

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра комп'ютерних наук

«До захисту допущено»

В.о. завідувача кафедри

Ігор ШЕЛЕХОВ

(підпис)

01 червня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 – Комп'ютерні науки,

освітньо-професійної програми «Інформатика»

на тему: «Інформаційна система для автоматизації діяльності бібліотеки»

здобувачки групи ІН - 02 Заїченко Тетяни Вікторівни

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

Тетяна ЗАЇЧЕНКО

(підпис)

Керівник,

асистент кафедри комп'ютерних наук,

кандидат фізико-математичних наук

Олександр ВЛАСЕНКО

(підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»

В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ

(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 – Комп'ютерні науки, освітньо-професійної програми «Інформатика»
здобувача групи ІН-02 Заїченко Тетяни Вікторівни

1. Тема роботи: «Інформаційна система для автоматизації діяльності бібліотеки» затверджена наказом по СумДУ від «22» квітня 2024 р. № 0414-VI _____
2. Термін здачі здобувачем кваліфікаційної роботи до «01» червня 2024 року _____
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз проблеми предметної області, постановка й формування завдань дослідження.
2) Огляд технологій, що використовуються для створення систем автоматизації бібліотек.
3) Розробка інформаційної системи для автоматизації діяльності бібліотеки. 4) Аналіз результатів.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____
6. Консультанти до проєкту (роботи), із значенням розділів проєкту, що стосується їх _____

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «08» квітня 2024 р.

Завдання прийняв до виконання _____
(підпис)

Керівник _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	<i>Аналіз проблеми предметної області, постановка й формування завдань дослідження</i>		
2	<i>Огляд технологій, що використовуються для створення систем автоматизації бібліотек</i>		
3	<i>Розробка інформаційної системи для автоматизації діяльності бібліотеки</i>		
4	<i>Аналіз отриманих результатів</i>		
5	<i>Оформлення пояснювальної записки до кваліфікаційної роботи</i>		

Здобувач вищої освіти _____
(підпис)

Керівник _____
(підпис)

АНОТАЦІЯ

Записка: 82 стр., 65 рис., 1 додаток, 20 використаних джерел.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є актуальною, оскільки дозволяє бібліотеці оптимізувати процеси обробки та зберігання інформації, покращити обслуговування відвідувачів бібліотеки та забезпечить більшу точність даних за допомогою інформаційної системи.

Об’єкт дослідження – інформаційна система для бібліотекарів з метою управління книгами та користувачами.

Мета роботи – розробка інформаційної системи для бібліотекарів з метою управління книгами та користувачами за допомогою мови програмування Python.

Методи дослідження – ознайомлення з можливими способами реалізації.

Результати – Розроблено інформаційну систему, яка дозволяє відображати каталог книг, додавати нові книги, редагувати інформацію про книги, видаляти книги, відображати відвідувачів бібліотеки, додавати нових відвідувачів, редагувати інформацію про відвідувачів, видаляти відвідувачів, видавати книги відвідувачу та переглядати список виданих книг.

ІНФОРМАЦІЙНА СИСТЕМА, ОПТИМІЗАЦІЯ ПРОЦЕСІВ БІБЛІОТЕКИ,
PYTHON, PYQT, POSTGRESQL.

ЗМІСТ

ВСТУП	5
1. ІНФОРМАЦІЙНИЙ ОГЛЯД.....	7
1.1. Застосування інформаційних технологій в бібліотечній сфері	7
1.2. Вимоги до проекту	8
1.3. Постановка задачі.....	10
2. ВИБІР МЕТОДІВ РОЗВ'ЯЗАННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ.....	11
2.1. Мова програмування.....	11
2.2. Середовище програмування.....	13
2.3. Бібліотека для графічного інтерфейсу	15
2.4. Система керування базами даних	16
3. РОЗРОБКА ІНФОРМАЦІЙНОЇ МОДЕЛІ.....	19
3.1. Проектування бази даних	19
3.2. Тестування додатку	21
3.3. Огляд додатку	21
ВИСНОВКИ.....	34
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	35
ДОДАТОК А ЛІСТИНГ КОДУ	37

ВСТУП

Інформаційні технології відіграють ключову роль у сучасному світі, надаючи незрівнянні можливості для покращення та оптимізації діяльності у різних сферах життя. Вони стали невід'ємною складовою нашого повсякденного існування, проникаючи у майже всі аспекти нашого життя, від освіти та розваг до комунікації та виробництва. Завдяки інформаційним технологіям ми отримуємо доступ до величезного обсягу інформації за лічені секунди, зможемо швидко та ефективно виконувати завдання, які раніше займали б значно більше часу. Технології полегшують комунікацію, дозволяючи нам спілкуватися з людьми по всьому світу у режимі реального часу через електронні листи, соціальні мережі, чати та відеоконференції.

Однією з найважливіших галузей, яка стоїть перед завданням використання сучасних технологій для підвищення ефективності, є бібліотечна справа. В сучасному світі бібліотеки відіграють важливу роль у забезпеченні доступу до знань та культурних цінностей. З урахуванням тенденцій розвитку інформаційних технологій та зростаючих потреб користувачів, створення настільного додатку для бібліотеки є актуальним завданням. Такий додаток може стати потужним інструментом управління та сприяти оптимізації робочих процесів та підвищенню якості обслуговування.

Актуальність роботи визначається тим, що з розвитком інформаційних технологій бібліотеки повинні адаптуватися до нових умов, щоб задовольнити потреби сучасних користувачів. Створення настільного додатку для бібліотеки дозволить автоматизувати рутинні операції, покращити облік бібліотечних ресурсів та підвищити якість надання послуг.

Об'єкт дослідження – процес розробки настільного додатку для бібліотеки.

Метою роботи є розробка настільного додатку для бібліотеки з метою полегшення процесів обліку та надання послуг користувачам. Цей додаток буде спрямований на автоматизацію рутинних операцій, забезпечення доступу

до бібліотечних ресурсів та підвищення ефективності взаємодії з користувачами.

Гіпотеза дослідження полягає в тому, що використання настільного додатку для автоматизації бібліотечних процесів сприятиме підвищенню ефективності роботи бібліотек та покращенню обслуговування користувачів.

Новизна фінального результату полягає у розробленні настільного додатку, який стане важливим інструментом у роботі бібліотеки, підвищуючи її ефективність і зручність обслуговування користувачів. Цей додаток буде корисним як для індивідуального управління обліком ресурсів, так і для великих бібліотек з численними відвідувачами.

Структура роботи складається зі вступу, огляду сучасних технологій управління бібліотеками, постановки задачі, вибору мов програмування, бібліотек та СУБД для реалізації додатку, практичної реалізації та огляду використання програмного додатку, висновків, списку використаних джерел та додатків.

1. ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1. Застосування інформаційних технологій в бібліотечній сфері

За останні десятиліття бібліотеки, які колись були традиційними сховищами знань, зазнали значної трансформації, завдяки інтеграції інформаційних технологій. Поява цифрових інструментів докорінно змінила спосіб функціонування бібліотек, розширивши їхню роль, перетворивши з простого сховища книг на динамічні центри поширення інформації, сприяння науковим дослідженням і залучення громадськості до участі в суспільному житті. Така трансформація полягає не просто в переході від друкованих видань до цифрових, а в глибоких змінах у доступності інформації, еволюції бібліотечних послуг і, що найважливіше, у переосмисленні ролі бібліотекарів. Бібліотеки здавна були хранителями знань, але з появою передових технологій, таких як онлайн бази даних, електронні книги та цифрові архіви, вони стали доступнішими, ніж раніше. Завдяки системам управління електронними ресурсами бібліотеки можуть ефективно організовувати і надавати контент, гарантуючи, що користувачі зможуть легко знаходити і отримувати потрібну їм інформацію.

Технології автоматизації впорядкували різні бібліотечні процеси, що призвело до підвищення ефективності та оптимізації ресурсів [18]. Наприклад, інтегроване програмне забезпечення для управління бібліотекою дозволяє бібліотекарам керувати колекціями книг, відстежувати дані про обіг і аналізувати моделі використання, що сприяє ухваленню обґрунтованих рішень і розподілу ресурсів.

Хоча інформаційні технології пропонують величезний потенціал для покращення бібліотечних послуг, вони також створюють виклики і проблеми. Такі питання, як цифровий розрив та проблеми конфіденційності потребують ретельної уваги. Надійні заходи кібербезпеки та протоколи управління даними

мають важливе значення для захисту приватності користувачів і захисту конфіденційної інформації.

Застосування інформаційних технологій перетворило бібліотеки на сучасні центри знань, інновацій та залучення громадськості. Використовуючи цифрові інструменти та стратегії, бібліотеки можуть підвищити доступність, ефективність та рівень обслуговування користувачів. У результаті інформаційні технології допомагають бібліотекам виконувати свою місію – надавати доступ до інформації, сприяти навчанню впродовж усього життя та розвивати цифрову епоху.

1.2. Вимоги до проєкту

Перед розробкою будь-якої інформаційної системи критично важливо встановити чіткі та обґрунтовані вимоги до проєкту. Вимоги до системи задають фундамент для всієї подальшої роботи, визначаючи як функціонал, так і межі можливостей системи [17].

Вимоги до системи поділяються на дві основні категорії: функціональні та нефункціональні. Функціональні вимоги описують безпосередньо те, які завдання та дії має виконувати система для досягнення поставлених перед нею цілей [19]. Це можуть бути, наприклад, процеси каталогізації ресурсів, управління обліковими записами користувачів, автоматизація видачі та прийому книг та інші критичні операції. Нефункціональні вимоги, у свою чергу, визначають якісні характеристики системи, такі як надійність, безпека, швидкість реакції та сумісність. Вони не вказують на конкретні функції, але є вирішальними для загальної ефективності та задоволення користувачів.

Функціональні вимоги:

1. Користувач може зареєструватися в системі за допомогою відповідної кнопки.

2. Користувач може авторизуватися до системи за допомогою відповідної кнопки.

3. Користувач може переглянути інформацію про розробника.
4. Користувач може перейти на сторінку каталогу книг за допомогою відповідної кнопки.
5. Користувач може перейти на сторінку каталогу відвідувачів бібліотеки за допомогою відповідної кнопки.
6. Користувач може повернутися зі сторінки каталогу книг до головного меню.
7. Користувач може повернутися зі сторінки каталогу відвідувачів до головного меню.
8. Користувач може на сторінці каталогу книг здійснювати пошук книг по бажаним критеріям.
9. Користувач може на сторінці каталогу книг змінювати інформацію про книгу.
10. Користувач може на сторінці каталогу книг видаляти книгу.
11. Користувач може на сторінці каталогу книг додавати нову книгу.
12. Користувач може на сторінці каталогу відвідувачів здійснювати пошук відвідувача по бажаним критеріям.
13. Користувач може на сторінці каталогу відвідувачів змінювати інформацію про відвідувача.
14. Користувач може на сторінці каталогу відвідувачів видаляти інформацію про відвідувача.
15. Користувач може на сторінці каталогу відвідувачів переглянути профіль відвідувача.
16. Користувач може на сторінці каталогу відвідувачів додати нового відвідувача.
17. Користувач може на сторінці профіля відвідувача переглядати інформацію про відвідувача.
18. Користувач може на сторінці профіля відвідувача видавати книгу відвідувачу.

19. Користувач може на сторінці профіля відвідувача переглядати список виданих книг.

1.3. Постановка задачі

Виходячи з вищезазначених вимог, основна мета даної роботи полягає у розробці інформаційної системи, що буде використовуватися для автоматизації діяльності бібліотеки. Для успішного виконання цього проєкту необхідно реалізувати наступні задачі:

1. Вивчення та аналіз поточних процесів у бібліотеці для вимог до інформаційної системи.
2. Розробка архітектури системи, що включатиме всі необхідні модулі та інтерфейси.
3. Визначення технологічного стеку для реалізації інформаційної системи, включаючи бази даних та інтерфейси користувача.
4. Розробка необхідних модулів системи, таких як каталогізація ресурсів, управління відвідувачами бібліотеки, система обліку видачі та повернення книг.
5. Проведення комплексного тестування всіх компонентів системи для виявлення та усунення можливих помилок.

При успішній реалізації, нова інформаційна система не тільки полегшить роботу бібліотекарів, але й значно підвищить ефективність обслуговування відвідувачів. Система забезпечить швидкий доступ до актуальної інформації про наявність книг у фондах бібліотеки, а також профілюватиме дані про відвідувачів, їхні запити та переваги. Це дозволить не тільки оптимізувати процес пошуку та видачі літератури, але й адаптувати бібліотечні заходи та пропозиції до конкретних інтересів і потреб користувачів. Ця технологічна інновація не тільки спростить повсякденні задачі бібліотекарів, але й значно покращить досвід користувачів, перетворюючи бібліотеку на більш інтерактивний і залучаючий центр навчання та культури.

2. ВИБІР МЕТОДІВ РОЗВ'ЯЗАННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

2.1. Мова програмування

Python – це універсальна і поширена мова програмування, відома легкістю і простим синтаксисом. Дана мова програмування підтримує декілька парадигм програмування, включаючи процедурне, об'єктно-орієнтовне та функціональне програмування, що дозволяє використовувати її для різноманітних проєктів. Велика стандартна бібліотека Python у поєднанні з потужною екосистемою сторонніх пакетів полегшує ефективне вирішення широкого спектру завдань програмування [16].

Основні можливості Python:

1. Простота та зрозумілість: Синтаксис Python інтуїтивно зрозумілий, що робить її ідеальною мовою для початківців. Структура мови сприяє написанню читабельного та легкого з обслуговуванні коду, що має вирішальне значення при масштабуванні проєктів.

2. Інтерпретована мова: Python є інтерпретованою мовою, що означає, що вона виконує код рядок за рядком. Це полегшує налагодження і підходить для швидкої розробки додатків.

3. Динамічна типізація: Python не вимагає явного оголошення типів змінних, оскільки типи визначаються під час виконання. Ця особливість спрощує код і прискорює процес розробки.

4. Велика стандартна бібліотека: Python постачається з великою стандартною бібліотекою, яка включає модулі та пакети для широкого спектру застосувань. Це означає, що можна використовувати готові функції замість того, щоб писати свої власні з нуля.

5. Відкритий вихідний код з сильною підтримкою спільноти: Оскільки Python є мовою з відкритим вихідним кодом, вона має активну спільноту, яка сприяє створенню величезної кількості модулів і продовжує розширювати свої можливості.

Переваги мови програмування Python [15]:

1. Підвищена продуктивність: Простота Python дозволяє розробникам зосередитися на вирішенні проблеми, а не на розумінні мови програмування чи налагодженні незрозумілого коду.

2. Універсальність: Python можна використовувати для веброзробки, аналізу даних, штучного інтелекту, наукових обчислень тощо. Такі фреймворки, як Flask та Django, спрощують веброзробку, бібліотеки NumPy, SciPy та Pandas є чудовим рішенням для роботи з даними, а бібліотеки Tkinter та PyQt відмінно підходять для реалізації графічних інтерфейсів [**Error! Reference source not found.**].

3. Можливість інтеграції: Python можна інтегрувати з іншими мовами програмування, такими як Java, C та C++, що дозволяє використовувати її в різних середовищах та для різних завдань.

Недоліки мови програмування Python:

1. Обмежена продуктивність: Оскільки Python є інтерпретованою мовою програмування, вона зазвичай працює повільніше, ніж скомпільовані мови, такі як C++ чи Java. Це може бути суттєвим недоліком для додатків, що потребують високої швидкості та продуктивності.

2. Використання пам'яті: Гнучкість Python та простота у використанні структур даних досягаються за рахунок більшого використання пам'яті. Для проєктів, де використання пам'яті є критичною проблемою, Python може бути не найкращим вибором.

3. Проблеми з багатопотоковістю: Python не дозволяє реалізувати повний паралелізм через глобальне блокування інтерпретатора (Global Interpreter Lock, GIL) – механізм, який запобігає одночасному виконанню декількох потоків. Це може бути перешкодою у багатопотокових та прив'язаних до процесора додатках.

Python виділяється простотою вивчення, великими бібліотеками та підтримкою спільноти, що робить її доступною для початківців і достатньо універсальною для досвідчених фахівців. Хоча вона стикається з певними

проблемами, такими як продуктивність і використання пам'яті, переваги часто переважають ці недоліки в багатьох додатках.

Python є потужним інструментом для розробки систем управління бібліотеками завдяки своїй універсальності, простоті використання та широкому спектру доступних ресурсів. Для таких проєктів Python не лише спрощує процес розробки, але й забезпечує надійність та ефективність системи. Оскільки бібліотеки продовжують оцифровувати свої процеси, Python, безсумнівно, відіграватиме ключову роль у їхній трансформації, доводячи, що він є безцінним інструментом у розвитку сучасних систем управління бібліотеками.

2.2. Середовище програмування

Інтегроване середовище розробки (IDE) – це програмне забезпечення, яке забезпечує програмістів комплексними засобами для розробки програмного забезпечення. IDE зазвичай складається з редактору вихідного коду, відлагоджувача та інструментів автоматизації збірки [11]. Для себе я обрала PyCharm від JetBrains, бо це потужний інструмент для розробки на мові програмування Python. PyCharm надає широкий спектр інструментів, які призначені підвищити продуктивність та спростити процес розробки [10].

PyCharm відомий своїм всеосяжним набором функцій, які задовольняють потреби як початківців, так і досвідчених Python-розробників:

1. Розумний редактор коду: В основі PyCharm лежить інтелектуальний редактор коду, який забезпечує підсвічування синтаксису, доповнення коду та виявлення помилок відразу. Ці функції роблять процес кодингу більш ефективним та допомагають зменшити кількість потенційних помилок.

2. Надійні інструменти відлагодження: Відладчик PyCharm підтримує покрокове виконання, точки записок та стековий аналіз, які потрібні для швидкого діагностування та виправлення помилок.

3. Інтеграція контролю версій: Завдяки вбудованій підтримці Git та інших систем контролю версій, PyCharm спрощує управління змінами та спільну роботу над проектами.

Переваги PyCharm:

1. Підвищена продуктивність: Інтелектуальні функції IDE, такі як завершення коду та автоматизований рефакторинг, значно прискорюють процес розробки та зменшують навантаження на розробників.

2. Кастомізація: PyCharm пропонує широкі можливості налаштування відповідно до індивідуальних потреб розробника, включаючи теми, плагіни та конфігурації редактора.

Недоліки PyCharm:

1. Високе використання ресурсів: PyCharm є ресурсномістким, що може призвести до зниження продуктивності на менш потужних пристроях.

2. Складність розуміння редактору: Широкий набір функцій та можливостей налаштування може бути непосильним для початківців і може збільшити час навчання у порівнянні з більш простими IDE або текстовими редакторами.

3. Вартість: Хоча PyCharm пропонує безкоштовну версію для спільноти, професійна версія, яка включає ширший набір функцій, має певну ціну.

У порівнянні з більш легкими IDE, такими як Visual Studio Code або Atom, PyCharm надає більш поглиблені інструменти спеціально для розробки на Python. Однак, для деякого компроміс між продуктивністю та простотою може призвести до вибору цих легших альтернатив, які є менш спеціалізованими, але все ще високоефективними за умови використання правильних розширень.

PyCharm є найбільш популярною IDE для програмування на Python, яка оснащена різними інструментами, призначеними для оптимізації продуктивності та впорядкування робочого процесу розробки. Завдяки своїм інтелектуальним можливостям та всебічній підтримці PyCharm є найкращим вибором для розробки на мові програмування Python. У цілому, PyCharm є

надійною платформою, яка значно розширює можливості Python розробників, незважаючи на її незначні недоліки.

2.3. Бібліотека для графічного інтерфейсу

У світі програмування на Python створення графічних інтерфейсів користувача (GUI) є поширеною вимогою для багатьох додатків. Python, з його принципами простоти та зрозумілості, пропонує кілька бібліотек для полегшення розробки графічних інтерфейсів. Серед них Tkinter та PyQt є одними з найпопулярніших [14]. Можливість Python взаємодіяти з декількома сторонніми бібліотеками робить її чудовим вибором для розробки крос-платформних настільних додатків. Tkinter, який постачається разом з Python, є стандартним інструментарієм графічного інтерфейсу, який забезпечує простий спосіб створення елементів вікна. PyQt з іншого боку, є набором підключень Python для фреймворку додатків Qt, пропонуючи повний набір елементів та можливостей графічного інтерфейсу.

Порівняння Tkinter та PyQt:

Tkinter:

1. Простота: Tkinter відомий своєю простотою. Він надає достатньо віджетів для створення стандартних вікон, діалогів, кнопок та меню.

2. Інтеграція: Будучи частиною стандартної бібліотеки Python, він не потребує додаткового встановлення, що робить його дуже доступним і легким для інтеграції в додатки.

3. Продуктивність: Хоча Tkinter підходить для малих і середніх додатків, він може бути дещо обмеженим у роботі з більш динамічними, сучасними графічними інтерфейсами з широкими функціональними можливостями.

PyQt є більш багатофункціональною альтернативою, що дозволяє створювати складні та візуально привабливі інтерфейси [5]. Завдяки даній бібліотеці, я реалізовувала графічний інтерфейс.

PyQt переносить в Python всі можливості фреймворку додатків Qt, які включають в себе наступні особливості:

1. Великий вибір віджетів: PyQt пропонує широкий спектр віджетів, включаючи розширені таблиці, дерева, списки, діаграми та графічні редактори, які недоступні в Tkinter.

2. Підтримка графіки: Включає вбудовану систему малювання, інтеграцію з OpenGL та фреймворк графічних сцен, що дозволяє створювати складні візуальні ефекти.

3. Підтримка баз даних: PyQt надає класи для інтеграції з базами даних, підтримуючи бази даних SQL, які можна безпосередньо використовувати у графічному інтерфейсі.

4. Обробка XML та JSON: Вбудована підтримка розбору та генерації XML та JSON, що полегшує роботу з вебсервісами та конфігураціями.

Вибір між Tkinter та PyQt значною мірою залежить від потреб проєкту та досвіду розробника. Однак, для більш складних і сучасних графічних інтерфейсів, PyQt надає потужні можливості, хоча і ціною більш складного процесу навчання. Обидва мають свої переваги і здатні створювати ефективні десктопні програми на Python [5].

2.4. Система керування базами даних

У сучасному світі, заснованому на даних, вибір підходящої системи керування базами даних (СКБД) має вирішальне значення для успіху будь-якого проєкту. Серед безлічі доступних варіантів було обрано PostgreSQL. Дана система управління базами даних має надійну архітектуру, великий набір функцій і непохитну прихильність до дотримання стандартів.

PostgreSQL, яку часто називають Postgres, закріпили свої позиції провідної об'єктно-реляційної системи баз даних з відкритим вихідним кодом, що отримала широке поширення в різних галузях і сферах застосування. PostgreSQL – стабільна база даних, відома своєю суворою відповідністю

стандартам SQL [7]. Вона призначений для обробки різноманітних завдань, починаючи від роботи з окремими комп'ютерами і закінчуючи великими сховищами даних або вебсервісами з великою кількістю одночасних користувачів. PostgreSQL славиться своєю розширюваністю, що є однією з його ключових особливостей. Користувачі мають можливість визначати власні типи даних, створювати власні функції і навіть писати код на різних мовах програмування без необхідності перекомпіляції бази даних. PostgreSQL реалізує MVCC (Multi-Version Concurrency Control) без блокування читання, що підвищує пропускну здатність і масштабованість системи при великих одночасних навантаженнях [20]. Окрім цього, дана система надає численні механізми цілісності даних, такі як первинні ключі, зовнішні ключі, унікальність обмеження та обмеження виключення.

Переваги PostgreSQL:

1. Відкритий вихідний код: Це дозволяє користувачам використовувати, модифікувати та впроваджувати програмне забезпечення на свій розсуд.
2. Можливість гнучкого налаштування: Завдяки відкритому вихідному коду систему можна легко налаштувати для різних сфер застосування. Ця гнучкість є безцінною для підприємств, які потребують специфічних модифікацій або функцій, недоступних у більш жорстких системах.
3. Надійна підтримка транзакцій: PostgreSQL має потужну систему підтримки транзакцій, яка забезпечує цілісність і надійність даних.

Недоліки PostgreSQL:

1. Значні витрати на продуктивність: Через великий набір функцій і складність, Postgres може не забезпечувати таку ж продуктивність, як інші бази даних, такі як MySQL, у сценаріях, що включають прості операції читання або коли висока швидкість читання є критичною.
2. Складність реплікації: PostgreSQL підтримує кілька конфігурацій реплікації, налаштування та керування реплікацією може бути складнішим у порівнянні з іншими базами даних, такими як MySQL.

Якщо порівнювати з іншими базами даних, такими як MySQL, Oracle і SQL Server, PostgreSQL виділяється своєю відповідністю стандартам SQL і підтримкою розширених функціональних можливостей SQL, таких як віконні функції і загальні табличні вирази (CTE). На відміну від MySQL, яка зазвичай більше підходить для вебдодатків, що вимагають високошвидкісних операцій читання, PostgreSQL надійніше обробляє складні запити і забезпечує цілісність даних, що робить її придатною для корпоративних додатків. Oracle і SQL Server, будучи комерційними продуктами, пропонують широку підтримку і розширені можливості, але за значну ціну. PostgreSQL пропонує альтернативу з багатьма схожими функціями, але без ліцензійних платежів [13].

3. РОЗРОБКА ІНФОРМАЦІЙНОЇ МОДЕЛІ

3.1. Проєктування бази даних

Для бази даних додатку «Library Assistant» було виділено наступні сутності:

1. Книга
2. Користувач
3. Відвідувач
4. Борг

Користувач – це працівник бібліотеки, який має доступ до додатку. Відвідувач – особа, яка користується послугами бібліотеки. Книга – це об’єкт, який призначено для читання. Борг – це інформація про видану книгу певному відвідувачу.

Нижче наведено ER-діаграму, яка була сформована у додатку PgAdmin4.

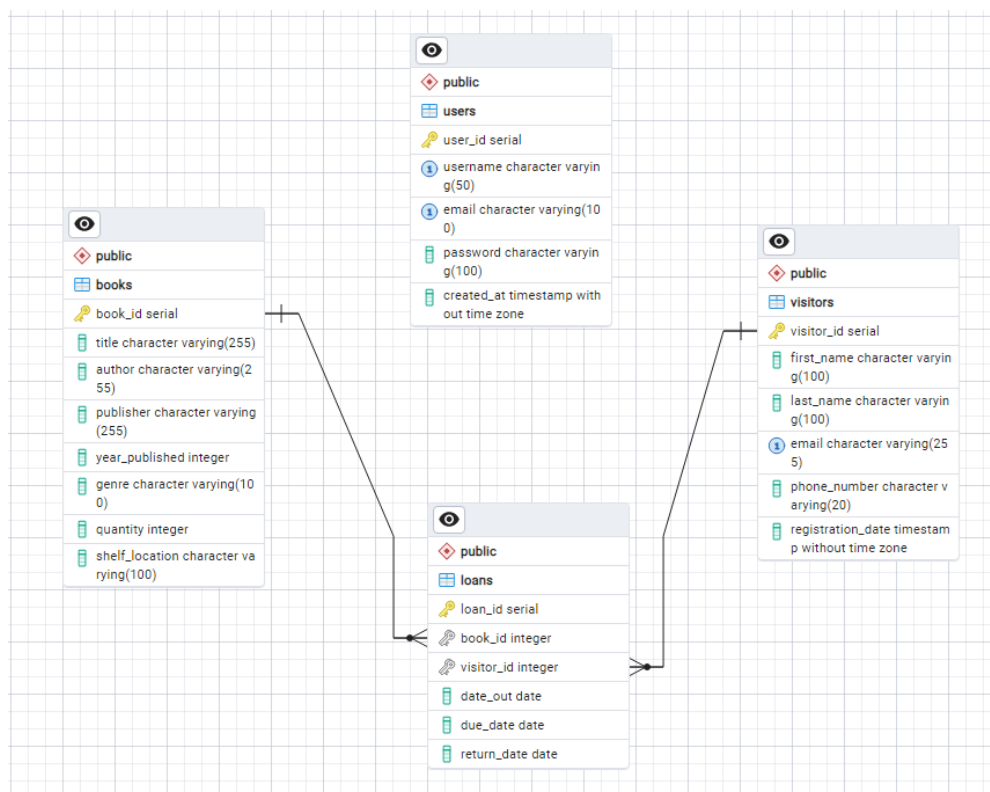


Рисунок 3.1 – ER-діаграма бази даних

Нижче наведено структуру та опис полів таблиць у базі даних.

Таблиця 3.1 – Структура та опис таблиць бази даних додатку

Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
Books	book_id	Ідентифікатор книги	serial	PK	NOT NULL, UNIQUE
	title	Назва книги	varchar		NOT NULL
	author	Автор книги	varchar		NOT NULL
	publisher	Видавництво	varchar		NOT NULL
	year_published	Рік видавництва	integer		NOT NULL
	genre	Жанр книги	varchar		NOT NULL
	quantity	Кількість	integer		NOT NULL
	shelf_location	Розташування	varchar		NOT NULL
Users	user_id	Ідентифікатор користувача	integer	PK	NOT NULL, UNIQUE
	username	Ім'я та прізвище	varchar		NOT NULL
	email	Електронна пошта	varchar		NOT NULL
	password	Пароль	varchar		NOT NULL
	created_at	Дата створення	timestamp		
Visitors	visitor_id	Ідентифікатор відвідувача	integer	PK	NOT NULL, UNIQUE
	first_name	Ім'я	varchar		NOT NULL
	last_name	Прізвище	varchar		NOT NULL
	email	Електронна пошта	varchar		NOT NULL
	phone_number	Номер телефону	varchar		NOT NULL
	registration_date	Дата реєстрації	timestamp		
Loans	loan_id	Ідентифікатор видачі	integer	PK	NOT NULL
	book_id	Ідентифікатор книги	integer	FK	NOT NULL
	visitor_id	Ідентифікатор відвідувача	integer	FK	NOT NULL
	date_out	Дата видачі	date		NOT NULL
	due_date	Дата закінчення терміну	date		NOT NULL
	return_date	Дата повернення	date		

3.2. Тестування додатку

Тестування є основою на шляху розробки будь-якого програмного додатку, оскільки є кінцевою оцінкою його функціональності, надійності та загальної продуктивності. Після завершення розробки додатку було проведено всебічне тестування з метою перевірки правильності роботи всіх компонентів програми. Тестування включало наступні етапи:

1. Перевірка інтерфейсу користувача. Було проведено детальну перевірку всіх елементів інтерфейсу, таких як кнопки, форми, меню та навігаційні панелі. В процесі тестування аналізувалася зручність та інтуїтивність використання інтерфейсу. Було перевірено, чи легко користувачам знайти необхідні функції та чи зрозумілі їм підказки та повідомлення додатку.

2. Тестування правильності роботи функціоналу. Цей етап включає в себе перевірку того, чи працюють всі функції додатку так, як очікується від них за специфікацією.

Результати всебічного тестування показали, що додаток повністю відповідає всім зазначеним вимогам. У процесі тестування не було виявлено критичних проблем або недоліків, що свідчить про високу якість, надійність та стабільність розробленого програмного продукту.

3.3. Огляд додатку

Під час запуску програми створюється вітальне вікно додатку «Library Assistant». Дане меню відповідає за автентифікацію користувача. Перед користувачем додатку з'являється дві кнопки, перша відповідає за авторизацію у систему, а друга – реєстрація нового користувача.

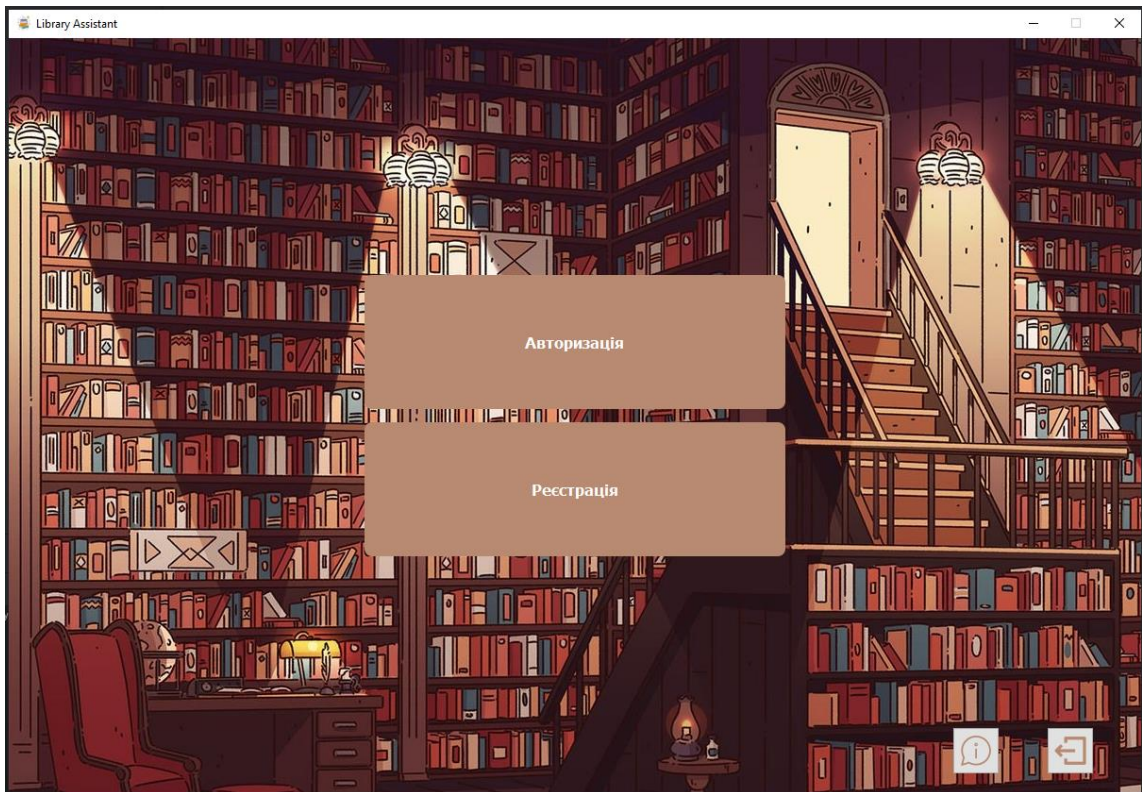


Рисунок 3.2 – Початкове меню для вибору способу автентифікації

Окрім цього, у даному вікні є дві кнопки, які розташовані у правому нижньому кутку вікна. Перша кнопка – інформація про розробника, а друга відповідає за вихід із додатку.

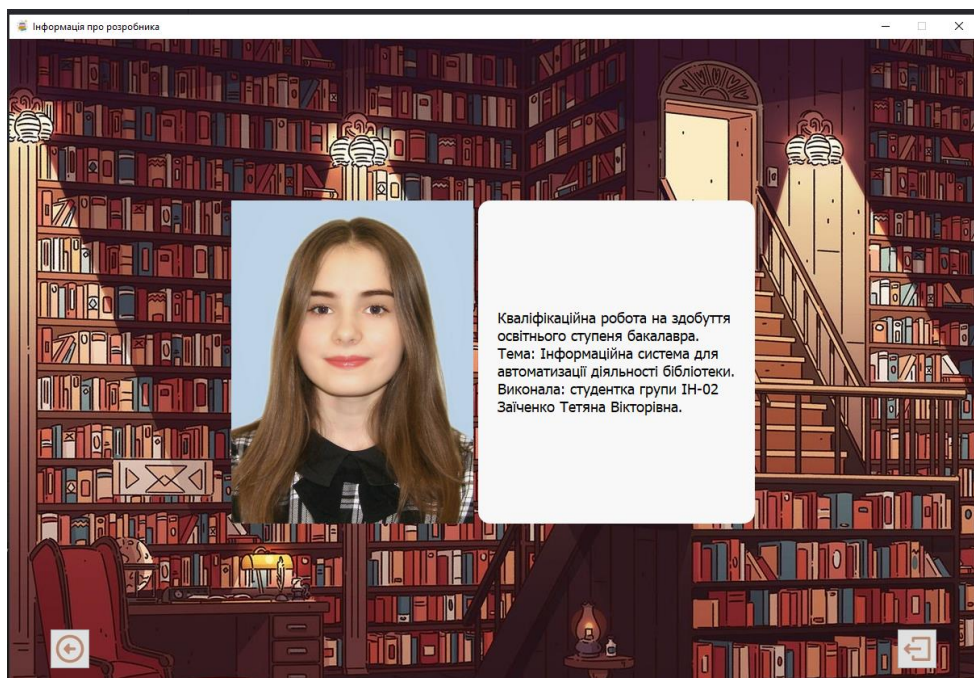


Рисунок 3.3 – Сторінка з інформацією про розробника

Після того, як користувач натисне кнопку «Авторизація», то він буде направлений на сторінку для авторизації. Користувач повинен ввести електронну пошту та пароль. Обидва поля перевіряються на коректність введених даних. Якщо користувач ввів коректні дані, які є в базі даних, бо буде виведено відповідне повідомлення, а якщо дані будуть невірними, то також буде виведено повідомлення про помилку. Після успішної авторизації користувача буде направлено до головного меню додатку. В даному вікні з'являється нова кнопка, яка знаходиться у лівому нижньому кутку, яка відповідає за повернення на початкову сторінку додатку.

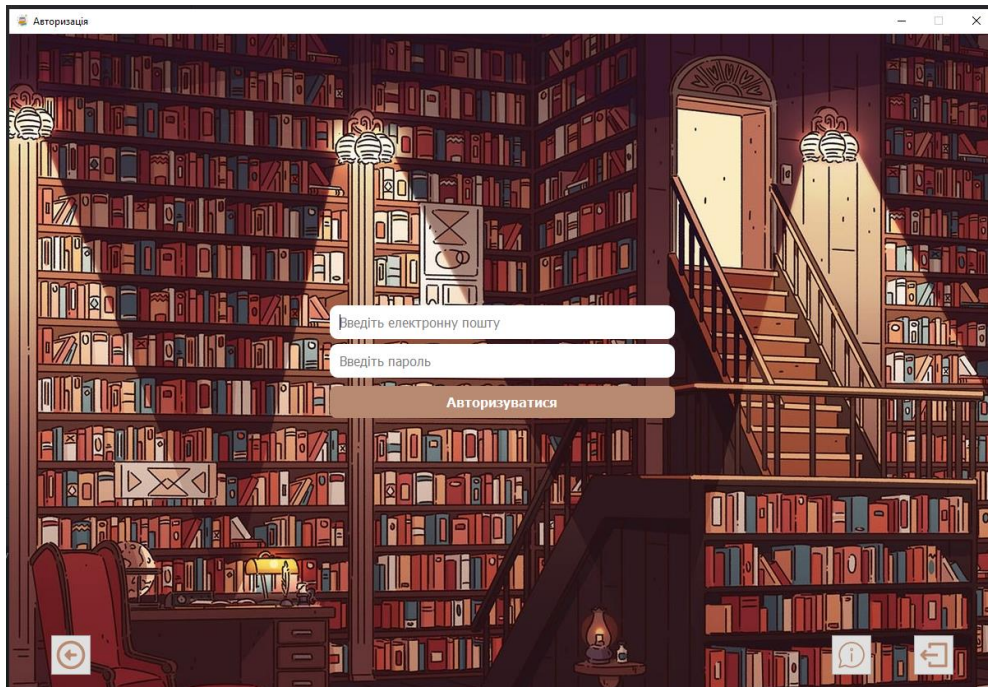


Рисунок 3.4 – Сторінка для авторизації користувача

Якщо користувач натисне кнопку «Реєстрація», то відкриється вікно, в якому користувач повинен ввести ім'я, електронну пошту та пароль. Після введення коректних даних, користувачу буде виведено повідомлення про успішну реєстрацію та відкриється вікно авторизації.

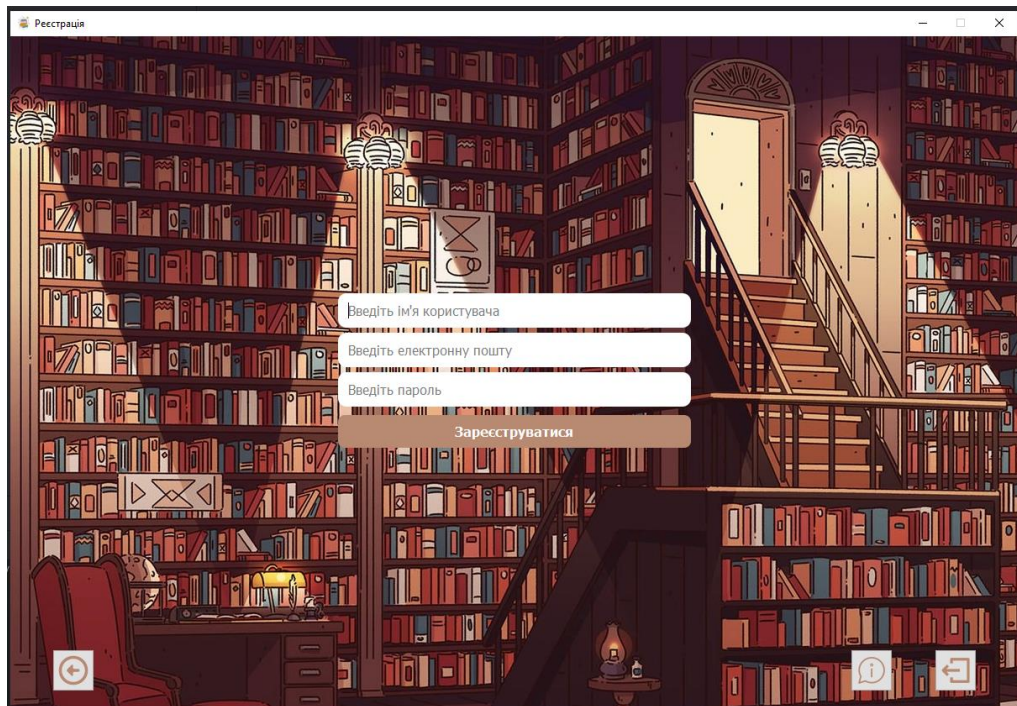


Рисунок 3.5 – Сторінка для реєстрації користувача

Головне меню додатку складається із двох основних кнопок, перша – «Каталог книг», яка перенаправляє на сторінку з реєстром книг, а друга – «Каталог користувачів», яка перенаправляє до реєстру відвідувачів бібліотеки.

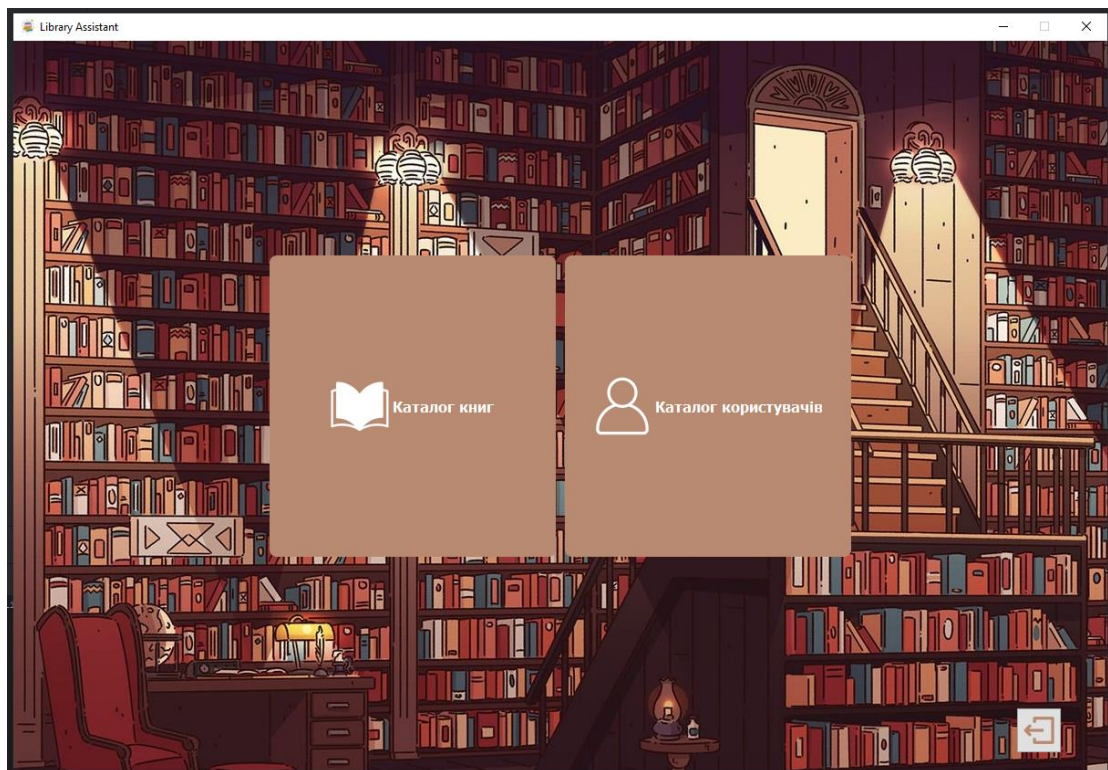


Рисунок 3.6 – Головне меню додатку

На сторінці «Каталог книг» у нас є таблиця, у якій перелічено всі наявні книги у бібліотеці. Є дві кнопки для навігації по таблиці, а саме перегортання сторінок.

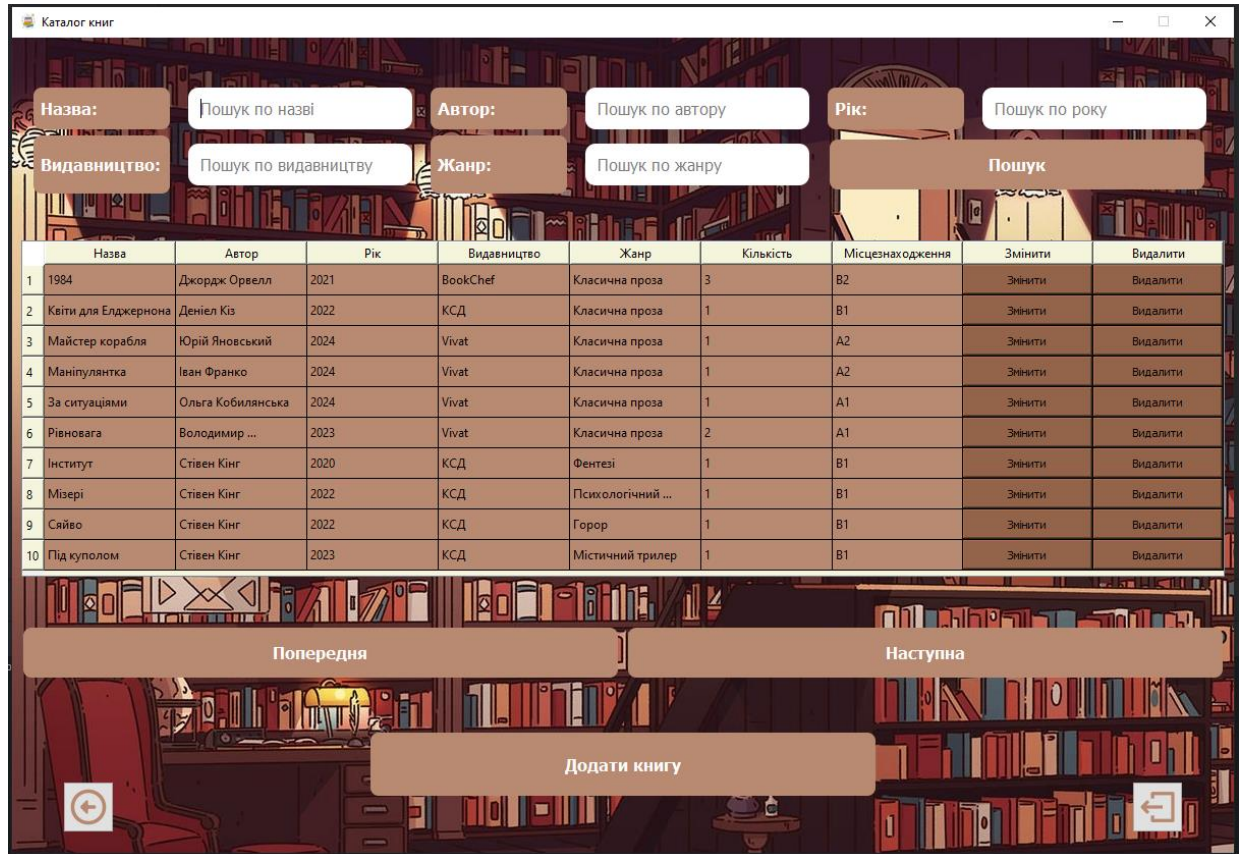


Рисунок 3.7 – Меню каталогу книг

Окрім цього на сторінці можна здійснити пошук книги по таким критеріям: назва, автор, рік видавництва, видавництво та жанр. Після введення потрібного критерію потрібно натиснути кнопку «Пошук». Якщо є збіг у базі даних із критерієм, то в таблиці буде виведено результати пошуку, а якщо результатів немає, то з такими критеріями книга відсутня у каталозі бібліотеки. Можна здійснювати пошук за декількома критеріями одночасно.

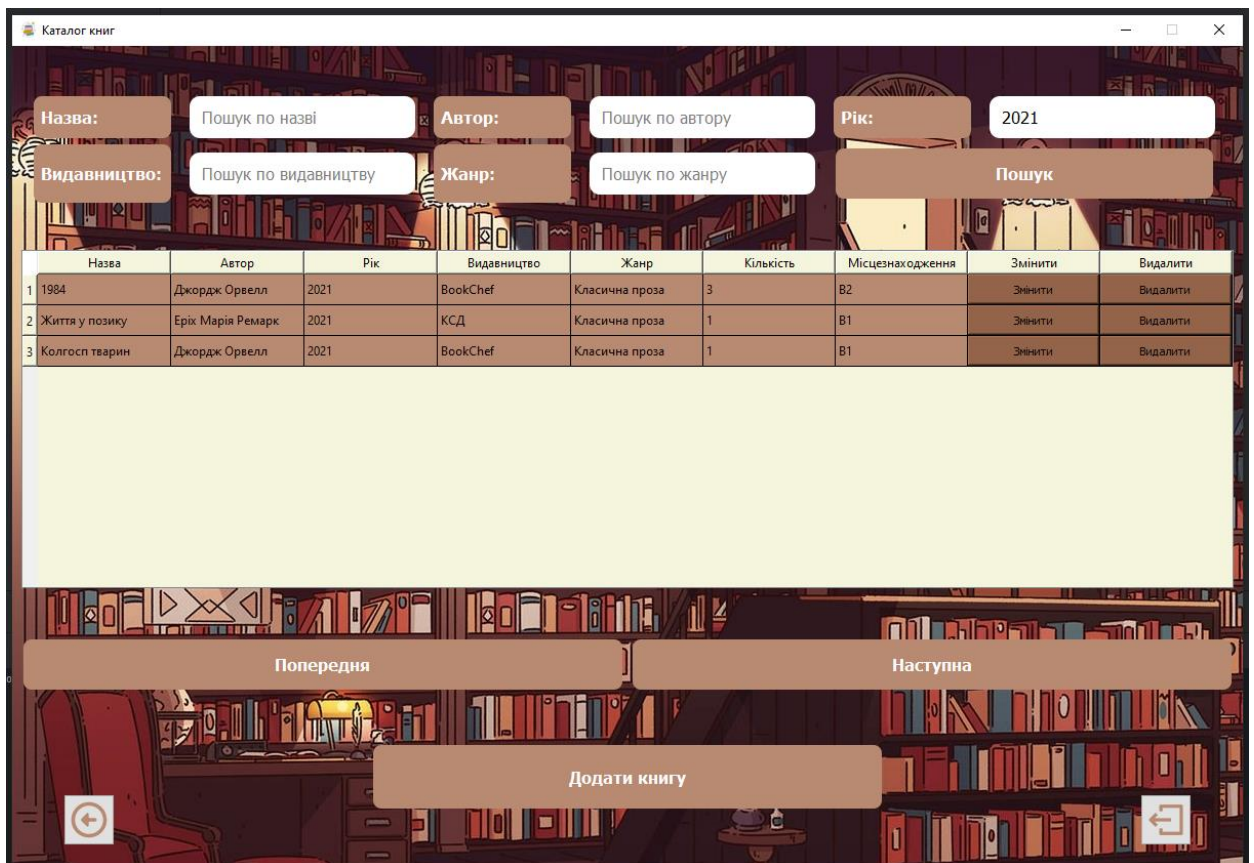


Рисунок 3.8 – Демонстрація пошуку книги за критерієм

Біля кожної книги у таблиці є дві кнопки «Змінити» та «Видалити». При натисканні першої, відкриється нове вікно, де буде виведено інформацію про книгу та можна змінити будь-яке поле, але воно повинно бути коректним. Зберігається інформація натисканням кнопки «Зберегти зміни». При натисканні другої кнопки, можна видалити книгу із каталогу, але потрібно підтвердити дію у спливаючому вікні.

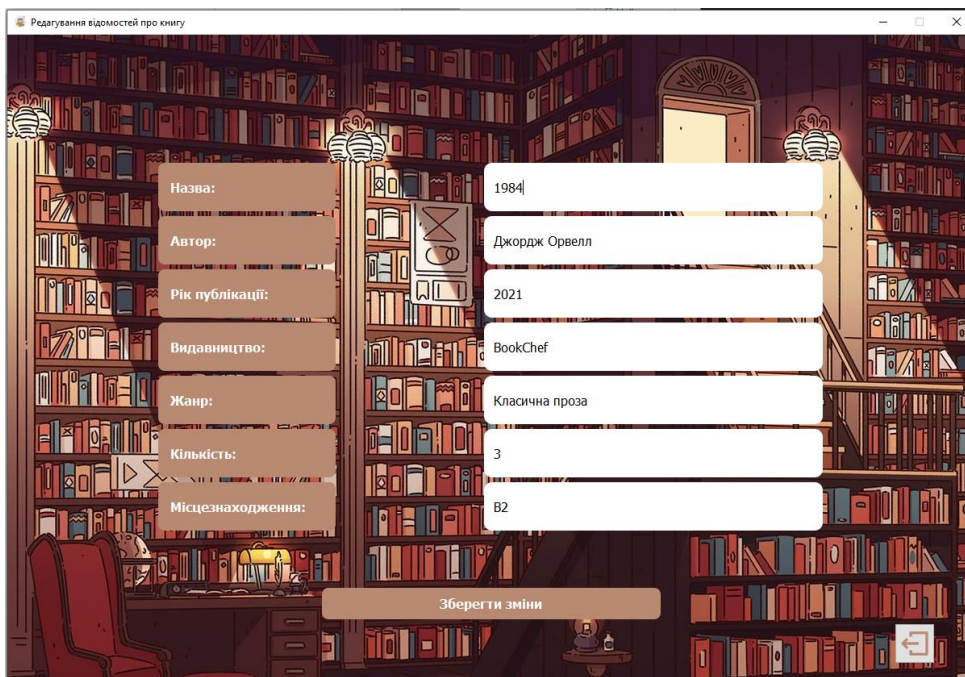


Рисунок 3.9 – Меню редагування інформації про книгу

Також на сторінці є кнопка «Додати книгу», яка відповідає за додавання нової книги до каталогу бібліотеки. Після натискання кнопки відкриється нове вікно, в якому потрібно ввести всю інформацію про книгу. Обов'язково потрібно заповнити всі поля, аби успішно додати книгу. Якщо всі дані введено коректно, то книгу буде додано до каталогу.

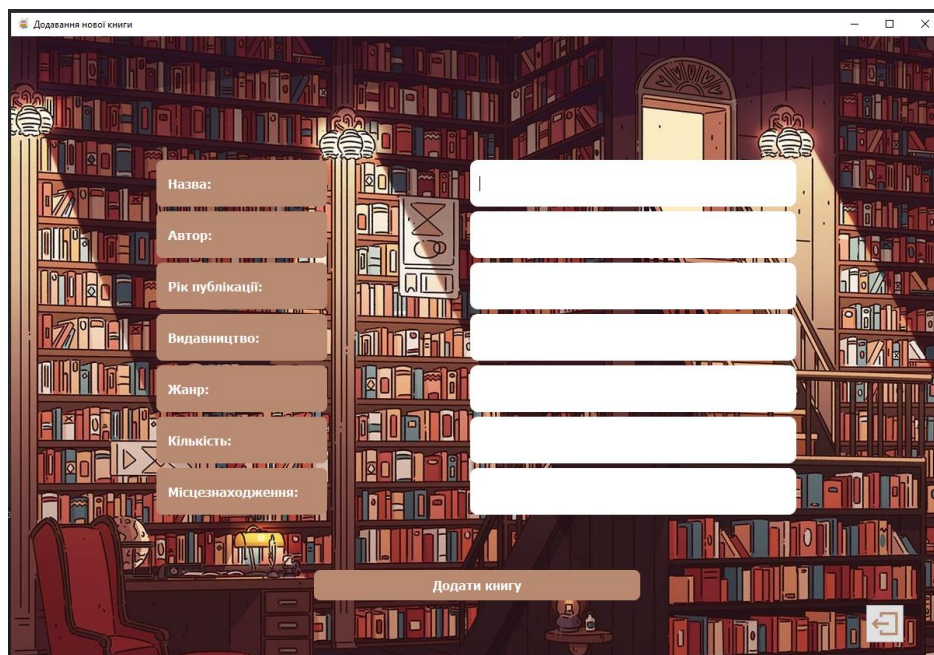


Рисунок 3.10 – Меню додавання нової книги до каталогу

Друга основна кнопка – «Користувачі бібліотеки». На даній сторінці ми маємо аналогічну сторінку з таблицею, в якій виведено інформацію про відвідувачів бібліотеки.



Рисунок 3.11 – Меню каталогу користувачів

Аналогічним чином можна здійснити пошук відвідувача за критеріями: ім'я, прізвище та електронна пошта. Пошук здійснюється за допомогою кнопки «Пошук».

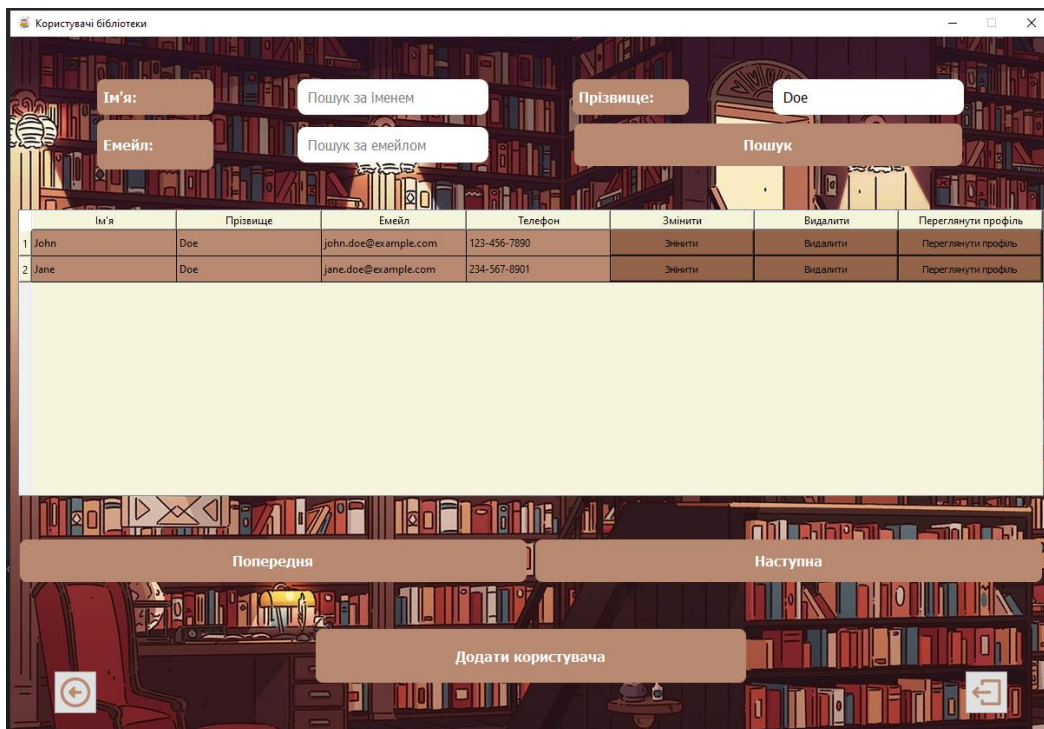


Рисунок 3.12 – Демонстрація пошуку користувачів за критерієм

Біля кожного відвідувача є кнопки «Змінити», «Видалити» та «Переглянути профіль». Перша кнопка відкриває нове вікно, де можна змінити інформацію про відвідувача і зберегти зміни за допомогою кнопки «Зберегти зміни».

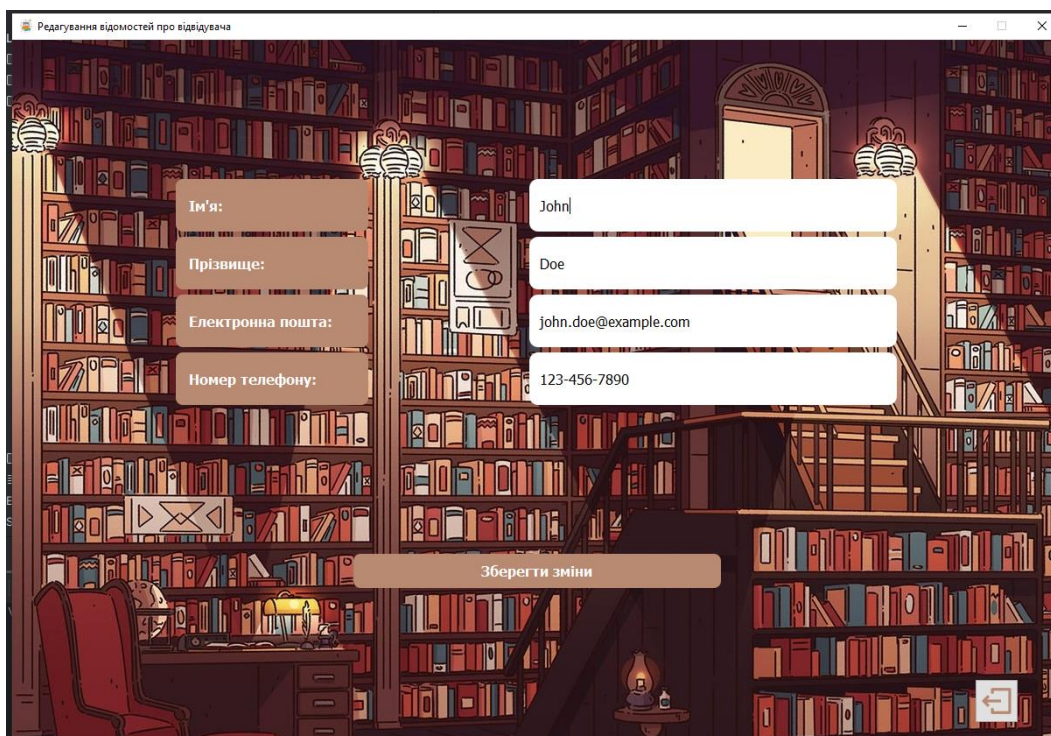


Рисунок 3.13 – Меню редагування інформації про користувача

Також на сторінці каталогу відвідувачів є кнопка для додавання нового користувача, після натискання якої відкриється нове вікно, в якому потрібно буде ввести всю необхідну інформацію про користувача. Після вірно введеної інформації потрібно натиснути кнопку «Додати відвідувача» і дані будуть додані до бази даних.

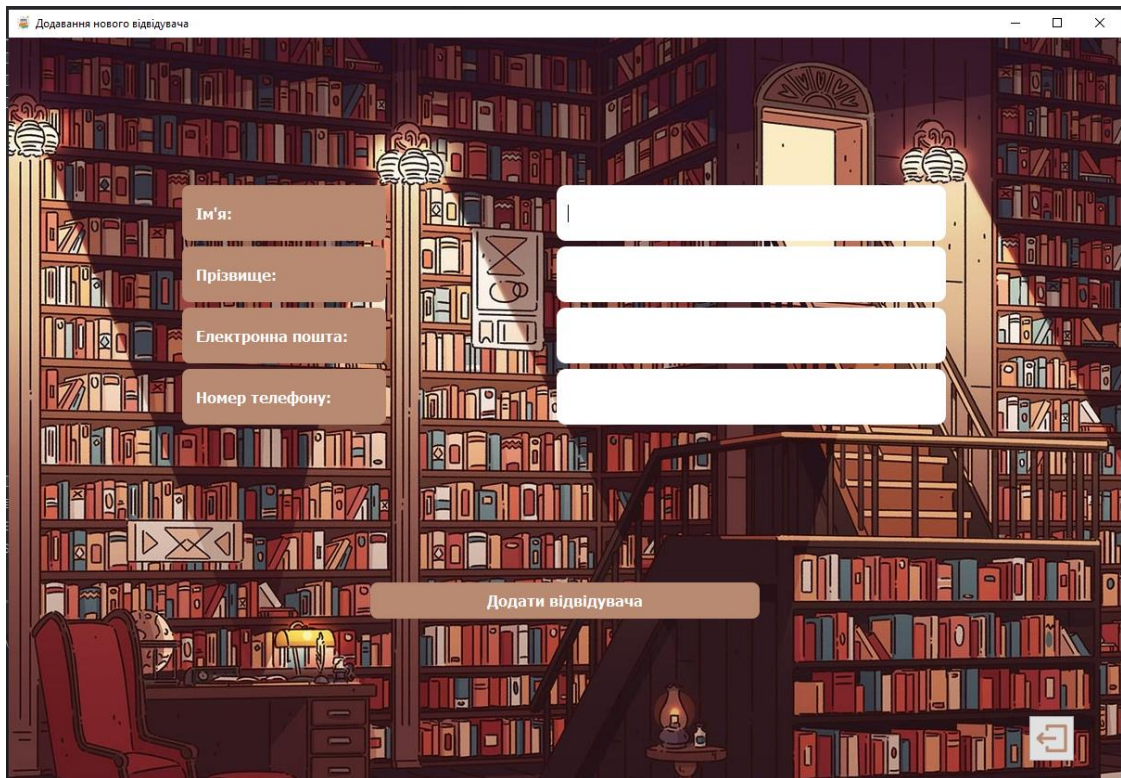


Рисунок 3.14 – Меню додавання нового користувача

Остання кнопка біля кожного користувача відповідає за відкриття профілю відвідувача. У цьому вікні у нас буде виведено всю наявну інформацію про відвідувача та дві кнопки «Видані книги» та «Видати книгу».

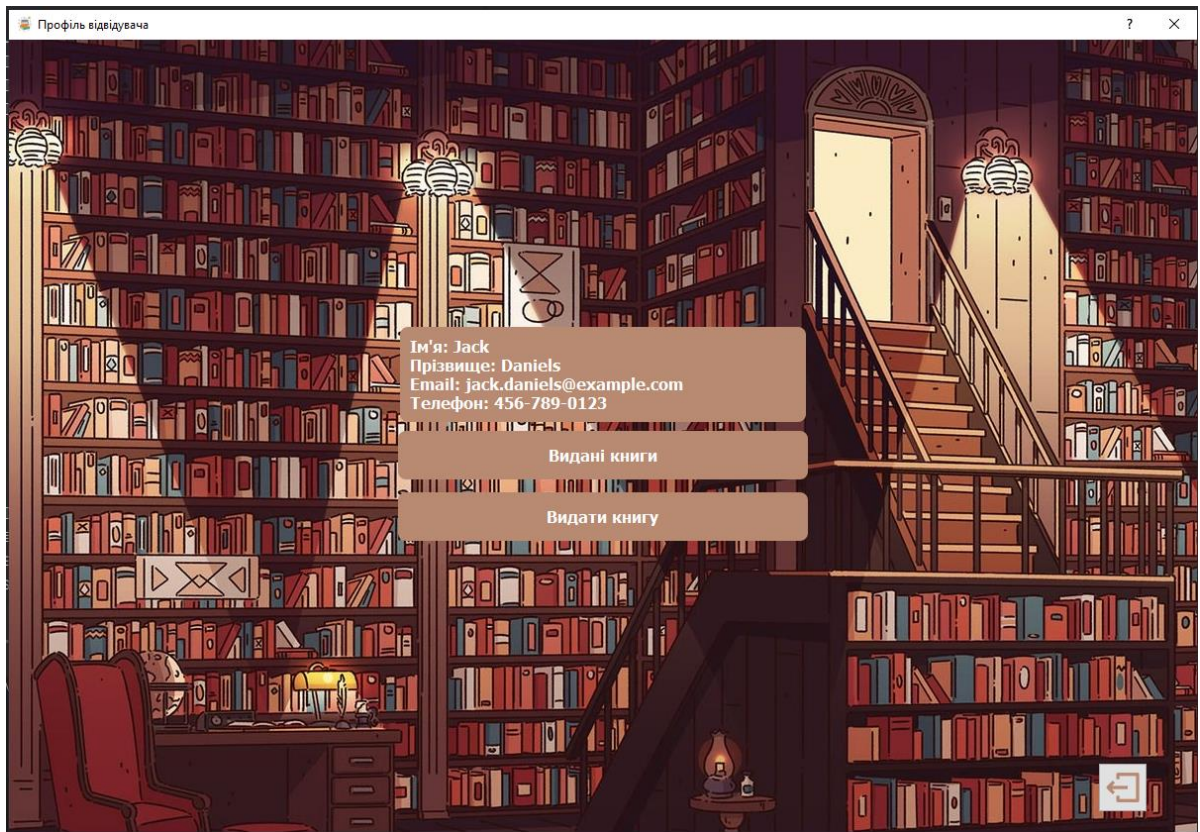


Рисунок 3.15 – Демонстрація профілю користувача

Якщо ми натискаємо на кнопку «Видати книгу» відкривається нове вікно, де із розкритого списку потрібно обрати наявну книгу у каталозі, а потім завдяки календарю обрати дату, коли потрібно повернути книгу. Після цього натиснути кнопку «Видати книгу» і книгу буде видано даному відвідувачу. При цьому книгу буде прибрано з каталогу, якщо вона в єдиному екземплярі, а якщо її декілька, то її кількість зменшиться.

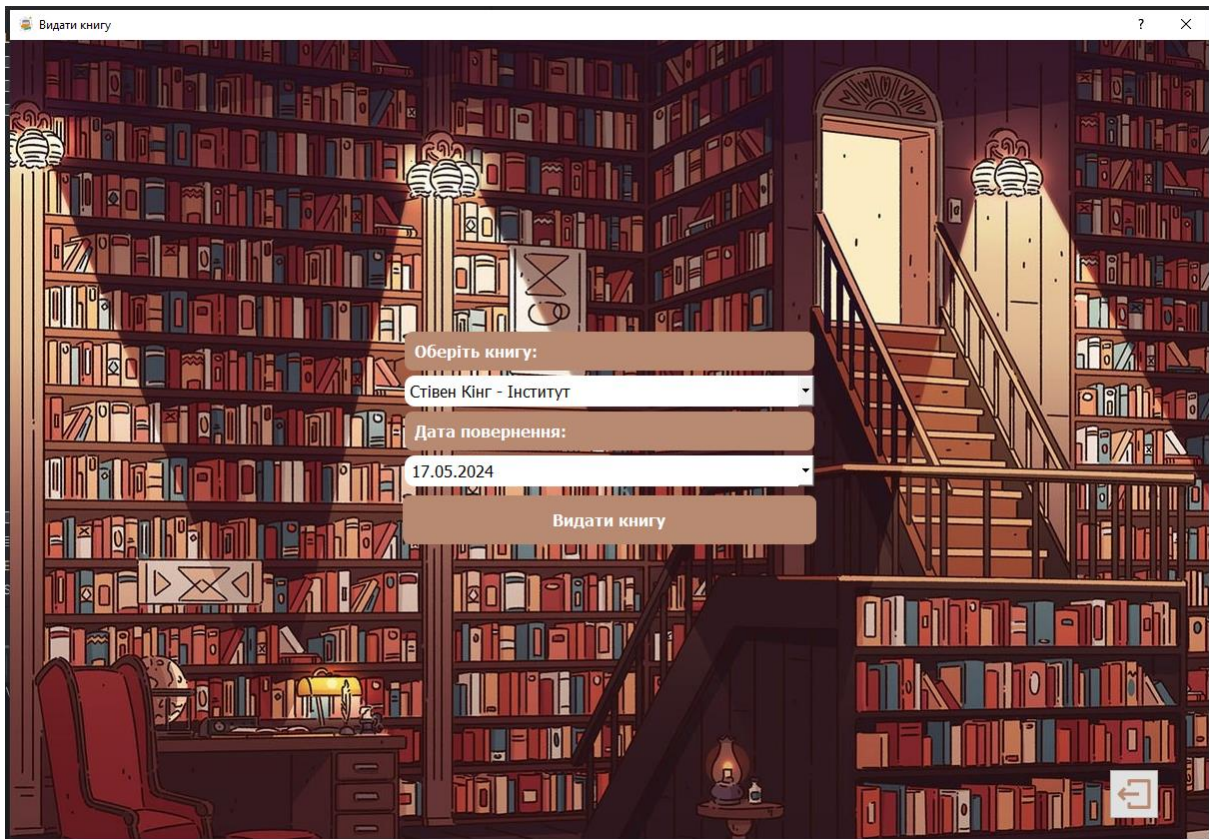


Рисунок 3.16 – Меню видачі книги певному користувачу

Аби переглянути список книг, які зараз є у відвідувача потрібно натиснути кнопку «Видані книги». Після цього у нас з'являється нове вікно, де у таблиці у нас виведено всі видані книги, а якщо таблиця порожня, то ніяких книг у відвідувача немає. Саме на цій сторінці ми можемо помітити, що книгу відвідувач повернув, це робиться за допомогою кнопки «Повернути». Після цього книжка стає знову доступною у каталозі.

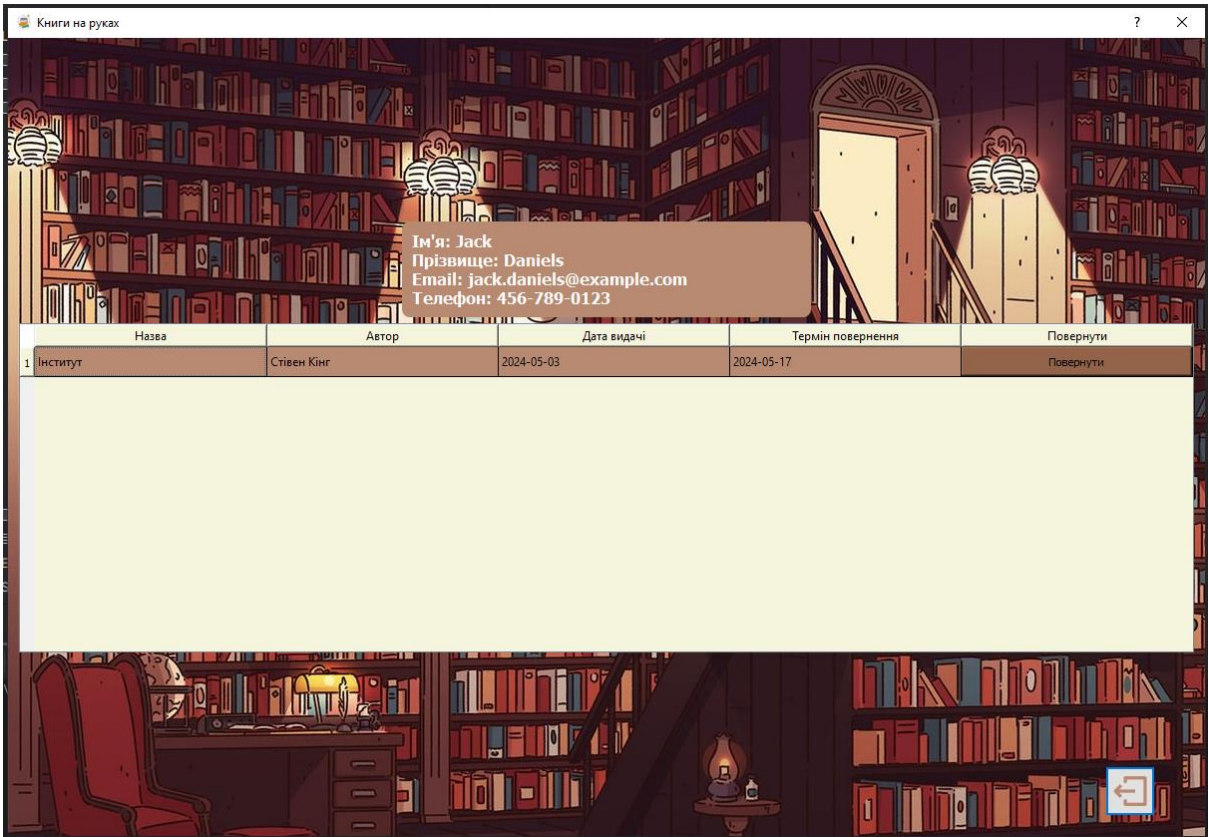


Рисунок 3.17 – Меню перегляду виданих книг певному користувачу

На цьому процес розробки і тестування застосунку є успішно закінченим.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено настільний додаток для бібліотеки з метою полегшення процесів обліку та надання послуг користувачам. Проект включав в себе аналіз поточного стану бібліотечної роботи, визначення потреб працівників бібліотек та розробку функціоналу, спрямованого на їх задоволення. Також був розроблений зручний і інтуїтивно зрозумілий інтерфейс для забезпечення зручності використання додатку користувачами будь-якого рівня експертизи.

Інформаційна система, розроблена у рамках цієї роботи, призначена для автоматизації рутинних операцій, забезпечення доступу до бібліотечних ресурсів та підвищення ефективності взаємодії з користувачами. Її створення відповідає вимогам сучасності та відображає стрімкий розвиток цифрових технологій у бібліотечній справі. Цей настільний додаток має потенціал стати важливим інструментом для підвищення ефективності та зручності обслуговування користувачів бібліотеки, а також оптимізації робочих процесів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Welcome to Python.org. Python.org. URL: <https://www.python.org/doc/> (дата звернення: 24.04.2024).
2. The Python Tutorial. Python documentation. URL: <https://docs.python.org/3/tutorial/index.html> (дата звернення: 24.04.2024).
3. Python Tutorial. W3Schools Online Web Tutorials. URL: <https://www.w3schools.com/python/default.asp> (дата звернення: 24.04.2024).
4. PyQt5 Reference Guide – PyQt Documentation v5.15.7. Riverbank Computing | News. URL: <https://www.riverbankcomputing.com/static/Docs/PyQt5/> (дата звернення: 24.04.2024).
5. Uddin Q. Python GUI : Difference between tkinter, PyQt, and Kivy. Medium. URL: <https://medium.com/@qasim.coder/python-gui-smackdown-unleashing-the-power-of-tkinter-pyqt-and-kivy-e7b05d0e862> (дата звернення: 28.04.2024).
6. Panicker A. Tkinter vs PyQt: Choosing the Right GUI Library for Your Python Projects. DEV Community. URL: <https://dev.to/abpanic/tkinter-vs-pyqt-choosing-the-right-gui-library-for-your-python-projects-1oj0> (дата звернення: 28.04.2024).
7. PostgreSQL: Documentation. PostgreSQL: The world's most advanced open source database. URL: <https://www.postgresql.org/docs/> (дата звернення: 24.04.2024).
8. PostgreSQL Tutorial. PostgreSQL Tutorial. URL: <https://www.postgresqltutorial.com/> (дата звернення: 24.04.2024).
9. PostgreSQL Tutorial. W3Schools Online Web Tutorials. URL: <https://www.w3schools.com/postgresql/index.php> (дата звернення: 24.04.2024).
10. JetBrains. PyCharm Features - JetBrains Python IDE. JetBrains. URL: <https://www.jetbrains.com/pycharm/features/#python-code-editing> (дата звернення: 26.05.2024).

11. Contributors to Wikimedia projects. Integrated development environment - Wikipedia. Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Integrated_development_environment (дата звернення: 26.05.2024).
12. Libraries in Python - GeeksforGeeks. GeeksforGeeks. URL: <https://www.geeksforgeeks.org/libraries-in-python/> (дата звернення: 26.05.2024).
13. MYSQL vs PostgreSQL vs ORACLE - ByteScout. ByteScout. URL: <https://bytescout.com/blog/mysql-vs-postgresql-vs-oracle.html#7> (дата звернення: 26.05.2024).
14. Fitzpatrick M. PyQt vs. Tkinter: Which Should You Choose for Your Next Python GUI?. Python GUIs. URL: <https://www.pythonguis.com/faq/pyqt-vs-tkinter/> (дата звернення: 26.05.2024).
15. Python Pros and Cons in 2024. Digital Acceleration Company | Netguru. URL: <https://www.netguru.com/blog/python-pros-and-cons> (дата звернення: 26.05.2024).
16. Lundh F. Python standard library. Beijing : O'Reilly, 2001. 281 с.
17. Young R. R. Project requirements: A guide to best practices. Vienna, Va : Management Concepts, 2006.
18. Murthy, T. A. V., and V. S. Cholin. Library automation, 2003.
19. Malan, R., & Bredemeyer, D. (2001). Functional requirements and use cases. Bredemeyer Consulting, 335-1653
20. Introduction MVCC. PostgreSQL Documentation. URL: <https://www.postgresql.org/docs/current/mvcc-intro.html> (дата звернення: 26.05.2024).

ДОДАТОК А ЛІСТИНГ КОДУ

```

add_book.py ×
1  import sys
2  import psycopg2
3  from PyQt5.QtCore import Qt
4  from PyQt5.QtGui import QIcon
5  from PyQt5.QtWidgets import QPushButton, QMessageBox, QMainWindow
6  import utils
7  1 usage
8  def create_connection():
9      try:
10         conn = psycopg2.connect(
11             host="localhost",
12             database="Library Assistant",
13             user="postgres",
14             password="1111")
15         return conn
16     except psycopg2.Error as e:
17         print(f"Помилка підключення до бази даних: {e}")
18         QMessageBox.critical(None, "Помилка підключення до бази даних", f"Помилка підключення до бази даних: {e}")
19         sys.exit(1)
20
21 from PyQt5.QtWidgets import QVBoxLayout, QHBoxLayout, QLabel, QLineEdit, QWidget
22 1 usage
23 class AddBookDialog(QMainWindow):
24     def __init__(self):
25         super().__init__()
26         self.setGeometry(100, 100, 1200, 800)
27         utils.set_background(self)
28         self.setWindowIcon(QIcon('src/logo.png'))
29         self.setWindowTitle('Додавання нової книги')
30         self.conn = create_connection()
31         self.initUI()
32     def initUI(self):
33         self.central_widget = QWidget(self)
34         self.setCentralWidget(self.central_widget)
35         self.layout = QVBoxLayout(self.central_widget)
36         self.layout.addSpacing(150)
37
38         self.create_form_entry('Назва:', 'title_edit')
39         self.create_form_entry('Автор:', 'author_edit')
40         self.create_form_entry('Рік публікації:', 'year_edit')
41         self.create_form_entry('Видавництво:', 'publisher_edit')
42         self.create_form_entry('Жанр:', 'genre_edit')
43         self.create_form_entry('Кількість:', 'quantity_edit')
44         self.create_form_entry('Місцезнаходження:', 'shelf_location_edit')
45
46         save_button = QPushButton('Додати книгу', self)
47         save_button.clicked.connect(self.add_book)
48         save_button.setStyleSheet(self.button_style())
49         self.layout.addWidget(save_button, alignment=Qt.AlignCenter)
50
51         exit_btn = utils.create_exit_button(self)
52         exit_btn.clicked.connect(self.close)

```

Рисунок 6.1 – 1 частина вмісту файлу add_book.py

```

add_book.py ×
50     exit_btn = utils.create_exit_button(self)
51     exit_btn.clicked.connect(self.close)
52     exit_btn.move(1100, 730)
53
54     7 usages
55     def create_form_entry(self, label_text, edit_name):
56         layout = QHBoxLayout()
57         label = QLabel(label_text, self)
58         label.setStyleSheet(self.label_style())
59         setattr(self, edit_name, QLineEdit(self))
60         getattr(self, edit_name).setStyleSheet(self.line_edit_style())
61         layout.addWidget(label)
62         layout.addWidget(getattr(self, edit_name))
63         self.layout.addLayout(layout)
64
65     1 usage
66     def add_book(self):
67         title = self.title_edit.text().strip()
68         author = self.author_edit.text().strip()
69         year = self.year_edit.text().strip()
70         publisher = self.publisher_edit.text().strip()
71         genre = self.genre_edit.text().strip()
72         quantity = self.quantity_edit.text().strip()
73         shelf_location = self.shelf_location_edit.text().strip()
74
75         if not title or not author or not year or not publisher or not genre or not quantity or not shelf_location:
76             QMessageBox.warning(self, 'Помилка', 'Будь ласка, заповніть всі поля.')
77             return
78
79         if not year.isdigit() or not quantity.isdigit() or int(quantity) < 0:
80             QMessageBox.warning(self, 'Помилка',
81                                 'Рік публікації та кількість мають бути коректними числами. Кількість не може '
82                                 'бути меншою за 0.')
83             return
84
85         year = int(year)
86         quantity = int(quantity)
87
88         try:
89             query = ("INSERT INTO books (title, author, year_published, publisher, genre, quantity, shelf_location) "
90                     "VALUES (%s, %s, %s, %s, %s, %s, %s);")
91             cur = self.conn.cursor()
92             cur.execute(query, (title, author, year, publisher, genre, quantity, shelf_location))
93             self.conn.commit()
94             cur.close()
95             QMessageBox.information(self, 'Книга додана', 'Нова книга успішно додана до каталогу.')
96             self.close()
97         except psycopg2.Error as e:
98             QMessageBox.critical(self, "Помилка бази даних", f"Помилка при додаванні книги: {e}")
99             self.conn.rollback()
100
101     1 usage
102     def label_style(self):
103         return ("QLabel {"
104                 "background-color: #b88a71; color: white; font-size: 16px;"
105                 "border-radius: 8px; padding: 10px;"

```

Рисунок 6.2 – 2 частина вмісту файлу add_book.py

```
100         "min-width: 200px; max-width: 150px; font-weight: bold;"
101     """
102     1 usage
103     def line_edit_style(self):
104         return ("QLineEdit {"
105             "font-size: 16px; border-radius: 10px; padding: 10px;"
106             "background-color: white; color: black;"
107             "min-width: 400px; max-width: 400px;"
108             "min-height: 40px; max-height: 40px;"
109         })
110     1 usage
111     def button_style(self):
112         return ("""
113             QPushButton {
114                 font-weight: bold; background-color: #b88a71; color: white;
115                 border: none; padding: 10px; text-align: center;
116                 display: inline-block; font-size: 16px; margin: 4px 2px;
117                 cursor: pointer; border-radius: 8px;
118                 min-width: 400px; max-width: 400px;
119             }
120             QPushButton:hover {
121                 background-color: #94644a;
122             }
123         """)
```

Рисунок 6.3 – 3 частина вмісту файлу add_book.py

```

add_visitor.py ×
1 import sys
2 import psycopg2
3 from PyQt5.QtCore import Qt
4 from PyQt5.QtGui import QIcon
5 from PyQt5.QtWidgets import QPushButton, QMessageBox, QMainWindow
6 import utils
7
8 usage
9 def create_connection():
10     try:
11         conn = psycopg2.connect(
12             host="localhost",
13             database="Library Assistant",
14             user="postgres",
15             password="1111")
16         return conn
17     except psycopg2.Error as e:
18         print(f"Помилка підключення до бази даних: {e}")
19         QMessageBox.critical(None, "Помилка підключення до бази даних", f"Помилка підключення до бази даних: {e}")
20         sys.exit(1)
21
22 from PyQt5.QtWidgets import QVBoxLayout, QHBoxLayout, QLabel, QLineEdit, QWidget
23
24 usage
25 class AddVisitorDialog(QMainWindow):
26     def __init__(self):
27         super().__init__()
28         self.setGeometry(100, 100, 1200, 800)
29         utils.set_background(self)
30         self.setWindowIcon(QIcon('src/logo.png'))
31         self.setWindowTitle('Додавання нового відвідувача')
32         self.conn = create_connection()
33         self.initUI()
34
35     def initUI(self):
36         self.central_widget = QWidget(self)
37         self.setCentralWidget(self.central_widget)
38         self.layout = QVBoxLayout(self.central_widget)
39         self.layout.addSpacing(150)
40
41         self.create_form_entry("Ім'я:", 'first_name_edit')
42         self.create_form_entry("Прізвище:", 'last_name_edit')
43         self.create_form_entry("Електронна пошта:", 'email_edit')
44         self.create_form_entry("Номер телефону:", 'phone_number_edit')
45
46         save_button = QPushButton('Додати відвідувача', self)
47         save_button.clicked.connect(self.add_visitor)
48         save_button.setStyleSheet(self.button_style())
49         self.layout.addWidget(save_button, alignment=Qt.AlignCenter)
50
51         exit_btn = utils.create_exit_button(self)
52         exit_btn.clicked.connect(self.close)
53         exit_btn.move(1100, 730)

```

Рисунок 6.4 – 1 частина вмісту файлу add_visitor.py


```

add_visitor.py ×
4 usages
53 def create_form_entry(self, label_text, edit_name):
54     layout = QHBoxLayout()
55     label = QLabel(label_text, self)
56     label.setStyleSheet(self.label_style())
57     setattr(self, edit_name, QLineEdit(self))
58     getattr(self, edit_name).setStyleSheet(self.line_edit_style())
59     layout.addWidget(label)
60     layout.addWidget(getattr(self, edit_name))
61     self.layout.addLayout(layout)
62
1 usage
63 def add_visitor(self):
64     first_name = self.first_name_edit.text().strip()
65     last_name = self.last_name_edit.text().strip()
66     email = self.email_edit.text().strip()
67     phone_number = self.phone_number_edit.text().strip()
68
69     if not first_name or not last_name or not email or not phone_number:
70         QMessageBox.warning(self, 'Помилка', 'Будь ласка, заповніть всі поля.')
71         return
72
73     try:
74         query = "INSERT INTO visitors (first_name, last_name, email, phone_number) VALUES (%s, %s, %s, %s);"
75         cur = self.conn.cursor()
76         cur.execute(query, (first_name, last_name, email, phone_number))
77         self.conn.commit()
78         cur.close()
79         QMessageBox.information(self, 'Відвідувача додано', 'Новий відвідувач бібліотеки успішно доданий.')
80         self.close()
81     except psycopg2.Error as e:
82         QMessageBox.critical(self, "Помилка бази даних", f"Помилка при додаванні відвідувача: {e}")
83         self.conn.rollback()
84
1 usage
85 def label_style(self):
86     return ("QLabel {"
87         "background-color: #b88a71; color: white; font-size: 16px;"
88         "border-radius: 8px; padding: 10px;"
89         "min-width: 200px; max-width: 150px; font-weight: bold;"
90         "}")
91
1 usage
92 def line_edit_style(self):
93     return ("QLineEdit {"
94         "font-size: 16px; border-radius: 10px; padding: 10px;"
95         "background-color: white; color: black;"
96         "min-width: 400px; max-width: 400px;"
97         "min-height: 40px; max-height: 40px;"
98         "}")
99

```

Рисунок 6.5 – 2 частина вмісту файлу add_visitor.py

```
1 usage
100 def button_style(self):
101     return """
102         QPushButton {
103             font-weight: bold; background-color: #b88a71; color: white;
104             border: none; padding: 10px; text-align: center;
105             display: inline-block; font-size: 16px; margin: 4px 2px;
106             cursor: pointer; border-radius: 8px;
107             min-width: 400px; max-width: 400px;
108         }
109         QPushButton:hover {
110             background-color: #94644a;
111         }
112     """
113
```

Рисунок 6.6 – 3 частина вмісту файлу add_visitor.py

```

auth_menu.py ×
1 import hashlib
2 import psycpg2
3 from PyQt5.QtCore import Qt
4 from PyQt5.QtGui import QIcon
5 from PyQt5.QtWidgets import QWidget, QLineEdit, QPushButton, QVBoxLayout, QMessageBox, QSpacerItem, QSizePolicy
6 import creator_info
7 from utils import set_background, create_exit_button, create_info_button, show_creator_info, create_return_button
8 from main_page import MainPage
9
10 class AuthWindow(QWidget):
11     def __init__(self, main_window=None):
12         super().__init__()
13         self.main_window = main_window
14         self.initUI()
15
16     def initUI(self):
17         set_background(self)
18         self.setFixedSize(1200, 800)
19         self.setWindowIcon(QIcon('src/logo.png'))
20
21         label_style = "background-color: #b88a71; color: white; font-size: 18px; border-radius: 8px; padding: 5px;"
22         line_edit_style = """
23             QLineEdit {
24                 font-size: 16px;
25                 border-radius: 10px;
26                 padding: 10px;
27                 background-color: white;
28                 color: black;
29                 min-width: 400px;
30                 max-width: 400px;
31             }
32         """
33
34         self.email = QLineEdit(self)
35         self.email.setPlaceholderText('Введіть електронну пошту')
36         self.email.setStyleSheet(line_edit_style)
37
38         self.password = QLineEdit(self)
39         self.password.setPlaceholderText('Введіть пароль')
40         self.password.setEchoMode(QLineEdit.Password)
41         self.password.setStyleSheet(line_edit_style)
42
43         self.loginButton = QPushButton("Авторизуватися", self)
44         self.loginButton.clicked.connect(self.check_credentials)
45         self.loginButton.setStyleSheet("""
46             QPushButton {
47                 font-weight: bold;
48                 background-color: #b88a71;
49                 color: white;
50                 border: none;
51                 padding: 10px;
52                 text-align: center;

```

Рисунок 6.7 – 1 частина вмісту файлу auth_menu.py

```

auth_menu.py ×
51         padding: 10px;
52         text-align: center;
53         text-decoration: none;
54         display: inline-block;
55         font-size: 16px;
56         margin: 4px 2px;
57         cursor: pointer;
58         border-radius: 8px;
59         min-width: 400px;
60         max-width: 400px;
61     }
62     QPushButton:hover {
63         background-color: #94644a;
64     }
65     """)
66
67     layout = QVBoxLayout(self)
68     spacer_top = QSpacerItem(20, 40, QSizePolicy.Minimum, QSizePolicy.Expanding)
69     layout.addItem(spacer_top)
70
71     layout.addWidget(self.email)
72     layout.setAlignment(self.email, Qt.AlignCenter)
73     layout.addWidget(self.password)
74     layout.setAlignment(self.password, Qt.AlignCenter)
75     layout.addWidget(self.loginButton)
76     layout.setAlignment(self.loginButton, Qt.AlignCenter)
77
78     spacer_bottom = QSpacerItem(20, 40, QSizePolicy.Minimum, QSizePolicy.Expanding)
79     layout.addItem(spacer_bottom)
80
81     exit_btn = create_exit_button(self)
82     exit_btn.clicked.connect(self.close)
83     exit_btn.move(1100, 730)
84
85     info_btn = create_info_button(self)
86     info_btn.clicked.connect(lambda: show_creator_info(self, self.creator_info_window))
87     info_btn.move(1000, 730)
88
89     back_btn = create_return_button(self)
90     back_btn.clicked.connect(self.go_back)
91     back_btn.move(50, 730)
92
93     self.creator_info_window = creator_info.CreatorInfoWindow()
94     self.setWindowTitle('Авторизація')
95     self.setLayout(layout)
96
97     1 usage
98     def check_credentials(self):
99         email = self.email.text().strip()
100         password = self.password.text().strip()
101
102         if not email or not password:
103             QMessageBox.warning(self, 'Помилка!', 'Будь ласка, введіть email та пароль!', QMessageBox.Ok,

```

Рисунок 6.8 – 2 частина вмісту файлу auth_menu.py

```

103         QMessageBox.Ok)
104     return
105
106     password_hash = hashlib.sha256(password.encode()).hexdigest() # Хешування пароля
107
108     try:
109         with psycopg2.connect(user='postgres', password='1111', host='127.0.0.1', port='5432',
110                               database='Library Assistant') as conn:
111             with conn.cursor() as cursor:
112                 cursor.execute('SELECT password FROM users WHERE email = %s', (email,))
113                 record = cursor.fetchone()
114                 if record and record[0] == password_hash:
115                     QMessageBox.information(self, 'Успішно', 'Ви успішно авторизувалися!', QMessageBox.Ok,
116                                             QMessageBox.Ok)
117                     self.open_main_page()
118                 else:
119                     QMessageBox.warning(self, 'Помилка!', 'Невірна пошта або пароль!', QMessageBox.Ok,
120                                         QMessageBox.Ok)
121             except psycopg2.Error as e:
122                 QMessageBox.critical(self, 'Помилка бази даних', 'Виникла помилка при роботі з базою даних: ' + str(e),
123                                     QMessageBox.Ok, QMessageBox.Ok)
124
125     1 usage
126     def open_main_page(self):
127         self.main_page = MainPage()
128         self.main_page.show()
129         self.close()
130
131     1 usage
132     def go_back(self):
133         self.close()
134         if self.main_window:
135             self.main_window.show()

```

Рисунок 6.9 – 3 частина вмісту файлу auth_menu.py

```

books_page.py ×
1 import sys
2 import psycopg2
3 from PyQt5.QtCore import Qt
4 from PyQt5.QtGui import QIcon
5 from PyQt5.QtWidgets import (QMainWindow, QVBoxLayout, QWidget, QPushButton,
6                             QTableWidgetItem, QTableWidgetItem, QHBoxLayout, QMessageBox, QLineEdit, QLabel, QHeaderView,
7                             QGridLayout)
8
9 import utils
10 import edit_book
11 import add_book, main_page
12
13 usage
14
15 def create_connection():
16     try:
17         conn = psycopg2.connect(
18             host="localhost",
19             database="Library Assistant",
20             user="postgres",
21             password="1111")
22         return conn
23     except psycopg2.Error as e:
24         print(f"Помилка підключення до бази даних: {e}")
25         QMessageBox.critical(None, "Помилка підключення до бази даних", f"Помилка підключення до бази даних: {e}")
26         sys.exit(1)
27
28 usage
29
30 class LibraryCatalog(QMainWindow):
31     def __init__(self):
32         super().__init__()
33         self.setGeometry(100, 100, 900, 600)
34         self.setWindowTitle('Каталог книг')
35         self.setWindowIcon(QIcon('src/logo.png'))
36         utils.set_background(self)
37         self.conn = create_connection()
38         self.current_page = 0
39         self.books_per_page = 10
40         self.current_filters = {}
41         self.initUI()
42
43     def initUI(self):
44         self.central_widget = QWidget()
45         self.setCentralWidget(self.central_widget)
46
47         label_style = ("background-color: #b88a71; color: white; font-size: 16px; border-radius: 8px; padding: 2px; "
48                       "min-width: 130px; max-width: 120px; font-weight: bold;")
49
50         line_edit_style = """
51             QLineEdit {
52                 font-size: 16px;
53                 border-radius: 10px;
54                 padding: 10px;
55                 background-color: white;
56                 color: black;

```

Рисунок 6.10 – 1 частина вмісту файлу books_page.py

```

books_page.py ×
50         background-color: white;
51         color: black;
52         min-width: 200px;
53         max-width: 200px;
54     }
55     """
56
57     main_layout = QVBoxLayout(self.central_widget)
58     self.setFixedSize(1200, 800)
59
60     search_grid = QGridLayout()
61
62     self.title_search = QLineEdit()
63     self.title_search.setPlaceholderText("Пошук по назві")
64     self.title_search.setStyleSheet(line_edit_style)
65     self.author_search = QLineEdit()
66     self.author_search.setPlaceholderText("Пошук по автору")
67     self.author_search.setStyleSheet(line_edit_style)
68     self.year_search = QLineEdit()
69     self.year_search.setPlaceholderText("Пошук по року")
70     self.year_search.setStyleSheet(line_edit_style)
71     self.publisher_search = QLineEdit()
72     self.publisher_search.setPlaceholderText("Пошук по видавництву")
73     self.publisher_search.setStyleSheet(line_edit_style)
74     self.genre_search = QLineEdit()
75     self.genre_search.setPlaceholderText("Пошук по жанру")
76     self.genre_search.setStyleSheet(line_edit_style)
77
78     search_grid.addWidget(QLabel("Назва:", styleSheet=label_style), 0, 0)
79     search_grid.addWidget(self.title_search, 0, 1)
80
81     search_grid.addWidget(QLabel("Автор:", styleSheet=label_style), 0, 2)
82     search_grid.addWidget(self.author_search, 0, 3)
83     search_grid.addWidget(QLabel("Рік:", styleSheet=label_style), 0, 4)
84     search_grid.addWidget(self.year_search, 0, 5)
85
86     search_grid.addWidget(QLabel("Видавництво:", styleSheet=label_style), 1, 0)
87     search_grid.addWidget(self.publisher_search, 1, 1)
88     search_grid.addWidget(QLabel("Жанр:", styleSheet=label_style), 1, 2)
89     search_grid.addWidget(self.genre_search, 1, 3)
90
91     self.search_button = QPushButton("Пошук")
92     self.search_button.clicked.connect(self.perform_search)
93     search_grid.addWidget(self.search_button, 1, 4, 1, 2)
94     self.search_button.setStyleSheet("""
95         QPushButton {
96             font-weight: bold;
97             background-color: #b88a71;
98             color: white;
99             border: none;
100            padding: 15px 32px;
101            text-align: center;
102            text-decoration: none;

```

Рисунок 6.11 – 2 частина вмісту файлу books_page.py

```

books_page.py ×
101         text-align: center;
102         text-decoration: none;
103         display: inline-block;
104         font-size: 16px;
105         margin: 4px 2px;
106         cursor: pointer;
107         border-radius: 8px;
108     }
109     QPushButton:hover {
110         background-color: #94644a;
111     }
112     """)
113
114     main_layout.addLayout(search_grid)
115
116     self.table = QTableWidgetItem(0, 9, self)
117     self.table.setHorizontalHeaderLabels(["Назва", "Автор", "Рік", "Видавництво", "Жанр", "Кількість",
118     "Місцезнаходження", "Змінити", "Видалити"])
119     main_layout.addWidget(self.table)
120
121     self.table.horizontalHeader().setStretchLastSection(True)
122     self.table.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)
123     self.table.horizontalHeader().setSectionsClickable(False)
124     self.table.horizontalHeader().setSectionsMovable(False)
125
126     row_height = 30
127     self.table.verticalHeader().setDefaultSectionSize(row_height)
128     self.table.setFixedHeight(row_height * 11)
129     self.table.setStyleSheet("""
130     QTableWidgetItem {
131         background-color: #F5F5DC;
132         gridline-color: #000000;
133     }
134     QHeaderView::section {
135         background-color: #F5F5DC;
136     }
137     QTableWidgetItem::item {
138         background-color: #b88a71;
139     }
140     """)
141
142     pagination_layout = QHBoxLayout()
143     self.prev_button = QPushButton('Попередня')
144     self.prev_button.clicked.connect(self.load_previous_page)
145     self.prev_button.setStyleSheet("""
146     QPushButton {
147         font-weight: bold;
148         background-color: #b88a71;
149         color: white;
150         border: none;
151         padding: 15px 32px;
152         text-align: center;
153         text-decoration: none;

```

Рисунок 6.12 – 3 частина вмісту файлу books_page.py


```

books_page.py ×
152         text-align: center;
153         text-decoration: none;
154         display: inline-block;
155         display: inline-block;
156         font-size: 16px;
157         margin: 4px 2px;
158         cursor: pointer;
159         border-radius: 8px;
160     }
161     QPushButton:hover {
162         background-color: #94644a;
163     }
164     """
165     self.next_button = QPushButton('Наступна')
166     self.next_button.clicked.connect(self.load_next_page)
167     self.next_button.setStyleSheet("""
168         QPushButton {
169             font-weight: bold;
170             background-color: #b88a71;
171             color: white;
172             border: none;
173             padding: 15px 32px;
174             text-align: center;
175             text-decoration: none;
176             display: inline-block;
177             font-size: 16px;
178             margin: 4px 2px;
179             cursor: pointer;
180             border-radius: 8px;
181         }
182         QPushButton:hover {
183             background-color: #94644a;
184         }
185     """)
186     pagination_layout.addWidget(self.prev_button)
187     pagination_layout.addWidget(self.next_button)
188     main_layout.addLayout(pagination_layout)
189
190     add_book_layout = QHBoxLayout()
191     self.add_book_button = QPushButton('Додати книгу')
192     self.add_book_button.clicked.connect(self.open_add_book_dialog)
193     self.add_book_button.setStyleSheet("""
194         QPushButton {
195             font-weight: bold;
196             background-color: #b88a71;
197             color: white;
198             border: none;
199             padding: 15px 32px;
200             text-align: center;
201             text-decoration: none;
202             display: inline-block;
203             font-size: 16px;
204             margin: 4px 2px;

```

Рисунок 6.13 – 4 частина вмісту файлу books_page.py

```

books_page.py x
203         font-size: 16px;
204         margin: 4px 2px;
205         cursor: pointer;
206         border-radius: 8px;
207     }
208     QPushButton:hover {
209         background-color: #94644a;
210     }
211     """
212     add_book_layout.addStretch()
213     add_book_layout.addWidget(self.add_book_button)
214     add_book_layout.addStretch()
215     self.add_book_button.setFixedSize(500, 70)
216     main_layout.addLayout(add_book_layout)
217
218     exit_btn = utils.create_exit_button(self)
219     exit_btn.clicked.connect(self.close)
220     exit_btn.move(1100, 730)
221
222     back_btn = utils.create_return_button(self)
223     back_btn.clicked.connect(self.go_to_main_page)
224     back_btn.move(50, 730)
225
226     self.load_data()
227
228     1 usage
229     def perform_search(self):
230         self.current_filters = {
231             'title': self.title_search.text().strip(),
232             'author': self.author_search.text().strip(),
233             'year': self.year_search.text().strip(),
234             'publisher': self.publisher_search.text().strip(),
235             'genre': self.genre_search.text().strip()
236         }
237         self.current_page = 0
238         self.load_data()
239
240     1 usage
241     def open_add_book_dialog(self):
242         self.add_book_window = add_book.AddBookDialog()
243         self.add_book_window.show()
244
245     1 usage
246     def go_to_main_page(self):
247         self.main_page = main_page.MainPage()
248         self.main_page.show()
249         self.close()
250
251     5 usages
252     def load_data(self):
253         conditions = ["quantity > 0"]
254         parameters = []
255         for field, value in self.current_filters.items():

```

Рисунок 6.14 – 5 частина вмісту файлу books_page.py

```

books_page.py ×
250 parameters = []
251 for field, value in self.current_filters.items():
252     if value:
253         if field == 'year':
254             conditions.append(f"{field}_published = %s")
255         else:
256             conditions.append(f"{field} ILIKE %s")
257     parameters.append(f"%{value}%" if field != 'year' else value)
258
259 query = "SELECT book_id, title, author, year_published, publisher, genre, quantity, shelf_location FROM books"
260 if conditions:
261     query += " WHERE " + " AND ".join(conditions)
262 query += f" LIMIT {self.books_per_page + 1} OFFSET {self.current_page * self.books_per_page};"
263
264 cur = self.conn.cursor()
265 cur.execute(query, parameters)
266 records = cur.fetchall()
267 cur.close()
268
269 has_next_page = len(records) > self.books_per_page
270 records = records[:self.books_per_page] if has_next_page else records
271
272 self.table.setRowCount(len(records))
273 for row_number, row_data in enumerate(records):
274     for column_number, data in enumerate(row_data[1:], 1):
275         item = QTableWidgetItem(str(data))
276         item.setFlags(item.flags() & ~Qt.ItemIsEditable)
277         self.table.setItem(row_number, column_number - 1, item)
278
279 edit_btn = QPushButton('Змінити')
280 edit_btn.clicked.connect(lambda _, book_id=row_data[0]: self.edit_book(book_id))
281 edit_btn.setStyleSheet("""
282         QPushButton {
283             background-color: #94644a;
284         }
285     """)
286 self.table.setCellWidget(row_number, 7, edit_btn)
287
288 delete_btn = QPushButton('Видалити')
289 delete_btn.clicked.connect(lambda _, book_id=row_data[0]: self.delete_book(book_id))
290 delete_btn.setStyleSheet("""
291         QPushButton {
292             background-color: #94644a;
293         }
294     """)
295 self.table.setCellWidget(row_number, 8, delete_btn)
296
297 self.prev_button.setEnabled(self.current_page > 0)
298 self.next_button.setEnabled(has_next_page)
299
300 1 usage
301 def edit_book(self, book_id):
302     self.edit_window = edit_book.EditBookDialog(book_id)

```

Рисунок 6.15 – 6 частина вмісту файлу books_page.py

```
302     self.edit_window.show()
303
304     1 usage
305     def delete_book(self, book_id):
306         reply = QMessageBox.question(self, 'Підтвердіть видалення', 'Ви впевнені, що хочете видалити цю книгу?',
307                                     QMessageBox.Yes | QMessageBox.No, QMessageBox.No)
308         if reply == QMessageBox.Yes:
309             cur = self.conn.cursor()
310             cur.execute("DELETE FROM books WHERE book_id = %s;", (book_id,))
311             self.conn.commit()
312             cur.close()
313             self.load_data()
314
315     1 usage
316     def load_next_page(self):
317         self.current_page += 1
318         self.load_data()
319
320     1 usage
321     def load_previous_page(self):
322         if self.current_page > 0:
323             self.current_page -= 1
324             self.load_data()
```

Рисунок 6.16 – 7 частина вмісту файлу books_page.py

```

creator_info.py ×
1  from PyQt5.QtWidgets import QVBoxLayout, QHBoxLayout, QWidget, QLabel
2  from PyQt5.QtGui import QPixmap, QIcon
3  from PyQt5.QtCore import Qt
4  from utils import set_background, create_exit_button, create_return_button
5
6  3 usages
7  class CreatorInfoWindow(QWidget):
8      def __init__(self, main_window=None):
9          super().__init__()
10         self.main_window = main_window
11         self.setWindowIcon(QIcon('D:/LibraryAssistant/src/logo.png'))
12         self.initUI()
13
14     def initUI(self):
15         self.setWindowTitle("Інформація про розробника")
16         set_background(self)
17         self.setFixedSize(1200, 800)
18
19         main_layout = QVBoxLayout(self)
20
21         content_layout = QHBoxLayout()
22
23         label_image = QLabel(self)
24         pixmap = QPixmap('D:/LibraryAssistant/src/creator_photo.jpg')
25         label_image.setPixmap(pixmap.scaled(400, 400, Qt.KeepAspectRatio, Qt.SmoothTransformation))
26         content_layout.addWidget(label_image)
27         label_image.setStyleSheet("border-radius: 15px;")
28
29         label_description = QLabel("Кваліфікаційна робота на здобуття освітнього ступеня бакалавра. '
30                                     'Тема: Інформаційна система для автоматизації діяльності бібліотеки. '
31                                     'Виконала: студентка групи ІН-02 Заїченко Тетяна Вікторівна.')
32         label_description.setWordWrap(True)
33         label_description.setStyleSheet("""
34             QLabel {
35                 background-color: #f8f8f8;
36                 color: black;
37                 font-size: 18px;
38                 border-radius: 15px;
39                 padding: 20px;
40             }
41         """)
42         content_layout.addWidget(label_description)
43
44         main_layout.addLayout(content_layout)
45
46         main_layout.setAlignment(Qt.AlignCenter)
47
48         exit_btn = create_exit_button(self)
49         exit_btn.clicked.connect(self.close)
50         exit_btn.move(1100, 730)
51
52         back_btn = create_return_button(self)
53         back_btn.clicked.connect(self.go_back)

```

Рисунок 6.17 – 1 частина вмісту файлу creator_info.py

```

53         back_btn.move(50, 730)
54
55         self.setLayout(main_layout)
56
57     1 usage
58     def go_back(self):
59         self.close()
60         if self.main_window:
61             self.main_window.show()

```

Рисунок 6.18 – 2 частина вмісту файлу creator_info.py

```

edit_book.py ×
1 import sys
2 import psycopg2
3 from PyQt5.QtCore import Qt
4 from PyQt5.QtGui import QIcon
5 from PyQt5.QtWidgets import QMainWindow, QWidget, QVBoxLayout, QHBoxLayout, QLineEdit, QLabel, QPushButton,
6     QMessageBox, QApplication)
7 import utils
8
9 1 usage
10 def create_connection():
11     try:
12         conn = psycopg2.connect(
13             host="localhost",
14             database="Library Assistant",
15             user="postgres",
16             password="1111")
17         return conn
18     except psycopg2.Error as e:
19         print(f"Error connecting to the database: {e}")
20         QMessageBox.critical(None, "Database Connection Error", f"Failed to connect to database: {e}")
21         sys.exit(1)
22
23 1 usage
24 class EditBookDialog(QMainWindow):
25     def __init__(self, book_id):
26         super().__init__()
27         self.book_id = book_id
28         self.setGeometry(100, 100, 350, 200)
29         utils.set_background(self)
30         self.setWindowIcon(QIcon('src/logo.png'))
31         self.setWindowTitle('Редагування відомостей про книгу')
32         self.conn = create_connection()
33         self.initUI()
34
35     def initUI(self):
36         self.central_widget = QWidget()
37         self.setCentralWidget(self.central_widget)
38         self.setFixedSize(1200, 800)
39
40         self.layout = QVBoxLayout(self.central_widget)
41         self.layout.addSpacing(150)
42         self.create_form_entry("Назва:", "title_edit")
43         self.create_form_entry("Автор:", "author_edit")
44         self.create_form_entry("Рік публікації:", "year_edit")
45         self.create_form_entry("Видавництво:", "publisher_edit")
46         self.create_form_entry("Жанр:", "genre_edit")
47         self.create_form_entry("Кількість:", "quantity_edit")
48         self.create_form_entry("Місцезнаходження:", "shelf_location_edit")
49
50         save_button = QPushButton('Зберегти зміни', self)
51         save_button.clicked.connect(self.save_changes)
52         save_button.setStyleSheet(self.button_style())
53         self.layout.addWidget(save_button, alignment=Qt.AlignCenter)

```

Рисунок 6.19 – 1 частина вмісту файлу edit_book.py


```

edit_book.py ×
50     save_button.setStyleSheet(self.button_style())
51     self.layout.addWidget(save_button, alignment=Qt.AlignCenter)
52
53     exit_btn = utils.create_exit_button(self)
54     exit_btn.clicked.connect(self.close)
55     exit_btn.move(1100, 730)
56
57     self.load_book_details()
58
59     7 usages
60     def create_form_entry(self, label_text, edit_name):
61         layout = QHBoxLayout()
62         label = QLabel(label_text, self)
63         label.setStyleSheet(self.label_style())
64         setattr(self, edit_name, QLineEdit(self))
65         getattr(self, edit_name).setStyleSheet(self.line_edit_style())
66         layout.addWidget(label)
67         layout.addWidget(getattr(self, edit_name))
68         self.layout.addLayout(layout)
69
70     1 usage
71     def label_style(self):
72         return ("QLabel {"
73             "background-color: #b88a71; color: white; font-size: 16px;"
74             "border-radius: 8px; padding: 10px;"
75             "min-width: 200px; max-width: 150px; font-weight: bold;"
76             "}")
77
78     1 usage
79     def line_edit_style(self):
80         return ("QLineEdit {"
81             "font-size: 16px; border-radius: 10px; padding: 10px;"
82             "background-color: white; color: black;"
83             "min-width: 400px; max-width: 400px;"
84             "min-height: 40px; max-height: 40px;"
85             "}")
86
87     1 usage
88     def button_style(self):
89         return ("""
90             QPushButton {
91                 font-weight: bold; background-color: #b88a71; color: white;
92                 border: none; padding: 10px; text-align: center;
93                 display: inline-block; font-size: 16px; margin: 4px 2px;
94                 cursor: pointer; border-radius: 8px;
95                 min-width: 400px; max-width: 400px;
96             }
97             QPushButton:hover {
98                 background-color: #94644a;
99             }
100         """)

```

Рисунок 6.20 – 2 частина вмісту файлу edit_book.py

```

edit_book.py x
96
97
1 usage
98 def load_book_details(self):
99     query = (f"SELECT title, author, year_published, publisher, genre, quantity, shelf_location "
100             f"FROM books WHERE book_id = {self.book_id};")
101     cur = self.conn.cursor()
102     cur.execute(query)
103     book = cur.fetchone()
104     cur.close()
105     if book:
106         self.title_edit.setText(book[0])
107         self.author_edit.setText(book[1])
108         self.year_edit.setText(str(book[2]))
109         self.publisher_edit.setText(book[3])
110         self.genre_edit.setText(book[4])
111         self.quantity_edit.setText(str(book[5]))
112         self.shelf_location_edit.setText(book[6])
113
114 1 usage
115 def save_changes(self):
116     title = self.title_edit.text().strip()
117     author = self.author_edit.text().strip()
118     year_str = self.year_edit.text().strip()
119     publisher = self.publisher_edit.text().strip()
120     genre = self.genre_edit.text().strip()
121     quantity = self.quantity_edit.text().strip()
122     shelf_location = self.shelf_location_edit.text().strip()
123
124     if not title or not author or not year_str or not publisher or not genre or not quantity or not shelf_location:
125         QMessageBox.warning(self, "Помилка", "Будь ласка, заповніть всі поля.")
126         return
127
128     try:
129         year = int(year_str)
130         quantity = int(quantity)
131     except ValueError:
132         QMessageBox.warning(self, "Помилка", "Рік публікації та кількість мають бути числами.")
133         return
134
135     if year <= 0 or quantity <= 0:
136         QMessageBox.warning(self, "Помилка", "Рік публікації та кількість повинні бути більше 0.")
137         return
138
139     try:
140         query = ("UPDATE books SET title = %s, author = %s, year_published = %s, publisher = %s, genre = %s, "
141                 " quantity = %s, shelf_location = %s WHERE book_id = %s;")
142         cur = self.conn.cursor()
143         cur.execute(query, (title, author, year, publisher, genre, quantity, shelf_location, self.book_id))
144         self.conn.commit()
145         cur.close()
146         QMessageBox.information(self, "Зміни збережено", "Інформація про книгу успішно оновлена.")
147         self.close()
148     except psycopg2.Error as e:
149         QMessageBox.critical(self, "Помилка бази даних", f"Помилка при оновленні даних: {e}")
150         self.conn.rollback()
151         cur.close()

```

Рисунок 6.21 – 3 частина вмісту файлу edit_book.py

```

edit_visitor.py ×
1 import sys
2 import psycopg2
3 from PyQt5.QtCore import Qt
4 from PyQt5.QtGui import QIcon
5 from PyQt5.QtWidgets import (QMainWindow, QWidget, QVBoxLayout, QHBoxLayout, QLineEdit, QLabel, QPushButton,
6                             QMessageBox, QApplication)
7
8 import utils
9
10 usage
11 def create_connection():
12     try:
13         conn = psycopg2.connect(
14             host="localhost",
15             database="Library Assistant",
16             user="postgres",
17             password="1111")
18         return conn
19     except psycopg2.Error as e:
20         print(f"Error connecting to the database: {e}")
21         QMessageBox.critical(None, "Database Connection Error", f"Failed to connect to database: {e}")
22         sys.exit(1)
23
24 usage
25 class EditVisitorDialog(QMainWindow):
26     def __init__(self, visitor_id):
27         super().__init__()
28         self.visitor_id = visitor_id
29         self.setGeometry(100, 100, 350, 200)
30         utils.set_background(self)
31         self.setWindowIcon(QIcon('src/logo.png'))
32         self.setWindowTitle('Редагування відомостей про відвідувача')
33         self.conn = create_connection()
34         self.initUI()
35
36     def initUI(self):
37         self.central_widget = QWidget()
38         self.setCentralWidget(self.central_widget)
39         self.setFixedSize(1200, 800)
40
41         self.layout = QVBoxLayout(self.central_widget)
42         self.layout.addSpacing(150)
43         self.create_form_entry("Ім'я:", 'first_name_edit')
44         self.create_form_entry("Прізвище:", 'last_name_edit')
45         self.create_form_entry("Електронна пошта:", 'email_edit')
46         self.create_form_entry("Номер телефону:", 'phone_number_edit')
47
48         save_button = QPushButton('Зберегти зміни', self)
49         save_button.clicked.connect(self.save_changes)
50         save_button.setStyleSheet(self.button_style())
51         self.layout.addWidget(save_button, alignment=Qt.AlignCenter)
52
53         exit_btn = utils.create_exit_button(self)

```

Рисунок 6.22 – 1 частина вмісту файлу edit_visitor.py

```

edit_visitor.py ×
50
51     exit_btn = utils.create_exit_button(self)
52     exit_btn.clicked.connect(self.close)
53     exit_btn.move(1100, 730)
54
55     self.load_visitor_details()
56
57     4 usages
58     def create_form_entry(self, label_text, edit_name):
59         layout = QHBoxLayout()
60         label = QLabel(label_text, self)
61         label.setStyleSheet(self.label_style())
62         setattr(self, edit_name, QLineEdit(self))
63         getattr(self, edit_name).setStyleSheet(self.line_edit_style())
64         layout.addWidget(label)
65         layout.addWidget(getattr(self, edit_name))
66         self.layout.addLayout(layout)
67
68     1 usage
69     def label_style(self):
70         return ("QLabel {"
71             "background-color: #b88a71; color: white; font-size: 16px;"
72             "border-radius: 8px; padding: 10px;"
73             "min-width: 200px; max-width: 150px; font-weight: bold;"
74             "}")
75
76     1 usage
77     def line_edit_style(self):
78         return ("QLineEdit {"
79             "font-size: 16px; border-radius: 10px; padding: 10px;"
80             "background-color: white; color: black;"
81             "min-width: 400px; max-width: 400px;"
82             "min-height: 40px; max-height: 40px;"
83             "}")
84
85     1 usage
86     def button_style(self):
87         return ("
88             QPushButton {
89                 font-weight: bold; background-color: #b88a71; color: white;
90                 border: none; padding: 10px; text-align: center;
91                 display: inline-block; font-size: 16px; margin: 4px 2px;
92                 cursor: pointer; border-radius: 8px;
93                 min-width: 400px; max-width: 400px;
94             }
95             QPushButton:hover {
96                 background-color: #94644a;
97             }
98         ")
99
100     1 usage
101     def load_visitor_details(self):
102         query = (f"SELECT first_name, last_name, email, phone_number "

```

Рисунок 6.23 – 2 частина вмісту файлу edit_visitor.py

```

97     query = (f"SELECT first_name, last_name, email, phone_number "
98             f"FROM visitors WHERE visitor_id = {self.visitor_id};")
99     cur = self.conn.cursor()
100    cur.execute(query)
101    visitor = cur.fetchone()
102    cur.close()
103    if visitor:
104        self.first_name_edit.setText(visitor[0])
105        self.last_name_edit.setText(visitor[1])
106        self.email_edit.setText(str(visitor[2]))
107        self.phone_number_edit.setText(visitor[3])
108
109    1 usage
110    def save_changes(self):
111        first_name = self.first_name_edit.text().strip()
112        last_name = self.last_name_edit.text().strip()
113        email = self.email_edit.text().strip()
114        phone_number = self.phone_number_edit.text().strip()
115
116        if not first_name or not last_name or not email or not phone_number:
117            QMessageBox.warning(self, 'Помилка', 'Будь ласка, заповніть всі поля.')
118            return
119
120        try:
121            query = ("UPDATE visitors SET first_name = %s, last_name = %s, email = %s, phone_number = %s "
122                    "WHERE visitor_id = %s;")
123            cur = self.conn.cursor()
124            cur.execute(query, (first_name, last_name, email, phone_number, self.visitor_id))
125            self.conn.commit()
126            cur.close()
127            QMessageBox.information(self, "Зміни збережено", "Інформація про відвідувача успішно оновлена.")
128            self.close()
129        except psycopg2.Error as e:
130            QMessageBox.critical(self, "Помилка бази даних", f"Помилка при оновленні даних: {e}")
131            self.conn.rollback()
132            cur.close()

```

Рисунок 6.24 – 3 частина вмісту файлу edit_visitor.py

```

issue_book.py ×
1 from PyQt5.QtCore import Qt
2 from PyQt5.QtGui import QIcon
3 from PyQt5.QtWidgets import QDialog, QVBoxLayout, QLabel, QComboBox, QDateEdit, QPushButton, QMessageBox
4 from datetime import date, timedelta
5 import utils
6
7 usage
8 class IssueBookDialog(QDialog):
9     def __init__(self, visitor_id, conn):
10         super().__init__()
11         self.visitor_id = visitor_id
12         self.conn = conn
13         self.setGeometry(100, 100, 900, 600)
14         self.setWindowIcon(QIcon('src/logo.png'))
15         utils.set_background(self)
16         self.initUI()
17
18     def initUI(self):
19         self.setWindowTitle('Видати книги')
20         layout = QVBoxLayout(self)
21         self.setFixedSize(1200, 800)
22
23         layout.setContentsMargins(10, 10, 10, 10)
24         layout.setSpacing(5)
25
26         label_style = ("background-color: #b88a71; color: white; font-size: 16px; "
27                      "border-radius: 8px; padding: 10px 5px; min-width: 400px; "
28                      "max-width: 400px; font-weight: bold; text-align: center;")
29
30         combo_box_style = """
31         QComboBox {
32             font-size: 16px;
33             border-radius: 10px;
34             padding: 5px;
35             min-width: 400px;
36             max-width: 400px;
37         }
38         QComboBox QAbstractItemView {
39             selection-background-color: #b88a71;
40             height: 120px;
41         }
42         """
43
44         date_edit_style = """
45         QDateEdit {
46             font-size: 16px;
47             border-radius: 10px;
48             padding: 5px;
49             min-width: 400px;
50             max-width: 400px;
51             background-color: white;
52         }
53         QDateEdit QCalendarWidget {

```

Рисунок 6.25 – 1 частина вмісту файлу issue_book.py


```

51     }
52     QDateEdit QCalendarWidget {
53         font-size: 14px;
54     }
55     """
56
57     button_style = """
58         QPushButton {
59             font-weight: bold;
60             background-color: #b88a71;
61             color: white;
62             border: none;
63             padding: 15px 32px;
64             text-align: center;
65             text-decoration: none;
66             display: inline-block;
67             font-size: 16px;
68             margin: 4px 2px;
69             cursor: pointer;
70             border-radius: 8px;
71             min-width: 350px;
72             max-width: 350px;
73         }
74         QPushButton:hover {
75             background-color: #94644a;
76         }
77     """
78
79     layout.addStretch(1)
80     label = QLabel('Оберіть книгу:', styleSheet=label_style)
81     self.combo_box = QComboBox()
82     self.combo_box.setStyleSheet(combo_box_style)
83     layout.addWidget(label, alignment=Qt.AlignCenter)
84     layout.addWidget(self.combo_box, alignment=Qt.AlignCenter)
85
86     self.load_books()
87
88     label_date = QLabel('Дата повернення:', styleSheet=label_style)
89     self.date_edit = QDateEdit()
90     self.date_edit.setStyleSheet(date_edit_style)
91     self.date_edit.setCalendarPopup(True)
92     self.date_edit.setDate(date.today() + timedelta(days=14))
93     layout.addWidget(label_date, alignment=Qt.AlignCenter)
94     layout.addWidget(self.date_edit, alignment=Qt.AlignCenter)
95
96     issue_button = QPushButton('Видати книгу')
97     issue_button.setStyleSheet(button_style)
98     issue_button.clicked.connect(self.issue_book)
99     layout.addWidget(issue_button, alignment=Qt.AlignCenter)
100    layout.addStretch(1)
101
102    exit_btn = utils.create_exit_button(self)
103    exit_btn.clicked.connect(self.close)

```

Рисунок 6.26 – 2 частина вмісту файлу issue_book.py

```

104     exit_btn.move(1100, 730)
105
106     1 usage
107     def load_books(self):
108         cur = self.conn.cursor()
109         cur.execute(
110             "SELECT book_id, author, title FROM books WHERE book_id NOT IN "
111             "(SELECT book_id FROM loans WHERE return_date IS NULL) AND quantity > 0"
112         )
113         books = cur.fetchall()
114         cur.close()
115         for book_id, author, title in books:
116             display_text = f"{author} - {title}"
117             self.combo_box.addItem(display_text, book_id)
118
119     1 usage
120     def issue_book(self):
121         book_id = self.combo_box.currentData()
122         due_date = self.date_edit.date().toPyDate()
123         cur = self.conn.cursor()
124
125         try:
126             cur.execute("SELECT COUNT(*) FROM loans WHERE visitor_id = %s AND return_date IS NULL", (self.visitor_id,))
127             books_count = cur.fetchone()[0]
128             if books_count >= 10:
129                 QMessageBox.warning(self, 'Ліміт перевищено', 'Ви не можете взяти більше 10 книг одночасно.')
130                 return
131
132             cur.execute("SELECT quantity FROM books WHERE book_id = %s", (book_id,))
133             quantity = cur.fetchone()[0]
134             if quantity > 0:
135                 cur.execute("INSERT INTO loans (book_id, visitor_id, date_out, due_date) VALUES (%s, %s, %s, %s)",
136                             (book_id, self.visitor_id, date.today(), due_date))
137                 cur.execute("UPDATE books SET quantity = quantity - 1 WHERE book_id = %s", (book_id,))
138                 self.conn.commit()
139                 QMessageBox.information(self, 'Книга видана', 'Книга успішно видана!')
140                 self.close()
141             else:
142                 QMessageBox.warning(self, 'Немає в наявності', 'На жаль, книги більше немає в наявності.')
143         except Exception as e:
144             self.conn.rollback()
145             QMessageBox.critical(self, 'Помилка', f'Сталася помилка при виданні книги: {e}')
146         finally:
147             cur.close()

```

Рисунок 6.27 – 3 частина вмісту файлу issue_book.py

```

loans_profile.py ×
1 from PyQt5.QtCore import Qt
2 from PyQt5.QtGui import QIcon
3 from PyQt5.QtWidgets import (QDialog, QVBoxLayout, QPushButton, QTableWidgetItem, QMessageBox,
4                               QHeaderView, QLabel)
5
6 import utils
7
8 usage
9
10 class LoansDialog(QDialog):
11     def __init__(self, visitor_id, conn):
12         super().__init__()
13         self.visitor_id = visitor_id
14         self.conn = conn
15         self.setGeometry(100, 100, 900, 600)
16         self.setWindowIcon(QIcon('src/logo.png'))
17         utils.set_background(self)
18         self.initUI()
19
20     def initUI(self):
21         self.setWindowTitle('Книги на руках')
22         layout = QVBoxLayout(self)
23         self.setFixedSize(1200, 800)
24
25         cur = self.conn.cursor()
26         cur.execute("SELECT first_name, last_name, email, phone_number FROM visitors WHERE visitor_id = %s",
27                   (self.visitor_id,))
28         data = cur.fetchone()
29         cur.close()
30
31         info_label_style = ("background-color: #b88a71; color: white; font-size: 16px; "
32                             "border-radius: 8px; padding: 10px 5px; min-width: 400px; "
33                             "max-width: 400px; font-weight: bold; text-align: center;")
34         layout.addStretch(1)
35         info_label = QLabel(f"Ім'я: {data[0]}\nПрізвище: {data[1]}\nEmail: {data[2]}\nТелефон: {data[3]}",
36                            styleSheet=info_label_style)
37         layout.addWidget(info_label, alignment=Qt.AlignCenter)
38
39         self.table = QTableWidgetItem(0, 5, self)
40         self.table.setHorizontalHeaderLabels(['Назва', 'Автор', 'Дата видачі', 'Термін повернення', 'Повернути'])
41         self.table.setStyleSheet("""
42             QTableWidgetItem {
43                 background-color: #F5F5DC;
44                 gridline-color: #000000;
45             }
46             QHeaderView::section {
47                 background-color: #F5F5DC;
48             }
49             QTableWidgetItem::item {
50                 background-color: #b88a71;
51             }
52         """)
53         layout.addWidget(self.table)
54
55         self.table.horizontalHeader().setStretchLastSection(True)
56         self.table.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)

```

Рисунок 6.28 – 1 частина вмісту файлу loans_profile.py

```

loans_profile.py ×
52 self.table.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)
53 self.table.horizontalHeader().setSectionsClickable(False)
54 self.table.horizontalHeader().setSectionsMovable(False)
55
56 row_height = 30
57 self.table.verticalHeader().setDefaultSectionSize(row_height)
58 self.table.setFixedHeight(row_height * 11)
59
60 layout.addStretch(1)
61
62 exit_btn = utils.create_exit_button(self)
63 exit_btn.clicked.connect(self.close)
64 exit_btn.move(1100, 730)
65
66 self.load_loans()
67
68 2 usages
69 def load_loans(self):
70     query = """
71     SELECT l.loan_id, b.title, b.author, l.date_out, l.due_date
72     FROM loans l
73     JOIN books b ON l.book_id = b.book_id
74     WHERE l.visitor_id = %s AND l.return_date IS NULL
75     """
76     cur = self.conn.cursor()
77     cur.execute(query, (self.visitor_id,))
78     records = cur.fetchall()
79     cur.close()
80
81     self.table.setRowCount(len(records))
82     for row_number, (loan_id, title, author, date_out, due_date) in enumerate(records):
83         item_title = QTableWidgetItem(title)
84         item_title.setFlags(item_title.flags() & ~Qt.ItemIsEditable)
85         self.table.setItem(row_number, 0, item_title)
86
87         item_author = QTableWidgetItem(str(author))
88         item_author.setFlags(item_author.flags() & ~Qt.ItemIsEditable)
89         self.table.setItem(row_number, 1, item_author)
90
91         item_date_out = QTableWidgetItem(str(date_out))
92         item_date_out.setFlags(item_date_out.flags() & ~Qt.ItemIsEditable)
93         self.table.setItem(row_number, 2, item_date_out)
94
95         item_due_date = QTableWidgetItem(str(due_date))
96         item_due_date.setFlags(item_due_date.flags() & ~Qt.ItemIsEditable)
97         self.table.setItem(row_number, 3, item_due_date)
98
99     return_btn = QPushButton('Повернути')
100     return_btn.setStyleSheet("""
101         QPushButton {
102             background-color: #94644a;
103         }
104     """)

```

Рисунок 6.29 – 2 частина вмісту файлу loans_profile.py

```

103         """
104         return_btn.clicked.connect(lambda _: ln=loan_id: self.return_book(ln))
105         self.table.setCellWidget(row_number, 4, return_btn)
106
107     1 usage
108     def return_book(self, loan_id):
109         cur = self.conn.cursor()
110         try:
111             cur.execute("UPDATE loans SET return_date = CURRENT_DATE WHERE loan_id = %s", (loan_id,))
112             cur.execute(
113                 "UPDATE books SET quantity = quantity + 1 WHERE book_id = (SELECT book_id FROM loans WHERE "
114                 "loan_id = %s)",
115                 (loan_id,))
116             self.conn.commit()
117             QMessageBox.information(self, 'Книга повернута', 'Книга успішно повернута до бібліотеки!')
118             self.load_loans()
119         except Exception as e:
120             self.conn.rollback()
121             QMessageBox.critical(self, 'Помилка', f'Сталася помилка при поверненні книги: {e}')
122         finally:
123             cur.close()

```

Рисунок 6.30 – 3 частина вмісту файлу loans_profile.py

```

main.py ×
1 import sys
2 from PyQt5.QtWidgets import QApplication
3 import welcome_menu
4
5 app = QApplication(sys.argv)
6 window = welcome_menu.MainWindow()
7 window.show()
8 sys.exit(app.exec_())

```

Рисунок 6.31 – Вмісту файлу main.py

```

main_page.py ×
1  from PyQt5.QtCore import Qt, QSize
2  from PyQt5.QtGui import QIcon
3  from PyQt5.QtWidgets import QMainWindow, QGridLayout, QWidget, QPushButton
4  import books_page
5  import users_page
6  import utils
7
8  4 usages
9  class MainPage(QMainWindow):
10     def __init__(self):
11         super().__init__()
12         self.setGeometry(100, 100, 300, 200)
13         self.setWindowTitle('Library Assistant')
14         self.setWindowIcon(QIcon('src/logo.png'))
15         utils.set_background(self)
16         self.initUI()
17
18     def initUI(self):
19         self.central_widget = QWidget()
20         self.setCentralWidget(self.central_widget)
21         self.setFixedSize(1200, 800)
22
23         grid_layout = QGridLayout()
24         grid_layout.setAlignment(Qt.AlignCenter)
25         self.central_widget.setLayout(grid_layout)
26
27         self.books_button = QPushButton('Каталог книг', self)
28         self.books_button.setMinimumSize(200, 50)
29         self.books_button.setIcon(QIcon('D:/LibraryAssistant/src/book.png'))
30         self.books_button.setIconSize(QSize(64, 64))
31         self.books_button.clicked.connect(self.goto_book_catalog)
32         self.books_button.setStyleSheet("""
33             QPushButton {
34                 font-weight: bold;
35                 background-color: #b88a71;
36                 color: white;
37                 border: none;
38                 padding: 15px 32px;
39                 text-align: center;
40                 text-decoration: none;
41                 display: inline-block;
42                 font-size: 16px;
43                 margin: 4px 2px;
44                 cursor: pointer;
45                 border-radius: 8px;
46                 min-width: 250px;
47                 max-width: 250px;
48                 min-height: 300px;
49                 max-height: 300px;
50             }
51             QPushButton:hover {
52                 background-color: #94644a;
53             }
54         """)

```

Рисунок 6.32 – 1 частина вмісту файлу main_page.py

```

51         background-color: #94644a;
52     }
53     """
54     grid_layout.addWidget(self.books_button, 0, 0)
55
56     self.visitors_button = QPushButton('Каталог користувачів', self)
57     self.visitors_button.setMinimumSize(200, 50)
58     self.visitors_button.setIcon(QIcon('D:/LibraryAssistant/src/user.png'))
59     self.visitors_button.setIconSize(QSize(64, 64))
60     self.visitors_button.clicked.connect(self.goto_visitor_catalog)
61     self.visitors_button.setStyleSheet("""
62         QPushButton {
63             font-weight: bold;
64             background-color: #b88a71;
65             color: white;
66             border: none;
67             padding: 15px 32px;
68             text-align: center;
69             text-decoration: none;
70             display: inline-block;
71             font-size: 16px;
72             margin: 4px 2px;
73             cursor: pointer;
74             border-radius: 8px;
75             min-width: 250px;
76             max-width: 250px;
77             min-height: 300px;
78             max-height: 300px;
79         }
80         QPushButton:hover {
81             background-color: #94644a;
82         }
83     """)
84     grid_layout.addWidget(self.visitors_button, 0, 1)
85
86     exit_btn = utils.create_exit_button(self)
87     exit_btn.clicked.connect(self.close)
88     exit_btn.move(1100, 730)
89
90     1 usage
91     def goto_book_catalog(self):
92         self.close()
93         self.book_catalog = books_page.LibraryCatalog()
94         self.book_catalog.show()
95
96     1 usage
97     def goto_visitor_catalog(self):
98         self.close()
99         self.users_page = users_page.UsersCatalog()
100        self.users_page.show()

```

Рисунок 6.33 – 2 частина вмісту файлу main_page.py

```

reg_menu.py x
1 import hashlib
2 import re
3 import psycpg2
4 from PyQt5.QtCore import Qt
5 from PyQt5.QtGui import QIcon
6 from PyQt5.QtWidgets import QLineEdit, QWidget, QPushButton, QVBoxLayout, QMessageBox, QSpacerItem, QSizePolicy
7 from utils import set_background, create_exit_button, create_info_button, show_creator_info, create_return_button
8 import creator_info
9 from auth_menu import AuthWindow
10
11 usage
12 class RegisterWindow(QWidget):
13     def __init__(self, main_window=None):
14         super().__init__()
15         self.main_window = main_window
16         self.initUI()
17
18     def initUI(self):
19         set_background(self)
20         self.setFixedSize(1200, 800)
21         self.setWindowIcon(QIcon('src/logo.png'))
22
23         line_edit_style = """
24             QLineEdit {
25                 font-size: 16px;
26                 border-radius: 10px;
27                 padding: 10px;
28                 background-color: white;
29                 color: black;
30                 min-width: 400px;
31                 max-width: 400px;
32             }
33         """
34
35         self.username = QLineEdit(self)
36         self.username.setPlaceholderText("Введіть ім'я користувача")
37         self.username.setStyleSheet(line_edit_style)
38
39         self.email = QLineEdit(self)
40         self.email.setPlaceholderText("Введіть електронну пошту")
41         self.email.setStyleSheet(line_edit_style)
42
43         self.password = QLineEdit(self)
44         self.password.setPlaceholderText("Введіть пароль")
45         self.password.setEchoMode(QLineEdit.Password)
46         self.password.setStyleSheet(line_edit_style)
47
48         self.registerButton = QPushButton("Зареєструватися", self)
49         self.registerButton.clicked.connect(self.register)
50         self.registerButton.setStyleSheet("""
51             QPushButton {
52                 font-weight: bold;
53                 background-color: #b88a71;

```

Рисунок 6.34 – 1 частина вмісту файлу reg_menu.py


```

reg_menu.py ×
51         font-weight: bold;
52         background-color: #b88a71;
53         color: white;
54         border: none;
55         padding: 10px;
56         text-align: center;
57         text-decoration: none;
58         display: inline-block;
59         font-size: 16px;
60         margin: 4px 2px;
61         cursor: pointer;
62         border-radius: 8px;
63         min-width: 400px;
64         max-width: 400px;
65     }
66     QPushButton:hover {
67         background-color: #94644a;
68     }
69     """
70
71     layout = QVBoxLayout(self)
72     spacer_top = QSpacerItem(20, 40, QSizePolicy.Minimum, QSizePolicy.Expanding)
73     layout.addItem(spacer_top)
74
75     layout.addWidget(self.username)
76     layout.setAlignment(self.username, Qt.AlignCenter)
77     layout.addWidget(self.email)
78     layout.setAlignment(self.email, Qt.AlignCenter)
79     layout.addWidget(self.password)
80     layout.setAlignment(self.password, Qt.AlignCenter)
81     layout.addWidget(self.registerButton)
82     layout.setAlignment(self.registerButton, Qt.AlignCenter)
83
84     spacer_bottom = QSpacerItem(20, 40, QSizePolicy.Minimum, QSizePolicy.Expanding)
85     layout.addItem(spacer_bottom)
86
87     exit_btn = create_exit_button(self)
88     exit_btn.clicked.connect(self.close)
89     exit_btn.move(1100, 730)
90
91     info_btn = create_info_button(self)
92     info_btn.clicked.connect(lambda: show_creator_info(self, self.creator_info_window))
93     info_btn.move(1000, 730)
94
95     back_btn = create_return_button(self)
96     back_btn.clicked.connect(self.go_back)
97     back_btn.move(50, 730)
98
99     self.creator_info_window = creator_info.CreatorInfoWindow()
100     self.setWindowTitle('Регістрація')
101     self.setLayout(layout)
102

```

Рисунок 6.34 – 2 частина вмісту файлу reg_menu.py

```

103     1 usage
104     def register(self):
105         username = self.username.text().strip()
106         email = self.email.text().strip()
107         password = self.password.text().strip()
108
109         if not username or not email or not password:
110             QMessageBox.critical(self, 'Невірне введення', 'Всі поля повинні бути заповнені!', QMessageBox.Ok,
111                                 QMessageBox.Ok)
112             return
113
114         if not re.match(r"^[^@]+@[^@]+\.[^@]+$", email):
115             QMessageBox.critical(self, 'Невірний email', 'Будь ласка, введіть валідний email.', QMessageBox.Ok,
116                                 QMessageBox.Ok)
117             return
118
119         if len(password) < 8:
120             QMessageBox.critical(self, 'Короткий пароль', 'Пароль повинен містити мінімум 8 символів.', QMessageBox.Ok,
121                                 QMessageBox.Ok)
122             return
123
124         password_hash = hashlib.sha256(password.encode()).hexdigest() # Хешування пароля
125
126         try:
127             with psycopg2.connect(user='postgres', password='1111', host='127.0.0.1', port='5432',
128                                 database='Library Assistant') as conn:
129                 with conn.cursor() as cursor:
130                     cursor.execute('SELECT * FROM users WHERE username = %s OR email = %s', (username, email))
131                     if cursor.fetchone():
132                         QMessageBox.critical(self, 'Дублікація даних',
133                                             'Користувач з таким іменем користувача або email вже існує.',
134                                             QMessageBox.Ok, QMessageBox.Ok)
135                         return
136
137                     cursor.execute('INSERT INTO users (username, email, password) VALUES (%s, %s, %s)',
138                                   (username, email, password_hash))
139
140                     conn.commit()
141                     QMessageBox.information(self, 'Успішно', 'Ви успішно зареєструвалися!', QMessageBox.Ok,
142                                           QMessageBox.Ok)
143                     self.open_auth_window()
144         except psycopg2.Error as e:
145             QMessageBox.critical(self, 'Помилка бази даних', 'Виникла помилка при роботі з базою даних: ' + str(e),
146                                 QMessageBox.Ok, QMessageBox.Ok)
147
148     1 usage
149     def open_auth_window(self):
150         self.auth_window = AuthWindow()
151         self.auth_window.show()
152         self.close()
153
154     1 usage
155     def go_back(self):
156         self.close()
157         if self.main_window:
158             self.main_window.show()

```

Рисунок 6.35 – 3 частина вмісту файлу reg_menu.py

```

users_page.py ×
1 import sys
2 import psycopg2
3 from PyQt5.QtCore import Qt
4 from PyQt5.QtGui import QIcon
5 from PyQt5.QtWidgets import (QMainWindow, QVBoxLayout, QWidget, QPushButton, QTableWidgetItem,
6                             QTableWidgetItem, QHBoxLayout, QMessageBox, QLineEdit, QLabel,
7                             QHeaderView, QGridLayout)
8
9 import utils
10 import edit_visitor
11 import add_visitor
12 import visitor_profile, main_page
13
14
15 usage
16 def create_connection():
17     try:
18         conn = psycopg2.connect(
19             host="localhost",
20             database="Library Assistant",
21             user="postgres",
22             password="1111")
23         return conn
24     except psycopg2.Error as e:
25         print(f"Помилка підключення до бази даних: {e}")
26         QMessageBox.critical(None, "Помилка підключення до бази даних", f"Помилка підключення до бази даних: {e}")
27         sys.exit(1)
28
29 usage
30 class UsersCatalog(QMainWindow):
31     def __init__(self):
32         super().__init__()
33         self.setGeometry(100, 100, 900, 600)
34         self.setWindowTitle('Користувачі бібліотеки')
35         self.setWindowIcon(QIcon('src/Logo.png'))
36         self.conn = create_connection()
37         self.current_page = 0
38         self.records_per_page = 10
39         utils.set_background(self)
40         self.initUI()
41
42     def initUI(self):
43         self.central_widget = QWidget()
44         self.setCentralWidget(self.central_widget)
45         self.setFixedSize(1200, 800)
46
47         label_style = ("background-color: #b88a71; color: white; font-size: 16px; border-radius: 8px; padding: 2px; "
48                       "min-width: 130px; max-width: 120px; font-weight: bold;")
49
50         line_edit_style = """
51             QLineEdit {
52                 font-size: 16px;

```

Рисунок 6.36 – 1 частина вмісту файлу users_page.py

```

users_page.py ×
50         QLineEdit {
51             font-size: 16px;
52             border-radius: 10px;
53             padding: 10px;
54             background-color: white;
55             color: black;
56             min-width: 200px;
57             max-width: 200px;
58         }
59         """
60
61     main_layout = QVBoxLayout(self.central_widget)
62
63     search_grid = QGridLayout()
64     self.first_name_search = QLineEdit()
65     self.first_name_search.setPlaceholderText("Пошук за іменем")
66     self.first_name_search.setStyleSheet(line_edit_style)
67     self.last_name_search = QLineEdit()
68     self.last_name_search.setPlaceholderText("Пошук за прізвищем")
69     self.last_name_search.setStyleSheet(line_edit_style)
70     self.email_search = QLineEdit()
71     self.email_search.setPlaceholderText("Пошук за емейлом")
72     self.email_search.setStyleSheet(line_edit_style)
73
74     search_grid.addWidget(QLabel("Ім'я:", styleSheet=label_style), 0, 0)
75     search_grid.addWidget(self.first_name_search, 0, 1)
76     search_grid.addWidget(QLabel("Прізвище:", styleSheet=label_style), 0, 2)
77     search_grid.addWidget(self.last_name_search, 0, 3)
78     search_grid.addWidget(QLabel("Емейл:", styleSheet=label_style), 1, 0)
79     search_grid.addWidget(self.email_search, 1, 1)
80
81     self.search_button = QPushButton("Пошук")
82     self.search_button.clicked.connect(self.load_data)
83     search_grid.addWidget(self.search_button, 1, 2, 1, 2)
84     self.search_button.setStyleSheet("""
85         QPushButton {
86             font-weight: bold;
87             background-color: #b88a71;
88             color: white;
89             border: none;
90             padding: 15px 32px;
91             text-align: center;
92             text-decoration: none;
93             display: inline-block;
94             font-size: 16px;
95             margin: 4px 2px;
96             cursor: pointer;
97             border-radius: 8px;
98         }
99         QPushButton:hover {
100             background-color: #94644a;
101         }
102     """)

```

Рисунок 6.37 – 2 частина вмісту файлу users_page.py

```

users_page.py ×
104 main_layout.addLayout(search_grid)
105
106 self.table = QTableWidgetItem(0, 7, self)
107 self.table.setStyleSheet("""
108     QTableWidgetItem {
109         background-color: #F5F5DC;
110         gridline-color: #000000;
111     }
112     QHeaderView::section {
113         background-color: #F5F5DC;
114     }
115     QTableWidgetItem::item {
116         background-color: #b88a71;
117     }
118     """)
119 self.table.setHorizontalHeaderLabels(["Ім'я", "Прізвище", "Емейл", "Телефон", "Змінити",
120     "Видалити", "Переглянути профіль"])
121 main_layout.addWidget(self.table)
122
123 self.table.horizontalHeader().setStretchLastSection(True)
124 self.table.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)
125 self.table.horizontalHeader().setSectionsClickable(False)
126 self.table.horizontalHeader().setSectionsMovable(False)
127
128 row_height = 30
129 self.table.verticalHeader().setDefaultSectionSize(row_height)
130 self.table.setFixedHeight(row_height * 11)
131
132 pagination_layout = QHBoxLayout()
133 self.prev_button = QPushButton('Попередня')
134 self.prev_button.clicked.connect(self.load_previous_page)
135 self.prev_button.setStyleSheet("""
136     QPushButton {
137         font-weight: bold;
138         background-color: #b88a71;
139         color: white;
140         border: none;
141         padding: 15px 32px;
142         text-align: center;
143         text-decoration: none;
144         display: inline-block;
145         font-size: 16px;
146         margin: 4px 2px;
147         cursor: pointer;
148         border-radius: 8px;
149     }
150     QPushButton:hover {
151         background-color: #94644a;
152     }
153     """)
154 self.next_button = QPushButton('Наступна')
155 self.next_button.clicked.connect(self.load_next_page)
156 self.next_button.setStyleSheet("""

```

Рисунок 6.38 – 3 частина вмісту файлу users_page.py

```

users_page.py ×
155 self.next_button.clicked.connect(self.load_next_page)
156 self.next_button.setStyleSheet("""
157     QPushButton {
158         font-weight: bold;
159         background-color: #b88a71;
160         color: white;
161         border: none;
162         padding: 15px 32px;
163         text-align: center;
164         text-decoration: none;
165
166         display: inline-block;
167         font-size: 16px;
168         margin: 4px 2px;
169         cursor: pointer;
170         border-radius: 8px;
171     }
172     QPushButton:hover {
173         background-color: #94644a;
174     }
175     """)
176 pagination_layout.addWidget(self.prev_button)
177 pagination_layout.addWidget(self.next_button)
178 main_layout.addLayout(pagination_layout)
179
180 add_visitor_layout = QHBoxLayout()
181 self.add_visitor_button = QPushButton('Додати користувача')
182 self.add_visitor_button.clicked.connect(self.open_add_visitor_dialog)
183 self.add_visitor_button.setStyleSheet("""
184     QPushButton {
185         font-weight: bold;
186         background-color: #b88a71;
187         color: white;
188         border: none;
189         padding: 15px 32px;
190         text-align: center;
191         text-decoration: none;
192         display: inline-block;
193         font-size: 16px;
194         margin: 4px 2px;
195         cursor: pointer;
196         border-radius: 8px;
197     }
198     QPushButton:hover {
199         background-color: #94644a;
200     }
201     """)
202 add_visitor_layout.addStretch()
203 add_visitor_layout.addWidget(self.add_visitor_button)
204 add_visitor_layout.addStretch()
205 self.add_visitor_button.setFixedSize(500, 70)
206 main_layout.addLayout(add_visitor_layout)
207

```

Рисунок 6.39 – 4 частина вмісту файлу users_page.py

```

users_page.py ×
206     main_layout.addLayout(add_visitor_layout)
207
208     exit_btn = utils.create_exit_button(self)
209     exit_btn.clicked.connect(self.close)
210     exit_btn.move(1100, 730)
211
212     back_btn = utils.create_return_button(self)
213     back_btn.clicked.connect(self.go_to_main_page)
214     back_btn.move(50, 730)
215
216     self.load_data()
217
218     5 usages
219     def load_data(self):
220         conditions = []
221         parameters = []
222         if self.first_name_search.text().strip():
223             conditions.append("first_name ILIKE %s")
224             parameters.append(f"%{self.first_name_search.text().strip()}%")
225         if self.last_name_search.text().strip():
226             conditions.append("last_name ILIKE %s")
227             parameters.append(f"%{self.last_name_search.text().strip()}%")
228         if self.email_search.text().strip():
229             conditions.append("email ILIKE %s")
230             parameters.append(f"%{self.email_search.text().strip()}%")
231
232         query = "SELECT visitor_id, first_name, last_name, email, phone_number FROM visitors"
233         if conditions:
234             query += " WHERE " + " AND ".join(conditions)
235         query += " ORDER BY visitor_id ASC LIMIT %s OFFSET %s;"
236
237         offset = self.current_page * self.records_per_page
238         parameters.extend([self.records_per_page + 1, offset])
239
240         cur = self.conn.cursor()
241         cur.execute(query, parameters)
242         records = cur.fetchall()
243         cur.close()
244
245         more_pages_available = len(records) > self.records_per_page
246         if more_pages_available:
247             records = records[:-1]
248
249         self.table.setRowCount(len(records))
250         for row_number, row_data in enumerate(records):
251             for column_number, data in enumerate(row_data[1:], 1):
252                 item = QTableWidgetItem(str(data))
253                 item.setFlags(item.flags() & ~Qt.ItemIsEditable)
254                 self.table.setItem(row_number, column_number - 1, item)
255
256         edit_btn = QPushButton('Змінити')
257         edit_btn.clicked.connect(lambda _, visitor_id=row_data[0]: self.edit_user(visitor_id))
258         edit_btn.setStyleSheet("""

```

Рисунок 6.39 – 5 частина вмісту файлу users_page.py

```

users_page.py ×
258                                     QPushButton {
259                                     |     background-color: #94644a;
260                                     | }
261                                     """
262     self.table.setCellWidget(row_number, 4, edit_btn)
263
264     delete_btn = QPushButton('Видалити')
265     delete_btn.clicked.connect(lambda _, visitor_id=row_data[0]: self.delete_user(visitor_id))
266     delete_btn.setStyleSheet("""
267                                     QPushButton {
268                                     |     background-color: #94644a;
269                                     | }
270                                     """
271
272     self.table.setCellWidget(row_number, 5, delete_btn)
273
274     view_profile_btn = QPushButton('Переглянути профіль')
275     view_profile_btn.clicked.connect(lambda _, visitor_id=row_data[0]: self.view_profile(visitor_id))
276     view_profile_btn.setStyleSheet("""
277                                     QPushButton {
278                                     |     background-color: #94644a;
279                                     | }
280                                     """
281
282     self.table.setCellWidget(row_number, 6, view_profile_btn)
283     self.update_navigation_buttons()
284
285 1 usage
286 def update_navigation_buttons(self):
287     cur = self.conn.cursor()
288     cur.execute("SELECT COUNT(*) FROM visitors")
289     total_records = cur.fetchone()[0]
290     cur.close()
291
292     max_pages = (total_records + self.records_per_page - 1) // self.records_per_page
293     self.next_button.setEnabled(self.current_page < max_pages - 1)
294     self.prev_button.setEnabled(self.current_page > 0)
295
296 1 usage
297 def go_to_main_page(self):
298     self.main_page = main_page.MainPage()
299     self.main_page.show()
300     self.close()
301
302 1 usage
303 def open_add_visitor_dialog(self):
304     self.add_visitor_window = add_visitor.AddVisitorDialog()
305     self.add_visitor_window.show()
306
307 1 usage
308 def load_next_page(self):
309     self.current_page += 1
310     self.load_data()
311

```

Рисунок 6.39 – 6 частина вмісту файлу users_page.py


```
306 def load_previous_page(self):
307     if self.current_page > 0:
308         self.current_page -= 1
309         self.load_data()
310
311 1 usage
311 def edit_user(self, visitor_id):
312     self.edit_window = edit_visitor.EditVisitorDialog(visitor_id)
313     self.edit_window.show()
314
315 1 usage
315 def delete_user(self, visitor_id):
316     reply = QMessageBox.question(self, 'Підтвердіть видалення',
317                                 'Ви впевнені, що хочете видалити даного користувача?',
318                                 QMessageBox.Yes | QMessageBox.No, QMessageBox.No)
319     if reply == QMessageBox.Yes:
320         cur = self.conn.cursor()
321         cur.execute("DELETE FROM visitors WHERE visitor_id = %s;", (visitor_id,))
322         self.conn.commit()
323         cur.close()
324         self.load_data()
325
326 1 usage
326 def view_profile(self, visitor_id):
327     self.profile_dialog = visitor_profile.VisitorProfileDialog(visitor_id, self.conn)
328     self.profile_dialog.show()
```

Рисунок 6.40 – 7 частина вмісту файлу users_page.py

```

utils.py ×
1  from PyQt5.QtWidgets import QPushButton, QLabel
2  from PyQt5.QtGui import QIcon
3  from PyQt5.QtCore import QSize
4
5
6  def set_background(widget):
7      background_label = QLabel(widget)
8      background_label.setGeometry(0, 0, 1200, 800)
9      background_label.setStyleSheet("background-image: url('D:/LibraryAssistant/src/background.jpg');")
10
11
12  def create_exit_button(parent_widget):
13      exit_btn = QPushButton(parent_widget)
14      exit_btn.setIcon(QIcon('D:/LibraryAssistant/src/exit.png'))
15      exit_btn.setIconSize(QSize(50, 50))
16      exit_btn.setFixedSize(50, 50)
17      return exit_btn
18
19
20  6 usages
21  def create_info_button(parent_widget):
22      info_btn = QPushButton(parent_widget)
23      info_btn.setIcon(QIcon('D:/LibraryAssistant/src/info.png'))
24      info_btn.setIconSize(QSize(50, 50))
25      info_btn.setFixedSize(50, 50)
26      return info_btn
27
28  8 usages
29  def create_return_button(parent_widget):
30      return_btn = QPushButton(parent_widget)
31      return_btn.setIcon(QIcon('D:/LibraryAssistant/src/return_welcome.png'))
32      return_btn.setIconSize(QSize(50, 50))
33      return_btn.setFixedSize(50, 50)
34      return return_btn
35
36  6 usages
37  def show_creator_info(current_window, creator_info_window):
38      current_window.close()
39      creator_info_window.show()
40
41  def show_welcome_window(current_window, welcome_window):
42      current_window.close()
43      welcome_window.show()
44
45
46  def open_auth_window(parent_widget):
47      parent_widget.close()
48      parent_widget.auth_window.show()
49

```

Рисунок 6.41 – Вмісту файлу utils.py

```

visitor_profile.py ×
1  from PyQt5.QtCore import Qt
2  from PyQt5.QtGui import QIcon
3  from PyQt5.QtWidgets import QPushButton, QLabel, QVBoxLayout, QDialog
4  import issue_book, loans_profile, utils
5
6  1 usage
7  class VisitorProfileDialog(QDialog):
8      def __init__(self, visitor_id, conn):
9          super().__init__()
10         self.visitor_id = visitor_id
11         self.conn = conn
12         self.setGeometry(100, 100, 900, 600)
13         self.setWindowIcon(QIcon('src/Logo.png'))
14         utils.set_background(self)
15         self.initUI()
16
17     def initUI(self):
18         self.setWindowTitle('Профіль відвідувача')
19         layout = QVBoxLayout(self)
20         self.setFixedSize(1200, 800)
21
22         layout.setContentsMargins(10, 10, 10, 10)
23         layout.setSpacing(5)
24
25         button_style = """
26             QPushButton {
27                 font-weight: bold;
28                 background-color: #b88a71;
29                 color: white;
30                 border: none;
31                 padding: 15px 32px;
32                 text-align: center;
33                 text-decoration: none;
34                 display: inline-block;
35                 font-size: 16px;
36                 margin: 4px 2px;
37                 cursor: pointer;
38                 border-radius: 8px;
39                 min-width: 350px;
40                 max-width: 350px;
41             }
42             QPushButton:hover {
43                 background-color: #94644a;
44             }
45         """
46
47         cur = self.conn.cursor()
48         cur.execute("SELECT first_name, last_name, email, phone_number FROM visitors WHERE visitor_id = %s",
49                   (self.visitor_id,))
50         data = cur.fetchone()
51         cur.close()
52
53         info_label_style = ("background-color: #b88a71; color: white; font-size: 16px; "

```

Рисунок 6.42 – 1 частина вмісту файлу visitor_profile.py

```

53         "border-radius: 8px; padding: 10px 5px; min-width: 400px; "
54         "max-width: 400px; font-weight: bold; text-align: center;"
55     layout.addStretch(1)
56     info_label = QLabel(f"Ім'я: {data[0]}\nПрізвище: {data[1]}\nEmail: {data[2]}\nТелефон: {data[3]}",
57                       styleSheet=info_label_style)
58     layout.addWidget(info_label, alignment=Qt.AlignCenter)
59
60     view_loans_button = QPushButton('Видані книги')
61     view_loans_button.setStyleSheet(button_style)
62     view_loans_button.clicked.connect(self.view_loans)
63     layout.addWidget(view_loans_button, alignment=Qt.AlignCenter)
64
65     issue_book_button = QPushButton('Видати книги')
66     issue_book_button.setStyleSheet(button_style)
67     issue_book_button.clicked.connect(self.issue_book)
68     layout.addWidget(issue_book_button, alignment=Qt.AlignCenter)
69
70     layout.addStretch(1)
71
72     exit_btn = utils.create_exit_button(self)
73     exit_btn.clicked.connect(self.close)
74     exit_btn.move(1100, 730)
75
76     1 usage
77     def view_loans(self):
78         self.loans_dialog = loans_profile.LoansDialog(self.visitor_id, self.conn)
79         self.loans_dialog.show()
80
81     1 usage
82     def issue_book(self):
83         self.issue_dialog = issue_book.IssueBookDialog(self.visitor_id, self.conn)
84         self.issue_dialog.show()

```

Рисунок 6.43 – 2 частина вмісту файлу visitor_profile.py

```
welcome_menu.py ×
1 from PyQt5.QtWidgets import QWidget, QPushButton, QVBoxLayout
2 from PyQt5.QtGui import QIcon
3 from PyQt5.QtCore import Qt
4 from utils import set_background, create_exit_button, create_info_button, show_creator_info
5 import creator_info
6 import auth_menu
7 import reg_menu
8
9 usage
10 class MainWindow(QWidget):
11     def __init__(self):
12         super().__init__()
13         self.initUI()
14
15     def initUI(self):
16         self.setFixedSize(1200, 800)
17         set_background(self)
18         self.setWindowTitle("Library Assistant")
19         self.setWindowIcon(QIcon('src/logo.png'))
20
21         layout = QVBoxLayout(self)
22         layout.setAlignment(Qt.AlignHCenter)
23
24         self.auth_button = QPushButton('Авторизація', self)
25         self.auth_button.setFixedSize(450, 150)
26         self.auth_button.clicked.connect(self.open_auth_window)
27         self.auth_button.setStyleSheet("""
28             QPushButton {
29                 font-weight: bold;
30                 background-color: #b88a71;
31                 color: white;
32                 border: none;
33                 padding: 15px 32px;
34                 text-align: center;
35                 text-decoration: none;
36                 display: inline-block;
37                 font-size: 16px;
38                 margin: 4px 2px;
39                 cursor: pointer;
40                 border-radius: 8px;
41             }
42             QPushButton:hover {
43                 background-color: #94644a;
44             }
45         """)
46         layout.addWidget(self.auth_button)
47
48         self.reg_button = QPushButton('Реєстрація', self)
49         self.reg_button.setFixedSize(450, 150)
50         self.reg_button.clicked.connect(self.open_reg_window)
51         self.reg_button.setStyleSheet("""
52             QPushButton {
53                 font-weight: bold;
```

Рисунок 6.44 – 1 частина вмісту файлу welcome_menu.py

```

53         background-color: #b88a71;
54         color: white;
55         border: none;
56         padding: 15px 32px;
57         text-align: center;
58         text-decoration: none;
59         display: inline-block;
60         font-size: 16px;
61         margin: 4px 2px;
62         cursor: pointer;
63         border-radius: 8px;
64     }
65     QPushButton:hover {
66         background-color: #94644a;
67     }
68     """
69     layout.addWidget(self.reg_button)
70
71     exit_btn = create_exit_button(self)
72     exit_btn.clicked.connect(self.close)
73     exit_btn.move(1100, 730)
74
75     info_btn = create_info_button(self)
76     info_btn.clicked.connect(lambda: show_creator_info(self, self.creator_info_window))
77     info_btn.move(1000, 730)
78
79     self.creator_info_window = creator_info.CreatorInfoWindow(self)
80     self.auth_window = auth_menu.AuthWindow(self)
81     self.reg_window = reg_menu.RegisterWindow(self)
82     self.setLayout(layout)
83
84     1 usage
85     def open_auth_window(self):
86         self.close()
87         self.auth_window.show()
88
89     1 usage
90     def open_reg_window(self):
91         self.close()
92         self.reg_window.show()

```

Рисунок 6.45 – 2 частина вмісту файлу welcome_menu.py