

**Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення**

**КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»**

**РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ БІБЛІОТЕКОЮ
ВИКОРИСТОВУЮЧИ C# ТА MS SQL SERVER**

**DEVELOPING A LIBRARY MANAGEMENT SYSTEM USING C# AND MS
SQL SERVER**

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗ-41
Сорочук Яна Миколаївна

Керівник:
Паливода Данило Ігорович

Кваліфікаційну роботу
допущено до захисту
«__» _____ 2026 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

Луцьк – 2026 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних та інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Галузь знань: 12 «Інформаційні технології»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма: «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
ЗАВІДУВАЧ КАФЕДРИ

«__» _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Сорочук Яні Миколаївні

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «Розробка системи управління бібліотекою використовуючи C# та MS SQL Server»

Керівник роботи: асистент Паливода Д. І.

затверджені наказом закладу вищої освіти від «31» грудня 2025 р. № 541/01-02

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра _____
«29» травня 2026 р.

3. Вихідні дані до роботи: C#, .NET, Windows Forms, Entity Framework Core, Microsoft SQL Server, Visual Studio, QuestPDF, ClosedXML, вимоги до системи управління бібліотекою, методичні вказівки до виконання кваліфікаційної роботи бакалавра.

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): аналіз сучасного стану проблем автоматизації обліку персоналу та управління задачами, постановка завдання, формування вимог до розроблюваної інформаційної системи, обґрунтування вибору технологій та засобів розробки, практична реалізація об'єкта проєктування, розробка структури бази даних, обґрунтування принципів захисту інформаційно-комп'ютерної системи, тестування та налагодження програмного забезпечення.

5. Перелік графічного матеріалу: 20 рисунків, 8 лістингів, 4 таблиць.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	Паливода Д. І.		
Специфікація вимог до розробленої системи	Паливода Д. І.		
Розробка об'єкта проектування	Паливода Д. І.		
Нормоконтроль	Суринович О. М.		
Гарант ОП	Ліщина Н. М.		
Показник запозичень тексту	%		
Академічна доброчесність	Паливода Д. І.		

7. Дата видачі завдання «3» лютого 2026 р.

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 14.02.2026 р.	
2	Аналіз проблеми розробки та впровадження об'єкту проектування	до 03.03.2026 р.	
3	Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 14.03.2026 р.	
4	Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних	до 14.04.2026 р.	
5	Практична реалізація об'єкта проектування та розробка бази даних	до 20.04.2026 р.	
6	Тестування та налагодження об'єкта проектування	до 04.05.2026 р.	
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедрі	до 29.05.2026 р.	

Здобувач вищої освіти

Яна СОРОЧУК

Керівник кваліфікаційної роботи

Данило ПАЛИВОДА

АНОТАЦІЯ

Сорочук Я. М. Розробка системи управління бібліотекою використовуючи C# та MS SQL Server. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності 121 «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2026.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел і додатків.

У роботі розроблено систему управління бібліотекою з використанням C# та MS SQL Server. У першому розділі досліджено предметну область бібліотечного обліку, проаналізовано сучасні підходи до автоматизації роботи бібліотек і наявні програмні аналоги. У другому розділі сформовано вимоги до програмного продукту, описано користувацькі ролі, функціональні можливості, структуру бази даних і вибір інструментів реалізації. У третьому розділі наведено процес практичної розробки системи, реалізацію модулів роботи з книгами, читачами, видачами, поверненнями й резервуваннями, а також питання захисту інформації та результати тестування. Розроблена система спрощує облік бібліотечного фонду та підвищує ефективність обслуговування читачів.

Ключові слова: бібліотека, C#, MS SQL Server, база даних, облік книг, читачі, автоматизація.

ABSTRACT

Sorochuk Y. M. Development of a Library Management System Using C# and MS SQL Server. Manuscript. Qualification work for bachelor's degree in Software Engineering, specialty 121 Software Engineering. Lutsk National Technical University. Lutsk, 2026.

The bachelor's qualification work consists of an introduction, three chapters, conclusions, a list of references, and appendices.

The work presents the development of a library management system using C# and MS SQL Server. The first chapter examines the subject area of library accounting, analyzes modern approaches to library workflow automation, and reviews existing software analogues. The second chapter defines the software requirements and describes user roles, functionality, database structure, and implementation tools. The third chapter presents the practical development of the system, including modules for books, readers, loans, returns, and reservations, as well as information security and testing results. The developed system simplifies library collection accounting and improves reader service efficiency.

Keywords: library, C#, MS SQL Server, database, book accounting, readers, automation.

ЗМІСТ

ВСТУП	7
РОЗДІЛ 1 АНАЛІЗ МЕТОДІВ РОЗРОБКИ ТА ПОСТАНОВКА ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ	9
1.1 Аналіз сучасного стану проблеми	9
1.2 Постановка завдання на кваліфікаційну роботу бакалавра	12
РОЗДІЛ 2 СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ УПРАВЛІННЯ БІБЛІОТЕКОЮ	13
2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проєктування програмного забезпечення	13
2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання ...	19
РОЗДІЛ 3 РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ БІБЛІОТЕКОЮ	25
3.1 Практична реалізація об'єкта проєктування	25
3.2 Розробка бази даних	31
3.3 Захист інформаційно-комп'ютерної системи	34
3.4 Тестування та налагодження інформаційно-комп'ютерної системи	36
ВИСНОВКИ	41
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	43
ДОДАТКИ	44

ВСТУП

Актуальність теми. Бібліотека працює з великою кількістю взаємопов'язаних даних: книжковим фондом, читачами, видачами, поверненнями, простроченнями, резервуваннями та обліковими записами працівників. Якщо ці процеси виконуються вручну або в окремих файлах, ускладнюється контроль доступних примірників, пошук історії читача, своєчасне виявлення прострочених повернень і підготовка звітів. Тому актуальним є створення системи управління бібліотекою, яка об'єднує основні операції обліку в одному настільному застосунку та підтримує різні ролі користувачів.

Для розроблення настільного інтерфейсу системи використано Windows Forms, що надає засоби створення форм, елементів керування та подієвої взаємодії користувача з програмою [1]. Зберігання бібліотечних даних реалізовано з використанням Entity Framework Core та провайдера SQL Server, що дає змогу працювати з реляційними таблицями через моделі C# і підтримувати зв'язки між книгами, читачами, видачами, резервуваннями та користувачами [2]. Для формування PDF-звіту про прострочені книги застосовано QuestPDF, який забезпечує програмне створення документів із таблицями та форматованим вмістом [3].

Мета кваліфікаційної роботи полягає у розробленні системи управління бібліотекою з використанням C# та MS SQL Server, яка забезпечує облік книг і читачів, керування видачами та поверненнями, резервування примірників, контроль прострочень, рольовий доступ і формування звітів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- здійснити аналіз процесів обліку бібліотечного фонду, читачів, видач, повернень і резервувань у межах бібліотечної системи;
- сформулювати функціональні вимоги до настільного застосунку з урахуванням ролей адміністратора, бібліотекаря та користувача з правами перегляду;

- спроектувати структуру бази даних для зберігання книг, читачів, видач, резервувань і облікових записів користувачів;
- обґрунтувати вибір C#, MS SQL Server та інших технологій для реалізації системи;
- реалізувати модулі керування книжковим фондом, читачами, видачами, поверненнями, резервуваннями, користувачами та звітами;
- перевірити працездатність системи на типових сценаріях роботи бібліотеки.

Об'єктом дослідження є процеси інформаційного супроводу роботи бібліотеки, пов'язані з обліком фонду, читачів, видач, повернень і резервувань.

Предметом дослідження є методи та програмні засоби розроблення настільної бібліотечної системи з рольовим доступом, реляційною базою даних і формуванням звітності.

Практична цінність розробки полягає у створенні програмного засобу, який може використовуватися для централізованого ведення бібліотечних даних, швидкого пошуку книг і читачів, контролю доступних примірників, відстеження прострочених повернень, розрахунку штрафів, роботи з резервуваннями та підготовки звітів у зручних форматах.

РОЗДІЛ 1

АНАЛІЗ МЕТОДІВ РОЗРОБКИ ТА ПОСТАНОВКА ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

Бібліотечні інформаційні системи автоматизують ведення книжкового фонду, читацьких карток, видачі та повернення примірників, бронювання і формування звітності. Для розроблюваної системи важливими є облік книг і читачів, контроль строків користування, розрахунок штрафів, рольовий доступ, формування PDF звітів про заборгованість та експорт даних у табличний формат.

Одним із найвідоміших аналогів є Koha – інтегрована бібліотечна система з відкритим кодом, що охоплює каталогізацію, обіг примірників, роботу з читачами, комплектування та онлайнкаталог [4]. На рисунку 1.1 наведено приклад інтерфейсу Koha.

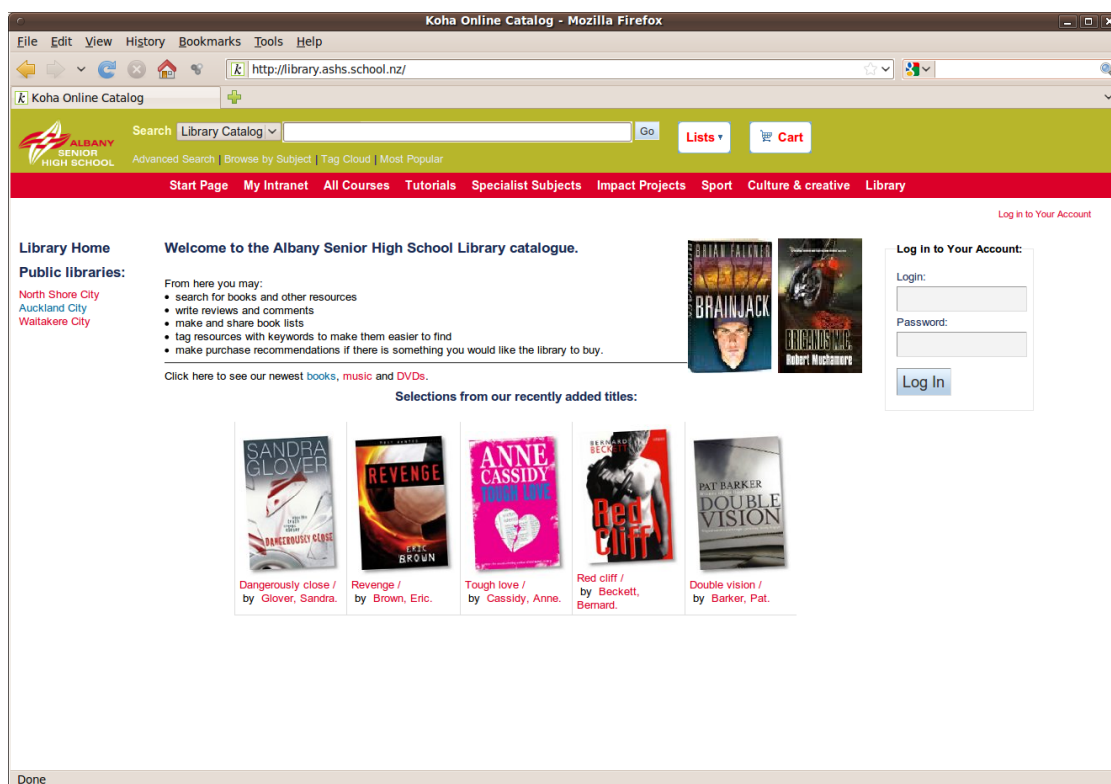


Рисунок 1.1 – Приклад інтерфейсу бібліотечної системи Koha [5]

Перевагою Koha є повнота бібліотечних процесів і придатність для реального впровадження у закладах різного масштабу. Недоліком для задачі кваліфікаційної роботи є складність розгортання та адміністрування: система орієнтована на вебсервер, каталогізаційні стандарти і значно ширший набір процесів, ніж потрібен для локального настільного застосунку.

Іншим рішенням є Evergreen ILS – відкрита бібліотечна система для каталогізації, циркуляції фонду, роботи з читачами та керування бібліотечними мережами [6]. На рисунку 1.2 показано приклад робочого екрана клієнтської частини Evergreen.

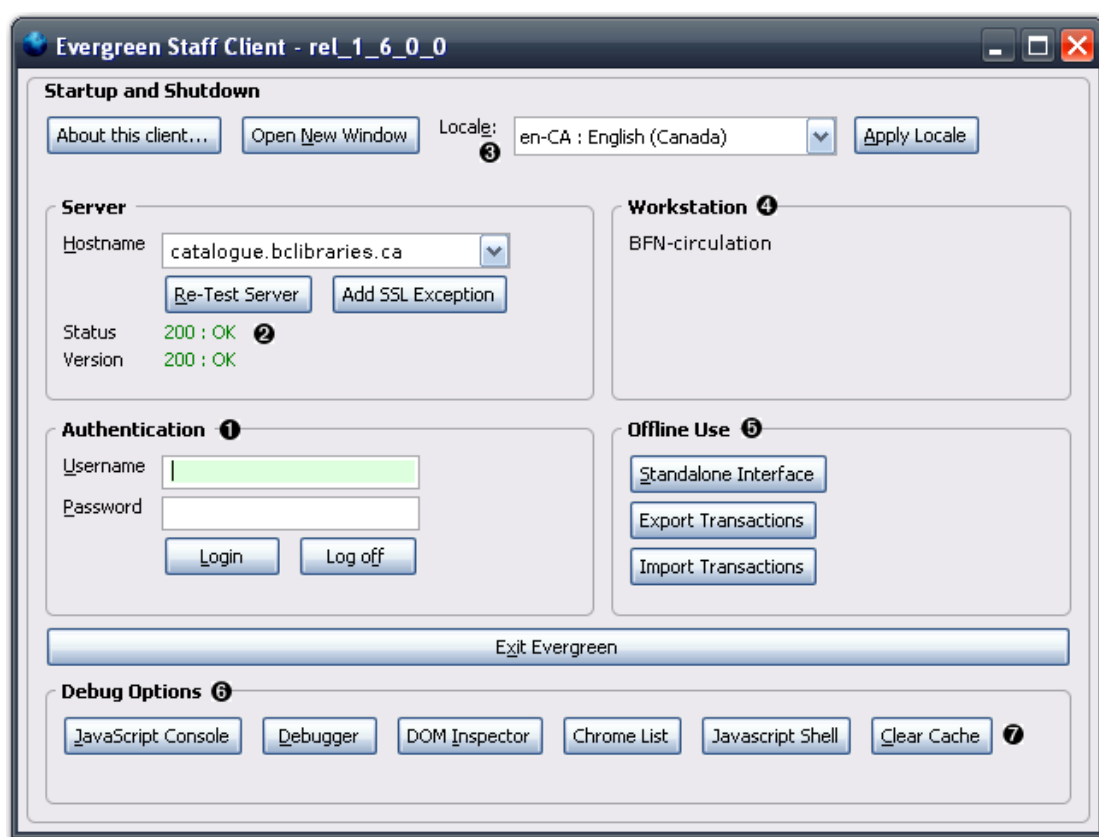


Рисунок 1.2 – Робочий екран клієнтської частини Evergreen [7]

Evergreen добре підходить для великих бібліотек і консорціумів, де потрібні розподілені каталоги, обмін записами та складна логіка обслуговування. Для невеликої локальної системи такий підхід є надмірним, оскільки потребує складнішої інфраструктури і навчання персоналу.

Система SLiMS також належить до відкритих бібліотечних систем і підтримує каталог, обслуговування читачів, членство, звіти та вебінтерфейс для доступу до бібліотечних записів [8]. Її перевагою є доступність і орієнтація на типові бібліотечні процеси, однак вебмодель роботи та ширший функціональний набір не завжди потрібні для невеликого настільного рішення з конкретними операціями видачі, повернення і резервування.

У таблиці 1.1 наведено порівняння розглянутих аналогів з розроблюваною системою.

Таблиця 1.1 – Порівняння аналогів бібліотечної інформаційної системи

Система	Переваги	Недоліки щодо поставленої задачі
Koha	повноцінна інтегрована бібліотечна система, відкритий код, онлайнкаталог	складне веброзгортання і надмірність для локального настільного застосунку
Evergreen ILS	підтримка великих бібліотечних мереж, каталогізація, циркуляція, робота з читачами	орієнтація на масштабну інфраструктуру та складніше адміністрування
SLiMS	відкритість, типові бібліотечні модулі, вебдоступ до каталогу	ширша вебсистема, ніж потрібно для локального обліку видач і штрафів
Розроблювана система	простий настільний інтерфейс, ролі користувачів, видача, повернення, бронювання, штрафи, PDF та Excel звіти	не охоплює повний набір професійних бібліотечних стандартів і мережних сценаріїв

Отже, існуючі бібліотечні системи забезпечують дуже широкий функціонал, але часто потребують серверного середовища та професійного адміністрування. Розроблювана система доцільна як легший настільний інструмент для автоматизації основних операцій невеликої бібліотеки, де ключовими є швидкий облік книг, читачів, видач, повернень, резервувань і формування зрозумілих звітів.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

В результаті аналізу процесів роботи бібліотеки та функціональних можливостей розроблюваної системи було сформовано наступні завдання:

- здійснити аналіз процесів обліку бібліотечного фонду, читачів, видач, повернень і резервувань у межах бібліотечної системи, визначивши основні інформаційні об'єкти, типові операції бібліотекаря, проблеми контролю доступних примірників і потребу в історії взаємодії читача з бібліотекою;

- сформулювати функціональні вимоги до настільного застосунку з урахуванням ролей адміністратора, бібліотекаря та користувача з правами перегляду, передбачивши різний доступ до керування користувачами, видалення даних, редагування книг і читачів, створення видач та резервувань;

- спроектувати структуру бази даних для зберігання книг, читачів, видач, резервувань і облікових записів користувачів, врахувавши кількість примірників, доступні копії, дати видачі й повернення, строк повернення, статус резервування та хешовані паролі;

- обґрунтувати вибір C#, MS SQL Server та інших технологій для реалізації системи, беручи до уваги потребу в настільному інтерфейсі, реляційному зберіганні даних, роботі зі звітами у PDF та експорті прострочених видач в електронну таблицю;

- реалізувати модулі керування книжковим фондом, читачами, видачами, поверненнями, резервуваннями, користувачами та звітами, щоб система підтримувала повний цикл роботи від додавання книги до контролю повернення і повідомлення про активне резервування;

- перевірити працездатність системи на типових сценаріях роботи бібліотеки, зокрема авторизації користувача, пошуку книги або читача, створенні видачі, поверненні книги, зміні кількості доступних примірників, створенні резервування, визначенні прострочень і формуванні звітів.

РОЗДІЛ 2

СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ УПРАВЛІННЯ БІБЛІОТЕКОЮ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проєктування програмного забезпечення

Під час аналізу предметної області встановлено, що система управління бібліотекою повинна підтримувати облік книжкового фонду, читачів, видач, повернень, резервувань, користувачів і звітів. У межах такої системи важливо не лише зберігати перелік книг, а й бачити актуальний стан кожного бібліотечного ресурсу: скільки примірників є в наявності, які з них видані читачам, які записи прострочені та чи очікують інші читачі на повернення конкретної книги. Центральним процесом є контроль руху примірників: під час видачі кількість доступних книг зменшується, після повернення збільшується, а прострочені записи повинні відображатися у звітності. Для цього система має зберігати не лише довідкові дані про книгу, а й історію взаємодії книги з читачами, включаючи дату видачі, планову дату повернення, фактичне повернення та пов'язані резервування.

Користувачі системи поділяються на адміністратора, бібліотекаря та користувача з правами перегляду. Адміністратор керує обліковими записами та має доступ до службових операцій, які впливають на загальну конфігурацію системи. Бібліотекар виконує основні дії з книжковим фондом, читачами, видачами, поверненнями й резервуваннями, тобто відповідає за щоденний цикл роботи бібліотеки. Користувач з правами перегляду може аналізувати дані без можливості змінювати критичну інформацію, що корисно для контролю, звітності або ознайомлення зі станом фонду. Такий поділ зменшує ризик випадкового або несанкціонованого редагування бібліотечного фонду, а також дозволяє чітко відокремити перегляд інформації від операцій, які змінюють стан книг і записів видачі. Основні варіанти використання системи наведено на рисунку 2.1.

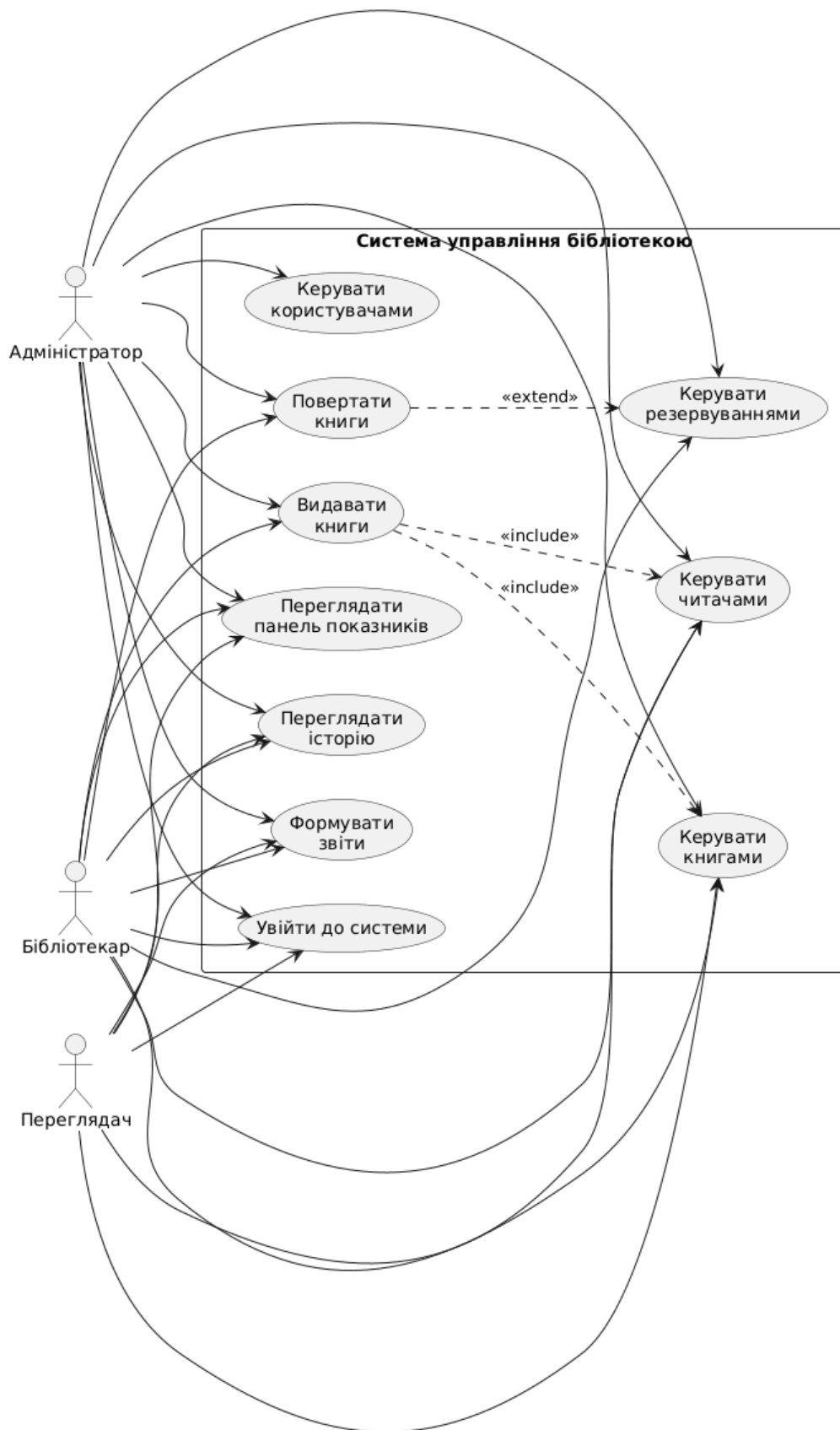


Рисунок 2.1 – UML-діаграма варіантів використання системи управління бібліотекою

Функціональні вимоги до системи охоплюють автентифікацію користувачів, рольовий доступ, облік книг, облік читачів, видачу й повернення книг, резервування, перегляд історії, пошук, фільтрацію та формування звітів. Система повинна відкривати основний інтерфейс лише після успішного входу користувача. Пароль має перевірятися не у відкритому вигляді, а через SHA256-хеш. Після входу роль користувача визначає доступні дії та форми.

Для роботи з книжковим фондом система повинна зберігати назву книги, автора, жанр, ISBN, рік видання, загальну кількість примірників і кількість доступних примірників. Для читачів необхідно зберігати ПІБ, email, телефон, адресу та дату реєстрації. Під час видачі книги система повинна створювати запис Loan, встановлювати дату видачі, планову дату повернення та зменшувати AvailableCopies. Під час повернення встановлюється ReturnDate, після чого кількість доступних примірників збільшується. Якщо для книги існує активне резервування, бібліотекар повинен отримати повідомлення з даними читача, який очікує на книгу.

Окрема група вимог пов'язана з резервуванням і звітністю. Резервування має зберігати книгу, читача, дату створення, статус і примітки. Система повинна запобігати повторному активному резервуванню однієї й тієї самої книги одним читачем. Статус резервування може бути активним, виконаним або скасованим. Для контролю прострочень система повинна визначати видачі, у яких дата повернення ще не встановлена, а планова дата повернення вже минула. Такі записи використовуються для формування PDF та Excel звітів.

Нефункціональні вимоги визначають якість роботи системи. Застосунок має бути реалізований як настільна Windows Forms система з окремими формами для книг, читачів, видач, резервувань, користувачів, історії та звітів [1]. Дані повинні зберігатися у реляційній базі через Entity Framework Core і MS SQL Server, що дає змогу описати зв'язки між сутностями та виконувати запити через об'єктну модель [2]. Формування PDF звітів доцільно реалізувати засобами QuestPDF [3], а експорт прострочених записів у табличний формат – засобами ClosedXML.

Проектування системи виконано як настільний C#/.NET застосунок із розділенням на форми, моделі, контекст бази даних, сервіси та допоміжні класи. AppDbContext містить набори Books, Readers, Loans, Reservations і Users. BookService, ReaderService, LoanService, ReservationService, UserService, AuthService, ReportService і PdfReportService виконують основну прикладну логіку. Форми Windows Forms відповідають за введення, перегляд і редагування даних. Узагальнену компонентну структуру системи наведено на рисунку 2.2.

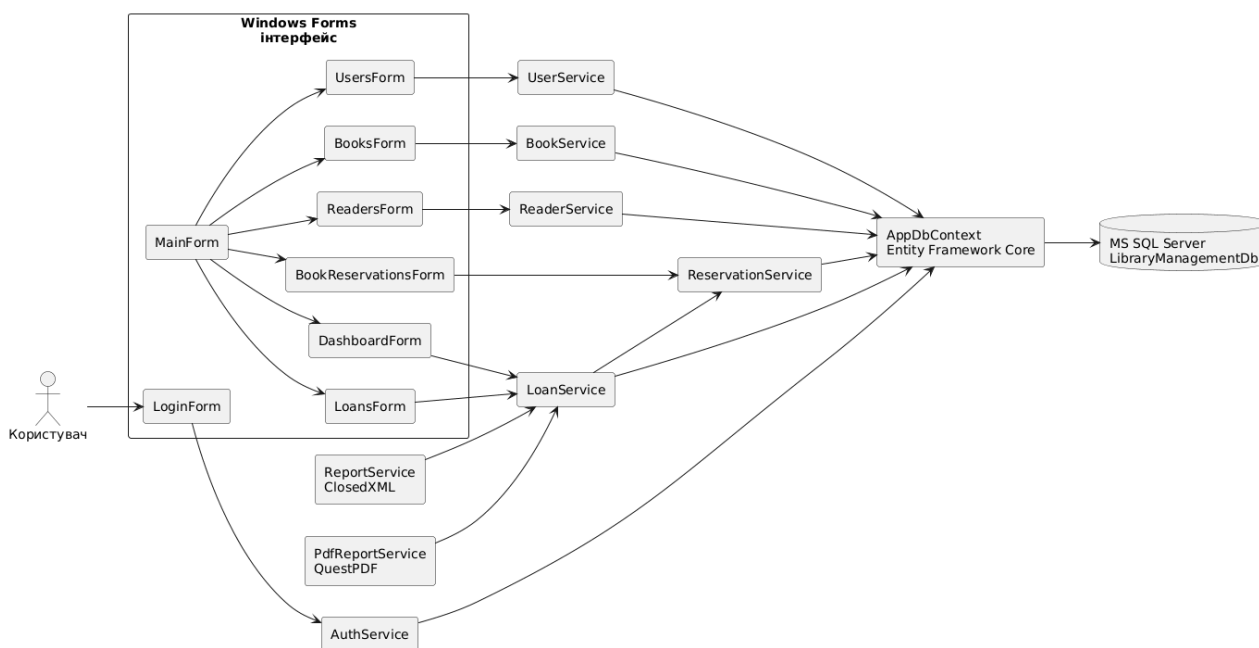


Рисунок 2.2 – UML-діаграма компонентів системи управління бібліотекою

Проектна модель даних побудована навколо сутностей Book, Reader, Loan, Reservation і User. Book описує книгу бібліотечного фонду та має колекції видач і резервувань. Reader описує читача та також пов'язаний з видачами й резервуваннями. Loan фіксує факт видачі книги читачу, дату видачі, планову дату повернення і фактичну дату повернення. Reservation описує бронювання книги читачем і містить статус. User зберігає обліковий запис працівника або користувача системи з роллю Admin, Librarian або Viewer. UML-діаграму класів проектної моделі наведено на рисунку 2.3.

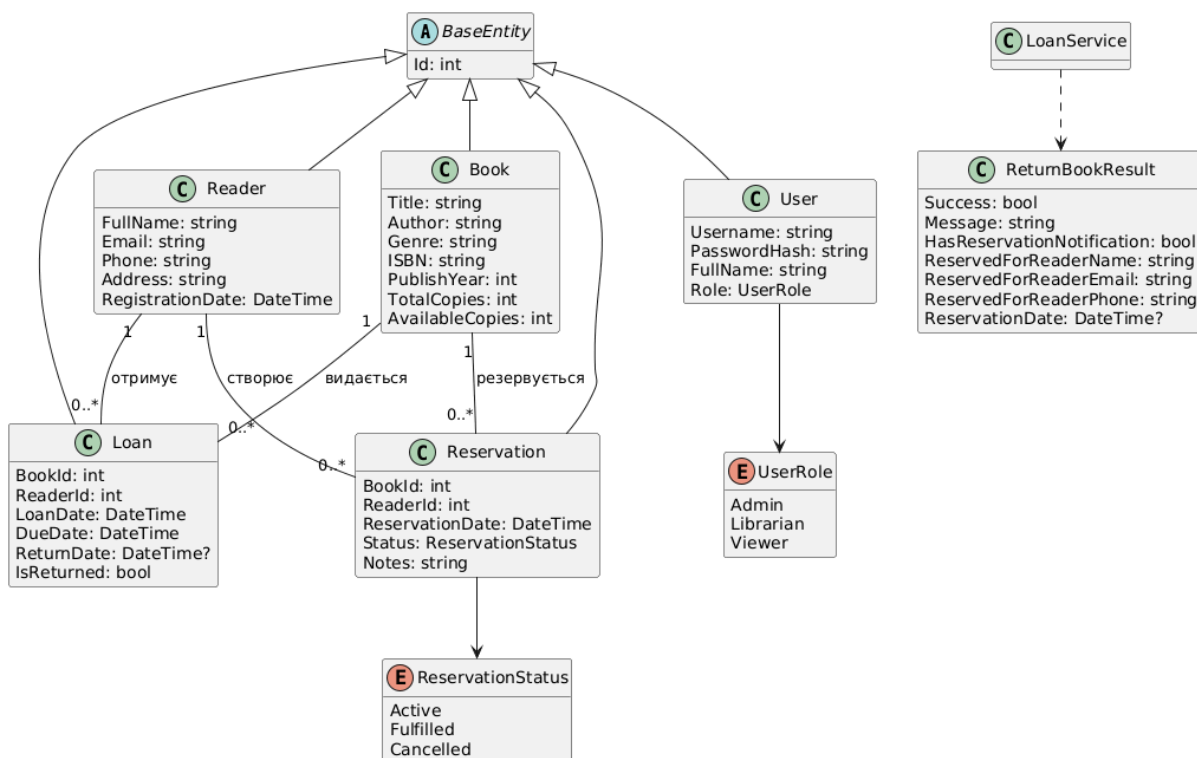


Рисунок 2.3 – UML-діаграма класів системи управління бібліотекою

Послідовність видачі книги наведено на рисунку 2.4.

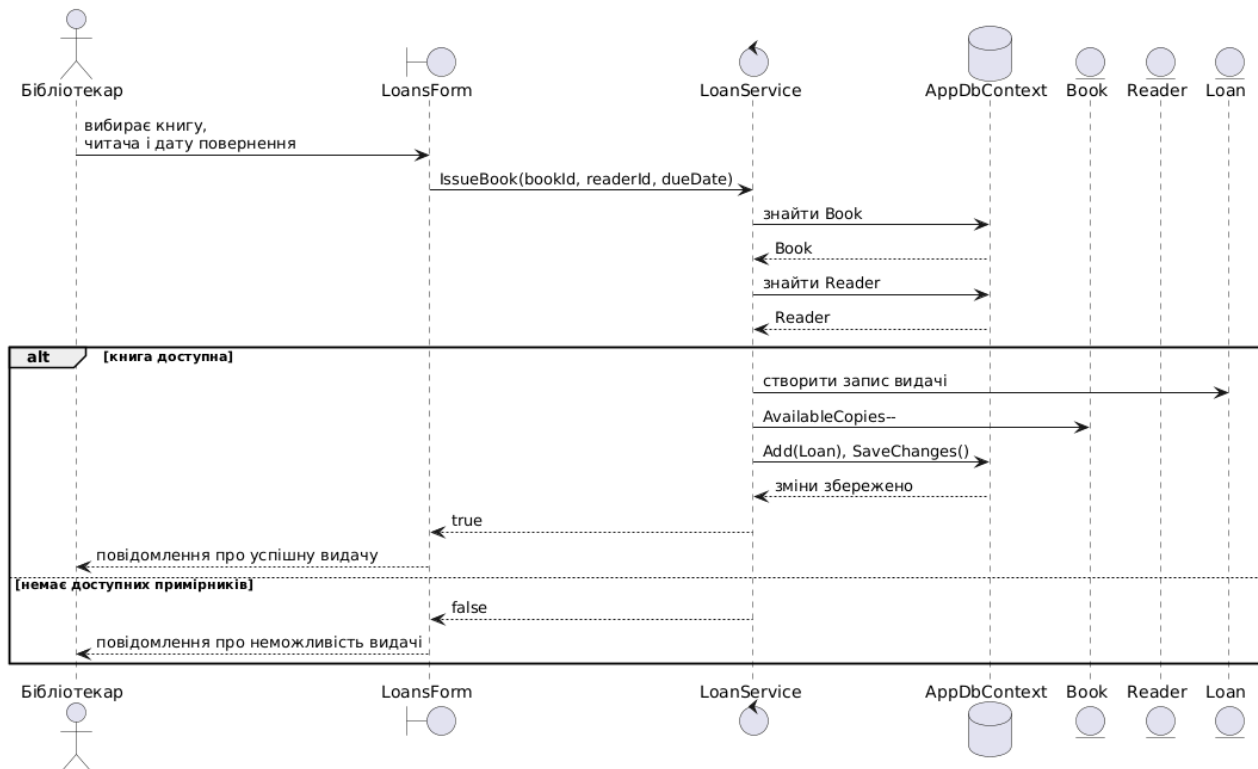


Рисунок 2.4 – UML-діаграма послідовності видачі книги читачу

Сценарій видачі книги починається з вибору читача та доступної книги. LoanService перевіряє, чи книга існує і чи має доступні примірники. Якщо AvailableCopies дорівнює нулю або книга не знайдена, видача не виконується. Якщо примірник доступний, створюється запис Loan з датою видачі та плановою датою повернення, а кількість доступних примірників зменшується на одиницю. Після збереження змін запис відображається у списку видач.

Сценарій повернення книги починається з вибору активної видачі. Система перевіряє, чи запис існує, чи книга пов'язана з видачею та чи вона ще не була повернена раніше. Після успішної перевірки встановлюється ReturnDate, а AvailableCopies збільшується на одиницю. Далі система перевіряє наявність першого активного резервування для цієї книги. Якщо таке резервування є, результат повернення містить повідомлення з ПІБ, email, телефоном і датою резервування читача, який очікує на книгу. Послідовність повернення книги наведено на рисунку 2.5.

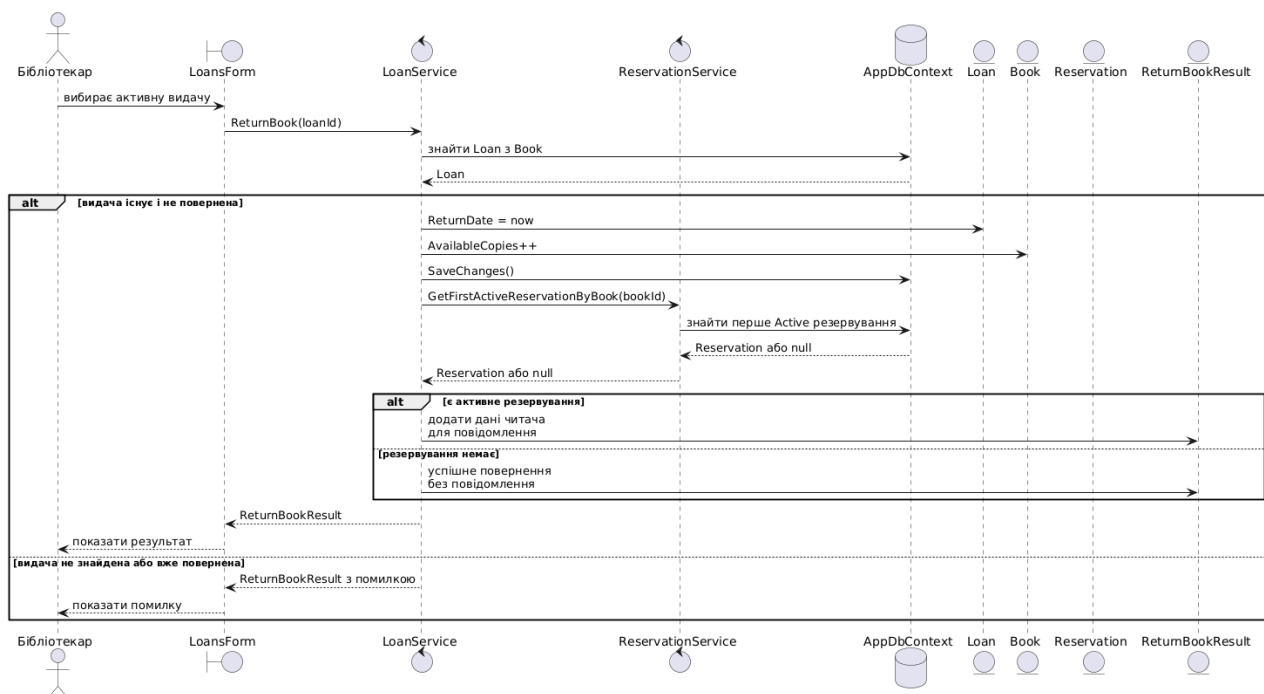


Рисунок 2.5 – UML-діаграма послідовності повернення книги та перевірки резервування

Сценарій формування звітності базується на вибірці прострочених видач. LoanService визначає записи, у яких ReturnDate відсутня, а DueDate менша за поточну дату. Після цього PdfReportService формує PDF документ із переліком прострочених книг, читачів і кількістю днів прострочення, а ReportService може експортувати ці дані у Excel файл. Такий підхід дає змогу бібліотекарю контролювати проблемні видачі та швидко отримувати документ для подальшої роботи.

Отже, у підрозділі визначено ролі користувачів, функціональні та нефункціональні вимоги, основні компоненти, модель даних і ключові сценарії роботи системи управління бібліотекою. Запропонована структура відповідає фактичній реалізації проєкту: Windows Forms використовується для інтерфейсу, Entity Framework Core і MS SQL Server – для реляційного зберігання даних, сервіси – для бізнес-логіки, а QuestPDF і ClosedXML – для формування звітів.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Вибір засобів реалізації системи управління бібліотекою здійснювався з урахуванням того, що програма повинна працювати як локальний настільний застосунок для бібліотекаря, адміністратора та користувача з правами перегляду. Основними вимогами до технологічного стеку є зручна робота з табличними даними, підтримка реляційних зв'язків між книгами, читачами, видачами й резервуваннями, формування звітів у PDF та XLSX форматах, а також контроль доступу до адміністративних дій.

Для побудови інтерфейсу обрано Windows Forms. Ця технологія платформи .NET призначена для створення настільних застосунків Windows з класичними вікнами, елементами керування та подієвою моделлю взаємодії [1]. Для бібліотечної системи такий підхід є доцільним, оскільки основні сценарії виконуються через форми книг, читачів, видач, резервувань, користувачів і звітів, де важливими є таблиці, кнопки дій, фільтри та діалогові вікна.

Під час вибору клієнтської технології розглянуто Windows Forms, WPF і вебзастосунок. Порівняння наведено в таблиці 2.1.

Таблиця 2.1 – Порівняння підходів до реалізації клієнтської частини бібліотечної системи

Підхід	Переваги	Недоліки щодо поставленого завдання	Висновок
Windows Forms	Швидко створюються довідникові форми, таблиці, кнопки, діалоги та вікна звітів; добре підходить для локального робочого місця бібліотекаря.	Менш гнучкий для складного сучасного дизайну порівняно з новішими технологіями інтерфейсу.	Обрано як основний підхід, оскільки система орієнтована на щоденну роботу з формами, списками і локальною базою даних.
WPF	Підтримує розвинене компонування, стилі, прив'язування даних і кращі можливості візуального оформлення.	Потребує більше часу на побудову XAML інтерфейсу, що не є визначальним для задачі обліку бібліотечного фонду.	Може бути використаний для майбутнього оновлення інтерфейсу, але для поточної реалізації є надмірним.
Вебзастосунок	Дає доступ через браузер і може бути зручним для мережевої бібліотечної системи.	Потребує вебсервера, налаштування безпеки, маршрутизації та іншої архітектури розгортання.	Не обрано, оскільки кваліфікаційна робота передбачає настільну систему для локального використання.

Прикладну логіку реалізовано мовою C# у середовищі .NET 8.0 для Windows. Код системи поділено на моделі, форми, сервіси, допоміжні класи та модуль доступу до даних. Така структура дозволяє винести операції з книгами, читачами, видачами, резервуваннями, користувачами і звітами в окремі сервіси, а форми використовувати переважно для взаємодії з користувачем.

Для роботи з базою даних використано Entity Framework Core та провайдер Microsoft SQL Server. Документація EF Core описує SQL Server provider як засіб

підключення EF Core до SQL Server і Azure SQL через пакет Microsoft.EntityFrameworkCore.SqlServer [2]. У системі AppDbContext містить набори Books, Readers, Loans, Reservations і Users, а також конфігурацію обов'язкових полів, максимальних довжин, унікальності логіна та зв'язків між сутностями.

Вибір системи керування базою даних обґрунтовано в таблиці 2.2.

Таблиця 2.2 – Порівняння засобів зберігання даних бібліотечної системи

Засіб	Переваги	Недоліки щодо поставленого завдання	Висновок
MS SQL Server LocalDB	Підтримує реляційну модель, зовнішні ключі, індекси, транзакції, інтеграцію з EF Core і локальне розгортання на Windows.	Потребує наявності компонентів SQL Server LocalDB на робочому місці.	Обрано для реалізації, оскільки система має реляційну структуру даних і працює як настільний Windows застосунок.
SQLite	Легка файлова база, проста для перенесення і не потребує сервера.	Менш зручна для майбутнього багатокористувацького сценарію та централізованого адміністрування бібліотечних даних.	Може бути використана для простого прототипу, але для поточної системи перевагу надано SQL Server LocalDB.
PostgreSQL	Надійна серверна СКБД, придатна для масштабованих і мережевих інформаційних систем.	Потребує окремого серверного встановлення і супроводу.	Доцільна для розширеної мережевої версії, але є надмірною для локальної настільної реалізації.

Параметри підключення винесено в appsettings.json і зчитуються через пакети Microsoft.Extensions.Configuration. Документація .NET описує систему конфігурації як механізм отримання налаштувань із різних джерел, зокрема JSON файлів [9]. Такий підхід дає змогу змінити рядок підключення без редагування програмного коду.

Для захисту облікових записів використано хешування паролів. У сервісі AuthService введений пароль перетворюється на SHA256 хеш і порівнюється з

PasswordHash у таблиці Users. Клас SHA256 у .NET належить до простору System.Security.Cryptography і реалізує алгоритм безпечного хешування SHA256 [10]. У межах навчальної системи це дозволяє не зберігати пароль у відкритому вигляді.

Для формування PDF звітів застосовано QuestPDF. Бібліотека надає Fluent API для створення PDF документів із текстом, таблицями, сторінками, стилями й компонованням [3]. У системі PdfReportService формує звіт по прострочених книгах, додає дату створення, таблицю з книгою, автором, читачем, строком повернення і кількістю днів прострочення.

Для експорту звітності в XLSX використано ClosedXML. За документацією ClosedXML, бібліотека дозволяє створювати файли Excel без встановлення Microsoft Excel, додавати аркуші й зберігати книгу у форматі XLSX [11]. У ReportService створюється аркуш Прострочені книги, налаштовуються заголовки, стилі, межі, колір прострочених рядків і автоматична ширина колонок.

Обрані засоби реалізації системи наведено в таблиці 2.3.

Таблиця 2.3 – Обрані засоби реалізації системи управління бібліотекою

Засіб	Призначення в системі
C# і .NET 8.0	Реалізація прикладної логіки, моделей, сервісів, форм і допоміжних класів бібліотечної системи.
Windows Forms	Побудова настільного інтерфейсу для книг, читачів, видач, резервувань, користувачів і звітів.
Entity Framework Core	Опис реляційної моделі, виконання запитів, збереження змін і робота з міграціями структури бази даних.
MS SQL Server LocalDB	Локальне зберігання бібліотечного фонду, читачів, видач, резервувань і користувачів.
Microsoft.Extensions.Configuration	Зчитування рядка підключення з appsettings.json.
QuestPDF	Створення PDF звіту по прострочених книгах.
ClosedXML	Експорт звіту по простроченнях у XLSX файл.
SHA256	Хешування паролів користувачів під час авторизації.

Алгоритм авторизації починається з нормалізації введеного логіна та обчислення SHA256 хешу введеного пароля. Далі система виконує пошук користувача за логіном і хешем пароля без відстеження змін, після чого зберігає дані поточного користувача у SessionService. Роль користувача визначає доступ до форм і критичних операцій: адміністратор керує користувачами, бібліотекар працює з основними бібліотечними процесами, а користувач з правами перегляду не змінює дані.

Алгоритм видачі книги реалізовано у LoanService. Спочатку за ідентифікатором шукається книга і перевіряється кількість доступних примірників. Якщо доступні копії відсутні, операція не виконується. Якщо книга доступна, створюється запис Loan з датою видачі та строком повернення, після чого AvailableCopies зменшується на одиницю і зміни зберігаються в базі даних.

Алгоритм повернення книги виконує зворотну операцію. Система знаходить активний запис видачі, встановлює дату повернення, збільшує кількість доступних примірників і перевіряє перше активне резервування для цієї книги. Якщо резервування існує, користувачу повертається повідомлення з іменем, телефоном та електронною адресою читача, для якого книга очікується.

Алгоритм резервування побудовано за принципом запобігання дублюванню активних заявок. Перед створенням резервування система перевіряє наявність книги, читача і вже активного резервування цього читача на ту саму книгу. Нове резервування отримує статус Active, дату створення і примітку. Під час скасування або виконання статус змінюється на Cancelled або Fulfilled, що дозволяє зберігати історію заявок.

Алгоритм формування звітів використовує вибірку прострочених видач. Простроченим вважається запис, у якого ReturnDate відсутня, а DueDate менша за поточну дату. Для PDF звіту дані передаються в PdfReportService, де створюється документ з таблицею і підрахунком загальної кількості прострочень. Для XLSX звіту ReportService створює книгу, додає заголовок, дату формування, рядки видач і кількість днів прострочення.

Отже, для реалізації системи управління бібліотекою обрано C#, Windows Forms, Entity Framework Core, MS SQL Server LocalDB, QuestPDF, ClosedXML і засоби конфігурації .NET. Такий набір технологій відповідає настільному характеру розробки, забезпечує реляційне зберігання бібліотечних даних, контроль ролей, облік руху примірників, резервування та формування звітів у зручних для користувача форматах.

РОЗДІЛ 3

РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ БІБЛІОТЕКОЮ

3.1 Практична реалізація об'єкта проєктування

Практична реалізація системи управління бібліотекою виконана як настільний застосунок Windows Forms. У коді проєкту виділено модулі форм, сервісів, моделей, допоміжних класів і доступу до даних. Такий поділ дає змогу залишити у формах логіку взаємодії з користувачем, а операції з книгами, читачами, видачами, резервуваннями, користувачами та звітами виконувати через окремі сервісні класи.

Фрагмент середовища розробки з відкритою структурою проєкту наведено на рисунку 3.1.

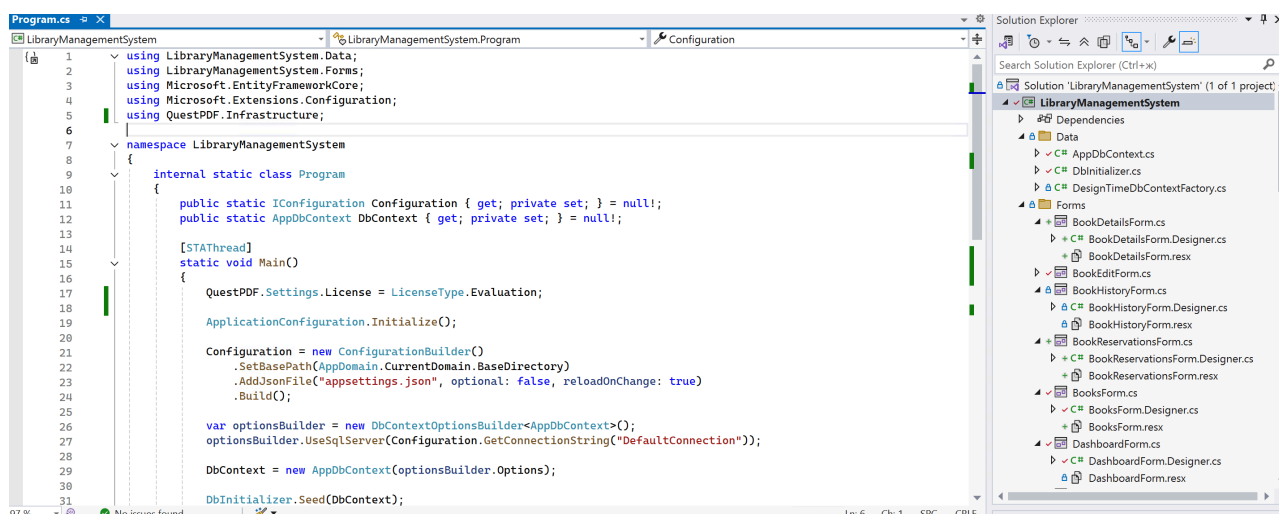


Рисунок 3.1 – Структура проєкту LibraryManagementSystem у середовищі розробки

Запуск застосунку починається з ініціалізації конфігурації, створення контексту бази даних і відображення форми авторизації. Після успішного входу відкривається головна форма системи, що забезпечує доступ до функціональних модулів. Фрагмент коду запуску наведено у лістингу 3.1.

Лістинг 3.1 – Ініціалізація застосунку і підключення до бази даних

```
[STAThread]
static void Main()
{
    QuestPDF.Settings.License = LicenseType.Evaluation;
    ApplicationConfiguration.Initialize();

    Configuration = new ConfigurationBuilder()
        .SetBasePath(AppDomain.CurrentDomain.BaseDirectory)
        .AddJsonFile("appsettings.json", optional: false,
reloadOnChange: true)
        .Build();

    var optionsBuilder = new
DbContextOptionsBuilder<AppDbContext>();

optionsBuilder.UseSqlServer(Configuration.GetConnectionString("Def
aultConnection"));

    DbContext = new AppDbContext(optionsBuilder.Options);
    DbInitializer.Seed(DbContext);

    using var loginForm = new LoginForm();
    if (loginForm.ShowDialog() == DialogResult.OK)
        Application.Run(new MainForm());
}
```

Кінець лістингу 3.1

Після авторизації користувач потрапляє на головну інформаційну панель. Вона відображає кількість книг, доступних примірників, читачів, активних видач, прострочених записів, резервувань і суму поточних штрафів, що дає бібліотекарю швидке уявлення про стан фонду. Головну панель системи показано на рисунку 3.2.

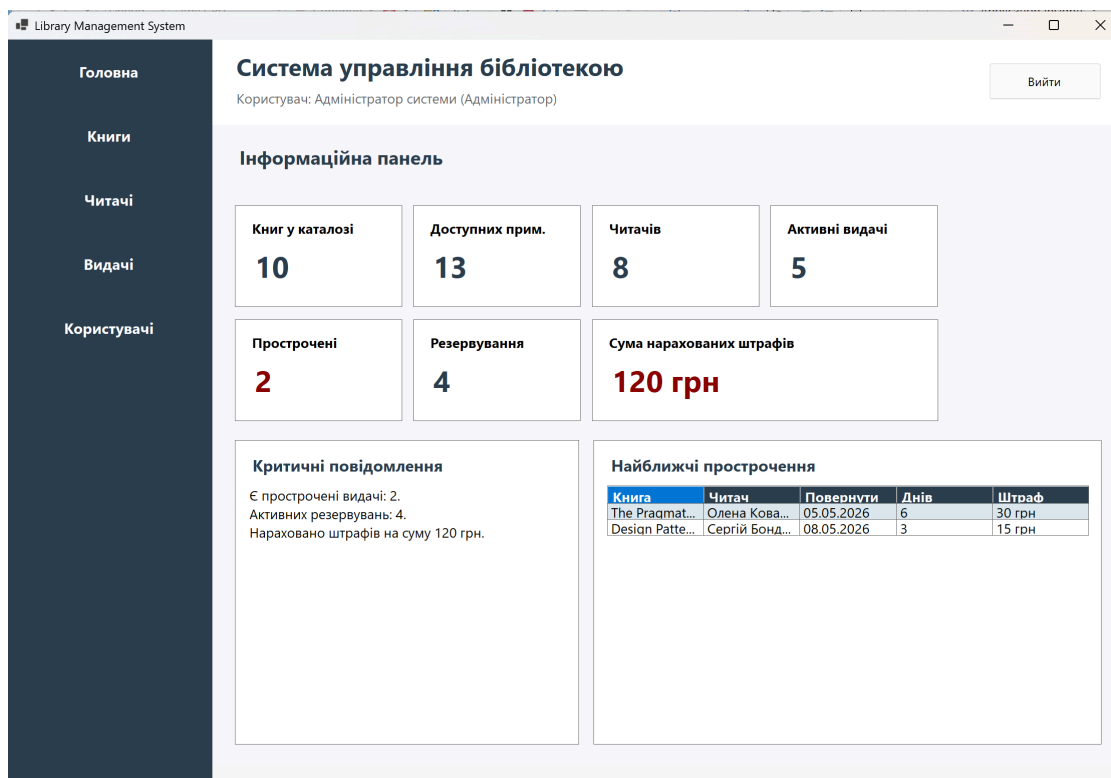


Рисунок 3.2 – Головна інформаційна панель системи управління бібліотекою

Модуль керування книгами реалізує перегляд, пошук, сортування, додавання, редагування, видалення, перегляд історії видач і відкриття картки книги. У сервісі BookService пошук виконується за назвою, автором або ISBN, а результат впорядковується за назвою книги. Відповідний фрагмент наведено у лістингу 3.2.

Лістинг 3.2 – Пошук і отримання списку книг у BookService

```
public List<Book> GetAll(string? search = null)
{
    var query = _context.Books.AsQueryable();
    if (!string.IsNullOrEmpty(search))
    {
        query = query.Where(b =>
            b.Title.Contains(search) ||
            b.Author.Contains(search) ||
            b.ISBN.Contains(search));
    }
}
```

```
return query.OrderBy(b => b.Title).ToList();
}
```

Кінець лістингу 3.2

Інтерфейс керування книжковим фондом наведено на рисунку 3.3. Він поєднує таблицю фонду, поле пошуку і кнопки основних операцій, тому бібліотекар може швидко перейти від перегляду до редагування або видачі книги.

Назва	Автор	Жанр	ISBN	Рік	Всього	Доступно
Artificial Intelligen...	Stuart Russell, Pete...	Штучний інтелект	9780134610993	2020	2	1
Clean Code	Robert C. Martin	Програмування	9780132350884	2008	4	2
Code Complete	Steve McConnell	Програмування	9780735619678	2004	2	0
Computer Networks	Andrew S. Tanenba...	Комп'ютерні мер...	9780132126953	2010	3	1
Database System ...	Abraham Silbersch...	Бази даних	9780073523323	2019	4	2
Design Patterns	Erich Gamma, Rich...	Програмування	9780201633610	1994	2	0
Introduction to AL...	Thomas H. Cormen	Алгоритми	9780262033848	2009	5	2
Operating System ...	Abraham Silbersch...	Операційні систе...	9781118063330	2018	4	3
Refactoring	Martin Fowler	Програмування	9780201485677	1999	3	1
The Pragmatic Pro...	Andrew Hunt, Davi...	Програмування	9780201616224	1999	3	1

Рисунок 3.3 – Форма керування книжковим фондом

Приклад форми редагування книги наведено на рисунку 3.4.

Книга

Назва: Introduction to Algorithms

Автор: Thomas H. Cormen

Жанр: Алгоритми

ISBN: 9780262033848

Рік: 2009

Всього: 5

Доступ: 2

Зберегти Скасувати

Рисунок 3.4 – Діалогова форма додавання або редагування книги

Додавання та редагування книги винесено в окрему діалогову форму. У ній користувач вводить назву, автора, жанр, ISBN, рік видання, загальну кількість примірників і кількість доступних примірників.

Операція видачі книги реалізована через сервіс `LoanService`. Перед створенням запису видачі система перевіряє наявність книги та кількість доступних примірників. Якщо примірник доступний, створюється запис `Loan`, а поле `AvailableCopies` зменшується на одиницю. Фрагмент коду операції видачі наведено у лістингу 3.3.

Лістинг 3.3 – Оформлення видачі книги читачу

```
public bool IssueBook(int bookId, int readerId, DateTime
dueDate)
{
    var book = _context.Books.Find(bookId);
    if (book == null || book.AvailableCopies <= 0)
        return false;
    var loan = new Loan
    {
        BookId = bookId,
        ReaderId = readerId,
        LoanDate = DateTime.Now,
        DueDate = dueDate
    };

    book.AvailableCopies--;
    _context.Loans.Add(loan);
    _context.SaveChanges();
    return true;
}
```

Кінець лістингу 3.3

Форма оформлення видачі містить книгу, читача, дату видачі та заплановану дату повернення. Приклад цієї форми наведено на рисунку 3.5.

Нова видача — Computer Networks

Оформлення видачі

Книга: Computer Networks

Читач: Андрій Мельник

Дата видачі: 11.05.2026

Повернути до: 25.05.2026

Оформити Скасувати

Рисунок 3.5 – Форма оформлення видачі книги

Картку книги показано на рисунку 3.6.

Картка книги — Design Patterns

Картка книги

Назва:	Design Patterns
Автор:	Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides
Жанр:	Програмування
ISBN:	9780201633610
Рік видання:	1994
Усього примірників:	2
Доступно:	0
Активні резервування:	2
Статус:	Недоступна

Усього видач
1

Активні
1

Повернуті
0

Прострочені
1

Видати книгу

Видати

Забронювати

Список резервувань

Історія видач

Редагувати

Закрити

Рисунок 3.6 – Картка книги з діями та статистикою використання

Картка книги використовується для швидкого перегляду стану конкретного видання: загальної кількості примірників, доступності, активних видач, прострочень, резервувань і переходу до пов'язаних операцій. Розширений фрагмент коду сервісу видач і повернення книг наведено в додатку А.

3.2 Розробка бази даних

База даних системи побудована у реляційній моделі та містить таблиці Books, Readers, Loans, Reservations і Users. Основні дані книжкового фонду зберігаються в Books, персональні дані читачів – у Readers, операції видачі – у Loans, заявки на очікування книги – у Reservations, а облікові записи працівників – у Users. Зв'язки між таблицями забезпечують історію руху кожної книги та історію обслуговування кожного читача.

Схему бази даних інформаційної системи наведено на рисунку 3.7.

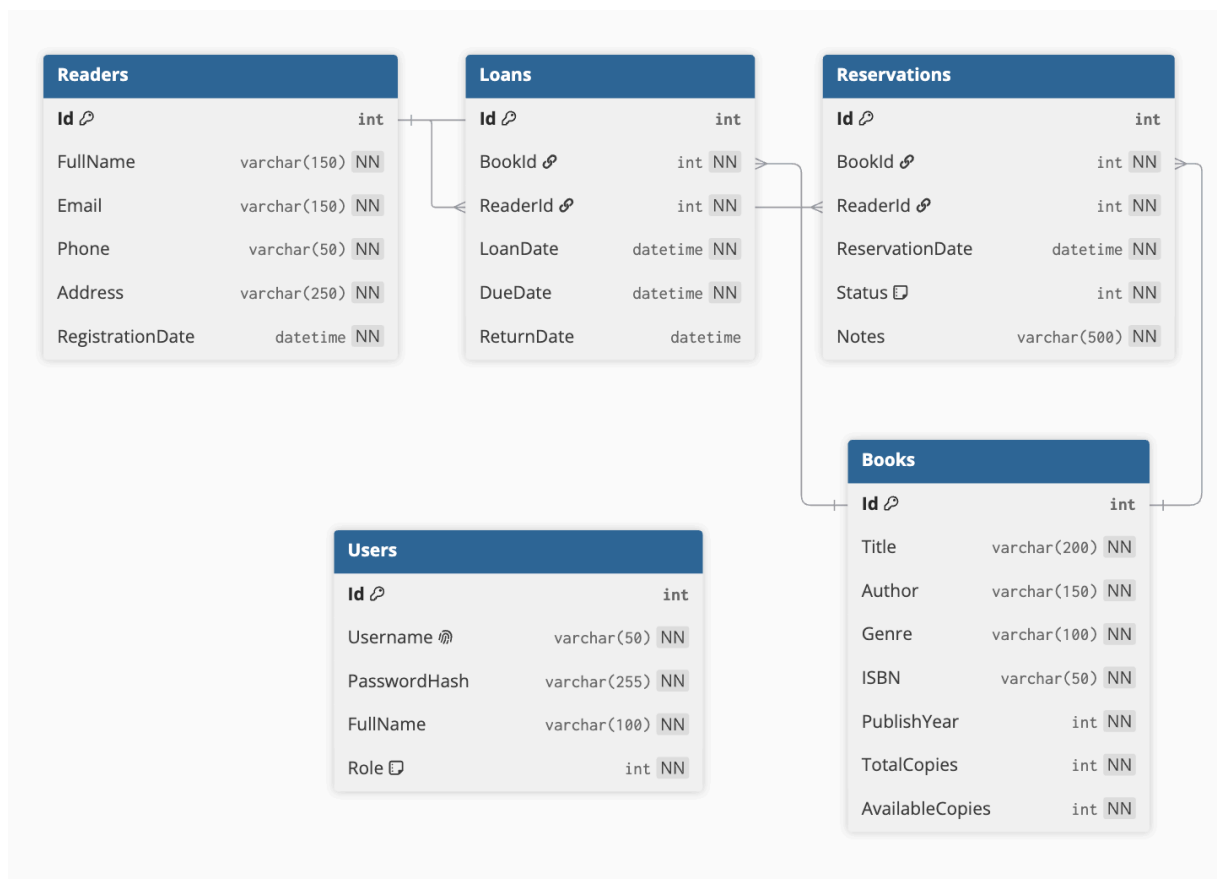


Рисунок 3.7 – Схема бази даних системи управління бібліотекою

Фізичну структуру таблиць бази даних у середовищі SQL Server показано на рисунку 3.8. На прикладі таблиці Books видно поля для назви, автора, жанру, ISBN, року видання, загальної кількості примірників і кількості доступних примірників.

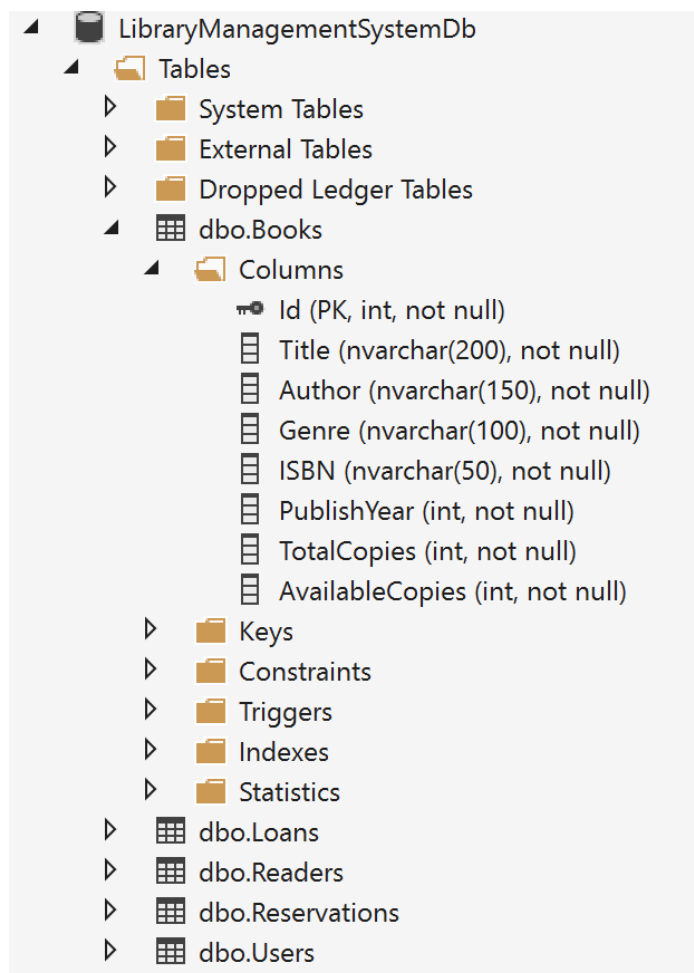


Рисунок 3.8 – Перегляд таблиць бази даних у SQL Server Object Explorer

Опис зв'язків і обмежень реалізовано в AppDbContext засобами Entity Framework Core. Для ключових текстових полів задано обов'язковість і максимальну довжину, для логіна користувача створено унікальний індекс, а зв'язки Loan і Reservation з книгами та читачами налаштовано з обмеженням каскадного видалення. Це запобігає випадковому вилученню книги або читача, якщо з ними вже пов'язані записи видач чи резервування. Фрагмент конфігурації наведено у лістингу 3.4.

Лістинг 3.4 – Конфігурація сутностей і зв'язків у AppDbContext

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    modelBuilder.Entity<Book>()
        .Property(b => b.Title)
        .IsRequired()
        .HasMaxLength(200);

    modelBuilder.Entity<User>()
        .HasIndex(u => u.Username)
        .IsUnique();

    modelBuilder.Entity<Loan>()
        .HasOne(l => l.Book)
        .WithMany(b => b.Loans)
        .HasForeignKey(l => l.BookId)
        .OnDelete(DeleteBehavior.Restrict);

    modelBuilder.Entity<Reservation>()
        .HasOne(r => r.Reader)
        .WithMany(rd => rd.Reservations)
        .HasForeignKey(r => r.ReaderId)
        .OnDelete(DeleteBehavior.Restrict);
}
```

Кінець лістингу 3.4

У таблиці Books поле TotalCopies відображає загальну кількість примірників, а AvailableCopies – поточну кількість доступних примірників. Під час видачі AvailableCopies зменшується, під час повернення збільшується. Таблиця Loans зберігає BookId, ReaderId, LoanDate, DueDate і ReturnDate, тому активною вважається видача без дати повернення. Таблиця Reservations містить статус Active, Cancelled або Fulfilled, що дозволяє не втрачати історію резервувань після їх скасування або виконання.

Для створення початкових даних використано DbInitializer. Він наповнює базу демонстраційними книгами, читачами й користувачами, що дає змогу одразу перевіряти пошук, видачу, повернення, прострочення і звітність без ручного введення великого набору тестових записів.

3.3 Захист інформаційно-комп'ютерної системи

Захист інформаційно-комп'ютерної системи реалізовано на рівні авторизації користувачів, розмежування ролей, збереження хешів паролів і обмеження критичних операцій у формах. Користувач не переходить до головного вікна без успішного входу, а після входу дані поточного користувача зберігаються в SessionService.

У системі передбачено три ролі: адміністратор, бібліотекар і користувач з правами перегляду. Адміністратор може керувати обліковими записами та виконувати всі операції, бібліотекар працює з книгами, читачами, видачами й резервуваннями, а користувач з правами перегляду не може змінювати критичні дані. Відображення поточного користувача і ролі в інтерфейсі показано на рисунку 3.9.

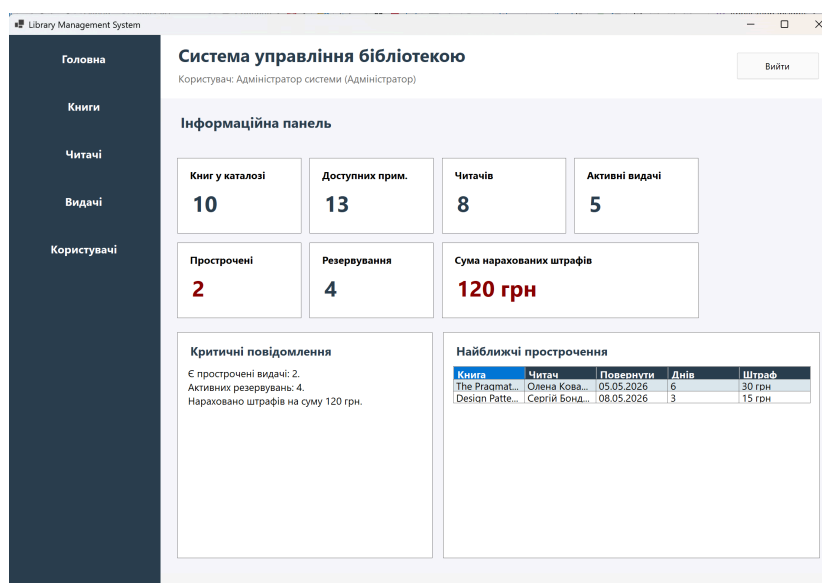


Рисунок 3.9 – Відображення поточного користувача після авторизації

Пароль не порівнюється у відкритому вигляді. Під час входу введене значення перетворюється на SHA256 хеш, після чого виконується пошук користувача з відповідним логіном і хешем пароля. Фрагмент коду авторизації наведено у лістингу 3.5.

Лістинг 3.5 – Перевірка логіна і хешу пароля в AuthService

```
public User? Authenticate(string username, string password)
{
    string normalizedUsername = username.Trim();
    string passwordHash = ComputeSha256(password);

    return _context.Users
        .AsNoTracking()
        .FirstOrDefault(u =>
            u.Username == normalizedUsername &&
            u.PasswordHash == passwordHash);
}

public static string ComputeSha256(string value)
{
    using var sha256 = SHA256.Create();
    byte[] bytes =
sha256.ComputeHash(Encoding.UTF8.GetBytes(value));
    return Convert.ToHexString(bytes);
}
```

Кінець лістингу 3.5

Розмежування доступу виконано через набір методів SessionService. Форми викликають ці методи під час налаштування кнопок і перед виконанням дій. Наприклад, видалення книг і керування користувачами доступні лише адміністратору, а оформлення видачі або повернення – адміністратору та бібліотекарю. Фрагмент перевірки ролей наведено у лістингу 3.6.

Лістинг 3.6 – Перевірка прав користувача в SessionService

```
public static bool CanEditBooks()  
{  
    return IsAdmin() || IsLibrarian();  
}  
  
public static bool CanCreateLoans()  
{  
    return IsAdmin() || IsLibrarian();  
}  
  
public static bool CanManageUsers()  
{  
    return IsAdmin();  
}
```

Кінець лістингу 3.6

Додатковий рівень цілісності забезпечують перевірки перед видаленням даних. Книга не видаляється, якщо має активні видачі, а користувач не може видалити власний обліковий запис. Такі обмеження зменшують ризик втрати службових даних під час повсякденної роботи.

Захист від некоректних операцій також підтримується на рівні бази даних: унікальний індекс для Username унеможливорює дублювання логінів, а зв'язки з DeleteBehavior.Restrict не дозволяють видаляти пов'язані записи без попередньої перевірки бізнес-логіки.

3.4 Тестування та налагодження інформаційно-комп'ютерної системи

Тестування та налагодження системи виконувалося за основними сценаріями роботи бібліотеки: вхід користувача, перегляд головної панелі, пошук книг і читачів, додавання та редагування записів, оформлення видачі, повернення книги, створення резервування, визначення прострочених записів, розрахунок штрафу, формування PDF звіту та експорт даних у XLSX.

Під час перевірки інтерфейсу окремо тестувалася форма видач. У ній користувач може фільтрувати записи за статусом, датами та текстовим пошуком,

а також швидко сформувати звіт по прострочених книгах. Приклад екрана тестування модуля видач наведено на рисунку 3.10.

Система управління бібліотекою

Користувач: Адміністратор системи (Адміністратор)

Управління видачами

Статус: Пошук:

З дати: По дату:

Книга	Читач	Дата видачі	Повернути до	Дата повернення	Статус	Днів прострочення	Штраф
Introduction to ...	Наталія Роман...	08.05.2026	22.05.2026	11.05.2026	Повернуто	0	0 грн
Clean Code	Іван Петренко	06.05.2026	20.05.2026		Активна	0	0 грн
Artificial Intellig...	Тетяна Гришук	04.05.2026	18.05.2026		Активна	0	0 грн
Computer Netw...	Дмитро Савчук	26.04.2026	10.05.2026	09.05.2026	Повернуто	0	0 грн
Design Patterns	Сергій Бондар	23.04.2026	08.05.2026		Прострочено	3	15 грн
The Pragmatic P...	Олена Ковальч...	21.04.2026	05.05.2026		Прострочено	6	30 грн
Refactoring	Андрій Мельник	11.04.2026	25.04.2026	24.04.2026	Повернуто	0	0 грн
Code Complete	Марія Шевчук	01.04.2026	16.04.2026	21.04.2026	Повернуто із з...	5	25 грн
Database Syste...	Іван Петренко	02.03.2026	17.03.2026	22.03.2026	Повернуто із з...	5	25 грн
Clean Code	Олена Ковальч...	10.02.2026	25.02.2026	02.03.2026	Повернуто із з...	5	25 грн

Рисунок 3.10 – Перевірка фільтрації видач і відображення прострочених записів

Коректність розрахунку прострочення перевірялася за датою повернення, плановою датою DueDate і поточною датою. Якщо книга вже повернута, кількість днів прострочення рахується від фактичної дати повернення; якщо ні – від поточної дати. Фрагмент допоміжного класу для розрахунку прострочень і штрафів наведено у лістингу 3.7.

Лістинг 3.7 – Розрахунок кількості днів прострочення та штрафу

```
public static int GetOverdueDays(Loan loan)
{
    if (loan == null)
        return 0;

    DateTime dueDate = loan.DueDate.Date;

    if (loan.ReturnDate.HasValue)
```

```

        return Math.Max(0, (loan.ReturnDate.Value.Date -
dueDate).Days);

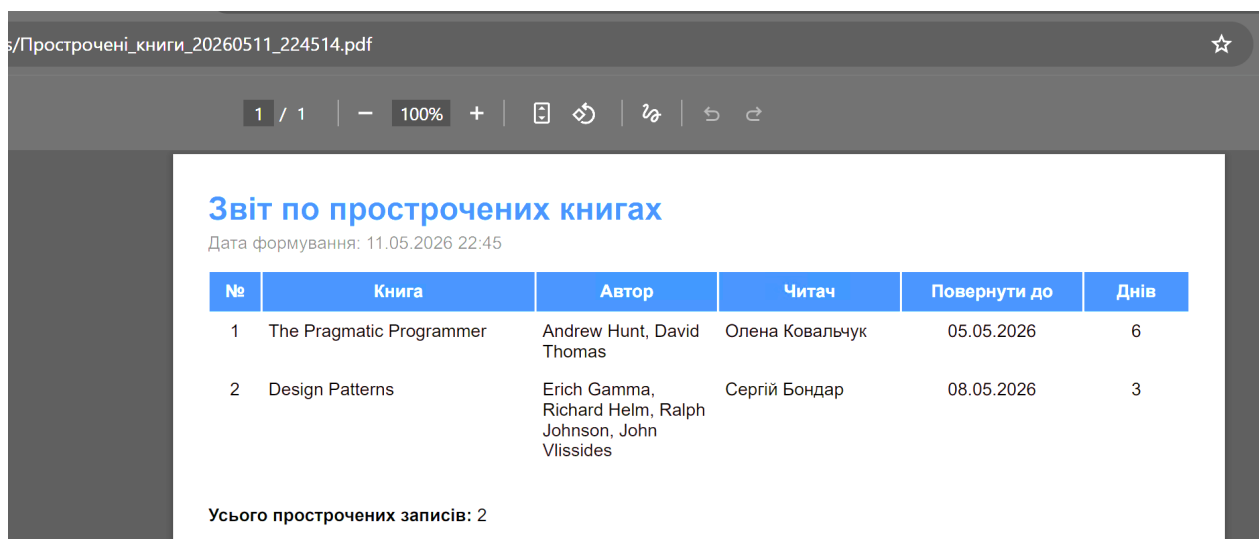
        return Math.Max(0, (DateTime.Today - dueDate).Days);
    }

    public static decimal GetFineAmount(Loan loan)
    {
        return GetOverdueDays(loan) * FinePerDay;
    }

```

Кінець лістингу 3.7

Для перевірки звітності сформовано PDF документ із переліком прострочених книг. У звіті виводяться книга, автор, читач, дата повернення і кількість днів прострочення. Результат формування PDF звіту наведено на рисунку 3.11.



Звіт по прострочених книгах

Дата формування: 11.05.2026 22:45

№	Книга	Автор	Читач	Повернути до	Днів
1	The Pragmatic Programmer	Andrew Hunt, David Thomas	Олена Ковальчук	05.05.2026	6
2	Design Patterns	Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides	Сергій Бондар	08.05.2026	3

Усього прострочених записів: 2

Рисунок 3.11 – Сформований PDF звіт по прострочених книгах

Експорт у XLSX перевірявся шляхом відкриття сформованого файлу в табличному редакторі. Прострочені рядки виділяються кольором, а колонки автоматично підлаштовуються під вміст. Фрагмент коду формування рядків

Excel-звіту наведено у лістингу 3.8, а результат експорту показано на рисунку 3.12.

Лістинг 3.8 – Формування рядків Excel-звіту з простроченими книгами

```
foreach (var loan in overdueLoans)
{
    int overdueDays = (DateTime.Today - loan.DueDate.Date).Days;

    worksheet.Cell(row, 2).Value = loan.Book?.Title ?? "";
    worksheet.Cell(row, 4).Value = loan.Reader?.FullName ?? "";
    worksheet.Cell(row, 7).Value =
loan.DueDate.ToString("dd.MM.yyyy");
    worksheet.Cell(row, 8).Value = overdueDays;

    var dataRange = worksheet.Range(row, 1, row, 8);
    dataRange.Style.Border.OutsideBorder =
XLBorderStyleValues.Thin;

    if (overdueDays > 0)
        dataRange.Style.Fill.BackgroundColor = XLColor.MistyRose;

    row++;
}
```

Кінець лістингу 3.8

Звіт по прострочених книгах					
Дата формування: 11.05.2026 22:46					
№	Книга	Автор	Читач	Email	Телефо
1	The Pragmatic Programmer	Andrew Hunt, David Thomas	Олена Ковальчук	olena.kovalchuk@example.com	+380671234
2	Design Patterns	Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides	Сергій Бондар	serhii.bondar@example.com	+380661112

Рисунок 3.12 – Перевірка експорту прострочених видач у XLSX

Також перевірено роботу форм редагування читачів і книг. Тестові дії включали введення контактних даних, зміну кількості примірників, збереження запису, повторне відкриття форми та відображення змінених даних у таблиці. Приклад форми редагування читача наведено на рисунку 3.13.

The screenshot displays a web application interface for managing readers. At the top, there is a search bar with the placeholder text "Пошук за ПІБ, email, телефон..." and a "Пошук" button. To the right of the search bar are five buttons: "Додати" (Add), "Редагувати" (Edit), "Видалити" (Delete), "Історія" (History), and "Деталі" (Details). Below these buttons is a table with the following columns: "ПІБ" (Full Name), "Email", "Телефон" (Phone), "Адреса" (Address), and "Реєстрація" (Registration). The table contains several rows of data, with the row for "Марія Шевчук" highlighted. A modal form titled "Читач" (Reader) is open in the foreground, allowing for the editing of the selected reader's information. The form includes fields for "ПІБ:" (Full Name), "Email:", "Телефон:" (Phone), "Адреса:" (Address), and "Дата реєстрації:" (Registration Date). The "ПІБ:" field contains "Марія Шевчук", the "Email:" field contains "maria.shevchuk@example.com", the "Телефон:" field contains "+380991234567", the "Адреса:" field contains "м. Рівне, вул. Соборна, 15", and the "Дата реєстрації:" field contains "11.05.2026". At the bottom of the modal form are two buttons: "Зберегти" (Save) and "Скасувати" (Cancel).

ПІБ	Email	Телефон	Адреса	Реєстрація
Мельник			вул. Волі, 8	11.11.202
Савчук			Конякіна, 5	11.03.202
Івченко			Лесі Українк...	11.07.202
Шевчук			Соборна, 15	11.12.202
Романюк			вул. Руська, ...	11.02.202
Ковальчук			Незалежно...	11.09.202
Бондар			Ордоцька, ...	11.01.202
Гришук			вул. Данил...	11.04.202

Рисунок 3.13 – Перевірка форми додавання та редагування читача

За результатами тестування підтверджено, що система коректно обробляє основні бібліотечні операції, не дозволяє видавати книгу за відсутності доступних примірників, оновлює кількість доступних копій після видачі й повернення, відображає активні резервування та формує звіти у двох форматах. Розширений фрагмент коду формування PDF звіту наведено в додатку Б.

ВИСНОВКИ

У кваліфікаційній роботі розроблено систему управління бібліотекою з використанням C#, Windows Forms, Entity Framework Core, MS SQL Server, QuestPDF і ClosedXML. Розроблений програмний засіб поєднує настільний інтерфейс, реляційну модель даних і набір сервісів для виконання основних бібліотечних операцій. Система забезпечує ведення книжкового фонду, облік читачів, видачі та повернення книг, резервування примірників, рольовий доступ і формування звітів. Її використання дає змогу перейти від розрізненого ручного обліку до структурованого збереження даних, у якому кожна книга, читач, видача або резервування має визначене місце в загальній моделі.

Здійснено аналіз процесів обліку бібліотечного фонду, читачів, видач, повернень і резервувань. Визначено, що для бібліотеки важливо контролювати кількість доступних примірників, бачити історію читача, швидко виявляти прострочені книги та не допускати втрати інформації про активні резервування. Особливу увагу приділено руху примірників, оскільки саме він впливає на актуальність даних про доступність книги. Якщо видача або повернення фіксуються несвоєчасно, працівник бібліотеки може отримати неправильну інформацію про фонд, тому автоматизація цього процесу є практично важливою.

Сформовано функціональні вимоги до настільного застосунку з урахуванням ролей адміністратора, бібліотекаря та користувача з правами перегляду. Такий підхід дає змогу відокремити адміністративні операції від повсякденної роботи з книгами й читачами та обмежити небажане редагування даних. Адміністратор відповідає за облікові записи й службові налаштування, бібліотекар виконує основні операції з фондом, видачами та резервуваннями, а користувач з правами перегляду може аналізувати дані без зміни записів. Завдяки цьому система підтримує контрольований доступ до різних частин функціональності.

Спроектовано структуру бази даних для зберігання книг, читачів, видач, резервувань і користувачів. Модель даних враховує зв'язки між книгою та її

видачами, між читачем і його історією, а також статуси резервувань, що забезпечує логічну цілісність бібліотечних операцій. Окреме зберігання видач і резервувань дає змогу не змішувати фактичне користування книгою з очікуванням її отримання. Це важливо для коректного формування повідомлень після повернення книги та для аналізу попиту на окремі видання.

Обґрунтовано вибір технологій реалізації. C# і Windows Forms використано для створення настільного інтерфейсу, Entity Framework Core та MS SQL Server – для роботи з реляційною базою даних, QuestPDF – для формування PDF-звітів, ClosedXML – для експорту даних у формат електронних таблиць. Обраний стек відповідає характеру системи, оскільки вона орієнтована на роботу працівника бібліотеки за локальним робочим місцем і не потребує складної серверної інфраструктури. Використання Entity Framework Core спрощує роботу з пов'язаними сутностями, а окремі сервіси дають змогу винести бізнес-логіку за межі форм інтерфейсу.

Реалізовано модулі керування книжковим фондом, читачами, видачами, поверненнями, резервуваннями, користувачами та звітами. Окремо реалізовано хешування паролів, визначення прострочених видач, розрахунок штрафу, підготовку PDF-звіту й експорт прострочених записів до Excel. Під час реалізації враховано ситуації, коли книгу неможливо видати через відсутність доступних примірників, коли видачу вже було повернено, або коли для поверненої книги існує активне резервування. Такі перевірки зменшують ризик некоректних операцій і допомагають підтримувати актуальний стан бібліотечного фонду.

Перевірено працездатність системи на типових сценаріях роботи бібліотеки: додавання й редагування книг, реєстрація читачів, створення видачі, повернення книги, резервування примірника, перегляд історії, формування звітів і робота з користувачами різних ролей. Результати підтверджують, що розроблений застосунок може використовуватися для автоматизації основних операцій бібліотечного обліку, підвищення зручності роботи працівників бібліотеки та швидшого отримання актуальної інформації про стан книжкового фонду.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is Windows Forms. Microsoft Learn. URL: <https://learn.microsoft.com/dotnet/desktop/winforms/overview/> (дата звернення: 05.03.2026).
2. Microsoft SQL Server Database Provider. EF Core. Microsoft Learn. URL: <https://learn.microsoft.com/ef/core/providers/sql-server/> (дата звернення: 07.03.2026).
3. Quick start. QuestPDF Documentation. URL: <https://www.questpdf.com/quick-start.html> (дата звернення: 20.03.2026).
4. Koha Library Software. Koha Community. URL: <https://koha-community.org/> (дата звернення: 04.04.2026).
5. Koha Screenshots. Koha Wiki. URL: https://wiki.koha-community.org/wiki/Koha_Screenshots (дата звернення: 05.04.2026).
6. Evergreen ILS. Evergreen. URL: <https://evergreen-ils.org/> (дата звернення: 15.04.2026).
7. Evergreen Staff Client Screenshots. Evergreen Wiki. URL: https://wiki.evergreen-ils.org/doku.php?id=scratchpad:staff_client_screenshots (дата звернення: 15.04.2026).
8. SLiMS. Senayan Library Management System. URL: <https://slims.web.id/> (дата звернення: 23.04.2026).
9. Configuration in .NET. Microsoft Learn. URL: <https://learn.microsoft.com/dotnet/core/extensions/configuration> (дата звернення: 02.05.2026).
10. SHA256 Class. Microsoft Learn. URL: <https://learn.microsoft.com/dotnet/api/system.security.cryptography.sha256> (дата звернення: 12.05.2026).
11. ClosedXML Documentation. URL: <https://docs.closedxml.io/> (дата звернення: 15.05.2026).

ДОДАТКИ

ДОДАТОК А

ФРАГМЕНТ КОДУ СЕРВІСУ ВИДАЧ І ПОВЕРНЕННЯ КНИГ

```

public ReturnBookResult ReturnBook(int loanId)
{
    var result = new ReturnBookResult();
    var loan = _context.Loans
        .Include(l => l.Book)
        .FirstOrDefault(l => l.Id == loanId);
    if (loan == null || loan.IsReturned || loan.Book == null)
    {
        result.Success = false;
        result.Message = "Не вдалося виконати повернення.";
        return result;
    }
    loan.ReturnDate = DateTime.Now;
    loan.Book.AvailableCopies++;
    _context.SaveChanges();
    var reservationService = new ReservationService(_context);
    var reservation =
reservationService.GetFirstActiveReservationByBook(loan.BookId);
    result.Success = true;
    result.Message = "Книгу успішно повернено.";
    if (reservation != null && reservation.Reader != null)
    {
        result.HasReservationNotification = true;
        result.ReservedForReaderName =
reservation.Reader.FullName;
        result.ReservedForReaderEmail = reservation.Reader.Email
?? "";
        result.ReservedForReaderPhone = reservation.Reader.Phone
?? "";
        result.ReservationDate = reservation.ReservationDate;
    }
    return result;
}

```

ДОДАТОК Б

ФРАГМЕНТ КОДУ ФОРМУВАННЯ PDF ЗВІТУ

```

var document = Document.Create(container =>
{
    container.Page(page =>
    {
        page.Size(PageSizes.A4);
        page.Margin(20);
        page.DefaultTextStyle(x =>
x.FontSize(10).FontFamily("Arial"));
        page.Header().Column(column =>
        {
            column.Item().Text("Звіт по прострочених книгах")

.FontSize(18).Bold().FontColor(Colors.Blue.Darken3);
            column.Item().Text($"Дата формування:
{generatedAt:dd.MM.yyyy HH:mm}");
        });
        page.Content().PaddingVertical(10).Column(column =>
        {
            column.Item().Table(table =>
            {
                table.ColumnsDefinition(columns =>
                {
                    columns.ConstantColumn(30);
                    columns.RelativeColumn(3);
                    columns.RelativeColumn(2);
                    columns.RelativeColumn(2);
                    columns.RelativeColumn(2);
                    columns.ConstantColumn(60);
                });

                foreach (var loan in overdueLoans)
                {
                    int overdueDays = Math.Max(0,
                        (DateTime.Today -
loan.DueDate.Date).Days);
                    table.Cell().Text(loan.Book?.Title ?? "-");
                    table.Cell().Text(loan.Reader?.FullName ?? "-");

                    table.Cell().AlignCenter().Text(overdueDays.ToString());
                }
            });
        });
    });
});

document.GeneratePdf(filePath);

```