

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____ Комп'ютерних наук
(повна назва)
Кафедра _____ Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____ перший (бакалаврський)

Інтелектуальна система автоматизованого обліку та інвентаризації
товарно-матеріальних цінностей на базі алгоритмів комп'ютерного зору
(тема)

Виконав:
здобувач _____ четвертого _____ року навчання,
групи _____ ІТШ-22-1

Владіслав ОНІЩЕНКО
(власне ім'я, прізвище)

Спеціальність 122 Комп'ютерні науки

(код і повна назва спеціальності)

Тип програми _____ освітньо-професійна

Освітня програма _____ Штучний інтелект

(повна назва освітньої програми)

Керівник _____ ас. Дмитро МАЛЄСВ
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ

(підпис)

Лариса ЧАЛА
(власне ім'я, прізвище)

2026 р.

Харківський національний університет радіоелектроніки

Факультет _____ Комп'ютерних наук _____

Кафедра _____ Штучного інтелекту _____

Рівень вищої освіти _____ перший (бакалаврський) _____

Спеціальність _____ 122 Комп'ютерні науки _____
(код і повна назва)

Тип програми _____ освітньо-професійна _____

Освітня програма _____ Штучний інтелект _____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

« _____ » _____ 20 ____ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві _____ Оніщенку Владіславу Андрійовичу _____
(прізвище, ім'я, по батькові)

1. Тема роботи _____ Інтелектуальна система автоматизованого обліку та інвентаризації
товарно-матеріальних цінностей на базі алгоритмів комп'ютерного зору _____

затверджена наказом університету від 5 червня 2026 р. № 488Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 8 липня 2026 р.

3. Вихідні дані до роботи _____ Цифрові зображення товарно-матеріальних цінностей, лінійні
штрих-коди та матричні QR-коди товарів, нормалізовані файли розмітки у форматі YOLO
(.txt), архітектури та ваги моделей штучного інтелекту (YOLO11n, YOLO-World). _____

4. Перелік питань, що потрібно опрацювати в роботі _____

1) Аналіз предметної галузі та постановка задачі _____

2) Архітектура системи _____

3) Програмна реалізація моделей штучного інтелекту _____

4) Інтерфейс користувача та тестування моделей _____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / термін виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	08.06.2026	виконано
2	Аналіз предметної галузі	09.06.2026	виконано
3	Аналіз метрик та критеріїв оцінювання моделей комп'ютерного зору	12.06.2026	виконано
4	Аналіз підходів вирішення задачі детекції	15.06.2026	виконано
5	Експериментальне моделювання та навчання моделі	20.06.2026	виконано
6	Написання пояснювальної записки	25.06.2026	виконано
7	Перевірка на академічний плагіат	26.06. 2026	виконано
8	Нормоконтроль	26.06. 2026	виконано
9	Підготовка презентації та доповіді	29.06. 2026	виконано
10	Попередній захист	30.06. 2026	виконано
11	Рецензування	01.07.2026	виконано
12	Захист перед ЕК	08.07.2026	виконано

Дата видачі завдання 8 червня 2026 р.

Здобувач _____
(підпис)

Керівник роботи _____
(підпис)

ас. Дмитро МАЛЄЄВ _____
(посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 68 с., 7 рис., 1 дод., 18 джерел.

АВТОМАТИЗАЦІЯ ІНВЕНТАРИЗАЦІЇ, ДЕТЕКЦІЯ ОБ'ЄКТІВ, КОМП'ЮТЕРНИЙ ЗІР, РЕЛЯЦІЙНІ БАЗИ ДАНИХ, РОЗПІЗНАВАННЯ ШТРИХ-КОДІВ, СКЛАДСЬКИЙ ОБЛІК, ТОВАРНО-МАТЕРІАЛЬНІ ЦІННОСТІ, YOLO.

Об'єкт дослідження – процес обліку, інвентаризації та сортування товарно-матеріальних цінностей на підприємствах сучасної логістики та складського господарства.

Предмет дослідження – методи та алгоритми комп'ютерного зору для автоматичного виявлення, розпізнавання, трекінгу об'єктів та зчитування їхніх ідентифікаційних кодів (штрих-кодів/QR-кодів) для інтеграції з базами даних.

Мета роботи – розробити концепцію та реалізувати прототип інтелектуальної системи автоматизованого обліку ТМЦ, що дозволить у реальному часі здійснювати детекцію товарів, розпізнавати їхній тип, зчитувати маркування та автоматично оновлювати інформацію про залишки в SQL-базі даних під час прийому, сортування або продажу.

Методи дослідження – порівняльний аналіз сучасних архітектур нейронних мереж для детекції об'єктів (зокрема моделей сімейства YOLO), аналіз методів обробки цифрових зображень та алгоритмів декодування графічного маркування, а також проектування реляційних баз даних.

У роботі розглянуто сучасні виклики автоматизації складської логістики та обліку товарно-матеріальних цінностей в умовах хаотичного надходження вантажів, вивчено існуючі програмні застосунки для мобільного зчитування даних, їхні переваги та недоліки.

ABSTRACT

Bachelor's thesis contains: 68 pp., 7 fig., 1 ann., 18 references.

AUTOMATION OF INVENTORY, BARCODE RECOGNITION, COMPUTER VISION, MATERIAL ASSETS, OBJECT DETECTION, RELATIONAL DATABASES, WAREHOUSE ACCOUNTING, YOLO.

The object of study is the process of accounting, inventorying, and sorting of goods and materials at enterprises of modern logistics and warehouse management.

The subject of research is methods and algorithms of computer vision for automatic detection, recognition, tracking of objects, and reading their identification codes (barcodes/QR codes) for integration with databases.

The purpose of the work is to develop a concept and implement a prototype of an intelligent automated inventory accounting system, which will allow real-time detection of goods, recognition of their type, reading of labels, and automatic updating of stock information in an SQL database during receipt, sorting, or sale.

Research methods – comparative analysis of modern neural network architectures for object detection (in particular, models of the YOLO family), analysis of digital image processing methods and graphic tagging decoding algorithms, as well as the design of relational databases.

The thesis examines current challenges in the automation of warehouse logistics and inventory accounting of goods and materials under conditions of chaotic cargo arrivals, and studies existing software applications for mobile data reading, their advantages and disadvantages.

ЗМІСТ

Вступ.....	10
1 Аналіз предметної галузі та постановка задачі.....	12
1.1 Опис предметної галузі	12
1.2 Штучний інтелект у системах управління складськими запасами	14
1.3 Мета роботи та постановка задачі.....	16
2 Архітектура системи.....	21
2.1 Трирівнева клієнт-серверна модель (UI – логіка – моделі)	21
2.2 Модель для виявлення продукції на складі.....	24
2.3 Модель для автоматичного маркування даних	27
2.4 Конвеєр автоматичного донавчання моделі.....	30
2.5 Потік даних у системі та інвентаризаційному менеджменті.....	33
2.6 Структура даних та зберігання	37
3 Програмна реалізація моделей штучного інтелекту.....	39
3.1 Використані технології та компоненти.....	39
3.1.1 Інструменти Python (FastAPI, Ultralytics YOLO, OpenCV).....	39
3.1.2 Інструменти логічного рівня (Java, Spring Boot, Spring Data JPA)	41
3.2 Модуль класичного сканування штрих-кодів (PyZbar)	44
3.2.1 Архітектура REST API та структура обміну даними	44
3.3 Модуль інтелектуального автоматичного маркування товарів	47
3.4 Навчання та оцінка якості моделі детекції товарів	49
3.4.1 Аналіз загальних та покласових метрик ефективності	49
4 Інтерфейс користувача та тестування моделей.....	52
4.1 Інтерфейс користувача (UI)	52
4.2 Інтеграція моделей у загальний конвеєр складського обліку	55
4.3. Експериментальне тестування програмного комплексу та верифікація моделей комп'ютерного зору	56
Висновки	59

Перелік джерел посилення	62
Додаток А Код класу інтелектуальної розмітки YOLOWorldLabeler	65
Додаток Б Відомість кваліфікаційної роботи.....	68

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ТМЦ – товарно-матеріальні цінності;

ТСД – термінал збору даних;

СКБД – система керування базами даних;

AI – Artificial Intelligence – штучний інтелект;

API – Application Programming Interface – інтерфейс прикладного програмування;

CSS3 – Cascading Style Sheets 3 – каскадні таблиці стилів третього покоління;

DOM – Document Object Model – об'єктна модель документа для маніпуляції елементами веб-сторінки;

ERP – Enterprise Resource Planning – система планування та управління ресурсами підприємства;

HTML5 – HyperText Markup Language 5 – мова розмітки гіпертексту п'ятого покоління;

HTTP – HyperText Transfer Protocol – протокол передачі гіпертексту;

JPA – Java Persistence API – специфікація збереження об'єктів Java у реляційних базах даних;

JSONB – JavaScript Object Notation Binary – бінарний тип даних для збереження документів JSON у СКБД PostgreSQL;

mAP – Mean Average Precision – інтегральна метрика середньої точності, що використовується для оцінки якості моделей детекції об'єктів;

MLOps – Machine Learning Operations – набір практик та методологій для автоматизації й життєвого циклу розгортання моделей машинного навчання;

REST – Representational State Transfer – архітектурний стиль взаємодії клієнт-серверних компонентів через протокол HTTP;

SKU – Stock Keeping Unit – унікальний ідентифікатор товарної позиції (артикул);

UI – User Interface – інтерфейс користувача;

WMS – Warehouse Management System – система автоматизації та управління складськими процесами;

YOLO – You Only Look Once – сімейство сучасних нейромережових архітектур глибокого навчання для детекції та класифікації об'єктів у реальному часі.

ВСТУП

Сучасний розвиток світової та вітчизняної логістики вимагає від підприємств високої швидкості обробки вантажів, мінімізації витрат та жорсткого контролю над товарно-матеріальними цінностями (ТМЦ). Традиційні підходи до ведення складського обліку, що базуються на ручному скануванні або паперовому документообігу, дедалі частіше стають причиною виникнення «вузьких місць» у логістичних ланцюгах [1]. Через вплив людського фактора гостро постають проблеми пересортиці, затримок під час приймання та відвантаження товарів, а також низької швидкості проведення інвентаризації, що в умовах динамічного ринку призводить до фінансових втрат.

Ефективне управління складом потребує врахування багатьох факторів: від специфіки геометрії й типів пакувань товарів до оперативного зчитування маркування (штрих-кодів та QR-кодів) безпосередньо в процесі їхнього руху конвеєром. Традиційні методи контролю є трудомісткими, повільними та часто суб'єктивними, особливо в умовах великих обсягів різнотипної продукції. Як наслідок, виникає потреба у докорінній зміні підходів до моніторингу руху ТМЦ та автоматизації рутинних складських операцій.

На сьогодні існує значна кількість систем управління складом (WMS), проте більшість із них вимагає безпосередньої участі людини для фіксації кожної одиниці товару або обмежується стандартним ручним скануванням. На вітчизняному ринку практично відсутні доступні та гнучкі інтелектуальні рішення, здатні в реальному часі за допомогою однієї камери автоматично розпізнавати категорію об'єкта, відстежувати його переміщення, зчитувати графічне маркування та миттєво синхронізувати ці дані з реляційними базами даних без зупинки логістичного потоку.

У зв'язку з цим особливо актуальним є створення інтелектуальної системи автоматизованого обліку та інвентаризації на базі алгоритмів

комп'ютерного зору. Розробка та впровадження такої системи дозволить не лише підвищити пропускну здатність складських комплексів та оптимізувати використання людських ресурсів, але й забезпечить прозорий, безпомилковий контроль за залишками продукції.

У рамках цієї роботи досліджується предметна галузь, пов'язана з автоматизацією складської логістики, визначаються основні виклики під час розпізнавання товарів на складських приміщеннях. Особлива увага приділяється інноваційним аспектам системи – інтеграції нейромережевих моделей детекції об'єктів із модулями оптичного зчитування ідентифікаційних кодів PyZbar.

Розроблений програмний продукт має високий потенціал для практичного застосування. Система є універсальною та масштабованою, завдяки чому її можуть використовувати не лише комерційні підприємства чи великі логістичні хаби, але й волонтерські організації та гуманітарні штаби. Впровадження системи волонтерами дозволить значно прискорити сортування, облік та розподіл великих обсягів гуманітарної допомоги, забезпечить прозорість звітності та допоможе оперативно закривати потреби постраждалого населення за умов обмеженого людського ресурсу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної галузі

Зі зміною глобальних та внутрішніх економічних умов в Україні, починаючи з 2022 року, логістична інфраструктура зазнала критичних трансформацій. За даними профільних аналітичних звітів та досліджень ринку комерційної нерухомості, сотні тисяч квадратних метрів сучасних складських площ класу «А» та «В» були або фізично пошкоджені, або заблоковані через зміну логістичних маршрутів [2]. Це спровокувало колосальне навантаження на збережені та новостворені хаби, особливо в центральних та західних регіонах країни. Крім суто комерційних збитків, виникла гостра потреба в управлінні величезними потоками товарно-матеріальних цінностей гуманітарного призначення. Великі обсяги медикаментів, харчових продуктів, засобів гігієни та обладнання щодня надходять від міжнародних донорів, утворюючи критичну масу об'єктів для обліку.

Перед логістичними центрами та волонтерськими організаціями постало пріоритетне завдання – забезпечити безперебійний, точний та швидкий облік ТМЦ як в умовах кризового стану, так і під час подальшої оптимізації процесів розподілу. З огляду на колосальні обсяги вантажів, обмежені фінансові ресурси та хронічний брак кваліфікованого персоналу на складах, стає очевидною неможливість швидкого оброблення всього потоку товарів за допомогою традиційних засобів. Потреба у чіткому сортуванні та визначенні черговості відвантаження товарів першої необхідності набуває критичного значення.

На сьогоднішній день для розв'язання цієї проблеми впроваджуються різноманітні системи управління складами. Проте класичний процес інвентаризації та обліку, як правило, складається з багатьох тривалих етапів: ручне сканування штрих-кодів кожної палети чи коробки, візуальна

перевірка цілісності пакування, ручне внесення даних до застарілих облікових таблиць, погодження документів та фіксація залишків фахівцями [4]. Цей ланцюжок займає значний час, критично залежить від людського фактора, супроводжується помилками через пересортицю та затримується через паперову бюрократію.

Важливим недоліком існуючих підходів до розподілу та сортування ТМЦ (особливо в гуманітарному та волонтерському секторах) є відсутність чітких автоматизованих критеріїв оцінки пріоритетності обробки вантажів. Під час приймання різнотипної продукції виникає потреба у дотриманні балансу між категорією товару, терміном його придатності та критичністю потреби в ньому на місцях. Суттєвим викликом є також відсутність прозорості: кінцеві споживачі, донори чи керівництво організацій часто не мають доступу до реальних даних про те, яка кількість товарів зараз перебуває на складі, чому певні позиції затримуються на етапі сортування або за якими критеріями формуються черги на відвантаження. Наприклад, під час масового надходження допомоги через обмеженість людського ресурсу автоматично обробляються лише великі однотипні партії, тоді як дрібні, але критично важливі вантажі можуть годинами чекати на ручне сканування, що викликає об'єктивні запитання щодо ефективності розподілу.

Таким чином, процес класифікації, сортування та обліку споруд на логістичному майданчику перетворюється на складну багатofакторну задачу. Її вирішення вимагає одночасного врахування типу пакування, візуальних характеристик об'єкта, маркування та його логістичного статусу. Отже, виникає гостра потреба у створенні інтелектуальної системи, що дозволила б за допомогою алгоритмів комп'ютерного зору автоматизувати процеси детекції товарів, розпізнавання їхнього типу, зчитування ідентифікаційних штрих-/QR-кодів та миттєвого оновлення баз даних. Впровадження подібних інтелектуальних систем дозволить зробити

складський облік надійним, швидким, а процес розподілу ТМЦ – повністю об'єктивним та прозорим для всіх учасників процесу.

1.2 Штучний інтелект у системах управління складськими запасами

Протягом останнього десятиліття спостерігається інтенсивна інтеграція технологій штучного інтелекту у логістичні системи та процеси управління складськими запасами й товарно-матеріальними цінностями. Впровадження інтелектуальних алгоритмів дозволяє реорганізувати традиційні підходи до проведення інвентаризації, забезпечуючи безперервний моніторинг наявності продукції на стелажах у режимі реального часу, мінімізацію похибок, зумовлених людським фактором, та предиктивне моделювання динаміки попиту з метою запобігання дефіциту або надлишку номенклатури.

Одним із провідних напрямків оптимізації є автоматизований аналіз візуальних даних, що надходять із камер спостереження, мобільних пристроїв персоналу або спеціалізованих терміналів збору даних. Візуальний моніторинг забезпечує оперативну верифікацію відповідності фактичної кількості товарно-матеріальних цінностей даним транзакційних баз даних, контроль коректності просторового розміщення об'єктів та ідентифікацію пересортиці. Використання методів комп'ютерного зору трансформує рутинні операції комірників у високоавтоматизовані процеси з покроковим трекінгом руху продукції.

Центральним елементом підсистеми розпізнавання є архітектури глибоких нейронних мереж для детекції об'єктів, які забезпечують локалізацію та класифікацію елементів на зображенні. Сучасні однопрохідні детектори, зокрема останні покоління лінійки YOLO, такі як YOLOv8 та YOLO11, здійснюють інференс у режимі реального часу з високими показниками точності. Дані моделі здатні одночасно локалізувати десятки різномірних об'єктів у межах одного кадру, визначаючи просторові

координати обмежувальних рамок та обчислюючи вектори ймовірностей приналежності до заданих класів.

Для забезпечення стабільного функціонування системи в динамічному складському середовищі обчислювальні процеси інтегруються у гнучку мікросервісну архітектуру. При цьому серверна частина на базі Java Spring Boot відповідає за збереження бізнес-логіки, транзакційний облік та взаємодію з реляційними базами даних, тоді як обчислювальні сервіси на базі Python FastAPI забезпечують виконання ресурсомістких завдань комп'ютерного зору та інференсу моделей глибокого навчання. На першому етапі обробки кадру застосовуються класичні методи детекції, зокрема алгоритми OpenCV та PyZbar, для розпізнавання лінійних і матричних штрих-кодів, а у випадку незадовільних умов освітлення, пошкодження етикетки або виявлення нової номенклатурної позиції система автоматично задіює нейромережевий конвеєр.

Важливим аспектом розробки є реалізація концепції Data-Centric AI, в межах якої автоматизуються процеси підготовки даних та функціонування MLOps-конвеєрів. Інтеграція нейромереж із відкритим словником, таких як YOLO-World, дозволяє побудувати конвеєр автоматичного маркування. У разі неможливості класифікації об'єкта локальною моделлю YOLO11 оператор ініціює сканування штрих-коду та фіксацію серії знімків. Використовуючи текстове найменування товару з реляційної бази даних як семантичну підказку, модель YOLO-World здійснює просторову локалізацію об'єкта та генерує відповідні файли анотацій у форматі .txt. Акумуляовані графічні масиви зберігаються в об'єктному сховищі MinIO, що уможливорює запуск планового циклу донавчання локальної моделі за розкладом без залучення сторонніх фахівців з розмітки.

Ефективність запропонованого підходу є найбільш критичною на великих логістичних комплексах із високою інтенсивністю товарообігу, де ручний аудит супроводжується значними трудовитратами та ризиками

виникнення помилок через втому персоналу. Застосування мобільних додатків із підтримкою комп'ютерного зору дозволяє виконувати поточну інвентаризацію шляхом безперервного сканування стелажів, автоматично оновлюючи записи в системі керування базами даних та візуалізуючи проблемні зони безпосередньо на екрані пристрою оператора.

Попри високу ефективність, функціонування системи обмежене низкою дестабілізуючих факторів, серед яких ключове місце посідають оптичні завади, такі як недостатній рівень освітленості складських приміщень чи сильні бліки від пакувальної плівки, що нівелюється за допомогою попередньої матричної обробки OpenCV та нормалізації гістограми кадру. Іншим суттєвим обмеженням є оклюзія, тобто взаємне перекриття об'єктів у глибині складських полиць, для усунення якого впроваджується багатокутний трекінг та обов'язкова фіксація об'єктів з кількох ракурсів. Крім того, наявність візуальної конвергенції у вигляді високої схожості пакувань різних брендів вимагає впровадження комбінованого класифікатора з урахуванням семантичних маркерів тексту, а апаратна гетерогенність камер ТСД та смартфонів компенсується аугментацією навчальної вибірки шляхом штучного додавання розмиття, шуму та зміни роздільної здатності зображень.

Таким чином, розвиток сучасних логістичних систем спрямований на створення інтелектуальних комплексів обліку, здатних до самонавчання в процесі експлуатації. Впровадження автоматизованих конвеєрів розпізнавання та MLOps-контролю мінімізує вплив людського фактора, оптимізує операційні витрати підприємства та забезпечує високу прозорість, швидкість і точність управління складськими запасами.

1.3 Мета роботи та постановка задачі

Ця інтелектуальна система має за мету допомогти комірникам, логістам та керівникам складських комплексів значно прискорити й

спростити процес проведення інвентаризації товарно-матеріальних цінностей (ТМЦ), мінімізувати вплив людського фактора та автоматизувати облік навіть у разі надходження абсолютно нових партій вантажів. Додаток має бути простим, зрозумілим і дозволяти кожному – від звичайного працівника складу до начальника логістики – легко проаналізувати поточний стан залишків: навести камеру на стелаж, миттєво розпізнати наявні товари, звірити їх із базою даних та автоматично зафіксувати результати ревізії. Мета – створити інструмент, який перетворить виснажливі багатогодинні перевірки на швидкий цифровий процес, а у майбутньому стане невіддільною частиною великих корпоративних систем управління складом (WMS).

Користувач зможе використовувати мобільний пристрій або термінал збору даних (ТСД) з інтегрованою камерою для сканування полиць із товарами. Програма за допомогою алгоритмів комп'ютерного зору самостійно детектує об'єкти, підрахує їхню кількість і відобразить інформацію про кожну позицію, включаючи її характеристики та специфічні умови зберігання (якщо товар потребує особливого температурного режиму чи конкретного місця розміщення). Усе це буде подано за допомогою: зручний інтерфейс із підсвічуванням знайдених предметів, цифрові звіти про надлишки чи нестачу та чіткі підказки щодо правильного розташування ТМЦ на складі.

Програма розробляється на базі гнучкої мікросервісної архітектури, що дозволяє легко масштабувати її, покращувати окремі модулі та підключати до вже існуючих ERP-систем підприємства. Вона виконує три головні завдання: автоматично ідентифікувати товари на складах, оперативно коригувати записи у базі даних та самостійно оновлювати власну базу знань без залучення сторонніх розробників.

Для досягнення поставленої мети було визначено такі задачі:

- розробка архітектури інтелектуальної системи автоматизованого обліку на основі мікросервісного підходу із розподілом обов'язків між сервісом бізнес-логіки та модулем штучного інтелекту;
- дослідження та впровадження комбінованого методу ідентифікації ТМЦ, який поєднує класичне зчитування штрих-кодів за допомогою OpenCV та PyZbar із сучасними алгоритмами детекції об'єктів;
- створення автоматизованого конвеєра розмітки даних (auto-labeling) на основі Zero-Shot моделі YOLO-World для розпізнавання нових товарів за текстовим описом із бази даних;
- реалізація механізму щотижневого донавчання (Fine-tuning) локальної моделі детекції YOLO11 для підвищення точності розпізнавання специфічного асортименту складу в реальних умовах;
- тестування розробленого прототипу системи, оцінка метрик точності моделей (mAP, Precision, Recall), аналіз швидкодії мікросервісів та визначення шляхів подальшого вдосконалення системи.

Перша частина системи – це мікросервіс бізнес-логіки та управління даними, реалізований за допомогою фреймворку Java Spring Boot. Він виступає ядром системи, яке координує всі процеси: зберігає інформацію про номенклатуру товарів, поточну кількість одиниць на складах та зони їхнього розміщення. Цей сервіс обробляє класичні CRUD-запити, керує користувачами та забезпечує транзакційність і надійність збереження даних. Для збереження інформації про товари, історію інвентаризацій та логів використовується потужна реляційна база даних PostgreSQL. Завдяки Spring Boot система здатна швидко обробляти сотні запитів від багатьох працівників одночасно, забезпечуючи стабільну координацію складських процесів.

Друга частина – це інтелектуальний модуль комп'ютерного зору, створений на мові програмування Python за допомогою швидкого асинхронного фреймворку FastAPI. Ця частина системи відповідає за первинний аналіз зображень. Коли працівник робить фото або транслює

відеопотік, модуль за допомогою бібліотек OpenCV та PyZbar намагається локалізувати та дешифрувати лінійні штрих-коди. Якщо штрих-код успішно розпізнано, Python-сервіс робить REST-запит до Java-сервісу, миттєво підтягуючи назву товару (рисунок 1.1). Якщо ж об'єкт не має коду або він пошкоджений, система активує нейромережеві алгоритми для візуальної детекції.



Рисунок 1.1 – Результат зчитування штрих-коду та QR-коду за допомогою бібліотек OpenCV і PyZbar

Третя частина – це конвеєр автоматичного маркування та донавчання моделі. Для вирішення проблеми появи нових товарів інтегровано модель з відкритим словником YOLO-World. Вона виступає в ролі «вчителя»: отримуючи текстову назву невідомого товару від Spring Boot як підказку (prompt), YOLO-World автоматично знаходить об'єкт на нових

фотографіях, знятих персоналом, і генерує файли анотацій з координатами обмежувальних рамок. Наприкінці тижня за розкладом запускається скрипт донавчання локальної моделі YOLO11. Модель YOLO11 тренується на оновленому датасеті, адаптується під конкретний асортимент складу і замінює стару версію на сервері.

Для створення клієнтської частини та взаємодії користувача із мікросервісами розробляється мобільний інтерфейс, який дозволяє легко надсилати знімки, отримувати інформацію з бази даних та бачити результати аналізу. Зв'язок між Java та Python сервісами налаштовано через гнучкі REST API контролери з використанням JSON та передачею медіафайлів через MultipartFile.

Створена програма являє собою комплексне інженерне рішення, що поєднує в собі надійність Enterprise-технологій Java, гнучкість штучного інтелекту на Python та передові підходи Data-Centric AI. Вона стане першим кроком до створення повністю автономних складських комплексів, допомагаючи бізнесу оптимізувати логістику, знизити витрати та назавжди позбутися помилок, пов'язаних із людським фактором під час обліку ТМЦ.

2 АРХІТЕКТУРА СИСТЕМИ

Цей розділ описує архітектуру системи управління інвентарем, яка інтегрує три ключові компоненти для автоматизації обліку товарно-матеріальних цінностей та забезпечення високої відмовостійкості процесу. Структура системи розподілена за функціональними задачами: інтелектуальна локалізація графічних маркувань за допомогою методів комп'ютерного зору, програмне декодування ідентифікаторів та резервне апаратне зчитування у разі виникнення помилок розпізнавання. Архітектура розроблена з урахуванням вимог до масштабованості, швидкодії в режимі реального часу та стабільності роботи за умов нестабільного освітлення або дефектів пакування. Нижче детально описано кожен модель, її архітектуру, принципи роботи та інтеграцію в загальну систему.

2.1 Трирівнева клієнт-серверна модель (UI – логіка – моделі)

Під час розробки програмного забезпечення для системи було обрано сучасну сервіс-орієнтовану мікросервісну архітектуру, що дозволяє чітко розмежувати логіку користувача, бізнес-логіку інвентаризації, збереження медіафайлів та обчислювальні модулі штучного інтелекту. Такий підхід забезпечує високу відмовостійкість, горизонтальне масштабування окремих компонентів, незалежне розгортання в Docker-контейнерах та можливість гнучкого оновлення моделей машинного навчання без зупинки всієї системи.

Архітектурна структура в даному проєкті функціонально поділяється на три ключові рівні.

Презентаційний рівень – це перший рівень взаємодії користувача із системою. У системі користувацький інтерфейс реалізовано у вигляді мобільного додатка (або спеціалізованого ПЗ для ТСД), що дозволяє

будувати гнучкі, адаптивні та швидкі інтерфейси для працівників складських комплексів.

Інтерфейс надає користувачеві змогу:

- керувати вбудованою камерою для захоплення зображень ТМЦ у реальному часі;
- запускати автоматичне зчитування штрих-кодів;
- активувати інтелектуальну детекцію товарів на полицях;
- надсилати нові знімки невідомих об'єктів для автоматизованої розмітки;
- переглядати результати обробки (кількість знайдених одиниць, назви товарів, підсвічені рамки об'єктів);
- зберігати та оновлювати дані про поточні залишки на складі.

Елементи керування (кнопки сканування, поля введення кількості, вікна виводу відеотрансляції, інформаційні картки товарів) розміщено на основній формі екрана інвентаризації. Для кожного елемента передбачено відповідний обробник подій у коді клієнтського застосунку.

Логічний рівень – відповідає за опрацювання всієї логіки взаємодії між користувачем, базою даних та мікросервісами штучного інтелекту. Він реалізований на базі технології Java Spring Boot, яка виступає центральним ядром та оркестратором системи, де:

- обробляються запити від клієнтського додатка;
- здійснюється валідація та перевірка вхідних даних;
- координується взаємодія з реляційною базою даних PostgreSQL (CRUD-операції з ТМЦ);
- генеруються текстові підказки (prompts) на основі назв товарів для модуля авторозмітки;
- асинхронно викликається обчислювальний Python-сервіс через REST API для обробки зображень та детекції.

Серверна частина містить сервісні класи для керування бізнес-процесами складу: перевірка наявності товару за штрих-кодом, списання чи додавання

кількості, логування результатів ревізії. Запити до AI-модуля виконуються асинхронно, щоб забезпечити високу швидкість відгуку інтерфейсу користувача. Приклад логіки активації додаткових функцій інтерфейсу залежно від результату зчитування штрих-коду класичним методом (OpenCV/PyZbar) або передачі на AI-детекцію представлено у лістингу 2.1.

Лістинг 2.1 – Програмний код активації відповідних елементів керування клієнтського додатка відповідно до результатів ідентифікації товару

```

        if (response.getStatusCode().is2xxSuccessful() &&
response.getBody().isDetected())
    {
        ManualScanButton.setVisibility(View.GONE);
        UpdateQuantityInput.setVisibility(View.VISIBLE);
        ConfirmInventoryButton.setVisibility(View.VISIBLE);
    }
else
    {
        ManualScanButton.setVisibility(View.VISIBLE);
        UpdateQuantityInput.setVisibility(View.GONE);
        ConfirmInventoryButton.setVisibility(View.GONE);
        ProductStatusText.setText("Об'єкт не розпізнано. Зробіть
фото для авторозмітки.");
    }

```

Обчислювальний рівень штучного інтелекту – цей рівень відповідає за всю інтелектуальну обробку мультимедійних даних. До нього входить окремий високопродуктивний мікросервіс на базі Python (FastAPI), який завантажує, запускає та донавчає моделі комп'ютерного зору. Для кожного аналітичного завдання передбачено свій функціональний модуль:

- BarcodeScannerModule – за допомогою бібліотек OpenCV та PyZbar виконує первинний пошук, локалізацію та декодування лінійних штрих-кодів на зображенні;

- YoloLocalDetector – реалізує швидку детекцію та підрахунок товарів у реальному часі за допомогою локальної моделі YOLO11, адаптованої під поточний асортимент складу;

- YoloWorldAutoLabeler – велика Zero-Shot модель комп'ютерного зору YOLO-World, яка використовується для автоматичної генерації розмітки (координат обмежувальних рамок) нових товарів за текстовим описом, що надходить від Java-сервісу.

Всі моделі інтегровані в єдиний MLOps-конвеєр штучного інтелекту. Локальна модель YOLO11 щотижнево проходить цикл автоматичного донавчання (Fine-tuning) в Python-середовищі на нових даних, які були розмічені за допомогою YOLO-World. Це забезпечує автономне розширення бази знань системи без залучення Data Scientist-ів та без втрати загальної продуктивності складського обліку. Дані про координати об'єктів передаються між обчислювальним та логічним рівнем у вигляді структурованих JSON-пакетів.

2.2 Модель для виявлення продукції на складі

Першою частиною є модель для виявлення продукції призначена для швидкої детекції та класифікації об'єктів товарно-матеріальних цінностей на складських стелажах, щоб визначити їхню точну кількість, назву та відповідність поточним записам у системі обліку. Тут використовується однопрохідна згорткова нейронна мережа глибокого навчання серії YOLO (зокрема, сучасна архітектура YOLO11 або YOLOv8) із вбудованою технікою регуляризації за допомогою блоків залишкових зв'язків (Residual/Shortcut Connections) для стабільного й ефективного навчання. Ця техніка дозволяє уникнути проблеми згасання градієнта при

збільшенні глибини мережі, додаючи прямі зв'язки між шарами, що допомагають інформації та градієнтам легше проходити крізь архітектуру, щоб модель не втрачала важливі дрібні деталі товарів при високій щільності їхнього розміщення на полицях.

Для активації прихованих шарів нейромережі, а також у модулях просторової уваги та згорткових блоках (таких як C2f або C2PSA), обрано покращені функції активації типу SiLU (Sigmoid Linear Unit) або PreLU (Parametric Rectified Linear Unit) – вони є значно розумнішими за звичайну ReLU, оскільки пропускають невеликі негативні значення, що дозволяє підлаштовувати параметри мережі у процесі оптимізації, щоб навчання йшло швидше й без збоїв. На виході детекційна голова моделі (Detection Head) розділяє завдання на два напрямки: обчислення координат обмежувальних рамок за допомогою функції втрат CIoU або DFL, а також видачу ймовірностей приналежності до конкретного класу продукту через функцію Sigmoid або Softmax, що забезпечує коректну інтерпретацію впевненості моделі у знайденому товарі. Після детекції об'єктів на наступному етапі аналізуються координати тих кадрів, де локалізовано ТМЦ. Це дозволяє одразу зрозуміти, чи відповідає реальна кількість пакувань даним із бази, чи варто задіяти ручне сканування штрих-коду працівником складу. Архітектуру системи детекції можна побачити на рисунку 2.1.

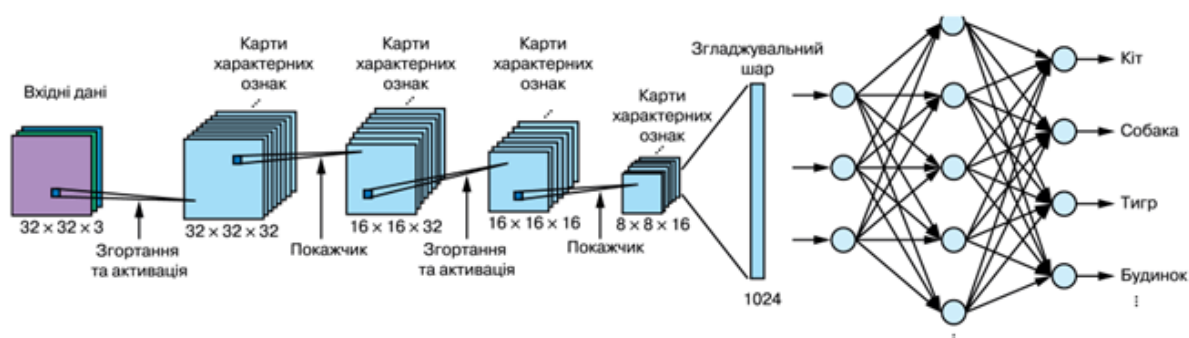


Рисунок 2.1 – Архітектура згорткової нейронної мережі детекції об'єктів

Створення та конфігурування моделі відбуватиметься за допомогою екосистеми штучного інтелекту на мові Python та бібліотеки ultralytics. Але оскільки безпосередня інтеграція важких процесів навчання та детекції в основне корпоративне середовище Java Spring Boot є технічно складною та неефективною, обчислення винесено в окремий асинхронний мікросервіс на базі фреймворку FastAPI. Готова та донавчена модель зберігається у стандартному бінарному форматі (.pt), а для оптимізації швидкодії за потреби може бути переведена у формат ONNX або TensorRT, що дозволяє їй миттєво обробляти медіафайли прямо на сервері та відправляти JSON-структури назад до Java-частини.

Знімки для аналізу беруться безпосередньо з вбудованих камер мобільних пристроїв персоналу, терміналів збору даних (ТСД) або стаціонарних купольних камер, які фіксують заповненість зон зберігання. Перед тим, як пустити зображення в модель, вони проходять обов'язкову попередню підготовку: зміну роздільної здатності до базового розміру вікна моделі (наприклад, 640×640 пікселів) із збереженням пропорцій, нормалізацію колірних каналів для вирівнювання контрастності за умов змінного складського освітлення, а також агресивну аугментацію (наприклад, випадкові повороти, мозаїчне поєднання кількох фото, додавання шуму та зміну яскравості) для покращення узагальнюючої здатності моделі та її стійкості до варіацій у ракурсах зйомки.

Ця модель інтегрується в загальну архітектуру як перший і головний етап візуального аналізу. Користувач робить фотографію стелажа через клієнтський мобільний інтерфейс, і модель за лічені мілісекунди видає результат: локалізує всі знайомі товари. Якщо якісь ділянки кадру або окремі предмети класифікуються як «невідомий об'єкт», користувач робить декілька зображень об'єкта та передає до фоновій моделі з відкритим словником YOLO-World для детальнішого аналізу та автоматичної розмітки. Результати успішного розпізнавання передаються у вигляді JSON-

пакетів до логічного рівня Spring Boot для модуля перенавчання моделі детекції.

2.3 Модель для автоматичного маркування даних

Друга модель відповідає за автоматичне маркування даних (auto-labeling) і призначена для детального аналізу зображень, класифікованих як «невідомий об'єкт» або пов'язаних із новими надходженнями продукції, щоб визначити точне розташування, межі, координати та клас нових товарно-матеріальних цінностей на фотографіях. Для цього впроваджено архітектуру YOLO-World – сучасну згорткову нейронну мережу глибокого навчання з концепцією Open-Vocabulary Object Detection (виявлення об'єктів із відкритим словником), яка широко застосовується для безвибіркової детекції (Zero-Shot Detection), зокрема в задачах автоматизації підготовки складських датасетів. YOLO-World складається з візуального енкодера, який поступово зменшує розмірність зображення та виділяє ознаки, і текстового енкодера (на базі трансформерних архітектур на кшталт CLIP), що кодує назви товарів, дозволяючи точно пов'язати текстовий опис із контурами, геометрією, типом та розташуванням предметів, що є критично важливим для подальшого формування анотацій без участі людини. Дана архітектура ефективно виділяє межі нових товарів, таких як пакунки, коробки чи пляшки, завдяки своїй здатності зіставляти багатомодальну інформацію (текст і візуальні ознаки) через пропускні з'єднання та спеціальні блоки взаємної уваги (Vision-Language Cross-Attention).

Крім того, YOLO-World демонструє високу стійкість до змін умов зйомки на складах, що є особливо важливим при роботі з розмитими фотографіями персоналу та зображеннями з мобільних пристроїв низької якості. Завдяки цьому модель забезпечує точні результати розмітки навіть

за умов слабкої роздільної здатності чи невдалого ракурсу вхідних зображень. Дану архітектуру представлено на рисунку 2.2.

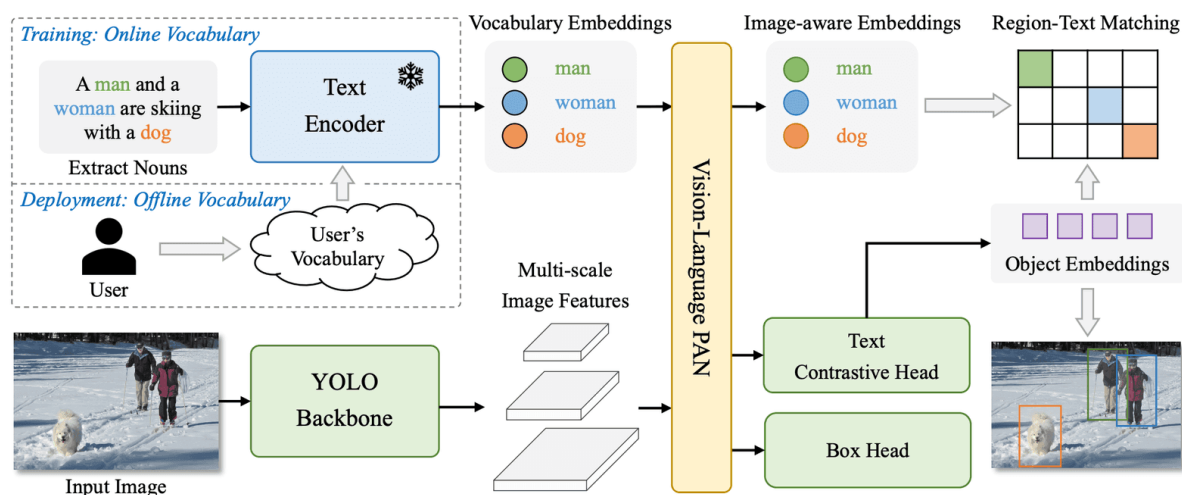


Рисунок 2.2 – Архітектура YOLO-World

Візуальна частина YOLO-World складається з кількох згорткових шарів Backbone, які поступово зменшуює розмірність зображення, виділяючи ключові ознаки, такі як краї пакувань чи текстура брендування продукту. Текстова частина обробляє вхідні запити, створюючи семантичні вектори, де кожен об'єкт на фото позначається відповідним ім'ям із бази даних. Для активації шарів знову використовується функція SiLU або PReLU, щоб забезпечити стабільну роботу моделі.

Модель функціонує в Python за допомогою бібліотеки ultralytics та асинхронного фреймворку FastAPI, оскільки цей стек пропонує гнучкі інструменти для роботи з багатомодальними мережами та обробки великих медіафайлів. Для генерації розмітки беруться фотографії товарів, які працівники складу зробили під час сканування, з текстовими підказками, що позначають назву ТМЦ. Назва продукту є особливо ефективною для виявлення меж конкретного об'єкта, тому використовуємо її як основний

орієнтир, додаючи ідентифікатори товарів із Java Spring Boot для покращення точності зіставлення класів.

Попередня обробка в цій моделі потребує формування тимчасового датасету з фотографій нерозпізнаних товарів та метаданих, таких як штрих-код, базова назва продукту та артикул. Метадані структуруються та стандартизуються для забезпечення коректності подальшої генерації текстових підказок. Також виконується попередня обробка зображень, зокрема зміна розміру до стандартних пікселів, нормалізація яскравості та контрастності, видалення завад за допомогою фільтрації, щоб уніфікувати дані перед подачею в YOLO-World.

Наступним кроком є тестування та валідація моделі M2. Використовуються метрики Mean Squared Error (MSE) для оцінки точності локалізації центрів об'єктів, Intersection over Union (IoU) – для визначення збігу площі згенерованих рамок із реальними межами предметів, а також Recall, Precision та F1-score для правильної класифікації товарів. Верифікація моделі M2 виконується на контрольному наборі даних, включаючи раніше невикористані специфічні складські товари, щоб оцінити її узагальнену здатність до розпізнавання слів із відкритого словника.

Концептуальний аналіз моделі M2 зосереджується на виявленні помилок розмітки (наприклад, хибних спрацьовувань на фонові об'єкти), їх усуненні через оптимізацію текстових запитів та порогових значень впевненості (confidence threshold). Оцінюється вплив щільності розміщення товарів та якості освітлення, а також можливість інтеграції текстових описів із ERP-системи для покращення автоматичного маркування.

Для оцінки якості автоматичної розмітки застосовуються критерії та (Mean Average Precision), Intersection over Union (IoU) – для визначення точності охоплення об'єкта, а також Recall та Precision для класифікації нових позицій. Ці метрики дозволяють переконатися, що модель точно виділяє межі товарів із мінімальними помилками. Після успішного

відпрацювання алгоритму результати автоматично зберігаються у вигляді текстових файлів конфігурації у форматі YOLO (.txt), де кожен рядок містить індекс класу та нормовані координати рамки. Ці файли поповнюють навчальний датасет і використовуються для щотижневого конвеєра Fine-tuning локальної мережі YOLO11.

У системі ця модель працює як другий етап: після того, як перша модель або сканер штрих-кодів ідентифікували товар як новий чи невідомий для поточної бази знань, YOLO-World створює детальну карту координат, чітко окреслюючи контури об'єктів. Сгенеровані рамки та мітки класів обробляються у фоновому режимі Python-сервісу, а результати у вигляді звітів надсилаються до Java-сервера, де адміністратор складу може перевірити правильність сформованого датасету. Така автоматизація допомагає логістам та комірникам швидко вносити нові партії ТМЦ до інтелектуальної системи обліку та планувати подальші інвентаризації без ручної праці дата-саєнтистів.

2.4 Конвеєр автоматичного донавчання моделі

Третьою частиною архітектурного проектування є конвеєр автоматичного донавчання моделі (Model Retraining Pipeline), призначений для систематичного, ізольованого та контрольованого перенавчання локальних алгоритмів комп'ютерного зору на основі щойно сформованих авторозмічених даних, з метою безперервного розширення бази знань системи та адаптації нейромережі до нових видів складської продукції. Тут використовується концепція спеціалізованого MLOps-конвеєра донавчання (Continuous Training Pipeline), який автоматизує процеси валідації датасету, розподілу вибірок та інтеграції оновлених ваг нейромережі у виробниче середовище без зупинки основних сервісів обліку ТМЦ. Цей підхід гарантує, що накопичена за допомогою моделі штучного інтелекту YOLO-World інформація проходить повний цикл перевірки та

оптимізації, перетворюючи сирі знімки у стабільне та точне програмне забезпечення для щоденних інвентаризацій.

Основним завданням цієї стадії конвеєра є запобігання деградації моделі, усунення проблеми зсуву даних (Data Drift) та забезпечення високої узагальнюючої здатності алгоритмів детекції за умов появи нових брендів чи пакувань товарів на складі. Весь життєвий цикл функціонування даного модульного конвеєра розділений на кілька послідовних фаз, кожна з яких виконує суворо визначену роль. Структурну схему та етапи проходження даних у конвеєрі автоматичного донавчання представлено на рисунку 2.4

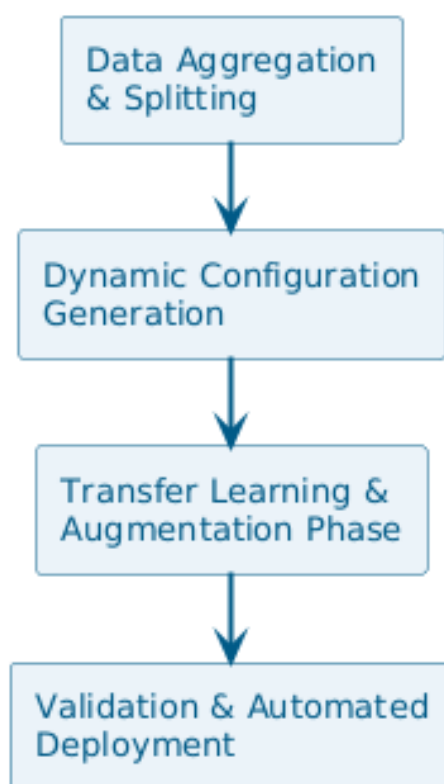


Рисунок 2.4 – Конвеєр автоматичного донавчання моделі (Model Retraining Pipeline)

Перша фаза конвеєра – це Агрегація та автоматичний розподіл даних (Data Aggregation & Splitting). Після того, як модель авторозмітки YOLO-World завершує формування пулу нових файлів, конвеєр підраховує

кількість унікальних зображень і відповідних їм анотаційних .txt специфікацій, збережених у буферній зоні сервера (Staging Area). Скрипт автоматично переміщує наявні файли для уникнення кластеризації за часом зйомки та розділяє їх у жорсткій пропорції: 80% даних відправляється у навчальну вибірку (Train Dataset), а 20% – у валідаційну вибірку (Validation Dataset). Це необхідно для об'єктивного контролю процесу навчання.

Друга фаза – це Динамічна генерація конфігурації (Dynamic Configuration Generation). Конвеєр автоматично ініціює REST-запит до логічного рівня Java Spring Boot, отримуючи актуальний повний перелік зареєстрованих товарних позицій зі СКБД PostgreSQL. На основі цих даних Python-сервіс динамічно генерує або оновлює конфігураційний файл у форматі, сумісному з бібліотекою Ultralytics. У цьому файлі чітко прописуються абсолютні шляхи до папок навчання та валідації, а також формується структурований масив індексів та назв класів для детекції.

Третя фаза – це переносне навчання та комп'ютерна аугментація (Transfer Learning & Augmentation Phase). Конвеєр завантажує поточні робочі ваги базової локальної моделі YOLO11 та запускає процес тонкого налаштування (Fine-tuning). На цьому етапі до вхідних даних застосовується агресивний конвеєр трансформацій: технологія мозаїчного колажування (Mosaic Augmentation), випадкове масштабування, колірні зсуви та розмиття. Навчання проводиться протягом оптимальної кількості епох (як правило, від 30 до 50) із використанням оптимізатора AdamW та динамічного кроку навчання, що дозволяє швидко адаптувати вагові коефіцієнти детекційної голови під геометрію нових товарів без втрати знань про старі об'єкти.

Четверта, фінальна фаза конвеєра – це валідація та автоматичне розгортання (Validation & Automated Deployment). Після завершення обчислень конвеєр аналізує фінальні метрики якості нової моделі на прихованій валідаційній вибірці:

- якщо узагальнена точність нової версії перевищує встановлений поріг або метрики попередньої моделі, конвеєр виконує безпечну заміну бінарного файлу .pt у виробничій директорії мікросервісу FastAPI;

- якщо ж метрики виявляються незадовільними (наприклад, через низьку якість початкових фотографій), система блокує розгортання, надсилає сповіщення адміністратору складу та зберігає лог помилки для подальшого аналізу.

Для оцінки якості роботи конвеєра автоматичного донавчання застосовуються такі академічні метрики, як функція втрат локалізації рамок об'єктів Box Loss (CIoU), точність класифікації Class Loss, а також інтегральні показники mean Average Precision (mAP_{50} та mAP_{50-95}). Такий автоматизований MLOps-пайплайн дозволяє побудувати повністю автономну систему, яка самостійно розвивається та покращує точність розпізнавання ТМЦ безпосередньо у процесі повсякденної експлуатації складського комплексу, повністю усуваючи потребу в залученні інженерів з розмітки даних та максимізуючи економічну ефективність підприємства.

2.5 Потік даних у системі та інвентаризаційному менеджменті

У системі було реалізовано чіткий і структурований потік даних, що забезпечував безперебійну обробку інформації та злагоджену взаємодію між усіма компонентами програми: інтерфейсом користувача мобільного додатка, логікою обробки сервера Java Spring Boot, моделями штучного інтелекту (модулем штрих-кодів, локальним детектором YOLO11, фоновим маркувальником YOLO-World). Такий підхід дозволяв інтелектуальній системі складського обліку не лише проводити точну ревізію залишків товарно-матеріальних цінностей, а й надавати користувачу практичні рекомендації щодо зон оптимального розміщення та умов зберігання вантажів, роблячи процес інвентаризації інтуїтивно зрозумілим навіть для людей без спеціальної технічної підготовки.

Потік даних у системі розпочинався із захоплення кадру або трансляції відеопотоку стелажа з товарами працівником складу через мобільний додаток або термінал збору даних (ТСД). Після цього метадані автоматично передавалися до логічного ядра програми на базі Java Spring Boot, яке координувало подальшу обробку та асинхронно перенаправляло зображення до обчислювального Python-мікросервісу. На першому етапі дані надходили до класичного модуля ідентифікації PyZbar, який виконував пошук та декодування лінійних штрих-кодів. Результат зчитування – унікальний числовий ідентифікатор – передавався назад на логічний рівень. Якщо штрих-код успішно розпізнавався і знаходився в реляційній базі даних PostgreSQL, обробка візуального ряду для цієї одиниці завершувалася: система автоматично інкрементувала кількість товару, записувала результат у таблицю інвентаризацій і пропонувала користувачу перейти до наступного об'єкта.

У разі якщо локальна модель штучного інтелекту YOLO11 не здатна автоматично ідентифікувати або класифікувати об'єкт на полиці (наприклад, через відсутність ваг для нової номенклатурної одиниці або специфічні умови зйомки), програмний комплекс активує альтернативний сценарій верифікації. У такому випадку система надсилає оператору відповідне сповіщення та просить співробітника складу вручну відсканувати матричний QR-код або лінійний штрих-код продукту для примусового пошуку та виведення повної інформації про товар із центральної бази даних.

На завершальному етапі, якщо локальна модель фіксувала невідомий предмет, потік даних активував фоновий конвеєр автоматичного маркування. Працівник складу вносив базові метадані товару (назву або артикул) через ручне сканування, після чого модель з відкритим словником YOLO-World отримувала ці метадані як текстовий промпт разом із серією опорних фотографій. Використовуючи механізми взаємної уваги (Cross-Attention), YOLO-World прогнозувала межі нового об'єкта та автоматично

генерувала сумісні файли розмітки у форматі .txt. Усі результати накопичувалися в буферній зоні сервера (Staging Area) для щотижневого конвеєра донавчання. Після успішного завершення чергового циклу перенавчання оновлені ваги моделі інтегрувалися в систему, а користувач бачив у своєму інтерфейсі сповіщення про успішне розширення асортименту розпізнавання, супроводжуване оновленням цифровим каталогом та порадами щодо правильного розміщення нових ТМЦ у специфічних зонах складу.

Структура системи управління інвентаризацією була побудована за трирівневою архітектурою:

- рівень інтерфейсу користувача: забезпечував зручне захоплення зображень камерою, відображення результатів детекції у вигляді підсвічених рамок, вивід карток товарів та інвентаризаційних звітів. Інтерфейс був інтуїтивним, щоб комірники, вантажники чи оператори обліку могли легко працювати з програмою без глибоких знань в AI;

- рівень логіки обробки: координував потік даних між мікросервісами, виконував валідацію запитів, керував транзакційним збереженням інформації та зв'язком із базою даних PostgreSQL. Цей рівень також відповідав за контроль винятків, наприклад, перевірку прав доступу персоналу або обробку помилок зв'язку з обчислювальними модулями;

- рівень моделей III: включав три взаємопов'язані аналітичні компоненти (модуль декодування штрих-кодів PyZbar, локальний швидкий детектор YOLO11 та модель авторозмітки YOLO-World), які обробляли візуальні дані та генерували результати. Моделі були інтегровані через асинхронні REST API контролери FastAPI на мові Python, що забезпечувало високу продуктивність обчислень.

Така організація потоку даних у поєднанні зі структурою інтелектуального логістичного менеджменту робила програму ефективним інструментом для швидкої оцінки заповненості складу та планування ревізій. У майбутньому потік даних міг бути вдосконалений шляхом

інтеграції з великими корпоративними ERP-системами підприємства (наприклад, SAP або Oracle WMS) для роботи з мільйонними обсягами номенклатури товарів або підключення до мережі стаціонарних стельових камер для повністю автономного моніторингу складу в реальному часі.

Перспективи подальшого розвитку та масштабування архітектури розробленого рішення полягають у його вертикальній та горизонтальній інтеграції з існуючим корпоративним середовищем підприємств. Насамперед, інформаційний потік системи «Fasty» може бути модернізований шляхом розробки спеціалізованих API-шлюзів для безшовної інтеграції з ERP-системами та WMS-платформами промислового рівня, такими як SAP, Oracle WMS або Microsoft Dynamics 365. Це дозволить підсистемі штучного інтелекту оперувати мільйонними обсягами номенклатурних одиниць (SKU) у режимі реального часу, забезпечуючи двосторонню синхронізацію даних, автоматичне формування замовлень постачальникам при досягненні критичного мінімуму товарів та гнучке керування ланцюгами постачань на основі предиктивної аналітики нейромереж.

Іншим стратегічним вектором вдосконалення є перехід від концепції періодичного моніторингу за допомогою мобільних пристроїв та ТСД до створення повністю автономної інфраструктури «Розумного складу» (Smart Warehouse). Даний крок передбачає інтеграцію програмного комплексу з розгалуженою мережею стаціонарних стельових та міжстелажних камер високої роздільної здатності, а також купольних камер із функцією PTZ (Pan-Tilt-Zoom). Автоматизований аналіз постійного відеопотоку з фіксованих точок спостереження дозволить реалізувати безперервний просторовий моніторинг складських зон, миттєво фіксувати будь-які переміщення палет чи окремих коробок, автоматично виявляти випадки неправильного позиціонування вантажів і надсилати пуш-сповіщення

адміністраторам у разі виникнення позаштатних ситуацій чи критичного дефіциту ТМЦ без залучення людського ресурсу.

2.6 Структура даних та зберігання

Структура даних розробленої системи обліку ТМЦ організована таким чином, щоб забезпечити зручність обміну між компонентами, швидкий доступ до проміжних результатів та можливість фіксації результатів інвентаризації в базі даних без втрати інформації. Враховуючи поетапну обробку та мікросервісну модульність архітектури, дані зберігаються у вигляді записів у реляційній базі даних, структурованих обмінних файлів у форматі JSON, а також графічних об'єктів у файловій системі.

Після захоплення або завантаження зображення товару працівником складу, файл фотографії зберігається безпосередньо у виділеному схові зображень (Image Storage) у вигляді графічного об'єкта, а унікальний відносний шлях до нього фіксується у базі даних у вигляді рядка текстового типу. Сама фотографія завантажується модулями комп'ютерного зору в оперативну пам'ять як об'єкт матриці NumPy для подальшої високопродуктивної обробки. Результати класифікації та детекції, отримані від нейромереж, кодуються та передаються між мікросервісами у вигляді структурованих повідомлень формату JSON, які містять координати обмежувальних рамок (bounding boxes), мітки класів та коефіцієнти впевненості. Це дозволяє використовувати ці дані для динамічного накладання графічних рамок поверх фотографій в інтерфейсі ТСД. Усі ці об'єкти передаються до логічного ядра системи на Java Spring Boot у структурованому вигляді, без необхідності повторного проходження важких обчислювальних конвеєрів ІІІ.

Для забезпечення узгодженості, довговічності та високої швидкості доступу між модулями у системі використовуються реляційні таблиці СКБД

PostgreSQL. Логічна схема збереження інформації про товарні запаси та топологію складу базується на трьох основних сутностях:

- категорії товарів (category): таблиця визначає верхній рівень класифікації продукції. Окрім стандартних текстових полів (name, description), вона містить поле category_fields у форматі JSON. Це дозволяє динамічно визначати схему метаданих та специфічних характеристик для різних типів продукції без необхідності постійної зміни структури таблиць у базі даних;

- продукція (product): ядро інформаційної структури, де зберігаються унікальні ідентифікатори товарів, їхні назви, опис, поточна кількість на складі (quantity) та одиниці виміру (uof). Для зв'язку з логістичними процесами кожна позиція має артикул (sku) та унікальний штрих-код. Таблиця містить зовнішній ключ, що посилається на категорії, а також гнучке поле attributes у форматі JSON, куди записуються динамічні властивості конкретного товару (наприклад, термін придатності, серійний номер або габарити);

- місця зберігання (storage_location): таблиця описує ієрархічну структуру складських приміщень (зони, ряди, стелажі, полиці). Завдяки рекурсивному зв'язку parent_id, який посилається на первинний ключ цієї ж таблиці, система здатна будувати деревоподібні маршрути та точно вказувати специфічні комірки, де повинен зберігатися або куди має бути переміщений розпізнаний за допомогою комп'ютерного зору товар.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДЕЛЕЙ ШТУЧНОГО ІНТЕЛЕКТУ

3.1 Використані технології та компоненти

Для створення системи було використано сучасний набір технологій і інструментів, що забезпечували ефективну реалізацію функціоналу машинного навчання, зручного клієнтського інтерфейсу, обробки зображень, транзакційного збереження даних та автоматизації MLOps- процесів. Архітектура системи була спроектована з акцентом на продуктивність, модульність і можливість масштабування, що дозволяло легко адаптувати її до нових вимог і забезпечувало стабільну роботу моделей штучного інтелекту (ШІ) у реальних умовах складського комплексу.

3.1.1 Інструменти Python (FastAPI, Ultralytics YOLO, OpenCV)

Основним середовищем для розгортання, виконання, автоматичного маркування та періодичного донавчання моделей штучного інтелекту була обрана мова програмування Python, яка завдяки своїй універсальності та потужній екосистемі спеціалізованих бібліотек ідеально підходить для розв'язання складних завдань комп'ютерного зору. Обчислювальний рівень системи реалізовано у вигляді ізольованого асинхронного мікросервісу, в межах якого використовувалися такі ключові інструменти та бібліотеки:

- FastAPI: сучасний високопродуктивний веб-фреймворк, який використовувався для створення REST API інтерфейсу обчислювального ядра. Завдяки вбудованій асинхронності (асинхронні обробники `async/await`) та повній підтримці валідації даних за допомогою Pydantic, FastAPI забезпечує миттєве приймання медіафайлів у форматі `MultipartFile` від Java-сервера, паралельний запуск аналітичних конвеєрів ШІ та швидке повернення результатів детекції у структурованому форматі JSON;

– Ultralytics (YOLO11): спеціалізована бібліотека, яка стала основою для інтеграції, навчання та запуску локальної моделі детекції об'єктів. Програмні інструменти Ultralytics дозволили гнучко реалізувати механізм тонкого налаштування (Fine-tuning) нейромережі на специфічних наборах даних складських ТМЦ. Бібліотека цінується за високу швидкість виконання інференсу, автоматичну оптимізацію обчислень на графічних процесорах (CUDA) та зручні інструменти для валідації моделей безпосередньо у виробничому середовищі;

– YOLO-World: інтегрований програмний модуль на базі технологій трансформерів комп'ютерного зору з відкритим словником (Open-Vocabulary Detection). Цей інструмент використовувався у фазі автоматичного маркування (Auto-labeling) для генерації файлів розмітки нових або нерозпізнаних товарів за текстовими підказками (prompts), що надходять від логічного ядра Java Spring Boot, повністю виключаючи потребу в ручній анотації даних;

– OpenCV (Open Source Computer Vision Library): застосовувалася для виконання низькорівневих числових та матричних операцій над зображеннями перед їх подачею в нейромережі. Основними функціями OpenCV у конвеєрі стали: зміна роздільної здатності кадру до базового розміру 640×640 пікселів, вирівнювання гістограми для нормалізації яскравості та контрастності в умовах змінного складського освітлення, а також попереднє колірне перетворення матриць;

– PyZbar: вузькоспеціалізована бібліотека класичного комп'ютерного зору, інтегрована як перший етап гібридної ідентифікації. Вона забезпечує миттєве сканування, локалізацію та декодування лінійних штрих-кодів на пакуваннях товарів, суттєво заощаджуючи обчислювальні ресурси сервера за умов наявності чіткого стандартного маркування на об'єктах ревізії.

Вбудовані засоби аугментації Ultralytics: забезпечували динамічну модифікацію зображень під час циклу щотижневого донавчання моделі YOLO11. Завдяки алгоритмам мозаїчного колажування (Mosaic

Augmentation), випадковим поворотам, афінним перетворенням та додаванню штучного цифрового шуму, було досягнуто високої стійкості локальної моделі до варіацій у ракурсах зйомки та якості камер мобільних пристроїв персоналу.

Після завершення щотижневих циклів перенавчання точність нових ваг автоматично оцінювалася за метриками mAP_{50} та mAP_{50-95} за допомогою інструментів валідації Ultralytics. Оновлена версія моделі зберігалася у бінарному форматі (.pt), що дозволяло використовувати її в асинхронних потоках FastAPI без переривання життєвого циклу всього складського комплексу. Такий підхід забезпечив швидку інтеграцію моделей комп'ютерного зору в загальну мікросервісну архітектуру проєкту та оптимізував загальну продуктивність і швидкість проведення інвентаризацій.

3.1.2 Інструменти логічного рівня (Java, Spring Boot, Spring Data JPA)

Логічне ядро та серверна частина системи були реалізовані на сучасній програмній платформі Java 25 із використанням екосистеми Spring, що гарантувало високу транзакційну стабільність, корпоративний рівень безпеки та повну підтримку асинхронної архітектури при роботі з потоками даних.

У межах цього рівня використовувалися такі ключові технології:

- Java Spring Boot: цей фреймворк став основою для побудови мікросервісної архітектури та оркестрації всієї бізнес-логіки складського комплексу. Spring Boot забезпечив розгортання високопродуктивних REST API контролерів, які координують взаємодію між клієнтським додатком ТСД та обчислювальними модулями комп'ютерного зору на базі FastAPI. Завдяки вбудованому серверу Tomcat фреймворк підтримує паралельне опрацювання великої кількості запитів на інвентаризацію від багатьох працівників складу одночасно;

– Spring Data JPA (Hibernate): архітектурний інструмент для об'єктно-реляційного відображення (ORM), який використовувався для забезпечення швидкого, надійного та безпечного доступу до даних. Завдяки абстракції репозиторіїв та автоматичній генерації SQL-запитів, Spring Data JPA оптимізував CRUD-операції з номенклатурою товарів, автоматизував контроль залишків (quantity) на стелажах та мінімізував ризик виникнення помилок при одночасному оновленні даних декількома операторами;

– Jackson (Spring Boot Starter Web Components): спеціалізована бібліотека, інтегрована для серіалізації та десеріалізації обмінних пакетів. Оскільки обчислювальний Python-сервіс повертає результати AI-детекції (координати рамок, назви та класи) у вигляді JSON-структур, а таблиці категорій та продуктів у СКБД PostgreSQL містять поля типу JSONB, Jackson забезпечує їх швидку та безпомилкову трансформацію у Java-об'єкти (Entities/DTO) всередині оперативної пам'яті;

– асинхронне програмування (@Async та CompletableFuture): механізми Java, які застосовувалися для розпаралелювання ресурсомістких процесів. Оскільки передача зображень високої роздільної здатності до AI-моделей та очікування відповіді від YOLO-World вимагають певного часу, Spring-сервер обробляє ці запити асинхронно у фонових потоках. Це дозволяє миттєво повертати відповідь на клієнтський інтерфейс, уникаючи блокування та затримок у роботі персоналу складу.

3.1.3 Зовнішні джерела даних та інфраструктура (СКБД, ERP, REST API)

Для повноцінного функціонування системи інтелектуального інвентаризаційного менеджменту, забезпечення цілісності номенклатурних довідників та автоматизації MLOps-конвеєра, архітектура передбачає гнучку інтеграцію з внутрішніми та зовнішніми інформаційними системами підприємства:

– реляційна СКБД PostgreSQL: виступає головним сховищем даних проєкту. На відміну від файлових або локальних сховищ, PostgreSQL гарантує суворе дотримання принципів ACID для транзакцій складського обліку. Наявність підтримки неструктурованих даних типу JSONB дозволила зберігати динамічні атрибути товарів (терміни придатності, габарити) та специфічні характеристики категорій без ускладнення та постійної перебудови реляційної схеми бази даних;

– локальне файлове сховище (Image Storage System): виділена інфраструктурна зона на сервері, куди Java-сервер записує оригінальні фотографії товарів. Зберігання бінарних медіафайлів безпосередньо у файловій системі з фіксацією лише текстових шляхів (`image_url`) у PostgreSQL дозволило розвантажити базу даних, оптимізувати використання оперативної пам'яті та забезпечити миттєвий доступ до знімків під час щотижневого Fine-tuning навчання;

– інтеграційні інтерфейси ERP/WMS корпоративного рівня: логічний рівень Spring Boot спроектований із розрахунком на підключення до зовнішніх систем автоматизації підприємства через стандартні захищені REST API та веб-сокети. Це необхідно для синхронізації артикулів (`sku`), імпорту нових штрих-кодів, автоматичного формування накладних за результатами розпізнавання ШІ та ведення наскрізної аналітики залишків ТМЦ в масштабах усієї компанії.

Інтеграція зазначених джерел та інфраструктурних компонентів робить систему гнучкою і готовою до розгортання в різних умовах, наприклад, на базі хмарних платформ (AWS, Google Cloud) або на локальних серверах підприємств (On-Premise). У майбутньому архітектуру системи можна легко розширити шляхом додавання брокерів повідомлень (наприклад, Apache Kafka або RabbitMQ) для ще швидшої обробки черг великих потоків відеоданих з камер спостереження в реальному часі.

Поєднання асинхронного Python-мікросервісу на FastAPI (для виконання обчислень штучного інтелекту) та Java Spring Boot із PostgreSQL (для стабільного керування бізнес-процесами та транзакціями) забезпечило ідеальний баланс між високою продуктивністю машинного навчання та промисловою надійністю системи автоматизації. Обрані інструменти дозволили реалізувати комплексну мікросервісну екосистему, яка оптимізує загальну продуктивність і швидкість проведення складських інвентаризацій.

3.2 Модуль класичного сканування штрих-кодів (PyZbar)

Цей модуль відповідає за початкову обробку вхідного зображення, тобто за визначення того, чи містить воно стандартне маркування у вигляді лінійного штрих-коду. Таке сканування дозволяє ефективно ідентифікувати товари «на льоту» та відсіювати інформацію, економлячи важкі обчислювальні ресурси нейромереж штучного інтелекту для подальшої обробки лише тих зображень, де маркування відсутнє або пошкоджене. Побудований на основі комбінації алгоритмів бібліотеки OpenCV для попередньої фільтрації матриці кадру та спеціалізованого декодера PyZbar, модуль забезпечує високу швидкість і точність розпізнавання навіть на знімках із різними умовами складського освітлення чи геометричними викривленнями.

3.2.1 Архітектура REST API та структура обміну даними

Для інтеграції модуля сканування штрих-кодів у загальну мікросервісну систему було розроблено виділений ендпоінт на базі асинхронного фреймворку FastAPI. Взаємодія між сервером бізнес-логіки Java Spring Boot та обчислювальним Python-сервісом відбувається через захищений HTTP-протокол. Передача бінарного файлу зображення

здійснюється за допомогою структурованого MIME-формату multipart/form-data.

Лістинг 3.1 – Приклад cURL-запиту для активації процесу сканування

```
curl -X 'POST' \  
  'http://localhost:8003/api/barcode/scan' \  
  -H 'accept: application/json' \  
  -H 'Content-Type: multipart/form-data' \  
  -F 'file=@OIP.webp;type=image/webp'
```

Після отримання запиту сервіс переводить бінарні дані у матричний вигляд, локалізує графічні контури штрих-коду та виконує бінаризацію. Результат декодування повертається логічному ядру у вигляді структурованого JSON-пакета, який містить прапорець успішності пошуку (found), закодоване числове значення (data), символіку штрих-коду (barcode_type), а також просторові координати обмежувальної рамки bbox (координати x, y, ширину та висоту) для візуалізації на екрані ТСД. Приклад структури серверної відповіді наведено у лістингу 3.2.

Лістинг 3.2 – Структура вихідного JSON-файлу за результатами сканування

```
{  
  "found": true,  
  "barcodes": [  
    {  
      "data": "1234567890",  
      "barcode_type": "CODE128",  
      "bbox": {  
        "x": 12,
```

Продовження лістингу 3.2

```

        "y": 111,
        "width": 445,
        "height": 208
    }
}
]
}

```

Програмний код для реалізації ендпоінту та алгоритму декодування зображення на стороні Python FastAPI представлено у лістингу 3.3.

Лістинг 3.3 – Програмний код обробника запиту сканування штрих-кодів

```

def scan_enhanced(self, image: np.ndarray) ->
List[BarcodeResult]:
    results = self._decode(image)
    if results:
        return results
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    results = self._decode(gray)
    if results:
        return results

    clahe = cv2.createCLAHE(clipLimit=2.0,
tileGridSize=(8, 8))
    enhanced = clahe.apply(gray)
    results = self._decode(enhanced)
    if results:
        return results
    thresh = cv2.adaptiveThreshold(
        enhanced, 255,
        cv2.ADAPTIVE_THRESH_GAUSSIAN_C,

```

Продовження лістингу 3.3

```

        cv2.THRESH_BINARY, 11, 2,
    )
    results = self._decode(thresh)
    if results:
        return results

    kernel = np.array([[ -1, -1, -1],
                       [ -1,  9, -1],
                       [ -1, -1, -1]])
    sharp = cv2.filter2D(gray, -1, kernel)
    results = self._decode(sharp)
    if results:
        return results

    big = cv2.resize(image, None, fx=2, fy=2,

                     interpolation=cv2.INTER_CUBIC)
    results = self._decode(big)
    if results:
        return results

    logger.info("No barcode found after all strategies")
    return []

```

3.3 Модуль інтелектуального автоматичного маркування товарів

Для взаємодії з модулем автоматичної розмітки було реалізовано спеціалізований Python-клас `YOLOWorldLabeler`. Його головною архітектурною особливістю є здатність динамічно перевизначати робочий словник класів через метод `set_classes()`. Коли користувач ініціює реєстрацію нового продукту (наприклад, із категорійним типом `toy` – іграшка), система передає бінарну матрицю BGR зображення та цільовий текстовий промпт у метод `label_image()`.

Процес обробки та переведення координат виконується за чітким математичним алгоритмом. Неймережа знаходить межі об'єкта у стандартному піксельному форматі $[x_1, y_1, x_2, y_2]$, після чого система автоматично трансформує їх у нормалізований формат YOLO, який вимагається для навчання локальних моделей серії YOLO11:

$$c_x = \frac{x_1 + x_2}{2 \cdot w}, \quad (3.1)$$

$$c_y = \frac{y_1 + y_2}{2 \cdot h}, \quad (3.2)$$

$$b_w = \frac{x_2 - x_1}{w}, \quad (3.3)$$

$$b_h = \frac{y_2 - y_1}{h}, \quad (3.4)$$

де c_x, c_y – відносні координати центру обмежувальної рамки;

b_w, b_h – відносна ширина та висота об'єкта;

а w, h – геометричні параметри вихідного кадру.

Приклад результату роботи модуля (детекція класу `toy` із високим коефіцієнтом впевненості) та генерація відповідного YOLO-сумісного анотаційного рядка представлені у лістингу 3.4.

Лістинг 3.4 – Текстовий файл розмітки за результатами автомаркування

```
0 0.466904 0.625960 0.674542 0.503968
```

Для масової підготовки великих масивів даних у фоновому режимі розроблено метод `label_dataset_from_minio()`. Він асинхронно завантажує оригінальні файли з об'єктного сховища MinIO за допомогою клієнта `boto3`,

виконує пакетну детекцію через OpenCV та автоматично записує готові файли анотацій .txt поруч із зображеннями. Програмну реалізацію логіки сервісу автоматичного маркування наведено у додатку А.

У разі, якщо максимальний коефіцієнт впевненості детекції падає нижче порогу 0.55, об'єкт отримує прапорець `needs_review=True`. Це сигналізує логічному рівню про необхідність верифікації даного кадру оператором складу, що запобігає потраплянню хибних або зашумлених анотацій у підсумковий навчальний набір даних та гарантує стабільно високу точність подальшого Fine-tuning етапу.

3.4 Навчання та оцінка якості моделі детекції товарів

Цей підрозділ присвячено етапу розгортання, тонкого налаштування (Fine-tuning) та експериментального аналізу результатів навчання локальної нейромережі архітектури YOLO11n на сформованому за допомогою авторозмітки складському датасеті. Метою цього етапу є валідація отриманих ваг моделі та оцінка її здатності безпомилково локалізувати й класифікувати цільові комерційні об'єкти на полицях під час проведення складської ревізії. Навчання проводилося протягом 100 epochs із застосуванням апаратного прискорення CUDA, динамічного кроку навчання та оптимізатора AdamW.

3.4.1 Аналіз загальних та покласових метрик ефективності

Для об'єктивного оцінювання якості навченої моделі детекції на прихованій валідаційній вибірці було розраховано стандартні метрики комп'ютерного зору: точність (Precision), повноту (Recall), а також інтегральну середню точність (mAP). Зведені результати загальної ефективності конвеєра детекції для всіх класів продукції мають такий вигляд:

– точність локалізації (Box P): становить 0.855. Це свідчить про те, що 85.5% усіх обмежувальних рамок (bounding boxes), сформованих нейромережею YOLO11n, є повністю коректними, що мінімізує ризик хибнопозитивних спрацювань (False Positives);

– повнота виявлення (Recall): досягла показника 0.989. Дане значення вказує на те, що модель успішно ідентифікувала 98.9% фактично наявних об'єктів на складських полицях, практично виключаючи пропуски товарів (False Negatives);

– середня точність mAP_{50} : становить 0.995. Показник Mean Average Precision при поргу перетину $IoU = 0.5$ є відмінним і підтверджує стабільність розпізнавання об'єктів у стандартних умовах;

– інтегральна точність mAP_{50-95} : дорівнює 0.962. Високе значення цієї метрики (усереднене за строгими порогоми IoU від 0.5 до 0.95) доводить, що модель демонструє високу геометричну точність підгонки меж рамок навіть за умов часткового перекриття товарів на стелажі.

Детальний покласовий аналіз (Per-Class Performance) дозволяє оцінити якість розпізнавання конкретних номенклатурних одиниць, що були внесені до бази даних і пройшли етап автоматичного маркування:

– Funko-pop-Deadpool: демонструє ідеальну точність (Precision= 1.0) та вкрай високу повноту виявлення (Recall = 0.967) у поєднанні з максимальними оцінками кривої mAP . Модель безпомилково виділяє цей візуальний клас серед інших об'єктів;

– Funko-pop-Notaru: фіксує високу точність (0.81) та абсолютний показник повноти (Recall = 1.0) що гарантує стовідсоткове виявлення даної позиції на кадрі;

– Mini-crafts-baby-ogre: показує стабільну точність на рівні 0.754 та бездоганний показник повноти (Recall = 1.0) із високими супутніми інтегральними оцінками mAP .

Отримані статистичні метрики однозначно вказують на те, що архітектура YOLO11n успішно узагальнила візуальні ознаки предметів із

наданого датасету. Стабільна висока повнота виявлення (Recall) за всіма категоріями є критично важливою перевагою для логістичного менеджменту, оскільки розроблена інтелектуальна система не пропускає завантажені на склад товари, забезпечуючи точний автоматичний підрахунок кількості залишків у реальному часі.

Високі аналітичні показники підтверджують життєздатність концепції Data-Centric AI: автоматична розмітка за допомогою семантичних підказок у YOLO-World сформувала якісний повний набір даних, навчання на якому дозволило локальній моделі YOLO11n досягти промислових стандартів точності без ручного втручання експертів. Даний підхід маркує фундаментальне зміщення парадигми від класичного інжинірингу неймережевих архітектур (Model-Centric approach) до систематичного й автоматизованого керування якістю навчальних вибірок. Завдяки використанню open-vocabulary детектора YOLO-World, який базується на вирівнюванні мовних та візуальних ембеддингів, система отримала здатність трансформувати текстові назви номенклатурних позицій безпосередньо з реляційної СКБД у точні просторові координати обмежувальних рамок у режимі Zero-Shot. Це дозволило повністю усунути найбільш критичне «вузьке місце» розгортання сучасних систем штучного інтелекту – тривалий, фінансово витратний та схильний до суб'єктивних помилок процес ручної анотації графічних даних.

4 ІНТЕРФЕЙС КОРИСТУВАЧА ТА ТЕСТУВАННЯ МОДЕЛЕЙ

4.1 Інтерфейс користувача (UI)

Інтерфейс розробленої інтелектуальної системи автоматизації складського обліку спроектований у вигляді адаптивного веб-застосунку з оптимізацією під мобільні термінали збору даних (ТСД) та смартфони персоналу. Клієнтська частина базується на поєднанні мови розмітки HTML5, каскадних таблиць стилів CSS3 та асинхронної логіки на мові JavaScript, що забезпечує гнучкість розгортання та швидке опрацювання візуальних елементів без затримок. Головне робоче вікно системи збудовано за модульним принципом: ліворуч розташовані елементи виклику ключових підсистем, а центральна область виділена під інтерактивні вікна для відображення результатів комп'ютерного зору, трансляцій відеопотоків та генерації звітів.

Після проходження авторизації користувач отримує доступ до панелі керування, яка містить три основні робочі екрани та функціональні етапи інвентаризації:

- екран «Дашборд» (Dashboard): відображає зведену аналітику складу, поточний стан MLOps-конвеєра, критичні залишки ТМЦ та статус активності фонових процесів донавчання моделей ШІ;

- екран «Каталог товарів» (Products): надає інтерфейс для перегляду повної номенклатури, редагування артикулів (SKU), динамічних атрибутів у форматі JSONB та ручного коригування кількості одиниць;

- екран «Сканування» (Scan): виступає основним інструментом проведення ревізії, забезпечуючи інтеграцію з апаратною камерою пристрою, гібридне сканування штрих-кодів та детекцію товарів (YOLO11 / YOLO-World) (рисунок 4.1);

- екран «Перенавчання» (Retraining): забезпечує повний контроль над життєвим циклом моделей глибокого навчання та процесами Data-Centric

AI; відображає чергу номенклатурних позицій без актуальних ML-міток із їхнім поточним статусом детекції (рисунок 4.2).

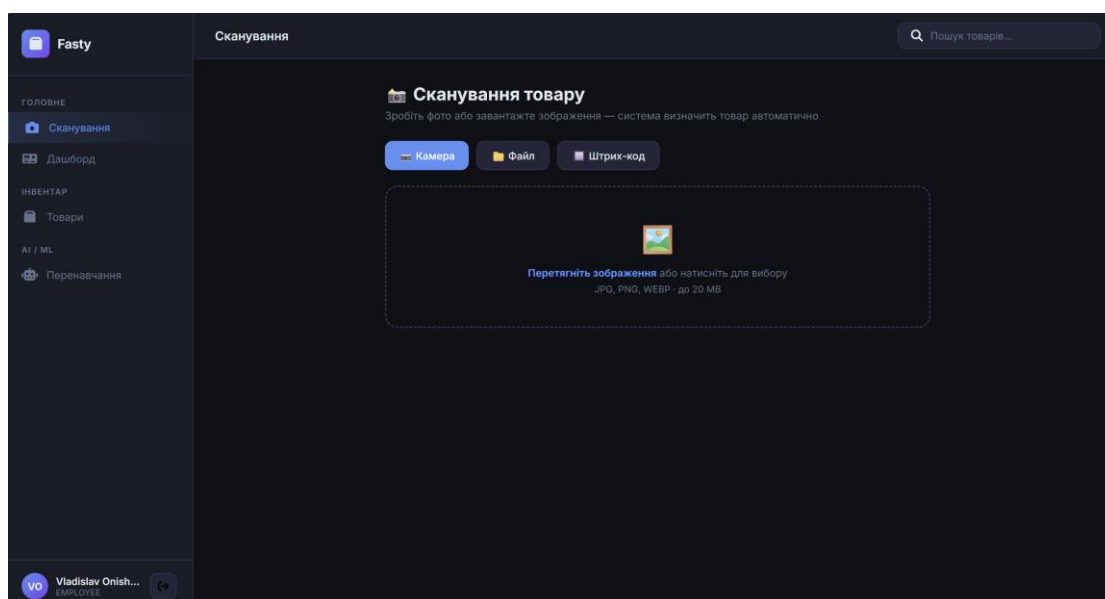


Рисунок 4.1 – Графічний інтерфейс користувача для завантаження зображень товарів та вибору методів сканування

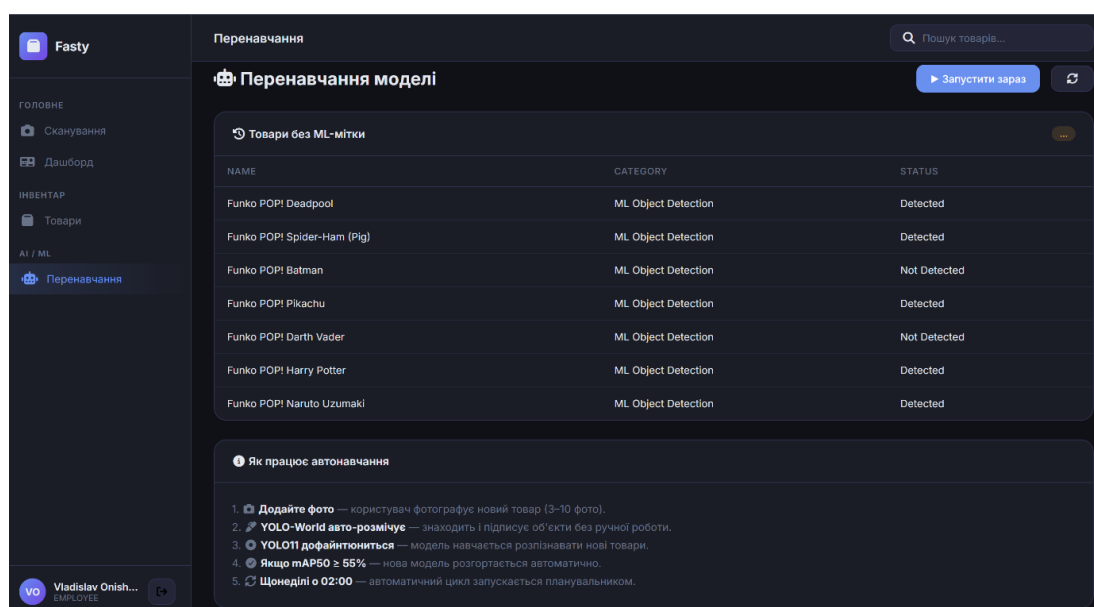


Рисунок 4.2 – Панель керування MLOps-конвеєром та моніторингу черги планового перенавчання моделей у системі «Fasty»

Логіка автоматичного сканування штрих-кодів у реальному часі реалізована через циклічне захоплення кадрів із відеопотоку HTML5-елемента `<video>`. Кадри з періодичністю у 400 мс перетворюються у бінарні об'єкти (Blob) та асинхронно відправляються на REST API ендпоінт обчислювального мікросервісу. Програмний код обробника процесу автоматичного сканування, який є одним із центральних алгоритмів клієнтського інтерфейсу, наведено у лістингу 4.1.

Лістинг 4.1 – Програмний код для автоматичного сканування та обробки кадрів

```
barcodeInterval = setInterval(async () => {
    if (video.paused || video.ended) return;
    canvas.width = video.videoWidth;
    canvas.height = video.videoHeight;
    ctx.drawImage(video, 0, 0, canvas.width, canvas.height);
    canvas.toBlob(async (blob) => {
        if (!blob) return;

        try {

            const res = await API.barcode.scan(blob);

            if (res.found) {
                stopAutoScanning();
                stopCamera();
                handleBarcodeResult(res.barcodes[0].data);
            }
        } catch (err) {
        }

        }, 'image/jpeg', 0.7);

    }, 400);
```

Усі результати аналізу візуалізуються безпосередньо поверх графічного інтерфейсу у реальному часі: знайдені штрих-коди та об'єкти підсвічуються накладеними двовимірними рамками, а розпізнані числові характеристики миттєво оновлюють текстові поля карток товарів завдяки маніпуляціям з DOM-деревом сторінки.

4.2 Інтеграція моделей у загальний конвеєр складського обліку

Розроблена мікросервісна система забезпечує наскрізну автоматизацію обліку товарно-матеріальних цінностей (ТМЦ) на стелажах, мінімізуючи вплив людського фактора під час проведення інвентаризації та реєстрації нових номенклатурних одиниць. Завдяки гібридному поєднанню алгоритмів комп'ютерного зору та моделей глибокого навчання, процес аналізу та верифікації кожного об'єкта побудований як послідовний багатовекторний конвеєр (Processing Pipeline), що дозволяє гнучко розподіляти обчислювальні ресурси сервера й оптимізувати повсякденну роботу персоналу.

Процес обробки вхідних даних у межах конвеєра реалізовано за такими послідовними етапами:

- первинна інтелектуальна детекція об'єктів (YOLO11n): після активації камери на мобільному пристрої або ТСД система насамперед надсилає зображення на обчислювальний мікросервіс FastAPI. Нейромережа YOLO11n намагається автоматично виявити, ідентифікувати та підрахувати кількість одиниць товару на кадрі за специфічними візуальними ознаками (наприклад, Funko-pop-Deadpool, Funko-pop-Notaru тощо). Якщо об'єкти успішно розпізнано, модель повертає масив координат обмежувальних рамок, а логічний сервер Java Spring Boot миттєво оновлює інвентаризаційну відомість у СКБД PostgreSQL;

- альтернативне сканування штрих-кодів (PyZbar): у разі, якщо нейромережева модель не змогла детектувати об'єкт на кадрі (наприклад,

через відсутність товару в локальних вагах або зміну дизайну пакування), конвеєр переходить у режим ручного захоплення, де працівник складу виконує сканування лінійного штрих-коду. Клієнтська частина через сервіс `API.barcode.scan` декодує маркування за допомогою бібліотеки `PyZbar`, зв'язується з реляційною базою даних для ідентифікації товарної позиції та виконує точковий інкремент кількості ТМЦ;

– акумуляція даних та планове донавчання: для запобігання подібним пропускам у майбутньому, після сканування штрих-коду нерозпізаного товару оператор за допомогою функції `addTrainingImage` додає серію нових фотографій об'єкта з різних ракурсів безпосередньо у базу даних та об'єктне сховище `MinIO`. Система реєструє ці знімки у фоновому `MLOps`-сервісі анотацій, де за допомогою моделі з відкритим словником `YOLO-World` автоматично генеруються нормалізовані `.txt` файли розмітки. Накопичені графічні та текстові дані стають основою для щотижневого `Fine-tuning` перенавчання моделі за розкладом (`on schedule`).

Усі результати опрацювання – включаючи артикули, штрих-коди, розпізані класи товарів, поточну кількість на стелажі та коефіцієнти впевненості ШІ – консоліднуються на логічному рівні сервера `Java Spring Boot`. Якщо під час первинної детекції точність розпізнавання ШІ падає нижче критичного порогу (0.55), система виставляє прапорець `needs_review=True`. Це сигналізує логічному рівню про необхідність верифікації даного кадру оператором складу, що запобігає потраплянню хибних або зашумлених анотацій у підсумковий навчальний набір даних до моменту запуску планового перенавчання.

4.3. Експериментальне тестування програмного комплексу та верифікація моделей комп'ютерного зору

Експериментальна верифікація та тестування розробленої інтелектуальної системи складського обліку проводилися з метою перевірки

працездатності мікросервісної архітектури, оцінки транзакційної стабільності логічного ядра на базі Java Spring Boot та вимірювання точності нейромережевого конвеєра обробки візуальних даних у середовищі Python FastAPI. Тестування інтеграційної взаємодії компонентів охоплювало перевірку REST API endpoints, асинхронного передавання бінарних медіафайлів з клієнтських мобільних терміналів, а також коректності збереження метаданих та маркувальних прапорців (зокрема, атрибуту ручного перегляду `needs_review`) у СКБД PostgreSQL. Окрему увагу було приділено стрес-тестуванню підсистеми збереження даних під час інтенсивного завантаження графічних масивів невідомої номенклатури в об'єктне сховище MinIO.

Функціональне тестування підсистеми комп'ютерного зору фокусувалося на оцінці стійкості інференсу локальної моделі YOLO11n. Модель продемонструвала стабільність геометричної апроксимації обмежувальних рамок навіть у випадках, коли кут зйомки оператора суттєво відхилявся від перпендикулярної осі.

Для демонстрації практичної ефективності алгоритмів детекції у реальних умовах експлуатації було проведено серію експериментальних запусків. На рисунку 4.1 наведено приклад успішної просторової локалізації та класифікації об'єкта сувенірної продукції класу `mini-crafts-baby-ogre`. Як свідчать результати інференсу, модель безпомилково згенерувала координати двовимірної обмежувальної рамки та присвоїла цільовий класифікаційний маркер із високим інтегральним коефіцієнтом впевненості (`Confidence Score = 0.93`), що підтверджує надійність розпізнавання дрібних товарних одиниць на фоні складних текстур складського середовища. Отримане значення показника впевненості, яке суттєво перевищує критичний поріг валідації, дозволило системі автоматично синхронізувати дані із транзакційним ядром без необхідності додаткового ручного сканування матричних кодів.



Рисунок 4.3 – Графічний інтерфейс додатка під час успішного тестування інференсу моделі YOLO11n

Статистичний аналіз результатів тестування на всій контрольній вибірці показав, що загальний інтегральний показник середньої точності моделі досяг значення $mAP_{50} = 0.955$, а повнота виявлення (Recall) склала 0.989 при значенні $mAP_{50-95} = 0.962$. Покласове оцінювання продемонструвало стабільно високі результати для всіх тестованих категорій, включаючи суміжні бренди Funko-pop-Deadpool та Funko-pop-Notaru, що свідчить про мінімізацію ризиків пересортиці та пропусків ТМЦ під час проведення автоматизованого аудиту складських залишків. Таким чином, експериментальне тестування підтвердило повну відповідність розробленого програмного продукту встановленим технічним вимогам, високу швидкість обробки кадрів та готовність інтелектуальної системи до промислової експлуатації на підприємстві.

ВИСНОВКИ

Результатом розробки є мікросервісна інтелектуальна система, яка стала відповіддю на нагальну потребу автоматизації та підвищення точності обліку товарно-матеріальних цінностей (ТМЦ) на сучасних логістичних комплексах і складах. Робота над проєктом включала ґрунтовне дослідження предметної галузі, аналіз класичних засобів автоматизації (WMS/ERP), а також вивчення викликів, із якими стикається персонал під час проведення ручних або напівавтоматичних інвентаризацій (пропуски номенклатури, затирання маркування, трудовитрати на розмітку нових надходжень). Це допомогло не лише зрозуміти проблему, а й запропонувати багаторівневе рішення, яке реально працює та оптимізує бізнес-процеси підприємства.

Окрім того, застосування таких інструментів сприяє ефективнішому управлінню ланцюгами постачань, мінімізації логістичних помилок та прискоренню перевірки залишків, що напряду підтримує цифровізацію та оптимізацію індустріальних секторів економіки. Це підкреслює значення технологій глибокого навчання не лише в контексті операційного менеджменту, а й як стратегічного інструменту для довгострокового сталого розвитку цифрової комерції та промислової автоматизації.

Взаємодія компонентів системи всередині розробленого конвеєра обробки даних (Processing Pipeline) побудована навколо трьох послідовних технологічних етапів:

- первинна інтелектуальна детекція об'єктів на основі локальної неймережі архітектури YOLO11n;
- альтернативне класичне сканування лінійних штрих-кодів за допомогою декодера PyZbar та інструментів OpenCV;
- інтелектуальна розмітка та MLOps-контроль невідомих товарів з використанням Zero-Shot моделі з відкритим словником YOLO-World.

Така послідовність обробки дозволила реалізованій системі:

- швидко виконувати паралельний підрахунок та розпізнавання наявних товарів на стелажі за допомогою комп'ютерного зору;
- автоматично переходити до сканування штрих-коду персоналом у разі відсутності ваг для об'єкта в локальній моделі детекції;
- надавати операторам зручний інтерфейс додавання фотографій нерозпізнаних ТМЦ (`addTrainingImage`), автоматично маркувати їх за текстовими промптами з бази даних і накопичувати у сховищі MinIO для планового перенавчання;
- формувати актуальні інвентаризаційні відомості, оновлювати кількісні показники залишків у СКБД PostgreSQL та маркувати зашумлені кадри прапорцем ручного перегляду (`needs_review`) при падінні впевненості ШІ нижче 0.55.

Обчислювальні моделі були розгорнуті в асинхронному середовищі Python FastAPI. Експериментальна валідація YOLO11n після 100 епох навчання підтвердила високу точність: загальна точність (Box P) склала 0.855, а повнота виявлення (Recall) досягла 0.989 при інтегральних показниках $mAP_{50} = 0.995$ та $mAP_{50-95} = 0.962$. Покласовий аналіз брендів (Funko-pop-Deadpool, Funko-pop-Hotaru, mini-crafts-baby-ogre) довів здатність моделі успішно узагальнювати візуальні ознаки та повністю виключати пропуски продукції.

Для побудови логічного ядра та оркестрації транзакцій використано фреймворк Java Spring Boot та Spring Data JPA, що у зв'язці з адаптивним веб-інтерфейсом (HTML5, CSS3, JavaScript) забезпечило стабільну роботу на мобільних терміналах збору даних (ТСД). Клієнтська частина спроектована інтуїтивно зрозумілою, працює через асинхронні HTTP-запити до REST API та візуалізує обмежувальні рамки виявлених об'єктів у реальному часі.

Дослідження підтвердило, що гібридне поєднання нейромереж із відкритим словником та класичних методів декодування є потужним архітектурним підходом для обробки великих потоків складських

медіаданих. Завдяки автоматизованому MLOps-конвеєру розмітки та плановому донавчанню моделі за розкладом (on schedule), система демонструє високу адаптивність до постійної зміни та розширення асортименту товарів без потреби залучення сторонніх фахівців з розмітки даних.

Розробка враховувала практичні потреби кінцевих користувачів:

- потребу у миттєвому автоматичному підрахунку однотипних товарів на полицях;
- усунення тривалих затримок при внесенні нових надходжень до облікових реєстрів;
- запит на кросплатформений, легкий веб-додаток, інтегрований із поточною базою даних підприємства.

Для реалізації системи було обрано технології, що поєднують промислову надійність та масштабованість. Такий підхід зробив систему повністю модульною та готовою до розгортання як локально (On-Premise), так і в хмарних інфраструктурах (AWS, Google Cloud).

У процесі роботи над проектом сформувалося чітке розуміння вимог до сучасного програмного продукту: він має бути модульним для підтримки додавання нових типів маркувань; транзакційно стабільним для уникнення збоїв у кількісних залишках; та аналітично глибоким для надання точних даних менеджменту компанії.

У підсумку, розроблена система не лише довела свою ефективність у вирішенні поставлених завдань, а й стала прикладом успішного впровадження методів штучного інтелекту у сферу складської логістики. Система має значний потенціал для подальшого розвитку: її логіку можна інтегрувати з великими корпоративними ERP-платформами (SAP, Oracle WMS), підключити брокери повідомлень (Apache Kafka) для обробки потокового відео з купольних камер складу в реальному часі або додати модулі предиктивної аналітики для прогнозування попиту на товари на основі динаміки зміни залишків.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Dewa P. K., Pujawan I. N. Human errors in warehouse operations: an improvement model. *International Journal of Logistics Systems and Management*. 2017. Vol. 27, no. 3. P. 298–317. URL: <https://doi.org/10.1504/IJLSM.2017.084468> (date of access: 12.06.2026).
2. How's Ukraine's warehouse market functioning during the war?. *Property Forum*. 2022. 3 July. URL: <https://www.property-forum.eu/news/how-ukraines-warehouse-market-functioning-during-the-war/12743> (date of access: 10.06.2026).
3. Ukraine Emergency. *Logistics Cluster*. URL: <https://logcluster.org/en/wherewe-work/ukr/ukraine> (date of access: 15.06.2026).
4. Lightweight object detection model for food freezer warehouses / J. Yang et al. *Scientific Reports*. 2025. URL: <https://doi.org/10.1038/s41598-025-86662-z> (date of access: 16.06.2026).
5. Aurora to rebuild Kyiv logistics hub with \$63M investment. *Inventure*. URL: <https://inventure.com.ua/en/news/ukraine/aurora-to-rebuild-kyiv-logistics-hub-with-dollar63m-investment> (date of access: 12.06.2026).
6. Revolutionizing warehouse management: how AI drives operational efficiency. *IJARESM*. 2023. Vol. 11, no. 11. P. 1855–1861. URL: https://www.researchgate.net/publication/389516065_Revolutionizing_Warehouse_Management_How_AI_Drives_Operational_Efficiency (date of access: 20.06.2026).
7. Microsoft COCO: common objects in context / T.-Y. Lin et al. *arXiv*. 2014. 19 c. URL: <https://arxiv.org/abs/1405.0312> (date of access: 22.06.2026).
8. Building damage assessment in conflict zones: a deep learning approach using geospatial sub-meter resolution data / M. Risso et al. *IEEE. 2025. 6th International Conference on Image Processing, Applications and Systems (IPAS)*. Lyon, France, 2025. P. 1–6.

URL: <https://doi.org/10.1109/IPAS63548.2025.10924534>. (date of access: 22.06.2026).

9. Ukraine – annual operation overview, 2024. *Logistics Cluster*. January 2025. URL: <https://logcluster.org/en/node/47321> (date of access: 18.06.2026).

10. Xu Y., Qu Z., Zheng S. High-speed barcode recognition system based on OpenCV and Zbar. Proc. SPIE 11179, *Eleventh International Conference on Digital Image Processing* (ICDIP 2019). 2019. URL: <https://doi.org/10.1117/12.2539756>. (date of access: 22.06.2026).

11. Cybertec. entity-attribute-value (EAV) design in PostgreSQL – don't do it! *Cybertec PostgreSQL*. URL: <https://www.cybertec-postgresql.com/en/entity-attribute-value-eav-design-in-postgresql-dont-do-it/> (date of access: 23.06.2026).

12. Object detection YOLO algorithms and their industrial applications: overview and comparative analysis / S. Kang et al. *Electronics*. 2025. Vol. 14, no. 6. P. 1104. URL: <https://doi.org/10.3390/electronics14061104>. (date of access: 23.06.2026).

13. Anyline mobile scanning sdk for logistics. *Anyline*. URL: <https://anyline.com/industries/logistics> (date of access: 03.04.2026).

14. Modernizing Ukraine's transport and logistics infrastructure. *CSIS*. URL: <https://www.csis.org/analysis/modernizing-ukraines-transport-and-logistics-infrastructure> (date of access: 15.06.2026).

15. Anyline barcode scanner SDK. *Anyline*. URL: <https://anyline.com/news/barcode-scanner> (date of access: 11.04.2026).

16. Real-time object detection and counting for inventory management using fine-tuned YOLOv11. *ResearchGate*. URL: https://www.researchgate.net/publication/402816360_Real-time_object_detection_and_counting_for_inventory_management_using_fine-tuned_YOLOv11 (date of access: 23.06.2026).

17. Coussej J. Replacing EAV with JSONB in PostgreSQL. *GitHub*. 2016. URL: <https://coussej.github.io/2016/01/14/Replacing-EAV-with-JSONB-in-PostgreSQL/> (date of access: 15.06.2026).

18. Scientia report academic journal archive issue (12.12.2025). *Scientia Report Platform*. URL: <https://previous.scientia.report/index.php/archive/issue/view/12.12.2025> (date of access: 19.06.2026).