НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ

Завідувач кафедри
комп'ютерних наук
(назва кафедри)

_____ Голуб Б.Л.
(підпис) (ПІБ)

"___" _____ 2025 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему

«Програмне забезпечення системи управління складом із застосуванням технології RFID для автоматизації обліку»

Спеціальність 121 – «Інженерія програмного забезпечення»

Гарант освітньої програми

к.т.н., доц. _____ Вайганг Г.О.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник бакалаврської кваліфікаційної роботи

к.е.н., ст.викладач _____ Ніколаєнко Д.В.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав

_____ Шелудько О. О.
(підпис) (ПІБ)

КИЇВ – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ
Факультет інформаційних технологій

ЗАТВЕРДЖУЮ

Завідувач кафедри
комп'ютерних наук
(назва кафедри)

_____ Голуб Б.Л.
(підпис) (ПІБ)

"___" _____ 2025 р.

З А В Д А Н Н Я
на виконання бакалаврської кваліфікаційної роботи студенту
Шелудько Олександр Олегович

Спеціальність 121 – «Інженерія програмного забезпечення»

1. Тема бакалаврської кваліфікаційної роботи «Програмне забезпечення системи управління складом із застосуванням технології RFID для автоматизації обліку» затверджена наказом ректора НУБіП України від 16.12.2024 № 2248
2. Термін подання завершеної роботи на кафедру_____
(рік, місяць, число)

3. Вихідні дані:

_____

4. Перелік питань, що розглядаються:
- Аналіз проблемної області
- Моделювання предметної області
- Проєктування програмної системи
- Впровадження та експлуатація системи

Дата видачі завдання "_____"_____20___ р.
Керівник бакалаврської кваліфікаційної роботи

к.е.н., ст.викладач _____ Ніколаєнко Д.В.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Завдання прийняв до виконання

_____ Шелудько О. О.
(підпис) (ПІБ)

Календарний план

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	21 лютого 2025	
2	Аналіз предметної області	березень 2025	
3	Моделювання предметної області	березень 2025	
4	Проєктування програмної системи	квітень 2025	
5	Розгортання та експлуатація програмного забезпечення	квітень- травень 2025	
6	Економічне дослідження розробки та експлуатації програмної системи	квітень- травень 2025	
7	Оформлення записки	травень 2025	
8	Перевірка на плагіат	2 червня 2025	
9	Проходження нормо контроля	29 травня – 4 червня 2025	
10	Проходження передзахисту	5-7 червня 2025	
11	Захист роботи	9-16 червня 2025	

Керівник бакалаврської кваліфікаційної роботи

к.е.н., ст.викладач		Ніколаєнко Д.В.
(науковий ступінь та вчене звання)	(підпис)	(ПІБ)

Студент

		Шелудько О. О.
	(підпис)	(ПІБ)

Анотація

Бакалаврська кваліфікаційна робота «Програмне забезпечення системи управління складом із застосуванням технології RFID для автоматизації обліку» присвячена розробці прототипу інформаційної системи, що дозволяє здійснювати автоматизований контроль руху товарно-матеріальних цінностей на складі. Основною метою дослідження є створення програмного забезпечення, яке забезпечує ефективне зчитування, ідентифікацію та облік об'єктів за допомогою RFID-міток, а також надання користувачу інтерфейсу для контролю, аналітики та керування процесами.

Запропонована система поєднує у собі сучасні засоби автоматичної ідентифікації (RFID), реляційні бази даних для зберігання інформації (PostgreSQL), серверну частину (Flask), а також інтуїтивно зрозумілий веб-інтерфейс. Реалізована функціональність включає реєстрацію товарів, облік їх переміщення, контроль залишків та формування звітності. Впровадження технології RFID дозволяє мінімізувати людський фактор, зменшити час обробки складських операцій та підвищити точність інвентаризації.

Особливу увагу в роботі приділено структурі програмної архітектури, побудові логічної моделі даних, модульності системи та можливостям масштабування. Система підтримує як онлайн-режим із реальним зчитуванням RFID, так і тестовий режим із симуляцією даних. Розроблене рішення є універсальним, платформонезалежним та базується виключно на відкритих технологіях, що робить його придатним для впровадження у складській логістиці малих та середніх підприємств.

ABSTRACT

The bachelor's qualification thesis titled "Software for Warehouse Management System Using RFID Technology for Inventory Automation" focuses on the development of a software prototype designed to automate inventory tracking in a warehouse environment. The primary objective of the project is to create a system capable of reading, identifying, and registering warehouse items using RFID tags, while also providing a user-friendly interface for monitoring, analysis, and process management.

The proposed solution integrates modern automatic identification technology (RFID), a relational database (PostgreSQL) for structured data storage, a backend service (Flask), and a web-based user interface. The system implements features such as item registration, movement tracking, stock control, and reporting. The application of RFID enables significant reduction of human error, faster processing of warehouse operations, and increased accuracy during inventory audits.

Special attention is paid to the system’s modular architecture, logical data modeling, scalability, and ease of deployment. The software supports both live RFID data acquisition and a testing mode using simulated events.The developed system is platform-independent, open-source, and cost-effective, making it a viable solution for logistics automation in small and medium-sized enterprises.

ЗМІСТ

ВСТУП

У сучасних умовах стрімкого розвитку логістичних процесів та електронної комерції ефективне управління складськими ресурсами набуває особливого значення. Точність і швидкість обліку товарів на складі безпосередньо впливають на загальну продуктивність підприємства, рівень обслуговування клієнтів та конкурентоспроможність бізнесу. Саме тому дедалі більшої популярності набувають автоматизовані системи управління складом (Warehouse Management Systems — WMS), що інтегрують сучасні інформаційні технології, зокрема засоби автоматичної ідентифікації.

Однією з найперспективніших технологій в цій сфері є RFID (Radio Frequency Identification) — радіочастотна ідентифікація, яка дозволяє здійснювати безконтактне зчитування унікальних ідентифікаторів товарів у режимі реального часу. У порівнянні з традиційними штрихкодами, RFID забезпечує вищу швидкість зчитування, більший обсяг збережених даних і можливість обробки без прямої видимості. Це відкриває нові можливості для автоматизації обліку, контролю переміщення та зберігання товарів на складі.

Метою даної роботи є розробка програмного забезпечення для системи управління складом із використанням технології RFID, яке забезпечує автоматичну фіксацію руху товарів, ведення обліку та візуалізацію складських операцій через веб-інтерфейс. Така система дозволяє значно знизити ризик людських помилок, прискорити інвентаризацію та забезпечити оперативний доступ до інформації про стан запасів.

У роботі застосовано такі технології, як мікроконтролери з підтримкою RFID-модулів, мова програмування Python, фреймворк Flask для створення веб-серверної частини, а також PostgreSQL для зберігання даних. Особливу

увагу приділено питанням структуризації бази даних, моделюванню складських процесів та розробці механізмів автоматизованого обліку.

Апробація результатів розробки здійснювалася в тестовому середовищі з використанням симуляторів RFID-подій та демонстраційних наборів, що підтвердило функціональність основних модулів системи.

Структура пояснювальної записки включає чотири розділи:

- у першому розділі проаналізовано предметну область, сформульовано вимоги та постановку задачі;
- у другому розділі розроблено архітектуру системи та UML-діаграми;
- у третьому розділі описано реалізацію системи, інструментарій та особливості програмування;
- у четвертому розділі проведено тестування і наведено рекомендації щодо впровадження.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Склад є одним із ключових елементів логістичної інфраструктури підприємства, який забезпечує зберігання, облік, обробку та переміщення матеріальних ресурсів. Від ефективності управління складом залежить стабільність виробничих процесів, швидкість виконання замовлень, а також загальний рівень обслуговування клієнтів. У сучасних умовах зростання обсягів товарообігу, розширення асортименту продукції та необхідності оперативного контролю над залишками дедалі гострішою постає проблема автоматизації складських операцій.

Традиційні підходи до обліку на складах передбачають ручне введення даних, використання паперових носіїв або систем із ручним скануванням штрихкодів. Такі методи є трудомісткими, схильними до людських помилок і мають обмежену продуктивність при великому потоці товарів. Для подолання цих недоліків доцільним є впровадження систем автоматичної ідентифікації, зокрема на основі технології RFID (Radio Frequency Identification).
RFID-технологія передбачає використання міток (тегів), що містять унікальний ідентифікаційний код, та зчитувачів, які отримують дані з міток за допомогою радіохвиль. На відміну від штрихкодів, RFID-мітки не потребують прямої видимості та можуть зчитуватися одночасно у великій кількості. Це дозволяє значно пришвидшити операції надходження, відвантаження та інвентаризації товарів.

У рамках даної роботи предметною областю є система управління складом (Warehouse Management System — WMS), яка інтегрує облік товарів із RFID-скануванням. Така система дозволяє автоматизувати:

- ідентифікацію товарних одиниць на основі RFID-міток;
- реєстрацію операцій надходження, переміщення та відвантаження товарів;
- ведення бази даних із історією переміщень і залишків;
- моніторинг складських процесів у реальному часі;
- інвентаризацію без перерви у роботі складу.

Ключовими компонентами такої системи є:

- RFID-зчитувачі - пристрої, що приймають сигнал від міток та передають дані до сервера;
- RFID-мітки (пасивні або активні) - ідентифікатори, прикріплені до товарів, коробок або палет;
- Програмне забезпечення обліку - серверна частина, що фіксує, обробляє та зберігає інформацію про товарообіг;
- База даних - централізоване сховище інформації про товари, операції та склади;
- Інтерфейс користувача - веб-додаток для перегляду, аналізу та управління складськими процесами.

Таким чином, автоматизація обліку на основі RFID-технології дозволяє значно підвищити ефективність роботи складу, зменшити витрати на інвентаризацію, запобігти помилкам при ручному введенні даних та забезпечити прозорість операцій. Ця предметна область є актуальною для впровадження в багатьох сферах: від роздрібної торгівлі та дистрибуції до логістичних центрів, фармацевтики та виробництва.

1.2 Аналіз вимог до системи управління складом

На етапі розробки програмного забезпечення важливим кроком є формулювання чітких вимог до системи, що дозволяє забезпечити її відповідність очікуванням користувача, функціональність, надійність та зручність у використанні.
Вимоги до системи поділяються на функціональні та нефункціональні.

1.2.1 Функціональні вимоги

Функціональні вимоги описують, які саме дії повинна виконувати система в рамках управління складом із застосуванням RFID. Основні функції:

- Ідентифікація товарів за допомогою RFID-міток
  Система має забезпечувати зчитування унікального ідентифікатора кожної товарної одиниці, яка оснащена RFID-міткою.
- Реєстрація операцій надходження та відвантаження
  При проходженні товару через RFID-зчитувач відбувається автоматична фіксація події в базі даних із зазначенням часу, типу операції, ідентифікатора товару та користувача.
- Облік товарних залишків
  Система повинна забезпечити ведення точного обліку поточних запасів на складі з можливістю фільтрації за найменуванням, типом товару, датою надходження.
- Інвентаризація в реальному часі
  За допомогою ручного RFID-сканера користувач повинен мати змогу оперативно проводити звірку фізичних залишків із даними в системі.
- Ведення бази даних товарів
  Система повинна підтримувати створення, редагування та видалення записів про товари, включаючи інформацію про найменування, кількість, місце зберігання, статус.
- Формування журналу подій
  Всі дії в системі повинні реєструватися у вигляді журналу подій: що, коли і ким було виконано.

- Інтерфейс для перегляду та управління

  Передбачено веб-інтерфейс, у якому користувач може переглядати товарні залишки, історію операцій, керувати складськими одиницями та RFID-пристроями.

1.2.2 Нефункціональні вимоги

Нефункціональні вимоги визначають якісні характеристики системи, які не залежать від конкретних функцій:

- Продуктивність

  Система повинна обробляти RFID-зчитування в реальному часі з мінімальними затримками (до 1 секунда на подію).
- Масштабованість

  Архітектура повинна дозволяти підключення нових складів, зон зберігання, RFID-зчитувачів і користувачів без необхідності значної модифікації коду.
- Надійність

  Система має функціонувати у цілодобовому режимі (24/7) із мінімальним ризиком втрати даних.
- Безпека

  Впровадження авторизації користувачів із розмежуванням прав доступу (оператор, адміністратор, перегляд).
- Платформна незалежність

  Можливість запуску системи на різних ОС (Linux, Windows), а також використання через браузер.
- Простота встановлення та оновлення

  Використання Docker для контейнеризації і простого розгортання.
- Резервне копіювання даних

  Підтримка автоматичного створення резервних копій бази даних.

1.2.3 Обмежання системи

- У першій версії системи не передбачена обробка фото- або відеозображень товару.
- Зчитування RFID-міток можливе лише в межах радіуса дії пристроїв.
- Для тестування використовуються демо-набори RFID,або емулятор.

1.3 Моделювання процесів складського обліку

Для кращого розуміння логіки роботи складської системи обліку з використанням RFID доцільно застосувати інструменти системного аналізу та побудувати графічні моделі, що описують основні процеси, об’єкти та взаємодії у предметній області. Використання методів моделювання дозволяє:

- формалізувати складські бізнес-процеси;
- визначити взаємозв’язки між об’єктами системи;
- забезпечити цілісне бачення логіки функціонування програмного продукту;
- закласти основу для подальшого проєктування бази даних та програмних модулів.

UML-діаграма прецедентів (Use Case Diagram) - описує взаємодію користувачів із системою.

Основними акторими виступають:

- Оператор - здійснює облік товарів, проводить інвентаризацію, додає нові позиції;
- Адміністратор - налаштовує RFID-зчитувачі, користувачів, переглядає журнал подій;
- Система - автоматично реєструє RFID-зчитування та зберігає події.

Основними прецедентами виступають:

- Пошук товару за ID;

- Додавання нової товарної одиниці;
- Автоматичне зчитування RFID-мітки;
- Інвентаризація складу;
- Перегляд журналу подій;
- Налаштування RFID-зчитувачів.

UML-діаграма активності (Activity Diagram) - ілюструє типовий робочий процес:

1. Оператор підносить товар до RFID-зчитувача.
2. Зчитувач надсилає ідентифікатор у систему.
3. Система перевіряє товар у базі.
4. Якщо товар знайдено — реєструється операція (наприклад, надходження чи відвантаження).
5. Якщо не знайдено — система пропонує додати новий товар.
6. Оператор підтверджує дію, і система фіксує подію в базі.

UML-діаграма послідовності (Sequence Diagram) - демонструє взаємодію між об’єктами:

- RFID-зчитувач передає ID у систему.
- Обробник подій перевіряє відповідність ID у базі.
- База даних повертає інформацію про товар.
- Логіка обліку реєструє операцію.
- Інтерфейс користувача оновлює журнал.

Таблиця 1.1 - Опис ключових сутностей предметної області

Сутність	Атрибути	Опис
Товар (Item)	ID, назва, кількість, локація, RFID-мітка	Одиниця обліку
Операція (Transaction)	ID, тип (надходження/відвантаження), час, товар	Запис у журналі

Користувач (User)	ID, логін, роль	Суб’єкт, що взаємодіє з системою
Зчитувач (Reader)	ID, зона дії, статус	Обладнання для зчитування RFID

1.4 Огляд інформаційних джерел та існуючих рішень

Перед розробкою власного програмного забезпечення доцільно здійснити аналіз наукових, технічних та практичних джерел, а також ознайомитися з існуючими комерційними і відкритими рішеннями в галузі автоматизації складського обліку з використанням RFID. Це дозволяє:

- глибше зрозуміти тенденції у сфері автоматизованих WMS-систем;
- оцінити переваги та недоліки різних підходів;
- обґрунтувати вибір технологій і методів реалізації власної системи.

1.4.1 Аналіз інформаційних джерел

У процесі дослідження було вивчено:

- Наукові статті та дисертації з тематики RFID-технологій, логістичних систем, оптимізації обліку;
- Документацію до апаратних RFID-рішень (наприклад, RC522, PN532, UHF RFID-модулі);
- Опис функціональних можливостей WMS-систем провідних виробників (Oracle WMS, SAP EWM, 1С:Логістика, Odoo Inventory);
- Огляд відкритих рішень на GitHub, що демонструють інтеграцію RFID у Python-додатках;
- Державні та галузеві стандарти з обліку та ідентифікації товарів (ДСТУ, ISO 18000, EPCglobal).

Таблиця 1.2 - Огляд існуючих рішень

Назва системи	Тип ліцензії	Основні можливості	Переваги	Недоліки
SAP Extended Warehouse Management (EWM)	Комерційна	Повна автоматизація логістики, RFID-підтримка	Інтеграція з ERP, масштабованість	Висока вартість, складність налаштування
Odoo Inventory + IoT	Відкрита/комерційна	Підтримка RFID через додаткові модулі	Гнучка архітектура, інтеграція з іншими модулями	Потребує окремої інтеграції з апаратними модулями
RFID WMS Lite (проєкти GitHub)	Open Source	Базова фіксація RFID-міток, веб-інтерфейс	Простота, доступність	Відсутність підтримки багатьох функцій
1С:Підприємство (Склад)	Комерційна	Базовий облік, RFID (опціонально)	Поширення, підтримка	Прив’язаність до середовища 1С

1.4.2 Висновки за результатами огляду

1. Комерційні рішення, як-от SAP EWM, є потужними, але дорогими та складними у впровадженні, що не підходить для малих і середніх підприємств або навчальних проєктів.

2. Відкриті системи, зокрема на базі Odoo чи Python-стеків, забезпечують гнучкість і можливість кастомізації, але часто потребують доопрацювання.

3. Найбільш оптимальним підходом для навчального або демонстраційного проєкту є створення власної системи з нуля на базі Python (Flask) + PostgreSQL із підтримкою інтеграції з простими RFID-зчитувачами (наприклад, RC522).

Таким чином, обраний шлях створення власного програмного забезпечення з відкритими інструментами дозволяє досягти гнучкості, контролю над логікою обліку та можливості майбутнього розширення функціоналу.

1.5 Постановка завдання

Метою даної дипломної роботи є розробка програмного забезпечення, яке реалізує автоматизовану систему управління складом на основі технології RFID. Така система має забезпечити безперервний контроль над переміщенням товарів, мінімізувати втручання людини в облік, а також підвищити точність і ефективність інвентаризації.

Формулювання задачі

На підставі проведеного аналізу предметної області, вимог до системи та огляду існуючих рішень було сформульовано наступні ключові функції, які повинна реалізувати система:

1. Автоматичне зчитування RFID-міток

   Забезпечення прийому ідентифікаторів товарів від RFID-зчитувачів (на базі модулів типу RC522) у режимі реального часу.
2. Реєстрація складських подій

Фіксація в базі даних операцій надходження, переміщення або відвантаження товару із зазначенням дати, часу, типу операції та ідентифікатора товару.

3. Облік і управління залишками

   Автоматичне оновлення даних про залишки на складі та відображення поточного статусу кожного товару.

4. Інвентаризація

   Проведення інвентаризації шляхом сканування RFID-міток і звірки з даними бази.

5. Веб-інтерфейс користувача

   Реалізація інтерфейсу для перегляду товарів, пошуку, фільтрації за параметрами, перегляду історії подій, а також управління налаштуваннями.

6. База даних PostgreSQL

   Проєктування та реалізація структурованої бази даних для зберігання інформації про товари, події та мітки.

7. Контейнеризація системи

   Підготовка середовища розгортання за допомогою Docker для спрощення встановлення та підтримки.

8. Режим тестування

   Забезпечення можливості тестування системи за допомогою емулятора зчитування або симулятора RFID-подій.

Очікувані результати

- Робочий прототип, здатний зчитувати RFID-мітки, фіксувати події та взаємодіяти з базою даних;

- Веб-інтерфейс для користувачів з базовим функціоналом перегляду та обліку;
- Повністю структурована і протестована база даних;

- Інсталяційний пакет або Docker-контейнер із документацією;
- Проведене тестування основних функцій на реальному або симульованому обладнанні.

Критерії успішності розробки

- Виявлення і реєстрація RFID-мітки відбувається не довше ніж за 1 секунду;
- Події коректно фіксуються в БД із гарантією цілісності;
- Інтерфейс забезпечує доступ до даних у режимі реального часу;
- Система працює стабільно без помилок при базовому навантаженні.

1.6 Особливості застосування RFID у складських системах

Системи управління складом (WMS) постійно еволюціонують, переходячи від ручного та напівавтоматичного обліку до повноцінної цифрової автоматизації. Одним із найефективніших підходів до оптимізації обліку є впровадження технології радіочастотної ідентифікації (RFID), що дозволяє безконтактно та оперативно фіксувати переміщення товарів.

1.6.1 **Класифікація систем управління складом**

Залежно від рівня автоматизації та принципів обліку, складські системи поділяють на:

- Ручні системи – облік ведеться вручну, на папері або в Excel-файлах. Схильні до помилок, повільні, не масштабовані.
- Напівавтоматичні системи – використовують штрихкод, ручне сканування, прості WMS-програми.
- Автоматизовані системи з RFID – забезпечують швидке, безконтактне зчитування товарів, інтегровані з WMS або ERP.

Таблиця 1.3 - Переваги RFID над штрихкодами

Параметр	Штрихкод	RFID
Зчитування	Лінійне, візуальний контакт	Безконтактне, одразу кілька міток
Швидкість	1-2 шт./сек	До 1000 міток/сек
Зносостійкість	Чутливий до подряпин/бруду	Працює навіть через упаковку
Обсяг даних	До 20 символів	До 128 байтів і більше
Можливість перезапису	Ні	Так (для перезаписуваних міток)

Таким чином, RFID дозволяє зчитувати одночасно десятки або сотні товарних одиниць без прямої видимості, що критично важливо для швидкої інвентаризації чи фіксації поставок.

1.6.2 Компоненти RFID-систем

1. RFID-мітка (тег)

   Носій даних, закріплений на об’єкті. Буває:

   - *пасивною* (живиться від зчитувача);
   - *активною* (має власне джерело живлення).

2. RFID-зчитувач (рідер)

   Пристрій, що випромінює радіосигнал і приймає дані з міток. З’єднується з обліковою системою через USB, Ethernet або UART.

3. Програмне забезпечення

   Приймає дані від зчитувача, обробляє події, зберігає інформацію в БД, забезпечує інтерфейс для перегляду та аналітики.

1.6.3 Архітектурні підходи до інтеграції RFID у WMS

- Локальна обробка - зчитувач напряму підключений до ПК або контролера, який обробляє події.
- Мережева модель - зчитувачі надсилають дані на сервер через TCP/IP або MQTT.

- Хмарна інтеграція - дані збираються з усіх локацій у централізованій хмарній WMS-системі.

У межах цієї дипломної роботи реалізується локально-серверна модель, де RFID-рідер передає дані у програмний модуль, розроблений на Python (Flask), із записом до БД PostgreSQL і доступом до журналу подій через браузер.

1.6.4 Недоліки RFID і виклики реалізації

- Потребує закупівлі зчитувачів та міток.
- Можливе накладення сигналів (колізії), якщо багато міток у зоні зчитування.
- Обмежена дальність у пасивних рішень (зазвичай до 1 метра).
- Необхідність відповідного програмного забезпечення.

Проте переваги технології суттєво переважають її обмеження, особливо в завданнях автоматизованого обліку.

1.7 Обґрунтування вибору технологічного рішення

На основі аналізу предметної області, технічних вимог і доступних інструментів для реалізації системи управління складом із підтримкою RFID було прийнято рішення щодо використання відкритих, гнучких та економічно доцільних технологій. Основні фактори, що вплинули на вибір архітектури та інструментарію:

1. Вимога до реального часу

RFID-зчитування має фіксуватися практично миттєво — тому вибір пав на асинхронну архітектуру з легковагим веб-фреймворком (Flask), що дозволяє швидко приймати та обробляти події.

2. Відкритість та розширюваність

Система повинна бути модульною, адаптованою до майбутнього підключення нових типів зчитувачів, API, форм фільтрації та аналітики. Використання мови

програмування Python із широкою екосистемою дозволяє швидко розробляти функціонал, тестувати та масштабувати проєкт.

3. Простота розгортання та кросплатформеність

Використання **Docker** забезпечує ізольоване середовище для запуску серверної частини незалежно від ОС, спрощуючи тестування і розгортання на будь-яких машинах.

4. Надійність і масштабованість бази даних

Для зберігання інформації про RFID-мітки, товари, події та користувачів обрано PostgreSQL — потужну об’єктно-реляційну СУБД з підтримкою транзакцій, фільтрів, індексів та розширень.

5. Доступність і низька вартість обладнання

Для тестування та демонстрації використовується дешевий модуль RC522 як зчитувач RFID-міток (13.56 MHz). Він легко інтегрується з контролерами Arduino, Raspberry Pi або напряму з ПК через адаптери.

Таблиця 1.4 – вибір інструментів/технологій для системи управління складом

Компонент	Інструмент / Технологія	Причина вибору
Ядро системи	Python 3	Простота, гнучкість, багата екосистема
Веб-сервер	Flask	Мінімалізм, REST API, швидка розробка
База даних	PostgreSQL	Продуктивність, транзакції, SQL
Зчитування міток	RC522 + pySerial	Низька вартість, простота інтеграції
Інтерфейс	HTML/CSS (Jinja2 шаблони)	Кросплатформеність, зрозумілий доступ
Розгортання	Docker + docker-compose	Уніфікація середовища, масштабованість

Таким чином, комбінація відкритих технологій забезпечує надійну основу для побудови повноцінної системи управління складом, яка не потребує дорогих ліцензій, може бути розгорнута у навчальних або виробничих умовах, легко адаптується під конкретні потреби підприємства й може еволюціонувати з часом.

2. ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Логічна модель даних (ER-діаграма)

Логічна модель даних є основою для побудови бази даних інформаційної системи. Вона дозволяє відобразити сутності предметної області, їх атрибути та взаємозв’язки. Побудова ER-діаграми (Entity–Relationship Diagram) — це критично важливий етап при проєктуванні системи обліку, оскільки саме на її основі реалізуються таблиці в реляційній базі даних.

У межах системи управління складом з підтримкою RFID-технології було визначено такі ключові сутності:

Сутність: Item (Товар)

- id: унікальний ідентифікатор товару
- name: назва товару
- rfid_tag: унікальний код RFID-мітки
- category: категорія товару
- location: місце розташування на складі
- status: активний/списаний
- quantity: кількість одиниць у групі (якщо застосовується)

Сутність: Transaction (Операція)

- id: ідентифікатор операції
- timestamp: дата та час події
- type: тип події (надходження, переміщення, відвантаження)
- item_id: зовнішній ключ до товару
- user_id: зовнішній ключ до користувача
- location_from: звідки переміщено (може бути NULL)
- location_to: куди переміщено

Сутність: User (Користувач)

- id: ідентифікатор користувача

- username: логін
- password_hash: хеш пароля
- role: роль (адміністратор, оператор)

Сутність: RFID Reader (Зчитувач)

- id: ідентифікатор зчитувача
- zone: зона покриття на складі
- device_id: унікальний код пристрою
- status: активний / неактивний

Зв’язки між сутностями

- Item 1:M Transaction - один товар може мати багато операцій.
- User 1:M Transaction - один користувач здійснює багато подій.
- RFID Reader 1:M Transaction - кожна операція може бути пов’язана з конкретним зчитувачем (опціонально).

Нормалізація

Модель приведена до третьої нормальної форми (ЗНФ):

- Відсутнє дублювання інформації;
- Кожен атрибут залежить тільки від первинного ключа;
- Зовнішні ключі забезпечують логічну цілісність між таблицями.

2.2 Діаграма класів та кооперацій

Об’єктно-орієнтований підхід до проєктування програмного забезпечення передбачає представлення системи у вигляді сукупності класів, кожен з яких відповідає за зберігання певних даних та реалізацію логіки обробки. UML-діаграма класів є важливим інструментом, що дозволяє наочно зобразити структуру програмної системи, її модулі, атрибути та методи, а також зв’язки між об’єктами.

У межах системи управління складом з використанням RFID-технології виділено такі основні класи:

Клас: Item (Товар)

Атрибути:

- id: унікальний ідентифікатор
- name: назва товару
- rfid_tag: унікальний RFID-код
- category: категорія
- location: розміщення на складі
- quantity: кількість одиниць

Методи:

- update_location()
- adjust_quantity()

Клас: Transaction (Операція)

Атрибути:

- id
- timestamp
- type (вхід, вихід, переміщення)
- item_id
- user_id
- location_from
- location_to

Методи:

- register()
- rollback()

Клас: User (Користувач)

Атрибути:

- id

- username
- password_hash
- role (оператор / адміністратор)

Методи:

- login()
- view_logs()

Клас: RFIDReader (Зчитувач)

Атрибути:

- id
- zone
- device_id
- status

Методи:

- scan_tag()
- send_data()

2.2.1 Кооперації (Взаємодія між об’єктами)

Для демонстрації логіки взаємодії об’єктів у ключових процесах побудовано кооперації — сценарії міжкласової взаємодії:

Кооперація 1: Зчитування RFID-мітки

RFIDReader викликає scan_tag() → передає RFID-дані до модуля обліку → Item ідентифікується → створюється Transaction

Кооперація 2: Переміщення товару

User ініціює переміщення → система запитує у Item поточне розміщення → створює нову Transaction з type=move

Кооперація 3: Інвентаризація

RFIDReader у ручному режимі сканує зону складу → зчитуються всі активні Item → порівнюються з базою → виводяться розбіжності через інтерфейс

Кооперація 4: Авторизація користувача

User викликає login() → перевіряється password_hash → у разі успіху відкривається доступ до дій відповідно до role

2.3 Діаграма пакетів

Для побудови логічно структурованого, підтримуваного і масштабованого програмного забезпечення важливо на етапі проєктування визначити модульну архітектуру. UML-діаграма пакетів (Package Diagram) дозволяє візуально відобразити поділ системи на окремі функціональні частини (пакети), визначити їх залежності та взаємодії.

У розробленій системі управління складом виділено наступні основні пакети:

Таблиця 2.1 - Опис пакетів

Пакет	Призначення
core	Основна логіка: обробка RFID-даних, ініціація транзакцій, облік товарів
api	REST API маршрути, що надають зовнішній інтерфейс до функцій системи
db	Робота з базою даних PostgreSQL (ORM-моделі, запити, з'єднання)
models	Визначення класів-сутностей: Item, Transaction, User, Reader
rfid	Взаємодія з апаратними модулями RFID (RC522, PN532 тощо)
ui	Веб-інтерфейс користувача: HTML-шаблони, стилі, JS-логіка
utils	Допоміжні модулі: логування, конфігурація, перевірка прав доступу

Взаємодія між пакетами

- api залежить від core, db, models;
- core взаємодіє з rfid, db, models;
- ui працює через api;
- utils використовується всіма іншими пакетами.

Такий поділ дозволяє ізолювати відповідальність кожного модуля:

- логіка зберігається окремо від API;
- модель даних відділена від інтерфейсу;
- драйвери RFID можуть бути легко замінені або оновлені;
- UI-пакет можна адаптувати під мобільний інтерфейс без зміни серверної частини.

2.4 Діаграма компонентів

Діаграма компонентів (Component Diagram) у UML дозволяє візуально описати фізичну структуру програмного забезпечення — тобто, які саме модулі (компоненти) складають систему, які функції виконує кожен з них, та які зв’язки між ними існують. Це особливо важливо для демонстрації архітектури при розгортанні, супроводі або масштабуванні системи.

У контексті системи управління складом з використанням RFID, основні компоненти розподілені за принципами модульності, взаємозамінності та відкритості до інтеграції.

Таблиця 2.2 - Основні компоненти системи

Компонент	Призначення
RFID Controller	Приймає сигнали зчитувача RFID, формує події на основі отриманих тегів
Core Logic	Реалізує бізнес-логіку: облік товару, створення транзакцій, валідація подій
Database (PostgreSQL)	Центральне сховище: таблиці items, transactions, users, readers

REST API (Flask)	Надає зовнішній інтерфейс для веб-клієнта (HTTP-запити)
Web UI (HTML/Jinja)	Інтерфейс користувача для перегляду складу, операцій, звітів
Config & Auth Module	Зберігає параметри підключення, облікові дані користувачів, сесії
Logger	Журналює події системи: доступ, помилки, дії користувачів
Docker Container	Охоплює всі компоненти для запуску в ізольованому середовищі

Опис взаємодії

- RFID Controller → передає подію до Core Logic;
- Core Logic → створює нову транзакцію та звертається до Database;
- REST API → забезпечує доступ зовнішніх запитів до даних і функцій;
- Web UI → звертається до API для отримання інформації;
- Logger → слідкує за кожною дією;
- Docker → інкапсулює все для зручного розгортання.

3. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Система управління інформаційною базою (PostgreSQL)

Для зберігання інформації про товари, RFID-мітки, транзакції та користувачів у системі обліку використовується PostgreSQL — об’єктно-реляційна система управління базами даних з відкритим вихідним кодом, що забезпечує високу продуктивність, масштабованість та підтримку транзакцій.

3.1.1 Причини вибору PostgreSQL

Серед альтернатив PostgreSQL обрано завдяки таким перевагам:

- Висока надійність — підтримка транзакцій ACID, відновлення після збоїв.
- Масштабованість — ефективна робота з великими обсягами даних.
- Гнучкість запитів — розвинуті засоби фільтрації, агрегації, індексів.
- Відкрите ПЗ — безкоштовне використання, активна спільнота.
- Інтеграція з Python — через ORM-бібліотеки (SQLAlchemy, psycopg2).

Структура бази даних

Таблиця 3.1 - Основні таблиці, реалізовані в базі даних

Таблиця	Призначення
items	Зберігає дані про товари: ID, назва, RFID, кількість, місце
transactions	Облік подій: надходження, відвантаження, переміщення
users	Користувачі системи: логін, пароль, роль
readers	Дані про RFID-зчитувачі: ID, зона дії, статус

Усі таблиці мають первинні ключі та зовнішні ключі відповідно до логічної моделі (ER-діаграми, див. п. 2.1). Наприклад, transactions.item_id — зовнішній ключ на items.id.

3.1.2 Інтеграція з програмним забезпеченням

Інтеграція з PostgreSQL здійснюється за допомогою ORM-бібліотеки SQLAlchemy, що дозволяє:

- створювати структуру БД через Python-класи;
- виконувати запити як через SQL, так і в об’єктному стилі;
- легко мігрувати БД між середовищами (наприклад, із SQLite на PostgreSQL).

3.1.3 Контейнеризація PostgreSQL

Для зручного розгортання PostgreSQL використовується Docker, де база працює у власному контейнері, налаштованому через docker-compose.yml. Це спрощує запуск, тестування та ізоляцію середовища.

3.2 Розробка структури бази даних для обліку товарів і подій

На основі логічної моделі даних, побудованої в підрозділі 2.1, було реалізовано фізичну структуру бази даних у PostgreSQL, яка дозволяє забезпечити повний цикл обліку складських одиниць, подій зчитування RFID-міток та взаємодію користувачів із системою.

Основні таблиці бази даних

items — Товари

- id — первинний ключ
- name — назва товару
- rfid_tag — унікальний RFID-код
- category — категорія товару
- location — місце зберігання
- quantity — кількість одиниць
- status — активний / списаний

transactions — Операції

- id — первинний ключ
- timestamp — час події
- type — тип (вхід / вихід / переміщення)
- item_id — зовнішній ключ на items
- user_id — зовнішній ключ на users
- reader_id — зовнішній ключ на readers
- location_from — звідки переміщено
- location_to — куди переміщено

users — Користувачі

- id — первинний ключ
- username — логін
- password_hash — захищений пароль
- role — адміністратор / оператор

readers — Зчитувачі RFID

- id — первинний ключ
- device_id — серійний номер
- zone — зона дії
- status — активний / неактивний

Типи зв’язків між таблицями

- Один користувач може створити багато транзакцій → users 1:M transactions
- Один товар може бути об'єктом багатьох операцій → items 1:M transactions
- Один зчитувач може передавати багато подій → readers 1:M transactions

Нормалізація структури

База даних приведена до третьої нормальної форми (ЗНФ):

- Відсутнє дублювання даних;

- Усі атрибути залежать лише від первинного ключа;
- Використано зовнішні ключі для забезпечення цілісності зв’язків.

Підтримка індексів

Індексація здійснена для полів:

- rfid_tag у таблиці items
- timestamp у таблиці transactions
- username у таблиці users

Це забезпечує прискорений пошук при великому обсязі даних.

3.3 Вибір інструментарію (Python, Flask, Docker, PostgreSQL)

Для реалізації програмного забезпечення системи управління складом із застосуванням RFID було обрано стек технологій, який забезпечує ефективність, відкритість, простоту розгортання та підтримку масштабування. Кожен компонент системи підібрано з урахуванням відповідності вимогам до продуктивності, сумісності та зручності інтеграції.

RFID-модулі (апаратна частина)

Для зчитування RFID-міток обрано модуль RC522 (13.56 MHz), що підтримує пасивні мітки стандарту ISO/IEC 14443. Причини вибору:

- Дешевизна (~2–5$);
- Підтримка бібліотек для Arduino, Raspberry Pi та Python;
- Висока точність зчитування на відстані до 5 см;
- Поширеність серед навчальних та демонстраційних проєктів.

Python 3

Основна мова розробки серверної частини. Переваги:

- Швидкість розробки;
- Велика кількість бібліотек для інтеграції з апаратурою (наприклад, MFRC522, serial, gpiozero);
- Простота написання REST API;
- Інтеграція з PostgreSQL через SQLAlchemy або psycopg2.

Flask

Фреймворк для створення REST API. Обраний через:

- Мінімалістичність і швидкий старт;
- Добру підтримку шаблонізації (Jinja2);
- Підтримку розширень для авторизації, логування, тестування;
- Просту інтеграцію з клієнтським інтерфейсом і базою даних.

PostgreSQL

Потужна СУБД для зберігання інформації про товари, події, користувачів:

- ACID-транзакції;
- Розширена підтримка SQL-запитів;
- Підтримка JSON, масивів, індексів;
- Добра сумісність із Python/ORM.

Docker + Docker Compose

Використовується для контейнеризації проєкту, що дає змогу:

- Розгортати систему на будь-якому сервері/ПК без налаштування середовища;
- Запускати одночасно базу даних, сервер, інтерфейс;
- Забезпечити відтворюваність конфігурації (особливо для демонстрації).

Файли:

- Dockerfile — збирає Flask-додаток;
- docker-compose.yml — запускає PostgreSQL і Flask одночасно.
-

Таблиця 3.2 - Характеристика основних технологічних компонентів системи

Бібліотека	Призначення
SQLAlchemy	ORM для взаємодії з базою даних
Flask-RESTful	Спрощення створення REST API
Flask-CORS	Доступ до API з фронтенду
dotenv	Зберігання конфігурацій у .env файлі
PyMFRC522	Робота з модулем RC522 (на Raspberry Pi)
requests / httpx	Тестування і виклики API

3.4 Алгоритмізація та програмування ключових модулів

На цьому етапі здійснено реалізацію основних функціональних модулів системи, які забезпечують повноцінну роботу програмного забезпечення для автоматизованого обліку товарів на складі з використанням RFID. Алгоритми орієнтовані на ефективність, надійність та простоту інтеграції з апаратним забезпеченням.

1. Модуль зчитування RFID-міток

Призначення: приймати ідентифікатори RFID-міток та ініціювати відповідну подію у системі.

Алгоритм:

1. Очікування зчитування мітки з пристрою RC522.
2. Отримання UID мітки.
3. Пошук товару з таким RFID у базі.
4. Якщо знайдено — створення транзакції.
5. Якщо не знайдено — створення нового запису (якщо дозволено оператором).

2. Модуль обліку та реєстрації транзакцій

Призначення: створювати записи про переміщення товарів.

Типи транзакцій:

- in — надходження;
- out — відвантаження;
- move — переміщення між локаціями.

Логіка:

- Автоматичне визначення дії за подією RFID;
- Запис часу, місця, користувача та об’єкта;
- Збереження у таблицю transactions.

3. Модуль взаємодії з базою даних (ORM)

Призначення: абстрагування SQL-запитів через об’єктно-орієнтовану модель.

Класи SQLAlchemy:

4. Модуль REST API

Призначення: надати доступ до функцій системи через HTTP.

Маршрути:

- GET /api/items — перегляд усіх товарів
- GET /api/transactions — журнал подій
- POST /api/item — додати товар вручну
- GET /api/item/<id> — деталі товару

Flask-фрагмент:

5. Веб-інтерфейс користувача

Призначення: відображення інформації про товари, події, форм для додавання / пошуку.

Інструменти: HTML + Jinja2 + Bootstrap

Функції:

- Таблиця товарів з пошуком
- Історія переміщень
- Додавання товарів вручну

6. Модуль конфігурації та налаштувань

- Використання .env файлу для зберігання:
    - URI бази даних;
    - портів API;
    - шляхів до логів.

7. Docker-контейнери

- Окремий контейнер для:
    - Flask-серверу;
    - PostgreSQL;
- Автоматичне з’єднання між ними;
- Спрощення налаштувань і запуску:

4. ТЕСТУВАННЯ ТА РЕКОМЕНДАЦІЇ ДО ВПРОВАДЖЕННЯ

4.1 Алгоритмізація та програмування ключових модулів

Тестування є важливим етапом розробки програмного забезпечення, який дозволяє перевірити відповідність реалізованого функціоналу технічному завданню, виявити можливі помилки та оцінити надійність системи у реальних умовах. Для системи управління складом з RFID-технологією тестування охоплює як апаратні, так і програмні компоненти.

Мета тестування

- Перевірити коректність зчитування RFID-міток.
- Оцінити правильність запису подій до бази даних.
- Перевірити відповідність вмісту REST API очікуваним даним.
- Оцінити роботу веб-інтерфейсу.
- Виявити збої в роботі при втраті з’єднання з БД або пристроєм.

Методика

Тестування проводилося у два етапи:

1. Функціональне тестування - ручна перевірка кожного сценарію використання.
2. Інтеграційне тестування - одночасна взаємодія кількох компонентів (RFID + БД + API + UI).

Таблиця 4.1 - Тестове середовище

Компонент	Характеристика
ОС	Ubuntu 22.04 / Windows 10
Процесор	Intel Core i5 / AMD Ryzen 5
Оперативна пам’ять	8 ГБ
Python	Версія 3.10
Бібліотеки	Flask, SQLAlchemy, psycopg2, dotenv
СУБД	PostgreSQL 15

RFID-модуль	RC522

Таблиця 4.2 - Приклади тестових сценаріїв

№	Тестовий випадок	Вхідні дані	Очікуваний результат	Статус
1	Зчитування RFID-мітки	Мітка UID = 04A1...	Виведено товар, створено транзакцію	+
2	Відсутня мітка в БД	Нова мітка	Повідомлення "товар не знайдено"	+
3	Перегляд подій через API	GET /api/transactions	JSON-список подій з коректними полями	+
4	Додавання нового товару вручну	POST /api/item	Новий запис у items, збереження в БД	+
5	Відсутнє з'єднання з базою	Зупинено PostgreSQL	Помилка з повідомленням "Connection failed"	+
6	Веб-інтерфейс — перегляд складу	Відкрито сторінку /ui	Відображення таблиці з товарами	+
7	Інвентаризація — ручне сканування	Серія RFID-міток	Таблиця зі статусами: "знайдено / не знайдено"	+

Результати тестування

- Система стабільно реагує на зчитування RFID-міток та створює події.
- Усі API-запити обробляються коректно.
- У разі помилок (відсутність БД, недоступний зчитувач) виводяться повідомлення.
- Веб-інтерфейс працює без збоїв у Chrome, Firefox.
- Контейнеризоване розгортання (docker-compose) відбувається без помилок.

4.2 Визначення технічних вимог до використання системи

Для забезпечення коректної та стабільної роботи програмного забезпечення системи управління складом з RFID-технологією необхідно враховувати як апаратні, так і програмні вимоги. Розроблена система може бути розгорнута як у локальному середовищі (на одному ПК), так і в контейнеризованому або серверному режимі.

Таблиця 4.3 - Мінімальні системні вимоги (для тестування)

Компонент	Мінімальні характеристики
Операційна система	Ubuntu 20.04 / Windows 10 / macOS 11+
Процесор	2-ядерний (Intel Core i3 / AMD Athlon)
Оперативна пам’ять	4 ГБ
Місце на диску	1–2 ГБ (для бази даних, логів, зображень)
Python	Версія 3.8 або вище
База даних	PostgreSQL 12 або вище
RFID-зчитувач	RC522 (через USB/UART/Raspberry Pi GPIO)

Таблиця 4.4 - Рекомендовані вимоги (реальне середовище)

Компонент	Рекомендовані характеристики
Операційна система	Ubuntu 22.04 LTS / Debian / Windows Server 2022
Процесор	4-ядерний (Intel Core i5 / AMD Ryzen 5)
Оперативна пам’ять	8–16 ГБ
Диск	SSD, 10+ ГБ вільного місця
Python	3.10+
Docker	Docker Engine + Docker Compose
Мережевий інтерфейс	Ethernet або Wi-Fi для інтеграції з сервером

Таблиця 4.5 - Програмні залежності

Компонент	Версія / Залежність
Flask	2.2+
SQLAlchemy	ORM для PostgreSQL
psycopg2	PostgreSQL-драйвер для Python
dotenv	Для конфігурації параметрів
RFID-бібліотека	PyMFRC522 (на Raspberry Pi) або аналог
Jinja2	Шаблонізатор для HTML-інтерфейсу
Bootstrap (UI)	CSS-фреймворк для адаптивного дизайну

Мережеві вимоги (опційно)

Якщо система використовується в режимі клієнт-сервер:

- Відкритий порт 5000 для доступу до REST API (Flask);
- Внутрішня або публічна IP-адреса для з'єднання клієнтів з сервером;
- Можливість передачі JSON-запитів між фронтендом і бекендом.

Гнучкість у розгортанні

Система підтримує кілька варіантів запуску:

- Локально, з прямим підключенням зчитувача;
- У Docker, із конфігурацією через docker-compose.yml;
- У локальній мережі, з доступом до інтерфейсу через браузер.

Розроблена система не потребує дорогого чи потужного обладнання для запуску. Її гнучка архітектура дозволяє адаптуватися під різні умови: від демонстраційного використання на ноутбуці до реального складу з кількома зчитувачами та окремим сервером. Мінімальні вимоги роблять її доступною для широкого кола підприємств і навчальних закладів.

4.3 Алгоритмізація та програмування ключових модулів

Для зручного розгортання, перевірки працездатності та демонстрації розробленого програмного забезпечення було сформовано інсталяційний пакет, який включає всі необхідні компоненти, залежності та інструкції. Пакет дозволяє швидко запускати систему як локально, так і в Docker-середовищі без складного налаштування.

Таблиця 4.6 - Структура проєкту

Назва файлу / директорії	Призначення
app/	Основний код серверної частини (Flask)
app/routes.py	REST API: обробка HTTP-запитів
app/models.py	SQLAlchemy-моделі бази даних
app/database.py	Підключення до PostgreSQL
app/rfid_reader.py	Взаємодія з RFID-зчитувачем
static/	Статичні файли (CSS, зображення)
templates/	HTML-шаблони для інтерфейсу
main.py	Точка входу у Flask-додаток
config.py	Конфігураційні параметри
.env	Приховані змінні середовища (паролі, URI БД)
requirements.txt	Список Python-залежностей
Dockerfile	Побудова образу Flask-додатку
docker-compose.yml	Запуск Flask + PostgreSQL у контейнерах
README.md	Інструкція з налаштування та запуску

Python-залежності (requirements.txt)

flask

sqlalchemy

psycopg2-binary

flask-cors

python-dotenv

opencv-python

Інструкція зі встановлення (README.md)

Запуск локально (без Docker):

git clone https://github.com/your-warehouse-rfid-project

cd project

pip install -r requirements.txt

python main.py

Контейнер автоматично створить базу PostgreSQL, підключить Flask-сервер, і зробить API доступним на http://localhost:5000

Формати розповсюдження:

- .zip-архів усього каталогу
- Git-репозиторій
- .tar.gz (опціонально для Linux)
- Образ DockerHub (у майбутньому)

Висновок

У ході виконання бакалаврської кваліфікаційної роботи на тему «Програмне забезпечення системи управління складом із застосуванням технології RFID для автоматизації обліку» було досягнуто поставлену мету — спроєктовано, реалізовано та протестовано прототип інформаційної системи, яка забезпечує ефективний контроль за рухом товарно-матеріальних цінностей із використанням RFID-технологій.

На етапі аналізу предметної області було вивчено сучасні підходи до автоматизації складських процесів, проаналізовано функціональні вимоги, математично описано модель обліку, а також обґрунтовано вибір інструментів та архітектури.

У процесі проєктування системи було створено:

- логічну ER-модель бази даних,
- UML-діаграми класів, кооперацій, пакетів і компонентів,
- специфікацію функціональних сценаріїв і структуру модулів.

На етапі реалізації:

- створено повноцінну серверну частину на Flask;
- реалізовано API для взаємодії з клієнтом;
- інтегровано підтримку RFID-зчитувача;
- розроблено веб-інтерфейс для відображення облікових даних;
- використано PostgreSQL для надійного збереження інформації;
- впроваджено Docker-контейнери для спрощеного розгортання.

Проведене тестування підтвердило працездатність ключових модулів: зчитування міток, реєстрація транзакцій, обробка API-запитів, робота з інтерфейсом. Система стабільно працює в тестовому середовищі, легко масштабується та може бути адаптована для використання в реальних умовах.

Основні досягнення:

- Автоматизовано зчитування RFID-міток та облік подій;
- Забезпечено інтеграцію апаратного і програмного забезпечення;

- Реалізовано повноцінну архітектуру клієнт-серверної взаємодії;
- Побудовано гнучку, відкриту систему на основі безкоштовних технологій.

Перспективи подальшого розвитку:

- Розширення підтримки різних типів RFID-пристроїв;
- Мультискладська модель обліку;
- Інтеграція з мобільним додатком;
- Розширення функціональності для аналітики та звітності;
- Хмарне розгортання з підтримкою масштабування.

Додаток А

app/database.py

```
import os
from sqlalchemy import create_engine
from dotenv import load_dotenv

load_dotenv()
DATABASE_URL = os.getenv("DATABASE_URL")

engine = create_engine(DATABASE_URL, echo=True)
```

app/models.py

```
from sqlalchemy import Column, Integer, String, ForeignKey, DateTime
from sqlalchemy.orm import relationship, declarative_base
from datetime import datetime

Base = declarative_base()

class Item(Base):
    __tablename__ = 'items'
    id = Column(Integer, primary_key=True)
    name = Column(String)
    rfid_tag = Column(String, unique=True)
    category = Column(String)
    location = Column(String)
    quantity = Column(Integer)
    status = Column(String)
```

```
class User(Base):
    __tablename__ = 'users'
    id = Column(Integer, primary_key=True)
    username = Column(String, unique=True)
    password_hash = Column(String)
    role = Column(String)

class Reader(Base):
    __tablename__ = 'readers'
    id = Column(Integer, primary_key=True)
    device_id = Column(String, unique=True)
    zone = Column(String)
    status = Column(String)

class Transaction(Base):
    __tablename__ = 'transactions'
    id = Column(Integer, primary_key=True)
    timestamp = Column(DateTime, default=datetime.utcnow)
    type = Column(String)
    item_id = Column(Integer, ForeignKey('items.id'))
    user_id = Column(Integer, ForeignKey('users.id'))
    reader_id = Column(Integer, ForeignKey('readers.id'))
    location_from = Column(String)
    location_to = Column(String)
```

app/rfid_reader.py

```
def emulate_rfid_scan(uid):
    print(f"[EMULATOR] Зчитано RFID: {uid}")
```

```
    return uid

def process_rfid_event(uid):
    from app.models import Item, Transaction
    from app.database import engine
    from sqlalchemy.orm import sessionmaker
    Session = sessionmaker(bind=engine)
    session = Session()

    item = session.query(Item).filter_by(rfid_tag=uid).first()
    if item:
        txn = Transaction(type='scan', item_id=item.id, user_id=1)
        session.add(txn)
        session.commit()
        print("[INFO] Транзакція записана")
    else:
        print("[WARN] Товар не знайдено")
```

app/routes.py

```
from flask import Flask, jsonify, request
from sqlalchemy.orm import sessionmaker
from app.database import engine
from app.models import Item, Transaction, Base

app = Flask(__name__)
Session = sessionmaker(bind=engine)

@app.route("/api/items", methods=["GET"])
def get_items():
```

```
    session = Session()
    items = session.query(Item).all()
    return jsonify([{"id": i.id, "name": i.name, "rfid_tag": i.rfid_tag} for i in items])

@app.route("/api/item", methods=["POST"])
def add_item():
    data = request.get_json()
    session = Session()
    new_item = Item(**data)
    session.add(new_item)
    session.commit()
    return jsonify({"status": "created"}), 201
```

templates/ index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Складський облік RFID</title>
</head>
<body>
    <h1>Товари на складі</h1>
    <div id="items"></div>

    <script>
        fetch("/api/items").then(res => res.json()).then(data => {
            document.getElementById("items").innerHTML = data.map(
                item => `<p>${item.name} (${item.rfid_tag})</p>`
            ).join('');
        });
```

```
  </script>
</body>
</html>
```

.env

```
DATABASE_URL=postgresql://postgres:password@db:5432/warehouse
FLASK_ENV=development
SECRET_KEY=supersecretkey
```

docker-compose.yml

```
version: '3.8'
services:
 web:
  build: .
  ports:
   - "5000:5000"
  env_file:
   - .env
  depends_on:
   - db
 db:
  image: postgres:15
  restart: always
  environment:
   POSTGRES_USER: postgres
   POSTGRES_PASSWORD: password
   POSTGRES_DB: warehouse
  volumes:
```

```
      - pgdata:/var/lib/postgresql/data
volumes:
  pgdata:
```

Dockerfile

```
FROM python:3.10
WORKDIR /app
COPY . .
RUN pip install -r requirements.txt
CMD ["python", "main.py"]
```

main.py

```
from app.routes import app

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

requirements.txt

```
flask
sqlalchemy
psycopg2-binary
flask-cors
python-dotenv
```

Додаток Б

Додаток В

Додаток Г

Додаток Ґ

Інструкція з встановлення та запуску системи обліку з RFID

Встановлення
Крок 1. Встановити залежності:
pip install ultralytics opencv-python flask sqlalchemy

Крок 2. Створити базу даних:
python3 -c "from app.database import init_db; init_db()"

Крок 3. Запустити детекцію:
python3 main.py

Крок 4. Запустити API:
python3 app/rfid_reader.py

Структура проєкту:

Файл / Директорія	Призначення
main.py	Точка входу в застосунок Flask
app/routes.py	REST API: обробка HTTP-запитів
app/models.py	SQLAlchemy-моделі для таблиць: товарів, операцій, користувачів
app/database.py	Логіка підключення до PostgreSQL та ініціалізація схеми
app/rfid_reader.py	Отримання UID з RFID-зчитувача (реального або симульованого)
templates/	HTML-шаблони веб-інтерфейсу
static/	Статичні файли: стилі, іконки, зображення
.env	Файл з параметрами середовища (паролі, URI бази тощо)
requirements.txt	Список Python-бібліотек, необхідних для запуску
Dockerfile	Скрипт збірки контейнера для Flask-додатку
docker-compose.yml	Контейнеризований запуск PostgreSQL та Flask
README.md	Документація з налаштування системи

Примітка

У разі використання Docker-середовища, запуск системи виконується командою:
docker-compose up --build
Це автоматично піднімає і серверну частину, і базу даних.

Список використаних джерел

1. ISO/IEC 18000-6:2013. Information technology – Radio frequency identification for item management – Part 6: Parameters for air interface communications at 860 MHz to 960 MHz.
2. Dobkin D. M. The RF in RFID: UHF RFID in Practice. — 2nd ed. — Newnes, 2012. — 504 p.
3. Постригань І. Г. Основи логістики. — Київ: Центр навчальної літератури, 2015. — 340 с.
4. Чайковський Ю. Ю., Коваль І. С. Системи управління складом (WMS): концепції, моделі, впровадження. — Львів: Видавництво ЛНУ, 2020. — 196 с.
5. PostgreSQL documentation — https://www.postgresql.org/docs/
6. Flask documentation — https://flask.palletsprojects.com/
7. SQLAlchemy ORM documentation — https://docs.sqlalchemy.org/
8. MFRC522 RFID Module User Manual — NXP Semiconductors. https://www.nxp.com/docs/en/data-sheet/MFRC522.pdf
9. Docker documentation — https://docs.docker.com/
10. Шевчук В. Я. Проєктування баз даних: Навчальний посібник. — Київ: КНУ, 2021. — 248 с.
11. Грейс Т., Хеллман Р. Python і Flask для розробників веб-сервісів. — Харків: Фабула, 2020. — 272 с.
12. RFID Journal — https://www.rfidjournal.com
13. Ultralytics YOLOv5 Documentation — https://docs.ultralytics.com/
14. RFC 7519 — JSON Web Token (JWT). https://datatracker.ietf.org/doc/html/rfc7519
15. Nhfjhfgjnjkgkfghb mjcbgbbgfghkkdmdkl