1. Які конкретні задачі планували вирішувати за допомогою цієї бібліотеки?

   У цьому проекті основною задачею було створення користувацького інтерфейсу (UI) для додатку, який зберігає велику кількість інформації про медіа, зокрема книги, ігри та комікси. Інтерфейс мав надати користувачеві можливість ефективно керувати цією базою даних, включаючи перегляд, додавання, редагування та видалення елементів для усіх користувачів системи. Крім того, додаток повинен дозволяти користувачам додавати медіа до своїх власних списків, зберігаючи при цьому персональні налаштування, такі як статус (прочитано, переглянуто, пройдено тощо) або оцінка. Цей підхід дозволяє створити інтерактивне середовище, де кожен користувач може відстежувати свою взаємодію з медіа-контентом, а також мати доступ до загальної бази даних, що надає зручний інтерфейс для пошуку та фільтрації медіа за різними критеріями.

2. Чому було обрано саме цю бібліотеку, а не аналоги?

   Windows Forms було обрано як найбільш простий і зручний фреймворк для створення графічного інтерфейсу для додатків на платформі Windows. Це класичний інструмент для створення настільних додатків, що дозволяє швидко створити простий інтерфейс. Крім того, я вже мав досвід роботи з цим фреймфорком, що значно спростило процес розробки.

3. Наскільки просто та зрозуміло було отримати, встановити, налаштувати та почати використовувати цю бібліотеку?

   Оскільки Windows Forms є частиною стандартного набору фрейморків для .NET, установка та налаштування були дуже простими. Інтерфейс і елементи керування легко налаштовуються, що значно спрощує процес використання.

4. Наскільки зрозумілою та корисною була документація бібліотеки?

   Документація Windows Forms була достатньо зрозумілою. Корисною ж для мене вона була у випадку базових завдань. Вона містила описи класів, методів і подій, а також приклади використання. Проте для деяких складніших аспектів розробки мені було набагато легше розібратися завдяки використанню додаткових ресурсів, наприклад відеоуроків на YouTube та форумів (в моєму випадку Stack Overflow), де надаються пояснення та приклади практичного застосування. Такі джерела часто пропонували швидші і зрозуміліші рішення, ніж офіційна документація. Окрім цього, я знаходив готові фрагменти коду та рекомендації для вирішення конкретних задач, таких як обробка подій або налаштування специфічних компонентів, що значно економило час. Також я активно використовував ChatGPT для швидкого ознайомлення з проблемою та пошуку оптимальних способів її вирішення. Це дозволяло оперативно отримувати відповіді на вузькопрофільні питання та уникати довготривалого пошуку інформації в інтернеті.

5. Наскільки було зрозуміло, як саме використовувати бібліотеку, які класи/методи/функції використовувати для вирішення поставлених задач?

   Як вже було сказано у пункті 4, використовувати фреймворк для вирішення базових задач було нескладно. Стандартна документація, хоч і була англійською, містила достатньо інформації для ознайомлення з базовими класами, такими як **Form**, **Button**, **TextBox**, та методами на зразок **Click** чи **Add()**. Це дозволяло легко зрозуміти, як реалізувати основні функції додатку, включаючи створення та налаштування елементів інтерфейсу. Однак через моє недостатнє володіння англійською мовою могли виникати труднощі з розумінням більш складних розділів документації. У таких випадках я звертався до зовнішніх джерел, як-от відеоуроки на YouTube, форуми та ChatGPT. Ці ресурси надавали простіші та локалізовані пояснення, що значно полегшувало роботу з фреймворком та допомагало швидше розібратися з використанням специфічних класів і методів. Таким чином, навіть попри потенційний мовний бар'єр, всі необхідні завдання вдалося успішно реалізувати

6. Наскільки зручно було використовувати бібліотеку, чи не треба було писати багато надлишкового коду?

   На початку роботи та ознайомлення з фреймворку дійсно були моменти написання надлишкового коду. Це траплялося через недостатнє розуміння того, як оптимально використовувати певні класи чи методи. Однак, з часом, коли я набув більше досвіду в роботі з фреймворком та краще зрозумів принципи її роботи, вдалося позбутися зайвого коду. Windows Forms забезпечує зручний інструментарій для створення користувацького інтерфейсу з мінімальною кількістю написаного коду, особливо завдяки дизайнеру у Visual Studio. Наприклад, функції автоматичної прив’язки даних до елементів керування або використання подій значно спрощували розробку. Таким чином, після подолання початкових труднощів, використання фреймворку стало зручним і ефективним.

7. Наскільки зрозумілою була поведінка класів/методів/функцій з бібліотеки?

   Як вже було сказано у пункті 4, поведінка класів, методів та функцій Windows Forms була доволі зрозумілою для вирішення базових задач. Стандартні класи, такі як **Form**, **Button**, **TextBox**, працювали передбачувано, а документація та додаткові ресурси допомагали швидко освоїти їхню роботу. Для складніших випадків я використовував зовнішні джерела або ChatGPT, що дозволяло уникати непорозумінь та отримувати чітке уявлення про поведінку компонентів.

8. Наскільки зрозумілою була взаємодія між різними класами/методами/функціями цієї бібліотеки, а також взаємодія між бібліотекою та власним кодом?

   Як вже було сказано у пункті 4, взаємодія між класами та методами фреймворка, а також інтеграція їх з власним кодом, була досить зрозумілою. Windows Forms пропонує просту модель роботи, де різні компоненти легко взаємодіють між собою, наприклад, через події та властивості. Використання таких механізмів, як події **Click** для кнопок, забезпечувало передбачуваний результат. Завдяки зрозумілому дизайну фреймоврку та підтримці з боку додаткових ресурсів робота з взаємодією класів була ефективною.

9. Чи виникали якісь проблеми з використанням бібліотеки? Чи вдалось їх вирішити, як саме?

   Найбільша проблема, яка мені запам’яталася, стосувалася того, що елементи інтерфейсу не змінювали свої розміри динамічно відносно вікна. Це ускладнювало роботу зі створення адаптивного дизайну. Проте вдалося знайти рішення: я використав **TableLayoutPanel** для гнучкого розташування елементів і написав спеціальні **resize**-функції для тих елементів, які вимагали більш тонкого налаштування. Це дозволило забезпечити належну адаптивність інтерфейсу.

10. Що хорошого можна сказати про цю бібліотеку, які були позитивні аспекти використання бібліотеки?

- **Зручність та зрозумілість**: Це один із найпростіших фреймворків для створення настільних додатків, особливо для початківців.
- **Швидкий старт**: Навіть із базовими знаннями можна швидко створити робочий додаток.
- **Гнучкість у налаштуванні**: Є можливість вручну налаштовувати майже всі аспекти елементів керування, що додає універсальності.
- **Інтеграція із Visual Studio**: Інтуїтивний дизайнер інтерфейсу значно прискорює розробку.

11. Що поганого можна сказати про цю бібліотеку, які були негативні аспекти використання бібліотеки?

- **Обмежена платформа**: Windows Forms створює додатки, які підтримуються лише на платформі Windows. Це обмежує можливість кросплатформеного використання, особливо якщо потрібно розробляти додатки для macOS або Linux.
- **Моральна застарілість**: Хоча фреймворк ще підтримується, вона вважається застарілою порівняно з сучаснішими рішеннями, такими як WPF або MAUI, які пропонують більше можливостей для створення адаптивного дизайну.
- **Складнощі з адаптивним інтерфейсом**: Як згадувалося раніше, базові елементи не завжди автоматично підлаштовуються під зміну розмірів вікна, що вимагає додаткових зусиль для розробки.

12. Якби довелось вирішувати аналогічну задачу, але вже враховуючи досвід використання в цій лабораторній роботі, що варто було б робити так само, а що змінити? Можливо, використати інші бібліотеки, чи використати інші можливості цієї бібліотеки, чи інакше організувати код, чи ще щось?

Якщо б я поставив перед собою цю ж задачу, то цього разу я б не використовував Windows Forms, оскільки, незважаючи на зручність, мені не дуже сподобалося працювати з цим фреймворком. Основні причини полягають у його обмеженнях щодо сучасного дизайну, адаптивності інтерфейсу та кросплатформенності.

***<u>Що я б зробив інакше:</u>***

- **Перехід на веб-додаток**: Я б написав веб-додаток замість настільного. Це надало б можливість створити сучасний, адаптивний інтерфейс, який працюватиме на будь-якому пристрої з браузером.
- **Фронтенд**: Використав би **React**, який забезпечує гнучкість у створенні динамічних інтерфейсів, зручний компонентний підхід і великий вибір готових бібліотек для різних потреб, наприклад, **Material-UI** для стилізації.
- **Бекенд**: Для серверної частини я б використав: **Node.js** з фреймворком **Express** — це чудовий вибір для побудови API, які швидко обробляють запити та добре інтегруються з фронтендом на React.

***<u>Що залишив би без змін:</u>***

- **Організація функціональності**: Я б зберіг принцип, за яким користувач має змогу переглядати, додавати, редагувати, видаляти елементи, а також керувати своїми персональними списками.
- **Фокус на зручність інтерфейсу**: Незалежно від фреймворка, важливо створити інтуїтивно зрозумілий і легкий у використанні UI.

## Переваги такого підходу:

- **Кросплатформенність**: Додаток працюватиме на будь-якому пристрої.
- **Сучасний інтерфейс**: Використання React дозволяє легко реалізувати адаптивний та привабливий дизайн.
- **Масштабованість**: Легко додати нові функції чи адаптувати систему до зростаючих потреб.
- **Велика спільнота**: Як React, так і Node.js мають активні спільноти, що забезпечує доступ до великої кількості готових рішень та документації.