Міністерство освіти і науки України

Чернівецький національний університет імені Юрія Федьковича

\_\_\_\_\_\_\_Програмного забезпечення комп’ютерних систем\_\_\_\_\_\_\_

(повна назва кафедри, циклової комісії)

**КУРСОВИЙ ПРОЄКТ**

з Принципи конструювання програмного забезпечення

(назва дисципліни)

на тему:\_ **Система керування базою даних**

**«Довідник власника відеотеки» засобами мови С#**

Виконав: студент (ка) \_\_3\_\_\_ курсу, групи\_\_\_343\_ск\_\_\_

спеціальності\_\_\_\_\_\_\_\_\_\_\_\_\_\_121\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр спеціальності)

\_\_\_\_Інженерія програмного забезпечення\_\_\_\_\_

(назва спеціальності)

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_Євинчук Д. В. \_\_\_\_

(підпис) (прізвище, ініціали)

Керівник \_\_\_\_\_\_\_Комісарчук В.В.\_\_\_\_\_\_

(прізвище, ініціали)

|  |  |  |  |
| --- | --- | --- | --- |
| **До захисту допущено:** |  |  | **Дата захисту** «\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_р. |
| від «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_р. |  |  | **Оцінка:** |
| Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |  | за національною шкалою \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| (підпис) |  |  | (словами) |
| Нормоконтролер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |  | кількість балів \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| (підпис) |  |  | (цифра) |
|  |  |  | за шкалою ECTS \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |  | (літера) |

Чернівці, 2023

Чернівецький національний університет імені Юрія Федьковича

Кафедра Програмного забезпечення комп’ютерних систем\_\_\_

Спеціальність Інженерія програмного забезпечення

# ЗАВДАННЯ

**на курсовий проєкт студента**

\_ Євинчука Дениса Віталійовича

(прізвище, ім’я, по-батькові)

1. Тема проєкту Система управління базою даних «Довідник власника відеотеки»

2. Вихідні дані до проєкту:

- розробити можливість переглядати записи в базі даних

- розробити можливість змінювати записи в базі даних

- розробити можливість видаляти записи з бази даних

- розробити можливість фільтрувати записи в базі даних за заданою умовою

3. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити):

- розробити загальні вимоги до програми

- описати методи програми

- описати користувацький інтерфейс

- зробити висновки по розробленій програмі

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

- скріншоти роботи з програмою

- блок-схеми роботи з програмою

- відео робити з програмою

Завдання прийняв (ла) до виконання \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис студента)

Керівник проєкту \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис керівника)

# РЕФЕРАТ

В курсовому проєкті розроблено систему управління базою даних «Довідник власника відеотеки» для комп’ютерів, які працюють під керуванням операційної системи Windows.

Область застосування – програмне забезпечення для комп’ютерів, що працюють на операційній системі Windows.

Програма реалізована засобами розробки середовища Microsoft Visual Studio мовою C# із застосуванням графічної підсистеми WPF.

Ця система може бути розширена в майбутньому із додаванням нового функціоналу та зміною користувацького інтерфейсу.

Курсовий проєкт містить \_\_ с., \_ рис., \_ табл., \_ додатки, \_ джерел

*ОПЕРАЦІЙНА СИСТЕМА, БАЗА ДАНИХ, СИСТЕМА УПРАВЛІННЯ БАЗОЮ ДАНИХ, VISUAL STUDIO, WINDOWS, C#, WPF.*

# SUMMARY

The course project developed database management system «Video Library Owner’s Handbook» for computers running the Windows operating system.

Scope - software for computers running on the Windows operating system.

The program is implemented by means of the Microsoft Visual Studio development environment in C# with the use of the WPF graphic subsystem. This system can be expanded in the future with the addition of new functionality and changes to the user interface.

The course project contains \_\_ pages, \_ figures, \_ tables, \_ appendix, \_ sources

OPERATING SYSTEM, DATABASE, DATABASE MANAGEMENT SYSTEM, VISUAL STUDIO, WINDOWS, C#, WPF.

# ЗМІСТ

Зміст

[ЗАВДАННЯ 2](#_Toc147355843)

[РЕФЕРАТ 3](#_Toc147355844)

[SUMMARY 4](#_Toc147355845)

[ЗМІСТ 5](#_Toc147355846)

[ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 6](#_Toc147355847)

[1. АРХІТЕКТУРА ТА ФУНКЦІОНАЛЬНІ ПОКАЗНИКИ 6](#_Toc147355848)

[1.1 Загальні вимоги до програми 7](#_Toc147355849)

[1.1.1 Вимоги до графічного інтерфейсу користувача 7](#_Toc147355850)

[1.1.2 Вимоги до архітектури програми 7](#_Toc147355851)

[1.1.3 Вимоги до функціональності додатка 7](#_Toc147355852)

[1.2 Призначення та область застосування 8](#_Toc147355853)

[1.3 Функціональні вимоги 8](#_Toc147355854)

[2. ОПИС ПРОГРАМИ 8](#_Toc147355855)

[2.1 Структура програми 8](#_Toc147355856)

[2.1.1 Модулі програми 9](#_Toc147355857)

[2.1.2 Алгоритми роботи програми 9](#_Toc147355858)

[2.2 Опис методів програми 9](#_Toc147355859)

[2.3 Програмні засоби 11](#_Toc147355860)

[2.4 Опис користувацького інтерфейсу 11](#_Toc147355861)

[ВИСНОВКИ 14](#_Toc147355862)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 15](#_Toc147355863)

[ДОДАТКИ 16](#_Toc147355864)

[ДОДАТОК А. Скролінг (текст) програми 16](#_Toc147355865)

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

СУБД – система управління базою даних.

ОС – операційна система.

# АРХІТЕКТУРА ТА ФУНКЦІОНАЛЬНІ ПОКАЗНИКИ

## 1.1 Загальні вимоги до програми

### 1.1.1 Вимоги до графічного інтерфейсу користувача

1. Робоча мова інтерфейсу – українська
2. Використовувати терміни зрозумілі для користувача
3. Навігаційна панель, яка повинна забезпечувати перегляд, редагування, додавання, видалення даних
4. Панель для фільтрації, сортування та пошуку по ключу
5. Посібник користувача з усією потрібною інформацією для користування системою
6. Кольорова гама повинна наслідувати загальноприйняті рекомндації.

### 1.1.2 Вимоги до архітектури програми

1. Використання стандартних класів, класів нащадків, абстрактних класів та класів-інтерфейсів, які реалізують:

− подання вхідних даних користувачем на GUI;

− відображення результатів у віконному режимі;

− збереження даних у файл та зчитування даних з файлу.

2) Стійкість програми:

– програма повинна не втрачати працездатності при будь-яких діях користувача;

– інформація, що вводиться, скрізь, де це можливо, піддається логічному забезпеченню цілісності даних;

– при будь-яких діях користувача не повинні втрачатись дані або їх цілісність.

### 1.1.3 Вимоги до функціональності додатка

1) Запис даних в базу даних.

2) Зчитування файлів з бази даних.

3) Засіб для обробки даних.

* Фільтрування записів
* Сортування записів
* Видалення записів
* Створення записів
* Редагування записів

4) Вікно з інформацією про програму.

## 1.2 Призначення та область застосування

**Мета роботи:** керування базою даних «Довідник власника відеотеки» для користувачів ОС Windows.

Реалізована версія системи контролю «Довідник власника відеотеки» призначена для ділового використання працівниками відеотек для керування інформацією про відеофільми, користувачів та запии в клієнтських картах.

Область застосування – програмне забезпечення для комп’ютерів підкеруванням ОС Windows.

## 1.3 Функціональні вимоги

До програмного забезпечення висуваються такі функціональні вимоги:

1. Сторінка на якій відбуваються всі операції з керуванням інформацією про фільми у відеотеці.
2. Сторінка на якій відбуваються всі операції з керуванням інформацією про користувачів відеотеки.
3. Сторінка на якій відбуваються всі операції з керуванням інформацією про видачу та повернення відеофільмів.
4. Сторінка, яка містить довідку по інструкції з використання системи.
5. Наявність меню для навігації між сторінками керування інформацією.
6. Функціональні вимоги до фільтрації записів:

* Фільтрація записів за вказаним ключем.

1. Функціональні вимоги до сортування записів:

* Сортування записів за вказаною умовою.

1. Функціональні вимоги до пошуку записів:

* Пошук записів за вказаним текстом.

1. Функціональні вимоги до додавання записів:

* Оновлення бази даних при додаванні нових записів.
* Оновлення записів у вкладці перегляду записів в базі даних.

# ОПИС ПРОГРАМИ

## 2.1 Структура програми

### 2.1.1 Модулі програми

Робота розробленого програмного забезпечення реалізується наступними модулями:

1. VideoFilm – абстрактний клас, що представляє основні властивості фільмів
2. VideoFilmOnCassette – клас, що наслідує клас VideoFilm та додає властивості, специфічні для касетного носія.
3. VideoFilmOnDisc – клас, що наслідує клас VideoFilm та додає властивості, специфічні для дискового носія.
4. Record – клас, що представляє запис про взяття фільму користувачем.
5. UserCard – клас, що містить інформацію про користувача та його взяті фільми.
6. VideoFilmLibrary – клас, який забезпечує функціональність бібліотеки фільмів та користувачів.
7. Enums – модуль, що містить перелічення для жанрів фільмів, типів касет та дисків.
8. FileManager – клас, що забезпечує запис та зчитування інформації з файлів.
9. IStringConvertible – клас-інтерфейс, що надає набір методів для перетворення об’єктів класів, в яких його імплементовано, в зручний для запису у файли та читання людиною вигляд.
10. MainWindow – клас, в якому реалізовано головне меню та методи для пошуку і фільтрації об’єктів

### 2.1.2 Алгоритми роботи програми

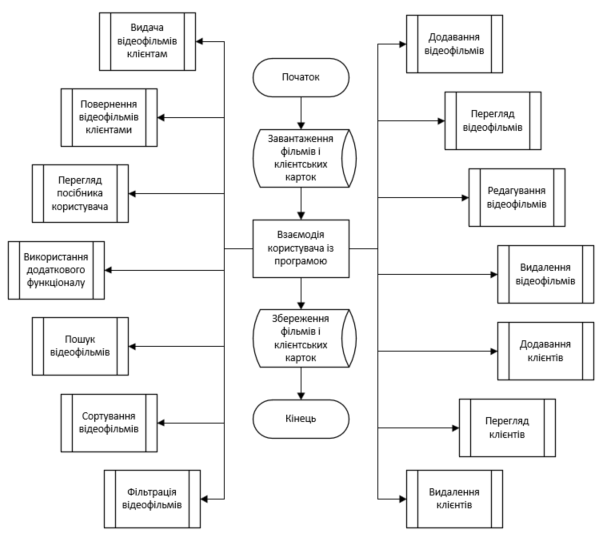
****

Рисунок 1 – Узагальнена блок-схема взаємодії користувача із програмою

## 2.2 Опис методів програми

Список методів класу MainWindow та їх опис наведено в табл.1

Таблиця 1 - Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Назва методу | Короткий опис |
| 1. | InitialSettings() | Здійснює початкові налаштування програми, завантаження даних з БД |
| 2. | MainWindow\_Closing(object sender, System.ComponentModel.CancelEventArgs e) | Здійснює збереження даних в БД при закритті програми |
| 3. | PrintMessage(string message) | Здійснює вивід текстового повідомлення на поле виводу |
| 4. | Help\_AboutProgram\_Click(object sender, RoutedEventArgs e) | Здійснює вивід загальної інформації про програму |
| 5. | Help\_LoadFromDB\_Click(object sender, RoutedEventArgs e) | Здійснює вивід інформації щодо завантаження даних із БД |
| 6. | Help\_SaveToDB\_Click(object sender, RoutedEventArgs e) | Здійснює вивід інформації щодо збереження інформації до БД |
| 7. | Help\_GiveFilm\_Click(object sender, RoutedEventArgs e | Здійснює вивід інформації щодо видачі відеофільмів |
| 8. | Help\_ReturnFilm\_Click(object sender, RoutedEventArgs e) | Здійснює вивід інформації щодо повернення відеофільмів |
| 9. | Help\_ClientBase\_Click(object sender, RoutedEventArgs e) | Здійснює вивід інформації щодо бази клієнтів |
| 10. | Help\_FilmsList\_Click(object sender, RoutedEventArgs e) | Здійснює вивід інформації щодо списку фільмів |
| 11. | Help\_AddFilm\_Click(object sender, RoutedEventArgs e) | Здійснює вивід інформації щодо додавання фільмів |
| 12. | Help\_UpdateFilm\_Click(object sender, RoutedEventArgs e) | Здійснює вивід інформації щодо редагування фільмів |
| 13. | Help\_DeleteFilm\_Click(object sender, RoutedEventArgs e) | Здійснює вивід інформації щодо видалення фільмів |
| 14. | Help\_Search\_Click(object sender, RoutedEventArgs e) | Здійснює вивід інформації щодо пошуку фільмів |
| 15. | Help\_Filter\_Click(object sender, RoutedEventArgs e) | Здійснює вивід інформації щодо фільтрації фільмів |
| 16. | Help\_Sort\_Click(object sender, RoutedEventArgs e) | Здійснює вивід інформації щодо сортування фільму |
| 17. | Help\_AdditionalFeatures\_Click(object sender, RoutedEventArgs e) | Здійснює вивід інформації щодо додаткового функціоналу |
| 18. | WriteAllFilms\_Click(object sender, RoutedEventArgs e) | Здійснює вивід інформації на екран |
| 19. | AddNewFilm\_Click(object sender, RoutedEventArgs e) | Здійснює додавання нового фільму |
| 20. | DeleteFilm\_Click(object sender, RoutedEventArgs e) | Здійснює видалення фільмів |
| 21. | UpdateFilm\_Click(object sender, RoutedEventArgs e) | Здійснює оновлення фільмів |
| 22. | Search\_Click(object sender, RoutedEventArgs e) | Здійснює пошук фільмів |
| 23. | Filter\_Click(object sender, RoutedEventArgs e) | Здійснювання фільтрації фільмів |
| 24. | Sort\_Click(object sender, RoutedEventArgs e) | Здійснення сортування фільмів |
| 25. | PrintAllClients\_Click(object sender, RoutedEventArgs e) | Здійснення виводу списку клієнтів |
| 26. | AddNewClient\_Click(object sender, RoutedEventArgs e) | Здійснення додавання нового клієнта |
| 27. | DeleteClient\_Click(object sender, RoutedEventArgs e) | Здійснення видалення клієнта |
| 28. | GiveFilm\_Click(object sender, RoutedEventArgs e) | Здійснення видачі фільмів клієнтам |
| 29. | ReturnFilm\_Click(object sender, RoutedEventArgs e) | Здійснення повернення фільмів клієнтами |
| 30. | GetUnreturnedFilmsByUserId(object sender, RoutedEventArgs e) | Здійснення перегляду списку неповернених клієнтом фільмів |
| 31. | FavoriteGenreByUserID\_Click(object sender, RoutedEventArgs e) | Додаткова функція, здійснює визначення улюбленого жанру користувача |
| 32. | RandomNotViewedFilmByUserID\_Click(object sender, RoutedEventArgs e) | Додаткова функція, здійснює вивід випадкового непереглянутого користувачем фільму |
| 33. | Top3MostWatchedFilmsInLibrary\_Click(object sender, RoutedEventArgs e) | Додаткова функція, здійснює вивід трьох найпереглядуваніших фільмів відеотеки |
| 34. | AverageFilmReturningTimeByUserID\_Click(object sender, RoutedEventArgs e) | Додаткова функція, здійснює вивід середнього часу повернення фільмів певним користувачем |
| 35. | AllWatchedFilmsByUserID(object sender, RoutedEventArgs e) | Додаткова функція, здійснює вивід інформації про фільми, переглянуті певним користувачем |
| 36. | OfferListOfSnacksByFilmID\_Click(object sender, RoutedEventArgs e) | Додаткова функція, здійснює підбір снеків під конкретний фільм |
| 37. | List<string> GetSnacksForGenre(Genres genre) | Допоміжна функція для підбору снеків |

Список методів інтерфейсу IStringConvertible та їх опис наведено в табл.2

Таблиця 2 – Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | ToStringForFile() | Перетворення об’єкту класу в форму рядка, придатного для запису до файлу |
| 2. | ToStringForWrite(); | Перетворення об’єкту класу в форму рядка, придатного для виводу не екран |

Список методів класу FileManager та їх опис наведено в табл.3

Таблиця 3 – Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | WriteFilmsToFile() | Запис фільмів в текстовий файл |
| 2. | ReadFilmsFromFile() | Зчитування фільмів з текстового файлу |
| 3. | WriteUserCardsToFile() | Запис клієнтських карток в текстовий файл |
| 4. | ReadUserCardsFromFile() | Зчитування клієнтських карток з текстового файлу |

Список методів класу UserCard та їх опис наведено в табл.4

Таблиця 4 – Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | FindMinUnusedRecordID() | Здійснює пошук незайнятого ідентифікатора запису в клієнтському квитку |
| 2. | AddRecordToUserCard(Record record) | Додає запис до списку клієнтських квитків |
| 3. | List<Record> GetRecords() | Повертає список записів клієнтського квитка |

Список методів класу UserCard та їх опис наведено в табл.5

Таблиця 5 – Основні методи класу

|  |  |  |
| --- | --- | --- |
| № | Метод | Короткий опис |
| 1. | GetVideoFilms() | Повертає список фільмів відеотеки |
| 2. | GetUserCards() | Повертає список клієнтських карток |
| 3. | AddFilm(VideoFilm videoFilm) | Додає фільм до відеотеки |
| 4. | AddUserCard(UserCard userCard) | Додає клієнтський квиток до відеотеки |
| 5. | FindMinUnusedUserID() | Знаходить незайнятий ідентифікатор користувача |
| 6. | FindMinUnusedFilmID() | Знаходить незайнятий ідентифікатор фільму |
| 7. | RemoveFilmById(int filmId) | Видаляє фільм за ідентифікатором |
| 8. | RemoveUserCardById(int userId) | Видаляє клієнтську картку за ідентифікатором |
| 9. | GetFilmById(int filmId) | Повертає фільм за ідентифікатором |
| 10. | GetUserCardById(int userId) | Повертає клієнтську картку за ідентифікатором |

## 2.3 Програмні засоби

Розробка реалізована засобами середовища Microsoft Visual Studio на мові програмування C# з використанням технологій WPF (Windows Presentation Foundation).

Microsoft Visual Studio - це інтегроване середовище розробки (IDE) від компанії Microsoft, яке використовується для створення програмного забезпечення для різних платформ.

Windows – це операційна система, орієнтована на керування за допомогою графічних інструментів. Також широкого поширення набула через її повноту взаємодії з компонентами комп’ютера. Оскільки операційна система націлена на графічний метод керування, широкого поширення набули настільні додатки, та інше додаткове ПЗ.

## 2.4 Опис користувацького інтерфейсу

При запуску програми користувач потрапляє одразу в меню посібника користувача, звідки він може перейти до будь-якого іншого підменю (рис.2).

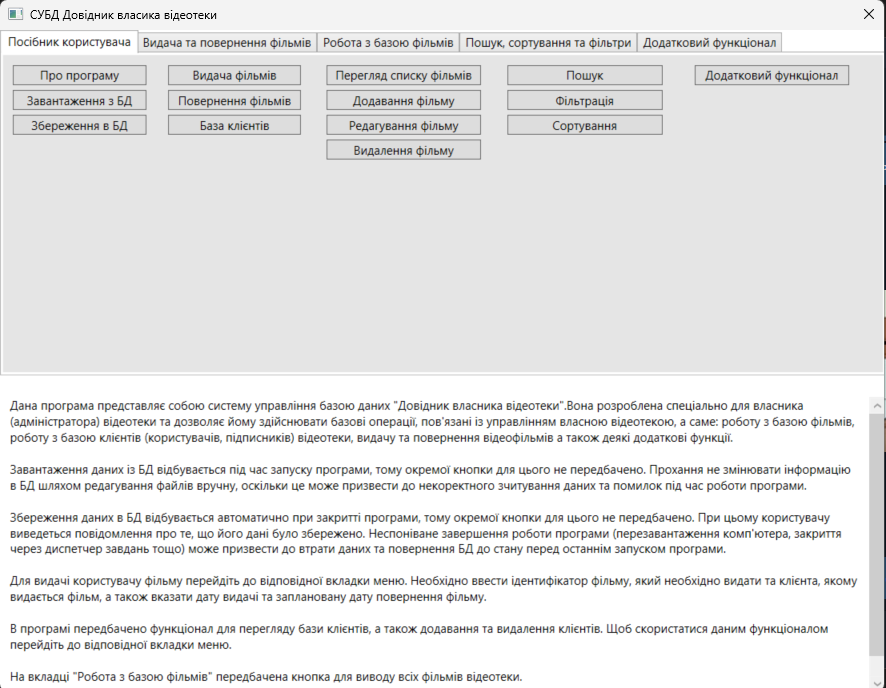


Рисунок 2 (посібник користувача)

На сторінці знаходяться кнопки для отримання довідкової інформації щодо користування програмою.

При переході до меню «Видача та повернення фільмів» користувач зможе скористатися можливостями програми з видачі та повернення фільмів, а також створення та видалення клієнтів відеотеки (рис. 3-9).

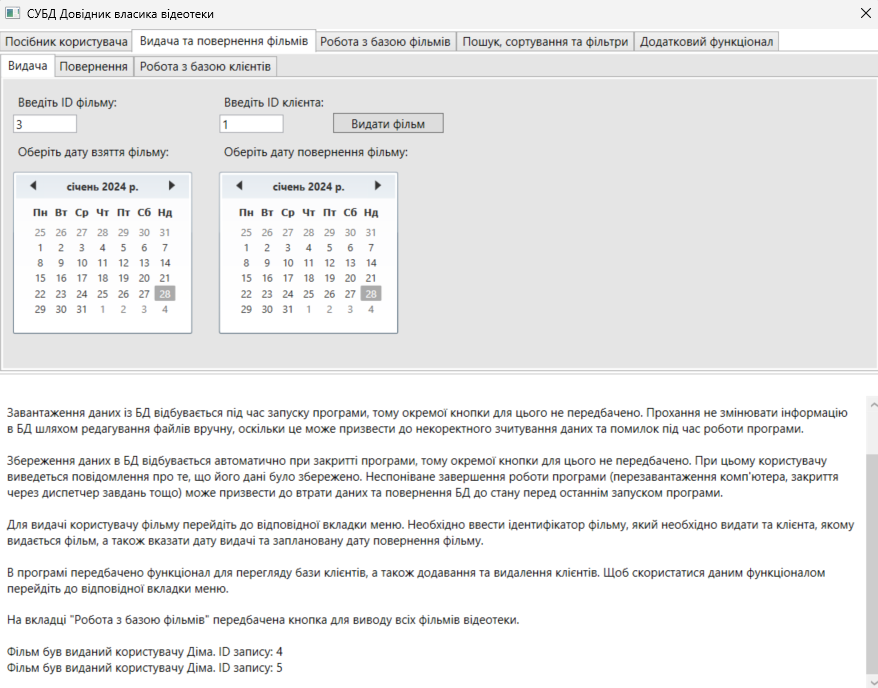
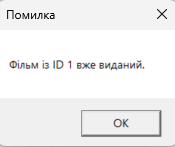
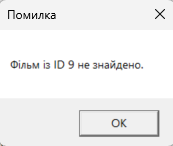
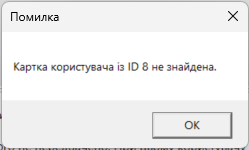


Рисунок 3 (видача фільмів)



Рисунки 4-6 (помилки під час видачі фільмів)

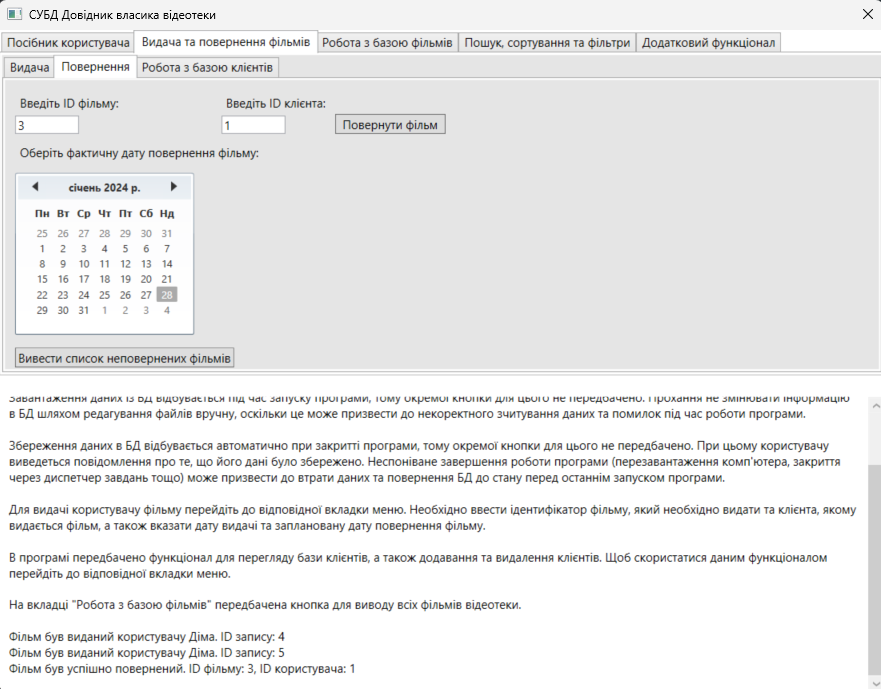


Рисунок 7 (повернення фільмів)

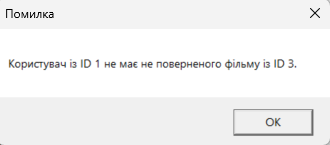


Рисунок 8 (помилка під час повернення фільмів)

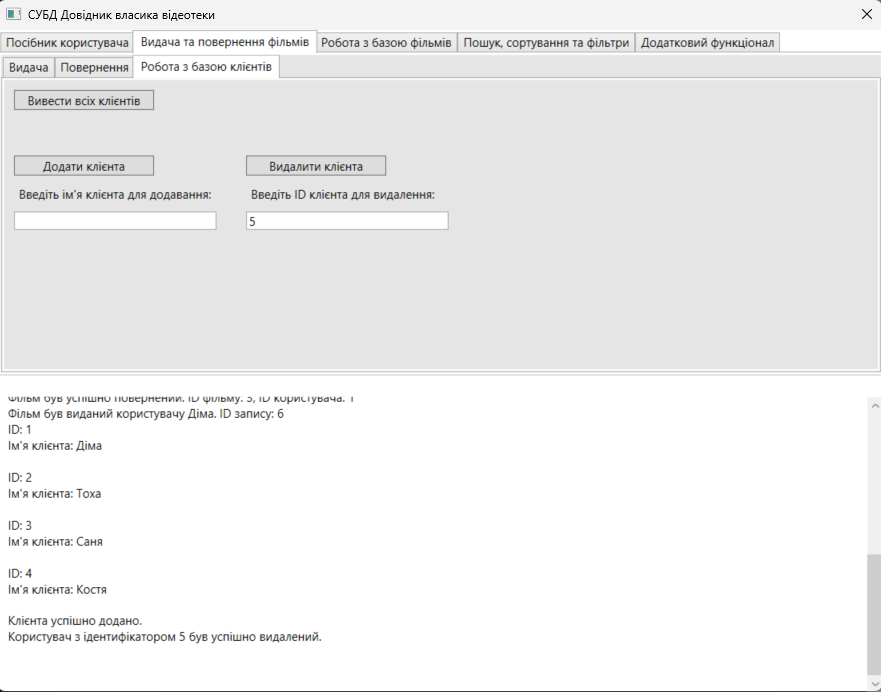
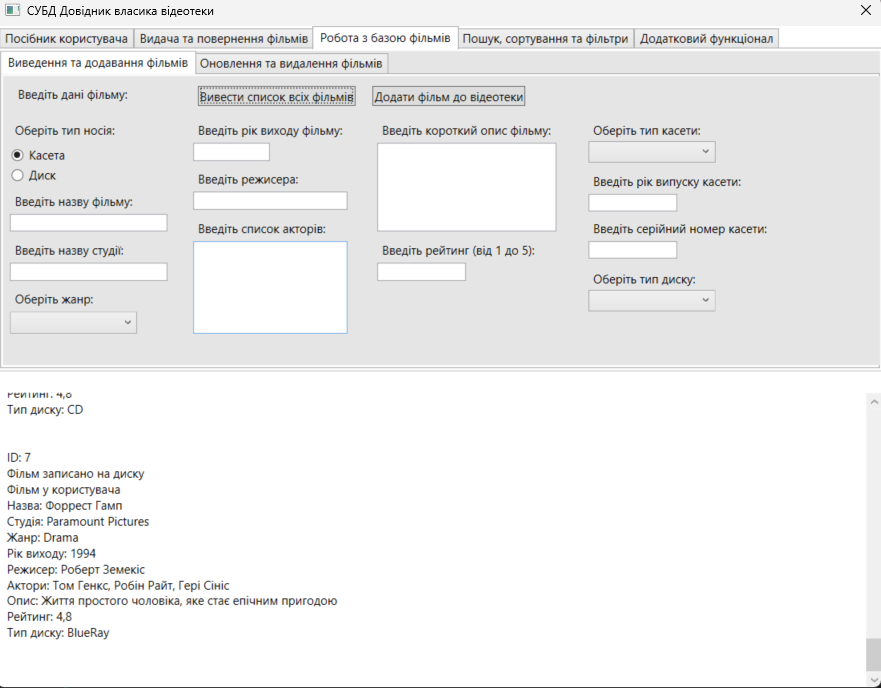
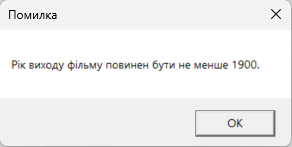
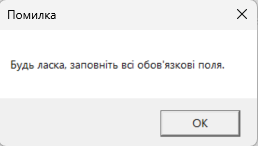


Рисунок 9 (меню додавання та видалення клієнтів)

При переході до меню «Робота з базою фільмів» користувач отримує можливість виконувати операції створення, перегляду, редагування та видалення фільмів (рис. 10-13).

Рисунок 10 (перегляд та додавання фільмів)



Рисунки 11-12 (помилки під час додавання фільмів)

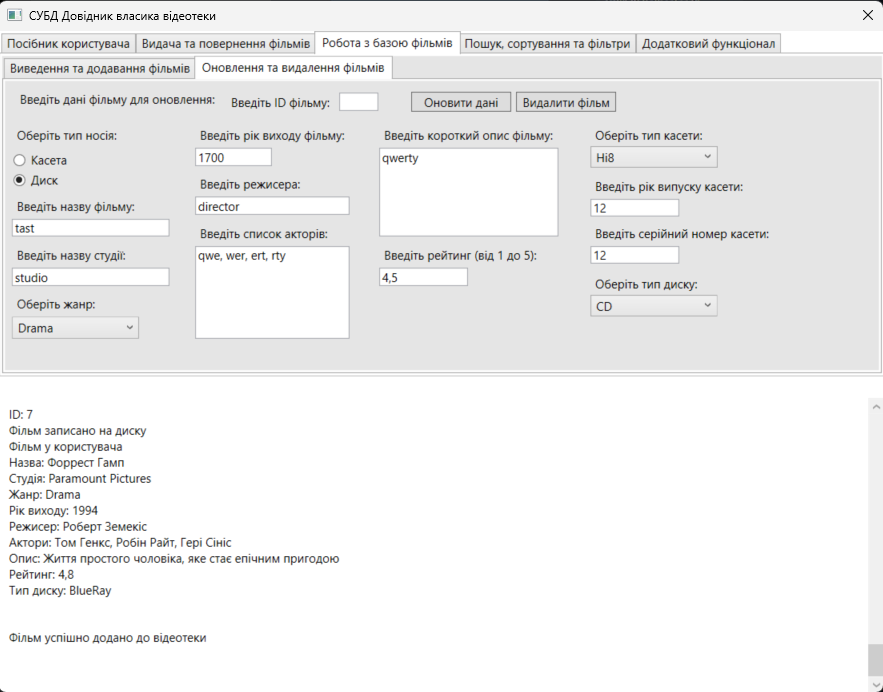


Рисунок 13 (редагування та видалення фільмів)

При переході до меню «Пошук, сортування та фільтри» користувач отримає доступ до відповідного функціоналу (рис. 14).

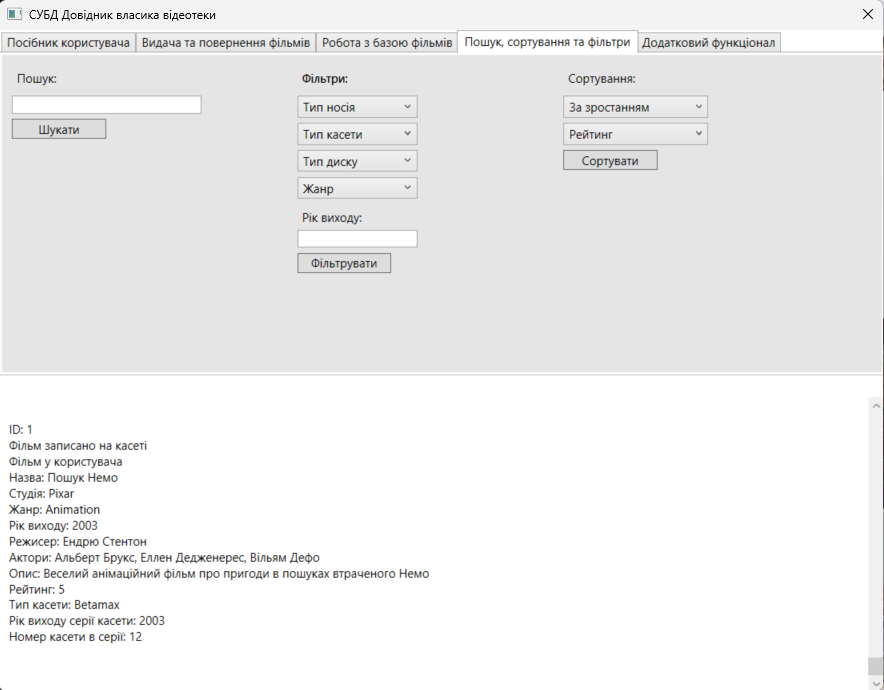


Рисунок 14 (пошук, сортування та фільтрація)

При переході до меню «додатковий функціонал» користувач отримає доступ до додаткових функцій, доступних в програмі (рис. 15).

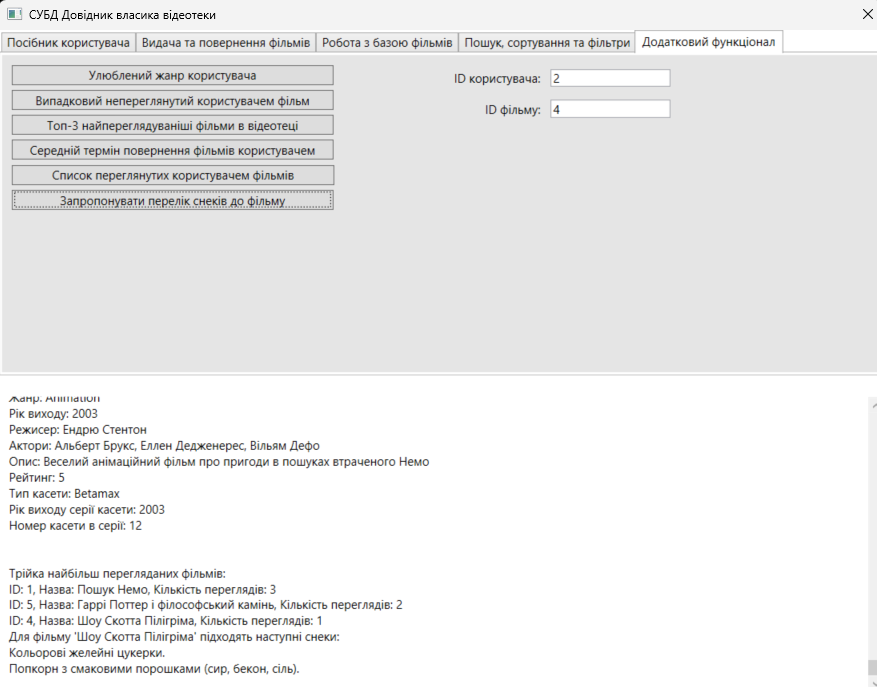


Рисунок 15 (додатковий функціонал)

# ВИСНОВКИ

В результаті виконання курсового проекту розроблена система управління

базою даних «Довідник власника відеотеки» для ОС Windows.

Розроблене програмне забезпечення має наступні можливості:

1. Можливість додавання, видалення, редагування записів.
2. Фільтрація, сортування та пошук серед записів в базі даних.
3. Видача та повернення відеофільмів користувачами.
4. Перегляд посібника користувача.

В даній роботі було застосовано технології WPF для створення графічних інтерфейсів користувача (GUI) у програмах для операційної системи Windows. Наведена розробка у майбутньому може бути розширена із додаванням нового функціоналу.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. <https://riverbankcomputing.com/software/pyqt/intro>

2. <https://forum.qt.io/topic/136650/python2-error-during-webengine-build-qt-5-12-5>

3. <https://github.com/pywinauto/pywinauto/issues/463>

# ДОДАТКИ

## ДОДАТОК А. Скролінг (текст) програми

from PySide6.QtCore import (QCoreApplication, QDate, QDateTime, QLocale,

QMetaObject, QObject, QPoint, QRect,

QSize, QTime, QUrl, Qt)

from PySide6.QtGui import (QBrush, QColor, QConicalGradient, QCursor,

QFont, QFontDatabase, QGradient, QIcon,

QImage, QKeySequence, QLinearGradient, QPainter,

QPalette, QPixmap, QRadialGradient, QTransform)

from PySide6.QtWidgets import (QApplication, QLabel, QMainWindow, QMenuBar,

QPushButton, QSizePolicy, QStatusBar, QWidget, QMessageBox, QTableWidgetItem)

from Databases.BuyerDatabase import BuyerDatabase

from Databases.DealDatabase import DealDatabase

from DesignFiles.buyerUi import Ui\_Buyer

from Models.Buyer import Buyer

class BuyerView(QWidget, Ui\_Buyer):

def \_\_init\_\_(self, parent=None):

"""Ініціалізація компонентів та методів вікна"""

super().\_\_init\_\_(parent)

self.setupUi(self)

self.retranslateUi(self)

self.buyer\_DB = BuyerDatabase()

self.deals\_DB = DealDatabase()

self.delete\_buyer.clicked.connect(self.delete\_buyer\_method)

self.addBuyer.clicked.connect(self.add\_buyer\_method)

self.edit\_buyer.clicked.connect(self.buyer\_update)

self.search\_buyer.clicked.connect(self.search\_buyers)

self.show\_all\_buyer.clicked.connect(self.fill\_seller\_table\_manage)

def fill\_seller\_table\_manage(self):

self.fill\_buyer\_table(self.buyer\_DB.read())

def buyer\_update(self):

"""Редагування даних про покупців"""

buyer\_list = []

for row in range(self.buyer\_table.rowCount()):

buyer\_list.append(

Buyer(self.buyer\_table.item(row, 0).text(), int(self.buyer\_table.item(row, 1).text()),

int(self.buyer\_table.item(row, 2).text())))

self.buyer\_DB.write(buyer\_list)

def delete\_buyer\_method(self):

"""Видалення покупців"""

new\_list = self.buyer\_DB.read()

updated\_list = []

deals\_list = self.deals\_DB.read()

updated\_deals\_list = []

for deal in deals\_list:

if self.buyer\_table.item(self.buyer\_table.currentRow(), 0).text() != deal.buyer.full\_name:

updated\_deals\_list.append(deal)

for buyer in new\_list:

if self.buyer\_table.item(self.buyer\_table.currentRow(), 0).text() != buyer.full\_name:

updated\_list.append(buyer)

self.buyer\_DB.write(updated\_list)

self.deals\_DB.write(updated\_deals\_list)

self.fill\_buyer\_table(updated\_list)

def fill\_buyer\_table(self, sellers\_list: list[Buyer]) -> None:

"""Заповнення таблиці покупців даними"""

self.buyer\_table.clearContents()

self.buyer\_table.setRowCount(len(sellers\_list))

for i in range(0, len(sellers\_list)):

self.buyer\_table.setItem(i, 0, QTableWidgetItem(str(sellers\_list[i].full\_name)))

self.buyer\_table.setItem(i, 1, QTableWidgetItem(str(sellers\_list[i].age)))

self.buyer\_table.setItem(i, 2, QTableWidgetItem(str(sellers\_list[i].balance)))

def search\_buyers(self):

"""Пошук покупців по всіх полях"""

if self.searched\_seller.text().strip() == "":

QMessageBox.warning(self, "Помилка", "Заповніть поле пошуку")

return

buyer\_result: [Buyer] = []

buyer\_list = self.buyer\_DB.read()

for buyer in buyer\_list:

if self.searched\_seller.text().strip() in [buyer.full\_name, str(buyer.age), buyer.balance]:

buyer\_result.append(buyer)

if len(buyer\_result) == 0:

QMessageBox.warning(self, "Попередження", "Немає шуканих покупців")

return

self.fill\_buyer\_table(buyer\_result)

def add\_buyer\_method(self):

"""Додавання продавця"""

try:

buyers: list[Buyer] = self.buyer\_DB.read()

if self.nameText.text().strip() == "" or self.ageNum.text().strip() == "" or self.balanceNum.text().strip() == "":

QMessageBox.warning(self, "Помилка", "Заповніть всі поля")

return

for item in buyers:

if item.full\_name == self.nameText.text().strip():

QMessageBox.warning(self, "Помилка", "Такий покупець вже існує")

return

new\_seller = Buyer(self.nameText.text(), int(self.ageNum.value()), int(self.balanceNum.text().strip()))

buyers.append(new\_seller)

self.buyer\_DB.write(buyers)

buyers = self.buyer\_DB.read()

for item in buyers:

print(item)

QMessageBox.information(self, "Успіх", "Покупця успішно додано")

except:

QMessageBox.warning(self, "Помилка", "ОЙ, ЩОСЬ ПІШЛО НЕ ТАК")

from PySide6.QtCore import (QCoreApplication, QDate, QDateTime, QLocale,

QMetaObject, QObject, QPoint, QRect,

QSize, QTime, QUrl, Qt)

from PySide6.QtGui import (QBrush, QColor, QConicalGradient, QCursor,

QFont, QFontDatabase, QGradient, QIcon,

QImage, QKeySequence, QLinearGradient, QPainter,

QPalette, QPixmap, QRadialGradient, QTransform)

from PySide6.QtWidgets import (QApplication, QLabel, QMainWindow, QMenuBar,

QPushButton, QSizePolicy, QStatusBar, QWidget, QMessageBox, QTableWidgetItem)

from Databases.BuyerDatabase import BuyerDatabase

from Databases.DealDatabase import DealDatabase

from Databases.SmartphoneDatabase import SmartphoneDatabase

from DesignFiles.dealsUi import Ui\_Deals

from Models.Buyer import Buyer

from Models.Deal import Deal

from Models.Smartphone import Smartphone

class DealsView(QWidget, Ui\_Deals):

def \_\_init\_\_(self, parent=None) -> None:

"""Ініціалізація компонентів та методів вікна"""

super().\_\_init\_\_(parent)

self.setupUi(self)

self.retranslateUi(self)

self.deals\_DB = DealDatabase()

self.buyer\_DB = BuyerDatabase()

self.smartphone\_DB = SmartphoneDatabase()

self.init\_combobox()

self.fill\_deals\_table\_manage()

self.delete\_deals.clicked.connect(self.remove\_deal)

self.buy.clicked.connect(self.add\_deal)

self.show\_all\_deals.clicked.connect(self.fill\_deals\_table\_manage)

self.search\_deals.clicked.connect(self.search\_deal)

def init\_combobox(self):

"""Формування випадних списків необхідних для створенння покупки"""

items\_list: [str] = []

smartphones: list[Smartphone] = self.smartphone\_DB.read()

for phone in smartphones:

items\_list.append(str(phone.article))

self.phones.clear()

self.phones.addItems(items\_list)

items\_list.clear()

for buyer\_item in self.buyer\_DB.read():

items\_list.append(buyer\_item.full\_name)

self.buyer.clear()

self.buyer.addItems(items\_list)

def fill\_deals\_table\_manage(self):

self.fill\_deals\_table(self.deals\_DB.read())

def remove\_deal(self):

"""Видалення угод про покупку"""

updated\_deal\_list: list[Deal] = []

for deal in self.deals\_DB.read():

if deal.toy.article != int(self.deals\_table.item(self.deals\_table.currentRow(),

0).text()) or deal.seller.full\_name != self.deals\_table.item(

self.deals\_table.currentRow(), 1).text():

updated\_deal\_list.append(deal)

self.deals\_DB.write(updated\_deal\_list)

self.fill\_deals\_table(updated\_deal\_list)

def search\_deal(self) -> None:

"""пошук угод по всіх полях"""

if self.searched\_deals.text().strip() == "":

QMessageBox.warning(self, "Помилка", "Заповніть поле пошуку")

return

deals\_result: [Deal] = []

deals\_list = self.deals\_DB.read()

for deal in deals\_list:

if (self.searched\_deals.text().strip() in str(deal.smartphone.article) or

self.searched\_deals.text().strip() in deal.buyer.full\_name):

deals\_result.append(deal)

if len(deals\_result) == 0:

QMessageBox.warning(self, "Попередження", "Немає таких угод")

self.fill\_deals\_table(deals\_list)

return

self.fill\_deals\_table(deals\_result)

def fill\_deals\_table(self, deals\_list: list[Deal]) -> None:

"""Заповнення таблиці угод даними"""

self.deals\_table.clearContents()

self.deals\_table.setRowCount(len(deals\_list))

for i in range(0, len(deals\_list)):

self.deals\_table.setItem(i, 0, QTableWidgetItem(str(deals\_list[i].smartphone.article)))

self.deals\_table.setItem(i, 1, QTableWidgetItem(str(deals\_list[i].buyer.full\_name)))

def add\_deal(self) -> None:

"""Додавання покупки"""

try:

deals: list[Deal] = self.deals\_DB.read()

if self.phones.currentText() == "" or self.buyer.currentText() == "":

QMessageBox.warning(self, "Помилка", "Заповніть всі поля")

return

smartphone: Smartphone

buyer: Buyer

for toy\_item in self.smartphone\_DB.read():

if toy\_item.article == int(self.phones.currentText()):

smartphone = toy\_item

break

for buyer\_item in self.buyer\_DB.read():

if buyer\_item.full\_name == self.buyer.currentText():

buyer = buyer\_item

break

new\_deal = Deal(smartphone, buyer)

deals.append(new\_deal)

self.deals\_DB.write(deals)

self.fill\_deals\_table\_manage()

QMessageBox.information(self, "Успіх", "Покупку успішно зроблено")

except:

QMessageBox.warning(self, "Помилка", "ОЙ, ЩОСЬ ПІШЛО НЕ ТАК")

from PySide6.QtCore import (QCoreApplication, QDate, QDateTime, QLocale,

QMetaObject, QObject, QPoint, QRect,

QSize, QTime, QUrl, Qt)

from PySide6.QtGui import (QBrush, QColor, QConicalGradient, QCursor,

QFont, QFontDatabase, QGradient, QIcon,

QImage, QKeySequence, QLinearGradient, QPainter,

QPalette, QPixmap, QRadialGradient, QTransform)

from PySide6.QtWidgets import (QApplication, QLabel, QMainWindow, QMenuBar,

QPushButton, QSizePolicy, QStatusBar, QWidget, QTableWidgetItem, QMessageBox)

from DesignFiles.phonesUi import Ui\_Form

from Models.Smartphone import Smartphone

from Databases.DealDatabase import DealDatabase

from Databases.SmartphoneDatabase import SmartphoneDatabase

class PhonesView(QWidget, Ui\_Form):

def \_\_init\_\_(self, parent=None):

"""Ініціалізація компонентів та методів вікна"""

super().\_\_init\_\_(parent)

self.setupUi(self)

self.retranslateUi(self)

self.smartphone\_DB = SmartphoneDatabase()

self.deals\_DB = DealDatabase()

self.fill\_phones\_table\_manage()

self.edit\_phone.clicked.connect(self.phones\_update)

self.delete\_phone.clicked.connect(self.delete\_phone\_method)

self.addPhone.clicked.connect(self.add\_phone)

self.show\_all\_phones.clicked.connect(self.fill\_phones\_table\_manage)

def add\_phone(self):

"""Додавання телефону"""

try:

smartphones: list[Smartphone] = self.smartphone\_DB.read()

if self.nameText.text().strip() == "" or self.articleNum.text().strip() == "" or self.priceNum.text().strip() == "" or self.brandText.text().strip() == "":

QMessageBox.warning(self, "Помилка", "Заповніть всі поля")

return

for item in smartphones:

if item.article == self.articleNum.value():

QMessageBox.warning(self, "Помилка", "Такий смартфон вже існує")

return

new\_phone = Smartphone(int(self.articleNum.value()), self.nameText.text(), int(self.priceNum.value()),

self.brandText.text())

smartphones.append(new\_phone)

self.smartphone\_DB.write(smartphones)

self.fill\_phones\_table\_manage()

QMessageBox.information(self, "Успіх", "Смартфон успішно додано")

except:

QMessageBox.warning(self, "Помилка", "ОЙ, ЩОСЬ ПІШЛО НЕ ТАК")

def fill\_phones\_table\_manage(self):

self.fill\_phones\_table(self.smartphone\_DB.read())

def phones\_update(self):

"""Редагування даних про смартфони"""

phones\_list = []

for row in range(self.phones\_table.rowCount()):

phones\_list.append(

Smartphone(int(self.phones\_table.item(row, 0).text()), self.phones\_table.item(row, 1).text(),

int(self.phones\_table.item(row, 2).text()),

self.phones\_table.item(row, 3).text()))

print(phones\_list)

self.smartphone\_DB.write(phones\_list)

def fill\_phones\_table(self, phones\_list: list[Smartphone]) -> None:

"""Заповнення таблиці телефонів даними"""

self.phones\_table.clearContents()

self.phones\_table.setRowCount(len(phones\_list))

for i in range(0, len(phones\_list)):

self.phones\_table.setItem(i, 0, QTableWidgetItem(str(phones\_list[i].article)))

self.phones\_table.setItem(i, 1, QTableWidgetItem(str(phones\_list[i].name)))

self.phones\_table.setItem(i, 2, QTableWidgetItem(str(phones\_list[i].brand)))

self.phones\_table.setItem(i, 3, QTableWidgetItem(str(phones\_list[i].price)))

def search\_phones\_method(self):

"""пошук телефонів по всіх полях"""

if self.searched\_phone.text().strip() == "":

QMessageBox.warning(self, "Помилка", "Заповніть поле пошуку")

return

phones\_result: [Smartphone] = []

phones\_list = self.smartphone\_DB.read()

for phone in phones\_list:

if self.searched\_phone.text().strip() in [phone.name, str(phone.article), str(phone.price), phone.brand]:

phones\_result.append(phone)

if len(phones\_result) == 0:

QMessageBox.warning(self, "Попередження", "Немає шуканих смартфонів")

self.fill\_phones\_table(phones\_list)

return

self.fill\_phones\_table(phones\_result)

def delete\_phone\_method(self):

"""видалення телефонів"""

phones\_list = self.smartphone\_DB.read()

updated\_phone\_list = []

updated\_deals\_list = []

for deal in self.deals\_DB.read():

if deal.smartphone.article != int(self.phones\_table.item(self.phones\_table.currentRow(), 0).text()):

updated\_deals\_list.append(deal)

for phone in phones\_list:

if int(self.phones\_table.item(self.phones\_table.currentRow(), 0).text()) != phone.article:

updated\_phone\_list.append(phone)

self.smartphone\_DB.write(updated\_phone\_list)

self.deals\_DB.write(updated\_deals\_list)

self.fill\_phones\_table(updated\_phone\_list)

from Databases.BasicDatabase import BasicDatabase

from Models.Smartphone import Smartphone

class SmartphoneDatabase(BasicDatabase):

def read(self):

result\_list: [Smartphone] = []

reading\_result = ""

with open("TxtFiles/smartphones.txt", "r") as openfile:

reading\_result += openfile.read()

reading\_result = reading\_result.split("|")

for i in range(0, len(reading\_result)):

if i != 0:

temp = reading\_result[i].split("@")

result\_list.append(Smartphone(int(temp[0]), temp[1], int(temp[3]), temp[2]))

return result\_list

def write(self, items: list) -> None:

result\_string = ""

for item in items:

result\_string += str(item)

with open("TxtFiles/smartphones.txt", "w") as openfile:

openfile.write(result\_string)