Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота бакалавра

на тему : «Розроблення інформаційного та програмного забезпечення електронної бібліотеки»

Виконав: студент групи ІПЗ20-1
Спеціальність 121 «Інженерія програмного забезпечення»
Гащак А.В.
(прізвище та ініціали)
Керівник к.ф.-м.н., доц. Мормуль М.Ф.
(науковий ступінь, вчене звання, прізвище та ініціали)
Рецензент Університет митної справи та фінансів
(місце роботи)
Доцент кафедри кібербезпеки та інформаційних технологій
(посада)
к.т.н., доцент Прокопович- Ткаченко Д.І.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2024

АНОТАЦІЯ

Гащак А.В. Розроблення інформаційного та програмного забезпечення електронної бібліотеки.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю 121 «Інженерія програмного забезпечення». – Університет митної справи та фінансів, Дніпро, 2024.

Сучасний розвиток інформаційних технологій відкриває нові можливості для розвитку електронних бібліотек як інструментів для доступу до знань та інформації. Ця дипломна робота спрямована на розробку інформаційного та програмного забезпечення для електронної бібліотеки з метою поліпшення якості обслуговування користувачів та забезпечення їм зручного доступу до ресурсів.

У роботі проводиться аналіз потреб користувачів електронних бібліотек, враховуючи їхні очікування та вимоги до функціоналу. На основі цього аналізу розробляється проект інформаційної системи, в якому визначаються основні компоненти та їхні взаємозв'язки.

Особлива увага приділяється процесу реалізації програмного забезпечення, яке включає в себе розробку інтерфейсу користувача, створення бази даних, реалізацію алгоритмів пошуку та обробки інформації.

Після завершення розробки виконується тестування та валідація програмного забезпечення з метою перевірки його працездатності, безпеки та відповідності вимогам.

Головний висновок роботи полягає в тому, що розроблене програмне забезпечення дозволяє значно покращити доступ користувачів до інформаційних ресурсів та полегшує процес їхнього користування, що є важливим в контексті сучасних вимог до розвитку освіти та науки.

Ключові слова: PHP, мова програмування, електронна бібліотека, WAMP Server, MySQL, база даних, HTML, CSS.

ABSTRACT

Haschak A.V. Development of information and software for the digital library.

Qualification work for a bachelor's degree in specialty 121 «Software Engineering». – University of Customs and Finance, Dnipro, 2024.

The modern development of information technologies opens up new opportunities for the development of electronic libraries as tools for access to knowledge and information. This thesis is aimed at the development of information and software for the electronic library in order to improve the quality of user service and provide them with convenient access to resources.

The work analyzes the needs of users of electronic libraries, taking into account their expectations and requirements for functionality. Based on this analysis, an information system project is developed, in which the main components and their interrelationships are defined.

Special attention is paid to the process of software implementation, which includes the development of the user interface, the creation of a database, the implementation of search and information processing algorithms.

After the development is completed, the software is tested and validated in order to check its functionality, security and compliance with the requirements.

The main conclusion of the work is that the developed software allows to significantly improve the access of users to information resources and facilitates the process of their use, which is important in the context of modern requirements for the development of education and science.

Keywords: PHP, programming language, electronic library, WAMP Server, MySQL, database, HTML, CSS

ЗМІСТ

ВСТУП

Електронна бібліотека – це вебплатформа або програмне забезпечення, яке надає доступ до цифрових версій книг, журналів, статей, аудіозаписів, відео, фотографій та інших матеріалів. Основна функція електронної бібліотеки полягає в тому, щоб забезпечити користувачам можливість знаходити, переглядати, читати або завантажувати цифрові ресурси за допомогою інтернету. Ключові функції електронної бібліотеки можна описати наступним чином.

1) Пошук інформації – користувачі можуть шукати матеріали за назвою, автором, ключовими словами або іншими критеріями.

2) Перегляд та читання – електронна бібліотека надає можливість переглядати та читати матеріали безпосередньо в онлайн-режимі через веббраузер або за допомогою спеціальних програм для читання електронних книг.

3) Завантаження – користувачі можуть завантажувати цифрові ресурси для подальшого використання офлайн або на пристроях з обмеженим доступом до Інтернету.

4) Організація та маркування – бібліотеки можуть надавати інструменти для організації збережених матеріалів у вигляді колекцій, списків бажань, закладок тощо.

5) Підтримка форматів – більшість електронних бібліотек підтримують різні формати файлів, такі як PDF, EPUB, MOBI, MP3, MPEG, JPEG тощо.

6) Доступність та мобільність – користувачі можуть отримувати доступ до електронних бібліотек з різних пристроїв, таких як комп'ютери, планшети, смартфони тощо, що робить їх доступними в будь-який час та в будь-якому місці.

7) Поширення та розповсюдження – електронні бібліотеки можуть допомагати в розповсюдженні цифрових книг і матеріалів шляхом надання їх авторам або видавцям можливості опублікувати їх на платформі.

8) Управління правами – бібліотеки можуть включати функції управління цифровими правами, такі як обмеження доступу, контроль копіювання або друку, захист від копіювання тощо.

Загалом, електронні бібліотеки створюють можливість доступу до широкого спектру культурних та освітніх ресурсів через Інтернет, полегшуючи процес пошуку та отримання інформації для користувачів.

Тема «Розробка інформаційного та програмного забезпечення електронної бібліотеки» є актуальною і важливою у сучасному світітому тому, що:

– відбувається цифрова трансформація у багатьох галузях, включаючи освіту та культуру. Електронні бібліотеки стають невід'ємною частиною цього процесу, дозволяючи мати доступ до знань та інформації в онлайн-форматі.

– електронні бібліотеки дозволяють людям з усього світу мати доступ до широкого спектру літератури та інформації без обмежень, які пов'язані з місцем проживання чи часом.

– розвиток технологій дозволяє постійно удосконалювати функціональні можливості електронних бібліотек, забезпечуючи користувачам зручний та ефективний інтерфейс для пошуку, читання та взаємодії з контентом.

– розробка програмного забезпечення для електронних бібліотек дозволяє автоматизувати багато рутинних процесів, таких як управління колекціями книг, обробка запитів користувачів, контроль доступу до ресурсів та інші.

– з підвищенням популярності онлайн-освіти зростає і важливість електронних бібліотек як засобу для навчання, досліджень та саморозвитку.

Отже, розробка програмного забезпечення для електронних бібліотек залишається актуальною та перспективною темою для дипломної роботи.

Метою даної дипломної роботи є створення інформаційної та програмної системи для електронної бібліотеки, яка забезпечить:

1) Зручний та швидкий доступ користувачів до колекцій електронних книг.

2) Можливість ефективного управління бібліотечними фондами для адміністраторів.

3) Надійне зберігання даних та їх захист від несанкціонованого доступу.

Завдання до дипломної роботи:

– Провести дослідження для визначення потреб та вимог цільової аудиторії електронної бібліотеки.

– Розробити детальний план структури та функціоналу програмного забезпечення для електронної бібліотеки. Визначити основні функції, можливості пошуку, систему авторизації та інші ключові аспекти системи.

– Розробити програмне забезпечення на основі розробленого проекту, використовуючи відповідні технології та інструменти розробки. Забезпечити ефективну роботу системи та її зручний інтерфейс для користувачів.

– Провести тестування програмного забезпечення для перевірки його працездатності, безпеки та відповідності вимогам. Виявити та виправити всі можливі помилки та недоліки.

– Підготувати документацію, включаючи технічне описання системи, інструкції з використання та адміністрування, а також звіт про роботу над дипломною роботою.

Кваліфікаційна робота складається зі вступу, 3-х розділів, висновків, використаних джерел з найменувань, додатків.

Обсяг роботи складається з 48 сторінок основного тексту з 115 сторінки кваліфікаційної роботи, 16 рисунків та 8 таблиць.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Огляд існуючого програмного забезпечення електронної бібліотеки

Огляд існуючого програмного забезпечення для електронних бібліотек включає розгляд різних рішень та їхніх характеристик з точки зору функціональності, ефективності та зручності використання. На ринку існує широкий вибір програм для електронних бібліотек, які можуть варіюватися від простих до складних систем з розширеним функціоналом.

Багато програмних рішень для електронних бібліотек пропонують основні функції, такі як каталогізація та організація колекцій книг, можливість пошуку та фільтрації матеріалів, підтримку різних форматів файлів, інтеграцію з системами керування бібліотекою та іншими додатками. Деякі програми також пропонують розширені функції, такі як взаємодія з читачами через віртуальні читальні зали, можливість ведення статистики використання та відстеження популярних запитів.

При виборі програмного забезпечення для електронної бібліотеки важливо враховувати потреби та вимоги конкретного закладу, а також його можливості щодо інтеграції та підтримки. Окрім того, слід звертати увагу на питання безпеки даних, підтримку користувацьких інтерфейсів та можливість розширення функціоналу у майбутньому.

Ось кілька прикладів найбільш відомих електронних бібліотек.

1) Project Gutenberg (рис. 1.1)

Project Gutenberg – це один з найстаріших та найвідоміших проектів цифрової бібліотеки, який надає доступ до великої кількості книг у вільному доступі. Заснований у 1971 році Майклом Хартом, цей проект має на меті створення та розповсюдження електронних копій книг, які перебувають у публічному домені.

Основна мета Project Gutenberg – сприяти поширенню знань, надаючи

безкоштовний доступ до літературних творів для всіх бажаючих. Проект орієнтований на збереження класичної літератури та забезпечення її доступності для сучасних та майбутніх поколінь.

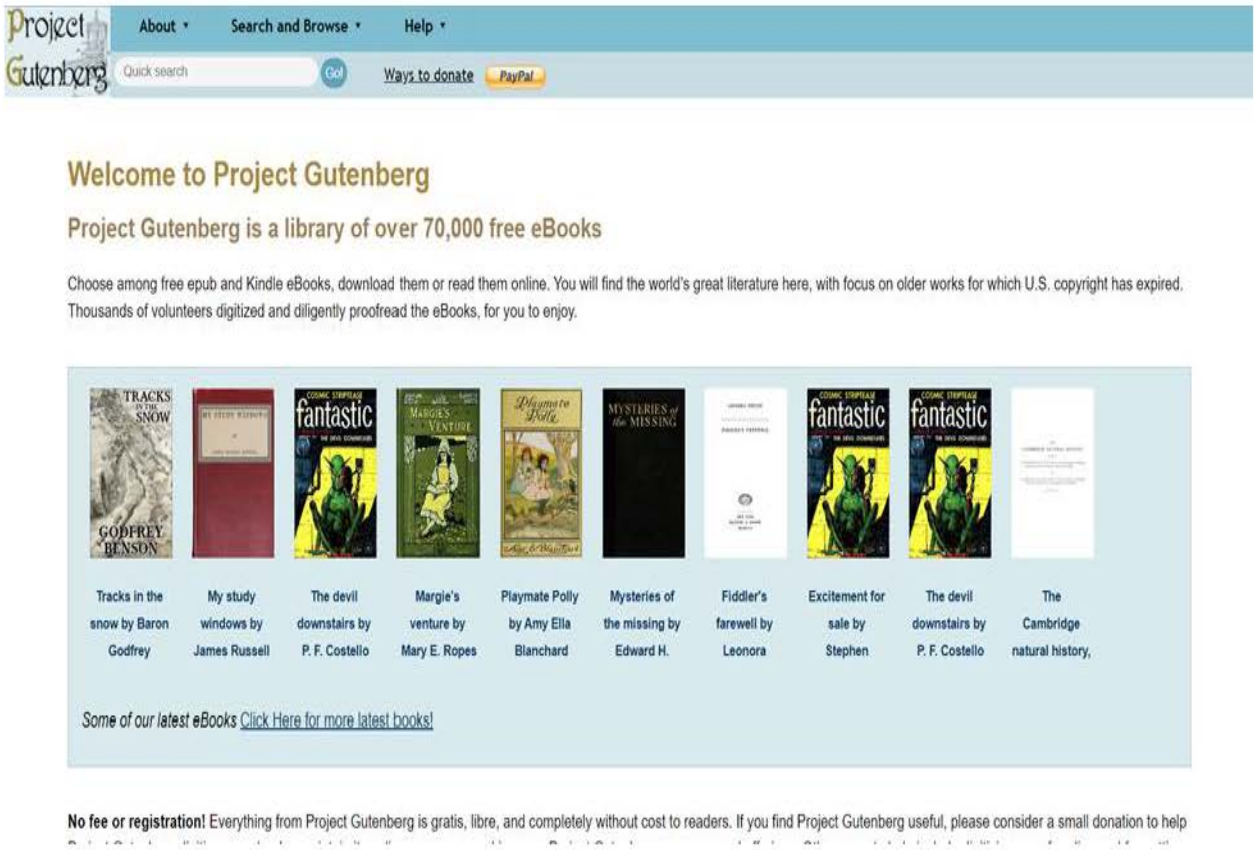

Рисунок 1.1 – Інтерфейс Project Gutenberg

Ключовою особливістю є велика колекція книг. Project Gutenberg пропонує понад 60 000 електронних книг, які включають класичну літературу, наукові роботи, історичні документи та інші важливі тексти. Всі ці книги перебувають у публічному домені, що означає, що їх можна вільно завантажувати, читати і поширювати.

Всі книги на Project Gutenberg доступні для безкоштовного завантаження. Користувачі можуть отримати тексти в різних форматах, включаючи plain text, HTML, ePub, Kindle та інші.

Бібліотека містить книги різними мовами, хоча більшість текстів написані англійською. Однак, колекція постійно поповнюється новими

текстами на різних мовах.

Проект значною мірою покладається на роботу волонтерів, які займаються оцифруванням, редагуванням та перевіркою текстів. Це дозволяє постійно збільшувати колекцію доступних книг.

Усі тексти, доступні на Project Gutenberg, перебувають у публічному домені, тобто вони більше не захищені авторським правом. Це дозволяє користувачам вільно використовувати ці матеріали для будь-яких цілей.

Як користуватися:

1) Пошук книг – користувачі можуть шукати книги за автором, назвою, мовою або за темою через інтерфейс вебсайту.

2) Завантаження – після знаходження потрібної книги, користувачі можуть завантажити її у відповідному форматі, що підтримується їхніми пристроями (наприклад, ePub для електронних рідерів або Kindle для пристроїв Amazon Kindle).

3) Читання онлайн – крім завантаження, багато книг можна читати безпосередньо на вебсайті Project Gutenberg у форматі HTML.

Project Gutenberg є прикладом успішної реалізації електронної бібліотеки, яка забезпечує доступ до великої кількості книг у цифровому форматі. Розглянемо цю систему з точки зору вирішення основних задач електронної бібліотеки:

Основні задачі електронної бібліотеки:

1) Збір та оцифрування контенту

– Project Gutenberg фокусується на книгах, які перебувають у публічному домені. Це дозволяє уникнути проблем з авторськими правами та забезпечує вільний доступ до текстів.

– Процес оцифрування включає сканування друкованих книг, розпізнавання тексту за допомогою OCR (оптичного розпізнавання символів) та перевірку якості тексту волонтерами. Це забезпечує високу якість цифрових копій.

2) Каталогізація та організація контенту

– Всі книги в Project Gutenberg каталогізовані за автором, назвою, темою, мовою та іншими атрибутами. Це полегшує пошук та навігацію по колекції.

– До кожної книги додаються метадані, такі як рік публікації, жанр, мова та короткий опис, що покращує пошукові можливості та організацію бібліотеки.

3) Доступ та розповсюдження

– Всі книги доступні для завантаження безкоштовно. Це робить літературу доступною для широкої аудиторії, включаючи студентів, дослідників та звичайних читачів.

– Книги доступні у різних форматах (plain text, HTML, ePub, Kindle), що дозволяє користувачам обрати найзручніший для їх пристроїв формат.

– Багато книг можна читати безпосередньо на вебсайті, що зручно для користувачів, які не хочуть або не можуть завантажити файли.

4) Пошук та навігація

– Project Gutenberg надає потужні пошукові інструменти, які дозволяють знаходити книги за різними критеріями (автор, назва, тема, мова).

– Користувачі можуть переглядати книги за категоріями, тематичними списками, а також рекомендуваннями та популярними завантаженнями.

5) Збереження та архівування

– Всі книги зберігаються у цифровому форматі, що забезпечує їх тривале збереження та доступність. Використання різних форматів файлів також сприяє сумісності з різними пристроями та системами.

– Проект дбає про резервне копіювання та захист даних, що гарантує збереження контенту у випадку технічних проблем або збоїв.

6) Підтримка користувачів

– Значна частина роботи, включаючи оцифрування та перевірку текстів, виконується волонтерами, що дозволяє постійно оновлювати та розширювати колекцію.

– Інтерфейс Project Gutenberg простий у використанні та інтуїтивно зрозумілий, що полегшує навігацію та доступ до ресурсів.

2) Free-eBooks.net (рис. 1.2)

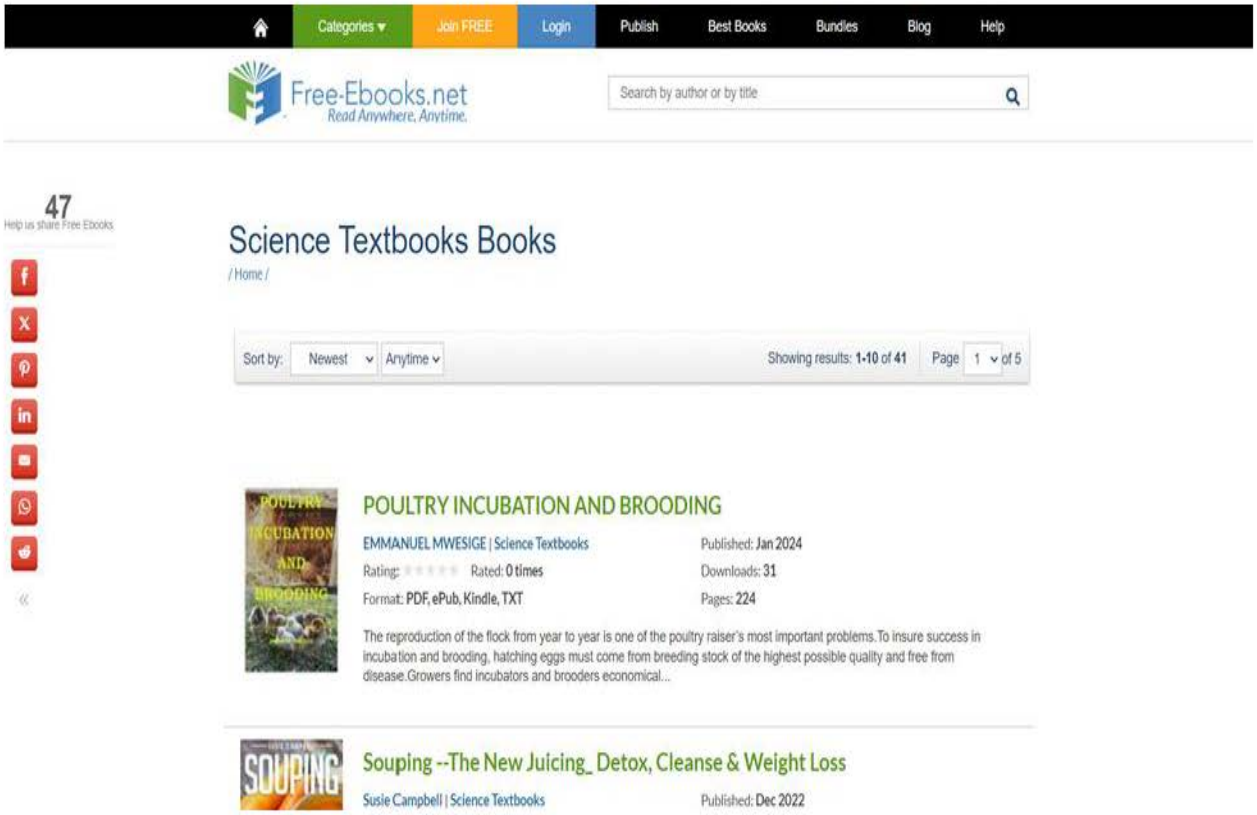

Рисунок 1.2 – Інтерфейс Free-eBooks.net

Free-eBooks.net – це онлайн-платформа, яка надає безкоштовний доступ до широкого асортименту електронних книг у різних жанрах та тематиках. Основна мета Free-eBooks.net полягає в тому, щоб надати користувачам можливість безкоштовно читати та завантажувати електронні книги з Інтернету. Ключові характеристики та особливості Free-eBooks.net:

1) Основна особливість Free-eBooks.net полягає в тому, що всі електронні книги на платформі доступні для читання та завантаження безкоштовно. Користувачам не потрібно платити за доступ до книг або їх використання.

2) Free-eBooks.net пропонує широкий вибір книг у різних жанрах, таких як романи, детективи, фентезі, наука, бізнес, саморозвиток та багато інших. Це дозволяє користувачам знайти книги, які відповідають їхнім інтересам та потребам.

3) Книги на Free-eBooks.net доступні у різних форматах, включаючи PDF, EPUB та MOBI. Це дозволяє користувачам використовувати їх на різних пристроях та платформах.

4) Користувачі можуть залишати відгуки та оцінки для книг, які вони прочитали. Це допомагає іншим користувачам знайти якісні матеріали та визначити, які книги є популярними серед читачів.

5) Книги на Free-eBooks.net організовані за різними категоріями та тегами, що полегшує пошук інтересних матеріалів для користувачів.

6) Користувачі можуть підписатися на розсилку Free-eBooks.net, щоб отримувати сповіщення про нові книги, акції та інші цікаві події.

Free-eBooks.net, як і більшість електронних бібліотек, працює на основі вебсайту або вебпорталу, який надає користувачам доступ до своїх ресурсів через Інтернет. Загальний опис того, як працює це програмне забезпечення:

1) Free-eBooks.net має вебсайт, де користувачі можуть зареєструватися або увійти, щоб мати доступ до всіх доступних функцій та ресурсів.

2) Користувачі можуть створити обліковий запис на Free-eBooks.net, введенням необхідних особистих даних та обранням логіна та пароля.

3) Після входу в систему користувачі можуть використовувати пошуковий інтерфейс для пошуку книг за автором, назвою, жанром або іншими критеріями. Знайдені книги можуть бути переглянуті на вебсайті.

4) Користувачі можуть читати книги прямо на вебсайті за допомогою онлайн-читача або завантажити їх у форматі PDF, EPUB або MOBI для подальшого читання офлайн або на інших пристроях.

5) Користувачі можуть залишати відгуки та оцінки для книг, які вони прочитали, допомагаючи іншим користувачам обирати якісні матеріали.

6) Користувачі можуть управляти своїм обліковим записом, змінюючи особисті дані, паролі та налаштування приватності.

7) Free-eBooks.net може надсилати сповіщення про нові книги, акції та інші події користувачам через електронну пошту або повідомлення на вебсайті.

8) Адміністратори Free-eBooks.net відповідають за управлінням та модерацію контенту на платформі, включаючи додавання нових книг, виправлення помилок, вирішення скарг та інше.

3) ManyBooks (рис. 1.3)

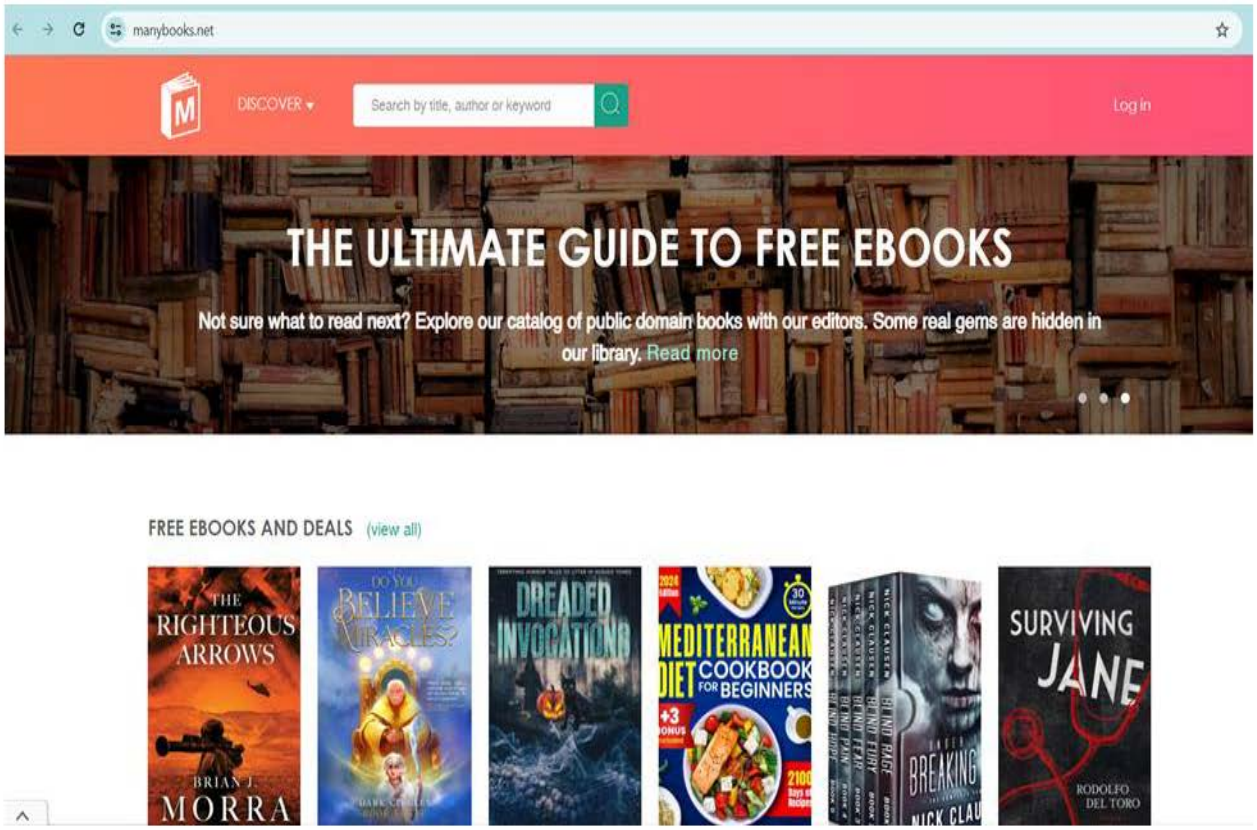

Рисунок 1.3 – Інтерфейс ManyBooks

ManyBooks – це онлайн-бібліотека, яка пропонує великий вибір безкоштовних електронних книг у різних форматах. Ключові функції цього ресурсу:

1) ManyBooks надає доступ до тисяч електронних книг з різних жанрів і категорій. Підтримка різних форматів, таких як ePub, Kindle, PDF, TXT та інші.

2) Зручний пошук і система фільтрації, яка дозволяє користувачам знаходити книги за жанром, автором, популярністю, датою додавання та ін.

3) Інтуїтивний інтерфейс з категоріями та підкатегоріями для легкого перегляду колекції.

4) Можливість створити обліковий запис для отримання додаткових

функцій, таких як збереження обраних книг і персоналізація рекомендацій.

5) Вхід через обліковий запис ManyBooks або використання облікових записів соціальних мереж.

6) Безкоштовне завантаження книг у різних форматах. Користувачі можуть завантажити книги без необхідності реєстрації. Можливість перегляду книги онлайн або завантаження для читання офлайн.

7) Персоналізовані рекомендації на основі історії читання та оцінок користувачів.

8) Користувачі можуть залишати рецензії та оцінки книг, що допомагає іншим у виборі.

9) Сторінки з детальною інформацією про авторів, їхні твори та літературну діяльність. Оновлення та новини про популярних авторів і нові надходження.

9) Вбудований онлайн-читач для читання книг безпосередньо на сайті без необхідності завантаження. Регулювання розміру шрифту, кольору фону та інших параметрів для зручного читання.

10) Платформа для обговорення книг, обміну враженнями та рекомендаціями з іншими користувачами. Можливість слідкувати за улюбленими авторами та іншими читачами.

4) HathiTrust Digital Library (рис. 1.4)

HathiTrust Digital Library – це велика цифрова бібліотека, яка містить мільйони книг, журналів, періодичних видань, документів та інших матеріалів. Загальний опис функціональності, яка доступна на цьому вебсайті:

1) Користувачі можуть використовувати різноманітні критерії для пошуку матеріалів, включаючи назву, автора, рік видання та інші метадані. Після знаходження потрібного матеріалу, користувачі можуть переглядати його онлайн або завантажувати для офлайн-читання.

2) Частина матеріалів в HathiTrust доступна вільно для загального користування, тоді як інші можуть бути доступні лише з обмеженими правами або в рамках угод з інституційними підписниками.

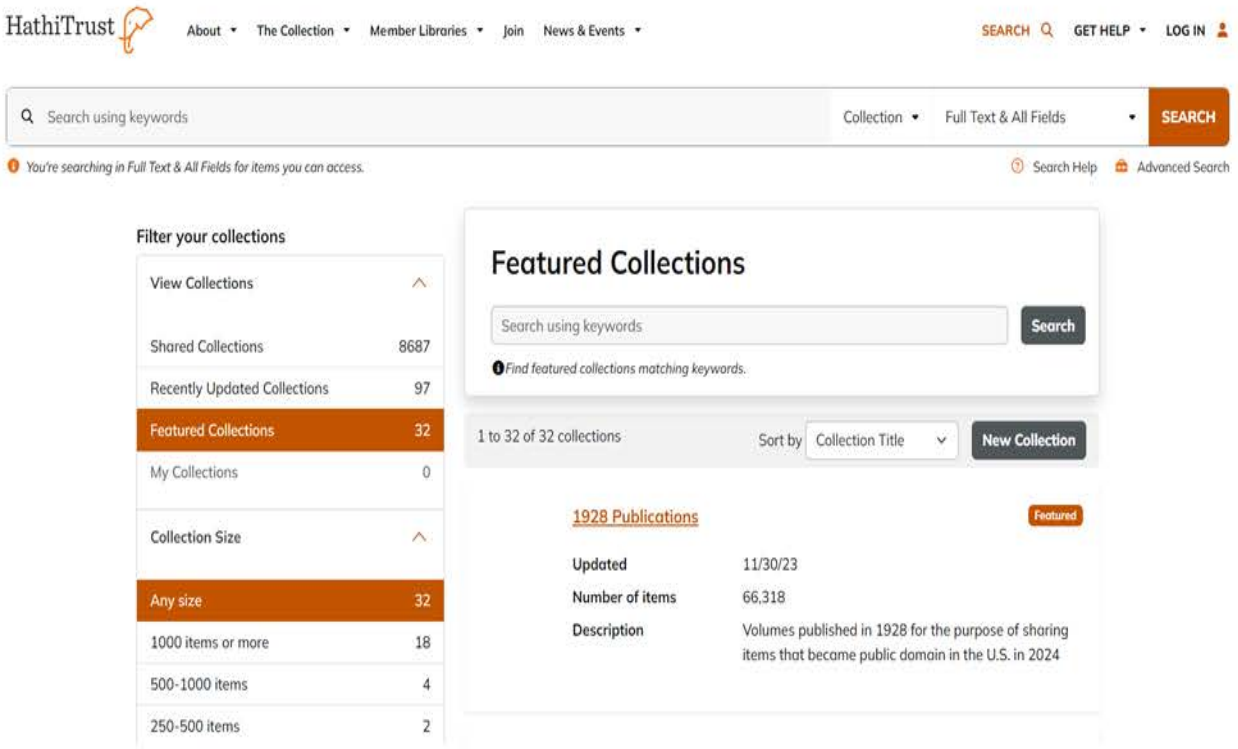

Рисунок 1.4 – Інтерфейс HathiTrust

3) Система пошуку HathiTrust пропонує різноманітні фільтри та параметри для деталізації результатів пошуку, такі як мова, формат файлу, тип матеріалу тощо.

4) HathiTrust має різноманітні колекції та вибірки, які об'єднують матеріали за певними темами, авторами або іншими критеріями.

5) Користувачі можуть отримати доступ до бібліографічних даних про матеріали у HathiTrust для додавання до своїх досліджень, бібліографічних списків тощо.

6) HathiTrust надає підтримку дослідникам та освітнім установам, сприяючи проведенню наукових досліджень, розробці курсів тощо.

7) HathiTrust забезпечує безпеку та конфіденційність даних користувачів, використовуючи сучасні заходи безпеки та шифрування.

Ці функції роблять HathiTrust важливим інструментом для дослідження та навчання, а також для загального доступу до культурного надбання у вигляді електронних ресурсів.

4) Smashwords (рис. 1.5)

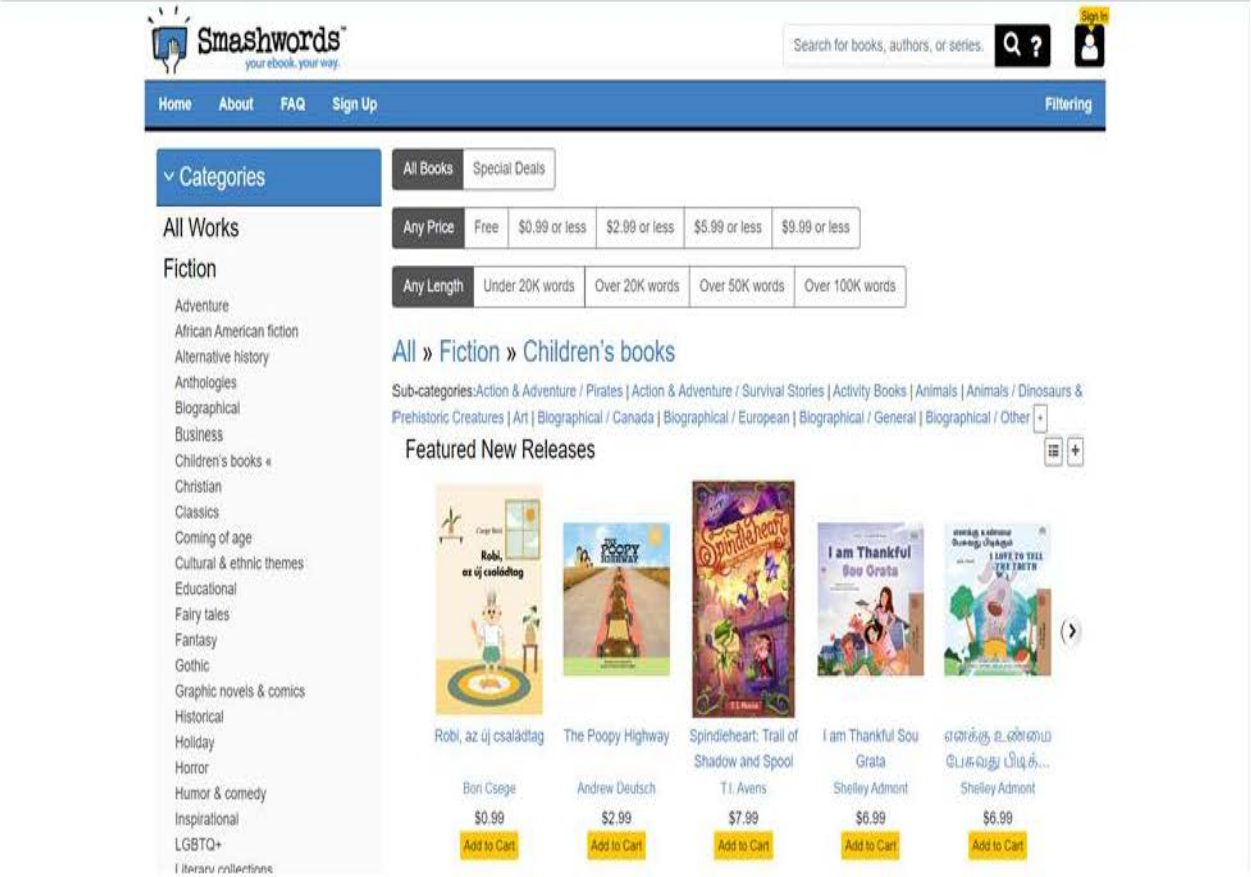

Рисунок 1.5 – Інтерфейс Smashwords

Smashwords – це платформа для самовидавців та незалежних авторів, яка дозволяє публікувати, розповсюджувати та продавати електронні книги. Ключові функції цього ресурсу:

1) Smashwords надає авторам інструменти для завантаження, форматування та публікації електронних книг. Підтримка різних форматів, таких як ePub, mobi, PDF та інші. Докладні інструкції та посібники з публікації, що допомагають авторам підготувати книги до випуску.

2) Smashwords співпрацює з багатьма дистриб'юторами, такими як Apple Books, Barnes & Noble, Kobo та іншими, що забезпечує широку аудиторію. Оптимізація метаданих для поліпшення видимості та продажів у магазинах-партнерах.

3) Інструменти для маркетингу, включаючи можливості встановлення

знижок, купонів та акцій. Автори можуть встановлювати власні ціни на книги або пропонувати їх безкоштовно. Прозора система виплат роялті авторам, що забезпечує справедливі виплати.

4) Зручна система пошуку та навігації по книгах, фільтрація за жанрами, авторами, популярністю та іншими критеріями. Можливість завантаження книг у різних форматах для різних пристроїв. Вбудований онлайн-читач для читання книг безпосередньо на платформі.

5) Авторам надається можливість створити персоналізований профіль з біографією, фотографією та списком публікацій. Інтеграція з соціальними мережами для просування книг та взаємодії з читачами.

6) Інструменти для відстеження продажів, завантажень та інших ключових показників. Регулярні звіти про продажі та роялті, що допомагають авторам аналізувати результати.

7) Розгорнуті посібники та FAQ для допомоги авторам у процесі публікації та маркетингу. Підтримка через електронну пошту та форуми для вирішення проблем та питань.

1.2 Основні функції програмного забезпечення електронної бібліотеки

Програмне забезпечення електронної бібліотеки – це комплекс програмних інструментів та додатків, призначених для створення, управління, зберігання, пошуку та доступу до цифрових бібліотечних ресурсів. Таке програмне забезпечення забезпечує ефективну організацію електронної бібліотеки, підтримку користувачів та управління бібліотечними процесами.

Основні функції програмного забезпечення електронної бібліотеки можна представити наступним чином.

1) Каталогізація та індексація

Каталогізація – організація та класифікація книг, статей, журналів та інших ресурсів за різними критеріями (автор, назва, тема, дата публікації тощо).

Індексація – автоматичне створення індексів для швидкого пошуку та доступу до матеріалів.

2) Зберігання та управління даними

Зберігання – надійне зберігання цифрових копій книг, статей, мультимедійних ресурсів у базах даних.

Управління – адміністрування бібліотечних ресурсів, включаючи контроль за доступом, оновлення контенту та управління правами.

3) Пошук та навігація

Пошукові інструменти – надання потужних інструментів пошуку, таких як повнотекстовий пошук, фасетний пошук та пошук за метаданими.

Навігація – зручна навігація по бібліотечних ресурсах, включаючи категорії, тематичні списки та рекомендації.

4) Доступ та використання

Доступ – забезпечення доступу до ресурсів через вебінтерфейс або мобільні додатки.

Використання – підтримка завантаження, читання онлайн та інших видів використання бібліотечних матеріалів.

5) Аналітика та звітність

Аналітика – відстеження використання ресурсів, аналіз пошукових запитів та активності користувачів.

Звітність – генерація звітів для адміністраторів та керівництва бібліотеки для прийняття рішень щодо розвитку бібліотеки.

6) Підтримка користувачів

Інтерфейс – інтуїтивно зрозумілий інтерфейс для користувачів, що забезпечує зручний доступ до ресурсів.

Особистий кабінет – можливість створення особистих кабінетів для користувачів з функціями збереження пошукових запитів, створення списків обраних книг та налаштування сповіщень про нові надходження.

1.3 Висновки до першого розділу. Постановка завдань дослідження

Метою даної дипломної роботи є створення інформаційної та програмної системи для електронної бібліотеки

Для досягнення поставленої мети необхідно вирішити наступніт задачі.

1) Аналіз існуючих рішень:

– Провести огляд сучасних електронних бібліотек та аналіз їх функціональних можливостей.

– Визначити вимоги до інформаційного та програмного забезпечення електронної бібліотеки на основі потреб користувачів та адміністрації.

2) Проектування бази даних

Розробити структуру бази даних для зберігання інформації про книги, авторів, користувачів, позички та інші дані.

– Створити діаграми ERD (діаграми сутність-зв'язок) для наочності структури бази даних.

– Визначити типи даних, первинні та зовнішні ключі для кожної таблиці.

3) Розробка серверної частини системи:

– Вибрати відповідні технології та мови програмування для реалізації серверної частини.

– Реалізувати API для взаємодії з базою даних, включаючи CRUD операції для книг, авторів, користувачів тощо.

– Забезпечити автентифікацію та авторизацію користувачів з різними рівнями доступу.

4) Розробка клієнтської частини системи:

– Вибрати відповідні технології та фреймворки для розробки інтерфейсу користувача.

– Розробити зручний та інтуїтивний інтерфейс для роботи з бібліотекою, включаючи пошук, перегляд, завантаження книг тощо.

– Забезпечити адаптивний дизайн для підтримки різних пристроїв (ПК, планшети, смартфони).

5) Інтеграція та тестування системи:

– Здійснити інтеграцію клієнтської та серверної частин системи.

– Провести модульне та інтеграційне тестування для перевірки коректності роботи всіх компонентів.

– Забезпечити функціональне та нефункціональне тестування (перформанс, безпека тощо).

У першому розділі було проведено огляд сучасних електронних бібліотек та аналіз їх функціональних можливостей.

Хоча всі ці електронні бібліотеки мають схожу мету – надання доступу до різних видів культурного контенту, вони відрізняються за різними критеріями, такими як доступність матеріалів, обсяг колекцій, функціональність та спеціалізація. Основні відмінності між ними насттиупні.

1) Обсяг та різноманітність матеріалів:

Деякі бібліотеки, такі як Project Gutenberg, спеціалізуються на класичній літературі та текстових документах. Інші, наприклад Google Books, мають широкий спектр матеріалів, включаючи книги, фільми, аудіозаписи, картини тощо. Деякі електронні бібліотеки, такі як National Library of Medicine, спеціалізуються на конкретних областях, таких як медицина та наука.

2) Функціональність та властивості платформи:

Деякі бібліотеки надають можливість читати книги онлайн безпосередньо на їхніх вебсайтах, тоді як інші дозволяють завантажувати матеріали для подальшого використання. Деякі мають інструменти для пошуку та організації знайдених матеріалів, такі як фільтри, теги та рейтинги.

3) Ліцензування та авторські права:

Деякі бібліотеки, такі як Project Gutenberg, спеціалізуються на роботах, які перебувають у суспільному надбанні або мають вільні ліцензії, тому їх матеріали можна використовувати безкоштовно. Інші мають комерційний аспект, пропонуючи доступ до платних книг або фрагментів.

4) Спеціалізація та географічне охоплення:

Деякі бібліотеки мають географічну спеціалізацію та зосереджуються на

культурних ресурсах певних регіонів або країн. Інші можуть мати спеціальні колекції або розділи для певних тематик або груп користувачів.

Ці відмінності дозволяють користувачам вибирати ту електронну бібліотеку, яка найкращим чином відповідає їхнім потребам.

РОЗДІЛ 2 АНАЛІЗ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Вибір програмних засобів для реалізації проекту

Електронні бібліотеки використовують широкий спектр технологій для створення, управління та обслуговування своїх ресурсів. Під час розробки програмного забезпечення можуть бути використані наступні програмні засоби:

1) PHP для створення бекенд-логіки електронної бібліотеки;

2) WAMP Server для спрощення створення та розгортання вебзастосунків на платформі Windows;

3) популярна система управління реляційними базами даних (DBMS) MySQL;

4) HTMLвикористовується для створення структури вебсторінок;

5) CSSCMS полегшує створення, редагування та управління контентом.

2.2 PHP

PHP (Hypertext Preprocessor) – це популярна мова серверного програмування, яка використовується для створення динамічних вебсайтів та вебдодатків. В електронних бібліотеках PHP відіграє важливу роль у забезпеченні функціональності, інтерактивності та динамічної генерації контенту. Використовується для написання серверних скриптів, які виконуються на вебсервері перед тим, як сторінка буде відправлена на клієнтський браузер [1–5].

PHP дозволяє створювати динамічні вебсторінки, які змінюються залежно від дій користувача або інших факторів.

Динамічне генерування HTML-сторінок – це процес створення HTML-коду вебсторінок на сервері в реальному часі, базуючись на даних, отриманих від користувача, з бази даних або з інших джерел. Це забезпечує відображення

актуального та персоналізованого контенту для кожного користувача.

Опис, як працює динамічне генерування HTML-сторінок.

1) Користувач взаємодіє з вебсайтом, відправляючи запит до сервера. Це може бути запит на перегляд сторінки, пошук інформації, заповнення форми тощо (запит користувача).

2) Сервер отримує запит і використовує серверний скрипт (наприклад, на мові PHP) для його обробки. Серверний скрипт визначає, яку інформацію потрібно отримати і як її обробити (обробка запиту сервером).

3) Серверний скрипт часто звертається до бази даних для отримання необхідної інформації. Це може бути інформація про книги, користувачів, категорії тощо. Сервер виконує SQL-запити для отримання відповідних даних (доступ до бази даних).

4) Отримавши необхідні дані, серверний скрипт створює HTML-код. Цей код включає динамічні елементи, які змінюються залежно від даних, отриманих з бази даних або введених користувачем. Наприклад, список книг у бібліотеці може бути створений на основі результатів пошуку або фільтрації (генерація HTML-коду).

5) Сервер відправляє згенерований HTML-код назад до браузера користувача. Браузер рендерить цю сторінку, відображаючи динамічно створений контент (відправка HTML-коду клієнту).

Опис переваг динамічного генерування HTML-сторінок.

– Користувачі отримують свіжу та актуальну інформацію, оскільки сторінки генеруються в реальному часі на основі поточних даних з бази даних (актуальність даних).

– Сторінки можуть бути адаптовані під конкретного користувача, відображаючи персоналізований контент, наприклад, рекомендації книг, історію переглядів тощо (персоналізація).

– Адміністратори можуть легко оновлювати та управляти контентом через базу даних без необхідності змінювати HTML-код вручну (зручне управління контентом).

– Замість зберігання тисяч статичних сторінок, сервер генерує їх на льоту, що зменшує обсяг збережених даних і полегшує адміністрування сайту (ефективність).

Динамічне генерування HTML-сторінок можна використовувати в електронних бібліотеках.

Динамічне генерування дозволяє відображати актуальні списки книг, їх деталі, рейтинги та рецензії (каталог книг).

Користувачі можуть шукати книги за різними критеріями, а результати пошуку генеруються динамічно на основі введених параметрів (пошук та фільтрація).

Динамічне генерування можна використовувати для відображення персоналізованих профілів, історії читання та рекомендацій (профілі користувачів); актуальних новин та інформації про події, що оновлюються у реальному часі

Таким чином, динамічне генерування HTML-сторінок є ключовим елементом сучасних електронних бібліотек, забезпечуючи зручний, персоналізований та актуальний досвід для користувачів.

PHP забезпечує інтеграцію з різними базами даних, такими як MySQL, PostgreSQL, SQLite, що дозволяє зберігати та отримувати дані.

PHP використовується для створення механізмів авторизації та аутентифікації користувачів, зберігання сесій та управління правами доступу; дозволяє обробляти дані, введені користувачами через форми, включаючи валідацію та зберігання цих даних у базі даних.

PHP може взаємодіяти з різними вебслужбами, API та іншими технологіями для забезпечення додаткових функцій.

PHP є важливою складовою для створення та підтримки функціональності електронних бібліотек. Він забезпечує динамічне генерування контенту, інтеграцію з базами даних, управління користувачами, пошук, фільтрацію, безпеку та інтеграцію з іншими сервісами. Завдяки своїй гнучкості та широкому спектру можливостей, PHP залишається одним із

найпопулярніших інструментів для розробки вебдодатків, включаючи електронні бібліотеки.

2.3 WAMP Server

WAMP Server (Windows, Apache, MySQL, PHP) є одним із типів локальних серверів, розроблених для спрощення створення та розгортання вебзастосунків на платформі Windows. Він поєднує в собі різні компоненти, які необхідні для виконання вебзастосунків на мові програмування PHP та роботи з базами даних MySQL [6–10].

Основні компоненти WAMP Server:

1) Windows – операційна система, на якій буде встановлюватися сервер.

2) Apache – вебсервер, який відповідає за обробку та відправку запитів від клієнтів (браузерів) до вебзастосунків. Apache дозволяє розміщувати вебсторінки та взаємодіяти з ними через HTTP-протокол.

3) MySQL – система управління базами даних (СУБД), яка забезпечує можливість створення та управління базами даних, зберігання та оптимізацію даних для вебзастосунків.

4) PHP – мова програмування, яка використовується для написання вебзастосунків. PHP інтерпретується на сервері, що дозволяє відправляти динамічний вміст до клієнтів та взаємодіяти з базами даних.

WAMP Server дозволяє розгортати локальне середовище розробки, де розробники можуть тестувати свої вебзастосунки перед розміщенням їх на живому сервері. Він є зручним інструментом для розробки та тестування вебзастосунків під платформу Windows, оскільки об'єднує усі необхідні компоненти для створення та розгортання вебпроектів в одному пакеті.

2.4 MySQL

MySQL – це відкрита реляційна система управління базами даних

(СУБД), яка зазвичай використовується для зберігання та управління структурованими даними. Вона є однією з найпопулярніших СУБД у світі, широко використовується для веброзробки, а також для створення різноманітних програмних застосунків та систем [11–15].

Опишемо основні характеристики MySQL.

MySQL використовує реляційну модель даних.

Основні концепції реляційної моделі:

1) Дані зберігаються у вигляді таблиць, де кожен рядок представляє окремий запис, а кожний стовпець представляє атрибут (поле) даних (таблиці – реляції).

2) Ключі:

– Унікальний ідентифікатор для кожного запису в таблиці. Він гарантує, що кожен рядок у таблиці має унікальне ідентифікаційне значення (первинний ключ – Primary Key).

– Використовується для встановлення зв'язку між двома таблицями. Він вказує на значення поля в іншій таблиці, що дозволяє створювати відносини між записами цих таблиць (зовнішній ключ Foreign – Key).

3) Процес організації даних у таблицях таким чином, щоб уникнути зберігання дубльованої інформації та забезпечити цілісність даних. Нормалізація допомагає уникнути аномалій при оновленні, вставці та видаленні даних (Нормалізація).

4) Операції:

– CRUD-операції: Створення (Create), Читання (Read), Оновлення (Update), Видалення (Delete). Ці операції дозволяють взаємодіяти з даними в таблицях.

– З'єднання (Join): Операція, що дозволяє комбінувати дані з двох або більше таблиць на основі спільних полів.

Мова SQL (Structured Query Language): Використовується для взаємодії з базами даних за допомогою стандартних запитів. SQL надає можливість створення, читання, оновлення та видалення даних в реляційних таблицях.

Реляційна модель є однією з найпоширеніших та ефективних моделей для організації даних у базах даних. Вона дозволяє ефективно зберігати, керувати та отримувати доступ до даних, що робить її дуже популярною для використання в різних сферах індустрії та програмування.

Для взаємодії з базою даних MySQL використовується мова структурованих запитів SQL (Structured Query Language). SQL дозволяє виконувати різноманітні операції, такі як створення, читання, оновлення та видалення даних.

MySQL підтримує транзакції, що дозволяє виконувати групу операцій як атомарну одиницю роботи, що виконується повністю або анульована; забезпечує можливість масштабування, що дозволяє працювати з великими обсягами даних та високими навантаженнями; надає багато різноманітних функцій та можливостей, таких як індексація даних для прискорення пошуку, підтримка зовнішніх ключів для забезпечення цілісності даних, реплікація для забезпечення високої доступності та бекапи для забезпечення безпеки даних.

MySQL є відкритим програмним забезпеченням, що означає, що вона безкоштовна для використання та має велику активну спільноту користувачів і розробників, яка надає підтримку та розвиває СУБД.

MySQL може бути використаний в електронній бібліотеці для зберігання, управління та доступу до різних типів даних.

2.5 HTML

HTML (HyperText Markup Language) – це мова розмітки, яка використовується для створення та структурування вмісту вебсторінок. Основне призначення HTML можна описати наступним чином [16–20].

1) використовується для організації та структурування контенту вебсторінок за допомогою різних елементів.

2) дозволяє форматувати текст, роблячи його більш читабельним та естетично привабливим.

3) дозволяє створювати гіперпосилання, які зв'язують різні вебсторінки або ресурси.

4) підтримує вбудовування зображень, відео та інших мультимедійних елементів.

5) дозволяє створювати інтерактивні форми для збору даних від користувачів.

6) включає семантичні елементи, які надають додаткову інформацію про структуру та зміст документа.

7) дозволяє включати метадані, які описують документ та його властивості.

HTML є основною мовою розмітки для створення вебсторінок. Він забезпечує структурування контенту, форматування тексту, створення посилань, вбудовування мультимедіа, створення інтерактивних форм, семантичну розмітку та включення метаданих. Використання HTML є фундаментальним кроком у розробці будь-якого вебдодатка або сайту, оскільки він визначає, як контент буде відображатися та взаємодіяти з користувачами.

2.6 CSS

CSS (Cascading Style Sheets) – це мова, яка використовується для стилізації вебсторінок, надання їм зовнішнього вигляду та форматування. Основне призначення CSS – розділення представлення вебсторінки (так звана «візуальна частина») від її змісту (структури та даних), що дозволяє забезпечити більш гнучке та ефективне управління виглядом вебсторінок [21–25].

Основні призначення CSS:

1) Стилізація елементів – дозволяє задавати різноманітні стилі для HTML-елементів, такі як колір тексту, фон, розміри, шрифти, відступи, рамки та інші властивості, що впливають на зовнішній вигляд.

2) Модульність та реюзабельність – дозволяє визначати стилі в окремих файлах, які можна потім використовувати на всій вебсторінці або навіть на кількох сторінках, що робить код більш модульним та легким для управління.

3) Адаптивний та відзивчивий дизайн – дозволяє створювати стилі, які змінюються в залежності від розміру та орієнтації пристрою, що дозволяє створювати адаптивні та відзивчиві вебсайти, які добре виглядають на різних пристроях і розмірах екранів.

4) Каскадність та спадкування стилів – має механізм каскаду, що дозволяє визначати пріоритетність стилів, які можуть бути застосовані до одного елемента, а також механізм спадкування, що дозволяє елементам унаслідувати стилі від їхніх батьківських елементів.

5) Анімація та переходи – дозволяє створювати анімації та переходи, які змінюють стилі елементів з плинними ефектами, що додає динамічності та привабливості вебсторінкам.

Загалом, CSS є важливою складовою веброзробки, що дозволяє створювати привабливі, модерні та функціональні вебсайти, що відповідають потребам користувачів та сучасним стандартам дизайну.

2.7 Висновки до другого розділу

Електронні бібліотеки використовують широкий спектр технологій для створення, управління та обслуговування своїх ресурсів. Нижче наведено опис основних технологій та їх призначення.

1) HTML (HyperText Markup Language)

Призначення: HTML використовується для створення структури вебсторінок електронної бібліотеки. Він забезпечує розмітку тексту, зображень, відео та інших елементів, що складають вебсторінки.

Застосування: HTML визначає заголовки, параграфи, списки, посилання, форми для пошуку та реєстрації, а також інтеграцію мультимедійних елементів, таких як зображення обкладинок книг і відео-

презентації.

2) CSS (Cascading Style Sheets)

Призначення: CSS використовується для оформлення та стилізації HTML-елементів на вебсторінках.

Застосування: В CSS задаються кольори, шрифти, відступи, межі, вирівнювання, анімації та інші візуальні аспекти, що роблять інтерфейс електронної бібліотеки привабливим та зручним для користувачів.

3) PHP (Hypertext Preprocessor)

Призначення: PHP – це серверна мова програмування, яка використовується для обробки даних та динамічного генерування HTML-сторінок.

Застосування: PHP використовується для створення бекенд-логіки електронної бібліотеки, включаючи обробку запитів користувачів, взаємодію з базою даних, управління сесіями користувачів та обробку форм.

4) SQL (Structured Query Language)

Призначення: SQL – це мова запитів, яка використовується для взаємодії з реляційними базами даних.

Застосування: SQL використовується для зберігання, запиту, оновлення та видалення даних в базах даних електронної бібліотеки. Це включає управління інформацією про книги, авторів, користувачів, замовлення та інші метадані.

5) MySQL

Призначення: MySQL – це популярна системи управління реляційними базами даних (DBMS).

Застосування: Вона використовується для зберігання та управління даними електронної бібліотеки. Ці системи забезпечують надійне зберігання, високу продуктивність та масштабованість.

Електронні бібліотеки використовують широкий спектр технологій для забезпечення надійного зберігання, швидкого доступу та ефективного управління бібліотечними ресурсами. HTML та CSS створюють інтерфейс

користувача; PHP та інші серверні мови обробляють логіку бекенду; SQL та MySQL бази даних забезпечують зберігання та управління даними. Використання цих технологій дозволяє створити потужну, масштабовану та ефективну електронну бібліотеку.

РОЗДІЛ 3 РОЗРОБКА ІНФОРМАЦІЙНОГО ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЕЛЕКТРОННОЇ БІБЛІОТЕКИ

3.1 Опис бази даних

Назва бази даних – Library.

База даних призначена для управління електронною бібліотекою, включаючи інформацію про книги, авторів, користувачів та їхні взаємодії з бібліотекою.

У зв'язку з цим можна виділити наступні сутності:

1) «Автори» – зберігається інформація про авторів книг;
2) «Книги» – зберігається інформація про книги;
3) «Жанр» – зберігається інформація про жанр книги;
4) «Користувачі» – зберігається інформація про користувачів;
5) «Ролі» – зберігається інформація про ролі користувачів;
6) «Автори книг» – зберігається інформація відношення книг та авторів;
7) «Бронювання книг» – зберігається інформація про ролі заброньовані книги;
8) «Міста» – зберігається інформація про міста видавництв;
9) «Видавництва» – зберігається інформація про видавництва.

Опис структури бази даних включає основні таблиці, їхні поля та зв'язки між ними (таблиці 3.1-3.8).

Таблиця 3.1

Жанри книг (genre)

Найменування атрибутів		Тип поля	Розмір поля	Обмеження	Зв'язок
Код жанру	id	int	11	Not null	PK
Назва жанру	name	varchar	150	Not null	-

Таблиця 3.2

Міста видавництв (city)

Найменування атрибутів		Тип поля	Розмір поля	Обмеження	Зв’язок
Код міста	id	int	11	Not null	PK
Назва	name	varchar	150	Not null	-

Таблиця 3.3

Автори (author)

Найменування атрибутів		Тип поля	Розмір поля	Обмеження	Зв’язок
Код автору	id	int	11	Not null	PK
Ім’я	name	varchar	150	Not null	-
Прізвище	surname	varchar	150	Not null	-
По батькові	middle_name	varchar	150	Null	-

Таблиця 3.4

Видавництва (publisher)

Найменування атрибутів		Тип поля	Розмір поля	Обмеження	Зв’язок
Код видавництва	id	int	11	Not null	PK
Найменування	name	varchar	150	Not null	-
Код міста	city_id	int	11	Not null	FK

Таблиця 3.5

Автори книг (book_author)

Найменування атрибутів		Тип поля	Розмір поля	Обмеження	Зв’язок
Код запису	id	int	11	Not null	PK
Шифр книги	book_id	int	11	Not null	FK
Код автору	author_id	int	11	Not null	FK

Таблиця 3.6

Користувачі (customer)

Найменування атрибутів		Тип поля	Розмір поля	Обмеження	Зв'язок
Код користувача	id	int	11	Not null	PK
Ім'я	name	varchar	100	Not null	-
Прізвище	surname	varchar	100	Not null	-
По батькові	middle_name	varchar	100	Null	-
Телефон	phone	varchar	20	Not null	-
Адреса	address	varchar	150	Null	-

Таблиця 3.7

Книги (book)

Найменування атрибутів		Тип поля	Розмір поля	Обмеження	Зв'язок
Шифр книги	id	int	11	Not null	PK
Назва	name	varchar	150	Not null	-
Код жанру	genre_id	int	11	Not null	FK
Код видавництва	publisher_id	int	11	Not null	FK
Рік видавництва	year	int	11	Null	-
Кількість сторінок	pages	int	11	Null	-
Ціна книги	price	decimal	10,2	Not null	-
Кількість екземплярів	amount	int	11	Not null	-

Таблиця 3.8

Бронювання книг (book_reservation)

Найменування атрибутів		Тип поля	Розмір поля	Обмеження	Зв'язок
Код броні	id	int	11	Not null	PK

Шифр книги	book_id	int	11	Not null	FK
Код користувача	customer_id	int	11	Not null	FK
Дата замовлення	order_date	date	10	Not null	-

Для кожного поля в таблицях бази даних вказуються назва поля, його розмір, тип даних і, якщо є, зв'язок з іншими таблицями. Для первинних ключів встановлюється обмеження на відсутність невизначених значень.

3.2 Діаграма бази даних

Діаграма бази даних (ER-діаграма, Entity-Relationship Diagram) – це візуальне представлення структури бази даних, яке показує, як різні сутності (об'єкти або таблиці) в базі даних взаємопов'язані. Діаграми баз даних використовуються для проектування, документування та аналізу баз даних. Вони допомагають розробникам, адміністраторам баз даних і іншим зацікавленим сторонам зрозуміти, як дані організовані і як вони взаємодіють.

Основні Елементи діаграми бази даних можна описати наступним чином.

1) Сутності (Entities):

– сутності представляють об'єкти або концепції, що зберігаються в базі даних;

– у діаграмі сутності зображуються у вигляді прямокутників.

Приклади: Книга, Автор, Користувач.

2) Атрибути (Attributes):

– атрибути представляють характеристики або властивості сутностей;

– вони зображуються у вигляді овалів, прикріплених до сутностей, або як частина прямокутника в ERD (Enhanced Entity-Relationship Diagram).

Приклади: Назва, Рік публікації, Ім'я автора.

3) Зв'язки (Relationships):

– зв'язки показують, як сутності взаємопов'язані одна з одною;

– вони зображуються у вигляді ромбів або ліній, що з'єднують сутності. Приклади: Має, Належить, Позичає.

4) Ключі (Keys):

– первинний Ключ (Primary Key): Унікальний ідентифікатор сутності. Зазвичай позначається підкресленням або жирним шрифтом;

– зовнішній Ключ (Foreign Key): Поле в одній таблиці, яке посилається на первинний ключ іншої таблиці. Використовується для встановлення зв'язків між таблицями.

Згідно структури з табл. 3.1-3.8 можна побудувати діаграму бази даних, яка буде мати вигляд наступний вигляд (рис. 3.1).

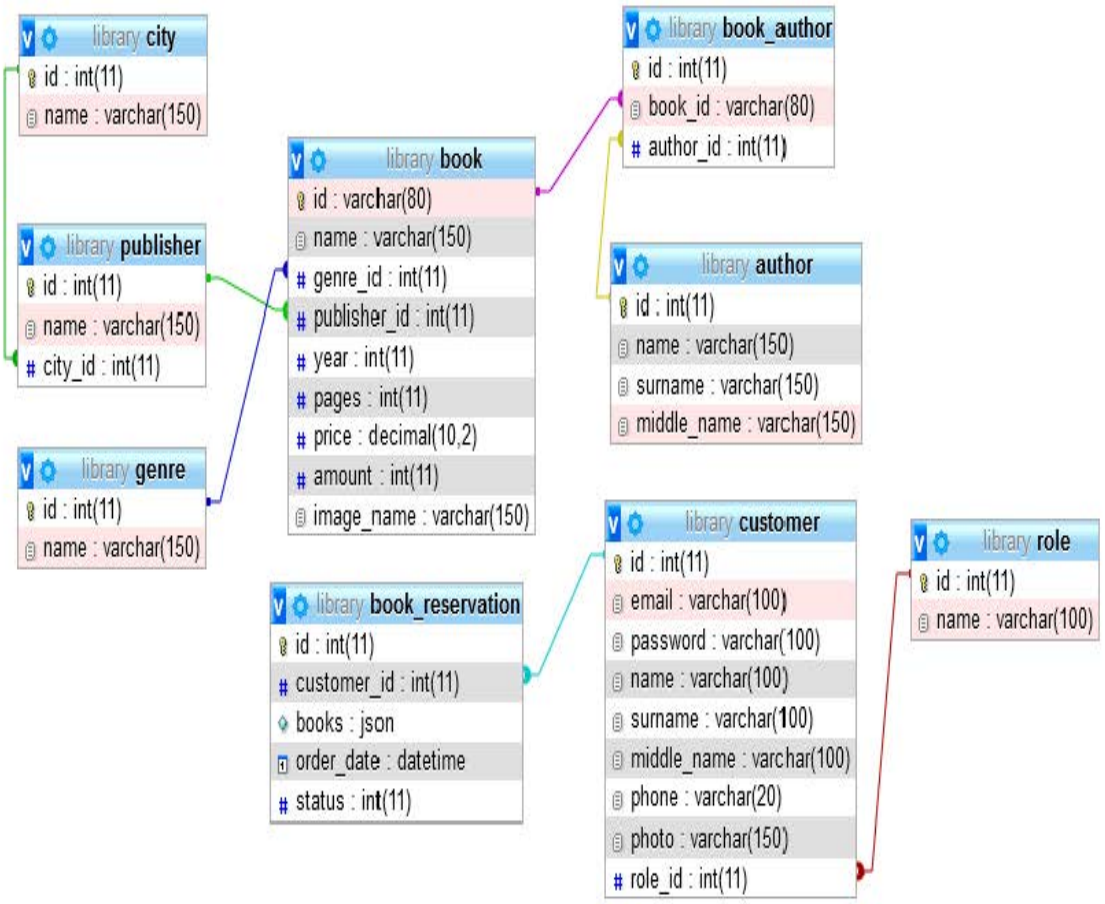

Рисунок 3.1 – Діаграма бази даних Library

3.3 Ролі користувачів

Даний динамічний web-засіб представляє три види користувацьких профілів, кожен з яких має свої права та обов'язки:

1) Адміністратор

Адміністратор є користувачем з найвищим рівнем доступу до системи. Цей профіль призначений для управління та налаштування всіх аспектів web-засобу.

Функції:

– управління обліковими записами користувачів (створення, редагування, видалення);

– модерація контенту (перегляд, редагування, видалення матеріалів, завантажених користувачами);

– налаштування системних параметрів і конфігурацій;

– створення та управління категоріями контенту;

– моніторинг активності користувачів і безпеки системи;

– управління резервними копіями бази даних і відновленням даних.

2) Зареєстрований користувач

Опис: Зареєстрований користувач має середній рівень доступу до системи і може виконувати більше дій порівняно з гостем, але менше, ніж адміністратор. Цей профіль призначений для активних учасників системи, які регулярно використовують її можливості.

Функції:

– перегляд і пошук контенту в базі даних;

– завантаження та завантаження книг і інших матеріалів (з можливим обмеженням за кількістю;

– бронювання книг;

– оцінка та рецензування контенту;

– управління своїм профілем (оновлення особистих даних, зміна пароля);

– доступ до історії своєї активності та позичок;

– взаємодія з іншими користувачами через коментарі та обговорення.

3) Гість

Опис: Гість має найнижчий рівень доступу і не потребує створення облікового запису для використання базових функцій системи. Цей профіль призначений для нових або тимчасових користувачів, які хочуть ознайомитися з можливостями системи.

Функції:

– перегляд обмеженого контенту в базі даних (без доступу до завантаження);

– пошук книг та іншого матеріалу;

– перегляд загальної інформації про книги (опис, рецензії, рейтинги);

– реєстрація нового облікового запису для отримання доступу до додаткових функцій;

– можливість перегляду коментарів і обговорень без можливості додавання власних.

Взаємодія ролей

– адміністратор взаємодіє з усіма користувачами для забезпечення належного функціонування системи та дотримання правил;

– зареєстровані користувачі можуть взаємодіяти один з одним через коментарі та обговорення, а також з адміністратором у разі потреби допомоги або вирішення питань;

– гості мають обмежений доступ і можуть стати зареєстрованими користувачами для отримання більшого функціоналу.

Ця структура ролей забезпечує гнучкість і безпеку системи, дозволяючи різним категоріям користувачів отримувати доступ до відповідного рівня функціональності.

Незареєстровані користувачі (гості) мають можливість лише переглядати головну сторінку, здійснювати пошук на ній (рис. 3.2) та переглядати додаткову інформацію, натиснувши на вибрану книгу (рис. 3.3).

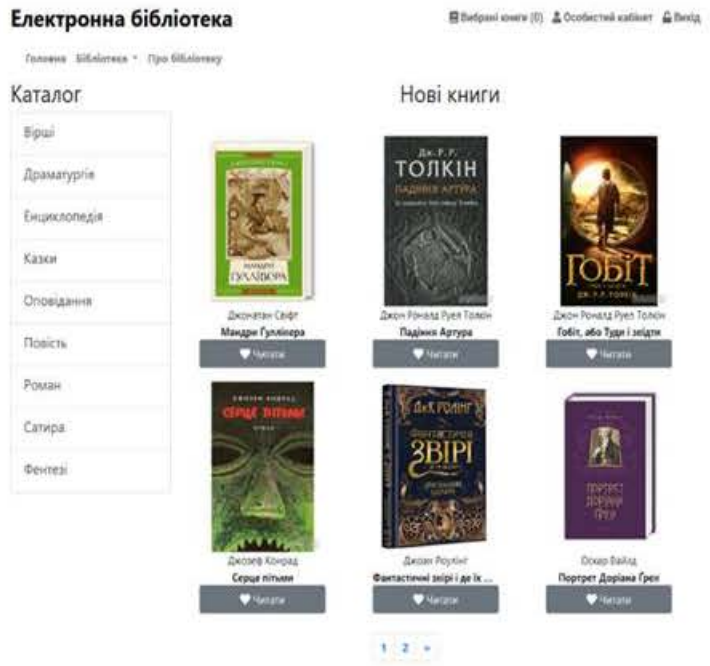

Рисунок 3.2 – Стартова сторінка

Рисунок 3.3 – Інформація про обрану книгу

3.4 Реєстрація та авторизація користувачів

Після завантаження головної сторінки сайту гість може зареєструватися. Для цього необхідно користувач повинен ввести свої персональні дані у спеціальну форму (рис.3.4). Після реєстрації користувачу призначається роль у системі – «user».

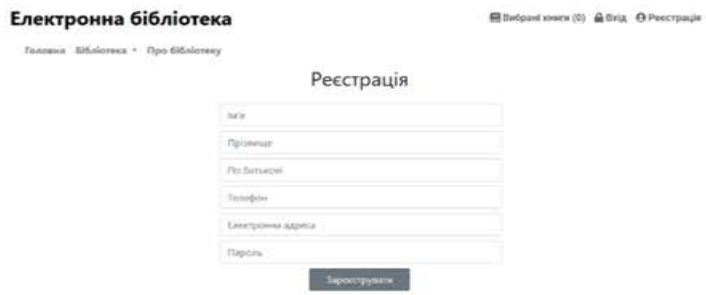

Рисунок 3.4 – Форма реєстрації користувача

Після того, як користувач пройшов реєстрацію, він має можливість авторизуватися (рис. 3.5).

Рисунок 3.5 – Форма авторизації користувача

3.5 Користування контентом

Після того, як користувач пройшов авторизацію, він потрапляє на стартову сторінку, де має можливість перейти до особистого кабінету (рис. 3.6).

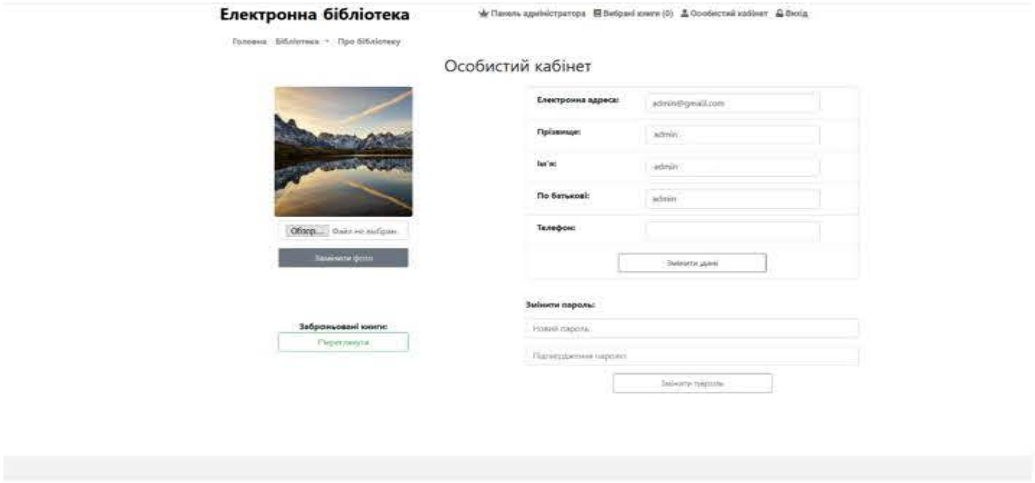

Рисунок 3.6 – Особистий кабінет користувача

Після натискання на посилання «Забронювані книги» користувач може переглянути список книг, що він забронював (рис. 3.7). Ця інформація представлена у вигляді таблиці: код книги, назва книги, автор та видавництво.

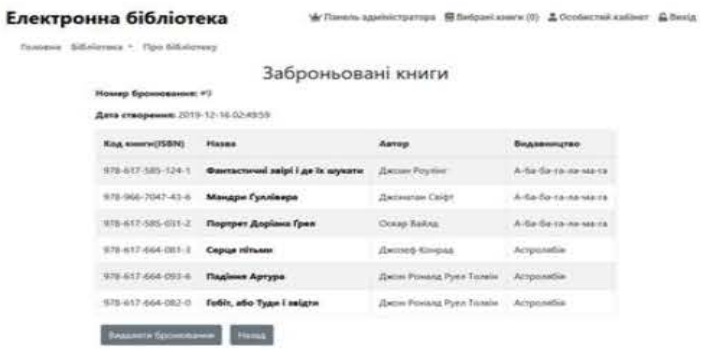

Рисунок 3.7 – Забронювані книги

3.6 Додавання, перегляд і редагування книг та жанрів

Щоб виконувати дії з додавання, редагування, видалення або перегляду книг та жанрів, потрібно авторизуватися як адміністратор. На сторінках рис. 3.8–3.10 продемонстровано можливості додавання, редагування та перегляду книг.

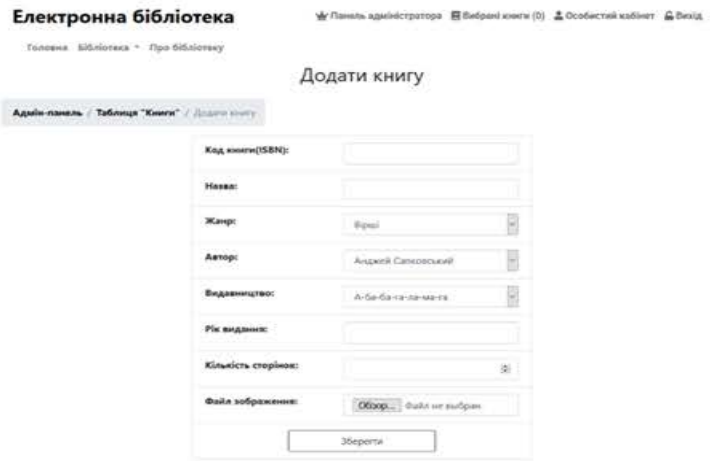

Рисунок 3.8 – Форма додання книг

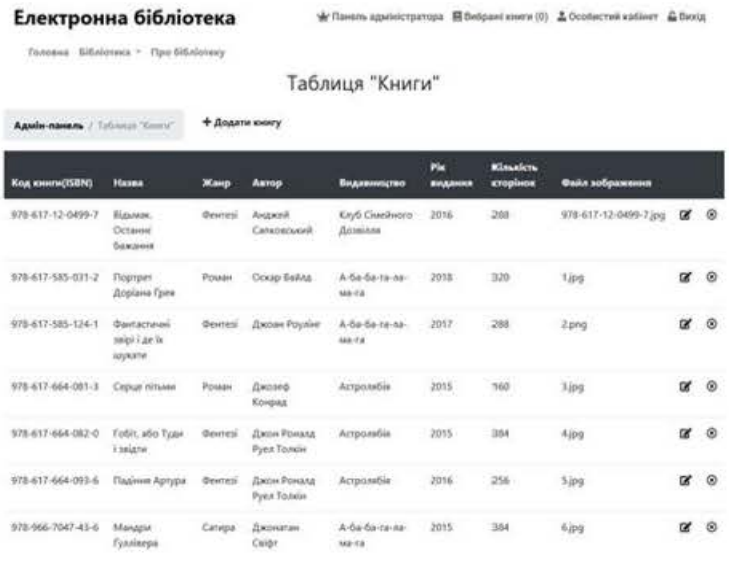

Рисунок 3.9 – Форма перегляду книг

Після авторизації з правами адміністратора на адмінпанелі з'являється можливість редагувати обкладинку книжки.

3.7 Бронювання книг

Користувачі мають можливість бронювати книг, після того як обрали потрібну книгу та натиснули кнопку «Читати» на сторінці каталогу. Після чого книга додається у кошик (рис. 3.10).

Також користувачам доступна опція резервації книг на сторінці кошика, де вони можуть натиснути відповідну кнопку.

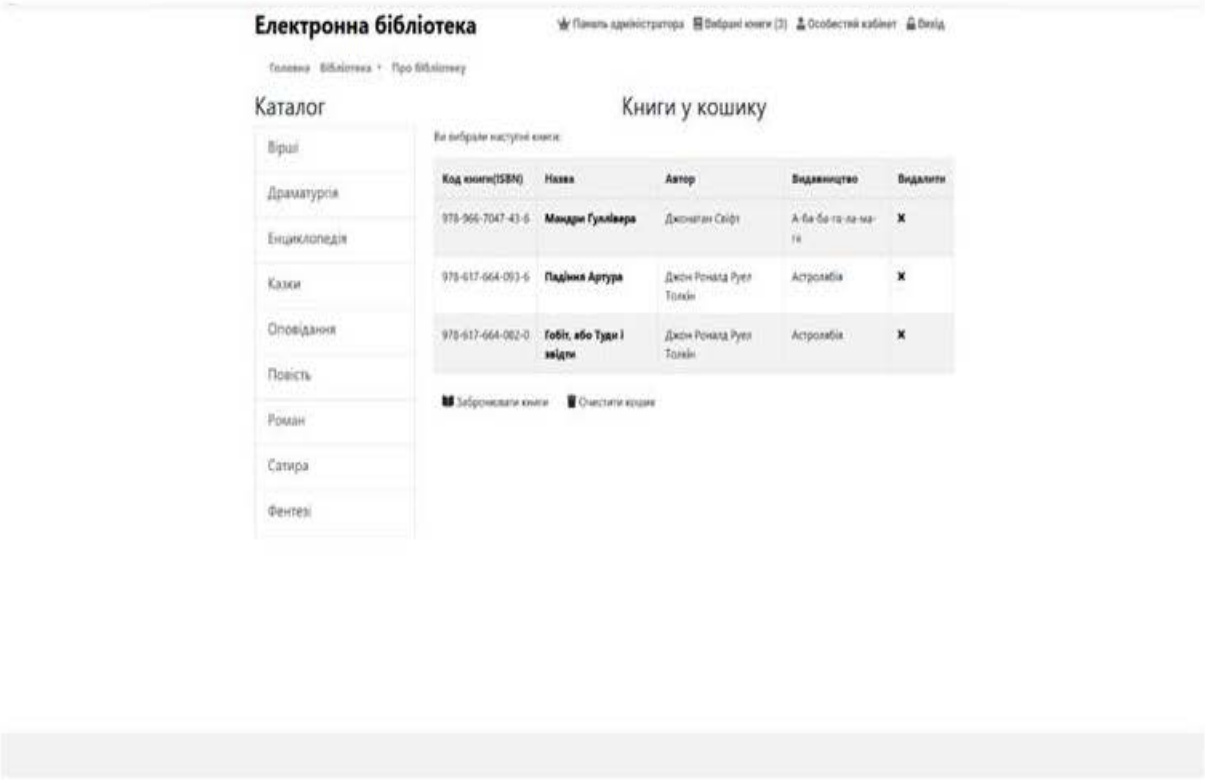

Рисунок 3.10 – Сторінка перегляду кошика

3.8 Варіативність у функціях користувачів

Різноманітність у користуванні контентом включає в себе те, що адміністратор має повні права на всі існуючі записи, включаючи можливість

додавання та редагування. Зареєстрований користувач має право на бронювання книг, тоді як незареєстрований користувач обмежений лише переглядом інформації про книги та категорії. Якщо користувач спробує забронювати книги без авторизації, він отримає повідомлення про помилку (рис. 3.11).

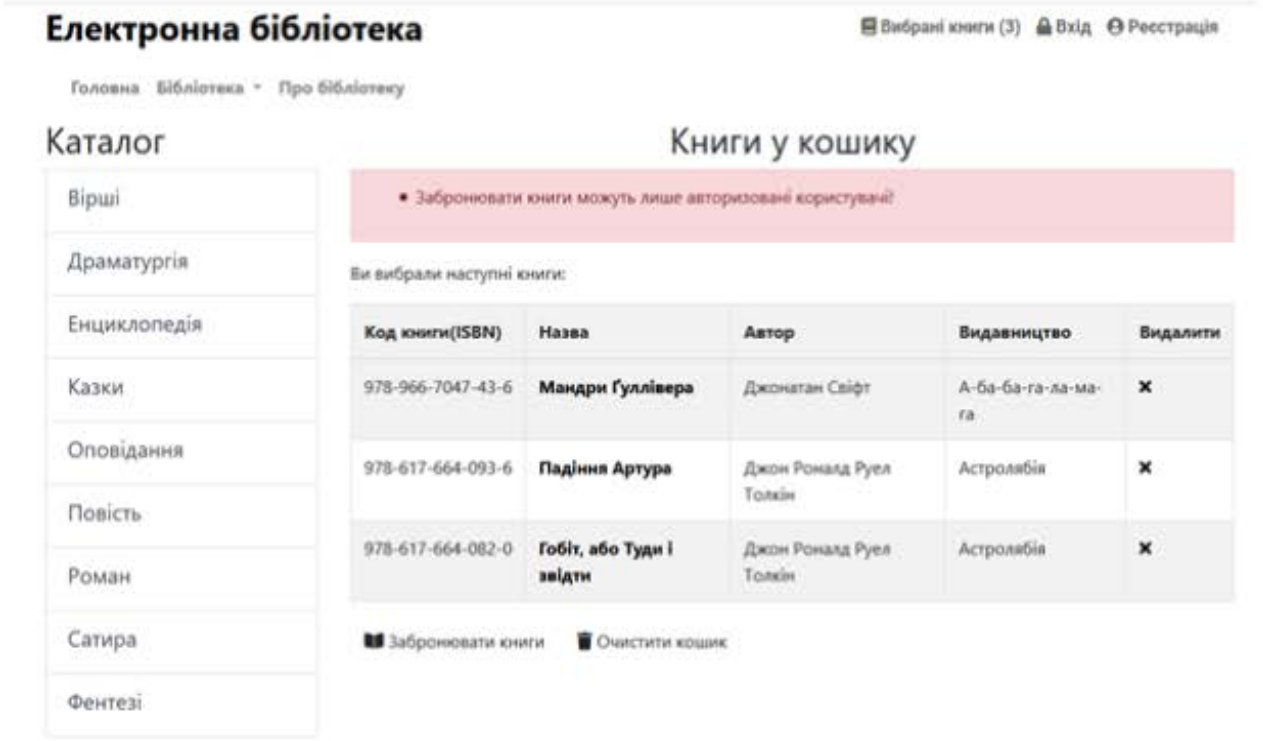

Рисунок 3.11 – Відображення помилки

3.9 Висновки до третього розділу

У розділі «Розробка інформаційного та програмного забезпечення електронної бібліотеки» було вивчено та розглянуто ключові аспекти розробки програмного забезпечення для електронної бібліотеки. Це включає детальний опис бази даних, яка складається з декількох важливих складових. Діаграма бази даних відображає структуру та зв'язки між різними сутностями системи, включаючи ролі користувачів, процес реєстрації та авторизації користувачів, а також функції користувачів у взаємодії з контентом. Зокрема,

увага приділяється можливостям додавання, перегляду та редагування книг та жанрів, а також процесу бронювання книг. Зазначена варіативність у функціях користувачів підкреслює гнучкість системи та можливість налаштування прав доступу відповідно до різних ролей користувачів. В цілому, цей розділ надає глибокий інсайт у процес створення програмного забезпечення для електронної бібліотеки, включаючи його функціональні можливості та структурну організацію бази даних.

В розділі «Розробка інформаційного та програмного забезпечення електронної бібліотеки» було детально проаналізовано не лише базу даних, а й інші ключові аспекти системи. У цьому розділі також описано архітектуру програмного забезпечення, використані технології та інструменти розробки. Деякі з інших аспектів, що розглядаються, включають в себе процес розробки інтерфейсу користувача, методи забезпечення безпеки даних, а також тестування та валідацію програмного забезпечення. Окрім цього, розглянуто аспекти взаємодії з користувачем, такі як процес реєстрації та авторизації, а також інструкції щодо користування функціоналом системи.

Крім того, розглянуто аспекти взаємодії з користувачем, такі як процес реєстрації та авторизації, а також детально описано інструкції щодо користування різноманітним функціоналом системи. Особлива увага була приділена створенню зручного та інтуїтивно зрозумілого інтерфейсу для користувачів, з метою забезпечення зручності в їхній взаємодії з програмою. Висвітлено також питання валідації даних під час процесу реєстрації та входу в систему, щоб забезпечити безпеку та надійність обробки інформації користувачами.

Висновок до даного розділу відображає загальний огляд роботи над розробкою програмного забезпечення для електронної бібліотеки, включаючи інформацію про базу даних та інші важливі компоненти системи.

ВИСНОВКИ

Метою даної дипломної роботи є створення інформаційної та програмної системи для електронної бібліотеки, яка забезпечує:

1) Зручний та швидкий доступ користувачів до колекцій електронних книг.

2) Можливість ефективного управління бібліотечними фондами для адміністраторів.

3) Надійне зберігання даних та їх захист від несанкціонованого доступу.

Для досягнення поставленої мети було вирішено наступніт задачі.

1) Аналіз існуючих рішень:

– Проведено огляд сучасних електронних бібліотек та аналіз їх функціональних можливостей.

– Визначено вимоги до інформаційного та програмного забезпечення електронної бібліотеки на основі потреб користувачів та адміністрації.

2) Проектування бази даних

Розроблено структуру бази даних для зберігання інформації про книги, авторів, користувачів, позички та інші дані.

– Створено діаграми ERD (діаграми сутність-зв'язок) для наочності структури бази даних.

– Визначено типи даних, первинні та зовнішні ключі для кожної таблиці.

3) Розроблено серверну частину системи:

– Вибрано відповідні технології та мови програмування для реалізації серверної частини.

– Реалізувано API для взаємодії з базою даних, включаючи CRUD операції для книг, авторів, користувачів тощо.

– Забезпечено автентифікацію та авторизацію користувачів з різними рівнями доступу.

4) Розроблено клієнтську частину системи:

– Вибрано відповідні технології та фреймворки для розробки інтерфейсу

користувача.

– Розроблено зручний та інтуїтивний інтерфейс для роботи з бібліотекою, включаючи пошук, перегляд, завантаження книг тощо.

– Забезпечено адаптивний дизайн для підтримки різних пристроїв (ПК, планшети, смартфони).

5) Інтеграція та тестування системи:

– Здійснено інтеграцію клієнтської та серверної частин системи.

– Проведено модульне та інтеграційне тестування для перевірки коректності роботи всіх компонентів.

– Забезпечено функціональне та нефункціональне тестування (перформанс, безпека тощо).

В розділі «Розробка інформаційного та програмного забезпечення електронної бібліотеки» було детально проаналізовано не лише базу даних, а й інші ключові аспекти системи. У цьому розділі також описано архітектуру програмного забезпечення, використані технології та інструменти розробки. Деякі з інших аспектів, що розглядаються, включають в себе процес розробки інтерфейсу користувача, методи забезпечення безпеки даних, а також тестування та валідацію програмного забезпечення. Окрім цього, розглянуто аспекти взаємодії з користувачем, такі як процес реєстрації та авторизації, а також інструкції щодо користування функціоналом системи.

Крім того, розглянуто аспекти взаємодії з користувачем, такі як процес реєстрації та авторизації, а також детально описано інструкції щодо користування різноманітним функціоналом системи. Особлива увага була приділена створенню зручного та інтуїтивно зрозумілого інтерфейсу для користувачів, з метою забезпечення зручності в їхній взаємодії з програмою. Висвітлено також питання валідації даних під час процесу реєстрації та входу в систему, щоб забезпечити безпеку та надійність обробки інформації користувачами.

У роботі були вирішені наступні завдання:

– Проведено дослідження для визначення потреб та вимог цільової

аудиторії електронної бібліотеки.

– Розробдено детальний план структури та функціоналу програмного забезпечення для електронної бібліотеки. Визначено основні функції, можливості пошуку, систему авторизації та інші ключові аспекти системи.

– Розроблено програмне забезпечення на основі розробленого проекту, було використано відповідні технології та інструменти розробки, щоб забезпечити ефективну роботу системи та її зручний інтерфейс для користувачів.

– Проведено тестування програмного забезпечення для перевірки його працездатності, безпеки та відповідності вимогам.

– Підготовлено документацію, включаючи технічне описання системи, інструкції з використання та адміністрування, а також звіт про роботу над дипломною роботою.

Об'єкт дослідження – це електронна бібліотека як цілісна система. Предмет дослідження – це процес розробки та впровадження інформаційного та програмного забезпечення для цієї бібліотеки.

Розробка інформаційного та програмного забезпечення для електронної бібліотеки є актуальною та важливою в сучасному світі. Електронні бібліотеки стають невід'ємною складовою цифрової трансформації, забезпечуючи глобальний доступ до інформації, підвищення зручності користування та підвищення ефективності освіти та науки.

Кваліфікаційна робота виконана у відповідності до стандарту спеціальності 121 – «Інженерія програмного забезпечення» і демонструє володіння такими компетентностями як:

- Здатність ідентифікувати, класифікувати та формулювати вимоги до програмного забезпечення;

- Здатність брати участь у проектуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його структури, поведінки та процесів функціонування;

- Здатність розробляти архітектури, модулі та компоненти програмних

систем;

- Здатність формулювати та забезпечувати вимоги щодо якості програмного забезпечення у відповідності з вимогами замовника, технічним завданням та стандартами;

- Здатність дотримуватися специфікацій, стандартів, правил і рекомендацій в професійній галузі при реалізації процесів життєвого циклу;

- Володіння знаннями про інформаційні моделі даних, здатність створювати програмне забезпечення для зберігання, видобування та опрацювання даних;

- Здатність застосовувати фундаментальні і міждисциплінарні знання для успішного розв'язання завдань інженерії програмного забезпечення;

- Здатність обґрунтовано обирати та освоювати інструментарій з розробки та супроводження програмного забезпечення;

- Здатність до алгоритмічного та логічного мислення тощо.

Серед результатів навчання, визначених стандартом, кваліфікаційна робота реалізує наступні:

- Аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки;

- Вміти розробляти людино-машинний інтерфейс;

- Знати та вміти використовувати методи та засоби збору, формулювання та аналізу вимог до програмного забезпечення;

- Проводити передпроектне обстеження предметної області, системний аналіз об'єкта проектування;

- Вибирати вихідні дані для проектування, керуючись формальними методами опису вимог та моделювання;

- Застосовувати на практиці ефективні підходи щодо проектування програмного забезпечення;

- Знати і застосовувати методи розробки алгоритмів, конструювання програмного забезпечення та структур даних;

- Мотивовано обирати мови програмування та технології розробки для розв’язання завдань створення і супроводження програмного забезпечення;

- Знати та вміти застосовувати інформаційні технології обробки, зберігання та передачі даних;

- Вміти документувати та презентувати результати розробки програмного забезпечення тощо.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Jon Duckett. PHP & MySQL: Server-side Web Development, 2022.

2. Robin Nixon. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5, 6th Edition, 2021.

3. Larry Ullman. PHP for the Web: Visual QuickStart Guide, 5th Edition, 2019.

4. Josh Lockhart. Modern PHP: New Features and Good Practices, 2nd Edition, 2020.

5. M. Younsi, M. Kalin. PHP 8 Objects, Patterns, and Practice, 6th Edition, 2021.

6. Jyotiswarup Raiturkar. Mastering WampServer, 2019.

7. Michael Ross. Local Development with WampServer, 2020.

8. Sunil Gulabani. Getting Started with WampServer, 2021.

9. Sushant Roy. Efficient Development with WampServer, 2022.

10. Mark Redman. WampServer Essentials, 2023.

11. Karthik Appigatla. MySQL 8.0 Cookbook: Over 150 recipes for high-performance database solutions, 2nd Edition, 2022.

12. Vinicius M. Grippa, Sergey Kuzmichev. Learning MySQL: Get a Handle on Your Data, 2021.

13. Chintan Mehta, Ankit Patel, Ivan Osuna. MySQL 8 Administrator's Guide, 2019.

14. Renee M. P. Teate. SQL for Data Scientists: A Beginner's Guide for Building Datasets for Analysis, 2021.

15. Jesper Wisborg Krogh. MySQL 8 Query Performance Tuning, 2021.

16. Jon Duckett. HTML and CSS: Design and Build Websites, 2019.

17. Andy Harris. HTML5 and CSS3 All-in-One For Dummies, 4th Edition, 2020.

18. Jennifer Robbins. Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics, 5th Edition, 2018.

19. Mark Pilgrim. HTML5: Up and Running, 2nd Edition, 2019.

20. Rob Crowther, Joe Lennon, Ash Blue. HTML5 in Action, 2021.

21. Eric A. Meyer, Estelle Weyl. CSS: The Definitive Guide, 4th Edition, 2017.

22. Lea Verou. CSS Secrets: Better Solutions to Everyday Web Design Problems, 2015.

23. Keith J. Grant. CSS in Depth, 2018.

24. Ben Frain. Responsive Web Design with HTML5 and CSS, 4th Edition, 2022.

25. David Sawyer McFarland. CSS: The Missing Manual, 4th Edition, 2019.

ДОДАТОК А

Лістинг створення БД

```
CREATE DATABASE `library`;
DROP TABLE IF EXISTS `author`;
CREATE TABLE IF NOT EXISTS `author` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(150) NOT NULL,
  `surname` varchar(150) NOT NULL,
  `middle_name` varchar(150) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8;
INSERT INTO `author` (`id`, `name`, `surname`, `middle_name`) VALUES
(1, 'Джоан', 'Роулінг', NULL),
(2, 'Джонатан', 'Свіфт', NULL),
(3, 'Оскар', 'Вайлд', NULL),
(4, 'Джозеф', 'Конрад', NULL),
(5, 'Джон', 'Роналд Руел Толкін', NULL),
(6, 'Анджей', 'Сапковський', NULL);
DROP TABLE IF EXISTS `book`;
CREATE TABLE IF NOT EXISTS `book` (
  `id` varchar(80) NOT NULL,
  `name` varchar(150) NOT NULL,
  `genre_id` int(11) NOT NULL,
  `publisher_id` int(11) NOT NULL,
  `year` int(11) DEFAULT NULL,
  `pages` int(11) DEFAULT NULL,
  `price` decimal(10,2) NOT NULL DEFAULT '0.00',
  `amount` int(11) NOT NULL DEFAULT '1',
  `image_name` varchar(150) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL DEFAULT 'no-
image.png',
  PRIMARY KEY (`id`),
  KEY `publisher_id` (`publisher_id`),
  KEY `genre_id` (`genre_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
INSERT INTO `book` (`id`, `name`, `genre_id`, `publisher_id`, `year`, `pages`, `price`, `amount`, `image_name`)
VALUES
('978-617-12-0499-7', 'Відьмак. Останнє бажання', 9, 5, 2016, 288, '0.00', 1, '978-617-12-0499-7.jpg'),
('978-617-585-031-2', 'Портрет Доріана Ґрея', 5, 1, 2018, 320, '110.00', 43, '1.jpg'),
('978-617-585-124-1', 'Фантастичні звірі і де їх шукати', 9, 1, 2017, 288, '186.00', 10, '2.png'),
('978-617-664-081-3', 'Серце пітьми', 5, 2, 2015, 160, '116.00', 8, '3.jpg'),
('978-617-664-082-0', 'Гобіт, або Туди і звідти\r\n', 9, 2, 2015, 384, '174.00', 63, '4.jpg'),
('978-617-664-093-6', 'Падіння Артура', 9, 2, 2016, 256, '198.00', 11, '5.jpg'),
('978-966-7047-43-6', 'Мандри Ґуллівера', 10, 1, 2015, 384, '130.00', 24, '6.jpg');
DROP TABLE IF EXISTS `book_author`;
CREATE TABLE IF NOT EXISTS `book_author` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `book_id` varchar(80) NOT NULL,
  `author_id` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `author_id` (`author_id`),
  KEY `book_id` (`book_id`)
) ENGINE=InnoDB AUTO_INCREMENT=40 DEFAULT CHARSET=utf8;
INSERT INTO `book_author` (`id`, `book_id`, `author_id`) VALUES
(2, '978-966-7047-43-6', 2),
(3, '978-617-585-031-2', 3),
(4, '978-617-585-124-1', 1),
(5, '978-617-664-081-3', 4),
(6, '978-617-664-093-6', 5),
(7, '978-617-664-082-0', 5),
(33, '978-617-12-0499-7', 6);
```

```
DROP TABLE IF EXISTS `book_reservation`;
CREATE TABLE IF NOT EXISTS `book_reservation` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `customer_id` int(11) NOT NULL,
  `books` json NOT NULL,
  `order_date` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  `status` int(11) NOT NULL DEFAULT '1',
  PRIMARY KEY (`id`),
  KEY `customer_id` (`customer_id`)
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8;

INSERT INTO `book_reservation` (`id`, `customer_id`, `books`, `order_date`, `status`) VALUES
(1, 5, 'null', '2019-12-09 20:41:32', 1);

DROP TABLE IF EXISTS `city`;
CREATE TABLE IF NOT EXISTS `city` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(150) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=25 DEFAULT CHARSET=utf8;
INSERT INTO `city` (`id`, `name`) VALUES
(1, 'Київ'),
(2, 'Львів'),
(3, 'Харків'),
(4, 'Вінниця'),
(5, 'Дніпро'),
(6, 'Житомир'),
(7, 'Запоріжжя'),
(8, 'Івано-Франківськ'),
(9, 'Кропивницький'),
(10, 'Луганськ'),
(11, 'Луцьк'),
(12, 'Миколаїв'),
(13, 'Одеса'),
(14, 'Полтава'),
(15, 'Рівне'),
(16, 'Суми'),
(17, 'Тернопіль'),
(18, 'Ужгород'),
(19, 'Донецьк'),
(20, 'Херсон'),
(21, 'Хмельницький'),
(22, 'Черкаси'),
(23, 'Чернівці'),
(24, 'Чернігів');
DROP TABLE IF EXISTS `customer`;
CREATE TABLE IF NOT EXISTS `customer` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `email` varchar(100) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL,
  `password` varchar(100) NOT NULL,
  `name` varchar(100) NOT NULL,
  `surname` varchar(100) NOT NULL,
  `middle_name` varchar(100) DEFAULT NULL,
  `phone` varchar(20) NOT NULL,
  `photo` varchar(150) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL DEFAULT
'default_avatar.png',
  `role_id` int(11) NOT NULL DEFAULT '2',
  PRIMARY KEY (`id`),
  KEY `role_id` (`role_id`)
) ENGINE=InnoDB AUTO_INCREMENT=14 DEFAULT CHARSET=utf8;
INSERT INTO `customer` (`id`, `email`, `password`, `name`, `surname`, `middle_name`, `phone`, `photo`, `role_id`)
VALUES
```

```
(1, 'user@gmail.com', '827ccb0eea8a706c4c34a16891f84e7b', 'Петро', 'Петренко', 'Петрович', '380676767676',
'default_avatar.png', 2),
(5, 'admin@gmail.com', 'b59c67bf196a4758191e42f76670ceba', 'admin', 'admin', 'admin', '380000000000',
'user5_avatar.jpg', 1),
(8, 'test@gmail.com', '827ccb0eea8a706c4c34a16891f84e7b', 'Іван', 'Іванченко', 'Іванович', '380999999999',
'user8_avatar.jpg', 2),
(12, 'volod@gmail.com', 'b59c67bf196a4758191e42f76670ceba', 'qqqqq', 'qqqqq', 'qqqqq', '380666666666',
'default_avatar.png', 2),
(13, 'volod1@gmail.com', 'b59c67bf196a4758191e42f76670ceba', 'qqqqqq', 'wwwww', 'fdsfddsf', '0505050505',
'user13_avatar.jpg', 2);
DROP TABLE IF EXISTS `genre`;
CREATE TABLE IF NOT EXISTS `genre` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(150) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=14 DEFAULT CHARSET=utf8;
INSERT INTO `genre` (`id`, `name`) VALUES
(1, 'Казки'),
(2, 'Вірші'),
(3, 'Оповідання'),
(4, 'Повість'),
(5, 'Роман'),
(6, 'Драматургія'),
(7, 'Енциклопедія'),
(9, 'Фентезі'),
(10, 'Сатира');
DROP TABLE IF EXISTS `publisher`;
CREATE TABLE IF NOT EXISTS `publisher` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(150) DEFAULT NULL,
  `city_id` int(11) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `city_id` (`city_id`)
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8;
INSERT INTO `publisher` (`id`, `name`, `city_id`) VALUES
(1, 'А-ба-ба-га-ла-ма-га', 1),
(2, 'Астролябія', 2),
(3, 'Веселка', 1),
(4, 'Зелений Пес', 1),
(5, 'Клуб Сімейного Дозвілля', 3);
DROP TABLE IF EXISTS `role`;
CREATE TABLE IF NOT EXISTS `role` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(100) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;
INSERT INTO `role` (`id`, `name`) VALUES
(1, 'admin'),
(2, 'user');
ALTER TABLE `book`
  ADD CONSTRAINT `book_ibfk_1` FOREIGN KEY (`publisher_id`) REFERENCES `publisher` (`id`) ON
DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `book_ibfk_2` FOREIGN KEY (`genre_id`) REFERENCES `genre` (`id`) ON DELETE
CASCADE ON UPDATE CASCADE;
ALTER TABLE `book_author`
  ADD CONSTRAINT `book_author_ibfk_2` FOREIGN KEY (`author_id`) REFERENCES `author` (`id`) ON
DELETE CASCADE ON UPDATE CASCADE,
  ADD CONSTRAINT `book_author_ibfk_3` FOREIGN KEY (`book_id`) REFERENCES `book` (`id`) ON DELETE
CASCADE ON UPDATE CASCADE;
ALTER TABLE `book_reservation`
  ADD CONSTRAINT `book_reservation_ibfk_2` FOREIGN KEY (`customer_id`) REFERENCES `customer` (`id`)
ON DELETE CASCADE ON UPDATE CASCADE;
```

```
ALTER TABLE `customer`
  ADD CONSTRAINT `customer_ibfk_1` FOREIGN KEY (`role_id`) REFERENCES `role` (`id`) ON DELETE
CASCADE ON UPDATE CASCADE;
ALTER TABLE `publisher`
  ADD CONSTRAINT `publisher_ibfk_1` FOREIGN KEY (`city_id`) REFERENCES `city` (`id`) ON DELETE
CASCADE ON UPDATE CASCADE;
COMMIT;
```

ДОДАТОК Б

Лістинг створення вебсторінок

Лістинг файлу AdminBase.php

```
<?php
abstract class AdminBase
{
   public static function checkAdmin()
   {
      $userId = User::checkLogged();
      $user = User::getUserById($userId);
      if (User::isAdmin()) {
         return true;
      }
      die('Доступ заборонено!');
   }
}
```

Лістинг файлу Autoload.php

```
<?php
spl_autoload_register(function ($class_name)
{
   $array_paths = array(
      '/models/',
      '/components/',
      '/controllers/',
   );
   foreach ($array_paths as $path) {
      $path = ROOT . $path . $class_name . '.php';
      if (is_file($path)) {
         include_once $path;
      }
   }
});
```

Лістинг файлу Cart.php

```
<?php
class Cart
{
   public static function addBook($id)
   {
      $cartItems = array();
      if (isset($_SESSION['books'])) {
         $cartItems = $_SESSION['books'];
      }

      if (array_key_exists($id, $cartItems)) {
         unset($cartItems[$id]);
      } else {
         $cartItems[$id] = 1;
      }

      $_SESSION['books'] = $cartItems;
      return self::itemsCount();
   }

   public static function itemsCount()
   {
      if (isset($_SESSION['books'])) {
```

```
            $count = 0;
            foreach ($_SESSION['books'] as $id => $quantity) {
                $count = $count + $quantity;
            }
            return $count;
        } else {
            return 0;
        }
    }

    public static function getBooks()
    {
        if (isset($_SESSION['books'])) {
            return $_SESSION['books'];
        }
        return false;
    }

    public static function clear()
    {
        if (isset($_SESSION['books'])) {
            unset($_SESSION['books']);
        }
    }

    public static function deleteBook($id)
    {
        $cartItems = self::getBooks();

        unset($cartItems[$id]);
        $_SESSION['books'] = $cartItems;
    }
}
```

Лістинг файлу Db.php

```
<?php
class Db
{
    public static function getConnection()
    {
        $paramsPath = ROOT . '/config/db_params.php';
        $params = include($paramsPath);
        $dsn = "mysql:host={$params['host']};dbname={$params['dbname']}";
        $db = new PDO($dsn, $params['user'], $params['password']);
        $db->exec("set names utf8");
        return $db;
    }
}
```

Лістинг файлу Pagination.php

```
<?php
class Pagination
{
    private $max = 10; //nav link amount
    private $index = 'page'; //url key
    private $current_page;
    private $total;  //records count
    private $limit;  //items limit
```

```
public function __construct($total, $currentPage, $limit, $index)
{
   $this->total = $total;
   $this->limit = $limit;
   $this->index = $index;
   $this->amount = $this->amount();
   $this->setCurrentPage($currentPage);
}

public function get()
{
  $html="";
  if ($this->total > $this->limit) {
    $links = null;
    $limits = $this->limits();

    $html = '<nav aria-label="page-navigation"><ul class="pagination">';
    for ($page = $limits[0]; $page <= $limits[1]; $page++) {
        if ($page == $this->current_page) {
            $links .= '<li class="page-item"><a class="page-link" href="#">'. $page .'</a></li>';
        } else {
            $links .= $this->generateHtml($page);
        }
    }
    if (!is_null($links)) {
        if ($this->current_page > 1)
            $links = $this->generateHtml(1, '&laquo;') . $links;
        if ($this->current_page < $this->amount)
            $links .= $this->generateHtml($this->amount, '&raquo;');
    }
    $html .= $links . '</ul></nav>';
  }
   return $html;
}

private function generateHtml($page, $text = null)
{
   if (!$text)
      $text = $page;
   $currentURI = rtrim($_SERVER['REQUEST_URI'], '/') . '/';
   $currentURI = preg_replace('~/page-[0-9]+~', '', $currentURI);
   return '<li><a class="page-link" href="'. $currentURI . $this->index . $page . '">'. $text . '</a></li>';
}

private function limits()
{
   $left = $this->current_page - round($this->max / 2);
   $start = $left > 0 ? $left : 1;
   if ($start + $this->max <= $this->amount) {
      $end = $start > 1 ? $start + $this->max : $this->max;
   } else {
      $end = $this->amount;
      $start = $this->amount - $this->max > 0 ? $this->amount - $this->max : 1;
   }
   return array($start, $end);
}

private function setCurrentPage($currentPage)
{
   $this->current_page = $currentPage;
   if ($this->current_page > 0) {
      if ($this->current_page > $this->amount)
```

```
            $this->current_page = $this->amount;
        } else
            $this->current_page = 1;
    }

    private function amount()
    {
        return ceil($this->total / $this->limit);
    }
}
```

Лістинг файлу Router.php

```
<?php
class Router
{
    private $routes;
    public function __construct()
    {
        $routesPath = ROOT . '/config/routes.php';
        $this->routes = include($routesPath);
    }

    private function getURI()
    {
        if (!empty($_SERVER['REQUEST_URI'])) {
            return trim($_SERVER['REQUEST_URI'], '/');
        }
    }

    public function run()
    {
        $uri = $this->getURI();
        foreach ($this->routes as $uriPattern => $path) {
            if (preg_match("~$uriPattern~", $uri)) {
                $internalRoute = preg_replace("~$uriPattern~", $path, $uri);
                $segments = explode('/', $internalRoute);
                $controllerName = array_shift($segments) . 'Controller';
                $controllerName = ucfirst($controllerName);
                $actionName = 'action' . ucfirst(array_shift($segments));
                $parameters = $segments;
                $controllerFile = ROOT . '/controllers/' .
                        $controllerName . '.php';

                if (file_exists($controllerFile)) {
                    include_once($controllerFile);
                }
                $controllerObject = new $controllerName;
                $result = call_user_func_array(array($controllerObject, $actionName), $parameters);
                if ($result != null) {
                    break;
                }
            }
        }
    }
}
```

Лістинг файлу db_params.php

```
<?php
return array(
    'host' => 'localhost:3308',
    'dbname' => 'library',
```

```
    'user' => 'prestige2',
    'password' => '12345',
);
```

Лістинг файлу routes.php

```
<?php

return array(

    'page-([0-9]+)' => 'site/index/$1',
    'book/([0-9]+)' => 'book/view/$1',

    'catalog' => 'catalog/index',

    'category/([0-9]+)/page-([0-9]+)' => 'site/category/$1/$2',
    'category/([0-9]+)' => 'site/category/$1',

    'cart/checkout' => 'cart/checkout',
    'cart/clear' =>'cart/clear',
    'cart/delete/([0-9]+)' => 'cart/delete/$1',
    'cart/add/([0-9]+)' => 'cart/add/$1',
    'cart/addAjax/([0-9]+)' => 'cart/addAjax/$1',
    'cart' => 'cart/index',

    'user/register' => 'user/register',
    'user/login' => 'user/login',
    'user/logout' => 'user/logout',
    'cabinet/order' => 'cabinet/bookOrder',
    'cabinet/delete/order' => 'cabinet/delete',
    'cabinet/book/list' => 'cabinet/bookList',
    'cabinet/change/photo' => 'cabinet/change',
    'cabinet/change/password' => 'cabinet/passchange',
    'cabinet/edit' => 'cabinet/edit',
    'cabinet' => 'cabinet/index',

    'admin/book/create' => 'adminBook/create',
    'admin/book/update/([0-9]+)' => 'adminBook/update/$1',
    'admin/book/delete/([0-9]+)' => 'adminBook/delete/$1',
    'admin/book' => 'adminBook/index',

    'admin/publisher/create' => 'adminPublisher/create',
    'admin/publisher/update/([0-9]+)' => 'adminPublisher/update/$1',
    'admin/publisher/delete/([0-9]+)' => 'adminPublisher/delete/$1',
    'admin/publisher' => 'adminPublisher/index',

    'admin/genre/create' => 'adminGenre/create',
    'admin/genre/update/([0-9]+)' => 'adminGenre/update/$1',
    'admin/genre/delete/([0-9]+)' => 'adminGenre/delete/$1',
    'admin/genre' => 'adminGenre/index',

    'admin/reservation/update/([0-9]+)' => 'adminReserve/update/$1',
    'admin/reservation/delete/([0-9]+)' => 'adminReserve/delete/$1',
    'admin/reservation/view/([0-9]+)' => 'adminReserve/view/$1',
    'admin/reservation' => 'adminReserve/index',
    // Админпанель:
    'admin' => 'admin/index',

    'contacts' => 'site/contact',
    'about' => 'site/about',
    // Главная страница
    'index.php' => 'site/index',
    '' => 'site/index',
```

```
);
```

Лістинг файлу AdminBookController.php

```
<?php

class AdminBookController extends AdminBase
{
    public function actionIndex()
    {
        self::checkAdmin();
        $bookList = Book::getBookList();
        require_once(ROOT . '/views/admin_book/index.php');
        return true;
    }

    public function actionCreate()
    {
        self::checkAdmin();
        $genres = Genre::getGenreList();
        $publisherList = Publisher::getPublisherList();
        $authorList = Author::getAuthorList();
        $errors = false;
        $messages = false;
        $options = false;

        if (isset($_POST['submit'])) {
            $author_id = htmlspecialchars($_POST['author_id']);
            $options['id'] = htmlspecialchars($_POST['book_id']);
            $options['name'] = htmlspecialchars($_POST['name']);
            $options['genre_id'] = htmlspecialchars($_POST['genre_id']);
            $options['publisher_id'] = htmlspecialchars($_POST['publisher_id']);
            $options['year'] = htmlspecialchars($_POST['year']);
            $options['pages'] = htmlspecialchars($_POST['page_count']);

            if (Book::checkBookIdExists($options['id'])) {
                $errors[] = 'Книга з заданим кодом вже існує!';
            }
            if (intval($options['year']) < 1000 || intval($options['year']) > intval(date("Y"))) {
                $errors[] = 'Рік видання задано не корректно!';
            }
            if (intval($options['pages']) <= 0) {
                $errors[] = 'Кількість сторінок задано не корректно!';
            }

            $fileName = NULL;
            if (isset($_FILES['img_file']) && $_FILES['img_file']['error'] == 0) {
                $photoFolder = ROOT . '/template/images/books/';
                $fileType = explode('.', $_FILES['img_file']['name'])[1];
                $fileName = $options['id'] . '.' . $fileType;
                $uploadFile = $photoFolder . $fileName;
                $types = ['gif', 'png', 'jpeg', 'jpg', 'bmp'];
                if (in_array($fileType, $types))
                {
                    if ($_FILES['img_file']['size'] <= 5 * 1024 * 1024)
                    {
                        copy($_FILES['img_file']['tmp_name'], $uploadFile);
                    }
                    else
                    {
                        $errors[] = "Зображення має бути розміром не більше 5 Мб!";
                    }
                }
```

```
            else
            {
                $errors[] = "Файл не відповідає типам (jpg, jpeg, png, gif, bmp)!";
            }
        }

        if ($errors == false) {
            $id = Book::createBook($options);
            if ($id) {
                if (!is_null($fileName)) {
                    Book::changeImageById($id, $fileName);
                }
                if (Book::addAuthor($id, $author_id)) {
                    $messages[] = 'Книгу успішно додано.';
                } else {
                    $errors[] = 'Помилка запиту до бази даних!';
                }
            }
        }
    }

    require_once(ROOT . '/views/admin_book/create.php');
    return true;
}

public function actionUpdate($id)
{
    self::checkAdmin();
    $book = Book::getBookById($id);
    $genres = Genre::getGenreList();
    $publisherList = Publisher::getPublisherList();
    $authorList = Author::getAuthorList();
    $errors = false;
    $messages = false;
    $options = false;

    if (isset($_POST['submit'])) {
        $author_id = htmlspecialchars($_POST['author_id']);
        $options['name'] = htmlspecialchars($_POST['name']);
        $options['genre_id'] = htmlspecialchars($_POST['genre_id']);
        $options['publisher_id'] = htmlspecialchars($_POST['publisher_id']);
        $options['year'] = htmlspecialchars($_POST['year']);
        $options['pages'] = htmlspecialchars($_POST['page_count']);

        if (intval($options['year']) < 1000 || intval($options['year']) > intval(date("Y"))) {
            $errors[] = 'Рік видання задано не корректно!';
        }
        if (intval($options['pages']) <= 0) {
            $errors[] = 'Кількість сторінок задано не корректно!';
        }

        $fileName = NULL;
        if (isset($_FILES["img_file"]) && $_FILES['img_file']['error'] == 0) {
            $photoFolder = ROOT . '/template/images/books/';
            $fileType = explode('.', $_FILES['img_file']['name'])[1];
            $fileName = $id . '.' . $fileType;
            $uploadFile = $photoFolder . $fileName;
            $types = ['gif', 'png', 'jpeg', 'jpg', 'bmp'];
            if (in_array($fileType, $types))
            {
                if ($_FILES['img_file']['size'] <= 5 * 1024 * 1024)
                {
```

```
                copy($_FILES['img_file']['tmp_name'], $uploadFile);
            }
            else
            {
                $errors[] = "Зображення має бути розміром не більше 5 Мб!";
            }
        }
        else
        {
            $errors[] = "Файл не відповідає типам (jpg, jpeg, png, gif, bmp)!";
        }
    }

    if ($errors == false) {
        $result = Book::updateBookById($id, $options);
        if ($result) {
            if (!is_null($fileName)) {
                Book::changeImageById($id, $fileName);
            }
            if (Book::changeAuthor($id, $author_id)) {
                $messages[] = 'Дані книги успішно змінено.';
            } else {
                $errors[] = 'Помилка запиту до бази даних!';
            }
        }
    }
    require_once(ROOT . '/views/admin_book/update.php');
    return true;
}

public function actionDelete($id)
{
    self::checkAdmin();
    Book::deleteBookById($id);
    header("Location: /admin/book");
    return true;
}
}
```

Лістинг файлу AdminController.php

```
<?php

class AdminController extends AdminBase
{
    public function actionIndex()
    {
        self::checkAdmin();
        $userId = User::checkLogged();
        $user = User::getUserById($userId);

        require_once(ROOT . '/views/admin/index.php');
        return true;
    }
}
```

Лістинг файлу AdminGenreController.php

```
<?php
```

```
class AdminGenreController extends AdminBase
{
    public function actionIndex()
    {
        self::checkAdmin();
        $genreList = Genre::getGenreList();
        require_once(ROOT . '/views/admin_genre/index.php');
        return true;
    }

    public function actionCreate()
    {
        self::checkAdmin();
        $errors = false;
        $messages = false;
        $options = false;
        $genreName = false;

        if (isset($_POST['submit'])) {
            $genreName = htmlspecialchars($_POST['genre_name']);
            if ($errors == false) {
                if (Genre::createGenre($genreName)) {
                    $messages[] = 'Жанр успішно додано.';
                } else {
                    $errors[] = 'Помилка запиту до бази даних!';
                }
            }
        }
        require_once(ROOT . '/views/admin_genre/create.php');
        return true;
    }

    public function actionUpdate($id)
    {
        self::checkAdmin();
        $errors = false;
        $messages = false;
        $options = false;
        $genre = Genre::getGenreById($id);

        if (isset($_POST['submit'])) {
            $genreName = htmlspecialchars($_POST['genre_name']);
            if ($errors == false) {
                if (Genre::updateGenreById($id, $genreName)) {
                    $messages[] = 'Жанр успішно змінено.';
                } else {
                    $errors[] = 'Помилка запиту до бази даних!';
                }
            }
        }
        require_once(ROOT . '/views/admin_genre/update.php');
        return true;
    }

    public function actionDelete($id)
    {
        self::checkAdmin();
        Genre::deleteGenreById($id);
        header("Location: /admin/genre");
        return true;
    }
```

```
}
```

Лістинг файлу AdminPublisherController.php

```
<?php

class AdminPublisherController extends AdminBase
{
    public function actionIndex()
    {
        self::checkAdmin();
        $publisherList = Publisher::getPublisherList();
        require_once(ROOT . '/views/admin_publisher/index.php');
        return true;
    }

    public function actionCreate()
    {
        self::checkAdmin();
        $errors = false;
        $messages = false;
        $options = false;
        $publisherName = false;
        $cityId = false;
        $cities = Publisher::getCitiesList();

        if (isset($_POST['submit'])) {
            $publisherName = htmlspecialchars($_POST['publisher_name']);
            $cityId = htmlspecialchars($_POST['city_id']);

            if ($errors == false) {
                if (Publisher::createPublisher($publisherName, $cityId)) {
                    $messages[] = 'Видавництво успішно додано.';
                } else {
                    $errors[] = 'Помилка запиту до бази даних!';
                }
            }
        }
        require_once(ROOT . '/views/admin_publisher/create.php');
        return true;
    }

    public function actionUpdate($id)
    {
        self::checkAdmin();
        $errors = false;
        $messages = false;
        $options = false;
        $cityId = false;
        $cities = Publisher::getCitiesList();
        $publisher = Publisher::getPublisherById($id);

        if (isset($_POST['submit'])) {
            $publisherName = htmlspecialchars($_POST['publisher_name']);
            $cityId = htmlspecialchars($_POST['city_id']);

            if ($errors == false) {
                if (Publisher::updatePublisherById($id, $publisherName, $cityId)) {
                    $messages[] = 'Видавництво успішно змінено.';
                } else {
                    $errors[] = 'Помилка запиту до бази даних!';
                }
            }
```

```
        }
        require_once(ROOT . '/views/admin_publisher/update.php');
        return true;
    }

    public function actionDelete($id)
    {
        self::checkAdmin();
        Publisher::deletePublisherById($id);
        header("Location: /admin/publisher");
        return true;
    }

}
```

Лістинг файлу BookController.php

```
<?php

class BookController
{
    public function actionView($bookId)
    {
        $genres = Genre::getGenreList();
        $book = Book::getBookById($bookId);

        require_once(ROOT . '/views/book/view.php');
        return true;
    }
}
```

Лістинг файлу CabinetController.php

```
<?php
class CabinetController
{
    public function actionIndex()
    {
        $userId = User::checkLogged();
        $user = User::getUserById($userId);
        $orderItems = Reservation::getReservationByUserId($userId);

        require_once(ROOT . '/views/cabinet/index.php');
        return true;
    }

    public function actionEdit()
    {
        $userId = User::checkLogged();
        $user = User::getUserById($userId);

        $name = false;
        $surname = false;
        $middle_name = false;
        $email = false;
        $phone = false;
        $messages = false;

        if (isset($_POST['submit'])) {

            $name = htmlspecialchars($_POST['name']);
            $surname = htmlspecialchars($_POST['surname']);
            $middle_name = htmlspecialchars($_POST['middle_name']);
            $email = htmlspecialchars($_POST['email']);
            $phone = htmlspecialchars($_POST['phone']);
```

```
        $errors = false;

        if (!User::checkName($name)) {
            $errors[] = 'Значення поля "Ім'я" має бути більше, ніж 2 символи';
        }
        if (!User::checkName($surname)) {
            $errors[] = 'Значення поля "Прізвище" має бути більше, ніж 2 символи';
        }
        if (!User::checkName($middle_name)) {
            $errors[] = 'Значення поля "По батькові" має бути більше, ніж 2 символи';
        }
        if (!User::checkEmail($email)) {
            $errors[] = 'Неправильна електрона адреса';
        }
        if ($errors == false) {
            if (User::edit($userId, $name, $surname, $middle_name, $phone, $email)) {
                $messages[] = 'Ваші дані успішно змінені.';
            }
        }
    }
    unset($_SESSION['errors']);
    $_SESSION['errors'] = $errors;
    unset($_SESSION['messages']);
    $_SESSION['messages'] = $messages;

    header("Location: /cabinet/");
    return true;
}

public function actionChange()
{
    $userId = User::checkLogged();
    $user = User::getUserById($userId);
    $errors = false;
    $messages = false;

    $photoFolder = ROOT . '/template/images/user/';
    $fileType = explode('.', $_FILES['photo']['name'])[1];
    $photoName = 'user' . $userId . '_avatar.' . $fileType;
    $uploadFile = $photoFolder . $photoName;
    $types = array('gif', 'png', 'jpeg', 'jpg', 'bmp');

    if (in_array($fileType, $types))
    {
        if ($_FILES['photo']['size'] <= 5 * 1024 * 1024)
        {
            copy($_FILES['photo']['tmp_name'], $uploadFile);
            if (User::updatePhotoById($userId, $photoName)) {
                $messages[] = 'Зображення успішно змінено.';
            }
        }
        else
        {
            $errors[] = "Зображення має бути розміром не більше 5 Мб!";
        }
    }
    else
    {
        $errors[] = "Файл не відповідає типам (jpg, jpeg, png, gif, bmp)!";
    }
```

```
        unset($_SESSION['errors']);
        $_SESSION['errors'] = $errors;
        unset($_SESSION['messages']);
        $_SESSION['messages'] = $messages;

        header("Location: /cabinet/");
        return true;
    }

    public function actionPasschange()
    {
        $userId = User::checkLogged();
        $user = User::getUserById($userId);

        $password = false;
        $pass_repeat = false;
        $errors = false;
        $messages = false;

        if (isset($_POST['submit'])) {

            $password = $_POST['pass'];
            $pass_repeat = $_POST['pass_repeat'];

            if ($password != $pass_repeat) {
                $errors[] = 'Паролі не співпадають!';
            }
            if ($errors == false) {
                if (User::updatePassword($userId, $password)) {
                    $messages[] = 'Пароль успішно змінено.';
                }
            }
        }

        unset($_SESSION['errors']);
        $_SESSION['errors'] = $errors;
        unset($_SESSION['messages']);
        $_SESSION['messages'] = $messages;

        header("Location: /cabinet/");
        return true;
    }

    public function actionBookList()
    {
        $userId = User::checkLogged();
        $user = User::getUserById($userId);

        $order_id = 0;
        if(isset($_SESSION['order_id'])) {
          $order_id = $_SESSION['order_id'];
          $orderItems = Reservation::getReservationById($order_id);
          $items = json_decode($orderItems['books'], true);
        }

        require_once(ROOT . '/views/cabinet/book_list.php');
        return true;
    }
    public function actionBookOrder()
    {
        $userId = User::checkLogged();
        $user = User::getUserById($userId);
```

```
        $orderItems = false;
        $items = false;

        $order_id = 0;
        if(isset($_POST['order_id'])) {
          $order_id = htmlspecialchars($_POST['order_id']);

          $orderItems = Reservation::getReservationById($order_id);
          $items = json_decode($orderItems['books'], true);
        }

        require_once(ROOT . '/views/cabinet/book_list.php');
        return true;
    }

    public function actionDelete()
    {
        $userId = User::checkLogged();
        $user = User::getUserById($userId);
        $errors = false;
        $messages = false;

        $order_id = 0;
        if(isset($_SESSION['order_id'])) {
          $order_id = $_SESSION['order_id'];
          Reservation::deleteReservationById($order_id);
          unset($_SESSION['order_id']);
          $messages[] = "Успішно видалено.";
        }
        unset($_SESSION['messages']);
        $_SESSION['messages'] = $messages;
        require_once(ROOT . '/views/cabinet/book_list.php');
        return true;
    }

}
```

Лістинг файлу CartController.php

```
<?php
class CartController
{
    public function actionAdd($id)
    {
        Cart::addBook($id);
        $referrer = $_SERVER['HTTP_REFERER'];
        header("Location: $referrer");
    }

    public function actionAddAjax($id)
    {
        echo Cart::addBook($id);
        return true;
    }

    public function actionDelete($id)
    {
        Cart::deleteBook($id);
        header("Location: /cart");
    }

    public function actionClear()
    {
        Cart::clear();
```

```
        header("Location: /cart");
    }

    public function actionIndex()
    {
        $genres = Genre::getGenreList();
        $cartItems = Cart::getBooks();

        if ($cartItems) {
            $itemsId = array_keys($cartItems);
            $items = Book::getBooksByIds($itemsId);
            // total
        }

        require_once(ROOT . '/views/cart/index.php');
        return true;
    }
    public function actionCheckout()
    {
        $cartItems = Cart::getBooks();
        $errors = false;

        if ($cartItems == false) {
            $errors[] = "Кошик порожній!";
        }

        $itemsId = array_keys($cartItems);
        $items = Book::getBooksByIds($itemsId);
        $result = false;
        $messages = false;

        if (!User::isGuest()) {
            $userId = User::checkLogged();
            $user = User::getUserById($userId);
            $userName = $user['name'];
        } else {
            $userId = false;
            $errors[] = "Забронювати книги можуть лише авторизовані користувачі!";
        }

        if ($errors == false) {
            $result = Reservation::save($userId, $items);
            if ($result) {
                Cart::clear();
                $order_id = Reservation::getLastId();
                $_SESSION['order_id'] = $order_id;
                $messages[] = "Книжки успішно заброньовано.";
            }
        }
        unset($_SESSION['errors']);
        $_SESSION['errors'] = $errors;
        unset($_SESSION['messages']);
        $_SESSION['messages'] = $messages;

        header("Location: /cart/");
        return true;
    }
}
```

Лістинг файлу SiteController.php

```
<?php

class SiteController
```

```
{
    public function actionIndex($page = 1)
    {
        $genres = Genre::getGenreList();
        $latestBooks = Book::getLatestBooks($page);
        $count = Book::getCount();

        $pagination = new Pagination($count, $page, Book::SHOW_BY_DEFAULT, 'page-');

        require_once(ROOT . '/views/site/index.php');
        return true;
    }
    public function actionCategory($genreId, $page = 1)
    {
        $genres = Genre::getGenreList();

        $latestBooks = Book::getBookListByGenre($genreId, $page);
        $count = Book::getTotalBooksInCategory($genreId);
        $pagination = new Pagination($count, $page, Book::SHOW_BY_DEFAULT, 'page-');

        require_once(ROOT . '/views/site/index.php');
        return true;
    }
    public function actionAbout()
    {
        require_once(ROOT . '/views/site/about.php');
        return true;
    }
}
```

Лістинг файлу UserController.php

```
<?php
class UserController
{
    public function actionRegister()
    {
        $name = false;
        $surname = false;
        $middle_name = false;
        $email = false;
        $password = false;
        $phone = false;
        $result = false;


        if (isset($_POST['submit'])) {

            $name = htmlspecialchars($_POST['name']);
            $surname = htmlspecialchars($_POST['surname']);
            $middle_name = htmlspecialchars($_POST['middle_name']);
            $email = htmlspecialchars($_POST['email']);
            $password = $_POST['password'];
            $phone = htmlspecialchars($_POST['phone']);

            $errors = false;

            if (!User::checkName($name)) {
                $errors[] = 'Значення поля "Ім'я" має бути більше, ніж 2 символи';
            }
            if (!User::checkName($surname)) {
                $errors[] = 'Значення поля "Прізвище" має бути більше, ніж 2 символи';
            }
```

```
            if (!User::checkName($middle_name)) {
                $errors[] = 'Значення поля "По батькові" має бути більше, ніж 2 символи';
            }
            if (!User::checkEmail($email)) {
                $errors[] = 'Неправильна електрона адреса';
            }
            if (!User::checkPassword($password)) {
                $errors[] = 'Некоректний пароль';
            }
            if (User::checkEmailExists($email)) {
                $errors[] = 'Задана електронна адреса вже використовується';
            }

            if ($errors == false) {
                $result = User::register($name, $surname, $middle_name, $phone, $email, $password);
                if($result) {
                    $userId = User::checkUserData($email, $password);
                    User::auth($userId, $user_info['role_id']);
                    unset($_SESSION['order_id']);
                    header("Location: /");
                }
            }
        }

        require_once(ROOT . '/views/user/register.php');
        return true;
    }

    public function actionLogin()
    {
        $email = false;
        $password = false;

        if (isset($_POST['submit'])) {
            $email = htmlspecialchars($_POST['email']);
            $password = htmlspecialchars($_POST['password']);

            $errors = false;

            if (!User::checkEmail($email)) {
                $errors[] = 'Неправильна електрона адреса';
            }
            if (!User::checkPassword($password)) {
                $errors[] = 'Некоректний пароль';
            }

            $userId = User::checkUserData($email, $password);
            $user_info = User::getUserById($userId);

            if ($userId == false) {
                $errors[] = 'Неправильна електронна адреса або пароль.';
            } else {
                User::auth($userId, $user_info['role_id']);
                unset($_SESSION['order_id']);
                header("Location: /");
            }
        }

        require_once(ROOT . '/views/user/login.php');
        return true;
    }
```

```
    public function actionLogout()
    {
        session_start();
        unset($_SESSION["user"]);
        unset($_SESSION["user_role"]);
        header("Location: /");
    }
}
```

Лістинг файлу Author.php

```
<?php

class Author
{
    public static function getAuthorList()
    {
        $db = Db::getConnection();
        $result = $db->query('SELECT id, name, surname, middle_name FROM author ORDER BY name ASC');
        return $result->fetchAll();
    }

    public static function deleteAuthorById($id)
    {
        $db = Db::getConnection();
        $sql = 'DELETE FROM author WHERE id = :id';
        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);
        return $result->execute();
    }

    public static function updateAuthorById($id, $name, $surname, $middle_name)
    {
        $db = Db::getConnection();
        $sql = "UPDATE publisher SET name = :name, surname = :surname,
         middle_name = :middle_name WHERE id = :id";
        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);
        $result->bindParam(':name', $name, PDO::PARAM_STR);
        $result->bindParam(':surname', $surname, PDO::PARAM_STR);
        $result->bindParam(':middle_name', $middle_name, PDO::PARAM_STR);
        return $result->execute();
    }

    public static function getAuthorById($id)
    {
        $db = Db::getConnection();
        $sql = 'SELECT * FROM author WHERE id = :id';
        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);
        $result->setFetchMode(PDO::FETCH_ASSOC);
        $result->execute();
        return $result->fetch();
    }

    public static function createAuthor($name, $surname, $middle_name)
    {
        $db = Db::getConnection();
        $sql = "INSERT INTO author (name, surname, middle_name)
                VALUES (:name, :surname, :middle_name)";
        $result = $db->prepare($sql);
        $result->bindParam(':name', $name, PDO::PARAM_STR);
        $result->bindParam(':surname', $surname, PDO::PARAM_STR);
```

```
        $result->bindParam(':middle_name', $middle_name, PDO::PARAM_STR);
        return $result->execute();
    }

}
```

Лістинг файлу Book.php

```
<?php

class Book
{
    const SHOW_BY_DEFAULT = 6;

    public static function getLatestBooks($page = 1)
    {
        $limit = Book::SHOW_BY_DEFAULT;
        $offset = ($page - 1) * self::SHOW_BY_DEFAULT;

        $db = Db::getConnection();
        $sql = 'SELECT book.id, book.name, author.name as author_name, author.surname as author_surname,
author.middle_name as author_midname, '
                .'genre.name as genre_name, publisher.name as publisher_name, year, pages, price, amount FROM book
'
                .'INNER JOIN genre ON genre.id = book.genre_id '
                .'INNER JOIN publisher ON publisher.id = book.publisher_id '
                .'INNER JOIN book_author ON book_author.book_id = book.id '
                .'INNER JOIN author ON book_author.author_id = author.id '
                .'ORDER BY book.id DESC LIMIT :limit OFFSET :offset';

        $result = $db->prepare($sql);
        $result->bindParam(':limit', $limit, PDO::PARAM_INT);
        $result->bindParam(':offset', $offset, PDO::PARAM_INT);
        $result->setFetchMode(PDO::FETCH_ASSOC);
        $result->execute();
        return $result->fetchAll();
    }

    public static function getBookListByGenre($genreId, $page = 1)
    {
        $limit = Book::SHOW_BY_DEFAULT;
        $offset = ($page - 1) * self::SHOW_BY_DEFAULT;

        $db = Db::getConnection();
        $sql = 'SELECT book.id, book.name, author.name as author_name, author.surname as author_surname,
author.middle_name as author_midname,'
                .'genre.name as genre_name, genre_id, publisher.name as publisher_name, year, pages, price, amount
FROM book '
                .'INNER JOIN genre ON genre.id = book.genre_id '
                .'INNER JOIN publisher ON publisher.id = book.publisher_id '
                .'INNER JOIN book_author ON book_author.book_id = book.id '
                .'INNER JOIN author ON book_author.author_id = author.id '
                ."WHERE book.genre_id = :genre_id "
                ."ORDER BY book.id ASC LIMIT :limit OFFSET :offset";

        $result = $db->prepare($sql);
        $result->bindParam(':genre_id', $genreId, PDO::PARAM_INT);
        $result->bindParam(':limit', $limit, PDO::PARAM_INT);
        $result->bindParam(':offset', $offset, PDO::PARAM_INT);
        $result->execute();
        return $result->fetchAll();
    }

    public static function getBookById($id)
```

```
    {
        $db = Db::getConnection();
        $sql = "SELECT book.id, book.name, author.name as author_name, author.surname as author_surname,
author.middle_name as author_midname,
                genre.name as genre_name, genre.id as genre_id, publisher.id as publisher_id, author.id as author_id,
                publisher.name as publisher_name, year, pages, price, amount FROM book
                INNER JOIN genre ON genre.id = book.genre_id
                INNER JOIN publisher ON publisher.id = book.publisher_id
                INNER JOIN book_author ON book_author.book_id = book.id
                INNER JOIN author ON book_author.author_id = author.id
                WHERE book.id = :id";
        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_STR);
        $result->setFetchMode(PDO::FETCH_ASSOC);
        $result->execute();
        return $result->fetch();
    }

    public static function getTotalBooksInCategory($genreId)
    {
        $db = Db::getConnection();
        $sql = 'SELECT count(id) AS count FROM book WHERE genre_id = :genre_id';

        $result = $db->prepare($sql);
        $result->bindParam(':genre_id', $genreId, PDO::PARAM_INT);
        $result->execute();
        $row = $result->fetch();
        return $row['count'];
    }

    public static function getCount()
    {
        $db = Db::getConnection();
        $sql = 'SELECT count(*) AS count FROM book';
        $result = $db->prepare($sql);
        $result->execute();
        $row = $result->fetch();
        return $row['count'];
    }

    public static function getBooksByIds($idsArray)
    {
        $db = Db::getConnection();
        $stringId = "'". implode("','", $idsArray). "'";
        $sql = 'SELECT book.id, book.name, author.name as author_name, author.surname as author_surname,
author.middle_name as author_midname,'
                .'genre.name as genre_name, genre.id as genre_id, publisher.id as publisher_id, author.id as author_id,'
                .'publisher.name as publisher_name, year, pages, price, amount FROM book '
                .'INNER JOIN genre ON genre.id = book.genre_id '
                .'INNER JOIN publisher ON publisher.id = book.publisher_id '
                .'INNER JOIN book_author ON book_author.book_id = book.id '
                .'INNER JOIN author ON book_author.author_id = author.id '
                .'WHERE book.id IN (' . $stringId . ')';
        $result = $db->query($sql);
        $result->setFetchMode(PDO::FETCH_ASSOC);
        return $result->fetchAll();
    }

    public static function getBookList()
    {
      $db = Db::getConnection();
      $sql = 'SELECT book.id, book.name, author.name as author_name, author.surname as author_surname,
```

```
author.middle_name as author_midname,'
            .'genre.name as genre_name, publisher.name as publisher_name, year, pages, price, amount FROM book '
            .'INNER JOIN genre ON genre.id = book.genre_id '
            .'INNER JOIN publisher ON publisher.id = book.publisher_id '
            .'INNER JOIN book_author ON book_author.book_id = book.id '
            .'INNER JOIN author ON book_author.author_id = author.id '
            .'ORDER BY id ASC';

    $result = $db->prepare($sql);
    $result->setFetchMode(PDO::FETCH_ASSOC);
    $result->execute();
    return $result->fetchAll();
  }


  public static function deleteBookById($id)
  {
    $db = Db::getConnection();
    $sql = 'DELETE FROM book WHERE id = :id';

    $result = $db->prepare($sql);
    $result->bindParam(':id', $id, PDO::PARAM_STR);
    return $result->execute();
  }

  public static function updateBookById($id, $options)
  {
    $db = Db::getConnection();
    $sql = 'UPDATE book SET name=:name, genre_id=:genre_id, publisher_id=:publisher_id, '
        .'year=:year, pages=:pages '
        .'WHERE id=:id';

    $result = $db->prepare($sql);
    $result->bindParam(':id', $id, PDO::PARAM_STR);
    $result->bindParam(':name', $options['name'], PDO::PARAM_STR);
    $result->bindParam(':genre_id', $options['genre_id'], PDO::PARAM_INT);
    $result->bindParam(':publisher_id', $options['publisher_id'], PDO::PARAM_INT);
    $result->bindParam(':year', $options['year'], PDO::PARAM_INT);
    $result->bindParam(':pages', $options['pages'], PDO::PARAM_INT);
    return $result->execute();
  }


  public static function createBook($options)
  {
    $db = Db::getConnection();
    $sql = 'INSERT INTO book (id, name, genre_id, publisher_id, year, pages)'
        .' VALUES (:id, :name, :genre_id, :publisher_id, :year, :pages)';

    $result = $db->prepare($sql);
    $result->bindParam(':id', $options['id'], PDO::PARAM_STR);
    $result->bindParam(':name', $options['name'], PDO::PARAM_STR);
    $result->bindParam(':genre_id', $options['genre_id'], PDO::PARAM_INT);
    $result->bindParam(':publisher_id', $options['publisher_id'], PDO::PARAM_INT);
    $result->bindParam(':year', $options['year'], PDO::PARAM_INT);
    $result->bindParam(':pages', $options['pages'], PDO::PARAM_INT);
    if ($result->execute()) {
      // return $db->lastInsertId();
      return $options['id'];
    }
    return 0;
  }
```

```
public static function getImageNameById($id) {
    $db = Db::getConnection();
    $sql = 'SELECT image_name FROM book WHERE id = :id';
    $result = $db->prepare($sql);
    $result->bindParam(':id', $id, PDO::PARAM_STR);
    $result->setFetchMode(PDO::FETCH_ASSOC);
    $result->execute();
    return $result->fetch()['image_name'];
}

public static function addAuthor($book_id, $author_id) {
    $db = Db::getConnection();
    $sql = "INSERT INTO book_author (book_id, author_id) VALUES (:book_id, :author_id)";
    $result = $db->prepare($sql);
    $result->bindParam(':book_id', $book_id, PDO::PARAM_STR);
    $result->bindParam(':author_id', $author_id, PDO::PARAM_INT);
    return $result->execute();
}

public static function changeAuthor($book_id, $author_id) {
    $db = Db::getConnection();
    $sql = "UPDATE book_author SET author_id=:author_id WHERE book_id=:book_id";
    $result = $db->prepare($sql);
    $result->bindParam(':author_id', $author_id, PDO::PARAM_INT);
    $result->bindParam(':book_id', $book_id, PDO::PARAM_STR);
    return $result->execute();
}

public static function checkBookIdExists($bookId)
{
    $db = Db::getConnection();
    $sql = 'SELECT COUNT(*) FROM book WHERE id = :book_id';
    $result = $db->prepare($sql);
    $result->bindParam(':book_id', $bookId, PDO::PARAM_STR);
    $result->execute();
    if ($result->fetchColumn())
        return true;
    return false;
}

public static function changeImageById($id, $imageName)
{
    $db = Db::getConnection();
    $sql = "UPDATE book SET image_name=:img_name WHERE id=:id";
    $result = $db->prepare($sql);
    $result->bindParam(':img_name', $imageName, PDO::PARAM_STR);
    $result->bindParam(':id', $id, PDO::PARAM_STR);
    return $result->execute();
}

public static function getImage($id)
{
    $noImage = 'no-image.png';
    $path = '/template/images/books/';
    $imageName = self::getImageNameById($id);

    $imagePath = $path . $imageName;
    if (file_exists($_SERVER['DOCUMENT_ROOT'].$imagePath)) {
        return $imagePath;
    }
```

```
        return $path . $noImage;
    }
}
```

Лістинг файлу Genre.php

```
<?php

class Genre
{
    public static function getGenreList()
    {
        $db = Db::getConnection();
        $result = $db->query('SELECT id, name FROM genre ORDER BY name ASC');
        return $result->fetchAll();
    }

    public static function deleteGenreById($id)
    {
        $db = Db::getConnection();
        $sql = 'DELETE FROM genre WHERE id = :id';

        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);
        return $result->execute();
    }

    public static function updateGenreById($id, $name)
    {
        $db = Db::getConnection();
        $sql = "UPDATE genre SET name = :name WHERE id = :id";
        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);
        $result->bindParam(':name', $name, PDO::PARAM_STR);
        return $result->execute();
    }

    public static function getGenreById($id)
    {
        $db = Db::getConnection();
        $sql = 'SELECT * FROM genre WHERE id = :id';
        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);
        $result->setFetchMode(PDO::FETCH_ASSOC);
        $result->execute();
        return $result->fetch();
    }

    public static function createGenre($name)
    {
        $db = Db::getConnection();
        $sql = 'INSERT INTO genre (name) VALUES (:name)';
        $result = $db->prepare($sql);
        $result->bindParam(':name', $name, PDO::PARAM_STR);
        return $result->execute();
    }
}
```

Лістинг файлу Publisher.php

```
<?php
class Publisher
{
    public static function getPublisherList()
    {
        $db = Db::getConnection();
```

```
        $result = $db->query('SELECT id, name, city_id FROM publisher ORDER BY name ASC');
        return $result->fetchAll(PDO::FETCH_ASSOC);
    }

    public static function getCitiesList()
    {
        $db = Db::getConnection();
        $result = $db->query('SELECT id, name FROM city ORDER BY name');
        return $result->fetchAll(PDO::FETCH_ASSOC);
    }

    public static function deletePublisherById($id)
    {
        $db = Db::getConnection();
        $sql = 'DELETE FROM publisher WHERE id = :id';
        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);
        return $result->execute();
    }

    public static function updatePublisherById($id, $name, $cityId)
    {
        $db = Db::getConnection();
        $sql = "UPDATE publisher SET name = :name, city_id=:city_id WHERE id = :id";
        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);
        $result->bindParam(':name', $name, PDO::PARAM_STR);
        $result->bindParam(':city_id', $cityId, PDO::PARAM_INT);
        return $result->execute();
    }

    public static function getPublisherById($id)
    {
        $db = Db::getConnection();
        $sql = 'SELECT * FROM publisher WHERE id = :id';
        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);
        $result->setFetchMode(PDO::FETCH_ASSOC);
        $result->execute();
        return $result->fetch();
    }

    public static function createPublisher($name, $cityId)
    {
        $db = Db::getConnection();
        $sql = "INSERT INTO publisher (name, city_id) VALUES (:name, :city_id)";
        $result = $db->prepare($sql);
        $result->bindParam(':name', $name, PDO::PARAM_STR);
        $result->bindParam(':city_id', $cityId, PDO::PARAM_INT);
        return $result->execute();
    }
}
```

Лістинг файлу Reservation.php

```
<?php
class Reservation
{
    public static function save($userId, $books)
    {
        $db = Db::getConnection();

        $sql = 'INSERT INTO book_reservation (customer_id, books) '
```

```
            .'VALUES (:user_id, :books)';

        $books = json_encode($books);

        $result = $db->prepare($sql);
        $result->bindParam(':user_id', $userId, PDO::PARAM_INT);
        $result->bindParam(':books', $books, PDO::PARAM_STR);

        return $result->execute();
    }
    public static function getReservationList()
    {
        $db = Db::getConnection();
        $result = $db->query("SELECT book_reservation.id, customer.name, customer.surname,
                                customer.phone, customer_id, books, order_date FROM
book_reservation
                                INNER JOIN customer ON customer_id = customer.id ORDER
BY id DESC");

        return $result->fetchAll();
    }
    public static function getStatusText($status)
    {
        switch ($status) {
            case '1':
                return 'Нове бронювання';
                break;
            case '0':
                return 'Закрито';
                break;
        }
    }
    public static function getReservationById($id)
    {
        $db = Db::getConnection();

        $sql = "SELECT book_reservation.id as res_id, customer.name, customer.surname,
                                customer.phone, customer_id, books, order_date FROM book_reservation
                                INNER JOIN customer ON customer_id = customer.id WHERE
book_reservation.id = :id";

        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);
        $result->setFetchMode(PDO::FETCH_ASSOC);
        $result->execute();
        return $result->fetch();
    }
    public static function getReservationByUserId($id)
    {
        $db = Db::getConnection();

        $sql = "SELECT book_reservation.id as res_id, customer.name, customer.surname,
                                customer.phone, customer_id, books, order_date FROM book_reservation
                                INNER JOIN customer ON customer_id = customer.id WHERE customer.id =
:id";

        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);
        $result->setFetchMode(PDO::FETCH_ASSOC);
        $result->execute();
        return $result->fetchAll();
    }
```

```
    public static function getLastId()
    {
        $db = Db::getConnection();
        $sql = "SELECT MAX(id) as last_id FROM book_reservation";
        $result = $db->prepare($sql);
        $result->setFetchMode(PDO::FETCH_ASSOC);
        $result->execute();
        return intval($result->fetch()['last_id']);
    }

    public static function deleteReservationById($id)
    {
        $db = Db::getConnection();

        $sql = 'DELETE FROM book_reservation WHERE id = :id';
        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);
        return $result->execute();
    }

    public static function updateReservationById($id, $date, $status)
    {
        $db = Db::getConnection();

        $sql = "UPDATE book_reservation
            SET
                order_date = :date,
                status = :status
            WHERE id = :id";

        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);
        $result->bindParam(':date', $date, PDO::PARAM_STR);
        $result->bindParam(':status', $status, PDO::PARAM_INT);
        return $result->execute();
    }
}
```

Лістинг файлу User.php

```
<?php
class User
{
    public static function register($name, $surname, $middle_name, $phone, $email, $password)
    {
        $db = Db::getConnection();

        $sql = 'INSERT INTO customer (email, password, name, surname, middle_name, phone) '
            . 'VALUES (:email, :password, :name, :surname, :middle_name, :phone)';

        $hashPassword = md5($password);

        $result = $db->prepare($sql);
        $result->bindParam(':email', $email, PDO::PARAM_STR);
        $result->bindParam(':password', $hashPassword, PDO::PARAM_STR);
        $result->bindParam(':name', $name, PDO::PARAM_STR);
        $result->bindParam(':surname', $surname, PDO::PARAM_STR);
        $result->bindParam(':middle_name', $middle_name, PDO::PARAM_STR);
        $result->bindParam(':phone', $phone, PDO::PARAM_STR);
        return $result->execute();
    }

    public static function updatePhotoById($id, $photoName)
    {
```

```
    $db = Db::getConnection();
    $response = false;
    $sql = "UPDATE customer SET photo = :photo_name WHERE id = :id";

    $result = $db->prepare($sql);
    $result->bindParam(':id', $id, PDO::PARAM_INT);
    $result->bindParam(':photo_name', $photoName, PDO::PARAM_STR);
    $response = $result->execute();
    return $response;
}

public static function updatePassword($id, $newPassword)
{
    $db = Db::getConnection();
    $response = false;
    $sql = "UPDATE customer SET password = :password WHERE id = :id";

    $hashPassword = md5($newPassword);

    $result = $db->prepare($sql);
    $result->bindParam(':id', $id, PDO::PARAM_INT);
    $result->bindParam(':password', $hashPassword, PDO::PARAM_STR);
    $response = $result->execute();
    return $response;
}

public static function edit($id, $name, $surname, $middle_name, $phone, $email)
{
    $db = Db::getConnection();
    $response = false;
    $sql = "UPDATE customer
        SET email = :email, name = :name, surname = :surname,
        middle_name = :middle_name, phone = :phone
        WHERE id = :id";

    $result = $db->prepare($sql);
    $result->bindParam(':email', $email, PDO::PARAM_STR);
    $result->bindParam(':name', $name, PDO::PARAM_STR);
    $result->bindParam(':surname', $surname, PDO::PARAM_STR);
    $result->bindParam(':middle_name', $middle_name, PDO::PARAM_STR);
    $result->bindParam(':phone', $phone, PDO::PARAM_STR);
    $result->bindParam(':id', $id, PDO::PARAM_INT);
    $response = $result->execute();
    return $response;
}

public static function checkUserData($email, $password)
{
    $db = Db::getConnection();
    $sql = 'SELECT * FROM customer WHERE email = :email AND password = :password';

    $hashPassword = md5($password);
    $result = $db->prepare($sql);
    $result->bindParam(':email', $email, PDO::PARAM_STR);
    $result->bindParam(':password', $hashPassword, PDO::PARAM_STR);
    $result->execute();
    $user = $result->fetch();

    if ($user) {
        return $user['id'];
    }
    return false;
```

```
}

public static function auth($userId, $userRole)
{
   $_SESSION['user'] = $userId;
   $_SESSION['user_role'] = $userRole;
}

public static function checkLogged()
{
   if (isset($_SESSION['user'])) {
      return $_SESSION['user'];
   }

   header("Location: /user/login");
}

public static function isGuest()
{
   if (isset($_SESSION['user'])) {
      return false;
   }
   return true;
}

public static function isAdmin()
{
   if (isset($_SESSION['user']) && isset($_SESSION['user_role'])
      && $_SESSION['user_role'] == 1) {
      return true;
   }
   return false;
}

public static function checkName($name)
{
   if (strlen($name) >= 2) {
      return true;
   }
   return false;
}

public static function checkPhone($phone)
{
   if (strlen($phone) >= 10) {
      return true;
   }
   return false;
}

public static function checkPassword($password)
{
   if (strlen($password) >= 3) {
      return true;
   }
   return false;
}

public static function checkEmail($email)
{
```

```
        if (filter_var($email, FILTER_VALIDATE_EMAIL)) {
            return true;
        }
        return false;
    }

    public static function checkEmailExists($email)
    {
        $db = Db::getConnection();

        $sql = 'SELECT COUNT(*) FROM customer WHERE email = :email';
        $result = $db->prepare($sql);
        $result->bindParam(':email', $email, PDO::PARAM_STR);
        $result->execute();

        if ($result->fetchColumn())
            return true;
        return false;
    }

    public static function getUserById($id)
    {
        $db = Db::getConnection();
        $sql = 'SELECT * FROM customer WHERE id = :id';

        $result = $db->prepare($sql);
        $result->bindParam(':id', $id, PDO::PARAM_INT);
        $result->setFetchMode(PDO::FETCH_ASSOC);
        $result->execute();

        return $result->fetch();
    }
}
```

Лістинг файлу views/admin/index.php

```
<?php include ROOT . '/views/layouts/header.php'; ?>

<section>
<div class="row justify-content-center">
    <h2 class="title text-center mb-4">Панель адміністратора</h2>
</div>
<div class="row justify-content-center">
  <div class="col-md-4 col-sm-4 col-lg-4">

      <div class="card">
        <div class="card-header pb-3">
          Вітаємо <strong><?= $user['name'] ?></strong>, вам доступні для управління наступні таблиці:
        </div>
         <div class="list-group list-group-flush">
            <a href="/admin/book" class="list-group-item list-group-item-action">Таблиця "Книги"</a>
            <a href="/admin/genre" class="list-group-item list-group-item-action">Таблиця "Жанри"</a>
            <a href="/admin/publisher" class="list-group-item list-group-item-action">Таблиця
"Видавництва"</a>
         </div>
      </div>
   </div>
 </div>
</section>

<?php include ROOT . '/views/layouts/footer.php'; ?>
```

Лістинг файлу admin_book/create.php

```
<?php include ROOT . '/views/layouts/header.php'; ?>
```

```
<section>
  <div class="container">
    <div class="row justify-content-center">
      <h2 class="title text-center mb-4">Додати книгу</h2>
    </div>
    <div class="row">
      <nav aria-label="breadcrumb">
       <ol class="breadcrumb">
          <li class="breadcrumb-item"><a href="/admin">Адмін-панель</a></li>
          <li class="breadcrumb-item"><a href="/admin/book">Таблиця "Книги"</a></li>
          <li class="breadcrumb-item active" aria-current="page">Додати книгу</li>
       </ol>
      </nav>
    </div>
    <div class="row justify-content-center">
     <div class="col-sm-6 col-md-6 col-lg-6">

      <?php if (isset($errors) && is_array($errors)): ?>
        <div class="alert alert-danger" role="alert">
          <ul>
            <?php foreach ($errors as $error): ?>
              <li><?= $error;?></li>
            <?php endforeach; ?>
          </ul>
        </div>
      <?php endif; ?>
      <?php if (isset($messages) && is_array($messages)): ?>
        <div class="alert alert-success" role="alert">
          <ul>
            <?php foreach ($messages as $message): ?>
              <li><?= $message;?></li>
            <?php endforeach; ?>
          </ul>
        </div>
      <?php endif; ?>

      <form action="/admin/book/create" method="post" enctype="multipart/form-data">
        <ul class="list-group">
          <li class="list-group-item">
            <div class="row">
              <div class="col-md-5 col-sm-5 col-lg-5">
                <strong>Код книги(ISBN):</strong>
              </div>
              <div class="col-md-7 col-sm-7 col-lg-7">
                <input class="form-control" required type="text" name="book_id" value="<?php if ($options)
echo $options['id'];?>"
                  pattern="^[0-9-●]{7,20}$">
              </div>
            </div>
          </li>
          <li class="list-group-item">
            <div class="row">
              <div class="col-md-5 col-sm-5 col-lg-5">
                <strong>Назва:</strong>
              </div>
              <div class="col-md-7 col-sm-7 col-lg-7">
                <input class="form-control" required type="text" name="name" value="<?php if ($options) echo
$options['name'];?>" maxlength="150">
              </div>
            </div>
          </li>
          <li class="list-group-item">
```

```
  <div class="row">
    <div class="col-md-5 col-sm-5 col-lg-5">
      <strong>Жанр:</strong>
    </div>
    <div class="col-md-7 col-sm-7 col-lg-7">
      <select class="form-control" name="genre_id">
        <?php if (is_array($genres)): ?>
          <?php foreach ($genres as $genre): ?>
            <option value="<?=$genre['id'];?>"
              <?php if ($options && $options['genre_id']==$genre['id']) echo "selected";?>
              >
              <?=$genre['name'];?>
            </option>
          <?php endforeach; ?>
        <?php endif; ?>
      </select>
    </div>
  </div>
</li>
<li class="list-group-item">
  <div class="row">
    <div class="col-md-5 col-sm-5 col-lg-5">
      <strong>Автор:</strong>
    </div>
    <div class="col-md-7 col-sm-7 col-lg-7">
      <select class="form-control" name="author_id">
        <?php if (is_array($authorList)): ?>
          <?php foreach ($authorList as $author): ?>
            <option value="<?=$author['id'];?>"
              <?php if (isset($author_id) && $author_id==$author['id']) echo "selected";?>
              >
                <?=$author['name']." ".$author['surname']." ".$author['middle_name'];?>
            </option>
          <?php endforeach; ?>
        <?php endif; ?>
      </select>
    </div>
  </div>
</li>
<li class="list-group-item">
  <div class="row">
    <div class="col-md-5 col-sm-5 col-lg-5">
      <strong>Видавництво:</strong>
    </div>
    <div class="col-md-7 col-sm-7 col-lg-7">
      <select class="form-control" name="publisher_id">
        <?php if (is_array($publisherList)): ?>
          <?php foreach ($publisherList as $publisher): ?>
            <option value="<?= $publisher['id']; ?>"
              <?php if ($options && $options['publisher_id']==$publisher['id']) echo "selected";?>
              >
                <?php echo $publisher['name']; ?>
            </option>
          <?php endforeach; ?>
        <?php endif; ?>
      </select>
    </div>
  </div>
</li>
<li class="list-group-item">
  <div class="row">
    <div class="col-md-5 col-sm-5 col-lg-5">
```

```
                <strong>Рік видання:</strong>
              </div>
              <div class="col-md-7 col-sm-7 col-lg-7">
                <input class="form-control" required type="text" name="year" value="<?php if ($options) echo $options['year'];?>">
              </div>
            </div>
          </li>
          <li class="list-group-item">
            <div class="row">
              <div class="col-md-5 col-sm-5 col-lg-5">
                <strong>Кількість сторінок:</strong>
              </div>
              <div class="col-md-7 col-sm-7 col-lg-7">
                <input class="form-control" required type="number" name="page_count" value="<?php if ($options) echo $options['pages'];?>">
              </div>
            </div>
          </li>
          <li class="list-group-item">
            <div class="row">
              <div class="col-md-5 col-sm-5 col-lg-5">
                <strong>Файл зображення:</strong>
              </div>
              <div class="col-md-7 col-sm-7 col-lg-7">
                <input class="form-control" name="img_file" type="file">
              </div>
            </div>
          </li>
          <li class="list-group-item">
            <div class="row justify-content-center">
              <div class="col-md-6 col-sm-6 col-lg-6">
                <button type="submit" name="submit" class="btn btn-outline-dark btn-block">Зберегти</button>
              </div>
            </div>
          </li>
        </ul>
      </form>
    </div>
  </div>
</section>

<?php include ROOT . '/views/layouts/footer.php'; ?>
```

Лістинг файлу admin_book/index.php

```
<?php include ROOT . '/views/layouts/header.php'; ?>

<section>
  <div class="container">
    <div class="row justify-content-center">
      <h2 class="title text-center mb-4">Таблиця "Книги"</h2>
    </div>
    <div class="row">
      <nav aria-label="breadcrumb">
        <ol class="breadcrumb">
          <li class="breadcrumb-item"><a href="/admin">Адмін-панель</a></li>
```

```
            <li class="breadcrumb-item active" aria-current="page">Таблиця "Книги"</li>
          </ol>
        </nav>
        <a href="/admin/book/create" class="btn btn-default ml-3"><i class="fas fa-plus"></i> <strong>Додати книгу</strong></a>
      </div>
      <div class="row">
        <table class="table">
        <thead class="thead-dark">
          <tr>
            <th scope="col">Код книги(ISBN)</th>
            <th scope="col">Назва</th>
            <th scope="col">Жанр</th>
            <th scope="col">Автор</th>
            <th scope="col">Видавництво</th>
            <th scope="col">Рік видання</th>
            <th scope="col">Кількість сторінок</th>
            <th scope="col">Файл зображення</th>
            <th scope="col"></th>
            <th scope="col"></th>
          </tr>
        </thead>
        <tbody>
          <?php foreach ($bookList as $book): ?>
            <tr>
              <td><?= $book['id'];?></td>
              <td><?= $book['name'];?></td>
              <td><?= $book['genre_name'];?></td>
              <td><?= $book['author_name'] . " ". $book['author_surname'];?></td>
              <td><?= $book['publisher_name'];?></td>
              <td><?= $book['year'];?></td>
              <td><?= $book['pages'];?></td>
              <td><?= Book::getImageNameById($book['id']);?></td>  <?php #?>
              <td><a  name="edit-item"  href="/admin/book/update/<?=  $book['id']  ?>" title="Редагувати"><i class="fas fa-edit"></i></a></td>
              <td><a name="delete-item" href="/admin/book/delete/<?= $book['id'] ?>" title="Видалити">
                  <i class="far fa-times-circle"></i></a>
              </td>
            </tr>
          <?php endforeach; ?>
        </tbody>
       </table>

      </div>
    </div>
</section>

<?php include ROOT . '/views/layouts/footer.php'; ?>
```

Лістинг файлу admin_book/update.php

```
<?php include ROOT . '/views/layouts/header.php'; ?>

<section>
    <div class="container">
        <div class="row justify-content-center">
            <h2 class="title text-center mb-4">Редагування книги #<?=$id;?></h2>
        </div>
        <div class="row">
            <nav aria-label="breadcrumb">
              <ol class="breadcrumb">
                <li class="breadcrumb-item"><a href="/admin">Адмін-панель</a></li>
                <li class="breadcrumb-item"><a href="/admin/book">Таблиця "Книги"</a></li>
                <li class="breadcrumb-item active" aria-current="page">Редагування книги</li>
```

```
        </ol>
      </nav>
    </div>
    <div class="row justify-content-center">
      <div class="col-sm-6 col-md-6 col-lg-6">

        <?php if (isset($errors) && is_array($errors)): ?>
          <div class="alert alert-danger" role="alert">
            <ul>
              <?php foreach ($errors as $error): ?>
                <li><?= $error;?></li>
              <?php endforeach; ?>
            </ul>
          </div>
        <?php endif; ?>
        <?php if (isset($messages) && is_array($messages)): ?>
          <div class="alert alert-success" role="alert">
            <ul>
              <?php foreach ($messages as $message): ?>
                <li><?= $message;?></li>
              <?php endforeach; ?>
            </ul>
          </div>
        <?php endif; ?>

        <form action="/admin/book/update/<?=$id;?>" method="post" enctype="multipart/form-data">
          <ul class="list-group">
            <li class="list-group-item">
              <div class="row">
                <div class="col-md-5 col-sm-5 col-lg-5">
                  <strong>Назва:</strong>
                </div>
                <div class="col-md-7 col-sm-7 col-lg-7">
                  <input class="form-control" required type="text" name="name" value="<?=$book['name'];?>"
maxlength="150">
                </div>
              </div>
            </li>
            <li class="list-group-item">
              <div class="row">
                <div class="col-md-5 col-sm-5 col-lg-5">
                  <strong>Жанр:</strong>
                </div>
                <div class="col-md-7 col-sm-7 col-lg-7">
                  <select class="form-control" name="genre_id">
                    <?php if (is_array($genres)): ?>
                      <?php foreach ($genres as $genre): ?>
                        <option value="<?=$genre['id'];?>"
                          <?php if ($book['genre_id']==$genre['id']) echo "selected";?>
                          >
                          <?=$genre['name'];?>
                        </option>
                      <?php endforeach; ?>
                    <?php endif; ?>
                  </select>
                </div>
              </div>
            </li>
            <li class="list-group-item">
              <div class="row">
                <div class="col-md-5 col-sm-5 col-lg-5">
                  <strong>Автор:</strong>
```

```
                </div>
                <div class="col-md-7 col-sm-7 col-lg-7">
                  <select class="form-control" name="author_id">
                   <?php if (is_array($authorList)): ?>
                     <?php foreach ($authorList as $author): ?>
                       <option value="<?=$author['id'];?>"
                        <?php if ($book['author_id']==$author['id']) echo "selected";?>
                        >
                         <?=$author['name']." ".$author['surname']." ".$author['middle_name'];?>
                       </option>
                     <?php endforeach; ?>
                   <?php endif; ?>
                  </select>
                </div>
              </div>
            </li>
            <li class="list-group-item">
              <div class="row">
                <div class="col-md-5 col-sm-5 col-lg-5">
                 <strong>Видавництво:</strong>
                </div>
                <div class="col-md-7 col-sm-7 col-lg-7">
                  <select class="form-control" name="publisher_id">
                   <?php if (is_array($publisherList)): ?>
                     <?php foreach ($publisherList as $publisher): ?>
                       <option value="<?= $publisher['id']; ?>"
                        <?php if ($book['publisher_id']==$publisher['id']) echo "selected";?>
                        >
                         <?php echo $publisher['name']; ?>
                       </option>
                     <?php endforeach; ?>
                   <?php endif; ?>
                  </select>
                </div>
              </div>
            </li>
            <li class="list-group-item">
              <div class="row">
                <div class="col-md-5 col-sm-5 col-lg-5">
                 <strong>Рік видання:</strong>
                </div>
                <div class="col-md-7 col-sm-7 col-lg-7">
                  <input class="form-control" required type="text" name="year" value="<?=$book['year'];?>">
                </div>
              </div>
            </li>
            <li class="list-group-item">
              <div class="row">
                <div class="col-md-5 col-sm-5 col-lg-5">
                 <strong>Кількість сторінок:</strong>
                </div>
                <div class="col-md-7 col-sm-7 col-lg-7">
                  <input class="form-control" required type="number" name="page_count"
value="<?=$book['pages'];?>">
                </div>
              </div>
            </li>
            <li class="list-group-item">
              <div class="row">
                <div class="col-md-5 col-sm-5 col-lg-5">
                 <strong>Файл зображення:</strong>
                </div>
```

```
                <div class="col-md-7 col-sm-7 col-lg-7">
                  <input class="form-control" name="img_file" type="file">
                </div>
              </div>
            </li>
            <li class="list-group-item">
              <div class="row justify-content-center">
                <div class="col-md-6 col-sm-6 col-lg-6">
                  <button type="submit" name="submit" class="btn btn-outline-dark btn-
block">Зберегти</button>
                </div>
              </div>
            </li>
          </ul>
        </form>
      </div>
    </div>
  </div>
</section>

<?php include ROOT . '/views/layouts/footer.php'; ?>
```

Лістинг файлу admin_genre/create.php

```
<?php include ROOT . '/views/layouts/header.php'; ?>

<section>
  <div class="container">
    <div class="row justify-content-center">
      <h2 class="title text-center mb-4">Додати жанр</h2>
    </div>
    <div class="row">
      <nav aria-label="breadcrumb">
        <ol class="breadcrumb">
          <li class="breadcrumb-item"><a href="/admin">Адмін-панель</a></li>
          <li class="breadcrumb-item"><a href="/admin/genre">Таблиця "Жанри"</a></li>
          <li class="breadcrumb-item active" aria-current="page">Додати жанр</li>
        </ol>
      </nav>
    </div>
    <div class="row justify-content-center">
      <div class="col-sm-6 col-md-6 col-lg-6">

        <?php if (isset($errors) && is_array($errors)): ?>
          <div class="alert alert-danger" role="alert">
            <ul>
              <?php foreach ($errors as $error): ?>
                <li><?= $error;?></li>
              <?php endforeach; ?>
            </ul>
          </div>
        <?php endif; ?>
        <?php if (isset($messages) && is_array($messages)): ?>
          <div class="alert alert-success" role="alert">
            <ul>
              <?php foreach ($messages as $message): ?>
                <li><?= $message;?></li>
              <?php endforeach; ?>
            </ul>
          </div>
        <?php endif; ?>

        <form action="/admin/genre/create" method="post">
          <ul class="list-group">
```

```
            <li class="list-group-item">
              <div class="row">
                <div class="col-md-5 col-sm-5 col-lg-5">
                  <strong>Назва жанру:</strong>
                </div>
                <div class="col-md-7 col-sm-7 col-lg-7">
                  <input class="form-control" required type="text" name="genre_name"
value="<?=$genreName;?>" maxlength="150">
                </div>
              </div>
            </li>
            <li class="list-group-item">
              <div class="row justify-content-center">
                <div class="col-md-6 col-sm-6 col-lg-6">
                  <button type="submit" name="submit" class="btn btn-outline-dark btn-
block">Зберегти</button>
                </div>
              </div>
            </li>
          </ul>
        </form>
      </div>
    </div>
  </div>
</section>

<?php include ROOT . '/views/layouts/footer.php'; ?>
```

Лістинг файлу admin_genre/index.php

```
<?php include ROOT . '/views/layouts/header.php'; ?>

<section>

  <div class="container">
    <div class="row justify-content-center">
      <h2 class="title text-center mb-4">Таблиця "Жанри"</h2>
    </div>
    <div class="row">
      <nav aria-label="breadcrumb">
        <ol class="breadcrumb">
          <li class="breadcrumb-item"><a href="/admin">Адмін-панель</a></li>
          <li class="breadcrumb-item active" aria-current="page">Таблиця "Жанри"</li>
        </ol>
      </nav>
      <a href="/admin/genre/create" class="btn btn-default ml-3"><i class="fas fa-plus"></i> <strong>Додати
жанр</strong></a>
    </div>

    <div class="row">
      <table class="table">
      <thead class="thead-dark">
        <tr>
          <th scope="col">Код жанру</th>
          <th scope="col">Назва жанру</th>
          <th scope="col"></th>
          <th scope="col"></th>
        </tr>
      </thead>
      <tbody>
        <?php foreach ($genreList as $genre): ?>
          <tr>
            <td><?= $genre['id'];?></td>
            <td><?= $genre['name'];?></td>
```

```
            <td><a name="edit-item" href="/admin/genre/update/<?= $genre['id'] ?>" title="Редагувати"><i
class="fas fa-edit"></i></a></td>
            <td><a name="delete-item" href="/admin/genre/delete/<?= $genre['id'] ?>" title="Видалити">
              <i class="far fa-times-circle"></i></a>
            </td>
          </tr>
        <?php endforeach; ?>
        </tbody>
      </table>
    </div>
  </div>
</section>

<?php include ROOT . '/views/layouts/footer.php'; ?>
```

Лістинг файлу admin_genre/update.php

```
<?php include ROOT . '/views/layouts/header.php'; ?>

<section>
  <div class="container">
    <div class="row justify-content-center">
      <h2 class="title text-center mb-4">Редагування жанру #<?=$id;?></h2>
    </div>
    <div class="row">
      <nav aria-label="breadcrumb">
       <ol class="breadcrumb">
          <li class="breadcrumb-item"><a href="/admin">Адмін-панель</a></li>
          <li class="breadcrumb-item"><a href="/admin/genre">Таблиця "Жанри"</a></li>
          <li class="breadcrumb-item active" aria-current="page">Редагувати жанр</li>
       </ol>
      </nav>
    </div>
    <div class="row justify-content-center">
     <div class="col-sm-6 col-md-6 col-lg-6">

      <?php if (isset($errors) && is_array($errors)): ?>
        <div class="alert alert-danger" role="alert">
          <ul>
            <?php foreach ($errors as $error): ?>
              <li><?= $error;?></li>
            <?php endforeach; ?>
          </ul>
        </div>
      <?php endif; ?>
      <?php if (isset($messages) && is_array($messages)): ?>
        <div class="alert alert-success" role="alert">
          <ul>
            <?php foreach ($messages as $message): ?>
              <li><?= $message;?></li>
            <?php endforeach; ?>
          </ul>
        </div>
      <?php endif; ?>

      <form action="/admin/genre/update/<?=$id;?>" method="post" enctype="multipart/form-data">
        <ul class="list-group">
          <li class="list-group-item">
            <div class="row">
              <div class="col-md-5 col-sm-5 col-lg-5">
                <strong>Назва жанру:</strong>
              </div>
              <div class="col-md-7 col-sm-7 col-lg-7">
                <input class="form-control" required type="text" name="genre_name"
```

```
value="<?=$genre['name'];?>" maxlength="150">
                    </div>
                  </div>
                </li>
                <li class="list-group-item">
                  <div class="row justify-content-center">
                    <div class="col-md-6 col-sm-6 col-lg-6">
                      <button type="submit" name="submit" class="btn btn-outline-dark btn-
block">Зберегти</button>
                    </div>
                  </div>
                </li>
              </ul>
            </form>
          </div>
        </div>
      </div>
    </section>

    <?php include ROOT . '/views/layouts/footer.php'; ?>
```

Лістинг файлу admin_publisher/create.php

```
<?php include ROOT . '/views/layouts/header.php'; ?>

<section>
  <div class="container">
    <div class="row justify-content-center">
      <h2 class="title text-center mb-4">Додати видавництво</h2>
    </div>
    <div class="row">
      <nav aria-label="breadcrumb">
        <ol class="breadcrumb">
          <li class="breadcrumb-item"><a href="/admin">Адмін-панель</a></li>
          <li class="breadcrumb-item"><a href="/admin/publisher">Таблиця "Видавництва"</a></li>
          <li class="breadcrumb-item active" aria-current="page">Додати видавництво</li>
        </ol>
      </nav>
    </div>
    <div class="row justify-content-center">
      <div class="col-sm-6 col-md-6 col-lg-6">

        <?php if (isset($errors) && is_array($errors)): ?>
          <div class="alert alert-danger" role="alert">
            <ul>
              <?php foreach ($errors as $error): ?>
                <li><?= $error;?></li>
              <?php endforeach; ?>
            </ul>
          </div>
        <?php endif; ?>
        <?php if (isset($messages) && is_array($messages)): ?>
          <div class="alert alert-success" role="alert">
            <ul>
              <?php foreach ($messages as $message): ?>
                <li><?= $message;?></li>
              <?php endforeach; ?>
            </ul>
          </div>
        <?php endif; ?>

        <form action="/admin/publisher/create" method="post">
          <ul class="list-group">
            <li class="list-group-item">
```

```
            <div class="row">
              <div class="col-md-5 col-sm-5 col-lg-5">
                <strong>Назва видавництва:</strong>
              </div>
              <div class="col-md-7 col-sm-7 col-lg-7">
                <input class="form-control" required type="text" name="publisher_name"
value="<?=$publisherName;?>" maxlength="150">
              </div>
            </div>
          </li>
          <li class="list-group-item">
            <div class="row">
              <div class="col-md-5 col-sm-5 col-lg-5">
                <strong>Місто:</strong>
              </div>
              <div class="col-md-7 col-sm-7 col-lg-7">
                <select class="form-control" name="city_id">
                  <?php if (is_array($cities)): ?>
                    <?php foreach ($cities as $city): ?>
                      <option value="<?=$city['id'];?>"
                        <?php if ($cityId==$city['id']) echo "selected";?>
                        >
                        <?=$city['name'];?>
                      </option>
                    <?php endforeach; ?>
                  <?php endif; ?>
                </select>
              </div>
            </div>
          </li>
          <li class="list-group-item">
            <div class="row justify-content-center">
              <div class="col-md-6 col-sm-6 col-lg-6">
                <button type="submit" name="submit" class="btn btn-outline-dark btn-
block">Зберегти</button>
              </div>
            </div>
          </li>
        </ul>
      </form>
    </div>
  </div>
</section>

<?php include ROOT . '/views/layouts/footer.php'; ?>
```

Лістинг файлу admin_publisher/index.php

```
<?php include ROOT . '/views/layouts/header.php'; ?>

<section>

  <div class="container">
    <div class="row justify-content-center">
      <h2 class="title text-center mb-4">Таблиця "Видавництва"</h2>
    </div>
    <div class="row">
      <nav aria-label="breadcrumb">
        <ol class="breadcrumb">
          <li class="breadcrumb-item"><a href="/admin">Адмін-панель</a></li>
          <li class="breadcrumb-item active" aria-current="page">Таблиця "Видавництва"</li>
        </ol>
      </nav>
```

```
        <a href="/admin/publisher/create" class="btn btn-default ml-3"><i class="fas fa-plus"></i>
<strong>Додати видавництво</strong></a>
      </div>

      <div class="row">
        <table class="table">
        <thead class="thead-dark">
          <tr>
            <th scope="col">Код видавництва</th>
            <th scope="col">Назва видавництва</th>
            <th scope="col">Код міста</th>
            <th scope="col"></th>
            <th scope="col"></th>
          </tr>
        </thead>
        <tbody>
          <?php foreach ($publisherList as $publisher): ?>
            <tr>
              <td><?= $publisher['id'];?></td>
              <td><?= $publisher['name'];?></td>
              <td><?= $publisher['city_id'];?></td>
              <td><a name="edit-item" href="/admin/publisher/update/<?= $publisher['id'] ?>"
title="Редагувати"><i class="fas fa-edit"></i></a></td>
              <td><a name="delete-item" href="/admin/publisher/delete/<?= $publisher['id'] ?>"
title="Видалити">
                <i class="far fa-times-circle"></i></a>
              </td>
            </tr>
          <?php endforeach; ?>
        </tbody>
       </table>

      </div>
    </div>
</section>

<?php include ROOT . '/views/layouts/footer.php'; ?>
```

Лістинг файлу admin_publisher/update.php

```
<?php include ROOT . '/views/layouts/header.php'; ?>

<section>
  <div class="container">
    <div class="row justify-content-center">
      <h2 class="title text-center mb-4">Редагування видавництва #<?=$id;?></h2>
    </div>
    <div class="row">
      <nav aria-label="breadcrumb">
        <ol class="breadcrumb">
          <li class="breadcrumb-item"><a href="/admin">Адмін-панель</a></li>
          <li class="breadcrumb-item"><a href="/admin/publisher">Таблиця "Видавництва"</a></li>
          <li class="breadcrumb-item active" aria-current="page">Редагувати видавництво</li>
        </ol>
      </nav>
    </div>
    <div class="row justify-content-center">
      <div class="col-sm-6 col-md-6 col-lg-6">

        <?php if (isset($errors) && is_array($errors)): ?>
          <div class="alert alert-danger" role="alert">
            <ul>
              <?php foreach ($errors as $error): ?>
                <li><?= $error;?></li>
```

```
                <?php endforeach; ?>
              </ul>
            </div>
          <?php endif; ?>
          <?php if (isset($messages) && is_array($messages)): ?>
            <div class="alert alert-success" role="alert">
              <ul>
                <?php foreach ($messages as $message): ?>
                  <li><?= $message;?></li>
                <?php endforeach; ?>
              </ul>
            </div>
          <?php endif; ?>

          <form action="/admin/publisher/update/<?=$id;?>" method="post">
            <ul class="list-group">
              <li class="list-group-item">
                <div class="row">
                  <div class="col-md-5 col-sm-5 col-lg-5">
                    <strong>Назва видавництва:</strong>
                  </div>
                  <div class="col-md-7 col-sm-7 col-lg-7">
                    <input class="form-control" required type="text" name="publisher_name"
value="<?=$publisher['name'];?>" maxlength="150">
                  </div>
                </div>
              </li>
              <li class="list-group-item">
                <div class="row">
                  <div class="col-md-5 col-sm-5 col-lg-5">
                    <strong>Місто:</strong>
                  </div>
                  <div class="col-md-7 col-sm-7 col-lg-7">
                    <select class="form-control" name="city_id">
                      <?php if (is_array($cities)): ?>
                        <?php foreach ($cities as $city): ?>
                          <option value="<?=$city['id'];?>"
                            <?php if ($publisher['city_id']==$city['id']) echo "selected";?>
                            >
                            <?=$city['name'];?>
                          </option>
                        <?php endforeach; ?>
                      <?php endif; ?>
                    </select>
                  </div>
                </div>
              </li>
              <li class="list-group-item">
                <div class="row justify-content-center">
                  <div class="col-md-6 col-sm-6 col-lg-6">
                    <button type="submit" name="submit" class="btn btn-outline-dark btn-
block">Зберегти</button>
                  </div>
                </div>
              </li>
            </ul>
          </form>
        </div>
      </div>
    </div>
  </section>
```

```
<?php include ROOT . '/views/layouts/footer.php'; ?>
```

Лістинг файлу book/view.php

```
<?php include ROOT . '/views/layouts/header.php'; ?>

<section>
   <div class="container">
      <div class="row">

         <?php include ROOT . '/views/layouts/left_menu.php'; ?>

         <div class="col-sm-9 col-md-9 col-lg-9">
           <div class="row">
             <div class="col-sm-6 col-md-6 col-lg-6">
               <img src="<?php echo Book::getImage($book['id']); ?>" alt="" style="width: 100%; height: 100%;"
/>
               <div class="row justify-content-center pt-2">
                 <div class="col-sm-6 col-md-6 col-lg-6">
                   <a href="#" class="btn btn-secondary add-to-cart btn-block" data-id="<?= $book['id']; ?>">
                   <i class="fas fa-heart"></i><span id="button-text"> Читати</span></a>
                 </div>
               </div>
             </div>
             <div class="col-sm-6 col-md-6 col-lg-6">
               <ul class="list-group">
                 <li class="list-group-item">
                   <div class="row">
                     <div class="col-md-6 col-sm-6 col-lg-6">
                       <strong>Код книги:</strong>
                     </div>
                     <div class="col-md-6 col-sm-6 col-lg-6">
                        <?= $book['id']; ?>
                     </div>
                   </div>
                 </li>
                 <li class="list-group-item">
                   <div class="row">
                     <div class="col-md-6 col-sm-6 col-lg-6">
                       <strong>Назва книги:</strong>
                     </div>
                     <div class="col-md-6 col-sm-6 col-lg-6">
                        <?= $book['name']; ?>
                     </div>
                   </div>
                 </li>
                 <li class="list-group-item">
                   <div class="row">
                     <div class="col-md-6 col-sm-6 col-lg-6">
                       <strong>Жанр книги:</strong>
                     </div>
                     <div class="col-md-6 col-sm-6 col-lg-6">
                        <?= $book['genre_name']; ?>
                     </div>
                   </div>
                 </li>
                 <li class="list-group-item">
                   <div class="row">
                     <div class="col-md-6 col-sm-6 col-lg-6">
                       <strong>Видавництво:</strong>
                     </div>
                     <div class="col-md-6 col-sm-6 col-lg-6">
                        <?= $book['publisher_name']; ?>
                     </div>
```

```
                    </div>
                  </li>
                  <li class="list-group-item">
                    <div class="row">
                      <div class="col-md-6 col-sm-6 col-lg-6">
                       <strong>Рік видання:</strong>
                      </div>
                      <div class="col-md-6 col-sm-6 col-lg-6">
                        <?= $book['year']; ?>
                      </div>
                    </div>
                  </li>
                  <li class="list-group-item">
                    <div class="row">
                      <div class="col-md-6 col-sm-6 col-lg-6">
                       <strong>Кількість сторінок:</strong>
                      </div>
                      <div class="col-md-6 col-sm-6 col-lg-6">
                        <?= $book['pages']; ?>
                      </div>
                    </div>
                  </li>
                </ul>
              </div>
            </div>
          </div>
        </div>
      </div>
    </section>
    <?php include ROOT . '/views/layouts/footer.php'; ?>
```

Лістинг файлу cabinet/book_list.php

```
<?php include ROOT . '/views/layouts/header.php'; ?>

<section>
  <div class="container">
    <div class="row justify-content-center">

      <div class="col-sm-9 padding-right">
        <div class="features_items">
          <h2 class="title text-center">Заброньовані книги</h2>

          <?php
            if (isset($_SESSION['errors'])) {
              $errors = $_SESSION['errors'];
              if (is_array($errors))  {
               echo '<div class="alert alert-danger" role="alert"><ul>';
              foreach ($errors as $error) {
                echo '<li>'. $error .'</li>';
              }
              echo '</ul></div>';
              unset($_SESSION['errors']);
            }
          }
          if (isset($_SESSION['messages'])) {
            $messages = $_SESSION['messages'];
            if (is_array($messages))  {
             echo '<div class="alert alert-success" role="alert"><ul>';
            foreach ($messages as $message) {
              echo '<li>'. $message .'</li>';
            }
            echo '</ul></div>';
            unset($_SESSION['messages']);
```

```
            }
        }
    ?>

    <?php if (!empty($orderItems) && !empty($items)): ?>
        <p><strong>Номер бронювання: </strong>#<?=$orderItems['res_id'];?></p>
        <p><strong>Дата створення: </strong><?=$orderItems['order_date'];?></p>
        <table id="book-info" class="table-bordered table-striped table">
            <tr>
                <th>Код книги(ISBN)</th>
                <th>Назва</th>
                <th>Автор</th>
                <th>Видавництво</th>
            </tr>
        <tbody class="table-body">
            <?php foreach ($items as $item): ?>
                <tr class="data">
                    <td id="book_id"><?php echo $item['id'];?></td>
                    <td>
                        <a href="/product/<?php echo $item['id'];?>">
                            <?php echo $item['name'];?>
                        </a>
                    </td>
                    <td><?php echo $item['author_name'] ." ". $item['author_surname'];?></td>
                    <td><?php echo $item['publisher_name'];?></td>
                </tr>
            <?php endforeach; ?>
        </tbody>
        </table>
        <a class="btn btn-secondary mx-2" href="/cabinet/delete/order"> Видалити бронювання</a>
    <?php else: ?>
        <p>На жаль, нічого немає. <i class="far fa-frown-open"></i></i></p>
    <?php endif; ?>
    <a class="btn btn-secondary" href="/cabinet"> Назад</a>

            </div>
        </div>
    </div>
</div>
</section>

<?php include ROOT . '/views/layouts/footer.php'; ?>
```

Лістинг файлу cabinet/index.php

```
<?php include ROOT . '/views/layouts/header.php'; ?>

<section>

    <div class="row justify-content-center">
        <h2 class="title text-center mb-4">Особистий кабінет</h2>
    </div>

<div class="row justify-content-center">
    <div class="col-md-4 col-sm-4 col-lg-4" align="center">
        <div>
            <img class="rounded" src="/template/images/user/<?= $user['photo']; ?>" height="256px"
width="256px">
                        <form enctype="multipart/form-data" action="/cabinet/change/photo" method="post">
            <div class="row justify-content-center py-2">
                                <div align="center" class="col-5">
                                    <input required class="form-control" type="file" name="photo">
                                </div>
            </div>
```

```
                                    <div class="row justify-content-center py-2">
                                        <div align="center" class="col-5">
                                            <button class="btn btn-secondary btn-block"
type="submit">Замінити фото</button>
                                        </div>
                                    </div>
                                </form>

        <?php if (!empty($orderItems)): ?>
        <form action="/cabinet/order" method="post">
        <div class="row justify-content-center py-5 my-5">
          <div align="center" class="col-5">
            <strong>Заброньовані книги:</strong>
            <select class="form-control my-3" name="order_id">
              <?php foreach ($orderItems as $order): ?>
                <option value="<?=$order['res_id'];?>">
                  <?=$order['res_id']. ' - '. $order['order_date'];?>
                </option>
              <?php endforeach; ?>
            </select>
            <button class="btn btn-outline-success btn-block my-3" type="submit">Переглянути</button>
          </div>
        </div>
       </form>
        <?php endif; ?>
           </div>
  </div>


  <div class="col-md-4 col-lg-4 col-sm-4">

   <?php
     if (isset($_SESSION['errors'])) {
         $errors = $_SESSION['errors'];
         if (is_array($errors))  {
          echo '<div class="alert alert-danger" role="alert"><ul>';
         foreach ($errors as $error) {
            echo '<li>'. $error .'</li>';
         }
         echo '</ul></div>';
         unset($_SESSION['errors']);
       }
      }
      if (isset($_SESSION['messages'])) {
          $messages = $_SESSION['messages'];
          if (is_array($messages))  {
           echo '<div class="alert alert-success" role="alert"><ul>';
          foreach ($messages as $message) {
             echo '<li>'. $message .'</li>';
          }
          echo '</ul></div>';
          unset($_SESSION['messages']);
        }
       }
   ?>


   <form action="/cabinet/edit" method="post">
     <ul class="list-group">
       <li class="list-group-item">
         <div class="row">
           <div class="col-md-4 col-sm-4 col-lg-4">
```

```
            <strong>Електронна адреса:</strong>
          </div>
          <div class="col-md-7 col-sm-7 col-lg-7">
            <input class="form-control" required type="email" name="email" value="<?= $user['email']; ?>"
minlength="5" maxlength="100">
          </div>
        </div>
      </li>
      <li class="list-group-item">
        <div class="row">
          <div class="col-md-4 col-sm-4 col-lg-4">
            <strong>Прізвище:</strong>
          </div>
          <div class="col-md-7 col-sm-7 col-lg-7">
            <input class="form-control" required type="text" name="surname" value="<?= $user['surname'];
?>" minlength="4" maxlength="100">
          </div>
        </div>
      </li>
      <li class="list-group-item">
        <div class="row">
          <div class="col-md-4 col-sm-4 col-lg-4">
            <strong>Ім'я:</strong>
          </div>
          <div class="col-md-7 col-sm-7 col-lg-7">
            <input class="form-control" required type="text" name="name" value="<?= $user['name']; ?>"
minlength="4" maxlength="100">
          </div>
        </div>
      </li>
      <li class="list-group-item">
        <div class="row">
          <div class="col-md-4 col-sm-4 col-lg-4">
            <strong>По батькові:</strong>
          </div>
          <div class="col-md-7 col-sm-7 col-lg-7">
            <input class="form-control" required type="text" name="middle_name" value="<?=
$user['middle_name']; ?>" minlength="4" maxlength="100">
          </div>
        </div>
      </li>
      <li class="list-group-item">
        <div class="row">
          <div class="col-md-4 col-sm-4 col-lg-4">
            <strong>Телефон:</strong>
          </div>
          <div class="col-md-7 col-sm-7 col-lg-7">
            <input class="form-control" required type="text" name="phone" value="<?= $user['phone']; ?>"
pattern="[+]?([0-9]{1,3})?[0-9]{10}">
          </div>
        </div>
      </li>
      <li class="list-group-item">
        <div class="row justify-content-center">
          <div class="col-md-6 col-sm-6 col-lg-6">
            <button type="submit" name="submit" class="btn btn-outline-dark btn-block">Змінити
дані</button>
          </div>
        </div>
      </li>
    </ul>
  </form>
```

```
  <div class="page-buffer">
    <form action="/cabinet/change/password" method="post">
      <div class="row">
        <div class="col-sm-12 col-md-12 col-lg-12 py-2">
          <strong>Змінити пароль:</strong>
        </div>
        <div class="col-sm-12 col-md-12 col-lg-12 py-2">
          <input class="form-control" required type="password" name="pass" minlength="4" maxlength="32" placeholder="Новий пароль">
        </div>
        <div class="col-sm-12 col-md-12 col-lg-12 py-2">
          <input class="form-control my-container" required type="password" name="pass_repeat" minlength="4" maxlength="32" placeholder="Підтвердження паролю">
        </div>
      </div>
      <div class="row justify-content-center">
        <div class="col-sm-6 col-md-6 col-lg-6 mt-2">
          <button type="submit" name="submit" class="btn btn-outline-secondary btn-block my-container">Змінити пароль</button>
        </div>
      </div>
    </form>
  </div>
 </div>
</div>
</section>

<?php include ROOT . '/views/layouts/footer.php'; ?>
```

Лістинг файлу cart/index.php

```
<?php include ROOT . '/views/layouts/header.php'; ?>

<section>
  <div class="container">
    <div class="row">

      <?php include ROOT . '/views/layouts/left_menu.php'; ?>

      <div class="col-sm-9 padding-right">
        <div class="features_items">
          <h2 class="title text-center">Книги у кошику</h2>

          <?php
            if (isset($_SESSION['errors'])) {
              $errors = $_SESSION['errors'];
              if (is_array($errors)) {
                echo '<div class="alert alert-danger" role="alert"><ul>';
                foreach ($errors as $error) {
                  echo '<li>'. $error .'</li>';
                }
                echo '</ul></div>';
                unset($_SESSION['errors']);
              }
            }
            if (isset($_SESSION['messages'])) {
              $messages = $_SESSION['messages'];
              if (is_array($messages)) {
                echo '<div class="alert alert-success" role="alert"><ul>';
                foreach ($messages as $message) {
                  echo '<li>'. $message .'</li>';
                }
                echo '</ul></div>';
```

```
                unset($_SESSION['messages']);
              }
            }
          ?>

          <?php if ($cartItems): ?>
            <p>Ви вибрали наступні книги:</p>
            <table id="book-info" class="table-bordered table-striped table">
              <tr>
                <th>Код книги(ISBN)</th>
                <th>Назва</th>
                <th>Автор</th>
                <th>Видавництво</th>
                <th>Видалити</th>
              </tr>
              <tbody class="table-body">
              <?php foreach ($items as $item): ?>
                <tr class="data">
                  <td id="book_id"><?php echo $item['id'];?></td>
                  <td>
                    <a href="/product/<?php echo $item['id'];?>">
                      <?php echo $item['name'];?>
                    </a>
                  </td>
                  <td><?php echo $item['author_name'] . ' '. $item['author_surname'];?></td>
                  <td><?php echo $item['publisher_name'];?></td>
                  <td>
                    <a href="/cart/delete/<?php echo $item['id'];?>">
                      <i class="fa fa-times"></i>
                    </a>
                  </td>
                </tr>
              <?php endforeach; ?>
              </tbody>
            </table>
            <a class="btn btn-default checkout" href="/cart/checkout"><i class="fas fa-book-open"></i>
Забронювати книги</a>
            <a class="btn btn-default checkout" href="/cart/clear"><i class="fa fa-trash"></i> Очистити
кошик</a>
          <?php else: ?>
            <p>Ви нічого не вибрали. <i class="far fa-frown-open"></i></p>

            <a class="btn btn-default checkout" href="/"><i class="fas fa-home"></i> Головна сторінка</a>
          <?php endif; ?>

        </div>
      </div>
    </div>
  </div>
</section>
<?php include ROOT . '/views/layouts/footer.php'; ?>
```

Лістинг файлу header.php

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Електронна бібліотека</title>

    <link href="/template/css/bootstrap.min.css" rel="stylesheet" type="text/css">
    <link href="/template/css/all.min.css" rel="stylesheet" type="text/css">
    <link href="/template/css/my_styles.css" rel="stylesheet" type="text/css">
```

```
    <link href="/template/css/bootstrap-toggle.min.css" rel="stylesheet" type="text/css">
  </head>

  <body class="d-flex flex-column">
  <div id="page-content">
      <div class="container">
          <div class="row">
                  <div class="col-md-5 col-sm-5 col-lg-5">
                    <div class="float-left">
                      <a          class="site-title"          href="/"><h2          class=""><strong>Електронна
бібліотека</strong></h2></a>
                    </div>
                  </div>
                  <div class="col-md-7 col-sm-7 col-lg-7">

                    <div class="shop-menu float-right">
                      <ul class="nav">

                        <?php if (User::isAdmin()): ?>
                          <li  class="nav-item"><a  class="nav-link"  href="/admin/"><i  class="fas  fa-
crown"></i></i> Панель адміністратора</a></li>
                        <?php endif; ?>

                        <li class="nav-item"><a class="nav-link" href="/cart">
                            <i class="fa fa-book"></i> Вибрані книги
                            (<span id="cart-count"><?php echo Cart::itemsCount(); ?></span>)
                          </a>
                        </li>

                        <?php if (User::isGuest()): ?>
                          <li  class="nav-item"><a  class="nav-link"  href="/user/login/"><i  class="fa  fa-
lock"></i> Вхід</a></li>
                          <li class="nav-item"><a class="nav-link" href="/user/register"><i class="fa fa-user-
circle"></i> Реєстрація</a></li>
                        <?php else: ?>
                          <li  class="nav-item"><a  class="nav-link"  href="/cabinet/"><i  class="fa  fa-
user"></i> Особистий кабінет</a></li>
                          <li  class="nav-item"><a  class="nav-link"  href="/user/logout/"><i  class="fa  fa-
unlock"></i> Вихід</a></li>
                        <?php endif; ?>
                      </ul>
                    </div>
                  </div>
          </div>

          <div class="row">
              <div class="col-md-12 col-sm-12 col-lg-12">
                <div class="float-left">
                  <nav class="navbar navbar-expand-lg navbar-light">
                    <ul class="navbar-nav">
                      <li class="nav-item"><a class="nav-link" href="/">Головна</a></li>
                      <li class="nav-item dropdown">
                        <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-
toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                          Бібліотека
                        </a>
                        <div class="dropdown-menu" aria-labelledby="navbarDropdown">
                          <a class="dropdown-item" href="/">Каталог книг</a>
                          <a class="dropdown-item" href="/cart/">Корзина</a>
                        </div>
                      </li>
```

```
                    <li class="nav-item"><a class="nav-link" href="/about/">Про бібліотеку</a></li>
                  </ul>
                </nav>
              </div>
            </div>
          </div>

</div>
```

Лістинг файлу footer.php

```
</div>
<footer id="sticky-footer" class="py-3">
  <div class="container">
    <div class="float-right">
    </div>
  </div>
 </footer>

<script src="template/js/popper.min.js" type="text/javascript"></script>
<script src="template/js/jquery.min.js" type="text/javascript"></script>
<script src="template/js/bootstrap.min.js" type="text/javascript"></script>
<script src="template/js/main-scripts.js" type="text/javascript"></script>
<script src="template/js/bootstrap.bundle.min.js" type="text/javascript"></script>
</body>
</html>
```

Лістинг файлу left_menu.php

```
<div class="col-sm-3 col-md-3 col-lg-3">
  <div class="left-sidebar list-group">
    <h2>Каталог</h2>
    <?php foreach ($genres as $genreItem): ?>
    <a class="list-group-item list-group-item-action" href="/category/<?php echo $genreItem['id']; ?>">
      <div id="category-item">
        <h5><?php echo $genreItem['name']; ?></h5>
      </div>
    </a>
    <?php endforeach; ?>
  </div>
</div>
```

Лістинг файлу site/about.php

```
<?php include ROOT . '/views/layouts/header.php'; ?>
<section>
  <div class="container">
    <div class="row justify-content-center">

      <div class="col-sm-8 col-md-8 col-lg-8">
        <div>
          <h2 class="title text-center mb-4">Про бібліотеку</h2>
          <p>Онлайн-бібліотеки – це архіви книг, брошур, статей тощо, які зберігаються у всесвітній павутині
          Сьогодні різноманіття таких ресурсів величезне, проте не всі архіви заслуговують на увагу.
          Аби не втрачати час і не розпорошуватися в пошуках, слід визначитися, що саме ви шукаєте.</p>
        </div>
    </div>
  </div>
 </section>
<?php include ROOT . '/views/layouts/footer.php'; ?>
```

Лістинг файлу site/index.php

```
<?php include ROOT . '/views/layouts/header.php'; ?>
```

```
<section>
  <div class="container">
    <div class="row">

      <?php include ROOT . '/views/layouts/left_menu.php'; ?>

      <div class="col-sm-9 col-md-9">
        <div class="items">
          <h2 class="title text-center mb-4">Нові книги</h2>

          <?php if (!empty($latestBooks)): ?>

          <?php foreach ($latestBooks as $book): ?>
            <div class="col-sm-4 col-md-4 float-left py-3" >
              <div class="item-container item-info text-center">
                <a href="/book/<?php echo $book['id']; ?>">
                  <img class="item-image" src="<?php echo Book::getImage($book['id']); ?>" alt="" />
                </a>
                <div class="row justify-content-center">
                  <div class="col-12 text-truncate">
                    <?php echo $book['author_name'] . ' ' . $book['author_surname']; ?>
                  </div>
                        </div>
<div class="row justify-content-center font-weight-bold" >
                  <div class="col-12 text-truncate">
                    <?php echo $book['name']; ?>
                  </div>
                </div>
                <div class="row justify-content-center">
                  <div class="col-12">
                    <a href="#" class="btn btn-secondary add-to-cart btn-block" data-id="<?php echo
$book['id']; ?>">
          <i class="fas fa-heart"></i><span id="button-text"> Читати</span></a>
                  </div>
                </div>
              </div>
            </div>
          <?php endforeach; ?>
          <?php else: ?>
            <p>Нічого не знайдено. <i class="far fa-frown-open"></i></p>
          <?php endif; ?>
          </div>

        <div class="col-sm-12 col-md-12 col-lg-12 offset-4 float-left pt-4">
          <?php echo $pagination->get();?>
        </div>

      </div>
    </div>
  </section>

<?php include ROOT . '/views/layouts/footer.php'; ?>
```

Лістинг файлу login.php

```
<?php include ROOT . '/views/layouts/header.php'; ?>

<section>
  <div class="container">
          <?php if (isset($errors) && is_array($errors)): ?>
            <ul>
              <?php foreach ($errors as $error): ?>
                <li> - <?php echo $error; ?></li>
```

```
            <?php endforeach; ?>
          </ul>
        <?php endif; ?>
        <div class="signup-form">
          <h2 class="title text-center mb-4">Вхід на сайт</h2>

          <form action="#" method="post">
            <div class="row justify-content-center">
              <div class="col-12 col-md-5 pb-2">
                <input type="email" class="form-control" required name="email" placeholder="E-mail"
value="<?php echo $email; ?>"/>
              </div>
            </div>
            <div class="row justify-content-center">
              <div class="col-12 col-md-5 pb-2">
                <input type="password" class="form-control" required name="password" placeholder="Пароль"
value="<?php echo $password; ?>"/>
              </div>
            </div>
            <div class="row justify-content-center">
              <div class="col-6 col-md-2">
                <input type="submit" class="btn btn-secondary btn-block" name="submit" value="Вхід" />
              </div>
            </div>
          </form>
        </div>
    </div>
</section>

<?php include ROOT . '/views/layouts/footer.php'; ?>
```

Лістинг файлу register.php

```
<?php include ROOT . '/views/layouts/header.php'; ?>

<section>
    <div class="container">
        <?php if ($result): ?>
            <p>Ви зареєстровані!</p>
        <?php else: ?>

            <?php if (isset($errors) && is_array($errors)): ?>
                <div class="alert alert-danger" role="alert">
                    <ul>
                        <?php foreach ($errors as $error): ?>
                            <li><?php echo $error; ?></li>
                        <?php endforeach; ?>
                    </ul>
                </div>
            <?php endif; ?>


            <div class="signup-form">
              <h2 class="title text-center mb-4">Реєстрація</h2>

                <form action="#" method="post">
                    <div class="row justify-content-center">
                        <div class="col-12 col-md-5 pb-2">
                            <input type="text" class="form-control" required name="name" placeholder="Ім'я"
value="<?php echo $name; ?>"/>
                        </div>
                    </div>
                    <div class="row justify-content-center">
                        <div class="col-12 col-md-5 pb-2">
```

```
                    <input type="text" class="form-control" required name="surname" placeholder="Прізвище" value="<?php echo $surname; ?>"/>
                  </div>
                </div>
                <div class="row justify-content-center">
                  <div class="col-12 col-md-5 pb-2">
                    <input type="text" class="form-control" required name="middle_name" placeholder="По батькові" value="<?php echo $middle_name; ?>"/>
                  </div>
                </div>
                <div class="row justify-content-center">
                  <div class="col-12 col-md-5 pb-2">
                    <input type="text" class="form-control" required name="phone" pattern="[+]?([0-9]{1,3})?[0-9]{10}" placeholder="Телефон" value="<?php echo $phone; ?>"/>
                  </div>
                </div>
                <div class="row justify-content-center">
                  <div class="col-12 col-md-5 pb-2">
                    <input type="email" class="form-control" required name="email" placeholder="Електронна адреса" value="<?php echo $email; ?>"/>
                  </div>
                </div>
                <div class="row justify-content-center">
                  <div class="col-12 col-md-5 pb-2">
                    <input type="password" class="form-control" required name="password" placeholder="Пароль" value="<?php echo $password; ?>"/>
                  </div>
                </div>
                <div class="row justify-content-center">
                  <div class="col-6 col-md-2">
                    <input type="submit" class="btn btn-secondary btn-block" name="submit" value="Зареєструвати" />
                  </div>
                </div>
              </form>
            </div>
          <?php endif; ?>
    </div>
</section>
<?php include ROOT . '/views/layouts/footer.php'; ?>
```

Лістинг файлу Index.php

```
<?php

ini_set('display_errors',1);
error_reporting(E_ALL);
session_start();
define('ROOT', dirname(__FILE__));
require_once(ROOT.'/components/Autoload.php');

$router = new Router();
$router->run();
```

Лістинг файлу main-scripts.js

```
$(document).ready(function(){
  $(".list-group-item-action").hover(function() {
     $(this).addClass("active")
  });
  $(".list-group-item-action").mouseleave(function() {
     $(this).removeClass("active")
  });
});
$(".add-to-cart").click(function() {
```

```
  $(this).toggleClass('btn-primary btn-secondary');

  if ($(this).hasClass("btn-secondary")) {
    $(this).children("#button-text").text(" Читати");
  } else {
    $(this).children("#button-text").text("Не читати");
  }
  let id = $(this).attr("data-id");
  $.post("/cart/addAjax/"+id, {}, function (data) {
     $("#cart-count").html(data);
  });
  return false;
});
```

Лістинг файлу my_styles.css

```
@import   url(http://fonts.googleapis.com/css?family=Roboto:400,300,400italic,500,700,100&subset=latin,cyrillic-ext,cyrillic);
@import url(http://fonts.googleapis.com/css?family=Open+Sans:400,800,300,600,700);
@import url(http://fonts.googleapis.com/css?family=Abel);

html, body {
    height: 100%;
    margin: 0;
    padding: 0;
    font-size: 16px;
}
a, a:hover {
  color: black;
  text-decoration: none;
}
a,.category-group :hover {
   font-weight: bold;
}
.site-title {
    color: black;
}
.page-wrapper {
   height: 100%;
}
a.site-title:hover, a.site-title:active {
  color: black;
  text-decoration: none;
}
/*---header----*/
.shop-menu ul li a {
    background: #FFFFFF;
    color: #696763;
    font-family: 'Roboto', sans-serif;
    padding-left: 0;
    padding-right: 15px;
}
.shop-menu ul li a:hover {
    color:#4c6bcc;
    background:#fff;
}
.mainmenu ul li a {
    color: #696763;
    font-family: 'Roboto', sans-serif;
    font-size: 17px;
    font-weight: 300;
    padding: 0;
    padding-bottom: 10px;
}
```

```
.mainmenu ul li a:hover, .mainmenu ul li a.active,  .shop-menu ul li a.active{
    background:none;
    color:#fdb45e;
}
/*----footer----*/
#page-content {
  flex: 1 0 auto;
}

#sticky-footer {
  flex-shrink: none;
  background: #f2f2f2;
}
/*-----------*/
.item-container {
   width: 200px;
   height: 300px;
   font-size: 15px;
}
.item-image {
   width: 180px;
   height: 230px;
}
.category-group {
   font-size: 5px;
}
#category-item {
  /* padding-top: 5px;
   padding-bottom: 5px;*/
}
.page-buffer {
   margin-top: 30px;
}
```