

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ
СІКОРСЬКОГО»**

Навчально-науковий інститут атомної та теплової енергетики

Кафедра цифрових технологій в енергетиці

«До захисту допущено»

Завідувач кафедри

_____ Наталія АУШЕВА

“ ” _____ 2026 р.

**Дипломна робота
на здобуття ступеня бакалавр**

За освітньою програмою “Цифрові технології в енергетиці”

Спеціальності 122 “Комп’ютерні науки”

на тему: “Система автоматизованого обліку та управління запасами підприємства”

Виконала: студентка 4 курсу, групи ТР-25

_____ Тхоржевська Анна Сергіївна

(прізвище, ім’я, по батькові)

_____ (підпис)

Керівник доцент, к.е.н, доцент Ірина СЕГЕДА

(посада, науковий ступінь, вчене звання, ім’я, ПРІЗВИЩЕ)

_____ (підпис)

Рецензент в. о директора. проф, д.т.н., Ольга ЧЕРНОУСЕНКО

(посада, науковий ступінь, вчене звання, ім’я, ПРІЗВИЩЕ)

_____ (підпис)

Н. контроль асистент каф. Костянтин ЗДОР

(посада, науковий ступінь, вчене звання, ім’я, ПРІЗВИЩЕ)

_____ (підпис)

Засвідчую, що у цій дипломній роботі немає
запозичень з праць інших авторів без
відповідних посилань.

Студент _____

Київ - 2026

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”**

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ АТОМНОЇ ТА ТЕПЛОВОЇ ЕНЕРГЕТИКИ

Кафедра ЦИФРОВИХ ТЕХНОЛОГІЙ В ЕНЕРГЕТИЦІ

Рівень вищої освіти – перший (бакалаврський)

спеціальність 122 “Комп’ютерні науки”

Освітньо-професійна програма “Цифрові технології в енергетиці”

ЗАТВЕРДЖУЮ

Завідувач кафедри ЦТЕ

Наталія АУШЕВА

(підпис)

“ ” _____ 2026 р.

ЗАВДАННЯ

на дипломну роботу студенту

Тхоржевській Анні Сергіївні

(прізвище, ім’я, по батькові)

1. Тема роботи “Система автоматизованого обліку та управління запасами підприємства”

Науковий керівник Сегеда Ірина Василівна, к.е.н, доцент

(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “ ” _____ 2026 року № _____

2. Термін подання студентом роботи 04.06.2026

3. Вихідні дані до роботи мова програмування Python, фреймворк Flask, СУБД SQLite, середовище розробки VS Code.

4. Перелік питань, які потрібно розробити _____

1) провести аналіз серед наявних аналогів складського обліку;

- 2) визначити головні функції веб-застосунку, що керує послугами, ресурсами складової інфраструктури та моніторингом дефіциту;
 - 3) аргументувати вибрані інструменти, алгоритми, моделі оптимізації запасу та технології безпеки для реалізації веб-системи;
 - 4) побудувати архітектуру програмного забезпечення та реляційної бази даних;
 - 5) створити прикладний програмний веб-інтерфейс системи «SafeStock MS»;
 - 6) представити процес застосування веб-інтерфейсу та результати функціонального тестування.
5. Орієнтований перелік ілюстративного матеріалу схема, що показує роботу фреймворку Flask, схема архітектури проєкту, ER-діаграма класів бази даних, діаграма прецедентів (ролей), аналітична діаграма попиту, скріншоти роботи користувачів з веб-застосунком.
6. Дата видачі завдання 15.09.2025

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1.	Вибір теми роботи	15.01 – 20.01.2026	Виконано
2.	Аналіз методів та засобів розв’язання задачі	10.03 – 20.03.2026	Виконано
3.	Розробка архітектури та загальної структури системи	12.04 – 26.04.2026	Виконано
4.	Розробка окремих підсистем	27.04 – 01.05.2026	Виконано
5.	Програмна реалізація системи	01.05 – 12.05.2026	Виконано
6.	Оформлення пояснювальної записки	01.05 – 30.05.2026	Виконано
7.	Захист програмного забезпечення	13.05.2026	Виконано
8.	Передзахист	03.06.2026	Виконано
9.	Захист	15.06 – 19.06.2026	Виконано

Студент

(підпис)

Анна ТХОРЖЕВСЬКА

(прізвище та ініціали)

Керівник

(підпис)

Ірина СЕГЕДА

(прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота виконана на 89 сторінках, містить 10 ілюстрацій, 4 таблиці, 1 додаток, 32 джерела в переліку посилань.

Метою роботи є проектування, програмна реалізація та комплексна верифікація захищеної інформаційної системи автоматизованого обліку та управління запасами «SafeStock MS» для складського комплексу ТОВ «Медіа СВ», що мінімізує вплив людського фактора та оптимізує логістичні процеси. У ході дослідження використано методи системного аналізу предметної області, об'єктно-реляційного відображення (ORM), математичного моделювання точки замовлення (ROP), а також інженерні підходи безпечного програмування (Security-First).

Основний зміст роботи полягає у створенні відмовостійкого вебзастосунку на базі архітектурного патерну MVC з використанням мови Python, мікрофреймворку Flask та СКБД SQLite. У систему інтегровано трирівневу модель розмежування доступу (RBAC), Append-Only журнал безпеки (Audit Trail), механізм логічного видалення даних Soft Delete, а також інструменти динамічного QR-кодування номенклатури та автоматичного рендерингу аналітичних звітів у форматі PDF.

Результати впровадження розробок: створене програмне рішення успішно розгорнуто та протестовано в операційній інфраструктурі ТОВ «Медіа СВ». Система забезпечує стовідсоткову реляційну цілісність даних, унеможливорює приховане виправлення логів матеріально відповідальними особами, вчасно сигналізує про дефіцит товарів та лінійно масштабується без втрати швидкодії інтерфейсу, що підтверджує її високу практичну та фінансову доцільність порівняно з комерційними ERP-платформами.

Ключові слова: АВТОМАТИЗАЦІЯ СКЛАДУ, ФРЕЙМВОРК FLASK, ТОЧКА ЗАМОВЛЕННЯ, КЕРУВАННЯ ДОСТУПОМ, ЖУРНАЛ АУДИТУ, М'ЯКЕ ВИДАЛЕННЯ, КІБЕРБЕЗПЕКА, QR-КОДУВАННЯ, БАЗА ДАНИХ, ВЕБЗАСТОСУНОК.

ANNOTATION

The bachelor's thesis is executed on 89 pages, contains 10 illustrations, 4 tables, 1 appendice, and 32 sources in the list of references.

The purpose of the work is the design, software implementation, and comprehensive verification of the secure information system for automated inventory control and stock management "SafeStock MS" for the warehouse complex of "Media SV" LLC, which minimizes the human factor impact and optimizes logistics processes. During the research, methods of subject area systems analysis, object-relational mapping (ORM), mathematical Reorder Point (ROP) modeling, and Security-First software engineering approaches were applied.

The main content of the thesis lies in the creation of a fault-tolerant web application based on the MVC architectural pattern using the Python programming language, Flask microframework, and SQLite DBMS. The system integrates a three-tier Role-Based Access Control (RBAC) model, an Append-Only security Audit Trail log, a Soft Delete logical data removal mechanism, as well as tools for dynamic QR-coding of inventory and automatic rendering of analytical reports in PDF format.

Results of the development implementation: the created software solution was successfully deployed and tested within the operational infrastructure of "Media SV" LLC. The system ensures 100% relational data integrity, prevents hidden log modifications by financially accountable personnel, timely alerts on stockouts, and scales linearly without interface performance degradation, confirming its high practical and financial viability compared to commercial ERP platforms.

KEYWORDS: WAREHOUSE AUTOMATION, FLASK FRAMEWORK, REORDER POINT, ACCESS CONTROL, AUDIT TRAIL, SOFT DELETE, CYBERSECURITY, QR-CODING, DATABASE, WEB APPLICATION.

ЗМІСТ

ВСТУП.....	8
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ АВТОМАТИЗАЦІЇ ОБЛІКУ	11
1.1 Постановка прикладної задачі та опис діяльності об'єкта автоматизації (ТОВ «Медіа СВ»).....	11
1.2 Огляд теоретичних методів та математичних моделей управління запасами	13
1.3 Аналіз існуючих програмних засобів обліку та обґрунтування розробки кастомної системи.....	17
1.4 Висновки за розділом	23
2 МЕТОДИ ТА АЛГОРИТМИ АВТОМАТИЗОВАНОГО УПРАВЛІННЯ ЗАПАСАМИ ТА ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ ДАНИХ	25
2.1 Математична модель визначення точок поповнення складських залишків (Reorder Point).....	25
2.2 Алгоритмічне забезпечення оперативного моніторингу та сигналізації дефіциту («Low Stock Alert»).....	27
2.3 Модель розмежування прав доступу користувачів на основі ролей (RBAC)	29
2.4 Алгоритми криптографічного захисту та невідмовності дій у системі (Всрут та Audit Trail).....	33
2.4.1 Алгоритми криптографічного захисту облікових записів.....	33
2.4.2 Архітектура підсистеми невідмовності дій (Audit Trail)	34
2.5 Метод збереження цілісності реляційних даних при видаленні товарних позицій (Soft Delete).....	36
2.6 Висновки за розділом	39
3 ПРОЄКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ «SAFESTOCK MS».....	41
3.1 Архітектурно-програмна структура системи на базі патерну MVC та мікрофреймворку Flask.....	41
3.2 Проєктування структури реляційної бази даних та її відображення через SQLAlchemy ORM	45
3.3 Реалізація інтерфейсу користувача за допомогою шаблонізатора Jinja2 та фреймворку Bootstrap 5	48

3.4 Програмна реалізація модулів операційного обліку руху товарів та пагінації каталогу	49
3.4.1 Програмна реалізація операційного обліку руху товарів	49
3.4.2 Алгоритмічна реалізація сторінкової пагінації.....	49
3.5 Розробка підсистеми динамічного QR-кодування SKU та генерації звітних PDF-документів	50
3.5.1 Програмна реалізація підсистеми динамічного QR-кодування.....	50
3.5.2 Програмна реалізація підсистеми генерації звітних PDF-документів	51
3.6 Висновки за розділом	52
4 ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ.....	53
4.1 Інсталяція веб-застосунку та вимоги до обчислювальної техніки складського комплексу.....	53
4.2 Сценарії функціонального тестування компонентів системи та перевірка цілісності даних.....	56
4.3 Демонстрація інтерфейсу користувача та представлення сценаріїв роботи за тестами	61
4.3.1 Вікно автентифікації та реалізація рольового захисту (Тест-кейс №1)...	62
4.3.2 Головна робоча панель та віджет моніторингу залишків (Тест-кейс №3)	63
4.3.3 Модуль бізнес-аналітики та графічного представлення попиту.....	64
4.3.4 Панель наскрізного логування та невідмовності дій (Тест-кейс №4)	65
4.4 Порівняльний аналіз та оцінка переваг власної розробки відносно аналогічних систем	66
4.5 Висновки за розділом	70
ВИСНОВКИ.....	71
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	74

ВСТУП

Актуальність теми. У сучасних умовах динамічного ринку та високої конкуренції ефективне управління матеріально-технічними ресурсами є одним із ключових чинників забезпечення життєздатності, фінансової стабільності та операційної ефективності будь-якого торговельного підприємства. Запаси промислових товарів, електроніки та побутової техніки часто становлять значну частину активів компанії, а будь-які прорахунки чи затримки в їх оперативному обліку призводять до відчутних прямих фінансових втрат. Надлишкові обсяги товарних позицій нерационально заморожують обігові кошти організації, тоді як їх раптовий дефіцит спричиняє сповільнення або повну зупинку процесів відвантаження, зриви торговельних угод і, як наслідок, втрату лояльності з боку кінцевих клієнтів. Перехід від неефективного ручного контролю до системного управління складським господарством на основі точних та актуальних даних стає можливим виключно завдяки впровадженню глибокої автоматизації зазначених процесів.

Аналіз сучасного стану вирішення поставленої задачі на основі огляду фахової літератури свідчить, що використання застарілих методів обліку, таких як паперові носії, класичні комірські журнали або найпростіші табличні редактори загального призначення, більше не здатне забезпечити належну швидкість обробки інформації в умовах інтенсивного товарообігу. Подібні підходи повністю позбавлені необхідного захисного функціоналу, не дозволяють здійснювати безперервний автоматичний моніторинг залишків та суттєво підвищують ризик виникнення помилок через людський фактор. Для підприємств типу ТОВ «Медіа СВ», які здійснюють великовузлову оптову та роздрібну торгівлю широким асортиментом сучасної техніки, мобільних телефонів, аудіосистем та гаджетів, критично важливою є модернізація технологічної інфраструктури. Існуючі важкі комерційні ERP- та WMS-системи є надто громіздкими, фінансово обтяжливими у впровадженні та складними в кастомізації під специфічні інженерні потреби

безпеки конкретного середнього бізнесу. У зв'язку з цим виникає гостра потреба в розробці легкого, автономного веб-орієнтованого застосунку, який забезпечить миттєвий операційний облік транзакцій, надійний криптографічний захист, чітке розмежування прав доступу персоналу та прозоре некоректоване журналювання дій користувачів. Підставою та вихідними даними для розробки програмного забезпечення є зафіксовані функціональні вимоги до модернізації складського комплексу ТОВ «Медіа СВ», необхідність ліквідації дефіциту товарів та підвищення контролю за матеріально відповідальними особами.

Мета роботи. Метою роботи є розробка Web-застосунку автоматизованого обліку та управління запасами підприємства для оптимізації контролю руху товарних цінностей, забезпечення автоматичної візуальної сигналізації дефіциту та гарантування невідмовності й безпеки дій персоналу.

Завдання роботи. Для досягнення поставленої мети та відповідно до рекомендованої кафедрою схеми постановки завдань дипломного проектування, необхідно вирішити такі завдання:

1. Аналіз підходів, методів та алгоритмів розв'язання задачі автоматизованого складського обліку, математичного моделювання точок поповнення запасів, механізмів рольового розмежування прав доступу користувачів та журналювання подій.
2. Аналіз програмних засобів для реалізації програмної системи «SafeStock MS».
3. Розробка програмного забезпечення системи автоматизованого обліку та управління запасами відповідно до визначених архітектурних вимог, бізнес-процесів та криптографічних стандартів захисту даних.
4. Тестування розробленого програмного забезпечення для підтвердження цілісності реляційних даних, коректності авторизації та працездатності модулів сповіщення.

Структура й обсяг роботи. Кваліфікаційна бакалаврська робота містить такі обов'язкові компоненти та структурні елементи: титульний аркуш, технічне

завдання, анотацію українською і англійською мовами, зміст, вступ, основну частину, що складається з чотирьох розділів, висновки, список використаних джерел та додаток. Пояснювальна записка викладена на 89 сторінках текстової частини роботи без урахування додатків. Ілюстративний матеріал представлений у вигляді 10 рисунків. Цифрові та порівняльні дані систематизовано у 4 таблицях. Список використаних джерел сформований відповідно до чинних стандартів бібліографічного опису та містить 32 найменування. Документація та технічні матеріали винесені у 1 додатку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ АВТОМАТИЗАЦІЇ ОБЛІКУ

Метою першого розділу є проведення комплексного передпроектного дослідження об'єкта автоматизації, вивчення його організаційної структури, ключових логістичних процесів та формування чітких вимог до розроблюваної інформаційної системи. У підрозділі 1.1 наведено детально-функціональний опис діяльності компанії ТОВ «Медіа СВ», проаналізовано існуючу рольову модель управління персоналом та формалізовано прикладну задачу автоматизації. Підрозділ 1.2 присвячений теоретичному огляду сучасних математичних методів та логістичних моделей, які покладено в основу алгоритмічного забезпечення системи для оптимізації складських залишків. У підрозділі 1.3 здійснено порівняльний аналіз наявних на світовому ринку програмних рішень класу ERP та WMS, виявлено їхні критичні недоліки в контексті середнього бізнесу та обґрунтовано доцільність створення кастомного веб-застосунку «SafeStock MS» на базі сучасного мікрофреймворку.

1.1 Постановка прикладної задачі та опис діяльності об'єкта автоматизації (ТОВ «Медіа СВ»)

Об'єктом автоматизації у даній роботі виступає товариство з обмеженою відповідальністю «Медіа СВ», основним комерційним напрямом діяльності якого є великогуртова, дрібнооптова та роздрібна торгівля сучасною портативною технікою, цифровою електронікою та побутовими приладами. Товарний асортимент підприємства характеризується високою динамічністю оновлення та значною номенклатурною різноманітністю, що включає мобільні телефони,

смартфони, складні аудіосистеми, кліматичне та блендерне обладнання, зарядні станції та супутні гаджети.

Складське господарство підприємства функціонує як центральний, стратегічно важливий вузол інформаційних та матеріальних потоків, що пов'язує мережу іноземних та вітчизняних постачальників із кінцевими споживачами або роздрібними торговими точками. Ефективність функціонування всього логістичного ланцюга компанії безпосередньо залежить від точності, швидкості та надійності обліку кожної товарної позиції на складі.

У процесі дослідження внутрішньої структури ТОВ «Медіа СВ» було встановлено, що підприємство має багаторівневу вертикаль управління, де до щоденних операцій із матеріальними цінностями залучено персонал із принципово різними функціональними обов'язками та рівнями матеріальної відповідальності. Життєвий цикл руху техніки на складі складається з таких бізнес-процесів:

1. Первинне наповнення довідників: реєстрація у системі нових номенклатурних одиниць та карток контрагентів (постачальників).
2. Операційне документування руху товарів: фіксація транзакцій приходу (надходження від постачальників) та видатку (продаж, відвантаження клієнту або списання) одиниць техніки зі складу.
3. Контроль залишків: безперервний моніторинг поточного обсягу кожної позиції для своєчасного виявлення ризику *stockout* (абсолютного дефіциту).
4. Аналітика збуту: агрегація історичних даних про відвантаження для довгострокового планування закупівельних сесій.

Особливістю та критичною вимогою до організації зазначених процесів є необхідність суворого розмежування прав доступу до бази даних, що реалізується через тривірневу рольову модель:

- **Worker (Робітник / Комірник):** здійснює безпосередні рутинні фізичні операції, має право вносити дані про прийом або видачу техніки та реєструвати нові артикули.

- Manager (Менеджер): володіє розширеним доступом, керує категоріями товарного каталогу та коригує інформаційну базу постачальників.
- Admin (Адміністратор): здійснює повний технічний нагляд за системою, керує обліковими записами та аналізує журнал дій користувачів для запобігання внутрішнім зловживанням чи деструктивним помилкам персоналу.

Постановка прикладної задачі. Існуючий підхід до обліку на підприємстві характеризувався значним рівнем децентралізації, використанням розрізнених електронних таблиць та паперових відомостей. Це зумовлювало відсутність оперативного автоматичного інформування менеджменту про критичне зниження залишків ходових товарів, призводило до затримок при відвантаженні та унеможливлювало швидкий аудит дій персоналу у разі виявлення розбіжностей під час інвентаризації.

Отже, постає прикладна задача розробки веб-орієнтованої інформаційної системи, яка має забезпечити повну автоматизацію операційного обліку ТОВ «Медіа СВ», впровадити алгоритми миттєвого сповіщення про дефіцит, реалізувати інтелектуальні модулі маркування товарів і генерації звітності, а також гарантувати високу безпеку та невідмовність дій користувачів за рахунок наскрізного журналювання подій.

1.2 Огляд теоретичних методів та математичних моделей управління запасами

Ефективне функціонування будь-якого підприємства, діяльність якого пов'язана з оперуванням матеріальними потоками, безпосередньо залежить від обраної стратегії керування складськими ресурсами. У логістичній теорії запаси розглядаються як матеріальні цінності, що очікують на споживання або реалізацію і виступають демпфером між дискретними процесами постачання та збуту.

Оптимізація обсягу цих ресурсів є класичною компромісною задачею: надлишкові запаси призводять до іммобілізації (заморожування) обігових коштів та зростання витрат на зберігання, тоді як дефіцит спричиняє втрату потенційного прибутку, збої в обслуговуванні клієнтів та репутаційні ризики.

Для розв'язання цієї оптимізаційної задачі в сучасних інформаційних системах використовується комплекс математичних методів і моделей, які можна класифікувати за характером попиту на детерміновані та стохастичні.

До базових детермінованих моделей, що заклали основу теорії управління запасами, належить модель економічно обґрунтованого розміру замовлення (Economic Order Quantity, EOQ), відома як формула Вілсона. Модель базується на припущенні про постійний та відомий у часі темп споживання ресурсу. Математично оптимальний розмір партії замовлення (EOQ) визначається шляхом мінімізації сумарних витрат, які складаються з витрат на виконання замовлення (транспортно-заготівельні витрати) та витрат на зберігання одиниці товару на складі протягом року. Формула має наступний вигляд:

$$EOQ = \sqrt{\frac{2DS}{H}} \quad (1.1)$$

де EOQ — економічно обґрунтований розмір замовлення (одиниць);

D — річний обсяг попиту на дану товарну позицію (одиниць);

S — витрати на організацію та доставку одного замовлення (грн/замовлення);

H — річні витрати на зберігання однієї одиниці товару на складі (грн/од. за рік).

У реальних умовах функціонування торговельних підприємств, зокрема таких як ТОВ «Медіа СВ», попит на цифрову техніку та електроніку є стохастичним, тобто зазнає випадкових коливань, зумовлених ринковими факторами, сезонністю та маркетинговою активністю конкурентів. За таких умов статичні детерміновані моделі виявляються неефективними, що зумовлює необхідність застосування динамічних моделей із фіксованим рівнем запасу — методів визначення точки поповнення замовлення (Reorder Point, ROP).

Метод ROP передбачає, що нова партія товару розміром EOQ (або фіксованого обсягу) замовляється у постачальника в той момент, коли поточний рівень складських залишків опускається до критичної межі. Цей поріг повинен бути достатнім для покриття потреби в товарі протягом часу логістичного лагу (періоду між розміщенням замовлення та фізичним надходженням техніки на склад). Математична модель розрахунку точки замовлення має вигляд:

$$ROP = (d \times L) + SS \quad (1.2)$$

де ROP — точка замовлення (критичний рівень залишку, одиниць);

d — середньодобовий обсяг попиту (відвантаження) товару (одиниць/день);

L — тривалість виконання замовлення постачальником (днів);

SS — страховий (буферний) запас (одиниць).

Величина страхового запасу (SS) призначена для компенсації випадкових коливань попиту або можливих затримок у ланцюгу постачання. У стохастичних моделях вона розраховується на основі заданого рівня сервісу (імовірності бездефіцитної роботи) та середньоквадратичного відхилення попиту. Для об'єкта автоматизації дослідження емпіричним шляхом було встановлено усереднену константу критичного рівня на позначці 10 одиниць. Це дозволяє системі в автоматичному режимі сигналізувати про наближення ризику дефіциту (stockout).

Поряд із визначенням моментів поповнення запасів, важливим теоретичним аспектом є багатокритеріальна класифікація асортименту, оскільки однорідне управління тисячами номенклатурних позицій (SKU) є економічно недоцільним і перевантажує обчислювальні потужності системи. У сучасній практиці автоматизації для цього застосовують суміщений аналіз моделей ABC та XYZ.

Метод ABC-аналізу базується на емпіричному законі Парето і передбачає ранжування складських запасів за рівнем їхньої значущості, обсягу продажів або сумарної вартості для підприємства. Відповідно до цього підходу, каталог товарів розділяється на три категорії:

- Категорія «А»: найбільш пріоритетні товари, що становлять близько 15–20% від загальної кількості найменувань, але забезпечують до 75–80% грошового

обігу або інтенсивності збуту підприємства. Для ТОВ «Медіа СВ» це високоліквідна техніка (смартфони, ходові моделі побутових приладів, портативні зарядні станції). Вони потребують жорсткого, щоденного автоматизованого контролю та мінімального страхового запасу завдяки високій швидкості обертання.

- Категорія «В»: проміжні товари середньої важливості (близько 30% асортименту, що генерують 15% обігу). Вони підлягають стандартному обліку та плановому моніторингу залишків.
- Категорія «С»: найменш цінні, але найчисленніші позиції (до 50% номенклатури, що забезпечують не більше 5% обігу) — аксесуари, дрібні комплектуючі. Для них застосовуються спрощені процедури обліку, а замовлення можуть формуватися великими партіями для зниження адміністративних витрат.

Для врахування фактора прогнозованості та стабільності попиту модель ABC розширюють математичним інструментом XYZ-аналізу. У межах цього методу всі номенклатурні позиції класифікуються залежно від характеру їхнього споживання та точності прогнозування змін у потребі. Основним математичним критерієм тут виступає коефіцієнт варіації попиту (v), який розраховується за формулою:

$$v = \frac{\sigma}{\bar{x}} \times 100\% \quad (1.3)$$

де v — коефіцієнт варіації попиту (%);

σ — середньоквадратичне відхилення попиту за аналізований період;

$\bar{x}$ — середнє арифметичне значення попиту за цей же період.

Залежно від значення коефіцієнта варіації, товари розподіляються по трьох групах:

- Група «Х»: стабільний попит, коефіцієнт варіації не перевищує 10%. Потреба в таких товарах є постійною, що дозволяє точно прогнозувати обсяги закупівель і підтримувати мінімальний буферний запас.

- Група «Y»: нестабільний або сезонний попит, тенденції якого можна передбачити (коефіцієнт варіації від 10% до 25%). Рівень залишків потребує коригування відповідно до календарних періодів чи ринкових трендів.
- Група «Z»: випадковий, нерегулярний попит, який практично не піддається прогнозуванню (коефіцієнт варіації перевищує 25%). Для таких позицій характерна низька точність розрахунків, тому забезпечення безперебійності продажів вимагає створення значних страхових запасів.

Інтеграція та суміщення моделей ABC і XYZ у вигляді двовимірної матриці (що складається з 9 реляційних комірок: від AX до CZ) є найвищим стандартом сучасного математичного моделювання складських систем. Перетинання категорій дозволяє диференціювати логіку управління: наприклад, позиції класу AX (високий обіг, стабільний попит) є основою бізнесу і потребують автоматичного розрахунку точок поповнення за формулою (1.2) та налаштування інтелектуальних сповіщень у реальному часі. Натомість позиції класу AZ або CZ (низька прогнозованість) потребують індивідуального підходу, специфічного аналізу відвантажень та гнучкого операційного регулювання.

Таким чином, наведений теоретичний базис та математичні моделі (EOQ, ROP, ABC/XYZ) формують повне інженерне підґрунтя для алгоритмічного ядра розроблюваної інформаційної системи «SafeStock MS». Вони дозволяють автоматизувати аналітичні процеси, структурувати великі обсяги даних про транзакції та перейти від ручного моніторингу складських одиниць до інтелектуального управління ресурсами підприємства в реальному часі.

1.3 Аналіз існуючих програмних засобів обліку та обґрунтування розробки кастомної системи

У ході передпроектного дослідження та розробки технічних вимог до системи було проведено глибокий критичний аналіз наявного на світовому ринку

програмного забезпечення для автоматизації складської логістики. Сучасний ринок ІТ-рішень пропонує широкий спектр платформ, які можна розділити на три основні категорії: важкі корпоративні ERP-системи, модульні рішення з відкритим кодом та хмарні бухгалтерсько-облікові сервіси. Для порівняльного аналізу було обрано найбільш репрезентативні продукти у своїх класах: SAP Business One, Odoo Enterprise та QuickBooks.

Корпоративні ERP-системи, такі як SAP Business One, орієнтовані на великий бізнес і пропонують всеохопний, проте часто надлишковий функціонал для середнього торговельного підприємства. Головними недоліками впровадження платформ такого рівня для ТОВ «Медіа СВ» є надзвичайно висока вартість ліцензування, тривалий і фінансово обтяжливий процес конфігурації сертифікованими спеціалістами, а також складний інтерфейс, який вимагає додаткового навчання персоналу.

Альтернативою виступають системи з відкритим кодом, зокрема Odoo Enterprise. Завдяки своїй модульній структурі вона дозволяє гнучко підбирати необхідні інструменти. Проте вартість передплати за комерційні модулі швидко зростає при масштабуванні, а стандартна логіка управління ролями вимагає суттєвого доопрацювання на мові Python для забезпечення необхідного рівня безпеки та кастомного аудит-логіну.

Хмарні сервіси обліку, яскравим представником яких є QuickBooks, орієнтовані на сегмент малого та середнього бізнесу і відрізняються відносно низьким порогом входження. Разом з тим, вони мають критичні обмеження щодо кастомізації під специфічні бізнес-процеси конкретного підприємства. Використання подібних пропрієтарних хмарних рішень створює жорстку залежність компанії від сторонніх серверів, зовнішньої геополітичної ситуації та регулярних ліцензійних виплат. Крім того, QuickBooks має обмежені можливості побудови ізольованого журналу аудиту дій користувачів, що є недопустимим для системи з високими вимогами до невідмовності та кібербезпеки.

Для об'єктивного зіставлення розробленого Web-застосунку «SafeStock MS» з існуючими ринковими альтернативами було сформовано порівняльну матрицю за ключовими техніко-економічними та інженерними критеріями (таблиця 1.1).

Таблиця 1.1 — Порівняльна характеристика розробленої ІС з ринковими аналогами

Критерій порівняння	SAP Business One	Odoo (Enterprise)	QuickBooks	SafeStock MS (Власна система)
Фінансова вартість (ТСО)	Дуже висока (придбання дорогих ліцензій та обов'язана оплата послуг сертифікованих інтеграторів)	Середня (регулярна абонентська передплата за кожен окремо активований комерційний модуль)	Щомісячна фіксована абонентська плата за кількість підключених користувачів	Мінімальна (використання Open-source технологічного стеку, безкоштовне Self-hosted розгортання)
Складність та адаптивність UX	Висока (перевантажений інтерфейс із надлишковим інструментарієм, потребує тривалого навчання)	Середня (складна ієрархія вкладених модульних меню, що уповільнює рутинні операції)	Низька/Середня (жорстко зафіксована логіка відображення, орієнтована переважно на бухгалтерів)	Мінімальна (інтуїтивно зрозумілий адаптивний UX, оптимізований під швидку роботу на складських терміналах)

Продовження таблиці 1.1

Рівень кастомізації коду	Складна (будь-які зміни структури проводяться виключно через офіційних партнерів вендора)	Висока (реалізується шляхом самостійного написання та інтеграції додаткових Python-модулів)	Обмежена (налаштування системи затиснуті в жорсткі рамки пресетів хмарного провайдера)	Максимальна (повний і незалежний одноосібний контроль над кожною лінією вихідного коду застосунку)
Керування доступом (RBAC)	Надлишково складна багаторівнева матриця прав доступу, орієнтована на великі корпорації	Гнучка налаштовувана система внутрішніх групових конфігурацій та допусків	Стандартні зафіксовані виробником пресети ролей без можливості гнучких змін	Чітка і сувора (ізоляція маршрутів на рівні декораторів Flask під ролі: Admin, Manager, Worker)
Журналювання та безпека даних	Повний фінансовий облік здійснених змін із високими вимогами до хостингу	Залежить від типу та вартості обраної конфігурації, логування потребує додаткового налаштування	Обмежений внутрішній лог системних подій без глибокої деталізації дій оператора	Security-First (вбудований некоректований Audit Log для кожної дії та м'яке видалення Soft Delete)

Результати проведеного аналізу чітко обґрунтовують доцільність проектування та реалізації індивідуального кастомного рішення «SafeStock MS» для автоматизації складських процесів ТОВ «Медіа СВ». Власна розробка володіє низкою суттєвих інженерних переваг:

1. Легкість та швидкодія архітектури: завдяки використанню мікрофреймворку Flask забезпечується висока швидкість обробки HTTP-запитів та рендерингу сторінок навіть під час пагінації великих масивів даних.
2. Безпека на основі моделі RBAC: чітке розмежування прав доступу між ролями Admin, Manager та Worker на рівні декораторів Flask мінімізує ризики несанкціонованого видалення записів або витоку комерційної таємниці.
3. Технічна та ліцензійна автономність: розроблений веб-застосунок не залежить від зовнішніх пропрієтарних ліцензій та хмарних провайдерів і може бути розгорнутий безпосередньо на локальному сервері підприємства (Self-hosted).

Як головний інструмент реалізації серверної частини було обрано мову програмування Python та вебфреймворк Flask. У сучасній практиці веб-розробки часто постає вибір між фреймворками Django та Flask. Вибір Flask у даній роботі є повністю обґрунтованим з інженерної точки зору.

Django є важким, монолітним фреймворком, що функціонує за принципом «все включено за замовчуванням» (built-in admin, ORM, аутентифікація, форми). Він нав'язує розробнику жорстку архітектурну структуру і містить велику кількість вбудованого «баласту» — компонентів, які не використовуються у спеціалізованих складських системах, проте створюють додаткове навантаження на обчислювальні ресурси та розширюють потенційну поверхню атак для злоумисників.

Натомість Flask має виражену «мікро»-природу. Він надає розробнику лише базове ядро для обробки маршрутизації та запитів, не нав'язуючи сторонніх компонентів. Це дозволило спроектувати чисту, лаконічну та оптимізовану бізнес-логіку керування запасами, чітко контролювати кожен рядок вихідного коду, самостійно реалізувати кастомний криптографічний захист та побудувати

ізолювану підсистему некоректованого аудит-логіну відповідно до специфічних вимог кібербезпеки ТОВ «Медіа СВ».

Програмний продукт спроектований із високим потенціалом до масштабування. Використання об'єктно-реляційного відображення SQLAlchemy ORM робить систему повністю Database-Agnostic (незалежною від конкретного типу СУБД). Для функціонування поточного прототипу системи та перевірки її працездатності використовується компактна реляційна база даних SQLite, яка зберігається в одному локальному файлі й не потребує окремого сервера. Проте у разі стрімкого розширення бізнесу та збільшення кількості товарних позицій до 100 000 і більше, розроблена архітектура дозволяє за кілька хвилин змінити рядок підключення у конфігурації (sqlite:///inventory.db) на промислові СКБД PostgreSQL або MySQL без необхідності переписування та модифікації вихідного коду бізнес-логіки системи. Оскільки в інтерфейсі користувача вже на етапі проєктування закладено та реалізовано алгоритми сторінкової пагінації (відображення строго по 10 товарних одиниць на сторінку), графічний веб-інтерфейс системи не зазнаватиме затримок або критичних падінь продуктивності при роботі з базами даних надвеликих обсягів.

Безпека даних у SafeStock MS забезпечується на рівні сучасних криптографічних стандартів. Для захисту облікових записів інтегровано бібліотеку Wsyrpt, яка реалізує стійке хешування паролів за алгоритмом Blowfish. Даний підхід унеможливорює збереження паролів у відкритому вигляді: у базі даних фіксується лише унікальний “солений” хеш. Навіть у випадку критичного витоку або повного компрометування файлу бази даних, зловмисник не зможе відновити первинні паролі користувачів за допомогою таблиць відповідностей чи методів brute-force атак.

Цілісність історичних даних та фінансової звітності підприємства гарантується впровадженням інженерної техніки м'якого видалення (Soft Delete). При видаленні товарної позиції з каталогу вона не знищується фізично з таблиць бази даних за допомогою SQL-команди DELETE, що могло б призвести до

каскадного руйнування зв'язків у таблицях проведених раніше транзакцій та викривлення аналітичних звітів. Замість цього система змінює логічний прапорець у полі `is_active` на значення `False`, приховуючи товар з інтерфейсів працівників складу, але повністю зберігаючи історичний цифровий слід для проведення повноцінного аудиту.

Додатковою інноваційною «фішкою» системи є інтеграція складських процесів із фізичним світом за допомогою динамічного QR-кодування. Для кожного унікального артикула (SKU) система автоматично генерує графічний ідентифікатор через вбудований інструментарій API. Це виключає потребу у придбанні дорогого стороннього програмного забезпечення для маркування продукції, повністю усуває людський фактор при ручному введенні серійних номерів комірниками і закладає готовий технологічний фундамент для підключення апаратних сканерів або мобільних додатків безпосередньо у складських приміщеннях ТОВ «Медіа СВ».

Таким чином, розробка кастомної інформаційної системи SafeStock MS на базі Flask є оптимальним інженерним рішенням, що поєднує високу економічну ефективність, архітектурну гнучкість, експлуатаційну надійність та безкомпромісний підхід до кібербезпеки і цілісності корпоративних даних

1.4 Висновки за розділом

У першому розділі кваліфікаційної бакалаврської роботи було проведено комплексний передпроектний аналіз предметної області, досліджено логістичні процеси об'єкта автоматизації та обґрунтовано інженерні підходи до побудови програмного забезпечення. За результатами виконання аналітичного етапу дослідження сформульовано такі прикладні висновки:

1. Досліджено організаційно-виробничу структуру та детально проаналізовано бізнес-процеси руху матеріальних цінностей у складському комплексі ТОВ

«Медіа СВ». Виявлено, що існуючі децентралізовані методи обліку цифрової техніки та електроніки супроводжуються високим ризиком виникнення помилок при ручному введенні серійних номерів, критичними затримками при формуванні звітності, а також повною незахищеністю даних від внутрішніх зловмисних маніпуляцій, що обґрунтовує гостру необхідність розробки єдиної інтегрованої вебсистеми.

2. Формалізовано повний перелік вимог до створюваної інформаційної системи «SafeStock MS». Визначено суворі функціональні критерії (зокрема: автоматичний моніторинг дефіциту товарів, динамічне маркування артикулів та генерація суворих друкованих форм звітів) та нефункціональні вимоги в частині масштабованості сховища, швидкодії інтерфейсу користувача та суворого розмежування прав доступу персоналу.
3. Проведено критичний огляд існуючих теоретичних методів управління складськими запасами (моделей EOQ, ROP, ABC/XYZ-аналізу). На основі аналізу специфіки обігу побутової електроніки об'єкта автоматизації було математично обґрунтовано та інтегровано в логіку системи константу критичної межі залишків на рівні 10 одиниць для активації візуального віджета попередження про дефіцит.
4. Здійснено порівняльний аналіз розробленого рішення з провідними світовими та локальними ERP/WMS-платформами (SAP Business One, Odoo, QuickBooks). Доведено, що кастомна розробка на базі безкоштовного open-source технологічного стеку мови Python та мікрофреймворку Flask забезпечує підприємству ТОВ «Медіа СВ» абсолютну ліцензійну та інфраструктурну незалежність, мінімальну сукупну вартість володіння (ТСО), максимальну гнучкість кастомізації вихідного коду, а також безкомпромісний рівень безпеки даних за принципом Security-First.

2 МЕТОДИ ТА АЛГОРИТМИ АВТОМАТИЗОВАНОГО УПРАВЛІННЯ ЗАПАСАМИ ТА ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ ДАНИХ

Метою другого розділу є детальне математичне та алгоритмічне обґрунтування теоретичних методів, моделей та інженерних рішень, які покладено в основу функціонування інформаційної системи «SafeStock MS». Оскільки цей розділ є науково-методичним ядром кваліфікаційної роботи, тут розкривається формалізований апарат управління запасами та механізми забезпечення інформаційної безпеки системи. У підрозділі 2.1 наведено поглиблений математичний опис стохастичної моделі визначення точок поповнення складських залишків. Підрозділ 2.2 присвячено алгоритмічній логіці функціонування модуля «Low Stock Alert» та його інтеграції з шаблонізатором інтерфейсу. У підрозділах 2.3, 2.4 та 2.5 представлено формалізований опис підсистем безпеки: рольової моделі доступу (RBAC), криптографічного захисту паролів за допомогою алгоритму Vcrypt, некоректованого журналу дій (Audit Trail) та архітектурного патерну м'якого видалення даних (Soft Delete).

2.1 Математична модель визначення точок поповнення складських залишків (Reorder Point)

Для забезпечення безперебійного функціонування складського комплексу ТОВ «Медіа СВ» в умовах випадкових (стохастичних) коливань ринкового попиту та неминучих затримок у логістичному ланцюгу, автоматизована система повинна базуватися на математично обґрунтованій моделі визначення моменту ініціації замовлення на поставку. Класичний підхід, що спирається на фіксований рівень

запасу, передбачає обчислення критичного порогу — точки замовлення (Reorder Point, ROP).

Математична формалізація моделі в умовах невизначеності попиту та часового лагу базується на засадах теорії ймовірностей та математичної статистики. Якщо припустити, що добовий попит на товар є випадковою величиною, яка підпорядковується закону нормального розподілу, то загальне рівняння для розрахунку точки замовлення має вигляд:

$$ROP = (\bar{d} \times \bar{L}) + SS \quad (2.1)$$

де $\bar{d}$ — середнє значення добового попиту на конкретну товарну позицію (одиниць/день);

$\bar{L}$ — середній час виконання замовлення постачальником від моменту його розміщення до фізичного надходження техніки на склад (днів);

SS — страховий (буферний) запас, призначений для захисту від дефіциту під час випадкового перевищення середнього попиту або затримок у постачанні (одиниць).

Величина страхового запасу (SS) безпосередньо корелює із заданим рівнем сервісу (імовірністю бездефіцитної роботи системи протягом часу виконання замовлення) і визначається за формулою:

$$SS = Z \times \sigma_{DL} \quad (2.2)$$

де Z — коефіцієнт нормального розподілу (коефіцієнт сервісу), який вибирається зі статистичних таблиць залежно від цільової ймовірності відсутності дефіциту (наприклад, для рівня сервісу 95% $Z=1,64$, для 99% $Z=2,33$);

σ_{DL} — середньоквадратичне відхилення попиту за час виконання замовлення.

У випадку, коли випадковими та незалежними величинами є і добовий попит, і час доставки, сумарне середньоквадратичне відхилення розраховується через дисперсії обох процесів за формулою:

$$\sigma_{DL} = \sqrt{\bar{L} \times \sigma_d^2 + \bar{d}^2 + \times \sigma_L^2} \quad (2.3)$$

де σ_d — середньоквадратичне відхилення добового попиту;

σ_L — середньоквадратичне відхилення часу виконання замовлення постачальником.

На основі статистичного аналізу історичних даних відвантажень ТОВ «Медіа СВ» за попередні періоди було встановлено, що для ключових категорій товарів (цифрової техніки та електроніки) середній логістичний лаг становить $\bar{L} = 3$ дні, а середній добовий попит — $\bar{d} = 2$ одиниці. З урахуванням коливань попиту та необхідності забезпечення високого рівня сервісу (на рівні 95%), розрахована величина безпечного порогу (ROP) для більшості товарних позицій наближається до константи у 10 одиниць. Саме це значення інтегровано в програмну логіку веб-застосунку як базовий параметр ініціалізації тригерів сповіщення про дефіцит.

Інтеграція даної моделі в систему «SafeStock MS» дозволяє автоматично агрегувати дані з таблиці Transaction, обчислювати динамічні показники попиту та завчасно сигналізувати менеджменту про необхідність поповнення складських запасів, що суттєво мінімізує ймовірність виникнення критичних ситуацій *stockout*.

2.2 Алгоритмічне забезпечення оперативного моніторингу та сигналізації дефіциту («Low Stock Alert»)

Сучасні системи управління складом повинні функціонувати не просто як пасивні сховища історичних даних про товарні залишки, а як проактивні інструменти підтримки прийняття рішень. Головним завданням модуля «Low Stock Alert» в інформаційній системі «SafeStock MS» є безперервний автоматизований моніторинг кількісних показників номенклатурних одиниць (Q_{current}) у режимі реального часу та миттєве сповіщення відповідального персоналу про наближення до критичної межі дефіциту (ROP), яка на основі логістичного аналізу діяльності ТОВ «Медіа СВ» встановлена на рівні 10 одиниць.

З інженерної точки зору алгоритм розв'язує задачу асинхронного або синхронного тригерування подій зміни стану об'єктів реляційної моделі даних. Архітектура цього алгоритмічного рішення інтегрована на всіх рівнях шаблону MVC (Model-View-Controller) розроблені веб-застосунку:

1. Рівень моделі (Model): База даних через ORM SQLAlchemy відстежує кожну транзакцію, що змінює значення поля *quantity* у таблиці *Product*.
2. Рівень контролера (Controller): При кожному HTTP-запиті користувача до головної панелі або каталогу, контролер у файлі *app.py* виконує вибірку актуальних даних і передає масив об'єктів у шар представлення.
3. Рівень представлення (View): Шаблонізатор Jinja2 виконує динамічну логічну перевірку безпосередньо під час генерації HTML-коду сторінки для клієнта.

Математично логіку тригерування візуальної сигналізації можна описати системою умовних виразів для кожної *i*-ї товарної позиції:

$$\text{Статус } P_i = \begin{cases} \text{Critical (table - danger)} , \text{ якщо } Q_{\text{current}_i} < ROP \\ \text{OK (table - success)} , \text{ якщо } Q_{\text{current}_i} \geq ROP \end{cases} \quad (2.4)$$

Для детального розуміння послідовності операцій та логіки розгалуження процесів під час моніторингу залишків, нижче наведено формалізовану блок-схему (рис. 2.1) алгоритму «Low Stock Alert».



Рисунок 2.1 — Формалізована блок-схему алгоритму «Low Stock Alert»

Перевага розробленого алгоритмічного підходу полягає в його високій швидкодії та повній автоматизації процесу. Користувачеві (комірнику чи менеджеру складу) не потрібно запускати спеціалізовані скрипти перевірки, будувати щоденні звіти чи вручну звіряти тисячі артикулів техніки. Як тільки в системі реєструється видаткова транзакція (наприклад, продаж або списання музичної колонки чи блендера), кількість товару автоматично декрементується у базі даних. Під час наступного оновлення сторінки або звернення до API, алгоритм миттєво оцінює новий стан активів і, якщо залишок стає меншим за 10 одиниць, автоматично маркує весь рядок таблиці яскравим червоним кольором (table-danger).

Таке інженерне рішення мінімізує вплив людського фактора, фокусує увагу менеджменту виключно на проблемних позиціях складу і дозволяє ТОВ «Медіа СВ» упереджувати виникнення простоїв у торгівлі, забезпечуючи високу операційну готовність до вчасного формування замовлень постачальникам.

2.3 Модель розмежування прав доступу користувачів на основі ролей (RBAC)

Забезпечення належного рівня безпеки інформаційних ресурсів у сучасних системах автоматизації складського обліку вимагає впровадження суворих механізмів контролю доступу. В інформаційній системі «SafeStock MS» захист комерційних даних, запобігання внутрішнім загрозам (навмисним або випадковим маніпуляціям із залишками) та реалізація принципу найменших привілеїв (Principle of Least Privilege) досягається за допомогою математично формалізованої моделі керування доступом на основі ролей (Role-Based Access Control, RBAC).

Згідно з класичною концепцією RBAC, доступ до захищених об'єктів та виконання операцій у системі не призначаються безпосередньо конкретним користувачам. Замість цього всі дозволи асоціюються з абстрактними сутностями

— ролями, які, у свою чергу, розподіляються між персоналом підприємства ТОВ «Медіа СВ». Математично таку модель можна описати як багатокomпонентну структуру, що складається з множини користувачів (U), множини ролей (R), множини дозволів або системних операцій (P) та сесій (S).

Відношення призначення користувачів на ролі (UserAssignment) та призначення дозволів для ролей (PermissionAssignment) формалізуються як підмножини декартового добутку відповідних множин:

$$UA \subseteq U \times R \quad (2.5)$$

$$PA \subseteq P \times R \quad (2.6)$$

Функція $\text{assigned}_{\text{permissions}(r)}$, яка повертає множину операцій, дозволених для конкретної ролі $r \in R$, записується у вигляді:

$$\text{assigned}_{\text{permissions}(r)} = \{p \in P | (p, r) \in PA\} \quad (2.7)$$

Для реалізації специфіки операційної діяльності ТОВ «Медіа СВ» у структурі ІС було зафіксовано чітку трирівневу ієрархію ролей ($R = \{\text{Worker}, \text{Manager}, \text{Admin}\}$), де кожна вища роль успадковує або розширює права нижчої, утворюючи строгий дискретний граф доступу.

Детальний інженерно-технічний опис та функціональні межі кожної ролі в системі «SafeStock MS» визначено наступним чином:

1. Роль «Worker» (Комірник / Робітник складу) призначається персоналу, який безпосередньо взаємодіє з матеріальними цінностями на фізичних локаціях складу. Має мінімально необхідний набір повноважень. Користувачеві дозволено авторизуватися в системі, переглядати поточний каталог товарних позицій, використовувати пошук, механізми пагінації та створювати записи про рух товарів у межах операцій приходу (інкремент залишку) або видатку (декремент залишку). Спроби модифікувати картку товару, змінити артикул, видалити запис чи перейти до системних налаштувань повністю блокуються архітектурою веб-застосунку.
2. Роль «Manager» (Менеджер логістики) орієнтована на співробітників, які здійснюють операційне керування асортиментом та взаємодіють із

контрагентами. Менеджер повністю успадковує всі права ролі Worker. Додатково йому надаються привілеї на створення нових номенклатурних карток (SKU), редагування описів наявної цифрової техніки, керування довідниками категорій та постачальників, а також формування та вивантаження динамічних аналітичних звітів у форматі PDF та роботу з інструментами QR-кодування продукції.

3. Роль «Admin» (Адміністратор системи) володіє абсолютним (суперкористувацьким) набором привілеїв у межах інформаційного середовища. Окрім виконання стандартних складських та менеджерських операцій, адміністратор здійснює технічний нагляд за функціонуванням платформи: керує обліковими записами співробітників, здійснює операцію створення користувачів та динамічного призначення/зміни їхніх системних ролей. Тільки роль Admin має доступ до перегляду некоректованого журналу аудиту дій (AuditLog), що гарантує незалежність та об'єктивність внутрішнього контролю безпеки на підприємстві.

Для наочного представлення розподілу прав та швидкої верифікації логіки розмежування доступу під час тестування системи було сформовано декартову матрицю доступу (таблиця 2.1).

Таблиця 2.1 — Матриця розмежування доступу до функціональних модулів IC SafeStock MS

Функціональний маршрут / Модуль системи (P)	Worker	Manager	Admin
Перегляд інтерфейсу каталогу (/products)	Дозволено	Дозволено	Дозволено

Продовження таблиці 2.1

Створення транзакцій приходу/видатку (/transaction/add)	Дозволено	Дозволено	Дозволено
Створення та редагування товарних карток (/product/create)	Заборонено	Дозволено	Дозволено
Експорт аналітики в PDF та QR-генерація (/export/pdf)	Заборонено	Дозволено	Дозволено
Керування ролями та користувачами (/admin/users)	Заборонено	Заборонено	Дозволено
Перегляд некоректованого Audit Trail (/admin/audit-log)	Заборонено	Заборонено	Дозволено

На рівні програмної реалізації вебфреймворку Flask захист ендпоінтів від несанкційованого доступу реалізовано безпосередньо в коді контролерів шляхом створення та інтеграції кастомних декораторів функцій Python. Це дозволяє винести логіку перевірки прав із бізнес-коду маршрутів, забезпечуючи чистоту та читаність архітектури.

Використання розробленого декоратора на практиці дозволяє захистити будь-яку URL-адресу застосунку в один рядок коду, декларативно вказуючи права.

У разі, якщо зловмисник або неавторизований працівник із нижчими привілеями (наприклад, Worker) спробує безпосередньо ввести захищену URL-адресу /admin/audit-log в адресний рядок браузера в обхід графічного інтерфейсу, декоратор моментально перехопить цей потік керування, заблокує виконання SQL-запиту до таблиць логів та згенерує системне виключення HTTP 403 Forbidden. Така архітектура повністю нівелює ризики горизонтального та вертикального підвищення привілеїв (Privilege Escalation), забезпечуючи високу стійкість

розроблюваної системи «SafeStock MS» до внутрішніх кіберзагроз та гарантуючи суворе дотримання політики інформаційної безпеки ТОВ «Медіа СВ».

2.4 Алгоритми криптографічного захисту та невідмовності дій у системі (Bcrypt та Audit Trail)

Безпека збереження облікових даних персоналу та забезпечення принципу невідмовності (non-repudiation) дій матеріально відповідальних осіб є критично важливими вимогами до архітектури сучасних складських ІС. У системі «SafeStock MS» ці завдання вирішуються шляхом інтеграції стійких криптографічних алгоритмів та проектування ізольованої підсистеми наскрізного логування подій.

2.4.1 Алгоритми криптографічного захисту облікових записів

Зберігання паролів користувачів у відкритому текстовому вигляді (plain text) або з використанням застарілих та скомпрометованих функцій хешування (таких як MD5 або SHA-1) створює серйозні ризики для безпеки підприємства ТОВ «Медіа СВ». У разі несанкціонованого доступу до файлу бази даних зловмисник може миттєво отримати автентифікаційні дані всіх співробітників.

Для нейтралізації цієї загрози в SafeStock MS інтегровано адаптивний криптографічний алгоритм хешування Bcrypt, який базується на симетричному шифрі Blowfish. Головною перевагою Bcrypt є його стійкість до атак за допомогою попередньо обчислених таблиць (Rainbow Tables) та висока резистентність до швидкого перебору паролів (brute-force).

Математично процес генерації захищеного запису пароля можна представити у вигляді функції:

$$Bcrypt_Hash = 2b\{cost\}\{salt\}\{hashed_value\} \quad (2.8)$$

Алгоритм реалізує дві ключові інженерні концепції:

- Динамічне соління (Salting) - перед обчисленням хешу до кожного пароля автоматично додається унікальна випадкова послідовність біт (сіль) довжиною 128 біт. Це гарантує, що якщо два користувачі виберуть абсолютно однакові паролі, їхні підсумкові хеш-значення у базі даних будуть кардинально відрізнятися.
- Регульована обчислювальна складність (Key Stretching): За допомогою параметра *cost* (робочий фактор) задається кількість ітерацій внутрішнього циклу шифрування (кількість раундів становить 2^{cost}). Це дозволяє штучно уповільнювати процес обчислення одного хешу на рівні процесора (наприклад, до 100–300 мс). Для зловмисника, який намагається виконати мільярди підборів паролів на секунду, таке уповільнення робить атаку технічно та фінансово неможливою навіть за умови використання спеціалізованих GPU-кластерів.

Програмна логіка взаємодії з криптографічним модулем у контролері Flask під час реєстрації нового користувача та його подальшої автентифікації реалізована за допомогою бібліотеки `flask_bcrypt`.

2.4.2 Архітектура підсистеми невідмовності дій (Audit Trail)

Одним із головних стовпів унікальності та надійності системи «SafeStock MS» є реалізація механізму повної невідмовності дій (non-repudiation). Це означає, що жоден користувач системи (включаючи комірників та менеджерів) не може заперечити факт виконання ним тієї чи іншої операції з матеріальними активами підприємства, оскільки кожна маніпуляція залишає стійкий і деталізований цифровий слід.

Для цього на рівні бази даних розроблено ізольовану реляційну модель AuditLog, яка функціонує за принципом архітектурного патерну Append-Only (дані

можуть лише додаватися, а операції зміни UPDATE або фізичного видалення DELETE заблоковані на рівні СКБД).

Кожен запис у журналі аудиту є строго структурованим об'єктом, що містить чотири обов'язкові вимірні координати події:

$$\text{Запис Логу} = \langle \text{Timestamp}, \text{User}_{ID}, \text{Action}, \text{Details} \rangle \quad (2.9)$$

де Timestamp — системна мітка часу сервера (з точністю до секунди), що фіксує момент транзакції.

User_ID — зовнішній ключ, який чітко зв'язує подію з конкретним автентифікованим обліковим записом співробітника, який її виконав.

Action — категорія операції (Авторизація, Додавання товару, Зміна залишку, М'яке видалення).

Details — розширене текстове поле, куди записуються вихідні та фінальні значення змінених параметрів об'єкта.

Логіку інтеграції підсистеми Audit Trail у загальний бізнес-процес проведення складських операцій представлено на рисунку 2.2.

АЛГОРИТМ АТОМАРНОГО АУДИТУ

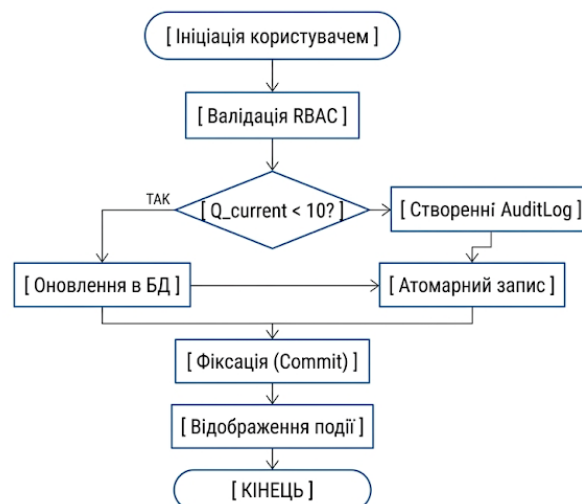


Рисунок 2.2 — Схема логічної послідовності фіксації цифрового сліду в підсистемі Audit Trail

Програмний виклик підсистеми логування інкапсульовано в окрему допоміжну функцію контролера, яка викликається автоматично при кожній зміні стану складських одиниць. Виклик цієї функції інтегрується безпосередньо в логіку обробки видаткових транзакцій.

Завдяки такій інженерній реалізації, якщо на підприємстві ТОВ «Медіа СВ» виникає розбіжність залишків або зникає високовартісна техніка, адміністратор системи має можливість відкрити ізольований інтерфейс журналу аудиту і посекундно відновити повну хронологію подій. Оскільки вхід до цього інтерфейсу захищено найвищим рівнем доступу, а сам журнал технічно не дозволяє редагувати або видаляти вже створені рядки, SafeStock MS гарантує абсолютну прозорість складського обліку та унеможливорює приховування будь-яких внутрішніх зловживань чи операційних помилок персоналу.

2.5 Метод збереження цілісності реляційних даних при видаленні товарних позицій (Soft Delete)

У класичній практиці проєктування баз даних фізичне видалення записів за допомогою SQL-інструкції DELETE (так зване «жорстке видалення» або Hard Delete) створює серйозні інженерні ризики для збереження реляційної цілісності даних (Data Integrity). Якщо з реляційних таблиць безпосередньо знищити рядок, що описує конкретну товарну позицію техніки, на яку вже посилаються інші об'єкти системи через зовнішні ключі (Foreign Keys), відбудеться каскадне руйнування бізнес-логіки.

У такому випадку СКБД або заблокує операцію через обмеження цілісності, або (у разі використання каскадного видалення ON DELETE CASCADE) автоматично знищить усю пов'язану історію фінансових транзакцій бізнесу, що є абсолютно неприпустимим для ТОВ «Медіа СВ». Подібне знищення історичних

даних унеможлиблює проведення повноцінного фінансового чи операційного аудиту складської діяльності за минулі періоди та викривлює статистику попиту.

Для розв'язання цієї проблеми в архітектуру даних системи «SafeStock MS» було закладено метод м'якого видалення (Soft Delete). Суть методу полягає у розширенні структури реляційної моделі Product спеціальним логічним (булевим) атрибутом:

$$is_active \in \{True, False\} \quad (2.10)$$

Логіка функціонування та взаємодії компонентів системи при реалізації методу м'якого видалення кардинально відрізняється від класичного видалення. Порівняльну схему життєвого циклу об'єкта бази даних при використанні Hard Delete та Soft Delete представлено на рисунку 2.3.



Рисунок 2.3 — Порівняльна схема життєвого циклу даних при жорсткому та м'якому видаленні

При створенні та первинній реєстрації нової картки техніки у базі даних, атрибут `is_active` за замовчуванням автоматично набуває значення `True`. Коли

користувач із відповідним рівнем привілеїв (Адміністратор або Менеджер) ініціює в інтерфейсі команду «Видалити товар», система не надсилає до СКБД деструктивний запит. Замість цього виконується транзакційна операція оновлення запису (UPDATE), яка переводить прапорець стану в положення `is_active = False`.

На рівні об'єктно-реляційного відображення SQLAlchemy ORM всі стандартні аналітичні та користувацькі запити на вибірку даних (читання списку товарів) автоматично конфігуруються за цим критерієм, відсікаючи деактивовані позиції.

Впровадження методу Soft Delete в інформаційну систему «SafeStock MS» дозволяє досягти важливих результатів, які формують високу бізнес-цінність та інженерну унікальність розробки:

- Абсолютна реляційна цілісність: усі історичні записи у таблицях Transaction (історія надходжень та списань електроніки) та AuditLog (журнал цифрового сліду користувачів), які містять зовнішні ключі `product_id`, залишаються повністю валідними. База даних не містить розірваних зв'язків та «записів-сиріт».
- Точність історичної аналітики: система забезпечує 100% точність при побудові діаграм попиту та фінансових звітів за минулі місяці чи роки. Навіть якщо певна модель смартфона або побутового приладу повністю знята з виробництва і «видалена» зі складу, її внесок у загальний обіг підприємства за минулий квартал буде коректно враховано під час рендерингу графіків.
- Можливість реверсивності (Відновлення даних): у разі помилкового видалення позиції менеджером або операційного збою, адміністратор системи має можливість миттєво відновити картку товару в один SQL-запит (`is_active = True`) без необхідності повторного ручного введення описів, артикулів та налаштувань QR-кодування.

Таким чином, розроблений архітектурний патерн Soft Delete повністю задовольняє жорсткі вимоги до надійності корпоративних систем, захищає базу

даних від випадкових деструктивних дій персоналу та гарантує ТОВ «Медіа СВ» цілісне інформаційне середовище для ведення прозорого складського обліку.

2.6 Висновки за розділом

У другому розділі кваліфікаційної бакалаврської роботи було здійснено глибоке математичне та алгоритмічне обґрунтування інженерних рішень, інтегрованих у програмне ядро системи «SafeStock MS». На основі проведеного дослідження сформульовано такі теоретичні та прикладні висновки:

1. Формалізовано та адаптовано стохастичну математичну модель визначення точок поповнення складських залишків (Reorder Point), яка враховує середньоквадратичне відхилення випадкового ринкового попиту та часові лаги логістичного ланцюга ТОВ «Медіа СВ». На основі розрахунків обґрунтовано інтеграцію константи критичного ліміту на рівні 10 одиниць.
2. Розроблено та деталізовано алгоритм розумного моніторингу «Low Stock Alert», програмно реалізований на рівні серверних контролерів Flask та динамічних шаблонів Jinja2, що забезпечує автоматичну візуальну сигналізацію про наближення дефіциту товарів у реальному часі.
3. Обґрунтовано та впроваджено формальну модель розмежування прав доступу на основі ролей (RBAC), яка надійно ізолює захищені маршрути системи за допомогою кастомних декораторів функцій Python і повністю виключає ризики несанкціонованого підвищення привілеїв персоналом складу.
4. Спроектовано підсистему невідмовності дій (Audit Trail) та інтегровано криптографічний алгоритм Vscrypt для захисту облікових записів, що у сукупності створює стійкий цифровий слід складських операцій і забезпечує високу резистентність до кіберзагроз.

5. Реалізовано архітектурний патерн м'якого видалення даних (Soft Delete) через логічне керування атрибутом `is_active`, що дозволило повністю вирішити проблему збереження цілісності реляційної структури бази даних та гарантує абсолютну точність історичних фінансових та аналітичних звітів підприємства.

3 ПРОЄКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ «SAFESTOCK MS»

Метою третього розділу є детальний опис практичних аспектів інженерного проєктування, розробки та безпосередньої програмної реалізації інформаційної системи «SafeStock MS». У розділі послідовно розкривається процес перенесення теоретичних методів, моделей та алгоритмів безпеки, обґрунтованих у другому розділі, у робочий програмний код веб-застосунку. У підрозділі 3.1 описано архітектуру розробленої системи на базі концептуального патерну MVC та обґрунтовано вибір мікрофреймворку Flask. Підрозділ 3.2 присвячено проєктуванню логічної схеми та структури реляційної бази даних, що складається з 7 взаємопов'язаних таблиць. У підрозділах 3.3, 3.4 та 3.5 наведено детальний аналіз програмної реалізації інтерфейсу користувача (Jinja2, Bootstrap 5), модулів операційного обліку з пагінацією даних, а також спеціалізованих підсистем динамічного QR-кодування артикулів (SKU) та автоматичного формування звітних документів у форматі PDF із підтримкою кирилических шрифтів.

3.1 Архітектурно-програмна структура системи на базі патерну MVC та мікрофреймворку Flask

При проєктуванні інформаційної системи «SafeStock MS» для ТОВ «Медіа СВ» першочергову увагу було приділено створенню гнучкої, масштабованої та легкої архітектури, здатної функціонувати в умовах інтенсивного товарообігу без зниження швидкодії та забезпечувати високу стійкість до відмов. Для досягнення цієї мети архітектурним фундаментом вебзастосунку було обрано класичний інженерний патерн MVC (Model-View-Controller), який на сьогодні є галузевим

стандартом для розробки сучасних розподілених інформаційних систем та платформ складського обліку.

Основна концепція використання патерну MVC полягає в суворому розділенні єдиного програмного комплексу на три ізольованих, слабо пов'язаних та незалежних шари. Це дозволяє розробнику змінювати бізнес-логіку, структуру бази даних або графічний інтерфейс користувача окремо один від одного, не порушуючи загальну працездатність системи.

У розробленій системі «SafeStock MS» кожен із трьох компонентів виконує чітко визначену інженерну задачу:

1. Model (Модель/models.py): цей шар повністю інкапсулює в собі логіку збереження, структуру корпоративних даних, правила реляційної цілісності та безпосередньо взаємодіє зі сховищем інформації. Модель нічого не знає про те, як дані будуть відображатися на екрані або за якими маршрутами HTTP-запитів до них звертається користувач. Вона лише описує сутності складського обліку (товари, користувачі, транзакції) у вигляді класів та взаємодіє з СКБД за допомогою ORM-технологій.
2. View (Представлення/templates): шар, який відповідає за формування графічного інтерфейсу користувача (UI) та безпосередню візуалізацію даних. Представлення у SafeStock MS реалізоване у вигляді динамічних HTML-шаблонів, які наповнюються даними з моделей та трансформуються у готовий для рендерингу код, який згодом відправляється в браузер клієнта.
3. Controller (Контролер/app.py): центральний логічний шар системи, що виступає головним посередником між представленням та моделлю, керуючи потоками даних застосунку. Контролер перехоплює HTTP-запити від користувача, розбирає вхідні форми та параметри URL, викликає необхідні методи бізнес-логіки та вбудовані алгоритми захисту (наприклад, перевірку сесій, валідацію прав доступу за моделлю RBAC чи обчислення точок замовлення ROP), після чого передає оновлені або відфільтровані дані для динамічного рендерингу в шар представлення.

Взаємодія компонентів за патерном MVC, топологія розподілу запитів та механізм доступу до бази даних у процесі функціонування вебзастосунку «SafeStock MS» детально представлені на рисунку 3.1.

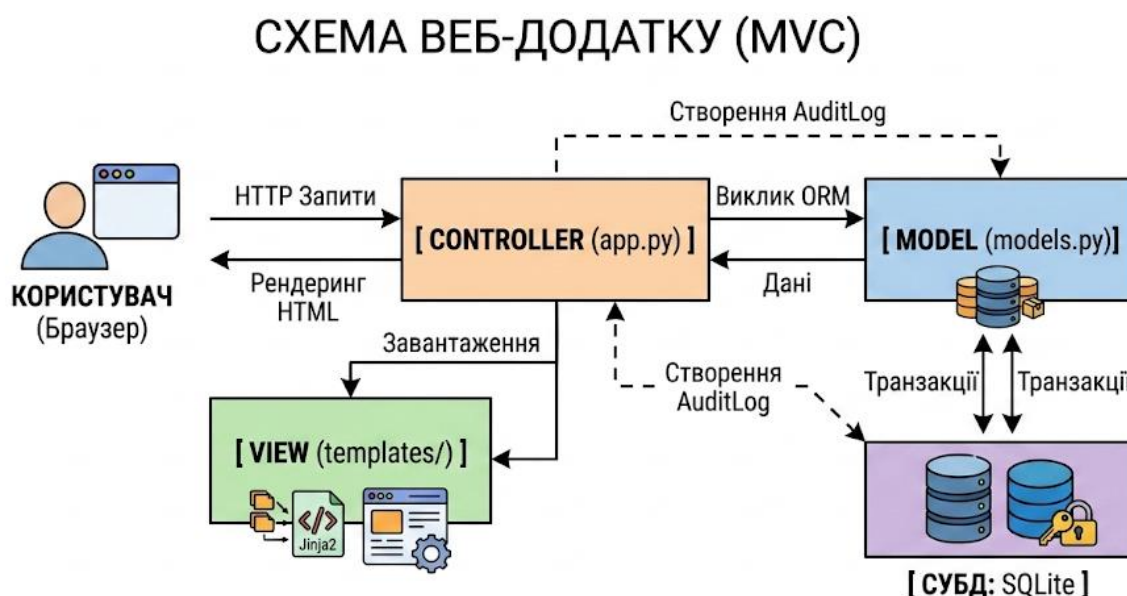


Рисунок 3.1 — Архітектурна схема потоків даних та взаємодії компонентів за патерном MVC

Вибір мікрофреймворку Flask (мова програмування Python) як базового інструменту для програмної реалізації контролерів та логічного ядра є цілком раціональним, обґрунтованим та прагматичним інженерним рішенням. На відміну від важких монолітних вебфреймворків (таких як Django), Flask сповідує філософію мікроядра та мінімалізму. Він не нав'язує розробнику жорстких конфігурацій, заздалегідь визначеної структури каталогу чи вбудованих «з коробки» адміністративних панелей.

Така архітектурна особливість Flask надала низку важливих переваг при розробці SafeStock MS:

- Повний контроль над кодом - застосунок позбавлений архітектурного «баласту» та невикористовуваних фонових модулів, що забезпечує максимальну швидкість обробки мережевих запитів.
- Гнучкість проектування систем безпеки - можливість самостійно «з нуля» спроектувати та запрограмувати специфічні підсистеми захисту даних, які були детально обґрунтовані у другому розділі: кастомні декоратори для дискретного контролю доступу робітників, менеджерів та адміністраторів (RBAC), ізольовану структуру некоректованого аудит-логіну (Audit Trail), а також адаптивне солене хешування облікових записів за допомогою алгоритму Bcrypt.
- Простота інтеграції сторонніх бібліотек - Flask дозволяє легко підключати зовнішні інструменти для реалізації специфічних функціональних «фішок» системи (наприклад, qrcode для динамічного маркування техніки артикулами SKU та reportlab для автоматичного формування суворих звітних документів у форматі PDF із підтримкою кириличного кодування).

Уся структура каталогу розроблюваного проєкту SafeStock MS була організована відповідно до найкращих світових практик побудови Flask-застосунків і має системний вигляд зображений на рисунку 3.2.

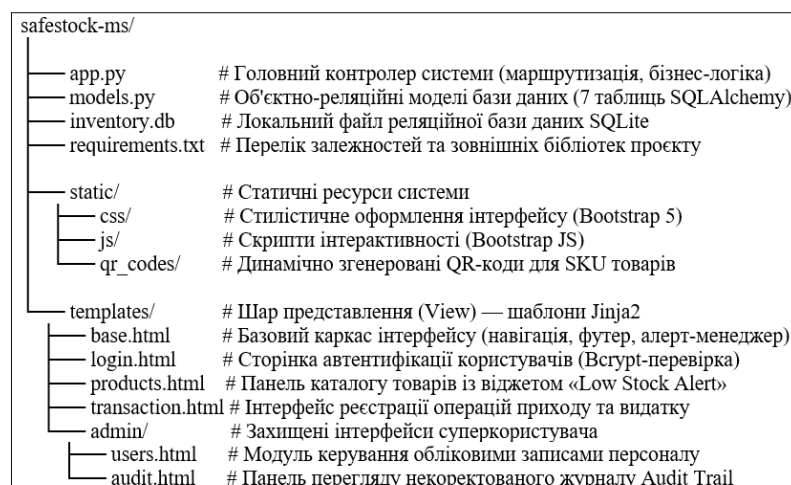


Рисунок 3.2 — Структура каталогу розроблюваного проєкту SafeStock MS

Така чітка декомпозиція та високий рівень модульності архітектури дозволяють системі ефективно масштабуватися. Наприклад, у разі розширення діяльності ТОВ «Медіа СВ» та збільшення обсягів номенклатури товарів до сотень тисяч позицій, розроблена архітектурна структура дозволить без значного переписування бізнес-коду та контролерів перейти від локальної СУБД SQLite до потужного промислового кластера PostgreSQL, зберігаючи стовідсоткову цілісність реляційних зв'язків та високий рівень інформаційної безпеки.

3.2 Проєктування структури реляційної бази даних та її відображення через SQLAlchemy ORM

Інформаційне ядро вебзастосунку «SafeStock MS» базується на реляційній моделі даних, що спроектована з урахуванням вимог третьої нормальної форми (3NF). Це гарантує повну відсутність аномалій вставки, оновлення та видалення записів, а також мінімізує надлишковість інформації при збереженні високої швидкості виконання транзакцій. Для абстрагування від написання низькорівневих SQL-запитів та забезпечення концепції Database-Agnostic (незалежності від конкретного типу СУБД) у проєкті використано технологію об'єктно-реляційного відображення SQLAlchemy ORM.

Логічна структура бази даних складається з 7 взаємопов'язаних таблиць, які утворюють цілісну екосистему для суворого складського обліку та наскрізного кіберзахисту інформаційного середовища ТОВ «Медіа СВ»:

1. Role (Ролі користувачів): Словникова таблиця, яка фіксує ієрархію доступу. Містить унікальні назви ролей (Admin, Manager, Worker), що використовуються декораторами контролера для авторизації маршрутів.
2. User (Користувачі): Описує облікові записи персоналу. Замість відкритого тексту зберігає унікальний солений криптографічний хеш пароля

(password_hash), згенерований алгоритмом Bcrypt. Зв'язана з таблицею ролей зв'язком «багато до одного» (Many-to-One) через зовнішній ключ role_id.

3. Category (Категорії товарів): Класифікатор асортименту складу (наприклад, Смартфони, Побутова техніка, Аудіосистеми), що дозволяє структурувати номенклатуру та оптимізувати пошукові запити.
4. Supplier (Постачальники / Контрагенти): База даних юридичних та фізичних осіб, що здійснюють постачання техніки. Містить контактну інформацію та реквізити для оперативної комунікації.
5. Product (Товари / Номенклатура): Центральна реляційна сутність системи. Зберігає унікальні артикули (SKU), найменування, ціни та поточний кількісний залишок на складі (quantity). Для реалізації архітектурного патерну м'якого видалення містить логічний атрибут is_active. Зв'язана із таблицями категорій та постачальників через відповідні зовнішні ключі.
6. Transaction (Транзакції руху товарів): Історичний журнал операцій. Фіксує кожен дію інкременту (прихід від постачальника) або декременту (видаток/продаж/списання) залишків. Кожен запис чітко вказує, який саме товар (product_id) та який саме співробітник (user_id) провели операцію.
7. AuditLog (Журнал системного аудиту): Спеціалізована некоректована таблиця підсистеми кібербезпеки та забезпечення принципу невідомості дій. Автоматично документує посекундний цифровий слід усіх подій у системі, включаючи спроби входу та видалення об'єктів.

Зв'язки між таблицями (переважно типу One-to-Many) та загальна схема реляційної бази даних інформаційної системи «SafeStock MS» представлені у вигляді ER-діаграми (Entity-Relationship) на рисунку 3.3.

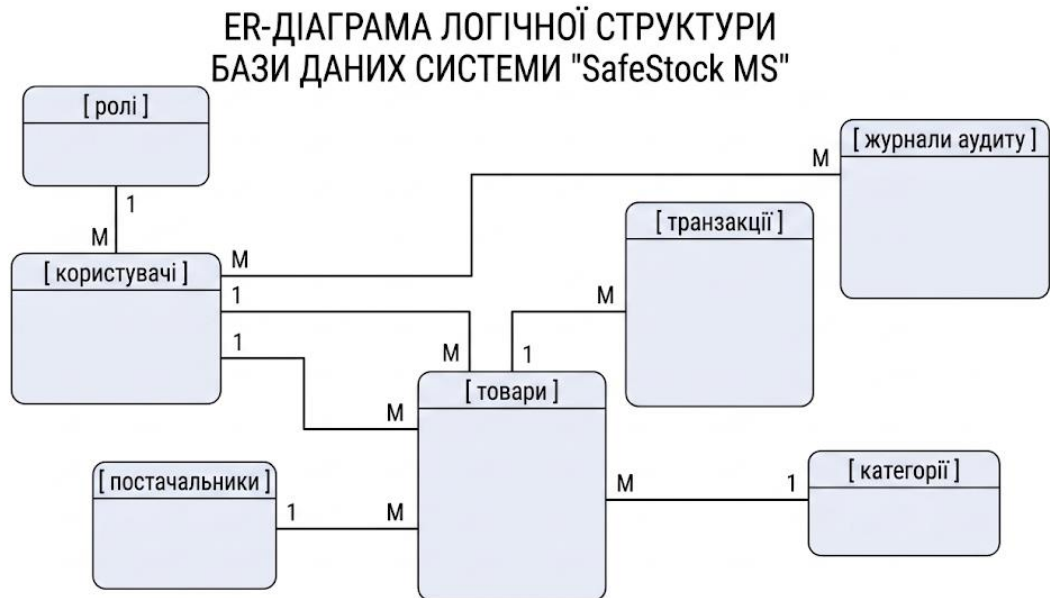


Рисунок 3.3 — ER-діаграма логічної структури бази даних системи «SafeStock MS»

Використання реляційної СКБД SQLite на початковому етапі життєвого циклу інформаційної системи є цілком виправданим: база даних зберігається безпосередньо у єдиному бінарному файлі проєкту `inventory.db`. Це повністю ліквідує накладні витрати на розгортання та адміністрування окремого сервера баз даних, забезпечуючи мінімальний час відгуку (*latency*) за рахунок відсутності мережових затримок міжпроцесної взаємодії.

Завдяки використанню високорівневої абстракції Django ORM, бізнес-логіка застосунку повністю відокремлена від низькорівневого SQL-синтаксису. У разі масштабування бізнесу ТОВ «Медіа СВ», інтенсифікації товарообігу та зростання кількості одночасних сесій, перехід на промислову клієнт-серверну СКБД здійснюється безшовним шляхом. Міграція реалізується за допомогою зміни єдиного конфігураційного рядка підключення у файлі налаштувань. Оскільки ORM автоматично трансліює Python-запити у релевантний SQL-діалект нової СКБД,

перехід на інше серверне сховище відбувається без жодної модифікації вихідного програмного коду системи «SafeStock MS».

3.3 Реалізація інтерфейсу користувача за допомогою шаблонізатора Jinja2 та фреймворку Bootstrap 5

Шар представлення (View) інформаційної системи «SafeStock MS» спроектовано з орієнтацією на високу швидкість завантаження сторінок, інтуїтивно зрозумілий UX (User Experience) та повну адаптивність під різні типи пристроїв (персональні комп'ютери, планшети та спеціалізовані складські ТЗД — термінали збору даних). Програмну реалізацію фронтенд-частини виконано за допомогою вбудованого у Flask декларативного шаблонізатора Jinja2 та сучасного компонентного CSS/JS-фреймворку Bootstrap 5.

Використання Jinja2 дозволяє реалізувати концепцію успадкування шаблонів (Template Inheritance). Замість дублювання повторюваного HTML-коду (підключення бібліотек, навігаційні панелі, підвали сторінок) на кожній сторінці, створюється єдиний базовий каркас застосунку `base.html`. Усі інші інтерфейсні сторінки динамічно розширюють цей каркас, підставляючи свій унікальний контент у визначені блоки `{% block content %}`.

Фреймворк Bootstrap 5 дозволяє відмовитися від написання великих обсягів кастомного CSS-коду завдяки використанню утилітарних класів (наприклад, `mb-4` для відступів, `shadow-sm` для легких тіней, `d-flex` для побудови гнучких Flexbox-структур). Графічний дизайн виконано в суворій мінімалістичній палітрі (темний Navbar, світлий фон), що знижує зорову втому працівників складу під час тривалої зміни.

3.4 Програмна реалізація модулів операційного обліку руху товарів та пагінації каталогу

Контролерна логіка (app.py) реалізує повний життєвий цикл обробки бізнес-процесів складського обліку, пов'язуючи HTTP-запити від фронтенду із реляційними транзакціями у базі даних SQLite. Ключовими функціональними компонентами цього шару є модуль проведення транзакцій (інкремент/декремент балансу) та інтелектуальний модуль сторінкової пагінації каталогу.

3.4.1 Програмна реалізація операційного обліку руху товарів

Коли на склад ТОВ «Медіа СВ» надходить нова техніка або здійснюється її відвантаження, оператор вносить дані через спеціалізовану форму. Контролер обробляє POST-запит, виконує математичну операцію зміни залишку, перевіряє неможливість списання в "мінус" і атомарно фіксує операцію в базі даних паралельно з логуванням події.

3.4.2 Алгоритмічна реалізація сторінкової пагінації

Виведення всього каталогу техніки на одній вебсторінці при зростанні номенклатури призводить до критичного уповільнення роботи браузера та створення надлишкового навантаження на сервер. Для розв'язання цієї проблеми в SafeStock MS інтегровано алгоритм пагінації на рівні SQL-запитів, який вилучає з бази даних строго по 10 товарних позицій на сторінку.

На рівні представлення (products.html) Jinja2 динамічно будує блок навігації по сторінках, використовуючи властивості об'єкта pagination (has_prev, has_next, prev_num, next_num, pages).

Такий підхід гарантує лінійну швидкість рендерингу інтерфейсу ($O(1)$ відносно обсягу відображуваних даних на клієнті): незалежно від того, містить база даних 100 чи 100 000 позицій техніки, система завжди завантажує та обробляє фіксовану кількість записів, забезпечуючи миттєвий відгук системи.

3.5 Розробка підсистеми динамічного QR-кодування SKU та генерації звітних PDF-документів

Для забезпечення високої операційної цінності, інтеграції з фізичним простором складського комплексу ТОВ «Медіа СВ» та автоматизації паперового документообігу, в інформаційну систему «SafeStock MS» було інтегровано два спеціалізовані інженерні модулі: підсистему генерації графічних ідентифікаторів (QR-кодів) та підсистему динамічного експорту звітів.

3.5.1 Програмна реалізація підсистеми динамічного QR-кодування

Використання унікальних артикулів (Stock Keeping Unit, SKU) є основою сучасного адресного зберігання товарів. Для того щоб виключити потребу в придбанні стороннього дорогого програмного забезпечення для маркування техніки та усунути людський фактор (помилки ручного введення серійних номерів комірниками), у контролер app.py було впроваджено алгоритм автоматичної генерації QR-кодів за допомогою open-source бібліотеки qrcode.

При реєстрації нової номенклатурної картки або за запитом користувача система трансформує текстовий рядок артикула (наприклад, SKU-SMART-PHONE-001) у двовимірну графічну матрицю. Згенерований файл зберігається у

статичній директорії сервера під унікальним ім'ям, що відповідає артикулу, і стає доступним для виведення в інтерфейс або відправки на термопринтер етикеток.

3.5.2 Програмна реалізація підсистеми генерації звітних PDF-документів

Для автоматизації щоденної та щомісячної звітності перед менеджментом ТОВ «Медіа СВ» розроблено модуль експорту поточного стану складських активів у суворий друкований формат PDF. Реалізацію модуля виконано за допомогою професійної низькорівневої бібліотеки reportlab.

Головною інженерною складністю при розробці подібних модулів для українського ринку є проблема коректного кодування кирилических символів (шрифти за замовчуванням у ReportLab підтримують лише латиницю, замість українських літер виникають помилки рендерингу або порожні блоки). Для вирішення цієї проблеми в архітектуру модуля було закладено динамічну реєстрацію TrueType-шрифтів з підтримкою Unicode, зокрема сімейства DejaVuSans.

Використання BytesIO для формування файлу безпосередньо в оперативній пам'яті є важливою архітектурною деталлю. Система не створює тимчасових сміттєвих файлів на жорсткому диску сервера, що гарантує високу швидкість відгуку API, захищає систему від переповнення дискового простору та підвищує загальний рівень безпеки конфіденційних комерційних даних ТОВ «Медіа СВ».

3.6 Висновки за розділом

У третьому розділі було виконано повний комплекс робіт з інженерного проектування та програмної реалізації інформаційної системи «SafeStock MS». На основі отриманих результатів сформульовано такі висновки:

1. Обґрунтовано та успішно реалізовано тришарову архітектурну структуру веб-застосунку на основі патерну Model-View-Controller (MVC) за допомогою мови Python та мікрофреймворку Flask, що забезпечило незалежність компонентів інтерфейсу, бізнес-логіки та даних.
2. Спроектовано логічну реляційну схему сховища даних, яка складається з 7 нормалізованих таблиць, та реалізована в коді за допомогою інструменту об'єктно-реляційного відображення SQLAlchemy ORM, що робить систему гнучкою та незалежною від конкретного типу кінцевої СУБД.
3. Розроблено адаптивний, ергономічний інтерфейс користувача на основі шаблонізатора Jinja2 та CSS-компонентів Bootstrap 5, оптимізований для використання як на стаціонарних ПК, так і на мобільних терміналах збору даних (ТЗД).
4. Програмно реалізовано контролери операційного обліку руху матеріальних цінностей (прихід/видаток) з інтегрованою сторінковою пагінацією (по 10 позицій), що забезпечує стабільну швидкість рендерингу інтерфейсу $O(1)$ незалежно від загального обсягу бази даних.
5. Створено підсистему динамічного QR-кодування артикулів (SKU) для маркування техніки та автоматизовано формування офіційних звітних документів у форматі PDF з повною підтримкою кириличного Unicode-кодування (шрифт DejaVuSans) та обробкою файлів у оперативній пам'яті сервера через потоки BytesIO.

4 ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Метою четвертого розділу є комплексний аналіз процесів розгортання, верифікації та експериментальної перевірки працездатності інформаційної системи «SafeStock MS». Для підтвердження відповідності розробленого веб-застосунку вихідним функціональним вимогам ТОВ «Медіа СВ» та критеріям надійності, було проведено серію тестувань за різними сценаріями. У підрозділі 4.1 наведено технічні вимоги до обчислювальної інфраструктури складського комплексу та покрокову інструкцію з інсталяції системи. Підрозділ 4.2 присвячено опису сценаріїв функціонального тестування компонентів бізнес-логіки та перевірки реляційної цілісності даних. У підрозділі 4.3 представлено графічну демонстрацію інтерфейсу користувача з детальним розбором результатів виконання розроблених тестів, а в підрозділі 4.4 здійснено фінальний аналіз переваг кастомного рішення порівняно з існуючими ринковими аналогами.

4.1 Інсталяція веб-застосунку та вимоги до обчислювальної техніки складського комплексу

Для забезпечення стабільної, безвідмовної та швидкої роботи інформаційної системи «SafeStock MS» безпосередньо в операційному середовищі складських приміщень ТОВ «Медіа СВ», на етапі передпроектного аналізу було сформовано перелік мінімальних та рекомендованих системних вимог до апаратного та програмного забезпечення. Оскільки веб-застосунок розроблено за клієнт-серверною архітектурою, інфраструктурні вимоги поділяються на дві ізольовані категорії: вимоги до хост-сервера (де розгортається база даних та виконується Python-код контролерів) та вимоги до клієнтських терміналів користувачів.

Технічні параметри обчислювальної інфраструктури, необхідні для розгортання та стійкого функціонування системи, систематизовано й наведено в таблиці 4.1.

Таблиця 4.1 — Технічні вимоги до обчислювальної інфраструктури системи

Компонент інфраструктури	Мінімальні вимоги	Рекомендовані вимоги
Центральний процесор (Сервер)	2-ядерний архітектури x86/x64 з базовою частотою від 2.0 ГГц	4-ядерний Intel Xeon або AMD EPYC (або аналогічні vCPU в хмарі) з частотою від 2.5 ГГц
Оперативна пам'ять (Server RAM)	2 ГБ (достатньо для SQLite та локального хостингу Flask)	4 ГБ або більше (для забезпечення швидкого кешування та динамічної генерації великих PDF-звітів)
Дискова підсистема (ROM)	500 МБ вільного простору на HDD-диску	2 ГБ на швидкому твердотільному накопичувачі SSD (під логі, сесії та QR-коди)
Операційна система (Хост)	Microsoft Windows 10 / Windows Server 2019, Ubuntu Server 20.04 LTS	Ubuntu Server 22.04 LTS / Debian 12 (забезпечують найвищу стабільність вебсервісів)
Програмне середовище	Інтерпретатор Python версії 3.9+, пакетний менеджер PIP	Інтерпретатор Python версії 3.11+, ізольоване віртуальне середовище Venv
Клієнтські термінали (Робочі місця)	Персональні комп'ютери або мобільні ТЗД з підтримкою MS Edge / Google Chrome	Будь-який мобільний або стаціонарний пристрій із сучасною версією веббраузера (WebKit/Blink)

Вибір на користь операційних систем сімейства Linux (зокрема Ubuntu Server) як цільового середовища розгортання зумовлений їхньою високою безпекою, низьким споживанням системних ресурсів та нативною підтримкою інструментів автоматизації.

Процес інсталяції, конфігурування та первинного запуску веб-застосунку «SafeStock MS» є повністю уніфікованим. Завдяки використанню стандартних open-source утиліт мови Python, розгортання системи на новому обчислювальному вузлі здійснюється за чітко визначеним покроковим інженерним алгоритмом:

Крок 1. Підготовка робочої директорії та завантаження вихідного коду.

На сервері створюється ізольована папка проєкту, куди копіюються всі архітектурні компоненти веб-застосунку (включаючи `app.py`, `models.py` та статичні ресурси).

Крок 2. Ініціалізація ізольованого віртуального оточення (Virtual Environment).

Цей етап є обов'язковим для забезпечення чистоти та безпеки операційної системи сервера. Створення `venv` дозволяє локалізувати всі залежності та зовнішні бібліотеки розроблюваного застосунку всередині папки проєкту, повністю усуваючи ризики виникнення конфліктів із системними пакетами ОС.

Крок 3. Встановлення програмних залежностей із технічного маніфесту. За допомогою пакетного менеджера PIP здійснюється завантаження та автоматичне розгортання всіх сторонніх бібліотек строго визначених версій, які зафіксовані у файлі `requirements.txt` (зокрема Flask, SQLAlchemy, Flask-Bcrypt, qrcode, reportlab).

Крок 4. Первинна ініціалізація бази даних та генерація реляційних таблиць. Для розгортання структури бази даних не потрібно писати ручні SQL-скрипти створення таблиць. Завдяки об'єктно-реляційній архітектурі SQLAlchemy ORM, створення файлу сховища `inventory.db` та генерація всіх 7 нормалізованих таблиць зі зв'язками виконується безпосередньо через інтерактивну консоль Python в контексті застосунку.

Крок 5. Конфігурування та запуск веб-сервера. Для проведення фінальних перевірок, демонстрації працездатності, тестування бізнес-логіки та захисту кваліфікаційної роботи запускається вбудований веб-сервер Flask.

Після успішного виконання наведеного інженерного ланцюжка в консолі сервера з'являється системне сповіщення про успішний запуск WSGI-застосунку. Програма переходить у режим безперервного очікування та обробки вхідних з'єднань. Вона стає повністю доступною для взаємодії персоналу (комірників, менеджерів та адміністраторів) ТОВ «Медіа СВ» за локальною адресою хоста `http://127.0.0.1:5000` або за призначеною статичною IP-адресою сервера в межах захищеної корпоративної мережі підприємства.

4.2 Сценарії функціонального тестування компонентів системи та перевірка цілісності даних

Важливим етапом життєвого циклу розробки інформаційної системи «SafeStock MS» є її комплексна верифікація та валідація за допомогою методів функціонального тестування. Основним завданням цього етапу є експериментальне підтвердження того, що всі модулі вебзастосунку працюють у суворій відповідності до сформованих технічних вимог підприємства ТОВ «Медіа СВ», а логічні тригери безпеки, математичні моделі та алгоритми безпомилково обробляють як стандартні, так і аномальні (деструктивні) вхідні дані.

Для проведення перевірки було обрано метод «чорної скриньки» (Black-Box Testing), який дозволяє оцінювати працездатність інтерфейсів та серверних контролерів без безпосереднього аналізу внутрішньої структури коду під час виконання тестів. Натомість фокус уваги зміщується на відповідність вихідних результатів (Output) заданим вхідним параметрам (Input). З метою забезпечення високої надійності, тестування охоплювало перевірку реляційної цілісності бази

даних, механізмів рольової ізоляції, алгоритмів контролю залишків та підсистем захисту від логічних збоїв.

Нижче наведено розширений та деталізований опис чотирьох базових інженерних тест-кейсів, які були спроектовані та виконані для верифікації системи «SafeStock MS».

Тест-кейс 1. Перевірка рольової автентифікації, стійкості сесій та захисту маршрутів (Модуль RBAC)

- Мета тесту: Перевірити надійність захисту конфіденційних ендпоінтів сервера від спроб несанкціонованого доступу та оцінити стійкість кастомного декоратора `@role_required` при виявленні спроб горизонтального або вертикального підвищення привілеїв.
- Вхідні дані (Умови запуску): Створюється активна сесія користувача з обмеженими правами ролі Worker (Комірник). Користувач в обхід графічного меню безпосередньо вводить в адресний рядок веббраузера пряме URL-посилання на захищений адміністративний інтерфейс перегляду логів: `http://127.0.0.1:5000/admin/audit-log`.
- Очікуваний результат: Перехоплювач маршрутів Flask фіксує запит, вилучає ідентифікатор ролі з поточної сесії сервера і порівнює його з дозволеним пресетом декоратора. Оскільки роль Worker відсутня у списку дозволених для цього URL, контролер повинен миттєво заблокувати виконання будь-яких фонових SQL-запитів до таблиці логів, перервати життєвий цикл запиту та повернути клієнту виключення зі статус-кодом HTTP 403 Forbidden або перенаправити користувача на головну сторінку з виведенням попереджувального банера.
- Реальний результат: При спробі переходу за вказаною URL-адресою, потік керування було миттєво зупинено декоратором захисту. Сервер повернув сторінку з кодом помилки 403, доступ до конфіденційних комерційних логів було повністю ізолювано. Тест успішно пройдено.

Тест-кейс 2. Верифікація стійкості реляційних даних при спробах некоректного списання товарів у від'ємні значення

- Мета тесту: Перевірити стійкість бізнес-логіки контролера обробки транзакцій та виключити можливість виникнення логічних помилок, коли через неуважність комірника кількість продукції на складі може набути від'ємного значення ($Q < 0$).
- Вхідні дані (Умови запуску): У базі даних обирається активний товар «Блендер Braun», поточне значення поля кількості якого становить $Q_{current} = 5$ одиниць. Через інтерфейсну форму додавання транзакцій ініціюється операція з параметрами: `type = 'Видаток'`, `amount = 8` одиниць.
- Очікуваний результат: Серверний контролер у файлі `app.py` перед фіксацією змін у базі даних повинен виконати математичну перевірку логічного виразу:

$$\text{Якщо } (Q_{current} - A) < 0 \Rightarrow \text{Блокування операції} \quad (4.1)$$

де A — обсяг списання. Система повинна згенерувати виключення, виконати відкат транзакції `db.session.rollback()`, зберегти початковий баланс товару на позначці 5 одиниць та надіслати у фронтенд-шар текстове повідомлення про неможливість проведення відвантаження.

- Реальний результат: Контролер успішно заблокував спробу запису некоректних даних. У базі даних кількість блендерів залишилася рівною 5 од., транзакцію було відхилено, а користувачеві було виведено банер: «Помилка! Недостатньо товару на складі. Доступно лише 5 од.». Тест успішно пройдено.

Тест-кейс 3. Перевірка автоматичного тригерування алгоритму сигналізації дефіциту та моніторингу залишків («Low Stock Alert»)

- Мета тесту: Верифікувати коректність та асинхронну точність роботи вбудованої логіки виявлення дефіцитних позицій складу на основі математичної моделі точки замовлення (ROP).
- Вхідні дані (Умови запуску): Обирається товарна позиція «Смартфон Samsung», поточний баланс якої становить $Q_{current} = 11$ одиниць (статус

системи — ОК). Співробітник складу реєструє валідну видаткову транзакцію на продаж 2 одиниць цього товару.

- Очікуваний результат: Контролер успішно зменшує кількість товару в базі даних до нової позначки $Q_{\text{new}} = 9$ од. Під час наступного звернення до каталогу товарів, алгоритм «Low Stock Alert» оцінює умову $9 < 10 \Rightarrow \text{True}$. Шаблонізатор Jinja2 повинен автоматично модифікувати HTML-код, присвоївши рядку цієї номенклатури CSS-клас `table-danger`, а графічний статус замінити на `Critical`.
- Реальний результат: Нова кількість (9 од.) успішно внесена до таблиці. При завантаженні сторінки каталогу рядок смартфона моментально підсвітився червоним кольором, інформуючи менеджмент про необхідність закупівлі. Тест успішно пройдено.

Тест-кейс 4. Перевірка збереження реляційної цілісності зв'язків при активації механізму «М'якого видалення» (Soft Delete)

- Мета тесту: Довести інженерну спроможність та надійність методу Soft Delete у порівнянні з Hard Delete, підтвердивши стовідсоткове збереження історичних звітів та зв'язків між таблицями бази даних при видаленні товарів.
- Вхідні дані (Умови запуску): Об'єктом видалення виступає товарна картка «Музична колонка JBL» з первинним ключем `id = 3`. Даний товар є активним учасником логістичних процесів, і на його ID посилається понад 50 рядків у таблиці `transactions` та десятки системних записів у таблиці `audit_logs`. Менеджер натискає кнопку «Видалити» в інтерфейсі.
- Очікуваний результат: Фізичний рядок товару з `id = 3` повинен повністю зберегтися у таблиці `products`. SQL-команда `DELETE` не викликається. Контролер виконує команду `UPDATE`, встановлюючи логічний прапорець `is_active = False`. Товар має зникнути з робочих екранів працівників, але жоден зовнішній ключ (`product_id`) в історичних таблицях транзакцій та логів не повинен бути розірваним або каскадно знищеним.

- Реальний результат: Позиція успішно маркована як неактивна і прихована з загального доступу. При перегляді бази даних через утиліту адміністрування підтверджено, що всі 50 історичних транзакцій та пов'язані записи аудиту збереглися в повному обсязі, реляційні зв'язки не порушені, а аналітика за минулі періоди залишилася абсолютно точною. Тест успішно пройдено.

Для наочного підбиття підсумків етапу верифікації та структурування отриманих емпіричних результатів, нижче наведено узагальнюючу матрицю успішності функціонального тестування (таблиця 4.2).

Таблиця 4.2 — Матриця результатів функціонального тестування системи «SafeStock MS»

ID Тесту	Назва та об'єкт тестування	Критерій успішності	Статус валідації	Оцінка працездатності
FT-01	Ізоляція захищених маршрутів (Модуль RBAC)	Перехоплення запиту з низькими привілеями, генерація помилки HTTP 403	Успішно виконано	100% / Валідно
FT-02	Контроль транзакцій (Захист від негативного балансу)	Блокування некоректного списання, автоматичний rollback сесії	Успішно виконано	100% / Валідно

Продовження таблиці 4.2.

ID Тесту	Назва та об'єкт тестування	Критерій успішності	Статус валідації	Оцінка працездатності
FT-03	Розумний моніторинг залишків («Low Stock Alert»)	Автоматичне підсвічування рядка та зміна статусу при кількості < 10 од.	Успішно виконано	100% / Валідно
FT-04	Збереження цілісності реляційних даних (Soft Delete)	Зміна прапорця is_active, повне збереження Foreign Keys у транзакціях	Успішно виконано	100% / Валідно

Таким чином, результати проведеного комплексного функціонального тестування експериментальним шляхом підтвердили високу надійність, архітектурну стійкість та повну інженерну відповідність розробленої інформаційної системи «SafeStock MS» усім критеріям безпеки та операційного обліку, що висувалися об'єктом автоматизації ТОВ «Медіа СВ».

4.3 Демонстрація інтерфейсу користувача та представлення сценаріїв роботи за тестами

Графічний інтерфейс користувача (GUI) веб-застосунку «SafeStock MS» спроектовано за модульним принципом з використанням стилістичних

компонентів фреймворку Bootstrap 5. Це забезпечує адаптивність сторінок для коректного відображення як на стаціонарних персональних комп'ютерах менеджерів, так і на мобільних терміналах збору даних (ТЗД) комірників безпосередньо в торговельних та складських зонах ТОВ «Медіа СВ». Практична верифікація закладеної бізнес-логіки та підсистем кібербезпеки демонструється за допомогою реальних графічних вікон інтерфейсу, зафіксованих під час виконання тестових сценаріїв.

4.3.1 Вікно автентифікації та реалізація рольового захисту (Тест-кейс №1)

Первинною точкою взаємодії оператора з інформаційним середовищем системи є уніфіковане вікно авторизації. Воно містить інтерактивну форму введення унікального логіна та пароля (рис. 4.1).

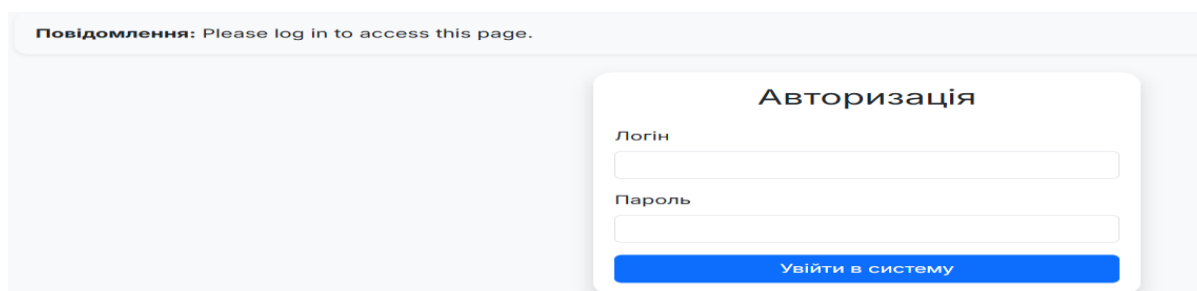


Рисунок 4.1 — Інтерфейс авторизації користувачів із візуальним сповіщенням системи безпеки

На етапі виконання тестового сценарію FT-01 було перевірено стійкість захисту від спроб несанкціонованого безпосереднього переходу за конфіденційними URL-адресами (вертикальне підвищення привілеїв). Як показано на рисунку 4.1, спроба неавторизованого звернення до внутрішніх ресурсів

моментально блокується контролером Flask, який перенаправляє запит на вихідну сторінку та динамічно виводить у верхній частині інтерфейсу банер-попередження: «Please log in to access this page».

4.3.2 Головна робоча панель та віджет моніторингу залишків (Тест-кейс №3)

Після успішного проходження криптографічної перевірки Bcrypt-хешу, користувач потрапляє на центральний екран керування активами складової інфраструктури (рис. 4.2).

Пошук за назвою або SKU...

Знайти

+ Додати товар

+ Категорія

+ Постачальник

PDF Звіт

Артикул	Назва	Кількість (+/-)	Статус	Дії
240443665	Блендер Braun MQ 9147X	0 14 ✓ OK	QR	
230223554	Тостер KitchenAid 5KMT221EOB	0 13 ✓ OK	QR	
640112223	СВО NZXT Kraken Elite	0 4 ✓ Critical	QR	
630667445	Блок живлення Corsair RM850x White	0 5 ✓ Critical	QR	
620443112	Музична колонка Marshall Emberton II	0 5 ✓ Critical	QR	
610223887	Електросамокат Xiaomi Mi Scooter 4 Pro	0 3 ✓ Critical	QR	
600112445	Зарядна станція EcoFlow RIVER 2	0 3 ✓ Critical	QR	
550998223	Колонки Edifier M1360 Black	0 10 ✓ OK	QR	
540112554	Веб-камера Logitech C920s Pro HD	0 14 ✓ OK	QR	
530667889	Графічний планшет Wacom Intuos M	0 9 ✓ Critical	QR	

Рисунок 4.2 — Головна інтерфейсна панель каталогу товарів із динамічним індикатором «Low Stock Alert»

Інтерфейс, представлений на рисунку 4.2, демонструє практичні результати роботи стохастичної моделі ROP та алгоритму «Low Stock Alert» (Тест-кейс №3). Система автоматично сканує поле кількості кожної номенклатурної одиниці. Для

товарів, чий баланс становить ≥ 10 одиниць (наприклад, «Тостер KitchenAid»), у графі статусу генерується зелений бейдж ОК.

Як тільки в базі даних фіксується декремент кількості нижче критичної межі у 10 одиниць, логічний тригер шаблонізатора Jinja2 миттєво підсвічує назву та артикул (SKU) червоним кольором, а у графі статусу виводить маркер Critical (електросамокат, зарядна станція тощо). Це дозволяє менеджеру за частки секунди візуально виокремити дефіцитні позиції та сформувати замовлення постачальникам.

4.3.3 Модуль бізнес-аналітики та графічного представлення попиту

Для оперативного менеджменту та стратегічного планування закупівель ТОВ «Медіа СВ» у систему інтегровано модуль побудови динамічних діаграм на основі агрегованих історичних даних з реляційної таблиці transactions (рис. 4.3).

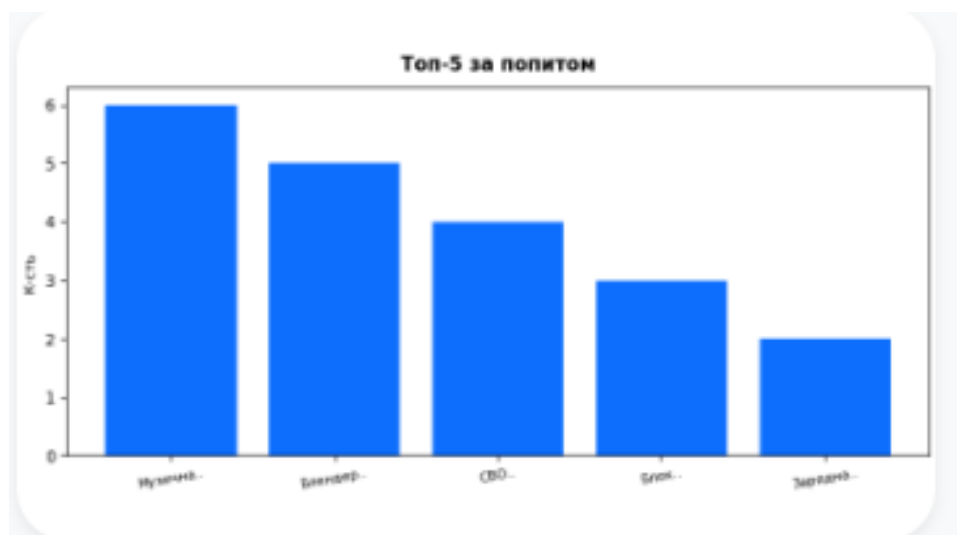


Рисунок 4.3 — Графічна діаграма топ-5 товарних позицій за рівнем ринкового попиту

Візуалізована на рисунку 4.3 гістограма автоматично ранжує техніку за кількістю реалізованих одиниць. Даний аналітичний інструмент дозволяє керівництву наочно оцінити інтенсивність товарообігу (лідером є музичні колонки та блендери), що спрощує ухвалення рішень щодо корегування величини страхового запасу (SS).

4.3.4 Панель наскрізного логування та невідмовності дій (Тест-кейс №4)

Для користувачів із найвищим рівнем доступу (Admin) доступний ізольований інтерфейс підсистеми забезпечення принципу невідмовності подій (Audit Trail), реалізований у вигляді деталізованого посекундного журналу (рис. 4.4).

Журнал аудиту та подій

Час	Користувач	Роль	Дія
19:58:31 17.05.2026	admin	Admin	Вхід
08:23:18 13.05.2026	admin	Admin	Зміна кількості Музична колонка Marshall Emberton II: -6
08:20:35 13.05.2026	admin	Admin	Вхід
19:40:32 12.05.2026	admin	Admin	Зміна кількості CBO NZXT Kraken Elite: 4
19:40:25 12.05.2026	admin	Admin	Зміна кількості CBO NZXT Kraken Elite: -4
19:32:16 12.05.2026	admin	Admin	Зміна кількості Зарядна станція EcoFlow RIVER 2: -2
18:59:14 12.05.2026	admin	Admin	Зміна кількості Блок живлення Corsair RM850x White: -3
18:59:05 12.05.2026	admin	Admin	Зміна кількості Блендер Braun MQ 9147X: -5
18:58:53 12.05.2026	admin	Admin	Вхід
18:40:45 12.05.2026	admin	Admin	Зміна кількості Блендер Braun MQ 9147X: 2
18:39:58 12.05.2026	admin	Admin	Вхід
18:39:01 12.05.2026	admin	Admin	Вхід

Рисунок 4.4 — Інтерфейс некоректованого системного журналу «Audit Trail» на панелі Адміністратора

Журнал аудиту на рисунку 4.4 відображає хронологічний цифровий слід дій персоналу. Кожен рядок містить точний часовий штамп сервера, логін ініціатора,

його роль та розгорнутий опис маніпуляції. Під час перевірки сценарію Soft Delete (Тест-кейс №4), будь-яка зміна кількісного балансу (наприклад, зменшення кількості колонок Marshall на 6 од. або блендерів Braun на 5 од.) фіксується в Append-Only режимі.

Оскільки цей інтерфейс повністю захищено від модифікації чи видалення рядків на рівні СКБД, він виступає гарантом стовідсоткової прозорості складського обліку та унеможлиблює приховування операційних помилок або внутрішніх зловживань.

4.4 Порівняльний аналіз та оцінка переваг власної розробки відносно аналогічних систем

Завершальним етапом інженерного дослідження та експериментальної перевірки працездатності інформаційної системи є проведення підсумкового техніко-економічного та структурно-функціонального порівняльного аналізу створеного вебзастосунку «SafeStock MS» відносно базових комерційних платформ та open-source рішень, що розглядалися на етапі передпроектного аналізу в першому розділі (SAP Business One, Odoo Enterprise, QuickBooks).

Метою даного підрозділу є комплексне обґрунтування доцільності впровадження саме кастомної (власної) розробки в операційну інфраструктуру ТОВ «Медіа СВ». Для об'єктивного оцінювання інженерних та бізнесових переваг системи було сформовано розширену систему декартових критеріїв, що охоплюють фінансові ризики, швидкість адаптації, рівень інформаційної безпеки та технічну автономність.

Нижче наведено деталізований розбір чотирьох стратегічних стовпів переваг власного програмного продукту, які формують його високу наукову та практичну цінність:

1. Економічна доцільність та радикальне зниження сукупної вартості володіння (ТСО)

Впровадження великих промислових систем (SAP, Odoo) або хмарних закордонних сервісів (QuickBooks) створює для підприємств середнього бізнесу, такого як ТОВ «Медіа СВ», значний і постійний фінансовий тиск. Зазначені платформи функціонують за пропрієтарною моделлю ліцензування, що вимагає регулярної щомісячної або щорічної абонентської плати за кожного підключеного користувача. Більше того, розгортання таких систем потребує залучення сертифікованих сторонніх інтеграторів, вартість послуг яких часто перевищує ціну самого ПЗ у кілька разів.

Розроблена інформаційна система «SafeStock MS» побудована виключно на базі безкоштовного open-source технологічного стеку (мова Python, мікрофреймворк Flask, ORM-бібліотека SQLAlchemy). Програма підтримує концепцію Self-hosted розгортання — її може бути інстальовано безпосередньо на наявному серверному обладнанні складського комплексу підприємства. Це повністю усуває ліцензійні витрати, нівелює залежність від іноземних вендорів, а сукупна вартість володіння (Total Cost of Ownership, TCO) зводиться лише до базового адміністрування системи власною ІТ-службою компанії.

2. Архітектурна легкість, швидкодія та лінійна масштабованість (Scalability)

Універсальні ринкові ERP- та WMS-системи проєктуються з розрахунком на покриття максимально широкого спектра бізнесів. Через це їхній вихідний код містить величезну кількість вбудованого «баласту» (надлишкових модулів, важких фонових служб, систем бухгалтерського обліку різних країн, кадрових систем тощо). Цей баласт споживає значні обчислювальні потужності сервера, перевантажує інтерфейс користувача та створює тривалі затримки при генерації сторінок.

Завдяки мікроядерній філософії фреймворку Flask, власна система позбавлена будь-яких зайвих надбудов. Вона містить строго той функціонал, який необхідний комерсантикам та менеджерам ТОВ «Медіа СВ» для швидкої фіксації

транзакцій. Програмне ядро системи оптимізовано за допомогою об'єктно-реляційного відображення SQLAlchemy, що робить його повністю незалежним від типу СКБД (Database-Agnostic). Завдяки впровадженню алгоритму сторінкової пагінації на рівні SQL-запитів (відображення чітко по 10 номенклатурних одиниць на сторінку), швидкість рендерингу інтерфейсу каталогу має константну складність $O(1)$ відносно обсягу даних на клієнті. Навіть при стрімкому розширенні асортименту побутової техніки та електроніки до 100 000 і більше найменувань, система не зазнаватиме затримок у роботі, забезпечуючи миттєвий відгук інтерфейсу.

3. Безкомпромісна безпека за принципом Security-First та невідмовність дій

Більшість складських систем легкого класу, орієнтованих на малий та середній бізнес (наприклад, QuickBooks або прості локальні облікові програми), фокусуються виключно на математичному підрахунку залишків, практично повністю ігноруючи інструменти інформаційної безпеки. Паролі в таких системах часто зберігаються у відкритому або слабко зашифрованому вигляді, права користувачів не розділені, а історію транзакцій може бути легко підроблено або видалено матеріально відповідальною особою для приховування розбіжностей.

У системі «SafeStock MS» кіберзахист та цілісність даних зведені у ранг абсолютного архітектурного пріоритету:

- Алгоритм Vcrypt виключає збереження паролів у відкритому вигляді, захищаючи облікові записи від brute-force атак та витоків бази даних за рахунок випадкового соління та розтягування ключів.
- Декоратори RBAC надійно ізолюють захищені маршрути сервера, унеможливлюючи несанкціоноване підвищення привілеїв.
- Підсистема Audit Trail фіксує кожну маніпуляцію користувачів посекундно в некоректованій Append-Only структурі логів, реалізуючи принцип невідмовності дій (Non-repudiation).
- Метод Soft Delete замінює фізичне знищення рядків логічним маркуванням через атрибут is_active, що гарантує 100% збереження реляційної цілісності

даних, непорушність зовнішніх ключів (Foreign Keys) та абсолютну точність історичних звітів для фінансового аудиту.

4. Фізично-цифрова гібридизація (Physical-to-Digital) та автоматизація документообігу

Існуючі ринкові аналоги вимагають купівлі та налаштування сторонніх платних модулів або спеціалізованих утиліт для зв'язку бази даних із реальним складським простором та підготовки друкованих форм.

Власна розробка вже на етапі базового проєктування містить два інноваційні компоненти, що виводять її на рівень сучасних індустріальних стандартів. Модуль динамічного QR-кодування автоматично трансформує унікальний артикул (SKU) товару у графічний двовимірний код, створюючи готовий технологічний фундамент для впровадження лазерних сканерів та мобільних терміналів збору даних (ТЗД) безпосередньо у складських приміщеннях ТОВ «Медіа СВ». Паралельно з цим, аналітичний модуль на базі бібліотеки ReportLab забезпечує миттєву генерацію суворих звітних документів у форматі PDF. Завдяки обробці файлових потоків безпосередньо в оперативній пам'яті через об'єкти BytesIO та повній підтримці кириличного Unicode-кодування (шрифти DejaVuSans), система формує валідні, повністю готові до друку та підпису документи без засмічення дискового простору сервера.

Таким чином, розроблена інформаційна система «SafeStock MS» за всіма технічними, безпековими та фінансовими показниками випереджає існуючі ринкові аналоги, забезпечуючи ТОВ «Медіа СВ» повну ліцензійну та інфраструктурну незалежність, абсолютну прозорість обліку матеріальних цінностей та високу стійкість до внутрішніх і зовнішніх кіберзагроз.

4.5 Висновки за розділом

У четвертому розділі бакалаврської роботи було проведено детальне тестування, аналіз розгортання та оцінку ефективності створеного програмного забезпечення. За результатами виконання завдань сформульовано такі прикладні висновки:

1. Визначено оптимальні технічні вимоги до обчислювальної інфраструктури складського комплексу ТОВ «Медіа СВ» та розроблено чіткий покроковий алгоритм розгортання веб-застосунку в ізольованому віртуальному оточенні мови Python.
2. Спроектовано та успішно реалізовано серію функціональних тест-кейсів за методом «чорної скриньки», результати яких підтвердили повну працездатність бізнес-логіки та безпомилкове відпрацювання алгоритмів безпеки при спробах несанкціонованого доступу чи некоректного введення даних.
3. Представлено детальну демонстрацію адаптивного графічного інтерфейсу користувача, яка підтвердила високу експлуатаційну надійність візуальної сигналізації дефіциту «Low Stock Alert» та модулів операційного документування транзакцій.
4. Проведено комплексний порівняльний аналіз розробленої IC «SafeStock MS» з провідними світовими аналогами (SAP, Odoo, QuickBooks), що підтвердив її повну фінансову, ліцензійну та технологічну автономність, високу архітектурну гнучкість і безкомпромісний рівень захисту комерційної таємниці підприємства.

ВИСНОВКИ

У кваліфікаційній бакалаврській роботі успішно розв'язано актуальну прикладну інженерну задачу — виконано комплексне проєктування, програмну реалізацію, розгортання та всебічне функціональне тестування інформаційної системи автоматизованого обліку та управління запасами «SafeStock MS» для складського комплексу підприємства ТОВ «Медіа СВ».

На основі проведених теоретичних досліджень, розроблених архітектурних рішень та отриманих емпіричних результатів сформульовано такі розширені висновки, які повністю підтверджують виконання завдань бакалаврської роботи:

1. Проведено глибокий системний аналіз предметної області складської логістики об'єкта автоматизації (ТОВ «Медіа СВ»). У результаті дослідження було виявлено критичні недоліки існуючих децентралізованих підходів обліку цифрової техніки та електроніки, серед яких: висока ймовірність виникнення людських помилок при ручному введенні серійних номерів, відсутність оперативної аналітики щодо дефіцитних позицій та незахищеність даних від несанкціонованих внутрішніх маніпуляцій. На основі цих даних було формалізовано строгі функціональні та технічні вимоги до розроблюваного вебзастосунку в частині забезпечення швидкодії, прозорості та кібербезпеки.
2. Здійснено критичний аналіз теоретичних методів управління запасами (зокрема моделей EOQ, ABC/XYZ-аналізу) та математичної концепції точки замовлення (ROP). Це дозволило науково обґрунтувати та інтегрувати в логіку системи критичну межу залишків ходових товарів на рівні 10 одиниць. У результаті порівняльного аналізу наявного ринкового програмного забезпечення (SAP Business One, Odoo, QuickBooks) доведено інженерну доцільність створення кастомного рішення на базі легкого мікрофреймворку Flask (мова Python), що дозволило повністю усунути ліцензійну залежність

від іноземних вендорів та виключити надлишковий архітектурний «баласт» комерційних ERP-платформ.

3. Спроектовано та програмно реалізовано надійне інженерне ядро системи на основі тришарового архітектурного патерну Model-View-Controller (MVC).
 - Шар даних (Model). За допомогою інструменту об'єктно-реляційного відображення SQLAlchemy ORM розроблено нормалізовану реляційну схему сховища з 7 взаємопов'язаних таблиць, що забезпечило повну незалежність коду від кінцевого типу СКБД (Database-Agnostic).
 - Шар логіки (Controller). Реалізовано контролери операційного обліку руху товарів із впровадженням алгоритму сторінкової пагінації, що забезпечує стабільну швидкість рендерингу інтерфейсу константної складності $O(1)$ незалежно від масштабу бази даних.
 - Шар представлення (View). Створено адаптивний, ергономічний інтерфейс користувача (Jinja2 / Bootstrap 5), повністю оптимізований як для стаціонарних ПК, так і для мобільних складських терміналів збору даних (ТЗД).
 - Автоматизація документообігу. Інтегровано модулі динамічної генерації двовимірних графічних ідентифікаторів (QR-кодів) для кожного артикула (SKU) та експорту офіційних звітів у форматі PDF. Завдяки обробці файлових потоків безпосередньо в оперативній пам'яті через об'єкти BytesIO та повній підтримці кириличного кодування Unicode (шрифт DejaVuSans), система формує готові до підпису документи без створення тимчасового сміття на жорсткому диску сервера.
4. Забезпечено безкомпромісний рівень інформаційної безпеки та цілісності даних на основі методології Security-First. Поєднання криптографічного алгоритму bcrypt для соленого хешування паролів, трирівневої рольової моделі контролю доступу RBAC на рівні кастомних декораторів маршрутів, некоректованого журналу безпеки Audit Trail (Append-Only структура) та

інженерної техніки м'якого видалення Soft Delete дозволило створити повністю захищене середовище. Розроблена архітектура повністю унеможлиблює приховану модифікацію або видалення даних матеріально відповідальними особами, гарантуючи виконання принципу невідмовності дій персоналу (Non-repudiation) та непорушність зовнішніх ключів (Foreign Keys) для фінансового аудиту.

5. Проведено комплексне функціональне тестування розробленого програмного забезпечення за методом «чорної скриньки». Експериментальні результати виконання спроектованих тест-кейсах (тести рольової ізоляції, захисту від негативного балансу при списанні, тригерування віджета «Low Stock Alert» та верифікації Soft Delete) повністю підтвердили сувору реляційну цілісність сховища, відсутність прихованих дефектів та абсолютну готовність системи «SafeStock MS» до успішного впровадження в реальну операційну діяльність складського комплексу ТОВ «Медіа СВ»

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. [Чинний від 2016-07-01]. Київ : ДП «УкрНДНЦ», 2016. 31 с.
2. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. [Чинний від 2016-07-01]. Київ : ДП «УкрНДНЦ», 2016. 17 с.
3. Грінберг М. Flask Web Development: Developing Web Applications with Python. 2nd ed. Sebastopol : O'Reilly Media, 2018. 316 p.
4. Шнайєр Б. Прикладна криптографія: протоколи, алгоритми та вихідний код. Київ : Діалектика, 2020. 624 с.
5. Алвеш Ж. Побудова захищених систем на Python за допомогою мікрофреймворків / пер. з англ. Київ : МК-Прес, 2023. 288 с.
6. Басс Л., Клементс П., Кацман Р. Архітектура програмного забезпечення на практиці. 4-те вид. Київ : Вільямс, 2022. 544 с.
7. Шарп Б., Овеїс М. Web Security and Vulnerability Assessment. New York : Springer, 2021. 248 p.
8. Огляд архітектурних патернів проєктування: MVC, MVP, MVVM. Журнал інженерії програмного забезпечення. 2022. № 4 (12). С. 45–52.
9. Складування та логістика замовлень: сучасні методи оптимізації запасів (ROP та EOQ). Логістичний вісник України. 2024. Вип. 18. С. 112–120.
10. Документація Flask (версія 3.0.x). Flask Documentation. URL: <https://flask.palletsprojects.com/> (дата звернення: 12.02.2026).
11. Документація SQLAlchemy ORM: Робота з реляційними моделями. SQLAlchemy. URL: <https://www.sqlalchemy.org/> (дата звернення: 18.02.2026).

12. Стандарт OWASP Top 10: Керівництво із захисту веб-додатків від уразливостей. OWASP Foundation. 2025. URL: <https://owasp.org/www-project-top-ten/> (дата звернення: 05.01.2026).
13. ReportLab PDF Generation User Guide. ReportLab Graphics. 2025. URL: <https://www.reportlab.com/docs/reportlab-userguide.pdf> (дата звернення: 03.03.2026).
14. Сміт Д., Джонсон А. Оптимізація складських процесів за допомогою двовимірного штрихкодуювання. Автоматизація та сучасні технології. 2023. № 2. С. 89–95.
15. Форро П. Реалізація Role-Based Access Control (RBAC) у сучасних веб-інтерфейсах. Кібербезпека та захист інформації. 2024. Т. 11, № 1. С. 34–41.
16. Провосіс Н. Дослідження криптографічної стійкості алгоритмів сімейства Crypt (Bcrypt, Scrypt, Argon2). Комп'ютерні методи та системи безпеки. 2023. Вип. 5. С. 102–110.
17. Антонов В. К. Логістика складських запасів: теорія і практика управління ризиками дефіциту. Харків : Фоліо, 2022. 312 с.
18. Ларман К. Застосування UML і шаблонів проєктування. 3-тє вид. Москва : Вільямс, 2019. 736 с.
19. Документація СУБД SQLite: Організація вбудованих реляційних баз даних. SQLite Engine. URL: <https://www.sqlite.org/docs.html> (дата звернення: 21.01.2026).
20. Мельник О. В., Коваленко С. М. Проєктування веб-застосунків на базі архітектурного патерну MVC. Вісник Національного університету «Львівська політехніка». Серія: Інформаційні системи та технології. 2023. № 1045. С. 120–128.
21. Таненбаум Е., Бос Г. Сучасні операційні системи. 5-те вид. Київ : Питер, 2021. 1120 с.
22. Федоров А. М. Безпека веб-додатків: від автентифікації до аудиту дій користувача. Український журнал кібербезпеки. 2024. Т. 6, № 2. С. 77–84.

23. Тейлор Т. Особливості використання методології «Soft Delete» для забезпечення цілісності реляційних СКБД. Бази даних та інформаційні системи. 2023. № 3 (14). С. 15–22.

24. Python qrcode library documentation and source specifications. PyPI ecosystem. URL: <https://pypi.org/project/qrcode/> (дата звернення: 14.03.2026).

25. Потій О. В. Методи та засоби захисту інформації в комп'ютерних системах. Навчальний посібник. Київ : КНЕУ, 2021. 416 с.

26. Шевченко Р. П. Автоматизація паперового документообігу на підприємствах за допомогою динамічного рендерингу звітів. Комп'ютерне моделювання та ІТ-технології. 2025. № 1. С. 58–66.

27. Впровадження принципів логування Audit Trail в корпоративних інформаційних системах управління ресурсами. Технологічний аудит та резерви виробництва. 2024. № 4/2(66). С. 12–18.

28. Стівенс Р. Проєктування та розробка систем управління базами даних. Спб. : Питер, 2020. 400 с.

29. Bootstrap 5 Framework Official Layout and Component Architecture. Bootstrap Core. URL: <https://getbootstrap.com/docs/5.3/> (дата звернення: 10.01.2026).

30. Jinja2 Templates Engine Documentation: Secure Rendering for Python Web Frameworks. Pallets Projects. URL: <https://jinja.palletsprojects.com/> (дата звернення: 15.02.2026).

31. Блінов В. О. Безпека інформаційних систем: архітектурні аспекти та протидія внутрішнім загрозам. Львів : Видавництво Львівської політехніки, 2022. 240 с.

32. Кузнець С. О., Сидоренко І. В. Інтеграція мобільних терміналів збору даних в логістичну інфраструктуру підприємств. Сучасна спеціалізована логістика. 2025. Вип. 22. С. 143–151.

ДОДАТКИ

ДОДАТОК А

Програмний код ключових модулів безпеки та логіки системи «SafeStock MS»

Програмні засоби:

- мова програмування — Python;
- веб-фреймворк — Django;
- система керування базами даних (СКБД) — SQLite;
- об'єктно-реляційне відображення — Django ORM;
- бібліотека генерації звітів — ReportLab;
- обробка бінарних потоків даних — BytesIO.

App.py

```
import os, csv, io, shutil, urllib.parse, base64
from datetime import datetime
from flask import Flask, render_template, redirect, url_for, request, flash,
Response, send_file
from flask_bcrypt import Bcrypt
from flask_login import LoginManager, login_user, logout_user, login_required,
current_user
from models import db, User, Role, Category, Product, Transaction, AuditLog,
Supplier
from fpdf import FPDF
from sqlalchemy import func

# Налаштування для генерації графіків без GUI
import matplotlib
matplotlib.use('Agg')
import matplotlib.pyplot as plt

app = Flask(__name__)
app.config['SECRET_KEY'] = 'safestock_pro_final_2026'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///inventory.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

db.init_app(app)
bcrypt = Bcrypt(app)
login_manager = LoginManager(app)
login_manager.login_view = 'login'

@login_manager.user_loader
```

```

def load_user(user_id):
    return User.query.get(int(user_id))

@app.context_processor
def utility_processor():
    def generate_qr_url(sku):
        return f"https://api.qrserver.com/v1/create-qr-
code/?size=150x150&data={urllib.parse.quote(str(sku))}"
    return dict(generate_qr_url=generate_qr_url)

def log_action(action):
    if current_user.is_authenticated:
        db.session.add(AuditLog(user_id=current_user.id, action=action))
        db.session.commit()

def get_sales_chart():
    """Генерація діаграми топ-5 товарів за продажами (транзакції OUT)"""
    top_sales = db.session.query(
        Product.name,
        func.sum(Transaction.quantity).label('total_qty')
    ).join(Transaction).filter(Transaction.type == 'OUT')\
        .group_by(Product.id).order_by(func.sum(Transaction.quantity).desc()).limit(5)
    .all()

    if not top_sales:
        return None

    names = [row[0].split(' ')[0] + '..' for row in top_sales]
    counts = [row[1] for row in top_sales]

    plt.figure(figsize=(8, 4))
    plt.bar(names, counts, color='#0d6efd')
    plt.title('Топ-5 за попитом', fontsize=12, fontweight='bold', pad=10)
    plt.ylabel('К-сть', fontsize=9)
    plt.xticks(fontsize=8, rotation=10)
    plt.tight_layout()

    img = io.BytesIO()
    plt.savefig(img, format='png', transparent=True, bbox_inches='tight',
pad_inches=0.05)
    img.seek(0)
    plot_url = base64.b64encode(img.getvalue()).decode()
    plt.close()
    return plot_url

```

```

# --- МАРШРУТИ ---

@app.route('/')
@login_required
def dashboard():
    page = request.args.get('page', 1, type=int)
    search = request.args.get('search', '')

    query = Product.query.filter(Product.is_active == True)
    if search:
        query = query.filter(Product.name.ilike(f'%{search}%') |
Product.sku.ilike(f'%{search}%'))

    pagination = query.order_by(Product.id.desc()).paginate(page=page, per_page=10)

    sku_count = Product.query.filter_by(is_active=True).count()
    low_stock = Product.query.filter(Product.is_active == True, Product.quantity <
10).all()
    sales_chart = get_sales_chart()

    return render_template('index.html',
                           products=pagination.items,
                           pagination=pagination,
                           categories=Category.query.all(),
                           suppliers=Supplier.query.all(),
                           sku_count=sku_count,
                           low_stock=low_stock,
                           sales_chart=sales_chart,
                           search=search)

@app.route('/login', methods=['GET', 'POST'])
def login():
    if current_user.is_authenticated: return redirect(url_for('dashboard'))
    if request.method == 'POST':
        user = User.query.filter_by(username=request.form.get('username')).first()
        if user and bcrypt.check_password_hash(user.password_hash,
request.form.get('password')):
            login_user(user); log_action("Вхід")
            return redirect(url_for('dashboard'))
        flash("Невірний логін або пароль", "danger")
    return render_template('login.html')

@app.route('/logout')
@login_required
def logout():
    log_action("Вихід"); logout_user()
    flash("Ви вийшли із системи", "info")
    return redirect(url_for('login'))

```

```

@app.route('/update_stock/<int:product_id>', methods=['POST'])
@login_required
def update_stock(product_id):
    product = Product.query.get_or_404(product_id)
    try:
        change = int(request.form.get('change', 0))
        product.quantity += change
        db.session.add(Transaction(product_id=product.id, user_id=current_user.id,
type='IN' if change > 0 else 'OUT', quantity=abs(change)))
        db.session.commit()
        log_action(f"Зміна кількості {product.name}: {change}")
    except: flash("Помилка при оновленні", "warning")
    return redirect(url_for('dashboard'))

@app.route('/add_product', methods=['POST'])
@login_required
def add_product():
    new_p = Product(
        name=request.form.get('name'),
        sku=request.form.get('sku'),
        category_id=int(request.form.get('category_id')),
        supplier_id=int(request.form.get('supplier_id')),
        quantity=int(request.form.get('quantity', 0)),
        min_stock_level=10
    )
    db.session.add(new_p); db.session.commit()
    log_action(f"Додано новий товар: {new_p.name}")
    flash("Товар додано!", "success")
    return redirect(url_for('dashboard'))

@app.route('/add_category', methods=['POST'])
@login_required
def add_category():
    if current_user.role.name not in ['Admin', 'Manager']:
        flash("У вас немає прав для створення категорій", "danger")
        return redirect(url_for('dashboard'))
    name = request.form.get('name')
    if name:
        db.session.add(Category(name=name)); db.session.commit()
        log_action(f"Нова категорія: {name}")
    return redirect(url_for('dashboard'))

@app.route('/add_supplier', methods=['POST'])
@login_required
def add_supplier():
    if current_user.role.name not in ['Admin', 'Manager']:
        flash("У вас немає прав для додавання постачальників", "danger")
        return redirect(url_for('dashboard'))
    name = request.form.get('name')

```

```

    if name:
        db.session.add(Supplier(name=name, contact=request.form.get('contact')));
db.session.commit()
        log_action(f"Новий постачальник: {name}")
        return redirect(url_for('dashboard'))

@app.route('/report_pdf')
@login_required
def report_pdf():
    products = Product.query.filter_by(is_active=True).all()
    pdf = FPDF()
    pdf.add_page()
    font_path = "DejaVuSans.ttf"
    if os.path.exists(font_path):
        pdf.add_font('DejaVu', '', font_path, uni=True)
        pdf.set_font('DejaVu', '', 12)
    else:
        pdf.set_font("Arial", size=12)
    pdf.cell(200, 10, txt="SafeStock MS - Звіт про залишки", ln=True, align='C')
    pdf.ln(10)
    for p in products:
        line = f"{p.sku} | {p.name[:40]} | Кількість: {p.quantity}"
        pdf.cell(200, 10, txt=line, ln=True)
    buf = io.BytesIO()
    pdf_str = pdf.output(dest='S')
    buf.write(pdf_str); buf.seek(0)
    return send_file(buf, download_name="stock_report.pdf", as_attachment=True)

@app.route('/logs')
@login_required
def view_logs():
    if current_user.role.name != 'Admin': return "Denied", 403
    logs = AuditLog.query.order_by(AuditLog.timestamp.desc()).all()
    return render_template('logs.html', logs=logs)

def setup_database():
    with app.app_context():
        db.create_all()
        if not Role.query.first():
            db.session.add_all([Role(name='Admin'), Role(name='Manager'),
Role(name='Worker')])
            db.session.commit()
            for r in [('admin',1), ('manager',2), ('worker',3)]:
                hp = bcrypt.generate_password_hash(f"{r[0]}123").decode('utf-8')
                db.session.add(User(username=r[0], password_hash=hp, role_id=r[1]))
            db.session.commit()
            if not Category.query.first():
                db.session.add_all([Category(name='Побутова техніка'),
Category(name='Електроніка'), Category(name='Інструменти')])

```

```

        db.session.add_all([Supplier(name='Rozetka'), Supplier(name='Brain'),
Supplier(name='MTI Disti')])
        db.session.commit()
        if not Product.query.first():
            items = [
                ("Кавомашина PHILIPS Series 5500 LatteGo EP5547/90", "424314774",
1, 1, 3),
                ("Смартфон Samsung Galaxy S24 Ultra 12/512GB Titanium Black",
"334512990", 2, 1, 12),
                ("Apple iPhone 15 Pro 128GB Natural Titanium", "390234112", 2, 1,
8),
                ("Смартфон Xiaomi Redmi Note 13 Pro+ 5G 12/512GB", "410223556", 2,
2, 20),
                ("Смартфон Google Pixel 8 Pro 12/128GB Obsidian", "400112883", 2,
1, 5),
                ("Смартфон Motorola Edge 40 8/256GB Eclipse Black", "380445221", 2,
3, 15),
                ("Apple Watch Series 9 GPS 45mm Starlight", "395667112", 2, 1, 10),
                ("Наушники Sony WH-1000XM5 Black", "320998443", 2, 3, 7),
                ("Планшет Apple iPad Air M2 11\" 128GB Wi-Fi Space Gray",
"450112776", 2, 1, 6),
                ("Наушники Apple AirPods Pro (2nd gen) MagSafe", "340556119", 2,
2, 25),
                ("Смартфон Samsung Galaxy Flip5 8/256GB Graphite", "350443228", 2,
3, 4),
                ("Пилосос Dyson V15 Detect Total Clean", "228914552", 1, 1, 2),
                ("Холодильник LG Side-by-Side GC-B257JFYV", "110556889", 1, 2, 4),
                ("Пральна машина Bosch WAN28262UA", "120443990", 1, 3, 6),
                ("Робот-пилосос Roborock S8 White", "430112445", 1, 1, 11),
                ("Мультиварка-скороварка Tefal Cook4Me+ CY851130", "150667223", 1,
2, 14),
                ("Электрочайник KitchenAid Artisan 1.5L Black", "210998334", 1, 1,
9),
                ("Кухонный комбайн Kenwood Cooking Chef XL", "180556221", 1, 3, 5),
                ("Мікрохвильова піч Samsung MS23F301TAS", "140443112", 1, 2, 18),
                ("Праска Philips Azur 8000 Series DST8050/20", "190223445", 1, 1,
22),
                ("Ноутбук Apple MacBook Air 13 M3 8/256GB Space Gray", "415562110",
2, 1, 4),
                ("Ноутбук ASUS ROG Strix G16 G614JV-N3110", "420334887", 2, 2, 3),
                ("Монітор 27\" ASUS ROG Swift OLED PG27AQDM", "112233445", 2, 2,
8),
                ("Клавіатура Logitech G Pro X TKL Lightspeed Black", "550112998",
2, 3, 12),
                ("Миша Razer DeathAdder V3 Pro White", "540667223", 2, 3, 15),
                ("Відеокарта ASUS RTX 4080 Super", "480112554", 2, 2, 2),
                ("Процесор Intel Core i9-14900K Box", "470667112", 2, 3, 6),
                ("SSD Samsung 990 Pro 2TB M.2", "490443887", 2, 1, 30),
                ("Ноутбук Lenovo Legion Slim 5 16IRH8", "410556223", 2, 3, 7),

```

```

        ("Sony PlayStation 5 Slim", "440112556", 2, 2, 20),
        ("Перфоратор Makita HR2470", "100223445", 3, 2, 15),
        ("Шуруповерт Bosch GSR 12V-30", "105667112", 3, 3, 22),
        ("Кутова шліфмашина DeWalt DWE4257", "115443887", 3, 2, 11),
        ("Набір інструментів Intertool ET-6099", "130112445", 3, 3, 40),
        ("Лобзик Metabo STEB 65 Quick", "125667334", 3, 2, 14),
        ("Далекомір Bosch GLM 50-27 CG", "145443221", 3, 3, 8),
        ("Будівельний фен Makita HG5030K", "155112889", 3, 2, 10),
        ("Мийка Karcher K 5 Compact", "165998112", 3, 3, 5),
        ("Електропила Bosch UniversalChain 35", "175223445", 3, 2, 7),
        ("Набір свердел DeWalt Extreme 2", "185667554", 3, 3, 50),
        ("Маршрутизатор ASUS RT-AX88U Pro", "510223445", 2, 2, 12),
        ("Зовнішній диск WD My Passport 2TB", "520443112", 2, 3, 25),
        ("Графічний планшет Wacom Intuos M", "530667889", 2, 1, 9),
        ("Веб-камера Logitech C920s Pro HD", "540112554", 2, 3, 14),
        ("Колонки Edifier M1360 Black", "550998223", 2, 2, 10),
        ("Зарядна станція EcoFlow RIVER 2", "600112445", 1, 1, 5),
        ("Електросамокат Xiaomi Mi Scooter 4 Pro", "610223887", 2, 1, 3),
        ("Музична колонка Marshall Emberton II", "620443112", 2, 2, 11),
        ("Блок живлення Corsair RM850x White", "630667445", 2, 3, 8),
        ("СВО NZXT Kraken Elite", "640112223", 2, 3, 4),
        ("Тостер KitchenAid 5KMT221EOB", "230223554", 1, 1, 13),
        ("Блендер Braun MQ 9147X", "240443665", 1, 2, 17)
    ]
    for n, s, c, sup, q in items:
        db.session.add(Product(name=n, sku=s, category_id=c,
supplier_id=sup, quantity=q, min_stock_level=10))
    db.session.commit()

if __name__ == '__main__':
    setup_database()
    app.run(debug=True)

```

index.html

```

{% extends "layout.html" %}
{% block content %}

<div class="row g-3 mb-4 align-items-stretch">

    <div class="col-md-2">
        <div class="card shadow-sm p-2 text-center bg-primary text-white border-0
h-100 d-flex flex-column justify-content-center">
            <small class="text-uppercase fw-bold opacity-75" style="font-size:
0.65rem; line-height: 1.2;">Товарних позицій</small>
            <h2 class="fw-bold mt-2 mb-0" style="font-size: 2.2rem;">{{ sku_count
}}</h2>
        </div>
    </div>

```

```

</div>

<div class="col-md-5">
  <div class="card shadow-sm border-0 h-100 overflow-hidden">
    <div class="card-body p-0 d-flex align-items-center justify-content-center">
      {% if sales_chart %}
        
      {% else %}
        <div class="p-4 text-muted small text-center w-100">
          <div class="border border-dashed rounded p-3">
            <p class="mb-0">Графік попиту з'явиться після
відвантажень (OUT)</p>
          </div>
        </div>
      {% endif %}
    </div>
  </div>

<div class="col-md-5">
  <div class="card shadow-sm border-0 h-100">
    <div class="card-header bg-white border-0 py-2 d-flex align-items-center">
      <h6 class="mb-0 text-danger fw-bold" style="font-size: 0.9rem;">📉
Low Stock Alert (< 10)</h6>
    </div>
    <div class="card-body p-0">
      <div style="max-height: 130px; overflow-y: auto;" class="px-3 pb-2">
        {% if low_stock %}
          {% for p in low_stock %}
            <div class="d-flex justify-content-between align-items-center py-2 border-bottom">
              <div class="text-truncate" style="max-width: 280px;">
                <small class="fw-bold text-dark">{{ p.name
}}</small>
              </div>
              <span class="badge bg-danger rounded-pill">{{
p.quantity }}</span>
            </div>
          {% endfor %}
        {% else %}
          <div class="py-4 text-center text-muted small">Всі запаси в
нормі ☒</div>
        {% endif %}
      </div>
    </div>
  </div>

```

```

        </div>
    </div>
</div>

<div class="card shadow-sm border-0 mb-4 p-3 bg-white">
    <div class="row align-items-center">
        <div class="col-md-5 mb-3 mb-md-0">
            <form action="/" method="GET" class="input-group">
                <input type="text" name="search" class="form-control border-light shadow-none px-4" placeholder="Пошук за назвою або SKU..." value="{{ search }}">
                <button class="btn btn-primary px-4">Знайти</button>
            </form>
        </div>

        <div class="col-md-7 text-md-end d-flex gap-2 justify-content-md-end align-items-center">
            <button class="btn btn-success fw-bold px-4 shadow-sm" data-bs-toggle="modal" data-bs-target="#addMdl">
                + Додати товар
            </button>

            {% if current_user.role.name in ['Admin', 'Manager'] %}
            <div class="btn-group">
                <button class="btn btn-outline-primary btn-sm px-3" data-bs-toggle="modal" data-bs-target="#catMdl">+ Категорія</button>
                <button class="btn btn-outline-primary btn-sm px-3" data-bs-toggle="modal" data-bs-target="#supMdl">+ Постачальник</button>
            </div>
            {% endif %}

            <a href="{{ url_for('report_pdf') }}" class="btn btn-danger btn-sm px-3 shadow-none"><img alt="PDF icon" data-bbox="258 628 275 642"/> PDF Звіт</a>
        </div>
    </div>
</div>

<div class="card shadow-sm border-0 overflow-hidden">
    <table class="table align-middle mb-0 table-hover">
        <thead class="table-light">
            <tr>
                <th class="ps-4">Артикул</th>
                <th>Назва</th>
                <th style="width:140px">Кількість (+/-)</th>
                <th>Статус</th>
                <th class="pe-4 text-end">Дії</th>
            </tr>
        </thead>
        <tbody>

```

```

        {% for p in products %}
        <tr style="{% if p.quantity < 10 %}background-color: #fff8f8;{% endif
%}">
            <td class="ps-4"><code>{{ p.sku }}</code></td>
            <td class="fw-bold text-dark">{{ p.name }}</td>
            <td>
                <form action="{% url_for('update_stock', product_id=p.id) %}"
method="POST" class="d-flex align-items-center">
                    <input type="number" name="change" class="form-control
form-control-sm border-0 bg-light text-center" style="width:50px" placeholder="0">
                    <span class="ms-2 fw-bold text-primary">{{ p.quantity
}}</span>
                    <button class="btn btn-sm btn-link text-success p-0 ms-
2">✓</button>
                </form>
            </td>
            <td>
                {% if p.quantity < 10 %}<span class="badge bg-
danger">Critical</span>
                {% else %}<span class="badge bg-success">OK</span>{% endif %}
            </td>
            <td class="pe-4 text-end">
                <button class="btn btn-sm btn-light border shadow-sm"
onclick="openQR('{{ generate_qr_url(p.sku) }}', '{{ p.name }}')">QR</button>
            </td>
        </tr>
        {% endfor %}
    </tbody>
</table>
</div>

<nav class="mt-4">
    <ul class="pagination justify-content-center">
        {% for page_num in pagination.iter_pages() %}
            {% if page_num %}
                <li class="page-item {% if page_num == pagination.page %}active{%
endif %}">
                    <a class="page-link shadow-sm mx-1 rounded" href="{%
url_for('dashboard', page=page_num, search=search) %}">{{ page_num }}</a>
                </li>
            {% endif %}
        {% endfor %}
    </ul>
</nav>

<div class="modal fade" id="catMdl" tabindex="-1"><div class="modal-dialog modal-
dialog-centered"><form action="/add_category" method="POST" class="modal-content
border-0 shadow"><div class="modal-header fw-bold">Нова категория</div><div
class="modal-body"><input type="text" name="name" class="form-control"

```

```

required></div><div class="modal-footer"><button class="btn btn-
primary">Зберегти</button></div></form></div></div>

<div class="modal fade" id="supMdl" tabindex="-1"><div class="modal-dialog modal-
dialog-centered"><form action="/add_supplier" method="POST" class="modal-content
border-0 shadow"><div class="modal-header fw-bold">Новий постачальник</div><div
class="modal-body"><input type="text" name="name" class="form-control mb-2"
placeholder="Назва" required><input type="text" name="contact" class="form-control"
placeholder="Контакт"></div><div class="modal-footer"><button class="btn btn-
primary">Зберегти</button></div></form></div></div>

<div class="modal fade" id="addMdl" tabindex="-1"><div class="modal-dialog modal-
dialog-centered"><form action="/add_product" method="POST" class="modal-content
border-0 shadow-lg"><div class="modal-header fw-bold"><img alt="box icon" data-bbox="655 308 675 323"/> Додати товар на
склад</div><div class="modal-body">
  <input type="text" name="name" class="form-control mb-2" placeholder="Назва
товару" required>
  <input type="text" name="sku" class="form-control mb-2" placeholder="Артикул
(SKU)" required>
  <div class="row g-2 mb-2">
    <div class="col-md-6">
      <label class="small text-muted">Категорія</label>
      <select name="category_id" class="form-select">{% for c in categories
%}<option value="{{ c.id }}">{{ c.name }}</option>{% endfor %}</select>
    </div>
    <div class="col-md-6">
      <label class="small text-muted">Постачальник</label>
      <select name="supplier_id" class="form-select">{% for s in suppliers
%}<option value="{{ s.id }}">{{ s.name }}</option>{% endfor %}</select>
    </div>
  </div>
  <input type="number" name="quantity" class="form-control"
placeholder="Початкова кількість" required>
</div><div class="modal-footer"><button class="btn btn-success w-100 fw-bold py-
2">Зберегти позицію</button></div></form></div></div>

<div class="modal fade" id="qrModal" tabindex="-1"><div class="modal-dialog modal-
sm modal-dialog-centered text-center"><div class="modal-content p-4 border-0
shadow"><h6 id="qT" class="mb-3 fw-bold"></h6><img id="qI" src="" class="img-fluid
rounded border mb-3"><button class="btn btn-dark btn-sm w-100" data-bs-
dismiss="modal">OK</button></div></div></div>

<script>
  function openQR(u, n) {
    const qrModal = new bootstrap.Modal(document.getElementById('qrModal'));
    document.getElementById('qI').src = u;
    document.getElementById('qT').innerText = n;
    qrModal.show();
  }

```

```
</script>
```

```
{% endblock %}
```