МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Таврійський державний агротехнологічний університет
імені Дмитра Моторного
Факультет енергетики і комп'ютерних технологій

ЗАТВЕРДЖУЮ
Зав. каф. "Комп'ютерні науки"
доц._____ Сергій ШАРОВ
"17" _лютого_ 2025 р.

Пояснювальна записка

до кваліфікаційної роботи здобувача СВО Магістр
(ступінь вищої освіти)

на тему: «Інтелектуальна система для формування рекомендацій користувачам бібліотеки»

15КНД.11323055.000000ПЗ

Виконав: здобувач вищої освіти 2 курсу, групи 21МБКН
спеціальності 122 Комп'ютерні науки
за ОПП Комп'ютерні науки
(шифр і назва спеціальності та ОПП)

_____ Коржилов Борис
(підпис)

Керівник_____ Юрій Сіциліцин
(підпис)

Нормоконтроль, ст.викл._____ Ольга ЗІНОВ'ЄВА
(підпис)

Рецензент_____ Альона ЧОРНА
(підпис)

Запоріжжя – 2025 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Таврійський державний агротехнологічний університет імені Дмитра Моторного

Факультет: Енергетики і комп'ютерних технологій
Кафедра: Комп'ютерні науки
Ступінь вищої освіти: Магістр
Спеціальність: 122 Комп'ютерні науки
ОПП: Комп'ютерні науки_____

ЗАТВЕРДЖУЮ
Завідувач кафедри_КН
к.пед.н., доц. _____ Сергій ШАРОВ
"04" _жовтня__2024__ року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

__Коржилов Борис Миколайович_____
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи__«Інтелектуальна система для формування рекомендацій користувачам бібліотеки»
Науковий керівник роботи __Сіциліцин Юрій Олександрович, доцент_
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом університету від "23" вересня_2024_року №_442-С__
2. Строк подання здобувачем вищої освіти роботи __17 лютого 2025_____
3. Вихідні дані до проекту: дані обстеження об'єкту, статистичні дані, технічне завдання на дипломне проєктування, матеріали виробничих практик, нормативні документи, науково-технічна література, електронні ресурси та ін.
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)__
___1. Опис предметної області та аналіз існуючих рішень._____
___2. Специфікація вимог до експертної системи._____
___3. Вибір та обґрунтування технологій розробки експертної системи.___
___4. Дослідна експлуатація експертної системи._____
___5. Тестування експертної системи._____
5. Перелік графічного матеріалу
Презентація – 10 слайдів:

1. Титульний слайд
2. Предмет, об’єкт, мета, завдання дослідження
3. Порівняльна характеристика аналогів програмного продукту
4. Діаграма варіантів використання
5. Інструментальні засоби
6. Інтерфейс і робота експертної системи
7. Інтерфейс і робота експертної системи
8. Тестування
9. Висновок

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання____04 вересня 2024 р._

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Аналіз предметної області	12.12.2024	виконано
2	Специфікація вимог до експертної системи	13.12.2024	виконано
3	Проектування експертної системи	16.12.2024	виконано
4	Обґрунтування інструментальних засобів	20.12.2024	виконано
5	Розробка експертної системи	30.01.2024	виконано
6	Тестування та налагодження експертної системи	15.01.2025	виконано
7	Підпис керівником проекту	16.02.2025	виконано
8	Підпис завідувачем кафедри	17.02.2025	виконано

Здобувач вищої освіти _____ __Коржилов_Борис__
(підпис) (власне ім’я та прізвище)

Керівник кваліфікаційної роботи _____ ___Юрій Сіциліцин___
(підпис) (власне ім’я та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра: 74 с., 15 рис., 24 посилань, 1 таблиця, 3 додатки.

Актуальність: Автоматизація бібліотек стає необхідністю, і програма керування з ШІ-асистентом значно підвищує ефективність. Вона спрощує облік і інвентаризацію, мінімізує помилки, оптимізує пошук літератури та надає персоналізовані рекомендації. ШІ аналізує популярність книг, прогнозує попит і допомагає оновлювати фонд, а також генерує звіти для прийняття управлінських рішень. Інтеграція з цифровими ресурсами забезпечує доступ до електронних книг та архівів. У результаті така система покращує обслуговування користувачів і робить управління бібліотекою сучасним та ефективним.

Об’єкт дослідження: процес управління бібліотечним фондом, зокрема методи автоматизації обліку, організації та оптимізації роботи бібліотеки за допомогою інформаційних технологій і штучного інтелекту.

Предмет дослідження: інтелектуальна система з можливістю формування рекомендацій користувачам бібліотеки, її функціональні можливості та вплив на ефективність управління бібліотечним фондом.

Мета роботи: створення програмного забезпечення для керування бібліотекою, з вбудованим штучним інтелектом у ролі асистента.

Задачі дослідження**:**

– вивчити та проаналізувати стан системи управління складом бібліотеки з вбудованим ШІ-асистентом;

– провести аналіз характеристик та сучасних програмних систем для автоматизації управління бібліотечним фондом;

– розробити функціональні вимоги до програмного забезпечення для моніторингу та управління складом бібліотеки з вбудованим ШІ-асистентом;

– розробити алгоритм функціонування програмного забезпечення для автоматизації управління бібліотечним фондом.

Практичне значення дослідження: полягатиме у впровадженні програмного забезпечення для автоматизації управління бібліотекою з вбудованим ШІ-асистентом, що забезпечить ефективне управління фондом та покращення взаємодії з користувачами.

Ключові слова: ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, КЕРУВАННЯ СКЛАДОМ, БІБЛІОТЕЧНИЙ ФОНД, ШІ-АСИСТЕНТ, АВТОМАТИЗАЦІЯ.

SUMMARY

Master's qualification work: 74 pages, 15 figures, 24 references, 1 table, 3 applications.

Relevance: Library automation is becoming a necessity, and a management program with an AI assistant significantly increases efficiency. It simplifies accounting and inventory, minimizes errors, optimizes literature search, and provides personalized recommendations. AI analyzes book popularity, forecasts demand, and helps update the collection while also generating reports for managerial decision-making. Integration with digital resources ensures access to electronic books and archives. As a result, such a system improves user service and makes library management modern and efficient.

Object of research: the process of managing the library collection, in particular, methods of automating accounting, organization, and optimization of library operations using information technology and artificial intelligence.

Subject of research: an intelligent system with the capability to generate recommendations for library users, its functional capabilities, and its impact on the efficiency of library collection management.

Purpose of the work: to create software for library management with embedded artificial intelligence as an assistant.

Research tasks:

– study and analyze the state of the library inventory management system with an embedded AI assistant;

– conduct an analysis of the characteristics and modern software systems for automating library collection management;

– develop functional requirements for software to monitor and manage the library inventory with an embedded AI assistant;

– develop an algorithm for the functioning of software for automating library collection management.

Practical significance of the research: will consist of implementing software for automating library management with an embedded AI assistant, which will ensure effective collection management and improved user interaction.

Keywords: SOFTWARE, INVENTORY MANAGEMENT, LIBRARY COLLECTION, AI ASSISTANT, AUTOMATION.

ЗМІСТ

ВСТУП

В умовах швидкого розвитку інформаційних технологій та зростання вимог до ефективності управління бібліотечними фондами, автоматизація процесів бібліотек стає важливою складовою модернізації цього сектору. Традиційні методи обліку та пошуку літератури, що базуються на ручному введенні та обробці даних, вже не відповідають сучасним потребам користувачів, які очікують швидкого та зручного доступу до інформації. Саме тому впровадження інтелектуальних систем є перспективним напрямом розвитку бібліотек, що дозволяє значно підвищити якість обслуговування, скоротити час на пошук необхідної літератури та зробити процес управління фондом більш ефективним.

Програма для керування бібліотекою з вбудованим ШІ-асистентом є сучасним інструментом, що дозволяє не тільки автоматизувати облік літератури, а й покращити взаємодію з користувачами. Інтелектуальна система аналізує історію читання, вподобання та пошукові запити користувачів, щоб формувати персоналізовані рекомендації. Це значно спрощує вибір книг, особливо для нових відвідувачів, які ще не знайомі з бібліотечним фондом.

Окрім цього, використання штучного інтелекту в бібліотечній сфері відкриває можливості для розширення функціоналу системи, зокрема інтеграції з цифровими архівами, онлайн-каталогами та іншими інформаційними ресурсами. Це сприяє забезпеченню користувачів актуальною інформацією та доступом до ширшого спектра літератури, включаючи електронні книги та рідкісні видання.

Використання інтелектуальної системи також допомагає бібліотекарям у швидкому внесенні змін до бази даних, спрощує облік нових надходжень і автоматизує частину адміністративної роботи. Такий підхід дозволяє не тільки підвищити ефективність роботи бібліотеки, а й зменшити навантаження на персонал. Таким чином, розробка та впровадження інтелектуальної системи для бібліотек є важливим кроком у напрямі цифровізації цієї сфери. Вона дає

змогу зробити процес управління фондом більш зручним, забезпечити швидкий доступ до потрібної літератури та покращити обслуговування користувачів, що відповідає сучасним викликам інформаційного суспільства.

Об’єкт дослідження – процес автоматизації програмного забезпечення управління бібліотечним фондом.

Предмет дослідження – програмне забезпечення для автоматизації управління бібліотечним фондом з вбудованим ШІ-асистентом, його функціональні можливості та вплив на ефективність бібліотечної діяльності.

Метою роботи є розробка та обґрунтування ефективності програми для керування бібліотекою з вбудованим ШІ-асистентом, що дозволяє автоматизувати облік, оптимізувати управління фондом і покращити взаємодію з користувачами.

Завдання дослідження включають: аналіз існуючих рішень у галузі автоматизації управління бібліотечними фондами, розробку функціональних вимог до програмного забезпечення, вибір технологій для розробки, тестування програмного продукту, а також оцінку його ефективності в реальних умовах.

Методи дослідження - У процесі виконання роботи використовувалися теоретичні методи, такі як аналіз літератури та існуючих програмних рішень, порівняння технологій автоматизації бібліотечних процесів. Практичними методами є розробка програмного забезпечення, тестування системи, а також інтеграція ШІ-асистента для покращення функціональності програмного продукту.

Наукова новизна роботи полягає в розробці нової концепції програмного забезпечення для автоматизації бібліотечних процесів з використанням штучного інтелекту, що дозволяє надавати персоналізовані рекомендації читачам і прогнозувати попит на літературу.

Практична значущість полягає в тому, що розроблений продукт дозволяє значно підвищити ефективність управління бібліотечними фондами,

автоматизувати облік та інвентаризацію, а також покращити обслуговування користувачів.

Робота складається з чотирьох розділів:

– у першому розділі «Опис предметної області та аналіз існуючих рішень» аналізуються основні проблеми управління складом бібліотеки та існуючі програмні рішення;

– другий розділ «Проектування програмного забезпечення» присвячений розробці концептуальної моделі системи, опису її підсистем та специфікації вимог;

– третій розділ «Вибір та обґрунтування технологій розробки програмного забезпечення» містить обґрунтування вибору технологій для розробки програмного продукту, включаючи вибір мови програмування та фреймворків;

– четвертий розділ «Дослідна експлуатація та тестування програмного забезпечення» охоплює етапи тестування, впровадження та оцінки ефективності розробленого продукту.

РОЗДІЛ 1
ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ ТА АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Узагальнена характеристика предметної області

Інформатизація різних сфер життя суспільства впливає як на культурні процеси всередині країни, так і на її міжнародний імідж. Рівень інформаційної культури як окремої особистості, так і держави загалом змінюється під впливом цих процесів. Постійне оновлення професійних підходів та світогляду фахівців зумовлене швидкими змінами в економічному та політичному середовищі. Ці трансформації охоплюють усі аспекти життя, а роль інновацій, новітніх технологій та творчих підходів до розвитку лише зростає.

Інформаційні технології відіграють важливу роль у сучасному суспільстві, зокрема у сфері бібліотечної справи. Автоматизація бібліотек сприяє підвищенню ефективності їхньої роботи, зручності для користувачів і збереженню бібліотечного фонду. Використання інформаційних технологій дає змогу швидко обробляти великі обсяги даних, зменшувати витрати на управління бібліотечними фондами та покращувати рівень обслуговування користувачів. Оцифровування книг, автоматизація обліку та інтелектуальні пошукові системи значно полегшують роботу бібліотекарів і підвищують зручність для читачів. Сучасні бібліотеки вже давно перетворилися з традиційних книгозбірень на інформаційні центри, які забезпечують доступ до знань у цифровому форматі. Швидкий розвиток технологій інформаційного суспільства суттєво змінив підхід до бібліотечної справи в усьому світі. Сьогодні головною місією бібліотек будь-якого типу є забезпечення вільного доступу до національних та міжнародних інформаційних ресурсів, оскільки вони відіграють ключову роль у соціально-економічному розвитку держави.

З поширенням сучасних інформаційних технологій у суспільстві бібліотечна сфера отримала новий напрям — розробку та впровадження ІТ-

рішень. Це зумовлено необхідністю адаптації бібліотек до сучасного рівня інформаційної взаємодії, де поєднуються технології роботи з документами, знаннями та інформацією. Використання новітніх цифрових інструментів у бібліотечній сфері також відбувається на тлі пришвидшення процесів поділу праці в інформаційному просторі.

Автоматизація та технологізація бібліотечних процесів стають дедалі важливішими через кілька ключових факторів: постійне зростання обсягів інформації, її швидке застарівання, питання авторства та боротьби з плагіатом, забезпечення вільного доступу до знань, а також збереження цінної інформації, що може мати національну чи державну значущість. Одним із ключових напрямів розвитку є впровадження штучного інтелекту (ШІ) для оптимізації роботи бібліотек. Використання інтелектуальних систем допомагає автоматизувати процеси пошуку, класифікації та рекомендацій книг. Зокрема, рекомендаційні системи, засновані на алгоритмах машинного навчання, дозволяють читачам швидко знаходити літературу, що відповідає їхнім вподобанням, історії читання та інтересам. Одним із перспективних напрямів розвитку є використання інтелектуальних систем для покращення обслуговування відвідувачів бібліотек.

Застосування ШІ у бібліотечних системах має низку переваг:

- персоналізація – системи можуть аналізувати вподобання користувачів та пропонувати відповідні книги;
- автоматизація обліку – забезпечення швидкого та точного обліку бібліотечного фонду;
- підвищення ефективності пошуку – швидке знаходження потрібної літератури за різними параметрами (автор, жанр, рік видання тощо);
- оптимізація управління фондами – аналіз популярності книг і прогнозування попиту на нові надходження.

Попри значні переваги, впровадження ШІ у бібліотечні системи має певні виклики:

– високі витрати на впровадження – розробка та підтримка інтелектуальних систем вимагає фінансових та технічних ресурсів.

– необхідність якісних даних – для ефективної роботи системи необхідно забезпечити коректну та повну базу даних книг.

– питання конфіденційності – використання персональних даних користувачів для персоналізованих рекомендацій потребує дотримання вимог безпеки та конфіденційності.

Однак, навіть з усіма своїми перевагами, інтеграція інтелектуальних систем у бібліотеки має певні виклики. Основними перешкодами є високі витрати на впровадження таких рішень, що вимагає значних фінансових і технічних ресурсів. Також важливим фактором є наявність великої кількості якісних даних для належної роботи інтелектуальних систем. Бібліотечні системи повинні мати доступ до актуальної і правильно організованої інформації про книги, авторів та користувачів. Однією з головних проблем є забезпечення конфіденційності користувацьких даних, оскільки персональні рекомендації ґрунтуються на обробці інформації про інтереси та вподобання кожного користувача. Не слід збувати про необхідність володіння бібліотекарями певною інформаційною культурою, щоб вони могли працювати з інформаційними ресурсами та програмними засобами.

Попри ці виклики, розвиток інформаційних технологій у бібліотечній справі продовжує розвиватися, пропонуючи нові можливості для удосконалення процесів обслуговування користувачів.

Українські бібліотеки можуть частково подолати ці виклики, зокрема через співпрацю з фахівцями зі штучного інтелекту або обмін досвідом із бібліотеками, які вже впровадили такі технології. Бібліотекарям важливо визначити напрями, що потребують удосконалення та інвестицій, а також підготувати персонал із необхідними навичками для управління та підтримки IoT-рішень.

Сучасні ІКТ-гаджети варто розглядати як інструменти для покращення надання інформаційних послуг. Крім того, бібліотечні працівники мають працювати над ефективністю використання цифрових технологій, регулярно підвищувати свою кваліфікацію та проходити відповідну підготовку для інтеграції штучного інтелекту. Важливо також постійно оцінювати якість бібліотечних послуг і вчасно адаптувати їх до нових умов. Впровадження інтелектуальних систем дозволяє значно покращити не лише якість обслуговування, але й сприяє оптимізації роботи бібліотек в цілому, знижуючи навантаження на бібліотечний персонал та створюючи умови для більш ефективного використання ресурсів. Використання інтелектуальних систем сприяє оптимізації пошуку, автоматизації обліку та персоналізації рекомендацій, що значно підвищує ефективність роботи бібліотек. Незважаючи на певні виклики, впровадження ШІ у бібліотечні системи є перспективним напрямом, що дозволяє адаптувати бібліотеки до вимог сучасного інформаційного суспільства. Аналіз даних також дозволяє бібліотекам здійснювати ефективний моніторинг популярності книг, що допомагає прогнозувати попит на певні теми або жанри літератури. Цей підхід дозволяє бібліотекам планувати закупівлі нових книг та оновлення фонду, що відповідають актуальним запитам користувачів. Впровадження таких технологій також дозволяє знизити витрати на управління фондами, завдяки автоматизації процесів обліку та збереження літератури.

Таким чином, інтелектуальні системи для формування рекомендацій користувачам бібліотеки стають важливим інструментом для адаптації бібліотек до сучасних потреб інформаційного суспільства. Вони дозволяють оптимізувати процеси обслуговування, персоналізувати рекомендації та підвищувати ефективність управління бібліотечними фондами, що робить бібліотеки більш доступними та зручними для користувачів у цифрову епоху.

1.2 Аналіз існуючих рішень

У рамках аналізу було обрано чотири бібліотечні системи: Koha, Alma, Evergreen, BiblioCommons, які є найпоширенішими рішеннями в галузі автоматизації бібліотек.

Koha:

- переваги: Відкрите програмне забезпечення, велика спільнота користувачів, можливість адаптації під потреби конкретної бібліотеки;
- недоліки: Потребує серверної інфраструктури, складність інтеграції з іншими системами.

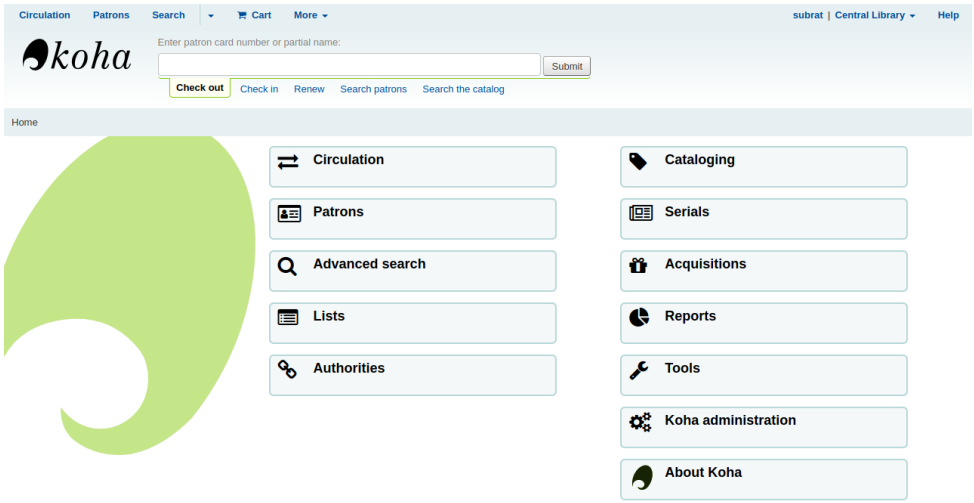

Рисунок 1 - Зовнішній вигляд koha

Alma:

- переваги: Хмарне рішення, що дозволяє централізоване управління бібліотечними ресурсами та інтеграцію з іншими бібліотечними платформами;
- недоліки: Висока вартість ліцензії, залежність від інтернет-з'єднання, що може бути критичним у випадку відсутності доступу до мережі.

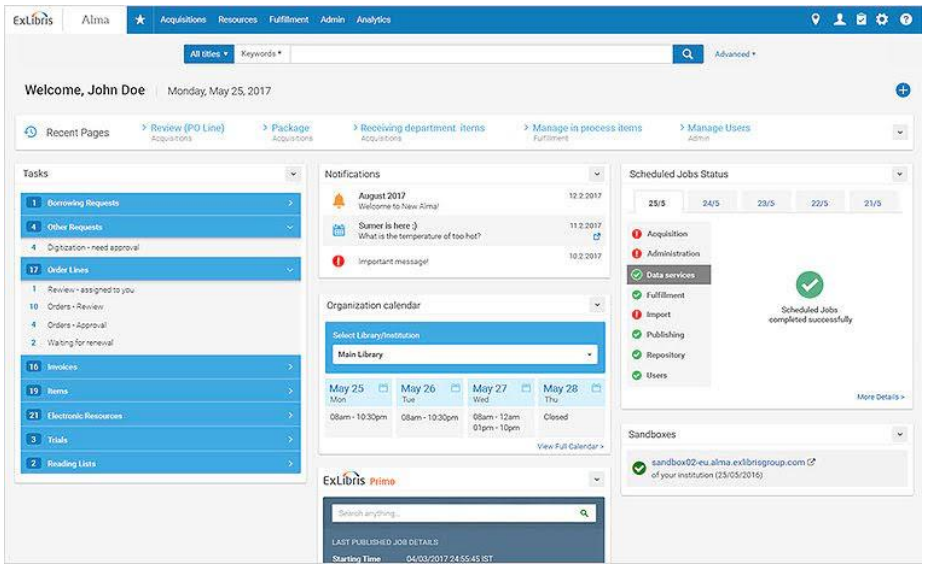

Рисунок 2 - Зовнішній вигляд Alma

Evergreen:

– переваги: Відкрите ПЗ, орієнтоване на великі бібліотечні мережі, підтримка розподіленої каталогізації;

– недоліки: Вимагає значних технічних ресурсів для встановлення та адміністрування.

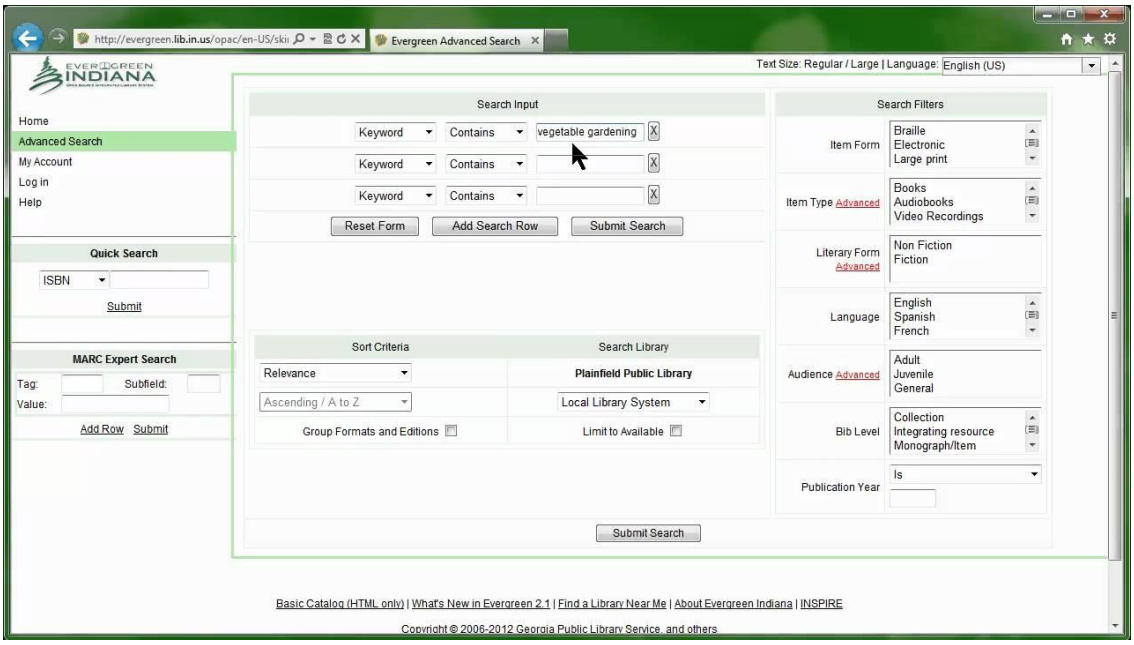

Рисунок 3 - Зовнішній вигляд Evergreen

BiblioCommons:

– переваги: Зручний користувацький інтерфейс, розширені функції соціальної взаємодії між читачами;

– недоліки: Ліцензійна модель передбачає постійні витрати, робота залежить від серверної інфраструктури розробника.

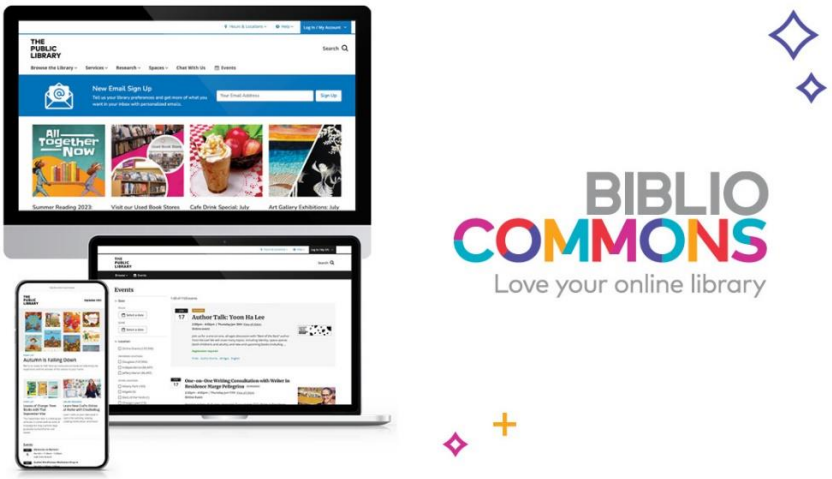

Рисунок 4 - Зовнішній вигляд BiblioCommons

У результаті можна підсумувати, що такі програми як Koha, Alma, Evergreen та BiblioCommons, виконують важливі функції з управління бібліотечними фондами, забезпечуючи автоматизацію обліку книг, пошук літератури, управління користувачами та інтеграцію з іншими цифровими ресурсами. Кожна з цих систем має свої переваги, зокрема гнучкість налаштувань, інтеграцію з каталогами та підтримку різноманітних бібліотечних процесів. Проте жодна з них не має вбудованого ШІ-асистента, який міг би надавати персоналізовані рекомендації користувачам, прогнозувати попит на книги чи відповідати на питання про літературу. Впровадження штучного інтелекту в ці системи могло б значно покращити взаємодію з користувачами та оптимізувати роботу бібліотек.

1.3 Технічне завдання на розробку ІС

Головними завданнями, які вирішує інтелектуальна система для бібліотек, є:

- автоматизація процесів керування бібліотекою – зменшення потреби у ручному введенні даних, пришвидшення операцій додавання, редагування, видалення книг;
- ефективний пошук книг – надання можливості швидко знаходити необхідну літературу за різними параметрами: назвою, жанром, роком видання, країною, автором, місцем зберігання;
- персоналізовані рекомендації – впровадження ШІ-асистента, що аналізує вподобання користувачів і пропонує книги, які можуть їх зацікавити;
- локальне збереження даних – забезпечення роботи без підключення до інтернету шляхом зберігання всієї інформації в локальному файлі TXT;
- зручний інтерфейс – спрощення взаємодії користувачів із системою за допомогою інтуїтивно зрозумілого графічного інтерфейсу, створеного за допомогою Qt.

Для ефективного функціонування системи необхідно виконати наступні вимоги:

- зручність використання – простий і зрозумілий інтерфейс, що дозволяє користувачам без труднощів додавати, редагувати, видаляти книги та взаємодіяти із ШІ-асистентом;
- швидкість обробки запитів – система повинна оперативно виконувати пошук і обробку запитів користувачів;
- надійність збереження даних – внесені зміни повинні бути збережені після натискання кнопки "Зберегти зміни", щоб уникнути втрати інформації;
- локальна робота без підключення до мережі – система має працювати автономно, використовуючи локальний файл для зберігання бібліотечних записів;
- гнучкість пошуку – можливість знаходження книг за різними критеріями для швидкого доступу до необхідної літератури.

Розробка інтелектуальної системи для бібліотек дозволяє покращити якість обслуговування читачів, підвищити ефективність роботи бібліотекарів і спростити процес пошуку та вибору книг.

Запропонована інтелектуальна система управління бібліотекою повинна забезпечувати повноцінний функціонал для автоматизованого керування каталогом, пошуку, збереження та рекомендацій книг.

Основна вимога – можливість автономної роботи з локальним збереженням даних.

Основні задачі проектування:

Інтерфейс користувача

– ергономічний та інтуїтивно зрозумілий дизайн;
– панель для відображення каталогу книг;
– поля для фільтрації та пошуку видань;
– функціональні кнопки для виконання основних операцій.

Обробка та збереження даних

– локальне збереження даних;
– надійні механізми читання та запису даних.

Пошукова система

– реалізація пошуку за ключовими параметрами (назва, жанр, автор, рік, країна, місце зберігання);
– оперативне оновлення та відображення результатів пошуку.

Інтеграція штучного інтелекту

– чат-бот для взаємодії з користувачем;
– реалізація алгоритмів рекомендаційної системи.

Вирішення технічних питань

– використання C++ та Qt для створення продуктивного інтерфейсу;
– оптимізація файлових операцій для швидкого обміну даними;
– використання методів обробки природної мови для аналізу запитів;
– гарантування стабільної роботи у режимі офлайн.

Таким чином, запропонована система стане ефективним рішенням для невеликих бібліотек, приватних колекцій або навчальних закладів, поєднуючи автоматизацію та можливості штучного інтелекту в локальному середовищі.

Вона дозволить суттєво спростити процес управління бібліотечним фондом, скоротити час, необхідний для пошуку та обробки інформації, а також підвищити зручність для користувачів. Завдяки автономному режиму роботи система не залежить від інтернет-з'єднання, що робить її придатною для використання у віддалених бібліотеках або установах з обмеженим доступом до мережі.

Штучний інтелект у системі не лише надає рекомендації щодо вибору книг, а й допомагає користувачам знаходити потрібну літературу, відповідаючи на запити у зручному форматі чату. Це дозволяє значно розширити функціональність традиційного бібліотечного пошуку, забезпечуючи персоналізований підхід до кожного читача. Інтерактивний ШІ-асистент може аналізувати запити користувачів та надавати релевантні відповіді, допомагаючи орієнтуватися у великому каталозі книг.

Крім того, система підтримує швидку обробку та оновлення бази даних, що забезпечує актуальність інформації про наявність літератури. Оптимізовані файлові операції гарантують надійне збереження даних навіть у випадку неочікуваних збоїв або завершення роботи програми. Завдяки використанню Qt створено зручний і сучасний інтерфейс, який забезпечує швидкий доступ до всіх основних функцій. У результаті розроблена система є універсальним інструментом для управління бібліотечним фондом, який поєднує простоту використання, автономність і можливості штучного інтелекту.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ІС

2.1 Концептуальна модель використання ІС

Перелік основних скорочень та використовуваних термінів в нашої предметної області представлений у таблиці 1.

Термін	Опис терміну
Фреймворк	програмні продукти, які спрощують створення і підтримку технічно складних або навантажених проектів.
UML (Unified Modeling Language)	мова для об'єктно-орієнтованого моделювання програмних систем, що дозволяє описувати структуру та поведінку системи за допомогою діаграм.
Qt	фреймворк для розробки графічних інтерфейсів, який використовується для створення зручного і інтуїтивно зрозумілого інтерфейсу користувача в програмі.
Visual Studio	інтегроване середовище розробки програмних продуктів компанії Microsoft.
Програмування	наукова і практична діяльність по створення програм.
Інтерфейс користувача (UI)	частина програми, з якою взаємодіє користувач, включаючи елементи управління та відображення даних.

Користувач	особа, яка використовує систему для пошуку книг, взаємодії з ШІ-асистентом та іншими функціями.
Локальне збереження	збереження даних на локальному пристрої без доступу до інтернету.
ШІ-асистент	програмний компонент, заснований на технології штучного інтелекту, який надає рекомендації або відповідає на питання користувача.
Програмне забезпечення	сукупність програм системи оброблення інформації та програмних документів, необхідних для експлуатації цих програм.
Автоматизація	є одним з напрямів науково-технічного прогресу, який спрямовано на застосування саморегульованих технічних засобів, економіко-математичних методів і систем керування, що звільняють людину від участі у процесах.
База даних	програмний або програмно-апаратний комплекс, розроблений для зберігання великого обсягу різноманітної інформації.
Інтегрована середа розробки	це комплекс програмного забезпечення, яке містить все необхідне для розробки, компіляції, лінкінга і налагодження коду.

Програма	набір інструкцій, які вказують комп'ютеру, що йому потрібно робити.
Концептуальна модель	абстрактне представлення основних елементів системи та їх взаємозв'язків на високому рівні.
Адміністратор	користувач, який має повні права для керування системою чи програмою.
Система	сукупність взаємодіючих компонентів, що виконують певну функцію або завдання.
ID (ідентифікатор)	унікальний код або номер, що використовується для ідентифікації об'єкта.
Портативність	здатність програми працювати на різних платформах без змін у коді.
ІС (інформаційна система)	організована сукупність ресурсів для збору, обробки та збереження даних.
C++	потужна мова програмування для розробки високопродуктивних та системних програм.
Масив структур	колекція елементів, де кожен елемент є структурою з визначеними полями.
Головний модуль	основна частина програми, що ініціалізує та керує виконанням.
Текстовий файл	файл, що містить дані у вигляді простого тексту без форматування.

Штучний інтелект (ШІ)	технологія, що дозволяє комп'ютерам виконувати завдання, що вимагають людського інтелекту.

Таблиця 1 – Глосарій

Концептуальна модель інтелектуальної бібліотечної системи (ІБС) визначає основні елементи, їхні взаємозв’язки, процеси взаємодії та управління інформаційними потоками у межах предметної області. Вона включає перелік ключових понять, їхні властивості, класифікацію та закони функціонування системи в умовах автономної роботи.

До основних об’єктів, що формують концептуальну модель ІБС, належать:

– книга – інформаційний ресурс, що має унікальні атрибути (назва, автор, рік видання, жанр, країна походження, місце зберігання);

– користувач – особа, що взаємодіє із системою для пошуку, перегляду чи отримання рекомендацій щодо книг;

– адміністратор – особа, відповідальна за наповнення бази даних, управління бібліотечним фондом та підтримку системи;

– ШІ-асистент – вбудований алгоритм, який аналізує запити користувачів та надає персоналізовані рекомендації.

Основні процеси в системі включають:

– додавання, редагування та видалення книг – адміністративна функція для підтримки актуальності бази даних;

– пошук книг – функціонал, що забезпечує вибірку книг за різними критеріями (назва, рік, жанр, автор, країна, місце зберігання);

– рекомендаційний механізм – система аналізу інтересів користувача на основі його пошукових запитів та переглянутих книг;

– збереження змін – запис оновлених даних у локальний текстовий файл, що забезпечує автономність системи.

Система працює в умовах локального використання, без необхідності підключення до Інтернету. Взаємодія з користувачами відбувається через графічний інтерфейс, реалізований на базі C++ та Qt. Дані зберігаються у текстовому файлі, що забезпечує простоту експлуатації та низькі вимоги до апаратного забезпечення.

Основною метою розробки є створення ефективного автономного бібліотечного помічника, який:

- автоматизує облік бібліотечного фонду;
- оптимізує процес пошуку необхідної літератури;
- підвищує рівень персоналізації рекомендацій за допомогою ШІ;
- мінімізує потребу в ручному керуванні бібліотечними процесами.

Інтерфейс дозволяє користувачам виконувати основні операції без необхідності спеціальних навичок, що робить систему зручною для використання в невеликих бібліотеках або особистих книжкових колекціях.

Концептуальна модель інтелектуальної бібліотечної системи відображає взаємодію її ключових компонентів та основні процеси функціонування. Вона забезпечує ефективне управління бібліотечним фондом та підтримку користувачів у виборі книг шляхом автоматизації та застосування ШІ-асистента. Роботу користувача з даною інтелектуальною системою, можна візуалізувати у вигляді певної моделі, у якій будуть відображатися функціональні та структурні особливості майбутнього програмного засобу.

Універсальною мовою для об'єктно-орієнтованого моделювання є UML (Unified Modeling Language). Її розробку розпочали в середині 90-х років XX століття, спираючись на кілька існуючих об'єктно-орієнтованих методів і нотацій для опису інформаційних систем. Мова UML використовується для візуального моделювання програмних систем, що допомагає розробникам, аналітикам і замовникам краще розуміти структуру та поведінку проєкту. Вона дозволяє описувати та документувати архітектуру, аналізувати систему перед її реалізацією, а також спрощує комунікацію між членами команди.

UML уніфікує підходи до моделювання, роблячи розробку більш структурованою та зрозумілою.

На сьогодні UML є загальноприйнятим стандартом для документування процесу розробки інформаційних систем і програмного забезпечення. Актуальною версією мови є UML 2.0. До популярних інструментів для роботи з UML належать Rational Rose, Microsoft Visio, Enterprise Architect та інші.

В UML усі моделі складних систем подаються у вигляді графічних конструкцій, відомих як діаграми. Діаграма в цьому контексті — це графічне представлення набору елементів моделі у формі зв'язного графа, де вершини та ребра мають певне значення.

UML 1.5 передбачає дванадцять типів діаграм, поділених на три категорії:

- Статична структура: діаграми класів, об'єктів, компонентів, розгортання.
- Поведінкові аспекти: діаграми прецедентів, послідовності, кооперації, станів, діяльності.
- Фізичні аспекти функціонування: діаграми реалізації.

Також важливо розуміти поняття нотації, яке використовується в UML-діаграмах. Нотація — це набір символів і правил їх використання для позначення об'єктів і зв'язків між ними. Оскільки UML є графічною мовою, моделі тут не записують текстом, а створюють у вигляді схем. В UML застосовують чотири основні елементи нотації:

- Фігури (для позначення інфраструктурних вузлів).
- Лінії (для зв'язку між вузлами).
- Значки та написи.

Побудуємо діаграму варіантів використання інтелектуальної бібліотечної системи, де буде наглядно продемонстровано як користувач може взаємодіяти з системою та у якому порядку:

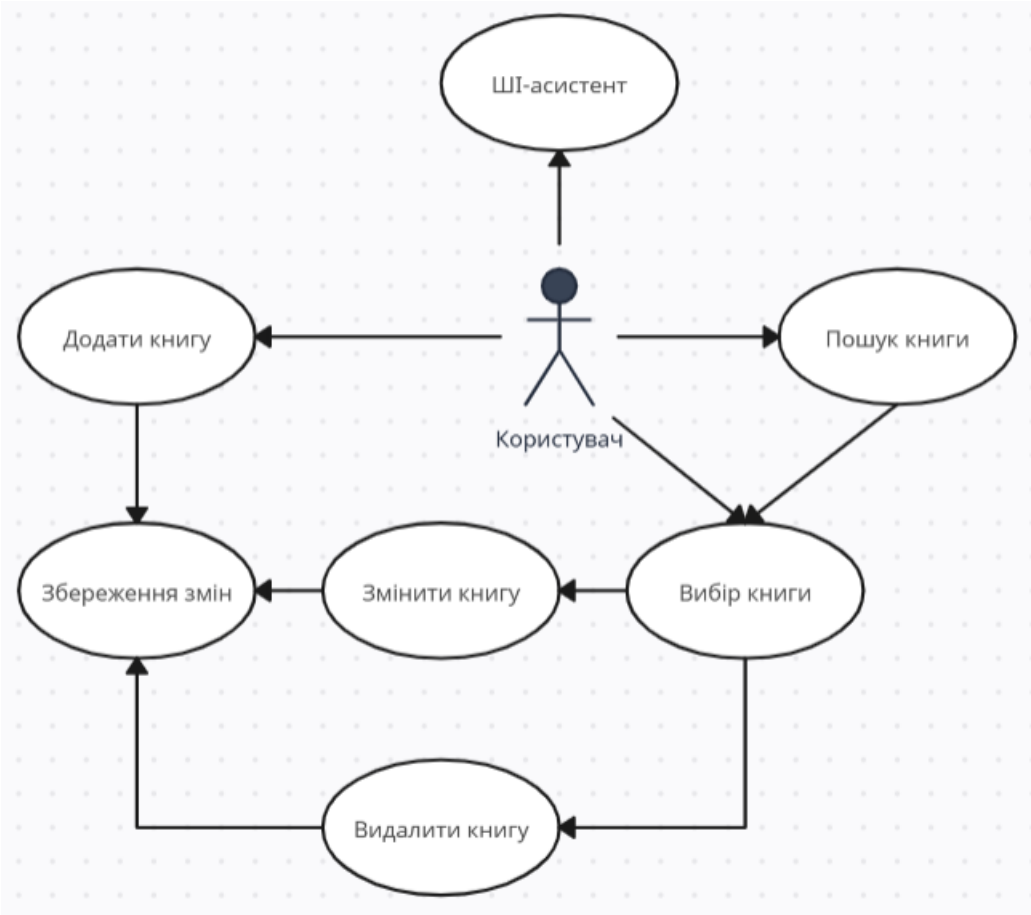

Рисунок 5 - Діаграма варіантів використання

Система програми дозволяє користувачеві здійснювати пошук книг за різними параметрами, після чого знайдену книгу можна видалити або змінити її дані, зокрема назву, жанр, рік тощо. Також є можливість додавати нові книги до списку. Усі внесені зміни потрібно зберігати, щоб вони були актуальними. Крім того, користувач може взаємодіяти з вбудованим ШІ-асистентом, який надає рекомендації щодо вибору книг.

2.2 Опис функціональних підсистем

Розроблена система управління бібліотекою є однокористувацькою, однак містить кілька функціональних модулів, які працюють у межах єдиної програми. Основні підсистеми:

Підсистема управління каталогом:

– додавання нових книг у базу даних;

– редагування існуючих записів;

– видалення книг.

Підсистема пошуку та фільтрації:

– пошук книг за назвою, жанром, автором, роком видання, раїною, місцем зберігання;

– відображення знайдених книг у списку.

Підсистема ШІ-асистента:

– чат-інтерфейс для взаємодії з користувачем;

– генерація рекомендацій книг на основі переваг користувача.

Підсистема роботи з файлами:

– збереження змін у текстовому файлі;

– завантаження списку книг під час запуску програми.

Функціональні модулі взаємопов’язані та забезпечують безперебійну роботу бібліотечної системи.

2.3 Специфікація функціональних та нефункціональних вимог

Функціональні вимоги:

– додавання нових книжок: Система повинна дозволяти користувачеві додавати нові книжки до бібліотечного складу, вказуючи наступні параметри: назва, автор, рік видання, жанр, країна, місце зберігання. У разі введення некоректних даних (наприклад, порожні поля) користувач може передивитись своє введення та змінити необхідні дані перед зберіганням;

– редагування даних книжок: Користувач має можливість редагувати дані вже наявних книжок (назва, автор, рік видання, жанр, країна, місце зберігання). При зміни будь-яких даних система повинна забезпечити механізм підтвердження змін, щоб уникнути випадкових помилок. Після зміни, користувач повинен виконати операції збереження;

– видалення книжок: Система повинна дозволяти користувачу видаляти книжки зі складу. Для запобігання випадковому видаленню, після виконання операцій користувач повинен зберегти зміни;

– пошук книжок: Користувач повинен мати можливість здійснювати пошук книжок за різними критеріями (назва, автор, жанр, рік видання, країна, місце зберігання);

– перегляд книжок: Користувач повинен мати можливість переглядати список усіх доступних книжок зі складом, а також переглядати окрему книжку з детальною інформацією;

– підтвердження змін: Після кожної операції (додавання, редагування, видалення книжок) користувач повинен здійснити функцію збереження, щоб уникнути випадкових помилок;

– інтерфейс повинен бути інтуїтивно зрозумілим і простим у використанні, дозволяти виконувати всі операції за кілька кроків. Система повинна бути здатна зберігати дані у вигляді файлів, що дозволяє зберігати всі зміни навіть після перезапуску програми.

Нефункціональні вимоги:

– продуктивність**:** Програма повинна забезпечувати швидку реакцію на будь-які операції: додавання, редагування, видалення та пошук книжок. Швидкість пошуку книжок не повинна перевищувати 2-3 секунд на великій кількості даних;

– безпека: Програма повинна гарантувати безпеку даних. Для цього важливо використовувати механізми підтвердження дій, щоб уникнути випадкових або помилкових змін;

– портативність: Програма повинна бути портативною, з можливістю запуску на різних комп’ютерах без необхідності встановлення. Для цього програма має бути компактною і займати мінімум системних ресурсів. Розмір програми не повинен перевищувати 50 МБ;

– сумісність: Програма повинна працювати на Windows 10 і новіших версіях, без потреби в додаткових драйверах або програмному забезпеченні;

– зручність використання: Інтерфейс програми повинен бути зрозумілим для користувачів з різним рівнем комп’ютерної грамотності. Всі функції програми повинні бути доступні через кілька простих кроків і мати чітке позначення.

2.4 Інформаційне забезпечення системи

Інформаційне забезпечення бібліотечної складської системи включає структуру зберігання, обробки та передачі даних між компонентами системи та користувачами. Воно охоплює взаємозв’язки між інформаційними потоками, базою даних і функціями предметної області. Програма працює з інформаційними потоками, що проходять через наступні етапи:

– введення даних: Користувач додає нові книжки або редагує наявні, вводячи інформацію про них (назва, автор, жанр, рік видання, країна, місце зберігання);

– обробка даних: Програма перевіряє коректність введених даних, обробляє їх та виконує необхідні операції (збереження, редагування, видалення);

– збереження даних: Дані записуються у файл, щоб забезпечити їхню цілісність та доступність після перезапуску програми;

– виведення інформації: Програма дозволяє користувачеві здійснювати пошук та перегляд інформації про книжки;

– ШІ-асистент: У програмі має буди вбудований ШІ-помічник, який буде шляхом розмов з користувачем, допомагати йому з підбором книг.

Система працює з наступними сутностями:

Книги**:**

– назва;

– автор;

– жанр;

– рік видання;

– країна;

– місце зберігання

Користувачі (за необхідності):

– id;

– ім’я;

– роль (адміністратор, оператор)

Фізичне середовище та вимоги:

– збереження інформації: Програма використовує файли для локального збереження даних, що забезпечує портативність;

– обмін даними: Передача інформації між компонентами системи відбувається через стандартні файли або внутрішню пам’ять;

– інструменти моделювання: Для створення інформаційних моделей можна використовувати MySQL Workbench, Microsoft Visio, Draw.io або Lucidchart.

Це забезпечує стабільну та ефективну роботу системи, мінімізує втрати даних та підвищує її зручність для користувача.

РОЗДІЛ 3
ВИБІР ТА ОБҐРУНТУВАННЯ ТЕХНОЛОГІЙ РОЗРОБКИ

3.1 Загальний огляд технологічного стеку

Для створення інтелектуальної системи управління бібліотекою, яка ефективно поєднує різноманітні функції для обробки інформації, управління бібліотечним фондом і персоналізації користувацького досвіду, було обрано кілька потужних інструментів. Ці інструменти дозволяють забезпечити високий рівень продуктивності, зручність для розробників та користувачів, а також надійність та масштабованість системи. Розглянемо докладніше основні інструменти, які були використані для реалізації системи.

Мова програмування C++:

C++ є однією з найбільш популярних мов програмування, що застосовується в розробці високопродуктивних програмних рішень. Вибір C++ для створення інтелектуальної системи управління бібліотекою обґрунтований кількома важливими факторами. По-перше, мова C++ відома своєю високою продуктивністю, що є критичним фактором для ефективної обробки великих обсягів даних, які необхідно зберігати та обробляти в бібліотечній системі. Швидкість роботи C++ дає змогу забезпечити миттєвий доступ до необхідної інформації та швидкий пошук за різними параметрами, що є важливим для зручного використання системи кінцевими користувачами.

Фреймворк Qt:

Qt є потужним та багатофункціональним інструментом для розробки кросплатформених додатків, що дозволяє створювати зручні та ефективні інтерфейси для користувачів. Однією з головних переваг Qt є його можливість забезпечувати платформонезалежність, що означає, що програма, розроблена з використанням цього фреймворку, буде працювати на різних операційних системах без необхідності значних змін у коді. Qt має великий набір вбудованих елементів інтерфейсу, таких як кнопки, текстові поля, таблиці та

інші компоненти, що значно полегшує розробку графічного інтерфейсу та прискорює процес створення додатка. Враховуючи, що інтерфейс є однією з найважливіших частин бібліотечної системи, оскільки він визначає взаємодію користувача з програмою, Qt став ідеальним вибором для реалізації простого, інтуїтивно зрозумілого та багатофункціонального інтерфейсу.

Qt надає програмісту не тільки зручний набір бібліотек класів, а й певну модель розробки додатків, певний каркас їх структури. Проходження принципам і правилам «хорошого стилю програмування на C++ / Qt» істотно знижує частоту таких важко відловлюють помилок в додатках, як витоку пам'яті (memory leaks), необроблені виключення, незакриті файли, чим нерідко страждають програми, написані «на голому C++ » без використання бібліотеки Qt. Важливою перевагою Qt є добре продуманий, логічний і стрункий набір класів, що надає програмісту дуже високий рівень абстракції. Завдяки цьому програмістам, які використовують Qt, доводиться писати значно менше коду, ніж це має місце при використанні, наприклад, бібліотеки класів MFC. Сам же код виглядає стрункішою і простіше, логічніше і зрозуміліше, ніж аналогічний за функціональністю код MFC. Його легше підтримувати і розвивати.

Крім того, навіть якщо програмісту в даний конкретний момент не потрібна кросплатформенність для його конкретного додатку (наприклад, планується версія тільки для Windows або тільки для Macintosh), ніхто не може знати, що знадобиться завтра. Плани можуть змінитися, і може виявитися і потрібним, і вигідним випустити версію для іншої операційної системи або інший апаратної платформи. У разі використання Qt для цього знадобиться всього лише перекомпіляція вихідного коду. У випадку ж використання, наприклад, MFC або «рідних» системних API знадобиться багато важкої роботи по портуванню, адаптації та налагодженні, а то й переписуванню з нуля існуючого вихідного коду для іншої ОС або апаратної платформи. Оскільки Qt можна застосовувати дуже широко, то можна написати код один раз і просто перекомпілювати його на іншій платформі, після чого він відразу виявиться працездатним.

Однією з важливих функцій Qt є підтримка інтерактивних вікон, що дозволяють створювати різноманітні діалогові вікна для введення даних, відображення результатів пошуку та інтерфейсів для управління бібліотечним фондом. Це забезпечує високу зручність користувачів і дозволяє швидко і без проблем виконувати необхідні дії, такі як додавання нових книг, редагування існуючих записів, пошук і видалення літератури.

Середовище розробки Visual Studio:

Для розробки та налагодження коду був обраний Visual Studio, яке є одним із найпопулярніших і найпотужніших середовищ для розробки програмного забезпечення.

Visual Studio надає розробникам широкий набір інструментів для написання, компіляції та налагодження коду, що робить процес розробки значно простішим і ефективнішим. Однією з основних переваг цього середовища є наявність потужного компілятора, який дозволяє забезпечити високу швидкість компіляції та ефективну роботу з великими проєктами.

Visual Studio також надає зручний інтерфейс для налагодження програм, що дає змогу швидко виявляти та виправляти помилки в коді, забезпечуючи стабільність і надійність роботи програми. Завдяки вбудованим засобам для відлагодження та підтримці багатьох мов програмування, Visual Studio є універсальним інструментом, який дозволяє ефективно розробляти програми, що працюють на різних платформах.

Крім того, Visual Studio підтримує роботу з великими проєктами, що робить його ідеальним вибором для розробки інтелектуальних систем, де необхідно керувати великою кількістю компонентів та модулів. Це дозволяє організувати роботу над проектом, що включає не тільки створення базових функцій системи, але й інтеграцію з іншими бібліотеками та інструментами для розширення можливостей програми.

3.2 Обґрунтування вибору мови програмування

Основною мовою розробки програмного засобу була обрана мова програмування C++, так як вона є компільованою статично типізованою мовою загального призначення. Вона підтримує такі парадигми програмування як процедурне програмування, об'єктно-орієнтоване програмування, узагальнене програмування, забезпечує модульність, роздільну компіляцію, обробку винятків, абстракцію даних, оголошення типів (класів) об'єктів, віртуальні функції.

C++ широко використовується для розробки програмного забезпечення, будучи однією з найбільш популярних мов програмування. Область її застосування включає створення операційних систем, різноманітних прикладних програм, драйверів пристроїв, додатків для вбудованих систем, високопродуктивних серверів, а також розважальних програмних засобів (ігор). Існує безліч реалізацій мови C++, як безкоштовних, так і комерційних і для різних платформ. Наприклад, на платформі x86 це GCC, Visual C++, Intel C++ Compiler, Embarcadero (Borland) C++ Builder та інші. C++ зробив величезний вплив на інші мови програмування, в першу чергу на Java і C#. Також мову C++ часто називають «покращеною C», так як при її розробці були згладжені деякі гострі кути мови C.

Вибір мови програмування C++ для розробки бібліотечної системи зумовлений її високою продуктивністю, гнучкістю у роботі з файлами, відсутністю необхідності додаткових залежностей, широкими можливостями створення графічного інтерфейсу та підтримкою об’єктно-орієнтованого програмування. Завдяки компіляції у машинний код C++ забезпечує швидке виконання операцій, що важливо для обробки великих обсягів даних. Його потужні засоби роботи з файлами дозволяють ефективно зчитувати, змінювати та зберігати інформацію у локальному текстовому файлі без використання серверних баз даних. Крім того, автономність C++ спрощує розгортання програми без додаткових налаштувань. Об’єктно-орієнтований підхід сприяє структурованості коду та можливості його масштабування. Завдяки цим

перевагам C++ є оптимальним вибором для реалізації поставлених завдань у межах дипломного проєкту.

Переваги C++:

– висока продуктивність - C++ забезпечує швидке виконання коду завдяки ефективному використанню пам’яті та обчислювальних ресурсів. Це особливо важливо для інтерактивних додатків, де швидкість роботи критично впливає на користувацький досвід;

– прямий контроль над пам’яттю - Завдяки ручному управлінню пам’яттю, C++ дозволяє мінімізувати використання зайвих ресурсів та оптимізувати роботу програми. Це дає можливість створювати швидкі та легкі додатки без зайвих накладних витрат;

– кросплатформність - Код на C++ може компілюватися під різні операційні системи (Windows, Linux, macOS), що робить його ідеальним вибором для розробки переносних застосунків;

– об’єктно-орієнтоване - програмування (ООП) Використання принципів ООП (класів, наслідування, поліморфізму) допомагає структурувати код, спрощувати його підтримку та повторне використання;

– можливість низькорівневого програмування - C++ дозволяє працювати як на високому рівні (зручний синтаксис, класи), так і на низькому (робота з указниками, системними ресурсами, асемблерними вставками), що робить його гнучким для будь-яких задач;

– висока стабільність і контроль над кодом - На відміну від мов з автоматичним збором сміття, C++ дає можливість уникати зайвих витрат пам’яті та забезпечує високу стабільність роботи програми;

– використання в критично важливих системах - C++ застосовується у розробці систем реального часу, фінансових додатків, комп’ютерних ігор, операційних систем, що підтверджує його надійність та ефективність;

– підтримка багатопотоковості - Завдяки можливості керування потоками та паралельним виконанням коду, C++ дозволяє створювати

ефективні, багатозадачні додатки, що є важливим для сучасних високонавантажених систем.

Недоліки C++:

– складність синтаксису - C++ має складний і багатий синтаксис, що ускладнює роботу з ним. Велика кількість можливостей, таких як шаблони, оператори перевантаження, складні механізми управління пам’яттю, може заплутати навіть досвідчених програмістів;

– ручне управління пам’яттю - Хоча прямий контроль пам’яті є перевагою, він також збільшує ризики витоків пам’яті, використання нульових або висячих покажчиків, що може призводити до важких у відлагодженні помилок;

– відсутність вбудованого збору сміття - На відміну від мов з автоматичним управлінням пам’яттю (наприклад, Python, Java), у C++ програміст сам відповідає за виділення та звільнення ресурсів, що підвищує ризик помилок і ускладнює розробку;

– тривалий час компіляції - C++ має повільніший процес компіляції порівняно з мовами інтерпретації (Python, JavaScript). Великі проекти можуть потребувати значного часу для збірки, особливо при використанні шаблонів, макросів та складних заголовкових файлів;

– відсутність стандартної бібліотеки для GUI - У C++ немає вбудованих інструментів для створення графічного інтерфейсу. Для розробки GUI потрібно використовувати сторонні бібліотеки (Qt, wxWidgets, GTK), що додає складності та залежностей до проєкту;

– менша портативність, ніж у високорівневих мовах - Хоча C++ є кросплатформною мовою, код все ж може вимагати перекомпіляції, адаптації під різні компілятори та операційні системи. У деяких випадках використання певних бібліотек може призвести до проблем із сумісністю;

– складність багатопотокового програмування - Хоча C++ підтримує багатопотоковість, її реалізація є складною та вимагає детального розуміння

потоків, синхронізації та блокувань, що може призводити до появи важких для усунення помилок;

– велика кількість мовних особливостей - Через довгу історію розвитку C++ включає в себе велику кількість застарілих можливостей, що ускладнює підтримку старого коду та створення нових проєктів відповідно до сучасних стандартів;

– висока складність відлагодження - Через особливості управління пам’яттю, вказівників і складного синтаксису відлагодження коду в C++ може бути складним та тривалим процесом, особливо у великих проєктах.

– менш зручний для швидкої розробки - C++ поступається мовам на кшталт Python у сфері швидкого прототипування та розробки. Написання навіть простої програми потребує більшої кількості коду та додаткової уваги до управління ресурсами.

3.3 Інструменти для розробки та середовище програмування

Розробка інтелектуальної бібліотечної системи здійснюється в одному з найпотужніших та найзручніших середовищ для програмування – Visual Studio 2022. Це середовище забезпечує високу стабільність роботи, високу продуктивність та надає великий набір інструментів для розробки, налагодження, тестування та профілювання коду, що робить його ідеальним для створення складних програмних рішень. Visual Studio 2022 підтримує не тільки роботу з C++, а й інтеграцію з іншими мовами програмування, що робить його універсальним інструментом для багатоплатформних проєктів.

Використання Visual Studio забезпечує ефективну організацію робочого процесу, надаючи можливість легко працювати з великими кодовими базами, здійснювати контроль версій, відлагоджувати програму в реальному часі, а також інтегрувати різноманітні інструменти для аналізу продуктивності програми. Це середовище має вбудовані інструменти для тестування коду, що

дозволяє своєчасно виявляти помилки та забезпечити високу якість програмного продукту на всіх етапах розробки.

Для створення графічного інтерфейсу використовується Qt Designer, що є частиною потужної бібліотеки Qt 6.8.2. Qt Designer дозволяє розробникам безпосередньо у графічному редакторі створювати складні інтерфейси з великою кількістю елементів управління, таких як кнопки, текстові поля, таблиці та інші компоненти, що значно прискорює процес розробки. Використання Qt Designer знижує необхідність написання великої кількості коду, дозволяючи зосередитись на логіці роботи системи та інтеграції її компонентів. Це також дозволяє розробляти інтерфейси, що забезпечують зручність використання та інтуїтивно зрозуміле управління для кінцевих користувачів.

Qt 6.8.2, крім Qt Designer, надає потужний набір інструментів для розробки кросплатформенних додатків, які можуть працювати на різних операційних системах без необхідності змінювати код. Це дозволяє досягти високої гнучкості та зручності в роботі з програмою, оскільки вона буде працювати як на Windows, так і на Linux або macOS, що важливо для ширшого кола користувачів.

Код програми написаний мовою C++, що є однією з найефективніших мов для створення продуктивних та високошвидкісних програм. Завдяки C++ система здатна ефективно працювати з великими обсягами даних, забезпечуючи швидкий доступ до інформації, що є важливим для бібліотечних систем, де необхідно оперативно знаходити, редагувати та зберігати інформацію про книги. Крім того, C++ надає можливість здійснювати точний контроль за пам’яттю, що дозволяє оптимізувати ресурси та забезпечити стабільну роботу програми при мінімальних витратах на ресурси.

Поєднання C++ і Qt дозволяє створити ефективний, швидкий, та кросплатформенний додаток, що працює без збоїв та оптимально використовує доступні ресурси. Це важливо для реалізації складних функцій, таких як обробка великих масивів даних, персоналізація рекомендацій та

інтерактивний графічний інтерфейс. Така архітектура також дозволяє забезпечити легкість масштабування додатка для роботи з більш великими бібліотечними фондами в майбутньому.

Для роботи з текстовими файлами, що використовуються як база даних для зберігання інформації про книги, авторів та інші дані, використовується стандартна бібліотека C++ – fstream. Вона забезпечує швидке та ефективне читання та записування даних у локальну базу, що є важливим для роботи системи без підключення до зовнішніх серверів. Використання цієї бібліотеки дозволяє забезпечити стабільність роботи програми, адже інформація зберігається локально, а доступ до неї здійснюється швидко та без зайвих затримок.

Таким чином, поєднання Visual Studio, Qt та C++ дозволяє створити потужну, ефективну та зручну бібліотечну систему, що оптимізує управління фондом, полегшує пошук та доступ до літератури, а також забезпечує інтуїтивно зрозумілий інтерфейс для користувачів.

РОЗДІЛ 4
ДОСЛІДНА ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ

4.1 Архітектура ІС

Розроблена інформаційна система є локальною монолітною програмою для управління бібліотекою. Вона працює без серверної частини та бази даних, зберігаючи інформацію про книги у текстовому файлі ("BOOKS.txt"). Програма використовує графічний інтерфейс, створений за допомогою Qt, і працює у віконному режимі.

Компоненти системи:

Головний модуль (MainApp)

– відповідає за ініціалізацію програми та головного вікна;

– завантажує дані про книги з файлу "BOOKS.txt" у масив структур;

– обробляє взаємодію з елементами інтерфейсу (кнопки, список книг, поля вводу).

Модуль зчитування та запису даних (файл "BOOKS.txt")

– використовує бібліотеку <fstream> для зчитування та запису інформації у текстовий файл;

– при завантаженні програми читає файл та зберігає книги у масив структур books[];

– при натисканні "Зберегти зміни" записує всі зміни у файл, повністю його перезаписуючи.

Модуль обробки даних (масив структур books[])

– всі книги зберігаються у масиві struct book books[2000];

– дозволяє виконувати пошук, редагування та видалення книг у масиві перед записом змін у файл.

Модуль управління інтерфейсом (Qt Designer + Qt Widgets)

– відповідає за відображення віконного інтерфейсу користувача;

– містить елементи GUI: список книг (QListWidget), кнопки для додавання, редагування, видалення та пошуку (QPushButton), інформаційні поля (QLabel);

– при виборі книги зі списку відображає детальну інформацію у відповідних полях.

Модуль ШІ-асистента

– використовує ONNX Runtime для локального запуску ІІ-моделі рекомендацій книг;

– інтегрований у нове вікно, яке відкривається при натисканні кнопки "ШІ-асистент".

Взаємодія компонентів:

– при запуску програми відкривається файл "BOOKS.txt", дані завантажуються у масив books[], а назви книг додаються у список booksList;

– користувач може вибрати книгу у списку, її інформація відобразиться у відповідних полях;

– при натисканні кнопки "Додати книгу" відкривається нове вікно для введення даних;

– при натисканні "Видалити" вибрана книга видаляється з масиву books[] і списку;

– пошук виконується за всіма параметрами (назва, автор, рік, жанр тощо), після чого у списку виділяється знайдений результат;

– натискання "Зберегти зміни" оновлює текстовий файл "BOOKS.txt" з урахуванням усіх змін.

Розподіл логіки:

– збереження інформації: Дані зберігаються у текстовому файлі, що спрощує розгортання програми, але обмежує масштабованість;

– обробка даних: Всі операції над книгами виконуються у масиві books[], що дозволяє швидко змінювати дані без зайвих викликів I/O;

– взаємодія з користувачем: Використовується графічний інтерфейс Qt з кнопками та списками, що робить програму зручною для роботи;

– ші-асистент: Взаємодіє з користувачем через окреме вікно, де ІІ-модель надає рекомендації на основі збережених книг.

Ця система має просту монолітну архітектуру, але завдяки Qt забезпечує зручний графічний інтерфейс. Надалі можливе розширення функціоналу, наприклад, інтеграція бази даних або поліпшення роботи ШІ-асистента.

4.2 Інструкція користувача

Інтелектуальна бібліотека - це система управління бібліотечним фондом, що дозволяє користувачам здійснювати пошук, додавати, змінювати та видаляти книги, а також отримувати рекомендації від вбудованого ШІ-асистента.

При запуску програми користувач бачить головне вікно, яке складається з таких елементів:

Список книг (зліва) - перелік наявних книг у бібліотеці (відображаються лише назви).

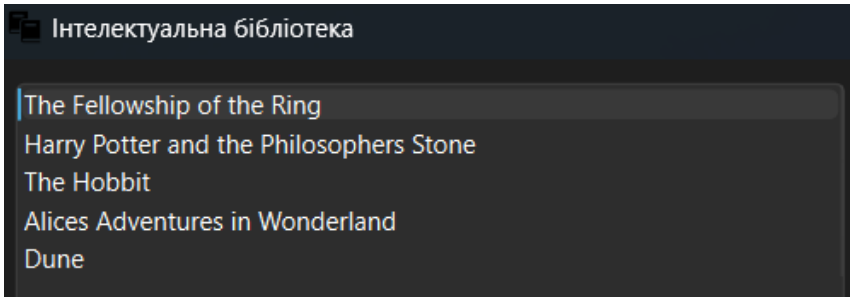

Рисунок 6 - Список книг

Панель деталей книги (знизу) - показує інформацію про вибрану книгу (назва, рік, жанр, автор, країна, місце зберігання).

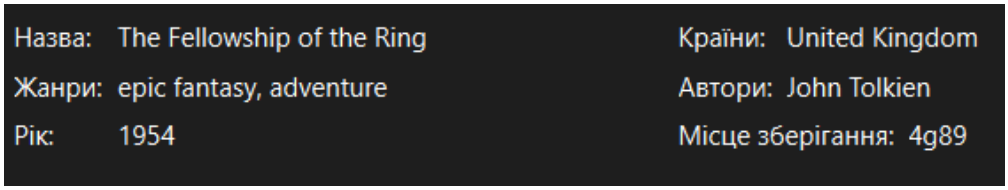

Рисунок 7 - Інформація про книги

Поле пошуку (праворуч угорі) - дозволяє здійснювати пошук книг за різними критеріями які впише користувач (назва, рік, номер полиці, жанр, автор тощо).

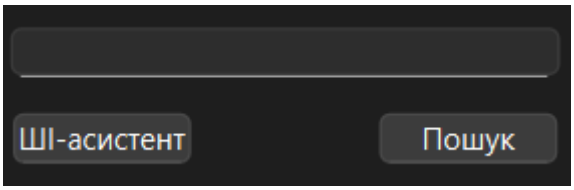

Рисунок 8 - Поле пошуку

Кнопки керування (зправа):

– "пошук" - виконує пошук за введеним критерієм;
– "додати" - відкриває вікно додавання нової книги;
– "змінити" - відкриває вікно редагування вибраної книги;
– "видалити" - видаляє вибрану книгу зі списку;
– "зберегти зміни" - записує всі внесені зміни у текстовий файл;
– "ШІ-асистент" - відкриває вікно взаємодії з вбудованим інтелектуальним асистентом.

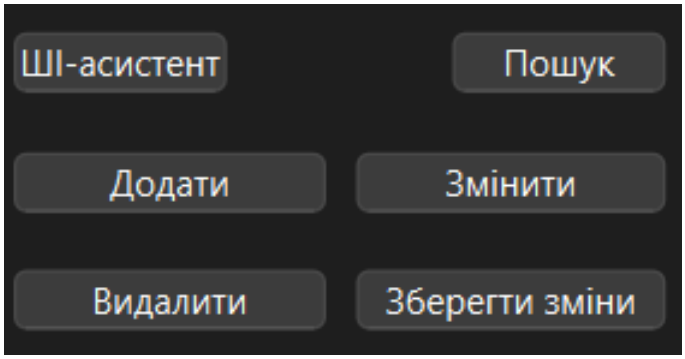

Рисунок 9 - Кнопки керування

Нижче представлена інструкція для роботи з усіма функціями програми:

Додавання книги:

– натиснути кнопку "Додати";
– у новому вікні ввести усі необхідні дані про книгу;
– підтвердити введення, натиснувши "Зберегти";

– книга з’явиться у списку.

Зміна книги:

– вибрати книгу зі списку;

– натиснути "Змінити";

– у відкритому вікні змінити необхідні дані;

– натиснути "Зберегти зміни".

Видалення книги:

– виберати книгу зі списку;

– натиснути "Видалити";

– підтвердити видалення.

Пошук книги:

– ввести критерій пошуку (назва, рік, жанр тощо) у поле пошуку;

– натиснути "Пошук";

– у списку з’являться відповідні результати.

Робота з ШІ-асистентом:

– натиснути "ШІ-асистент";

– у вікні чату задати питання або запит щодо книжкових рекомендацій;

– отримати відповідь від асистента.

Збереження змін:

– після будь-яких змін у бібліотеці натиснути "Зберегти зміни";

– дані буде збережено у текстовий файл.

Ця інструкція дозволить користувачам легко орієнтуватися у програмі та ефективно використовувати її функціонал.

4.3 Тестування

Функціональне тестування програми проводилось відповідно до заздалегідь розроблених тестових сценаріїв, щоб перевірити коректність роботи основних функцій. Нижче наведені результати тестування:

Тест 1: Введення користувачем запиту на рекомендацію книги

Сценарій: Користувач вводить запит на рекомендацію книги за жанром ("фантастика"). Програма надсилає запит до ШІ-асистента (GPT-3.5) і відображає результати у графічному інтерфейсі.

Очікуваний результат: Список книг, які відповідають запиту, відображається у графічному інтерфейсі.

Фактичний результат:

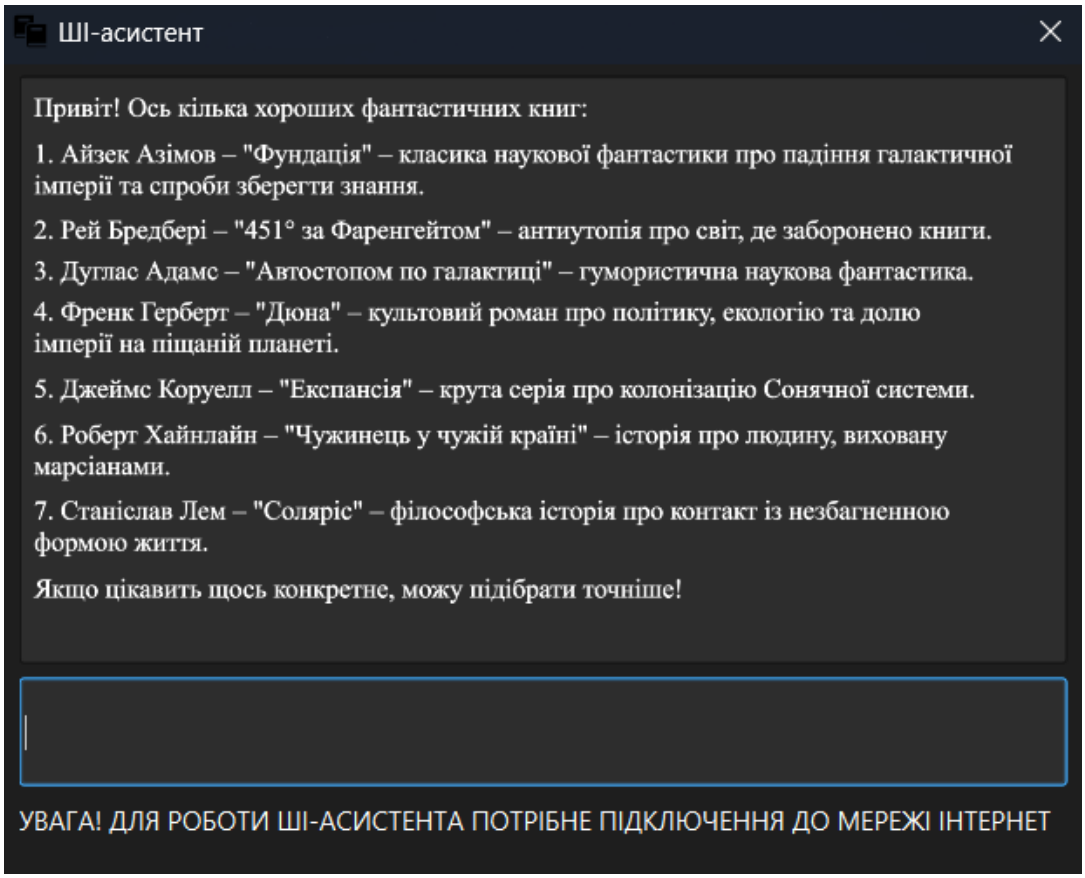

Рисунок 10 - Результати запиту

Список рекомендацій із 5 книг жанру "фантастика" успішно відображено у GUI.

Тест 2: Збереження даних у TXT-файл

Сценарій: Користувач додає нову книгу в базу даних (назва: "1984", автор: "Джордж Орвелл", рік видання "1949", країна "Великобританія") і зберігає її у локальному TXT-файлі.

Очікуваний результат: Дані зберігаються у файлі BOOKS.txt у форматі, що відповідає специфікації.

Фактичний результат:

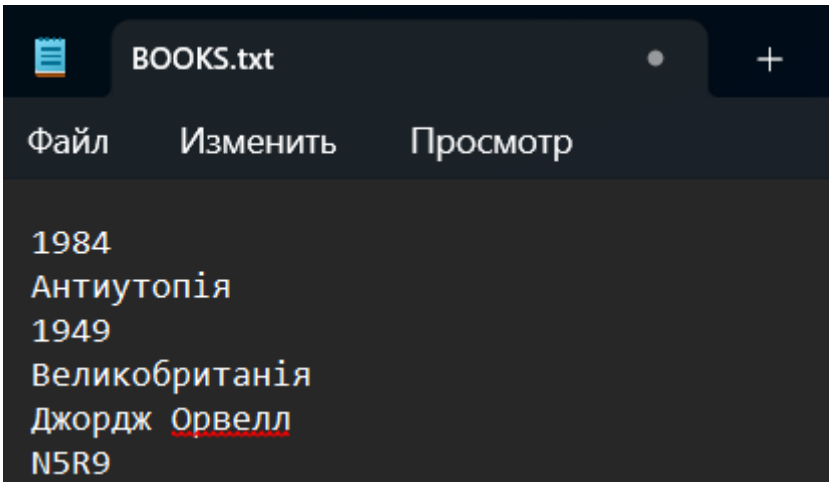

Рисунок 11 - Збережені дані

Дані успішно збережені.

Тест 3: Завантаження даних із TXT-файлу

Сценарій: Користувач запускає програму, і вона автоматично завантажує дані з файлу BOOKS.txt.

Очікуваний результат: Список книг відображається у графічному інтерфейсі.

Фактичний результат:

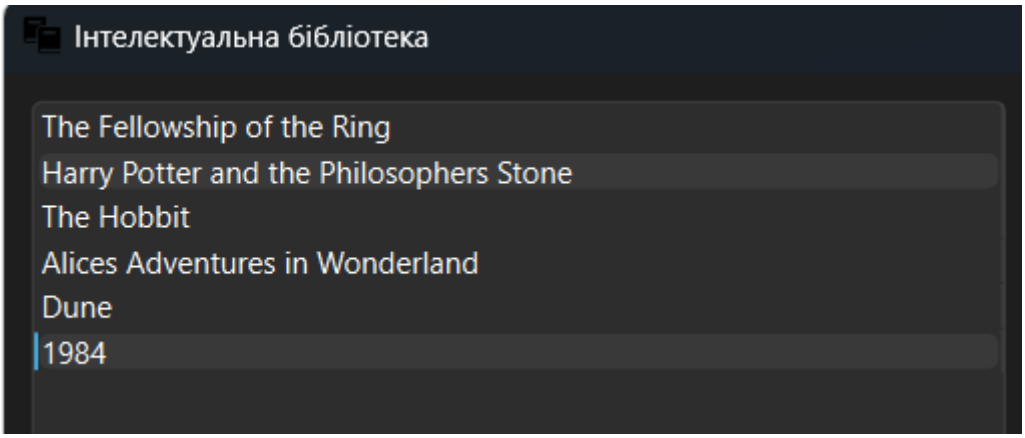

Рисунок 12 - Список книг у програмі

Дані з файлу успішно завантажені, список книг відображено в інтерфейсі.

Тест 4: Обробка некоректного запиту до ШІ-асистента

Сценарій: Користувач вводить некоректний запит (наприклад, набір випадкових символів "##!!&&").

Очікуваний результат: Програма відображає повідомлення про помилку та запитує користувача повторити запит.

Фактичний результат:

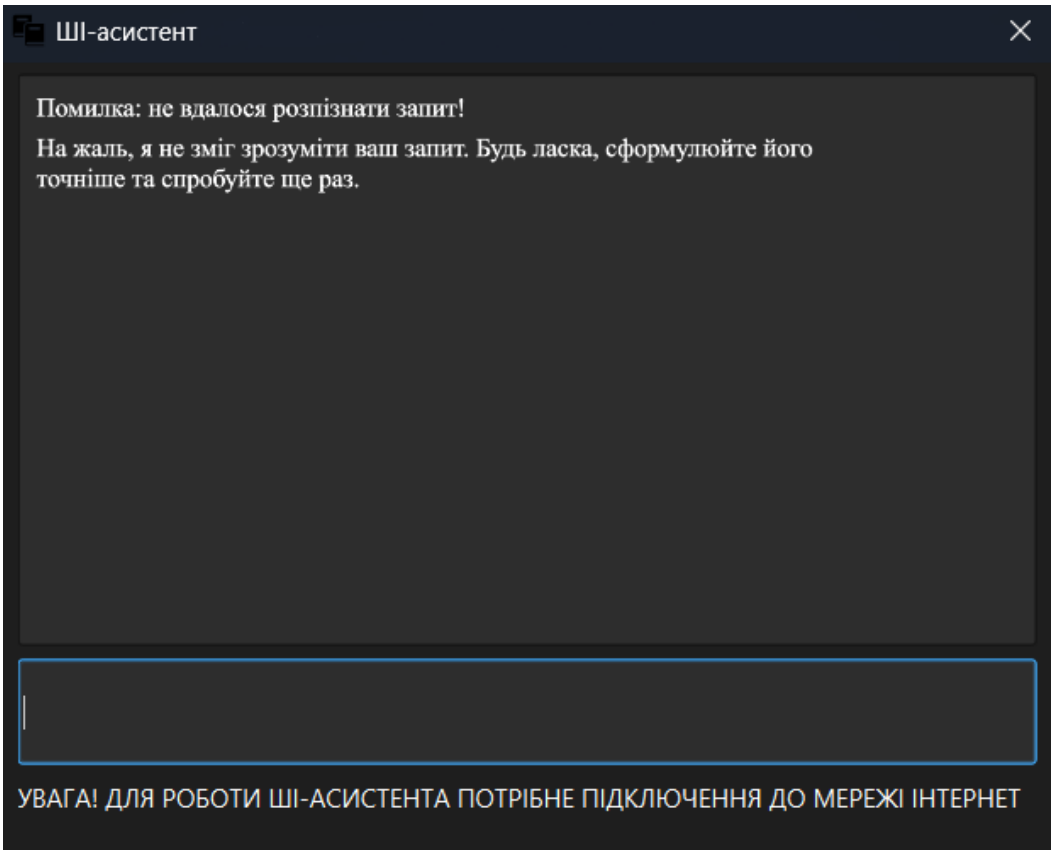

Рисунок 13 - Повідомлення про помилку

Відображено повідомлення: "Запит некоректний. Будь ласка, уточніть дані."

Тест 5: Взаємодія з графічним інтерфейсом

Сценарій: Користувач натискає кнопки "ШІ-асистент" та "Додати" у GUI.

Очікуваний результат: Всі кнопки виконують свої функції без помилок.

Фактичний результат:

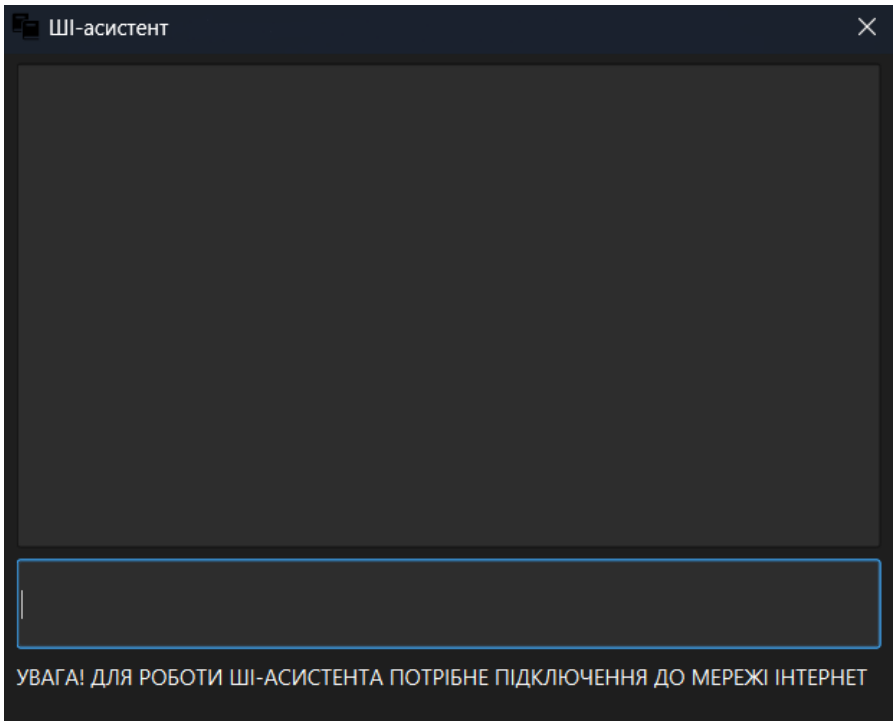

Рисунок 14 - Чат з ШІ-асистентом

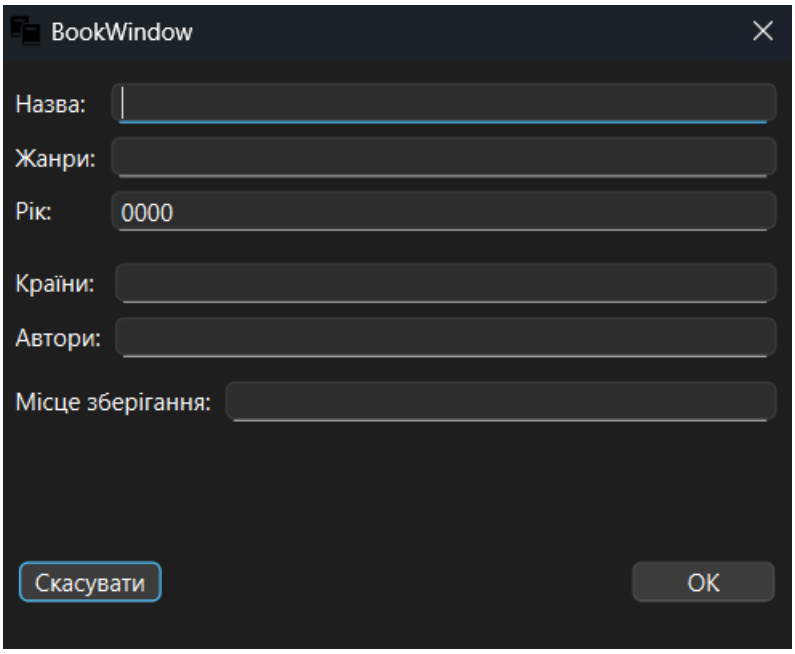

Рисунок 15 - Вікно додавання книги

Всі функції виконуються коректно, результат відображається у GUI відповідно до дій користувача.

Усі функції програми успішно пройшли функціональне тестування. Програма працює стабільно, основні сценарії взаємодії з користувачем виконуються без помилок. Коректність роботи з даними та інтеграція з ШІ-асистентом підтверджені.

ВИСНОВОК

У результаті виконаного дослідження було розроблено інтелектуальну систему для автоматизації управління бібліотечним фондом, що включає вбудований ШІ-асистент. Ця система значно підвищує ефективність роботи бібліотеки, оскільки автоматизує ключові процеси, такі як облік та інвентаризація книг, мінімізує людські помилки та оптимізує пошук літератури за різними параметрами. Важливою перевагою є інтеграція з цифровими ресурсами, що дає змогу забезпечити доступ до електронних книг та архівів, що відповідають сучасним вимогам цифрової ери.

Штучний інтелект у складі системи виконує роль асистента, який надає рекомендації користувачам на основі їхніх вподобань, історії читання та пошукових запитів. ШІ допомагає знаходити книги, що відповідають інтересам користувача, а також може відповідати на питання щодо наявності книг, їхнього змісту та інших характеристик. Це сприяє покращенню досвіду користувачів та підвищенню ефективності пошуку потрібної літератури. Система забезпечує доступ до більш точних і релевантних рекомендацій, що оптимізує взаємодію користувачів з бібліотечним фондом.

Було створено програмне забезпечення, яке не тільки автоматизує управлінські процеси в бібліотеці, але й значно покращує обслуговування користувачів, завдяки персоналізованим рекомендаціям. Вирішення поставлених задач забезпечило створення ефективної системи, яка відповідає потребам сучасних бібліотек.

Практичне значення дослідження полягає у можливості впровадження такої системи в реальних умовах роботи бібліотек, що дозволить знизити навантаження на персонал, підвищити якість обслуговування користувачів та оптимізувати роботу з бібліотечним фондом. Впровадження програмного забезпечення з ШІ-асистентом відкриває нові можливості для бібліотечних установ, роблячи їх більш доступними, ефективними та сучасними в умовах цифровізації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бондаренко О. В. Основи програмування на C++: навч. посіб. Київ: Видавництво «КНУ», 2021. 320 с.
2. Василенко П. І. Оптимізація алгоритмів у C++: практичний підхід. Одеса: ОНПУ, 2022. 310 с.
3. ДСТУ 9152:2022. Видавництво. Оформлення бібліографічних описів.
4. Демченко В. І. Автоматизація процесів аналізу даних за допомогою ШІ. Харків: ХНУРЕ, 2022. 280 с.
5. Дурєєва Т. Застосування інформаційних технологій у бібліотечній сфері. Вісник книжної палати. 2009. № 1. С. 23–25.
6. Дурєєва Т. А. Інформатизація бібліотечної сфери в умовах розвитку технологій інформаційного суспільства. Вісн. Харк. держ. акад. культури: зб. наук. праць. 2009. Т. 24. С. 145–156.
7. Гринько Д. О. Qt для початківців: створення графічних застосунків. Львів: Видавництво «Львівська політехніка», 2019. 256 с.
8. Косарєв В., Мельник І., Герасименко М. Розробка програмного забезпечення з використанням Qt. Львів: Видавництво Львівської політехніки, 2021. 256 с.
9. Ковальчук В., Демченко А., Сидоренко П., Черненко Ю. Інтелектуальні системи аналізу даних. Київ: Наукова думка, 2020. 384 с.
10. Коваленко А. С., Ткачук В. Г. Бази даних та їх інтеграція у застосунки на C++. Дніпро: ДНУ, 2021. 270 с.
11. Кравець О. П. Нейронні мережі та їх використання у програмуванні. Тернопіль: ТНТУ, 2023. 305 с.
12. Петренко І. М. Штучний інтелект: методи та алгоритми. Харків: Вид-во «Ранок», 2020. 288 с.
13. Шаповаленко О. Алгоритмічні основи програмування. Київ: Ліра-К, 2020. 312 с.

14. Шевченко М. Ю. Розробка багатопотокових застосунків у Qt: методологія та практика. Київ: Наук. думка, 2023. 290 с.
15. Сидоренко Л. В. Основи алгоритмізації та програмування: навч. посіб. Вінниця: ВНТУ, 2020. 345 с.
16. Смалько О. Оптимізація роботи нейромереж у завданнях класифікації текстових даних. Автореф. дис. … канд. техн. наук: 05.13.23. Київ, 2021. 28 с.
17. Шишкіна О. С. Використання штучного інтелекту в бібліотечній діяльності. Інформаційні технології і системи в документознавчій сфері. 2024. С. 124–126.
18. Шаров С. В. Сучасний стан розвитку штучного інтелекту та напрямки його використання. Українські студії в європейському контексті. 2023. № 6. С. 136–143.
19. Шаров С. В., Шамардак О. А. Концептуальне моделювання інформаційної системи за допомогою мови UML. Інформаційне суспільство: технологічні, економічні та технічні аспекти становлення. 2014. № 7. С. 10–14.
20. Юрченко В. А. Системне програмування на C++: концепції та підходи. Запоріжжя: ЗНУ, 2021. 275 с.
21. Automating C++ Code Testing with Google Test. Google Testing Blog: веб-сайт. URL: https://testing.googleblog.com (дата звернення: 16.02.2025).
22. Best Practices for Exception Handling in C++. IBM Developer: веб-сайт. URL: https://developer.ibm.com/articles/c-exception-handling/ (дата звернення: 13.02.2025).
23. Best IDEs for C++ Development in 2025. DevTools Review: веб-сайт. URL: https://devtools.com/best-cpp-ides (дата звернення: 15.02.2025).
24. Best Libraries for C++ Developers. Open Source Initiative: веб-сайт. URL: https://opensource.org/best-cpp-libraries (дата звернення: 21.02.2025).

25. Best Practices for Secure C++ Programming. Security Weekly: веб-сайт. URL: https://securityweekly.com/cpp-security (дата звернення: 02.02.2025).
26. Bishop C., Nasrabadi N., Barber D. Pattern Recognition and Machine Learning. Springer, 2016. 738 p.
27. Building GUI Applications with Qt. Qt Academy: веб-сайт. URL: https://academy.qt.io (дата звернення: 16.02.2025).
28. Blanchette J., Summerfield M. C++ GUI Programming with Qt 5. 2nd ed. Prentice Hall, 2016. 512 p.
29. Creating Cross-Platform Apps with Qt. Qt Dev Blog: веб-сайт. URL: https://blog.qt.io (дата звернення: 12.02.2025).
30. C++ and AI: How to Implement Machine Learning Algorithms. AI Alignment: веб-сайт. URL: https://www.ai-alignment.com/cpp-ml (дата звернення: 10.02.2025).
31. C++ and Python Interoperability. Real Python: веб-сайт. URL: https://realpython.com/cpp-python/ (дата звернення: 08.02.2025).
32. Deep Learning with C++ and OpenCV. PyImageSearch: веб-сайт. URL: https://www.pyimagesearch.com/deep-learning-opencv/ (дата звернення: 21.02.2025).
33. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016. 775 p.
34. ISO/IEC 14882:2020. Programming Language C++. International Organization for Standardization, 2020.
35. Lippman S., Lajoie J. C++ Primer. 5th ed. Addison-Wesley, 2012. 976 p.
36. Meyers S. Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14. O’Reilly Media, 2014. 334 p.
37. Stroustrup B. The C++ Programming Language. 4th ed. Addison-Wesley, 2013. 1366 p.
38. What’s new in C++20? CppReference: веб-сайт. URL: https://en.cppreference.com/w/cpp/20 (дата звернення: 13.02.2025).

39. Introduction to Qt for Beginners. Qt Documentation: веб-сайт. URL: https://doc.qt.io/qt-6/gettingstarted.html (дата звернення: 21.02.2025).
40. Modern C++ Best Practices. ISO C++: веб-сайт. URL: https://isocpp.org/get-started (дата звернення: 13.02.2025).
41. Qt Quick and QML Overview. Qt Wiki: веб-сайт. URL: https://wiki.qt.io/Qt_Quick (дата звернення: 14.02.2025).
42. Using C++ for AI Applications. Towards Data Science: веб-сайт. URL: https://towardsdatascience.com/cpp-for-ai (дата звернення: 12.02.2025).
43. Multithreading in Modern C++. GeeksforGeeks: веб-сайт. URL: https://www.geeksforgeeks.org/multithreading-in-cpp/ (дата звернення: 13.02.2025).
44. Memory Management in C++. Stack Overflow: веб-сайт. URL: https://stackoverflow.com/questions/657612 (дата звернення: 15.02.2025).
45. Performance Optimization Techniques in C++. JetBrains Blog: веб-сайт. URL: https://blog.jetbrains.com/cpp/performance-tips/ (дата звернення: 12.02.2025).
46. Using Qt for Database Applications. Qt Wiki: веб-сайт. URL: https://wiki.qt.io/Qt_for_Databases (дата звернення: 13.02.2025).
47. OpenAI API and C++. OpenAI Docs: веб-сайт. URL: https://platform.openai.com/docs/cpp-api (дата звернення: 21.02.2025).
48. Writing High-Performance C++ Code. Perf Programming: веб-сайт. URL: https://perf-programming.com/cpp-high-performance (дата звернення: 09.02.2025).
49. Using C++ in Embedded Systems. Embedded.com: веб-сайт. URL: https://www.embedded.com/cpp-in-embedded/ (дата звернення: 13.02.2025).
50. Machine Learning Libraries for C++ Developers. AI Weekly: веб-сайт. URL: https://aiweekly.com/cpp-ml-libraries (дата звернення: 13.02.2025).
51. Hugging Face – платформа для роботи з моделями ШІ. Hugging Face: веб-сайт. URL: https://huggingface.co/ (дата звернення: 16.02.2025).

52. Qt Quick and QML Overview. Qt Wiki: веб-сайт. URL: https://wiki.qt.io/Qt_Quick (дата звернення: 14.02.2025).
53. OpenAI API and C++. OpenAI Docs: веб-сайт. URL: https://platform.openai.com/docs/cpp-api (дата звернення: 21.02.2025).

ДОДАТОК А

Лістинг коду програми

main.cpp:

```
#include "stdafx.h"
#include "mainapp.h"
#include <QtWidgets/QApplication>
#include <iostream>
#include <list>
#include <string>
#include <fstream>
#include <cstdlib>
#include <Windows.h>
#include <time.h>
using namespace std;
extern int b, n, s, p, booksCount, count, choice, booksDeleted, temp;
struct book
{
    int year;
    string genre;
    string author;
    string country;
    string name;
    string place;
};
int main(int argc, char* argv[])
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    time_t mytime = time(NULL);
    struct tm* now = localtime(&mytime);
    QApplication a(argc, argv);
    MainApp w;
    w.show();
    return a.exec();
```

```
}
```

mainapp.cpp:

```
#include "stdafx.h"
#include "mainapp.h"
#include "addbook.h"
#include <QtWidgets/QApplication>
#include <iostream>
#include <list>
#include <string>
#include <fstream>
#include <cstdlib>
#include <Windows.h>
#include <time.h>
#include "AiChatWindow.h"
using namespace std;
struct book
{
    int year;
    string genre;
    string author;
    string country;
    string name;
    string place;
};
struct book books[2000];
int b = 1, n = 0, s = 1, p = 0, booksCount = 0, count = 1, choice = 0, booksDeleted = 0, temp =
0, currentSearchIndex = 0;
string strInput, tmp;
bool itemChangeButtonClicked = false;
int localMain()
{
```

```
fstream textFile("BOOKS.txt");

if (!textFile)
{
   ofstream outf("BOOKS.txt");
   outf.close();
   return 404;
}
while (textFile)
{
   while (getline(textFile, strInput))
   {
      switch (b)
      {
      case 1:
         books[n].name = strInput;
         break;
      case 2:
         books[n].genre = strInput;
         break;
      case 3:
         books[n].year = stoi(strInput);
         break;
      case 4:
         books[n].country = strInput;
         break;
      case 5:
         books[n].author = strInput;
         break;
      case 6:
         books[n].place = strInput;
         break;
```

```
            default:
                break;
            }
            if (b == 6)
            {
                b = 0;
                n++;
            }
            b++;
        }
    }
    while (true)
    {
        if (books[p].year == NULL)
        {
            break;
        }
        else
        {
            p++;
        }
    }
}
MainApp::MainApp(QWidget *parent)
        : QWidget(parent)
{
    localMain();
         ui.setupUi(this);
    addBook5 = new addBook;
    aiChat = new AiChatWindow;
    for (int i = 0; i < p - booksDeleted; i++)
    {
```

```
        ui.booksList->addItem(QString::fromStdString(books[i].name));
    }
    connect(addBook5, &addBook::signalAddBook, this,
&MainApp::on_signalAddBook_received);
    connect(ui.aiButton, &QPushButton::clicked, this, &MainApp::on_aiButton_clicked);
}
MainApp::~MainApp()
{
}
void MainApp::on_addBookButton_clicked()
{
    addBook5->show();
}
void MainApp::on_booksList_itemSelectionChanged()
{
        ui.nameLabel->setText(QString::fromStdString(books[ui.booksList-
>currentRow()].name));
        ui.genreLabel->setText(QString::fromStdString(books[ui.booksList-
>currentRow()].genre));
        ui.yearLabel->setText(QString::fromStdString(to_string(books[ui.booksList-
>currentRow()].year)));
        ui.countryLabel->setText(QString::fromStdString(books[ui.booksList-
>currentRow()].country));
        ui.authorLabel->setText(QString::fromStdString(books[ui.booksList-
>currentRow()].author));
        ui.placeLabel->setText(QString::fromStdString(books[ui.booksList-
>currentRow()].place));
}

void MainApp::on_deleteBookButton_clicked()
{
    choice = ui.booksList->currentRow();
```

```
    for (int w = choice; w < p; w++)
    {
        books[w] = books[w + 1];
    }
    booksDeleted++;
    ui.booksList->takeItem(choice);
}
void MainApp::on_saveBooksButton_clicked()
{
        ofstream outf("BOOKS.txt", ios::trunc);
        for (int f = 0; f < p - booksDeleted; f++)
        {
            outf << books[f].name << endl;
            outf << books[f].genre << endl;
            outf << books[f].year << endl;
            outf << books[f].country << endl;
            outf << books[f].author << endl;
            outf << books[f].place << endl;
        }
        outf.close();
}
void MainApp::on_signalAddBook_received(QString name, QString genre, QString year,
QString country, QString author, QString place)
{
    if (!itemChangeButtonClicked)
    {
        ui.booksList->addItem(name);
        books[p - booksDeleted].name = name.toUtf8().data();
        books[p - booksDeleted].genre = genre.toUtf8().data();
        books[p - booksDeleted].year = stoi(year.toUtf8().data());
        books[p - booksDeleted].country = country.toUtf8().data();
        books[p - booksDeleted].author = author.toUtf8().data();
```

```
        books[p - booksDeleted].place = place.toUtf8().data();
        booksDeleted--;
    }
    else
    {
        choice = ui.booksList->currentRow();
        books[choice].name = name.toUtf8().data();
        ui.booksList->currentItem()->setText(name);
        books[choice].genre = genre.toUtf8().data();
        books[choice].year = stoi(year.toUtf8().data());
        books[choice].country = country.toUtf8().data();
        books[choice].author = author.toUtf8().data();
        books[choice].place = place.toUtf8().data();
        itemChangeButtonClicked = false;
    }
}
void MainApp::on_changeBookButton_clicked()
{
    addBook5->show();
    itemChangeButtonClicked = true;
}
void MainApp::on_searchBookButton_clicked()
{
    bool isFilterMet = false;
    tmp = ui.findBookEdit->text().toUtf8().data();
    for (int i = currentSearchIndex; i <= p - booksDeleted; i++)
    {
        isFilterMet = false;
        if (books[i].name == tmp) { isFilterMet = true; }
        else if (books[i].genre == tmp) { isFilterMet = true; }
        else if (tmp == to_string(books[i].year)) { isFilterMet = true; }
        else if (books[i].country == tmp) { isFilterMet = true; }
```

```
        else if (books[i].author == tmp) { isFilterMet = true; }
        else if (books[i].place == tmp) { isFilterMet = true; }
        if (isFilterMet)
        {
            ui.booksList->setCurrentRow(i);
            currentSearchIndex = i+1;
            break;
        }
    }
    if(!isFilterMet) currentSearchIndex = 0;
}
void MainApp::on_aiButton_clicked()
{
    aiChat->show();
}
```

addBook.cpp:

```
#include "stdafx.h"
#include "addbook.h"
addBook::addBook(QWidget *parent)
        : QDialog(parent)
{
        setupUi(this);
}
addBook::~addBook()
{}
void addBook::on_okButton_clicked()
{
    QString name = nameEdit->text();
    QString genre = genreEdit->text();
    QString year = yearEdit->text();
    QString country = countryEdit->text();
```

```
    QString author = authorEdit->text();
    QString place = placeEdit->text();
    emit signalAddBook(name, genre, year, country, author, place);
}
```

mainapp.h:

```
#pragma once
#include <QWidget>
#include "ui_mainapp.h"
#include "addBook.h"
#include "aiChatWindow.h"

class MainApp : public QWidget
{
        Q_OBJECT
public:
        MainApp(QWidget *parent = nullptr);
        ~MainApp();

private:
        Ui::MainAppClass ui;
        addBook* addBook5;
        AiChatWindow* aiChat;
private slots:
        void on_addBookButton_clicked();
        void on_booksList_itemSelectionChanged();
        void on_deleteBookButton_clicked();
        void on_saveBooksButton_clicked();
        void on_changeBookButton_clicked();
        void on_searchBookButton_clicked();
public slots:
```

```
    void on_signalAddBook_received(QString name, QString genre, QString year, QString country, QString author, QString place);
    void on_aiButton_clicked();
};
```

addBook.h:

```
#pragma once
#include <QDialog>
#include "ui_addbook.h"
class addBook : public QDialog, public Ui::addBookClass
{
    Q_OBJECT
public:
    addBook(QWidget *parent = nullptr);
    ~addBook();
private:
signals:
    void signalAddBook(QString, QString, QString, QString, QString, QString);
public slots:
private slots:
    void on_okButton_clicked();
};
```

gpt_api.cpp:

```
#include <stdafx.h>
#include <windows.h>
#include <winhttp.h>
#include <QThread>
#include <QObject>
#include <QString>
#include <string>
#pragma comment(lib, "winhttp.lib")
```

```
class GPTClient : public QThread {
    Q_OBJECT
public:
    explicit GPTClient(const QString& prompt, QObject* parent = nullptr);
    GPTClient() = delete;
    void run() override;
signals:
    void responseReceived(const QString& response);

private:
    QString prompt;
};
GPTClient::GPTClient(const QString& prompt, QObject* parent) : QThread(parent),
prompt(prompt) {}
void GPTClient::run() {
    std::string apiKey = "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";
    std::string requestData = "{\"inputs\": \"" + prompt.toStdString() + "\"}";
    HINTERNET hSession = WinHttpOpen(L"GPT Client",
WINHTTP_ACCESS_TYPE_DEFAULT_PROXY, WINHTTP_NO_PROXY_NAME,
WINHTTP_NO_PROXY_BYPASS, 0);
    if (!hSession) return;
    HINTERNET hConnect = WinHttpConnect(hSession, L"api-inference.huggingface.co",
INTERNET_DEFAULT_HTTPS_PORT, 0);
    if (!hConnect) {
        WinHttpCloseHandle(hSession);
        return;
    }
    HINTERNET hRequest = WinHttpOpenRequest(hConnect, L"POST",
L"/models/jondurbin/airoboros-gpt-3.5-turbo-100k-7b", NULL, WINHTTP_NO_REFERER,
WINHTTP_DEFAULT_ACCEPT_TYPES, WINHTTP_FLAG_SECURE);
    if (!hRequest) {
        WinHttpCloseHandle(hConnect);
```

```
        WinHttpCloseHandle(hSession);
        return;
    }
    std::wstring headers = L"Authorization: Bearer " + std::wstring(apiKey.begin(),
apiKey.end()) + L"\r\nContent-Type: application/json\r\n";
    WinHttpAddRequestHeaders(hRequest, headers.c_str(), (ULONG)-1L,
WINHTTP_ADDREQ_FLAG_ADD);
    BOOL sent = WinHttpSendRequest(hRequest,
WINHTTP_NO_ADDITIONAL_HEADERS, 0, (LPVOID)requestData.c_str(),
requestData.size(), requestData.size(), 0);
    if (!sent) {
        WinHttpCloseHandle(hRequest);
        WinHttpCloseHandle(hConnect);
        WinHttpCloseHandle(hSession);
        return;
    }
    WinHttpReceiveResponse(hRequest, NULL);
    DWORD sizeAvailable = 0;
    WinHttpQueryDataAvailable(hRequest, &sizeAvailable);
    if (sizeAvailable == 0) {
        WinHttpCloseHandle(hRequest);
        WinHttpCloseHandle(hConnect);
        WinHttpCloseHandle(hSession);
        return;
    }
    std::string response(sizeAvailable, '\0');
    DWORD bytesRead = 0;
    if (!WinHttpReadData(hRequest, &response[0], sizeAvailable, &bytesRead)) {
        WinHttpCloseHandle(hRequest);
        WinHttpCloseHandle(hConnect);
        WinHttpCloseHandle(hSession);
        return;
```

```
    }
    WinHttpCloseHandle(hRequest);
    WinHttpCloseHandle(hConnect);
    WinHttpCloseHandle(hSession);
    emit responseReceived(QString::fromStdString(response));
}
```

gpt_api.h:

```
#ifndef GPT_API_H
#define GPT_API_H
#include <string>
std::string responseReceived(const std::string& prompt);
#endif
```

aiChatWindow:

```
#include "AiChatWindow.h"
#include <QVBoxLayout>
#include <QKeyEvent>
AiChatWindow::AiChatWindow(QWidget* parent) : QWidget(parent) {
    QVBoxLayout* layout = new QVBoxLayout(this);
    aiOutputEdit = new QTextEdit(this);
    aiOutputEdit->setReadOnly(true);
    layout->addWidget(aiOutputEdit);
    aiTextInputEdit = new QLineEdit(this);
    layout->addWidget(aiTextInputEdit);
    connect(aiTextInputEdit, &QLineEdit::returnPressed, this,
&AiChatWindow::sendMessage);
}
void AiChatWindow::sendMessage() {
    QString userMessage = aiTextInputEdit->text().trimmed();
    if (userMessage.isEmpty()) return;
    aiOutputEdit->append("Вы: " + userMessage);
```

```
    aiTextInputEdit->clear();
    GPTClient* client = new GPTClient(userMessage, this);
    connect(client, &GPTClient::responseReceived, this, &AiChatWindow::displayResponse);
    connect(client, &GPTClient::finished, client, &QObject::deleteLater);
    client->start();
}
void AiChatWindow::displayResponse(const QString& response) {
    aiOutputEdit->append("ИИ: " + response);
}
```

aiChatWindow.h:

```
#ifndef AICHATWINDOW_H
#define AICHATWINDOW_H
#include <QWidget>
#include <QLineEdit>
#include <QTextEdit>
#include "gpt_api.h"
class AiChatWindow : public QWidget {
    Q_OBJECT
public:
    explicit AiChatWindow(QWidget* parent = nullptr);
private slots:
    void sendMessage();
    void displayResponse(const QString& response);
private:
    QLineEdit* aiTextInputEdit;
    QTextEdit* aiOutputEdit;
};
#endif
```

ДОДАТОК Б

Стаття

Матеріали IV Всеукраїнської науково-практичної Інтернет-конференції «Сучасні комп'ютерні та інформаційні системи і технології»

УДК 004.8:027.7

ОГЛЯД МОЖЛИВОСТЕЙ ЗАСТОСУВАННЯ ШТУЧНОГО ІНТЕЛЕКТУ У БІБЛІОТЕЧНИХ СИСТЕМАХ

Коржилов Б.М., здобувач вищої освіти *e-mail: boris090901@gmail.com*
Сіциліцин Ю.О., Ph.D. *e-mail: yurii.sitsilitsin@tsatu.edu.ua*
Таврійський державний агротехнологічний університет імені Дмитра Моторного

Актуальність та постановка проблеми. Сучасні бібліотечні системи стикаються з викликами, пов'язаними зі стрімким зростанням обсягу інформації, потребою в її швидкій обробці та наданні користувачам доступу до релевантних ресурсів. Традиційні методи каталогізації, індексації та обслуговування вже не задовольняють вимог інформаційного суспільства. Штучний інтелект (ШІ) пропонує нові підходи для автоматизації процесів, персоналізації послуг та аналітики користувацьких даних, що робить бібліотеки більш ефективними та доступними. Проблема полягає у відсутності інтегрованих рішень, які б використовували можливості ШІ для покращення функціонування бібліотек. Постає необхідність у дослідженні потенціалу ШІ для впровадження інноваційних технологій у бібліотечну діяльність та розробці підходів для їх практичної реалізації.

Основні матеріали дослідження. Штучний інтелект стає ключовим інструментом для трансформації сучасних бібліотек, надаючи їм можливість автоматизувати рутинні процеси, вдосконалювати надання послуг і адаптуватися до потреб сучасного користувача. Одним із найбільш значущих напрямів впровадження ШІ у бібліотеках є автоматизація процесу каталогізації та індексації. Використання технологій обробки природної мови (NLP) дозволяє ШІ аналізувати текстові дані з книг та інших документів, автоматично генеруючи метадані, ключові слова та анотації. Покращення пошукових систем є наступним важливим кроком, який забезпечується завдяки машинному навчанню та аналізу даних. Бібліотечні пошукові системи, оснащені ШІ, здатні надавати більш релевантні результати, орієнтуючись на поведінку користувача та контекст його запитів. Важливим інструментом для модернізації бібліотек є технології розпізнавання тексту. Завдяки використанню систем OCR (оптичне розпізнавання символів), старі паперові книги, рукописи та архівні матеріали можуть бути оцифровані та включені до загальнодоступних баз даних. Однією з перспективних можливостей ШІ є створення віртуальних помічників, які функціонують на основі чат-ботів або голосових помічників. Віртуальні помічники можуть працювати у реальному часі, допомагаючи користувачам знаходити необхідну інформацію, оформлювати замовлення на книги чи отримувати довідкові послуги. Не менш значущим є впровадження систем автоматичного перекладу на основі ШІ. Бібліотеки, які працюють з іноземними документами, часто стикаються із проблемами мовного бар'єру. Таким чином, штучний інтелект відкриває широкі можливості для трансформації бібліотек у динамічні, інноваційні та доступні центри знань.

У результаті аналізу можливостей застосування штучного інтелекту у бібліотечних системах було сформульовано ключові вимоги до їх проєктування. Ці вимоги базуються на потребах сучасних бібліотек, а також враховують технічні можливості, що надає штучний інтелект для оптимізації процесів, покращення функціональності та підвищення зручності для користувачів.

Перша вимога – автоматизація рутинних завдань. Системи повинні використовувати алгоритми обробки природної мови (NLP) для автоматичної каталогізації, індексації та анотації інформаційних ресурсів. Ця функціональність дозволяє значно скоротити витрати часу працівників бібліотек на обробку нових надходжень.

Друга вимога – інтелектуальний пошук та рекомендації. Бібліотечні системи повинні підтримувати покращений пошук на основі машинного навчання та аналізу поведінки користувача. ШІ повинен надавати персоналізовані рекомендації, аналізуючи інтереси читача, історію його запитів та взаємодію з ресурсами.

Третя вимога – оцифровка та обробка архівних матеріалів. Важливим аспектом є впровадження технологій оптичного розпізнавання символів (OCR) для переведення паперових книг та рукописів у цифровий формат. Алгоритми штучного інтелекту мають забезпечувати високу точність розпізнавання, навіть для пошкоджених чи рукописних документів.

Четверта вимога – підтримка віртуальних помічників. Системи мають передбачати інтеграцію інтелектуальних чат-ботів або голосових помічників для оперативного обслуговування користувачів у режимі реального часу

П'ята вимога – аналітика даних та прогнозування трендів. Бібліотечні системи мають використовувати алгоритми аналітики великих даних (Big Data) для аналізу відвідуваності ресурсів, популярності книг та прогнозування попиту на певні категорії матеріалів. Отримані дані допоможуть бібліотекам оптимізувати свої фонди та ефективніше планувати закупівлі.

Шоста вимога – інтеграція систем автоматичного перекладу. Штучний інтелект повинен забезпечувати автоматичний переклад іноземних документів для розширення доступу користувачів до світових наукових та освітніх ресурсів.

Сьома вимога – забезпечення безпеки та захисту даних. Проєктування бібліотечних систем має передбачати надійний захист персональних даних користувачів та захист від несанкціонованого доступу.

Восьма вимога – адаптивний інтерфейс та доступність. Бібліотечні системи повинні мати зручний, інтуїтивно зрозумілий інтерфейс з можливістю адаптації для користувачів з обмеженими можливостями.

Рекомендації до стеку технологій. Для проєктування бібліотечних систем з використанням штучного інтелекту рекомендується використовувати сучасні мови програмування, такі як Python для обробки даних та розробки моделей машинного навчання, а також JavaScript для створення інтерактивного інтерфейсу. Інфраструктура повинна базуватися на хмарних рішеннях для забезпечення масштабованості та доступності, з використанням систем управління базами даних, таких як PostgreSQL для структурованих даних та Elasticsearch для швидкого пошуку й індексації. Бекенд-система має бути побудована на фреймворках Django або Flask для обробки запитів та забезпечення функціонування API, а фронтенд повинен забезпечувати зручний інтерфейс користувача за допомогою React або Vue.js. Важливо забезпечити інтеграцію інструментів для обробки природної мови, розпізнавання тексту та аналітики великих даних, а також впровадити механізми безпеки й автентифікації для захисту даних користувачів.

Висновки. Таким чином, впровадження штучного інтелекту у бібліотечні системи спрямоване на підвищення ефективності їх функціонування, оптимізацію процесів обробки та надання інформації, а також покращення обслуговування користувачів. Використання інтелектуальних алгоритмів забезпечує автоматизацію рутинних завдань, персоналізацію доступу до ресурсів та точний аналіз потреб відвідувачів.

Запропонований стек технологій дозволяє створювати сучасні,

масштабовані та безпечні інформаційні системи, що відповідають викликам цифрової епохи. Такі системи не лише забезпечують доступ до знань у нових форматах, а й сприяють збереженню культурної спадщини шляхом оцифровки та інтеграції архівних ресурсів у загальнодоступні бази даних.

Список використаних джерел:

1. Івашкевич О. Штучний інтелект в акустиці функціонування книгозбірень України. *Бібліотекознавство. Документознавство. Інформологія.* 2023. №2. С. 97–101.

2. Дятлюк І.С., Романюк О.В. Інтеграція сучасних технологій із програмним забезпеченням для ефективного управління ресурсами бібліотеки. *Матеріали LIII науково-технічної конференції підрозділів ВНТУ*, 2024. №22. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki2024/paper/view/20691.

ДОДАТОК В

Сертифікат