

**Міністерство освіти і науки України
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення**

**КВАЛІФІКАЦІЙНА РОБОТА
ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «БАКАЛАВР»**

**РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ ПРОДАЖАМИ ТА
ЗАКУПІВЛЯМИ ВИКОРИСТОВУЮЧИ C# ТА .NET**

**DEVELOPMENT OF A SALES AND PURCHASING MANAGEMENT
SYSTEM USING C# AND .NET**

спеціальність 121 «Інженерія програмного забезпечення»
освітня програма «Інженерія програмного забезпечення»

Виконав: здобувач вищої освіти
групи ІПЗ-42
Титов Максим Сергійович

Керівник:
Сичук Віктор Анатолійович

Кваліфікаційну роботу
допущено до захисту
«__» _____ 2026 р.
Гарант освітньої програми:
к.т.н., доцент
Ліщина Наталія Миколаївна

Луцьк – 2026 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних та інформаційних технологій

Кафедра інженерії програмного забезпечення

Ступінь вищої освіти бакалавр

Галузь знань: 12 «Інформаційні технології»

Спеціальність: 121 «Інженерія програмного забезпечення»

Освітня програма: «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
ЗАВІДУВАЧ КАФЕДРИ

«__» _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Титову Максиму Сергійовичу

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «Розробка системи управління продажами та закупівлями використовуючи C# та .NET»

Керівник роботи: к.т.н. доцент Сичук В. А.

затверджені наказом закладу вищої освіти від «31» грудня 2025 р. № 541/01-02

2. Строк подання здобувачем вищої освіти кваліфікаційної роботи бакалавра _____
«29» травня 2026 р.

3. Вихідні дані до роботи: C#, .NET, Entity Framework Core, Microsoft SQL Server LocalDB, Visual Studio, вимоги до системи управління продажами, закупівлями та складськими залишками, методичні вказівки до виконання кваліфікаційної роботи бакалавра.

4. Зміст розрахунково-пояснювальної записки (перелік питань, що потрібно розробити): аналіз сучасного стану проблем автоматизації обліку персоналу та управління задачами, постановка завдання, формування вимог до розроблюваної інформаційної системи, обґрунтування вибору технологій та засобів розробки, практична реалізація об'єкта проєктування, розробка структури бази даних, обґрунтування принципів захисту інформаційно-комп'ютерної системи, тестування та налагодження програмного забезпечення.

5. Перелік графічного матеріалу: 19 рисунків, 8 лістингів, 7 таблиць.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз предметної області	Сичук В. А.		
Специфікація вимог до розробленої системи	Сичук В. А.		
Розробка об'єкта проектування	Сичук В. А.		
Нормоконтроль	Суринович О. М.		
Гарант ОП	Ліщина Н. М.		
Показник запозичень тексту	%		
Академічна доброчесність	Сичук В. А.		

7. Дата видачі завдання «3» лютого 2026 р.

№	Назва етапів кваліфікаційної роботи бакалавра	Строк виконання етапів роботи	Примітка
1	Огляд літературних джерел по темі кваліфікаційної роботи бакалавра	до 14.02.2026 р.	
2	Аналіз проблеми розробки та впровадження об'єкту проектування	до 03.03.2026 р.	
3	Обґрунтування вибору шляхів, технологій і засобів вирішення поставленого завдання	до 14.03.2026 р.	
4	Розробка функціонально-структурної схеми роботи об'єкта проектування та проектування бази даних	до 14.04.2026 р.	
5	Практична реалізація об'єкта проектування та розробка бази даних	до 20.04.2026 р.	
6	Тестування та налагодження об'єкта проектування	до 04.05.2026 р.	
7	Здача чистового варіанту кваліфікаційної роботи бакалавра на кафедрі	до 29.05.2026 р.	

Здобувач вищої освіти

Максим Титов

Керівник кваліфікаційної роботи

Віктор Сичук

АНОТАЦІЯ

Титов М. С. Розробка системи управління продажами та закупівлями використовуючи C# та .NET. Рукопис. Кваліфікаційна робота бакалавра ОП «Інженерія програмного забезпечення» спеціальності 121 «Інженерія програмного забезпечення». Луцький національний технічний університет. Луцьк, 2026.

Кваліфікаційна робота бакалавра складається зі вступу, трьох розділів, висновків, списку використаних джерел і додатків.

У роботі розроблено систему управління продажами та закупівлями з використанням C# та .NET. У першому розділі досліджено предметну область обліку торговельних операцій, проаналізовано потреби автоматизації продажів, закупівель, складського обліку та існуючі програмні рішення. У другому розділі визначено вимоги до системи, описано її функціональні модулі, структуру бази даних, користувацькі ролі та обґрунтовано вибір технологій розробки. У третьому розділі наведено реалізацію модулів роботи з товарами, клієнтами, постачальниками, продажами, закупівлями і звітністю, а також розглянуто захист інформації та результати тестування. Розроблена система забезпечує централізоване ведення обліку продажів і закупівель та підтримує прийняття управлінських рішень.

Ключові слова: продажі, закупівлі, C#, .NET, база даних, складський облік, автоматизація.

ABSTRACT

Tytov M. S. Development of a Sales and Purchasing Management System Using C# and .NET. Manuscript. Qualification work for bachelor's degree in Software Engineering, specialty 121 Software Engineering. Lutsk National Technical University. Lutsk, 2026.

The bachelor's qualification work consists of an introduction, three chapters, conclusions, a list of references, and appendices.

The work presents the development of a sales and purchasing management system using C# and .NET. The first chapter examines the subject area of trade operation accounting, analyzes the need to automate sales, purchases, inventory accounting, and reviews existing software solutions. The second chapter defines the system requirements and describes its functional modules, database structure, user roles, and the choice of development technologies. The third chapter presents the implementation of modules for products, customers, suppliers, sales, purchases, and reporting, as well as information security and testing results. The developed system provides centralized accounting of sales and purchases and supports management decision-making.

Keywords: sales, purchasing, C#, .NET, database, inventory accounting, automation.

ЗМІСТ

ВСТУП.....	7
РОЗДІЛ 1 АНАЛІЗ МЕТОДІВ РОЗРОБКИ системи управління продажами ТА ПОСТАНОВКА ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ	9
1.1 Аналіз сучасного стану проблеми.....	9
1.2 Постановка завдання на кваліфікаційну роботу бакалавра.....	11
РОЗДІЛ 2 СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ	13
2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проектування програмного забезпечення	13
2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання	19
РОЗДІЛ 3 РОЗРОБКА СИСТЕМИ УПРАВЛІННЯ ПРОДАЖАМИ	24
3.1 Практична реалізація об'єкта проектування.....	24
3.2 Розробка бази даних	30
3.3 Захист інформаційно-комп'ютерної системи	33
3.4 Тестування та налагодження інформаційно-комп'ютерної системи.....	35
ВИСНОВКИ	40
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	42
ДОДАТКИ	44

ВСТУП

Актуальність теми. Підприємства, що працюють із товарами, постійно виконують операції закупівлі, продажу, контролю залишків, взаємодії з постачальниками та клієнтами. Якщо такі процеси ведуться вручну або в окремих таблицях, складно підтримувати актуальну кількість товарів на складі, швидко оцінювати обсяги продажів, бачити дефіцитні позиції та контролювати прибутковість. Тому актуальним є створення системи управління продажами та закупівлями, яка об'єднує товарний облік, документи руху товарів, складські залишки й звітність у єдиному настільному застосунку.

Для побудови графічного інтерфейсу системи використано Windows Forms, що дає змогу створювати настільні форми, таблиці, кнопки, панелі навігації та події взаємодії користувача з програмою [1]. Робота з даними реалізована через Entity Framework Core та провайдер SQL Server, який підтримує реляційне зберігання інформації про товари, постачальників, клієнтів, закупівлі, продажі й позиції документів [2]. Для локального розгортання під час розроблення використано SQL Server Express LocalDB, що дозволяє працювати з базою даних без окремого повноцінного серверного встановлення [3].

Мета кваліфікаційної роботи полягає у розробленні системи управління продажами та закупівлями з використанням C# та .NET, яка забезпечує облік товарів, постачальників, клієнтів, документів закупівель і продажів, автоматичне оновлення складських залишків, рольовий доступ і формування звітів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- здійснити аналіз процесів товарного обліку, закупівель, продажів, контролю залишків і взаємодії з контрагентами;
- сформулювати функціональні вимоги до настільної системи з урахуванням ролей адміністратора, менеджера та користувача з правами перегляду;
- спроектувати структуру бази даних для зберігання товарів, постачальників, клієнтів, закупівель, продажів і позицій документів;

- обґрунтувати вибір C#, .NET, Windows Forms, Entity Framework Core та MS SQL Server LocalDB для реалізації системи;

- реалізувати модулі довідників, закупівель, продажів, складського обліку, панелі показників, звітів і керування користувачами;

- перевірити працездатність системи на типових сценаріях обліку продажів, закупівель і складських залишків.

Об’єктом дослідження є процеси управління продажами, закупівлями та складськими залишками товарів.

Предметом дослідження є методи та програмні засоби розроблення настільної інформаційної системи товарного обліку з реляційною базою даних, рольовим доступом і звітністю.

Практична цінність розробки полягає у створенні програмного засобу, який дає змогу централізовано вести товари, постачальників і клієнтів, оформлювати закупівлі та продажі, автоматично змінювати залишки на складі, виявляти товари з низьким запасом, оцінювати обсяги операцій і експортувати звітні дані.

РОЗДІЛ 1

АНАЛІЗ МЕТОДІВ РОЗРОБКИ СИСТЕМИ УПРАВЛІННЯ ПРОДАЖАМИ ТА ПОСТАНОВКА ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

1.1 Аналіз сучасного стану проблеми

Системи обліку продажів і закупівель використовуються для контролю товарних залишків, оформлення надходжень, фіксації реалізації, аналізу прибутку та підготовки управлінських звітів. Для розроблюваного застосунку важливими є робота з товарами, постачальниками і клієнтами, автоматичне збільшення залишку під час закупівлі, зменшення під час продажу, перевірка доступної кількості та формування звітів із можливістю експорту.

Одним із аналогів є inFlow Inventory – система для керування запасами, продажами, закупівлями, замовленнями та складськими операціями [4]. На рисунку 1.1 показано приклад інтерфейсу inFlow Inventory.

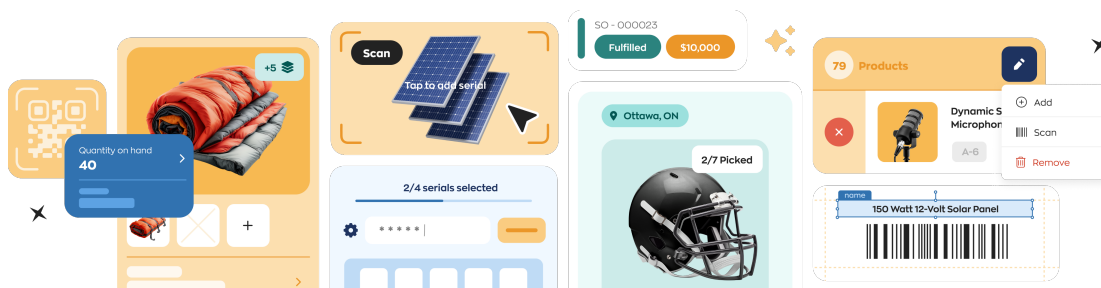


Рисунок 1.1 – Інтерфейс системи inFlow Inventory [4]

Перевагою inFlow є цілісне охоплення складського обліку, продажів, закупівель і звітності. Недоліком для поставленої задачі є орієнтація на готовий комерційний продукт з ширшим набором бізнесфункцій, тоді як кваліфікаційна робота передбачає розроблення локального настільного рішення з контрольованою логікою руху товарів.

Odoo Inventory є модулем великої ERP платформи Odoo і підтримує складські маршрути, операції переміщення, поповнення запасів та інтеграцію з

продажами і закупівлями [5]. Сильною стороною такого підходу є масштабованість і зв'язок з іншими модулями підприємства, але для невеликої системи обліку це створює надмірну складність розгортання та налаштування.

ERPNext також містить модулі для запасів, закупівель, продажів і бухгалтерського обліку, що дозволяє вести бізнесоперації у межах єдиної ERP системи [6]. На рисунку 1.2 наведено приклад інтерфейсу ERPNext.

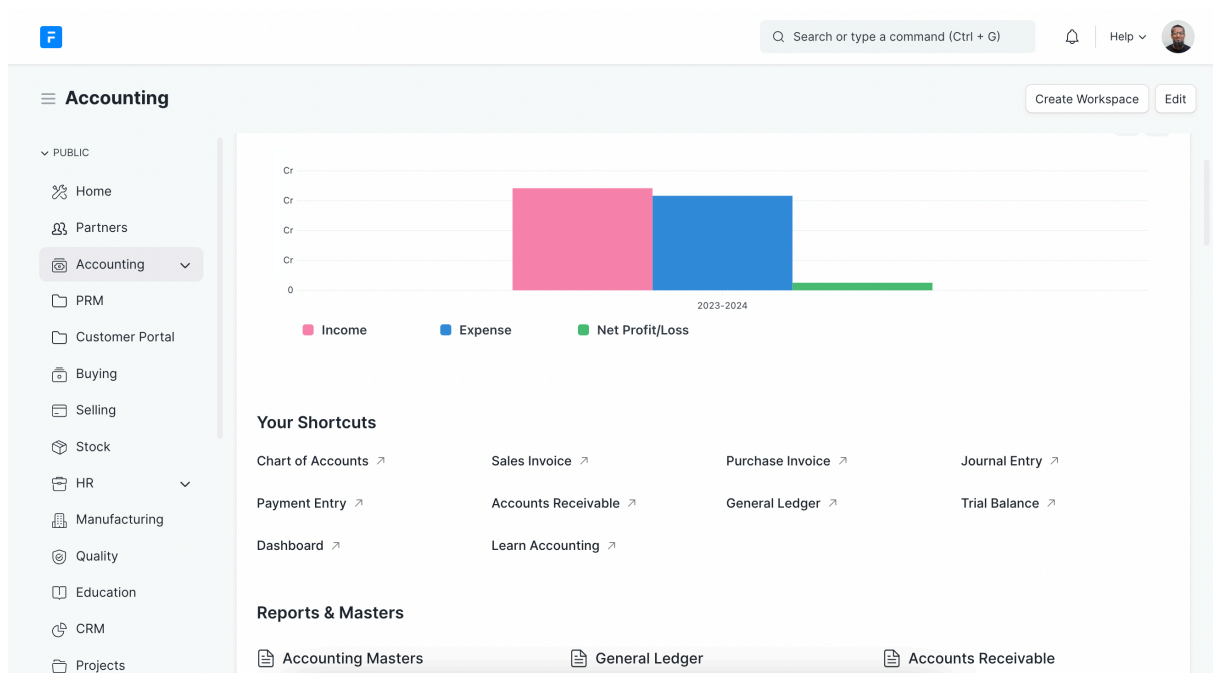


Рисунок 1.2 – Приклад інтерфейсу ERPNext [7]

Перевагою ERPNext є інтеграція обліку товарів з іншими управлінськими процесами підприємства. Водночас така система потребує серверного впровадження, адміністрування та адаптації бізнеслогіки, що є надмірним для локального застосунку, який має забезпечити базові операції закупівлі, продажу, контролю залишків і звітності.

Ще одним близьким рішенням є Zoho Inventory – онлайнсервіс для обліку запасів, замовлень, продажів і поставок [8]. Він зручний для малого бізнесу завдяки вебдоступу та інтеграціям, проте залежить від зовнішнього сервісу і не відповідає задачі створення автономного настільного застосунку з локальною базою даних.

Порівняння аналогів наведено в таблиці 1.1.

Таблиця 1.1 – Порівняння аналогів системи обліку продажів і закупівель

Система	Переваги	Недоліки щодо поставленої задачі
inFlow Inventory	облік запасів, продажів, закупівель, замовлень і звітів	комерційність і надмірність для навчального локального рішення
Odoo Inventory	масштабованість, інтеграція з ERP модулями, складська логіка	складне розгортання та потреба в налаштуванні платформи
ERPNext	єдина ERP система для запасів, закупівель, продажів і фінансів	серверна інфраструктура і ширший функціонал, ніж потрібно для задачі
Zoho Inventory	вебдоступ, інтеграції, підтримка замовлень і запасів	залежність від хмарного сервісу та обмежена автономність
Розроблюваний застосунок	локальна робота, простий WinForms інтерфейс, контроль залишків, звіти, ролі користувачів	не охоплює складні ERP процеси і багатоскладську логіку

Отже, наявні системи підтверджують важливість автоматизації товарного обліку, але здебільшого орієнтуються на вебсервіси або комплексні ERP платформи. Розроблювана система є доцільною як компактне настільне рішення для базового управління закупівлями, продажами, залишками та звітами без потреби у складній інфраструктурі.

1.2 Постановка завдання на кваліфікаційну роботу бакалавра

В результаті аналізу процесів товарного обліку та функціональних можливостей розроблюваної системи було сформовано наступні завдання:

- здійснити аналіз процесів товарного обліку, закупівель, продажів, контролю залишків і взаємодії з контрагентами, визначивши ключові операції менеджера, проблеми ручного ведення документів, потребу в актуальних складських залишках і контролі товарів із недостатнім запасом;

- сформувані функціональні вимоги до настільної системи з урахуванням ролей адміністратора, менеджера та користувача з правами перегляду, передбачивши різний доступ до довідників, документів, звітів, складської інформації, керування користувачами й повторного заповнення демонстраційних даних;

- спроектувати структуру бази даних для зберігання товарів, постачальників, клієнтів, закупівель, продажів і позицій документів, забезпечивши зв'язки між документом і його рядками, збереження цін, кількостей, сум, залишків, мінімального запасу та контактної інформації контрагентів;

- обґрунтувати вибір C#, .NET, Windows Forms, Entity Framework Core та MS SQL Server LocalDB для реалізації системи, враховуючи потребу в настільному інтерфейсі, локальній роботі з базою даних, транзакційному оновленні залишків і зручній підтримці реляційних зв'язків;

- реалізувати модулі довідників, закупівель, продажів, складського обліку, панелі показників, звітів і керування користувачами, щоб система підтримувала повний цикл роботи від створення товару до формування документів руху й аналізу результатів;

- перевірити працездатність системи на типових сценаріях обліку продажів, закупівель і складських залишків, зокрема створенні закупівлі з кількома позиціями, збільшенні залишку, продажу товару з перевіркою доступної кількості, редагуванні документа, формуванні звітів і експорті даних у CSV.

РОЗДІЛ 2

СПЕЦИФІКАЦІЯ ВИМОГ ДО РОЗРОБЛЮВАНОЇ СИСТЕМИ

2.1 Аналіз, визначення вимог до розроблюваного програмного забезпечення та проєктування програмного забезпечення

Під час аналізу предметної області встановлено, що система управління продажами та закупівлями повинна підтримувати повний цикл товарного обліку: створення довідника товарів, ведення постачальників і клієнтів, оформлення документів закупівлі, оформлення документів продажу, автоматичне оновлення складських залишків, контроль мінімального запасу, перегляд історії руху товарів і формування звітної інформації. Центральною вимогою до такої системи є узгодженість між документами руху товару та фактичним залишком на складі. Якщо товар надходить за документом закупівлі, його кількість має збільшуватися. Якщо товар реалізується через документ продажу, система повинна перевіряти доступний залишок і зменшувати його лише після успішного збереження документа.

У розроблюваній системі передбачено три ролі користувачів: Admin, Manager і Viewer. Адміністратор має повний доступ до функцій системи, зокрема до керування користувачами, довідниками, документами, складом і звітами. Менеджер виконує основні операції з товарами, постачальниками, клієнтами, закупівлями, продажами та переглядом складських даних. Користувач з роллю Viewer призначений для перегляду інформації, інвентаризаційних даних і звітів без виконання критичних операцій редагування. Такий розподіл дає змогу відокремити адміністративні дії від щоденної роботи з документами та зменшити ризик випадкового змінення важливих даних. Основні варіанти використання системи наведено на рисунку 2.1.

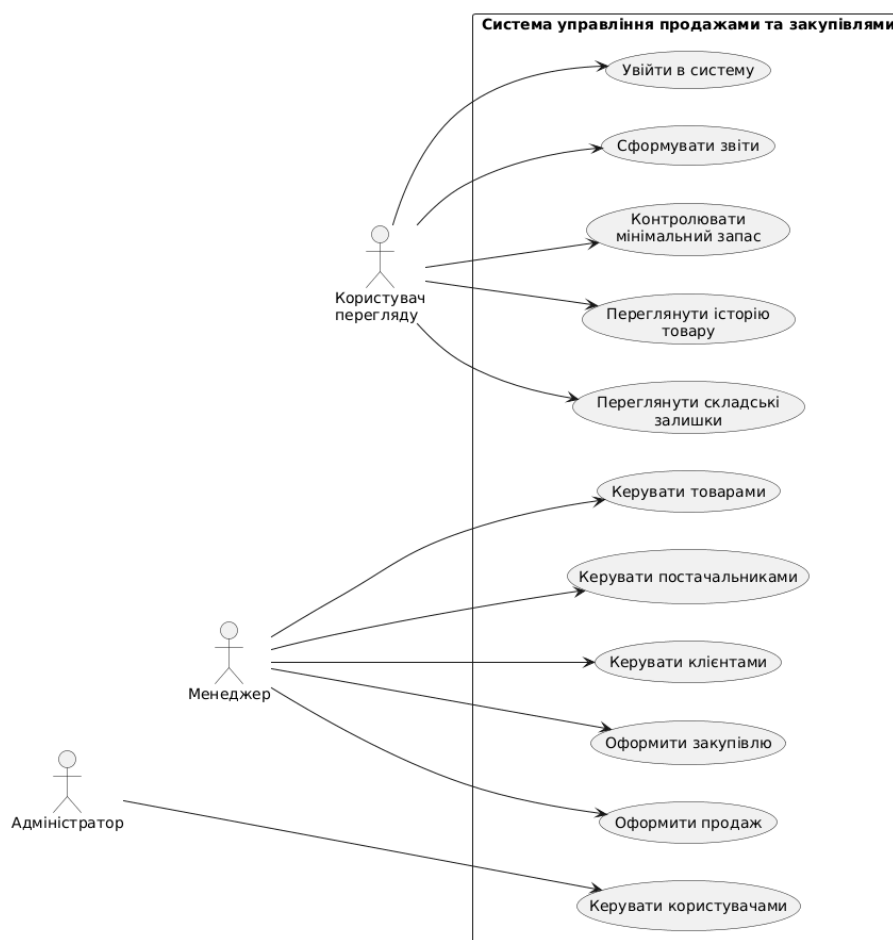


Рисунок 2.1 – UML-діаграма варіантів використання системи управління продажами та закупівлями

До функціональних вимог системи належить автентифікація користувача за логіном і паролем, відкриття головного вікна відповідно до ролі користувача, ведення довідника товарів, постачальників і клієнтів, створення та редагування документів закупівлі, створення та редагування документів продажу, автоматичний перерахунок сум у позиціях документів, оновлення кількості товару на складі після збереження документа, перевірка достатності залишку перед продажем, перегляд складських залишків, виявлення товарів із кількістю нижче мінімального рівня, перегляд історії руху товару та формування звітів. Окремою вимогою є підтримка зв'язку між документом і його позиціями: закупівля повинна складатися з однієї або кількох позицій закупівлі, а продаж – з однієї або кількох позицій продажу.

Нефункціональні вимоги пов'язані зі зручністю використання, надійністю обробки даних і підтримуваністю програмного коду. Інтерфейс має бути зрозумілим для користувачів, які щоденно працюють із товарами та документами, тому основні модулі розміщено у вигляді окремих форм: товари, постачальники, клієнти, закупівлі, продажі, склад, звіти, користувачі та панель показників. Дані повинні зберігатися в реляційній базі даних, а зв'язки між сутностями мають забезпечувати цілісність документів. Операції зі складським залишком повинні виконуватися разом зі збереженням документа, щоб уникнути ситуації, коли документ створено, але кількість товару не оновлено. Архітектурну організацію основних частин системи наведено на рисунку 2.2.

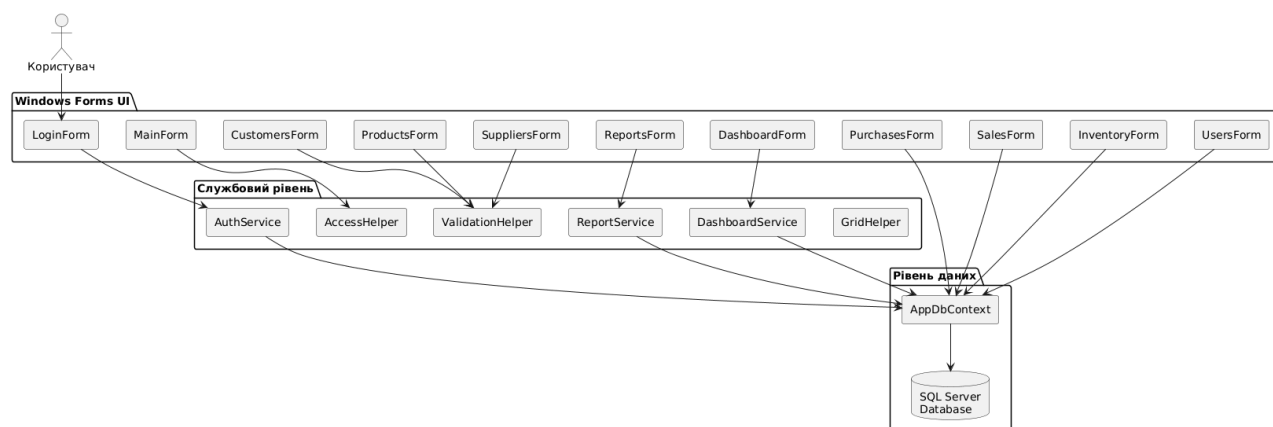


Рисунок 2.2 – UML-діаграма компонентів системи управління продажами та закупівлями

Проектування програмного забезпечення виконано з урахуванням настільної архітектури застосунку Windows Forms. На рівні інтерфейсу користувач працює з окремими формами для кожного функціонального модуля. Форми створення й редагування закупівель та продажів відповідають за введення загальних реквізитів документа, додавання позицій, розрахунок підсумкової суми та передавання змін до бази даних. Допоміжні класи використовуються для перевірки прав доступу, валідації введених даних, налаштування таблиць і стилізації елементів інтерфейсу. Сервіс автентифікації

виконує пошук користувача за логіном і паролем, сервіс звітів готує узагальнені дані, а сервіс панелі показників формує інформацію про залишки, суми закупівель, суми продажів і товари з низьким запасом.

Основу моделі даних становлять сутності Product, Supplier, Customer, Purchase, PurchaseItem, Sale, SaleItem і User. Сутність Product описує товар і містить назву, категорію, одиницю вимірювання, закупівельну ціну, ціну продажу, поточний залишок і мінімальний рівень запасу. Supplier зберігає інформацію про постачальника, Customer – дані клієнта. Purchase описує документ закупівлі та пов'язаний із постачальником, а PurchaseItem зберігає окремі позиції цього документа. Sale описує документ продажу та може бути пов'язаний із клієнтом, а SaleItem містить позиції продажу. User зберігає дані облікового запису та роль користувача. UML-діаграму класів проектної моделі наведено на рисунку 2.3.

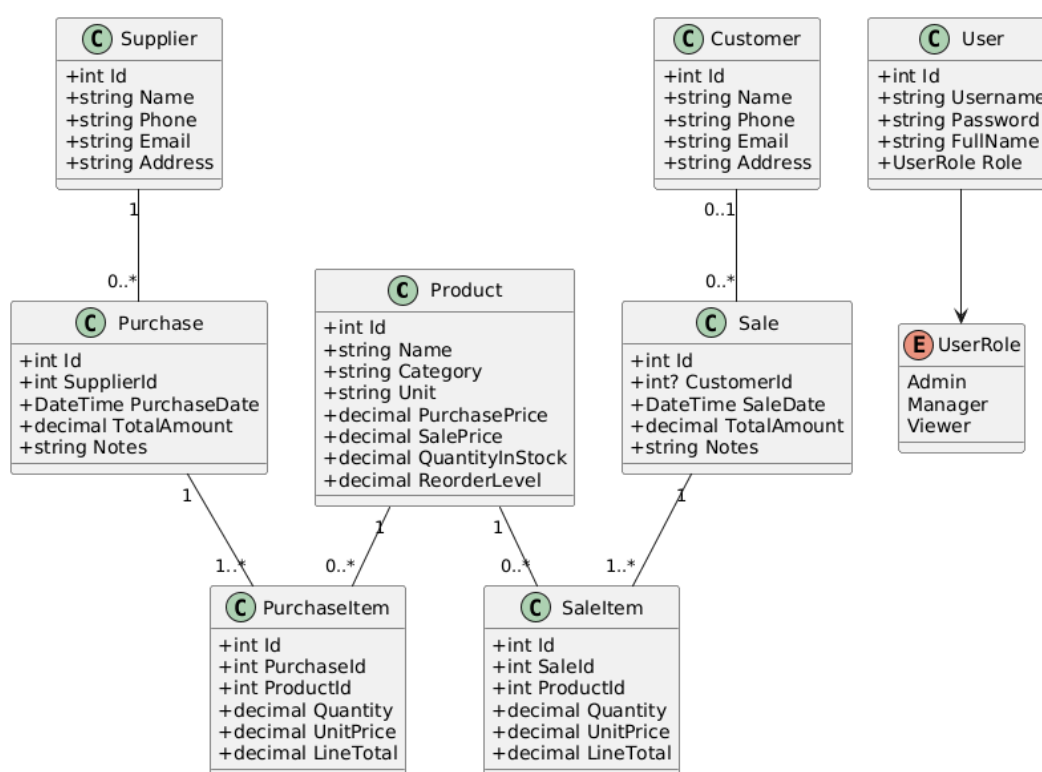


Рисунок 2.3 – UML-діаграма класів системи управління продажами та закупівлями

Сценарій оформлення закупівлі починається з вибору постачальника та дати документа. Після цього користувач додає до документа одну або кілька позицій, для кожної з яких обирає товар, вказує кількість і ціну. Система розраховує суму позиції як добуток кількості на ціну, а загальна сума документа формується як сума всіх позицій. Після збереження закупівлі створюється запис документа, зберігаються його позиції, а кількість відповідних товарів на складі збільшується. Якщо користувач редагує вже наявну закупівлю, система спочатку скасовує вплив попередніх позицій на залишки, після чого застосовує оновлені позиції. Послідовність оформлення закупівлі наведено на рисунку 2.4.

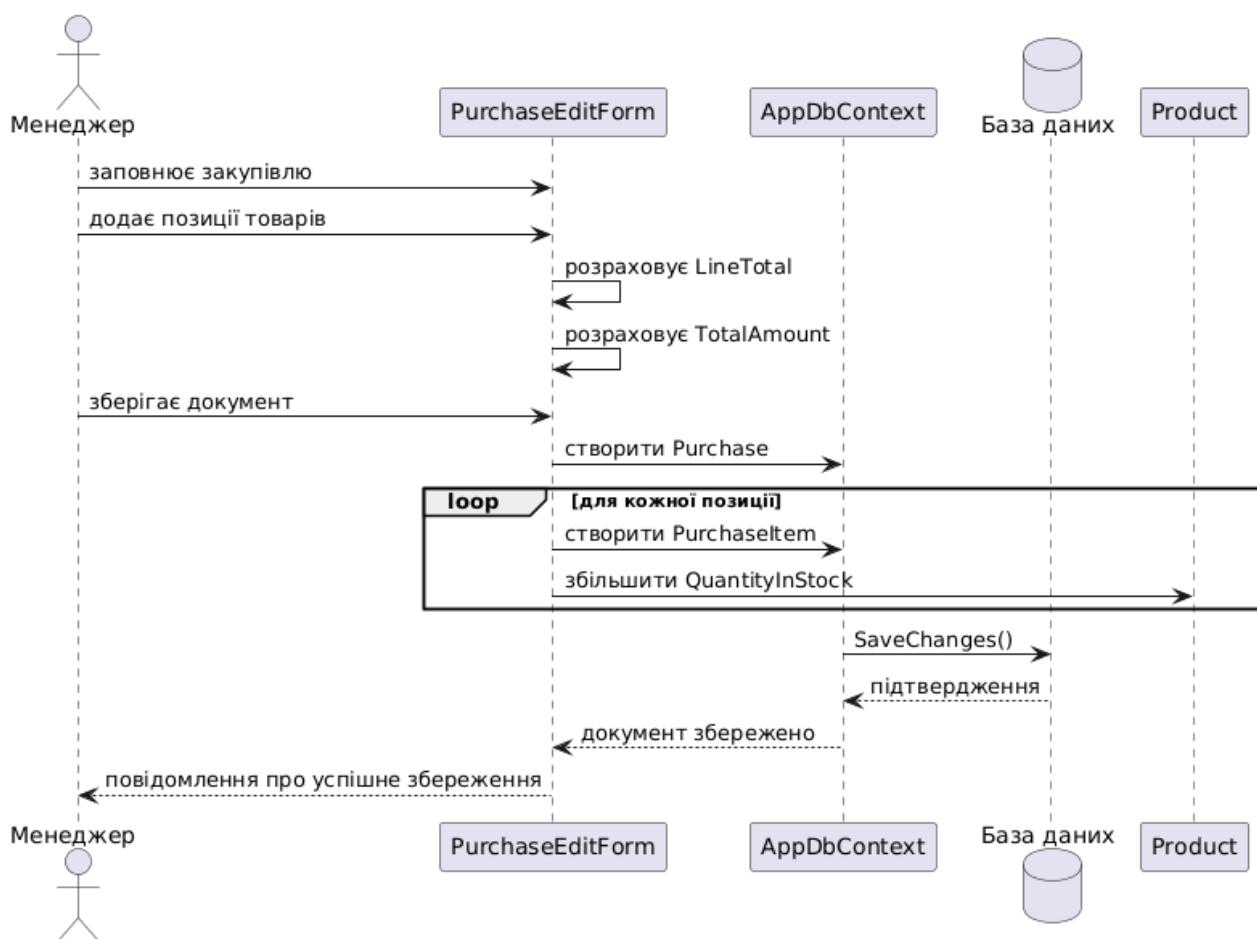


Рисунок 2.4 – UML-діаграма послідовності оформлення закупівлі

Сценарій оформлення продажу має схожу структуру, але містить додаткову перевірку складського залишку. Користувач обирає клієнта, дату документа та додає позиції продажу. Для кожної позиції система перевіряє, чи

достатньо товару на складі. Якщо кількість товару менша за потрібну, документ не повинен бути збережений без виправлення позиції. Після успішного збереження продажу створюється документ, зберігаються його позиції, а залишок кожного проданого товару зменшується. Під час редагування продажу система повертає на склад кількість із попередніх позицій, повторно перевіряє оновлені значення та списує нову кількість. Послідовність оформлення продажу наведено на рисунку 2.5.

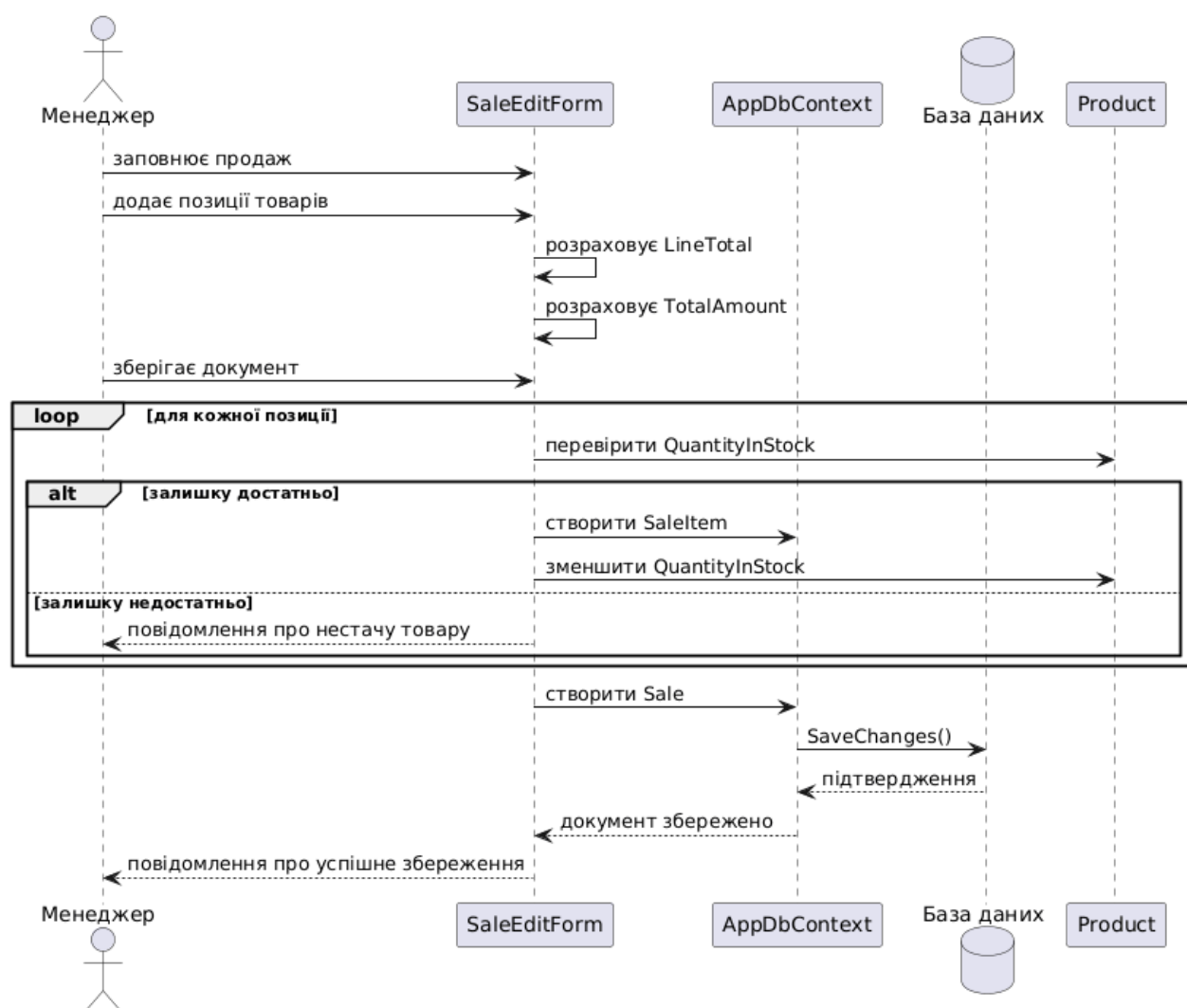


Рисунок 2.5 – UML-діаграма послідовності оформлення продажу

Отже, у результаті аналізу вимог визначено основні ролі користувачів, функціональні можливості системи, обмеження доступу, структуру даних і логіку руху товарів. Проєктна модель охоплює довідники, документи закупівель

і продажів, складський облік, звітність і адміністрування користувачів, що відповідає призначенню системи управління продажами та закупівлями.

2.2 Вибір засобів, методів і алгоритмів вирішення поставленого завдання

Вибір засобів реалізації системи управління продажами та закупівлями здійснювався з урахуванням того, що програма повинна працювати як локальний настільний інструмент для адміністратора, менеджера та користувача з правами перегляду. Основними вимогами до технологічного стеку є зручне ведення довідників, оформлення документів з кількома позиціями, коректне оновлення складських залишків, контроль мінімального запасу, формування управлінських звітів і експорт результатів у CSV.

Для побудови інтерфейсу обрано Windows Forms. Ця технологія платформи .NET призначена для створення настільних застосунків Windows з вікнами, елементами керування та подієвою моделлю взаємодії [1]. Для системи продажів і закупівель вона є доцільною, оскільки основні операції виконуються через форми товарів, постачальників, клієнтів, закупівель, продажів, складу, звітів і користувачів.

Під час вибору клієнтської технології розглядалися Windows Forms, WPF і вебзастосунок. Windows Forms обрано як основний варіант, оскільки ця технологія добре підходить для швидкого створення форм введення, табличного подання даних, діалогових вікон, кнопок дій і локального робочого місця менеджера. Її можливостей достатньо для реалізації довідників, документів руху товарів, перегляду залишків і звітів. WPF має ширші можливості для побудови складних інтерфейсів, стилів, шаблонів і прив'язування даних, однак для поточного завдання він є менш доцільним через більшу складність розробки. Вебзастосунок забезпечив би доступ з різних пристроїв, але потребував би вебсервера, налаштування мережевого доступу, додаткового захисту та іншої архітектури розгортання. Оскільки розроблювана система орієнтована на

локальне використання в межах робочого місця, перевагу надано Windows Forms.

Прикладну логіку реалізовано мовою C# у середовищі .NET 8.0 для Windows. Проєкт поділено на моделі, форми, допоміжні класи, сервіси та модуль доступу до даних. Такий поділ дозволяє відокремити опис сутностей від інтерфейсної логіки, а службові класи використовувати для авторизації, перевірки доступу, панелі показників і повторюваних операцій з таблицями.

Для роботи з даними використано Entity Framework Core та провайдер Microsoft SQL Server. Документація EF Core описує SQL Server provider як засіб підключення EF Core до SQL Server через пакет Microsoft.EntityFrameworkCore.SqlServer [2]. У системі ApplicationDbContext містить таблиці Users, Products, Suppliers, Customers, Purchases, PurchaseItems, Sales і SaleItems, а також описує типи грошових полів, зв'язки між документами та їхніми рядками і каскадне видалення позицій документа.

Для локального зберігання обрано SQL Server LocalDB. Microsoft Learn описує LocalDB як легку версію SQL Server Express, призначену для локальної розробки та простого запуску без повноцінного серверного адміністрування [3]. Це відповідає характеру розробки, оскільки база SalesPurchasesDb розміщується локально, але зберігає можливості реляційної моделі SQL Server.

Під час вибору засобу зберігання даних розглядалися SQL Server LocalDB, SQLite і PostgreSQL. SQL Server LocalDB обрано тому, що він підтримує реляційні зв'язки, транзакції, типи decimal, інтеграцію з Entity Framework Core і локальне розгортання на Windows. SQLite є простішим файловим рішенням і добре підходить для невеликих автономних застосунків, однак у контексті системи продажів і закупівель менш зручний для подальшого переходу до централізованого або багатокористувацького сценарію. PostgreSQL є потужною серверною СКБД, але для локального настільного застосунку потребує зайвого адміністрування, налаштування сервера та супроводу. Тому для поточної реалізації найбільш збалансованим варіантом є SQL Server LocalDB.

Параметри підключення винесено в `appsettings.json` і зчитуються через пакети `Microsoft.Extensions.Configuration`. Документація .NET описує систему конфігурації як механізм отримання налаштувань з різних джерел, зокрема JSON файлів [9]. Це дозволяє змінювати рядок підключення без редагування коду форм або контексту бази даних.

Для операцій, які змінюють залишки товарів, використано транзакції EF Core. У документації EF Core зазначено, що транзакції застосовуються для узгодженого збереження декількох змін у базі даних [10]. У формах закупівлі та продажу збереження документа виконується в межах `BeginTransaction`: якщо хоча б одна позиція не може бути застосована, транзакція відкочується, і складські залишки не переходять у частково змінений стан.

Для кількостей, цін і сум використано тип `decimal`. У документації C# зазначено, що `decimal` підходить для фінансових і грошових розрахунків, де потрібна точність десяткових значень [11]. У моделі даних поля `PurchasePrice`, `SalePrice`, `QuantityInStock`, `TotalAmount`, `Quantity`, `UnitPrice` і `LineTotal` налаштовано як `decimal(18,2)`, що зменшує ризик помилок округлення у звітах і документах.

Для експорту звітів у CSV використано класи `System.Text` і запис текстового файлу з кодуванням `UTF8Encoding`. Документація .NET описує `UTF8Encoding` як клас для кодування символів у формат UTF8 [12]. У системі значення таблиці екрануються лапками, розділяються крапкою з комою і зберігаються з BOM, щоб файл коректно відкривався у табличних редакторах.

Отже, для реалізації системи управління продажами та закупівлями обрано C# і .NET 8.0 для прикладної логіки, Windows Forms для настільного інтерфейсу, Entity Framework Core для роботи з реляційною моделлю, SQL Server LocalDB для локального зберігання даних, `Microsoft.Extensions.Configuration` для зчитування параметрів підключення, тип `decimal` для точних фінансових розрахунків і CSV як простий формат експорту звітів. Такий набір засобів відповідає локальному характеру системи, забезпечує роботу з довідниками,

документами, складськими залишками, користувачами і звітами без ускладнення архітектури.

Алгоритм авторизації починається з введення логіна і пароля у формі входу. AuthService виконує пошук користувача в таблиці Users і після успішного входу поточний користувач зберігається в Program.CurrentUser. Далі AccessHelper визначає права доступу: адміністратор може керувати користувачами і демонстраційними даними, менеджер працює з довідниками та документами, а користувач з правами перегляду має доступ до складу і звітів без редагування критичних даних.

Алгоритм створення закупівлі передбачає вибір постачальника, додавання однієї або кількох товарних позицій, перевірку кількості та ціни, розрахунок суми рядка як добутку кількості на ціну і підрахунок загальної суми документа. Після збереження створюється запис Purchase, додаються PurchaseItems, а QuantityInStock кожного товару збільшується на кількість у відповідному рядку.

Алгоритм редагування закупівлі враховує попередній вплив документа на склад. Спочатку система знаходить старі позиції, віднімає їхню кількість від залишків товарів і перевіряє, щоб залишок не став від'ємним. Після цього старі рядки видаляються, оновлюється заголовок документа, додаються нові позиції і повторно збільшуються залишки. Усі ці дії виконуються в межах транзакції.

Алгоритм створення продажу додатково перевіряє доступний залишок. Для нового документа доступною кількістю є поточний QuantityInStock, а під час редагування до нього тимчасово додається кількість із початкового документа. Якщо запитана кількість перевищує доступний залишок, позиція не додається. Після збереження створюється Sale, додаються SaleItems, а залишки товарів зменшуються.

Алгоритм редагування продажу також спочатку скасовує попередній вплив документа: старі позиції повертаються на склад, після чого перевіряється можливість застосування нового складу документа. Якщо для кожного окремого товару залишку недостатньо, операція скасовується і транзакція відкочується. Це

захищає систему від ситуацій, коли продаж збережено, але складські дані оновлено лише частково.

Алгоритм складського контролю ґрунтується на порівнянні QuantityInStock і ReorderLevel. Якщо залишок дорівнює нулю або менший за нього, товар отримує статус критичного залишку; якщо залишок не перевищує мінімального рівня, встановлюється статус низького залишку; в інших випадках товар вважається у нормі. Ці статуси використовуються у формі складу, панелі показників і звіті товарів з низьким залишком.

Алгоритм звітності підтримує п'ять типів звітів: продажі за період, закупівлі за період, прибуток за період, залишки товарів і товари з низьким залишком. Для періодичних звітів застосовується фільтр дат, для прибутку розраховується дохід, собівартість і різниця між ними, а для складських звітів використовується поточний стан таблиці Products. Поточну таблицю можна експортувати у CSV файл.

Отже, обрані засоби, методи й алгоритми забезпечують реалізацію основної логіки системи управління продажами та закупівлями: рольовий доступ, ведення довідників, створення документів, транзакційне оновлення залишків, контроль мінімального запасу, перегляд аналітичних показників і формування звітів.

РОЗДІЛ 3

РОЗРОБКА СИСТЕМИ УПРАВЛІНЯ ПРОДАЖАМИ

3.1 Практична реалізація об'єкта проєктування

Практична реалізація системи управління продажами та закупівлями виконана як настільний застосунок Windows Forms на платформі .NET. У проєкті виділено форми користувацького інтерфейсу, моделі предметної області, контекст бази даних, допоміжні класи доступу та сервіси для панелі показників і авторизації. Така структура дає змогу відокремити візуальну частину від логіки роботи з документами, залишками й звітами.

Запуск системи передбачає зчитування конфігурації, підключення до SQL Server LocalDB, створення бази даних за потреби, заповнення початкових даних і відкриття форми входу. Фрагмент коду запуску наведено у лістингу 3.1.

Лістинг 3.1 – Ініціалізація застосунку та підключення до бази даних

```
[STAThread]
static void Main()
{
    ApplicationConfiguration.Initialize();

    var configuration = new ConfigurationBuilder()
        .SetBasePath(AppDomain.CurrentDomain.BaseDirectory)
        .AddJsonFile("appsettings.json", optional: false,
reloadOnChange: true)
        .Build();

    var optionsBuilder = new
DbContextOptionsBuilder<AppDbContext>();

optionsBuilder.UseSqlServer(configuration.GetConnectionString("Def
aultConnection"));

    DbContext = new AppDbContext(optionsBuilder.Options);
    DbContext.Database.EnsureCreated();
    DbInitializer.Seed(DbContext);

    Application.Run(new LoginForm());
}
```

Кінець лістингу 3.1

Форма авторизації є першою точкою взаємодії користувача із системою. Її вигляд наведено на рисунку 3.1.

Рисунок 3.1 – Форма входу до системи продажів і закупівель

Після успішного входу відкривається головна панель, на якій зібрано підсумкові показники: кількість товарів, постачальників і клієнтів, загальний залишок, собівартість запасів, сума закупівель, сума продажів і орієнтовний прибуток. Головну панель системи показано на рисунку 3.2.

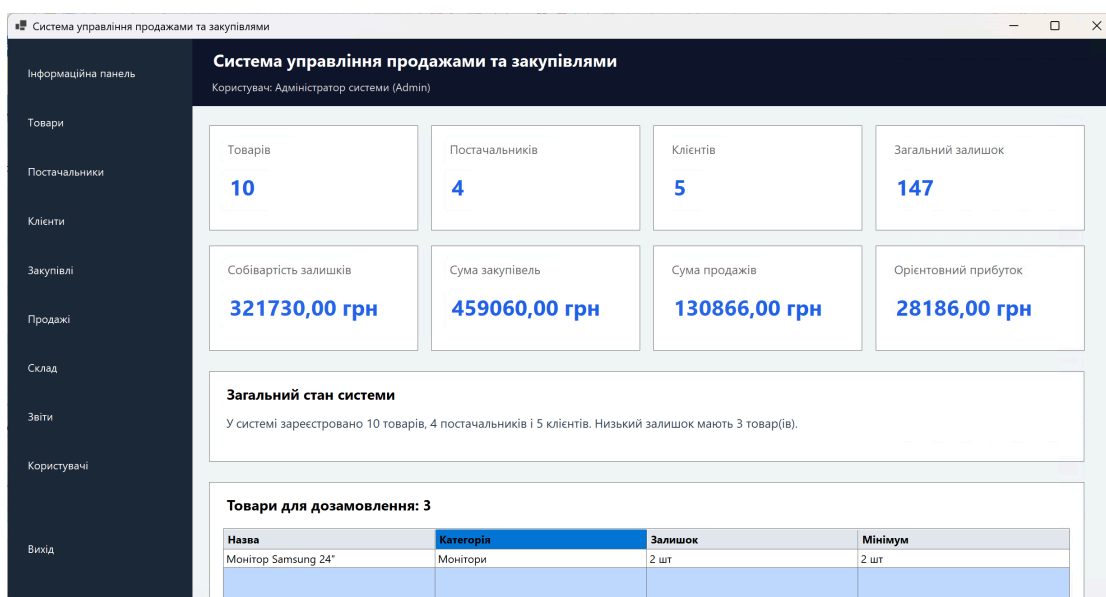


Рисунок 3.2 – Головна панель показників системи

Показники головної панелі формуються у `DashboardService`. Наприклад, орієнтовний прибуток визначається як різниця між сумою продажів і собівартістю проданих товарів, яка розраховується за закупівельною ціною товару. Відповідний фрагмент коду наведено у лістингу 3.2.

Лістинг 3.2 – Розрахунок орієнтовного прибутку в `DashboardService`

```
public decimal GetEstimatedProfit()
{
    decimal salesRevenue = _context.SaleItems.Any()
        ? _context.SaleItems.Sum(x => x.LineTotal)
        : 0;
    decimal salesCost = _context.SaleItems
        .Include(x => x.Product)
        .AsEnumerable()
        .Sum(x => x.Quantity * (x.Product?.PurchasePrice ?? 0));
    return salesRevenue - salesCost;
}
```

Кінець лістингу 3.2

Модуль товарів забезпечує перегляд номенклатури, додавання, редагування, видалення та перегляд історії руху окремого товару. Форма товарів наведена на рисунку 3.3, а діалог редагування товару – на рисунку 3.4.

Товари						
Кількість записів: 10						
Додати Редагувати Видалити Оновити Історія руху						
Назва	Категорія	Одиниця	Закупівельна ціна	Ціна продажу	Залишок	Мінімальний залишок
SSD Kingston 1TB	Комплектуючі	шт	1700,00	2299,00	17,00	5,00
USB Flash 64GB	Носії даних	шт	180,00	299,00	10,00	12,00
Вебкамера Full HD	Акcesуари	шт	950,00	1499,00	4,00	4,00
Клавіатура Defender Ele...	Периферія	шт	450,00	699,00	29,00	8,00
Миша Logitech M185	Периферія	шт	320,00	499,00	53,00	10,00
Монітор LG 27"	Монітори	шт	6500,00	7999,00	5,00	2,00
Монітор Samsung 24"	Монітори	шт	4200,00	5499,00	2,00	2,00
Навушники Havit	Акcesуари	шт	520,00	899,00	16,00	6,00
Ноутбук ASUS VivoBook...	Ноутбуки	шт	20500,00	24999,00	4,00	2,00
Ноутбук Lenovo IdeaPa...	Ноутбуки	шт	18000,00	21999,00	7,00	3,00

Рисунок 3.3 – Форма керування товарами

Рисунок 3.4 – Діалогова форма редагування товару

Оформлення закупівлі реалізовано через документ із заголовком і набором позицій. Після збереження створюється запис `Purchase`, додаються `PurchaseItems`, а залишок кожного товару збільшується. Фрагмент коду збереження нової закупівлі наведено у лістингу 3.3.

Лістинг 3.3 – Створення закупівлі та збільшення складського залишку

```
private void SaveNewPurchase()
{
    var purchase = new Purchase
    {
        SupplierId = ((Supplier)cmbSupplier.SelectedItem).Id,
        PurchaseDate = dtpPurchaseDate.Value,
        Notes = txtNotes.Text.Trim(),
        TotalAmount = _items.Sum(x => x.LineTotal)
    };

    Program.DbContext.Purchases.Add(purchase);
    Program.DbContext.SaveChanges();

    foreach (var row in _items)
    {
        Program.DbContext.PurchaseItems.Add(new PurchaseItem
        {
            PurchaseId = purchase.Id,
            ProductId = row.ProductId,
            Quantity = row.Quantity,
            UnitPrice = row.UnitPrice,
```

```

        LineTotal = row.LineTotal
    });

    var product = Program.DbContext.Products.First(x => x.Id == row.ProductId);
    product.QuantityInStock += row.Quantity;
}

Program.DbContext.SaveChanges();
}

```

Кінець лістингу 3.3

Список закупівель із підсумковими сумами наведено на рисунку 3.5.

Закупівлі
Кількість записів: 4

Пошук (постачальник / примітки)

Ві До

Загальна сума: **459060,00 грн** Середня сума: **114765,00 грн**

Дата	Постачальник	Сума	Примітки
24.03.2026	ФОП Андрій Мельник	16750,00	Дозакупівля периферії
20.03.2026	ТОВ Digital Trade	189260,00	Друга хвиля закупівель
16.03.2026	ПП ЕлектроМаркет	85000,00	Закупівля моніторів і SSD
12.03.2026	ТОВ ТехноПостач	168050,00	Стартова закупівля ноутбуків і периферії

Рисунок 3.5 – Форма перегляду закупівель

Модуль продажів перед додаванням позиції перевіряє доступний залишок. Якщо користувач намагається продати більше, ніж є на складі, система не додає позицію до документа і показує попередження. Фрагмент перевірки наведено у лістингу 3.4.

Лістинг 3.4 – Перевірка доступного залишку під час продажу

```

decimal effectiveStock =
    GetEffectiveAvailableStock(selectedProduct);
decimal requested = nudQuantity.Value;
decimal alreadyAdded =
    GetAlreadyReservedQuantity(selectedProduct.Id);
decimal newTotal = alreadyAdded + requested;

if (newTotal > effectiveStock)
{
    MessageBox.Show(
        $"Недостатньо товару на складі. Доступно: {effectiveStock:0.##}",

```

```

        "Недостатній залишок",
        MessageBoxButtons.OK,
        MessageBoxIcon.Warning);
    return;
}

items.Add(new SaleItemRow
{
    ProductId = selectedProduct.Id,
    ProductName = selectedProduct.Name,
    Quantity = requested,
    UnitPrice = nudUnitPrice.Value
});

```

Кінець лістингу 3.4

Форма продажів та перегляд складу документа наведені на рисунках 3.6 і 3.7.

Продажі
Кількість записів: 4

Пошук (клієнт / примітки) ☐ Фільтр за датою

Ві 11.04.2026 До 11.05.2026

Додати Редагувати Оновити Склад Видалити Скинути фільтри

Загальна сума: **130866,00 грн** Середня сума: **32716,50 грн**

Дата	Клієнт	Сума	Примітки
29.03.2026	ПП Офіс Центр	68994,00	Продаж робочих місць
27.03.2026	Марія Ковальчук	10190,00	Продаж комплектуючих
25.03.2026	ТОВ Бізнес Клієнт	29184,00	Продаж офісної техніки
22.03.2026	Іван Петренко	22498,00	Роздрібний продаж

Рисунок 3.6 – Форма перегляду продажів

Склад продажу №4
Позицій: 3

Тип документа: **Продаж** Контрагент: **І Офіс Центр** Оновити Закрити

Дата: **29.03.2026** Загальна сума: **994,00 грн**

Товар	Кількість	Ціна	Сума
Ноутбук ASUS VivoBook 15	2,00	24999,00	49998,00
Монітор LG 27"	2,00	7999,00	15998,00
Вебкамера Full HD	2,00	1499,00	2998,00

Рисунок 3.7 – Перегляд позицій документа продажу

Для аналізу руху конкретного товару реалізовано форму історії. Вона показує закупівлі, продажі та загальний рух за вибраною позицією, що допомагає швидко пояснити поточний залишок.

Приклад цієї форми наведено на рисунку 3.8. Розширений фрагмент коду редагування закупівлі з транзакційним перерахунком залишків наведено в додатку А.

Історія руху товару

Назва товару: **SSD Kingston 1TB** Закупівельна ціна: **1700,00 грн** Мінімальний залишок: **5 шт**

Категорія: **Комплектуючі** Ціна продажу: **2299,00 грн**

Одиниця: **шт** Поточний залишок: **17 шт**

Закупівлі товару Усього закуплено: 20 шт

Дата	Постачальник	Кількість	Ціна	Сума	Документ
16.03.2026	ПП Електро...	20,00	1700,00	34000,00	Закупівля №2

Продажі товару Усього продано: 3 шт

Дата	Клієнт	Кількість	Ціна	Сума	Документ
27.03.2026	Марія Ковал...	3,00	2299,00	6897,00	Продаж №3

Загальний журнал руху

Дата	Операція	Кількість	Ціна	Сума	Контрагент	Документ
27.03.2026	Продаж	-3	2299,00	6897,00	Марія Ковальчук	Продаж №3
16.03.2026	Закупівля	+20	1700,00	34000,00	ПП ЕлектроМаркет	Закупівля №2

Рисунок 3.8 – Історія руху товару

3.2 Розробка бази даних

База даних системи побудована як реляційна модель, у якій довідникові сутності відокремлені від документів руху товару.

Таблиця Products зберігає номенклатуру, ціни, одиницю виміру, кількість на складі та мінімальний залишок.

Таблиці Suppliers і Customers містять контактні дані контрагентів. Таблиці Purchases і Sales відповідають за заголовки документів, а PurchaseItems і SaleItems – за їхні рядки.

Схему бази даних інформаційної системи наведено на рисунку 3.9.

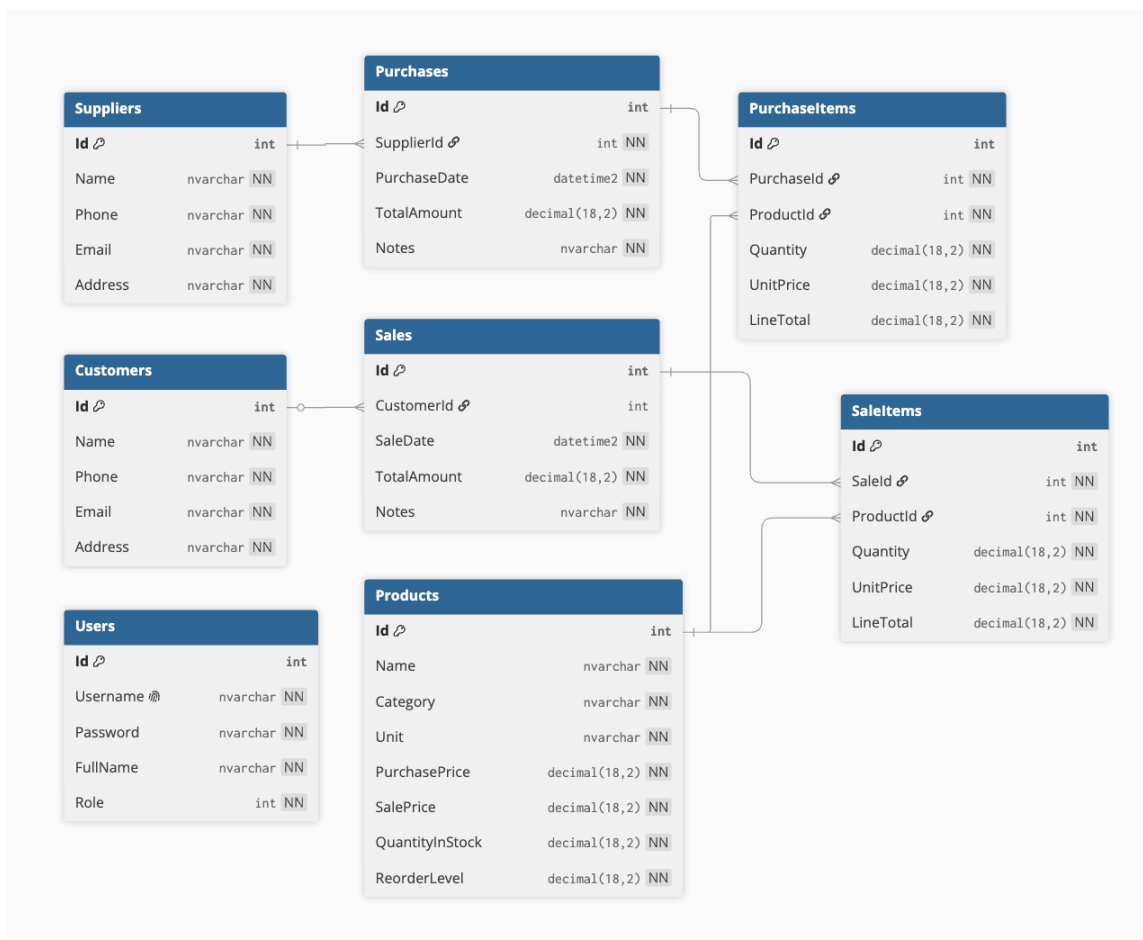


Рисунок 3.9 – Схема бази даних системи управління продажами та закупівлями

У моделі використано зв'язок один до багатьох між закупівлею та її позиціями, а також між продажем та його позиціями. Кожна позиція документа посилається на товар і зберігає кількість, ціну за одиницю та суму рядка. Такий підхід дозволяє зберігати історичну ціну документа навіть тоді, коли поточна ціна товару згодом зміниться.

Налаштування сутностей виконано в AppDbContext. Для фінансових і кількісних полів задано decimal(18,2), а позиції документів видаються

каскадно разом із відповідним заголовком документа. Фрагмент конфігурації наведено у лістингу 3.5.

Лістинг 3.5 – Конфігурація типів даних і зв'язків у AppDbContext

```

modelBuilder.Entity<Product>()
    .Property(p => p.PurchasePrice)
    .HasColumnType("decimal(18,2)");

modelBuilder.Entity<Product>()
    .Property(p => p.QuantityInStock)
    .HasColumnType("decimal(18,2)");

modelBuilder.Entity<PurchaseItem>()
    .HasOne(x => x.Purchase)
    .WithMany(x => x.Items)
    .HasForeignKey(x => x.PurchaseId)
    .OnDelete(DeleteBehavior.Cascade);

modelBuilder.Entity<SaleItem>()
    .HasOne(x => x.Sale)
    .WithMany(x => x.Items)
    .HasForeignKey(x => x.SaleId)
    .OnDelete(DeleteBehavior.Cascade);

```

Кінець лістингу 3.5

Під час створення закупівлі або продажу зміни впливають не лише на таблиці документів, а й на QuantityInStock у Products. Тому для збереження документів використовується транзакційний підхід: якщо запис заголовка, позицій або оновлення залишків не виконано, усі зміни відкочуються. Це особливо важливо під час редагування документів, коли система спочатку скасовує попередній вплив документа на склад, а потім застосовує новий набір рядків.

Постачальники та клієнти винесені в окремі таблиці, що дає змогу багаторазово використовувати їх у документах і звітах. Приклади довідників постачальників і клієнтів наведено на рисунках 3.10 і 3.11.

Постачальники			
Кількість записів: 4			
Додати Редагувати Видалити Оновити			
Назва	Телефон	Email	Адреса
ПП ЕлектроМаркет	+380502224455	sales@electro.ua	м. Львів, вул. Наукова, 22
TOB Digital Trade	+380503336677	office@digitaltrade.ua	м. Київ, вул. Січових Стрільців, 8
TOB ТехноПостач	+380501112233	tech@supply.ua	м. Луцьк, вул. Рівненська, 15
ФОП Андрій Мельник	+380504447788	melnyk.parts@gmail.com	м. Тернопіль, вул. Руська, 10

Рисунок 3.10 – Довідник постачальників

Товари						
Кількість записів: 10						
Додати Редагувати Видалити Оновити Історія DVXV						
Назва	Категорія	Одиниця	Закупівельна ціна	Ціна продажу	Залишок	Мінімальний залишок
SSD Kingston 1TB	Комплектуючі	шт	1700,00	2299,00	17,00	5,00
USB Flash 64GB	Носії даних	шт	180,00	299,00	10,00	12,00
Вебкамера Full HD	Аксесуари	шт	950,00	1499,00	4,00	4,00
Клавіатура Defender Ele...	Периферія	шт	450,00	699,00	29,00	8,00
Миша Logitech M185	Периферія	шт	320,00	499,00	53,00	10,00
Монітор LG 27"	Монітори	шт	6500,00	7999,00	5,00	2,00
Монітор Samsung 24"	Монітори	шт	4200,00	5499,00	2,00	2,00
Навушники Havit	Аксесуари	шт	520,00	899,00	16,00	6,00
Ноутбук ASUS VivoBook...	Ноутбуки	шт	20500,00	24999,00	4,00	2,00
Ноутбук Lenovo IdeaPa...	Ноутбуки	шт	18000,00	21999,00	7,00	3,00

Рисунок 3.11 – Табличне представлення номенклатури товарів

3.3 Захист інформаційно-комп'ютерної системи

Захист інформаційно-комп'ютерної системи реалізовано через авторизацію користувача, рольове розмежування доступу та перевірки перед виконанням операцій, що змінюють складські залишки. Поточний користувач зберігається у `Program.CurrentUser`, а допоміжний клас `AccessHelper` визначає, які розділи та дії доступні для його ролі.

У системі використовуються ролі адміністратора, менеджера та користувача з правами перегляду. Адміністратор має доступ до керування користувачами та демонстраційними даними, менеджер працює з довідниками й документами, а користувач перегляду може аналізувати склад і звіти без

редагування. Фрагмент коду авторизації та перевірки прав наведено у лістингу 3.6.

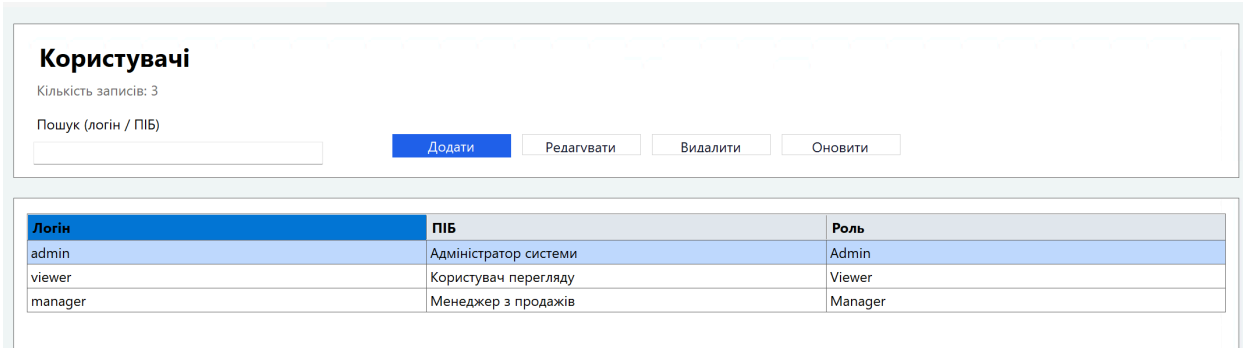
Лістинг 3.6 – Авторизація та рольові правила доступу

```
public User? Login(string username, string password)
{
    return _context.Users.FirstOrDefault(u =>
        u.Username == username && u.Password == password);
}

public static bool CanManageDocuments => IsAdmin || IsManager;
public static bool CanViewReports => IsAdmin || IsManager ||
    IsViewer;
public static bool CanManageUsers => IsAdmin;
```

Кінець лістингу 3.6

Форма керування користувачами наведена на рисунку 3.12. Через неї адміністратор може додавати, редагувати та видаляти облікові записи, а також призначати роль користувача.



Користувачі
Кількість записів: 3

Пошук (логін / ПІБ)

Логін	ПІБ	Роль
admin	Адміністратор системи	Admin
viewer	Користувач перегляду	Viewer
manager	Менеджер з продажів	Manager

Рисунок 3.12 – Форма керування користувачами

Діалог редагування користувача наведено на рисунку 3.13. Під час зміни облікового запису можна оновити логін, ПІБ, роль і пароль. Якщо поле пароля залишити порожнім, система не повинна вимагати зміну пароля при кожному редагуванні профілю.

Рисунок 3.13 – Діалог редагування користувача

Окремий аспект захисту пов'язаний із цілісністю складських даних. Продаж не може бути збережений, якщо кількість товару перевищує доступний залишок, а редагування закупівлі або продажу виконується в межах транзакції. Завдяки цьому система не переходить у стан, коли документ збережено частково, а залишки оновлено не для всіх позицій.

Для зменшення ризику випадкового пошкодження даних критичні дії відокремлені за ролями: керування користувачами доступне лише адміністратору, а робота з документами – адміністратору та менеджеру. Користувач із правами перегляду може відкривати склад і звіти, але не оформлює документи, що впливають на кількість товару.

3.4 Тестування та налагодження інформаційно-комп'ютерної системи

Тестування та налагодження системи виконувалося за основними сценаріями товарного обліку: авторизація, перегляд панелі показників, створення товару, оформлення закупівлі, оформлення продажу, перевірка недостатнього залишку, перегляд історії руху товару, фільтрація складу, формування звітів і експорт результатів у CSV.

Під час тестування складського модуля перевірявся розрахунок статусу товару за мінімальним залишком. Якщо кількість дорівнює нулю або менша за нуль, товар позначається як критичний; якщо кількість не перевищує ReorderLevel – як низький залишок; в інших випадках встановлюється нормальний статус. Фрагмент коду визначення й підсвічування статусів наведено у лістингу 3.7.

Лістинг 3.7 – Визначення статусу складського залишку

```
private string GetStockStatus(decimal quantity, decimal
reorderLevel)
{
    if (quantity <= 0)
        return "Критичний залишок";

    if (quantity <= reorderLevel)
        return "Низький залишок";

    return "Норма";
}

private void HighlightRows()
{
    foreach (DataGridViewRow row in dgvInventory.Rows)
    {
        string status = row.Cells["Статус"].Value?.ToString() ??
"";

        if (status == "Низький залишок")
            row.DefaultCellStyle.BackColor = Color.FromArgb(254,
240, 138);
        else if (status == "Критичний залишок")
            row.DefaultCellStyle.BackColor = Color.FromArgb(254,
202, 202);
    }
}
```

Кінець лістингу 3.7

Результат перевірки складського модуля наведено на рисунку 3.14. У таблиці видно поточні залишки, мінімальні рівні та статуси товарів, а фільтри дозволяють швидко знайти дефіцитні позиції.

Склад / Залишки

Кількість записів: 10

Пошук

Категорія

Статус

Усі категорії

Усі статуси

Оновити

Скинути фільтри

Назва	Категорія	Одиниця	Залишок	Мінімальний залишок	Статус
SSD Kingston 1TB	Комплектуючі	шт	17,00	5,00	Норма
USB Flash 64GB	Носії даних	шт	10,00	12,00	Низький залишок
Вебкамера Full HD	Аксесуари	шт	4,00	4,00	Низький залишок
Клавіатура Defender Element	Периферія	шт	29,00	8,00	Норма
Миша Logitech M185	Периферія	шт	53,00	10,00	Норма
Монітор LG 27"	Монітори	шт	5,00	2,00	Норма
Монітор Samsung 24"	Монітори	шт	2,00	2,00	Низький залишок
Навушники Havit	Аксесуари	шт	16,00	6,00	Норма
Ноутбук ASUS VivoBook 15	Ноутбуки	шт	4,00	2,00	Норма
Ноутбук Lenovo IdeaPad 3	Ноутбуки	шт	7,00	3,00	Норма

Рисунок 3.14 – Перевірка складських залишків і статусів товарів

Звітний модуль перевірявся для продажів за період, закупівель, прибутку, залишків і товарів із низьким запасом. Для періодичних звітів застосовується фільтр дат, а підсумок звіту оновлюється після кожної зміни параметрів. Приклад сформованого звіту наведено на рисунку 3.15.

Звіти

Тип звіту

Від

До

Продажі за період

01.01.2026

11.05.2026

Сформувати

Оновити

Скинути

Експорт CSV

Підсумок звіту

Продажів за період: 4. Загальна сума продажів: 130866,00 грн.

Дата	Клієнт	Сума	Примітки
29.03.2026	ПП Офіс Центр	68994,00	Продаж робочих місць
27.03.2026	Марія Ковальчук	10190,00	Продаж комплектуючих
25.03.2026	ТОВ Бізнес Клієнт	29184,00	Продаж офісної техніки
22.03.2026	Іван Петренко	22498,00	Роздрібний продаж

Рисунок 3.15 – Перевірка формування звіту продажів за період

Для експорту результатів використано CSV файл із розділенням значень крапкою з комою та кодуванням UTF-8 з BOM. Перед записом система бере лише видимі колонки таблиці, екранує лапки у значеннях і формує рядки для поточного звіту. Фрагмент експорту наведено у лістингу 3.8.

Лістинг 3.8 – Експорт поточного звіту у CSV файл

```

var visibleColumns = dgvReport.Columns
    .Cast<DataGridViewColumn>()
    .Where(c => c.Visible)
    .OrderBy(c => c.DisplayIndex)
    .ToList();

sb.AppendLine(string.Join(";", visibleColumns.Select(c =>
EscapeCsvValue(c.HeaderText))));

foreach (DataGridViewRow row in dgvReport.Rows)
{
    if (row.IsNewRow)
        continue;

    var values = new List<string>();
    foreach (var column in visibleColumns)
        values.Add(EscapeCsvValue(row.Cells[column.Name].Value));

    sb.AppendLine(string.Join(";", values));
}

File.WriteAllText(saveDialog.FileName, sb.ToString(), new
UTF8Encoding(true));

```

Кінець лістингу 3.8

Результат відкриття експортованого CSV файлу в табличному редакторі наведено на рисунку 3.16.

Дата	Клієнт	Сума	Примітки		
29.03.2026 0:00	ПП Офіс Центр	68994	Продаж робочих місць		
27.03.2026 0:00	Марія Ковальчук	10190	Продаж комплектуючих		
25.03.2026 0:00	ТОВ Бізнес Клієнт	29184	Продаж офісної техніки		
22.03.2026 0:00	Іван Петренко	22498	Роздрібний продаж		

Рисунок 3.16 – Перевірка експортованого CSV звіту

Також перевірено взаємозв'язок документів і складських залишків. Після створення закупівлі залишок товару збільшується, після продажу – зменшується, а форма історії руху відображає обидва типи операцій. Результати тестування

підтвердили коректність розрахунку сум рядків, загальних сум документів, фільтрації таблиць і рольових обмежень. Розширений фрагмент коду формування звітів і CSV-експорту наведено в додатку Б.

ВИСНОВКИ

У кваліфікаційній роботі розроблено систему управління продажами та закупівлями з використанням C#, .NET, Windows Forms, Entity Framework Core і MS SQL Server LocalDB. Система забезпечує облік товарів, постачальників, клієнтів, закупівель, продажів, складських залишків, користувачів і звітних показників.

Здійснено аналіз процесів товарного обліку, закупівель, продажів, контролю залишків і взаємодії з контрагентами. Визначено, що ключовими проблемами є потреба в актуальному залишку товарів, контролі дефіцитних позицій, коректному розрахунку сум документів і швидкому отриманні підсумкової інформації про діяльність.

Сформовано функціональні вимоги до настільної системи з урахуванням ролей адміністратора, менеджера та користувача з правами перегляду. Рольова модель дає змогу відокремити перегляд звітів і залишків від редагування довідників, оформлення документів та керування користувачами.

Спроектовано структуру бази даних для зберігання товарів, постачальників, клієнтів, закупівель, продажів і позицій документів. Модель даних підтримує зв'язки між закупівлею та її рядками, продажем та його рядками, а також зберігає ціни, кількості, суми, залишки й мінімальний рівень запасу.

Обґрунтовано вибір технологій реалізації. C# і .NET використано як основу застосунку, Windows Forms – для графічного інтерфейсу, Entity Framework Core – для доступу до даних, MS SQL Server LocalDB – для локального зберігання інформації під час роботи системи.

Реалізовано модулі довідників, закупівель, продажів, складського обліку, панелі показників, звітів і керування користувачами. У документах закупівель залишок товару збільшується, у документах продажу – зменшується після перевірки доступної кількості; для редагування документів використано транзакційний підхід із перерахунком впливу на склад.

Перевірено працездатність системи на типових сценаріях обліку продажів, закупівель і складських залишків. Результати підтверджують, що розроблений застосунок може використовуватися для автоматизації основних торговельних операцій, контролю товарних запасів і підготовки управлінської звітності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is Windows Forms. Microsoft Learn. URL: <https://learn.microsoft.com/dotnet/desktop/winforms/overview/> (дата звернення: 05.03.2026).
2. Microsoft SQL Server Database Provider. EF Core. Microsoft Learn. URL: <https://learn.microsoft.com/ef/core/providers/sql-server/> (дата звернення: 05.03.2026).
3. SQL Server Express LocalDB. Microsoft Learn. URL: <https://learn.microsoft.com/sql/database-engine/configure-windows/sql-server-express-localdb> (дата звернення: 14.03.2026).
4. inFlow Inventory. inFlow. URL: <https://www.inflowinventory.com/inflow-inventory> (дата звернення: 14.03.2026).
5. Odoo Inventory. Odoo. URL: <https://www.odoo.com/app/inventory> (дата звернення: 02.04.2026).
6. Open Source Inventory Management Software. ERPNext. URL: <https://frappe.io/erpnext> (дата звернення: 02.04.2026).
7. File: ERPNext accounts home page. Wikimedia Commons. URL: https://commons.wikimedia.org/wiki/File:ERPNext_accounts_home_page.png (дата звернення: 28.04.2026).
8. Zoho Inventory. Zoho. URL: <https://www.zoho.com/inventory/> (дата звернення: 28.04.2026).
9. Configuration in .NET. Microsoft Learn. URL: <https://learn.microsoft.com/dotnet/core/extensions/configuration> (дата звернення: 03.05.2026).
10. Transactions. EF Core. Microsoft Learn. URL: <https://learn.microsoft.com/ef/core/saving/transactions> (дата звернення: 17.05.2026).
11. Floating-point numeric types. C# reference. Microsoft Learn. URL: <https://learn.microsoft.com/dotnet/csharp/language-reference/builtin-types/floating-point-numeric-types> (дата звернення: 25.05.2026).

12. UTF8Encoding Class. Microsoft Learn. URL:
<https://learn.microsoft.com/dotnet/api/system.text.utf8encoding> (дата звернення:
25.05.2026).

ДОДАТКИ

ДОДАТОК А

ФРАГМЕНТ КОДУ РЕДАГУВАННЯ ЗАКУПІВЛІ З ПЕРЕРАХУНКОМ ЗАЛИШКІВ

```

private void UpdateExistingPurchase()
{
    var purchase = Program.DbContext.Purchases
        .Include(p => p.Items)
        .FirstOrDefault(p => p.Id == _purchaseId!.Value);

    if (purchase == null)
        throw new InvalidOperationException("Закупівлю не
знайдено.");

    var existingItems = Program.DbContext.PurchaseItems
        .Include(x => x.Product)
        .Where(x => x.PurchaseId == purchase.Id)
        .ToList();

    foreach (var oldItem in existingItems)
    {
        if (oldItem.Product == null)
            continue;

        oldItem.Product.QuantityInStock -= oldItem.Quantity;

        if (oldItem.Product.QuantityInStock < 0)
            throw new InvalidOperationException(
                $"Неможливо відредагувати закупівлю для товару
{oldItem.Product.Name}.");

        Program.DbContext.Products.Update(oldItem.Product);
    }

    Program.DbContext.PurchaseItems.RemoveRange(existingItems);
    Program.DbContext.SaveChanges();

    purchase.SupplierId = ((Supplier)cmbSupplier.SelectedItem).Id;
    purchase.PurchaseDate = dtpPurchaseDate.Value;
    purchase.Notes = txtNotes.Text.Trim();
    purchase.TotalAmount = _items.Sum(x => x.LineTotal);

    Program.DbContext.Purchases.Update(purchase);
    Program.DbContext.SaveChanges();

    foreach (var row in _items)
    {
        Program.DbContext.PurchaseItems.Add(new PurchaseItem
        {
            PurchaseId = purchase.Id,

```

```
        ProductId = row.ProductId,  
        Quantity = row.Quantity,  
        UnitPrice = row.UnitPrice,  
        LineTotal = row.LineTotal  
    });  
  
    var product = Program.DbContext.Products.First(x => x.Id  
== row.ProductId);  
    product.QuantityInStock += row.Quantity;  
    Program.DbContext.Products.Update(product);  
}  
  
Program.DbContext.SaveChanges();  
}
```

ДОДАТОК Б

ФРАГМЕНТ КОДУ ФОРМУВАННЯ ЗВІТУ ПРИБУТКУ

```

private void GenerateProfitReport()
{
    DateTime from = dtpDateFrom.Value.Date;
    DateTime to = dtpDateTo.Value.Date.AddDays(1).AddTicks(-1);

    var saleItems = Program.DbContext.SaleItems
        .AsNoTracking()
        .Include(x => x.Sale)
        .Include(x => x.Product)
        .Where(x => x.Sale != null && x.Sale.SaleDate >= from &&
x.Sale.SaleDate <= to)
        .Select(x => new
        {
            Товар = x.Product != null ? x.Product.Name : "",
            Кількість = x.Quantity,
            Ціна_продажу = x.UnitPrice,
            Дохід = x.LineTotal,
            Собівартість = x.Product != null ? x.Quantity *
x.Product.PurchasePrice : 0m,
            Прибуток = x.LineTotal - (x.Product != null ?
x.Quantity * x.Product.PurchasePrice : 0m)
        })
        .ToList();

    dgvReport.DataSource = null;
    dgvReport.DataSource = saleItems;

    decimal totalRevenue = saleItems.Sum(x => x.Дохід);
    decimal totalCost = saleItems.Sum(x => x.Собівартість);
    decimal totalProfit = saleItems.Sum(x => x.Прибуток);

    lblSummary.Text =
        $"Дохід: {totalRevenue:0.00} грн | " +
        $"Собівартість: {totalCost:0.00} грн | " +
        $"Прибуток: {totalProfit:0.00} грн";
}

```