачів, варіанти використання, бізнес-процеси, стани об'єктів, модель даних і правила доступу.

Основними акторами системи є гість, зареєстрований користувач, власник бібліотеки, працівник бібліотеки, читач і адміністратор системи.

Гість працює з публічною частиною. Він переглядає інформаційні сторінки, тарифи, статті, форму входу, реєстрації та звернення.

Зареєстрований користувач після входу може створити бібліотеку або приєднатися до вже створеної за кодом.

Власник бібліотеки керує бібліотекою, учасниками, оплатою, налаштуваннями й участю в обміні підручниками, а також може робити те, що й працівник бібліотеки.

Працівник бібліотеки же виконує облікові операції. Тобто додає книги, редагує фонд, створює читачів, оформлює видачу й повернення, формує документи та працює з підручниками.

Читач має окремий кабінет і бачить власний профіль, активні книги та історію видач.

Адміністратор керує платформою загалом. Сюди входять тарифи, підписки, звернення, користувачі, статті та службові дані.

Для узагальнення взаємодії користувачів із системою побудовано UML-діаграму варіантів використання. На ній відображено основних акторів і доступні їм сценарії. Діаграма дає змогу відокремити публічні дії гостя, бібліотечні операції працівника, дії власника бібліотеки, читацький кабінет і адміністративне керування. UML-діаграму варіантів використання системи BookCatalog наведено на рисунку 3.2.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

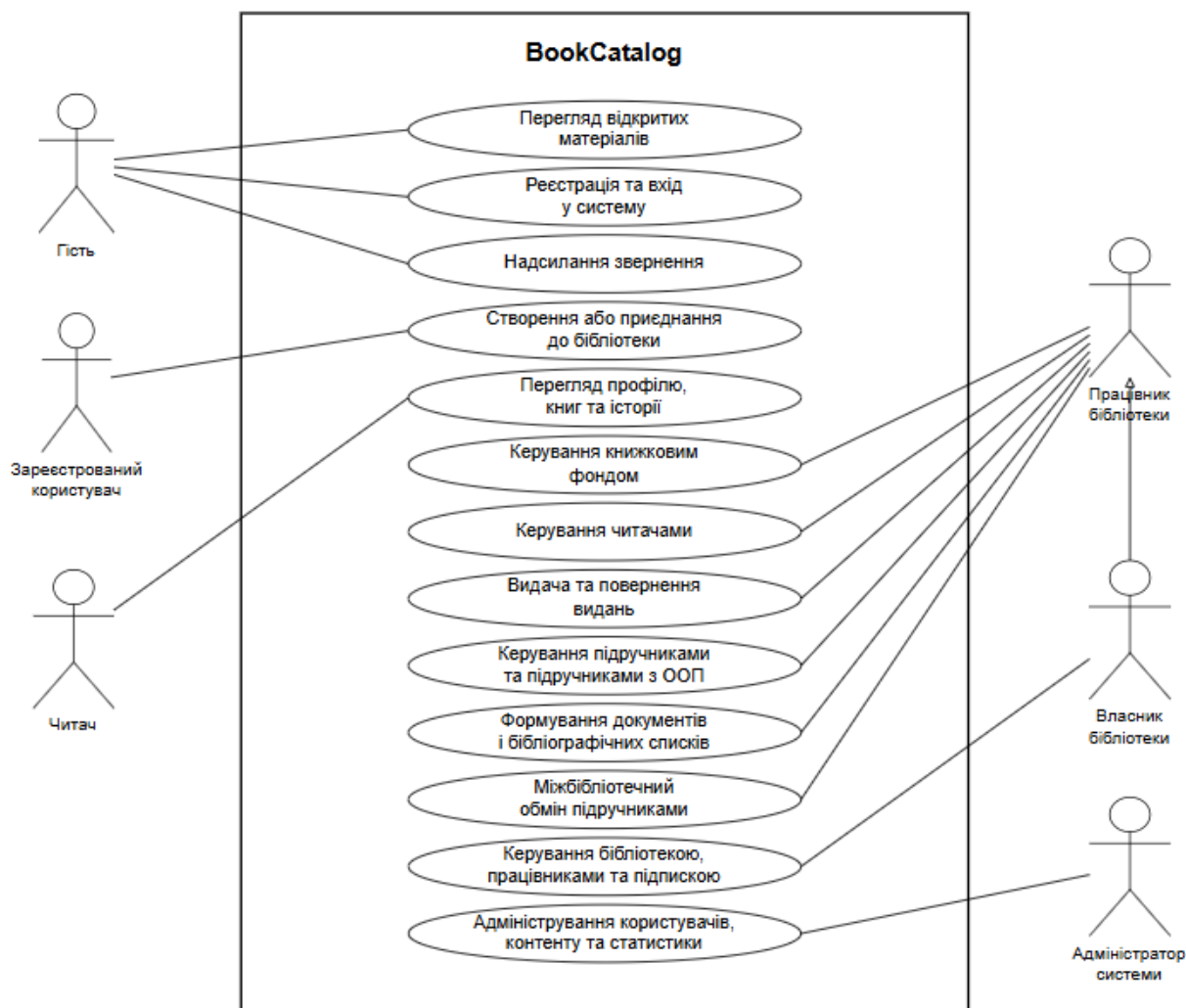


Рисунок 3.2 – UML-діаграма варіантів використання системи BookCatalog

Центральним процесом є створення бібліотеки. Після реєстрації користувач задає назву. Система створює окремий бібліотечний простір, до якого надалі належать книги, читачі, документи, підручники, заявки й фінансові записи.

Для приєднання працівників використовується код підключення. Після його введення система визначає бібліотеку, перевіряє можливість приєднання та створює зв'язок між користувачем і бібліотекою. Цей процес доцільно подати діаграмою діяльності, оскільки він містить послідовні перевірки та альтернативні результати виконання сценарію. Діаграму діяльності процесу приєднання користувача наведено на рисунку 3.3.

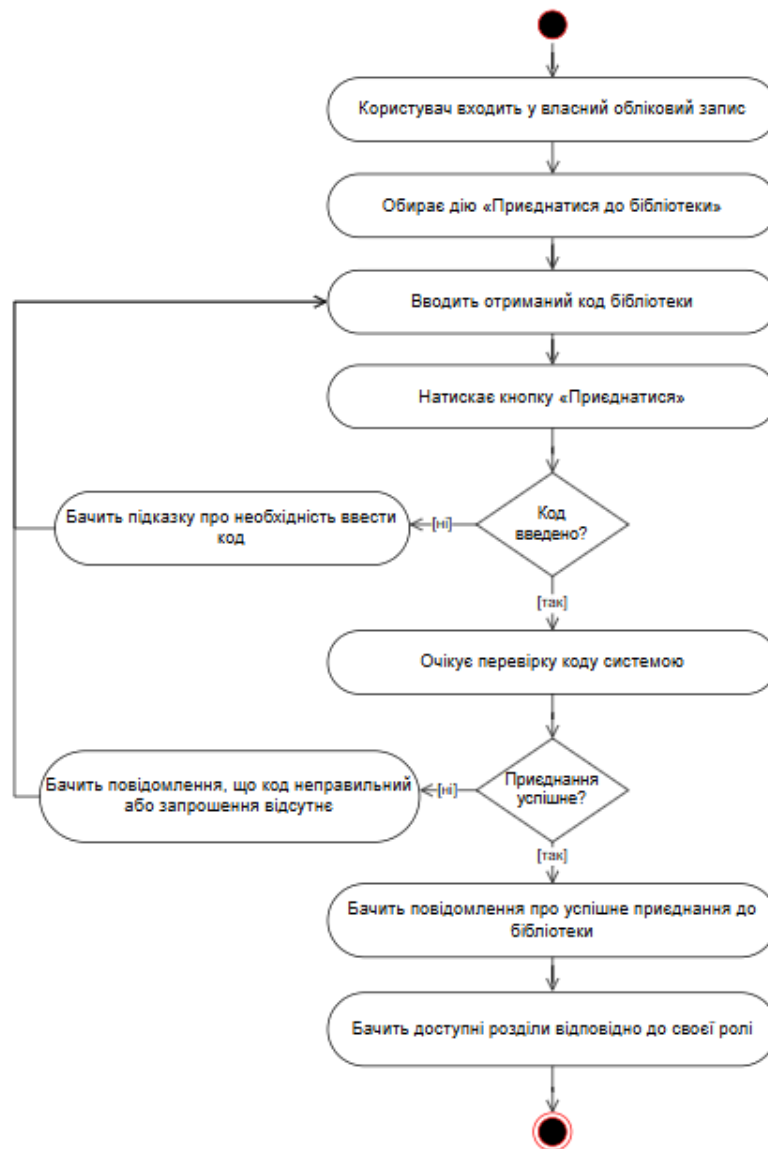


Рисунок 3.3 – UML-діаграма діяльності процесу приєднання користувача

Книжковий фонд моделюється як набір примірників, пов’язаних із конкретною бібліотекою.

Під час додавання книги працівник вносить бібліографічні та облікові дані. Перед збереженням система перевіряє роль користувача, належність до бібліотеки, оплату доступу та коректність даних. Редагування виконується за тими самими правилами. Списання книги не видаляє запис, а змінює її стан, що зберігає зв’язки з видачами, документами та статистикою.

Для модуля книжкового фонду побудовано діаграму станів книги. Вона показує переходи між станами “активна”, “видана”, “повернена” і “списана”, а

також окремі результати користування примірником, такі як “втрачена” або “пошкоджена”. Діаграму станів книги наведено на рисунку 3.4.

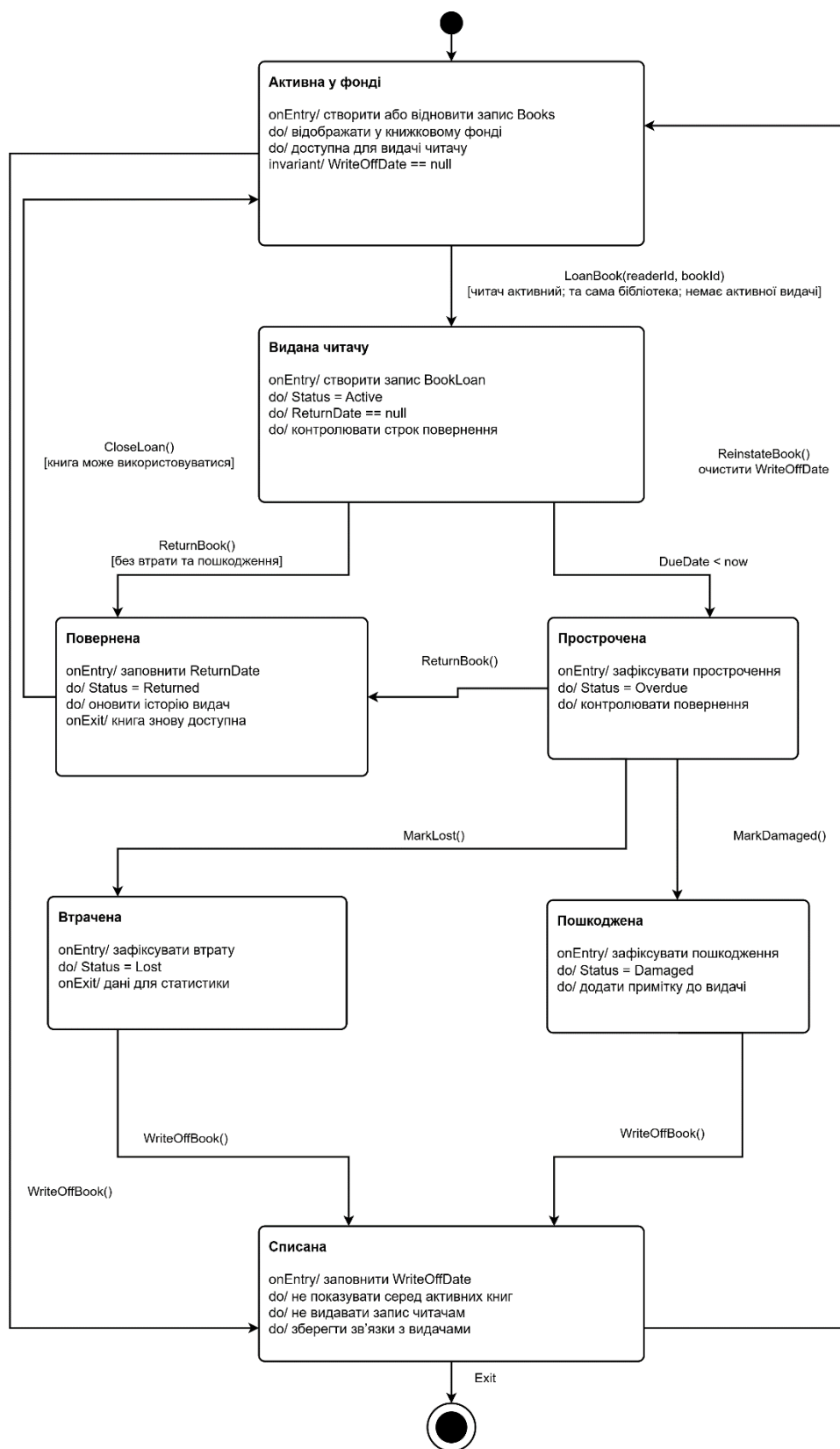


Рисунок 3.4 – UML-діаграма станів книги

Процес видачі книги починається з вибору читача й книги. Система перевіряє, чи читач активний, чи книга належить тій самій бібліотеці, чи не списана вона та чи не перебуває в активній видачі. Після успішних перевірок створюється запис видачі з датою видачі, плановою датою повернення, відповідальним працівником і початковим станом “активна”. Послідовність видачі подано у вигляді діаграми діяльності на рисунку 3.5.

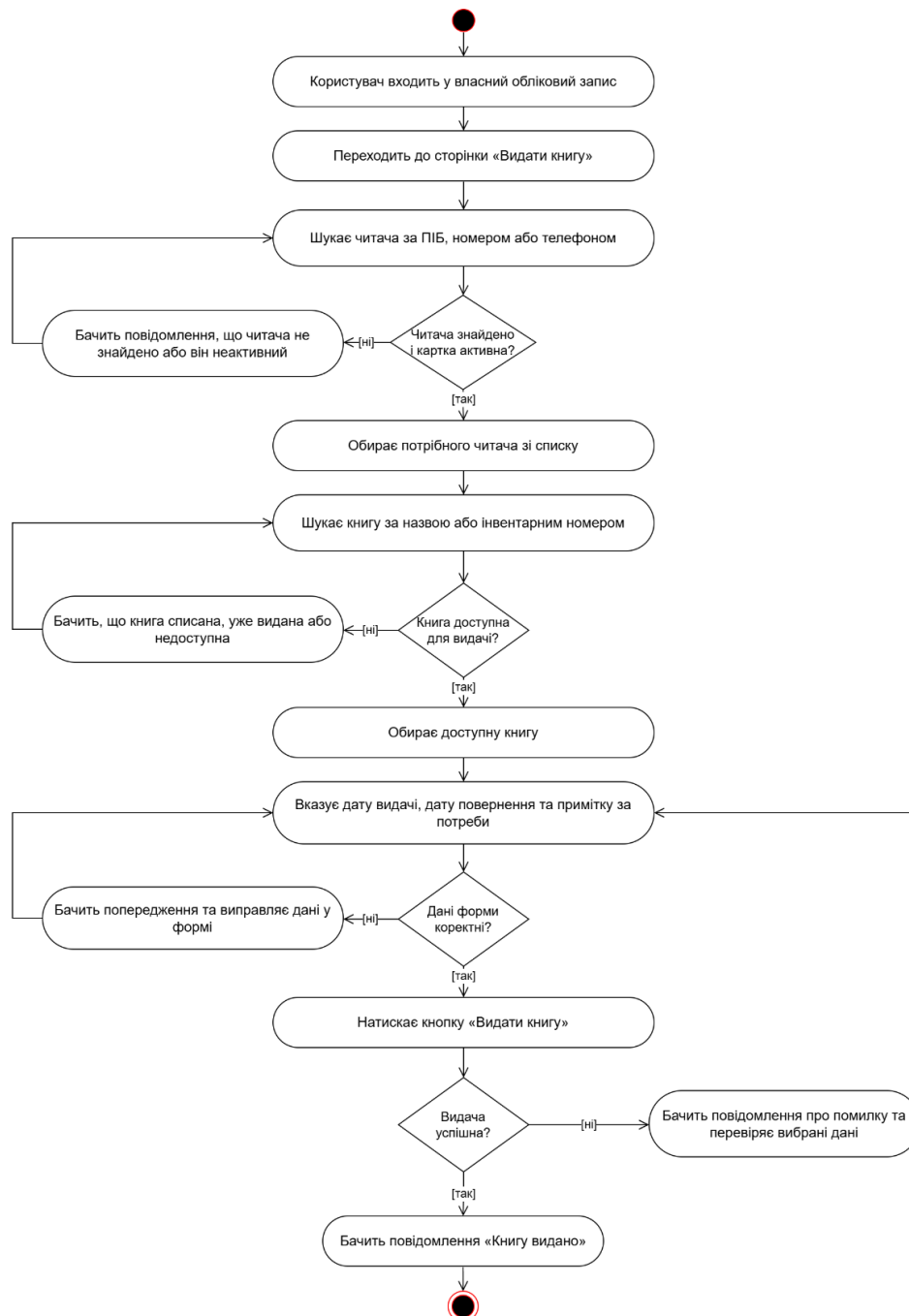


Рисунок 3.5 – UML-діаграма діяльності процесу видачі та повернення книги

Повернення книги змінює стан видачі на “повернена”, “прострочена”, “втрачена” або “пошкоджена”.

Модуль читачів забезпечує ведення читацьких формулярів. Для читача зберігаються персональні облікові дані, читацький номер, клас, статус активності та зв’язок із бібліотекою. Створення читацького формуляру наведено на рисунку 3.6.

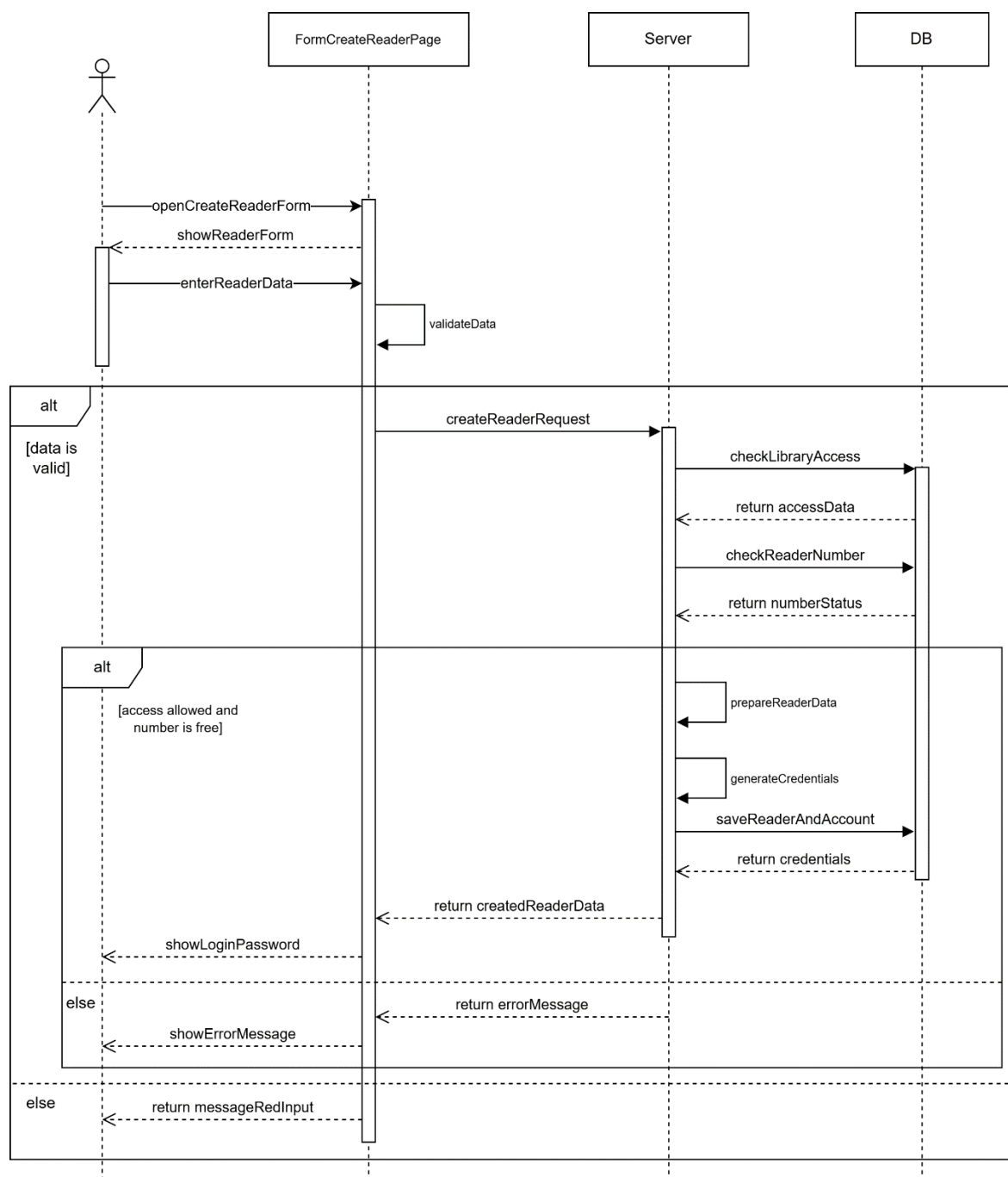


Рисунок 3.6 – UML-діаграма послідовності процесу створення читацької картки

Облік підручників відрізняється від обліку книг. Книга розглядається як окремий інвентарний примірник, а підручник — як позиція з кількісними показниками: потреба учнів, потреба вчителів, фонд, видані, втрачені, пошкоджені, доступні примірники, резерв і дефіцит. Під час видачі підручників система фіксує отримувача. Ними можуть бути клас або вчитель, а також фіксується кількість, дата та відповідальний працівник. Процес видачі підручників наведено на рисунку 3.7.

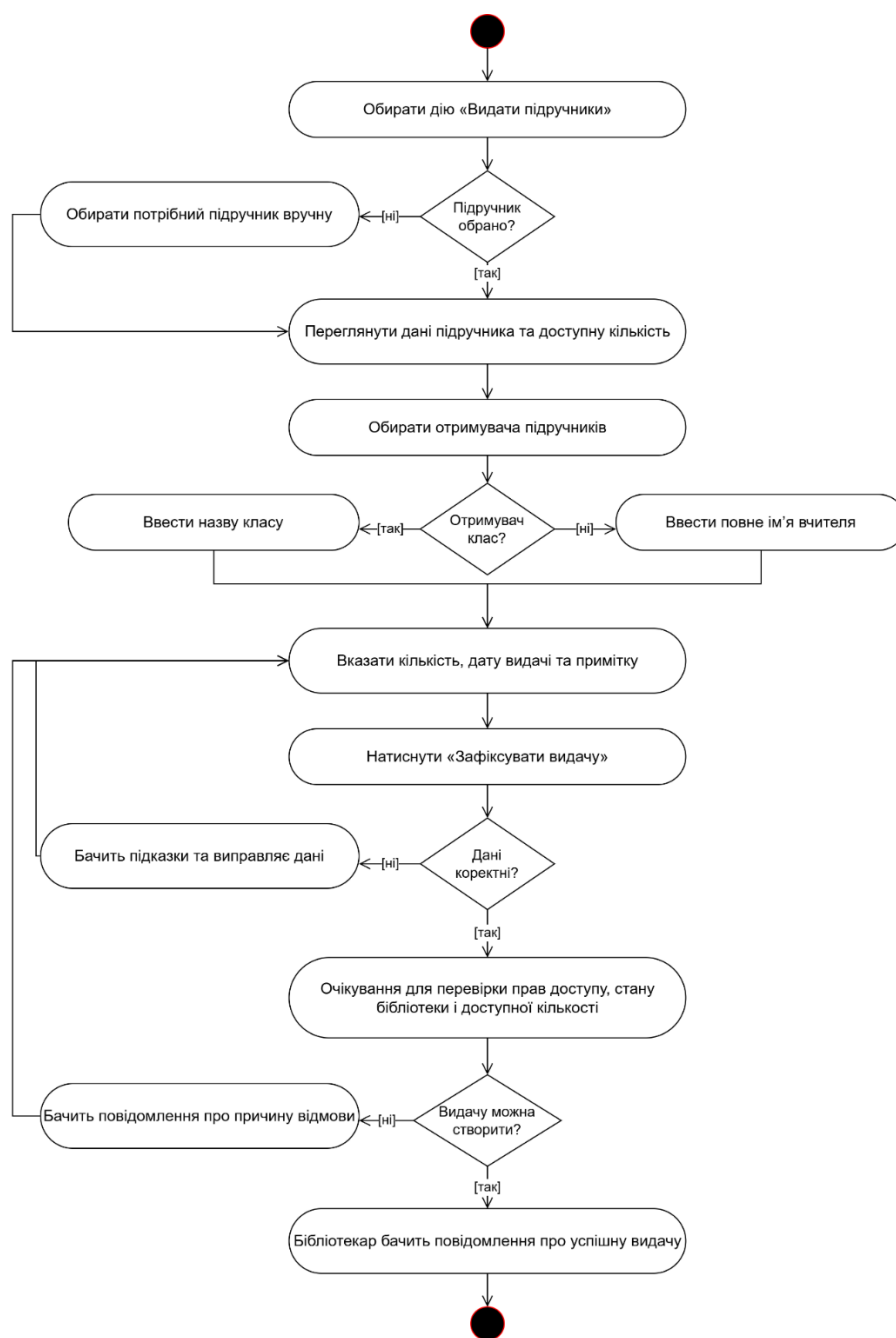


Рисунок 3.7 – UML-діаграма діяльності процесу видачі підручників

Повернення може бути частковим, оскільки частину примірників можуть повернути, а частину — позначити як втрачену або пошкоджену. Процес повернення підручників наведено на рисунку 3.8.

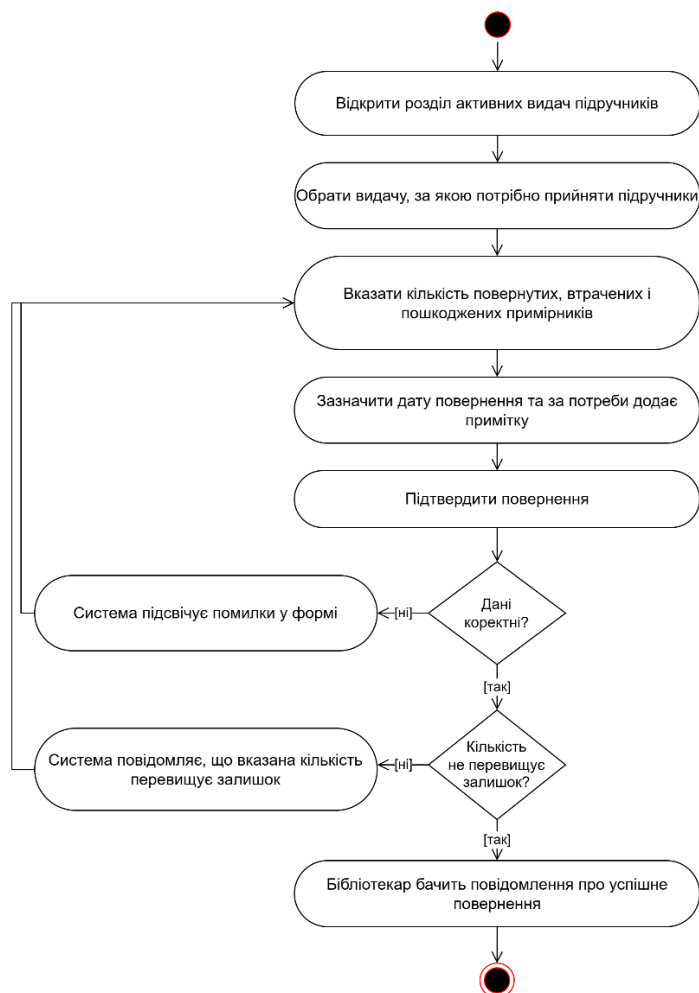


Рисунок 3.8 – UML-діаграма діяльності процесу повернення підручників

Окремо моделюються підручники для дітей з особливими освітніми потребами. Вони мають ті самі базові процеси. Відмінністю є додаткова ознака освітньої потреби.

Міжбібліотечний обмін підручниками є процесом взаємодії двох бібліотек. Бібліотека-запитувач шукає доступні позиції в інших бібліотеках. Система виключає власну бібліотеку з результатів, перевіряє участь бібліотеки-надавача в обміні, доступність підручника та резерв. Резерв визначається як різниця між фондом і потребою учнів та вчителів. Якщо запитана кількість не перевищує

резерв, створюється заявка. Вона проходить стани “очікує розгляду”, “погоджена”, “відхилена”, “скасована” або “завершена”. Послідовність створення заявки на обмін наведено на рисунку 3.9.

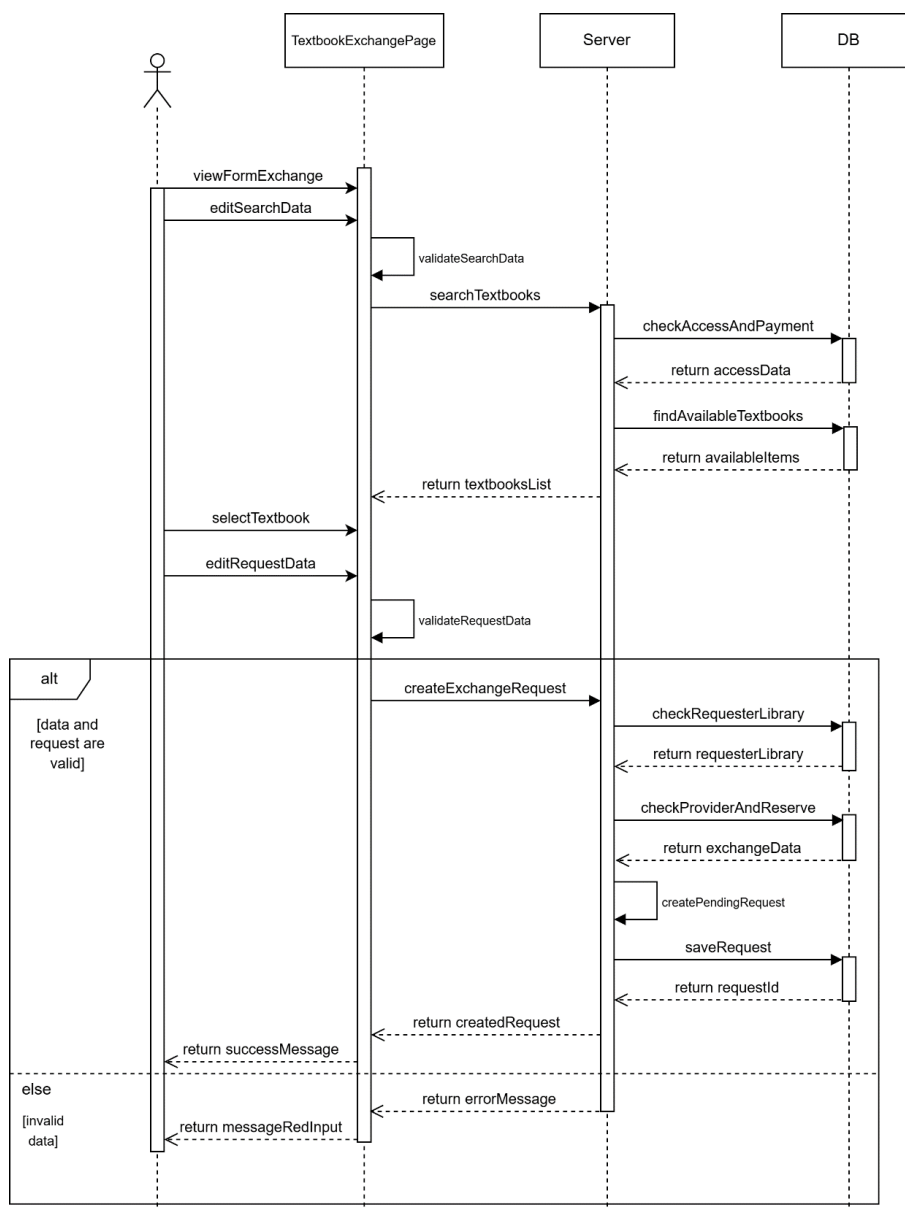


Рисунок 3.9 – UML-діаграма послідовності процесу створення заявки на міжбібліотечний обмін підручниками

Формування документів базується на вже збережених даних бібліотеки. Працівник обирає тип документа та потрібні параметри. Система отримує дані, формує файл і створює службовий запис із назвою, типом, датою створення та шляхом до файлу. Сам файл зберігається окремо від основних записів бази

даних. Взаємодію користувача, системних компонентів, бази даних і файлового сховища під час формування документа наведено на рисунку 3.10.

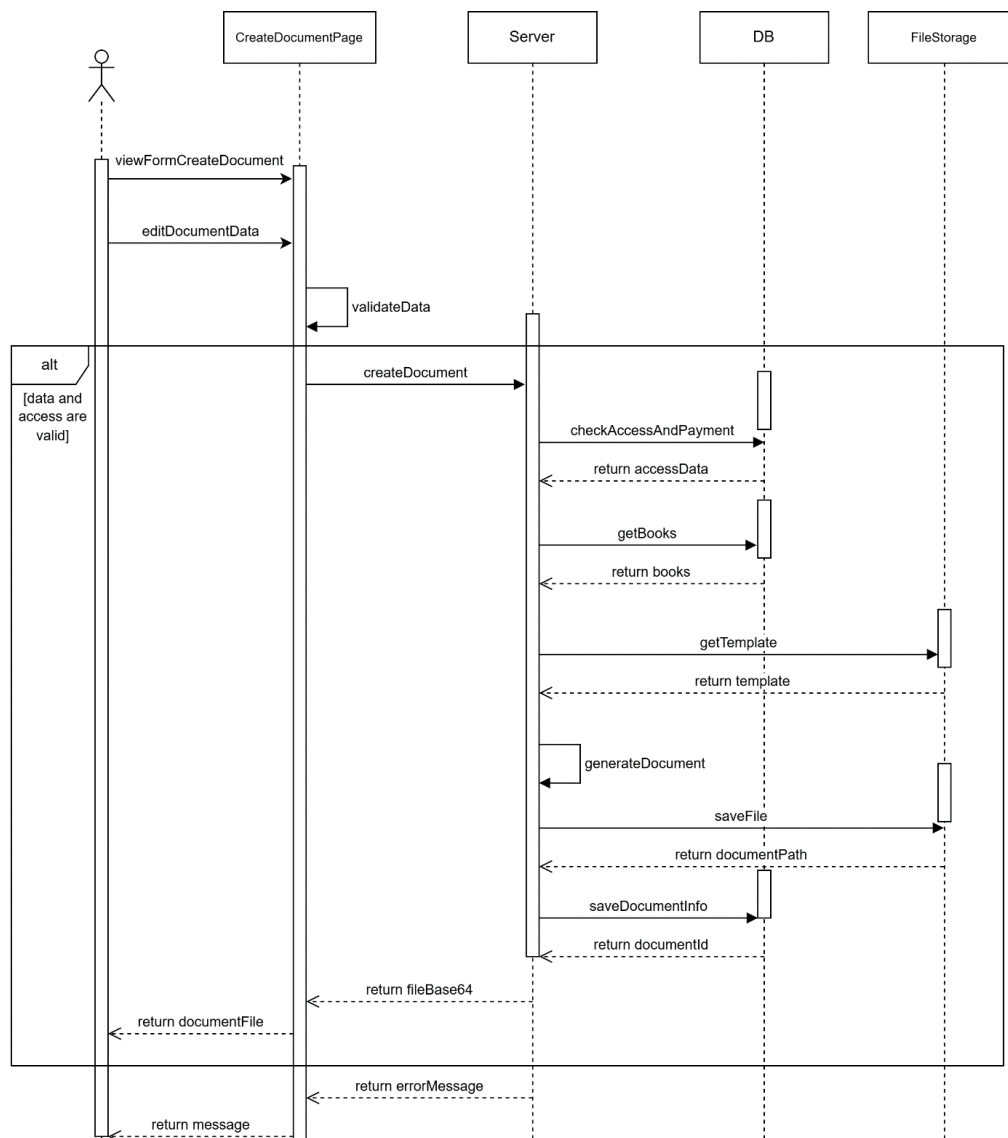


Рисунок 3.10 – UML-діаграма послідовності процесу формування документа

Фінансовий процес пов'язаний із тарифами й підписками. Власник бібліотеки обирає тариф, після чого створюється запит на підписку із сумою, тривалістю та станом підтвердження. До підтвердження оплати доступ до частини бібліотечних операцій може бути обмежений. Після підтвердження система активує підписку й оновлює фінансовий стан бібліотеки.

Адміністративний модуль охоплює керування тарифами, підписками, користувачами, зверненнями, статтями, тегами та службовими даними. Щоб не

перевантажувати загальну діаграму варіантів використання, адміністративні сценарії винесено в окрему діаграму. UML-діаграму варіантів використання адміністратора системи наведено на рисунку 3.11.



Рисунок 3.11 – UML-діаграма варіантів використання адміністратора системи

Інформаційна модель системи подається через ER-діаграму. Вона відображає основні сутності, їхні атрибути та зв'язки. ER-модель є основою для концептуальної схеми, зрозумілої як розробникам, так і користувачам предметної області. Центральною сутністю є бібліотека, оскільки через неї ізолюються дані різних організацій. Частину ER-діаграми бази даних системи BookCatalog, яка відповідає за бібліотеку, книжковий та підручниковий фонд, а також читачів, наведено на рисунку 3.12.

					БР.ІІІ - 18.00.00.000 ПЗ	Арк.
						49
Змн.	Арк.	№ докум.	Підпис	Дата		

Під час побудови ER-моделі застосовано принцип мінімальних даних. У моделі мають бути тільки ті дані, які потрібні для виконання системних операцій, без полів, що не використовуються жодним процесом.

Права доступу моделюються на двох рівнях: системному та бібліотечному. Системний рівень розмежовує гостя, користувача, читача й адміністратора. Бібліотечний рівень визначає дії власника та працівників у межах конкретної бібліотеки.

Дані різних бібліотек не змішуються, оскільки всі ключові облікові записи пов'язані з конкретною бібліотекою. Для більшості бібліотечних операцій додатково перевіряється активна оплата.

3.3 Вибір та обґрунтування технологій та інструментів розробки

Для реалізації BookCatalog обрано веборієнтований клієнт-серверний стек. Вибір технологій розглядається як частина архітектурних рішень, оскільки саме платформа, бібліотеки, СУБД та інфраструктурні засоби визначають спосіб реалізації функцій системи [32, с. 76].

Клієнтська частина реалізована як SPA на React 18.3.1. React обрано через компонентний підхід, де сторінки BookCatalog складаються з форм, таблиць, фільтрів, карток, кнопок і модальних вікон. Така структура підвищує зрозумілість і тестованість frontend, оскільки помилки легше локалізувати в невеликих відповідальних компонентах.

Компоненти React інкапсулюють логіку та відображення окремих частин інтерфейсу. Повторювана логіка завантаження даних, валідації, фільтрації та обробки помилок може бути винесена в окремі frontend-сервіси й допоміжні функції, що відповідає принципу DRY.

Оновлення стану сторінок у React пов'язане з використанням virtual DOM, завдяки чому змінюються лише потрібні частини інтерфейсу. Для сторінок із великими списками даних у React можуть застосовуватися техніки віртуалізації,

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		

коли відображається лише видима частина елементів, що зменшує навантаження на інтерфейс.

Vite використовується для запуску та збірки клієнтської частини, оскільки проєкт має багато сторінок і потребує швидкої перевірки змін під час розробки.

React Router DOM організовує маршрути для гостьової частини, користувацької панелі, адміністративної панелі та читацького кабінету.

Axios застосовується для запитів до REST API.

Для оформлення інтерфейсу використано Tailwind CSS. Він підходить для системи з багатьма dashboard-сторінками, формами, таблицями та панелями, де потрібен однаковий стиль без дублювання окремих CSS-файлів. Heroicons застосовується для позначення типових дій: перегляду, редагування, видалення, пошуку, завантаження та повернення.

React Helmet Async керує заголовками й метаданими публічних сторінок.

Серверна частина побудована на .NET 8 та ASP.NET Core Web API. .NET 8 обрано як платформу для типізованої backend-реалізації, оскільки серверна частина BookCatalog має підтримувати не лише функціональні операції, а й подальший розвиток, безпеку та супровід системи [29].

Ще однією причиною стало те, що ASP.NET Core фреймворк орієнтований на кросплатформну веброзробку та модульну побудову застосунків [35, с. 4–7]. Модульна структура дає змогу підключати лише потрібні компоненти платформи [35, с. 15].

Для збереження даних обрано Microsoft SQL Server 2019. Предметна область має багато пов'язаних сутностей. Реляційна модель подає дані як набір пов'язаних таблиць, тому підходить для структурованого бібліотечного обліку. Транзакційність потрібна для операцій, які змінюють кілька таблиць. Для українських назв книг, авторів, читачів і бібліотек використовується сортування Ukrainian_CI_AS.

Entity Framework Core виконує роль ORM-рівня між C#-моделями та SQL Server. Він дає змогу описувати сутності, налаштовувати відповідність між

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

класами й таблицями, виконувати запити, оновлювати дані та керувати змінами структури БД через міграції [29].

Авторизація реалізована на основі JWT. Після входу користувач або читач отримує токен, який передається із захищеними API-запитами. Backend перевіряє токен, роль і право виконання дії. Для цього використано Microsoft.AspNetCore.Authentication.JwtBearer, System.IdentityModel.Tokens.Jwt, ASP.NET Core Authorization і рольову модель доступу. Такий підхід відповідає структурі системи, де гість, користувач, власник бібліотеки, працівник, читач і адміністратор мають різні сценарії роботи. Безпека вебзастосунку також включає HTTPS, контроль CORS і захист API від типових вебризиків [35].

Для формування бібліотечних документів використано DocumentFormat.OpenXml. Він потрібен для створення Word-документів на основі даних системи.

Файли, зображення, аватари й шаблони зберігаються у файловому сховищі, а в БД достатньо зберігати пов'язані метадані та шляхи.

Для системних email-повідомлень використано MailKit і SMTP. Параметри поштового сервісу винесено в конфігурацію. Це відповідає рекомендації зберігати параметри середовища, рядки підключення й службові налаштування поза прикладною логікою [35, с. 304–305].

API документується й перевіряється через Swashbuckle.AspNetCore та Swagger UI. Це дає змогу переглядати endpoint-и, параметри запитів і структуру відповідей під час розробки.

Інфраструктурний рівень побудовано на Docker і Docker Compose. Контейнери ізолюють процеси та залежності, використовуючи ядро хостової операційної системи [29, с. 260–261]. Docker Compose описує застосунки, що складаються з кількох контейнерів, мереж і томів, та запускає їх як єдину систему.

У BookCatalog окремими сервісами запускаються frontend, backend і SQL Server. Для БД використовується volume для збереження даних, внутрішня

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

мережа для взаємодії сервісів, змінні середовища для конфігурації та healthcheck для перевірки готовності БД перед запуском backend.

Отже, обраний стек відповідає структурі BookCatalog. React формує динамічний інтерфейс, ASP.NET Core Web API реалізує захищений REST API, SQL Server зберігає пов'язані бібліотечні дані, EF Core забезпечує доступ до БД, JWT забезпечують авторизацію, OpenXML формує документи, MailKit обробляє email-повідомлення, Swagger і тестові інструменти перевіряють API та сценарії, а Docker Compose забезпечує узгоджений запуск основних сервісів.

3.4 Висновки по розділу

У третьому розділі спроектовано архітектуру автоматизованої бібліотечно-інформаційної системи BookCatalog. Систему визначено як веборієнтований клієнт-серверний застосунок із поділом на клієнтську та серверну частину, реляційну базу даних, файлове сховище та середовище розгортання.

У межах моделювання визначено основних акторів системи. Ними є гість, зареєстрований користувач, власник та працівник бібліотеки, читач й адміністратор. Для них описано сценарії роботи з бібліотекою, книжковим фондом, читачами, підручниками, документами, підписками, зверненнями та адміністративними функціями. Окремо змодельовано стани ключових об'єктів. ER-модель відображає основні сутності та зв'язки між ними.

Для реалізації системи обрано стек React, ASP.NET Core Web API, Microsoft SQL Server, Docker Compose. React використовується для клієнтського інтерфейсу, ASP.NET Core Web API — для серверної логіки та REST API, SQL Server — для збереження структурованих бібліотечних даних, JWT — для розмежування доступу. Docker Compose забезпечує запуск клієнтської, серверної частини і бази даних як пов'язаних сервісів.

Запропоновані архітектурні, модельні й технологічні рішення утворюють основу для реалізації BookCatalog.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 4. РЕАЛІЗАЦІЯ ПРОДУКТУ

4.1 Опис розробки програмного коду

Реалізація поділена на клієнтську та серверну частину, базу даних, файловий рівень та інфраструктуру запуску. Це не змішує інтерфейс, бізнес-логіку, доступ до даних і роботу з файлами, і по цій причині структура системи є зрозумілою для розвитку, тестування та супроводу [29, с. 485].

Серверна частина реалізована на ASP.NET Core Web API з використанням .NET 8. У корені проєкту розміщено конфігураційні файли, файл запуску Program.cs, файл проєкту BookCatAPI.csproj, Dockerfile тощо. Основний код поділено на каталоги відповідно до їхнього призначення. Узагальнену структуру каталогів серверної частини подано на рисунку 4.1.

```
oksanka@debian:~/Documents/book-catalog-v1.0.5/api$ tree -d -L 2 --noreport | tail -n +2
├── Controllers
├── Data
├── Migrations
├── Models
│   └── DTOs
├── Options
├── Properties
├── Seeders
├── Services
│   └── Interfaces
├── wwwroot
└── Templates
```

Рисунок 4.1 – Структура каталогів серверної частини BookCatalog

Файл Program.cs є точкою запуску. Тут реєструються залежності, підключення сервісів, налаштування контексту бази даних, авторизації, конфігураційних параметрів і middleware. У .NET 8 DI-контейнер відповідає за створення, передавання та життєвий цикл об'єктів, тому реєстрація сервісів у цьому файлі є частиною конфігурації застосунку [29, с. 134–135].

Обробка HTTP-запитів винесена в контролери. В ASP.NET Core маршрути зіставляють URL із контролерами та їхніми action-методами, після чого

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

потрібний контролер отримується через DI-контейнер [29 с. 438–440]. Контролери виконують базову перевірку вхідних даних, отримують інформацію про поточного користувача за необхідності та передають виконання відповідному сервісу. Не містять складної бізнес-логіки, оскільки їхнє завдання полягає в маршрутизації запиту й формуванні відповіді для фронтенд.

Бізнес-логіка реалізована в сервісному шарі. Саме сервіси виконують перевірку ролі, належності до бібліотеки чи оплати, після чого здійснюють відповідні операції бізнес-процесів. Для зменшення зв'язаності між контролерами та реалізаціями використано інтерфейси. Це спрощує тестування та розвиток АБІС [34 с. 41–43].

Послідовність обробки запиту побудована за одним принципом. Клієнтська частина надсилає HTTP-запит до відповідного endpoint-a, після чого контролер приймає його і передає до сервісного шару. Сервіс виконує перевірки, застосовує бізнес-правила та за потреби звертається до BookCatDbContext для отримання або зміни даних у базі. Після виконання результат перетворюється у DTO або службову відповідь і повертається клієнтській частині.

Моделі описують основні сутності (користувачів, бібліотеки, книги, читачів, підручники тощо). Окремо виділено DTO, які використовуються для передавання лише набору властивостей, необхідних для конкретної операції, без повної доменної сутності [29 с. 156].

Доступ до бази даних здійснюється через Entity Framework Core. Центральним елементом є BookCatDbContext, де описано набори сутностей, зв'язки між ними, індекси, правила видалення, обмеження та типи полів. DbContext у EF Core представляє частковий знімок бази даних у пам'яті, містить конфігурацію відображення сутностей і синхронізує зміни з реальною базою даних [29 с. 312].

Зміни структури бази даних фіксуються через міграції, що дозволяє послідовно оновлювати схему BookCatDB без ручного редагування [35 с. 129–130].

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

Роботу з файлами винесено в окремі сервіси. АБІС підтримує завантаження зображень, аватарів, файлів бібліотеки та формування документів DOCX. У базі даних зберігаються тільки службові відомості про файл. А самі файли не записуються в базу даних як бінарні дані. Операції з файлами та зовнішніми ресурсами належать до I/O-bound завдань, тому їх потрібно виконувати асинхронно й ізолювати в окремих компонентах [36 с. 217–218].

Клієнтська частина реалізована на React, який підтримує декларативний підхід, із використанням Vite.

Vite забезпечує швидке середовище розробки з мінімальною кількістю початкових налаштувань. У React використовується JSX, який робить дерево елементів читабельним завдяки XML-подібному синтаксису. Основний каталог src організовано за функціональним принципом. Структуру каталогів клієнтської частини наведено на рисунку 4.2.

```

• oksanka@debian:~/Documents/book-catalog-v1.0.5/frontend/src$ tree
--noreport | tail -n +2
├── api
├── assets
│   ├── image
│   └── logo
├── components
│   ├── shared
│   │   └── guest
│   └── ui
├── context
├── hooks
├── layouts
├── pages
│   ├── admin
│   ├── auth
│   ├── guest
│   │   └── sections
│   ├── reader
│   └── user
├── test
└── utils
• oksanka@debian:~/Documents/book-catalog-v1.0.5/frontend/src$ █

```

Рисунок 4.2 – Структура каталогів клієнтської частини BookCatalog

Маршрутизація налаштована у файлі App.jsx. Інтерфейс поділено на зони відповідно до ролей і сценаріїв користувачів. Це відкриті сторінки, дашборд бібліотекаря, адміністративна панель і кабінет читача. Для цих зон використано окремі layout-компоненти, які містять спільні елементи сторінок, зокрема

навігацію, хедер, сайдбар та основну область вмісту. Це прибирає дублювання однакової структури на сторінках.

Взаємодію з backend винесено в окремий API-шар. Сторінки та компоненти не формують HTTP-запити напряму, а звертаються до відповідних сервісних функцій. Завдяки цьому логіка взаємодії із сервером не змішується з логікою відображення інтерфейсу. Якщо змінюється адреса endpoint-а, формат відповіді або спосіб обробки помилки, ці зміни вносяться в API-сервісі, а не в кожному компоненті окремо. Такий підхід підтримує separation of concerns, де компонент відповідає за інтерфейс і поведінку сторінки, а робота з API ізолюється в окремому шарі [37, с. 52–53].

Для централізації запитів використано спільний екземпляр Axios. У файлі axiosInstance.jsx задається базова адреса API, службові заголовки, таймаут і interceptor для додавання JWT-токена до авторизованих запитів. Після входу користувача токен передається в заголовок Authorization, тому запити до захищених endpoint-ів проходять перевірку на backend. Це усуває повторення авторизаційної логіки в кожному сервісі й робить механізм взаємодії з API більш керованим. Фрагмент реалізації спільного екземпляра Axios подано на рисунку 4.4.

```
frontend > src > axiosInstance.jsx > ...
1  import axios from "axios"; 56.8k (gzipped: 21.2k)
2
3  const axiosInstance = axios.create({
4    baseURL: "https://localhost:7104/api",
5    withCredentials: true,
6    timeout: 10000,
7    headers: {
8      "X-Requested-From": "BookCatApp",
9    },
10  });
11
12  axiosInstance.interceptors.request.use((config) => {
13    const token = localStorage.getItem("token");
14
15    if (token) {
16      config.headers.Authorization = `Bearer ${token}`;
17    }
18
19    return config;
20  });
21
22  export default axiosInstance;
```

Рисунок 4.3 – Реалізації авторизованих запитів у axiosInstance.jsx

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

Глобальні дані винесено в контексти. AuthContext використовується для збереження стану авторизації, даних поточного користувача та ролей, а ThemeContext — для теми оформлення. Адже React Context API зручно використовувати для даних, які потрібні багатьом компонентам, оскільки це дає можливість передавати стан без прокидання props через кожен проміжний рівень дерева компонентів.

Повторювана логіка розміщена в hooks і utils, де зосереджено допоміжні дії для роботи з файлами, перевірки email, пароля, обов'язкових полів тощо.

У backend класи, моделі, контролери, сервіси й DTO називаються в PascalCase. Інтерфейси мають префікс I. Контролери завершуються суфіксом Controller, сервіси — Service, а об'єкти передавання даних — Dto.

У frontend React-компоненти та сторінки також називаються в PascalCase, а функції, змінні стану, об'єкти фільтрів і методи API-сервісів — у camelCase.

Змістовні назви файлів, змінних, функцій і компонентів полегшують розуміння коду та спрощують пошук помилок.

Запуск системи організовано через Docker Compose. Конфігурація об'єднує frontend, backend та MS SQL Server у пов'язане середовище виконання. Для збереження даних і файлів використовуються окремі volumes, що дає змогу не втрачати інформацію після перезапуску контейнерів.

Отже, реалізація BookCatalog побудована як клієнт-серверний вебзастосунок із поділом відповідальності. Така структура дає змогу розвивати модулі BookCatalog без змішування відповідальностей.

4.2 Використані алгоритми та структури даних

У BookCatalog алгоритми описують прикладні сценарії шкільної бібліотеки. Як от наприклад, пошук записів, видачу й повернення фондів, формування документів, статистику та міжбібліотечний обмін.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

Основна логіка винесена в сервіси. За роллю вони відповідають Application Service, оскільки координують сценарії застосунку й не переносять цю логіку в контролери [34 с. 41–43].

Доступ до даних реалізовано через Entity Framework Core. DbContext і DbSet інкапсулюють роботу з таблицями та дають змогу працювати з об'єктною моделлю без SQL-запитів у бізнес-логіці, що відповідає ідеї Repository pattern [29 с. 156–157]. Один екземпляр DbContext накопичує зміни й зберігає їх через SaveChangesAsync, тому в операціях видачі, повернення й обміну він виконує роль Unit of Work [29, с 313-333].

У каталогах книг, підручників, читачів, документів і заявок використано динамічну фільтрацію. Сервіс формує базову вибірку, після чого додає умови відповідно до параметрів із фронтенд. Після фільтрації виконується пагінація, де підраховується кількість результатів, перевіряється номер сторінки, а до запиту застосовуються Skip і Take. Послідовність цього алгоритму наведено на рисунку 4.4.

```

367     var query = _context.Books
368     .Include(b => b.Invoice)
369     .Include(b => b.BookAuthors).ThenInclude(ba => ba.Author)
370     .Include(b => b.BookEditors).ThenInclude(be => be.Editor)
371     .Where(b => b.LibraryId == userLibrary.LibraryId && b.WriteOffDate != null)
372     .AsQueryable();
373
374     if (!string.IsNullOrEmpty(filter.Title))
375     {
376         query = query.Where(b => b.Title.Contains(filter.Title));
377     }
378     if (!string.IsNullOrEmpty(filter.InventoryNumber))
379     {
380         query = query.Where(b => b.InventoryNumber == filter.InventoryNumber);
381     }
382     if (!string.IsNullOrEmpty(filter.Udc))
383     {
384         query = query.Where(b => b.Udc == filter.Udc);
385     }
386     if (filter.Price.HasValue)
387     {
388         query = query.Where(b => b.Price == filter.Price.Value);
389     }
390     if (filter.PublicationYear.HasValue)
391     {
392         query = query.Where(b => b.PublishYear == filter.PublicationYear.Value);
393     }
394     if (!string.IsNullOrEmpty(filter.Isbn))
395     {
396         query = query.Where(b => b.Isbn != null && b.Isbn.Contains(filter.Isbn));
397     }

```

```

447     var totalItems = await query.CountAsync();
448     var totalSum = await query.SumAsync(b => b.Price ?? 0);
449
450     var allMatchedIds = await query.Select(b => b.Id).ToListAsync();
451
452
453     int pageSize = 15;
454     var totalPages = (int)Math.Ceiling(totalItems / (double)pageSize);
455
456     if (currentPage > totalPages || currentPage < 1)
457     {
458         currentPage = 1;
459     }
460     var books = await query
461         .OrderByDescending(b => b.WriteOffDate)
462         .Skip((currentPage - 1) * pageSize)
463         .Take(pageSize)
464         .ToListAsync();

```

Рисунок 4.4 – Алгоритм фільтрації та пагінації каталогу

Це обмежує обсяг даних, які передаються користувачу за один запит [35, с. 131]. Приклад роботи фільтрів і пагінації в інтерфейсі каталогу підручників показано на рисунку 4.5.

```

77     public async Task<TextbooksPagedResponseDto> GetCatalogAsync(
78         int userId,
79         int roleType,
80         int currentPage,
81         TextbookSearchDto dto)
82     {
83         var userLibrary = await GetPaidUserLibraryAsync(userId, roleType);
84
85         var query = GetTextbookQuery()
86             .Where(t => t.LibraryId == userLibrary.LibraryId &&
87                 t.WriteOffDate == null);
88
89         query = ApplySearch(query, dto, includeWeededDate: false);
90
91         return await BuildPagedResponseAsync(query, currentPage, isWeeded: false);
92     }

```

```

507     private IQueryable<Textbook> ApplySearch(
508         IQueryable<Textbook> query,
509         TextbookSearchDto dto,
510         bool includeWeededDate)
511     {
512         if (dto.Grade.HasValue)
513             query = query.Where(t => t.Grade == dto.Grade.Value);
514
515         if (!string.IsNullOrEmpty(dto.Title))
516         {
517             var title = dto.Title.Trim().ToLower();
518
519             query = query.Where(t => t.Title.ToLower().Contains(title));
520         }
521
522         if (dto.PublicationYear.HasValue)
523             query = query.Where(t => t.PublicationYear == dto.PublicationYear.Value);

```

```

589     var items = await query
590         .OrderBy(t => t.Grade)
591         .ThenBy(t => t.Title)
592         .Skip((currentPage - 1) * PageSize)
593         .Take(PageSize)
594         .ToListAsync();

```

Рисунок 4.5 – Приклад фільтрації та пагінації каталогу підручників

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

Для пошуку й контролю цілісності в базі налаштовано індекси та унікальні обмеження. Вони застосовані до значень, які не можуть дублюватися. Це код підключення до бібліотеки, інвентарний номер книги та номер читача в межах бібліотеки, логін читачького акаунта тощо.

Для активних видач книг використано `filtered unique index`, щоб одна книга не могла одночасно перебувати у двох незакритих видачах. Індекси розміщено на колонках, що використовуються у фільтрації, групуванні та сортуванні [38, с.154–155].

В обліку підручників доступна кількість обчислюється як $\text{AvailableQuantity} = \text{LibraryFundQuantity} - \text{DistributedQuantity}$. Резерв для обміну визначається формулою $\text{Reserve} = \max(\text{LibraryFundQuantity} - \text{StudentNeed} - \text{TeacherNeed}, 0)$, а дефіцит — $\text{Deficit} = \max(\text{StudentNeed} + \text{TeacherNeed} - \text{LibraryFundQuantity}, 0)$. Логіку цих розрахунків показано на рисунку 4.6. Для одного запису складність таких обчислень становить $O(1)$, для каталогу — $O(n)$.

```

726     private TextbookDetailsDto MapDetails(Textbook textbook)
727     {
728         var totalNeed = textbook.StudentNeed + textbook.TeacherNeed;
729         var reserve = Math.Max(0, textbook.LibraryFundQuantity - totalNeed);
730         var deficit = Math.Max(0, totalNeed - textbook.LibraryFundQuantity);
731         var availableForDistribution = Math.Max(0, textbook.LibraryFundQuantity - textbook.DistributedQuantity);

```

Рисунок 4.6 – Розрахунок залишку, резерву та дефіциту підручників

Під час видачі підручників система перевіряє належність підручника до бібліотеки, ознаку списання, доступний залишок і тип отримувача. Після створення видачі збільшується `DistributedQuantity`. Повернення фіксує повернуті, втрачені й пошкоджені примірники. Їхня сума не може перевищувати залишок за видачею. Якщо повернено частину фонду, запис переходить у `PartiallyReturned`, якщо всю кількість — у `Closed`.

Операції, що змінюють кілька пов'язаних записів, виконуються транзакцією. Зміни зберігаються лише після успішного проходження всіх перевірок [35, с. 98]. Узагальнений порядок видачі та повернення підручників наведено на рисунку 4.7.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

```

291     var distribution = new TextbookDistribution
292     {
293         LibraryId = userLibrary.LibraryId,
294         TextbookId = textbook.Id,
295         ClassName = NormalizeNullable(dto.ClassName),
296         RecipientType = dto.RecipientType,
297         TeacherFullName = NormalizeNullable(dto.TeacherFullName),
298         Quantity = dto.Quantity,
299         DistributionDate = dto.DistributionDate ?? DateTime.UtcNow,
300         Status = TextbookDistributionStatus.Active,
301         Note = NormalizeNullable(dto.Note),
302         IssuedByUserId = userId,
303         CreatedAt = DateTime.UtcNow,
304         UpdatedAt = DateTime.UtcNow
305     };
306
307     textbook.DistributedQuantity += dto.Quantity;
308     textbook.UpdatedAt = DateTime.UtcNow;
309
310     _context.TextbookDistributions.Add(distribution);
311
312     await _context.SaveChangesAsync();
313     await transaction.CommitAsync();

```

```

416     textbook.DistributedQuantity -= newClosed;
417     textbook.LostQuantity += dto.LostQuantity;
418     textbook.DamagedQuantity += dto.DamagedQuantity;
419     textbook.LibraryFundQuantity -= dto.LostQuantity + dto.DamagedQuantity;
420
421     if (textbook.DistributedQuantity < 0)
422         textbook.DistributedQuantity = 0;
423
424     if (textbook.LibraryFundQuantity < 0)
425         textbook.LibraryFundQuantity = 0;
426
427     var totalClosedAfter = alreadyClosed + newClosed;
428
429     distribution.Status = totalClosedAfter == distribution.Quantity
430         ? TextbookDistributionStatus.Closed
431         : TextbookDistributionStatus.PartiallyReturned;

```

Рисунок 4.7 – Алгоритм видачі та повернення підручників

Статуси реалізовано через enum-типи: BookLoanStatus, TextbookDistributionStatus і TextbookExchangeRequestStatus. Вони замінюють довільні текстові значення фіксованим набором станів і спрощують перевірку переходів. У проєкті це не повний State pattern, оскільки окремі класи станів не створювались, а використано спрощену статусну модель на основі enum і умов [35, с. 49].

Міжбібліотечний обмін реалізовано в TextbookExchangeService. Пошук резервних підручників виключає власну бібліотеку користувача, враховує лише бібліотеки з увімкненим обміном, перевіряє доступність фонду й відстань між

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

бібліотеками. Завершення заявки оновлює фонди двох бібліотек у межах однієї транзакції. Послідовність міжбібліотечного обміну показано на рисунку 4.8.

```

323     var query = _context.Textbooks
324         .Include(t => t.Library)
325         .Where(t =>
326             t.LibraryId != currentLibrary.Id &&
327             t.WriteOffDate == null &&
328             t.IsAvailableForExchange &&
329             t.Library.IsExchangeEnabled
330         );

364         ReserveQuantity = CalculateReserve(t.LibraryFundQuantity, t.StudentNeed, t.TeacherNeed),
365         ExchangeNote = t.ExchangeNote
366     });
367     .Where(x => x.ReserveQuantity >= minReserve)
368     .OrderBy(x => x.DistanceKm ?? double.MaxValue)
369     .ThenBy(x => x.LibraryName)
370     .ToList();

503     if (quantity > reserve)
504         throw new InvalidOperationException($"Недостатньо резерву для завершення обміну. Доступно: {reserve}.");
505
506     providerTextbook.LibraryFundQuantity -= quantity;
507     providerTextbook.UpdatedAt = DateTime.UtcNow;

552     else
553     {
554         requesterTextbook.LibraryFundQuantity += quantity;
555         requesterTextbook.UpdatedAt = DateTime.UtcNow;
556     }

```

Рисунок 4.8 – Алгоритм міжбібліотечного обміну підручниками

Агрегування даних використано в AdminStatisticsService і MyLibraryStatisticsService. Перший сервіс формує загальні показники системи, другий обмежує статистику бібліотекою поточного користувача. Для підрахунків, сум і помісячних даних застосовується групування. Рівень деталізації визначається так, щоб додаткові колонки не змінювали кількість рядків у результаті [38, с. 17–25].

Модуль документів містить алгоритми генерації файлів. DocumentGenerationService створює службові документи за шаблонами, DocumentFileService надає доступ до готових файлів, а DocumentBibliographyService формує бібліографічні списки й каталожні картки. Для акту списання використовуються списані книги за періодом і шаблон writeOffAct.docx, для інвентарної книги — активні книги та шаблон InventoryBook.docx. Після створення файл зберігається у файловій системі, а запис про нього додається до таблиці Documents. Алгоритм генерації документа

наведено на рисунку 4.9.

```
63     if (dto.DocumentType == "ActOfWeeding")
64     {
65         if (dto.StartDate == null || dto.EndDate == null)
66             throw new ArgumentException("Для акту списання необхідно вказати початкову та кінцеву дати.");
67
68         var from = dto.StartDate.Value;
69         var to = dto.EndDate.Value;
70
71         books = await _context.Books
72             .Include(b => b.BookAuthors).ThenInclude(ba => ba.Author)
73             .Include(b => b.Invoice)
74             .Where(b =>
75                 b.LibraryId == libraryId &&
76                 b.WriteOffDate != null &&
77                 b.WriteOffDate >= from &&
78                 b.WriteOffDate <= to)
79             .OrderBy(b => b.InventoryNumber)
80             .ToListAsync();
81
82         GenerateWriteOffAct(fullFilePath, books);
83     }
84     else if (dto.DocumentType == "SummaryBook")
85     {
86         books = await _context.Books
87             .Include(b => b.BookAuthors).ThenInclude(ba => ba.Author)
88             .Include(b => b.Invoice)
89             .Where(b => b.LibraryId == libraryId && b.WriteOffDate == null)
90             .OrderBy(b => b.InventoryNumber)
91             .ToListAsync();
92
93         GenerateInventoryBook(fullFilePath, books, dto.DocumentName);
94     }
```

Рисунок 4.9 – Алгоритм генерації документа в BookCatalog

Бібліографічний опис формується на основі книги та пов'язаних з нею авторів, редакторів і відповідальних осіб. Сервіс використовує назву, дані про авторів та редакторів, місце видання, видавництво, рік, кількість сторінок та чи має книга ілюстрації. Для списку й каталожної картки застосовується різне подання, але джерело даних спільне.

CurrentUserService інкапсулює роботу з ClaimsPrincipal, JWT claims і типами токенів. Завдяки цьому контролери та сервіси не звертаються безпосередньо до HTTP-контексту, а отримують уже готові значення UserId, ReaderId, RoleType, ReaderLibraryId і тип авторизації. Схему отримання цих даних через CurrentUserService показано на рисунку 4.10. За призначенням цей сервіс відповідає Facade pattern, оскільки надає простий інтерфейс до технічних деталей HTTP-контексту й claims.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

```

19     public int? UserId
20     {
21         get
22         {
23             if (TokenType == "Reader")
24                 return null;
25
26             var user = _httpContextAccessor.HttpContext?.User;
27
28             var userIdClaim = user?.FindFirstValue(ClaimTypes.NameIdentifier)
29                             ?? user?.FindFirst(JwtRegisteredClaimNames.Sub)?.Value
30                             ?? user?.FindFirst("sub")?.Value;
31
32             return int.TryParse(userIdClaim, out int userId)
33                    ? userId
34                    : null;
35         }
36     }

```

Рисунок 4.10 – Отримання даних поточного користувача через
CurrentUserService

Інтеграції з іншими API винесено в окремі сервіси. SmtпEmailService відповідає за надсилання листів, а NominatimGeocodingService — за перетворення адреси бібліотеки на координати. Такі сервіси виконують роль адаптерів, оскільки бізнес-логіка працює з внутрішнім інтерфейсом і не залежить від формату зовнішнього API.

Для обміну даними між backend і frontend використовуються DTO. API не повертає внутрішні моделі бази напряду, а передає об'єкти, сформовані під конкретні сценарії.

Перед відправленням запиту frontend прибирає зайві пробіли, перетворює порожні значення на null і приводить числові поля до потрібного типу.

Асинхронне отримання даних у frontend реалізовано через API-сервіси та React Hooks. Типовий сценарій містить стан завантаження, запит, запис результату або помилки та завершення обробки. useEffect використовується для запитів із компонентів, а useMemo і useCallback — лише там, де є повторні обчислення або передавання функцій у залежності чи дочірні компоненти [37].

Інтерфейс побудовано через композицію компонентів. Загальні частини сторінок винесено в GuestLayout, DashboardLayout, DashboardAdminLayout тощо, а повторювані елементи — в окремі компоненти кнопок, полів, модальних вікон, фільтрів тощо.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

Для адміністративних параметрів використано словник. Записи AdminInfos зберігаються у форматі `FieldName — Content` і перетворюються на `Dictionary<string, string>`, де доступ за ключем у середньому має складність $O(1)$.

Унікальні значення формуються окремими механізмами. Код підключення до бібліотеки генерується як випадкова послідовність із 20 великих латинських літер і цифр, а неповторюваність контролюється індексом. Номер читача створює `LibraryReaderNumberService` через послідовне збільшення `ReaderNumber`. Для файлів і зображень використовується `Guid`, щоб уникати конфліктів назв у файловій системі.

Для частини сутностей застосовано `soft-delete`. Під час списання книги або підручника заповнюється `WriteOffDate`, а запис залишається в базі. Читач і читацький акаунт вимикаються через `IsActive`. Це дає змогу зберігати історію обліку, видач і повернень.

Отже, алгоритмічна частина `BookCatalog` охоплює фільтрацію, пагінацію, облік фондів, транзакційні операції, генерацію документів, статистичні запити та обмін між бібліотеками. Основними структурами даних є реляційні таблиці з індексами, DTO, enum-типи, словники, списки та стан `React`.

4.3 Модульне тестування та налагодження

Модульне тестування `BookCatalog` виконувалося для перевірки окремих backend- і frontend-модулів до етапу системного тестування. Такий рівень тестування призначений для ізольованої перевірки поведінки окремих методів або компонентів [35, с. 48–49].

Backend-частина тестувалася в окремому проєкті `BookCatalog.Tests` для платформи .NET 8. Для написання тестів використано `xUnit`, для перевірки очікуваних результатів — `FluentAssertions`, для створення mock-об'єктів — `Moq`. У тестах, пов'язаних із даними, застосовано `Entity Framework Core InMemory` та `SQLite in-memory`, щоб перевіряти бізнес-логіку без звернення до основної бази даних [28, с. 219–223].

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

Unit-тести backend охоплювали авторизацію, JWT-сервіс, відновлення пароля, акаунти, бібліотеки, книги, читачів, підручники, підписки, теги й статті. Основні перевірки стосувалися нормалізації email, хешування пароля, генерації інвентарного номера, заборони доступу до чужої бібліотеки, неможливості видати вже видану книгу, контролю кількості підручників і перевірки ролей користувачів тощо. Структуру backend unit-тестів наведено в таблиці 4.1.

Таблиця 4.1

Модульні тести backend-частини BookCatalog

Група тестів	Кількість тестів	Основні перевірки
Account / Avatar	19	видалення акаунта, передача ролі власника, перевірка аватарів
Auth	53	реєстрація, вхід, JWT, скидання паролю, claims, ролі користувачів
Libraries	59	створення бібліотеки, учасники, інвентарний номер, оплата, номер читача
Books	23	створення книги, автоматичний інвентарний номер, каталог, списання, відновлення
Readers	22	створення читача, акаунт читача, видача й повернення книги
Textbooks	74	підручники, спецпідручники, обмін, фонд, резерв, дефіцит
Users	30	профіль користувача, статус бібліотеки, аватар
Articles / Plans / Subscriptions / Tags	56	статті, тарифи, підписки, CRUD-операції з тегами
Разом	336	перевірка backend-сервісів, контролерів і бізнес-правил

Найбільше backend unit-тестів припадає на модулі підручників, бібліотек та авторизації, оскільки ці частини містять основні бізнес-правила АБІС.

Frontend-частина тестувалася за допомогою Vitest, React Testing Library та jsdom. Тести будувалися навколо поведінки користувача. Це введення даних у форми, повідомлення про помилки, loading-стани, empty-state, пагінація, фільтрація та виклики API-сервісів.

Для React-компонентів такий підхід дає змогу перевіряти не внутрішню реалізацію, а реакцію інтерфейсу на дії користувача [37 с. 130–143]. Структуру frontend unit/component-тестів наведено в таблиці 4.2.

Unit/component-тести frontend-частини BookCatalog

Група тестів	Кількість тестів	Основні перевірки
API-сервіси	192	формування HTTP-запитів, параметри, тіло запиту, обробка відповідей і помилок
Сторінки авторизації	88	вхід, реєстрація, скидання паролю, валідація форм
Сторінки читача	27	профіль читача, поточні книги, історія видач, токен, помилки API
Сторінки користувача / бібліотекаря	714	книги, читачі, видачі, підручники, оплата, налаштування бібліотеки
App-тест	1	базовий рендер головного компонента застосунку
Разом	1022	перевірка API-сервісів, форм, сторінок і станів інтерфейсу

Найбільша частина frontend-тестів стосується сторінок користувача та бібліотекаря, оскільки саме там реалізовано основні сценарії роботи BookCatalog, які включають додавання книг, створення читачів, видачу й повернення книг, роботу з підручниками, списання, оплату тарифу та налаштування бібліотеки.

Для ізоляції тестів використовувалися mock-об'єкти. На backend-рівні мокувалися email-сервіс, сервіс аватарів і JWT-сервіс. На frontend-рівні мокувалися axiosInstance, API-сервіси, React Router, AuthContext, модальні вікна підтвердження, localStorage і sessionStorage. Це дало змогу перевіряти окремі модулі без запуску всієї системи та без звернення до зовнішніх сервісів.

Для налагодження frontend-тестів було налаштовано очищення DOM після кожного тесту, скидання mock-функцій, очищення localStorage і sessionStorage, а також підміну браузерних API matchMedia, ResizeObserver, IntersectionObserver, scrollTo та scrollIntoView. Такі налаштування зменшують залежність тестів від попереднього стану jsdom.

Отже, у BookCatalog підготовлено тестове середовище для перевірки backend- і frontend-модулів, ізоляції залежностей через mock-об'єкти та локалізації помилок у бізнес-логіці, API або інтерфейсі.

4.4 Висновки по розділу

Описано реалізацію BookCatalog. Backend-частина побудована на ASP.NET Core Web API та .NET 8, frontend-частина — на React і Vite, а збереження даних організовано через MS SQL Server та Entity Framework Core.

У процесі реалізації код поділено за функціональним призначенням. На backend-рівні виділено контролери, сервіси, моделі, DTO, контекст бази даних і допоміжні компоненти. На frontend-рівні окремо організовано сторінки, компоненти, layout-компоненти, API-сервіси, контексти, hooks і утиліти.

Окремо розглянуто алгоритми, які реалізують основні сценарії роботи. Сюди належать фільтрація та пагінація, розрахунок резерву та дефіциту, міжбібліотечний обмін, формування документів і статистичних показників тощо. Для збереження цілісності даних використано індекси, унікальні обмеження, enum-статуси, транзакційні операції, DTO та допоміжні структури даних.

Також у розділі описано модульне тестування та налагодження. Backend-тести реалізовано в окремому проєкті BookCatalog.Tests із використанням xUnit, FluentAssertions, Moq, EF Core InMemory та SQLite in-memory. Frontend-тести побудовано на Vitest, React Testing Library і jsdom. Тестами охоплено сервіси, контролери, API-сервіси, форми, сторінки, стани інтерфейсу, обробку помилок і ключові бізнес-правила.

Отже, у межах розділу було показано, як реалізовано основні модулі BookCatalog, які алгоритми й структури даних використано для роботи системи та як організовано перевірку окремих backend- і frontend-компонентів.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 5. ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ

5.1 Стратегії та методи тестування

Тестування організовано за рівнями, які починаються від перевірки окремих сервісів і компонентів до наскрізних сценаріїв у браузері.

Основна стратегія поєднувала модульне, інтеграційне, компонентне, end-to-end та ручне функціональне тестування.

Модульні тести використовувалися для перевірки бізнес-логіки, інтеграційні — для API, авторизації та роботи з базою даних, frontend-тести — для сторінок, форм і API-сервісів клієнтської частини. E2E-сценарії перевіряли основні дії користувача в інтерфейсі, а ручне тестування застосовувалося для оцінки навігації, повідомлень і логіки переходів між сторінками. Загальну стратегію тестування наведено в таблиці 5.1.

Таблиця 5.1

Стратегія тестування програмного забезпечення BookCatalog

Рівень тестування	Об'єкт перевірки	Інструменти
Модульне тестування backend	сервіси (бізнес-логіка)	xUnit, FluentAssertions, Moq
Інтеграційне тестування backend	API, авторизація, база даних	WebApplicationFactory, Testcontainers.MsSql, Respawn
Тестування API-сервісів frontend	клієнтські сервіси взаємодії з backend	Vitest, mock axiosInstance
Компонентне тестування frontend	сторінки, форми, компоненти	Vitest, React Testing Library, jsdom
End-to-end тестування	наскрізні сценарії користувача	Playwright
Ручне функціональне тестування	інтерфейс і логіка взаємодії	браузер, тестові дані

Backend-частина перевірялася модульними та інтеграційними тестами. Хороший unit-тест має бути швидким, ізольованим, зрозумілим і стабільним за результатами [39, с. 15–16]. Тому для unit-тестів використано xUnit, FluentAssertions і Moq. Mock-об'єкти застосовувалися для залежностей, які не

повинні впливати на результат конкретної перевірки, наприклад email-сервісу під час тестування авторизації або керування акаунтом.

Для перевірок, пов'язаних із даними, використовувалися EF Core InMemory, SQLite in-memory та тестовий контейнер MS SQL Server.

У простіших unit-тестах перевіряється бізнес-логіка ізольовано, а в інтеграційних тестах — повний цикл HTTP-запиту: маршрут, авторизація, статус відповіді, виконання бізнес-правил і зміна даних у базі. Адже інтеграційний тест перевіряє одиницю роботи без повного контролю над усіма реальними залежностями, зокрема базою даних або зовнішніми сервісами [39, с. 18].

Окрема увага приділялася авторизації та контролю доступу. Перевірялися реєстрація, вхід, формування JWT-токена, визначення поточного користувача, обробка відсутнього або некоректного токена, недостатній рівень ролі та спроба доступу до ресурсів іншої бібліотеки. Для таких сценаріїв очікувалися відповіді Unauthorized, Forbidden або NotFound.

Функціональні модулі перевірялися на успішні та помилкові сценарії. Для книг тестувалися створення, фільтрація, списання й відновлення. Для читачів — створення профілю, деактивація, видача та повернення книг. Для підручників — розрахунок резерву, дефіциту, загальної потреби та заборона зменшення фонду нижче вже виданої кількості.

Frontend-частина тестувалася за допомогою Vitest і React Testing Library. Більша частині уваги приділялася поведінці сторінок. Наприклад, наявність потрібних полів, введення даних, повідомлення про помилки, переходи між сторінками та блокування некоректних дій.

Для стабільності тестового середовища після кожного тесту очищувалися DOM, localStorage, sessionStorage і mock-функції, а також підмінялися браузерні API, недоступні в jsdom.

Форми перевірялися на граничні й некоректні значення. Також контролювалося очищення введених даних від зайвих пробілів перед відправленням.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

API-сервіси frontend тестувалися окремо від сторінок. Такі перевірки підтверджували правильність HTTP-методу, endpoint-а, параметрів, тіла запиту та обробки помилок.

End-to-end тестування виконувалося за допомогою Playwright. Воно охоплювало основні користувацькі сценарії. Від реєстрації, входу, створенні бібліотеки та додаванні книг з підручниками до перегляду статистики та керування командою бібліотеки. E2E-тести перевіряють систему з позиції користувача й дають змогу оцінити узгоджену роботу маршрутизації, інтерфейсу та API [39, с. 202–203].

У стратегію також були включені негативні та регресійні перевірки.

Негативні сценарії охоплювали відсутність токена, недостатні права доступу, неоплачену бібліотеку, дублювання email або читацького номера, неіснуючі записи, неправильні параметри запиту й порожні обов’язкові поля.

Регресійне тестування застосовувалося після змін у бізнес-логіці або інтерфейсі, щоб перевірити, чи не були порушені вже реалізовані функції.

Повністю ручна регресійна перевірка є витратною за часом, тому автоматизовані тести зменшують обсяг повторюваних дій [39, с. 3].

Налагодження виконувалося відповідно до рівня, на якому виявлено проблему. Для помилки в окремому методі аналізувався unit-тест, для збою HTTP-запиту — інтеграційний тест API, для проблеми в інтерфейсі — frontend-тест або Playwright-сценарій. Це допомагає швидше визначити джерело дефекту.

5.2 Результати тестування системи

Перед формуванням висновків було виконано запуск backend-тестів у проєкті BookCatalog.Tests. Результат запуску наведено на рисунку 5.1.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		73

```
[testcontainers.org 00:00:45.56] Delete Docker container 77a8a427abb4
[xUnit.net 00:00:53.04] Finished: BookCatalog.Tests
BookCatalog.Tests test succeeded (54,1s)

Test summary: total: 612; failed: 0; succeeded: 612; skipped: 0; duration: 54,1s
Build succeeded in 56,1s
oksanka@debian:~/Documents/book-catalog-v1.0.5$
```

Рисунок 5.1 – Результат запуску backend-тестів BookCatalog

Виконано 612 перевірок, із яких всі завершилися успішно.

Наступним етапом виконано запуск frontend unit/component-тестів. Результат запуску наведено на рисунку 5.2.

```
✓ src/api/_tests_/textbook-services.test.jsx (47 tests) 39ms
✓ src/api/_tests_/special-needs-textbook-service.test.jsx (28 tests) 21ms
✓ src/api/_tests_/request-reader-services.test.jsx (47 tests) 22ms
✓ src/api/_tests_/tag-subscription-services.test.jsx (18 tests) 12ms

Test Files 42 passed (42)
Tests 1075 passed (1075)
Start at 03:49:11
Duration 42.93s (transform 4.34s, setup 6.41s, import 11.13s, tests 76.44s, environment 27.06s)
```

Рисунок 5.2 – Результат запуску frontend unit/component-тестів BookCatalog

За результатом запуску виконано 1075 перевірок, із яких всі завершилися успішно. Цей результат відображає стан перевірки API-сервісів, форм, сторінок та повідомлень про помилки чи реакції інтерфейсу на дії користувача.

Для оцінки покриття frontend-коду було сформовано coverage-звіт. Його результат наведено на рисунку 5.3.

% Coverage report from v8					
File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	58.24	52.61	55.42	58.8	
src	16.66	0	25	16.66	
App.jsx	100	100	100	100	
axiosInstance.jsx	33.33	0	0	33.33	13-19
index.jsx	0	100	100	0	8-17
reportWebVitals.jsx	0	0	0	0	1-8
src/api	97.63	65.64	100	97.61	
adminDataService.jsx	100	100	100	100	
articleService.jsx	100	100	100	100	
authService.jsx	81.57	100	100	81.57	...2-83,92-93,102-103,115-117,128-129,145-146
bookService.jsx	100	70	100	100	59-89
documentService.jsx	100	66.66	100	100	10-23,49,92
libraryService.jsx	100	50	100	100	123-155
planService.jsx	100	100	100	100	
readerService.jsx	100	100	100	100	
requestService.jsx	100	100	100	100	
specialNeedsTextbookService.jsx	100	53.57	100	100	13-90,106-203
subscriptionService.jsx	100	100	100	100	
tagService.jsx	100	100	100	100	
textbookExchangeService.jsx	100	57.14	100	100	13-29,44,59-136

Рисунок 5.3 — Результат покриття frontend-коду тестами

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						74
Змн.	Арк.	№ докум.	Підпис	Дата		

За результатом coverage-звіту показник покриття становив: Statements — 58,24 %, Branches — 52,61 %, Functions — 55,42 %, Lines — 58,8 %. Ці показники використано як допоміжну метрику, оскільки сам відсоток покриття не гарантує якості тестів, але допомагає виявити частини коду, які потребують додаткових перевірок [39, с. 211].

End-to-end тестування виконувалося за допомогою Playwright. Результат запуску E2E-сценаріїв наведено на рисунку 5.4.

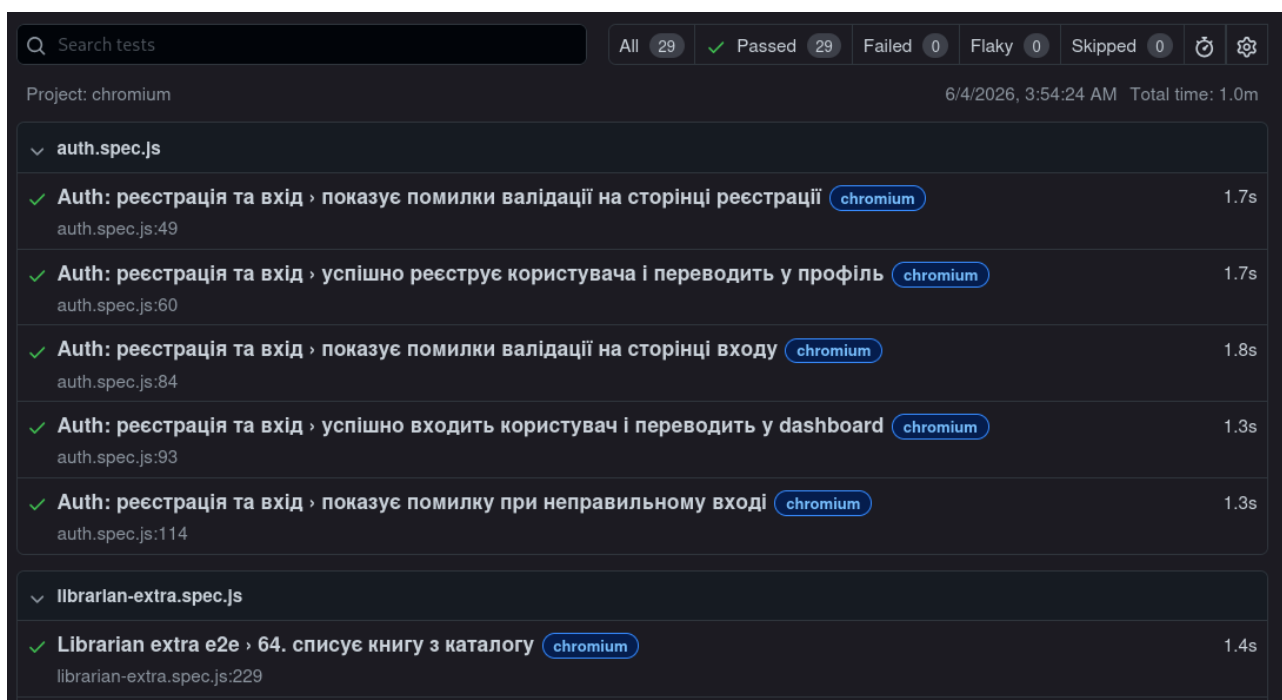


Рисунок 5.4 – Результат запуску Playwright E2E-сценаріїв BookCatalog

За результатом запуску E2E-тестів було виконано 29 сценаріїв, із яких всі завершилися успішно.

У процесі аналізу результатів враховувалися не лише успішні запуски, а й можливі відмови. У динамічному тестуванні дефект проявляється тоді, коли фактичний результат відрізняється від очікуваного.

Отже, результати запуску тестів дають змогу оцінити поточний стан готовності BookCatalog.

5.3 Аналіз ефективності впровадження

АБІС розгорталася за допомогою Docker Compose. Всі частини були запуснені як окремі контейнери в єдиному середовищі. Всі необхідні сервіси, порти, змінні оточення, томи для збереження даних і мережеві параметри описані в одному конфігураційному файлі.

Визначено три основні сервіси: client, bookcatapi і db. Вони працюють у спільній мережі bookcat-network, завдяки чому можуть взаємодіяти між собою за іменами сервісів. Наприклад, серверна частина підключається до бази даних за іменем db, яке використовується в рядку підключення до MS SQL Server.

Сервіс client відповідає за запуск клієнтської частини системи. Він створюється на основі образу Node.js, встановлює залежності проєкту та запускає frontend-застосунок у режимі розробки. Усередині контейнера клієнтська частина працює на порту 3000, а на локальному комп'ютері доступна через порт 8000.

Сервіс bookcatapi відповідає за роботу серверної частини системи, реалізованої на ASP.NET Core. Для роботи передаються змінні оточення, зокрема середовище виконання, рядок підключення до бази даних, параметри HTTPS-сертифіката, SMTP-параметри для надсилання електронних повідомлень тощо.

Сервіс db відповідає за роботу бази даних і використовує образ Microsoft SQL Server 2019. Дані бази зберігаються в директорії db-data, яка підключається до контейнера як том. Завдяки цьому інформація не втрачається після зупинки або перезапуску контейнерів.

Для бази даних налаштовано перевірку працездатності healthcheck, яка виконує тестовий SQL-запит. Це дозволяє запускати backend лише після того, як база даних стала доступною для підключення.

Завантажені файли та сформовані документи зберігаються в директорії uploads, яка монтується в контейнер серверної частини. Це забезпечує

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						76
Змн.	Арк.	№ докум.	Підпис	Дата		

збереження файлів поза межами контейнера та дає змогу використовувати їх після повторного запуску системи.

Основні сервіси Docker Compose для розгортання системи наведено в таблиці 5.2.

Таблиця 5.2

Сервіси Docker Compose для розгортання BookCatalog

Сервіс	Призначення	Технологія	Порти
client	Клієнтська частина системи	Node.js, React/Vite	8000:3000
bookcatapi	Серверна частина та API	ASP.NET Core 8.0	7104, 8080
db	База даних системи	MS SQL Server 2019	1433

Результат запуску контейнерів BookCatalog за допомогою Docker Compose наведено на рисунку 5.5.

```
[+] up 5/5
✓ Image book-catalog-v105-client          Built
✓ Network book-catalog-v105_bookcat-network Created
✓ Container book-catalog-v105-db-1        Healthy
✓ Container book-catalog-v105-bookcatapi-1 Started
✓ Container book-catalog-v105-client-1    Started
oksanka@debian:~/Documents/book-catalog-v1.0.5$
```

Рисунок 5.5 — Результат запуску контейнерів BookCatalog за допомогою Docker Compose

Якщо база даних ще не створена, застосовуються міграції Entity Framework Core, після чого формується структура таблиць, необхідна для роботи системи.

За допомогою сідерів, у базу даних вносяться дані про адміністратора та створення ролей як для користувачів, так і для працівників бібліотеки. Також створюються теги для статей та вносяться дані тарифних планів.

Ефективність впровадження АБІС BookCatalog оцінювалася за якісними критеріями, що відображають основні проблеми ручного ведення бібліотечного обліку. До таких критеріїв належать швидкість пошуку інформації, зменшення ризику дублювання, зручність обліку читачів, контроль видачі й повернення книг, автоматизація роботи з підручниками, формування документів та

розмежування прав доступу. Оскільки система не проходила повноцінної апробації на реальних даних конкретного закладу освіти, результати впровадження подано як очікувані зміни, що можуть бути перевірені під час практичного використання системи. Очікувані зміни після впровадження BookCatalog наведено в таблиці 5.3.

Таблиця 5.3

Очікувані зміни після впровадження BookCatalog

Критерій	До впровадження	Після впровадження
Облік книг	Журнали або окремі файли	Єдиний електронний каталог
Пошук інформації	Перегляд журналів чи таблиць	Пошук і фільтрація в системі
Інвентарні номери	Ризик дублювання	Автоматичний контроль номера
Читачі	Ручне ведення списків	Облік профілів читачів
Видача книг	Складно відстежувати історію	Фіксація видачі й повернення
Підручники	Ручні підрахунки	Розрахунок резерву й дефіциту
Документи	Повторне введення даних	Формування з даних системи
Доступ	Складно контролювати зміни	Ролі та обмеження прав

Отже, використання Docker Compose спрощує розгортання, оскільки всі основні компоненти системи запускаються в ізольованих контейнерах і працюють у спільній мережі. Очікуваний ефект від впровадження системи полягає в централізації бібліотечного обліку, пришвидшенні пошуку інформації, зменшенні ризику дублювання записів та автоматизації формування документів.

5.4 Висновки по розділу

У розділі розглянуто процес тестування та впровадження. Визначено загальну стратегію перевірки, яка поєднує модульне, інтеграційне, компонентне, end-to-end та ручне функціональне тестування. Таким чином змогли охопити як окремі частини системи, так і повні користувацькі сценарії роботи.

Для backend-частини перевірялася бізнес-логіка, API, авторизація та взаємодії з базою даних. Для frontend-частина перевірялися API-сервіси, форми, сторінки, повідомлення про помилки, переходи між маршрутами та реакції інтерфейсу на дії користувача.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						78
Змн.	Арк.	№ докум.	Підпис	Дата		

Окрему роль відіграло end-to-end тестування, яке дозволяє оцінити працездатність системи з позиції кінцевого користувача.

Крім тестування, описано особливості впровадження системи. Розгортання передбачене через Docker Compose. Для початкового запуску необхідно налаштувати середовище, застосувати міграції бази даних, створити за допомогою сідера адміністратора та ролі.

Очікуваний ефект від впровадження BookCatalog полягає в переході від ручного та розрізненого обліку до централізованого електронного каталогу. Система спрощує пошук книг, облік читачів, видач і повернень, розрахунок резерву та дефіциту підручників, а також формування документів на основі вже внесених даних.

Отже, BookCatalog є готовою до впровадження після налаштування серверного середовища, наповнення бази початковими даними та перевірки роботи на реальних бібліотечних записах. Запропонована стратегія тестування та впровадження забезпечує основу для подальшої експлуатації, виявлення помилок, удосконалення функціональності та оцінювання ефективності системи в умовах конкретної бібліотеки.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		79

ВИСНОВКИ

У результаті виконання бакалаврської роботи розроблено автоматизовану бібліотечну інформаційну систему BookCatalog для закладів загальної середньої освіти. Поставлену мету досягнуто, оскільки створений вебзастосунок забезпечує електронний облік книжкового фонду, підручників, читачів, бібліотек, документів і заявок на міжбібліотечний обмін.

Під час роботи проаналізовано особливості діяльності шкільної бібліотеки та наявні програмні рішення для автоматизації бібліотечних процесів. Визначено, що більшість готових систем орієнтовані на великі бібліотеки або потребують складного налаштування. Тому було сформовано вимоги до простого й адаптованого рішення, яке враховує щоденні потреби бібліотекаря.

Для реалізації системи обрано клієнт-серверну архітектуру. Клієнтську частину розроблено засобами React, серверну — за допомогою ASP.NET Core Web API, а для збереження даних використано Microsoft SQL Server.

Розроблена система дає змогу створювати й редагувати записи про книги, вести облік видачі та повернення, працювати з підручниками, формувати документи й переглядати статистичні дані. Це зменшує кількість ручних операцій, спрощує пошук інформації та підвищує точність бібліотечного обліку.

Проведене тестування підтвердило працездатність основних модулів.

Практичне значення роботи полягає в тому, що АБІС BookCatalog може бути використана як основа для цифровізації роботи шкільної бібліотеки. Подальший розвиток системи варто спрямувати на розширення звітності, додавання імпорту й експорту даних, покращення аналітики та перевірку роботи системи на реальних даних конкретного закладу освіти.

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						80
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Про затвердження Положення про бібліотеку загальноосвітнього навчального закладу Міністерства освіти України : Наказ від 14.05.1999 № 139 : станом на 4 серп. 1999 р. URL: <https://zakon.rada.gov.ua/rada/show/v0139281-99#Text> (дата звернення: 27.05.2026).

2. Про схвалення Стратегії розвитку бібліотечної справи на період до 2025 року “Якісні зміни бібліотек для забезпечення сталого розвитку України” : Розпорядж. Каб. Міністрів України від 23.03.2016 № 219-р. URL: <https://zakon.rada.gov.ua/laws/show/219-2016-p#Text> (дата звернення: 27.05.2026).

3. Про затвердження Типових норм часу на основні процеси бібліотечної роботи : Наказ від 29.12.2008 № 1631. URL: <https://zakon.rada.gov.ua/rada/show/v1631655-08#Text> (дата звернення: 28.05.2026).

4. Бруй О. Відставання на 20 років: цифрова трансформація бібліотек та реальні можливості змін. *Читомо | Культура читання і мистецтво книговидання | Усе про книги | Читомо - портал про культуру читання і мистецтво книговидання.* URL: <https://chytomo.com/vidstavannia-na-20-rokiv-tsyfrova-transformatsiia-bibliotek-ta-realni-mozhlyvosti-zmin/> (дата звернення: 27.05.2026).

5. Про схвалення Стратегії розвитку читання на період до 2032 року “Читання як життєва стратегія” та затвердження операційного плану її реалізації на 2023-2025 роки : Розпорядж. Каб. Міністрів України від 03.03.2023 № 190-р : станом на 31 січ. 2026 р. URL: <https://zakon.rada.gov.ua/laws/show/190-2023-p#Text> (дата звернення: 27.05.2026).

6. Про затвердження Інструкції з обліку документів, що знаходяться в бібліотечних фондах : Наказ М-ва культури і туризму України від 03.04.2007 № 22. URL: <https://zakon.rada.gov.ua/laws/show/z0530-07#Text> (дата звернення: 27.05.2026).

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						81
Змн.	Арк.	№ докум.	Підпис	Дата		

7. Бруй О. Як діджиталізувати бібліотечну систему й не наробити помилок. *Читомо | Культура читання і мистецтво книговидання | Усе про книги | Читомо - портал про культуру читання і мистецтво книговидання.*
URL: <https://chytomo.com/iak-didzhytalizuvaty-bibliotechnu-systemu-j-ne-narobyty-pomylok/> (дата звернення: 28.05.2026).

8. Бакан С. Система автоматизації бібліотек «ІРБІС». *Українська бібліотечна енциклопедія.*
URL: <https://ube.nlu.org.ua/article/Система%20автоматизації%20бібліотек%20«ІРБІС»> (дата звернення: 28.05.2026).

9. Обґрунтування закупівлі “Впровадження програмної платформи бібліосервісів для автоматизації бібліотечних процесів ILS FOLIO” / Національна наукова сільськогосподарська бібліотека НААН України. 2024. URL: <https://dnsgb.com.ua/assets/files/2024/11/obgruntuvannya-zakypivli-2024.pdf> (дата звернення: 29.05.2026).

10. Бакан С. Ф. Автоматизована бібліотечна інформаційна система. *Українська бібліотечна енциклопедія.*
URL: <https://ube.nlu.org.ua/article/Автоматизована%20бібліотечна%20інформаційна%20система> (дата звернення: 29.05.2026).

11. Про бібліотеки і бібліотечну справу : Закон України від 27.01.1995 № 32/95-ВР : станом на 28 серп. 2025 р.
URL: <https://zakon.rada.gov.ua/laws/show/32/95-вр#Text> (дата звернення: 29.05.2026).

12. Бібліографічний опис. *Бібліотека рівненського обласного інституту післядипломної педагогічної освіти.*
URL: <https://sites.google.com/roippo.org.ua/roippo-bibl/інформація/бібліографічний-опис> (дата звернення: 29.05.2026).

13. Про впровадження Універсальної десятикової класифікації в практику роботи бібліотек : Наказ від 26.06.2017 № 929.
URL: <https://zakon.rada.gov.ua/rada/show/v0929729-17#Text> (дата звернення: 29.05.2026).

					БР.ІІ - 18.00.00.000 ПЗ	Арк.
						82
Змн.	Арк.	№ докум.	Підпис	Дата		

14. Про захист персональних даних : Закон України від 01.06.2010 № 2297-VI : станом на 14 черв. 2025 р. URL: <https://zakon.rada.gov.ua/laws/show/2297-17/conv> (дата звернення: 29.05.2026).

15. Головна сторінка. Український Фондовий Дім. URL: <https://ush.com.ua/> (дата звернення: 29.05.2026).

16. UniLib – професійна платформа для сучасної бібліотеки. UniLib. URL: <https://unilib.com.ua/> (дата звернення: 29.05.2026).

17. Koha library software. Official Website of Koha Library Software. URL: <https://koha-community.org> (дата звернення: 29.05.2026).

18. Installing Koha – Koha manual documentation. Official Website of Koha Library Software. URL: <https://koha-community.org/manual/latest/en/html/installation.html> (дата звернення: 29.05.2026).

19. Гуралюк А. Г., Вараксіна Н. В. Сучасні автоматизовані бібліотечні інформаційні системи : аналітичний огляд. Київ : Державна науково-педагогічна бібліотека України імені В. О. Сухомлинського, 2022. 14 с. URL: https://dnpb.gov.ua/wp-content/uploads/2022/12/Avtomatyizovani_bibliotechni_informatsiyni_systemy-2022.pdf (дата звернення: 29.05.2026).

20. Безкоштовні АБІС – вільне відкрите програмне забезпечення | Національна бібліотека України імені В. І. Вернадського. Національна бібліотека України імені В. І. Вернадського. URL: <https://nbuv.gov.ua/node/1336> (дата звернення: 29.05.2026).

21. Evergreen Downloads. Evergreen. URL: <https://evergreen-ils.org/egdownloads/> (дата звернення: 29.05.2026).

22. OpenBiblio: An easy to use free integrated library software | AIMS. Homepage | AIMS. URL: <https://aims.fao.org/news/openbiblio-easy-use-free-integrated-library-software> (дата звернення: 29.05.2026).

23. PMB Integrated Library System – Nepal Knowledge Center Services. Nepal Knowledge Center Services. URL: <https://nkcs.org.np/wpress/index.php/pmb/> (дата звернення: 29.05.2026).

					БР.ІІІ - 18.00.00.000 ПЗ	Арк.
						83
Змн.	Арк.	№ докум.	Підпис	Дата		

24. Вільна автоматизована бібліотечна інтегрована система – АБІС. *Librarika: The Free Integrated Library System (ILS)*. URL: <https://librarika.com/> (дата звернення: 29.05.2026).

25. CDS/ISIS (Computer Documentation System / Integrated System Information Services) – пакет прикладних програм для побудови автоматизованих інформаційних систем бібліотек, архівів і музеїв | Національна бібліотека України імені В. І. Вернадського. *Національна бібліотека України імені В. І. Вернадського*. URL: <https://nbn.gov.ua/node/1743> (дата звернення: 29.05.2026).

26. About FOLIO. *FOLIO* | *folio.org*. URL: <https://folio.org/about/> (дата звернення: 29.05.2026).

27. The FOLIO Library Services Platform (LSP). *EBSCO Information Services, Inc.* | *about.ebsco.com*. URL: <https://about.ebsco.com/academic-libraries/products/ebsco-folio> (дата звернення: 29.05.2026).

28. Koelsch G. Requirements Writing for System Engineering. Berkeley, CA : Apress, 2016. URL: <https://doi.org/10.1007/978-1-4842-2099-3> (дата звернення: 30.05.2026).

29. Baptista G., Abbruzzese F. Software Architecture with C# 12 and .NET 8. 4-те вид. Birmingham : Packt Publishing Ltd., 2024. 726 с.

30. Patton J., Economy P. User Story Mapping / ред.: М. Treseler, А. Jollymore. Sebastopol, CA : O'Reilly Media, 2014. 276 с.

31. A guide to the project management body of knowledge. Pennsylvania : Project Management Institute, Inc., 2000. 216 с.

32. Bass L., Clements P., Kazman R. Software Architecture in Practice. 3-тє вид. Westford : Pearson Education, Inc., 2013. 589 с.

33. Бруй О. Збалансована система показників у стратегічному управлінні бібліотекою = Strategic Management of Library on Base of Balanced Scorecard : thesis. 2018. URL: http://eprints.rclis.org/38873/1/Bruy_diss.pdf (дата звернення: 30.05.2026).

34. Vernon V. Domain-Driven Design Distilled. Crawfordsville : Pearson Education, Inc., 2016. 134 с.

					БР.ІІІ - 18.00.00.000 ПЗ	Арк.
						84
Змн.	Арк.	№ докум.	Підпис	Дата		

35. Metzgar D. .NET Core in Action. Shelter Island, NY : Manning Publications, 2018. 288 с.

36. Terrell R. Concurrency in .NET: Modern patterns of concurrent and parallel programming. Manning Publications, 2018. 568 с.

37. Okonta A. O. React.js Design Patterns. London : BPB Online, 2023. 180 с.

38. SQL Query Design Patterns and Best Practices. Birmingham : Packt Publishing Ltd., 2023. 249 с.

39. Oshero R. Art of Unit Testing, Third Edition: With Examples in JavaScript. Manning Publications Co. LLC, 2022.

					БР.ІІ - 18.00.00.000 ІЗ	Арк.
						85
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

ДОДАТОК А

Таблиця А.1

Порівняння існуючих АБІС у контексті використання в українських шкільних бібліотеках

Назва	Походження	Для яких бібліотек	Чому для українських шкільних бібліотек не підходить
УФД/Бібліотека	Україна	публічні, університетські, спеціалізовані, відомчі	Є комерційною системою, потребує супроводу постачальника та може бути надмірною для школи. Не має повної орієнтації на облік підручників, класи, навчальні роки й типову шкільну звітність.
UNILIB / Unie Library	Україна	публічні, університетські, спеціалізовані, невеликі бібліотеки	Не є спеціалізованою саме для шкільних бібліотек. Потребує адаптації до фонду підручників, класів, навчальних років, забезпеченості учнів і внутрішньої звітності закладу освіти.
Koha	Нова Зеландія	публічні, університетські, наукові, муніципальні	Потребує встановлення, налаштування, локалізації та технічного супроводу. Не має готової моделі української шкільної бібліотеки з обліком підручників, класів і навчальних років.
ІРБІС64 / ІРБІС64+	Росія	публічні, наукові, університетські, спеціалізовані	Має російське походження, що створює безпекові та організаційні ризики. Також система є складною для типової шкільної бібліотеки й недоцільною для нового впровадження в українських закладах освіти.
ALEPH	Ізраїль	академічні, наукові, університетські, дослідницькі	Має високу вартість і складність упровадження. Потребує серверної інфраструктури, адміністрування та фахового супроводу, тому є надмірною для окремої шкільної бібліотеки.
Alma	Ізраїль	академічні, університетські, наукові, бібліотечні мережі	Орієнтована на великі бібліотечні екосистеми, електронні ресурси й аналітику. Для школи є занадто дорогою, складною та не адаптованою до обліку підручників і локальної документації.

Evergreen ILS	США	публічні, муніципальні, бібліотечні мережі, консорціуми	Розрахована переважно на мережеву роботу кількох бібліотек. Потребує адміністрування, локалізації та технічної підтримки, що є складним для окремої шкільної бібліотеки.
OPALS	США	шкільні, малі публічні, спеціальні	Має шкільну орієнтацію, але не адаптована до української нормативної бази, УДК, актів списання, фонду підручників, класів, навчальних років і локальної звітності.
OpenBiblio	США	малі, шкільні, громадські, приватні	Має обмежену функціональність і застарілий підхід. Не враховує специфіку української шкільної бібліотеки, зокрема облік підручників, інвентарні номери та шкільні звіти.
PMB / PhpMyBibli	Франція	публічні, навчальні, спеціалізовані, малі та середні	Потребує локалізації й адаптації до українських освітніх процесів. Не має готової підтримки фонду підручників, класів, навчальних років, інвентарних книг і актів списання.
Librarika	США	малі, шкільні, навчальні, громадські, приватні	Залежить від зовнішнього сервісу, тарифів і умов зберігання даних. Не адаптована до української шкільної документації, фонду підручників і вимог до локальної звітності.
FOLIO	США	університетські, наукові, академічні, бібліотечні екосистеми	Має складну модульну архітектуру й потребує фахового адміністрування. Для окремої школи є надмірною за масштабом, технічними вимогами та складністю впровадження.

ДОДАТОК Б

Наведено основні екрани АБІС «BookCatalog». Скріншоти демонструють роботу гостьової частини системи, процес авторизації користувача, функціональні можливості користувацької панелі, модулі обліку книжкового фонду, читачів, підручників, обміну, оплати, а також адміністративну частину системи.

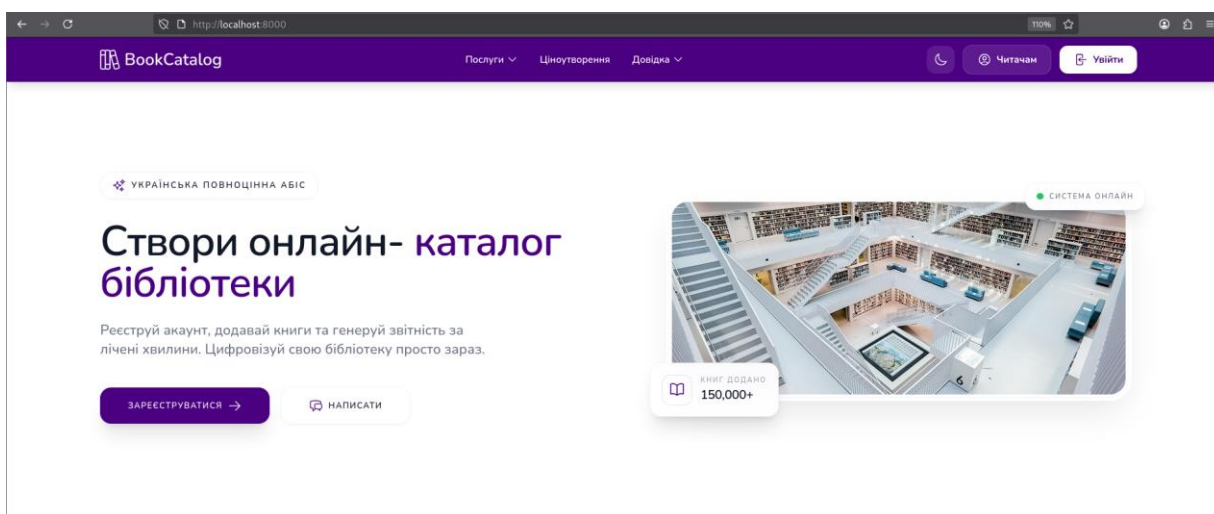


Рисунок Б.1 – Головна сторінка програмного забезпечення «BookCatalog»

На головній сторінці користувач отримує загальне уявлення про призначення АБІС, можливості та основні переваги.

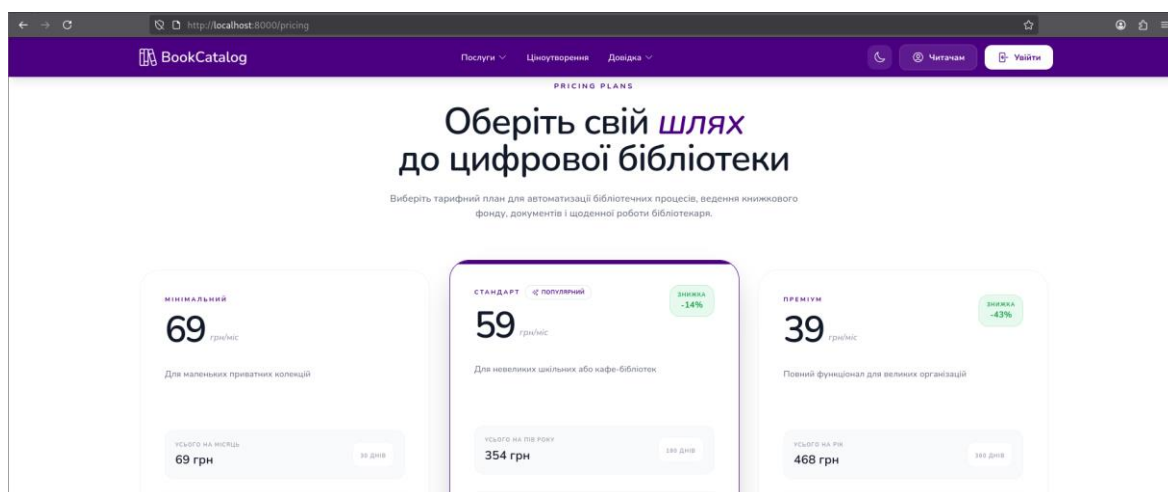


Рисунок Б.2 – Сторінка перегляду тарифних планів

Сторінка тарифних планів відображає доступні варіанти користування системою, їхню тривалість та вартість.

BookCatalog

Послуги Ціноутворення Довідка

Читачам Увійти

новий профіль

Створити акаунт

Приєднуйтеся до платформи цифровізації бібліотечних фондів, реєстрів і системної документації.

ВЖЕ Є ПРОФІЛЬ? Увійти в систему

ДАНІ РЕЄСТРАЦІЇ

ЗАПОВНІТЬ ПРОФІЛЬ

ІМ'Я КОРИСТУВАЧА

Наприклад: Ченна Оксана

ЕЛЕКТРОННА ПОШТА

name@example.com

ПАРОЛЬ

8-20 символів, Аа та цифра

☐ Я погоджуюся з Умовами використання та Політикою конфіденційності

ЗАРЕЄСТРУВАТИСЯ →

SECURE ENROLLMENT - V2.4.0

Рисунок Б.3 – Сторінка реєстрації користувача

Сторінка реєстрації призначена для створення нового облікового запису користувача.

BookCatalog

Послуги Ціноутворення Довідка

Читачам Увійти

з поверненням

Авторизація в системі

Введіть свої дані, щоб продовжити роботу з бібліотечними фондами, реєстрами та системними документами.

ЩЕ НЕМАЄ АКАУНТУ? Створити профіль

ВХІД

ВКАЖІТЬ ОБЛІКОВІ ДАНІ

ЕЛЕКТРОННА ПОШТА

name@example.com

ПАРОЛЬ

Введіть пароль

ЗАБУЛИ ПАРОЛЬ?

УВІЙТИ →

SECURE AUTHENTICATION - V2.4.0

Рисунок Б.4 – Сторінка авторизації користувача

Сторінка авторизації забезпечує вхід зареєстрованого користувача до системи.

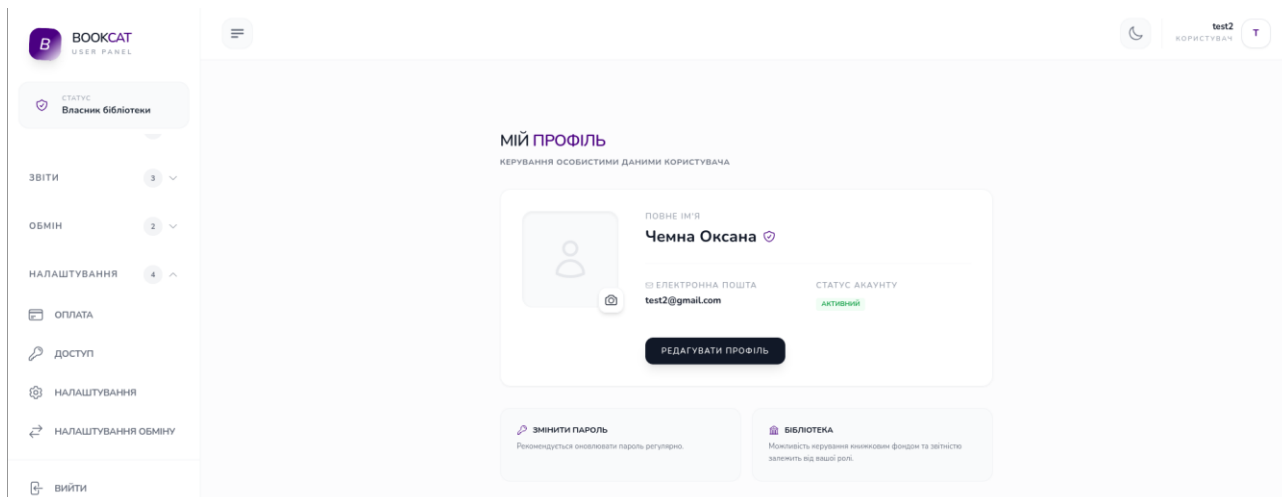


Рисунок Б.5 – Особистий кабінет користувача

Особистий кабінет користувача містить основні відомості про обліковий запис.

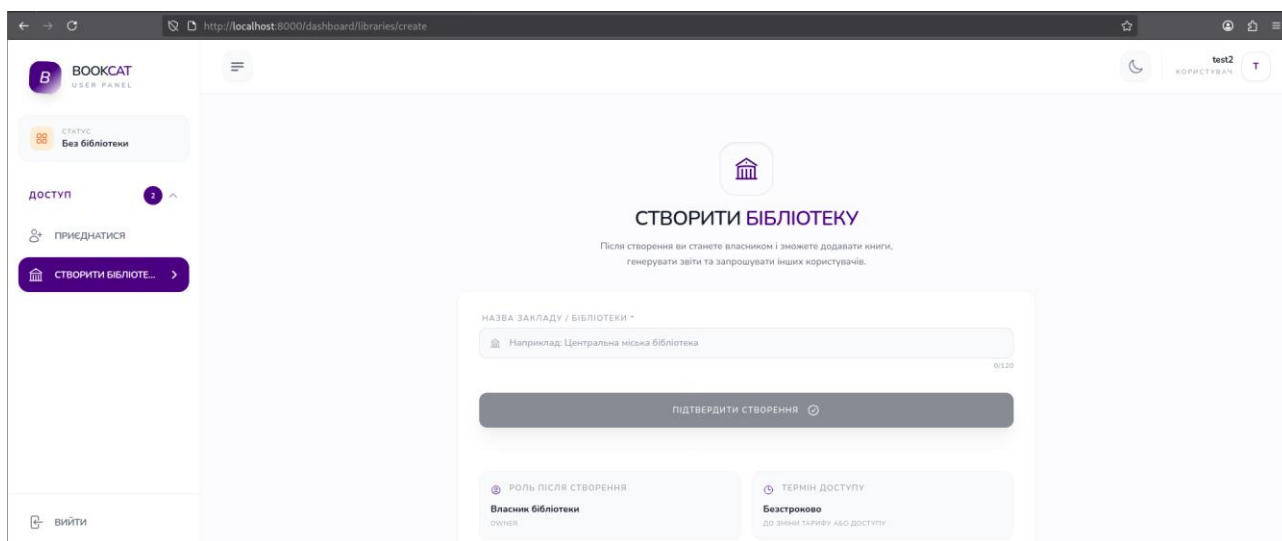


Рисунок Б.6 – Сторінка створення бібліотеки

Сторінка створення бібліотеки дозволяє користувачу зареєструвати нову бібліотеку в системі, вказати її назву та розпочати роботу з книжковим фондом.

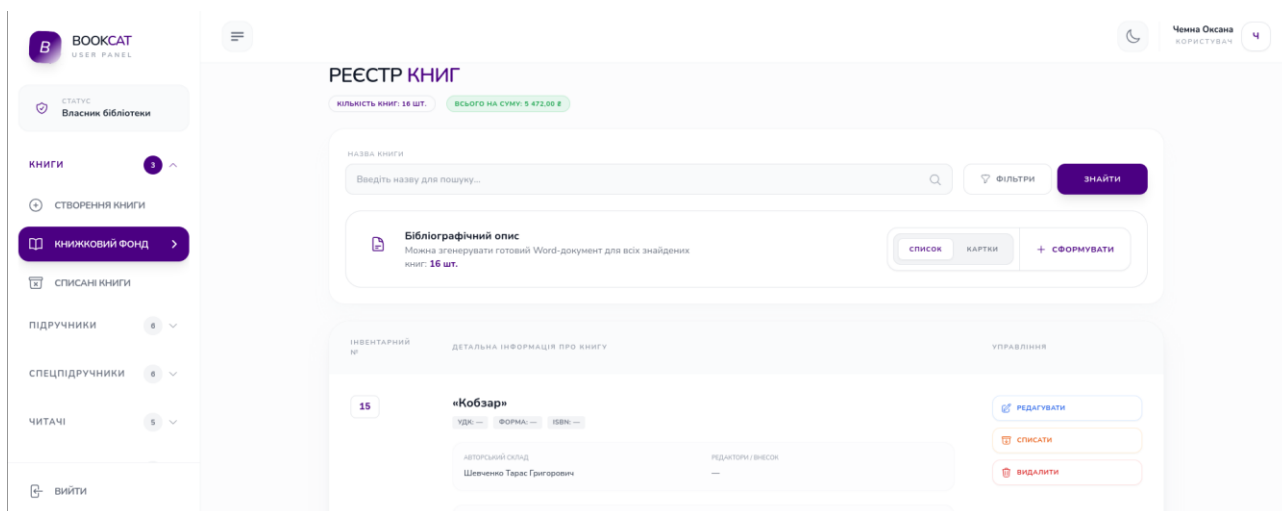


Рисунок Б.7 – Каталог книг бібліотеки

Каталог книг є одним з основних модулів системи. Він дає змогу переглядати книжковий фонд бібліотеки, здійснювати пошук, фільтрацію та перехід до роботи з окремими записами.

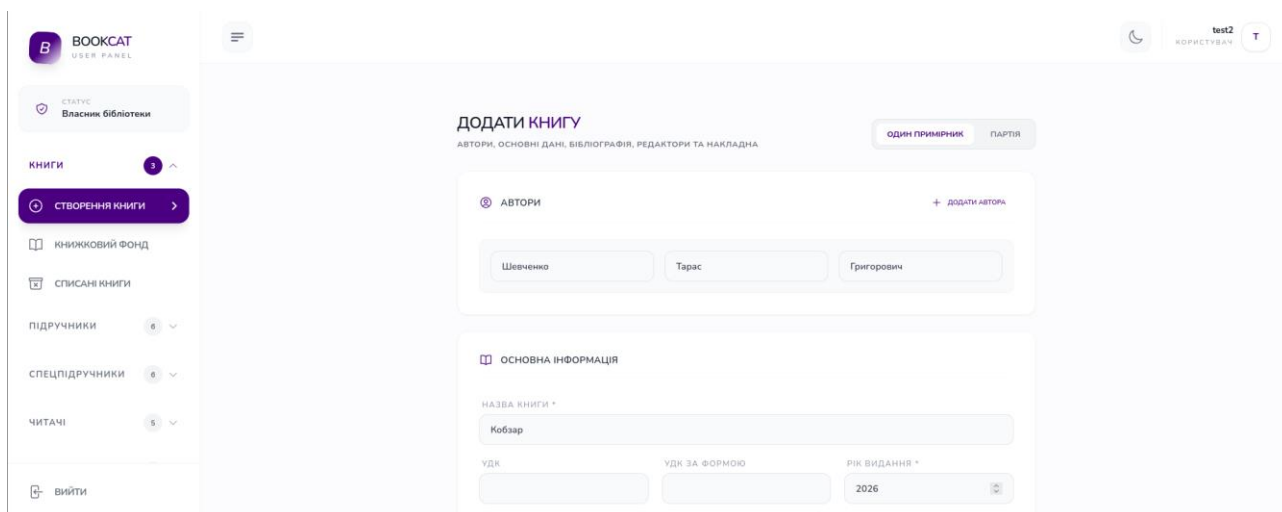


Рисунок Б.8 – Сторінка додавання книги до каталогу

Сторінка додавання книги використовується для внесення нової бібліографічної одиниці до фонду бібліотеки. Користувач заповнює дані про назву, авторів, редакторів, рік видання та інші характеристики книги.

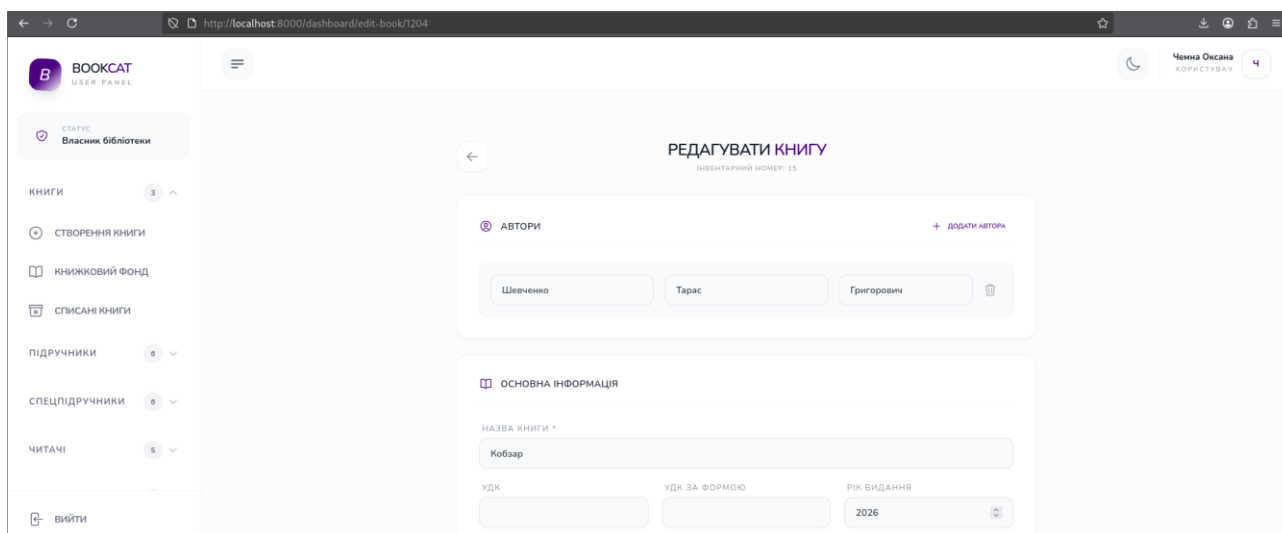


Рисунок Б.9 – Сторінка перегляду та редагування даних книги

Сторінка роботи з окремою книгою дозволяє переглядати детальну інформацію про запис та редагувати дані.

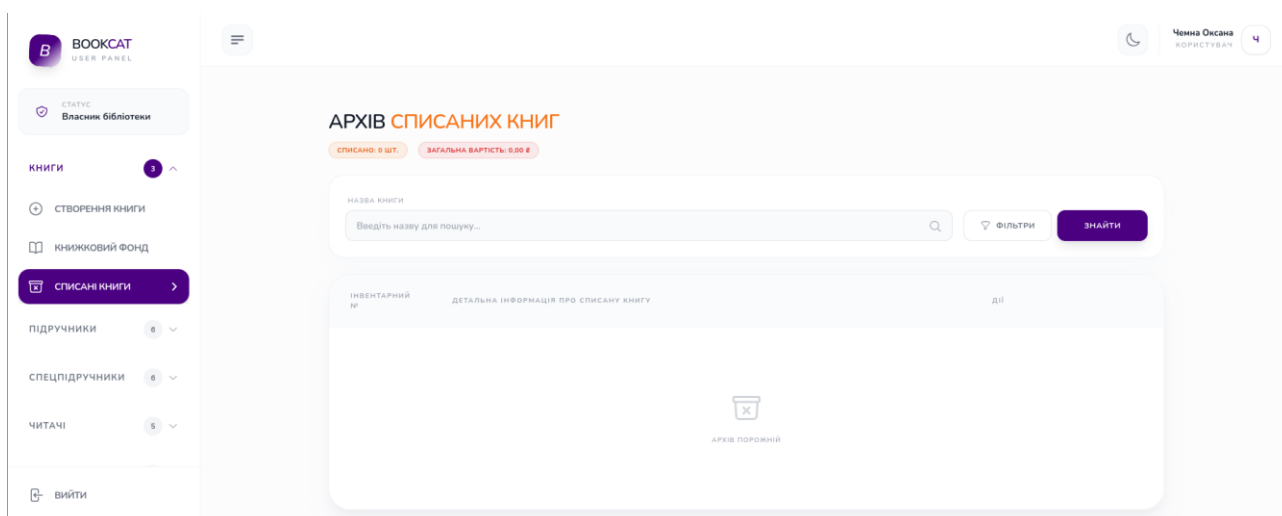


Рисунок Б.10 – Сторінка списаних книг

Сторінка списаних книг відображає записи, які були вилучені з активного фонду.

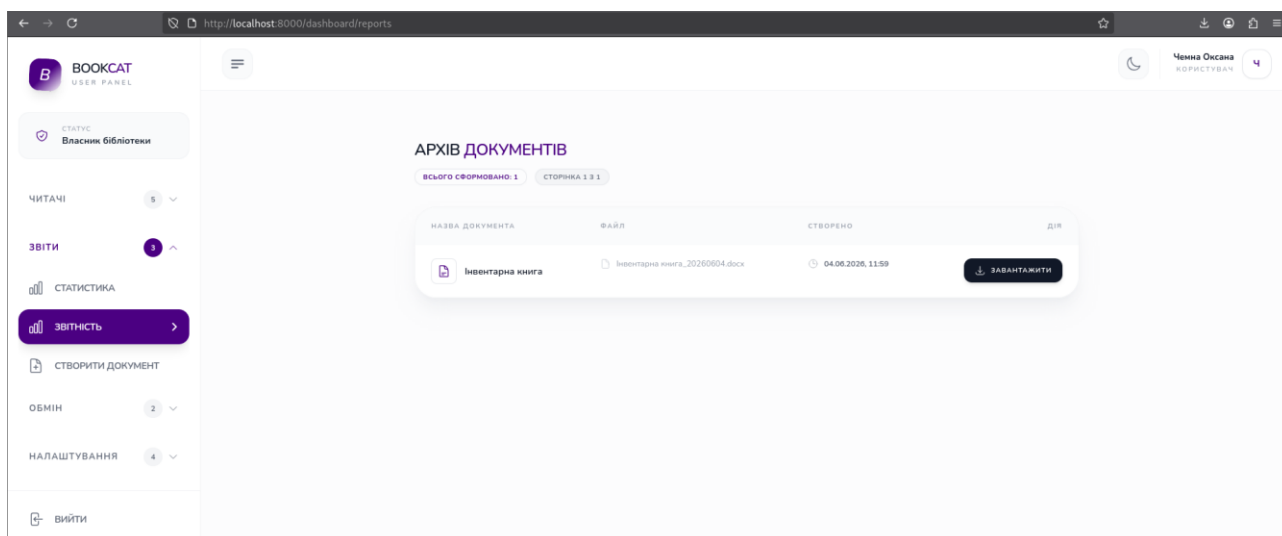


Рисунок Б.11 – Сторінка документів

Модуль документів і звітів призначений для формування службових документів бібліотеки, зокрема звітів, актів, бібліографічних списків та інших матеріалів, пов'язаних з обліком фонду.

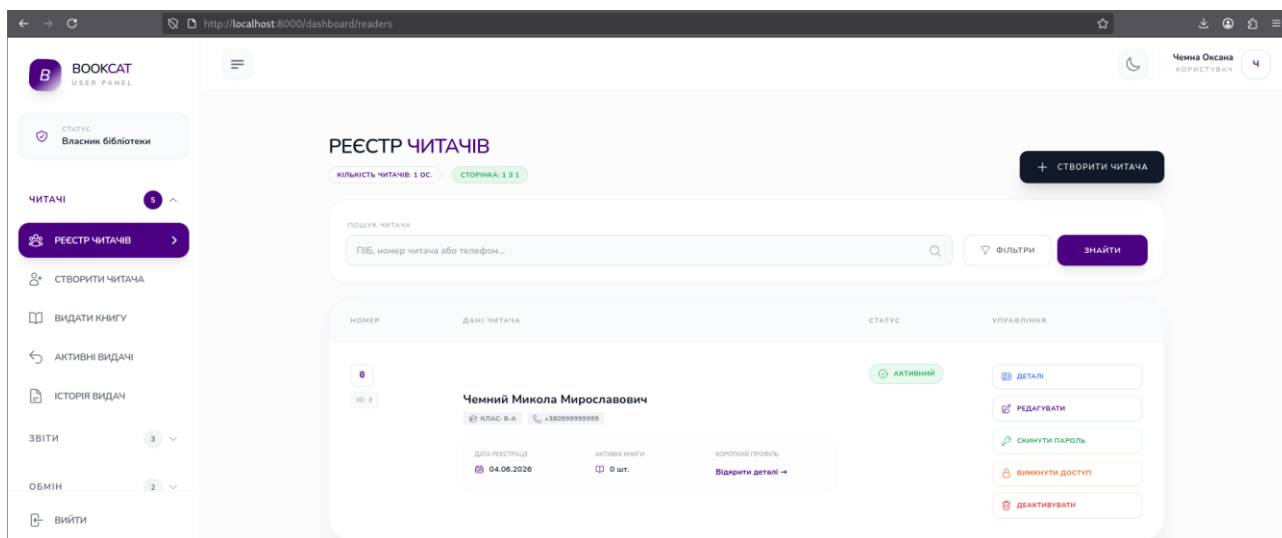


Рисунок Б.12 – Сторінка обліку читачів

Сторінка обліку читачів використовується для перегляду списку зареєстрованих читачів бібліотеки.

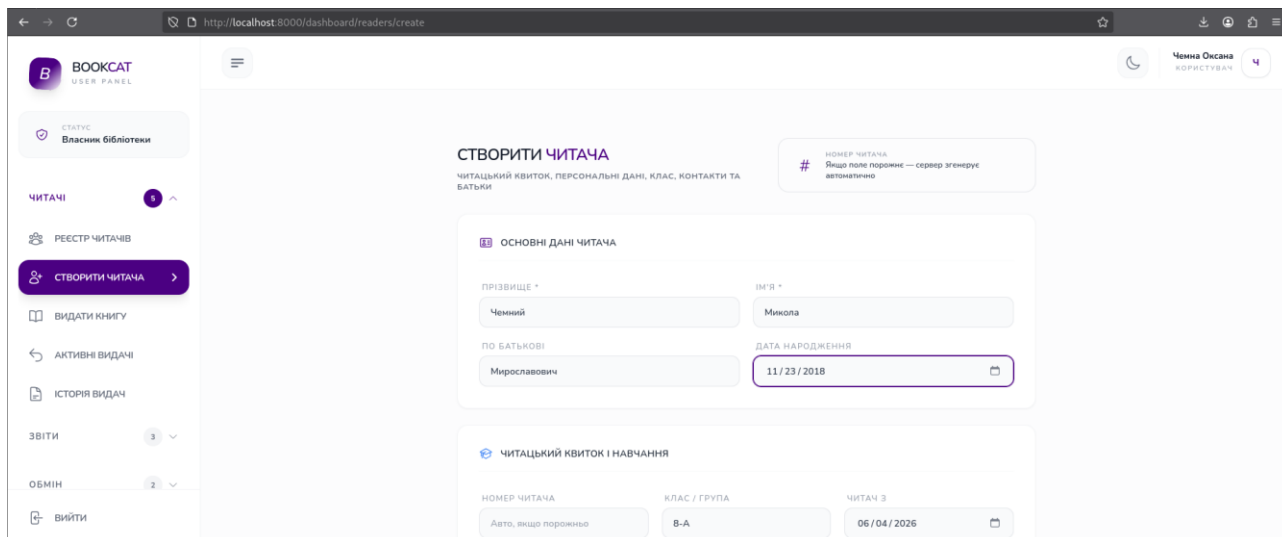


Рисунок Б.13 – Сторінка створення читацького запису

Сторінка створення читача дозволяє додати нового читача до системи, зберегти його персональні та контактні дані, а також надалі використовувати цей запис для видачі книг.

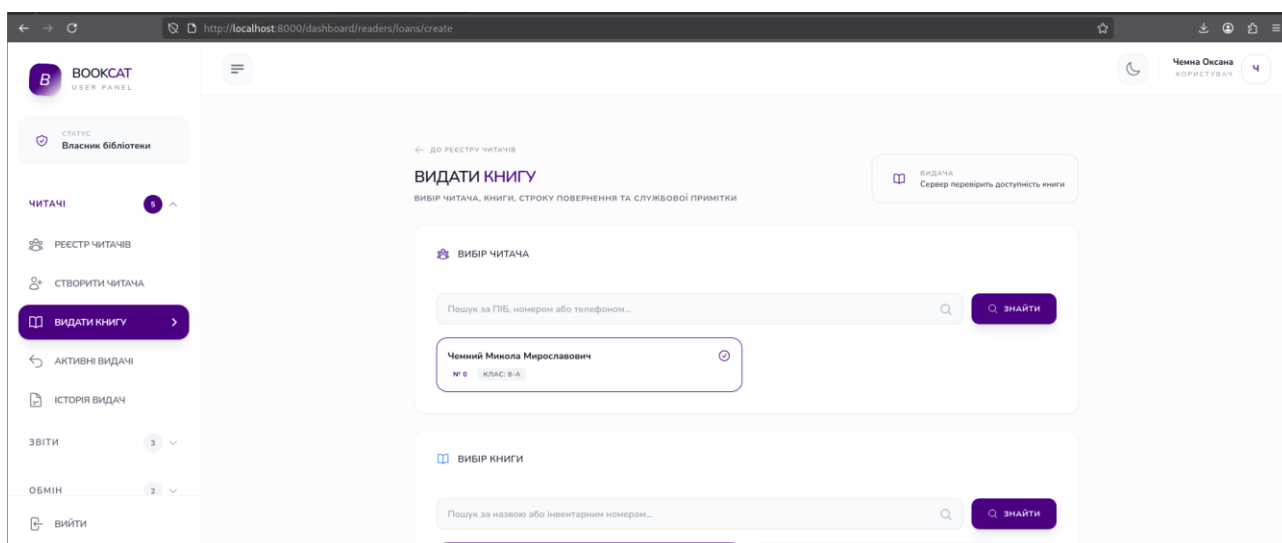


Рисунок Б.14 – Сторінка видачі книги читачу

Модуль видачі книг забезпечує фіксацію факту передання книги читачу.

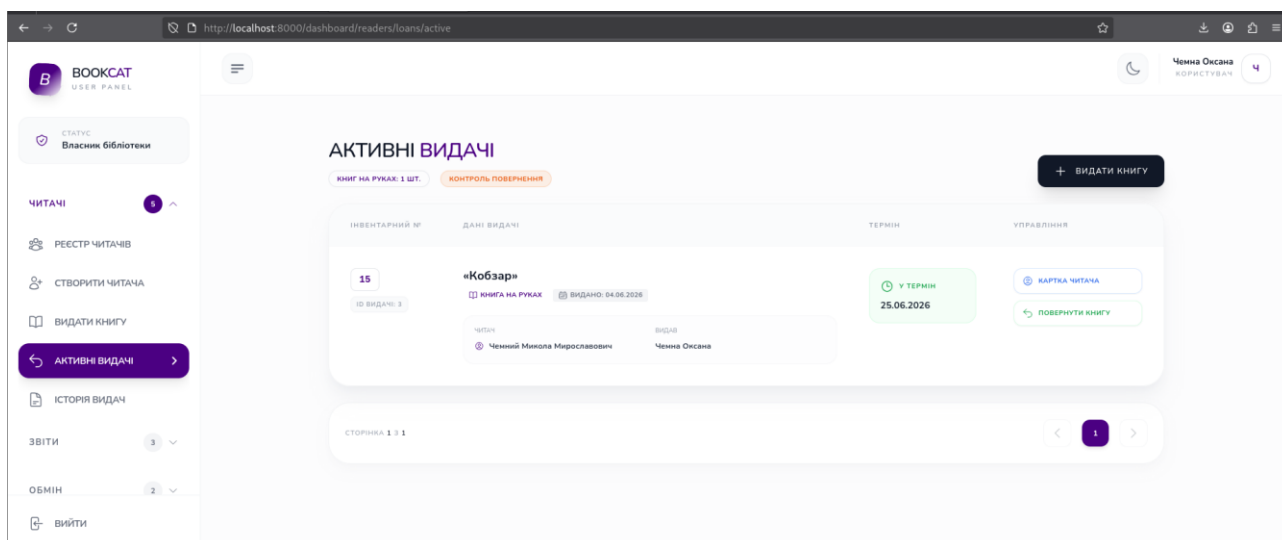


Рисунок Б.15 – Сторінка активних видач

Сторінка активних видач дозволяє переглядати поточні видані книги.

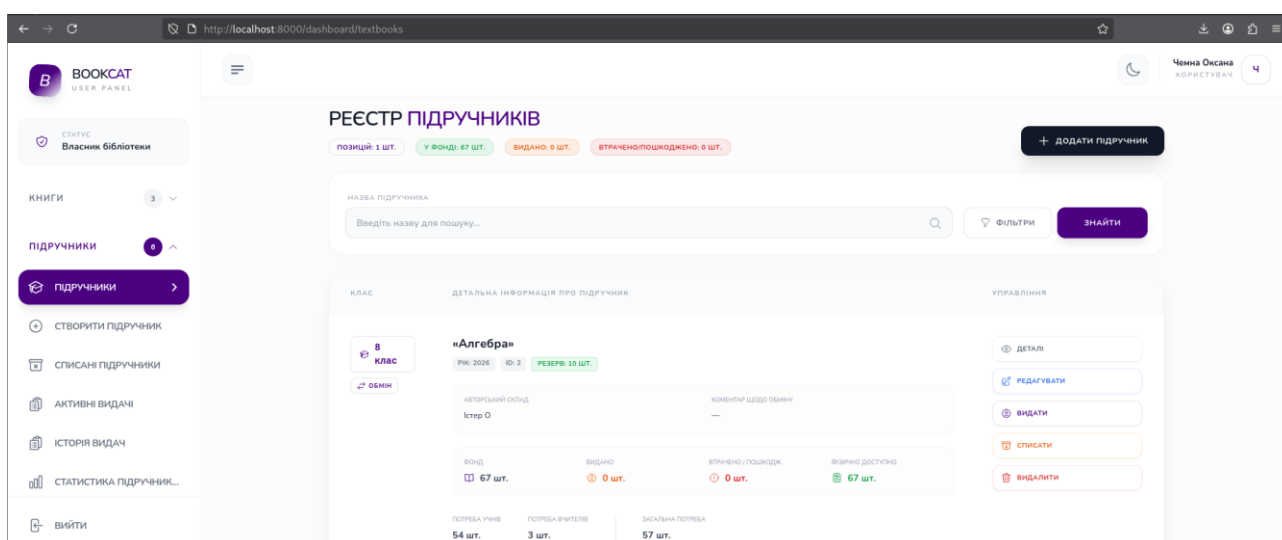


Рисунок Б.16 – Каталог підручників

Каталог підручників використовується для обліку навчальної літератури. У ньому зберігаються дані про підручники, класи, авторів, кількість примірників, потребу та стан фонду.

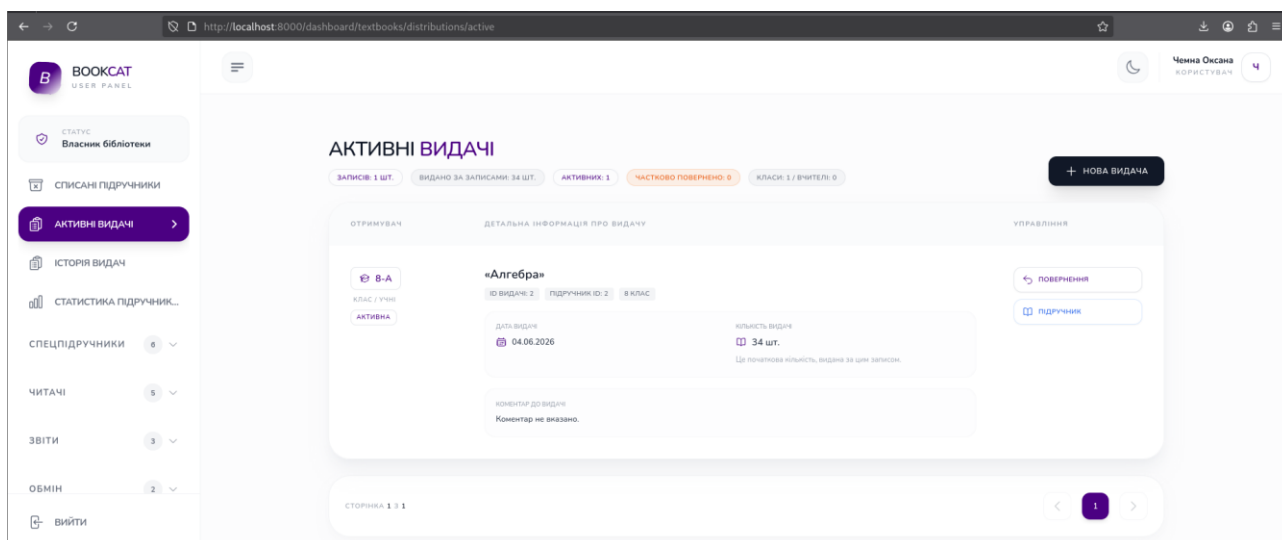


Рисунок Б.17 – Сторінка видачі підручників

Сторінка видачі підручників забезпечує облік руху навчальної літератури.

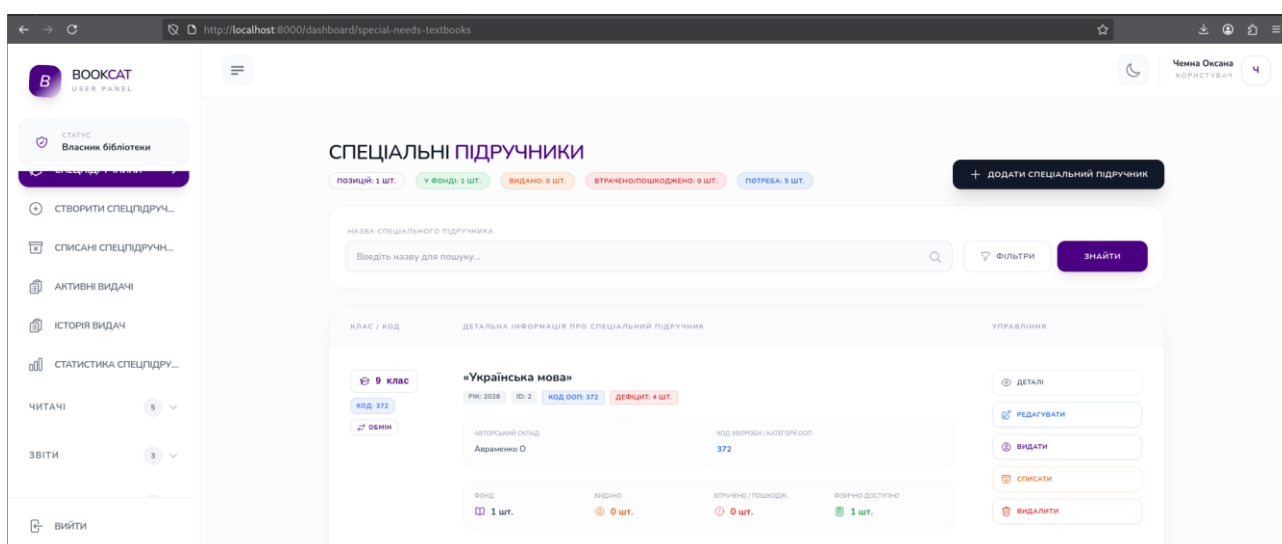


Рисунок Б.18 – Модуль підручників для дітей з особливими освітніми потребами

Окремий модуль підручників для дітей з особливими освітніми потребами дає змогу вести спеціалізований облік навчальної літератури з урахуванням додаткових параметрів, необхідних для такої категорії підручників.

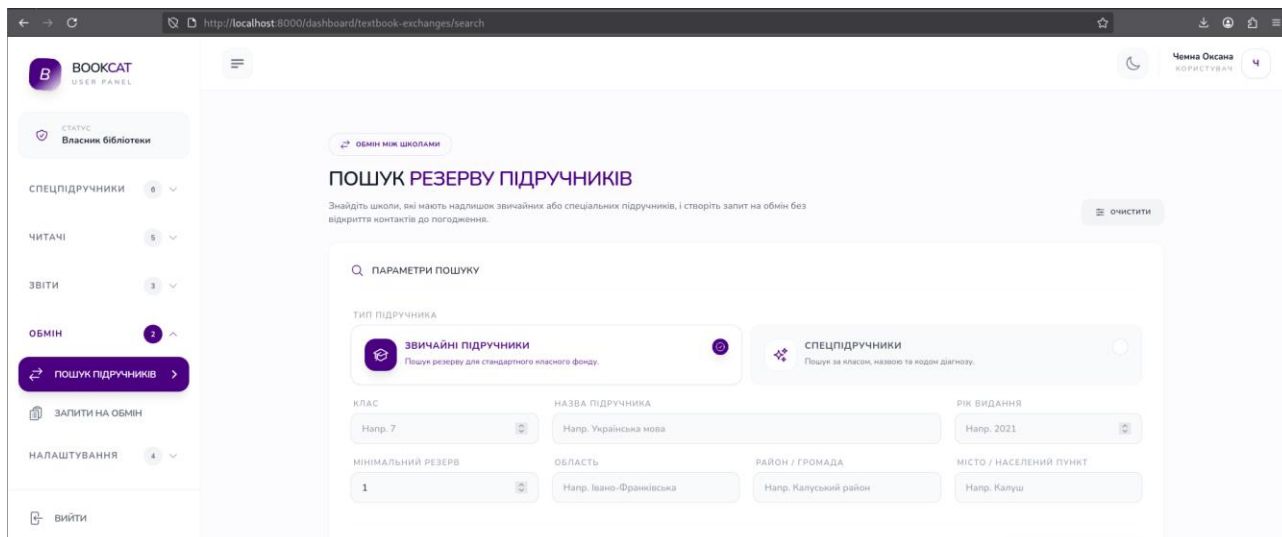


Рисунок Б.19 – Сторінка міжбібліотечного обміну підручниками

Модуль обміну підручниками дозволяє шукати доступні підручники в інших бібліотеках, створювати заявки на обмін та переглядати їхній поточний статус.

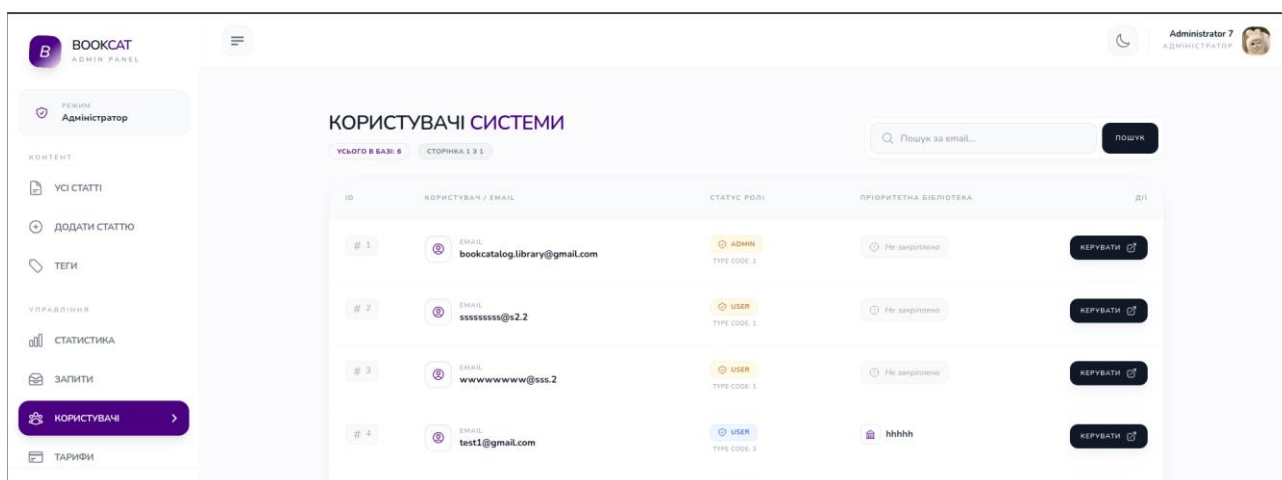


Рисунок Б.20 – Адміністративна панель системи

Адміністративна панель призначена для керування системними даними, користувачами, тарифними планами, статтями, тегами, заявками та загальними параметрами роботи програмного забезпечення.

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи: “Розробка АБІС для закладів загальної середньої освіти засобами React та ASP.NET”

Обсяг пояснювальної записки: 81 аркуш

Дата закінчення роботи 4 червня 2026 р.

Підпис _____