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	*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 отриманих результатів*		
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

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

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 масштабування 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 запинок 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 проєктами.

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 декого 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: Хочу 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.

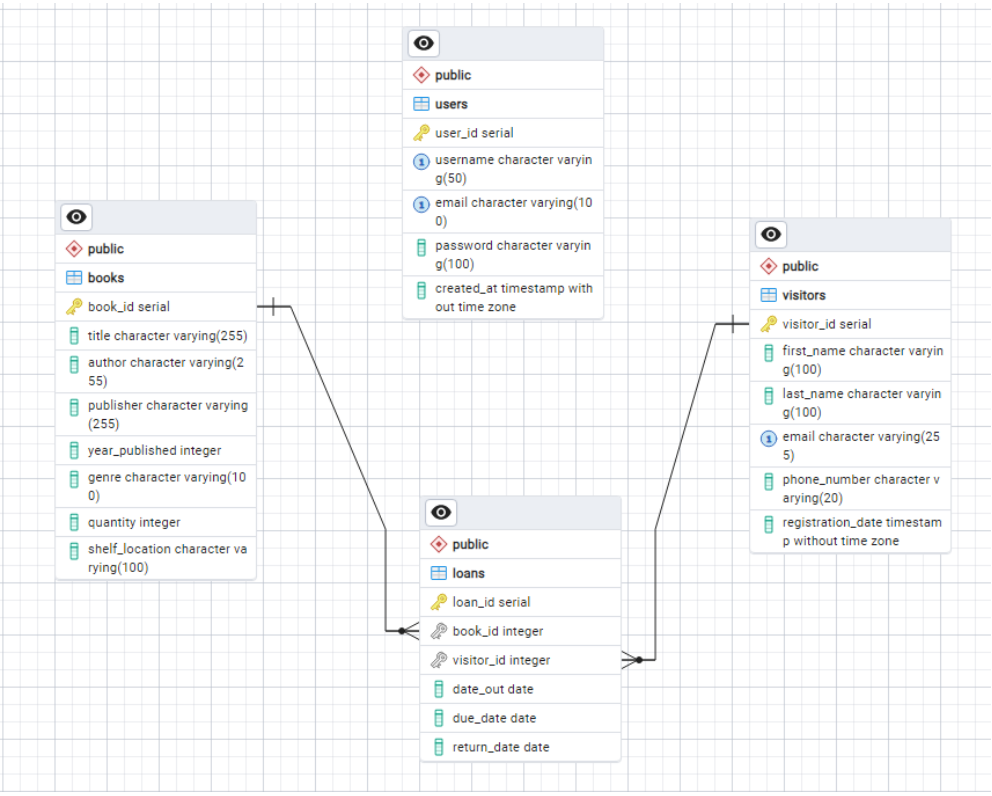


Рисунок 3.1 – ER-діаграма бази даних

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.

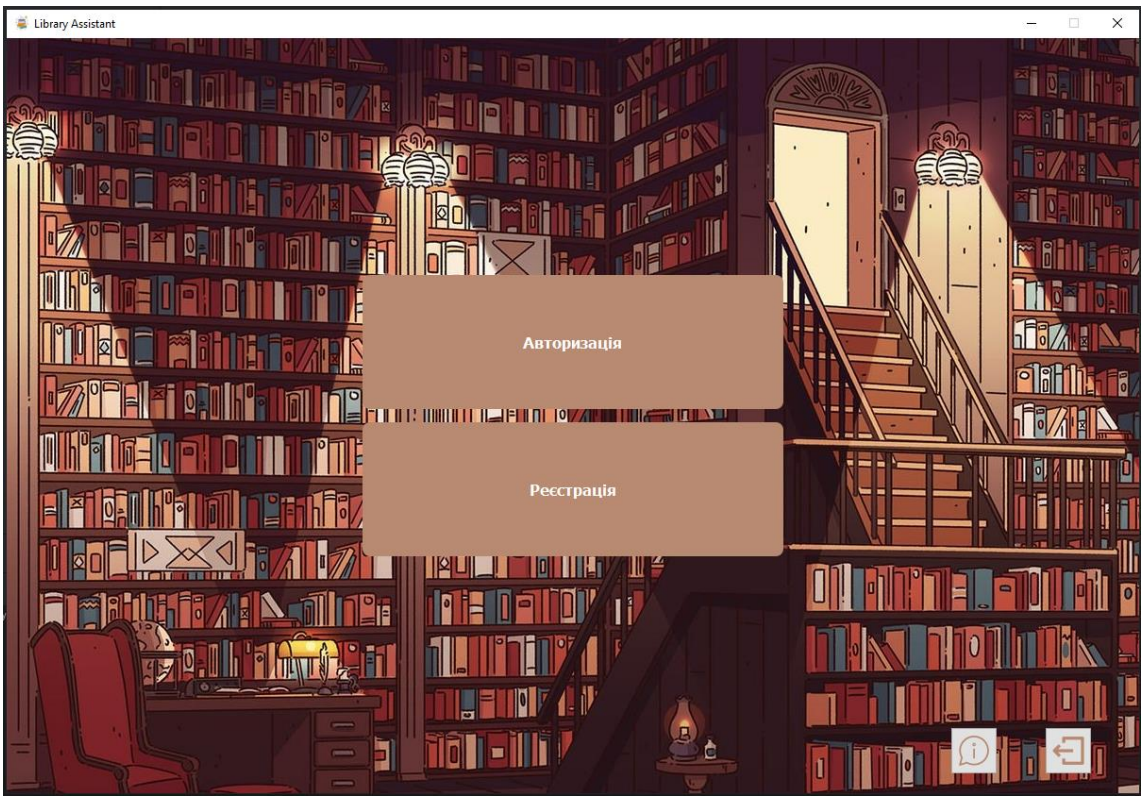


Рисунок 3.2 – Початкове меню для вибору способу автентифікації

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.

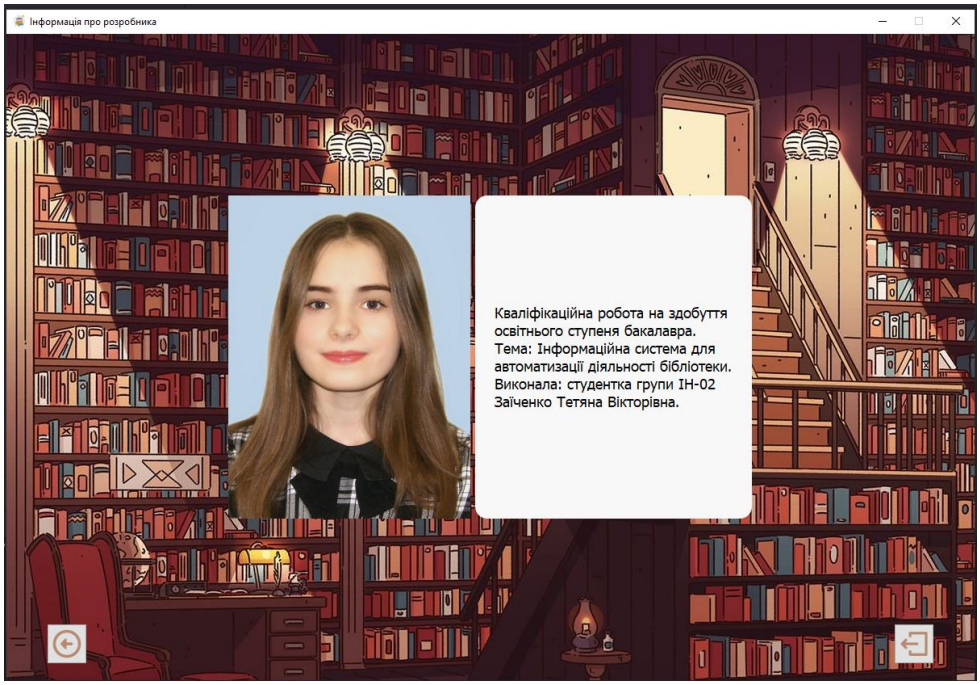


Рисунок 3.3 – Сторінка з інформацією про розробника

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.

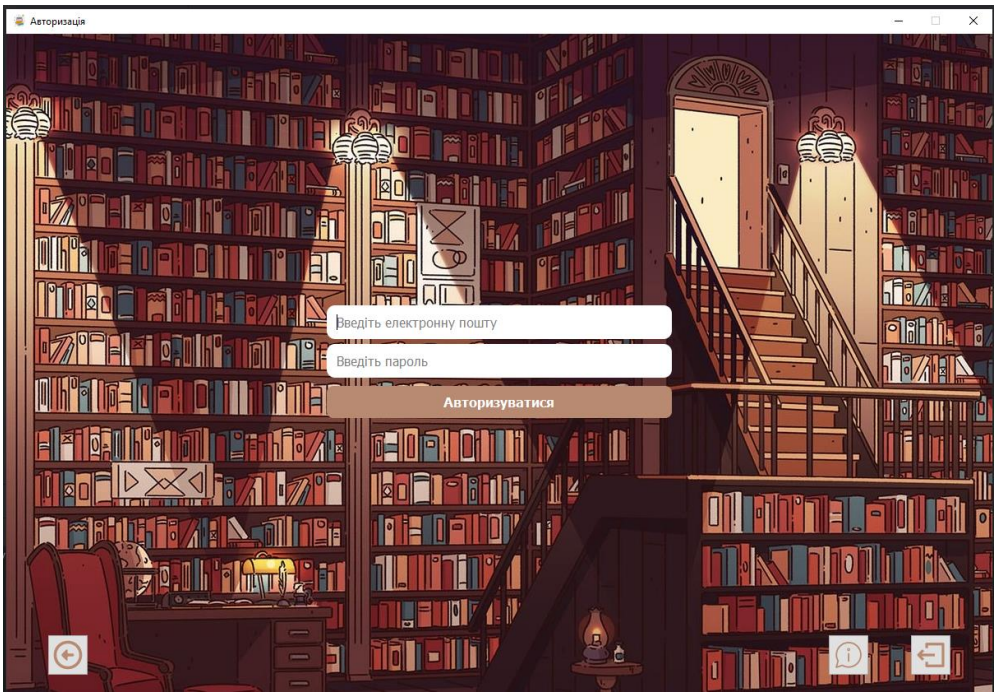


Рисунок 3.4 – Сторінка для авторизації користувача

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 виведене повідомлення про успішну реєстрацію та відкриється вікно авторизації.

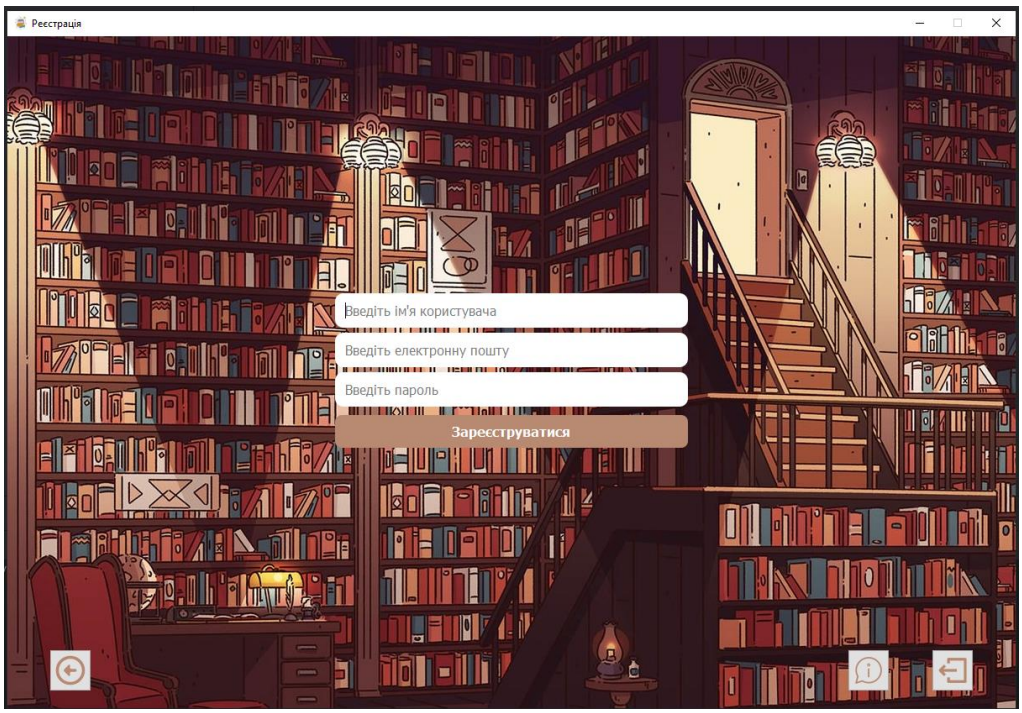


Рисунок 3.5 – Сторінка для реєстрації користувача

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.

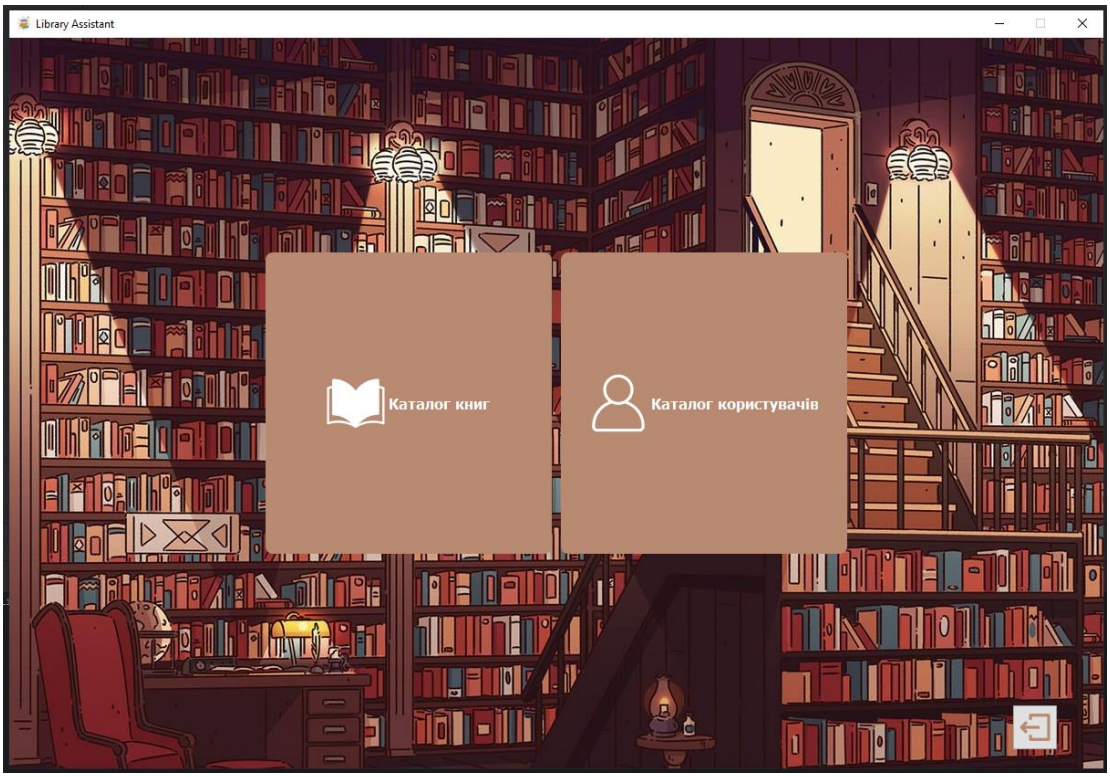
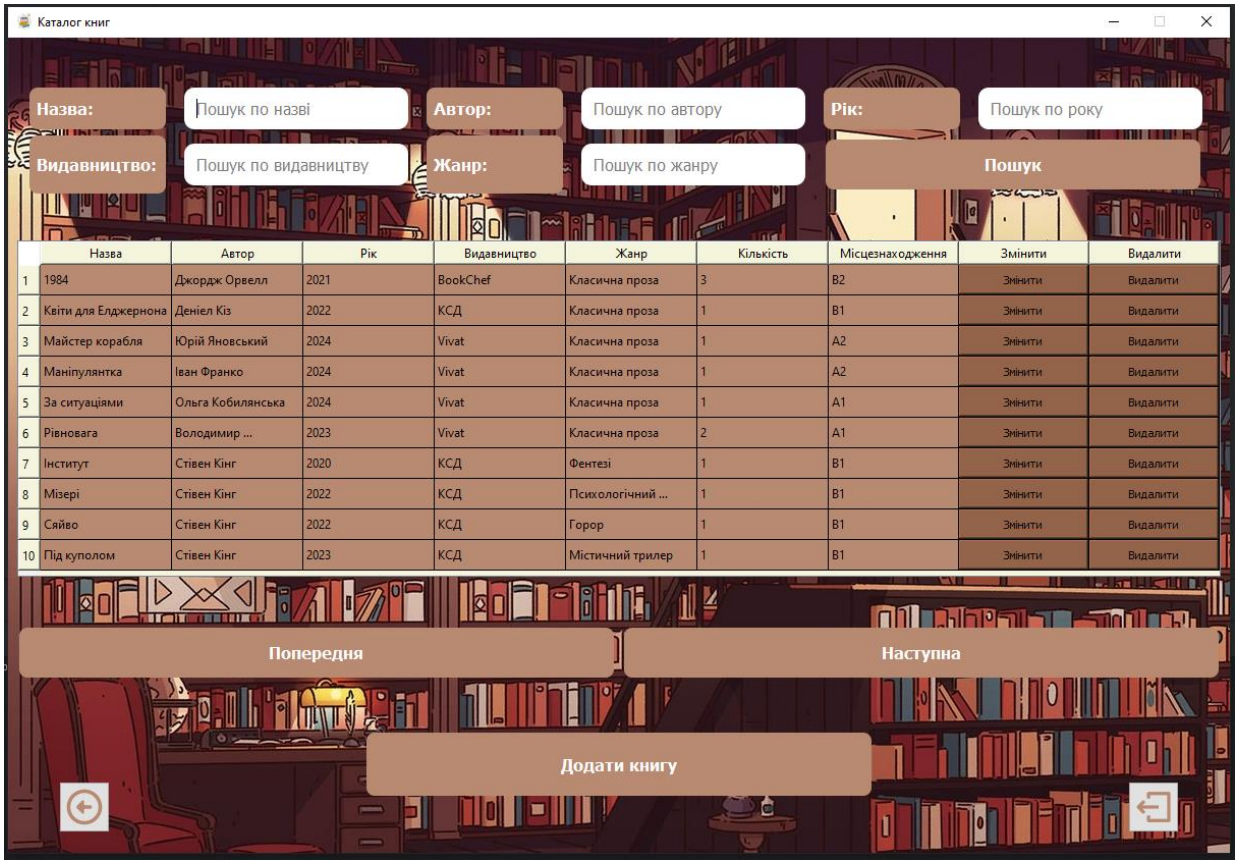


Рисунок 3.6 – Головне меню додатку

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.



	Назва	Автор	Рік	Видавництво	Жанр	Кількість	Місцезнаходження	Змінити	Видалити
1	1984	Джордж Орвелл	2021	BookChef	Класична проза	3	B2	Змінити	Видалити
2	Квіти для Елджернона	Деніел Кіз	2022	КСД	Класична проза	1	B1	Змінити	Видалити
3	Майстер корабля	Юрій Яновський	2024	Vivat	Класична проза	1	A2	Змінити	Видалити
4	Маніпулянтка	Іван Франко	2024	Vivat	Класична проза	1	A2	Змінити	Видалити
5	За ситуаціями	Ольга Кобилянська	2024	Vivat	Класична проза	1	A1	Змінити	Видалити
6	Рівновага	Володимир ...	2023	Vivat	Класична проза	2	A1	Змінити	Видалити
7	Інститут	Стівен Кінг	2020	КСД	Фентезі	1	B1	Змінити	Видалити
8	Мізері	Стівен Кінг	2022	КСД	Психологічний ...	1	B1	Змінити	Видалити
9	Сяйво	Стівен Кінг	2022	КСД	Горор	1	B1	Змінити	Видалити
10	Під куполом	Стівен Кінг	2023	КСД	Містичний трилер	1	B1	Змінити	Видалити

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 критерії 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.

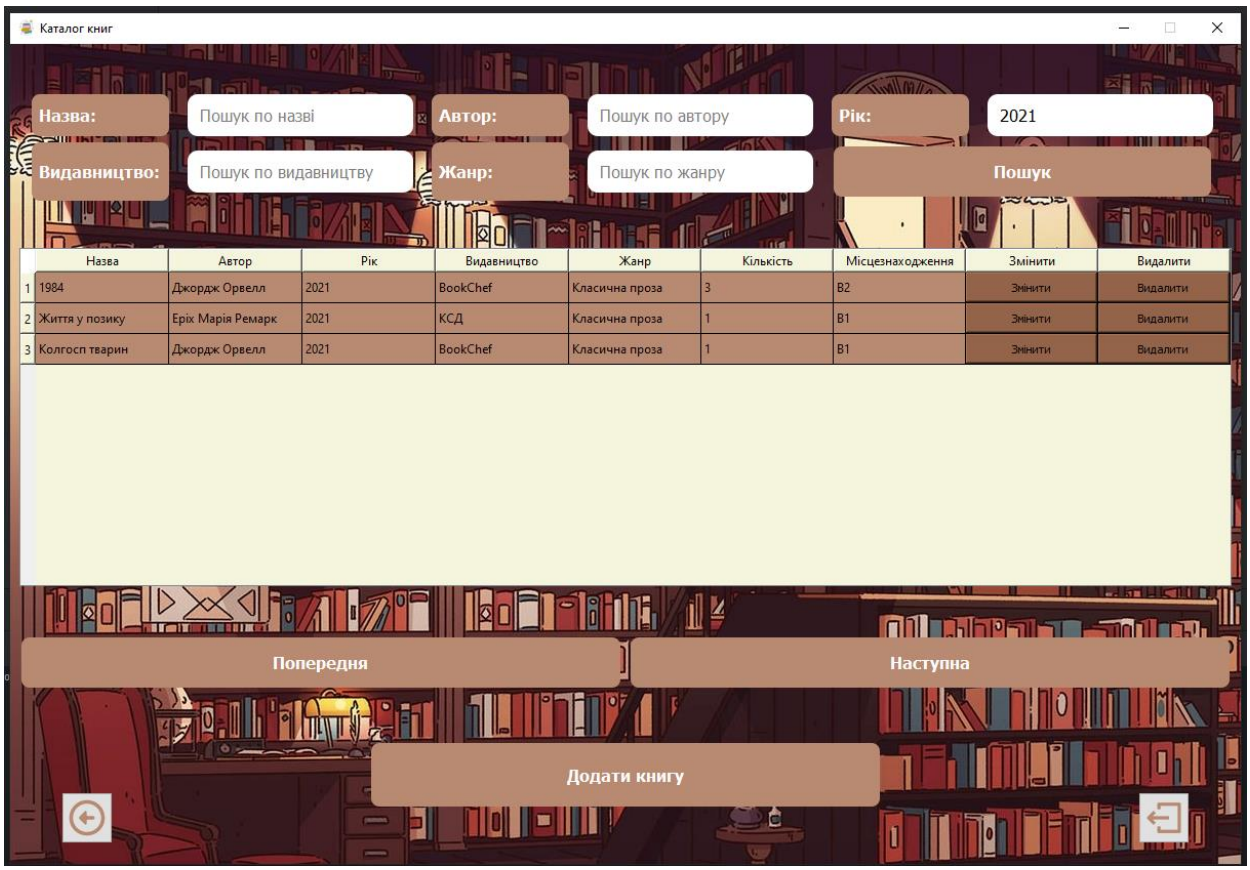


Рисунок 3.8 – Демонстрація пошуку книги за критерієм

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.

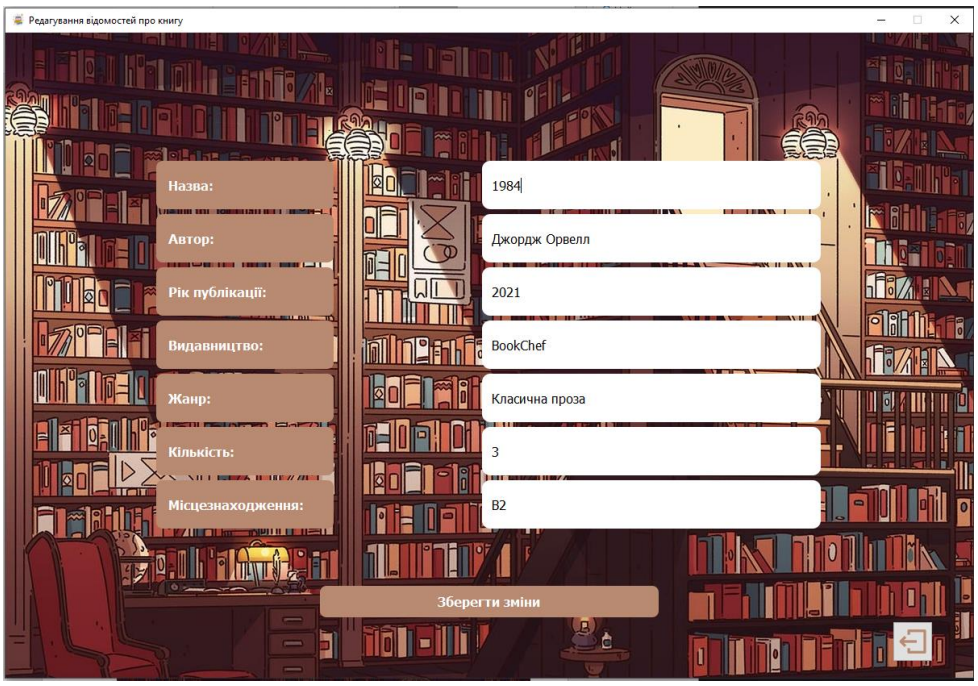


Рисунок 3.9 – Меню редагування інформації про книгу

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.

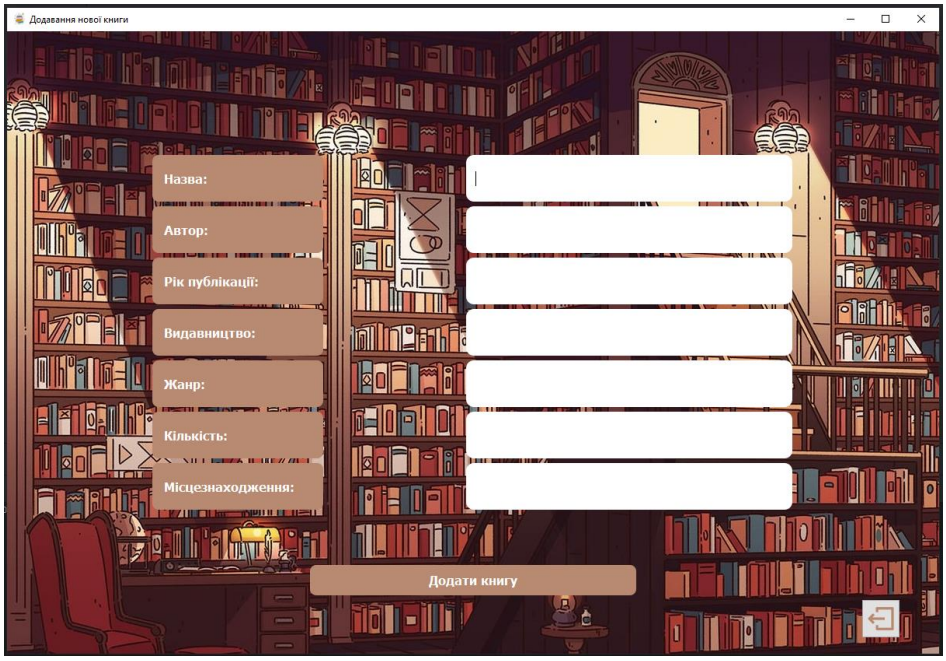
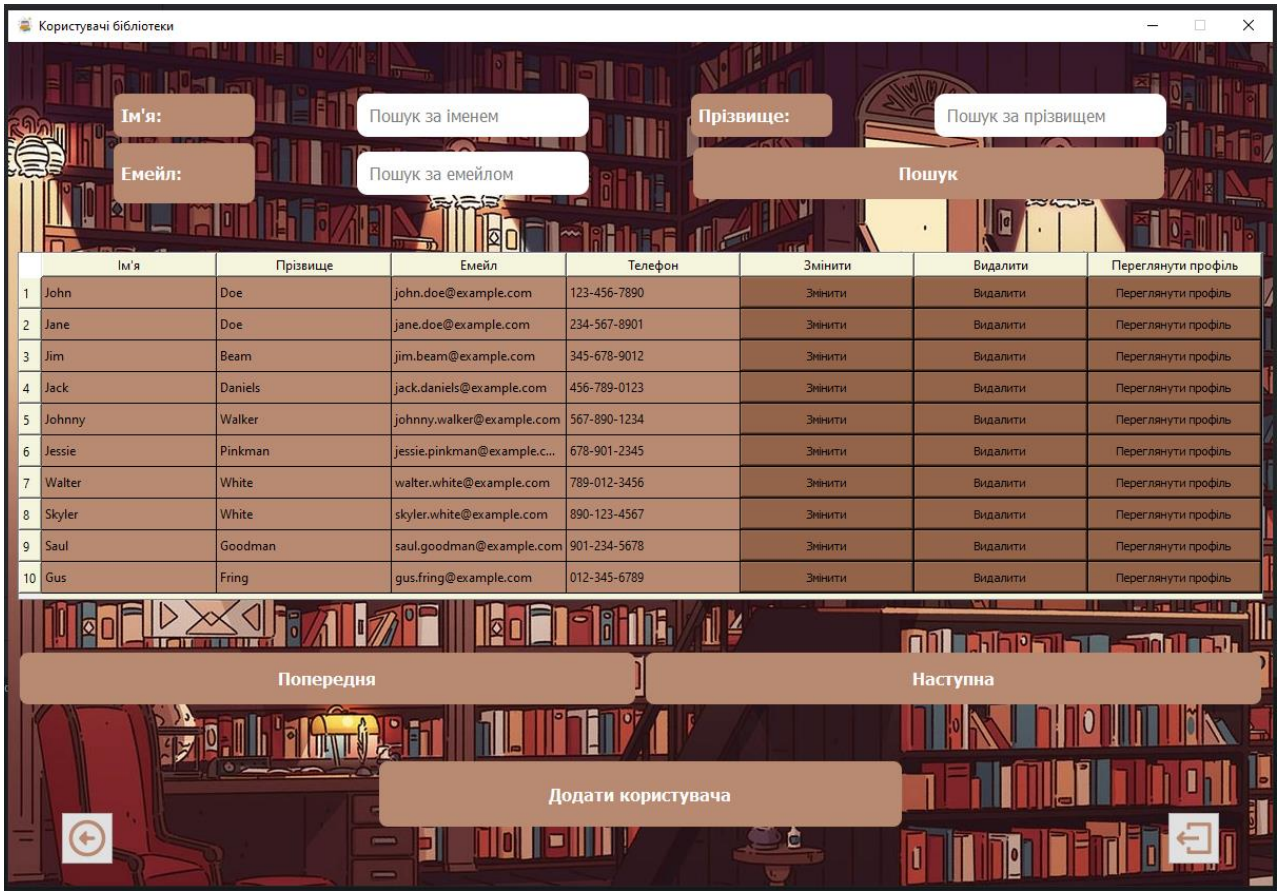


Рисунок 3.10 – Меню додавання нової книги до каталогу

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



	Ім'я	Прізвище	Емейл	Телефон	Змінити	Видалити	Переглянути профіль
1	John	Doe	john.doe@example.com	123-456-7890	Змінити	Видалити	Переглянути профіль
2	Jane	Doe	jane.doe@example.com	234-567-8901	Змінити	Видалити	Переглянути профіль
3	Jim	Beam	jim.beam@example.com	345-678-9012	Змінити	Видалити	Переглянути профіль
4	Jack	Daniels	jack.daniels@example.com	456-789-0123	Змінити	Видалити	Переглянути профіль
5	Johnny	Walker	johnny.walker@example.com	567-890-1234	Змінити	Видалити	Переглянути профіль
6	Jessie	Pinkman	jessie.pinkman@example.c...	678-901-2345	Змінити	Видалити	Переглянути профіль
7	Walter	White	walter.white@example.com	789-012-3456	Змінити	Видалити	Переглянути профіль
8	Skyler	White	skyler.white@example.com	890-123-4567	Змінити	Видалити	Переглянути профіль
9	Saul	Goodman	saul.goodman@example.com	901-234-5678	Змінити	Видалити	Переглянути профіль
10	Gus	Fring	gus.fring@example.com	012-345-6789	Змінити	Видалити	Переглянути профіль

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».

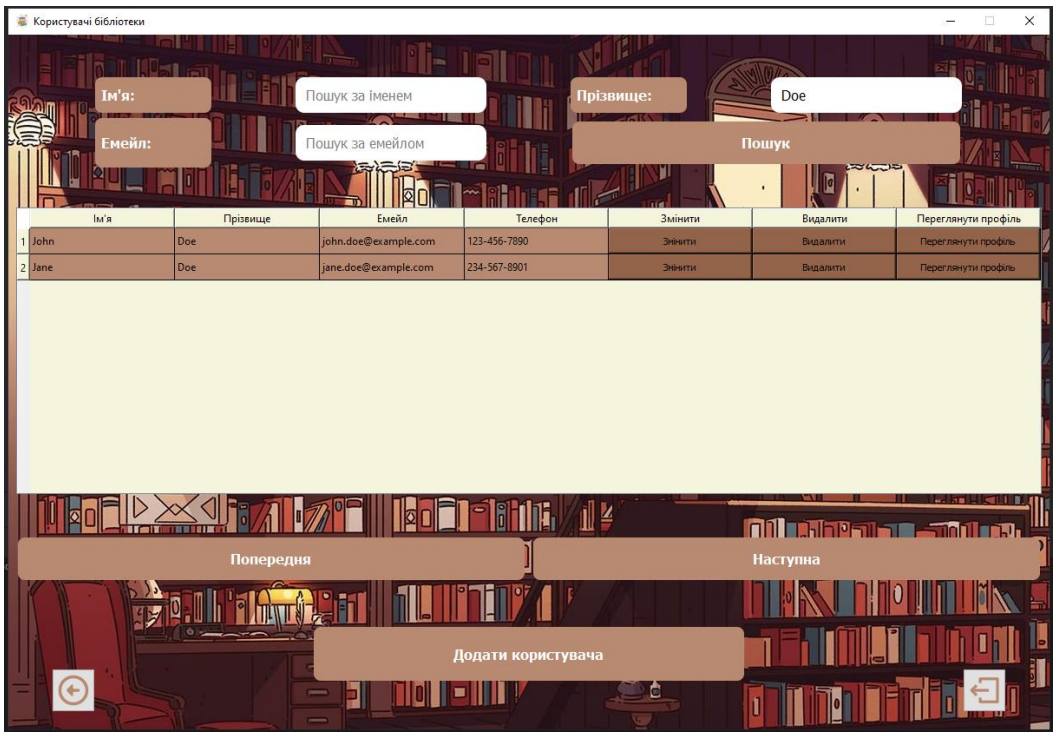


Рисунок 3.12 – Демонстрація пошуку користувачів за критерієм

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».

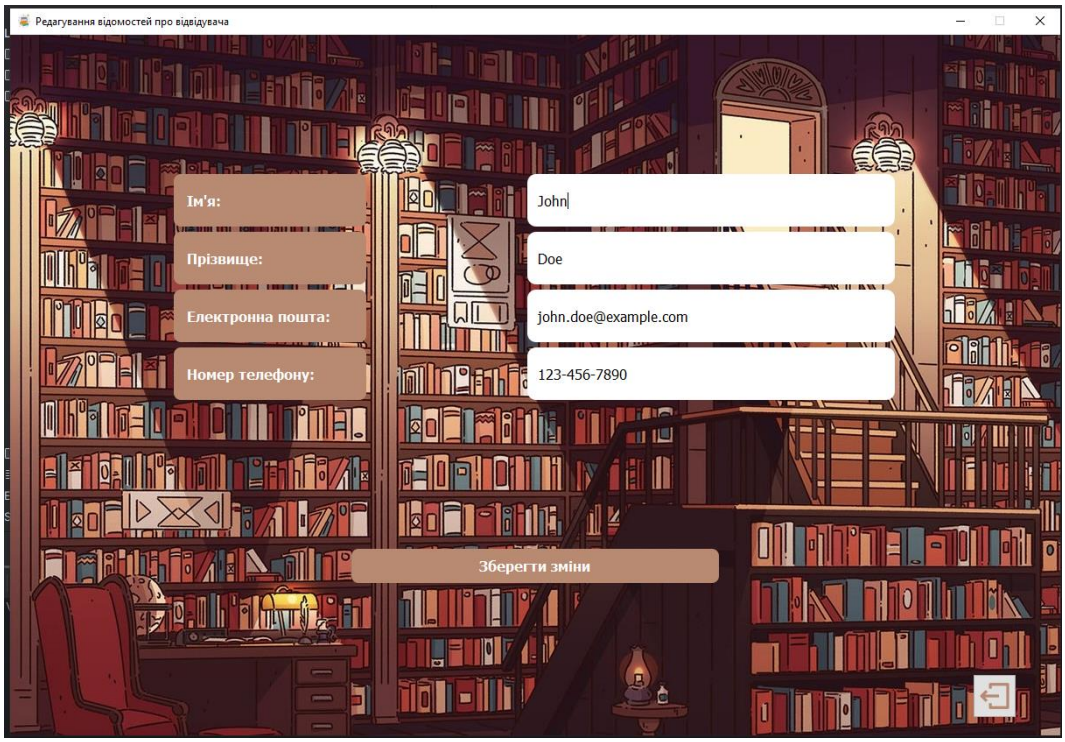


Рисунок 3.13 – Меню редагування інформації про користувача

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.

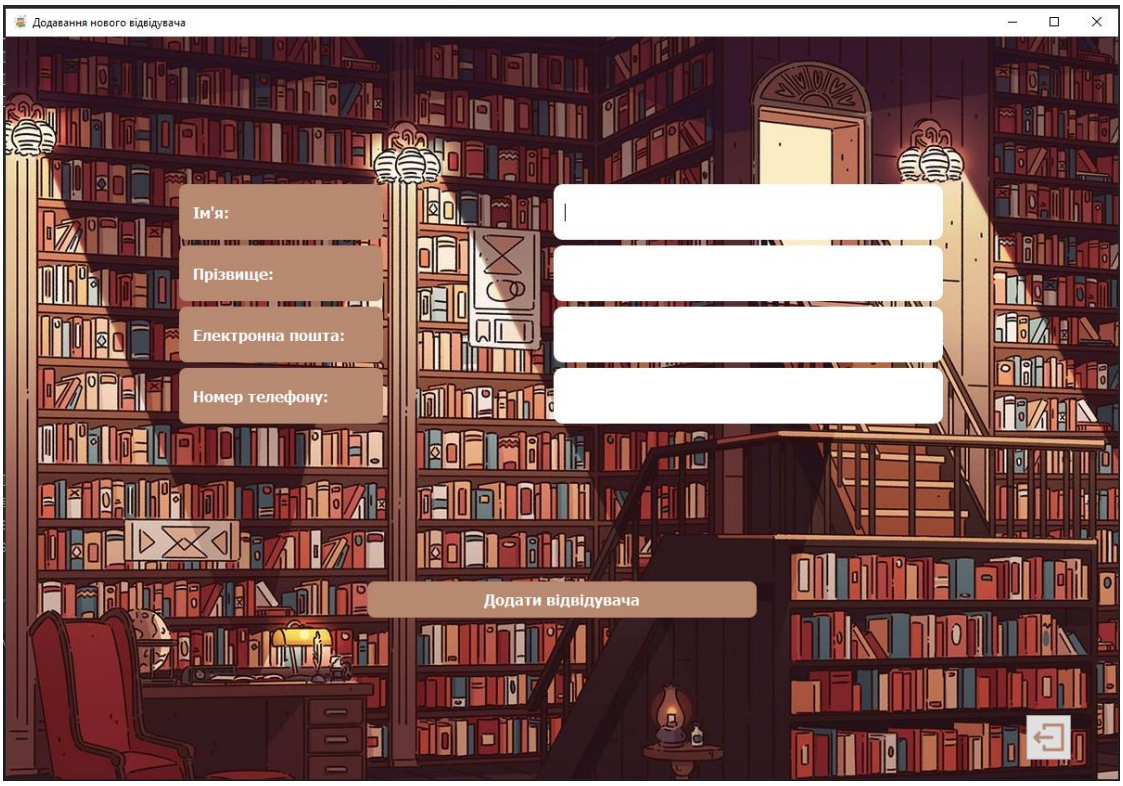


Рисунок 3.14 – Меню додавання нового користувача

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».

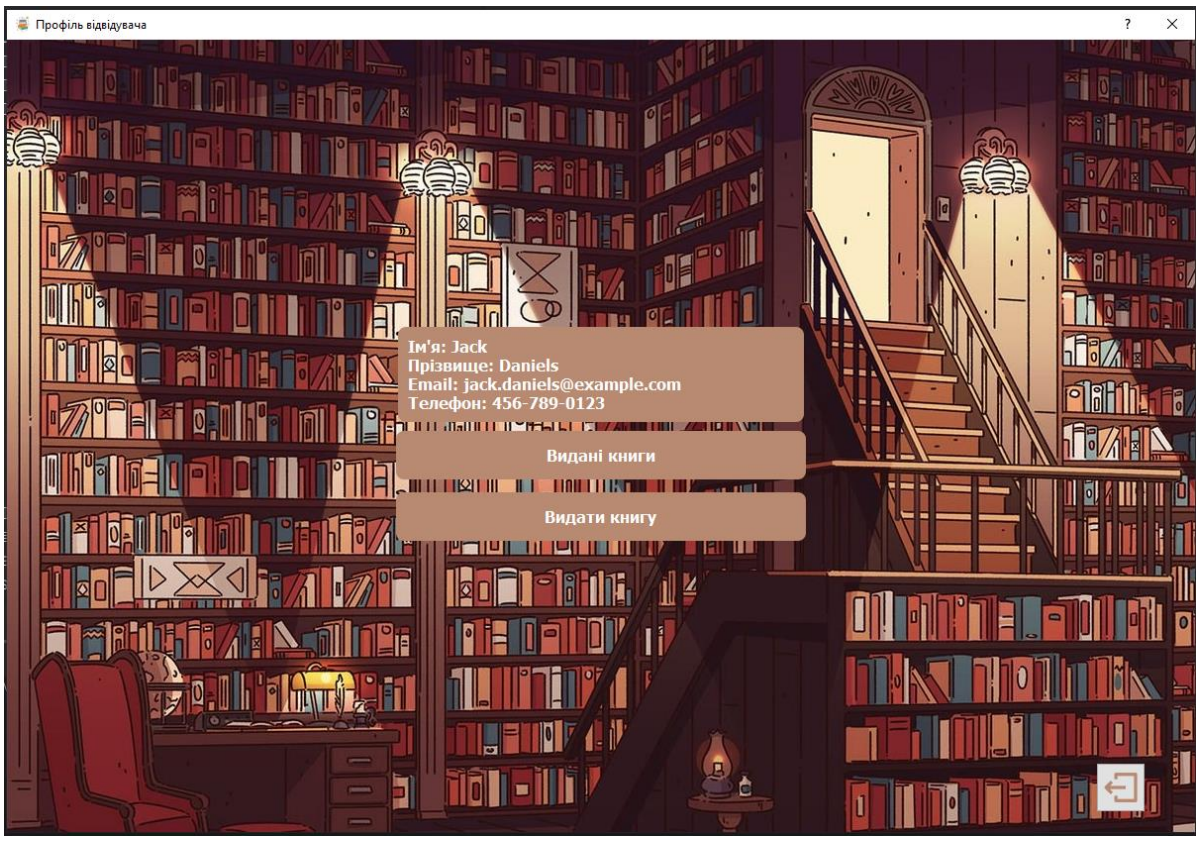


Рисунок 3.15 – Демонстрація профілю користувача

Якщо ми натискаємо на кнопку «Видати книгу» відкривається нове вікно, де із розкривного 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.

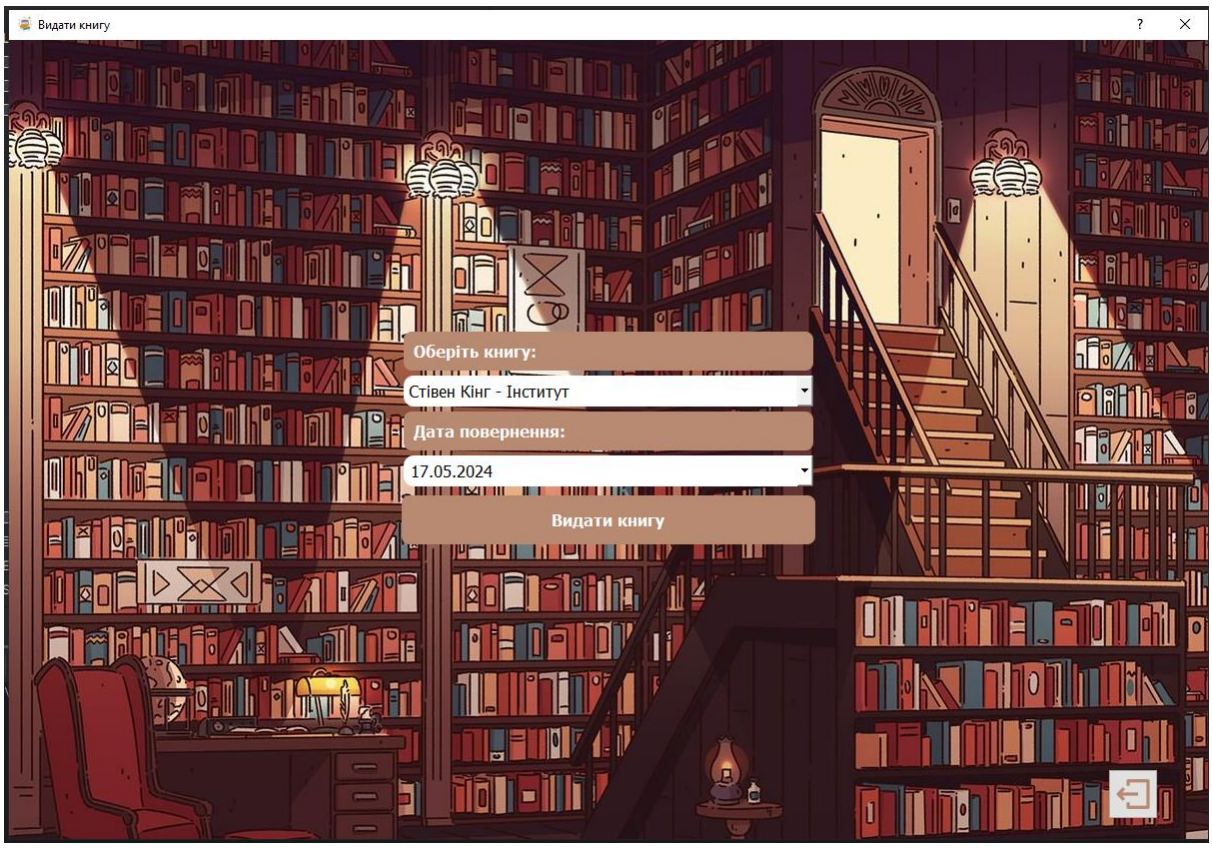


Рисунок 3.16 – Меню видачі книги певному користувачу

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.

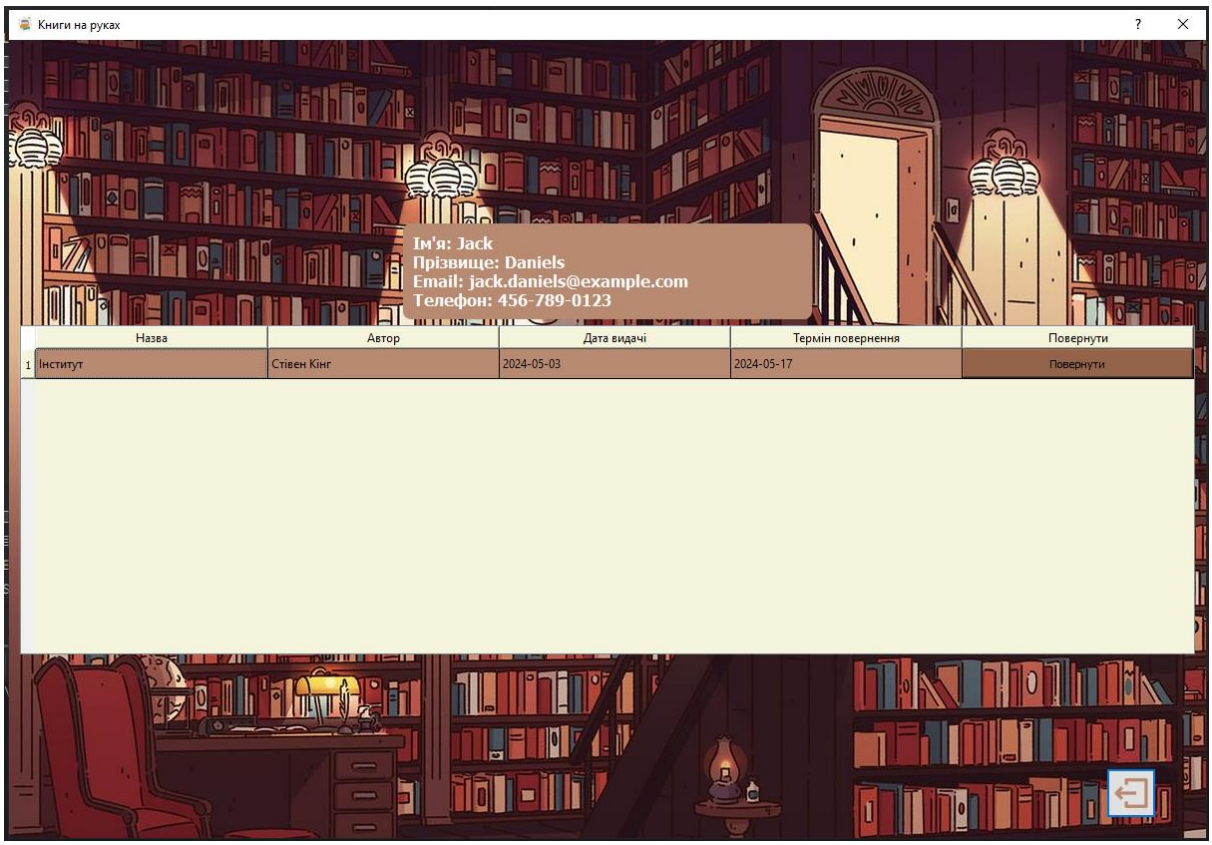


Рисунок 3.17 – Меню перегляду виданих книг певному користувачу

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. Проєкт 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 c.

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

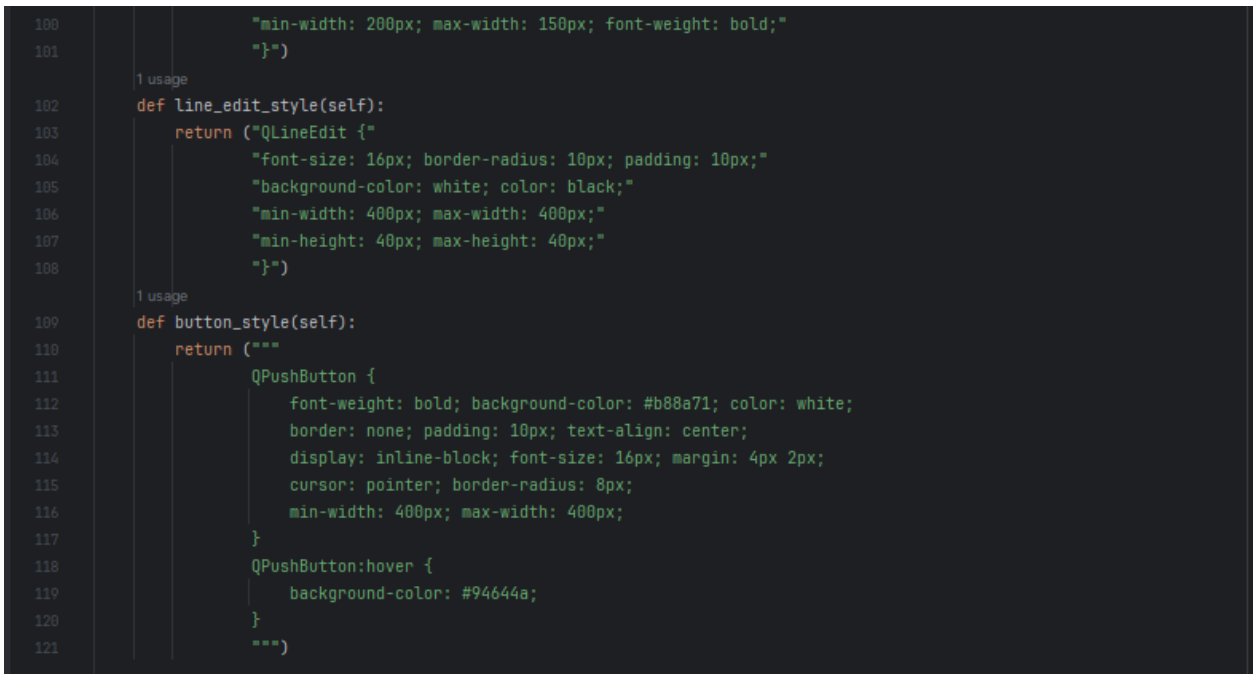
```python
import sys
import psycopg2
from PyQt5.QtCore import Qt
from PyQt5.QtGui import QIcon
from PyQt5.QtWidgets import QPushButton, QMessageBox, QMainWindow
import utils
1 usage
def create_connection():
    try:
        conn = psycopg2.connect(
            host="localhost",
            database="Library Assistant",
            user="postgres",
            password="1111")
        return conn
    except psycopg2.Error as e:
        print(f"Помилка підключення до бази даних: {e}")
        QMessageBox.critical(None, "Помилка підключення до бази даних", f"Помилка підключення до бази даних: {e}")
        sys.exit(1)

from PyQt5.QtWidgets import QVBoxLayout, QHBoxLayout, QLabel, QLineEdit, QWidget

1 usage
class AddBookDialog(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setGeometry(100, 100, 1200, 800)
        utils.set_background(self)
        self.setWindowIcon(QIcon('src/logo.png'))
        self.setWindowTitle('Додавання нової книги')
        self.conn = create_connection()
        self.initUI()
    def initUI(self):
        self.central_widget = QWidget(self)
        self.setCentralWidget(self.central_widget)
        self.layout = QVBoxLayout(self.central_widget)
        self.layout.addSpacing(150)

        self.create_form_entry('Назва:', 'title_edit')
        self.create_form_entry('Автор:', 'author_edit')
        self.create_form_entry('Рік публікації:', 'year_edit')
        self.create_form_entry('Видавництво:', 'publisher_edit')
        self.create_form_entry('Жанр:', 'genre_edit')
        self.create_form_entry('Кількість:', 'quantity_edit')
        self.create_form_entry('Місцезнаходження:', 'shelf_location_edit')

        save_button = QPushButton('Додати книгу', self)
        save_button.clicked.connect(self.add_book)
        save_button.setStyleSheet(self.button_style())
        self.layout.addWidget(save_button, alignment=Qt.AlignCenter)

        exit_btn = utils.create_exit_button(self)
        exit_btn.clicked.connect(self.close)
```

Рисунок 6.1 – 1 частина вмісту файлу add_book.py

add_book.py

```
        exit_btn = utils.create_exit_button(self)
        exit_btn.clicked.connect(self.close)
        exit_btn.move(1100, 730)
    7 usages
    def create_form_entry(self, label_text, edit_name):
        layout = QHBoxLayout()
        label = QLabel(label_text, self)
        label.setStyleSheet(self.label_style())
        setattr(self, edit_name, QLineEdit(self))
        getattr(self, edit_name).setStyleSheet(self.line_edit_style())
        layout.addWidget(label)
        layout.addWidget(getattr(self, edit_name))
        self.layout.addLayout(layout)
    1 usage
    def add_book(self):
        title = self.title_edit.text().strip()
        author = self.author_edit.text().strip()
        year = self.year_edit.text().strip()
        publisher = self.publisher_edit.text().strip()
        genre = self.genre_edit.text().strip()
        quantity = self.quantity_edit.text().strip()
        shelf_location = self.shelf_location_edit.text().strip()

        if not title or not author or not year or not publisher or not genre or not quantity or not shelf_location:
            QMessageBox.warning(self, 'Помилка', 'Будь ласка, заповніть всі поля.')
            return

        if not year.isdigit() or not quantity.isdigit() or int(quantity) < 0:
            QMessageBox.warning(self, 'Помилка',
                                'Рік публікації та кількість мають бути коректними числами. Кількість не може '
                                'бути меншою за 0.')
            return

        year = int(year)
        quantity = int(quantity)

        try:
            query = ("INSERT INTO books (title, author, year_published, publisher, genre, quantity, shelf_location) "
                     "VALUES (%s, %s, %s, %s, %s, %s, %s);")
            cur = self.conn.cursor()
            cur.execute(query, (title, author, year, publisher, genre, quantity, shelf_location))
            self.conn.commit()
            cur.close()
            QMessageBox.information(self, 'Книга додана', 'Нова книга успішно додана до каталогу.')
            self.close()
        except psycopg2.Error as e:
            QMessageBox.critical(self, "Помилка бази даних", f"Помилка при додаванні книги: {e}")
            self.conn.rollback()
    1 usage
    def label_style(self):
        return ("QLabel {"
                "background-color: #b88a71; color: white; font-size: 16px;"
                "border-radius: 8px; padding: 10px;"
```

Рисунок 6.2 – 2 частина вмісту файлу add_book.py



```
                "min-width: 200px; max-width: 150px; font-weight: bold;"
                "}")
    1 usage
    def line_edit_style(self):
        return ("QLineEdit {"
                "font-size: 16px; border-radius: 10px; padding: 10px;"
                "background-color: white; color: black;"
                "min-width: 400px; max-width: 400px;"
                "min-height: 40px; max-height: 40px;"
                "}")
    1 usage
    def button_style(self):
        return ("""
                QPushButton {
                    font-weight: bold; background-color: #b88a71; color: white;
                    border: none; padding: 10px; text-align: center;
                    display: inline-block; font-size: 16px; margin: 4px 2px;
                    cursor: pointer; border-radius: 8px;
                    min-width: 400px; max-width: 400px;
                }
                QPushButton:hover {
                    background-color: #94644a;
                }
                """)
```

Рисунок 6.3 – 3 частина вмісту файлу add_book.py

add_visitor.py

```python
import sys
import psycopg2
from PyQt5.QtCore import Qt
from PyQt5.QtGui import QIcon
from PyQt5.QtWidgets import QPushButton, QMessageBox, QMainWindow
import utils

1 usage
def create_connection():
    try:
        conn = psycopg2.connect(
            host="localhost",
            database="Library Assistant",
            user="postgres",
            password="1111")
        return conn
    except psycopg2.Error as e:
        print(f"Помилка підключення до бази даних: {e}")
        QMessageBox.critical(None, "Помилка підключення до бази даних", f"Помилка підключення до бази даних: {e}")
        sys.exit(1)

from PyQt5.QtWidgets import QVBoxLayout, QHBoxLayout, QLabel, QLineEdit, QWidget

1 usage
class AddVisitorDialog(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setGeometry(100, 100, 1200, 800)
        utils.set_background(self)
        self.setWindowIcon(QIcon('src/logo.png'))
        self.setWindowTitle('Додавання нового відвідувача')
        self.conn = create_connection()
        self.initUI()

    def initUI(self):
        self.central_widget = QWidget(self)
        self.setCentralWidget(self.central_widget)
        self.layout = QVBoxLayout(self.central_widget)
        self.layout.addSpacing(150)

        self.create_form_entry("Ім'я:", 'first_name_edit')
        self.create_form_entry('Прізвище:', 'last_name_edit')
        self.create_form_entry('Електронна пошта:', 'email_edit')
        self.create_form_entry('Номер телефону:', 'phone_number_edit')

        save_button = QPushButton('Додати відвідувача', self)
        save_button.clicked.connect(self.add_visitor)
        save_button.setStyleSheet(self.button_style())
        self.layout.addWidget(save_button, alignment=Qt.AlignCenter)

        exit_btn = utils.create_exit_button(self)
        exit_btn.clicked.connect(self.close)
        exit_btn.move(1100, 730)
```

Рисунок 6.4 – 1 частина вмісту файлу add_visitor.py

add_visitor.py

```
4 usages
def create_form_entry(self, label_text, edit_name):
    layout = QHBoxLayout()
    label = QLabel(label_text, self)
    label.setStyleSheet(self.label_style())
    setattr(self, edit_name, QLineEdit(self))
    getattr(self, edit_name).setStyleSheet(self.line_edit_style())
    layout.addWidget(label)
    layout.addWidget(getattr(self, edit_name))
    self.layout.addLayout(layout)

1 usage
def add_visitor(self):
    first_name = self.first_name_edit.text().strip()
    last_name = self.last_name_edit.text().strip()
    email = self.email_edit.text().strip()
    phone_number = self.phone_number_edit.text().strip()

    if not first_name or not last_name or not email or not phone_number:
        QMessageBox.warning(self, 'Помилка', 'Будь ласка, заповніть всі поля.')
        return

    try:
        query = "INSERT INTO visitors (first_name, last_name, email, phone_number) VALUES (%s, %s, %s, %s);"
        cur = self.conn.cursor()
        cur.execute(query, (first_name, last_name, email, phone_number))
        self.conn.commit()
        cur.close()
        QMessageBox.information(self, 'Відвідувача додано', 'Новий відвідувач бібліотеки успішно доданий.')
        self.close()
    except psycopg2.Error as e:
        QMessageBox.critical(self, "Помилка бази даних", f"Помилка при додаванні відвідувача: {e}")
        self.conn.rollback()

1 usage
def label_style(self):
    return ("QLabel {"
            "background-color: #b88a71; color: white; font-size: 16px;"
            "border-radius: 8px; padding: 10px;"
            "min-width: 200px; max-width: 150px; font-weight: bold;"
            "}")

1 usage
def line_edit_style(self):
    return ("QLineEdit {"
            "font-size: 16px; border-radius: 10px; padding: 10px;"
            "background-color: white; color: black;"
            "min-width: 400px; max-width: 400px;"
            "min-height: 40px; max-height: 40px;"
            "}")
```

Рисунок 6.5 – 2 частина вмісту файлу add_visitor.py

```
1 usage
def button_style(self):
    return ("""
            QPushButton {
                font-weight: bold; background-color: #b88a71; color: white;
                border: none; padding: 10px; text-align: center;
                display: inline-block; font-size: 16px; margin: 4px 2px;
                cursor: pointer; border-radius: 8px;
                min-width: 400px; max-width: 400px;
            }
            QPushButton:hover {
                background-color: #94644a;
            }
            """)
```

Рисунок 6.6 – 3 частина вмісту файлу add_visitor.py

auth_menu.py

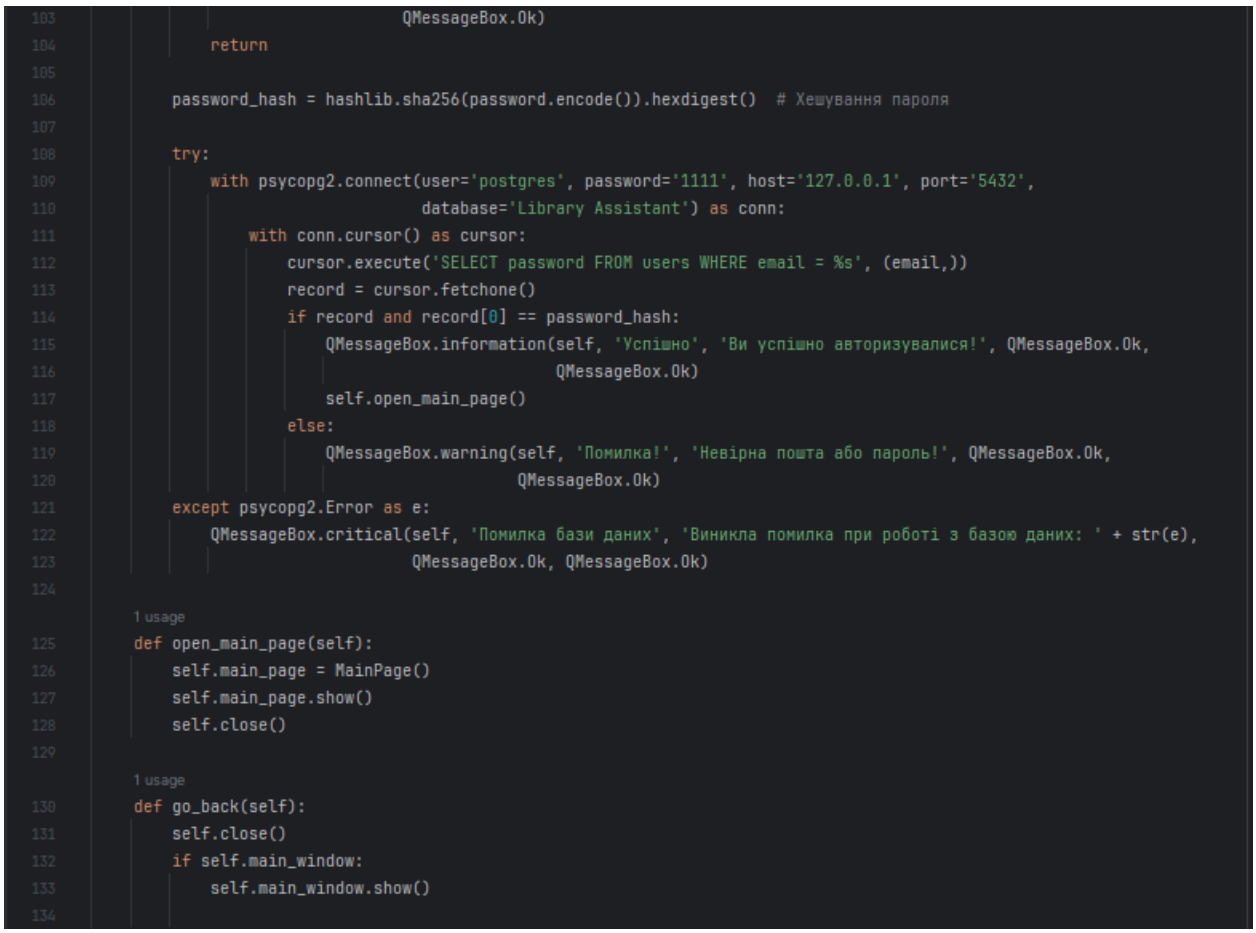
```python
import hashlib
import psycopg2
from PyQt5.QtCore import Qt
from PyQt5.QtGui import QIcon
from PyQt5.QtWidgets import QWidget, QLineEdit, QPushButton, QVBoxLayout, QMessageBox, QSpacerItem, QSizePolicy
import creator_info
from utils import set_background, create_exit_button, create_info_button, show_creator_info, create_return_button
from main_page import MainPage

3 usages
class AuthWindow(QWidget):
    def __init__(self, main_window=None):
        super().__init__()
        self.main_window = main_window
        self.initUI()

    def initUI(self):
        set_background(self)
        self.setFixedSize(1200, 800)
        self.setWindowIcon(QIcon('src/logo.png'))

        label_style = "background-color: #b88a71; color: white; font-size: 18px; border-radius: 8px; padding: 5px;"
        line_edit_style = """
                            QLineEdit {
                                font-size: 16px;
                                border-radius: 10px;
                                padding: 10px;
                                background-color: white;
                                color: black;
                                min-width: 400px;
                                max-width: 400px;
                            }
                        """

        self.email = QLineEdit(self)
        self.email.setPlaceholderText('Введіть електронну пошту')
        self.email.setStyleSheet(line_edit_style)

        self.password = QLineEdit(self)
        self.password.setPlaceholderText('Введіть пароль')
        self.password.setEchoMode(QLineEdit.Password)
        self.password.setStyleSheet(line_edit_style)

        self.loginButton = QPushButton('Авторизуватися', self)
        self.loginButton.clicked.connect(self.check_credentials)
        self.loginButton.setStyleSheet("""
                                    QPushButton {
                                        font-weight: bold;
                                        background-color: #b88a71;
                                        color: white;
                                        border: none;
                                        padding: 10px;
                                        text-align: center;
```

Рисунок 6.7 – 1 частина вмісту файлу auth_menu.py

auth_menu.py

```
                                    padding: 10px;
                                    text-align: center;
                                    text-decoration: none;
                                    display: inline-block;
                                    font-size: 16px;
                                    margin: 4px 2px;
                                    cursor: pointer;
                                    border-radius: 8px;
                                    min-width: 400px;
                                    max-width: 400px;
                                }
                                QPushButton:hover {
                                    background-color: #94644a;
                                }
                            """)

        layout = QVBoxLayout(self)
        spacer_top = QSpacerItem(20, 40, QSizePolicy.Minimum, QSizePolicy.Expanding)
        layout.addItem(spacer_top)

        layout.addWidget(self.email)
        layout.setAlignment(self.email, Qt.AlignCenter)
        layout.addWidget(self.password)
        layout.setAlignment(self.password, Qt.AlignCenter)
        layout.addWidget(self.loginButton)
        layout.setAlignment(self.loginButton, Qt.AlignCenter)

        spacer_bottom = QSpacerItem(20, 40, QSizePolicy.Minimum, QSizePolicy.Expanding)
        layout.addItem(spacer_bottom)

        exit_btn = create_exit_button(self)
        exit_btn.clicked.connect(self.close)
        exit_btn.move(1100, 730)

        info_btn = create_info_button(self)
        info_btn.clicked.connect(lambda: show_creator_info(self, self.creator_info_window))
        info_btn.move(1000, 730)

        back_btn = create_return_button(self)
        back_btn.clicked.connect(self.go_back)
        back_btn.move(50, 730)

        self.creator_info_window = creator_info.CreatorInfoWindow()
        self.setWindowTitle('Авторизація')
        self.setLayout(layout)

    1 usage
    def check_credentials(self):
        email = self.email.text().strip()
        password = self.password.text().strip()

        if not email or not password:
            QMessageBox.warning(self, 'Помилка!', 'Будь ласка, введіть email та пароль!', QMessageBox.Ok,
```

Рисунок 6.8 – 2 частина вмісту файлу auth_menu.py



```
                                QMessageBox.Ok)
            return

        password_hash = hashlib.sha256(password.encode()).hexdigest()  # Хешування пароля

        try:
            with psycopg2.connect(user='postgres', password='1111', host='127.0.0.1', port='5432',
                                  database='Library Assistant') as conn:
                with conn.cursor() as cursor:
                    cursor.execute('SELECT password FROM users WHERE email = %s', (email,))
                    record = cursor.fetchone()
                    if record and record[0] == password_hash:
                        QMessageBox.information(self, 'Успішно', 'Ви успішно авторизувалися!', QMessageBox.Ok,
                                                QMessageBox.Ok)
                        self.open_main_page()
                    else:
                        QMessageBox.warning(self, 'Помилка!', 'Невірна пошта або пароль!', QMessageBox.Ok,
                                            QMessageBox.Ok)
        except psycopg2.Error as e:
            QMessageBox.critical(self, 'Помилка бази даних', 'Виникла помилка при роботі з базою даних: ' + str(e),
                                 QMessageBox.Ok, QMessageBox.Ok)

    1 usage
    def open_main_page(self):
        self.main_page = MainPage()
        self.main_page.show()
        self.close()

    1 usage
    def go_back(self):
        self.close()
        if self.main_window:
            self.main_window.show()
```

Рисунок 6.9 – 3 частина вмісту файлу auth_menu.py

books_page.py

```python
import sys
import psycopg2
from PyQt5.QtCore import Qt
from PyQt5.QtGui import QIcon
from PyQt5.QtWidgets import (QMainWindow, QVBoxLayout, QWidget, QPushButton,
                             QTableWidget, QTableWidgetItem, QHBoxLayout, QMessageBox, QLineEdit, QLabel, QHeaderView,
                             QGridLayout)
import utils
import edit_book
import add_book, main_page
1 usage
def create_connection():
    try:
        conn = psycopg2.connect(
            host="localhost",
            database="Library Assistant",
            user="postgres",
            password="1111")
        return conn
    except psycopg2.Error as e:
        print(f"Помилка підключення до бази даних: {e}")
        QMessageBox.critical(None, "Помилка підключення до бази даних", f"Помилка підключення до бази даних: {e}")
        sys.exit(1)


1 usage
class LibraryCatalog(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setGeometry(100, 100, 900, 600)
        self.setWindowTitle('Каталог книг')
        self.setWindowIcon(QIcon('src/logo.png'))
        utils.set_background(self)
        self.conn = create_connection()
        self.current_page = 0
        self.books_per_page = 10
        self.current_filters = {}
        self.initUI()

    def initUI(self):
        self.central_widget = QWidget()
        self.setCentralWidget(self.central_widget)

        label_style = ("background-color: #b88a71; color: white; font-size: 16px; border-radius: 8px; padding: 2px; "
                       "min-width: 130px; max-width: 120px; font-weight: bold;")

        line_edit_style = """
                            QLineEdit {
                                font-size: 16px;
                                border-radius: 10px;
                                padding: 10px;
                                background-color: white;
                                color: black;
```

Рисунок 6.10 – 1 частина вмісту файлу books_page.py

books_page.py

```
                background-color: white;
                color: black;
                min-width: 200px;
                max-width: 200px;
            }
        """

    main_layout = QVBoxLayout(self.central_widget)
    self.setFixedSize(1200, 800)

    search_grid = QGridLayout()

    self.title_search = QLineEdit()
    self.title_search.setPlaceholderText("Пошук по назві")
    self.title_search.setStyleSheet(line_edit_style)
    self.author_search = QLineEdit()
    self.author_search.setPlaceholderText("Пошук по автору")
    self.author_search.setStyleSheet(line_edit_style)
    self.year_search = QLineEdit()
    self.year_search.setPlaceholderText("Пошук по року")
    self.year_search.setStyleSheet(line_edit_style)
    self.publisher_search = QLineEdit()
    self.publisher_search.setPlaceholderText("Пошук по видавництву")
    self.publisher_search.setStyleSheet(line_edit_style)
    self.genre_search = QLineEdit()
    self.genre_search.setPlaceholderText("Пошук по жанру")
    self.genre_search.setStyleSheet(line_edit_style)

    search_grid.addWidget(QLabel("Назва:", styleSheet=label_style), 0, 0)
    search_grid.addWidget(self.title_search, 0, 1)

    search_grid.addWidget(QLabel("Автор:", styleSheet=label_style), 0, 2)
    search_grid.addWidget(self.author_search, 0, 3)
    search_grid.addWidget(QLabel("Рік:", styleSheet=label_style), 0, 4)
    search_grid.addWidget(self.year_search, 0, 5)

    search_grid.addWidget(QLabel("Видавництво:", styleSheet=label_style), 1, 0)
    search_grid.addWidget(self.publisher_search, 1, 1)
    search_grid.addWidget(QLabel("Жанр:", styleSheet=label_style), 1, 2)
    search_grid.addWidget(self.genre_search, 1, 3)

    self.search_button = QPushButton("Пошук")
    self.search_button.clicked.connect(self.perform_search)
    search_grid.addWidget(self.search_button, 1, 4, 1, 2)
    self.search_button.setStyleSheet("""
                    QPushButton {
                        font-weight: bold;
                        background-color: #b88a71;
                        color: white;
                        border: none;
                        padding: 15px 32px;
                        text-align: center;
                        text-decoration: none;
```

Рисунок 6.11 – 2 частина вмісту файлу books_page.py

books_page.py

```
                            text-align: center;
                            text-decoration: none;
                            display: inline-block;
                            font-size: 16px;
                            margin: 4px 2px;
                            cursor: pointer;
                            border-radius: 8px;
                        }
                        QPushButton:hover {
                            background-color: #94644a;
                        }
                    """)

        main_layout.addLayout(search_grid)

        self.table = QTableWidget(0, 9, self)
        self.table.setHorizontalHeaderLabels(["Назва", "Автор", "Рік", "Видавництво", "Жанр", "Кількість",
                                              "Місцезнаходження", "Змінити", "Видалити"])
        main_layout.addWidget(self.table)

        self.table.horizontalHeader().setStretchLastSection(True)
        self.table.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)
        self.table.horizontalHeader().setSectionsClickable(False)
        self.table.horizontalHeader().setSectionsMovable(False)

        row_height = 30
        self.table.verticalHeader().setDefaultSectionSize(row_height)
        self.table.setFixedHeight(row_height * 11)
        self.table.setStyleSheet("""
            QTableWidget {
                background-color: #F5F5DC;
                gridline-color: #000000;
            }
            QHeaderView::section {
                background-color: #F5F5DC;
            }
            QTableWidget::item {
                background-color: #b88a71;
            }
        """)

        pagination_layout = QHBoxLayout()
        self.prev_button = QPushButton('Попередня')
        self.prev_button.clicked.connect(self.load_previous_page)
        self.prev_button.setStyleSheet("""
                            QPushButton {
                                font-weight: bold;
                                background-color: #b88a71;
                                color: white;
                                border: none;
                                padding: 15px 32px;
                                text-align: center;
                                text-decoration: none;
```

Рисунок 6.12 – 3 частина вмісту файлу books_page.py

books_page.py

```
                            text-align: center;
                            text-decoration: none;
                            display: inline-block;
                            display: inline-block;
                            font-size: 16px;
                            margin: 4px 2px;
                            cursor: pointer;
                            border-radius: 8px;
                        }
                        QPushButton:hover {
                            background-color: #94644a;
                        }
                    """)
        self.next_button = QPushButton('Наступна')
        self.next_button.clicked.connect(self.load_next_page)
        self.next_button.setStyleSheet("""
                        QPushButton {
                            font-weight: bold;
                            background-color: #b88a71;
                            color: white;
                            border: none;
                            padding: 15px 32px;
                            text-align: center;
                            text-decoration: none;
                            display: inline-block;
                            font-size: 16px;
                            margin: 4px 2px;
                            cursor: pointer;
                            border-radius: 8px;
                        }
                        QPushButton:hover {
                            background-color: #94644a;
                        }
                    """)
        pagination_layout.addWidget(self.prev_button)
        pagination_layout.addWidget(self.next_button)
        main_layout.addLayout(pagination_layout)

        add_book_layout = QHBoxLayout()
        self.add_book_button = QPushButton('Додати книгу')
        self.add_book_button.clicked.connect(self.open_add_book_dialog)
        self.add_book_button.setStyleSheet("""
            QPushButton {
                font-weight: bold;
                background-color: #b88a71;
                color: white;
                border: none;
                padding: 15px 32px;
                text-align: center;
                text-decoration: none;
                display: inline-block;
                font-size: 16px;
                margin: 4px 2px;
```

Рисунок 6.13 – 4 частина вмісту файлу books_page.py

books_page.py

```
                font-size: 16px;
                margin: 4px 2px;
                cursor: pointer;
                border-radius: 8px;
            }
            QPushButton:hover {
                background-color: #94644a;
            }
        """)
        add_book_layout.addStretch()
        add_book_layout.addWidget(self.add_book_button)
        add_book_layout.addStretch()
        self.add_book_button.setFixedSize(500, 70)
        main_layout.addLayout(add_book_layout)

        exit_btn = utils.create_exit_button(self)
        exit_btn.clicked.connect(self.close)
        exit_btn.move(1100, 730)

        back_btn = utils.create_return_button(self)
        back_btn.clicked.connect(self.go_to_main_page)
        back_btn.move(50, 730)

        self.load_data()

    1 usage
    def perform_search(self):
        self.current_filters = {
            'title': self.title_search.text().strip(),
            'author': self.author_search.text().strip(),
            'year': self.year_search.text().strip(),
            'publisher': self.publisher_search.text().strip(),
            'genre': self.genre_search.text().strip()
        }
        self.current_page = 0
        self.load_data()

    1 usage
    def open_add_book_dialog(self):
        self.add_book_window = add_book.AddBookDialog()
        self.add_book_window.show()

    1 usage
    def go_to_main_page(self):
        self.main_page = main_page.MainPage()
        self.main_page.show()
        self.close()

    5 usages
    def load_data(self):
        conditions = ["quantity > 0"]
        parameters = []
        for field, value in self.current_filters.items():
```

Рисунок 6.14 – 5 частина вмісту файлу books_page.py

books_page.py

```
        parameters = []
        for field, value in self.current_filters.items():
            if value:
                if field == 'year':
                    conditions.append(f"{field}_published = %s")
                else:
                    conditions.append(f"{field} ILIKE %s")
                parameters.append(f"%{value}%" if field != 'year' else value)

        query = "SELECT book_id, title, author, year_published, publisher, genre, quantity, shelf_location FROM books"
        if conditions:
            query += " WHERE " + " AND ".join(conditions)
        query += f" LIMIT {self.books_per_page + 1} OFFSET {self.current_page * self.books_per_page};"

        cur = self.conn.cursor()
        cur.execute(query, parameters)
        records = cur.fetchall()
        cur.close()

        has_next_page = len(records) > self.books_per_page
        records = records[:self.books_per_page] if has_next_page else records

        self.table.setRowCount(len(records))
        for row_number, row_data in enumerate(records):
            for column_number, data in enumerate(row_data[1:], 1):
                item = QTableWidgetItem(str(data))
                item.setFlags(item.flags() & ~Qt.ItemIsEditable)
                self.table.setItem(row_number, column_number - 1, item)

            edit_btn = QPushButton('Змінити')
            edit_btn.clicked.connect(lambda _, book_id=row_data[0]: self.edit_book(book_id))
            edit_btn.setStyleSheet("""
                                    QPushButton {
                                        background-color: #94644a;
                                    }
                                """)
            self.table.setCellWidget(row_number, 7, edit_btn)

            delete_btn = QPushButton('Видалити')
            delete_btn.clicked.connect(lambda _, book_id=row_data[0]: self.delete_book(book_id))
            delete_btn.setStyleSheet("""
                                    QPushButton {
                                        background-color: #94644a;
                                    }
                                """)
            self.table.setCellWidget(row_number, 8, delete_btn)

        self.prev_button.setEnabled(self.current_page > 0)
        self.next_button.setEnabled(has_next_page)

    1 usage
    def edit_book(self, book_id):
        self.edit_window = edit_book.EditBookDialog(book_id)
```

Рисунок 6.15 – 6 частина вмісту файлу books_page.py

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

Рисунок 6.16 – 7 частина вмісту файлу books_page.py

creator_info.py

```python
from PyQt5.QtWidgets import QVBoxLayout, QHBoxLayout, QWidget, QLabel
from PyQt5.QtGui import QPixmap, QIcon
from PyQt5.QtCore import Qt
from utils import set_background, create_exit_button, create_return_button

3 usages
class CreatorInfoWindow(QWidget):
    def __init__(self, main_window=None):
        super().__init__()
        self.main_window = main_window
        self.setWindowIcon(QIcon('D:/LibraryAssistant/src/logo.png'))
        self.initUI()

    def initUI(self):
        self.setWindowTitle("Інформація про розробника")
        set_background(self)
        self.setFixedSize(1200, 800)

        main_layout = QVBoxLayout(self)

        content_layout = QHBoxLayout()

        label_image = QLabel(self)
        pixmap = QPixmap('D:/LibraryAssistant/src/creator_photo.jpg')
        label_image.setPixmap(pixmap.scaled(400, 400, Qt.KeepAspectRatio, Qt.SmoothTransformation))
        content_layout.addWidget(label_image)
        label_image.setStyleSheet("border-radius: 15px;")

        label_description = QLabel('Кваліфікаційна робота на здобуття освітнього ступеня бакалавра. '
                                   'Тема: Інформаційна система для автоматизації діяльності бібліотеки. '
                                   'Виконала: студентка групи ІН-02 Заїченко Тетяна Вікторівна.')
        label_description.setWordWrap(True)
        label_description.setStyleSheet("""
                    QLabel {
                        background-color: #f8f8f8;
                        color: black;
                        font-size: 18px;
                        border-radius: 15px;
                        padding: 20px;
                    }
                """)
        content_layout.addWidget(label_description)

        main_layout.addLayout(content_layout)

        main_layout.setAlignment(Qt.AlignCenter)

        exit_btn = create_exit_button(self)
        exit_btn.clicked.connect(self.close)
        exit_btn.move(1100, 730)

        back_btn = create_return_button(self)
        back_btn.clicked.connect(self.go_back)
```

Рисунок 6.17 – 1 частина вмісту файлу creator_info.py

```python
        back_btn.move(50, 730)

        self.setLayout(main_layout)

    1 usage
    def go_back(self):
        self.close()
        if self.main_window:
            self.main_window.show()
```

Рисунок 6.18 – 2 частина вмісту файлу creator_info.py

edit_book.py

```python
import sys
import psycopg2
from PyQt5.QtCore import Qt
from PyQt5.QtGui import QIcon
from PyQt5.QtWidgets import (QMainWindow, QWidget, QVBoxLayout, QHBoxLayout, QLineEdit, QLabel, QPushButton,
                             QMessageBox, QApplication)
import utils

1 usage
def create_connection():
    try:
        conn = psycopg2.connect(
            host="localhost",
            database="Library Assistant",
            user="postgres",
            password="1111")
        return conn
    except psycopg2.Error as e:
        print(f"Error connecting to the database: {e}")
        QMessageBox.critical(None, "Database Connection Error", f"Failed to connect to database: {e}")
        sys.exit(1)

1 usage
class EditBookDialog(QMainWindow):
    def __init__(self, book_id):
        super().__init__()
        self.book_id = book_id
        self.setGeometry(100, 100, 350, 200)
        utils.set_background(self)
        self.setWindowIcon(QIcon('src/logo.png'))
        self.setWindowTitle('Редагування відомостей про книгу')
        self.conn = create_connection()
        self.initUI()

    def initUI(self):
        self.central_widget = QWidget()
        self.setCentralWidget(self.central_widget)
        self.setFixedSize(1200, 800)

        self.layout = QVBoxLayout(self.central_widget)
        self.layout.addSpacing(150)
        self.create_form_entry("Назва:", "title_edit")
        self.create_form_entry("Автор:", "author_edit")
        self.create_form_entry("Рік публікації:", "year_edit")
        self.create_form_entry("Видавництво:", "publisher_edit")
        self.create_form_entry("Жанр:", "genre_edit")
        self.create_form_entry("Кількість:", "quantity_edit")
        self.create_form_entry("Місцезнаходження:", "shelf_location_edit")

        save_button = QPushButton('Зберегти зміни', self)
        save_button.clicked.connect(self.save_changes)
        save_button.setStyleSheet(self.button_style())
        self.layout.addWidget(save_button, alignment=Qt.AlignCenter)
```

Рисунок 6.19 – 1 частина вмісту файлу edit_book.py

edit_book.py

```
        save_button.setStyleSheet(self.button_style())
        self.layout.addWidget(save_button, alignment=Qt.AlignCenter)

        exit_btn = utils.create_exit_button(self)
        exit_btn.clicked.connect(self.close)
        exit_btn.move(1100, 730)

        self.load_book_details()

    7 usages
    def create_form_entry(self, label_text, edit_name):
        layout = QHBoxLayout()
        label = QLabel(label_text, self)
        label.setStyleSheet(self.label_style())
        setattr(self, edit_name, QLineEdit(self))
        getattr(self, edit_name).setStyleSheet(self.line_edit_style())
        layout.addWidget(label)
        layout.addWidget(getattr(self, edit_name))
        self.layout.addLayout(layout)

    1 usage
    def label_style(self):
        return ("QLabel {"
                "background-color: #b88a71; color: white; font-size: 16px;"
                "border-radius: 8px; padding: 10px;"
                "min-width: 200px; max-width: 150px; font-weight: bold;"
                "}")

    1 usage
    def line_edit_style(self):
        return ("QLineEdit {"
                "font-size: 16px; border-radius: 10px; padding: 10px;"
                "background-color: white; color: black;"
                "min-width: 400px; max-width: 400px;"
                "min-height: 40px; max-height: 40px;"
                "}")

    1 usage
    def button_style(self):
        return ("""
                QPushButton {
                    font-weight: bold; background-color: #b88a71; color: white;
                    border: none; padding: 10px; text-align: center;
                    display: inline-block; font-size: 16px; margin: 4px 2px;
                    cursor: pointer; border-radius: 8px;
                    min-width: 400px; max-width: 400px;
                }
                QPushButton:hover {
                    background-color: #94644a;
                }
                """)
```

Рисунок 6.20 – 2 частина вмісту файлу edit_book.py

edit_book.py

```
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
    1 usage
114 def save_changes(self):
115     title = self.title_edit.text().strip()
116     author = self.author_edit.text().strip()
117     year_str = self.year_edit.text().strip()
118     publisher = self.publisher_edit.text().strip()
119     genre = self.genre_edit.text().strip()
120     quantity = self.quantity_edit.text().strip()
121     shelf_location = self.shelf_location_edit.text().strip()
122
123     if not title or not author or not year_str or not publisher or not genre or not quantity or not shelf_location:
124         QMessageBox.warning(self, "Помилка", "Будь ласка, заповніть всі поля.")
125         return
126
127     try:
128         year = int(year_str)
129         quantity = int(quantity)
130     except ValueError:
131         QMessageBox.warning(self, "Помилка", "Рік публікації та кількість мають бути числами.")
132         return
133
134     if year <= 0 or quantity <= 0:
135         QMessageBox.warning(self, "Помилка", "Рік публікації та кількість повинні бути більше 0.")
136         return
137
138     try:
139         query = ("UPDATE books SET title = %s, author = %s, year_published = %s, publisher = %s, genre = %s,"
140                  " quantity = %s, shelf_location = %s WHERE book_id = %s;")
141         cur = self.conn.cursor()
142         cur.execute(query, (title, author, year, publisher, genre, quantity, shelf_location,  self.book_id))
143         self.conn.commit()
144         cur.close()
145         QMessageBox.information(self, "Зміни збережено", "Інформація про книгу успішно оновлена.")
146         self.close()
147     except psycopg2.Error as e:
148         QMessageBox.critical(self, "Помилка бази даних", f"Помилка при оновленні даних: {e}")
149         self.conn.rollback()
150         cur.close()
```

Рисунок 6.21 – 3 частина вмісту файлу edit_book.py

edit_visitor.py

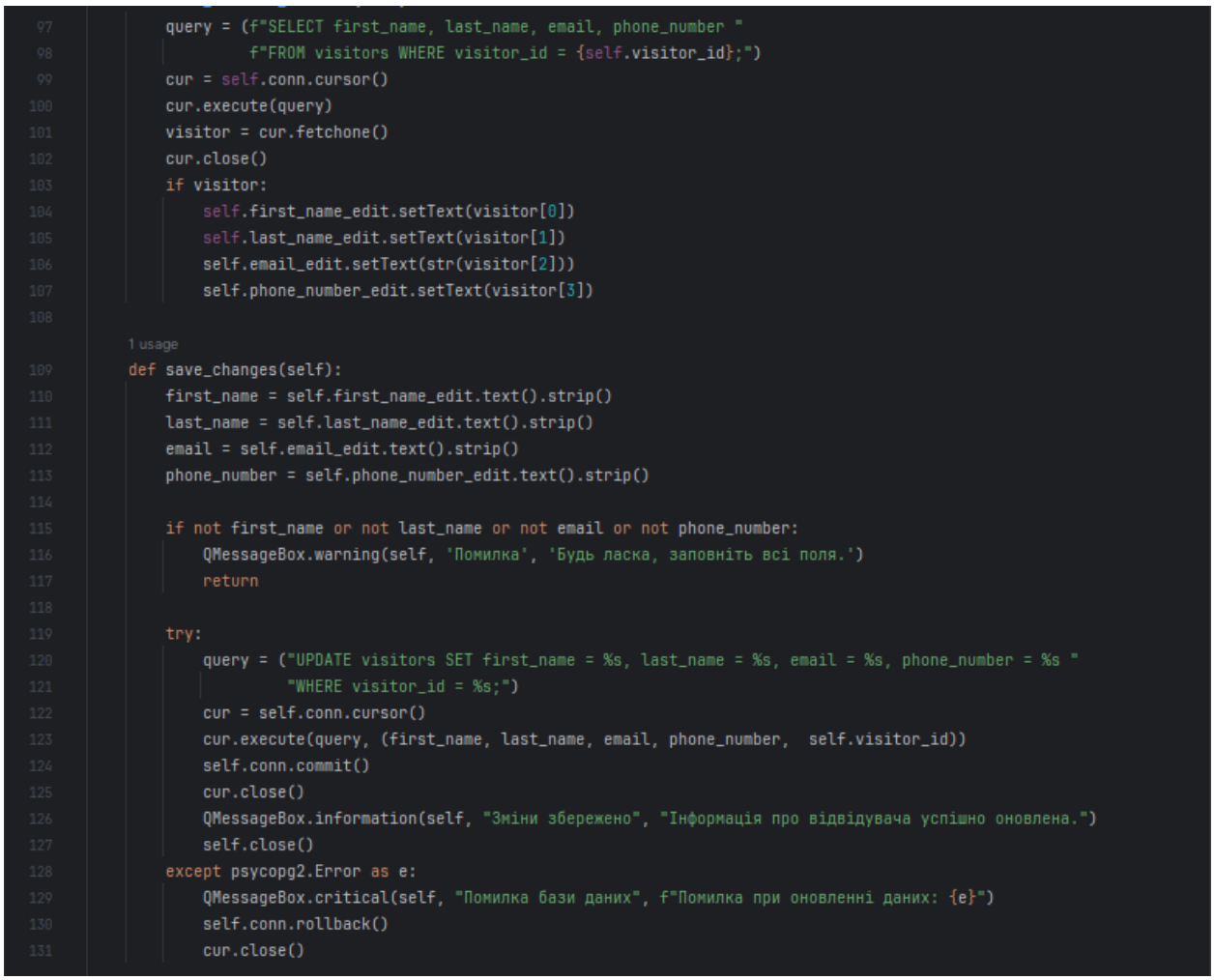
```python
import sys
import psycopg2
from PyQt5.QtCore import Qt
from PyQt5.QtGui import QIcon
from PyQt5.QtWidgets import (QMainWindow, QWidget, QVBoxLayout, QHBoxLayout, QLineEdit, QLabel, QPushButton,
                             QMessageBox, QApplication)

import utils

1 usage
def create_connection():
    try:
        conn = psycopg2.connect(
            host="localhost",
            database="Library Assistant",
            user="postgres",
            password="1111")
        return conn
    except psycopg2.Error as e:
        print(f"Error connecting to the database: {e}")
        QMessageBox.critical(None, "Database Connection Error", f"Failed to connect to database: {e}")
        sys.exit(1)

1 usage
class EditVisitorDialog(QMainWindow):
    def __init__(self, visitor_id):
        super().__init__()
        self.visitor_id = visitor_id
        self.setGeometry(100, 100, 350, 200)
        utils.set_background(self)
        self.setWindowIcon(QIcon('src/logo.png'))
        self.setWindowTitle('Редагування відомостей про відвідувача')
        self.conn = create_connection()
        self.initUI()

    def initUI(self):
        self.central_widget = QWidget()
        self.setCentralWidget(self.central_widget)
        self.setFixedSize(1200, 800)

        self.layout = QVBoxLayout(self.central_widget)
        self.layout.addSpacing(150)
        self.create_form_entry("Ім'я:", 'first_name_edit')
        self.create_form_entry('Прізвище:', 'last_name_edit')
        self.create_form_entry('Електронна пошта:', 'email_edit')
        self.create_form_entry('Номер телефону:', 'phone_number_edit')

        save_button = QPushButton('Зберегти зміни', self)
        save_button.clicked.connect(self.save_changes)
        save_button.setStyleSheet(self.button_style())
        self.layout.addWidget(save_button, alignment=Qt.AlignCenter)

        exit_btn = utils.create_exit_button(self)
```

Рисунок 6.22 – 1 частина вмісту файлу edit_visitor.py

edit_visitor.py

```
        exit_btn = utils.create_exit_button(self)
        exit_btn.clicked.connect(self.close)
        exit_btn.move(1100, 730)

        self.load_visitor_details()

    4 usages
    def create_form_entry(self, label_text, edit_name):
        layout = QHBoxLayout()
        label = QLabel(label_text, self)
        label.setStyleSheet(self.label_style())
        setattr(self, edit_name, QLineEdit(self))
        getattr(self, edit_name).setStyleSheet(self.line_edit_style())
        layout.addWidget(label)
        layout.addWidget(getattr(self, edit_name))
        self.layout.addLayout(layout)

    1 usage
    def label_style(self):
        return ("QLabel {"
                "background-color: #b88a71; color: white; font-size: 16px;"
                "border-radius: 8px; padding: 10px;"
                "min-width: 200px; max-width: 150px; font-weight: bold;"
                "}")

    1 usage
    def line_edit_style(self):
        return ("QLineEdit {"
                "font-size: 16px; border-radius: 10px; padding: 10px;"
                "background-color: white; color: black;"
                "min-width: 400px; max-width: 400px;"
                "min-height: 40px; max-height: 40px;"
                "}")

    1 usage
    def button_style(self):
        return ("""
                QPushButton {
                    font-weight: bold; background-color: #b88a71; color: white;
                    border: none; padding: 10px; text-align: center;
                    display: inline-block; font-size: 16px; margin: 4px 2px;
                    cursor: pointer; border-radius: 8px;
                    min-width: 400px; max-width: 400px;
                }
                QPushButton:hover {
                    background-color: #94644a;
                }
                """)

    1 usage
    def load_visitor_details(self):
        query = (f"SELECT first_name, last_name, email, phone_number "
```

Рисунок 6.23 – 2 частина вмісту файлу edit_visitor.py



```
        query = (f"SELECT first_name, last_name, email, phone_number "
                 f"FROM visitors WHERE visitor_id = {self.visitor_id};")
        cur = self.conn.cursor()
        cur.execute(query)
        visitor = cur.fetchone()
        cur.close()
        if visitor:
            self.first_name_edit.setText(visitor[0])
            self.last_name_edit.setText(visitor[1])
            self.email_edit.setText(str(visitor[2]))
            self.phone_number_edit.setText(visitor[3])

    1 usage
    def save_changes(self):
        first_name = self.first_name_edit.text().strip()
        last_name = self.last_name_edit.text().strip()
        email = self.email_edit.text().strip()
        phone_number = self.phone_number_edit.text().strip()

        if not first_name or not last_name or not email or not phone_number:
            QMessageBox.warning(self, 'Помилка', 'Будь ласка, заповніть всі поля.')
            return

        try:
            query = ("UPDATE visitors SET first_name = %s, last_name = %s, email = %s, phone_number = %s "
                     "WHERE visitor_id = %s;")
            cur = self.conn.cursor()
            cur.execute(query, (first_name, last_name, email, phone_number,  self.visitor_id))
            self.conn.commit()
            cur.close()
            QMessageBox.information(self, "Зміни збережено", "Інформація про відвідувача успішно оновлена.")
            self.close()
        except psycopg2.Error as e:
            QMessageBox.critical(self, "Помилка бази даних", f"Помилка при оновленні даних: {e}")
            self.conn.rollback()
            cur.close()
```

Рисунок 6.24 – 3 частина вмісту файлу edit_visitor.py

issue_book.py

```python
from PyQt5.QtCore import Qt
from PyQt5.QtGui import QIcon
from PyQt5.QtWidgets import QDialog, QVBoxLayout, QLabel, QComboBox, QDateEdit, QPushButton, QMessageBox
from datetime import date, timedelta
import utils

1 usage
class IssueBookDialog(QDialog):
    def __init__(self, visitor_id, conn):
        super().__init__()
        self.visitor_id = visitor_id
        self.conn = conn
        self.setGeometry(100, 100, 900, 600)
        self.setWindowIcon(QIcon('src/logo.png'))
        utils.set_background(self)
        self.initUI()

    def initUI(self):
        self.setWindowTitle('Видати книгу')
        layout = QVBoxLayout(self)
        self.setFixedSize(1200, 800)

        layout.setContentsMargins(10, 10, 10, 10)
        layout.setSpacing(5)

        label_style = ("background-color: #b88a71; color: white; font-size: 16px; "
                       "border-radius: 8px; padding: 10px 5px; min-width: 400px; "
                       "max-width: 400px; font-weight: bold; text-align: center;")

        combo_box_style = """
            QComboBox {
                font-size: 16px;
                border-radius: 10px;
                padding: 5px;
                min-width: 400px;
                max-width: 400px;
            }
            QComboBox QAbstractItemView {
                selection-background-color: #b88a71;
                height: 120px;
            }
        """

        date_edit_style = """
            QDateEdit {
                font-size: 16px;
                border-radius: 10px;
                padding: 5px;
                min-width: 400px;
                max-width: 400px;
                background-color: white;
            }
            QDateEdit QCalendarWidget {
```

Рисунок 6.25 – 1 частина вмісту файлу issue_book.py

issue_book.py

```
            }
            QDateEdit QCalendarWidget {
                font-size: 14px;
            }
        """

        button_style = """
                QPushButton {
                    font-weight: bold;
                    background-color: #b88a71;
                    color: white;
                    border: none;
                    padding: 15px 32px;
                    text-align: center;
                    text-decoration: none;
                    display: inline-block;
                    font-size: 16px;
                    margin: 4px 2px;
                    cursor: pointer;
                    border-radius: 8px;
                    min-width: 350px;
                    max-width: 350px;
                }
                QPushButton:hover {
                    background-color: #94644a;
                }
            """

        layout.addStretch(1)
        label = QLabel('Оберіть книгу:', styleSheet=label_style)
        self.combo_box = QComboBox()
        self.combo_box.setStyleSheet(combo_box_style)
        layout.addWidget(label, alignment=Qt.AlignCenter)
        layout.addWidget(self.combo_box, alignment=Qt.AlignCenter)

        self.load_books()

        label_date = QLabel('Дата повернення:', styleSheet=label_style)
        self.date_edit = QDateEdit()
        self.date_edit.setStyleSheet(date_edit_style)
        self.date_edit.setCalendarPopup(True)
        self.date_edit.setDate(date.today() + timedelta(days=14))
        layout.addWidget(label_date, alignment=Qt.AlignCenter)
        layout.addWidget(self.date_edit, alignment=Qt.AlignCenter)

        issue_button = QPushButton('Видати книгу')
        issue_button.setStyleSheet(button_style)
        issue_button.clicked.connect(self.issue_book)
        layout.addWidget(issue_button, alignment=Qt.AlignCenter)
        layout.addStretch(1)

        exit_btn = utils.create_exit_button(self)
        exit_btn.clicked.connect(self.close)
```

Рисунок 6.26 – 2 частина вмісту файлу issue_book.py

issue_book.py

```
        exit_btn.move(1100, 730)

    1 usage
    def load_books(self):
        cur = self.conn.cursor()
        cur.execute(
            "SELECT book_id, author, title FROM books WHERE book_id NOT IN "
            "(SELECT book_id FROM loans WHERE return_date IS NULL) AND quantity > 0"
        )
        books = cur.fetchall()
        cur.close()
        for book_id, author, title in books:
            display_text = f"{author} - {title}"
            self.combo_box.addItem(display_text, book_id)

    1 usage
    def issue_book(self):
        book_id = self.combo_box.currentData()
        due_date = self.date_edit.date().toPyDate()
        cur = self.conn.cursor()

        try:
            cur.execute("SELECT COUNT(*) FROM loans WHERE visitor_id = %s AND return_date IS NULL", (self.visitor_id,))
            books_count = cur.fetchone()[0]
            if books_count >= 10:
                QMessageBox.warning(self, 'Ліміт перевищено', 'Ви не можете взяти більше 10 книг одночасно.')
                return

            cur.execute("SELECT quantity FROM books WHERE book_id = %s", (book_id,))
            quantity = cur.fetchone()[0]
            if quantity > 0:
                cur.execute("INSERT INTO loans (book_id, visitor_id, date_out, due_date) VALUES (%s, %s, %s, %s)",
                            (book_id, self.visitor_id, date.today(), due_date))
                cur.execute("UPDATE books SET quantity = quantity - 1 WHERE book_id = %s", (book_id,))
                self.conn.commit()
                QMessageBox.information(self, 'Книга видана', 'Книга успішно видана!')
                self.close()
            else:
                QMessageBox.warning(self, 'Немає в наявності', 'На жаль, книги більше немає в наявності.')
        except Exception as e:
            self.conn.rollback()
            QMessageBox.critical(self, 'Помилка', f'Сталася помилка при виданні книги: {e}')
        finally:
            cur.close()
```

Рисунок 6.27 – 3 частина вмісту файлу issue_book.py

loans_profile.py

```python
from PyQt5.QtCore import Qt
from PyQt5.QtGui import QIcon
from PyQt5.QtWidgets import (QDialog, QVBoxLayout, QPushButton, QTableWidget, QTableWidgetItem, QMessageBox,
                             QHeaderView, QLabel)
import utils
1 usage
class LoansDialog(QDialog):
    def __init__(self, visitor_id, conn):
        super().__init__()
        self.visitor_id = visitor_id
        self.conn = conn
        self.setGeometry(100, 100, 900, 600)
        self.setWindowIcon(QIcon('src/logo.png'))
        utils.set_background(self)
        self.initUI()

    def initUI(self):
        self.setWindowTitle('Книги на руках')
        layout = QVBoxLayout(self)
        self.setFixedSize(1200, 800)

        cur = self.conn.cursor()
        cur.execute("SELECT first_name, last_name, email, phone_number FROM visitors WHERE visitor_id = %s",
                    (self.visitor_id,))
        data = cur.fetchone()
        cur.close()

        info_label_style = ("background-color: #b88a71; color: white; font-size: 16px; "
                            "border-radius: 8px; padding: 10px 5px; min-width: 400px; "
                            "max-width: 400px; font-weight: bold; text-align: center;")
        layout.addStretch(1)
        info_label = QLabel(f"Ім'я: {data[0]}\nПрізвище: {data[1]}\nEmail: {data[2]}\nТелефон: {data[3]}",
                            styleSheet=info_label_style)
        layout.addWidget(info_label, alignment=Qt.AlignCenter)

        self.table = QTableWidget(0, 5, self)
        self.table.setHorizontalHeaderLabels(['Назва', 'Автор', 'Дата видачі', 'Термін повернення', 'Повернути'])
        self.table.setStyleSheet("""
                            QTableWidget {
                                background-color: #F5F5DC;
                                gridline-color: #000000;
                            }
                            QHeaderView::section {
                                background-color: #F5F5DC;
                            }
                            QTableWidget::item {
                                background-color: #b88a71;
                            }
                        """)
        layout.addWidget(self.table)

        self.table.horizontalHeader().setStretchLastSection(True)
        self.table.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)
```

Рисунок 6.28 – 1 частина вмісту файлу loans_profile.py

loans_profile.py

```
        self.table.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)
        self.table.horizontalHeader().setSectionsClickable(False)
        self.table.horizontalHeader().setSectionsMovable(False)

        row_height = 30
        self.table.verticalHeader().setDefaultSectionSize(row_height)
        self.table.setFixedHeight(row_height * 11)

        layout.addStretch(1)

        exit_btn = utils.create_exit_button(self)
        exit_btn.clicked.connect(self.close)
        exit_btn.move(1100, 730)

        self.load_loans()

    2 usages
    def load_loans(self):
        query = """
        SELECT l.loan_id, b.title, b.author, l.date_out, l.due_date
        FROM loans l
        JOIN books b ON l.book_id = b.book_id
        WHERE l.visitor_id = %s AND l.return_date IS NULL
        """
        cur = self.conn.cursor()
        cur.execute(query, (self.visitor_id,))
        records = cur.fetchall()
        cur.close()

        self.table.setRowCount(len(records))
        for row_number, (loan_id, title, author, date_out, due_date) in enumerate(records):
            item_title = QTableWidgetItem(title)
            item_title.setFlags(item_title.flags() & ~Qt.ItemIsEditable)
            self.table.setItem(row_number, 0, item_title)

            item_author = QTableWidgetItem(str(author))
            item_author.setFlags(item_author.flags() & ~Qt.ItemIsEditable)
            self.table.setItem(row_number, 1, item_author)

            item_date_out = QTableWidgetItem(str(date_out))
            item_date_out.setFlags(item_date_out.flags() & ~Qt.ItemIsEditable)
            self.table.setItem(row_number, 2, item_date_out)

            item_due_date = QTableWidgetItem(str(due_date))
            item_due_date.setFlags(item_due_date.flags() & ~Qt.ItemIsEditable)
            self.table.setItem(row_number, 3, item_due_date)

            return_btn = QPushButton('Повернути')
            return_btn.setStyleSheet("""
                                    QPushButton {
                                        background-color: #94644a;
                                    }
                                """)
```

Рисунок 6.29 – 2 частина вмісту файлу loans_profile.py

```
                                    """)
            return_btn.clicked.connect(lambda _, ln=loan_id: self.return_book(ln))
            self.table.setCellWidget(row_number, 4, return_btn)

    1 usage
    def return_book(self, loan_id):
        cur = self.conn.cursor()
        try:
            cur.execute("UPDATE loans SET return_date = CURRENT_DATE WHERE loan_id = %s", (loan_id,))
            cur.execute(
                "UPDATE books SET quantity = quantity + 1 WHERE book_id = (SELECT book_id FROM loans WHERE "
                "loan_id = %s)",
                (loan_id,))
            self.conn.commit()
            QMessageBox.information(self, 'Книга повернута', 'Книга успішно повернута до бібліотеки!')
            self.load_loans()
        except Exception as e:
            self.conn.rollback()
            QMessageBox.critical(self, 'Помилка', f'Сталася помилка при поверненні книги: {e}')
        finally:
            cur.close()
```

Рисунок 6.30 – 3 частина вмісту файлу loans_profile.py

main.py

```
import sys
from PyQt5.QtWidgets import QApplication
import welcome_menu

app = QApplication(sys.argv)
window = welcome_menu.MainWindow()
window.show()
sys.exit(app.exec_())
```

Рисунок 6.31 – Вмісту файлу main.py

main_page.py

```python
from PyQt5.QtCore import Qt, QSize
from PyQt5.QtGui import QIcon
from PyQt5.QtWidgets import QMainWindow, QGridLayout, QWidget, QPushButton
import books_page
import users_page
import utils

4 usages
class MainPage(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setGeometry(100, 100, 300, 200)
        self.setWindowTitle('Library Assistant')
        self.setWindowIcon(QIcon('src/logo.png'))
        utils.set_background(self)
        self.initUI()

    def initUI(self):
        self.central_widget = QWidget()
        self.setCentralWidget(self.central_widget)
        self.setFixedSize(1200, 800)

        grid_layout = QGridLayout()
        grid_layout.setAlignment(Qt.AlignCenter)
        self.central_widget.setLayout(grid_layout)

        self.books_button = QPushButton('Каталог книг', self)
        self.books_button.setMinimumSize(200, 50)
        self.books_button.setIcon(QIcon('D:/LibraryAssistant/src/book.png'))
        self.books_button.setIconSize(QSize(64, 64))
        self.books_button.clicked.connect(self.goto_book_catalog)
        self.books_button.setStyleSheet("""
                            QPushButton {
                                font-weight: bold;
                                background-color: #b88a71;
                                color: white;
                                border: none;
                                padding: 15px 32px;
                                text-align: center;
                                text-decoration: none;
                                display: inline-block;
                                font-size: 16px;
                                margin: 4px 2px;
                                cursor: pointer;
                                border-radius: 8px;
                                min-width: 250px;
                                max-width: 250px;
                                min-height: 300px;
                                max-height: 300px;
                            }
                            QPushButton:hover {
                                background-color: #94644a;
                            }
```

Рисунок 6.32 – 1 частина вмісту файлу main_page.py

main_page.py

```
                                background-color: #94644a;
                            }
                        """)
        grid_layout.addWidget(self.books_button, 0, 0)

        self.visitors_button = QPushButton('Каталог користувачів', self)
        self.visitors_button.setMinimumSize(200, 50)
        self.visitors_button.setIcon(QIcon('D:/LibraryAssistant/src/user.png'))
        self.visitors_button.setIconSize(QSize(64, 64))
        self.visitors_button.clicked.connect(self.goto_visitor_catalog)
        self.visitors_button.setStyleSheet("""
                            QPushButton {
                                font-weight: bold;
                                background-color: #b88a71;
                                color: white;
                                border: none;
                                padding: 15px 32px;
                                text-align: center;
                                text-decoration: none;
                                display: inline-block;
                                font-size: 16px;
                                margin: 4px 2px;
                                cursor: pointer;
                                border-radius: 8px;
                                min-width: 250px;
                                max-width: 250px;
                                min-height: 300px;
                                max-height: 300px;
                            }
                            QPushButton:hover {
                                background-color: #94644a;
                            }
                        """)
        grid_layout.addWidget(self.visitors_button, 0, 1)

        exit_btn = utils.create_exit_button(self)
        exit_btn.clicked.connect(self.close)
        exit_btn.move(1100, 730)

    1 usage
    def goto_book_catalog(self):
        self.close()
        self.book_catalog = books_page.LibraryCatalog()
        self.book_catalog.show()

    1 usage
    def goto_visitor_catalog(self):
        self.close()
        self.users_page = users_page.UsersCatalog()
        self.users_page.show()
```

Рисунок 6.33 – 2 частина вмісту файлу main_page.py

reg_menu.py

```python
import hashlib
import re
import psycopg2
from PyQt5.QtCore import Qt
from PyQt5.QtGui import QIcon
from PyQt5.QtWidgets import QLineEdit, QWidget, QPushButton, QVBoxLayout, QMessageBox, QSpacerItem, QSizePolicy
from utils import set_background, create_exit_button, create_info_button, show_creator_info, create_return_button
import creator_info
from auth_menu import AuthWindow

1 usage
class RegisterWindow(QWidget):
    def __init__(self, main_window=None):
        super().__init__()
        self.main_window = main_window
        self.initUI()

    def initUI(self):
        set_background(self)
        self.setFixedSize(1200, 800)
        self.setWindowIcon(QIcon('src/logo.png'))

        line_edit_style = """
                    QLineEdit {
                        font-size: 16px;
                        border-radius: 10px;
                        padding: 10px;
                        background-color: white;
                        color: black;
                        min-width: 400px;
                        max-width: 400px;
                    }
                """

        self.username = QLineEdit(self)
        self.username.setPlaceholderText("Введіть ім'я користувача")
        self.username.setStyleSheet(line_edit_style)

        self.email = QLineEdit(self)
        self.email.setPlaceholderText('Введіть електронну пошту')
        self.email.setStyleSheet(line_edit_style)

        self.password = QLineEdit(self)
        self.password.setPlaceholderText('Введіть пароль')
        self.password.setEchoMode(QLineEdit.Password)
        self.password.setStyleSheet(line_edit_style)

        self.registerButton = QPushButton('Зареєструватися', self)
        self.registerButton.clicked.connect(self.register)
        self.registerButton.setStyleSheet("""
                            QPushButton {
                                font-weight: bold;
                                background-color: #b88a71;
```

Рисунок 6.34 – 1 частина вмісту файлу reg_menu.py

reg_menu.py

```
                                font-weight: bold;
                                background-color: #b88a71;
                                color: white;
                                border: none;
                                padding: 10px;
                                text-align: center;
                                text-decoration: none;
                                display: inline-block;
                                font-size: 16px;
                                margin: 4px 2px;
                                cursor: pointer;
                                border-radius: 8px;
                                min-width: 400px;
                                max-width: 400px;
                            }
                            QPushButton:hover {
                                background-color: #94644a;
                            }
                        """)

        layout = QVBoxLayout(self)
        spacer_top = QSpacerItem(20, 40, QSizePolicy.Minimum, QSizePolicy.Expanding)
        layout.addItem(spacer_top)

        layout.addWidget(self.username)
        layout.setAlignment(self.username, Qt.AlignCenter)
        layout.addWidget(self.email)
        layout.setAlignment(self.email, Qt.AlignCenter)
        layout.addWidget(self.password)
        layout.setAlignment(self.password, Qt.AlignCenter)
        layout.addWidget(self.registerButton)
        layout.setAlignment(self.registerButton, Qt.AlignCenter)

        spacer_bottom = QSpacerItem(20, 40, QSizePolicy.Minimum, QSizePolicy.Expanding)
        layout.addItem(spacer_bottom)

        exit_btn = create_exit_button(self)
        exit_btn.clicked.connect(self.close)
        exit_btn.move(1100, 730)

        info_btn = create_info_button(self)
        info_btn.clicked.connect(lambda: show_creator_info(self, self.creator_info_window))
        info_btn.move(1000, 730)

        back_btn = create_return_button(self)
        back_btn.clicked.connect(self.go_back)
        back_btn.move(50, 730)

        self.creator_info_window = creator_info.CreatorInfoWindow()
        self.setWindowTitle('Реєстрація')
        self.setLayout(layout)

```

Рисунок 6.34 – 2 частина вмісту файлу reg_menu.py

```
    1 usage
    def register(self):
        username = self.username.text().strip()
        email = self.email.text().strip()
        password = self.password.text().strip()

        if not username or not email or not password:
            QMessageBox.critical(self, 'Невірне введення', 'Всі поля повинні бути заповнені!', QMessageBox.Ok,
                                 QMessageBox.Ok)
            return

        if not re.match(r"[^@]+@[^@]+\.[^@]+", email):
            QMessageBox.critical(self, 'Невірний email', 'Будь ласка, введіть валідний email.', QMessageBox.Ok,
                                 QMessageBox.Ok)
            return

        if len(password) < 8:
            QMessageBox.critical(self, 'Короткий пароль', 'Пароль повинен містити мінімум 8 символів.', QMessageBox.Ok,
                                 QMessageBox.Ok)
            return

        password_hash = hashlib.sha256(password.encode()).hexdigest()  # Хешування пароля

        try:
            with psycopg2.connect(user='postgres', password='1111', host='127.0.0.1', port='5432',
                                  database='Library Assistant') as conn:
                with conn.cursor() as cursor:
                    cursor.execute('SELECT * FROM users WHERE username = %s OR email = %s', (username, email))
                    if cursor.fetchone():
                        QMessageBox.critical(self, 'Дублікація даних',
                                             'Користувач з таким іменем користувача або email вже існує.',
                                             QMessageBox.Ok, QMessageBox.Ok)
                        return

                    cursor.execute('INSERT INTO users (username, email, password) VALUES (%s, %s, %s)',
                                   (username, email, password_hash))
                conn.commit()
                QMessageBox.information(self, 'Успішно', 'Ви успішно зареєструвалися!', QMessageBox.Ok,
                                        QMessageBox.Ok)
                self.open_auth_window()
        except psycopg2.Error as e:
            QMessageBox.critical(self, 'Помилка бази даних', 'Виникла помилка при роботі з базою даних: ' + str(e),
                                 QMessageBox.Ok, QMessageBox.Ok)

    1 usage
    def open_auth_window(self):
        self.auth_window = AuthWindow()
        self.auth_window.show()
        self.close()

    1 usage
    def go_back(self):
        self.close()
        if self.main_window:
            self.main_window.show()
```

Рисунок 6.35 – 3 частина вмісту файлу reg_menu.py

users_page.py

```python
import sys
import psycopg2
from PyQt5.QtCore import Qt
from PyQt5.QtGui import QIcon
from PyQt5.QtWidgets import (QMainWindow, QVBoxLayout, QWidget, QPushButton, QTableWidget,
                             QTableWidgetItem, QHBoxLayout, QMessageBox, QLineEdit, QLabel,
                             QHeaderView, QGridLayout)

import utils
import edit_visitor
import add_visitor
import visitor_profile, main_page


1 usage
def create_connection():
    try:
        conn = psycopg2.connect(
            host="localhost",
            database="Library Assistant",
            user="postgres",
            password="1111")
        return conn
    except psycopg2.Error as e:
        print(f"Помилка підключення до бази даних: {e}")
        QMessageBox.critical(None, "Помилка підключення до бази даних", f"Помилка підключення до бази даних: {e}")
        sys.exit(1)


1 usage
class UsersCatalog(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setGeometry(100, 100, 900, 600)
        self.setWindowTitle('Користувачі бібліотеки')
        self.setWindowIcon(QIcon('src/logo.png'))
        self.conn = create_connection()
        self.current_page = 0
        self.records_per_page = 10
        utils.set_background(self)
        self.initUI()

    def initUI(self):
        self.central_widget = QWidget()
        self.setCentralWidget(self.central_widget)
        self.setFixedSize(1200, 800)

        label_style = ("background-color: #b88a71; color: white; font-size: 16px; border-radius: 8px; padding: 2px; "
                       "min-width: 130px; max-width: 120px; font-weight: bold;")

        line_edit_style = """
                            QLineEdit {
                                font-size: 16px;
```

Рисунок 6.36 – 1 частина вмісту файлу users_page.py

users_page.py

```
                    QLineEdit {
                        font-size: 16px;
                        border-radius: 10px;
                        padding: 10px;
                        background-color: white;
                        color: black;
                        min-width: 200px;
                        max-width: 200px;
                    }
                """

        main_layout = QVBoxLayout(self.central_widget)

        search_grid = QGridLayout()
        self.first_name_search = QLineEdit()
        self.first_name_search.setPlaceholderText("Пошук за іменем")
        self.first_name_search.setStyleSheet(line_edit_style)
        self.last_name_search = QLineEdit()
        self.last_name_search.setPlaceholderText("Пошук за прізвищем")
        self.last_name_search.setStyleSheet(line_edit_style)
        self.email_search = QLineEdit()
        self.email_search.setPlaceholderText("Пошук за емейлом")
        self.email_search.setStyleSheet(line_edit_style)

        search_grid.addWidget(QLabel("Ім'я:", styleSheet=label_style), 0, 0)
        search_grid.addWidget(self.first_name_search, 0, 1)
        search_grid.addWidget(QLabel("Прізвище:", styleSheet=label_style), 0, 2)
        search_grid.addWidget(self.last_name_search, 0, 3)
        search_grid.addWidget(QLabel("Емейл:", styleSheet=label_style), 1, 0)
        search_grid.addWidget(self.email_search, 1, 1)

        self.search_button = QPushButton("Пошук")
        self.search_button.clicked.connect(self.load_data)
        search_grid.addWidget(self.search_button, 1, 2, 1, 2)
        self.search_button.setStyleSheet("""
                    QPushButton {
                        font-weight: bold;
                        background-color: #b88a71;
                        color: white;
                        border: none;
                        padding: 15px 32px;
                        text-align: center;
                        text-decoration: none;
                        display: inline-block;
                        font-size: 16px;
                        margin: 4px 2px;
                        cursor: pointer;
                        border-radius: 8px;
                    }
                    QPushButton:hover {
                        background-color: #94644a;
                    }
                """)
```

Рисунок 6.37 – 2 частина вмісту файлу users_page.py

users_page.py

```
        main_layout.addLayout(search_grid)

        self.table = QTableWidget(0, 7, self)
        self.table.setStyleSheet("""
                    QTableWidget {
                        background-color: #F5F5DC;
                        gridline-color: #000000;
                    }
                    QHeaderView::section {
                        background-color: #F5F5DC;
                    }
                    QTableWidget::item {
                        background-color: #b88a71;
                    }
                """)
        self.table.setHorizontalHeaderLabels(["Ім'я", "Прізвище", "Емейл", "Телефон", "Змінити",
                                              "Видалити", "Переглянути профіль"])
        main_layout.addWidget(self.table)

        self.table.horizontalHeader().setStretchLastSection(True)
        self.table.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)
        self.table.horizontalHeader().setSectionsClickable(False)
        self.table.horizontalHeader().setSectionsMovable(False)

        row_height = 30
        self.table.verticalHeader().setDefaultSectionSize(row_height)
        self.table.setFixedHeight(row_height * 11)

        pagination_layout = QHBoxLayout()
        self.prev_button = QPushButton('Попередня')
        self.prev_button.clicked.connect(self.load_previous_page)
        self.prev_button.setStyleSheet("""
                                    QPushButton {
                                        font-weight: bold;
                                        background-color: #b88a71;
                                        color: white;
                                        border: none;
                                        padding: 15px 32px;
                                        text-align: center;
                                        text-decoration: none;
                                        display: inline-block;
                                        font-size: 16px;
                                        margin: 4px 2px;
                                        cursor: pointer;
                                        border-radius: 8px;
                                    }
                                    QPushButton:hover {
                                        background-color: #94644a;
                                    }
                                """)
        self.next_button = QPushButton('Наступна')
        self.next_button.clicked.connect(self.load_next_page)
        self.next_button.setStyleSheet("""
```

Рисунок 6.38 – 3 частина вмісту файлу users_page.py

users_page.py

```
        self.next_button.clicked.connect(self.load_next_page)
        self.next_button.setStyleSheet("""
                                    QPushButton {
                                        font-weight: bold;
                                        background-color: #b88a71;
                                        color: white;
                                        border: none;
                                        padding: 15px 32px;
                                        text-align: center;
                                        text-decoration: none;

                                        display: inline-block;
                                        font-size: 16px;
                                        margin: 4px 2px;
                                        cursor: pointer;
                                        border-radius: 8px;
                                    }
                                    QPushButton:hover {
                                        background-color: #94644a;
                                    }
                                """)
        pagination_layout.addWidget(self.prev_button)
        pagination_layout.addWidget(self.next_button)
        main_layout.addLayout(pagination_layout)

        add_visitor_layout = QHBoxLayout()
        self.add_visitor_button = QPushButton('Додати користувача')
        self.add_visitor_button.clicked.connect(self.open_add_visitor_dialog)
        self.add_visitor_button.setStyleSheet("""
                    QPushButton {
                        font-weight: bold;
                        background-color: #b88a71;
                        color: white;
                        border: none;
                        padding: 15px 32px;
                        text-align: center;
                        text-decoration: none;
                        display: inline-block;
                        font-size: 16px;
                        margin: 4px 2px;
                        cursor: pointer;
                        border-radius: 8px;
                    }
                    QPushButton:hover {
                        background-color: #94644a;
                    }
                """)
        add_visitor_layout.addStretch()
        add_visitor_layout.addWidget(self.add_visitor_button)
        add_visitor_layout.addStretch()
        self.add_visitor_button.setFixedSize(500, 70)
        main_layout.addLayout(add_visitor_layout)

```

Рисунок 6.39 – 4 частина вмісту файлу users_page.py

users_page.py

```
        main_layout.addLayout(add_visitor_layout)

        exit_btn = utils.create_exit_button(self)
        exit_btn.clicked.connect(self.close)
        exit_btn.move(1100, 730)

        back_btn = utils.create_return_button(self)
        back_btn.clicked.connect(self.go_to_main_page)
        back_btn.move(50, 730)

        self.load_data()

    5 usages
    def load_data(self):
        conditions = []
        parameters = []
        if self.first_name_search.text().strip():
            conditions.append("first_name ILIKE %s")
            parameters.append(f"%{self.first_name_search.text().strip()}%")
        if self.last_name_search.text().strip():
            conditions.append("last_name ILIKE %s")
            parameters.append(f"%{self.last_name_search.text().strip()}%")
        if self.email_search.text().strip():
            conditions.append("email ILIKE %s")
            parameters.append(f"%{self.email_search.text().strip()}%")

        query = "SELECT visitor_id, first_name, last_name, email, phone_number FROM visitors"
        if conditions:
            query += " WHERE " + " AND ".join(conditions)
        query += " ORDER BY visitor_id ASC LIMIT %s OFFSET %s;"

        offset = self.current_page * self.records_per_page
        parameters.extend([self.records_per_page + 1, offset])

        cur = self.conn.cursor()
        cur.execute(query, parameters)
        records = cur.fetchall()
        cur.close()

        more_pages_available = len(records) > self.records_per_page
        if more_pages_available:
            records = records[:-1]

        self.table.setRowCount(len(records))
        for row_number, row_data in enumerate(records):
            for column_number, data in enumerate(row_data[1:], 1):
                item = QTableWidgetItem(str(data))
                item.setFlags(item.flags() & ~Qt.ItemIsEditable)
                self.table.setItem(row_number, column_number - 1, item)

            edit_btn = QPushButton('Змінити')
            edit_btn.clicked.connect(lambda _, visitor_id=row_data[0]: self.edit_user(visitor_id))
            edit_btn.setStyleSheet("""
```

Рисунок 6.39 – 5 частина вмісту файлу users_page.py

users_page.py

```
                                                                    QPushButton {
                                                                        background-color: #94644a;
                                                                    }
                                                                """)
            self.table.setCellWidget(row_number, 4, edit_btn)

            delete_btn = QPushButton('Видалити')
            delete_btn.clicked.connect(lambda _, visitor_id=row_data[0]: self.delete_user(visitor_id))
            delete_btn.setStyleSheet("""
                                                                    QPushButton {
                                                                        background-color: #94644a;
                                                                    }
                                                                """)
            self.table.setCellWidget(row_number, 5, delete_btn)

            view_profile_btn = QPushButton('Переглянути профіль')
            view_profile_btn.clicked.connect(lambda _, visitor_id=row_data[0]: self.view_profile(visitor_id))
            view_profile_btn.setStyleSheet("""
                                                                    QPushButton {
                                                                        background-color: #94644a;
                                                                    }
                                                                """)
            self.table.setCellWidget(row_number, 6, view_profile_btn)
        self.update_navigation_buttons()

    1 usage
    def update_navigation_buttons(self):
        cur = self.conn.cursor()
        cur.execute("SELECT COUNT(*) FROM visitors")
        total_records = cur.fetchone()[0]
        cur.close()

        max_pages = (total_records + self.records_per_page - 1) // self.records_per_page
        self.next_button.setEnabled(self.current_page < max_pages - 1)
        self.prev_button.setEnabled(self.current_page > 0)

    1 usage
    def go_to_main_page(self):
        self.main_page = main_page.MainPage()
        self.main_page.show()
        self.close()

    1 usage
    def open_add_visitor_dialog(self):
        self.add_visitor_window = add_visitor.AddVisitorDialog()
        self.add_visitor_window.show()

    1 usage
    def load_next_page(self):
        self.current_page += 1
        self.load_data()
```

Рисунок 6.39 – 6 частина вмісту файлу users_page.py

```
    1 usage
    def load_previous_page(self):
        if self.current_page > 0:
            self.current_page -= 1
        self.load_data()

    1 usage
    def edit_user(self, visitor_id):
        self.edit_window = edit_visitor.EditVisitorDialog(visitor_id)
        self.edit_window.show()

    1 usage
    def delete_user(self, visitor_id):
        reply = QMessageBox.question(self, 'Підтвердіть видалення',
                                     'Ви впевнені, що хочете видалити даного користувача?',
                                     QMessageBox.Yes | QMessageBox.No, QMessageBox.No)
        if reply == QMessageBox.Yes:
            cur = self.conn.cursor()
            cur.execute("DELETE FROM visitors WHERE visitor_id = %s;", (visitor_id,))
            self.conn.commit()
            cur.close()
            self.load_data()

    1 usage
    def view_profile(self, visitor_id):
        self.profile_dialog = visitor_profile.VisitorProfileDialog(visitor_id, self.conn)
        self.profile_dialog.show()
```

Рисунок 6.40 – 7 частина вмісту файлу users_page.py

utils.py

```python
from PyQt5.QtWidgets import QPushButton, QLabel
from PyQt5.QtGui import QIcon
from PyQt5.QtCore import QSize


def set_background(widget):
    background_label = QLabel(widget)
    background_label.setGeometry(0, 0, 1200, 800)
    background_label.setStyleSheet("background-image: url('D:/LibraryAssistant/src/background.jpg');")


def create_exit_button(parent_widget):
    exit_btn = QPushButton(parent_widget)
    exit_btn.setIcon(QIcon('D:/LibraryAssistant/src/exit.png'))
    exit_btn.setIconSize(QSize(50, 50))
    exit_btn.setFixedSize(50, 50)
    return exit_btn


6 usages
def create_info_button(parent_widget):
    info_btn = QPushButton(parent_widget)
    info_btn.setIcon(QIcon('D:/LibraryAssistant/src/info.png'))
    info_btn.setIconSize(QSize(50, 50))
    info_btn.setFixedSize(50, 50)
    return info_btn


8 usages
def create_return_button(parent_widget):
    return_btn = QPushButton(parent_widget)
    return_btn.setIcon(QIcon('D:/LibraryAssistant/src/return_welcome.png'))
    return_btn.setIconSize(QSize(50, 50))
    return_btn.setFixedSize(50, 50)
    return return_btn


6 usages
def show_creator_info(current_window, creator_info_window):
    current_window.close()
    creator_info_window.show()


def show_welcome_window(current_window, welcome_window):
    current_window.close()
    welcome_window.show()


def open_auth_window(parent_widget):
    parent_widget.close()
    parent_widget.auth_window.show()
```

Рисунок 6.41 – Вмісту файлу utils.py

visitor_profile.py

```
from PyQt5.QtCore import Qt
from PyQt5.QtGui import QIcon
from PyQt5.QtWidgets import QPushButton, QLabel, QVBoxLayout, QDialog
import issue_book, loans_profile, utils

1 usage
class VisitorProfileDialog(QDialog):
    def __init__(self, visitor_id, conn):
        super().__init__()
        self.visitor_id = visitor_id
        self.conn = conn
        self.setGeometry(100, 100, 900, 600)
        self.setWindowIcon(QIcon('src/logo.png'))
        utils.set_background(self)
        self.initUI()

    def initUI(self):
        self.setWindowTitle('Профіль відвідувача')
        layout = QVBoxLayout(self)
        self.setFixedSize(1200, 800)

        layout.setContentsMargins(10, 10, 10, 10)
        layout.setSpacing(5)

        button_style = """
                        QPushButton {
                            font-weight: bold;
                            background-color: #b88a71;
                            color: white;
                            border: none;
                            padding: 15px 32px;
                            text-align: center;
                            text-decoration: none;
                            display: inline-block;
                            font-size: 16px;
                            margin: 4px 2px;
                            cursor: pointer;
                            border-radius: 8px;
                            min-width: 350px;
                            max-width: 350px;
                        }
                        QPushButton:hover {
                            background-color: #94644a;
                        }
                    """

        cur = self.conn.cursor()
        cur.execute("SELECT first_name, last_name, email, phone_number FROM visitors WHERE visitor_id = %s",
                    (self.visitor_id,))
        data = cur.fetchone()
        cur.close()

        info_label_style = ("background-color: #b88a71; color: white; font-size: 16px; "
```

Рисунок 6.42 – 1 частина вмісту файлу visitor_profile.py

```
                        "border-radius: 8px; padding: 10px 5px; min-width: 400px; "
                        "max-width: 400px; font-weight: bold; text-align: center;")
        layout.addStretch(1)
        info_label = QLabel(f"Ім'я: {data[0]}\nПрізвище: {data[1]}\nEmail: {data[2]}\nТелефон: {data[3]}",
                            styleSheet=info_label_style)
        layout.addWidget(info_label, alignment=Qt.AlignCenter)

        view_loans_button = QPushButton('Видані книги')
        view_loans_button.setStyleSheet(button_style)
        view_loans_button.clicked.connect(self.view_loans)
        layout.addWidget(view_loans_button, alignment=Qt.AlignCenter)

        issue_book_button = QPushButton('Видати книгу')
        issue_book_button.setStyleSheet(button_style)
        issue_book_button.clicked.connect(self.issue_book)
        layout.addWidget(issue_book_button, alignment=Qt.AlignCenter)

        layout.addStretch(1)

        exit_btn = utils.create_exit_button(self)
        exit_btn.clicked.connect(self.close)
        exit_btn.move(1100, 730)

    1 usage
    def view_loans(self):
        self.loans_dialog = loans_profile.LoansDialog(self.visitor_id, self.conn)
        self.loans_dialog.show()

    1 usage
    def issue_book(self):
        self.issue_dialog = issue_book.IssueBookDialog(self.visitor_id, self.conn)
        self.issue_dialog.show()

```

Рисунок 6.43 – 2 частина вмісту файлу visitor_profile.py

welcome_menu.py

```python
from PyQt5.QtWidgets import QWidget, QPushButton, QVBoxLayout
from PyQt5.QtGui import QIcon
from PyQt5.QtCore import Qt
from utils import set_background, create_exit_button, create_info_button, show_creator_info
import creator_info
import auth_menu
import reg_menu

1 usage
class MainWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.initUI()

    def initUI(self):
        self.setFixedSize(1200, 800)
        set_background(self)
        self.setWindowTitle("Library Assistant")
        self.setWindowIcon(QIcon('src/logo.png'))

        layout = QVBoxLayout(self)
        layout.setAlignment(Qt.AlignHCenter)

        self.auth_button = QPushButton('Авторизація', self)
        self.auth_button.setFixedSize(450, 150)
        self.auth_button.clicked.connect(self.open_auth_window)
        self.auth_button.setStyleSheet("""
                    QPushButton {
                        font-weight: bold;
                        background-color: #b88a71;
                        color: white;
                        border: none;
                        padding: 15px 32px;
                        text-align: center;
                        text-decoration: none;
                        display: inline-block;
                        font-size: 16px;
                        margin: 4px 2px;
                        cursor: pointer;
                        border-radius: 8px;
                    }
                    QPushButton:hover {
                        background-color: #94644a;
                    }
                """)
        layout.addWidget(self.auth_button)

        self.reg_button = QPushButton('Реєстрація', self)
        self.reg_button.setFixedSize(450, 150)
        self.reg_button.clicked.connect(self.open_reg_window)
        self.reg_button.setStyleSheet("""
                    QPushButton {
                        font-weight: bold;
```

Рисунок 6.44 – 1 частина вмісту файлу welcome_menu.py

```
                    background-color: #b88a71;
                    color: white;
                    border: none;
                    padding: 15px 32px;
                    text-align: center;
                    text-decoration: none;
                    display: inline-block;
                    font-size: 16px;
                    margin: 4px 2px;
                    cursor: pointer;
                    border-radius: 8px;
                }
                QPushButton:hover {
                    background-color: #94644a;
                }
            """)
        layout.addWidget(self.reg_button)

        exit_btn = create_exit_button(self)
        exit_btn.clicked.connect(self.close)
        exit_btn.move(1100, 730)

        info_btn = create_info_button(self)
        info_btn.clicked.connect(lambda: show_creator_info(self, self.creator_info_window))
        info_btn.move(1000, 730)

        self.creator_info_window = creator_info.CreatorInfoWindow(self)
        self.auth_window = auth_menu.AuthWindow(self)
        self.reg_window = reg_menu.RegisterWindow(self)
        self.setLayout(layout)

    1 usage
    def open_auth_window(self):
        self.close()
        self.auth_window.show()

    1 usage
    def open_reg_window(self):
        self.close()
        self.reg_window.show()
```

Рисунок 6.45 – 2 частина вмісту файлу welcome_menu.py