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 62 с., 15 рис., 5 табл., 1 додатки, 6 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 62 pages, 15 figures, 5 tables, 1 appendix, 6 sources

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

# ЗМІСТ

Зміст

[ЗАВДАННЯ 2](#_Toc157342665)

[РЕФЕРАТ 3](#_Toc157342666)

[SUMMARY 4](#_Toc157342667)

[ЗМІСТ 5](#_Toc157342668)

[ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 6](#_Toc157342669)

[1. АРХІТЕКТУРА ТА ФУНКЦІОНАЛЬНІ ПОКАЗНИКИ 7](#_Toc157342670)

[1.1 Загальні вимоги до програми 7](#_Toc157342671)

[1.1.1 Вимоги до графічного інтерфейсу користувача 7](#_Toc157342672)

[1.1.2 Вимоги до архітектури програми 7](#_Toc157342673)

[1.1.3 Вимоги до функціональності додатка 8](#_Toc157342674)

[1.2 Призначення та область застосування 8](#_Toc157342675)

[1.3 Функціональні вимоги 9](#_Toc157342676)

[2. ОПИС ПРОГРАМИ 10](#_Toc157342677)

[2.1 Структура програми 10](#_Toc157342678)

[2.1.1 Модулі програми 10](#_Toc157342679)

[2.1.2 Алгоритми роботи програми 11](#_Toc157342680)

[2.2 Опис методів програми 12](#_Toc157342681)

[2.3 Програмні засоби 19](#_Toc157342682)

[2.4 Опис користувацького інтерфейсу 20](#_Toc157342683)

[ВИСНОВКИ 28](#_Toc157342684)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 29](#_Toc157342685)

[ДОДАТКИ 30](#_Toc157342686)

[ДОДАТОК А. Скролінг (текст) програми 30](#_Toc157342687)

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

СУБД – система управління базою даних.

ОС – операційна система.

БД – база даних.

VS – Visual Studio.

WPF – Windows Presentation Foundation.

GUI – graphical user interface (графічний інтерфейс користувача).

IDE – integrated development environment (інтегроване середовище розробки).

# 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

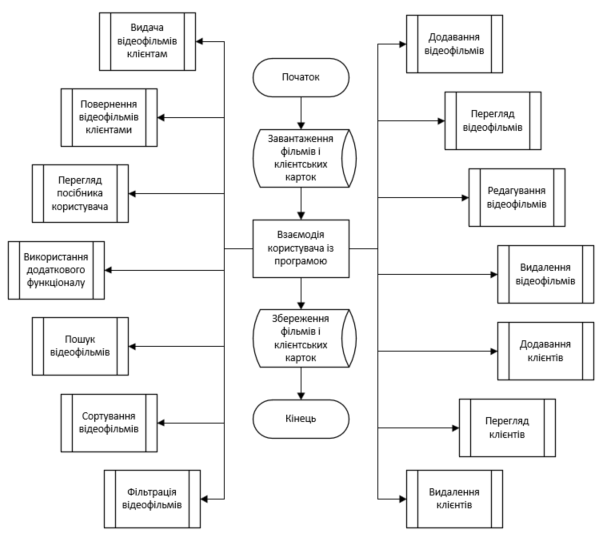
****

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).

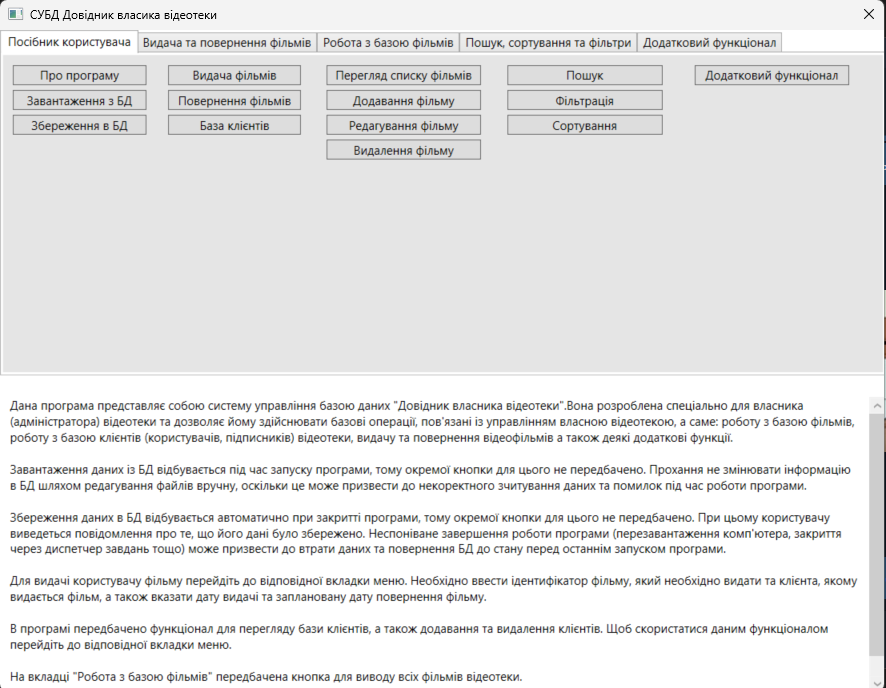


Рисунок 2 (посібник користувача)

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

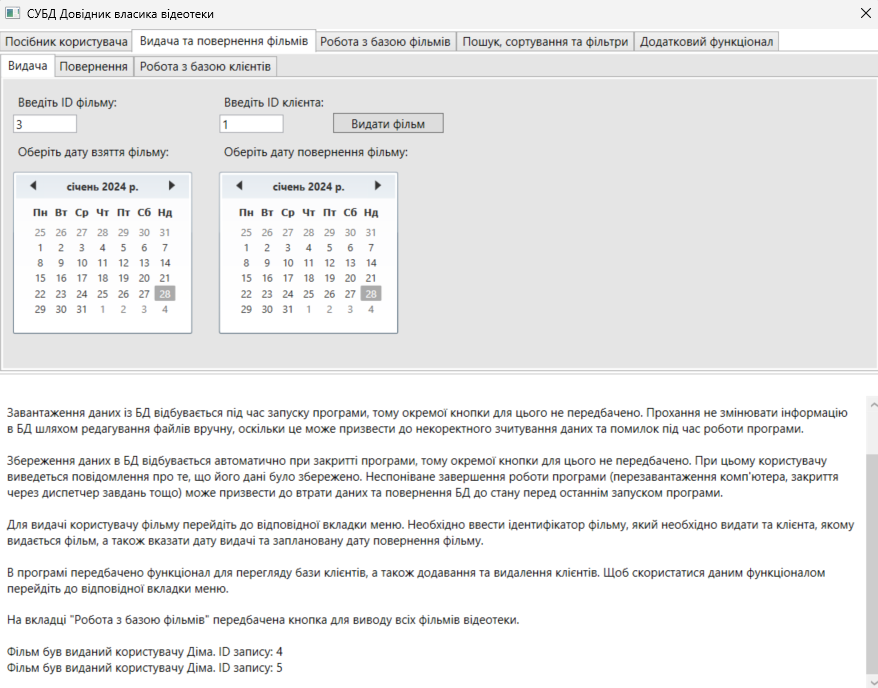
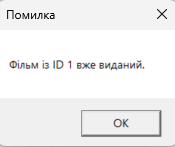
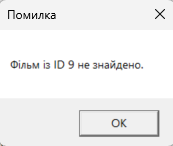
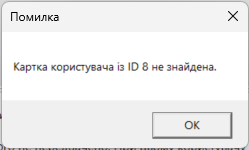
 При переході до меню «Видача та повернення фільмів» користувач зможе скористатися можливостями програми з видачі та повернення фільмів, а також створення та видалення клієнтів відеотеки (рис. 3-9).

Рисунок 3 (видача фільмів)



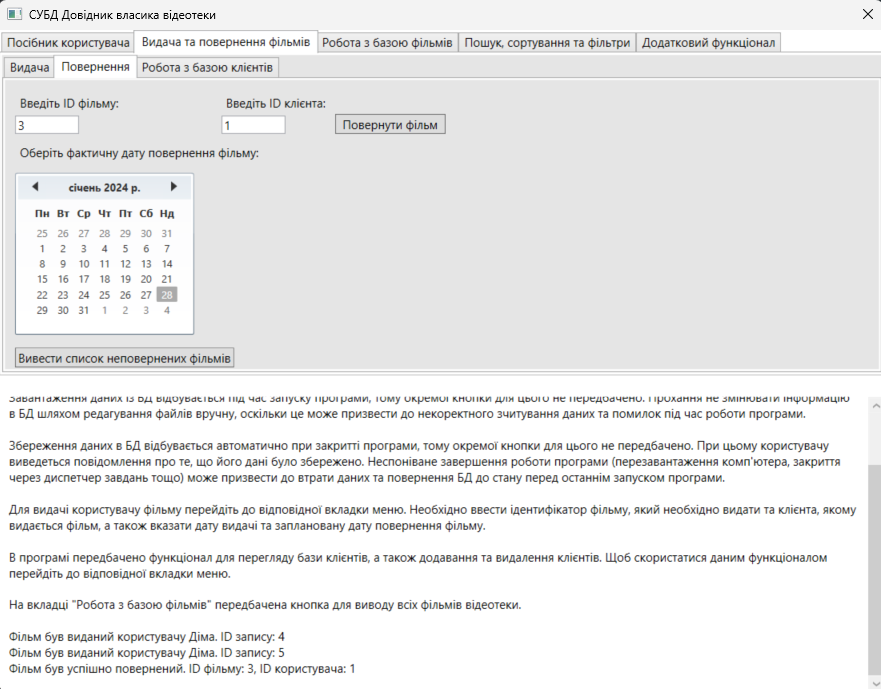
Рисунки 4-6 (помилки під час видачі фільмів)

Рисунок 7 (повернення фільмів)

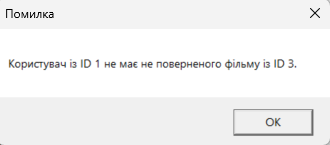


Рисунок 8 (помилка під час повернення фільмів)

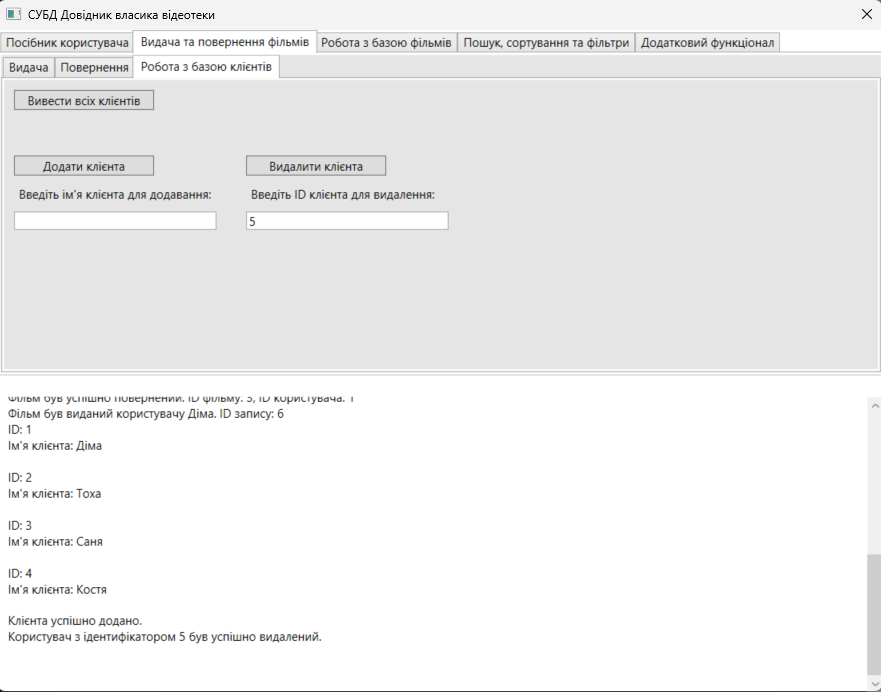
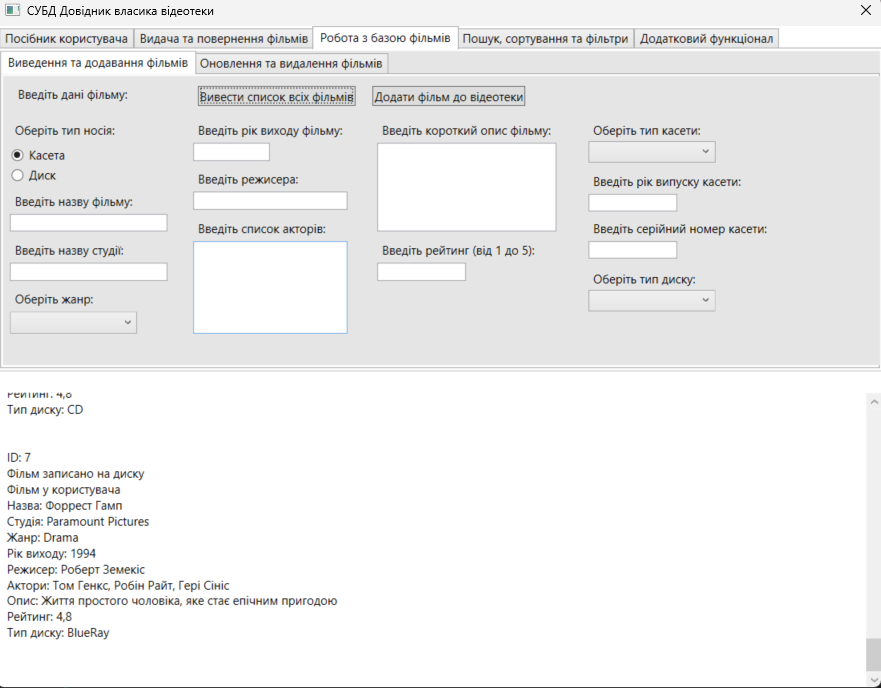
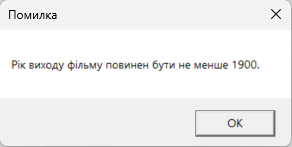
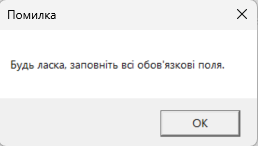


Рисунок 9 (меню додавання та видалення клієнтів)

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)

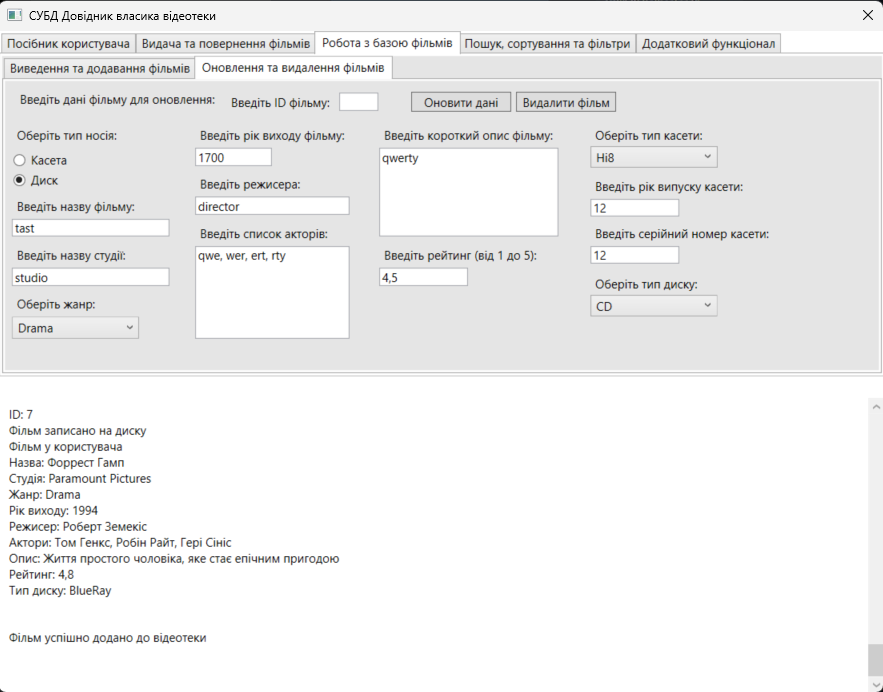


Рисунок 13 (редагування та видалення фільмів)

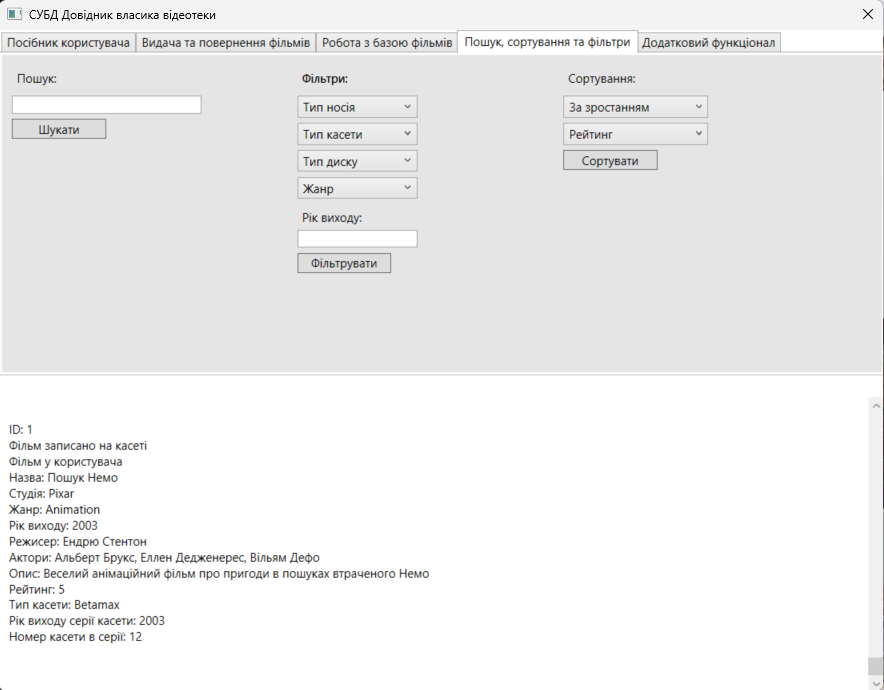
При переході до меню «Пошук, сортування та фільтри» користувач отримає доступ до відповідного функціоналу (рис. 14).

Рисунок 14 (пошук, сортування та фільтрація)

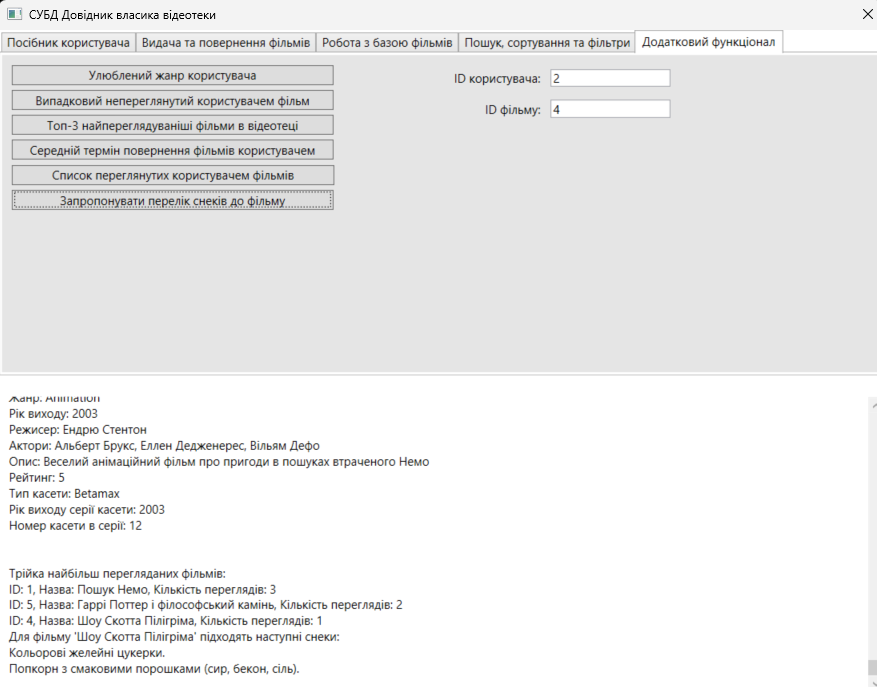
При переході до меню «додатковий функціонал» користувач отримає доступ до додаткових функцій, доступних в програмі (рис. 15).

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://learn.microsoft.com/en-us/dotnet/csharp/linq/>
2. <https://learn.microsoft.com/en-us/dotnet/csharp/>
3. <https://www.youtube.com/watch?v=oRFTxRTVxP0>
4. <https://learn.microsoft.com/en-us/dotnet/api/system.collections.generic?view=net-7.0>
5. <https://stackoverflow.com/questions/42758337/how-to-read-write-process-txt-file-like-database-in-c-sharp-console>
6. <https://github.com/Nibe34/Course-project>

# ДОДАТКИ

## ДОДАТОК А. Скролінг (текст) програми

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace Video\_library\_owner\_handbook

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

InitialSettings();

mediaTypeFilter.SelectedIndex = 0;

genreFilter.SelectedIndex = 0;

cassetteTypeFilter.SelectedIndex = 0;

discTypeFilter.SelectedIndex = 0;

sortType.SelectedIndex = 0;

sortCriterion.SelectedIndex = 0;

Closing += MainWindow\_Closing;

}

// початкові налаштування (завантаження даних)

private static void InitialSettings()

{

FileManager.ReadFilmsFromFile();

FileManager.ReadUserCardsFromFile();

}

// збереження даних

private void MainWindow\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

{

MessageBox.Show("Всі зміни будуть збережені.", "Збереження змін", MessageBoxButton.OK);

FileManager.WriteFilmsToFile();

FileManager.WriteUserCardsToFile();

}

// вивід повідомлень

public void PrintMessage(string message)

{

outputTextBlock.Text += message + Environment.NewLine;

}

// посібник користувача

private void Help\_AboutProgram\_Click(object sender, RoutedEventArgs e)

{

PrintMessage("Дана програма представляє собою систему управління базою даних \"Довідник власника відеотеки\"." +

"Вона розроблена спеціально для власника (адміністратора) відеотеки та дозволяє йому здійснювати базові " +

"операції, пов'язані із управлінням власною відеотекою, а саме: роботу з базою фільмів, роботу з базою " +

"клієнтів (користувачів, підписників) відеотеки, видачу та повернення відеофільмів а також деякі додаткові функції.\n");

}

private void Help\_LoadFromDB\_Click(object sender, RoutedEventArgs e)

{

PrintMessage("Завантаження даних із БД відбувається під час запуску програми, тому окремої кнопки для цього " +

"не передбачено. Прохання не змінювати інформацію в БД шляхом редагування файлів вручну, оскільки це " +

"може призвести до некоректного зчитування даних та помилок під час роботи програми.\n");

}

private void Help\_SaveToDB\_Click(object sender, RoutedEventArgs e)

{

PrintMessage("Збереження даних в БД відбувається автоматично при закритті програми, тому окремої кнопки для цього " +

"не передбачено. При цьому користувачу виведеться повідомлення про те, що його дані було збережено. " +

"Неспоніване завершення роботи програми (перезавантаження комп'ютера, закриття через диспетчер завдань тощо) " +

"може призвести до втрати даних та повернення БД до стану перед останнім запуском програми.\n");

}

private void Help\_GiveFilm\_Click(object sender, RoutedEventArgs e)

{

PrintMessage("Для видачі користувачу фільму перейдіть до відповідної вкладки меню. Необхідно ввести ідентифікатор " +

"фільму, який необхідно видати та клієнта, якому видається фільм, а також вказати дату видачі та заплановану " +

"дату повернення фільму.\n");

}

private void Help\_ReturnFilm\_Click(object sender, RoutedEventArgs e)

{

PrintMessage("Для повернення фільму перейдіть до відповідної вкладки меню. Необхідно ввести ідентифікатор " +

"фільму, який повертається клієнтом та самого клієнта, а також вказати фактичну дату повернення фільму. " +

"Якщо фактична дата повернення пізніше запланованої, на екран виведеться повідомлення із проханням ввести " +

"санкції стосовно клієнта. Розмір ти вид санкцій встановлює адміністратор відеотеки. \nТакож в програмі " +

"передбачена спеціальна кнопка для виводу списку неповернених клієнтом відеофільмів.\n");

}

private void Help\_ClientBase\_Click(object sender, RoutedEventArgs e)

{

PrintMessage("В програмі передбачено функціонал для перегляду бази клієнтів, а також додавання та видалення " +

"клієнтів. Щоб скористатися даним функціоналом перейдіть до відповідної вкладки меню.\n");

}

private void Help\_FilmsList\_Click(object sender, RoutedEventArgs e)

{

PrintMessage("На вкладці \"Робота з базою фільмів\" передбачена кнопка для виводу всіх фільмів відеотеки.\n");

}

private void Help\_AddFilm\_Click(object sender, RoutedEventArgs e)

{

PrintMessage("На вкладці \"Робота з базою фільмів\" передбачена кнопка для додавання фільму до відеотеки. " +

"Для додавання нового фільму необхідно спершу заповнити всі поля з інформацією щодо фільму. " +

"Також при додаванні фільму необхідно обрати тип носія, на якому записаний цей фільм (диск або відеокасета). " +

"Для кожного з цих носіїв передбачено унікальні поля (наприклад тип диску чи серійний номер касети). " +

"Заповнення полів, що не надежать обраному типу носія не має сенсу.\n");

}

private void Help\_UpdateFilm\_Click(object sender, RoutedEventArgs e)

{

PrintMessage("На вкладці \"Робота з базою фільмів\" передбачена кнопка для редагування (оновлення) фільму. " +

"Для редагування фільму необхідно заповнити всі поля з оновленою інформацією щодо фільму, а також поле ID.\n");

}

private void Help\_DeleteFilm\_Click(object sender, RoutedEventArgs e)

{

PrintMessage("На вкладці \"Робота з базою фільмів\" передбачена кнопка для видалення фільму. для видалення необхідно " +

"ввести ідентифікатор (не)потрібного фільму та натиснути на кнопку.\n");

}

private void Help\_Search\_Click(object sender, RoutedEventArgs e)

{

PrintMessage("Для пошуку фільму перейдіть на відповідну вкладку меню та введіть ключове слово (фрагмент). " +

"Пошук відбувається за назвою фільму, режисером та назвою студії.\n");

}

private void Help\_Filter\_Click(object sender, RoutedEventArgs e)

{

PrintMessage("Для фільтрації фільмів перейдіть на відповідну вкладку меню та оберіть бажані параметри фільтрації. " +

"Вибір \"Типу диску\" та \"Типу касети\" має сенс лише якщо ви обрали відповідний тип носія. Жоден фільтр " +

"не є обов'язковим.\n");

}

private void Help\_Sort\_Click(object sender, RoutedEventArgs e)

{

PrintMessage("Для сортування фільмів перейдіть на відповідну вкладку меню та оберіть тип та критерій сортування.\n");

}

private void Help\_AdditionalFeatures\_Click(object sender, RoutedEventArgs e)

{

PrintMessage("Вкладка \"Додатковий функціонал\" містить різні функції, що можуть бути корисними залежно від ситуації, " +

"а саме: \"Визначити улюблений жанр користувача\", \"Запропонувати випадковий непереглянутий користувачем фільм\", " +

"\"Вивести топ-3 найпереглядуваніші фільми в відеотеці\", \"Перевірити середній термін повернення фільмів користувачем\", " +

"\"Пеоеглянути весь список фільмів, які брав користувач\" та \"Запропонувати перелік снеків до фільму\".\n");

}

// CRUD

private void WriteAllFilms\_Click(object sender, RoutedEventArgs e)

{

foreach (var videoFilm in VideoFilmLibrary.GetVideoFilms())

{

if (videoFilm is VideoFilmOnCassette videoFilmOnCassette)

{

PrintMessage(videoFilmOnCassette.ToStringForWrite());

}

else if (videoFilm is VideoFilmOnDisc videoFilmOnDisc)

{

PrintMessage(videoFilmOnDisc.ToStringForWrite());

}

}

}

private void AddNewFilm\_Click(object sender, RoutedEventArgs e)

{

try

{

if (CRUD\_isCassette.IsChecked == true)

{

if (string.IsNullOrWhiteSpace(CRUD\_title.Text) ||

string.IsNullOrWhiteSpace(CRUD\_studio.Text) ||

CRUD\_genre.SelectedItem == null ||

string.IsNullOrWhiteSpace(CRUD\_director.Text) ||

string.IsNullOrWhiteSpace(CRUD\_actors.Text) ||

string.IsNullOrWhiteSpace(CRUD\_summary.Text) ||

string.IsNullOrWhiteSpace(CRUD\_cassetteType.SelectedItem?.ToString()) ||

string.IsNullOrWhiteSpace(CRUD\_cassetteYear.Text) ||

string.IsNullOrWhiteSpace(CRUD\_cassetteSerialNumber.Text))

{

MessageBox.Show("Будь ласка, заповніть всі обов'язкові поля.", "Помилка");

return;

}

double ratingValue;

if (!double.TryParse(CRUD\_rating.Text, out ratingValue) || ratingValue < 1 || ratingValue > 5)

{

MessageBox.Show("Поле 'Rating' повинно містити значення від 1 до 5.", "Помилка");

return;

}

if (string.IsNullOrWhiteSpace(CRUD\_releaseYear.Text) ||

int.Parse(CRUD\_releaseYear.Text) < 1900)

{

MessageBox.Show("Рік виходу фільму повинен бути не менше 1900.", "Помилка");

}

if (string.IsNullOrWhiteSpace(CRUD\_cassetteYear.Text) ||

int.Parse(CRUD\_cassetteYear.Text) < 1900)

{

MessageBox.Show("Рік виходу касети повинен бути не менше 1900.", "Помилка");

}

if (string.IsNullOrWhiteSpace(CRUD\_cassetteSerialNumber.Text) ||

int.Parse(CRUD\_cassetteSerialNumber.Text) < 0)

{

MessageBox.Show("Серійний номер касети повинен бути не менше 0.", "Помилка");

}

VideoFilmOnCassette videoFilmOnCassette = new VideoFilmOnCassette(

VideoFilmLibrary.FindMinUnusedFilmID(),

true,

CRUD\_title.Text,

CRUD\_studio.Text,

(CRUD\_genre.SelectedItem as ComboBoxItem)?.Content?.ToString(),

int.Parse(CRUD\_releaseYear.Text),

CRUD\_director.Text,

CRUD\_actors.Text,

CRUD\_summary.Text,

ratingValue,

(CRUD\_cassetteType.SelectedItem as ComboBoxItem)?.Content?.ToString(),

int.Parse(CRUD\_cassetteYear.Text),

int.Parse(CRUD\_cassetteSerialNumber.Text));

VideoFilmLibrary.AddFilm(videoFilmOnCassette);

}

if (CRUD\_isDisc.IsChecked == true)

{

if (string.IsNullOrWhiteSpace(CRUD\_title.Text) ||

string.IsNullOrWhiteSpace(CRUD\_studio.Text) ||

CRUD\_genre.SelectedItem == null ||

string.IsNullOrWhiteSpace(CRUD\_director.Text) ||

string.IsNullOrWhiteSpace(CRUD\_actors.Text) ||

string.IsNullOrWhiteSpace(CRUD\_summary.Text) ||

string.IsNullOrWhiteSpace(CRUD\_discType.SelectedItem?.ToString()))

{

MessageBox.Show("Будь ласка, заповніть всі обов'язкові поля.", "Помилка");

return;

}

double ratingValue;

if (!double.TryParse(CRUD\_rating.Text, out ratingValue) || ratingValue < 1 || ratingValue > 5)

{

MessageBox.Show("Поле 'Rating' повинно містити значення від 1 до 5.", "Помилка");

return;

}

if (string.IsNullOrWhiteSpace(CRUD\_releaseYear.Text) ||

int.Parse(CRUD\_releaseYear.Text) < 1900)

{

MessageBox.Show("Рік виходу фільму повинен бути не менше 1900.", "Помилка");

}

VideoFilmOnDisc videoFilmOnDisc = new VideoFilmOnDisc(

VideoFilmLibrary.FindMinUnusedFilmID(),

true,

CRUD\_title.Text,

CRUD\_studio.Text,

(CRUD\_genre.SelectedItem as ComboBoxItem)?.Content?.ToString(),

int.Parse(CRUD\_releaseYear.Text),

CRUD\_director.Text,

CRUD\_actors.Text,

CRUD\_summary.Text,

ratingValue,

(CRUD\_discType.SelectedItem as ComboBoxItem)?.Content?.ToString());

VideoFilmLibrary.AddFilm(videoFilmOnDisc);

}

PrintMessage("Фільм успішно додано до відеотеки\n\n");

}

catch (FormatException)

{

MessageBox.Show("Некоректний формат введених даних. Будь ласка, перевірте правильність введення числових значень.", "Помилка");

}

catch (Exception ex)

{

MessageBox.Show($"Сталася помилка: {ex.Message}", "Помилка");

}

}

private void DeleteFilm\_Click(object sender, RoutedEventArgs e)

{

try

{

int filmId;

if (!int.TryParse(CRUD\_filmID.Text, out filmId))

{

MessageBox.Show("Некоректний формат введеного ідентифікатора фільму.", "Помилка");

return;

}

string removalResult = VideoFilmLibrary.RemoveFilmById(filmId);

PrintMessage(removalResult);

}

catch (Exception ex)

{

MessageBox.Show($"Сталася помилка: {ex.Message}", "Помилка");

}

}

private void UpdateFilm\_Click(object sender, RoutedEventArgs e)

{

try

{

int filmId;

if (!int.TryParse(CRUD\_filmID.Text, out filmId))

{

MessageBox.Show("Некоректний формат введеного ідентифікатора фільму.", "Помилка");

return;

}

if (string.IsNullOrWhiteSpace(CRUD\_title.Text) ||

string.IsNullOrWhiteSpace(CRUD\_studio.Text) ||

CRUD\_genre.SelectedItem == null ||

string.IsNullOrWhiteSpace(CRUD\_director.Text) ||

string.IsNullOrWhiteSpace(CRUD\_actors.Text) ||

string.IsNullOrWhiteSpace(CRUD\_summary.Text) ||

string.IsNullOrWhiteSpace(CRUD\_cassetteType.SelectedItem?.ToString()) ||

string.IsNullOrWhiteSpace(CRUD\_cassetteYear.Text) ||

string.IsNullOrWhiteSpace(CRUD\_cassetteSerialNumber.Text))

{

MessageBox.Show("Будь ласка, заповніть всі обов'язкові поля.", "Помилка");

return;

}

double ratingValue;

if (!double.TryParse(CRUD\_rating.Text, out ratingValue) || ratingValue < 1 || ratingValue > 5)

{

MessageBox.Show("Поле 'Rating' повинно містити значення від 1 до 5.", "Помилка");

return;

}

if (string.IsNullOrWhiteSpace(CRUD\_releaseYear.Text) ||

int.Parse(CRUD\_releaseYear.Text) < 1900)

{

MessageBox.Show("Рік виходу фільму повинен бути не менше 1900.", "Помилка");

}

if (string.IsNullOrWhiteSpace(CRUD\_cassetteYear.Text) ||

int.Parse(CRUD\_cassetteYear.Text) < 1900)

{

MessageBox.Show("Рік виходу касети повинен бути не менше 1900.", "Помилка");

}

if (string.IsNullOrWhiteSpace(CRUD\_cassetteSerialNumber.Text) ||

int.Parse(CRUD\_cassetteSerialNumber.Text) < 0)

{

MessageBox.Show("Серійний номер касети повинен бути не менше 0.", "Помилка");

}

string removalResult = VideoFilmLibrary.RemoveFilmById(filmId);

PrintMessage(removalResult);

if (removalResult.StartsWith("Фільм з ідентифікатором"))

{

if (CRUD\_isCassette.IsChecked == true)

{

VideoFilmOnCassette videoFilmOnCassette = new VideoFilmOnCassette(

filmId,

true,

CRUD\_title.Text,

CRUD\_studio.Text,

(CRUD\_genre.SelectedItem as ComboBoxItem)?.Content?.ToString(),

int.Parse(CRUD\_releaseYear.Text),

CRUD\_director.Text,

CRUD\_actors.Text,

CRUD\_summary.Text,

ratingValue,

(CRUD\_cassetteType.SelectedItem as ComboBoxItem)?.Content?.ToString(),

int.Parse(CRUD\_cassetteYear.Text),

int.Parse(CRUD\_cassetteSerialNumber.Text));

VideoFilmLibrary.AddFilm(videoFilmOnCassette);

}

else if (CRUD\_isDisc.IsChecked == true)

{

VideoFilmOnDisc videoFilmOnDisc = new VideoFilmOnDisc(

filmId,

true,

CRUD\_title.Text,

CRUD\_studio.Text,

(CRUD\_genre.SelectedItem as ComboBoxItem)?.Content?.ToString(),

int.Parse(CRUD\_releaseYear.Text),

CRUD\_director.Text,

CRUD\_actors.Text,

CRUD\_summary.Text,

ratingValue,

(CRUD\_discType.SelectedItem as ComboBoxItem)?.Content?.ToString());

VideoFilmLibrary.AddFilm(videoFilmOnDisc);

}

PrintMessage("Фільм успішно оновлено.\n\n");

}

}

catch (FormatException)

{

MessageBox.Show("Некоректний формат введених даних. Будь ласка, перевірте правильність введення числових значень.", "Помилка");

}

catch (Exception ex)

{

MessageBox.Show($"Сталася помилка: {ex.Message}", "Помилка");

}

}

// пошук, сортування та фільтрація

private void Search\_Click(object sender, RoutedEventArgs e)

{

List<VideoFilm> foundVideoFilms = VideoFilmLibrary.GetVideoFilms()

.Where(obj => obj.Title.Contains(searchField.Text) || obj.Director.Contains(searchField.Text) || obj.Studio.Contains(searchField.Text))

.ToList();

if (foundVideoFilms.Count == 0)

{

PrintMessage("Фільми за заданим запитом не знайдено.");

}

else

{

PrintMessage("Фільми за заданим запитом:");

}

foreach(var videoFilm in foundVideoFilms)

{

if (videoFilm is VideoFilmOnCassette videoFilmOnCassette)

{

PrintMessage(videoFilmOnCassette.ToStringForWrite());

}

else if (videoFilm is VideoFilmOnDisc videoFilmOnDisc)

{

PrintMessage(videoFilmOnDisc.ToStringForWrite());

}

}

}

private void Filter\_Click(object sender, RoutedEventArgs e)

{

if (mediaTypeFilter.SelectedIndex == 0 &&

discTypeFilter.SelectedIndex == 0 &&

cassetteTypeFilter.SelectedIndex == 0 &&

genreFilter.SelectedIndex == 0 &&

string.IsNullOrWhiteSpace(yearFilter.Text))

{

MessageBox.Show("Oберіть принаймні один фільтр.", "Помилка");

return;

}

List<VideoFilm> filteredFilms = VideoFilmLibrary.GetVideoFilms();

if (!string.IsNullOrEmpty(yearFilter.Text))

{

if (int.TryParse(mediaTypeFilter.Text, out int value))

{

MessageBox.Show("Поле \"рік\" може містити лише цілі числа.", "Помилка");

return;

}

filteredFilms = filteredFilms

.Where(film => film.ReleaseYear == int.Parse(yearFilter.Text))

.ToList();

}

if (genreFilter.SelectedIndex != 0)

{

filteredFilms = filteredFilms

.Where(film => film.Genre == (Genres)Enum.Parse(typeof(Genres), ((ComboBoxItem)genreFilter.SelectedItem).Content.ToString()))

.ToList();

}

if (mediaTypeFilter.SelectedIndex == 1)

{

List<VideoFilmOnCassette> cassetteVideoFilms = filteredFilms

.OfType<VideoFilmOnCassette>()

.ToList();

if (cassetteTypeFilter.SelectedIndex != 0)

{

cassetteVideoFilms = cassetteVideoFilms

.Where(film => film.CassetteType == (CassetteTypes)Enum.Parse(typeof(CassetteTypes), ((ComboBoxItem)cassetteTypeFilter.SelectedItem).Content.ToString()))

.ToList();

}

foreach (var videoFilm in cassetteVideoFilms)

{

PrintMessage(videoFilm.ToStringForWrite());

}

return;

}

if (mediaTypeFilter.SelectedIndex == 2)

{

List<VideoFilmOnDisc> discVideoFilms = filteredFilms

.OfType<VideoFilmOnDisc>()

.ToList();

if (discTypeFilter.SelectedIndex != 0)

{

discVideoFilms = discVideoFilms

.Where(film => film.DiscType == (DiscTypes)Enum.Parse(typeof(DiscTypes), ((ComboBoxItem)discTypeFilter.SelectedItem).Content.ToString()))

.ToList();

}

foreach (var videoFilm in discVideoFilms)

{

PrintMessage(videoFilm.ToStringForWrite());

}

return;

}

foreach (var videoFilm in filteredFilms)

{

if (videoFilm is VideoFilmOnCassette videoFilmOnCassette)

{

PrintMessage(videoFilmOnCassette.ToStringForWrite());

}

else if (videoFilm is VideoFilmOnDisc videoFilmOnDisc)

{

PrintMessage(videoFilmOnDisc.ToStringForWrite());

}

}

}

private void Sort\_Click(object sender, RoutedEventArgs e)

{

if (sortType.SelectedIndex == 0)

{

MessageBox.Show("Oберіть тип сортування.", "Помилка");

return;

}

List<VideoFilm> sortedVideoFilms = new List<VideoFilm>();

if (sortCriterion.SelectedIndex == 0)

{

sortedVideoFilms = sortType.SelectedIndex == 1

? VideoFilmLibrary.GetVideoFilms().OrderBy(film => film.Id).ToList()

: VideoFilmLibrary.GetVideoFilms().OrderByDescending(film => film.Id).ToList();

}

else if (sortCriterion.SelectedIndex == 1)

{

sortedVideoFilms = sortType.SelectedIndex == 1

? VideoFilmLibrary.GetVideoFilms().OrderBy(film => film.Title).ToList()

: VideoFilmLibrary.GetVideoFilms().OrderByDescending(film => film.Title).ToList();

}

else if (sortCriterion.SelectedIndex == 2)

{

sortedVideoFilms = sortType.SelectedIndex == 1

? VideoFilmLibrary.GetVideoFilms().OrderBy(film => film.ReleaseYear).ToList()

: VideoFilmLibrary.GetVideoFilms().OrderByDescending(film => film.ReleaseYear).ToList();

}

else if (sortCriterion.SelectedIndex == 3)

{

sortedVideoFilms = sortType.SelectedIndex == 1

? VideoFilmLibrary.GetVideoFilms().OrderBy(film => film.Rating).ToList()

: VideoFilmLibrary.GetVideoFilms().OrderByDescending(film => film.Rating).ToList();

}

foreach (var videoFilm in sortedVideoFilms)

{

if (videoFilm is VideoFilmOnCassette videoFilmOnCassette)

{

PrintMessage(videoFilmOnCassette.ToStringForWrite());

}

else if (videoFilm is VideoFilmOnDisc videoFilmOnDisc)

{

PrintMessage(videoFilmOnDisc.ToStringForWrite());

}

}

}

// робота з клієнтами, видачею та поверненням фільмів

private void PrintAllClients\_Click(object sender, RoutedEventArgs e)

{

foreach (var client in VideoFilmLibrary.GetUserCards())

{

PrintMessage(client.ToStringForWrite());

}

}

private void AddNewClient\_Click(object sender, RoutedEventArgs e)

{

VideoFilmLibrary.AddUserCard(new UserCard(VideoFilmLibrary.FindMinUnusedUserID(), addClientName.Text));

PrintMessage("Клієнта успішно додано.");

}

private void DeleteClient\_Click(object sender, RoutedEventArgs e)

{

try

{

int userId;

if (!int.TryParse(deleteClientID.Text, out userId))

{

MessageBox.Show("Некоректний формат введеного ідентифікатора клієнта.", "Помилка");

return;

}

string removalResult = VideoFilmLibrary.RemoveUserCardById(userId);

PrintMessage(removalResult);

}

catch (Exception ex)

{

MessageBox.Show($"Сталася помилка: {ex.Message}", "Помилка");

}

}

private void GiveFilm\_Click(object sender, RoutedEventArgs e)

{

try

{

int filmId = int.Parse(getFilmID.Text);

DateTime takingDate = getTakeDate.SelectedDate ?? DateTime.Now;

DateTime returnDate = getReturnDate.SelectedDate ?? DateTime.Now;

int clientId = int.Parse(getClientID.Text);

VideoFilm film = VideoFilmLibrary.GetFilmById(filmId);

if (film == null)

{

MessageBox.Show($"Фільм із ID {filmId} не знайдено.", "Помилка");

return;

}

if (!film.InVideoLibrary)

{

MessageBox.Show($"Фільм із ID {filmId} вже виданий.", "Помилка");

return;

}

if (returnDate < takingDate)

{

MessageBox.Show("Дата повернення повинна бути пізніше за дату взяття.", "Помилка");

return;

}

UserCard client = VideoFilmLibrary.GetUserCardById(clientId);

if (client == null)

{

MessageBox.Show($"Картка користувача із ID {clientId} не знайдена.", "Помилка");

return;

}

int recordId = client.FindMinUnusedRecordID();

Record record = new Record(recordId, filmId, takingDate, returnDate, false);

client.AddRecordToUserCard(record);

VideoFilmLibrary.GetFilmById(filmId).InVideoLibrary = false;

PrintMessage($"Фільм був виданий користувачу {client.UserName}. ID запису: {recordId}");

}

catch (FormatException)

{

MessageBox.Show("Некоректний формат введених даних.", "Помилка");

}

catch (Exception ex)

{

MessageBox.Show($"Сталася помилка: {ex.Message}", "Помилка");

}

}

private void ReturnFilm\_Click(object sender, RoutedEventArgs e)

{

try

{

int filmId = int.Parse(returnFilmID.Text);

int clientId = int.Parse(returnClientID.Text);

DateTime date = factReturnDate.SelectedDate ?? DateTime.Now;

VideoFilm film = VideoFilmLibrary.GetFilmById(filmId);

if (film == null)

{

MessageBox.Show($"Фільм із ID {filmId} не знайдено.", "Помилка");

return;

}

UserCard client = VideoFilmLibrary.GetUserCardById(clientId);

if (client == null)

{

MessageBox.Show($"Картка користувача із ID {clientId} не знайдена.", "Помилка");

return;

}

Record recordToReturn = client.Records.FirstOrDefault(record =>

record.VideoFilmId == filmId && !record.WasReturned);

if (recordToReturn == null)

{

MessageBox.Show($"Користувач із ID {clientId} не має не поверненого фільму із ID {filmId}.", "Помилка");

return;

}

if (date > DateTime.Now)

{

MessageBox.Show("Фактична дата повернення не може бути в майбутньому.", "Помилка");

return;

}

if (date > recordToReturn.ReturningDate)

{

MessageBox.Show("Фактична дата повернення пізніше запланованої. Введіть санкції стосовно клієнта.", "Попередження");

}

film.InVideoLibrary = true;

recordToReturn.WasReturned = true;

PrintMessage($"Фільм був успішно повернений. ID фільму: {filmId}, ID користувача: {clientId}");

}

catch (FormatException)

{

MessageBox.Show("Некоректний формат введених даних.", "Помилка");

}

catch (Exception ex)

{

MessageBox.Show($"Сталася помилка: {ex.Message}", "Помилка");

}

}

private void GetUnreturnedFilmsByUserId(object sender, RoutedEventArgs e)

{

try

{

int clientId = int.Parse(returnClientID.Text);

UserCard client = VideoFilmLibrary.GetUserCardById(clientId);

if (client == null)

{

MessageBox.Show($"Картка користувача із ID {clientId} не знайдена.", "Помилка");

return;

}

List<Record> unreturnedRecords = client

.Records.Where(record => !record.WasReturned)

.ToList();

if (unreturnedRecords.Count == 0)

{

MessageBox.Show($"У користувача із ID {clientId} немає неповернених фільмів.", "Повідомлення");

}

else

{

PrintMessage($"Неповернені фільми користувача із ID {clientId}:");

foreach (Record record in unreturnedRecords)

{

VideoFilm film = VideoFilmLibrary.GetFilmById(record.VideoFilmId);

PrintMessage($"- ID фільму: {film.Id}, Назва: {film.Title}, Дата взяття: {record.TakingDate.ToShortDateString()}");

}

}

}

catch (FormatException)

{

MessageBox.Show("Некоректний формат введених даних.", "Помилка");

}

catch (Exception ex)

{

MessageBox.Show($"Сталася помилка: {ex.Message}", "Помилка");

}

}

// додаткові функції

private void FavoriteGenreByUserID\_Click(object sender, RoutedEventArgs e)

{

try

{

if (!int.TryParse(featuresUserID.Text, out int userId))

{

MessageBox.Show("Некоректний формат ID користувача.", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

UserCard user = VideoFilmLibrary.GetUserCardById(userId);

if (user == null)

{

MessageBox.Show($"Користувача з ID {userId} не знайдено.", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

List<Record> userRecords = user.GetRecords();

if (userRecords.Count == 0)

{

PrintMessage($"Користувач із ID {userId} ще не дивився жодного фільму.");

return;

}

var genreCounts = userRecords

.GroupBy(record => VideoFilmLibrary.GetFilmById(record.VideoFilmId)?.Genre)

.Where(group => group.Key != null)

.Select(group => new { Genre = group.Key, Count = group.Count() })

.OrderByDescending(item => item.Count)

.ToList();

if (genreCounts.Count == 0)

{

PrintMessage($"Користувач із ID {userId} ще не дивився фільми з визначеними жанрами.");

return;

}

int maxCount = genreCounts.First().Count;

var favoriteGenres = genreCounts

.Where(item => item.Count == maxCount)

.Select(item => item.Genre.ToString())

.ToList();

PrintMessage($"Улюблені жанри користувача із ID {userId}: {string.Join(", ", favoriteGenres)}");

}

catch (Exception ex)

{

MessageBox.Show($"Сталася помилка: {ex.Message}", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void RandomNotViewedFilmByUserID\_Click(object sender, RoutedEventArgs e)

{

try

{

if (!int.TryParse(featuresUserID.Text, out int userId))

{

MessageBox.Show("Некоректний формат ID користувача.", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

UserCard user = VideoFilmLibrary.GetUserCardById(userId);

if (user == null)

{

MessageBox.Show($"Користувача з ID {userId} не знайдено.", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

List<VideoFilm> notViewedFilms = VideoFilmLibrary.GetVideoFilms()

.Where(film => !user.GetRecords().Any(record => record.VideoFilmId == film.Id))

.ToList();

if (notViewedFilms.Count == 0)

{

PrintMessage($"Користувач із ID {userId} переглянув всі фільми.");

return;

}

Random random = new Random();

VideoFilm randomFilm = notViewedFilms[random.Next(notViewedFilms.Count)];

if (randomFilm is VideoFilmOnCassette videoFilmOnCassette)

{

PrintMessage($"Випадковий фільм для користувача із ID {userId}: \n{videoFilmOnCassette.ToStringForWrite()}");

}

else if (randomFilm is VideoFilmOnDisc videoFilmOnDisc)

{

PrintMessage($"Випадковий фільм для користувача із ID {userId}: \n{videoFilmOnDisc.ToStringForWrite()}");

}

}

catch (Exception ex)

{

MessageBox.Show($"Сталася помилка: {ex.Message}", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void Top3MostWatchedFilmsInLibrary\_Click(object sender, RoutedEventArgs e)

{

try

{

List<VideoFilm> allFilms = VideoFilmLibrary.GetVideoFilms();

if (allFilms.Count == 0)

{

PrintMessage("Відеотека порожня.");

return;

}

Dictionary<VideoFilm, int> filmViewsCount = new Dictionary<VideoFilm, int>();

foreach (UserCard userCard in VideoFilmLibrary.GetUserCards())

{

foreach (Record record in userCard.GetRecords())

{

VideoFilm film = allFilms.FirstOrDefault(f => f.Id == record.VideoFilmId);

if (film != null)

{

if (!filmViewsCount.ContainsKey(film))

{

filmViewsCount[film] = 1;

}

else

{

filmViewsCount[film]++;

}

}

}

}

if (filmViewsCount.Count == 0)

{

PrintMessage("Немає переглядів фільмів відеотеки.");

return;

}

List<VideoFilm> top3Films = filmViewsCount.OrderByDescending(kv => kv.Value).Take(3).Select(kv => kv.Key).ToList();

PrintMessage("Трійка найбільш перегляданих фільмів:");

foreach (VideoFilm film in top3Films)

{

PrintMessage($"ID: {film.Id}, Назва: {film.Title}, Кількість переглядів: {filmViewsCount[film]}");

}

}

catch (Exception ex)

{

MessageBox.Show($"Сталася помилка: {ex.Message}", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void AverageFilmReturningTimeByUserID\_Click(object sender, RoutedEventArgs e)

{

try

{

int userId = int.Parse(featuresUserID.Text);

UserCard userCard = VideoFilmLibrary.GetUserCardById(userId);

if (userCard == null)

{

MessageBox.Show($"Картка користувача із ID {userId} не знайдена.", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

List<Record> userRecords = userCard.GetRecords();

if (userRecords.Count == 0)

{

MessageBox.Show($"Користувач із ID {userId} ще не переглядав фільми.", "Інформація", MessageBoxButton.OK, MessageBoxImage.Information);

return;

}

List<TimeSpan> returningTimes = new List<TimeSpan>();

foreach (Record record in userRecords)

{

if (record.WasReturned)

{

TimeSpan returningTime = record.ReturningDate - record.TakingDate;

returningTimes.Add(returningTime);

}

}

if (returningTimes.Count == 0)

{

MessageBox.Show($"Користувач із ID {userId} ще не повертав фільми.", "Інформація", MessageBoxButton.OK, MessageBoxImage.Information);

return;

}

double averageReturningTime = returningTimes.Average(time => time.TotalDays);

PrintMessage($"Середній час повернення фільмів для користувача із ID {userId}: {averageReturningTime:F2} дні(в).");

}

catch (FormatException)

{

MessageBox.Show("Некоректний формат введених даних.", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

catch (Exception ex)

{

MessageBox.Show($"Сталася помилка: {ex.Message}", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void AllWatchedFilmsByUserID(object sender, RoutedEventArgs e)

{

try

{

int userId = int.Parse(featuresUserID.Text);

UserCard userCard = VideoFilmLibrary.GetUserCardById(userId);

if (userCard == null)

{

MessageBox.Show($"Картка користувача із ID {userId} не знайдена.", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

List<Record> userRecords = userCard.GetRecords();

if (userRecords.Count == 0)

{

MessageBox.Show($"Користувач із ID {userId} ще не переглядав фільми.", "Інформація", MessageBoxButton.OK, MessageBoxImage.Information);

return;

}

List<VideoFilm> watchedFilms = new List<VideoFilm>();

foreach (Record record in userRecords)

{

VideoFilm film = VideoFilmLibrary.GetFilmById(record.VideoFilmId);

if (film != null)

{

watchedFilms.Add(film);

}

}

if (watchedFilms.Count == 0)

{

MessageBox.Show($"Користувач із ID {userId} ще не переглядав фільми.", "Інформація", MessageBoxButton.OK, MessageBoxImage.Information);

return;

}

PrintMessage("Фільми, які переглядав користувач:");

foreach (VideoFilm film in watchedFilms)

{

if (film is VideoFilmOnCassette videoFilmOnCassette)

{

PrintMessage(videoFilmOnCassette.ToStringForWrite());

}

else if (film is VideoFilmOnDisc videoFilmOnDisc)

{

PrintMessage(videoFilmOnDisc.ToStringForWrite());

}

}

}

catch (FormatException)

{

MessageBox.Show("Некоректний формат введених даних.", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

catch (Exception ex)

{

MessageBox.Show($"Сталася помилка: {ex.Message}", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void OfferListOfSnacksByFilmID\_Click(object sender, RoutedEventArgs e)

{

try

{

int filmId = int.Parse(featuresFilmID.Text);

VideoFilm film = VideoFilmLibrary.GetFilmById(filmId);

if (film == null)

{

MessageBox.Show($"Фільм із ID {filmId} не знайдено.", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

List<string> snacksList = GetSnacksForGenre(film.Genre);

if (snacksList.Count == 0)

{

MessageBox.Show($"Для жанру {film.Genre} не знайдено варіантів снеків.", "Інформація", MessageBoxButton.OK, MessageBoxImage.Information);

return;

}

Random random = new Random();

List<string> selectedSnacks = snacksList.OrderBy(\_ => random.Next()).Take(2).ToList();

PrintMessage($"Для фільму '{film.Title}' підходять наступні снеки:");

foreach (string snack in selectedSnacks)

{

PrintMessage(snack);

}

}

catch (FormatException)

{

MessageBox.Show("Некоректний формат введених даних.", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

catch (Exception ex)

{

MessageBox.Show($"Сталася помилка: {ex.Message}", "Помилка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private List<string> GetSnacksForGenre(Genres genre)

{

List<string> snacksList = new List<string>();

switch (genre)

{

case Genres.Action:

snacksList.AddRange(new List<string>{

"Попкорн з карамеллю або солоний.",

"Спайсі чіпси.",

"Курятина або свинина в гострій паніровці.",

"Гострі крильця.",

"Газовані напої."});

break;

case Genres.Drama:

snacksList.AddRange(new List<string>{

"Сир і виноград.",

"Фруктовий салат.",

"Горішки.",

"Сирні палички.",

"Темний шоколад."});

break;

case Genres.Comedy:

snacksList.AddRange(new List<string>{

"Попкорн з смаковими порошками (сир, бекон, сіль).",

"Газовані напої.",

"Кольорові желейні цукерки.",

"Мафіни або кекси.",

"Зефірки."});

break;

case Genres.Thriller:

snacksList.AddRange(new List<string>{

"Сухарики.",

"Крекери з кетчупом.",

"Оливки.",

"Гострі сирні шарики.",

"Темний шоколад."});

break;

case Genres.Fantasy:

snacksList.AddRange(new List<string>{

"Маршмеллоу, драже, желе.",

"Кекси.",

"Фруктові коктейлі.",

"Морозиво з феєричними топінгами.",

"Кольорові напої."});

break;

case Genres.Romance:

snacksList.AddRange(new List<string>{

"Фруктові шашлики.",

"Шоколадні фонтани з фруктами.",

"Малиновий чи трюфельний мус.",

"Пудроване печиво.",

"Рожеві коктейлі."});

break;

case Genres.Animation:

snacksList.AddRange(new List<string>{

"Кольорові фруктові палички.",

"Мармуровані мафіни або кейкпопси.",

"Масляний попкорн.",

"Кольорові желейні цукерки.",

"Морозиво з веселими топінгами."});

break;

case Genres.Documentary:

snacksList.AddRange(new List<string>{

"Сухофрукти та горішки.",

"Сир.",

"Крекери з тунцем або лососем.",

"Солоний попкорн.",

"Нежирні йогурти."});

break;

default:

break;

}

return snacksList;

}

}

}

namespace Video\_library\_owner\_handbook

{

internal class FileManager

{

public static void WriteFilmsToFile()

{

File.WriteAllText("FilmsOnCassette.txt", string.Empty);

File.WriteAllText("FilmsOnDisc.txt", string.Empty);

foreach (var item in VideoFilmLibrary.GetVideoFilms())

{

if (item is VideoFilmOnCassette)

{

VideoFilmOnCassette videoFilm = (VideoFilmOnCassette)item;

File.AppendAllText("FilmsOnCassette.txt", videoFilm.ToStringForFile() + Environment.NewLine);

}

else if (item is VideoFilmOnDisc)

{

VideoFilmOnDisc videoFilm = (VideoFilmOnDisc)item;

File.AppendAllText("FilmsOnDisc.txt", videoFilm.ToStringForFile() + Environment.NewLine);

}

}

}

public static void ReadFilmsFromFile()

{

string[] lines = File.ReadAllLines("FilmsOnCassette.txt");

foreach (string line in lines)

{

string[] elements = line.Split('~');

VideoFilmOnCassette film = new VideoFilmOnCassette(int.Parse(elements[0]),

bool.Parse(elements[1]),

elements[2],

elements[3],

elements[4],

int.Parse(elements[5]),

elements[6],

elements[7],

elements[8],

double.Parse(elements[9]),

elements[10],

int.Parse(elements[11]),

int.Parse(elements[12]));

VideoFilmLibrary.AddFilm(film);

}

string[] lines1 = File.ReadAllLines("FilmsOnDisc.txt");

foreach (string line in lines1)

{

string[] elements = line.Split('~');

VideoFilmOnDisc film = new VideoFilmOnDisc(int.Parse(elements[0]),

bool.Parse(elements[1]),

elements[2],

elements[3],

elements[4],

int.Parse(elements[5]),

elements[6],

elements[7],

elements[8],

double.Parse(elements[9]),

elements[10]);

VideoFilmLibrary.AddFilm(film);

}

}

public static void WriteUserCardsToFile()

{

File.WriteAllText("UserCards.txt", string.Empty);

foreach (var userCard in VideoFilmLibrary.GetUserCards())

{

File.AppendAllText("UserCards.txt", userCard.ToStringForFile() + Environment.NewLine);

}

}

public static void ReadUserCardsFromFile()

{

string[] lines = File.ReadAllLines("UserCards.txt");

foreach (string line in lines)

{

string[] elements = line.Split('~');

UserCard userCard = new UserCard(int.Parse(elements[0]), elements[1]);

if (elements.Length > 2)

for (int i = 2; i < elements.Length; i += 5)

{

userCard.AddRecordToUserCard(new Record(int.Parse(elements[i]),

int.Parse(elements[i + 1]),

DateTime.Parse(elements[i + 2]),

DateTime.Parse(elements[i + 3]),

bool.Parse(elements[i + 4])));

}

VideoFilmLibrary.AddUserCard(userCard);

}

}

}

}

namespace Video\_library\_owner\_handbook

{

public enum Genres

{

Action,

Drama,

Comedy,

Thriller,

Fantasy,

Romance,

Animation,

Documentary

}

public enum CassetteTypes

{

VHS,

Betamax,

Video8,

Hi8

}

public enum DiscTypes

{

CD,

DVD,

BlueRay,

VCD

}

}

namespace Video\_library\_owner\_handbook

{

internal interface IStringConvertible

{

string ToStringForFile();

string ToStringForWrite();

}

}

namespace Video\_library\_owner\_handbook

{

internal class Record : IStringConvertible

{

private int recordId { get; set; }

private int videoFilmId { get; set; }

private DateTime takingDate { get; set; }

private DateTime returningDate { get; set; }

private bool wasReturned { get; set; }

public int RecordId

{

get { return recordId; }

set { recordId = value; }

}

public int VideoFilmId

{

get { return videoFilmId; }

set { videoFilmId = value; }

}

public DateTime TakingDate

{

get { return takingDate; }

set { takingDate = value; }

}

public DateTime ReturningDate

{

get { return returningDate; }

set { returningDate = value; }

}

public bool WasReturned

{

get { return wasReturned; }

set { wasReturned = value; }

}

public Record(int recordId, int videoFilmId, DateTime takingDate, DateTime returningDate, bool wasReturned)

{

this.recordId = recordId;

this.videoFilmId = videoFilmId;

this.takingDate = takingDate;

this.returningDate = returningDate;

this.wasReturned = wasReturned;

}

~Record()

{

Console.WriteLine("Деструктор для класу \"Record\" було викликано.");

}

public string ToStringForFile()

{

return "~" +

$"{this.recordId}~" +

$"{this.videoFilmId}~" +

$"{this.takingDate}~" +

$"{this.returningDate}~" +

$"{this.wasReturned}";

}

public string ToStringForWrite()

{

return

$"ID записа: {this.recordId}\n" +

$"ID фільму: {this.videoFilmId}\n" +

$"Дата взяття: {this.takingDate}\n" +

$"Дата повернення: {this.returningDate}\n" +

$"Факт повернення: " +

(this.wasReturned ? "повернено\n" : "не повернено\n");

}

}

}

namespace Video\_library\_owner\_handbook

{

internal class UserCard : IStringConvertible

{

private int id;

private string userName;

private List<Record> records { get; set; }

public int Id

{

get { return id; }

set { id = value; }

}

public string UserName

{

get { return userName; }

set { userName = value; }

}

public List<Record> Records

{

get { return records; }

set { records = value; }

}

public UserCard(int id, string userName)

{

this.id = id;

this.userName = userName;

this.records = new List<Record>();

}

public int FindMinUnusedRecordID()

{

if (records.Count == 0)

return 1;

List<int> sortedIds = records

.OrderBy(obj => obj.RecordId)

.Select(obj => obj.RecordId)

.ToList();

for (int i = 1; i < sortedIds.Max(); i++)

if (!sortedIds.Contains(i))

return i;

return sortedIds.Max() + 1;

}

public void AddRecordToUserCard(Record record)

{

records.Add(record);

}

~UserCard()

{

Console.WriteLine("Деструктор для класу \"UserCard\" було викликано.");

}

public List<Record> GetRecords()

{

return records;

}

public string ToStringForFile()

{

string result = $"{this.id}~" +

$"{this.userName}";

foreach (Record record in records)

{

result += record.ToStringForFile();

}

return result;

}

public string ToStringForWrite()

{

return

$"ID: {this.id}\n" +

$"Ім'я клієнта: {this.userName}\n";

}

}

}

namespace Video\_library\_owner\_handbook

{

internal abstract class VideoFilm

{

protected int id;

protected bool inVideoLibrary;

protected string title;

protected string studio;

protected Genres genre;

protected int releaseYear;

protected string director;

protected string actors;

protected string summary;

protected double rating;

public int Id

{

get { return id; }

set { id = value; }

}

public bool InVideoLibrary

{

get { return inVideoLibrary; }

set { inVideoLibrary = value; }

}

public string Title

{

get { return title; }

set { title = value; }

}

public string Studio

{

get { return studio; }

set { studio = value; }

}

public Genres Genre

{

get { return genre; }

set { genre = value; }

}

public int ReleaseYear

{

get { return releaseYear; }

set { releaseYear = value; }

}

public string Director

{

get { return director; }

set { director = value; }

}

public string Actors

{

get { return actors; }

set { actors = value; }

}

public string Summary

{

get { return summary; }

set { summary = value; }

}

public double Rating

{

get { return rating; }

set { rating = value; }

}

public VideoFilm(int id, bool inVideoLibrary, string title, string studio, string genreString, int releaseYear,

string director, string actors, string summary, double rating)

{

this.id = id;

this.inVideoLibrary = inVideoLibrary;

this.title = title;

this.studio = studio;

if (Enum.TryParse(genreString, out Genres genre))

{

this.genre = genre;

}

this.releaseYear = releaseYear;

this.director = director;

this.actors = actors;

this.summary = summary;

this.rating = rating;

}

~VideoFilm()

{

Console.WriteLine("Деструктор для класу \"VideoFilm\" було викликано.");

}

}

}

namespace Video\_library\_owner\_handbook

{

internal class VideoFilmLibrary

{

private static List<VideoFilm> videoFilms = new List<VideoFilm>();

private static List<UserCard> userCards = new List<UserCard>();

public VideoFilmLibrary()

{

videoFilms = new List<VideoFilm>();

userCards = new List<UserCard>();

}

~VideoFilmLibrary()

{

Console.WriteLine("Деструктор для класу \"VideoFilmLibrary\" було викликано.");

}

public static List<VideoFilm> GetVideoFilms()

{

return videoFilms;

}

public static List<UserCard> GetUserCards()

{

return userCards;

}

public static void AddFilm(VideoFilm videoFilm)

{

VideoFilmLibrary.videoFilms.Add(videoFilm);

}

public static void AddUserCard(UserCard userCard)

{

userCards.Add(userCard);

}

public static int FindMinUnusedUserID()

{

if (VideoFilmLibrary.userCards.Count == 0)

return 1;

List<int> sortedIds = VideoFilmLibrary.userCards

.OrderBy(obj => obj.Id)

.Select(obj => obj.Id)

.ToList();

for (int i = 1; i < sortedIds.Max(); i++)

if (!sortedIds.Contains(i))

return i;

return sortedIds.Max() + 1;

}

public static int FindMinUnusedFilmID()

{

if (VideoFilmLibrary.videoFilms.Count == 0)

return 1;

List<int> sortedIds = VideoFilmLibrary.videoFilms

.OrderBy(obj => obj.Id)

.Select(obj => obj.Id)

.ToList();

for (int i = 1; i < sortedIds.Max(); i++)

if (!sortedIds.Contains(i))

return i;

return sortedIds.Max() + 1;

}

public static string RemoveFilmById(int filmId)

{

VideoFilm filmToRemove = videoFilms.FirstOrDefault(film => film.Id == filmId);

if (filmToRemove != null)

{

videoFilms.Remove(filmToRemove);

return ($"Фільм з ідентифікатором {filmId} був успішно видалений.\n\n");

}

else

{

return ($"Ідентифікатор {filmId} не знайдено в бібліотеці.\n\n");

}

}

public static string RemoveUserCardById(int userId)

{

UserCard userCardToRemove = userCards.FirstOrDefault(user => user.Id == userId);

if (userCardToRemove != null)

{

userCards.Remove(userCardToRemove);

return ($"Користувач з ідентифікатором {userId} був успішно видалений.\n\n");

}

else

{

return ($"Ідентифікатор {userId} не знайдено в бібліотеці.\n\n");

}

}

public static VideoFilm GetFilmById(int filmId)

{

return videoFilms.FirstOrDefault(film => film.Id == filmId);

}

public static UserCard GetUserCardById(int userId)

{

return userCards.FirstOrDefault(userCard => userCard.Id == userId);

}

}

}

namespace Video\_library\_owner\_handbook

{

internal class VideoFilmOnCassette : VideoFilm, IStringConvertible

{

private CassetteTypes cassetteType;

private int cassetteSeriesReleaseYear;

private int cassetteNumberInSeries;

public CassetteTypes CassetteType

{

get { return cassetteType; }

set { cassetteType = value; }

}

public int CassetteSeriesReleaseYear

{

get { return cassetteSeriesReleaseYear; }

set { cassetteSeriesReleaseYear = value; }

}

public int CassetteNumberInSeries

{

get { return cassetteNumberInSeries; }

set { cassetteNumberInSeries = value; }

}

public VideoFilmOnCassette(int id, bool inVideoLibrary, string title, string studio, string genreString, int releaseYear,

string director, string actors, string summary, double rating,

string cassetteTypeString, int cassetteSeriesReleaseYear,

int cassetteNumberInSeries)

: base(id, inVideoLibrary, title, studio, genreString, releaseYear, director, actors, summary, rating)

{

if (Enum.TryParse(cassetteTypeString, out CassetteTypes cassetteType))

{

this.cassetteType = cassetteType;

}

this.cassetteSeriesReleaseYear = cassetteSeriesReleaseYear;

this.cassetteNumberInSeries = cassetteNumberInSeries;

}

~VideoFilmOnCassette()

{

Console.WriteLine("Деструктор для класу \"VideoFilmOnCassette\" було викликано.");

}

public string ToStringForFile()

{

return

$"{this.id}~" +

$"{this.inVideoLibrary}~" +

$"{this.title}~" +

$"{this.studio}~" +

$"{this.genre}~" +

$"{this.releaseYear}~" +

$"{this.director}~" +

$"{this.actors}~" +

$"{this.summary}~" +

$"{this.rating}~" +

$"{this.cassetteType}~" +

$"{this.cassetteSeriesReleaseYear}~" +

$"{this.cassetteNumberInSeries}";

}

public string ToStringForWrite()

{

return

$"ID: {this.id}\n" +

"Фільм записано на касеті\n" +

(this.inVideoLibrary ? "Фільм у відеотеці\n" : "Фільм у користувача\n") +

$"Назва: {this.title}\n" +

$"Студія: {this.studio}\n" +

$"Жанр: {this.genre}\n" +

$"Рік виходу: {this.releaseYear}\n" +

$"Режисер: {this.director}\n" +

$"Актори: {this.actors}\n" +

$"Опис: {this.summary}\n" +

$"Рейтинг: {this.rating}\n" +

$"Тип касети: {this.cassetteType}\n" +

$"Рік виходу серії касети: {this.cassetteSeriesReleaseYear}\n" +

$"Номер касети в серії: {this.cassetteNumberInSeries}\n\n";

}

}

}

namespace Video\_library\_owner\_handbook

{

internal class VideoFilmOnDisc : VideoFilm, IStringConvertible

{

private DiscTypes discType;

public DiscTypes DiscType

{

get { return discType; }

set { discType = value; }

}

public VideoFilmOnDisc(int id, bool inVideoLibrary, string title, string studio, string genreString, int releaseYear,

string director, string actors, string summary, double rating,

string discTypeString)

: base(id, inVideoLibrary, title, studio, genreString, releaseYear, director, actors, summary, rating)

{

if (Enum.TryParse(discTypeString, out DiscTypes discType))

{

this.discType = discType;

}

}

~VideoFilmOnDisc()

{

Console.WriteLine("Деструктор для класу \"VideoFilmOnDisc\" було викликано.");

}

public string ToStringForFile()

{

return

$"{this.id}~" +

$"{this.inVideoLibrary}~" +

$"{this.title}~" +

$"{this.studio}~" +

$"{this.genre}~" +

$"{this.releaseYear}~" +

$"{this.director}~" +

$"{this.actors}~" +

$"{this.summary}~" +

$"{this.rating}~" +

$"{this.discType}";

}

public string ToStringForWrite()

{

return

$"ID: {this.id}\n" +

"Фільм записано на диску\n" +

(this.inVideoLibrary ? "Фільм у відеотеці\n" : "Фільм у користувача\n") +

$"Назва: {this.title}\n" +

$"Студія: {this.studio}\n" +

$"Жанр: {this.genre}\n" +

$"Рік виходу: {this.releaseYear}\n" +

$"Режисер: {this.director}\n" +

$"Актори: {this.actors}\n" +

$"Опис: {this.summary}\n" +

$"Рейтинг: {this.rating}\n" +

$"Тип диску: {this.discType}\n\n";

}

}

}

<Window x:Class="Video\_library\_owner\_handbook.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Video\_library\_owner\_handbook"

mc:Ignorable="d"

Title="СУБД Довідник власика відеотеки" Height="700" Width="900"

ResizeMode="NoResize">

<Grid>

<TabControl Margin="0,0,0,317">

<TabItem Header="Посібник користувача">

<Grid Background="#FFE5E5E5">

<Button Click="Help\_GiveFilm\_Click" Content="Видача фільмів" HorizontalAlignment="Left" Margin="165,10,0,0" VerticalAlignment="Top" Width="133"/>

<Button Click="Help\_ReturnFilm\_Click" Content="Повернення фільмів" HorizontalAlignment="Left" Margin="165,35,0,0" VerticalAlignment="Top" Width="133"/>

<Button Click="Help\_ClientBase\_Click" Content="База клієнтів" HorizontalAlignment="Left" Margin="165,60,0,0" VerticalAlignment="Top" Width="133"/>

<Button Click="Help\_FilmsList\_Click" Content="Перегляд списку фільмів" HorizontalAlignment="Left" Margin="323,10,0,0" VerticalAlignment="Top" Width="155"/>

<Button Click="Help\_AddFilm\_Click" Content="Додавання фільму" HorizontalAlignment="Left" Margin="323,35,0,0" VerticalAlignment="Top" Width="155"/>

<Button Click="Help\_UpdateFilm\_Click" Content="Редагування фільму" HorizontalAlignment="Left" Margin="323,60,0,0" VerticalAlignment="Top" Width="155"/>

<Button Click="Help\_DeleteFilm\_Click" Content="Видалення фільму" HorizontalAlignment="Left" Margin="323,85,0,0" VerticalAlignment="Top" Width="155"/>

<Button Click="Help\_LoadFromDB\_Click" Content="Завантаження з БД" HorizontalAlignment="Left" Margin="10,35,0,0" VerticalAlignment="Top" Width="133"/>

<Button Click="Help\_SaveToDB\_Click" Content="Збереження в БД" HorizontalAlignment="Left" Margin="10,60,0,0" VerticalAlignment="Top" Width="133"/>

<Button Click="Help\_AboutProgram\_Click" Content="Про програму" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" Width="133"/>

<Button Click="Help\_Search\_Click" Content="Пошук" HorizontalAlignment="Left" Margin="504,10,0,0" VerticalAlignment="Top" Width="155"/>

<Button Click="Help\_Filter\_Click" Content="Фільтрація" HorizontalAlignment="Left" Margin="504,35,0,0" VerticalAlignment="Top" Width="155"/>

<Button Click="Help\_Sort\_Click" Content="Сортування" HorizontalAlignment="Left" Margin="504,60,0,0" VerticalAlignment="Top" Width="155"/>

<Button Click="Help\_AdditionalFeatures\_Click" Content="Додатковий функціонал" HorizontalAlignment="Left" Margin="691,10,0,0" VerticalAlignment="Top" Width="155"/>

</Grid>

</TabItem>

<TabItem Header="Видача та повернення фільмів">

<Grid Background="#FFE5E5E5">

<TabControl>

<TabItem Header="Видача">

<Grid Background="#FFE5E5E5">

<Label Content="Введіть ID фільму:" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top"/>

<TextBox x:Name="getFilmID" HorizontalAlignment="Left" Margin="10,36,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="64"/>

<Label Content="Оберіть дату взяття фільму:" HorizontalAlignment="Left" Margin="10,59,0,0" VerticalAlignment="Top"/>

<Calendar x:Name="getTakeDate" HorizontalAlignment="Left" Margin="10,90,0,0" VerticalAlignment="Top"/>

<Label Content="Оберіть дату повернення фільму:" HorizontalAlignment="Left" Margin="216,59,0,0" VerticalAlignment="Top"/>

<Calendar x:Name="getReturnDate" HorizontalAlignment="Left" Margin="216,90,0,0" VerticalAlignment="Top"/>

<Label Content="Введіть ID клієнта:" HorizontalAlignment="Left" Margin="216,10,0,0" VerticalAlignment="Top"/>

<TextBox x:Name="getClientID" HorizontalAlignment="Left" Margin="216,36,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="64"/>

<Button Content="Видати фільм" HorizontalAlignment="Left" Margin="330,34,0,0" VerticalAlignment="Top" Width="110" Click="GiveFilm\_Click"/>

</Grid>

</TabItem>

<TabItem Header="Повернення">

<Grid Background="#FFE5E5E5">

<Label Content="Введіть ID фільму:" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top"/>

<TextBox x:Name="returnFilmID" HorizontalAlignment="Left" Margin="10,36,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="64"/>

<Label Content="Введіть ID клієнта:" HorizontalAlignment="Left" Margin="216,10,0,0" VerticalAlignment="Top"/>

<TextBox x:Name="returnClientID" HorizontalAlignment="Left" Margin="216,36,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="64"/>

<Label Content="Оберіть фактичну дату повернення фільму:" HorizontalAlignment="Left" Margin="10,59,0,0" VerticalAlignment="Top"/>

<Calendar x:Name="factReturnDate" HorizontalAlignment="Left" Margin="10,90,0,0" VerticalAlignment="Top"/>

<Button Content="Повернути фільм" HorizontalAlignment="Left" Margin="330,34,0,0" VerticalAlignment="Top" Width="110" Click="ReturnFilm\_Click"/>

<Button Content="Вивести список неповернених фільмів" HorizontalAlignment="Left" Margin="10,268,0,0" VerticalAlignment="Top" Width="219" Click="GetUnreturnedFilmsByUserId"/>

</Grid>

</TabItem>

<TabItem Header="Робота з базою клієнтів">

<Grid Background="#FFE5E5E5">

<Button Content="Вивести всіх клієнтів" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" Width="140" Click="PrintAllClients\_Click"/>

<Button Content="Додати клієнта" HorizontalAlignment="Left" Margin="10,76,0,0" VerticalAlignment="Top" Width="140" Click="AddNewClient\_Click"/>

<Label Content="Введіть ім'я клієнта для додавання:" HorizontalAlignment="Left" Margin="10,101,0,0" VerticalAlignment="Top"/>

<TextBox x:Name="addClientName" HorizontalAlignment="Left" Margin="10,132,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="202"/>

<Button Content="Видалити клієнта" HorizontalAlignment="Left" Margin="242,76,0,0" VerticalAlignment="Top" Width="140" Click="DeleteClient\_Click"/>

<Label Content="Введіть ID клієнта для видалення:" HorizontalAlignment="Left" Margin="242,101,0,0" VerticalAlignment="Top"/>

<TextBox x:Name="deleteClientID" HorizontalAlignment="Left" Margin="242,132,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="202"/>

</Grid>

</TabItem>

</TabControl>

</Grid>

</TabItem>

<TabItem Header="Робота з базою фільмів">

<Grid Background="#FFE5E5E5">

<TabControl>

<TabItem Header="Виведення та додавання фільмів">

<Grid Background="#FFE5E5E5" Margin="0,0,-18,0">

<Label Content="Введіть дані фільму:" HorizontalAlignment="Left" Margin="10,5,0,0" VerticalAlignment="Top"/>

<Button Content="Вивести список всіх фільмів" HorizontalAlignment="Left" Margin="195,10,0,0" VerticalAlignment="Top" Click="WriteAllFilms\_Click"/>

<Button Content="Додати фільм до відеотеки" HorizontalAlignment="Center" Margin="0,10,0,0" VerticalAlignment="Top" Click="AddNewFilm\_Click"/>

</Grid>

</TabItem>

<TabItem Header="Оновлення та видалення фільмів">

<Grid Background="#FFE5E5E5">

<Label Content="Введіть дані фільму для оновлення:" HorizontalAlignment="Left" Margin="10,5,0,0" VerticalAlignment="Top"/>

<Label Content="Введіть ID фільму:" HorizontalAlignment="Left" Margin="221,8,0,0" VerticalAlignment="Top" RenderTransformOrigin="0.89,0.267"/>

<TextBox x:Name="CRUD\_filmID" HorizontalAlignment="Left" Margin="334,12,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="39"/>

<Button Content="Оновити дані" HorizontalAlignment="Left" Margin="406,11,0,0" VerticalAlignment="Top" RenderTransformOrigin="0.464,-0.144" Width="100" Click="UpdateFilm\_Click"/>

<Button Content="Видалити фільм" HorizontalAlignment="Left" Margin="511,11,0,0" VerticalAlignment="Top" RenderTransformOrigin="0.464,-0.144" Width="100" Click="DeleteFilm\_Click"/>

</Grid>

</TabItem>

</TabControl>

<Label Content="" HorizontalAlignment="Left" Margin="10,35,0,0" VerticalAlignment="Top"/>

<Label Content="Оберіть тип носія:" HorizontalAlignment="Left" Margin="10,66,0,0" VerticalAlignment="Top"/>

<RadioButton x:Name="CRUD\_isCassette" IsChecked="True" Content="Касета" HorizontalAlignment="Left" Margin="10,97,0,0" VerticalAlignment="Top"/>

<RadioButton x:Name="CRUD\_isDisc" Content="Диск" HorizontalAlignment="Left" Margin="10,117,0,0" VerticalAlignment="Top"/>

<Label Content="Введіть назву фільму:" HorizontalAlignment="Left" Margin="10,137,0,0" VerticalAlignment="Top"/>

<TextBox x:Name="CRUD\_title" HorizontalAlignment="Left" Margin="10,163,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="157"/>

<Label Content="Введіть назву студії:" HorizontalAlignment="Left" Margin="10,186,0,0" VerticalAlignment="Top"/>

<TextBox x:Name="CRUD\_studio" HorizontalAlignment="Left" Margin="10,212,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="157"/>

<Label Content="Оберіть жанр:" HorizontalAlignment="Left" Margin="10,235,0,0" VerticalAlignment="Top"/>

<ComboBox x:Name="CRUD\_genre" HorizontalAlignment="Left" Margin="10,261,0,0" VerticalAlignment="Top" Width="127">

<ComboBoxItem Content="Action" />

<ComboBoxItem Content="Drama" />

<ComboBoxItem Content="Comedy" />

<ComboBoxItem Content="Thriller" />

<ComboBoxItem Content="Fantasy" />

<ComboBoxItem Content="Romance" />

<ComboBoxItem Content="Animation" />

<ComboBoxItem Content="Documentary" />

</ComboBox>

<Label Content="Введіть рік виходу фільму:" HorizontalAlignment="Left" Margin="193,66,0,0" VerticalAlignment="Top"/>

<TextBox x:Name="CRUD\_releaseYear" HorizontalAlignment="Left" Margin="193,92,0,0" TextWrapping="Wrap" Text="" VerticalAlignment="Top" Width="77"/>

<Label Content="Введіть режисера:" HorizontalAlignment="Left" Margin="193,115,0,0" VerticalAlignment="Top"/>

<TextBox x:Name="CRUD\_director" HorizontalAlignment="Left" Margin="193,141,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="154"/>

<Label Content="Введіть список акторів:" HorizontalAlignment="Left" Margin="193,164,0,0" VerticalAlignment="Top"/>

<TextBox x:Name="CRUD\_actors" HorizontalAlignment="Left" Margin="193,190,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="154" Height="93"/>

<Label Content="Введіть короткий опис фільму:" HorizontalAlignment="Left" Margin="377,66,0,0" VerticalAlignment="Top"/>

<TextBox x:Name="CRUD\_summary" HorizontalAlignment="Left" Margin="377,92,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="179" Height="89"/>

<Label Content="Введіть рейтинг (від 1 до 5):" HorizontalAlignment="Left" Margin="377,186,0,0" VerticalAlignment="Top"/>

<TextBox x:Name="CRUD\_rating" HorizontalAlignment="Left" Margin="377,212,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="89"/>

<Label Content="Оберіть тип касети:" HorizontalAlignment="Left" Margin="588,66,0,0" VerticalAlignment="Top"/>

<ComboBox x:Name="CRUD\_cassetteType" HorizontalAlignment="Left" Margin="588,90,0,0" VerticalAlignment="Top" Width="127">

<ComboBoxItem Content="VHS" />

<ComboBoxItem Content="Betamax" />

<ComboBoxItem Content="Video8" />

<ComboBoxItem Content="Hi8" />

</ComboBox>

<Label Content="Введіть рік випуску касети:" HorizontalAlignment="Left" Margin="588,117,0,0" VerticalAlignment="Top"/>

<TextBox x:Name="CRUD\_cassetteYear" HorizontalAlignment="Left" Margin="588,143,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="89"/>

<Label Content="Введіть серійний номер касети:" HorizontalAlignment="Left" Margin="588,164,0,0" VerticalAlignment="Top"/>

<TextBox x:Name="CRUD\_cassetteSerialNumber" HorizontalAlignment="Left" Margin="588,190,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="89"/>

<Label Content="Оберіть тип диску:" HorizontalAlignment="Left" Margin="588,215,0,0" VerticalAlignment="Top"/>

<ComboBox x:Name="CRUD\_discType" HorizontalAlignment="Left" Margin="588,239,0,0" VerticalAlignment="Top" Width="127">

<ComboBoxItem Content="CD" />

<ComboBoxItem Content="DVD" />

<ComboBoxItem Content="BlueRay" />

<ComboBoxItem Content="VCD" />

</ComboBox>

</Grid>

</TabItem>

<TabItem Header="Пошук, сортування та фільтри">

<Grid Background="#FFE5E5E5">

<Label Content="Пошук:" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top"/>

<TextBox x:Name="searchField" HorizontalAlignment="Left" Margin="10,41,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="189"/>

<Button Content="Шукати" HorizontalAlignment="Left" Margin="10,64,0,0" VerticalAlignment="Top" Width="94" Click="Search\_Click"/>

<Label Content="Фільтри:" HorizontalAlignment="Left" Margin="295,10,0,0" VerticalAlignment="Top"/>

<ComboBox x:Name="mediaTypeFilter" HorizontalAlignment="Left" Margin="295,41,0,0" VerticalAlignment="Top" Width="120">

<ComboBoxItem Content="Тип носія" />

<ComboBoxItem Content="Касета" />

<ComboBoxItem Content="Диск" />

</ComboBox>

<ComboBox x:Name="genreFilter" HorizontalAlignment="Left" Margin="295,122,0,0" VerticalAlignment="Top" Width="120">

<ComboBoxItem Content="Жанр" />

<ComboBoxItem Content="Action" />

<ComboBoxItem Content="Drama" />

<ComboBoxItem Content="Comedy" />

<ComboBoxItem Content="Thriller" />

<ComboBoxItem Content="Fantasy" />

<ComboBoxItem Content="Romance" />

<ComboBoxItem Content="Animation" />

<ComboBoxItem Content="Documentary" />

</ComboBox>

<ComboBox x:Name="cassetteTypeFilter" HorizontalAlignment="Left" Margin="295,68,0,0" VerticalAlignment="Top" Width="120">

<ComboBoxItem Content="Тип касети" />

<ComboBoxItem Content="VHS" />

<ComboBoxItem Content="Betamax" />

<ComboBoxItem Content="Video8" />

<ComboBoxItem Content="Hi8" />

</ComboBox>

<ComboBox x:Name="discTypeFilter" HorizontalAlignment="Left" Margin="295,95,0,0" VerticalAlignment="Top" Width="120">

<ComboBoxItem Content="Тип диску" />

<ComboBoxItem Content="CD" />

<ComboBoxItem Content="DVD" />

<ComboBoxItem Content="BlueRay" />

<ComboBoxItem Content="VCD" />

</ComboBox>

<Label Content="Рік виходу:" HorizontalAlignment="Left" Margin="295,149,0,0" VerticalAlignment="Top"/>

<TextBox x:Name="yearFilter" HorizontalAlignment="Left" Margin="295,175,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>

<Button Content="Фільтрувати" HorizontalAlignment="Left" Margin="295,198,0,0" VerticalAlignment="Top" Width="94" Click="Filter\_Click"/>

<Label Content="Сортування:" HorizontalAlignment="Left" Margin="561,10,0,0" VerticalAlignment="Top"/>

<Label Content="Фільтри:" HorizontalAlignment="Left" Margin="295,10,0,0" VerticalAlignment="Top"/>

<ComboBox x:Name="sortType" HorizontalAlignment="Left" Margin="561,41,0,0" VerticalAlignment="Top" Width="145">

<ComboBoxItem Content="Тип сортування" />

<ComboBoxItem Content="За зростанням" />

<ComboBoxItem Content="За спаданням" />

</ComboBox>

<ComboBox x:Name="sortCriterion" HorizontalAlignment="Left" Margin="561,68,0,0" VerticalAlignment="Top" Width="145">

<ComboBoxItem Content="Критерій сортування" />

<ComboBoxItem Content="Назва" />

<ComboBoxItem Content="Рік виходу" />

<ComboBoxItem Content="Рейтинг" />

</ComboBox>

<Button Content="Сортувати" HorizontalAlignment="Left" Margin="561,95,0,0" VerticalAlignment="Top" Width="94" Click="Sort\_Click"/>

</Grid>

</TabItem>

<TabItem Header="Додатковий функціонал">

<Grid Background="#FFE5E5E5">

<Button Content="Улюблений жанр користувача" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" Width="321" Click="FavoriteGenreByUserID\_Click"/>

<Button Content="Випадковий непереглянутий користувачем фільм" HorizontalAlignment="Left" Margin="10,35,0,0" VerticalAlignment="Top" Width="321" Click="RandomNotViewedFilmByUserID\_Click"/>

<Button Content="Топ-3 найпереглядуваніші фільми в відеотеці" HorizontalAlignment="Left" Margin="10,60,0,0" VerticalAlignment="Top" Width="321" Click="Top3MostWatchedFilmsInLibrary\_Click"/>

<Button Content="Середній термін повернення фільмів користувачем" HorizontalAlignment="Left" Margin="10,85,0,0" VerticalAlignment="Top" Width="321" Click="AverageFilmReturningTimeByUserID\_Click"/>

<Button Content="Запропонувати перелік снеків до фільму" HorizontalAlignment="Left" Margin="10,135,0,0" VerticalAlignment="Top" Width="321" Click="OfferListOfSnacksByFilmID\_Click"/>

<Button Content="Список переглянутих користувачем фільмів" HorizontalAlignment="Left" Margin="10,110,0,0" VerticalAlignment="Top" Width="322" Click="AllWatchedFilmsByUserID"/>

<Label Content="ID користувача:" HorizontalAlignment="Left" Margin="447,10,0,0" VerticalAlignment="Top"/>

<TextBox x:Name="featuresUserID" HorizontalAlignment="Left" Margin="548,14,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>

<Label Content="ID фільму:" HorizontalAlignment="Left" Margin="478,41,0,0" VerticalAlignment="Top"/>

<TextBox x:Name="featuresFilmID" HorizontalAlignment="Left" Margin="548,45,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>

</Grid>

</TabItem>

</TabControl>

<ScrollViewer VerticalScrollBarVisibility="Auto" Margin="0,367,0,0">

<TextBlock x:Name="outputTextBlock" TextWrapping="Wrap" Margin="10, 0" Text=""/>

</ScrollViewer>

</Grid>

</Window>