Міністерство освіти і науки України
Криворізький національний університет
Кафедра моделювання і програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти магістра

за напрямом підготовки 121 – Інженерія програмного забезпечення

На тему: Аналіз ефективності використання програмних засобів для обліку функціонування літературного фонду

Засвідчую, що в цій кваліфікаційній роботі немає запозичень із праць інших авторів без відповідних посилань.
Студент гр. ІПЗ–24м
_____/ Д. В. Подоляченко /

Керівник кваліфікаційної роботи _____/ Д. В. Швець /

Завідувач кафедрою _____/ А. М. Стрюк /

Кривий Ріг
2025

Криворізький національний університет

Факультет: Інформаційних технологій

Кафедра: Моделювання і програмного забезпечення

Ступінь вищої освіти: магістр

Спеціальність: 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедрою

_____ А. М. Стрюк

«____» _____ 2025 р.

ЗАВДАННЯ

на кваліфікаційну роботу

студенту групи ІПЗ–24м Подоляченко Денису Володимировичу

1. Тема: Аналіз ефективності використання програмних засобів для обліку функціонування літературного фонду затверджена наказом по КНУ № 204с від «10» квітня 2025 р.

2. Термін подання студентом закінченої роботи: «01» грудня 2025 р.

3. Вихідні дані по роботі: програмне забезпечення, що розроблюється, має забезпечити реалізацію вимог, необхідних для обліку функціонування літературного фонду.

4. Зміст пояснювальної записки (перелік питань, що їх треба розробити): виконати аналіз існуючих на ринку програмного забезпечення рішень, визначити основний функціонал системи, спроектувати систему для обліку функціонування літературного фонду, реалізувати програмне забезпечення, провести тестування розробленого додатку.

5. Перелік ілюстративного матеріалу: функціональна схема, блок–схема розробленого алгоритму програми, схема баз даних, знімки екранних форм додатку.

Календарний план:

№	Найменування етапів кваліфікаційної роботи	Термін виконання
1	Аналіз літературних джерел та огляд інтернет-ресурсів з заданої тематики	10.01.25 – 20.02.25
2	Пошук існуючих методів вирішення проблеми	21.02.25 – 12.03.25
3	Формулювання актуальності роботи і аналіз предметної області	13.03.25 – 27.03.25
4	Оформлення матеріалів першого розділу роботи	28.03.25 – 15.04.25
5	Науковий огляд проблеми	16.04.25 – 02.05.25
6	Оформлення матеріалів другого розділу роботи	03.05.25 – 18.05.25
7	Проектування програмного комплексу	19.05.25 – 06.06.25
8	Оформлення матеріалів третього розділу роботи	07.06.25 – 26.06.25
9	Розробка функціональної схеми та алгоритму програми	27.06.25 – 14.07.25
10	Розробка програмного забезпечення системи	15.07.25 – 30.09.25
11	Оформлення матеріалів четвертого розділу роботи	01.10.25 – 09.10.25
12	Аналіз економічної ефективності розробки	10.10.25 – 31.10.25
13	Оформлення пояснювальної записки	01.11.25 – 30.11.25

Дата видачі завдання: «09» січня 2025 р.

Студент: _____ / Д. В. Подоляченко /

Керівник роботи: _____ / Д. В. Швець /

РЕФЕРАТ

ЛІТЕРАТУРНИЙ ФОНД, КНИГА, ПРОГРАМНІ ЗАСОБИ, .NET FRAMEWORK, MICROSOFT SQL SERVER.

Пояснювальна записка: 81 с., 4 табл., 13 рис., 2 дод., 25 джерел.

Мета кваліфікаційної роботи: аналіз ефективності використання програмних засобів для обліку функціонування літературного фонду.

Об’єкт проектування: програмні засоби для обліку функціонування літературного фонду.

У теоретичній частині кваліфікаційної роботи проаналізовано ринок існуючих програмних рішень для вирішення проблеми обліку функціонування літературного фонду. Виконана оцінка існуючого програмного забезпечення, визначено його недоліки та переваги. Обґрунтовані актуальність роботи та сформульовані завдання дипломної роботи.

У практичній частині реалізовано функціональну схему розроблюваної системи, структуру даних, використаних в програмному забезпеченні, та алгоритм його роботи. Виконані проектування та розробка баз даних, спроектовано та створено інтерфейс додатку та програмну логіку системи для обліку функціонування літературного фонду. Проведено тестування розробленого додатку.

Розроблене програмне забезпечення може бути застосоване як для спрощення роботи невеликих літературних фондів, так і для обліку роботи більш крупних закладів.

ABSTRACT

LITERARY FUND, BOOK, SOFTWARE, .NET FRAMEWORK, MICROSOFT SQL SERVER.

Explanatory note: 81 p., 4 tables, 13 pics, 2 pp., 25 sources.

The aim of the qualifying work: analysis of the effectiveness of software tools for accounting for the functioning of the literary fund.

Design object: software tools for accounting for the functioning of the literary fund.

The theoretical part of the qualification work analyses the market of existing software solutions for solving the problem of accounting for the functioning of the literary fund. The evaluation of the existing software has performed, its shortcomings and advantages have been determined. The relevance of the work has substantiated and the tasks of the qualifying work have formulated.

In the practical part, the functional scheme of the developed system, the structure of the data used in the software, and the algorithm of its work has implemented. Database design and development have completed, and the application interface and system software logic for accounting for the functioning of the literary fund have designed and created. The developed application has tested.

The developed software can be used both to simplify the work of small literary funds and to keep track of the work of larger institutions.

ЗМІСТ

ВСТУП

Сьогодні все більше спостерігається перехід до споживання інформації через мережу Інтернет на відміну від звичного читання книг та роботи з літературою. Це зумовлюється низкою факторів, серед яких значна розповсюдженість доступу до інтернету, наявність безлічі ресурсів для завантаження літератури в цифровому форматі, значне збільшення швидкості доступу до інформації, коли знаходження потрібних відомостей пришвидшується в десятки разів.

Але тим не менш, читання традиційних паперових книжок знаходить своїх прихильників і сьогодні. Паперові книги користуються попитом через свою тактильну приємність читачеві та можливість нелінійного пошуку інформації, який доволі складно та затратно здійснювати при використанні електронного рідера. Будь-які енциклопедії чи довідники з великою кількістю малюнків теж зручніше вивчати саме в надрукованому варіанті. Те ж саме стосується дитячої літератури – в більшості випадків дітям цікаві саме кольорові друковані видання, аніж чорно-білі зображення на екранах електронних книжок.

Одними з найбільш популярних джерел отримання літератури традиційно залишаються бібліотеки або літературні фонди. Абоненти бібліотек зазвичай користуються послугами безкоштовно, що є доволі важливим фактором сьогодні, коли друковані видання коштують відносно дорого. В літературних фондах користувачі можуть отримати не лише книги, але й періодичні друковані видання (газети, журнали тощо), що дозволяє зекономити додатково ще й на періодиці.

Зазвичай користувачами бібліотеки є більш літні люди, які в силу тих чи інших причин не займаються опануванням сучасних технологій та не користуються Інтернетом, в той же час деякий відсоток молоді теж є активними читачами та звертається у подібні заклади. Окремими категоріями є наукові бібліотеки з вузькопрофільною літературою, до яких можуть

навідуватися науковці для отримання вузькоспеціалізованих видань, які було опубліковано невеликим примірником та які неможливо знайти в Інтернеті.

З огляду на це, літературні фонди як заклади є достатньо актуальними на сьогоднішній день та продовжують користуватися попитом, не дивлячись на щоденну цифровізацію життя.

1 ОГЛЯД ПИТАННЯ ОБЛІКУ ФУНКЦІОНУВАННЯ ЛІТЕРАТУРНИХ ФОНДІВ

1.1 Актуальність теми та огляд функціонування літературних фондів

Друковані видання продовжують користуватися значним попитом серед широких верств населення, не зважаючи на широке розповсюдження електронних гаджетів, що спрощують доступ до літератури та підвищують зручність читання. Робота з паперовими книжками та виданнями традиційно більш звична для більшості людей, в той же час вона зумовлює менше навантаження на зір на відміну від електронних пристроїв (смартфони, електроні рідери тощо), які призводять до втоми очей значно швидше через наявність активної підсвітки.

Іншим важливим моментом стосовно паперових книг є достатньо висока їх вартість на сьогоднішній день, яка зумовлена підвищенням витрат видавництв та друкарень через збільшення вартості сировини та оплати праці робітників цих закладів. Придбання щойно виданої книги стає все більш дорогим задоволенням, тому попит на таку продукцію поступово зменшується, що спричиняє закриття книжкових магазинів.

Одним із варіантів виходу з цієї ситуації – користуванням паперовими виданнями та мінімізацією витрат на них – є використання літературних фондів та подібного роду сервісів. Бібліотечні заклади періодично оновлюють свої фонди та проводять закупівлю свіжих видань, отже, користувачі за невелику щомісячну абонентську плату (а іноді навіть і безкоштовно) можуть отримувати доступ до читання книг у зручному для них форматі. Бібліотеки продовжують існувати повсюдно, їх зазвичай можна знайти в будь-якому районі міста, до того ж вони поділяються за своїми профілями (дитячі, дорослі, наукові, художні тощо), що дозволяє кожному читачеві знайти саме те місце,

де він може знайти цікавлячий його напрям літератури та насолоджуватися тими жанрами книг та періодики, що цікавлять саме його.

Основою бібліотеки є бібліотечний фонд, тобто сукупність наявних в закладі видань. Для того, щоб користувачі бібліотеки мали змогу користуватися добре сформованим бібліотечним фондом, його треба відповідним чином організувати. В свою чергу, для того, щоб наявні видання бібліотеки відповідали її направленості та інтересам користувачів, має бути відповідним чином проведений не лише процес комплектування фонду (тобто процес закупівлі видань), але й організація фонду. Організація фонду, в свою чергу, базується на комплексі процесів обліку, прийому, розміщення, зберігання та обробки видань. Належним чином сформований процес організації фонду бібліотеки дозволяє ефективно розв'язувати завдання активного використання видань закладу та забезпечення їх схоронності протягом значних проміжків часу [1-3].

Варто зазначити, що до складу бібліотечного фонду входять як неперіодичні, так і періодичні видання. Неперіодичними виданнями вважаються такі, що, як правило, випускаються видавцем однократно. Вони можуть мати вигляд брошур або книг. В той же час неперіодичні видання можуть випускатися у вигляді багатотомників, що однаково оформлюються та мають за наповненням логічно однорідний вміст. Кожний том видання маркується своїм власним порядковим номером.

Що стосується класифікації неперіодичних видань згідно їх призначенню, то можна виділити наступні види:

а) офіційні видання – це публіковані матеріали, що випускаються державними або суспільними організаціями. Вони зазвичай носять нормативний чи законодавчий характер. До них можна віднести закони, правила, описи винаходів, державні стандарти.

б) науково-популярні видання – це книги та інші матеріали, що описують науково-технічні досягнення у зрозумілій для широкого кола читачів формі та мають на меті зацікавити та заохотити їх до вивчення певних

напрямів науки. Зазвичай розраховані на читачів, що мають принаймні середню освіту, на відміну від спеціалізованих наукових видань, які потребують від читача серйозної підготовки з предмету, розглянутого у них.

в) наукові видання, як зазначалося вище, більше розраховані на вчених та спеціалістів у певній галузі. Однією з найважливіших функцій наукових видань є взаємне інформування вчених стосовно результатів наукових досліджень. Наукові видання можна поділити на вчені труди наукових класиків, монографії та труди узагальнюючого характеру. Також сюди можна додати матеріали наукових конференцій та наукових конгресів.

г) навчальні видання – містять в собі певним чином систематизовані відомості з конкретного напрямку науки і техніки. До них можна віднести підручники, що забезпечують викладення дисципліни згідно зазначеного плану; навчальні програми, що формують певний порядок вивчення будь-якої дисципліни; навчально-методичні посібники, що здебільшого видаються у вищих навчальних закладах та орієнтовані на студентів для допомоги їм в опануванні певного предмету; наглядні матеріали, що забезпечують візуальне сприйняття матеріалу (креслення, схеми, таблиці) та покращують його опанування

д) довідники – це видання, що містять в собі інформацію з різних галузей науки чи техніки, яка розміщена в них таким чином, щоб користувачеві було зручно її знайти. Серед довідникових видань можна виділити словники, які містять переліки словосполучень та слів з відповідними роз'ясненнями (виділяють орфографічні, тлумачні, термінологічні словники). Варто зазначити, що в словниках матеріали розміщуються в алфавітному порядку. Також до довідників відносять енциклопедії, які включають найбільш суттєву інформацію стосовно певних наукових напрямків та галузей знань, та зазвичай містять велику кількість ілюстративного матеріалу, що спрощує сприйняття інформації; універсальні енциклопедії, що пропонують більш структуровану подачу матеріалу, ніж звичайні енциклопедії; довідники, що містять велику

кількість даних у цифровому форматі та використовуються як вузькопрофільними спеціалістами, так і широким загалом користувачів.

Відповідно до розгляду періодичних видань, варто зазначити їх основну відмінність від неперіодичних – це регулярність їх випуску. По класифікації періодичних видань згідно їх цільовому призначенню можна виділити такі наступні види:

а) газети – друковані видання, що виходять щотижнево (в деяких випадках – щоденно, або протягом інших проміжків часу). Можуть мати регіональну направленість, тобто висвітлювати новини певного регіону країни, також можуть мати і спеціалізовану направленість, висвітлюючи певну галузь людської діяльності.

б) журнали – в більшості випадків мають галузеву направленість, але можуть поділятися на науково-популярні, літературні, художні, наукові, професійно-промислові.

в) періодичні збірники – здебільшого включають в себе тексти доповідей, наукові статті, тези наукових конференцій. Протягом календарного року випускається певна кількість номерів видання, яка є попередньо запланованою. В більшості випадків, періодичні збірники висвітлюють саме результати наукових досліджень [4-6].

Що стосується організації зберігання даних про літературний фонд, то для цього зазвичай застосовується алфавітний каталог або картотека, що являє собою велику шафу зі значною кількістю висувних поличок. У висувних поличках, відповідно, знаходяться картки з інформацією про наявні в бібліотеці видання. Всі полички алфавітного каталогу пронумеровані із зазначенням першого та останнього складу слова, на який може починатися назва видання (або прізвище автора), інформація про яке зберігається в картотеці цієї полички. Для зручності картки з даними про видання можуть бути розділені за допомогою відповідних довідникових карток чи роздільників. Це спрощує пошук потрібної книги, а наявність нумерації

поличок зумовлює пришвидшення їх розстановки на місця після завершення роботи з картотекою.

Таким чином, завдяки використанню алфавітного каталогу можна виявити наявність в бібліотеці тих чи інших видань та з'ясувати, чи є в наявності книги того чи іншого автора.

Є очевидним факт, що використання алфавітних каталогів для здійснення пошуку необхідних користувачеві видань потребує значних витрат часу. По-перше, необхідно знайти відповідну висувну поличку серед їх різноманіття, по-друге – зазвичай потрібно перебрати велику кількість карток до моменту, як буде знайдена потрібна.

З огляду на це, доцільним виглядає створення і використання саме електронного каталогу видань бібліотеки. Розробка і впровадження системи для обліку бібліотечного фонду дозволить підвищити надійність збереження інформації стосовно наявних в бібліотеці друкованих видань, спростить роботу бібліотекарів та підвищить ефективність їх роботи. Паралельне використання алфавітних каталогів разом з електронним може бути виправданим в якості дублюючої функції.

Таким чином, розробка програмних засобів для обліку функціонування літературного фонду є актуальним та важливим завданням, що потребує вирішення.

1.2 Існуючі програмні засоби для обліку функціонування літературного фонду

На ринку програмного забезпечення сьогодні пропонується ряд рішень, що мають на меті спрощення обліку бібліотечного фонду. Всі вони відрізняються за своїм функціональним наповненням, за можливістю інтеграції у ті чи інші платформи, за особливостями інтерфейсу та, відповідно, за вартістю використання.

Серед систем для автоматизації бібліотечної роботи вирізняються відкриті рішення з вільним кодом і комерційні хмарні платформи. Наприклад, Koha [7] є повнофункціональною системою ILS [8], яка підтримує каталогізацію, облік фондів, видачу/повернення, управління підписками, серійні видання, статистику, друк штрих-кодів та веб-інтерфейс для користувачів. Koha сумісна з бібліотечними стандартами - це забезпечує ефективний обмін метаданими та інтеграцію з іншими бібліотечними системами [9, 10].

За рахунок відкритості коду, Koha підходить для літературних фондів, які мають обмежений бюджет або потребують кастомізації - немає ліцензійних витрат, можна самостійно налаштувати систему або звернутись до компаній-партнерів для підтримки.

Водночас для ефективного впровадження та підтримки потрібні або власні IT-ресурси, або залучення фахівців, що може бути непросто для малих або сільських бібліотек.

З іншого боку, комерційні платформи - як Alma від Ex Libris [11] - забезпечують сучасний «хмарний» підхід: управління друкованими, електронними та цифровими ресурсами в єдиному середовищі, аналітику, автоматизацію робочих процесів, інтеграцію з ERP [12] чи освітніми системами, а також підтримку з боку постачальника.

Такий вибір зручний для великих академічних бібліотек з великим обсягом фондів і потребами в аналітиці. Однак він потребує постійних підписок, створює залежність від постачальника, а також обмежує гнучкість кастомізації.

Для невеликих літературних фондів, наприклад, шкільних - де ресурси обмежені, а пріоритетом є базове ведення фонду та простота в роботі - підходять легші й менш ресурсомісткі системи: вони забезпечують основні функції каталогізації, обліку, резервування та звітності з мінімальними системними вимогами. Хоча такі системи мають обмежений функціонал і

менше підходять для майбутнього масштабування або інтеграції зі складними сервісами, вони дають змогу швидко запускати роботу без значних витрат.

1.3 Висновки за результатами аналізу наявного ПЗ для обліку функціонування літературного фонду

Проведений аналіз існуючих систем для обліку роботи літературного фонду показав, що запропоновані програмні рішення різняться як за своїми функціональними можливостями, так і за вартістю впровадження, використання і обслуговування.

Серед наявних на ринку рішень присутні як хмарні сервіси, що забезпечують зберігання та обробку інформації на віддалених серверах, так і додатки, що функціонують безпосередньо на локальній системі користувача. Використання SaaS-рішень, що базуються на хмарній роботі, з одного боку вирішує проблему централізації даних та дозволяє здійснювати доступ до інформаційної системи з різних локацій, але в той же час і підвищує ризики несанкціонованого доступу до даних в результаті дій зловмисників. Також виникають ризики втрати даних через проблеми провайдеру хмарного сервісу, що можуть включати хакерську атаку чи відказ у роботі обладнання. Відповідно існує і небезпека компрометації чутливих даних (наприклад, приватної інформації користувачів бібліотеки) через зловживання службовим становищем співробітниками хмарного сервісу, які можуть розпоряджатися цією інформацією у власних цілях.

Аналізуючи цінову політику використання розглянутих програмних продуктів, варто зауважити, що частина з них є доволі вартісними як з огляду на придбання, інсталяцію та налаштування, так і з точки зору абонентського обслуговування протягом циклу експлуатації.

Таким чином, завдання розробки програмних засобів для обліку функціонування літературного фонду є на сьогодні важливим та актуальним.

2 ДОСЛІДЖЕННЯ ПИТАНЬ ЕФЕКТИВНОСТІ ОБЛІКУ ФУНКЦІОНУВАННЯ ЛІТЕРАТУРНОГО ФОНДУ

2.1 Аналіз проблематики та дослідження існуючих методик вимірювання ефективності обліку функціонування літературного фонду

У науковій літературі ефективність діяльності бібліотеки розглядають як багатовимірне явище, що поєднує операційну продуктивність (використання фонду, людських і фінансових ресурсів) та якісно-соціальний вплив (задоволеність користувачів, освітній і культурний ефект). Для адекватної оцінки цього явища дослідники використовують комбінації кількісних індикаторів, опитувальних інструментів і економіко-математичних методів, кожен із яких дає свою частину картини та компенсує обмеження інших підходів [13].

У міжнародній практиці одним із ключових документів, що структурує підхід до вимірювання продуктивності бібліотек, є керівництво IFLA «Performance Measurement in Libraries». Воно пропонує уніфікований набір індикаторів для різних типів бібліотек, включно з показниками для електронних сервісів і показниками витратної ефективності. Керівництво описує принципи вибору індикаторів, методи їх збору і приклади інтерпретації, наголошуючи на необхідності поєднання входів, виходів і результатів для формування збалансованої системи моніторингу. Це практичне видання слугує основою для багатьох подальших емпіричних досліджень і надає стандартний набір критеріїв для порівняння бібліотек у часі та між регіонами [14].

Традиційні кількісні індикатори, запозичені з практики IFLA та національних методичних рекомендацій, дають змогу відслідковувати динаміку: відвідуваність, оборот фонду, кількість видань на одного користувача, витрати на обслуговування одного відвідувача тощо. Такі індикатори зручні для щорічної звітності й бюджетного планування, але вони

не вимірюють безпосередньо сприйняття якості послуг або довготривалого соціального ефекту, тож для розуміння впливу потрібні додаткові інструменти [15].

Оцінка якості обслуговування в бібліотеках набула поширення завдяки адаптаціям сервіс-орієнтованих підходів. Однією з найвідоміших практик є LibQUAL+ - стандартизований веб-опитувальник, розроблений і поширюваний через мережу ARL. LibQUAL+ базується на емпірично виведених вимірах сприйняття якості і застосовує статистичну обробку відповідей для виявлення розривів між очікуваннями й досвідом користувачів. Інструмент пропонує процедури вибірки, тестування та інтерпретації результатів, роблячи можливим систематичний моніторинг сприйняття сервісу та порівняння між установами. У практиці LibQUAL+ часто використовують як доповнення до статистичних показників, щоб отримати глибше розуміння якості послуг з боку клієнтів [16].

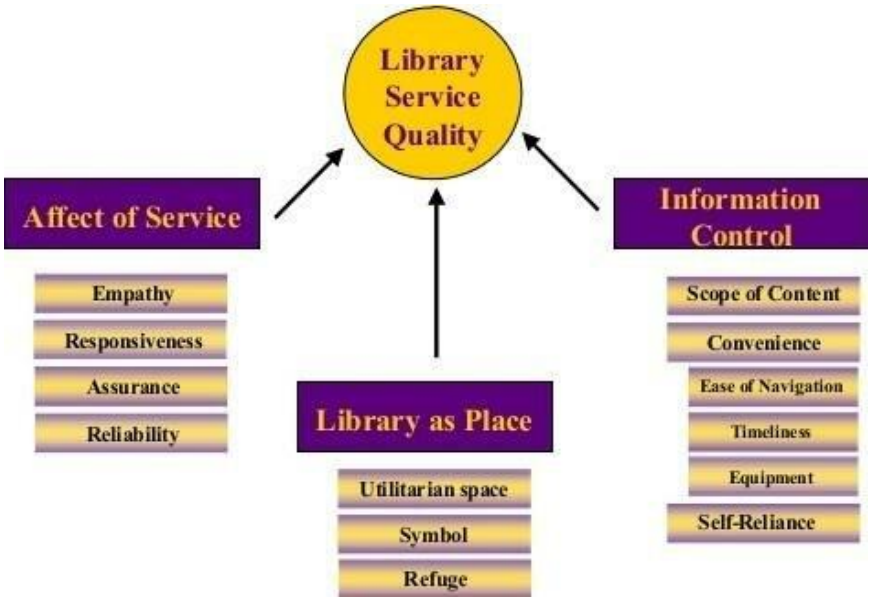


Рисунок 2.1 – Інфографіка LibQUAL+ [17]

Економіко-математичні методи, зокрема Data Envelopment Analysis (DEA), застосовуються для оцінки відносної продуктивності літературних фондів як множини одиниць прийняття рішень. DEA дозволяє одночасно

враховувати кілька входів і виходів і визначати «фронт виробництва», тобто літературні фонди, що працюють ефективно відносно інших. Застосування DEA в бібліотечній сфері дозволяє виявляти джерела неефективності, порівнювати організаційні моделі та моделювати можливі шляхи оптимізації витрат. Водночас метод має обмеження щодо оцінки якості виходів і вимагає ретельного підбору змінних; тому найкращі дослідження комбінують DEA з опитуваннями та аналізом контексту. Останні систематичні огляди підтверджують широке застосування DEA в академічних, публічних і спеціалізованих бібліотеках та виділяють потребу в стандартизації наборів входів/виходів для порівнянь на міжнародному рівні [18,19].

Вітчизняні методичні матеріали інтегрують міжнародні підходи з національними особливостями звітності та обліку. Науково-методичні посібники українських установ рекомендують поєднання стандартних статистичних показників із локально адаптованими опитуваннями, зокрема з урахуванням цифрових метрик (взаємодія через сайт, віддалений доступ до баз даних), оскільки сучасна цифровізація змінює співвідношення фізичної відвідуваності та загального обсягу взаємодій користувачів з послугами бібліотеки. Українські дослідження також звертають увагу на нерівномірність участі різних соціальних груп у бібліотечному житті як фактор, що потребує цілеспрямованих програм розширення охоплення [20].

У практичному плануванні дослідження доцільно формувати мультипарадигмовий дизайн [15], де стандартні показники продуктивності поєднані з опитуванням (наприклад, адаптованою формою LibQUAL+) та DEA-аналізом для порівняльної оцінки. Такий підхід дозволяє отримати не лише числову картину ресурсного використання, а й якісну інтерпретацію результатів і рекомендації щодо оптимізації. Для валідності висновків необхідна увага до вибірки в опитуваннях, коректний добір входів і виходів для DEA (з урахуванням локального контексту) та системний збір цифрових метрик.

2.2 Розробка методики аналізу ефективності обліку функціонування літературного фонду

В той же час варто зазначити, що оцінювання результативності роботи бібліотек потребує не лише аналізу якісних характеристик, таких як задоволеність користувачів чи рівень доступності інформаційних ресурсів, а й кількісного обчислення економічних параметрів. Це дає змогу представити діяльність бібліотеки у формі вимірюваного показника, який відображає співвідношення отриманих переваг і витрат на їх досягнення протягом певного періоду. Одним із найбільш зручних підходів є побудова математичної моделі, що інтерпретує ефективність як чисту щомісячну вигоду, виражену в грошовому вимірі.

У межах цієї роботи пропонується модель, яка базується на ідеї монетизації функціональних результатів діяльності літературного фонду. До ключових змінних включено кількість наявних книжкових найменувань, загальний обсяг примірників, кількість читачів за місяць, динаміку видач за той самий період та кадрові ресурси установи. Зазначені показники дозволяють не лише описати структуру фонду та інтенсивність його використання, а й перевести ці процеси у форму, що відображає грошову цінність вироблених послуг.

У модель закладено припущення, за яким кожна видача книги має умовну економічну вартість. Її можна інтерпретувати як гіпотетичну суму, яку користувач сплатив би за аналогічний доступ у комерційній інформаційній системі. Аналогічним чином трактуються індивідуальні відвідування читачів: кожне з них має власну цінність, що дорівнює середній вартості користування інформаційними ресурсами на ринку цифрових бібліотек. Сукупність цих величин дає змогу сформувати оцінку місячного «валового ефекту» від діяльності установи.

Поряд із вигодами враховуються витрати, які закладені у функціонування бібліотеки. Насамперед це фінансування оплати праці персоналу, що становить значну частку загального бюджету. Додатково

враховується амортизація книжкових ресурсів, оскільки кожен примірник має кінцевий строк використання, протягом якого його цінність поступово зменшується. Розподіл вартості фонду на строк експлуатації дозволяє оцінити задачу в економічних термінах і забезпечити коректність порівняння місячних витрат і вигод.

З урахуванням цих положень ефективність роботи літературного фонду у місячному вимірі визначається формулою:

$$E_m = p_L \cdot L + p_R \cdot R - w_s \cdot S - \frac{P_{avr} \cdot C}{T_{life}} \quad (2.1)$$

де E_m - економічна ефективність літературного фонду у розрахунку на місяць;

p_L - ринкова або умовна вартість однієї видачі;

L - кількість видач за місяць;

p_R - монетизована оцінка одного читача;

R - кількість читачів за місяць;

w_S - середня місячна вартість утримання одного працівника;

S - кількість працівників бібліотеки;

P_{avg} - середня вартість одного примірника книжки;

C - загальна кількість примірників;

T_{life} - середній строк експлуатації книжкового фонду, поданий у місяцях.

Структура рівняння охоплює ключові економічні компоненти. Перша частина формули відображає користь, що продукується бібліотекою у вигляді інформаційних послуг. Друга частина виражає витрати на забезпечення цих послуг, включно з кадровими ресурсами та матеріально-технічною базою. Таким чином, модель є балансом між обсягом фактично створеної користі та витратами, необхідними для її досягнення.

Змістовною перевагою цієї моделі є її прозорість. Вона побудована таким чином, що кожна змінна має чіткий фізичний та економічний сенс, а сам процес обчислення легко відтворити для будь-якої бібліотеки незалежно від

масштабу. Гнучкість моделі дозволяє адаптувати її до різних сценаріїв. Наприклад, установи, де основний акцент робиться на електронні ресурси, можуть скоригувати вагові коефіцієнти або замінити частину параметрів відповідно до своїх операційних процесів. Бібліотеки, орієнтовані на активну комунікацію з читачами, можуть застосовувати збільшене значення параметра p_R або використовувати окремі категорії відвідувачів.

Важливою перевагою є можливість оцінювання показника в динаміці. Якщо протягом кількох місяців або років обраховувати E_m, то тренд дозволить виявити залежності між розвитком фонду, змінами кадрової політики чи зростанням читацької активності. Фактично, модель може працювати як інструмент управлінської аналітики, що орієнтований на стратегічні рішення - від оптимізації фінансування до моделювання різних сценаріїв розвитку.

Загалом, запропонована формула дає змогу перетворити багатовимірну діяльність літературного фонду на числовий показник, що відображає її економічну спроможність та ефективність. Такий підхід поєднує простоту розрахунку та аналітичну глибину, зберігаючи при цьому високий рівень інтерпретованості результатів.

3 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОБЛІКУ ФУНКЦІОНУВАННЯ ЛІТЕРАТУРНОГО ФОНДУ

3.1 Функціональні можливості системи

Програмні засоби для обліку функціонування літературного фонду мають на меті полегшити роботу бібліотекарам та спростити облік виданих читачам друкованих видань, забезпечити облік книжок та періодики на полицях бібліотеки, зробити можливим автоматизоване зберігання даних читачів та створити можливості для зберігання іншої важливої інформації, що спростить роботу працівників літературного фонду та підвищить його надійність.

Для ефективного функціонування зазначених програмних засобів доцільно запровадити підтримку ними наступних можливостей:

– урахування наявної в бібліотеці колекції книжок, періодичних видань та інших видів друкованої продукції з зазначенням наступних їх параметрів та властивостей:

а) найменування друкованого видання;

б) зазначення авторства друкованого видання;

в) компанії, що видала книгу;

г) рік публікації;

д) оформлення переплетення книги;

е) мова публікації книги чи друкованого видання;

ж) міжнародний стандартний книжний номер друкованого видання;

з) об’єм друкованого видання;

і) кількість екземплярів книги чи друкованого видання в бібліотеці;

– облік виданих читачеві книг чи інших друкованих видань з реєстрацією даних про:

а) час видачі видання на руки;

б) день видачі;

в) найменування виданої літератури;

г) видану кількість книг читачеві;

д) працівника бібліотеки, що видав літературу читачеві;

е) інформації про читача, який взяв в користування друковане видання;

– здійснення реєстрації надходжень друкованих видань до літературного фонду зі збереженням інформації про:

а) час надходження літератури;

б) дату надходження літератури;

в) найменування книг, що надійшли до бібліотеки;

г) кількість друкованих видань, що надійшли в цій партії;

д) працівника бібліотеки, що зареєстрував отримане друковане видання в інформаційній системі;

– облік відомостей про працівників бібліотеки:

а) ПІБ співробітника бібліотеки;

б) контактна інформація співробітника (в першу чергу телефонний номер);

в) е-мейл адреса співробітника бібліотеки;

г) дата народження працівника;

д) інша інформація, що може бути корисна;

– облік даних про читачів бібліотеки:

а) ПІБ користувача бібліотеки;

б) контактна інформація читача;

в) е-мейл читача;

г) дата народження читача;

д) інша інформація, що може бути корисна.

Додатково необхідно реалізувати можливість пошуку інформації в переліку друкованих видань для швидкого знаходження необхідних користувачеві відомостей.

Сукупність зазначених вимог при їх реалізації забезпечить необхідну функціональну наповненість програмних засобів для обліку функціонування літературного фонду.

2.2 Вимоги до конфігурації системи для експлуатації програмних засобів

Програмні засоби для обліку функціонування літературного фонду мають працювати на комп'ютерах, що оснащені згідно із таблицею 2.1.

Таблиця 2.1 – Апаратні вимоги до системи

Тип	Діапазон вимог
Встановлена ОС	від Windows 8 до Windows 11 (можливе використання серверних версій)
Найменування ЦП та тактова частота	AMD або Intel (тактова частота більша за 2ГГц)
Об'єм ОЗП	4 Гб
Графічний адаптер	Інтегрований (не менше 64Мб відеопам'яті)
Екран	Не менше, ніж 1024x768
Вільне місце на жорсткому диску	200 Мб
Периферія	Клавіатура, миша

Розробка зазначеного програмного забезпечення здійснюватиметься з використанням об’єктно-орієнтованої мови програмування високого рівня C# [21,22]. Ця мова добре зарекомендувала себе, розробка з її використанням зазвичай є швидкою та ефективною, особливо при застосуванні рідного середовища Visual Studio [23]. Додатковою вимогою при створенні додатків на мові C# є використання програмної платформи .NET [24,25], що дозволяє пришвидшити процес розробки програмного забезпечення та спростити його.

4 РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ ДЛЯ ОБЛІКУ ФУНКЦІОНУВАННЯ ЛІТЕРАТУРНОГО ФОНДУ

4.1 Функціональна схема програмних засобів для обліку функціонування літературного фонду

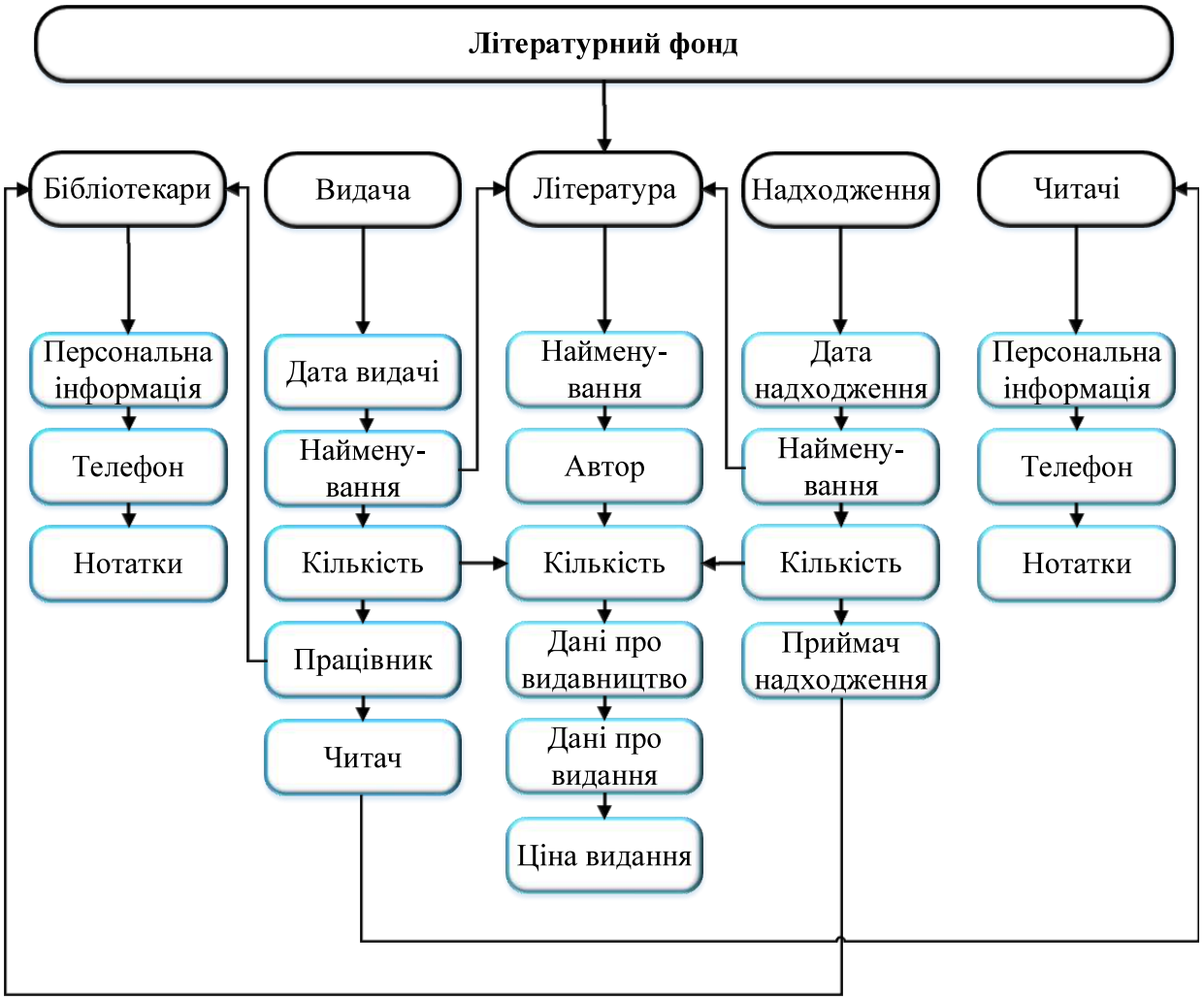


Рисунок 4.1 – Функціональна схема

Рисунок 4.1 демонструє спроектовану функціональну схему програмного забезпечення для обліку функціонування літературного фонду. Як видно з ілюстрації, система реалізує п’ять напрямків взаємодії з додатком – «Література», «Видача», «Бібліотекари», «Читачі», «Надходження». Позиція «Література» відповідає за роботу з наявними в бібліотеці книгами та іншими

видами видань, «Видача» забезпечує зберігання даних щодо книг, які видані з літературного фонду на руки читачам, «Бібліотекари» - дає змогу оперувати відомостями щодо бібліотекарів та іншого персоналу закладу, «Читачі», відповідно, дають змогу зберігати базу даних клієнтів бібліотеки, а «Надходження» забезпечують реєстрацію нових друкованих видань.

Перелік категорій по кожному з напрямків роботи з додатком детально наведений на рисунку 4.1 та відповідає сформульованим в попередньому підрозділі переліку функціональних можливостей системи, що проектується.

Відповідно, і взаємодії між тими чи іншими сутностями для кожного напрямку відображені на ілюстрації 4.1.

Таким чином, функціональна схема системи ілюструє відповідність проекту сформульованим попередньо вимогам.

4.2 База даних системи

Програмні засоби для обліку функціонування літературного фонду функціонують на базі СУБД «Microsoft SQL Server». База даних додатку містить в своєму складі п’ять таблиць. Зазначені таблиці є взаємопов’язаними за рахунок використання ключових полів. В перелік таблиць входять наступні:

- Literatura (Література);
- Vydacha (Видача);
- Pracivnyki (Працівники);
- Nadhodzhenia (Надходження);
- Chytachi (Читачі).

Таблиця «Literatura» забезпечує зберігання інформації про наявні в бібліотеці друковані видання; у таблиці «Vydacha» реалізоване збереження інформації про видачу літератури читачам бібліотеки; у «Pracivnyki» зберігаються відомості про бібліотекарів та інших працівників закладу; «Chytachi» відповідає за збереження інформації стосовно читачів бібліотеки; в таблиці «Nadhodzhenia» розміщено дані про надходження друкованих видань до бібліотечного фонду.

4.3 Алгоритм роботи програмних засобів

Рисунок 4.2 ілюструє алгоритм роботи програмних засобів для обліку функціонування літературного фонду.

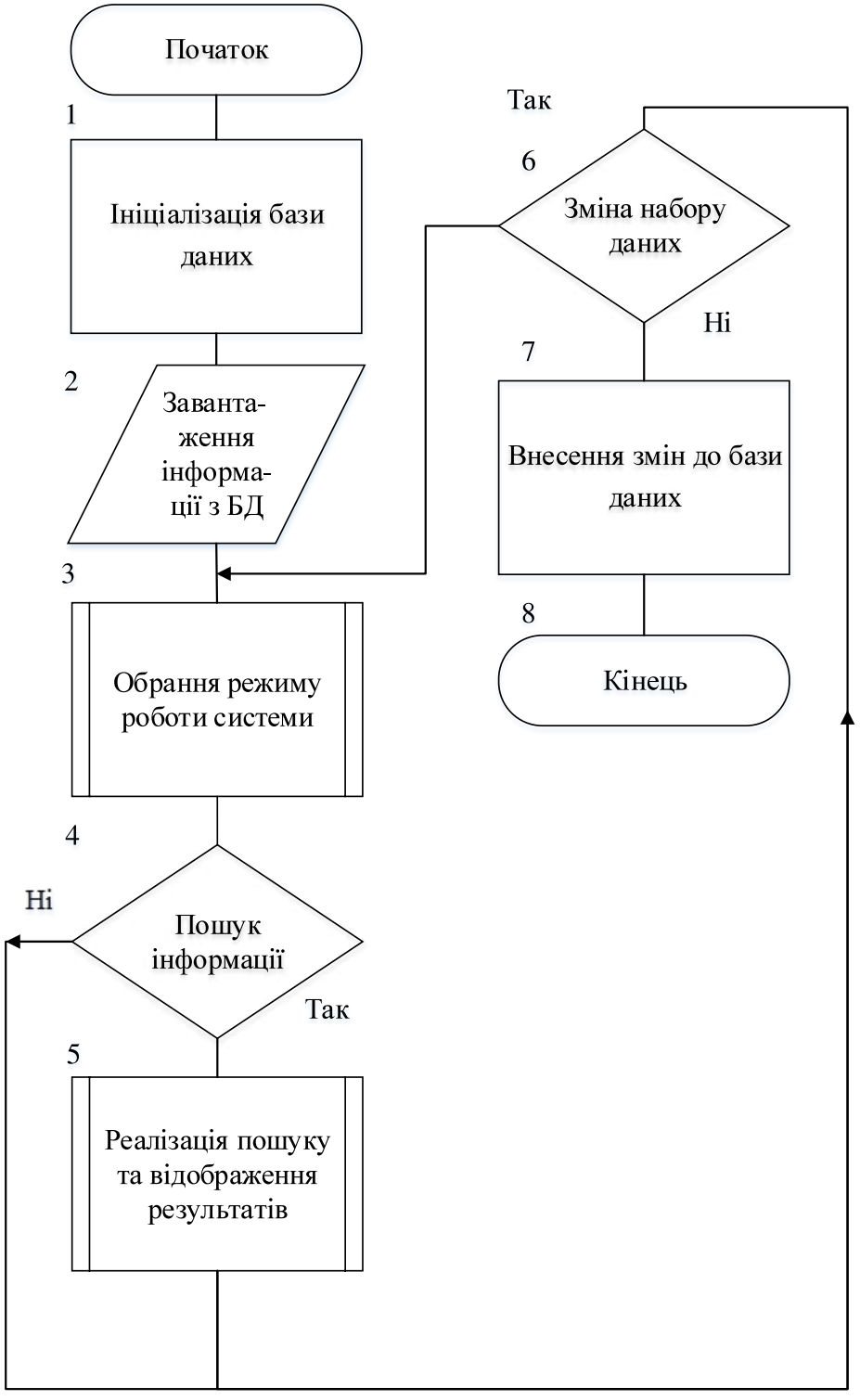


Рисунок 4.2 – Алгоритм роботи програмних засобів

Програмне забезпечення працює за наступним сценарієм: після запуску додатку в першу чергу відбувається звернення до бази даних та підключення до неї. За замовчуванням демонструється вікно роботи з літературним фондом, тобто наявною в бібліотеці літературою. Бібліотекар може працювати з цим списком в режимі перегляду даних, вносити необхідні зміни та видаляти неактуальні записи.

Також бібліотекар може змінити режими взаємодії з додатком на наступні:

– робота з переліком надходжень друкованої продукції до літературного фонду;

– реєстрація видачі книг та інших друкованих видань абонентам бібліотеки;

– облік інформації про працівників бібліотеки (бібліотекарів та допоміжного персоналу);

– реєстрація інформації про читачів бібліотеки (контактних та особистих даних);

– перегляд та редагування переліку видань, наявних в літературному фонді (режим за замовчуванням).

В системі також можливе використання опції пошуку, яка дозволяє виконувати фільтрування записів в таблиці згідно заданому пошуковому запиту.

При завершенні роботи з програмним забезпеченням, у базу даних вносяться зареєстровані в додатку зміни, та головне вікно додатку закривається.

Рисунок 4.3 ілюструє алгоритм реалізації зміни режиму програмного забезпечення. При запуску додатку відбувається формування таблиці з інформацією у головному вікні програми (за замовчуванням, це перелік літератури, наявної в літературному фонді), а після цього оператор системи може перемикатися на інші режими.

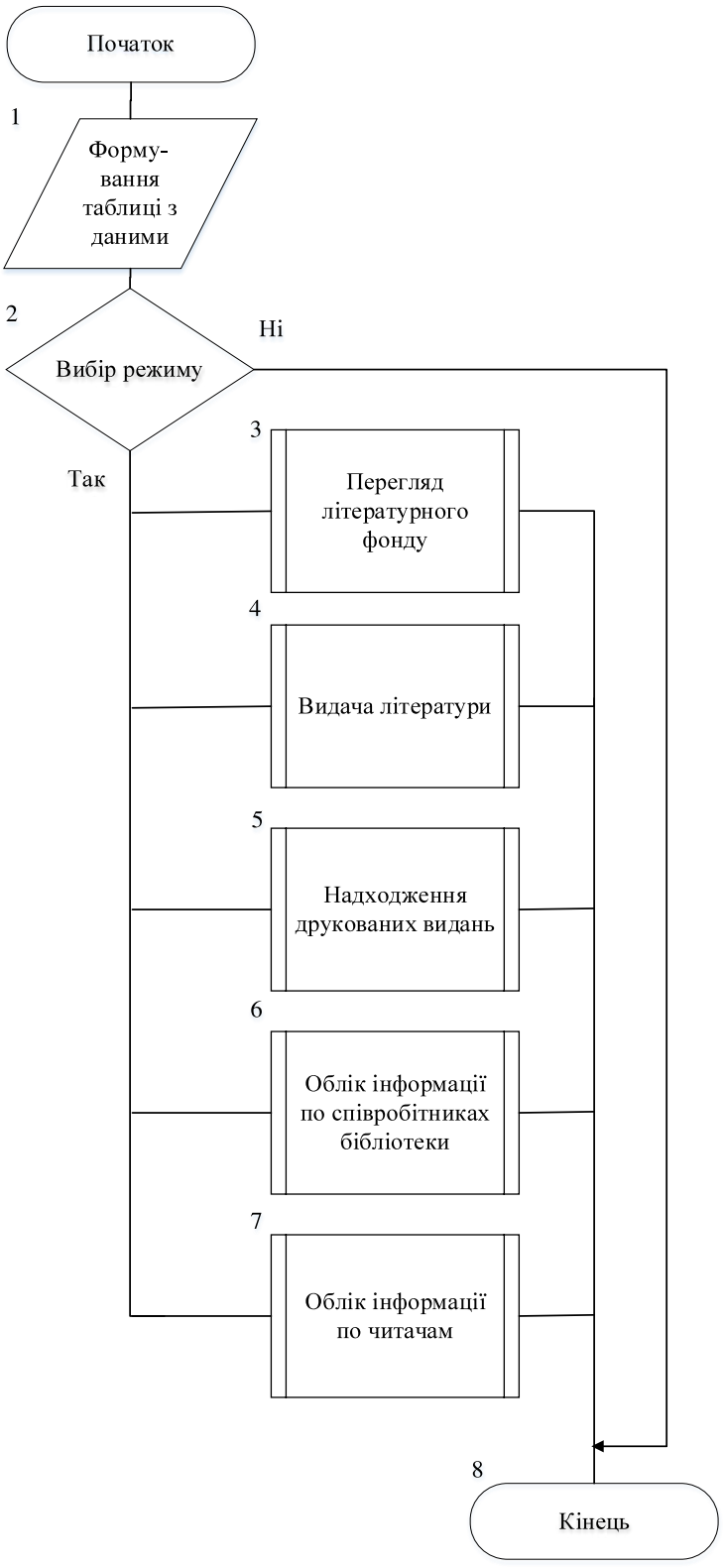
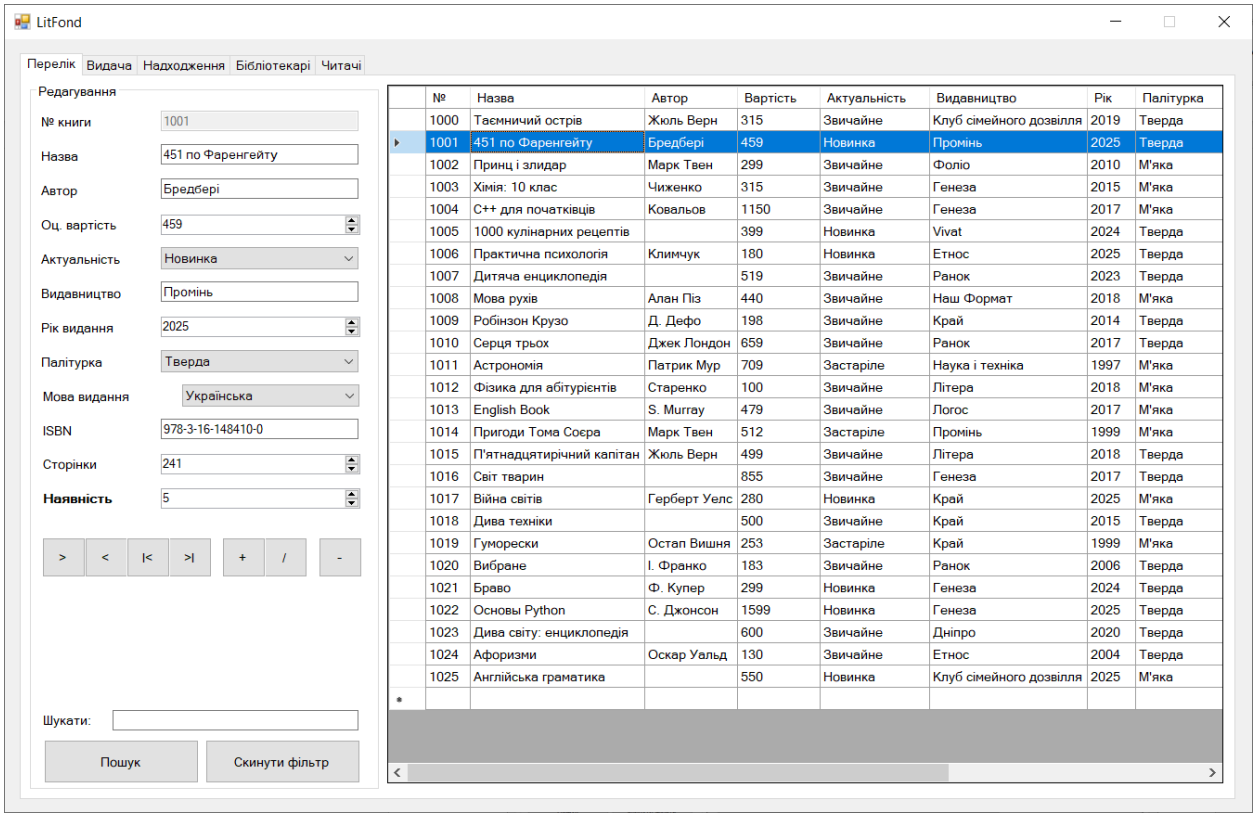


Рисунок 4.3 – Алгоритм вибору режиму

Проілюстровані алгоритми роботи програмних засобів для обліку функціонування літературного фонду демонструють шляхи їх використання для вирішення поставлених завдань обліку бібліотечної роботи.

4.4 Розроблений програмний продукт

Після запуску програмних засобів для обліку функціонування літературного фонду відображується головне вікно системи, що продемонстровано на рисунку 4.4. Воно містить в собі панель перемикання режимів взаємодії з додатком, таблицю для відображення інформації стосовно поточного режиму роботи, області редагування інформації, зони навігації і редагування та, відповідно, зони пошуку.



№	Назва	Автор	Вартість	Актуальність	Видавництво	Рік	Палітурка
1000	Таємничий острів	Жюль Верн	315	Звичайне	Клуб сімейного дозвілля	2019	Тверда
1001	451 по Фаренгейту	Бредбері	459	Новинка	Промінь	2025	Тверда
1002	Принц і злидар	Марк Твен	299	Звичайне	Фоліо	2010	М'яка
1003	Хімія: 10 клас	Чиженко	315	Звичайне	Генеза	2015	М'яка
1004	C++ для початківців	Ковальов	1150	Звичайне	Генеза	2017	М'яка
1005	1000 кулінарних рецептів		399	Новинка	Vivat	2024	Тверда
1006	Практична психологія	Климчук	180	Новинка	Етнос	2025	Тверда
1007	Дитяча енциклопедія		519	Звичайне	Ранок	2023	Тверда
1008	Мова рухів	Алан Піз	440	Звичайне	Наш Формат	2018	М'яка
1009	Робінзон Крузо	Д. Дефо	198	Звичайне	Край	2014	Тверда
1010	Серця трьох	Джек Лондон	659	Звичайне	Ранок	2017	Тверда
1011	Астрономія	Патрик Мур	709	Застаріле	Наука і техніка	1997	М'яка
1012	Фізика для абітурієнтів	Стаценко	100	Звичайне	Літера	2018	М'яка
1013	English Book	S. Murray	479	Звичайне	Логос	2017	М'яка
1014	Пригоди Тома Соєра	Марк Твен	512	Застаріле	Промінь	1999	М'яка
1015	П'ятнадцятирічний капітан	Жюль Верн	499	Звичайне	Літера	2018	Тверда
1016	Світ тварин		855	Звичайне	Генеза	2017	Тверда
1017	Війна світів	Герберт Уелс	280	Новинка	Край	2025	М'яка
1018	Дива техніки		500	Звичайне	Край	2015	Тверда
1019	Гуморески	Остап Вишня	253	Застаріле	Край	1999	М'яка
1020	Вибране	І. Франко	183	Звичайне	Ранок	2006	Тверда
1021	Браво	Ф. Купер	299	Новинка	Генеза	2024	Тверда
1022	Основы Python	С. Джонсон	1599	Новинка	Генеза	2025	Тверда
1023	Дива світу: енциклопедія		600	Звичайне	Дніпро	2020	Тверда
1024	Афоризми	Оскар Уальд	130	Звичайне	Етнос	2004	Тверда
1025	Англійська граматика		550	Новинка	Клуб сімейного дозвілля	2025	М'яка

Рисунок 4.4 – Головне вікно інформаційної системи, що демонструється після її запуску

Інформаційна таблиця є достатньо широкою, тому для повноцінного її використання необхідно використовувати прокрутку (рисунок 4.5).

Панель перемикання режимів містить у своєму складі 5 елементів – це вкладки «Перелік», «Видача», «Надходження», «Бібліотекари» та «Читачі». Активація відповідного режиму призводить до завантаження необхідних

наборів даних до таблиць, що дає змогу опрацьовувати необхідні користувачеві дані.

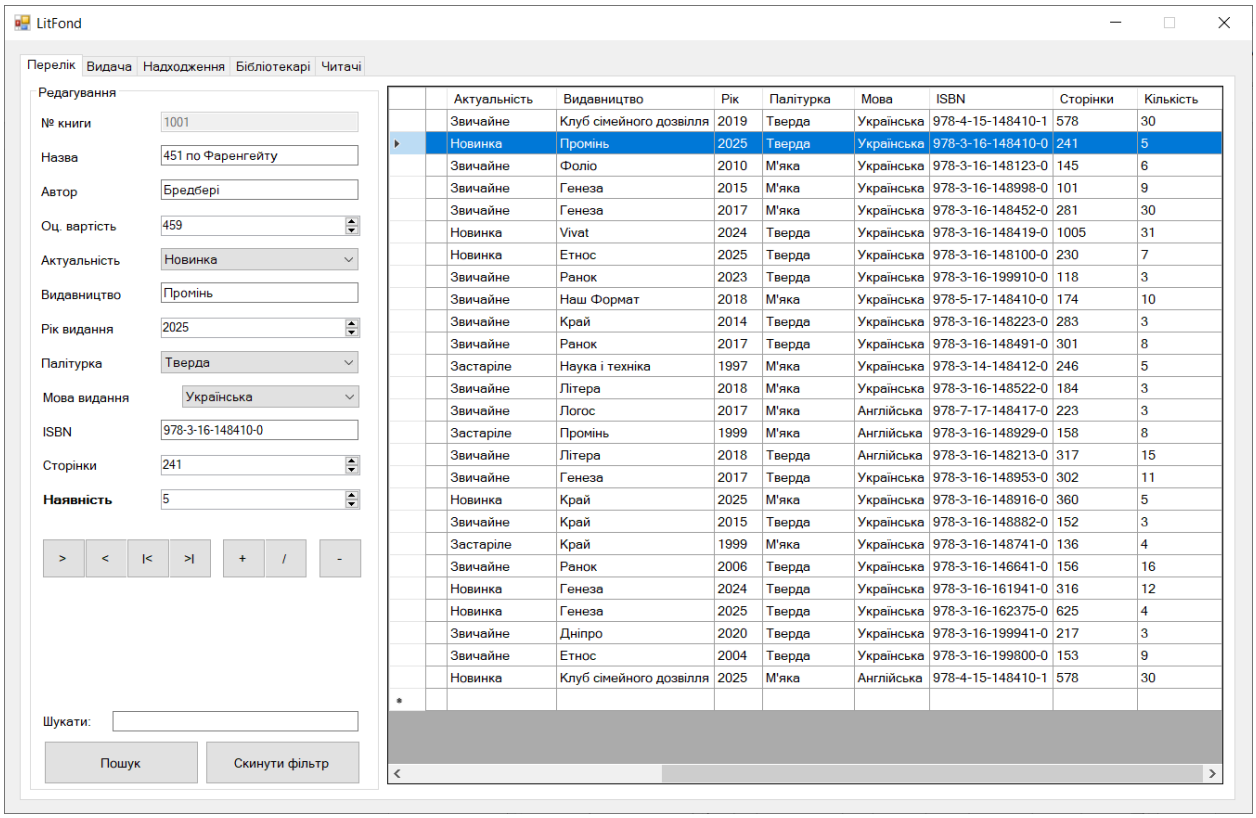


Рисунок 4.5 – Головне вікно інформаційної системи (прокрутка інформаційної таблиці)

З лівої сторони вікна розташована панель для внесення та редагування даних в таблиці. Вона дозволяє правити низку властивостей запису, серед яких найменування друкованого видання, його автори, страхова вартість, назва видавництва, тип перепльоту та інші (рисунок 4.6).

Для додавання нового запису необхідно обрати в нижній частині таблиці порожній рядок та заповнити даними поля панелі внесення даних. Далі необхідно натиснути кнопку «Додати» (+). Для редагування, відповідно, виділяється потрібний рядок, редагуються дані в панелі та натискається кнопка «Редагувати» (/). Для видалення зайвого друкованого видання виділяється строка з ним та натискається кнопка «Видалити» (-).

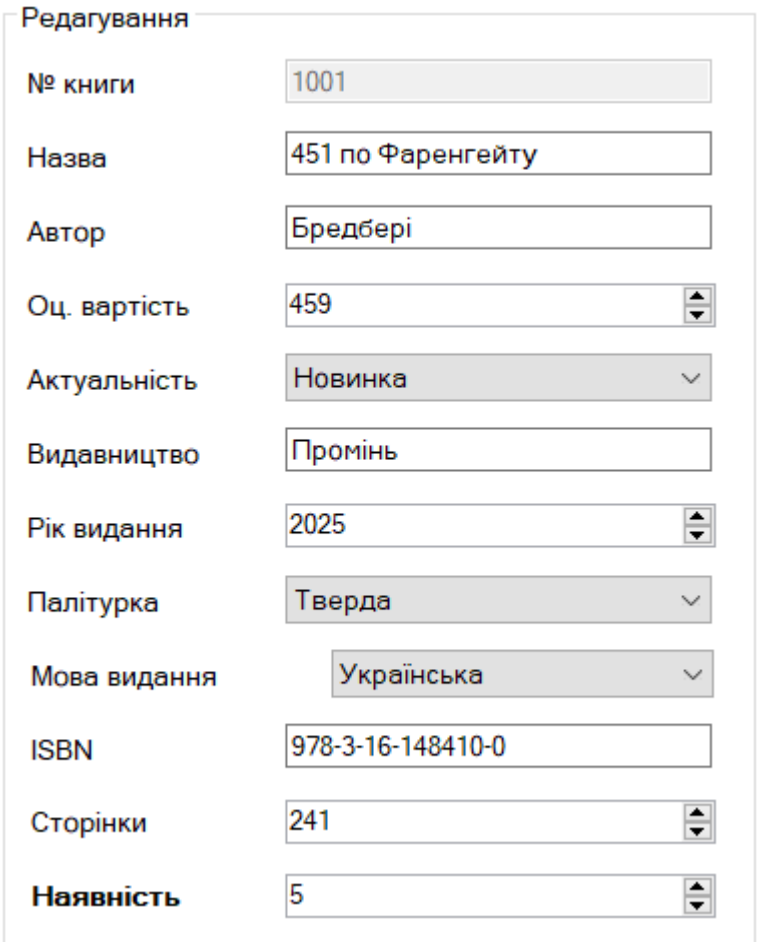


Рисунок 4.6 – Панель внесення даних та їх редагування

Панель редагування та навігації наведена на рисунку 4.7. Перші чотири кнопки дозволяють переміщуватися по інформаційній таблиці, останні три – правити вміст таблиці.

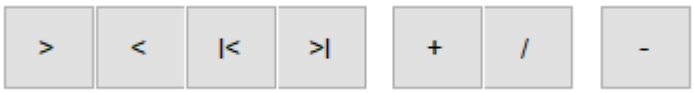


Рисунок 4.7 – Кнопки редагування даних та навігації

Панель пошуку представлена на рисунку 4.8. Робота з нею інтуїтивно зрозуміла.

Рисунок 4.8 – Елементи для реалізації пошуку в таблиці

Друга вкладка інформаційної системи «Видача» містить в собі дві інформаційні таблиці, перша з яких демонструє наявні в бібліотеці друковані видання («Наявність»), друга – показує, які з книг були видані читачам бібліотеки («Видано»). Відповідно, для реєстрації видачі друкованого видання користувачеві бібліотеки, необхідно обрати потрібну книгу в таблиці «Наявність», виділити її, та натиснути кнопку «Видати». Після цього в таблиці «Видано» буде сформовано запис, який свідчитиме про видачу видання читачеві. Вкладка «Видача» зображена на рисунку 4.9.

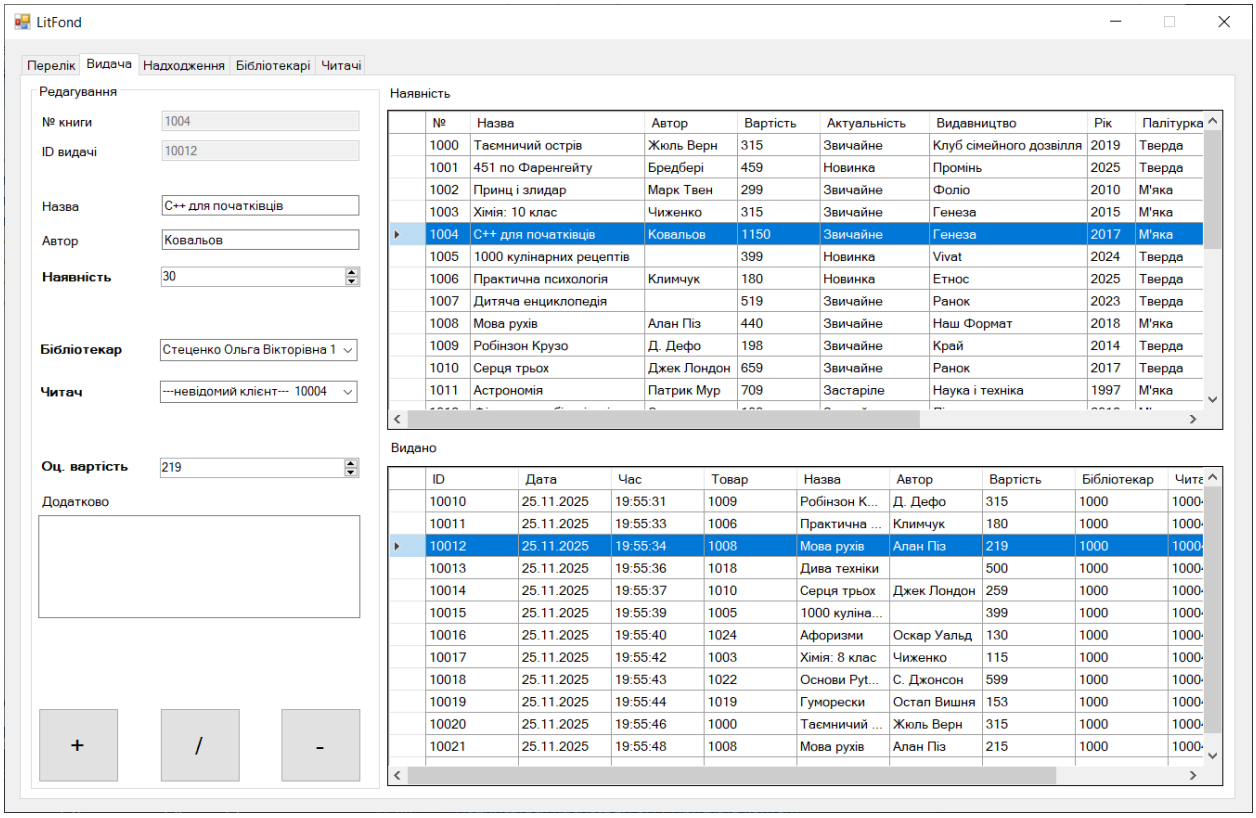


Рисунок 4.9 – Вкладка «Видача»

Третя вкладка «Надходження» певною мірою схожа за принципом свого функціонування зі вкладкою «Видача». Вона також викликає вікно, що складається з двох інформаційних таблиць – «Наявність» та «Поставка». За аналогією з попередньою вкладкою, перша таблиця свідчить про наповненість літературного фонду тими чи іншими виданнями, друга – являє собою реєстр надходжень, тобто забезпечує облік поставлених до бібліотеки друкованих видань.

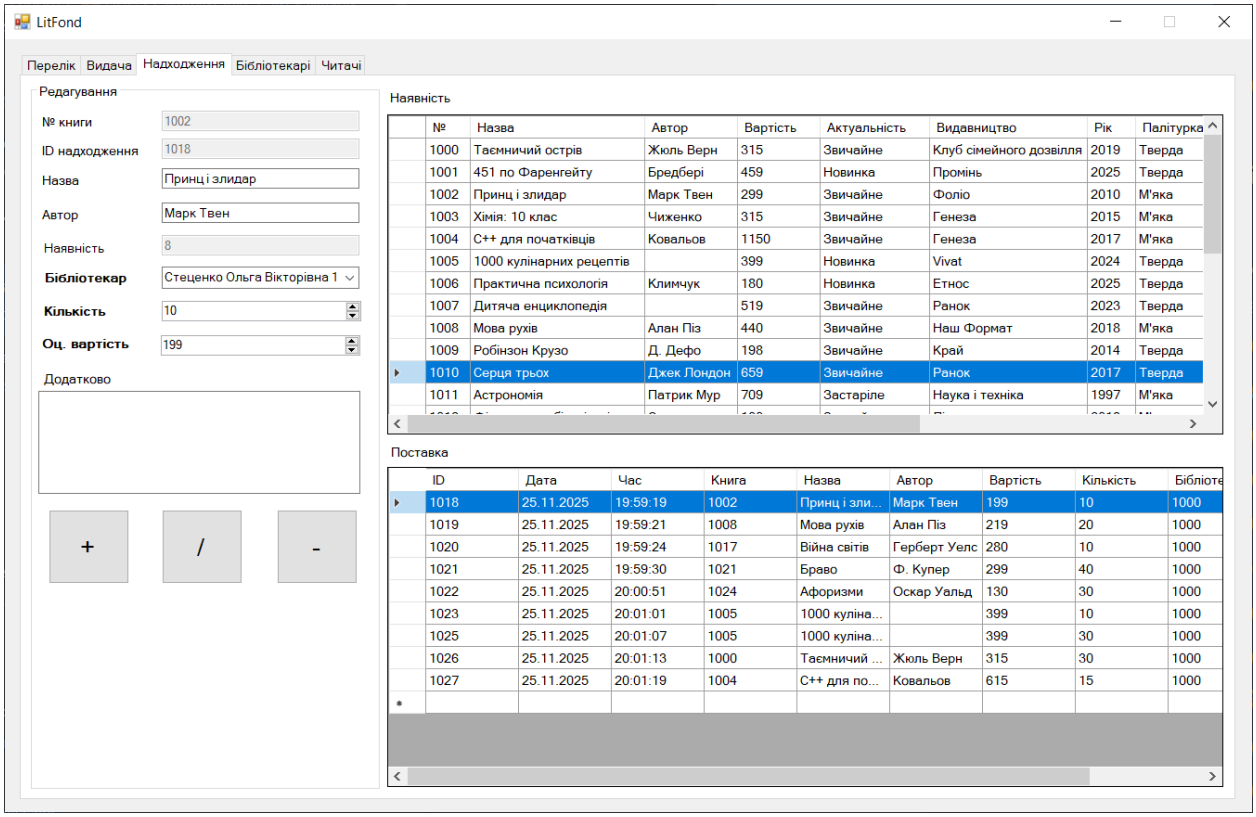


Рисунок 4.10 – Вкладка «Надходження»

Четверта вкладка «Бібліотекари» реалізує зберігання інформації про бібліотекарів та іншого персоналу закладу. Вона дає можливість зберігати основну інформацію про працівників, серед якої прізвище, ім’я, по батькові (персональні дані), дата народження, контакти та інші речі. Зовнішній вигляд четвертої вкладки зображено на рисунку 4.11.

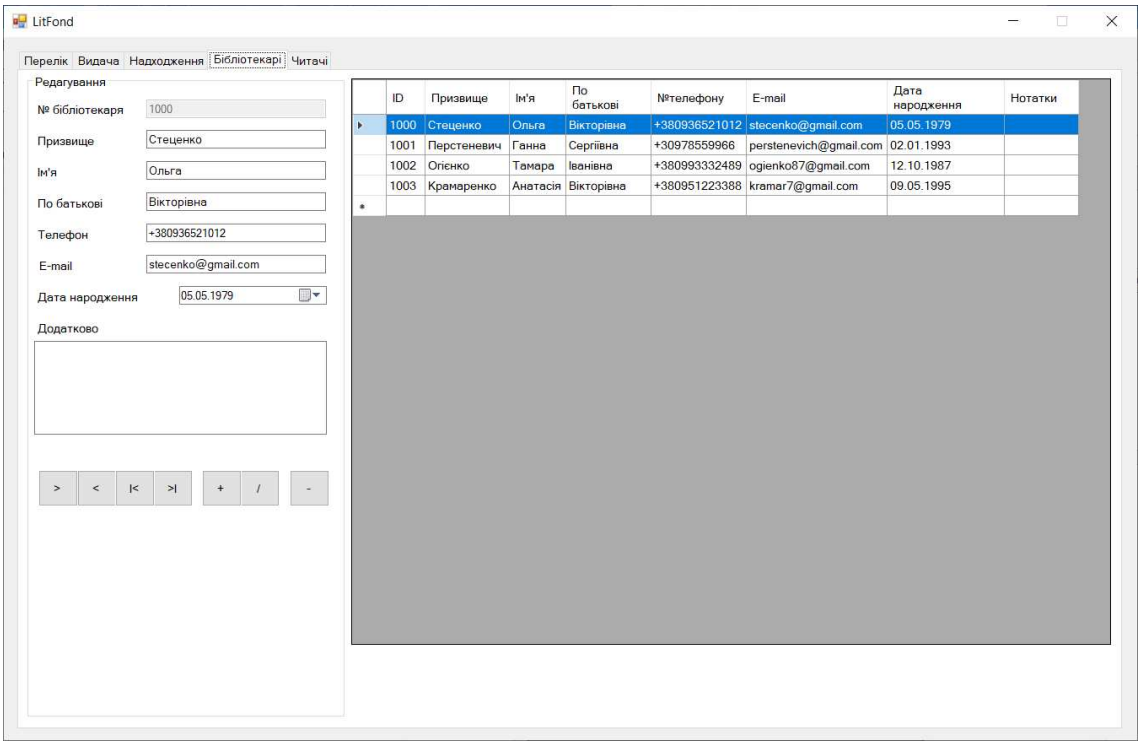


Рисунок 4.11 – Вкладка «Бібліотекари»

На п’ятій вкладці розміщено інформацію про читачів бібліотеки. На рисунку 4.12 наведено вміст вкладки «Читачі», що реалізує зберігання даних про клієнтів бібліотеки.

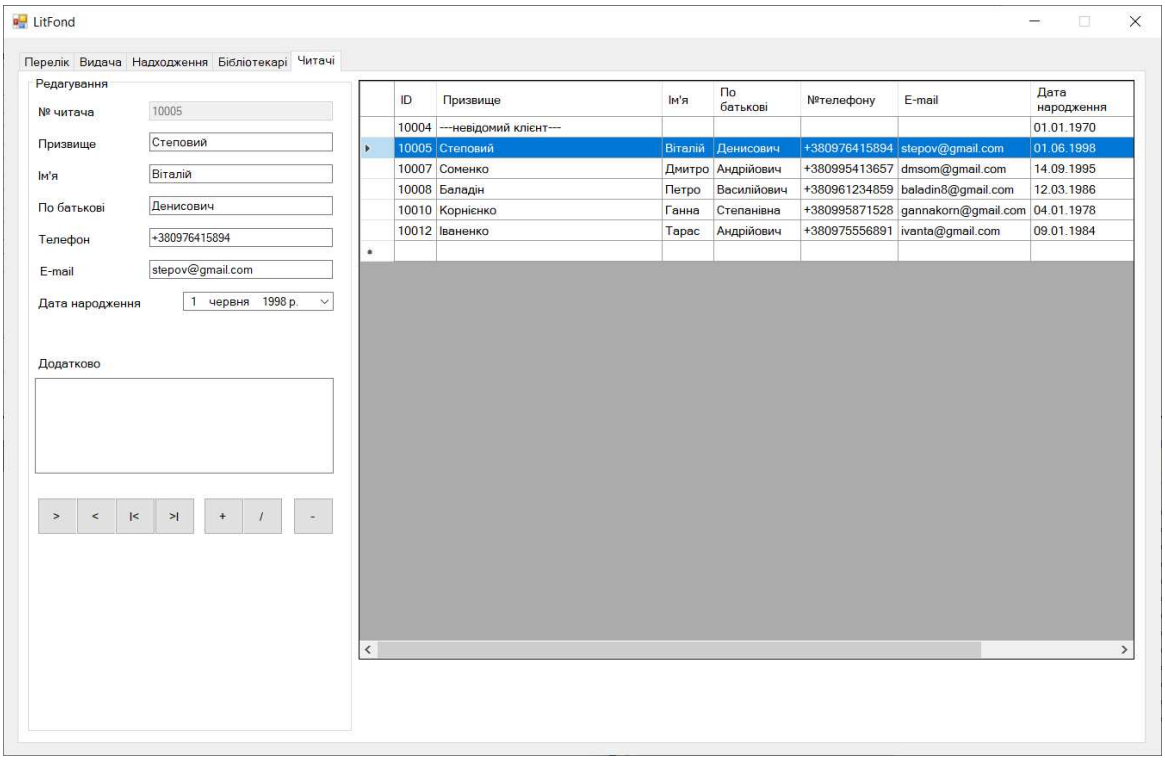


Рисунок 4.12 – Вкладка «Читачі»

Розроблені програмні засоби для обліку функціонування літературного фонду в повній мірі задовольняють зазначеним в попередніх розділах сформульованих функціональним вимогам, використовують попередньо спроектовану базу даних з зазначеними зв’язками між таблицями, та мають зручний для користувача інтерфейс. Таким чином, завдання на розробку програмного забезпечення можна вважати успішно виконаним.

ВИСНОВКИ

В ході проведення робіт з дослідження стану питання обліку роботи літературних фондів було визначено, що існуючі програмні засоби для їх оцифрування та для спрощення роботи бібліотекарів наявні на ринку програмного забезпечення, але є достатньо дорогими для придбання та використання. В той же час вони перенавантаженні великою кількістю функцій, які зазвичай не потрібні при використання у невеликих закладах. З огляду на це, було спроектовано додаток, що є зручним в роботі, реалізує основні необхідні функції (облік друкованих видань, постачань нової літератури, видачі книг читачам, зберігання персональних даних працівників бібліотеки та абонементів читачів), та може бути доволі швидко опанованим новими користувачами.

При виконанні дипломної роботи, окрім проведення детального та поглибленого аналізу стану ринку програмних продуктів-аналогів, було виконано наступні види робіт:

– сформульовано актуальність створення програмних засобів для обліку функціонування літературного фонду;

– проведено порівняння недоліків існуючих додатків та їх сильних сторін;

– визначено напрямки робіт проектування створюваного програмного продукту;

– сформульовано набір вимог до програмного забезпечення, що розроблювалося;

– визначено перелік вимог до програмних засобів;

– розроблено функціональну схему, що ілюструє принцип взаємодії компонентів системи;

– укладені алгоритми функціонування системи в цілому, так і окремих її компонентів;

– проведено проектування таблиць бази даних додатку;

– розроблено безпосередньо програмні засоби для обліку функціонування літературного фонду та проведено комплексне тестування його роботи в різних режимах.

Розроблене програмне забезпечення може бути застосоване як для спрощення роботи невеликих літературних фондів, так і для автоматизації більш масштабних закладів бібліотечного типу.

ПЕРЕЛІК ПОСИЛАНЬ

1. Дубровіна Л. А. Бібліотечна справа в Україні в ХХ столітті / Л. А. Дубровіна, О. О. Онищенко. – Київ : НБУВ, 2009. – 336 с.;

2. Бібліотечні каталоги як інформаційно-пошукові системи : навч. посіб. / авт.-уклад.: В. В. Сєдих [та ін.] ; наук. ред. Н. М. Кушнаренко ; Харк. держ. акад. культури. – Х. : ХДАК, 2003. – 193 с.

3. Систематизація інформації в контексті розвитку класифікацій наук : монографія / Олег Сербін. – К. : ВПЦ "Київський університет", 2015. – 431 с. ISBN 978-966-439-851-7

4. Бургін М. Оптимізація інформаційного наповнення автоматизованих каталогів наукових бібліотек / М. Бургін // Проблеми вдосконалення каталогів наукових бібліотек : матеріали доп. міжнар. наук. конф. (Київ, 14-17 жовт. 1997 р.). – К., 1997. – С. 73–74.

5. Зарічняк І. Систематичний каталог у системі інформаційних ресурсів наукової бібліотеки / І. Зарічняк, І. Багрій // Інформаційна діяльність наукової бібліотеки : матеріали доп. міжнар. наук. конф. (Київ, 1999 р.). – К., 2000. – С. 152–158.

6. Стрішинець Н. В. Сучасна американська бібліотечноінформаційна терміносистема : бібліотекознавчий аспект / Н. В. Стрішинець ; НАН України, Нац. б-ка України ім. В. І. Вернадського. – К., 2011. – 501 с.

7. Koha [Електронний ресурс] – Режим доступу до ресурсу: https://koha-community.org/download-koha/

8. Understanding Integrated Library Systems (ILS): Types and Categories [Електронний ресурс] – Режим доступу до ресурсу: https://lis.academy/ict-in-libraries/integrated-library-systems-types-categories/

9. АБІС Koha [Електронний ресурс] – Режим доступу до ресурсу: https://wiki.koha.org.ua

10. Koha integrated library system [Електронний ресурс] – Режим доступу до ресурсу: https://openlogictech.com/koha-library-system

11. Alma Library Management System [Електронний ресурс] – Режим доступу до ресурсу: https://exlibrisgroup.com/products/alma-library-services-platform/
12. ERP [Електронний ресурс] – Режим доступу до ресурсу: https://www.oracle.com/ua/erp/what-is-erp/
13. Kiran K., Diljit S. Modeling Web-based library service quality. Library & Information Science Research. 2012. Vol. 34, no. 3. P. 184–196. URL: https://doi.org/10.1016/j.lisr.2012.02.005
14. IFLA. Performance Measurement in Libraries. 2nd rev. ed. The Hague : IFLA, 2007. - 64 p. [Електронний ресурс] – Режим доступу до ресурсу: https://repository.ifla.org/bitstreams/dfdc2aad-9500-4e30-8be8-8d382393af22/download
15. IFLA [Електронний ресурс] – Режим доступу до ресурсу: https://repository.ifla.org/rest/api/core/bitstreams/dfdc2aad-9500-4e30-8be8-8d382393af22/content
16. Association of Research Libraries (ARL). LibQUAL+ Procedures Manual / D. Green, D. Meho та ін. 2012. [Електронний ресурс] – Режим доступу до ресурсу: https://www.libqual.org/documents/LibQual/publications/ProceduresManual.pdf
17. Users' Perception of the Quality of Public Library Services in the Greater Accra Region of Ghana: An Application of the LibQUAL+ Model / F. N.-A. Baada et al. Library Philosophy and Practice. 2019. 33 p.
18. Tian L. A Review on Studies of Library Efficiency Evaluation by Using DEA Method. Atlantis Press, 2018. [Електронний ресурс] – Режим доступу до ресурсу: https://www.atlantis-press.com/article/25897593_A_Review_on_Studies_of_Library_Efficiency_Evaluation_by_Using_DEA_Method
19. Najafi A., DEA-based Performance Evaluation of Libraries: systematic mapping study. SciELO / 2020. [Електронний ресурс] – Режим доступу до ресурсу: https://www.scielo.org.mx/scielo.php?pid=S0187-358X2020000400227&script=sci_arttext

20. Яковенко О. Г., Венідиктова А. В. Бібліотечні послуги: облік, статистика, ефективність : наук.-метод. посіб. Київ : НБУВ, 2019. - 132 с. [Електронний ресурс] – Режим доступу до ресурсу: https://irbis-nbuv.gov.ua/E_LIB/PDF/er-0003876.pdf
21. Gabriel Baptista, Francesco Abbruzzese. Software Architecture with C# 12 and .NET 8 - Fourth Edition: Build enterprise applications using microservices, DevOps, EF Core, and design patterns for Azure 4th ed. Edition. 2024. Pp 756.
22. Ian Griffiths. Programming C# 12: Build Cloud, Web, and Desktop Applications 1st Edition. 2024. Pp 834.
23. Visual Studio [Електронний ресурс] – Режим доступу до ресурсу: https://visualstudio.microsoft.com
24. .NET [Електронний ресурс] – Режим доступу до ресурсу: https://dotnet.microsoft.com/en-us/
25. Principal Design Features of .NET Framework [Електронний ресурс] – Режим доступу до ресурсу: http://www.zabalnet.com/overview-highlight-principal-design-features.html

Додаток А

Розрахунок собівартості розробленої системи.

Визначимо собівартість створеного програмного забезпечення для обліку функціонування літературного фонду.

Для здійснення розрахунку застосовувалися наступні показники, що представлені у таблиці А.1.

Таблиця А.1 - Початкові показники для визначення собівартості

Найменування	Показник
Складність створення програми, днів	35
Ставка програміста (місяць), грн	27200
Кількість годин в місяці, год	160
Додаткова зарплатня (%)	10
Відрахування до соц. фондів (%)	15
Загальновиробничі витрати компанії (%)	100
ПДВ (%)	20

Виконаємо розрахунок на період у 1 місяць, що включатиме в себе 20 робочих днів.

а) Пункт 1. Комплектуючі вироби - CD-диски для інсталяції системи:

$$З_k = \sum Ц_k * n_k \qquad (А.1)$$

де $З_К$ - витрати на CD-диски, грн.;

$Ц_К$ - вартість за 1 одиницю CD-диску, грн.;

n_k - кількість CD-дисків, шт.

Визначимо витрати на CD-диски:

$$З_k = 20*10 = 200грн \quad (А.2)$$

Витрати на CD-диски склали 200 грн.

б) Пункт 2. Витрати на електроенергію визначаємо згідно виразу А.3:

$$B_E = P_E \sum W_i * t_i \quad (А.3)$$

де P_E - вартість 1 кВт-год, грн.;

Wi - середня споживана потужність в ході роботи.

Вартість 1кВт електроенергії на момент написання кваліфікаційної роботи складає 4,32 грн.

Розрахуємо витрати виробництва на електроенергію згідно виразу А.4:

$$B_E = 4,32*2*160 = 1382,4грн \quad (А.4)$$

Витрати виробництва на електроенергію складають 1382,4 грн.

в) Пункт 3. Основна заробітна плата розраховується згідно виразу А.5:

$$З_{осн} = l_{год} * T_{год} \quad (А.5)$$

де $l_{год}$ - тарифна ставка робітника (годинна), грн.;

$T_{год}$ - кількість робочих годин у місяці (160 год.)

Годинна тарифна ставка програміста, що розробляє програмні засоби для обліку функціонування літературного фонду, дорівнює 170 грн.

Розрахуємо основну заробітну плату програміста згідно виразу А.6:

$$З_{осн} = 170 * 160 = 27200 грн \quad (А.6)$$

Основна заробітна плата програміста складає 27200 грн.

г) Пункт 4. Додаткова заробітна плата розраховується згідно виразу А.7:

$$З_{дод} = \frac{З_{осн} * Д\%}{100} \quad (А.7)$$

де $Здод$ - це додаткова заробітна плата розробника системи, грн.;

$Д\%$ - відсоток додаткової заробітної плати розробника системи (10%).

Визначимо додаткову заробітну плату згідно виразу А.8:

$$З_{дод} = \frac{27200 * 10}{100} = 2720 грн \quad (А.8)$$

Додаткова заробітна плата буде дорівнювати 2720 грн.

д) Пункт 5. Розмір відрахування в соц. фонди визначається згідно виразу А.9:

$$З_{соц} = \frac{(З_{осн} + З_{дод}) * С\%}{100} \quad (А.9)$$

де $Зсоц$ - розмір відрахування в соц. фонди, грн.;

$С\%$ - визначений відсоток відрахувань у соц. фонди (15%).

Обчислимо відрахування в соц. фонди згідно виразу А.10:

$$З_{соц} = \frac{(27200 + 2720) * 15}{100} = 4488 грн \quad (А.10)$$

Розраховане відрахування в соц. фонди складає 4488 грн;

е) Пункт 6. Загальновиробничі витрати обчислюються згідно виразу А.11:

$$З_{заг} = \frac{З_{осн} * Н_1 \%}{100} \quad (А.11)$$

де $З_{заг}$ - загальновиробничі витрати підприємства, грн.;

$Н_1 \%$ - запланований відсоток загальновиробничих витрат (100%)

Розрахуємо загальновиробничі витрати згідно виразу А.12:

$$З_{заг} = \frac{27200 * 100}{100} = 27200 грн \quad (А.12)$$

Загальновиробничі витрати підприємства складають 27200,00 грн.

Розрахуємо виробничу собівартість, виконавши послідовне додавання результатів здійснених розрахунків (вираз А.13)

$$S_{пп} = 200 + 1382,4 + 27200 + 2720 + 4488 + 27200 = 63190,4 грн \quad (А.13)$$

де $S_{пп}$ - виробнича собівартість, грн.

Таким чином, виробнича собівартість складає 63190,4 грн.

В таблиці А.2 представлено планову калькуляцію стосовно виробничої собівартості, вартості для підприємства та вартості для замовника на створення програмних засобів для обліку функціонування літературного фонду.

Створення системи включало в себе роботи, що виконувались протягом 35 робочих днів. Таким чином, собівартість розробленого програмного забезпечення складає 110583,2 грн.

Таблиця А.2 - Планова калькуляція

Пункти калькуляції	Сума, грн.
Комплектуючи вироби	200
Кошти на електроенергію	1382,4
Основна зар. плата	27200
Додаткова зар. плата	2720
Розмір відрахувань в соц. фонди	4488
Загальновиробничі витрати підприємства	27200
Виробнича собівартість в розрахунку на 1 місяць	63190,4
Собівартість розробки програмних засобів для обліку функціонування літературного фонду	110583,2

Розрахувати економічний ефект від застосування програмних засобів для обліку функціонування літературного фонду можна, використовуючи отриману при розробці математичної моделі формулу (2.1), наведену в другому розділі роботи. Для цього потрібно визначити числові значення її змінних. Емпірично визначенні числові показники даних параметрів наведені в таблиці А.3.

Таким чином, можна перейти до розрахунків терміну окупності розроблюваної системи за формулою:

$$T = \frac{C}{E_{еф}} \qquad (А.14)$$

де T – строк окупності розроблюваного проекту, міс.,

C – собівартість розроблюваного проекту, грн.,

$E_{еф}$ – розрахований місячний економічний ефект від розроблюваного проекту, грн.

Таблиця А.3 - Емпірично визначенні числові показники параметрів для розрахунку економ. ефективності впровадження розробки

Параметр	Пояснення	Значення
p_L	Ринкова або умовна вартість однієї видачі, грн	10
L	Кількість видач за місяць, шт	1500
p_R	Монетизована оцінка одного читача, грн	100
R	Кількість читачів за місяць	800
w_S	Середня місячна вартість утримання одного працівника, грн	10000
S	Кількість працівників бібліотеки	5
P_{avg}	Середня вартість одного примірника книжки, грн	450
C	Загальна кількість примірників, шт	20000
T_{life}	Середній строк експлуатації книжкового фонду, місяці	600

Таким чином, економічний ефект складе

$$E_{еф} = 10 * 1500 + 100 * 800 - 10000 * 5 - \frac{450*20000}{600} = 30000 \text{ грн} \quad (А.15)$$

Визначимо термін окупності розроблюваної системи згідно виразу:

$$T = 110583{,}2/30000 = 3{,}69 \text{ місяця} \quad (А.16)$$

Отже, собівартість системи складає 110583,2 грн, економефект від її впровадження складає 30000 грн на місяць, а витрачені засоби окупляться за 3,69 місяця. Таким чином, можна відмічати доцільність створення і впровадження зазначеного програмного забезпечення та економічну доцільність його застосування.

Додаток Б

Лістинг програми

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace LitFond
{
    public partial class LitFond: Form
    {
        SqlConnection sqlConnection;

        private BindingSource bindingSource = new BindingSource();
        private BindingSource bindingSource2 = new BindingSource();
        private BindingSource bindingSource3 = new BindingSource();
        private BindingSource bindingSource4 = new BindingSource();
        private BindingSource bindingSource5 = new BindingSource();
        DataSet dataset = new DataSet();
        DataSet dataset2 = new DataSet();
        DataSet dataset3 = new DataSet();
        DataSet dataset4 = new DataSet();
        DataSet dataset5 = new DataSet();

        public LitFond()
        {
            InitializeComponent();
            readData();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            InitBalance();

            comboBox1Actual.Items.Add("Застаріле");
            comboBox1Actual.Items.Add("Звичайне");
            comboBox1Actual.Items.Add("Новинка");
            comboBox1Actual.SelectedItem = 1;
            comboBox1Paliturka.Items.Add("Тверда");
            comboBox1Paliturka.Items.Add("М'яка");
            comboBox1Mova.Items.Add("Українська");
            comboBox1Mova.Items.Add("Англійська");
            comboBox1Mova.Items.Add("Інша");
        }

        private string getDate()
        {
            DateTime date = new DateTime();
            date = DateTime.Now;
            return date.ToShortDateString();
        }

        private string getTime()
```

```
        {
            DateTime time = new DateTime();
            time = DateTime.Now;
            return time.ToLongTimeString();
        }

        private void readData()
        {
            //connection with server
            SqlConnection connection = CreateConnection();
            SqlCommand command = new SqlCommand("SELECT * FROM [Product]");
            command.Connection = connection;
            SqlDataAdapter adapter = new SqlDataAdapter();
            adapter.SelectCommand = command;
            //fill dataset
            adapter.Fill(dataset);

            SqlDataAdapter adapterEmployee = new SqlDataAdapter();
            SqlCommand commandEmployee = new SqlCommand("SELECT * FROM
Employee", connection);
            adapterEmployee.SelectCommand = commandEmployee;
            adapterEmployee.Fill(dataset2);

            SqlDataAdapter adapterClient = new SqlDataAdapter();
            SqlCommand commandClient = new SqlCommand("SELECT * FROM Client",
connection);
            adapterClient.SelectCommand = commandClient;
            adapterClient.Fill(dataset3);

            SqlDataAdapter adapterSale = new SqlDataAdapter();
            SqlCommand commandSale = new SqlCommand("SELECT * FROM Sale",
connection);
            adapterSale.SelectCommand = commandSale;
            adapterSale.Fill(dataset4);

            SqlDataAdapter adapterSupply = new SqlDataAdapter();
            SqlCommand commandSupply = new SqlCommand("SELECT * FROM Supply",
connection);
            adapterSupply.SelectCommand = commandSupply;
            adapterSupply.Fill(dataset5);

            //close connection
            connection.Close();

            dataGridView1.AutoGenerateColumns = true;
            bindingSource.DataSource = dataset.Tables[0];
            dataGridView1.DataSource = bindingSource;
            dataGridView2.AutoGenerateColumns = true;
            bindingSource.DataSource = dataset.Tables[0];
            dataGridView2.DataSource = bindingSource;
            dataGridView3.AutoGenerateColumns = true;
            bindingSource.DataSource = dataset.Tables[0];
            dataGridView3.DataSource = bindingSource;

            dataGridView4.AutoGenerateColumns = true;
            bindingSource2.DataSource = dataset2.Tables[0];
            dataGridView4.DataSource = bindingSource2;
            dataGridView5.AutoGenerateColumns = true;
            bindingSource3.DataSource = dataset3.Tables[0];
            dataGridView5.DataSource = bindingSource3;

            dataGridView6.AutoGenerateColumns = true;
            bindingSource4.DataSource = dataset4.Tables[0];
```

```
            dataGridView6.DataSource = bindingSource4;
            dataGridView7.AutoGenerateColumns = true;
            bindingSource5.DataSource = dataset5.Tables[0];
            dataGridView7.DataSource = bindingSource5;
        }

        SqlConnection CreateConnection()
        {
            SqlConnection connection = new SqlConnection();
            connection.ConnectionString = connectionString;
            try
            {
                connection.Open();
            }
            catch
            {
                MessageBox.Show("Помилка з'єднання з БД!");
            }

            return connection;
        }

        //delete row
        private void buttonDeleteColumn_Click(object sender, EventArgs e)
        {
                if (dataGridView1.CurrentRow != null)
                {
                    dataGridView1.Rows.Remove(dataGridView1.CurrentRow);
                }
        }

        //close the app
        private void buttonCloseForm_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void buttonSearch_Click(object sender, EventArgs e)
        {
            SqlConnection connection = new SqlConnection();
            connection.ConnectionString = connectionString;

            SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM
Product", connection);
            adapter.SelectCommand = new SqlCommand("SELECT * FROM Product
WHERE Product.iTovar LIKE N'%" + textBoxSearch.Text.Trim() + "%' OR
Product.bookTitle LIKE N'%" + textBoxSearch.Text.Trim() + "%' OR
Product.author LIKE N'%" + textBoxSearch.Text.Trim() + "%' OR Product.cost
LIKE N'%" + textBoxSearch.Text.Trim() + "%' OR Product.relevance LIKE N'%" +
textBoxSearch.Text.Trim() + "%' OR Product.publishing LIKE N'%" +
textBoxSearch.Text.Trim() + "%' OR Product.year LIKE N'%" +
textBoxSearch.Text.Trim() + "%' OR Product.year LIKE N'%" +
textBoxSearch.Text.Trim() + "%' OR Product.cover LIKE N'%" +
textBoxSearch.Text.Trim() + "%' OR Product.language LIKE N'%" +
textBoxSearch.Text.Trim() + "%' OR Product.isbn LIKE N'%" +
textBoxSearch.Text.Trim() + "%' OR Product.page LIKE N'%" +
textBoxSearch.Text.Trim() + "%' OR Product.count LIKE N'%" +
textBoxSearch.Text.Trim() + "%'", connection);
            adapter.SelectCommand.Connection = connection;
            DataSet dataset = new DataSet();
            adapter.Fill(dataset);
            dataGridView1.DataSource = dataset.Tables[0];
        }
```

```
        private void buttonSearchClear_Click(object sender, EventArgs e)
        {
            textBoxSearch.Clear();

            //show new row (refresh DataGrid)
            SqlConnection connection = new SqlConnection();
            connection.ConnectionString = connectionString;

            DataSet dataset = new DataSet();

            SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM
Product", connection);
            adapter.Fill(dataset);
            dataGridView1.DataSource = dataset.Tables[0];

        }

        private void dataGridView3_CellEnter(object sender,
DataGridViewCellEventArgs e)
        {
            int indexRow = dataGridView3.CurrentRow.Index;
            dataGridView3.Rows[indexRow].Selected = true;
            dataGridView3.MultiSelect = false;

            //checking for last row
            if (dataGridView3[0,
dataGridView3.CurrentRow.Index].Value.ToString().Trim() == "")
            {
                textBox1Tovar.Clear();
                textBox1Nazva.Clear();
                textBox1Avtor.Clear();
                textBox1Vydavn.Clear();
            }
            else
            {
                textBox1Tovar.Text =
dataGridView3.CurrentRow.Cells["idKlient"].Value.ToString();
                textBox1Nazva.Text =
dataGridView3.CurrentRow.Cells["lastName"].Value.ToString();
                textBox1Avtor.Text =
dataGridView3.CurrentRow.Cells["firstName"].Value.ToString();
            }
        }

        private void button1Add_Click(object sender, EventArgs e)
        {
            int currentRow = dataGridView1.CurrentRow.Index;
            //open connection
            SqlConnection connection = new SqlConnection();
            connection.ConnectionString = connectionString;
            connection.Open();
            SqlCommand command = new SqlCommand();
            command.Connection = connection;

            //make INSERT query
            command.CommandText = "INSERT INTO [Product] (bookTitle, author,
cost, relevance, publishing, year, cover, language, isbn, page, count) VALUES
(@bookTitle, @author, @cost, @relevance, @publishing, @year, @cover,
@language, @isbn, @page, @count)";
            command.Parameters.Add(new SqlParameter("bookTitle",
textBox1Nazva.Text));
```

```
            command.Parameters.Add(new SqlParameter("author",
textBox1Avtor.Text));
            command.Parameters.Add(new SqlParameter("cost",
numericUpDown1Vartist.Text));
            command.Parameters.Add(new SqlParameter("relevance",
comboBox1Actual.Text.Trim()));
            command.Parameters.Add(new SqlParameter("publishing",
textBox1Vydavn.Text));
            command.Parameters.Add(new SqlParameter("year",
numericUpDown1Rik.Text));
            command.Parameters.Add(new SqlParameter("cover",
comboBox1Paliturka.Text.Trim()));
            command.Parameters.Add(new SqlParameter("language",
comboBox1Mova.Text.Trim()));
            command.Parameters.Add(new SqlParameter("isbn",
textBox1ISBN.Text));
            command.Parameters.Add(new SqlParameter("page",
numericUpDown1Storinky.Text));
            command.Parameters.Add(new SqlParameter("count",
numericUpDown1Nayavnist.Text));
            command.ExecuteNonQuery();
            connection.Close();

            DataSet dataset = new DataSet();
            SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM
[Product]", connection);
            adapter.Fill(dataset);
            dataGridView1.DataSource = dataset.Tables[0];

            //move to row selected before
            dataGridView1.CurrentCell = dataGridView1[0, currentRow];
        }

        private void button1Edit_Click(object sender, EventArgs e)
        {
            int currentRow = dataGridView1.CurrentRow.Index;

            //open connection
            SqlConnection connection = new SqlConnection();
            connection.ConnectionString = connectionString;
            connection.Open();
            SqlCommand command = new SqlCommand();
            command.Connection = connection;

            //make Update query
            command.CommandText = "UPDATE [Product] SET bookTitle =
@bookTitle, author = @author, cost = @cost, relevance = @relevance,
publishing = @publishing, year = @year, cover = @cover, language = @language,
isbn = @isbn, page = @page, count = @count WHERE iTovar = @iTovar";
            command.Parameters.Add(new SqlParameter("iTovar",
textBox1Tovar.Text));
            command.Parameters.Add(new SqlParameter("bookTitle",
textBox1Nazva.Text));
            command.Parameters.Add(new SqlParameter("author",
textBox1Avtor.Text));
            command.Parameters.Add(new SqlParameter("cost",
numericUpDown1Vartist.Text));
            command.Parameters.Add(new SqlParameter("relevance",
comboBox1Actual.Text.Trim()));
            command.Parameters.Add(new SqlParameter("publishing",
textBox1Vydavn.Text));
            command.Parameters.Add(new SqlParameter("year",
numericUpDown1Rik.Text));
```

```
            command.Parameters.Add(new SqlParameter("cover",
comboBox1Paliturka.Text.Trim()));
            command.Parameters.Add(new SqlParameter("language",
comboBox1Mova.Text.Trim()));
            command.Parameters.Add(new SqlParameter("isbn",
textBox1ISBN.Text));
            command.Parameters.Add(new SqlParameter("page",
numericUpDown1Storinky.Text));
            command.Parameters.Add(new SqlParameter("count",
numericUpDown1Nayavnist.Text));
            command.ExecuteNonQuery();
            connection.Close();

            //show new row (refresh DataGrid)
            DataSet dataset = new DataSet();
            SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM
[Product]", connection);
            adapter.Fill(dataset);
            dataGridView1.DataSource = dataset.Tables[0];
            dataGridView1.CurrentCell = dataGridView1[0, currentRow];

        }

        private void button1Delete_Click(object sender, EventArgs e)
        {
            {
                DialogResult dialogResult = MessageBox.Show("Ви впевнені?",
"", MessageBoxButtons.OKCancel);
                if (dialogResult == DialogResult.OK)
                {
                    //open connection
                    SqlConnection connection = new SqlConnection();
                    connection.ConnectionString = connectionString;
                    connection.Open();
                    SqlCommand command = new SqlCommand();
                    command.Connection = connection;

                    command.CommandText = "DELETE FROM [Product] WHERE iTovar
= @iTovar";
                    command.Parameters.Add(new SqlParameter("iTovar",
textBox1Tovar.Text));
                    command.Connection = connection;
                    command.ExecuteNonQuery();
                    connection.Close();

                    //show new row (refresh DataGrid)
                    DataSet dataset = new DataSet();
                    SqlDataAdapter adapter = new SqlDataAdapter("SELECT *
FROM [Product]", connection);
                    adapter.Fill(dataset);
                    dataGridView1.DataSource = dataset.Tables[0];
                }
            }
        }

        private void dataGridView1_CellEnter_1(object sender,
DataGridViewCellEventArgs e)
        {
            int indexRow = dataGridView1.CurrentRow.Index;
            dataGridView1.Rows[indexRow].Selected = true;
            dataGridView1.MultiSelect = false;
            //checking for last row
```

```
            if (dataGridView1[0,
dataGridView1.CurrentRow.Index].Value.ToString().Trim() == "")
            {
                textBox1Nazva.Clear();
                textBox1Avtor.Clear();
                numericUpDown1Vartist.Text = "0";
                comboBox1Actual.SelectedIndex = 1;
                textBox1Vydavn.Clear();
                numericUpDown1Rik.Text = "0";
                comboBox1Paliturka.SelectedIndex = 1;
                comboBox1Mova.SelectedIndex = 1;
                textBox1ISBN.Clear();
                numericUpDown1Storinky.Text = "0";
                numericUpDown1Nayavnist.Text = "0";
            }
            else
            {
                if (dataGridView1.CurrentRow.Cells["iTovar"].Value == null)
                {
                    textBox1Tovar.Clear();
                }
                else
                {
                    string sITovar =
dataGridView1.CurrentRow.Cells["iTovar"].Value.ToString();
                    textBox1Tovar.Text = sITovar;
                }

                //get bookTitle value
                if (dataGridView1.CurrentRow.Cells["bookTitle"].Value ==
null)
                {
                    textBox1Nazva.Text = "";
                }
                else
                {
                    string sbookTitle =
dataGridView1.CurrentRow.Cells["bookTitle"].Value.ToString();
                    textBox1Nazva.Text = sbookTitle;
                }

                //get author value
                if (dataGridView1.CurrentRow.Cells["author"].Value == null)
                {
                    textBox1Avtor.Text = "";
                }
                else
                {
                    string sAuthor =
dataGridView1.CurrentRow.Cells["author"].Value.ToString();
                    textBox1Avtor.Text = sAuthor;
                }

                //get costs value
                if (dataGridView1.CurrentRow.Cells["cost"].Value == null)
                {
                    numericUpDown1Vartist.Text = "";
                }
                else
                {
                    string sCost =
dataGridView1.CurrentRow.Cells["cost"].Value.ToString();
                    numericUpDown1Vartist.Text = sCost;
```

```
                }

                //get revelance value
                if (dataGridView1.CurrentRow.Cells["relevance"].Value ==
null)
                {
                    comboBox1Actual.SelectedIndex = 1;
                }
                else
                {
                    string sRelevance =
dataGridView1.CurrentRow.Cells["relevance"].Value.ToString();
                    comboBox1Actual.Text = sRelevance;
                }

                //get publishing value
                if (dataGridView1.CurrentRow.Cells["publishing"].Value ==
null)
                {
                    textBox1Vydavn.Text = "";
                }
                else
                {
                    string sPublishing =
dataGridView1.CurrentRow.Cells["publishing"].Value.ToString();
                    textBox1Vydavn.Text = sPublishing;
                }

                //get year
                if (dataGridView1.CurrentRow.Cells["year"].Value == null)
                {
                    numericUpDown1Rik.Text = "";
                }
                else
                {
                    string sYear =
dataGridView1.CurrentRow.Cells["year"].Value.ToString();
                    numericUpDown1Rik.Text = sYear;
                }

                //get cover
                if (dataGridView1.CurrentRow.Cells["cover"].Value == null)
                {
                    comboBox1Paliturka.Text = "";
                }
                else
                {
                    string sCover =
dataGridView1.CurrentRow.Cells["cover"].Value.ToString();
                    comboBox1Paliturka.Text = sCover;
                }

                //get language
                if (dataGridView1.CurrentRow.Cells["language"].Value == null)
                {
                    comboBox1Mova.Text = "";
                }
                else
                {
                    string sLanguage =
dataGridView1.CurrentRow.Cells["language"].Value.ToString();
                    comboBox1Mova.Text = sLanguage;
                }
```

```
                //get ISBN
                if (dataGridView1.CurrentRow.Cells["isbn"].Value == null)
                {
                    textBox1ISBN.Text = "";
                }
                else
                {
                    string sISBN =
dataGridView1.CurrentRow.Cells["isbn"].Value.ToString();
                    textBox1ISBN.Text = sISBN;
                }

                //get pages
                if (dataGridView1.CurrentRow.Cells["page"].Value == null)
                {
                    numericUpDown1Storinky.Text = "";
                }
                else
                {
                    string sPage =
dataGridView1.CurrentRow.Cells["page"].Value.ToString();
                    numericUpDown1Storinky.Text = sPage;
                }

                //get count
                if (dataGridView1.CurrentRow.Cells["count"].Value == null)
                {
                    numericUpDown1Nayavnist.Text = "";
                }
                else
                {
                    string sCount =
dataGridView1.CurrentRow.Cells["count"].Value.ToString();
                    numericUpDown1Nayavnist.Text = sCount;
                }
            }
        }

        private void button1Next_Click(object sender, EventArgs e)
        {
            int index = dataGridView1.CurrentRow.Index;
            dataGridView1.Rows[index].Selected = true;
            try
            {
                dataGridView1.CurrentCell = dataGridView1[0, index + 1];
            }
            catch
            {
            }
        }

        private void button1Prev_Click(object sender, EventArgs e)
        {
            int index = dataGridView1.CurrentRow.Index;
            int indexMax = dataGridView1.Rows.Count;
            dataGridView1.Rows[index].Selected = true;
            try
            {
                dataGridView1.CurrentCell = dataGridView1[0, index - 1];
            }
            catch
            {
```

```
            }
        }

        private void button1First_Click(object sender, EventArgs e)
        {
            int indexMax = dataGridView1.Rows.Count;
            dataGridView1.CurrentCell = dataGridView1[0, 0];
        }

        private void button1Last_Click(object sender, EventArgs e)
        {
            int indexMax = dataGridView1.Rows.Count;
            dataGridView1.CurrentCell = dataGridView1[0, indexMax - 1];
        }

        private void button2Add_Click(object sender, EventArgs e)
        {
            //checking for availability
            if (numericUpDown2Nayavnist.Value > 0)
            {
                numericUpDown2Nayavnist.Value--;

                int currentRow;
                try
                {
                    currentRow = dataGridView1.CurrentRow.Index;
                }
                catch
                {
                    currentRow = 0;
                }

                SqlConnection connection = new SqlConnection();
                connection.ConnectionString = connectionString;
                connection.Open();
                SqlCommand command = new SqlCommand();
                command.Connection = connection;

                string comboBox2ClientId =
comboBox2Client.Items[comboBox2Client.SelectedIndex].ToString();
                int iKlnt =
int.Parse(comboBox2ClientId.Substring(comboBox2ClientId.Length - 5, 5));

                string comboBox2SellerId =
comboBox2Seller.Items[comboBox2Seller.SelectedIndex].ToString();
                int iEmployee =
int.Parse(comboBox2SellerId.Substring(comboBox2SellerId.Length - 4, 4));


                try
                {
                    currentRow = dataGridView6.CurrentRow.Index;
                }
                catch
                {
                    currentRow = 0;
                }

                command.CommandText = "INSERT INTO [Sale] (dateSale,
timeSale, iTovar, bookTitle, author, price, iEmployee, iKlient, notes) VALUES
(@dateSale, @timeSale, @iTovar, @bookTitle, @author, @price, @iEmployee,
@iKlient, @notes)";
```

```
                command.Parameters.Add(new SqlParameter("dateSale",
getDate()));
                command.Parameters.Add(new SqlParameter("timeSale",
getTime()));
                command.Parameters.Add(new SqlParameter("iTovar",
textBox2Tovar.Text));
                command.Parameters.Add(new SqlParameter("bookTitle",
textBox2Nazva.Text));
                command.Parameters.Add(new SqlParameter("author",
textBox2Avtor.Text));
                command.Parameters.Add(new SqlParameter("price",
numericUpDown2Vartist.Text));
                //command.Parameters.Add(new SqlParameter("count",
numericUpDown2Nayavnist.Text));
                command.Parameters.Add(new SqlParameter("iEmployee",
iEmployee));
                command.Parameters.Add(new SqlParameter("iKlient", iKlnt));
                command.Parameters.Add(new SqlParameter("notes",
textBox2Info.Text));

                command.ExecuteNonQuery();
                connection.Close();

                connection = new SqlConnection();
                connection.ConnectionString = connectionString;
                connection.Open();
                command = new SqlCommand();
                command.Connection = connection;

                command.CommandText = "UPDATE [Product] SET count = @count
WHERE iTovar = @iTovar";
                command.Parameters.Add(new SqlParameter("iTovar",
textBox1Tovar.Text));
                command.Parameters.Add(new SqlParameter("count",
numericUpDown2Nayavnist.Text));
                command.ExecuteNonQuery();
                connection.Close();

                DataSet dataset = new DataSet();
                SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM
[Product]", connection);
                adapter.Fill(dataset);
                dataGridView2.DataSource = dataset.Tables[0];

                dataGridView2.CurrentCell = dataGridView2[0, currentRow];

                //show new row (refresh DataGrid)
                dataset = new DataSet();
                adapter = new SqlDataAdapter("SELECT * FROM [Sale]",
connection);
                adapter.Fill(dataset);
                dataGridView6.DataSource = dataset.Tables[0];

                //move to row selected before
                dataGridView6.CurrentCell = dataGridView6[0, currentRow];
            }
            else
            {
                MessageBox.Show("Товар скінчився!");
            }
        }

        private void button2Edit_Click(object sender, EventArgs e)
```

```
        {
            int currentRow;
            try
            {
                currentRow = dataGridView6.CurrentRow.Index;
            }
            catch
            {
                currentRow = 0;
            }

            //open connection
            SqlConnection connection = new SqlConnection();
            connection.ConnectionString = connectionString;
            connection.Open();
            SqlCommand command = new SqlCommand();
            command.Connection = connection;

            string comboBox2ClientId =
comboBox2Client.Items[comboBox2Client.SelectedIndex].ToString();
            int iKlnt =
int.Parse(comboBox2ClientId.Substring(comboBox2ClientId.Length - 5, 5));

            string comboBox2SellerId =
comboBox2Seller.Items[comboBox2Seller.SelectedIndex].ToString();
            int iEmployee =
int.Parse(comboBox2SellerId.Substring(comboBox2SellerId.Length - 4, 4));

            //make Update query
            command.CommandText = "UPDATE [Sale] SET dateSale = @dateSale,
timeSale = @timeSale, price = @price, iEmployee = @iEmployee, iKlient =
@iKlient, notes = @notes WHERE iSale = @iSale";
            command.Parameters.Add(new SqlParameter("iSale",
textBox2Sell.Text));
            command.Parameters.Add(new SqlParameter("dateSale", getDate()));
            command.Parameters.Add(new SqlParameter("timeSale", getTime()));
            command.Parameters.Add(new SqlParameter("iTovar",
textBox2Tovar.Text));
            command.Parameters.Add(new SqlParameter("price",
numericUpDown2Vartist.Text));

            command.Parameters.Add(new SqlParameter("iEmployee", iEmployee));
            command.Parameters.Add(new SqlParameter("iKlient", iKlnt));
            command.Parameters.Add(new SqlParameter("notes",
textBox2Info.Text));

            command.ExecuteNonQuery();
            connection.Close();

            //show new row (refresh DataGrid)
            DataSet dataset = new DataSet();
            SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM
[Sale]", connection);
            adapter.Fill(dataset);
            dataGridView6.DataSource = dataset.Tables[0];

            dataGridView6.CurrentCell = dataGridView6[0, currentRow];
        }

        private void button2Delete_Click(object sender, EventArgs e)
        {
            {
```

```
                DialogResult dialogResult = MessageBox.Show("Ви впевнені?",
"", MessageBoxButtons.OKCancel);
                if (dialogResult == DialogResult.OK)
                {
                    //decreaseCurrentBalance();

                    //open connection
                    SqlConnection connection = new SqlConnection();
                    connection.ConnectionString = connectionString;
                    connection.Open();
                    SqlCommand command = new SqlCommand();
                    command.Connection = connection;

                    command.CommandText = "DELETE FROM [Sale] WHERE iSale =
@iSale";
                    command.Parameters.Add(new SqlParameter("iSale",
textBox2Sell.Text));
                    command.Connection = connection;
                    command.ExecuteNonQuery();
                    connection.Close();

                    //show new row (refresh DataGrid)
                    DataSet dataset = new DataSet();
                    SqlDataAdapter adapter = new SqlDataAdapter("SELECT *
FROM [Sale]", connection);
                    adapter.Fill(dataset);
                    dataGridView6.DataSource = dataset.Tables[0];
                }
            }
        }

        private void dataGridView6_CellEnter(object sender,
DataGridViewCellEventArgs e)
        {
            int indexRow = dataGridView6.CurrentRow.Index;
            dataGridView6.Rows[indexRow].Selected = true;
            dataGridView6.MultiSelect = false;
            //checking for last row
            if (dataGridView6[0,
dataGridView6.CurrentRow.Index].Value.ToString().Trim() == "")
            {
                //textBox2Tovar.Clear();
                textBox2Nazva.Clear();
                textBox2Avtor.Clear();
                numericUpDown2Nayavnist.Value = 0;
                //comboBox2Seller.SelectedIndex = 0;
                //comboBox2Client.SelectedIndex = 0;
                numericUpDown2Znyzhka.Value = 0;
                numericUpDown2Vartist.Value = 0;
                textBox2Info.Clear();

            }
            else
            {}
                if (dataGridView6.CurrentRow.Cells["price"].Value == null)
                {
                    numericUpDown2Vartist.Value = 0;
                }
                else
                {
                    numericUpDown2Vartist.Text =
dataGridView6.CurrentRow.Cells["price"].Value.ToString();
                }
```

```
                if (dataGridView6.CurrentRow.Cells["iSale"].Value == null)
                {
                    textBox2Sell.Clear();
                }
                else
                {
                    textBox2Sell.Text =
dataGridView6.CurrentRow.Cells["iSale"].Value.ToString();
                }

                if (dataGridView6.CurrentRow.Cells["notes"].Value == null)
                {
                    textBox2Info.Clear();
                }
                else
                {
                    textBox2Info.Text =
dataGridView6.CurrentRow.Cells["notes"].Value.ToString();
                }
            }
        }

        private void dataGridView2_CellEnter_1(object sender,
DataGridViewCellEventArgs e)
        {
            int indexRow = dataGridView2.CurrentRow.Index;
            dataGridView2.Rows[indexRow].Selected = true;
            dataGridView2.MultiSelect = false;

            numericUpDown2Znyzhka.Text = "0";
            //checking for last row
            if (dataGridView2[0,
dataGridView2.CurrentRow.Index].Value.ToString().Trim() == "")
            {
                textBox2Tovar.Clear();
                textBox2Nazva.Clear();
                textBox2Avtor.Clear();
                numericUpDown2Nayavnist.Text = "0";
                numericUpDown2Vartist.Text = "0";
            }
            else
            {
                //get sITovar value
                if (dataGridView2.CurrentRow.Cells["iTovar"].Value == null)
                {
                    textBox2Tovar.Clear();
                }
                else
                {
                    textBox2Tovar.Text =
dataGridView2.CurrentRow.Cells["iTovar"].Value.ToString();
                }

                //get bookTitle value
                if (dataGridView2.CurrentRow.Cells["bookTitle"].Value ==
null)
                {
                    textBox2Nazva.Text = "";
                }
                else
                {
```

```
                    textBox2Nazva.Text =
dataGridView2.CurrentRow.Cells["bookTitle"].Value.ToString();
                }

                //get author value
                if (dataGridView2.CurrentRow.Cells["author"].Value == null)
                {
                    textBox2Avtor.Text = "";
                }
                else
                {
                    textBox2Avtor.Text =
dataGridView2.CurrentRow.Cells["author"].Value.ToString();
                }

                //get count
                if (dataGridView2.CurrentRow.Cells["count"].Value == null)
                {
                    numericUpDown2Nayavnist.Text = "";
                }
                else
                {
                    numericUpDown2Nayavnist.Text =
dataGridView2.CurrentRow.Cells["count"].Value.ToString();
                }

                if (dataGridView2.CurrentRow.Cells["cost"].Value == null)
                {
                    numericUpDown2Vartist.Text = "";
                }
                else
                {
                    numericUpDown2Vartist.Text =
dataGridView2.CurrentRow.Cells["cost"].Value.ToString();
                }


            }
        }

        private void InitEmployeeDropdown()
        {
            comboBox2Seller.Items.Clear();
            comboBox3Spivr.Items.Clear();

            SqlConnection connection = CreateConnection();
            SqlCommand commandSeller = new SqlCommand("SELECT iEmployee,
lastNameEmployee, firstNameEmployee, middleNameEmployee FROM [Employee]");
            commandSeller.Connection = connection;
            SqlDataReader sqlDataReader = commandSeller.ExecuteReader();
            while (sqlDataReader.Read())
            {

comboBox2Seller.Items.Add(sqlDataReader["lastNameEmployee"].ToString().Trim()
+ " " + sqlDataReader["firstNameEmployee"].ToString().Trim() + " " +
sqlDataReader["middleNameEmployee"].ToString().Trim() + " " +
sqlDataReader["iEmployee"].ToString().Trim());

comboBox3Spivr.Items.Add(sqlDataReader["lastNameEmployee"].ToString().Trim()
+ " " + sqlDataReader["firstNameEmployee"].ToString().Trim() + " " +
sqlDataReader["middleNameEmployee"].ToString().Trim() + " " +
sqlDataReader["iEmployee"].ToString().Trim());
            }
```

```
            sqlDataReader.Close();
            connection.Close();

            comboBox2Seller.SelectedIndex = 0;
            comboBox3Spivr.SelectedIndex = 0;
        }

        private void InitClientDropdown()
        {
            comboBox2Client.Items.Clear();

            SqlConnection connection = CreateConnection();
            SqlCommand commandClient = new SqlCommand("SELECT iKlnt,
lastNameClient, firstNameClient, middleNameClient FROM [Client]");
            commandClient.Connection = connection;
            SqlDataReader sqlDataReader = commandClient.ExecuteReader();
            while (sqlDataReader.Read())
            {
                //comboBox2Client.Items.Add("(" +
sqlDataReader["iKlnt"].ToString().Trim() + ") " +
sqlDataReader["lastNameClient"].ToString().Trim() + " " +
sqlDataReader["firstNameClient"].ToString().Trim() + " " +
sqlDataReader["middleNameClient"].ToString().Trim());

comboBox2Client.Items.Add(sqlDataReader["lastNameClient"].ToString().Trim() +
" " + sqlDataReader["firstNameClient"].ToString().Trim() + " " +
sqlDataReader["middleNameClient"].ToString().Trim() +
sqlDataReader["iKlnt"].ToString().Trim());
            }
            sqlDataReader.Close();
            connection.Close();

            comboBox2Client.SelectedIndex = 0;
        }

        //get client sale value
        private void comboBox2Client_SelectionChangeCommitted(object sender,
EventArgs e)
        {
            int comboBox2ClientIndex = comboBox2Client.SelectedIndex;
            string comboBox2ClientId =
comboBox2Client.Items[comboBox2ClientIndex].ToString();
            string iKlnt =
comboBox2ClientId.Substring(comboBox2ClientId.Length - 5, 5);
            //textBox2Info.Text = iKlnt;
            SqlConnection connection = CreateConnection();
            SqlCommand commandClient = new SqlCommand("SELECT discountClient
FROM [Client] WHERE iKlnt = @iKlnt ");
            commandClient.Parameters.Add(new SqlParameter("iKlnt", iKlnt));
            commandClient.Connection = connection;
            SqlDataReader sqlDataReader = commandClient.ExecuteReader();

            while (sqlDataReader.Read())
            {
                numericUpDown2Znyzhka.Value =
int.Parse(sqlDataReader["discountClient"].ToString().Trim());
            }
            sqlDataReader.Close();
            connection.Close();

            numericUpDown2Vartist.Value =
Math.Round(numericUpDown2Vartist.Value * (100 - numericUpDown2Znyzhka.Value)
/ 100);
```

```
        }

        private void button3Add_Click(object sender, EventArgs e)
        {

                string comboBox3SpivrId =
comboBox3Spivr.Items[comboBox2Seller.SelectedIndex].ToString();
                int iEmployee =
int.Parse(comboBox3SpivrId.Substring(comboBox3SpivrId.Length - 4, 4));

                decimal postavka = numericUpDown3Kilkist.Value;
                textBox3Nayavnist.Text =
Convert.ToString(int.Parse(textBox3Nayavnist.Text) + postavka);

                int currentRow = 0;
                try
                {
                    currentRow = dataGridView7.CurrentRow.Index;
                }
                catch { }

                SqlConnection connection = new SqlConnection();
                connection.ConnectionString = connectionString;
                connection.Open();
                SqlCommand command = new SqlCommand();
                command.Connection = connection;

            command.CommandText = "INSERT INTO [Supply] (dateSupply,
timeSupply, iTovar, bookTitle, author, price, count, iEmployee, notes) VALUES
(@dateSupply, @timeSupply, @iTovar, @bookTitle, @author, @price, @count,
@iEmployee, @notes)";
            command.Parameters.Add(new SqlParameter("dateSupply",
getDate()));
            command.Parameters.Add(new SqlParameter("timeSupply",
getTime()));
            command.Parameters.Add(new SqlParameter("iTovar",
textBox3Tovar.Text));
            command.Parameters.Add(new SqlParameter("bookTitle",
textBox3Nazva.Text));
            command.Parameters.Add(new SqlParameter("author",
textBox3Avtor.Text));
            command.Parameters.Add(new SqlParameter("price",
numericUpDown3Vartist.Value));
            command.Parameters.Add(new SqlParameter("count", postavka));
            command.Parameters.Add(new SqlParameter("iEmployee", iEmployee));
            command.Parameters.Add(new SqlParameter("notes",
textBox3Notes.Text));
            command.ExecuteNonQuery();
            connection.Close();

            connection = new SqlConnection();
            connection.ConnectionString = connectionString;
            connection.Open();
            command = new SqlCommand();
            command.Connection = connection;

            //make INSERT query
            command.CommandText = "UPDATE [Product] SET count = @count WHERE
iTovar = @iTovar";
                command.Parameters.Add(new SqlParameter("iTovar",
textBox3Tovar.Text));
                command.Parameters.Add(new SqlParameter("count",
textBox3Nayavnist.Text));
```

```
                command.ExecuteNonQuery();


            command.ExecuteNonQuery();
            connection.Close();

            //show new row (refresh DataGrid)
            dataset = new DataSet();
                SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM
[Supply]", connection);
                adapter.Fill(dataset);
                dataGridView7.DataSource = dataset.Tables[0];

                //move to row selected before
                dataGridView7.CurrentCell = dataGridView7[0, currentRow];

            dataset = new DataSet();
            adapter = new SqlDataAdapter("SELECT * FROM [Product]",
connection);
            adapter.Fill(dataset);
            dataGridView3.DataSource = dataset.Tables[0];
        }

        private void button3AddExisting_Click(object sender, EventArgs e)
        {
            int currentRow;
            try
            {
                currentRow = dataGridView3.CurrentRow.Index;
            }
            catch
            {
                currentRow = 0;
            }

            SqlConnection connection = new SqlConnection();
            connection.ConnectionString = connectionString;
            connection.Open();
            SqlCommand command = new SqlCommand();
            command.Connection = connection;

            command.CommandText = "UPDATE [Product] SET count = @count WHERE
iTovar = @iTovar";
            command.Parameters.Add(new SqlParameter("iTovar",
textBox1Tovar.Text));
            command.Parameters.Add(new SqlParameter("count",
numericUpDown2Nayavnist.Text));
            command.ExecuteNonQuery();
            connection.Close();

            DataSet dataset = new DataSet();
            SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM
[Product]", connection);
            adapter.Fill(dataset);
            dataGridView2.DataSource = dataset.Tables[0];

            dataGridView2.CurrentCell = dataGridView2[0, currentRow];
            comboBox3Spivr.SelectedIndex = 0;
        }

        private void button3Edit_Click(object sender, EventArgs e)
        {
            int currentRow = dataGridView7.CurrentRow.Index;
```

```
            string comboBox3SpivrId =
comboBox3Spivr.Items[comboBox3Spivr.SelectedIndex].ToString();
            int iEmployee =
int.Parse(comboBox3SpivrId.Substring(comboBox3SpivrId.Length - 4, 4));

            //open connection
            SqlConnection connection = new SqlConnection();
            connection.ConnectionString = connectionString;
            connection.Open();
            SqlCommand command = new SqlCommand();
            command.Connection = connection;

            //make Update query
            command.CommandText = "UPDATE [Supply] SET dateSupply =
@dateSupply, timeSupply = @timeSupply, bookTitle = @bookTitle, author =
@author, price = @price, count = @count, iEmployee = @iEmployee, notes =
@notes WHERE iSupply = @iSupply";
            command.Parameters.Add(new SqlParameter("iSupply",
textBox3Supply.Text));
            command.Parameters.Add(new SqlParameter("dateSupply",
getDate()));
            command.Parameters.Add(new SqlParameter("timeSupply",
getTime()));
            command.Parameters.Add(new SqlParameter("iTovar",
textBox3Tovar.Text));
            command.Parameters.Add(new SqlParameter("bookTitle",
textBox3Nazva.Text));
            command.Parameters.Add(new SqlParameter("author",
textBox3Avtor.Text));
            command.Parameters.Add(new SqlParameter("price",
numericUpDown3Vartist.Value));
            command.Parameters.Add(new SqlParameter("count",
numericUpDown3Kilkist.Value));
            command.Parameters.Add(new SqlParameter("iEmployee", iEmployee));
            command.Parameters.Add(new SqlParameter("notes",
textBox3Notes.Text));
            command.ExecuteNonQuery();
            connection.Close();

            //show new row (refresh DataGrid)
            DataSet dataset = new DataSet();
            SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM
[Supply]", connection);
            adapter.Fill(dataset);
            dataGridView7.DataSource = dataset.Tables[0];

            dataGridView7.CurrentCell = dataGridView7[0, currentRow];
        }

        private void button3Delete_Click(object sender, EventArgs e)
        {
            {
                DialogResult dialogResult = MessageBox.Show("Ви впевнені?",
"", MessageBoxButtons.OKCancel);
                if (dialogResult == DialogResult.OK)
                {
                    //open connection
                    SqlConnection connection = new SqlConnection();
                    connection.ConnectionString = connectionString;
                    connection.Open();
                    SqlCommand command = new SqlCommand();
                    command.Connection = connection;
```

```
                    command.CommandText = "DELETE FROM [Supply] WHERE iSupply
= @iSupply";
                    command.Parameters.Add(new SqlParameter("iSupply",
textBox3Supply.Text));
                    command.Connection = connection;
                    command.ExecuteNonQuery();
                    connection.Close();

                    //show new row (refresh DataGrid)
                    DataSet dataset = new DataSet();
                    SqlDataAdapter adapter = new SqlDataAdapter("SELECT *
FROM [Supply]", connection);
                    adapter.Fill(dataset);
                    dataGridView7.DataSource = dataset.Tables[0];
                }
            }
        }

        private void dataGridView3_CellEnter_1(object sender,
DataGridViewCellEventArgs e)
        {
            int indexRow = dataGridView3.CurrentRow.Index;
            dataGridView3.Rows[indexRow].Selected = true;
            dataGridView3.MultiSelect = false;
            //checking for last row
            if (dataGridView3[0,
dataGridView3.CurrentRow.Index].Value.ToString().Trim() == "")
            {
                textBox3Tovar.Clear();
                textBox3Nazva.Clear();
                textBox3Avtor.Clear();
                numericUpDown3Kilkist.Text = "0";
                numericUpDown2Vartist.Text = "0";
                textBox3Notes.Clear();
            }
            else
            {
                if (dataGridView3.CurrentRow.Cells["iTovar"].Value == null)
                {
                    textBox3Tovar.Clear();
                }
                else
                {
                    textBox3Tovar.Text =
dataGridView3.CurrentRow.Cells["iTovar"].Value.ToString(); ;
                }

                //get bookTitle value
                if (dataGridView3.CurrentRow.Cells["bookTitle"].Value ==
null)
                {
                    textBox3Nazva.Text = "";
                }
                else
                {
                    textBox3Nazva.Text =
dataGridView3.CurrentRow.Cells["bookTitle"].Value.ToString();
                }

                //get author value
                if (dataGridView3.CurrentRow.Cells["author"].Value == null)
                {
```

```
                    textBox3Avtor.Text = "";
                }
                else
                {
                    textBox3Avtor.Text =
dataGridView3.CurrentRow.Cells["author"].Value.ToString();
                }

                if (dataGridView3.CurrentRow.Cells["cost"].Value == null)
                {
                    numericUpDown3Vartist.Text = "";
                }
                else
                {
                    numericUpDown3Vartist.Text =
dataGridView3.CurrentRow.Cells["cost"].Value.ToString();
                }

                if (dataGridView3.CurrentRow.Cells["count"].Value == null)
                {
                    textBox3Nayavnist.Text = "";
                }
                else
                {
                    textBox3Nayavnist.Text =
dataGridView3.CurrentRow.Cells["count"].Value.ToString();
                }


            }
        }

        private void dataGridView7_CellEnter(object sender,
DataGridViewCellEventArgs e)
        {
            int indexRow = dataGridView7.CurrentRow.Index;
            dataGridView7.Rows[indexRow].Selected = true;
            dataGridView7.MultiSelect = false;
            //checking for last row
            if (dataGridView7[0,
dataGridView7.CurrentRow.Index].Value.ToString().Trim() == "")
            {
                //textBox3Tovar.Clear();
                textBox3Nazva.Clear();
                textBox3Avtor.Clear();
                //comboBox3Spivr.SelectedIndex = 0;
                numericUpDown3Kilkist.Text = "0";
                numericUpDown3Vartist.Text = "0";
                textBox3Notes.Clear();
            }
            else
            {
                if (dataGridView7.CurrentRow.Cells["iTovar"].Value == null)
                {
                    textBox3Tovar.Clear();
                }
                else
                {
                    textBox3Tovar.Text =
dataGridView7.CurrentRow.Cells["iTovar"].Value.ToString(); ;
                }

                //get bookTitle value
```

```
                if (dataGridView7.CurrentRow.Cells["bookTitle"].Value ==
null)
                {
                    textBox3Nazva.Text = "";
                }
                else
                {
                    textBox3Nazva.Text =
dataGridView7.CurrentRow.Cells["bookTitle"].Value.ToString();
                }

                //get author value
                if (dataGridView7.CurrentRow.Cells["author"].Value == null)
                {
                    textBox3Avtor.Text = "";
                }
                else
                {
                    textBox3Avtor.Text =
dataGridView7.CurrentRow.Cells["author"].Value.ToString();
                }

                if (dataGridView7.CurrentRow.Cells["iSupply"].Value == null)
                {
                    textBox3Supply.Text = "";
                }
                else
                {
                    textBox3Supply.Text =
dataGridView7.CurrentRow.Cells["iSupply"].Value.ToString();
                }

                if (dataGridView7.CurrentRow.Cells["count"].Value == null)
                {
                    numericUpDown3Kilkist.Text = "";
                }
                else
                {
                    numericUpDown3Kilkist.Text =
dataGridView7.CurrentRow.Cells["count"].Value.ToString();
                }

                if (dataGridView7.CurrentRow.Cells["price"].Value == null)
                {
                    numericUpDown3Vartist.Value = 0;
                }
                else
                {
                    numericUpDown3Vartist.Text =
dataGridView7.CurrentRow.Cells["price"].Value.ToString();
                }

            }
        }

        private void button4Add_Click(object sender, EventArgs e)
        {
            int currentRow = 0;
            try
            {
                currentRow = dataGridView4.CurrentRow.Index;
            }
            catch { }
```

```
            SqlConnection connection = new SqlConnection();
            connection.ConnectionString = connectionString;
            connection.Open();
            SqlCommand command = new SqlCommand();
            command.Connection = connection;

            command.CommandText = "INSERT INTO [Employee] (lastNameEmployee,
firstNameEmployee, middleNameEmployee, phoneEmployee, emailEmployee,
birthdayEmployee, infoEmployee) VALUES (@lastNameEmployee,
@firstNameEmployee, @middleNameEmployee, @phoneEmployee, @emailEmployee,
@birthdayEmployee, @infoEmployee)";
            // command.Parameters.Add(new SqlParameter("iTovar",
textBox1Tovar.Text));

            command.Parameters.Add(new SqlParameter("lastNameEmployee",
textBox4Prizv.Text));
            command.Parameters.Add(new SqlParameter("firstNameEmployee",
textBox4Imya.Text));
            command.Parameters.Add(new SqlParameter("middleNameEmployee",
textBox4PoBatkovi.Text));
            command.Parameters.Add(new SqlParameter("phoneEmployee",
textBox4Phone.Text));
            command.Parameters.Add(new SqlParameter("emailEmployee",
textBox4Email.Text));
            command.Parameters.Add(new SqlParameter("birthdayEmployee",
dateTimePicker4DataNarodzh.Value));
            command.Parameters.Add(new SqlParameter("infoEmployee",
textBox4Info.Text));

            command.ExecuteNonQuery();
            connection.Close();

            //show new row (refresh DataGrid)
            DataSet dataset = new DataSet();
            SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM
[Employee]", connection);
            adapter.Fill(dataset);
            dataGridView4.DataSource = dataset.Tables[0];

            //move to row selected before
            try
            { dataGridView4.CurrentCell = dataGridView4[0, currentRow]; }
            catch
            {}
        }

        private void button4Edit_Click(object sender, EventArgs e)
        {
            int currentRow = dataGridView1.CurrentRow.Index;

            //open connection
            SqlConnection connection = new SqlConnection();
            connection.ConnectionString = connectionString;
            connection.Open();
            SqlCommand command = new SqlCommand();
            command.Connection = connection;

            //make Update query
            command.CommandText = "UPDATE [Employee] SET lastNameEmployee =
@lastNameEmployee, firstNameEmployee = @firstNameEmployee, middleNameEmployee
= @middleNameEmployee, phoneEmployee = @phoneEmployee, emailEmployee =
```

```
@emailEmployee, birthdayEmployee = @birthdayEmployee, infoEmployee =
@infoEmployee WHERE iEmployee = @iEmployee";
            command.Parameters.Add(new SqlParameter("iEmployee",
textBox4Spivr.Text));
            command.Parameters.Add(new SqlParameter("lastNameEmployee",
textBox4Prizv.Text));
            command.Parameters.Add(new SqlParameter("firstNameEmployee",
textBox4Imya.Text));
            command.Parameters.Add(new SqlParameter("middleNameEmployee",
textBox4PoBatkovi.Text));
            command.Parameters.Add(new SqlParameter("phoneEmployee",
textBox4Phone.Text));
            command.Parameters.Add(new SqlParameter("emailEmployee",
textBox4Email.Text));
            command.Parameters.Add(new SqlParameter("birthdayEmployee",
dateTimePicker4DataNarodzh.Value));
            command.Parameters.Add(new SqlParameter("infoEmployee",
textBox4Info.Text));
            command.ExecuteNonQuery();
            connection.Close();

            //show new row (refresh DataGrid)
            DataSet dataset = new DataSet();
            SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM
[Employee]", connection);
            adapter.Fill(dataset);
            dataGridView4.DataSource = dataset.Tables[0];

            dataGridView4.CurrentCell = dataGridView4[0, currentRow];
        }

        private void button4Delete_Click(object sender, EventArgs e)
        {
            {
                DialogResult dialogResult = MessageBox.Show("Ви впевненi?",
"", MessageBoxButtons.OKCancel);
                if (dialogResult == DialogResult.OK)
                {
                    //open connection
                    SqlConnection connection = new SqlConnection();
                    connection.ConnectionString = connectionString;
                    connection.Open();
                    SqlCommand command = new SqlCommand();
                    command.Connection = connection;

                    command.CommandText = "DELETE FROM [Employee] WHERE
iEmployee = @iEmployee";
                    command.Parameters.Add(new SqlParameter("iEmployee",
textBox4Spivr.Text));
                    command.Connection = connection;
                    command.ExecuteNonQuery();
                    connection.Close();

                    //show new row (refresh DataGrid)
                    DataSet dataset = new DataSet();
                    SqlDataAdapter adapter = new SqlDataAdapter("SELECT *
FROM [Employee]", connection);
                    adapter.Fill(dataset);
                    dataGridView4.DataSource = dataset.Tables[0];
                }
            }
        }
```

```
        private void dataGridView4_CellEnter(object sender,
DataGridViewCellEventArgs e)
        {
            int indexRow = dataGridView4.CurrentRow.Index;
            dataGridView4.Rows[indexRow].Selected = true;
            dataGridView4.MultiSelect = false;
            //checking for last row
            if (dataGridView4[0,
dataGridView4.CurrentRow.Index].Value.ToString().Trim() == "")
            {
                textBox4Spivr.Clear();
                textBox4Prizv.Clear();
                textBox4Imya.Clear();
                textBox4PoBatkovi.Clear();
                textBox4Phone.Clear();
                textBox4Email.Clear();
                //dateTimePicker4DataNarodzh
                textBox4Info.Clear();
            }
            else
            {
                if (dataGridView4.CurrentRow.Cells["iEmployee"].Value ==
null)
                {
                    textBox4Spivr.Clear();
                }
                else
                {
                    textBox4Spivr.Text =
dataGridView4.CurrentRow.Cells["iEmployee"].Value.ToString();
                }

                if (dataGridView4.CurrentRow.Cells["lastNameEmployee"].Value
== null)
                {
                    textBox4Prizv.Clear();
                }
                else
                {

                    textBox4Prizv.Text =
dataGridView4.CurrentRow.Cells["lastNameEmployee"].Value.ToString();
                }

                //get author value
                if (dataGridView4.CurrentRow.Cells["firstNameEmployee"].Value
== null)
                {
                    textBox4Imya.Clear();
                }
                else
                {
                    textBox4Imya.Text =
dataGridView4.CurrentRow.Cells["firstNameEmployee"].Value.ToString();
                }

                if
(dataGridView4.CurrentRow.Cells["middleNameEmployee"].Value == null)
                {
                    textBox4PoBatkovi.Clear();
                }
                else
                {
```

```
                    textBox4PoBatkovi.Text =
dataGridView4.CurrentRow.Cells["middleNameEmployee"].Value.ToString();
                }

                if (dataGridView4.CurrentRow.Cells["phoneEmployee"].Value ==
null)
                {
                    textBox4Phone.Clear();
                }
                else
                {
                    textBox4Phone.Text =
dataGridView4.CurrentRow.Cells["phoneEmployee"].Value.ToString();
                }

                if (dataGridView4.CurrentRow.Cells["emailEmployee"].Value ==
null)
                {
                    textBox4Email.Clear();
                }
                else
                {
                    textBox4Email.Text =
dataGridView4.CurrentRow.Cells["emailEmployee"].Value.ToString();
                }

                if (dataGridView4.CurrentRow.Cells["birthdayEmployee"].Value
!= null)
                {
                    dateTimePicker4DataNarodzh.Value =
DateTime.Parse(dataGridView4.CurrentRow.Cells["birthdayEmployee"].Value.ToStr
ing());
                }

                if (dataGridView4.CurrentRow.Cells["infoEmployee"].Value ==
null)
                {
                    textBox4Info.Text = "";
                }
                else
                {
                    textBox4Info.Text =
dataGridView4.CurrentRow.Cells["infoEmployee"].Value.ToString();
                }
            }
        }

        private void button4Next_Click(object sender, EventArgs e)
        {
            int index = dataGridView4.CurrentRow.Index;
            dataGridView4.Rows[index].Selected = true;
            try
            {
                dataGridView4.CurrentCell = dataGridView4[0, index + 1];
            }
            catch {}
        }

        private void button4Prev_Click(object sender, EventArgs e)
        {
            int index = dataGridView4.CurrentRow.Index;
            int indexMax = dataGridView4.Rows.Count;
            dataGridView4.Rows[index].Selected = true;
```

```
            try
            {
                dataGridView4.CurrentCell = dataGridView4[0, index - 1];
            }
            catch {}
        }

        private void button4First_Click(object sender, EventArgs e)
        {
            int indexMax = dataGridView4.Rows.Count;
            dataGridView4.CurrentCell = dataGridView4[0, 0];
        }

        private void button4Last_Click(object sender, EventArgs e)
        {
            int indexMax = dataGridView4.Rows.Count;
            dataGridView4.CurrentCell = dataGridView4[0, indexMax - 1];
        }

        private void button5Add_Click(object sender, EventArgs e)
        {
            int currentRow = 0;
            try
            {
                currentRow = dataGridView5.CurrentRow.Index;
            }
            catch { }

            //int currentRow = dataGridView5.CurrentRow.Index;

            SqlConnection connection = new SqlConnection();
            connection.ConnectionString = connectionString;
            connection.Open();
            SqlCommand command = new SqlCommand();
            command.Connection = connection;

            command.CommandText = "INSERT INTO [Client] (lastNameClient,
firstNameClient, middleNameClient, phoneClient, emailClient, birthdayClient,
discountClient, infoClient) VALUES (@lastNameClient, @firstNameClient,
@middleNameClient, @phoneClient, @emailClient, @birthdayClient,
@discountClient, @infoClient)";
            command.Parameters.Add(new SqlParameter("lastNameClient",
textBox5Prizv.Text));
            command.Parameters.Add(new SqlParameter("firstNameClient",
textBox5Imya.Text));
            command.Parameters.Add(new SqlParameter("middleNameClient",
textBox5PoBatkovi.Text));
            command.Parameters.Add(new SqlParameter("phoneClient",
textBox5Phone.Text));
            command.Parameters.Add(new SqlParameter("emailClient",
textBox5Email.Text));
            command.Parameters.Add(new SqlParameter("birthdayClient",
dateTimePicker5DataNarodzh.Value));
            command.Parameters.Add(new SqlParameter("discountClient",
numericUpDown5Discount.Text));
            command.Parameters.Add(new SqlParameter("infoClient",
textBox5Info.Text));

            command.ExecuteNonQuery();
            connection.Close();

            //show new row (refresh DataGrid)
            DataSet dataset = new DataSet();
```

```
            SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM
[Client]", connection);
            adapter.Fill(dataset);
            dataGridView5.DataSource = dataset.Tables[0];

            //move to row selected before
            dataGridView5.CurrentCell = dataGridView5[0, currentRow];
        }

        private void button5Edit_Click(object sender, EventArgs e)
        {
            int currentRow = dataGridView5.CurrentRow.Index;

            //open connection
            SqlConnection connection = new SqlConnection();
            connection.ConnectionString = connectionString;
            connection.Open();
            SqlCommand command = new SqlCommand();
            command.Connection = connection;

            //make Update query
            command.CommandText = "UPDATE [Client] SET lastNameClient =
@lastNameClient, firstNameClient = @firstNameClient, middleNameClient =
@middleNameClient, phoneClient = @phoneClient, emailClient = @emailClient,
birthdayClient = @birthdayClient, discountClient = @discountClient,
infoClient = @infoClient WHERE iKlnt = @iKlnt";

            command.Parameters.Add(new SqlParameter("iKlnt",
textBox5Client.Text));
            command.Parameters.Add(new SqlParameter("lastNameClient",
textBox5Prizv.Text));
            command.Parameters.Add(new SqlParameter("firstNameClient",
textBox5Imya.Text));
            command.Parameters.Add(new SqlParameter("middleNameClient",
textBox5PoBatkovi.Text));
            command.Parameters.Add(new SqlParameter("phoneClient",
textBox5Phone.Text));
            command.Parameters.Add(new SqlParameter("emailClient",
textBox5Email.Text));
            command.Parameters.Add(new SqlParameter("birthdayClient",
dateTimePicker5DataNarodzh.Value));
            command.Parameters.Add(new SqlParameter("discountClient",
numericUpDown5Discount.Text));
            command.Parameters.Add(new SqlParameter("infoClient",
textBox5Info.Text));
            command.ExecuteNonQuery();
            connection.Close();

            //show new row (refresh DataGrid)
            DataSet dataset = new DataSet();
            SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM
[Client]", connection);
            adapter.Fill(dataset);
            dataGridView5.DataSource = dataset.Tables[0];
            dataGridView5.CurrentCell = dataGridView5[0, currentRow];
        }

        private void button5Delete_Click(object sender, EventArgs e)
        {
            {
                DialogResult dialogResult = MessageBox.Show("Ви впевнені?",
"", MessageBoxButtons.OKCancel);
                if (dialogResult == DialogResult.OK)
```

```
                {
                    //open connection
                    SqlConnection connection = new SqlConnection();
                    connection.ConnectionString = connectionString;
                    connection.Open();
                    SqlCommand command = new SqlCommand();
                    command.Connection = connection;

                    command.CommandText = "DELETE FROM [Client] WHERE iKlnt =
@iKlnt";
                    command.Parameters.Add(new SqlParameter("iKlnt",
textBox5Client.Text));
                    command.Connection = connection;
                    command.ExecuteNonQuery();
                    connection.Close();

                    //show new row (refresh DataGrid)
                    DataSet dataset = new DataSet();
                    SqlDataAdapter adapter = new SqlDataAdapter("SELECT *
FROM [Client]", connection);
                    adapter.Fill(dataset);
                    dataGridView5.DataSource = dataset.Tables[0];
                }
            }
        }

        private void dataGridView5_CellEnter(object sender,
DataGridViewCellEventArgs e)
        {
            int indexRow = dataGridView5.CurrentRow.Index;
            dataGridView5.Rows[indexRow].Selected = true;
            dataGridView5.MultiSelect = false;
            //checking for last row
            if (dataGridView5[0,
dataGridView5.CurrentRow.Index].Value.ToString().Trim() == "")
            {
                textBox5Client.Clear();
                textBox5Prizv.Clear();
                textBox5Imya.Clear();
                textBox5PoBatkovi.Clear();
                textBox5Phone.Clear();
                textBox5Email.Clear();
                numericUpDown5Discount.Value = 0;
                textBox5Info.Clear();

            }
            else
            {
                if (dataGridView5.CurrentRow.Cells["iKlnt"].Value == null)
                {
                    textBox5Client.Clear();
                }
                else
                {
                    textBox5Client.Text =
dataGridView5.CurrentRow.Cells["iKlnt"].Value.ToString();
                }

                if (dataGridView5.CurrentRow.Cells["lastNameClient"].Value ==
null)
                {
                    textBox5Prizv.Clear();
                }
```

```
                else
                {

                    textBox5Prizv.Text =
dataGridView5.CurrentRow.Cells["lastNameClient"].Value.ToString();
                }

                if (dataGridView5.CurrentRow.Cells["firstNameClient"].Value
== null)
                {
                    textBox5Imya.Clear();
                }
                else
                {
                    textBox5Imya.Text =
dataGridView5.CurrentRow.Cells["firstNameClient"].Value.ToString();
                }

                if (dataGridView5.CurrentRow.Cells["middleNameClient"].Value
== null)
                {
                    textBox5PoBatkovi.Clear();
                }
                else
                {
                    textBox5PoBatkovi.Text =
dataGridView5.CurrentRow.Cells["middleNameClient"].Value.ToString();
                }

                if (dataGridView5.CurrentRow.Cells["phoneClient"].Value ==
null)
                {
                    textBox5Phone.Clear();
                }
                else
                {
                    textBox5Phone.Text =
dataGridView5.CurrentRow.Cells["phoneClient"].Value.ToString();
                }

                if (dataGridView5.CurrentRow.Cells["emailClient"].Value ==
null)
                {
                    textBox5Email.Clear();
                }
                else
                {
                    textBox5Email.Text =
dataGridView5.CurrentRow.Cells["emailClient"].Value.ToString();
                }

                if (dataGridView5.CurrentRow.Cells["birthdayClient"].Value !=
null)
                {
                    dateTimePicker5DataNarodzh.Value =
DateTime.Parse(dataGridView5.CurrentRow.Cells["birthdayClient"].Value.ToStrin
g());
                }

                if (dataGridView5.CurrentRow.Cells["emailClient"].Value ==
null)
                {
                    textBox5Email.Clear();
```

```
                }
                else
                {
                    textBox5Email.Text =
dataGridView5.CurrentRow.Cells["emailClient"].Value.ToString();
                }

                if (dataGridView5.CurrentRow.Cells["discountClient"].Value ==
null)
                {
                    numericUpDown5Discount.Value = 0;
                }
                else
                {
                    numericUpDown5Discount.Text =
dataGridView5.CurrentRow.Cells["discountClient"].Value.ToString();
                }

                if (dataGridView5.CurrentRow.Cells["infoClient"].Value ==
null)
                {
                    textBox5Info.Clear();
                }
                else
                {
                    textBox5Info.Text =
dataGridView5.CurrentRow.Cells["infoClient"].Value.ToString();
                }
            }
        }

        private void button5Next_Click(object sender, EventArgs e)
        {
            int index = dataGridView5.CurrentRow.Index;
            dataGridView5.Rows[index].Selected = true;
            try
            {
                dataGridView5.CurrentCell = dataGridView5[0, index + 1];
            }
            catch {}
        }

        private void button5Prev_Click(object sender, EventArgs e)
        {
            int index = dataGridView5.CurrentRow.Index;
            int indexMax = dataGridView5.Rows.Count;
            dataGridView5.Rows[index].Selected = true;
            try
            {
                dataGridView5.CurrentCell = dataGridView5[0, index - 1];
            }
            catch { }
        }

        private void button5First_Click(object sender, EventArgs e)
        {
            int indexMax = dataGridView5.Rows.Count;
            dataGridView5.CurrentCell = dataGridView5[0, 0];
        }

        private void button5Last_Click(object sender, EventArgs e)
        {
            int indexMax = dataGridView5.Rows.Count;
```

```
            dataGridView5.CurrentCell = dataGridView5[0, indexMax - 1];
        }

        private void comboBox2Client_Click(object sender, EventArgs e)
        {
            InitClientDropdown();
        }

        private void comboBox2Seller_Click(object sender, EventArgs e)
        {
            InitEmployeeDropdown();
        }

        private void comboBox3Spivr_Click(object sender, EventArgs e)
        {
            InitEmployeeDropdown();
        }
    }
}
```