КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій

(кафедра)

КВАЛІФІКАЦІЙНА РОБОТА ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

на тему: «Розробка інформаційної системи управління запасами розподільчого центру сільськогосподарської продукції»

КОВТЮХ МАКСИМ ВІТАЛІЙОВИЧ

(прізвище, ім’я та по батькові студента повністю)

Київ – 2025р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ БУДІВНИЦТВА І АРХІТЕКТУРИ

автоматизації і інформаційних технологій

(факультет)

інформаційних технологій

(кафедра)

ЗАТВЕРДЖУЮ
Завідувачка кафедри ІТ
д.т.н., доцент Гончаренко Т.А.

„___” _____2025 року

КВАЛІФІКАЦІЙНА РОБОТА ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

на тему: «Розробка інформаційної системи управління запасами розподільчого центру сільськогосподарської продукції»

Я як здобувач вищої освіти КНУБА розумію і підтримую політику закладу з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволену допомогу під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Здобувач Ковтюх Максим Віталійович
(прізвище, ім’я та по батькові повністю)
122 «Комп’ютерні науки»
(спеціальність)
Інформаційні управляючі системи і технології
(освітня програма)
Групи КН-21-2
Керівник Саченко І. А.
(прізвище та ініціали)
доцент, кандидат технічних наук
(вчене звання, науковий ступінь)
Рецензент к.т.н., доц Баліна О.І.
(Прізвище та ініціали)

Ідентичність підтверджую

Київ, 2025 р.

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Факультет: автоматизації і інформаційних технологій
Випускова кафедра: інформаційних технологій
Ступінь вищої освіти: «бакалавр»
Спеціальність: 122 «Комп`ютерні науки»
Освітня програма: Інформаційні управляючі системи і технології

ЗАТВЕРДЖУЮ
Завідувачка кафедри ІТ
д.т.н., доцент Гончаренко Т.А.
_____
„___” _____2025 року

З А В Д А Н Н Я
ДО ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ
ЗДОБУВАЧА СТУПЕНЯ ВИЩОЇ ОСВІТИ БАКАЛАВР

Ковтюх Максим Віталійович
1. Тема роботи: Розробка інформаційної системи управління запасами розподільчого центру сільськогосподарської продукції
затверджена наказом ректора КНУБА № 2650/2 від 18.11.2025.
2. Керівник роботи: Саченко Ілля Анатолійович, к.т.н, доцент кафедри інформаційних технологій
3. Строк подання студентом роботи до захисту:
4. Зміст пояснювальної записки за розділами:
Р.1. Аналіз предметної області та постановка задачі
Р.2. Методи та моделі оптимізації управління запасами
Р.3. Програмна реалізація інформаційної системи управління запасами
Р.4. Техніко-ергономічне обґрунтування розробки підсистеми
5. Інформаційні слайди:
С.1. Причини дослідження та актуальність
С.2. Мета та елементи дослідження
С.3. Аналіз існуючих інформаційних систем
С.4. Висновки з аналізу ІС
С.5. Комплексне рішення для ефективної агрологістики: загальний підхід
С.6. Аналіз існуючих моделей та методів
С.7. Запропоновані моделі та алгоритми

6. Консультанти розділів кваліфікаційної випускної роботи

Розділ	Прізвище, ініціали та посада консультанта, представника комісії	дата	підпис
Ергономіка інформаційних технологій	доц. Рябчун Ю.В.		
Прийом програмного продукту	ас. Мацієвський О.О.		

7. Календарний план виконання кваліфікаційної випускної роботи

Види робіт та їх зміст	Дата виконання
Р. 1. Аналіз предметної області та постановка задачі	Січень 2025 р.
Р. 2. Моделювання та планування	Лютий 2025 р.
Р. 3. Розробка програмного забезпечення	Травень 2025 р.
Тестовий приклад програми	Травень 2025 р.
Р. 4. Ергономіка інформаційних технологій	Травень 2025 р.
Остаточне оформлення роботи	Травень 2025 р.
Направлення роботи на рецензування	Червень 2025 р.
Попередній захист роботи на кафедрі	Червень 2025 р.

8. Дата видачі завдання: 20.01.2025 р.

Завідувачка _____ Гончаренко Т.А.
(підпис) (прізвище та ініціали)

Керівник _____ Саченко І.А.
(підпис) (прізвище та ініціали)

Студент _____ Ковтюх М.В.
(підпис) (прізвище та ініціали)

<table>
<tr><td>РЕЗЮМЕ
(SUMMARY)
до кваліфікаційної випускної роботи Здобувача:</td><td colspan="3">Ковтюх Максим Віталійович

Kovtiukh Maksym</td></tr>
<tr><td>ЗВО</td><td colspan="3">Київський національний університет будівництва і архітектури</td></tr>
<tr><td>Тема (українською та англійською)</td><td colspan="3">Розробка інформаційної системи управління запасами розподільчого центру сільськогосподарської продукції. Development of an information system for managing inventories of an agricultural distribution center.</td></tr>
<tr><td>Освітній ступінь</td><td colspan="3">Бакалавр</td></tr>
<tr><td>Факультет</td><td colspan="3">Автоматизації і інформаційних технологій</td></tr>
<tr><td>Випускова кафедра</td><td colspan="3">Інформаційних технологій</td></tr>
<tr><td>Спеціальність</td><td colspan="3">122 «Комп’ютерні науки»</td></tr>
<tr><td>Освітня програма</td><td colspan="3">Інформаційні управляючі системи та технології</td></tr>
<tr><td>Керівник</td><td colspan="3">Саченко Ілля Анатолійович</td></tr>
<tr><td rowspan="2">Обсяг роботи:</td><td>пояснювальна записка, стор.</td><td>розділів</td><td>креслень формату А</td></tr>
<tr><td>79</td><td>4</td><td>0</td></tr>
<tr><td>Розділ 1.</td><td colspan="3">Аналіз предметної області та постановка задачі</td></tr>
<tr><td>Розділ 2.</td><td colspan="3">Моделювання та планування</td></tr>
<tr><td>Розділ 3.</td><td colspan="3">Розробка програмного забезпечення</td></tr>
<tr><td>Розділ 4.</td><td colspan="3">Ергономіка інформаційних технологій</td></tr>
<tr><td>Ключові слова:

Keywords:</td><td colspan="3">управління запасами, інформаційна система, складська логістика, автоматизація, прогнозування, штучний інтелект, сільськогосподарська продукція.

inventory management, information system, warehouse logistics, automation, forecasting, artificial intelligence, agricultural products.</td></tr>
</table>

Здобувач: _____ /Максим КОВТЮХ /

Керівник: _____/ Ілля Саченко /

“___” _____ 2025р.

АНОТАЦІЯ

Ковтюх М.В. Розробка інформаційної системи управління запасами розподільчого центру сільськогосподарської продукції.

Кваліфікаційна випускна робота бакалавра за спеціальністю: 122 «Комп'ютерні науки», освітньо–професійна програма: «Інформаційні управляючі системи і технології». – Київський національний університет будівництва та архітектури. – Київ, 2025.

Робота присвячена створенню інформаційної системи управління запасами сільськогосподарської продукції. Описується логістика складу, архітектура системи та основні модулі, які включають приймання, розміщення, комплектацію та відвантаження. Результати тестування підтверджують функціональність, гнучкість і придатність системи до використання в агрологістиці.

Ключові слова: управління запасами, інформаційна система, складська логістика, автоматизація, прогнозування, штучний інтелект, сільськогосподарська продукція.

SUMMARY

Kovtiukh M. V. Development of an information system for managing inventories of an agricultural distribution center.

Bachelor's thesis for a bachelor's degree in specialty: 122 “Computer Science”, specialization: "Information Management Systems and Technologies – Kyiv National University of Construction and Architecture – Kyiv, 2025.

The work is devoted to the creation of an information system for managing agricultural product inventories. The warehouse logistics, system architecture and main modules, which include receiving, placing, picking and shipping, are described. The testing results confirm the functionality, flexibility and suitability of the system for use in agrologistics.

Keywords: inventory management, information system, warehouse logistics, automation, forecasting, artificial intelligence, agricultural products.

ЗМІСТ

ВСТУП

У сучасних умовах глобалізації та високої конкуренції підприємства аграрного сектору, зокрема розподільчі центри, які працюють з овочами та фруктами, змушені шукати нові підходи до підвищення ефективності логістичних процесів.

Обмежений термін зберігання, сезонність постачання, вимоги до температури та вологості, а також високі вимоги торговельних мереж до якості та стабільності поставок є особливостями управління запасами такої продукції.

В умовах постійно змінного попиту неефективне управління запасами може призвести до значних втрат, включаючи псування товарів, перевищення витрат на зберігання, неефективне використання складських площ і зниження рівня обслуговування клієнтів. Таким чином, впровадження сучасних цифрових продуктів, які дозволяють автоматизувати процеси планування, моніторингу та обліку запасів, є надзвичайно важливим.

Удосконалення засобів автоматизації та інформаційних технологій створює нові можливості для цифрової трансформації агрологістичних процесів. Сучасні інформаційні системи не тільки дозволяють оптимізувати управління складськими запасами, але й дозволяють контролювати умови зберігання, робити прогнози та швидко оновлювати дані про рух товарів.

Таким чином, для підвищення ефективності логістичних процесів, зниження витрат на виробництво, зменшення втрат і підвищення конкурентоспроможності підприємства на ринку необхідно впровадити інформаційну систему управління запасами в розподільчі центри сільськогосподарської продукції.

Мета роботи: розробка інформаційної системи управління запасами для розподільчого центру сільськогосподарської продукції, яка забезпечить підвищення ефективності логістичних процесів, зменшення втрат продукції та оптимізацію витрат на зберігання.

Об'єкт дослідження: логістичні процеси розподільчого центру, що спеціалізується на зберіганні та обліку сільськогосподарської продукції.

Предмет дослідження: процеси автоматизованого управління запасами та обліку продукції на складі за допомогою інформаційної системи.

Методи дослідження: системний та функціональний аналіз, методи оптимізації, методи моделювання.

У першому розділі проведено аналіз проблем управління запасами в розподільчих центрах сільськогосподарської продукції, зокрема овочів і фруктів. Визначено особливості логістичних процесів у цій галузі, основні виклики, пов'язані з сезонністю, обмеженим терміном зберігання та потребою в спеціальних умовах зберігання. Розглянуто потенціал інформаційних систем у подоланні цих проблем, виконано структурний, функціональний та параметричний аналіз систем, визначено вимоги до майбутньої інформаційної системи та сформульовано постановку задачі.

У другому розділі розглядаються сучасні методи та моделі управління запасами сільськогосподарської продукції, такі як EOQ, JIT, VMI, а також ABC/XYZ-аналіз і штучний інтелект для прогнозування попиту. Поєднання моделей штучного інтелекту з методами класифікації було виявлено ефективним для продукції з коротким терміном зберігання. Інформаційна модель була розроблена з акцентом на точності прогнозування, адаптивності до змін і інтеграції з логістичними процесами.

У третьому розділі описано програмну реалізацію інформаційної системи, побудованої на основі Flask та PostgreSQL, яка має веб-інтерфейс, інтегрований з емулятором ТЗД. Розглядалися архітектура, функції, логіка роботи, сценарії використання, інсталяція та тестування. Тестування підтвердило працездатність основних сценаріїв, але виявило кілька обмежень, які вимагають додаткового вдосконалення. Система, яка була розроблена, є масштабованою, гнучкою та готовою до використання з урахуванням мінімальних доробок.

У четвертому розділі розглядалися техніко-економічні та ергономічні аспекти проєкту. Встановлено стандарти інтерфейсу, зручності, стандартизації

та адаптивності. Оцінка ринку, конкуренти, план виробництва, організаційна структура та фінансова модель були надані. Аналіз показав, що впровадження системи є економічно доцільним, і витрати можуть бути компенсовані за рахунок підвищення точності обліку та зменшення втрат. створено план фінансування, враховуючи інвестиції та потенційні ризики.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Загальні поняття управління запасами в розподільчих центрах

Розподільчий центр – це склад чи інша спеціалізована будівля, що використовується для тимчасового зберігання товару з подальшим його перерозподілом між роздрібними торговцями, оптовиками та споживачами.

Розподільчі центри сільського господарства мають багато характеристик, які впливають на управління запасами. Серед них – дотримання найкращих умов зберігання, забезпечення швидкого переміщення продукції та врахування сезонності попиту та поставок.[18]

Основні проблеми з якими стикаються такі центри є:

- **Правильні умови зберігання** – різні категорії продукції вимагають специфічних температурних режимів, вологості та вентиляції. Недотримання цих параметрів призводить до псування товару.
- **Сезонність обсягів постачання та попиту** – обумовлюють необхідність гнучкого управління запасами та ефективного прогнозування потреб ринку.
- **Швидка ротація товарів за принципом FIFO (First In, First Out)** – для уникнення псування продукції необхідно контролювати терміни її зберігання та дотримуватися правил відвантаження.

Основні складові управління запасами:

1. Облік запасів
   - Використання WMS та ERP для контролю залишків.
2. Прогнозування попиту
   - Урахування сезонності та тренду споживання;
   - Методи прогнозування.
3. Автоматизація замовлень
   - Визначення економічної кількості замовлення(EOQ);
   - Автозамовлення постачальникам.

4. Безперервне постачання
   - Управління постачанням;
   - Використання страхових запасів на випадок дефіциту.
5. Ефективне зберігання
   - Впровадження системи розміщення товарів;
6. Контроль управління запасами
   - Проведення інвентаризацій;
   - Аналіз ефективності.

Зі збільшенням кількості складів оптової торгівлі, збільшується необхідність у розподільчих центрах задля покращення логістичних процесів зв'язку «постачальник – споживачі».

1.2 Використання інформаційних систем у логістиці та складському управлінні.

1.2.1 IMS(Inventory Management Systems – Інформаційні системи управління запасами).

Програмні рішення IMS автоматизують облік, рух, контроль і оптимізацію запасів на складі. Вони полегшують ефективне управління постачанням, зменшують витрати на зберігання та зменшують втрати продукції через псування.

До основних функцій таких систем належать:

- Облік товарів – реєстрація надходжень, переміщень, реалізації та списання запасів.
- Контроль залишків – автоматичне відстеження рівня запасів та нагадування про необхідність поповнення.
- Прогнозування попиту – використання аналітики для оцінки майбутніх потреб у запасах.

- Управління партіями та термінами придатності – контроль ротації продукції за методами FIFO (First In, First Out) або FEFO (First Expired, First Out).
- Автоматизація замовлень – автоматичне формування заявок постачальникам на основі встановлених параметрів.
- Інтеграція з іншими системами – взаємодія з ERP–системами, сканерами –терміналами, CRM–системами.

Прикладами подібних систем є WMS (Warehouse Management Systems), ERP–системи (SAP, 1C, Odoo). [19]

1.2.2 WMS система.

WMS система – це програмно–апаратна система для керування складом, яка забезпечує комплексну автоматизацію логістичних операцій бізнесу. Сюди входить прийом товару чи матеріалів складу, їх упаковка, зберігання, переміщення, інвентаризація та інтеграція з іншими учасниками операційного процесу. Система приймає та аналізує дані щодо кожного з цих процесів, а потім використовує їх для створення звітів, які згодом може переглянути керуючий складом.[1]

Основні функції WMS:

- **Контроль залишків** – система надає точну інформацію про кількість залишку товару на складі та його строк зберігання;
- **Контроль розміщення товару** – система аналізує товар і доступні місця зберігання, та заселяє в оптимальне;
- **Контроль прийому** – товар автоматично надходить до системи після підтвердження прийому, зручна звірка з накладними;
- **Контроль комплектування** – використання ТЗД(Термінал збору даних) та штрих–кодів зменшує кількість помилок людського фактору.
- **Контроль відвантаження** – сканування завантажених піддонів з товару, з подальшим друкуванням документів переміщення.

Найпростіший сценарій роботи WMS:

1. Прийом товару та його ідентифікація.
2. Розміщення.
3. Зберігання.
4. Відбір.
5. Упаковка.
6. Відвантаження.
7. Інвентаризація.

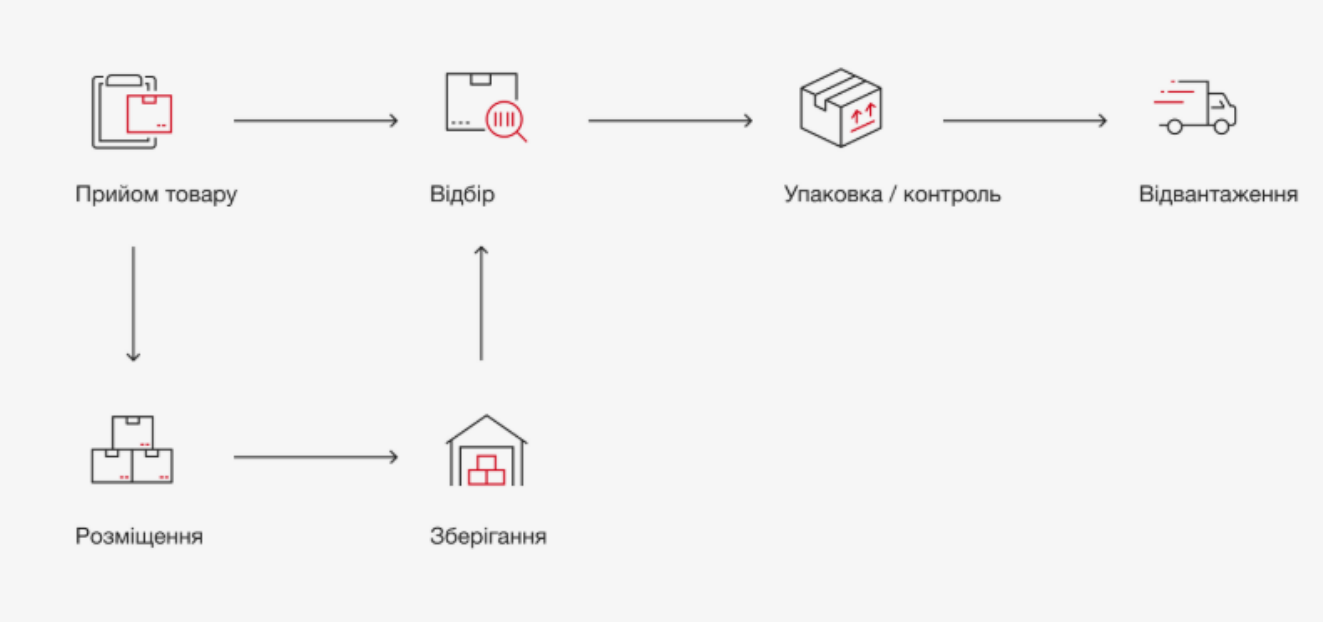

Рисунок 1.1 Базовий варіант роботи WMS.

Переваги впровадження WMS:

- Автоматизація процесів знижує потребу в ручній роботі та прискорює складські процеси;
- Завдяки використанню ТЗД, кількість помилок зводиться до мінімуму;
- Присутня можливість адаптувати під потреби складу, незалежно від його розміру;
- Система збирає дані про всі рухи товарів, що дає змогу, за необхідності, вести звітність.

Згідно з дослідженням Кочубея Дмитра В'ячеславовича[4] про впровадження системи управління складуванням «Infor Exceed WMS 4000» на українських підприємствах призвело до таких результатів:

- Підвищення оборотності товару на складі на 11,5%.
- Скорочення складського персоналу на 5,5%.
- Зменшення експлуатаційних витрат обладнання на 23%.
- Підвищення коефіцієнта використання складського простору на 20%.

Із результатів дослідження помітний значний потенціал підвищення ефективності складських операцій завдяки впровадженню WMS на українських підприємствах.

1.2.3 ERP–система(enterprise resource planning system – Планування ресурсів підприємства)

Для повноцінної роботи складу WMS поєднують з **ERP–системою** – це програмна система, яка допомагає організаціям оптимізувати основні бізнес–процеси, включаючи фінанси, управління персоналом, виробництво, ланцюжок поставок, продажу та закупівлі, з єдиним уявленням про діяльність та надає єдине джерело достовірної інформації.[2]

Класичні системи ERP забезпечують керування задачами:

- Керування фінансами;
- Керування виробництвом;
- Керування формуванням та розподілом запасів;
- Керування реалізацією та маркетингом;
- Керування утриманням покупців
- Керування постачанням;
- Керування проєктами;
- Керування сервісним обслуговуванням;
- Керування процедурами забезпечення якості продукції.

Переваги використання ERP для управління складом:

- **Єдина база даних** – ERP об’єднує інформацію про залишки, закупівлі, продажі, фінанси та логістику, що мінімізує дублювання даних і покращує аналітику.
- **Автоматизація обліку товарів** – кожна операція миттєво відображається в системі, що забезпечує актуальність даних у реальному часі.
- **Оптимізація логістики** – ERP дозволяє планувати закупівлі та поставки на основі аналітики попиту, сезонності та прогнозів.

Інтеграція ERP із WMS, ТЗД та іншими складськими технологіями створює **єдину систему управління запасами**, що дозволяє оперативно приймати рішення та мінімізувати ризики втрат продукції.

1.2.4 SCM–системи (Supply Chain Management – Управління ланцюгами постачання)

Системи SCM забезпечують координацію та оптимізацію процесів у всьому ланцюгу постачання, починаючи від закупівлі сировини і закінчуючи доставкою готової продукції клієнтам. Вони охоплюють взаємодію між логістичними операторами, складами, постачальниками та кінцевими споживачами. У розподільчих центрах, особливо в сільському господарстві, SCM має вирішальне значення для забезпечення своєчасного постачання товарів, мінімізації втрат і оптимізації транспортних витрат.[17]

Основні функції SCM–систем:

- планування попиту та постачання;
- контроль запасів уздовж усього ланцюга;
- управління замовленнями постачальників та клієнтів;
- оптимізація маршрутів доставки;
- моніторинг продуктивності постачальників;
- аналіз витрат на логістику та транспортування.

Переваги впровадження SCM–рішень:

- скорочення часу виконання замовлень;
- зменшення надлишкових запасів;
- прозорість руху товарів у реальному часі;
- покращення комунікації між усіма учасниками ланцюга.

У ланцюжку розподільчих центрів овочів і фруктів SCM дозволяє точно прогнозувати постачання в залежності від сезону, оптимізувати витрати на транспортування холодних продуктів і гарантувати відповідність умов зберігання на кожному етапі шляху продукту. Крім того, вона може працювати з системами контролю температури, моніторингом вантажів за допомогою GPS і обміном даними з агропостачальниками.

У той же час для малих і середніх підприємств часто потрібна спрощена або адаптована версія рішень SCM, які інтегруються з внутрішніми WMS і ERP-модулями. У таких ситуаціях краще використовувати модульні SaaS-рішення з відкритим API або створити власну систему [11].

Таким чином, SCM-системи є важливим інструментом для досягнення високої ефективності в ланцюгах постачання, особливо коли йдеться про продукти з обмеженим терміном зберігання. Інтегроване планування, Vendor-Managed Inventory і Just-in-Time стратегії в цифровій логістиці можна реалізувати за допомогою їх використання.

1.2.5 BI–системи (Business Intelligence – Бізнес–аналітика)

BI-системи є аналітичними платформами, які перетворюють необроблені дані у візуальну інформацію, яка допомагає управлінцям приймати розумні рішення. У логістиці та розподільчих центрах BI використовується для візуального представлення ключових логістичних процесів, виявлення відхилень, побудови прогнозів і моніторингу KPI. [20]

Основні функції BI–систем:

- зведена аналітика за залишками, обігом, попитом;

- інтерактивні дашборди з показниками продуктивності;
- автоматична генерація звітів за періодами;
- порівняння прогнозованих і фактичних показників;
- аналіз трендів, сезонності, втрат та відхилень.

BI в агрологістиці дозволяє швидко реагувати на зміни попиту, адаптувати логістичні стратегії до погодних змін і виявити неефективні процеси. Можливість інтеграції з WMS, ERP і системами збору даних дозволяє бізнес-аналізу надавати комплексне бачення роботи складу.

Системи BI особливо корисні для аналітиків і керівників, оскільки вони дозволяють не тільки бачити поточні дані, але й прогнозувати наслідки рішень, які приймаються. У той же час для їх використання потрібно мати якісну базу даних і базові навички роботи з аналітичними інструментами.[10]

Таким чином, системи BI є важливою частиною сучасної інформаційної інфраструктури складу. Вони доповнюють функції WMS та ERP, надаючи глибше розуміння поточних процесів і можливостей їх покращення.

1.2.6 Аналіз існуючих інформаційних рішень: переваги та недоліки

Часто використовувані в логістиці та складському управлінні інформаційні системи, такі як IMS, WMS, ERP, SCM і BI, відіграють важливу роль у цифровізації процесів управління запасами. Тим не менш, кожен із них має обмеження, які потрібно враховувати при розробці власного рішення для розподільчого центру, окрім функціональних переваг.

Таблиця 1.2.6.1 Порівняльна таблиця ІС

Тип системи	Основні переваги	Основні недоліки
IMS (Inventory Management System)	• Оперативне управління запасами • Простота реалізації • Підтримка ABC/XYZ–аналізу	• Обмежені можливості інтеграції • Недостатньо гнучка при масштабуванні • Відсутність глибокої аналітики
WMS (Warehouse Management System)	• Управління рухом товару на складі • Підтримка температурних зон • Висока точність обліку	• Складне впровадження • Потребує навчання персоналу • Низька інтегрованість із SCM
ERP–система	• Централізоване планування ресурсів • Інтеграція фінансів, постачання і продажу • Уніфікація обліку	• Висока вартість • Складність налаштування • Недостатня адаптація до агросектору
SCM–система	• Контроль усього ланцюга постачання • Оптимізація логістичних маршрутів • Прозорість операцій	• Слабке покриття внутрішньоскладських процесів • Високі вимоги до даних • Складність налаштування
BI–системи	• Візуалізація даних • Прогнозування трендів • Підтримка прийняття рішень	• Залежність від якості даних • Потребує інтеграції з іншими системами • Не здійснює управлінських дій напряму

1.3 Сучасні технології для реалізації інформаційної системи

1.3.1 База даних(БД)

Сучасні бази даних є основою для зберігання, обробки та аналізу даних у інформаційних системах управління запасами розподільчих центрів сільськогосподарської продукції.

В управлінні складом, основні пункти, що обов'язково повинна містити база даних це інформація про:

- Назва товару;
- Штрих–код;
- Термін зберігання;
- Логістичні дані товарів;
- Залишки товару на складі;
- Замовлення та відвантаження;
- Інформацію про постачальників.

Рекомендується використовувати реляційну базу даних на основі SQL для типової системи обліку товарів, яка містить основну інформацію про товари та постачальників. Це обґрунтовується структурованою табличною моделлю реляційних БД, яка забезпечує чіткі зв'язки між елементами. Використання запитів SQL дозволяє швидко шукати, фільтрувати та оновити дані.

Документоорієнтована БД краща для швидкої обробки та зберігання великої кількості даних. Це дозволить зберігати дані у гнучкому форматі, що стане зручно при роботі з даними, які постійно змінюються. [21, 22]

1.3.2 Використання IoT (Internet of Things)

Навіть тимчасове зберігання сільськогосподарських товарів вимагає особливих умов зберігання. Інтернет речей є перспективним рішенням для контролю зберігання товарів. Найкращим методом вирішення проблеми є

використання датчиків температури та вологості в камерах зберігання, які передають дані в режимі реального часу до інформаційної системи. Контроль за показниками відбувається, як з людським втручанням, так і автоматизовано.

Без ефективної системи ідентифікації товарів автоматизація складських процесів є неможливою. Штрих-коди та ТЗД для їх зчитування є одними з найпоширеніших рішень цієї проблеми. Штрих-коди полегшують прийом і відвантаження товарів, оскільки вони дозволяють швидко ідентифікувати товари. Це значно зменшує ймовірність помилок, пов’язаних із неправильною комплектацією або забутим товаром, який повинен бути відправлений. Автоматизація цього процесу прискорює роботу складу, покращує точність обліку та зменшує вплив людей. [23]

Визначення цілей дослідження

У центрах розподілу сільськогосподарської продукції ефективне управління запасами є важливим для стабільності постачань, мінімізації втрат і оптимізації логістичних операцій.

Основна мета дослідження – розробка інформаційної системи управління запасами. Ця система допоможе підвищити продуктивність складських операцій, автоматизуючи облік, аналіз і контроль товарних запасів.

Для досягнення цієї мети визначено такі основні завдання:

- **Проведення аналізу** сучасних підходів до управління запасами та їх застосування в розподільчих центрах, що працюють з овочами та фруктами;
- **Оцінка** існуючих **інформаційних систем** із автоматизації складських процесів та визначити їх переваги та недоліки;
- **Розробка концептуальної моделі інформаційної системи** управління запасами для визначення основних компонентів системи, їх взаємодії та функціональних можливостей

- **Формування** основних **вимог** до **інформаційної системи** та визначення необхідного функціоналу для її реалізації.
- **Дослідження**, що стосується інтеграції **IoT та автоматизації** складських **процесів**, зосереджується на можливостях використання штучного інтелекту, сенсорів температури та вологості для управління умовами зберігання продукції.
- **Тестування та оцінка** ефективності розробленої **ІС** передбачає впровадження системи для оцінки потенційних швидкостей роботи, втрат і точності обліку.

Очікується, що результати дослідження сприятимуть:

- **Зменшення помилок при обліку товарів** – пов’язано з тим, що автоматизація та використання технологій штрих-кодів і RFID значно скоротить кількість помилок, пов’язаних із неправильним обліком або забутими товарами.
- **Покращення умов зберігання продукції** – використання IoT для контролю температури та вологості знижує ризик псування товарів
- **Оптимізація процесів** – прийом, зберігання та відбір і комплектація замовлень будуть оптимізовані, щоб використовувати складський простір ефективніше.

Таким чином, впровадження інформаційної системи управління запасами в розподільчих центрах покращить логістичні та складські процеси та сприятиме розвитку компанії.

1.4 Аналіз системи «Управління запасами»

1.4.1 Класифікація систем

1. **За типом системи:**
   - Автономні – працюють без інтеграції з іншими підсистемами;

- Інтегровані – працюють у загальній інформаційній системі підприємства, інтегруючи дані з інших підсистем.

2. **За методом управління запасами:**
   - Періодичні – запаси перевіряються на певні проміжки часу;
   - Безперервні – система здійснює постійний моніторинг запасів і оновлює їхній рівень у реальному часі.
3. **За типом обробки інформації:**
   - Ручні – облікові та управлінські функції виконуються вручну;
   - Автоматизовані – використовують сучасні технології для автоматизації обліку запасів.
4. **За функціональністю:**
   - Управління закупівлею та постачанням – автоматизація процесу замовлення товарів;
   - Управління складом – внутрішнє управління запасами на складі: зберігання, переміщення, облік та контроль товарів;
   - Прогнозування та планування попиту – оцінка майбутнього попиту на товари та коригування рівня запасів у відповідності до цього попиту.
5. **За рівнем автоматизації:**
   - Частково автоматизовані – поєднують людський контроль з автоматизованими процесами;
   - Повністю автоматизовані – процеси управління запасами здійснюються автоматично на основі закладених алгоритмів та даних.

Дана класифікація допомагає визначити оптимальний підхід до організації управлінням запасів на підприємстві.

1.4.2 Структурний аналіз.

Структурний аналіз системи управління запасами включає в себе вивчення основних компонентів, які забезпечують ефективне управління товарними запасами на підприємстві.

Компоненти системи управління запасами:

- Програмне забезпечення – відповідає за обробку даних, підтримку алгоритмів, інтеграцію з іншими системами);
- База даних – місце зберігання інформації про запаси, постачальників, замовлення;
- Інтерфейси користувача – введення і відображення даних, доступу до звітів;
- Алгоритми і методи управління запасами – алгоритми для прогнозування попиту, методи обчислення оптимальних рівнів запасів тощо;
- Інфраструктура – сервери, мережі та інші технічні елементи, які підтримують роботу системи.

Взаємодія між цими елементами забезпечує ефективність усієї системи.

1.4.3 Функціональний аналіз.

У центрі уваги функціонального аналізу системи управління запасами знаходяться основні функції, які забезпечують ефективне управління запасами.

Основні функції системи управління запасами:

- Облік запасів – система здійснює точний облік товарів на складі, відстежуючи їх кількість, місце розташування;
- Прогнозування попиту – використовуються алгоритми прогнозування для оцінки потреби в запасах;
- Управління замовленнями – автоматично або вручну ініціювання замовлення товарів на основі рівнів запасів, попиту та прогнозів.
- Оптимізація запасів – забезпечення мінімальних запасів для безперервної роботи без надлишку;

- Моніторинг та звітність – автоматичне формування звітів про стан запасів, рух товарів, витрати, помилки та інші важливі показники;
- Контроль умов зберігання – інтеграція з датчиками та IoT–пристроями для моніторингу температури, вологості;
- Автоматизація процесів прийому та відвантаження товарів – використання штрих–кодів та ТЗД підв'язаних до інформаційної системи;
- Інтеграція з іншими системами підприємства – система повинна бути сумісною з WMS, ERP та іншими інструментами, що використовуються на підприємстві для створення єдиної ефективної інформаційної системи.

Цей аналіз дозволяє точно визначити, які функції необхідно реалізувати в інформаційній системі, щоб забезпечити ефективне управління запасами.

1.4.4 Параметричний аналіз.

Параметричний аналіз полягає у визначенні та оцінці основних параметрів, які впливають на ефективність управління запасами.

Основні параметри, які слід аналізувати:

- Мінімальний рівень запасів – кількість товарів, яка забезпечує безперервну роботу підприємства до наступного постачання;
- Максимальний рівень запасів – обсяг продукції, що може бути збережений без перевантаження складу товаром та збільшення витрат;
- Періодичність постачання – частота поповнення запасів залежно від попиту та логістичних можливостей;
- Прогнозований попит – розрахунки майбутніх потреб у товарах на основі існуючих даних, сезонних коливаннях та ринкових тенденцій.

Аналіз цих параметрів дозволяє змінити систему управління запасами, щоб відповідати змінам попиту, знизити витрати та підвищити ефективність роботи складу.

Таким чином, аналіз системи управління запасами дозволив визначити основні підходи, структуру, функціональні можливості та параметри, які впливають на ефективність управління товарними запасами в розподільчих центрах.

1.5 Постановка задачі

Оптимізація логістичних процесів залежить від створення інформаційної системи управління запасами в розподільчих центрах, які відповідають за зберігання та перерозподіл овочів і фруктів. Контроль умов зберігання, високі сезонні коливання та необхідність швидкої ротації продукції вимагають використання сучасних інформаційних технологій.

У результаті аналізу існуючих інформаційних систем, які використовуються в логістиці та управлінні запасами, було виявлено, що жодна з них не відповідає потребам розподільчих центрів зорієнтованих на категорію товарів «овочі та фрукти». Незважаючи на те, що кожна система виконує певні функції, її суттєві недоліки включають:

- **висока вартість впровадження та складність налаштування**;
- **відсутність гнучкості для агросектору**, особливо щодо швидкопсувності, сезонності та температурних вимог;
- **необхідність інтеграції кількох рішень для повноцінної роботи**, що ускладнює підтримку і супровід;
- **недостатня візуалізація та аналітика в режимі реального часу**, яка критично важлива для оперативного управління запасами.

Це означає, що **потрібно створити власну інформаційну систему**, яка враховує характеристики продукції, логістичні процедури та особливості складу.

Основні задачі, які ставляться перед системою:

1. **Інтеграція WMS, ERP та BI функціоналу** в єдиній платформі для управління всіма процесами: від приймання до аналітики.

2. **Можливість адаптації під температурні зони, терміни зберігання та принцип FIFO/FEFO**, актуальні для фруктів та овочів.
3. **Реалізація мобільного інтерфейсу** для роботи з ТЗД (терміналами збору даних), сканерами та іншими пристроями.
4. **Автоматизація ключових операцій**: прийомка, інвентаризація, комплектування, формування замовлень, звітність.
5. **Підтримка аналітичних та прогнозуючих модулів** на основі ABC/XYZ аналізу та простих алгоритмів машинного навчання.
6. **Гнучка база даних**, що дозволить масштабувати систему, інтегрувати зовнішні модулі або налаштовувати логіку під конкретний бізнес.
7. **Низький поріг входу для персоналу**, з простим та інтуїтивно зрозумілим інтерфейсом.

Таким чином, розробка інформаційної системи, адаптованої до потреб агрологістики, дозволить не тільки вирішити проблеми, пов’язані з існуючими рішеннями, але й створити ефективний інструмент для постійного управління товарними запасами, що сприятиме розвитку компанії та підвищить її конкурентоспроможність.

2. МЕТОДИ ТА МОДЕЛІ ОПТИМІЗАЦІЇ УПРАВЛІННЯ ЗАПАСАМИ

2.1 Аналіз існуючих методів та моделей управління запасами

Розподільчі центри дуже залежні від управління запасами, особливо коли йдеться про продукти харчування з обмеженим терміном придатності, такі як овочі та фрукти. У сучасному світі ефективне управління запасами стає все більш важливим через високу конкуренцію, сезонність поставок і необхідність дотримання специфічних умов зберігання, таких як температура, вологість і вентиляція [5].

Правильний вибір методів важливий через коливання попиту та температури при зберіганні, короткий термін придатності товарів і необхідність автоматизації складських процесів.

Розглянуто наступні групи моделей[6]:

- **Кількісні моделі управління запасами (EOQ, ROP, Safety Stock)** – традиційні методи визначення обсягу замовлення, точки повторного поповнення та страхового запасу. Через те, що попит і попит швидко змінюються, вони не підходять для агропродукції.
- **Методи класифікації запасів (ABC/XYZ аналіз)** – використовується для розподілу товарів на категорії відповідно до вартості та стабільності попиту. Дають змогу зосередитися на логістичних центрах.
- **Прогностичні моделі (Time Series, ML, AI)** – дозволяють враховувати сезонні коливання попиту. Потребують складної реалізації та великих обсягів даних.
- **Логістичні та стратегічні моделі (JIT, VMI, DRP, FEFO)** – організаційні підходи до управління товарними потоками. Частково використовуються за допомогою методу відбору FEFO.
- **Імітаційні та евристичні моделі (жадібні, rule-based)** – основа реалізованих систем, які забезпечують оптимізацію без складних обчислень.

2.1.1 Кількісні моделі управління запасами

Кількісні моделі управління запасами – це аналітичні підходи, які дозволяють оптимізувати запаси шляхом визначення необхідного страхового запасу, моменту повторного поповнення та обсягу замовлення. Багато галузей логістики використовують їх, особливо в виробництві та роздрібній торгівлі.

EOQ (Economic Order Quantity)

EOQ – це модель, що визначає оптимальний обсяг замовлення, при якому загальні витрати на зберігання та замовлення є мінімальними. Формула EOQ виглядає наступним чином:

EOQ = $\sqrt{2 * D * S/H}$

де: D – попит за період, S – витрати на оформлення одного замовлення, H – витрати на зберігання одиниці продукції за період.

Модель EOQ гарантує постійний попит і миттєве виконання замовлень. Це робить її менш придатною для швидкопсувної продукції, особливо в агрологістиці, де попит є сезонним і залежить від багатьох зовнішніх факторів.[3]

ROP (Reorder Point)

ROP визначає точку, у якій слід робити нове замовлення, щоб уникнути дефіциту запасів. Формула розрахунку виглядає наступним чином:

ROP = $d * L$

де: d – середньодобовий попит, L – час постачання (lead time).

ROP швидкопсувних товарів, таких як овочі та фрукти, може бути нестабільним через зміни попиту, затримки доставки або погодні умови, які впливають на врожайність.[27]

Safety Stock (страховий запас)

Safety Stock – це додатковий запас, який зберігається, щоб компенсувати непередбачувані коливання попиту або затримки в поставках. Стандартне відхилення попиту та бажаний рівень сервісу є статистичними розрахунками, на яких може базуватися формула визначення.

Наприклад: $SS = z * \sigma L$

де: z – коефіцієнт сервісного рівня (наприклад, 1.65 для 95%), σL – стандартне відхилення попиту під час lead time.[28]

В умовах стабільного, прогнозованого попиту та значного періоду зберігання продукції всі наведені моделі працюють. Однак агропродукція має короткий термін придатності, значну залежність від сезонності та нестабільний попит. У таких умовах використання EOQ, ROP та Safety Stock має обмежене застосування.

2.1.2 Методи класифікації запасів (ABC/XYZ аналіз)

2.1.2.1 ABC–аналіз(Activity–Based Classification)

Використання ABC-аналізу демонструє, що тенденції товарів сільськогосподарської продукції розділяються залежно від сезону. Це важлива частина правильного замовлення та доставки товарів.

Аналіз ABC – це метод класифікації товарних запасів на основі того, наскільки вони важливі для компанії. Він базується на принципі Парето, також відомому як правило 80/20, згідно з яким двадцять відсотків товарів можуть принести вісімдесят відсотків доходу або витрат.

Принципи ABC–аналізу:

1. Товари групуються на три категорії (A, B, C) залежно від їхнього впливу на загальну вартість запасів.
2. Основна мета – визначити, які товари потребують найбільшого контролю та уваги при управлінні запасами.

Таблиця 2.1.2.2.1 ABC - аналіз

Категорія	Частка товарів	Частка вартості товарів	Характеристика
A	20%	80%	Найцінніші товари, що приносять основний прибуток. Потребують точного контролю, прогнозування попиту та регулярного перегляду запасів.
B	30%	15%	Проміжні товари. Важливі, але менш критичні. Контролюються рідше, ніж категорія A.
C	50%	5%	Найбільш численна група, але з найменшою часткою у вартості запасів. Контролюються спрощено, можливо використовувати автозамовлення.

2.1.2.2 XYZ–аналіз

Крім ABC-аналізу, XYZ-аналіз є важливим доповненням; це метод класифікації товарів за стабільністю попиту. Його використовують для оптимізації логістики, планування закупівель і управління запасами.

XYZ-аналіз зосереджується на передбачуваності та мінливості попиту, на відміну від ABC-аналізу, який розподіляє товари за вартістю або важливістю.

Залежно від стабільності їх споживання XYZ-аналіз поділяє товари на три категорії:

- X – стабільний попит, мінімальні коливання;
- Y – помірні коливання попиту;
- Z – нестабільний попит, важке прогнозування.

Таблиця 2.1.2.2.2 XYZ - аналіз

Категорія	Характеристика	Стратегія управління запасами
X	Незначні коливання, передбачуваний попит.	Автоматизація закупівель, підтримка мінімальних запасів.
Y	Попит змінюється в залежності від сезонності або ринкових факторів.	Гнучке управління запасами, закупівлі перед піковим сезоном.
Z	Попит важко прогнозувати, значні коливання.	Закупівля під замовлення, уникнення надлишкових запасів.

Поєднуючи ABC та XYZ аналізи, отримуємо найкращий метод аналізу, що забезпечує розширений контроль за запасами, дозволяючи оптимізувати управління складом.[9]

Таблиця 2.1.2.2.2 ABC-XYZ аналіз

ABC\ XYZ	X	Y	Z
A	AX (найважливіші, потребують чіткого контролю)	AY (важливі, але їхній попит змінюється)	AZ (дорогі, але попит нестабільний)
B	BX (стабільні, але менш значущі)	BY (попит коливається, потребують планування)	BZ (непередбачувані, середньої важливості)
C	CX (стабільні, можна зберігати у великій кількості)	CY (інколи потрібні, але без чіткої закономірності)	CZ (рідко використовуються, краще уникати зберігання)

Де, X – стабільні товари, Y – сезонні товари, Z – нестабільні, A – ключові товари, B – товари середньої важливості, C – малозначущі.

У контексті аграрної логістики, зокрема для продукції, яка швидко псується (овочі, фрукти), застосування ABC/XYZ дає змогу:

- Визначити **товари з найвищим пріоритетом контролю (A)**
- Адаптувати логістику до **сезонних хвиль попиту (Y)**
- Виявляти **ризикові позиції (CZ)**, які можуть спричинити втрати

2.1.3 Прогностичні моделі попиту

Прогнозування попиту є однією з ключових функцій у системах управління запасами. Його мета – оцінити майбутній попит на товари, щоб:

- мінімізувати надлишкові запаси,
- уникнути дефіциту продукції,
- забезпечити стабільність постачання та ефективну логістику.

Для цього застосовують різні підходи: **статистичні моделі**, **методи часових рядів**, **моделі машинного навчання** та **гібридні рішення**.[25]

2.1.3.1 Методи часових рядів (Time Series Forecasting)

Ці методи передбачають аналіз послідовності попередніх значень попиту у часовому вимірі. Найбільш поширені моделі:

- **Ковзна середня (Moving Average)**

Згладжує випадкові коливання, усереднюючи попередні nnn значень:

$\acute{y}_t = \frac{1}{n}\sum_{i=1}^{n} y_{t-i}$

де, $\acute{y}_t$ – прогнозоване значення попиту в момент часу, n – кількість попередніх періодів для усереднення, y_{t-i} – фактичне значення попиту у попередньому періоді t-i.

Переваги: простота реалізації

Недоліки: нечутливість до трендів і сезонності

- **Експоненційне згладжування (Exponential Smoothing)**

Прогноз базується на попередньому значенні та попередньому прогнозі:

$\acute{y}_t = \alpha y_{t-1} + (1-\alpha)\acute{y}_{t-1}$

де $\acute{y}_t$ – прогноз на поточний момент, y_{t-1} – фактичне значення попиту за попередній період, $\acute{y}_{t-1}$ – прогноз попереднього періоду, α – коефіцієнт згладжування (від 0 до 1).

- **ARIMA (AutoRegressive Integrated Moving Average)**

Одна з найпотужніших моделей для стаціонарних часових рядів. Складові:

  - **AR (авторегресія)**: зв'язок між поточним значенням і попередніми
  - **I (інтегрування)**: усунення тренду шляхом диференціювання
  - **MA (середнє)**: врахування шуму

ARIMA добре працює при наявності достатнього обсягу історичних даних.

2.1.3.2 Класичні статистичні моделі

Окрім часових рядів, застосовують також класичні підходи:

- **Регресійний аналіз**

Залежність попиту від зовнішніх змінних:

$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_n x_n + \varepsilon$

де y – прогнозований попит, $x_1, x_2, \dots x_n$ – незалежні змінні (наприклад: день тижня, температура, ціна, категорія товару), β_0 – вільний член (зсув), $\beta_1, \dots \beta_n$ – коефіцієнти регресії, ε – похибка моделі.

- **Метод найменших квадратів (OLS)**

Використовується для оцінки параметрів регресійних моделей.

2.1.3.3 Моделі машинного навчання (Machine Learning)

Ці моделі краще враховують складні, нелінійні зв’язки, сезонні патерни, тренди та вплив різноманітних факторів.

- **Decision Tree / Random Forest / XGBoost**
  - Побудовані на ідеї розбиття даних за умовами
  - Добре працюють з великою кількістю незалежних змінних
  - Можуть враховувати взаємодії: SKU, регіон, тип тари, день тижня
- **Gradient Boosting (CatBoost, LightGBM)**

Покращують якість прогнозу шляхом послідовного усунення помилок попередніх дерев. Ефективні для структурованих табличних даних.

2.1.3.4 Глибоке навчання (Deep Learning)

Застосовується для обробки довгих послідовностей даних, особливо коли є складна динаміка.

- **RNN (Recurrent Neural Networks)**
  - Добре підходять для прогнозування часових рядів
  - Недолік – затухання градієнту
- **LSTM (Long Short-Term Memory)**
  - Розширення RNN з механізмом пам’яті
  - Використовується в складних серіях (наприклад, попит із нерегулярними піками)
- **GRU (Gated Recurrent Units)**
  - Альтернатива LSTM з меншою складністю

2.1.3.5 Гібридні та спеціалізовані моделі

- **Prophet (Meta)**

Модель з відкритим кодом, створена для бізнес-прогнозування:

- Виявляє тренд, сезонність, свята
- Автоматично підлаштовується до збоїв
- Простий інтерфейс для нетехнічних користувачів

- **AutoML-рішення**

Сервіси, які автоматично обирають найкращу модель:

- Google AutoML
- Amazon Forecast
- Azure AutoML

Прогнозування попиту є завданням, яке стосується багатьох дисциплін і включає статистику, аналіз часових рядів, алгоритмічне моделювання та машинне навчання.

Найбільш доцільними для логістики агропродукції є:

- **Короткострокові моделі** з адаптацією до сезонності
- **ML-рішення** для складної класифікації SKU
- **Гібридні моделі**, які враховують обсяг, вагу, упаковку, клієнта

На всіх етапах логістичного ланцюга розумне впровадження прогнозування зменшує втрати, запобігає перевантаженню складу та підвищує точність прийняття рішень.[26]

2.1.4 Логістичні та стратегічні моделі

Логістичні та стратегічні моделі є важливими для управління запасами, оскільки вони забезпечують управління ризиками, скорочення витрат і підвищення ефективності використання ресурсів. Щоденні операції на складі та планування на рівні ланцюга постачання можна виконувати за допомогою цих методів.

1. JIT (Just-in-Time)

Суть методу:

Модель, при якій товари замовляються і поставляються точно в той момент, коли вони потрібні для виробництва чи реалізації, без зберігання на складі.

Переваги:

- Мінімальні залишки
- Економія складських витрат
- **Недоліки:**
- Надзвичайно чутлива до збоїв постачання
- Вимагає високої точності прогнозів і безперебійної логістики

Застосовність до агропродукції:

Підходить лише частково, оскільки існує надзвичайно високий ризик того, що товар буде зіпсований через затримку транспорту. Може застосовуватися для постійних, передбачуваних поставок, які оновлюються щодня.[7]

2. VMI (Vendor-Managed Inventory)

Суть методу:

Постачальник самостійно відслідковує запаси на складі клієнта і ініціює поставку, коли обсяг зменшується.

Переваги:

- Зменшує навантаження на клієнта
- Постачальник краще планує виробництво

Недоліки:

- Потребує повної прозорості даних
- Високі вимоги до систем зв’язку

Застосовність:

Незважаючи на те, що це може працювати у великих аграрних мережах або переробних підприємствах, це важко зробити в децентралізованих логістичних центрах.[8]

3. DRP (Distribution Requirements Planning)

Суть методу:

Метод планування потреб у запасах для мережі складів. Враховує обсяги попиту, графіки постачання, маршрути.

Особливості:

- Забезпечує синхронізацію між рівнями ланцюга постачання
- Усуває дублювання закупівель

Недоліки:

- Складна реалізація без централізованої ІС
- Вимагає великої кількості даних і точної координації

4. FEFO / FIFO / LIFO – Методи відбору товарів

FIFO (First In, First Out)

- Перший прийшов – перший пішов
- Класичний метод для продукції з обмеженим терміном зберігання

LIFO (Last In, First Out)

- Використовується рідко у складуванні
- Переважно бухгалтерська практика

FEFO (First Expired, First Out)

- Відбір за терміном придатності, незалежно від дати приймання
- Критично важливий для швидкопсувної продукції

Застосовність:

Для агрологістики FEFO є найкращим. забезпечує контроль терміну придатності на рівні партій через базу даних і штрих-коди SSCC.

Таблиця 2.1.4.1 Логістичні та стратегічні моделі

Модель	Суть	Переваги	Недоліки
JIT	Поставка в момент потреби	Мінімальні запаси	Ризики затримки
VMI	Постачальник керує запасами	Зручність для клієнта	Високі вимоги до IT
DRP	Планування потреб мережі	Централізація	Складність реалізації
FIFO	Відбір за датою приймання	Простість	Ігнорує термін
FEFO	Відбір за терміном	Найточніше	Потребує відстеження партій

Стратегічні та логістичні моделі допомагають оптимізувати політику управління запасами розподільчого центру. Модель FEFO є найкращою для агропродукції, оскільки вона забезпечує мінімальні втрати завдяки правильному підбору. За умови наявності відповідної інфраструктури та систем прогнозування, інші моделі, такі як JIT або DRP, можуть бути впроваджені в довгостроковій перспективі.

2.1.5 Імітаційні та евристичні моделі

У системах управління запасами, які функціонують в умовах **високої мінливості, обмеженого часу на прийняття рішень та складних просторових обмежень**, традиційні математичні моделі часто поступаються **імітаційним і евристичним підходам**. Ці методи не завжди забезпечують оптимальне рішення, однак дозволяють знайти **задовільні результати в реальному часі** з прийнятним рівнем точності.

1. Евристичні моделі (Heuristic Methods)

Евристика – це наближений метод вирішення задачі, що базується на правилах, інтуїції, попередньому досвіді чи структурних закономірностях системи.

Класи евристик:

- **Жадібні алгоритми (greedy)** – будують рішення крок за кроком, обираючи локально найкраще.
- **Алгоритми з обмеженням правил (rule-based)** – реалізують чіткі умови прийняття рішень на основі стану системи.
- **Евристики класифікації та сортування** – застосовуються для пріоритезації SKU, маршрутизації, відбору, складання палет.
- **Евристики комбінаторної оптимізації** – спрощують складні задачі, наприклад, пакування товарів або поділ їх на групи.

Типові сфери застосування:

- Формування замовлень і партій
- Розміщення товарів по зонах складу
- Оптимізація палетизації та заповнення тари
- Визначення черговості відбору

Приклад жадібної евристики (палетизація):

1. Відсортувати товари за об’ємом (від більшого до меншого)
2. По черзі додавати їх до палети, поки не буде вичерпано допустимий обсяг
3. Повторити для залишку

2. Імітаційне моделювання (Simulation Modeling)

Імітація – це побудова комп’ютерної моделі реальної системи, яка дозволяє провести експерименти без втручання у фізичні процеси.

Типи імітацій:

- **Дискретно-подійна симуляція (DES)** – моделює події в часі (наприклад, прибуття замовлення, завершення комплектації)

- **Агентне моделювання (Agent-Based)** – кожен складський об’єкт (товар, комірка, працівник) має свою поведінку
- **Системна динаміка** – аналізує потоки запасів у стратегічному масштабі

Ключові переваги імітації:

- Можливість тестування «що-якщо» сценаріїв
- Вивчення поведінки системи в екстремальних ситуаціях
- Пошук вузьких місць (bottlenecks)

Основні етапи побудови моделі:

1. **Формалізація процесів** – визначення вхідних змінних, подій, обмежень
2. **Розробка симулятора** – створення моделі з часовими залежностями
3. **Верифікація** – перевірка, чи модель відповідає логіці реальної системи
4. **Експерименти та аналіз результатів** – зміна параметрів, збір статистики

3. Гібридні евристично-імітаційні підходи

У багатьох практичних задачах логістики використовується **гібридна схема**, де:

- **евристика** використовується для оперативного прийняття рішень (наприклад, формування черги на відбір),
- а **імітація** – для перевірки ефективності обраної політики або тестування системних змін (наприклад, перестановки зон зберігання).

Такі підходи дозволяють **перевірити евристичне рішення на віртуальному складі** і, при потребі, внести зміни до алгоритмів без впливу на реальний процес.

Імітаційні та евристичні моделі є чудовими інструментами для управління складськими процесами в умовах обмежених ресурсів, великої кількості змінних і відсутності аналітичної моделі. Вони дозволяють швидко реагувати на зміни, адаптувати логіку до особливостей продукції та складу, а також прогнозувати ефективність логістичних рішень ще до того, як вони стануть реальністю.

2.2 Обґрунтування вибору методів

Визначення найбільш ефективних методів для вирішення практичних завдань, які стоять перед розроблюваною інформаційною системою, вимагає ретельної оцінки існуючих підходів до управління запасами та класифікації моделей логістики.

Проводимо аргументований вибір певних моделей та методів, що були обрані для реалізації окремих функціональних компонентів системи. Основними факторами, які вплинули на вибір, були особливості складу агропродукції, вимоги до оперативності обробки запитів, обмеження на обчислювальні ресурси та необхідність адаптації до сезонного та нестабільного попиту.

Розгляд критеріїв вибору, порівняння альтернативних підходів і логіка обґрунтування реалізованих методів наведені у наступних підпунктах.

2.2.1 Аналіз задач, що вирішуються системою

Приймання товару, автоматичне розміщення товару у відповідних температурних діапазонах, резервування товару для виконання замовлень клієнтів, розрахунок і формування логістичних одиниць (лотів або палет), а також гарантування точного та своєчасного відбору та відвантаження відповідно до принципів ротації FIFO/FEFO є основними завданнями, які система повинна вирішити.

Крім того, система повинна забезпечувати повне управління запасами на складі. Це включає облік товарів за типами упаковки, місцями зберігання та партіями, а також дозволяє формувати звіти щодо обсягів товарів у наявності, розподілу їх за категоріями, термінами придатності та джерелами надходження.

Автоматизація комплектації також включає процес швидкої перевірки наявності товару в зоні відбору та запуску внутрішнього переміщення зі складу глибокого зберігання, якщо його немає. Крім того, обов’язком є створення

документів обліку та обміну даними, таких як рахунки-фактури, видаткові накладні та етикетки зі штрих-кодом SSCC.

Система повинна підтримувати як штучні, так і вагові види продукції, реалізовувати алгоритми для перерахунку ваги в ящики, враховувати кратність упаковки та дотримуватись вимог логістичних даних щодо габаритів, температури зберігання й умов перевезення. У майбутньому планується впровадити модуль прогнозування попиту, щоб покращити планування поставок і зменшити втрати неякісних товарів.

Таким чином, система повинна виконувати завдання як на операційному рівні (облік, звітність, планування) так і на аналітичному рівні (взаємодія з терміналом збору даних, контроль розміщення товарів). Завдяки правильному вибору відповідних моделей і алгоритмів для вирішення цих проблем можна досягти успіху. Ці моделі та алгоритми повинні враховувати особливості продукції, організаційну структуру складу та обмеження ресурсів.

2.2.2 Критерії вибору методів і моделей

Вибір моделей, алгоритмів і методів для впровадження інформаційної системи управління запасами залежав від цілей, характеристик продукції, обмежень інфраструктури складу та загальних вимог до роботи системи в реальному виробничому середовищі.

Висока мінливість середовища, включаючи постійні зміни асортименту, варіативність штучних і вагових товарів, сезонні коливання попиту та необхідність швидкого прийняття рішень з мінімальним обсягом даних, були основними факторами. Застосування цих функцій вимагає використання моделей, здатних адаптуватися до змін, працювати з неповними даними та надавати результати в режимі реального часу.

Основним критерієм була **технічна доступність реалізації**. Необхідно було уникати надмірно складних моделей (наприклад, повномасштабного

машинного навчання чи глибоких нейромереж), які потребують значної кількості даних для навчання, високої обчислювальної потужності або складного супроводу, оскільки проект реалізовувався в середовищі з обмеженими обчислювальними ресурсами.

Також була необхідна **швидкодія** та **масштабованість алгоритмів**, оскільки система повинна обробляти велику кількість комірок, партій, SKU і документів у середовищі, де працюють багато людей. Таким чином, пріоритетом були методи з низькою алгоритмічною складністю, які дозволяють досягати результатів за постійну або лінійну кількість кроків.

Окрему роль відіграла **потреба у пояснюваності логіки**. Важливо, щоб персонал знав поведінку системи, щоб вона могла працювати в логістичних цілях, і щоб результати були прогнозованими. Таким чином, пріоритет надавався підходам, заснованим на правилах і евристичним, які легко формалізуються у вигляді простих правил і умов.

Крім того, були враховані **параметри стійкості** та **адаптивності**. Методи, які були обрані, повинні бути легко переналаштовуватись і не вимагати повної реконфігурації системи в умовах частих змін конфігурації складу, кількості робочих зон, нових типів тари або SKU.

Зрештою, визначальним критерієм стало **відповідність моделі задачі**, тобто її здатність вирішити конкретну прикладну проблему. При прийманні та палетизації це швидке групування товарів відповідно до розмірів. У випадку відбору це оперативна перевірка залишків і створення запиту на переміщення. Використання таких методів прогнозування не вимагає великого історичного масиву для початку.

Усі вищезазначені критерії були основою для розгляду альтернатив і остаточного вибору методів, що використовуються в системі.

2.2.3 Порівняльний аналіз класичних та альтернативних підходів

У процесі проектування інформаційної системи управління запасами було розглянуто низку методів і моделей, традиційних у логістиці, виробництві та аналітиці. Їх аналіз стосувався відповідності проєкту, зокрема обробки аграрної продукції, обмеженого терміну зберігання, динамічного асортименту та швидкої реакції системи.

Один із найпоширеніших класичних підходів – це **кількісні моделі поповнення запасів**, такі як EOQ (економічна кількість замовлення), ROP (точка повторного замовлення) та розрахунок страхового запасу. Ці методи працюють добре в умовах з коротким терміном придатності, сезонною доступністю та великою кількістю SKU, характерною для складу сільськогосподарських товарів, але вони працюють добре в умовах з передбачуваним попитом. З цієї причини дані методи були оцінені як непрактичні в межах розроблюваної системи.

Крім того, розглядалося, чи можна використовувати **моделі прогнозування попиту**, засновані на статистиці (наприклад, ARIMA) або машинному навчанні (наприклад, градієнтний бустинг, рекурентні нейромережі тощо). Ці методи потребують складної інфраструктури підтримки, великих історичних даних і системи динамічного оновлення моделей, незважаючи на потенціал у плануванні логістики. На початку впровадження системи подібні умови були недоступні.[24]

Стратегічний аналіз включав оцінку таких підходів, як **JIT (поставка точно вчасно)**, **VMI (керування запасами постачальником)** та **DRP (планування потреб у розподілі)**. Ці моделі добре працюють у великих інтегрованих ланцюгах постачання, але вони вимагають централізованого управління, синхронізації між різними системами та значної автоматизації, яка зараз перевищує розмір цільової системи. У той же час метод FIFO визнаний найкращим для швидкопсувних товарів і залишається одним із основних принципів відбору в системі.

Евристичні алгоритми та **rule-based підходи** були основними кандидатами для реалізації, оскільки вони продемонстрували найкращий баланс між адаптивністю, простотою реалізації, швидкодією та прозорістю логіки. Вони

дозволяють виконувати операційні завдання, такі як формування лотів, перевірка доступності та розподіл по палетах, з мінімальними вимогами до інфраструктури та даних.

Таким чином, на основі порівняльного аналізу було прийнято рішення відмовитися від надмірно складних або слабко масштабованих методів і зосередитися на методах, які дозволяють досягти практичного результату в реальних умовах складу агропродукції, які мають високі динамічні коливання та короткий час прийняття рішень.

2.2.4 Обґрунтування вибору реалізації підходів

Для розробки інформаційної системи управління запасами планується використовувати кілька прикладних алгоритмів і моделей, які базуються на **евристичних**, **rule-based** і **імітаційних підходах**, відповідно до попереднього аналізу. Методи, які були обрані, забезпечують прозору логіку обробки складських процесів, дозволяють гнучку адаптацію до умов агрологістичного складу та швидку реакцію на зміни стану запасів.

У план реалізації закладено впровадження **жадібного алгоритму палетизації** (див. Додаток В, лістинг 1), який дозволятиме розподіляти товари з різними габаритами й упаковками по палетах із урахуванням залишкового об'єму. У модулі обробки замовлень буде реалізовано **алгоритм адаптації вагових товарів**, який автоматично розраховуватиме еквівалентну кількість ящиків на основі нетто-ваги, типу упаковки та кратності (див. Додаток В, лістинг 2, 4).

Також планується реалізація **rule-based логіки комплектації**, дозволить системі вирішувати, де відбирати товари, створювати запити на внутрішнє переміщення, якщо товари відсутні в зоні відбору, і перевіряти, чи відповідають SKU, партії та терміни придатності (див. Додаток В, лістинг 3, 6). Для агропродукції з обмеженим терміном зберігання принцип FIFO буде основою для логіки вибору.

Евристична модель сумісності товару та зони зберігання буде використана для визначення місця розміщення товару в системі. Ця модель враховуватиме температуру, рівень зберігання (наприклад, глибину зберігання чи відбір) і заповненість складських рядів. Це дозволить автоматизувати процес приймання та «поселення» товару(див. Додаток В, лістинг 5, 8).

Також планується впровадження **алгоритму генерації SSCC штрих-кодів** згідно зі стандартом GS1 (див. Додаток В, лістинг 7), що дозволить ідентифікувати кожну логістичну одиницю та гарантує точність при відвантаженні, відборі та перевірці.

У наступних етапах розробки планується впровадити **систему аналітики** для визначення найбільш затребуваних SKU, створення звітів по залишках, транспортування та приймання товарів. Підґрунтям для цього стане простий механізм агрегування даних, який можна буде потім інтегрувати з модулем прогнозування попиту.

Обрани методи гарантують оптимальний баланс точності, швидкості, гнучкості та простоти реалізації. У процесі проходження етапів розробки та тестування всі описані моделі та алгоритми будуть поступово впроваджені у функціональні модулі системи.

2.3 Планування інформаційної моделі управління запасами

2.3.1 Проєктування структури бази даних

Інформаційна система управління запасами не може працювати ефективно без ретельної структури бази даних. База даних є ключовою частиною забезпечення збереження, доступності та цілісності інформації про рух продукції, постачальників, замовлення, запаси та інші логістичні операції.

Пропонується створити реляційну базу даних, яка базується на чітко структурованих таблицях, пов’язаних між собою за допомогою ключів, для системи, яка обслуговує розподільчий центр сільськогосподарської продукції.

Основні таблиці такої бази:

- **Продукція** – зберігає номенклатуру товарів, включаючи назву, код, категорію (наприклад, овочі чи фрукти), одиницю виміру, термін придатності, температуру зберігання та критичний рівень залишку.
- **Постачальники** – ідентифікаційні дані, контактна інформація, країна походження, історія поставок.
- **Замовлення** – дата, статус (створено, підтверджено, доставлено), пов’язаний постачальник, перелік товарних позицій.
- **Складські залишки** – поточна кількість продукції на складі, зона зберігання, дата останнього оновлення.
- **Рух товару** – вхід/вихід, дата, відповідальний працівник, пов’язане замовлення.
- **Температурний моніторинг** – історія температури в різних зонах зберігання, відхилення від норми.
- **Прогнози** – результат роботи ШІ–модуля, дата прогнозу, товар, обсяг прогнозованого попиту, період дії прогнозу.

Усі таблиці повинні бути пов’язані зовнішніми ключами. Наприклад, у розділі «Рух товару» «Продукція» та «Складські залишки» пов’язані з «Постачальниками» за допомогою ID постачальника.

Щоб забезпечити безпеку даних, також важливо впровадити інструменти для логування, резервного копіювання та контролю прав доступу та засоби для контролю прав доступу, щоб гарантувати безпеку даних.

2.3.2 Моделювання ролей користувачів та сценаріїв взаємодії

Інформаційна система управління запасами повинна враховувати функціональні обов’язки користувачів і різні рівні доступу. Моделювання ролей дозволяє визначити, які функції доступні для кожного типу користувача, які дані

вони можуть бачити або змінювати, і за які процеси відповідають. Це захистить систему, запобігає несанкціонованим змінам і покращує управління процесами.

Основні ролі користувачів у системі:

- **Приймальник –** відповідає приймання товару на склад. Зчитує SSCC-коди, вводить вагу, кількість і підтверджує партію за допомогою терміналу збору даних. Він може використовувати лише функції приймання та базових звітів.
- **Комплектувальник –** комплектує замовлення відповідно до завдань, створених системою. Він може використовувати функції відбору товарів, підтвердження кількості та обробки лотів. Він працює за допомогою ТЗД у режимі відбору. Він не може мати додаткові модулі.
- **Оператор / Начальник зміни / Керівник –** повний доступ до більшості операцій системи. Може створювати, змінювати та затверджувати замовлення, контролювати запаси, запускати приймання та відвантаження, переглядати звіти та створювати документи. Контролює всі складські операції в режимі реального часу.
- **Менеджер із закупівель –** формує замовлення на постачання, відстежує потреби та аналізує залишки. Модуль управління закупівлями, контракти з постачальниками та звіти доступні для нього. Прогнозуюча підсистема та аналітик співпрацюють з ним.
- **Аналітик –** порівнює фактичні дані з прогнозами та оцінює ефективність логістичних рішень. Він має доступ до чат-бота, історичних даних, дашбордів і модуля аналітики. може розпочати перевірку точності прогнозу на основі оновлених даних.
- **ІТ-адміністратор –** відповідає за технічне керівництво, включаючи створення користувачів, резервне копіювання, налаштування інтеграції та оновлення компонентів системи. Він може переглядати конфігурації, внутрішні сервіси та журнали подій.

Логіка, необхідна для щоденного використання системи, включена в сценарії взаємодії.

Наприклад:

- Приймальник отримує товар, зчитує SSCC-код, вводить вагу та кількість через ТЗД, підтверджує приймання.
- Система оновлює залишки та передає дані у WMS.
- Менеджер переглядає залишки, ініціює закупівлю при виявленні дефіциту.
- Комплектувальник виконує відбір товару згідно з лотом, підтверджує кількість та завершення лота.
- Аналітик аналізує ефективність прогнозу, використовуючи звітність за результатами комплектації.
- Керівник має змогу переглядати весь ланцюг операцій, погоджувати або коригувати їх при потребі.

Рис. 75 (див. Додаток Б), де показана UML-діаграма прецедентів, показує основні сценарії взаємодії користувачів з інформаційною системою. З неї видно, що кожна роль має доступ лише до відповідних функцій.

Таким чином, модель ролей і сценаріїв гарантує чітку розподіл відповідальності, робить процеси більш прозорими та робить систему більш зручною для користувачів.

2.3.3 Інтеграція з іншими модулями

Інформаційна система управління запасами повинна бути пов’язана з іншими модулями підприємства, щоб усі логістичні процеси працювали ефективно та узгоджено. Для розподільчих центрів, які працюють із швидкопсувною сільськогосподарською продукцією, обмін даними між модулями має відбуватись у режимі, наближеному до реального часу, з високим рівнем точності та надійності.

Взаємодія з Warehouse Management System є основним напрямком інтеграції. WMS використовує прогнозуючу систему для початку внутрішніх операцій, переміщуючи продукцію до зон швидкого доступу, створюючи

буферні запаси та пріоритезуючи відвантаження [13]. Наприклад, якщо система передбачає, що попит на апельсини зросте, WMS може автоматично створити маршрут, який дозволить підготувати продукцію до виробництва. Обидві системи повинні бути синхронізовані за одиницями виміру, складськими зонами та ідентифікаторами товарів.

Прогнози можна побачити в фінансах, плануванні ресурсів і закупівлях за допомогою інтеграції з ERP. ERP складають плани постачання, оцінюють бюджет і визначають вимоги до замовлень [14]. Система може автоматично скоригувати обсяг замовлення відповідно до встановлених бізнес-правил у випадку браку фінансування на закупівлю. Фінансові обмеження, закупівлі та терміни постачання передаються в систему ERP.

ТЗД забезпечують миттєву передачу даних про операції на складі. Коли оператор сканує штрих-код, система отримує інформацію, оновлює залишки, передає її в систему управління запасами (WMS) і одночасно запускає актуалізацію прогнозу. Підвищення точності майбутніх розрахунків можна очікувати завдяки цьому зв’язку, яке сприяє створенню адаптивної моделі, яка навчається на поточних даних [12, 15].

Крім того, необхідно впровадити інтеграцію з такими системами:

- **CRM** – щоб ураховувати дані про замовлення клієнтів, тенденції та заплановані акції;
- **BI–системами** – для передачі зведених даних у вигляді аналітичних звітів;

Інформаційна взаємодія може бути реалізована за допомогою API (Application Programming Interface), RESTful сервісів, вебхуків або ETL-процесів. JSON, XML і CSV є найбільш поширеними форматами для обміну. Захищені протоколи, такі як HTTPS і SSL, використовуються для передавання даних, а автентифікація користувачів здійснюється за допомогою токенів або OAuth2. Для підвищення стабільності система повинна мати механізми черг обміну даними, також відомі як черги повідомлень, щоб запобігти втрат під час пікового навантаження [16].

Регулярність синхронізації залежить від типу модуля. ТЗД синхронізується в режимі реального часу, ERP синхронізується щогодинно або за подієвим принципом, а аналітичні системи синхронізуються щодня або на вимогу. Налаштовано автоматичне інформування відповідальних осіб щодо логіки реакції на критичні події, такі як перевищення температури або відсутність постачання.

2.3.4 Візуалізація та звітність

Ефективна візуалізація даних і можливість генерації звітності є одними з найважливіших компонентів сучасної інформаційної системи. Це дозволяє розподільчим центрам, які працюють з продукцією, яка швидко псується, оперативно відслідковувати стан запасів, аналізувати динаміку обігу товарів і приймати розумні рішення.

Візуалізація має на меті представити складну інформацію у зручному для людини вигляді, такому як графіки, діаграми, карти руху товарів та індикатори запасів. Такі інструменти дозволяють логістам, аналітикам і менеджерам складу швидко вирішувати проблеми, знаходити відхилення, контролювати залишки в зоні ризику та планувати дії з мінімальним використанням ручної праці.

Наступні типи візуалізації мають бути використані в системі:

- **Гістограми і лінійні графіки** – для аналізу динаміки попиту, виявлення пікових навантажень, порівняння прогнозованих і фактичних значень;
- **Кругові діаграми** – для структури запасів за категоріями (овочі, фрукти), постачальниками, регіонами;
- **Теплові карти** – для візуального відображення зон складу з найвищою або найнижчою ротацією продукції;
- **Інтерактивні панелі керування (дашборди)** – для огляду ключових показників (KPI) у реальному часі, з можливістю налаштування фільтрів,

оновленням даних, та побудови багаторівневих звітів за сценаріями використання.

Користувачі мають можливість створювати індивідуальні звіти за різними параметрами, включаючи час, групи товарів, категорії постачальників, критичні запаси або прострочені партії. Звітування повинні бути експортовані в популярні формати, такі як PDF, Excel і CSV, а також інтегровані з електронною поштою для автоматичного розсилання звітів за графіком, наприклад щодня, щотижня або щомісяця.

Крім того, дуже важливо створити зведені звіти для керівництва, які містять основні показники ефективності, такі як швидкість обігу запасів, точність прогнозів, частота помилок при комплектуванні та відсоток вчасно виконаних замовлень. Заздалегідь визначені шаблони повинні бути використані для створення таких звітів, які можна переглядати у вигляді графіків, таблиць або інтерактивних схем.

Крім візуалізації, система повинна мати аналітичні інструменти, які дозволяють виявити сезонні тенденції, прогнозувати надлишки та дефіцити, оцінювати ефективність постачальників і знайти приховані залежності за допомогою кластеризації або аналізу аномалій.

Для цього доцільним є використання **BI-інструментів**, спеціалізованого програмного забезпечення, яке забезпечує глибоку візуалізацію та інтерактивну аналітику. Найпоширенішими BI-платформами є **Tableau, Google Data Studio і Microsoft Power BI**, які дозволяють легко інтегруватися з базами даних, створювати зв’язки з іншими сервісами та створювати персоналізовані дашборди для кожного користувача.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ЗАПАСАМИ

3.1 Обґрунтування вибору програмних засобів та технології розробки

3.1.1 Вибір мови програмування та фреймворкі

Для створення інформаційної системи управління запасами розподільчого центру сільськогосподарської продукції було обрано мову програмування Python, яка є однією з найвідоміших сучасних мов програмування. Python забезпечує швидку розробку прототипів, ефективне налагодження коду та простий синтаксис. Крім того, велика бібліотека та активна спільнота дозволяють вирішувати проблеми різного рівня складності, зокрема аналітику, розробку веб-додатків і управління даними.

У якості веб–фреймворку було обрано Flask, оскільки він є ідеальним веб-фреймворком, оскільки він поєднує легку вагу, гнучкість і зручність налаштування. Flask не обмежує структуру застосунків, що дозволяє створювати адаптовані рішення для конкретних вимог розподільчого центру. Завдяки модульності Flask легко інтегрувати різні технології та сервіси; це важливе для побудови комплексних систем із взаємодією різних компонентів.

Під час вибору Flask альтернативні фреймворки, такі як Django та FastAPI, не розглядалися, оскільки основними критеріями для цього проєкту були гнучкість, мінімалізм і швидкість розробки. Flask забезпечує швидку інтеграцію в існуючі інфраструктури, менший поріг входження та просте масштабування.

Для створення зручного, адаптивного та інтуїтивного інтерфейсу було вирішено використовувати Bootstrap та JavaScript як фронтенд-технології. Високоякісний адаптивний дизайн, який забезпечує зручність роботи користувачів на різних пристроях, легко досягається за допомогою шаблону. JavaScript дозволяє користувачам інтерактивно та динамічно взаємодіяти з системою.

Docker-контейнеризація забезпечує високий рівень масштабованості та просте розгортання системи. Це рішення дозволяє швидко запускати програмний продукт на різних платформах і ефективно керувати ресурсами, що особливо важливо за умови очікуваного високого рівня навантаження в умовах роботи розподільчого центру.

3.1.2 Вибір системи управління базами даних

Для створення інформаційної системи управління запасами було обрано систему управління базами даних PostgreSQL, оскільки вона є надзвичайно надійним, стабільним і ефективним у роботі з великими кількостями даних. PostgreSQL має потужні механізми для забезпечення цілісності, послідовності та безпеки даних, а також підтримує розширені SQL-запити. Крім того, він має високу продуктивність і має відкритий вихідний код, що дозволяє зменшити витрати на розробку та підтримку програмного продукту завдяки активній підтримці спільноти.

Підтримка транзакцій, процедур і тригерів є важливими перевагами PostgreSQL для даного проєкту, оскільки це дозволяє ефективно автоматизувати бізнес-логіку та спростити підтримку даних у правильному стані. Здатність підтримувати високі навантаження та велику кількість одночасних підключень роблять PostgreSQL ідеальною для інформаційних систем розподільчого центру.

Під час вибору СУБД також були розглянуті альтернативні рішення, такі як SQLite та MongoDB. SQLite було відхилено через його обмеження, включаючи неможливість працювати з великою кількістю одночасних користувачів і нездатність ефективно масштабуватися під високими навантаженнями. Здатність MongoDB орієнтуватися на документоорієнтовану модель зберігання даних призвела до того, що він не підходить для

структурованих реляційних даних, які використовуються в інформаційних системах управління запасами.

З огляду на її функціональні можливості, надійність і те, наскільки вона відповідає вимогам системи управління запасами, PostgreSQL стала найкращим варіантом.

3.1.3 Вибір інструментів для розробки інтерфейсу

Популярний фреймворк Bootstrap був обраний для створення зручного та адаптивного користувацького інтерфейсу. Простота використання адаптивного та мобільного дизайну є важливою для користувачів, які працюють з різними типами пристроїв. Крім того, Bootstrap містить велику кількість готових компонентів і шаблонів, що значно скорочує час розробки, дозволяючи швидко створювати стильні та функціонально зручні інтерфейси.

Було вирішено використовувати JavaScript, щоб забезпечити інтерактивність і динамічну взаємодію з користувачем. JavaScript дозволяє створювати динамічні елементи інтерфейсу та підтримувати асинхронну взаємодію з сервером за допомогою запитів AJAX. Це дозволяє суттєво покращити досвід користування, роблячи інтерфейс швидшим і простішим для розуміння.

Бібліотека jQuery була використана для спрощення написання коду та покращення читабельності та підтримуваності коду. Це робить більш простим працювати з DOM-елементами та виконувати стандартні задачі JavaScript.

3.1.4 Обґрунтування вибору моделі життєвого циклу ПЗ

Інформаційна система управління запасами для розподільчого центру була створена за допомогою моделі життєвого циклу Agile і методології Scrum. Цей вибір виникає через необхідність гнучкого підходу до розробки, який може швидко реагувати на зміни в потребах і особливостях предметної області.

Методологія Agile/Scrum використовує невеликі ітерації для розробки, що дозволяє швидко отримувати проміжні результати, переглядати пріоритети завдань і змінювати їх відповідно до потреб замовника. Це особливо важливо для проєктів управління запасами, оскільки постійні зміни можуть значно вплинути на вимоги до функціонування системи.

Основними перевагами використання Agile/Scrum у цьому проєкті є:

- Процес розробки стає більш прозорим і контролованим завдяки регулярним перевіркам результатів;
- Можливість швидко змінити продукт відповідно до нових потреб;
- Покращення якості кінцевого продукту за допомогою постійних перевірок і виявлення та усунення помилок.

Таким чином, реалізація інформаційної системи за допомогою Agile/Scrum максимально відповідає потребам бізнесу та гарантує її стійку адаптивність до змін.

3.2 Архітектура програмної системи

3.2.1 Загальна (логічна) структура інформаційної системи

Інформаційна система управління запасами розподільчого центру має логічну структуру, яка складається з багатьох взаємопов’язаних компонентів, які забезпечують ефективне функціонування системи:

1. **Веб–сервер (Flask backend)** – обробка запитів користувачів, виконання бізнес-логіки, спілкування з базою даних та надання зовнішніх послуг;
2. **База даних (PostgreSQL)** – здійснює надійне зберігання та управління даними системи, включаючи інформацію про товари, замовлення, користувачів тощо.
3. **Фронтенд (Bootstrap, JavaScript, jQuery)** – надає адаптивний, інтерактивний інтерфейс, який робить систему користувача зручною для використання. FETCH-запити (AJAX) використовуються для взаємодії з бекендом;
4. **Компоненти автентифікації та авторизації** – дозволяють користувачам безпечно використовувати функції системи. Паролі, які використовуються для реєстрації користувачів, хешуються та зберігаються в базі даних, що гарантує безпеку;
5. **Компоненти для роботи з IoT–пристроями** – імітація терміналів збору даних (ТЗД) і симуляція вагового обладнання дозволяє без використання реального обладнання тестувати та демонструвати взаємодію системи з IoT-пристроями;
6. **Зовнішні сервіси** – функції чат-бота можна реалізувати за допомогою сторонньої API (https://openrouter.ai/), яка дозволяє користувачам швидко отримувати інформацію та отримувати допомогу в роботі з системою.

Взаємодія компонентів відбувається наступним чином:

- **Фронтенд** надсилає асинхронні запити до **бекенду** через FETCH–запити, що дозволяє створювати швидкий та зручний інтерфейс.
- **Бекенд (Flask)** обробляє запити користувача, виконує необхідну логіку, після чого надсилає відповідь у форматі JSON.
- Взаємодія бекенду з **базою даних** відбувається за допомогою ORM SQLAlchemy, що забезпечує зручну та безпечну роботу з даними.
- **Компоненти IoT** імітуються безпосередньо на бекенді, де обробляються команди та дані, що надходять від симуляторів.

- **Зовнішній сервіс OpenRouter.ai** використовується бекендом для забезпечення роботи чат–бота.

Доцільно використовувати діаграму компонентів UML для демонстрації взаємодії основних складових системи, яка показує інформаційні зв’язки між модулями бекенду, фронтенду, базою даних і зовнішніми сервісами. На рис. 73 (див. Додаток Б) показано спосіб, за допомогою якого різні частини об’єднуються в єдину архітектуру системи управління запасами.

3.2.2 Модульна (фізична) структура системи

Фізична структура системи складається з чітко розподілених модулів, що полегшує процес розробки, супроводу та подальшого масштабування. Програмний продукт складається з наступних основних модулів:

1. **Модуль авторизації та реєстрації користувачів** – відповідає за безпечну авторизацію та реєстрацію користувачів, а також управління ролями доступу до функцій системи.
2. **Модуль управління замовленнями** – дозволяє постачальникам створювати, переглядати, редагувати та видаляти замовлення, а також відстежувати їх статус.
3. **Модуль роботи з постачальниками та контрактами** – дозволяє керувати даними постачальників, а також їхніми позиціями та контрактами.
4. **Модуль роботи з товарами (SKU, асортимент)** – допомагає керувати каталогом товарів, асортиментом і позиціями на складі.
5. **Модуль управління розміщенням товару** – автоматизує розподіл і визначення місця зберігання товару на складі відповідно до його характеристик і умов зберігання (температурний режим).

6. **Модуль роботи з клієнтами** – дозволяє вести інформацію про клієнтів, створювати замовлення та відстежувати статуси замовлень, а також створювати та перевіряти документи клієнтів.
7. **Модуль симуляції IoT–пристроїв** – імітує роботу вагового обладнання та терміналів збору даних для тестування та демонстрації взаємодії з IoT-пристроями.
8. **Модуль інтеграції з зовнішніми сервісами** – забезпечує інтеграцію з API OpenRouter.ai для реалізації функціоналу чат–бота, що допомагає користувачам отримувати інформацію і допомогу.

Фізична організація файлової системи проєкту:

1. project

1.1 forms/ – Форми для взаємодії користувачів;
1.2 migrations/ – Міграції для бази даних;
1.3 models/ – Моделі для бази даних;
1.4 routes/ – Маршрути та обробники запитів;
1.5 static/ – Статичні файли(CSS, JavaScript)
1.6 templates/ – HTML–шаблони сторінок
1.7 utils/ – Утиліти та додаткові функції;
1.8 .env – Конфігураційні змінні середовища;
1.9 app.py – Основний файл програми Flask;
1.10 app_order.py – Логіка управління замовленнями;
1.11 app_supplier.py – Логіка управління постачальниками;
1.12 docker–compose.yml – Конфігурація Docker–контейнерів;
1.13 extensions.py – Розширення Flask(SQLAlchemy);
1.14 init_db.py – Скрипт ініціалізації бази даних;
1.15 LogisticsDate.xlsx – Файл з логістичними даними;
1.16 manage.py – Скрипт керування завданнями системи;

1.17 requirements.txt – Скрипт керування завданнями системи;

1.18 run.py – Точка запуску застосунку.

Така змінена структура полегшує розробку, тестування та супровід програмного продукту, оскільки дозволяє чітко розмежувати функціональність окремих компонентів системи.

Діаграма класів UML зручно демонструє структуру об’єктів і взаємозв’язків у розробленій системі. На рис. 74 (див. Додаток Б) показано класи, які відповідають сутностям бази даних, а також логічні зв’язки між ними.

3.3 Опис процесу функціонування програмної системи

Програмна система управління запасами розподільчого центру працює за допомогою багатьох основних сценаріїв, щоб повністю автоматизувати логістичні операції.

Нижче наведено короткий опис основних сценаріїв роботи системи, включаючи особливості їх реалізації:

1. Реєстрація та авторизація користувачів

Коли користувачі реєструються, їхні паролі хешуються та зберігаються в базі даних. Після авторизації користувач отримує доступ до певних функцій відповідно до своєї ролі.

2. Створення та обробка замовлень постачальникам і клієнтам

Для створення замовлення користувач вказує необхідні позиції та кількість. Система автоматично створює номер замовлення, зберігає дані в базі даних і надає можливість відстежувати стан замовлення.

3. Приймання товару (імітація роботи ТЗД)

Процес приймання товару виконується за допомогою імітації ТЗД. Користувач вводить інформацію про товар, систему перевіряє дані, створює записи у базі даних та автоматично генерує рахунок-фактуру на прийнятий товар.

4. Розміщення товару на складі

Відповідно до температурного режиму товару система автоматично визначає місце для поселення товару. Для кожної позиції визначається конкретне місце відбору товару.

5. Комплектація товару (імітація роботи ТЗД)

Комплектація товару здійснюється з використанням імітації ТЗД. Користувач відбирає товари відповідно до замовлень клієнтів, система перевіряє відповідність замовленню, підтверджує комплектацію та після вивантаження, генерує видаткову накладну.

6. Симуляція вагового обладнання (імітація IoT)

Система дозволяє імітувати роботу вагового обладнання для створення штрих-кодів, що зв’язані з вагою брутто піддона, це необхідно для контролю і точності ведення складського обліку.

7. Взаємодія з чат-ботом

Користувачі можуть взаємодіяти з чат-ботом, який обробляє запити за допомогою API OpenRouter.ai. Чат-бот автоматично перемикається на обробку стандартних попередньо прописаних запитів у разі відсутності зв’язку з цим сервісом.

Особливості процесів:

- Товар розміщується відповідно до його температурного режиму, щоб створити ідеальні умови зберігання.

- Автоматично створюється рахунок-фактура після отримання товару.
- Після обробки замовлень клієнтів автоматично формуються номери лотів.
- Клієнту автоматично створюється видаткова накладна після відвантаження товару.

Таким чином, сценарії функціонування, описані вище, гарантують високий рівень автоматизації та оптимізації складських процесів.

3.4 Докладний опис компонентів програмного забезпечення

3.4.1 Загальні відомості про розроблені програмні модулі

Програмна система управління запасами розподільчого центру складається з багатьох спеціалізованих модулів, які дозволяють виконувати всі важливі завдання.

Нижче наведено перелік основних модулів, їх функціональних призначення, використаних технологій і технічних вимог:

1. **Авторизація (app.py)**
   Призначення: Вхід у систему, логін користувача.
   Технології: Flask, Python.
   Вимоги: Web-браузер, сервер з Flask.
2. **Управління замовленнями (orders.py, app_order.py)**
   Призначення: Створення, перегляд, управління статусами замовлень.
   Технології: Flask, SQLAlchemy, PostgreSQL.
   Вимоги: БД PostgreSQL, сервер Flask.
3. **Комплектація (picking.py)**
   Призначення: Вибір товару за замовленням, оновлення статусу.

Технології: Flask, JavaScript, TSD Emulator, SQLAlchemy.
Вимоги: TSD UI, PostgreSQL, браузер.

4. **Обробка лотів (order_lot.py)**
   Призначення: Формування, відстеження, управління статусами лотів.
   Технології: SQLAlchemy, Flask.
   Вимоги: PostgreSQL.
5. **Приймання товару (tsd_batch_receiving.py, received_inventory.py)**
   Призначення: Приймання через термінал збору даних, фіксація SSCC.
   Технології: Flask, JavaScript, SQLAlchemy.
   Вимоги: IoT-емулятор, PostgreSQL.
6. **Складські залишки (warehouse_stock.py, inventory.py)**
   Призначення: Відображення стану складу, залишків, температурних зон.
   Технології: SQLAlchemy, Flask.
   Вимоги: PostgreSQL, браузер.
7. **Відвантаження/трансфери (transfers.py, pending_transfer.py)**
   Призначення: Формування та управління переміщеннями між зонами складу.
   Технології: Flask, SQLAlchemy.
   Вимоги: Серверна логіка.
8. **Клієнтські залишки (client_stock.py)**
   Призначення: Перегляд залишків, доступних клієнтам.
   Технології: Flask API.
   Вимоги: Web-клієнт, API.
9. **Звітність (inventory_report.py)**
   Призначення: Генерація звітів щодо приймання, залишків та відвантажень.
   Технології: Flask, PDF/Excel, SQLAlchemy.
   Вимоги: Сумісність з Excel, браузер.
10. **Контракти з постачальниками (app_supplier.py, supplier.py)**
    Призначення: Управління інформацією про постачальників та контракти.

Технології: Flask, SQLAlchemy.
Вимоги: PostgreSQL.

11. **Інтерфейс чат-бота (chatbot.py)**
Призначення: Обробка запитів та відповідей через AI (GPT/OpenRouter).
Технології: Flask, OpenRouter.ai, JavaScript.
Вимоги: Інтернет, API-доступ.

12. **Валідація (validation.py)**
Призначення: Внутрішні перевірки та валідація даних.
Технології: Python.
Вимоги: -

13. **UI-логіка TSD (tsd-emulator.py)**
Призначення: Симуляція терміналу збору даних, обробка клавіш.
Технології: JavaScript, jQuery, Bootstrap.
Вимоги: Браузер.

14. **Штрих-кодування (sscc_barcode.py)**
Призначення: Генерація та розпізнавання штрих-кодів SSCC.
Технології: Python, SQLAlchemy.
Вимоги: -

15. **Скалярні операції (scales.py)**
Призначення: Робота з вагами (імітація або інтеграція з реальними вагами).
Технології: Python, Flask.
Вимоги: Підключення до фізичних/віртуальних ваг.

16. **Поселення товару (picking_location.py)**
Призначення: Управління місцями розміщення та відбору товару.
Технології: Flask, SQLAlchemy, PostgreSQL.
Вимоги: PostgreSQL, браузер.

Структура модулів, представлена тут, забезпечує повне охоплення завдань складської логістики, ефективне управління запасами та простоту масштабування та супроводу системи.

3.4.2 Функціональне призначення модулів системи

Логічний поділ усіх компонентів системи на категорії відповідно до їхнього призначення в загальній архітектурі програмного забезпечення визначає функційне призначення модулів системи. Це дозволяє оптимізувати процеси управління запасами та ефективно виконувати основні завдання розподільчого центру.

- **Модулі взаємодії з користувачами:**
  - **Авторизація (app.py)** – надає користувачам можливість входу в систему з урахуванням їх ролей та рівнів доступу.
  - **Інтерфейс ТЗД (tsd-emulator.py, tsd-picking.js)** – забезпечує емуляцію терміналу збору даних для взаємодії з системою на складі.
  - **Чат-бот (chatbot.py)** – дозволяє користувачам отримувати інформацію та відповіді на запити через інтерфейс OpenRouter.ai або локальні сценарії.
- **Модулі управління бізнес-процесами:**
  - **Приймання товару (tsd_batch_receiving.py, received_inventory.py)** – обробка приймання товарів через термінал, фіксація SSCC та занесення в базу.
  - **Комплектація (picking.py)** – виконання відбору товару відповідно до замовлень.
  - **Обробка лотів (order_lot.py)** – формування та контроль статусів лотів.

- **Відвантаження та трансфери (transfers.py, pending_transfer.py)** – логіка переміщення товарів між зонами складу та підготовка до відвантаження клієнтам.
- **Управління замовленнями (orders.py, app_order.py)** – створення, редагування та контроль замовлень постачальникам та клієнтам.

➢ **Модулі управління довідниками:**

- **Контракти з постачальниками (app_supplier.py, supplier.py)** – управління постачальниками, їх реквізитами та контрактами.
- **Складські залишки (warehouse_stock.py, inventory.py)** – зберігання та відображення залишків на складі з урахуванням температурних зон.
- **Клієнтські залишки (client_stock.py)** – обмежений доступ клієнтів до даних про наявність товарів.
- **Поселення товару (picking_location.py)** – визначення місць відбору для кожної товарної позиції з урахуванням правил зберігання.

➢ **Модулі автоматизації:**

- **Штрих-кодування (sscc_barcode.py)** – генерація та обробка унікальних штрих-кодів SSCC.
- **Скалярні операції (scales.py)** – взаємодія з віртуальними або реальними вагами під час приймання або відвантаження товарів.
- **Автоматична генерація документів** – створення рахунків-фактур, видаткових накладних, номерів лотів після обробки.

➢ **Модулі контролю і звітності:**

- **Валідація (validation.py)** – внутрішні перевірки достовірності та повноти даних.

- **Звітність (inventory_report.py)** – формування звітів щодо приймання, залишків, відвантажень тощо.

У результаті цього розділення модулів можна підтримувати високу керованість системи, розширюваність функціоналу та простоту технічного супроводу.

3.4.3 Особливості інсталяції програмного продукту

Програмний продукт інсталюється локально або на сервері за допомогою контейнера Docker. Такий метод зменшує помилки налаштування середовища, полегшує розгортання системи на різних платформах і гарантує стабільність роботи.

Вимоги до середовища:

- **Операційна система**: Windows(x64) / Linux (рекомендовано: Ubuntu Server 20.04+)
- **Процесор**: Intel Core i5 або вище
- **Оперативна пам’ять**: від 4 ГБ
- **Сховище**: не менше 1 ГБ вільного місця
- **Потрібне ПЗ**:
  - Python 3.10+
  - Docker + Docker Compose (при використанні контейнеризації)
  - PostgreSQL (якщо запуск без Docker)
  - Git (для клонування репозиторію)

Етапи інсталяції (локально або у Docker):

1. Клонування проєкту з репозиторію.

   git clone <URL репозиторію>

cd <назва проєкту>

2. Налаштування середовища: Створити файл .env та docker-compose.yml з параметрами підключення до БД, API-ключами, Flask-секретами
3. Запуск із використанням Docker, це створить контейнери для Flask, PostgreSQL та інших сервісів:

    - *docker-compose up –d*

4. Ініціалізація бази даних:

    - У консолі Windows чи консолі будь-якої IDE вводимо: *python init_db.py*

      python run.py

5. Запуск системи у браузері:

    - У рядок адреси вводимо: *http://localhost:5000*

Альтернативний запуск без Docker:

1. Створюємо віртуальне середовище, у консолі вводимо дані команди:

   python -m venv venv

   source venv/bin/activate (або .\venv\Scripts\activate для Windows)

2. Встановлюємо залежності:

   pip install -r requirements.txt

3. Ініціалізуємо базу:

   python init_db.py

4. Запускаємо додаток:

   flask run

3.5 Тестування програмного продукту

3.5.1 Розробка методики тестування

Мета тестування програмного продукту полягає в тому, щоб переконатися, що всі основні функціональні модулі інформаційної системи управління запасами працюють правильно, а також виявити помилки, які можуть виникнути через неправильні дані або в результаті взаємодії з користувачем.

Типи використаних тестів

Для цього проєкту були застосовані такі види тестування:

- **Функціональне тестування** – перевірка коректної роботи кожного модуля відповідно до вимог.
- **Інтеграційне тестування** – перевірка взаємодії між модулями (наприклад, замовлення ↔ приймання ↔ відвантаження).
- **Тестування інтерфейсів** – перевірка правильності відображення елементів інтерфейсу та взаємодії з користувачем.
- **Тестування стійкості** – моделювання відмов з'єднання з API OpenRouter.ai, перевірка роботи чат-бота в автономному режимі.

Обрана методика

Метод, який було обрано, був метод тестування «білого ящика», який дозволяє перевірити, чи можна отримати доступ до внутрішньої структури програмного коду. Це дозволяє проводити аналіз логіки виконання, перевіряти всі гілки умов, цикли та обробку помилок і винятків. Такий метод надає широке

покриття функціоналу; це особливо важливо для важливих логістичних сценаріїв.

Ключові сценарії тестування

1. **Авторизація користувача**
    - Перевірка коректного логіну з валідними даними.
    - Відображення повідомлення при невірному паролі.
2. **Створення замовлення постачальнику**
    - Вибір постачальника, контракту, додавання позицій.
    - Перевірка автоматичної генерації номера замовлення.
3. **Приймання товару через ТЗД**
    - Введення SSCC, перевірка реєстрації прийнятого товару.
    - Генерація рахунку-фактури після приймання.
4. **Розміщення товару за температурним режимом**
    - Вибір комірки відповідно до типу товару.
    - Перевірка правильності призначення місця відбору.
5. **Комплектація клієнтського замовлення**
    - Відбір товару, оновлення статусу.
    - Генерація номера лота.
6. **Робота чат-бота**
    - Запит до OpenRouter.ai та отримання відповіді.
    - Перевірка fallback-сценарію при втраті підключення.
7. **Генерація документів**
    - Рахунок-фактура, видаткова накладна, звіти в Excel.
    - Перевірка цілісності PDF/Excel-документів.

Критерії прийнятності

- Всі функції працюють відповідно до специфікації.
- Дані записуються до бази без помилок.
- Відповіді на запити повертаються у встановленому форматі.

- Користувацький інтерфейс коректно відображає стани.

Тестування «білого ящика» дало можливість детально вивчити внутрішню логіку системи, перевірити контрольні точки коду, обробити виключення та переконатися, що модулі працюють стабільно в різних умовах.

3.5.2 Аналіз результатів тестування

Результати тестування представлені в Додатку А. Він показав, що всі основні модулі інформаційної системи управління запасами пройшли тестування функціональності. Тестування охоплювало весь процес, починаючи від авторизації користувача, закінчуючи формуванням звітності та взаємодією з чат-ботом.

Загальні висновки:

- Кожен основний функціональний модуль системи виконує технічні вимоги.
- Жодна з функцій, перевірених, не призвела до критичних помилок або збоїв у роботі системи.
- Автоматична генерація лотів, рахунків-фактур, накладних і номерів замовлень працює коректно.
- Приймання та комплектація товару через імітатор ТЗД виконуються стабільно, всі введені дані фіксуються в базі.
- Через відсутність оплати OpenRouter.ai наразі недоступний. Тому модуль чат-бота працює лише в режимі офлайн з використанням заздалегідь заданих відповідей. Проте, запити, на які бот може відреагувати обмежені та потребують корегування.

Виявлені недоліки:

- Під час тестування були помітні незначні затримки при взаємодії з інтерфейсом ТЗД у браузерах з низькою продуктивністю; необхідно оптимізувати обробку JavaScript-подій.
- У різних режимах ТЗД кнопки F1–F6 виконують різні дії. У кожному випадку необхідно надати чітке пояснення функцій цих клавіш у письмовій або візуальній формі.
- Низька різноманітність запитів, на які чат-бот може відреагувати коректно.

Загальна оцінка

Інформаційна система повністю стабільна та відповідає вимогам. Тим не менш, наявність проблем, таких як нечітка інструкція щодо функціональних клавіш і непрацюючий зовнішній чат-сервіс, вказує на те, що використовувати програму в реальному житті зараз передчасно. Необхідно доопрацювати логіку інтерфейсу та усунути обмеження щодо зовнішніх сервісів.

4. ТЕХНІКО-ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ РОЗРОБКИ ПІДСИСТЕМИ

4.1 Вступ

У сучасних умовах цифрової трансформації підприємств інформаційні технології є життєво важливими для ефективного управління матеріальними потоками, зокрема у галузі логістики та обліку запасів. Інформаційні системи управління запасами покращують контроль над ресурсами підприємства, зменшують помилки та прискорюють обробку замовлень.

Техніко-економічне обґрунтування розробки інформаційної системи є важливою частиною процесу впровадження, оскільки воно дозволяє оцінити доцільність інвестицій у створення програмного продукту, потенційні вигоди від його використання та ризики, пов’язані з реалізацією проєкту.

У цьому розділі розглядаються ергономічні функції інтерфейсу користувача, а також техніко-економічні аспекти створення та впровадження інформаційної системи управління запасами в умовах діяльності розподільчого центру сільськогосподарської продукції. Оцінка ринку збуту, аналіз конкуренції, планування виробництва програмного продукту, маркетингова стратегія, організаційні та фінансові складові, а також фактори ризику та стратегія фінансування представлені в розділі.

4.2 Ергономіка ІТ

Ергономіка інформаційних технологій включає розробку інтерфейсів, які дозволяють користувачам комфортно, ефективно та без помилок працювати з програмними продуктами. Інтерфейс повинен бути простим для користувача, візуально впорядкованим, мати логічно згруповані елементи керування та оперативну зворотну реакцію системи.

- **Основні ергономічні цілі:**

- Зменшення кількості дій, необхідних для виконання задачі;
- Зниження ймовірності мимовільних помилок;
- Забезпечення самонавчання інтерфейсу без зовнішньої інструкції;
- Стандартизований вигляд елементів керування;
- Мінімізація кількості введення користувачем;
- Візуальна привабливість і простота.

➢ **Особливості реалізації інтерфейсу в системі:**

Інтерфейс розробленої системи складається з таких компонентів:

- Веб-інтерфейс, створений за допомогою Bootstrap, забезпечує адаптивність і простоту навігації.
- Табличні форми відображення даних для замовлень, залишків, приймання та відвантаження.
- Емулятор терміналу збору даних (ТЗД), реалізований на JavaScript, дозволяє моделювати роботу операторів складу.
- Повідомлення системи про помилки, успішні дії та підтвердження – візуально виокремлені та контрастні.
- Простий чат-інтерфейс, який дозволяє ставити запитання системі та отримувати відповіді.

➢ **Виявлені ергономічні проблеми:**

- Функціональні клавіші F1–F6 в інтерфейсі ТЗД не мають чіткого опису дій, які вони виконують. Це викликає плутанину в користувачів.
- У деяких формах відсутні підказки щодо введення або уточнення помилок.
- Недостатня візуальна реакція на некоректні дії користувача в окремих сценаріях.
- Інтерфейс чат-бота не інформує користувача про втрату з’єднання з OpenRouter.ai, через що відповідь не надходить без видимих причин.

➢ **Оцінка зручності**

Інтерфейс системи загалом є зрозумілим і логічно структурованим. Базові задачі (створення замовлення, приймання, перегляд залишків) виконуються швидко, без додаткових пояснень. Після вдосконалення підказок до клавіш і покращення повідомлень система буде готовою до використання у виробничих умовах.

- **Висновок**

Інтерфейс розробленої системи відповідає більшості вимог до ергономічності ПЗ. Виявлені недоліки не є критичними та можуть бути усунені на етапі доопрацювання. Впровадження елементів інтерфейсної підтримки (контекстні підказки, індикатори статусу, документація) дозволить значно покращити зручність використання системи для кінцевого користувача.

4.3 Резюме проєкту

Назва проєкту: Інформаційна система управління запасами розподільчого центру сільськогосподарської продукції

Місце реалізації: логістичний склад або розподільчий центр, що займається прийманням, зберіганням, комплектацією та видачею овочів і фруктів.

Мета проєкту: розробка програмного продукту, який допоможе автоматизувати процеси управління запасами, забезпечити простий і ефективний облік товарів, мінімізувати вплив людей і підвищити точність складування та відвантаження продукції.

Суть проєкту полягає у створенні веб-орієнтованої інформаційної системи, що включає такі функціональні модулі:

- облік постачальників та контрактів;
- створення та обробка замовлень;
- приймання товару із симуляцією ТЗД;
- розміщення продукції за температурними зонами;
- управління лотами і комплектація замовлень клієнтів;

- автоматичне формування документів (рахунок-фактура, видаткова накладна);
- відображення залишків складу та клієнтських залишків;
- чат-бот для пошуку інформації та допомоги користувачу.

Система розроблена для малих і середніх аграрних підприємств, які потребують автоматизації обліку продукції, підвищення точності логістичних операцій і зменшення витрат, пов’язаних із ручним введенням даних. Програмний продукт має відкриту структуру, що дозволяє його адаптувати до потреб конкретного бізнесу.

4.4 Опис проєктованого продукту або виду послуг

Розроблений програмний продукт є веб-орієнтованою інформаційною системою, яка дозволяє аграрним логістичним підприємствам автоматизувати управління складськими запасами. Програмне забезпечення дозволяє вести єдиний облік для всіх складських операцій, відображає залишки в режимі реального часу та дозволяє керувати зв’язками з постачальниками та клієнтами.

Основні функціональні можливості продукту:

- Ведення бази даних постачальників, клієнтів та їхніх контрактів;
- Створення замовлень з контролем статусів та етапів виконання;
- Реєстрація приймання товару із вказанням SSCC-коду, температурного режиму, ваги та кількості;
- Автоматичне визначення зони зберігання на основі параметрів продукції;
- Комплектація замовлень з місць відбору із зазначенням лоту;
- Генерація супровідних документів (рахунки-фактури, видаткові накладні);
- Відображення складських залишків у розрізі температурних зон та клієнтів;
- Візуальна інтерфейсна частина для роботи з емульованими ТЗД і вагами;
- Довідковий чат-бот для допомоги користувачам у пошуку інформації.

Програмний продукт має відкриту архітектуру, підтримує масштабування та може бути інтегрований у існуючу IT-інфраструктуру компанії. Продукт пропонує гнучкість, стабільність і простоту підтримки завдяки використанню сучасних технологій (Flask, PostgreSQL, JavaScript і Docker).

Система розроблена для підприємств, які мають логістичні склади та потребують ретельного обліку овочів, фруктів та іншої сільськогосподарської продукції. Мета системи полягає в тому, щоб зменшити витрати на ручний контроль товарів, помилки персоналу та тривалість операційних процесів.

4.5 Оцінка ринку збуту

Ринок збуту інформаційної системи управління запасами включає логістичні компанії, торговельні мережі, фермерські господарства з власною логістичною інфраструктурою, а також розподільчі центри, які відповідають за прийом, зберігання, сортування та комплектацію овочів.

Більшість підприємств аграрної логістики потребують автоматизації процесів, оскільки вимоги до якості обліку та швидкості складських операцій постійно зростають. Попит на такі рішення зростає через низьку продуктивність ручного обліку, високу частоту людських помилок, необхідність відстеження партій продукції та необхідність контролю температури.

Цільовий ринок складають:

- Розподільчі центри аграрної продукції (овочі, фрукти);
- Логістичні комплекси оптових ринків та супермаркетів;
- Фермерські кооперативи з централізованим зберіганням та збутом продукції;
- Компанії-експортери, які мають склади з холодильною логістикою;
- Агрохолдинги з власними логістичними ланцюгами.

Згідно з відкритими даними, в Україні існує понад 100 спеціалізованих розподільчих центрів і овочесховищ, які можуть бути зацікавлені в сучасних IT-рішеннях для автоматизації. Крім того, все більше господарств починають використовувати складське ПЗ як частину ERP або WMS-платформи. Це дає можливість інтегрувати систему, пропоновану, на більший ринок корпоративних рішень.

Таким чином, ринок збуту є досить широким, а розроблений продукт може задовольнити потреби як малих підприємств, так і великих логістичних операторів завдяки своїй модульній структурі та можливості налаштування відповідно до потреб конкретного клієнта.

4.6 Аналіз конкуренції

На ринку програмного забезпечення для управління запасами в логістиці та сільському господарстві доступні локальні та міжнародні розробки.

Основними конкурентами запропонованої інформаційної системи є:

- **1С:Підприємство – Модуль "Управління торгівлею" / "Склад"**
  - **Переваги**: широка підтримка; інтеграція з бухгалтерією; гнучка адаптація під бізнес-процеси;
  - **Недоліки**: складність у налаштуванні; громіздкий інтерфейс; не всі модулі адаптовані до специфіки агрологістики.
- **SAP Extended Warehouse Management (EWM)**
  - **Переваги**: потужний інструмент для великих корпорацій; підтримка міжнародних стандартів логістики;
  - **Недоліки**: висока вартість ліцензії; складність впровадження; надлишкова функціональність для середніх компаній.
- **Zoho Inventory, Odoo, NetSuite**

- **Переваги**: хмарні сервіси, прості у впровадженні; зручний інтерфейс;
- **Недоліки**: обмежені можливості кастомізації; фокус на e-commerce, а не на склади з температурним режимом.

➢ **Локальні розробки та Excel-облік**

- **Переваги**: дешевизна, можливість власного впровадження;
- **Недоліки**: відсутність автоматизації, людський фактор, відсутність підтримки масштабування та інтеграції з IoT.

Розроблена система має низку конкурентних переваг:

- Спеціалізація саме на аграрному секторі (робота з фруктами, овочами);
- Наявність симуляції терміналів збору даних і вагового обладнання;
- Проста інтеграція, модульність, легкість налаштування;
- Відкритий код, можливість кастомізації під конкретного клієнта;
- Менша вартість впровадження порівняно з великими системами типу SAP.

Таким чином, запропонована система є привабливою для малого та середнього бізнесу в агрологістиці, оскільки вона займає проміжну нішу між дорогими корпоративними продуктами та примітивними локальними рішеннями.

4.7 Стратегія маркетингу

Стратегія маркетингу, розроблена для просування інформаційної системи управління запасами на ринку агрологістики, враховує цільову аудиторію, канали збуту, ціни та методи просування.

➢ **Цільова аудиторія:**

- Малі та середні аграрні підприємства з власними складами;
- Розподільчі центри супермаркетів і торгових мереж;
- Фермерські об’єднання та кооперативи;

- Компанії, що експортують плодоовочеву продукцію.

➢ **Основні канали збуту:**

- Прямий продаж через сайти і презентації;
- Співпраця з IT-компаніями, що впроваджують рішення для агросектору;
- Участь у виставках і форумах (AgroExpo, Фермер України, Smart Agro);
- Інтернет-маркетинг: SEO, email-розсилка, тематичні блоги, реклама в Facebook/LinkedIn.

➢ **Ціноутворення:**

- Базова версія з відкритим вихідним кодом (безкоштовна або з разовою оплатою за встановлення);
- Платна адаптація під замовника: кастомні модулі, інтеграція, підтримка;
- SaaS-модель (підписка) для хмарного використання у майбутньому.

➢ **Засоби просування:**

- Демонстраційний сайт з інтерфейсом системи та відео-інструкціями;
- Пробний період використання на обмеженій кількості SKU;
- Публікації у професійних ЗМІ аграрного сектору та IT-сфері;
- Відгуки перших клієнтів і кейси впровадження.

Реалізація маркетингової стратегії дозволить ефективно донести цінність продукту до потенційних клієнтів і поступово розширювати клієнтську базу.

4.8 План виробництва

Цільова аудиторія, канали збуту, ціни та методи просування є елементами стратегії маркетингу, яка розроблена для просування інформаційної системи управління запасами на ринку агрологістики.

➢ **Етапи виробничого процесу:**

1. **Аналіз вимог:**

- Збір функціональних вимог від потенційних користувачів: працівників складу, логістів, операторів.
- Вивчення процесів приймання, комплектації, розміщення і видачі товару на реальному складі.

2. **Проєктування системи:**
   - Створення логічної моделі інформаційної системи.
   - Розробка структури бази даних, опис основних сутностей: замовлення, партія, SSCC, місце зберігання тощо.
   - Проєктування інтерфейсів: веб-панелі, емуляторів ТЗД та чат-бота.
3. **Розробка програмного забезпечення:**
   - Реалізація бекенду на Flask з використанням SQLAlchemy для роботи з PostgreSQL.
   - Створення REST API для взаємодії з фронтендом.
   - Побудова інтерфейсу на Bootstrap + JavaScript + jQuery.
   - Розробка імітаторів ТЗД та вагового обладнання.
   - Генерація супровідної документації: рахунок-фактура, видаткова накладна, лоти.
4. **Тестування:**
   - Проведення тестування методом білого ящика.
   - Тестування приймання, комплектації, розміщення, обробки чат-запитів, генерації документів.
   - Виявлено декілька проблемних місць
5. **Впровадження та навчання:**
   - Підготовка інструкцій користувача.
   - Демонстрація роботи системи з використанням симулятора.
   - Оцінка зручності інтерфейсу та складання рекомендацій для майбутнього впровадження.

➢ **Необхідні ресурси:**

1. **Людські ресурси:**

- **Розробник програмного забезпечення:** реалізація логіки, бази даних, інтерфейсів, API.
- **Тестувальник:** ручне функціональне тестування, аналіз помилок.
- **Консультант зі складської логістики:** визначення реальних сценаріїв роботи та адаптація під них.

2. **Технічні ресурси:**
   - Середовище розробки (Python, Flask, PostgreSQL, GitHub).
   - Інструменти тестування та віртуалізації (Docker, браузери, SQL GUI).
   - Емулятори терміналів збору даних і ваг.

Програмний продукт був розроблений таким чином, щоб він міг адаптуватися до умов різних організацій.

4.9 Організаційний план

Організаційний план розробки програмного забезпечення визначає структуру взаємодії між технічними та функціональними напрямками та склад команди.

➢ **Організаційна структура проєкту:**

1. **Керівник проєкту:**
   - Координує всі етапи проєкту.
   - Визначає пріоритети розробки.
   - Відповідає за інтеграцію всіх модулів системи.
2. **Розробник програмного забезпечення:**
   - Реалізує логіку бекенду (Flask, SQLAlchemy).
   - Створює веб-інтерфейс користувача (HTML, CSS, Bootstrap, jQuery).
   - Впроваджує інтеграцію з базою даних PostgreSQL.
   - Розробляє інтерфейс ТЗД та ваг.
3. **Тестувальник:**
   - Виконує ручне тестування системи.

- Аналізує логіку роботи кожного модуля.
- Виявляє баги, перевіряє коректність збереження даних та взаємодії між компонентами.

➢ **Організація роботи над проєктом:**

- Розробка здійснювалася за гнучкою методологією (Agile/Scrum) із поетапною реалізацією функціональних частин.
- Проміжні результати тестувалися та доопрацьовувалися у відповідь на виявлені недоліки.
- Для організації версійного контролю використовувався Git.

Проєкт був розроблений індивідуально, що дозволить команді розширюватися після комерційного впровадження. Такий метод дозволив швидко приймати технічні рішення та зосередитися на важливих модулях системи. Це також дозволило створити повнофункціональний прототип, який може бути використаний у логістичній практиці.

4.10 Юридичний план

1. Форма власності:

- Якщо програмний продукт буде продаватися, планується створити товариство з обмеженою відповідальністю (ТОВ). Це дозволить легально вести бізнес, укладати договори з партнерами та клієнтами, а також обмежити відповідальність засновників.

2. Юридичні аспекти:

- **Реєстрація компанії:** офіційна реєстрація юридичної особи у відповідних державних органах України.
- **Ліцензування діяльності:** розробка та розробка програмного забезпечення не вимагають окремої ліцензії, однак у разі використання API сторонніх сервісів необхідно дотримуватись умов їх використання.

- **Дотримання чинного законодавства:** податкове та трудове законодавство, правила обробки персональних даних (Закон України «Про захист персональних даних»).

3. Захист інтелектуальної власності:

- **Авторські права:** програмний код, дизайн інтерфейсів та структура бази даних можуть бути зареєстровані як об’єкти авторського права.
- **Ліцензії на ПЗ:** система може розповсюджуватись за умовами відкритої (наприклад, MIT) або закритої ліцензії з підписанням відповідних угод з користувачами.

4. Контракти та угоди:

- **Трудові договори:** у разі розширення команди буде передбачено офіційне працевлаштування розробників, тестувальників, консультантів.
- **Договори з клієнтами:** комерційні угоди з чіткими умовами відповідальності та гарантіями укладаються на адаптацію системи, технічну підтримку та впровадження.

У всьому життєвому циклі продукту, від розробки до продажу та обслуговування клієнтів, юридичні плани гарантують його правову безпеку.

4.11 Фінансовий план та стратегія фінансування

- **Оцінка витрат:**

  - **Заробітна плата персоналу:** витрати на оплату праці основних учасників проєкту, таких як розробник, тестувальник, консультант з логістики та спеціаліст підтримки.
  - **Оренда офісу:** хоча розробка здійснювалася віддалено на початковому етапі, у разі розвитку проєкту можуть виникнути витрати на оренду приміщення.

- **Обладнання та програмне забезпечення:** витрати на комп'ютери та сервери, ліцензії на роботу з базами даних, фреймворки та тестові інструменти..
- **Розробка та тестування:** реалізація функціоналу, UX/UI-допрацювання, тестування модулів та інтеграції (одноразові та поточні витрати).
- **Маркетинг та реклама:** SEO-просування, участь у виставках, створення промоматеріалів, реклама в соціальних мережах.
- **Адміністративні витрати:** юридичне оформлення, бухгалтерія, реєстрація ТОВ/ФОП, підготовка технічної документації.

➢ **Прогноз доходів:**

- **Продаж програмного забезпечення:** разова оплата від клієнтів за інсталяцію та налаштування системи.
- **Консультаційні послуги:** впровадження, навчання персоналу, адаптація системи під потреби замовника.
- **Підписка на оновлення:** регулярна підтримка, резервне копіювання, оновлення модулів.

➢ **Аналіз рентабельності:**

- **Розрахунок чистого прибутку:** очікувана окупність проєкту можлива при залученні 2–3 клієнтів протягом перших 3–6 місяців.
- **Показники рентабельності:**
  - **ROS (рентабельність продажів):** оцінюється в межах 30–40%.
  - **ROA (рентабельність активів):** зростає при масштабуванні клієнтської бази.

➢ **Стратегія фінансування:**

- **Інвестори:** потенційне залучення венчурного капіталу, бізнес-ангелів або агрофінансових фондів, зацікавлених у цифровізації логістики.
- **Кредити та гранти:** можливе фінансування з боку банків, участь у грантових програмах для підтримки цифрових проєктів у агросекторі.

- **Доходи від попередніх продажів:** а допомогою моделі попередніх замовлень можна залучити стартовий капітал для впровадження мінімально життєздатного продукту, також відомого як MVP.

Таким чином, запропонована модель фінансування є гнучким і поєднує самофінансування, допомогу ззовні та стратегію повної окупності в середньостроковій перспективі.

4.12 Оцінка ризику

Розробка, впровадження та комерціалізація інформаційної системи можуть нести низку ризиків. Оцінка ризиків дозволяє планувати, як уникнути або мінімізувати наслідки.

➢ **Основні ризики:**

1. **Технічні ризики:**
   - Виникнення помилок у роботі системи в умовах високого навантаження;
   - Несумісність із технічним обладнанням замовника (наприклад, старі ТЗД або специфічні ваги);
   - Затримки в інтеграції з іншими внутрішніми системами підприємства.
2. **Фінансові ризики:**
   - Недостатнє фінансування на етапі впровадження або масштабування проєкту;
   - Низький попит у перші місяці, що може вплинути на окупність.
3. **Юридичні ризики:**
   - Неврахування змін у законодавстві щодо обробки персональних даних або комерційної діяльності;
   - Використання сторонніх сервісів без чіткого ліцензійного обґрунтування.
4. **Маркетингові ризики:**
   - Низька впізнаваність бренду серед потенційних клієнтів;
   - Висока конкуренція зі сторони великих рішень (1С, SAP тощо);

- Недостатня активність у рекламних каналах.

5. **Організаційні ризики:**
    - Затримки у формуванні команди або недостатній рівень кваліфікації учасників;
    - Відсутність зворотного зв’язку від перших клієнтів для подальшого вдосконалення.

➢ **Заходи щодо мінімізації:**

- Проведення пілотного впровадження на одному складі для виявлення технічних недоліків;
- Гнучке бюджетування з урахуванням резервів на непередбачені витрати;
- Періодичний аудит відповідності законодавству та дотримання умов API;
- Поступове розширення рекламної кампанії та створення реальних кейсів використання;
- Формування резервної команди підтримки та підготовка документації для користувачів.

Зазвичай IT-проєкти стикаються з ризиками, але їх можна ефективно контролювати за допомогою правильного планування, управління та гнучкої адаптації стратегії впровадження.

4.13 Страхування

Страхування розглядається як додатковий захід для мінімізації фінансових, технічних і юридичних ризиків під час впровадження інформаційної системи управління запасами. Що стосується комерційного використання продукту, особливо важливо передбачити механізми захисту.

➢ **Види страхування, які можуть бути актуальними:**

1. **Страхування професійної відповідальності розробника:**
   Захист від можливих претензій клієнтів у разі помилок у роботі системи або збоїв, які призвели до збитків.
2. **Страхування кіберризиків:**

Враховуючи використання баз даних і мережеву інфраструктуру, може бути актуальним захист від витоку інформації, збоїв через атаки, втрат доступу тощо.

3. **Майнове страхування офісного обладнання:**

   За наявності фізичного офісу – покриття ризиків, пов'язаних із пошкодженням чи крадіжкою техніки.

4. **Медичне страхування персоналу:**

   За умови розширення команди до повноцінної IT-компанії.

- **Додаткові заходи безпеки:**

  - Резервне копіювання бази даних і логів;
  - Впровадження політик безпеки (паролі, контроль доступу, логи);
  - Документація відповідальностей та гарантій у контрактах з клієнтами.

  Хоча страхування не є обов'язковим на початковому етапі розробки, у разі масштабування системи до комерційного продукту доцільно включити страхові механізми до загальної стратегії захисту бізнесу та клієнтських інтересів.

5. ВИСНОВКИ

У даній дипломній роботі було проведено дослідження, проєктування та реалізацію інформаційної системи управління запасами розподільчого центру сільськогосподарської продукції. Розглянуто особливості логістичних процесів в агропромисловості, визначено вимоги до обліку товарів з температурними режимами та розроблено комплексну систему, яка забезпечує приймання, зберігання, комплектацію та відвантаження продуктів.

Висока кількість помилок, затримки в роботі, відсутність контролю за залишками та складське розміщення були основними проблемами ручного обліку. Інформаційна система, запропонована тут, дозволить автоматизувати ці операції, зменшити вплив людського фактору та підвищити ефективність функціонування складу.

У системі є багато модулів, включаючи управління замовленнями, приймання товарів з імітацією ТЗД, облік залишків, чат-бот для підтримки користувачів і формування документів, таких як рахунок-фактура та видаткова накладна. У процесі реалізації використовувалися сучасні веб-технології, такі як Flask, PostgreSQL, JavaScript і Bootstrap, а також розроблено інтерфейсну логіку, зручну для операторів складу.

Проведено тестування системи за сценаріями прийому, розміщення та комплектації. Відсутність підказок до функціональних клавіш у ТЗД є одним із кількох ергономічних недоліків, які були виявлені. Незважаючи на це, система працювала без проблем і відповідала функціональним вимогам.

Здійснено техніко-економічне обґрунтування розробки, включаючи аналіз ринку, конкуренції, створення фінансової моделі та оцінку ризиків. Результати показали, що система може бути корисною для використання в малих та середніх підприємствах сільського господарства.

Таким чином, дана дипломна робота зробила значний внесок у розвиток агрологістичних IT-рішень. Розроблена система має потенціал для

масштабування та подальшого розвитку, і вона може служити основою для впровадження автоматизованого обліку на аграрних складах.

ДОДАТКИ

Додаток А. Інтерфейс та тестування програми

Тестування програмного продукту.

Таблиця 1 Основні сценарії тестування

№	Модуль	Сценарій	Очікуваний результат
1	Реєстрація\авторизація	Введення правильного/неправильного логіну і паролю	Вхід у систему або повідомлення про помилку
2	Закріплення місця відбору	Призначення зони відбору для SKU	Дані збережено, відображено в інтерфейсі
3	Створення постачальника\ контракта\замовлення на приймання	Послідовне створення постачальника, контракту, замовлення	Дані збережені, сформовано номер замовлення
4	Симуляція вагів\генерація штрих-кодів	Введення ваги брутто, генерація штрих-коду.	Згенеровано SSCC, прив’язаний до ваги-брутто
5	Приймання через ТЗД	Введення необхідних даних	Прийнята палета з'являється в системі

Продовження таблиці 1 Основні сценарії тестування

6	Рахунок-фактура	Перевірка автоматичного формування після приймання	Документ сформовано автоматично
7	Поселення товару	Автоматичне розміщення в комірку згідно температури	Місце зберігання оновлено
8	Створення клієнта\ замовлення клієнта	Створення нового клієнта та замовлення на відвантаження	Дані збережено, статус - «Створено»
9	Обробка замовлення\ генерація лотів	Переведення замовлення у роботу, генерація лотів	Лоти згенеровано, кожен має унікальний номер
10	Комплектування через ТЗД	Відбір продукції з місця відбору, підтвердження кількості	Статус лоту оновлено на «Зібрано»
11	Відвантаження замовлення	Формування видаткової накладної після відбору, переміщення кількості на залишки клієнта	Документ сформовано, статус - «Завершено»
12	Переміщення	На місці відбору недостатньо товару – створюється переміщення із зберігання	Палета поповнена на місце відбору
13	Чат-бот	Запит інформації щодо залишків, замовлень, статусу	Відповідь (у режимі офлайн при

			недоступності API)

1. Реєстрація\авторизація.

Вносимо власні дані для реєстрації.

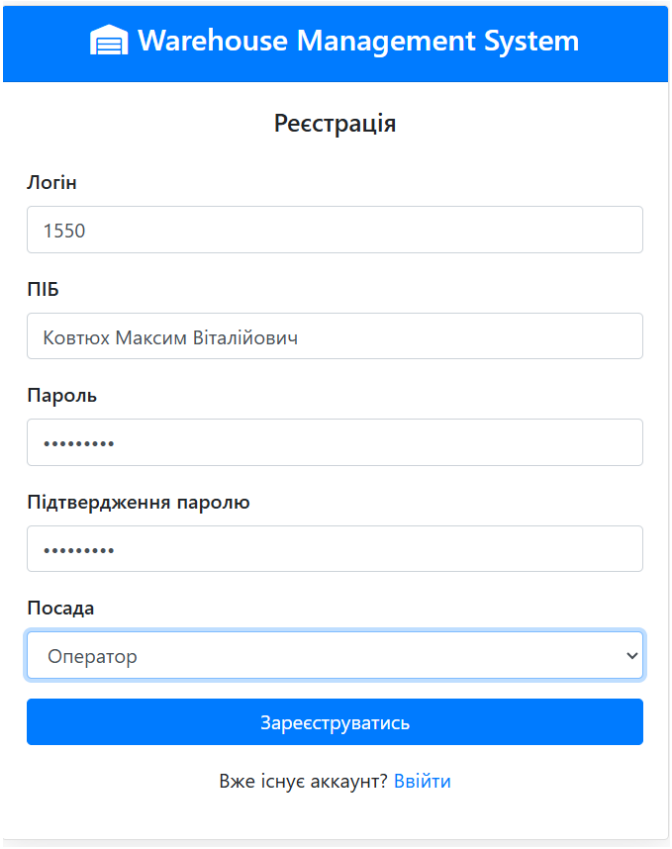

Рис. 1 Вікно реєстрації.

Перевіряємо чи можливо створити аккаунт з логіном, що вже існує. Отримуємо помилку, що такий логін вже існує.

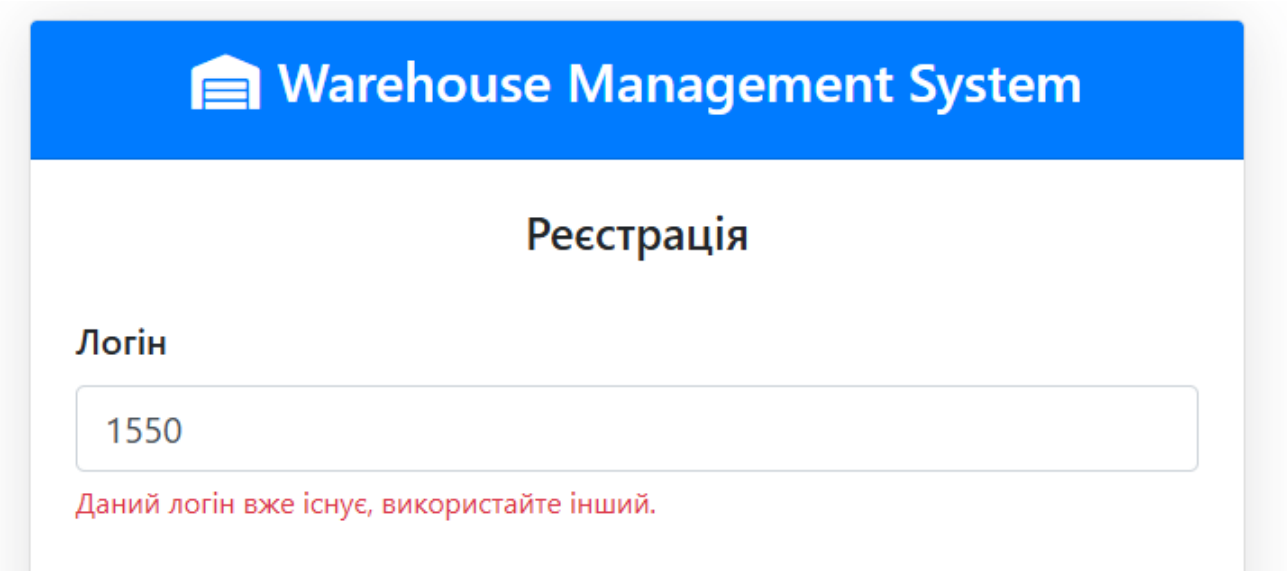

Рис. 2 Перевірка повторної реєстрації

Реєстрація успішна, отже заходимо до облікового запису.

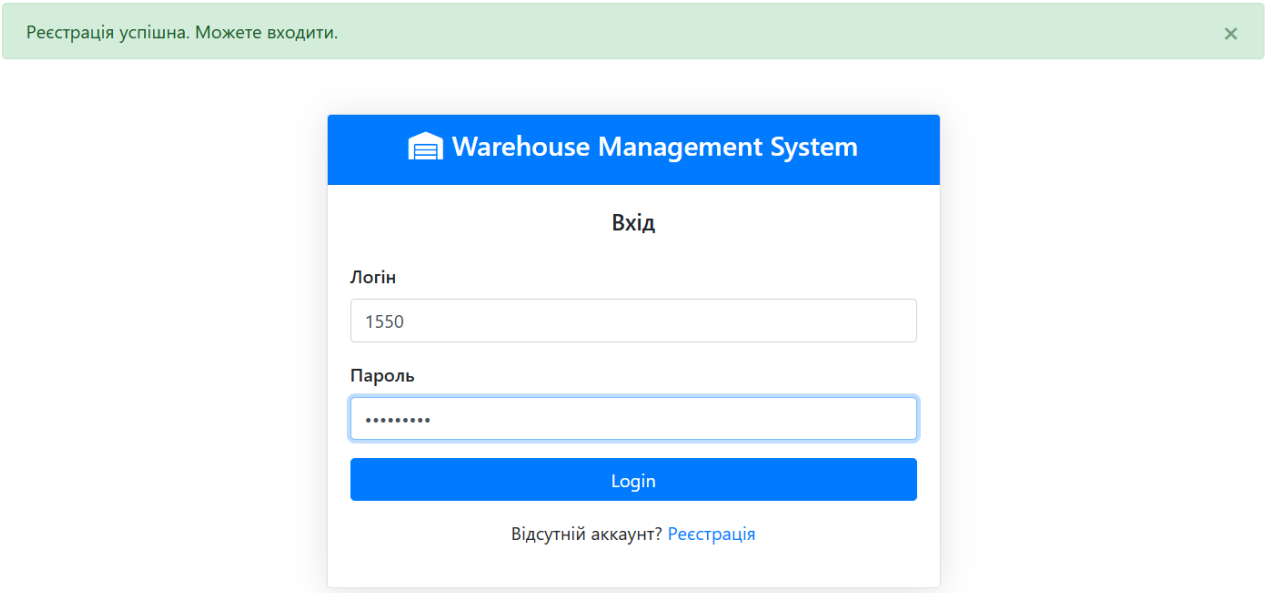

Рис 3. Вікно авторизації.

Ввійшли в обліковий запит, отже реєстрація\авторизація працює коректно.

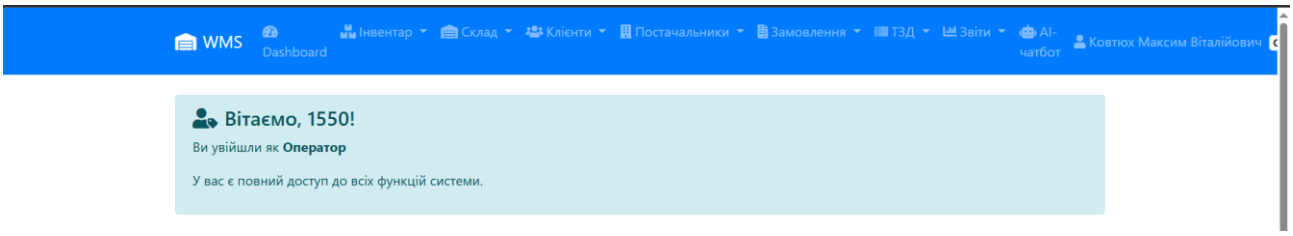

Рис. 4 Підтвердження входу в аккаунт

2. **Закріплення місця відбору.**

Переходимо у вікно «Склад» -> «Місця відбору». Та тиснемо на кнопку «Призначити нове місце комплектування».

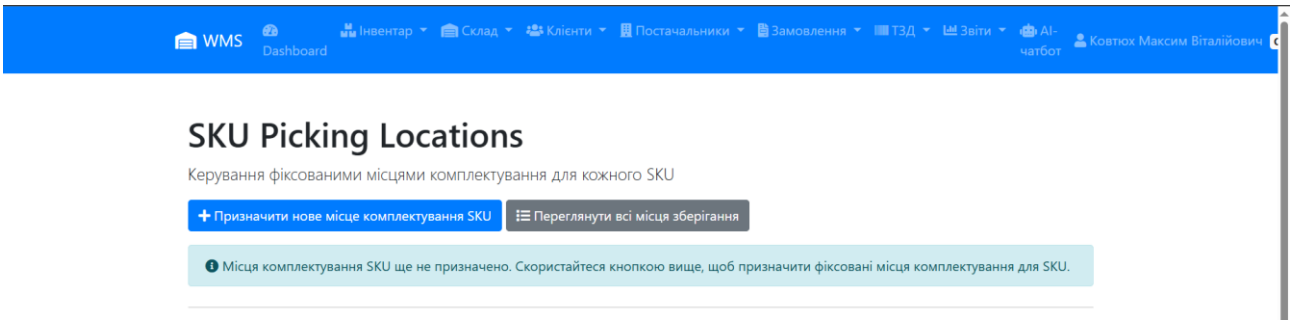

Рис. 5 Вікно місця відбору.

Додаємо місце відбору для двох артикулів.

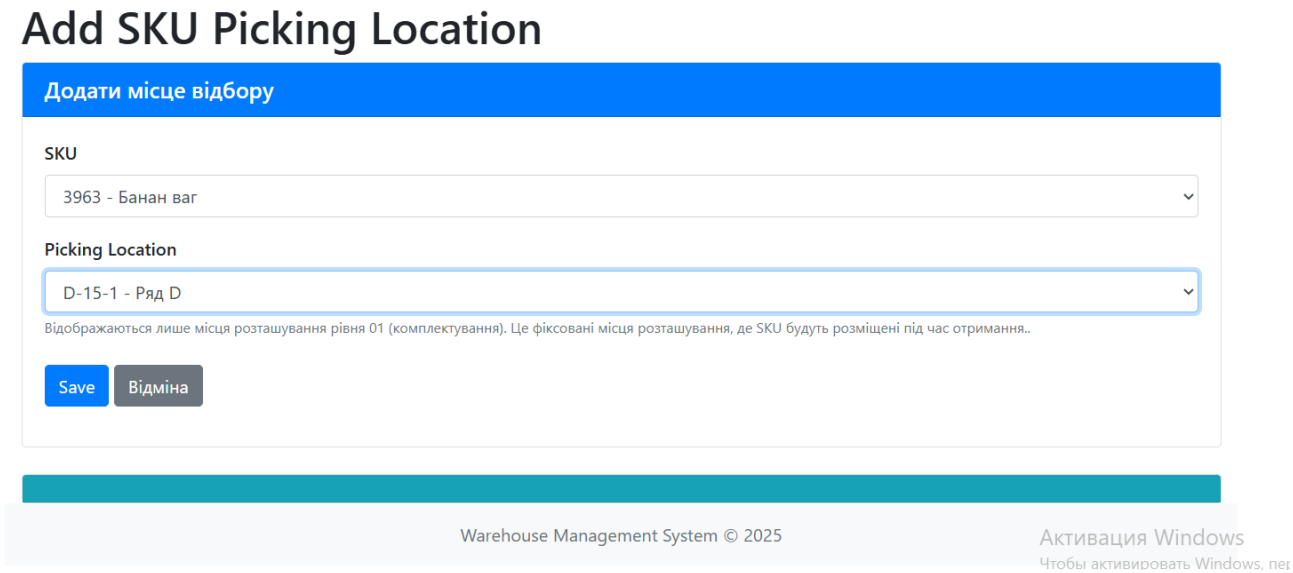

Рис. 6 Вікно додавання місця відбору.

Додавши місця відбору, бачимо що вони успішно збереглися.

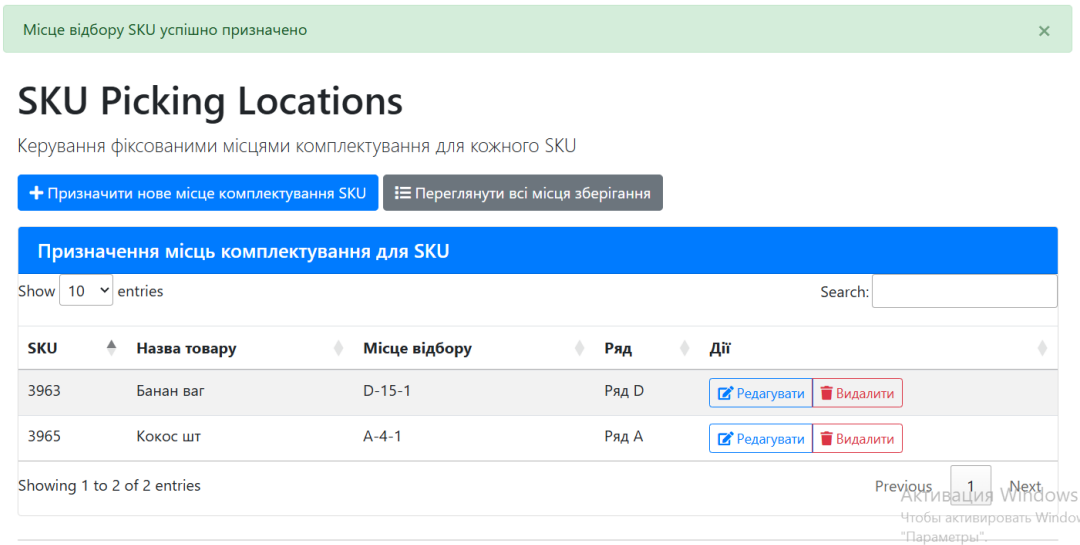

Рис. 7 Місця відбору збережено.

Якщо намагаємось назначити зайняте місце відбору іншому артикулу, отримуємо помилку.

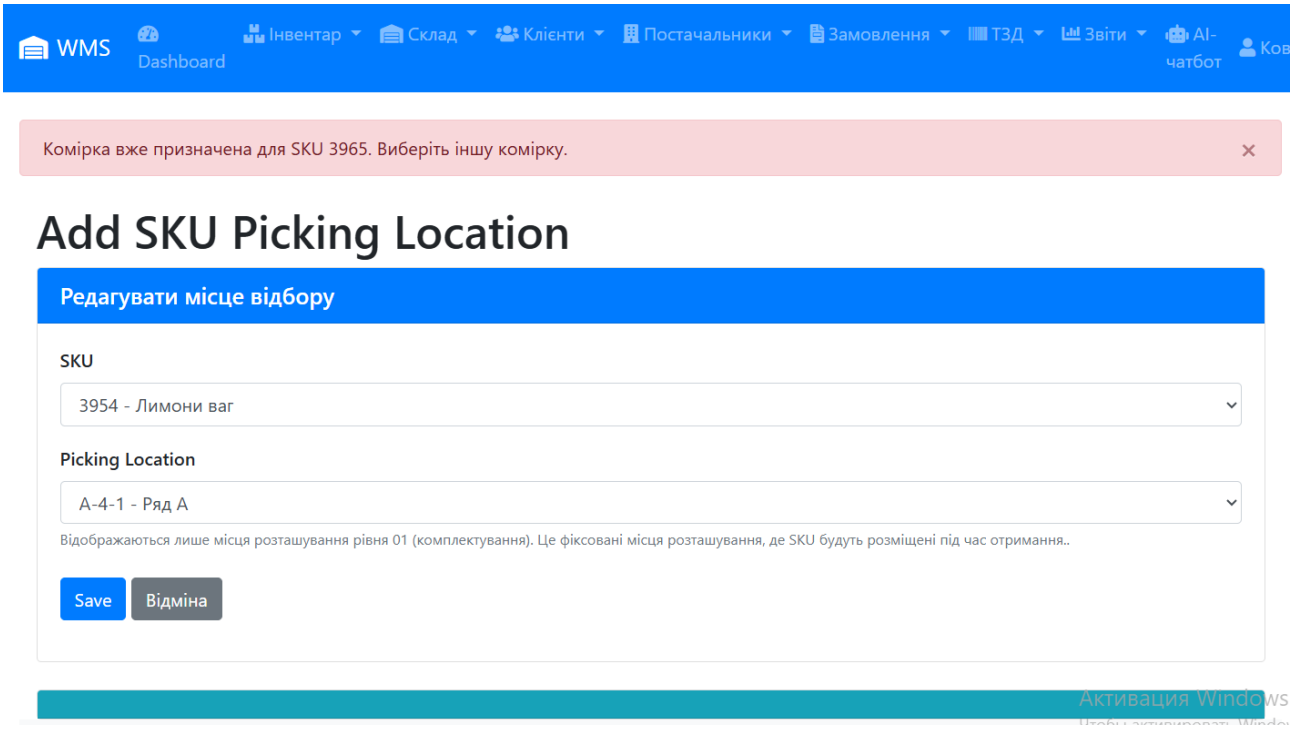

Рис. 8 Помилка при повторній підв’язці в одну комірку

3. Створення постачальника\контракта\замовлення на приймання.

Переходимо до вікна постачальники «Список постачальників».

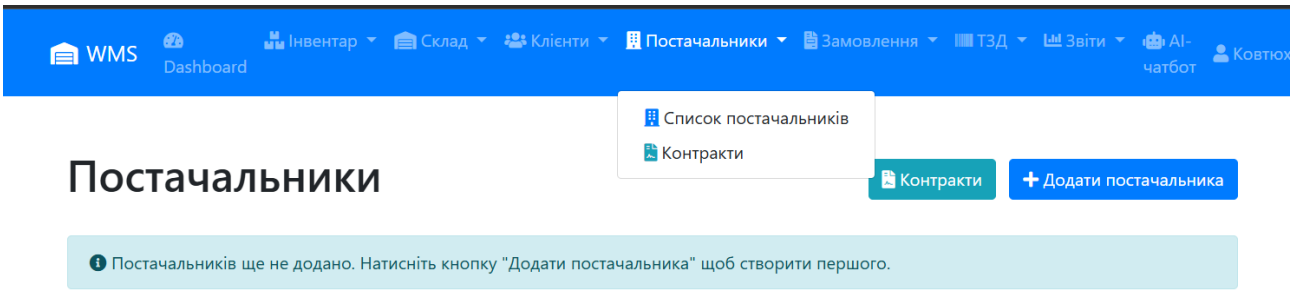

Рис. 9 Перелік постачальників.

Створюємо постачальника: вводимо необхідні дані та тиснемо «зберегти».

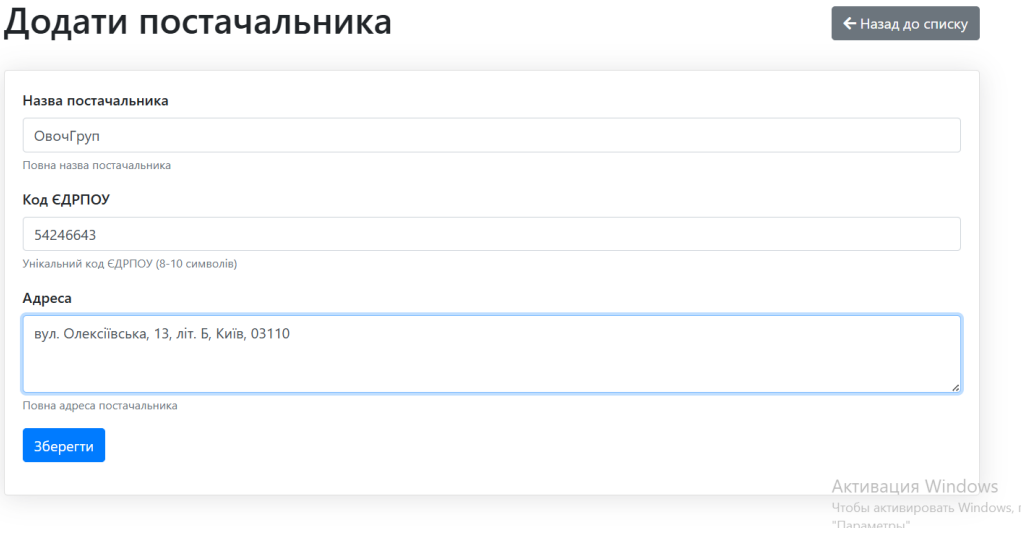

Рис. 10 Вікно створення постачальника

Бачимо, що постачальник додався до таблиці. Отже, створення відбулося.

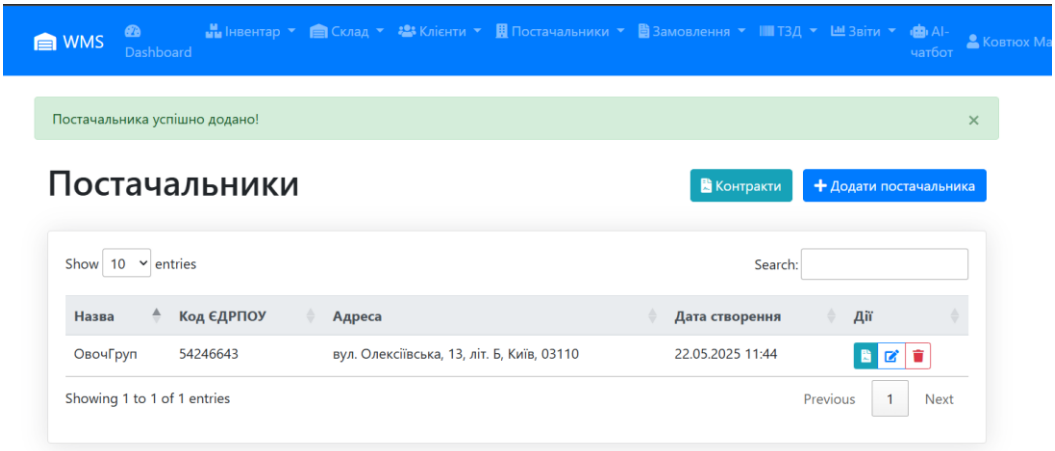

Рис 11. Таблиця постачальників

Тиснемо кнопку «Контракти» та переходимо до вікна з контрактами постачальників.

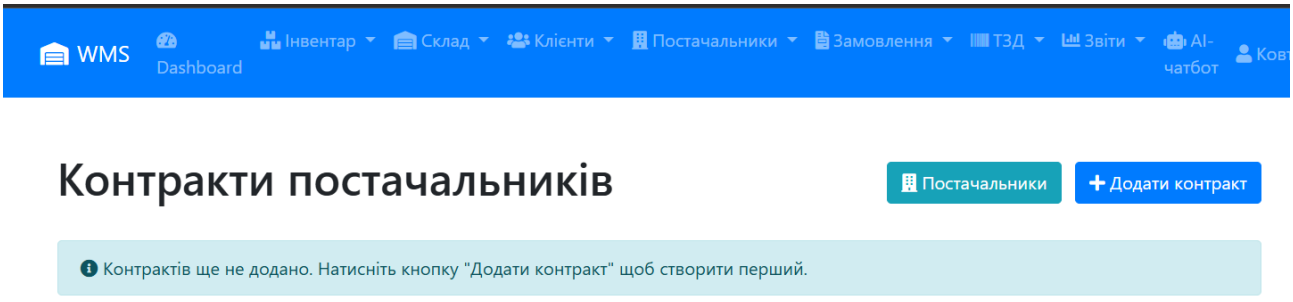

Рис. 12 Вікно контрактів постачальників

Ствоюємо контракт постачальника, згідно якого він має можливість возити певний товар. Внесли дані та тиснемо «Зберегти».

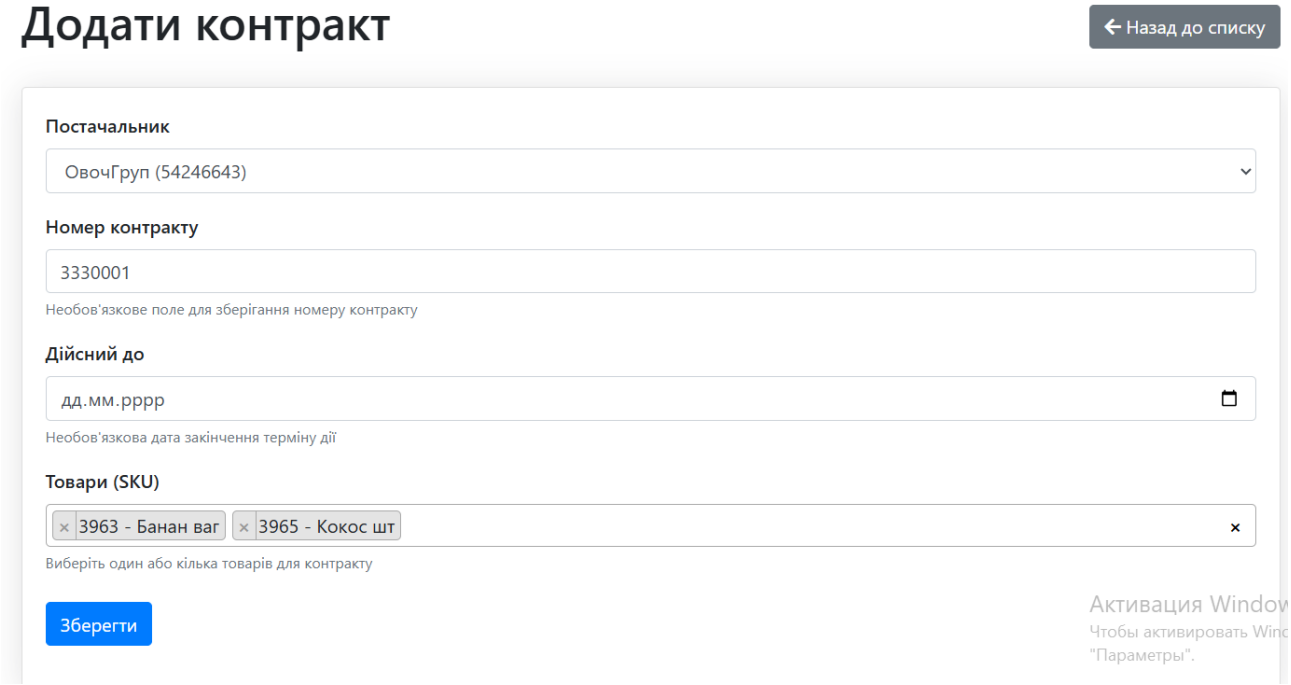

Рис. 13 Вікно створення контракту

Контракт додано до таблиці, отже успішно створено.

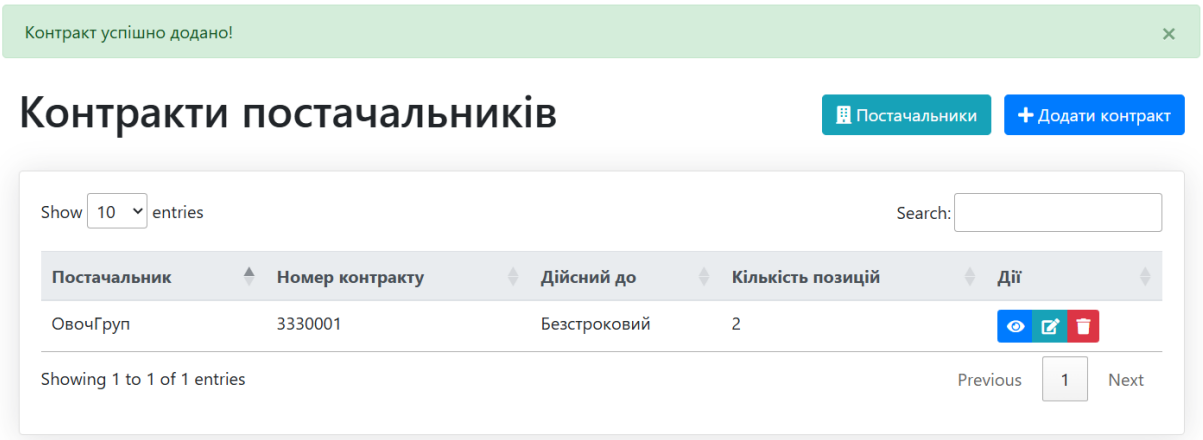

Рис. 14 Вікно контрактів.

Створюємо замовлення. Переходимо до «Замовлення» -> «Створити замовлення». Обираємо «Створити замовлення постачальнику.

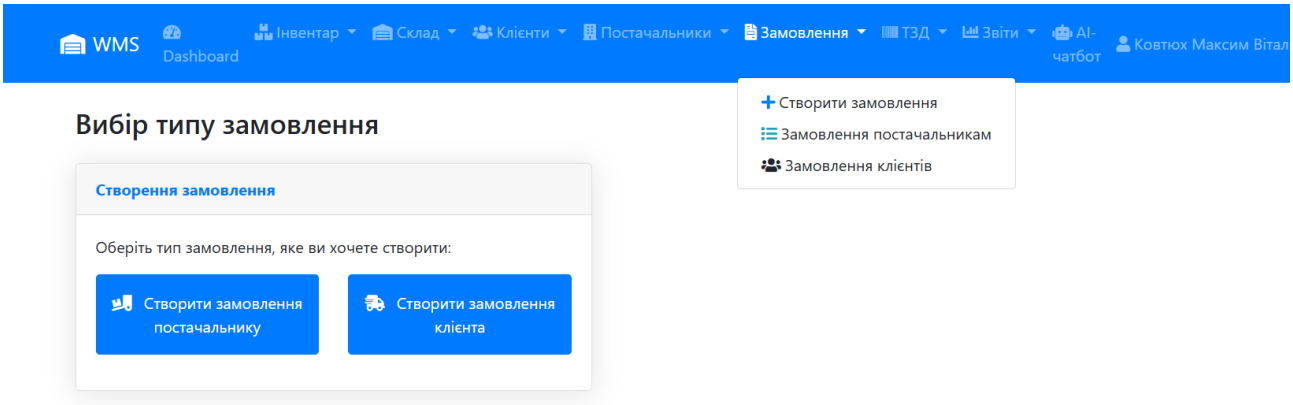

Рис. 15 Вікно вибору створення замовлення

Вписуємо код ЄДРПОУ постачальника, отримуємо можливість обрати номер контракту постачальника.

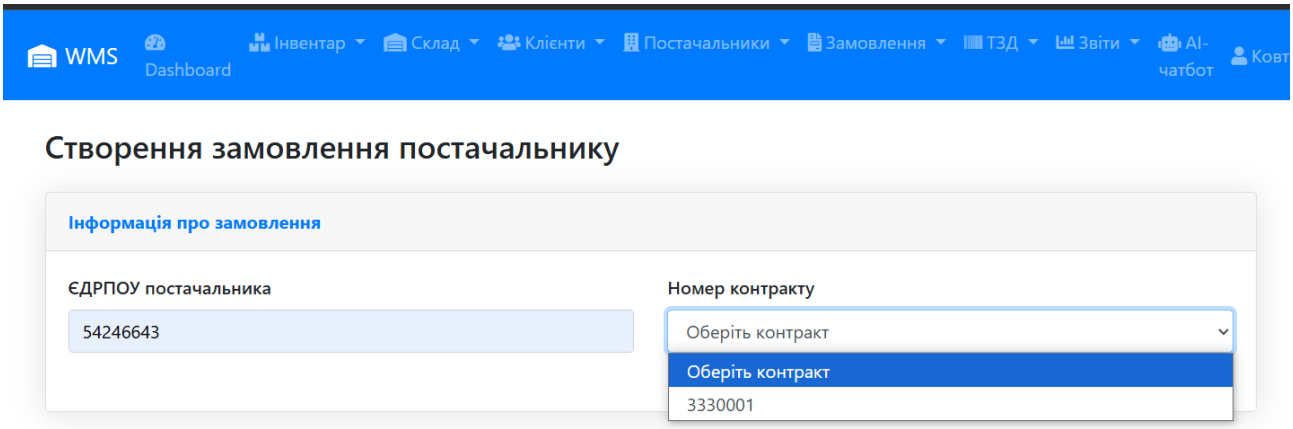

Рис. 16 Вікно вибору постачальника та контракту.

Обравши контракт постачальника, отримуємо можливість створити замовлення. Вносимо ціну одиниці товару, для вагового товару – ціна за 1кг, для штучного товару – ціна за 1шт.

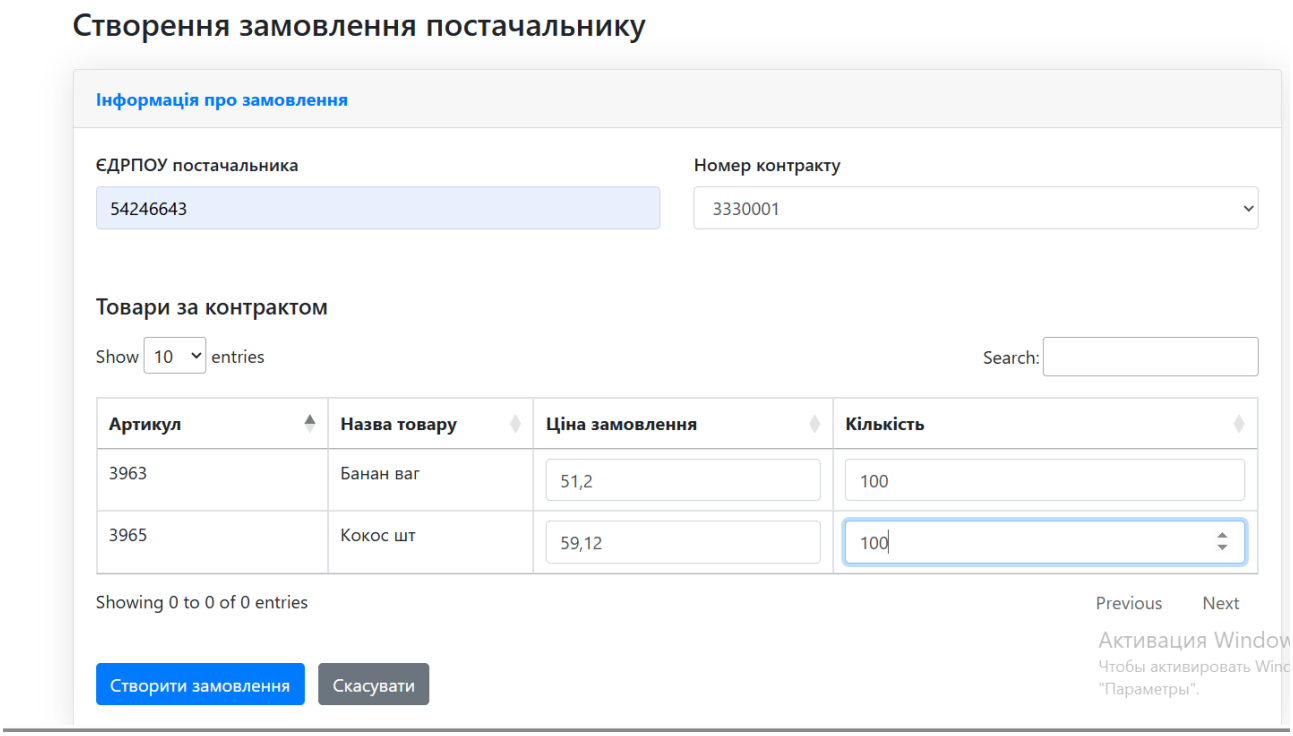

Рис. 17 Створення замовлення постачальнику.

Замовлення створено успішно. Статус «створено».

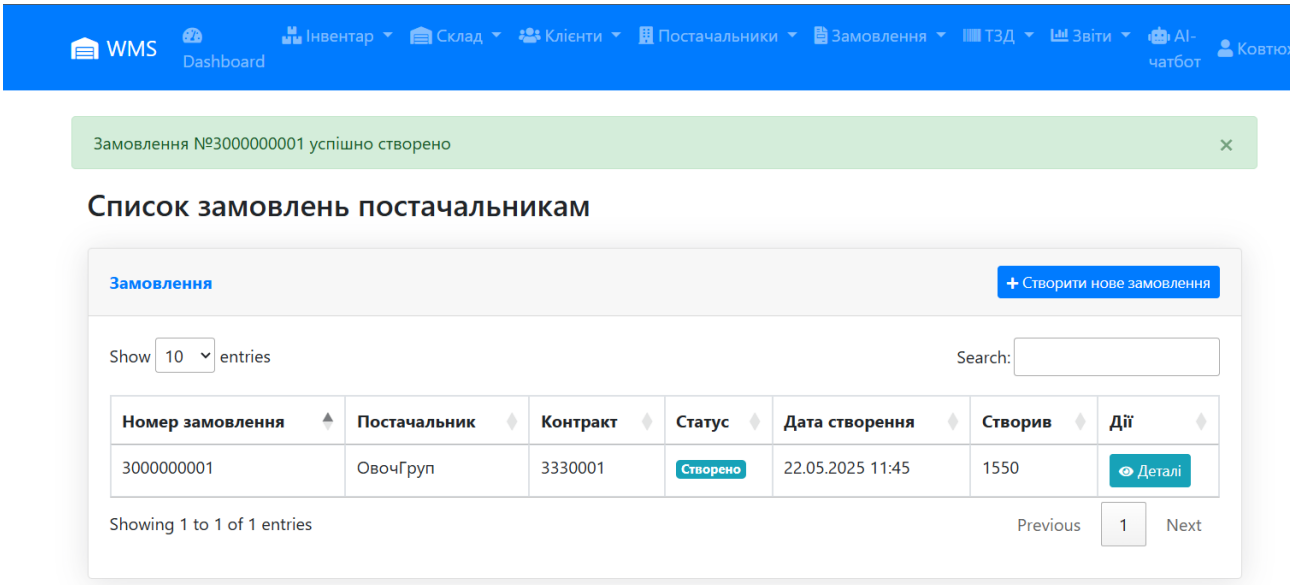

Рис. 18 Таблиця із замовленнями постачальників.

Натиснувши кнопку «Деталі», можемо переглянути дані про замовлення. Номер накладної (ПН) – номер згідно якого починаємо приймання товару.

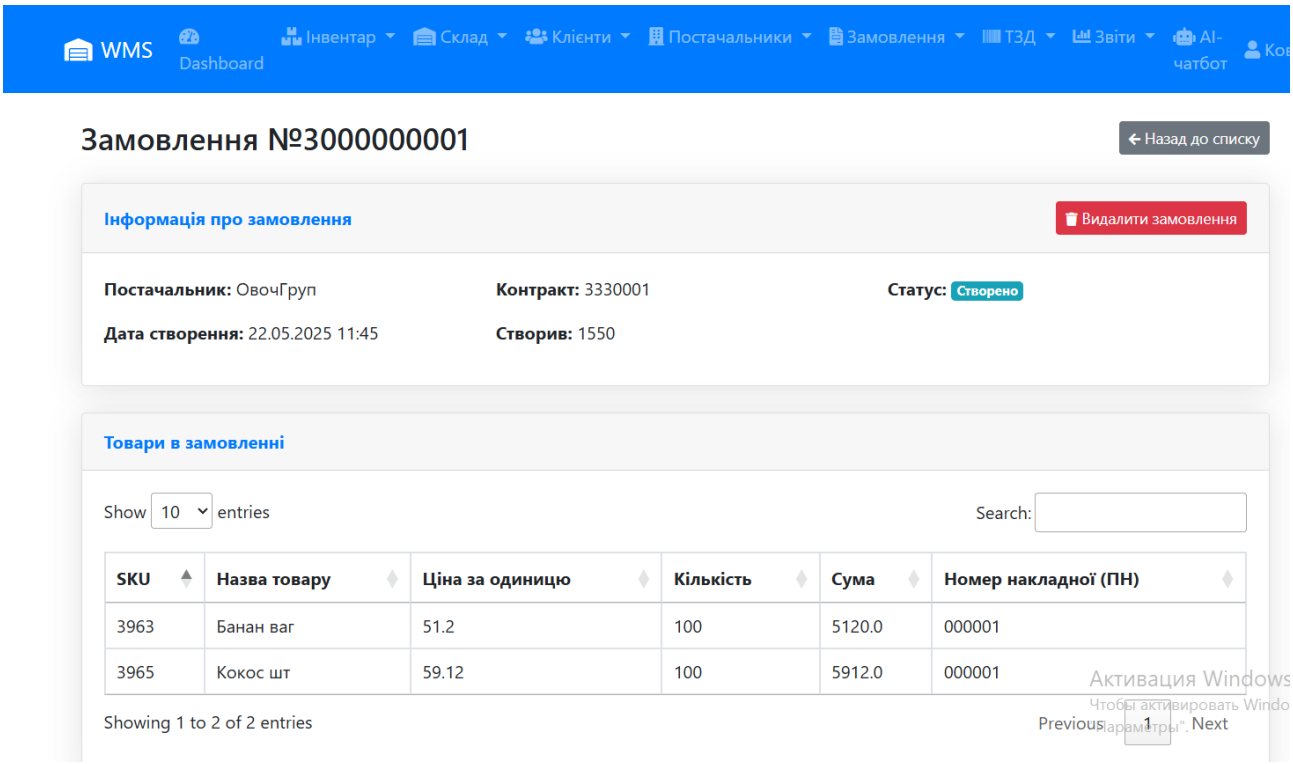

Рис. 19 Деталі створеного замовлення

4. Симуляція вагів\генерація штрих-кодів

Переходимо у вікно ТЗД -> «Симуляція вагів».

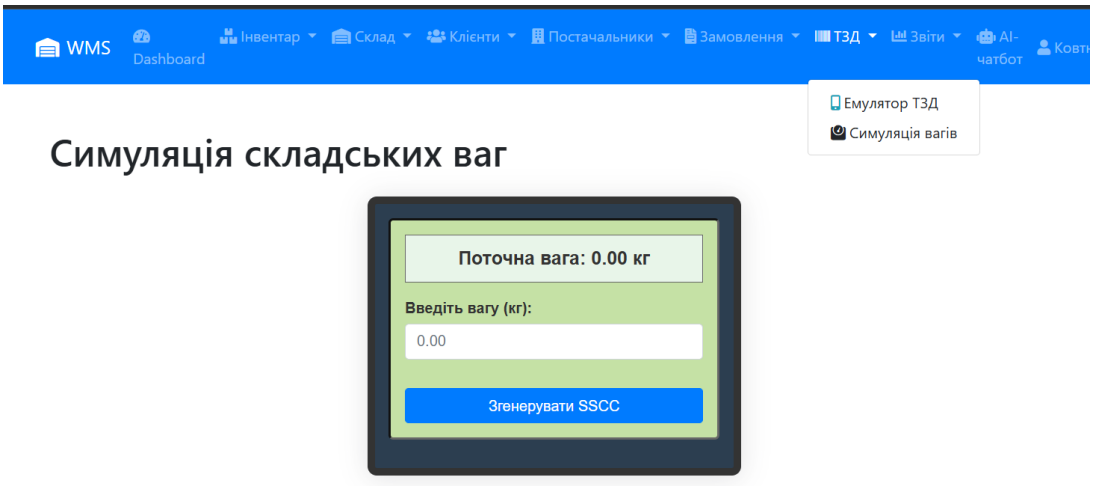

Рис. 20 Вікно симуляції складських вагів.

Вводимо вагу піддона, тиснемо кнопку «Згенерувати SSCC». Формується SSCC, штрих-код, та прив'язується вага брутто.

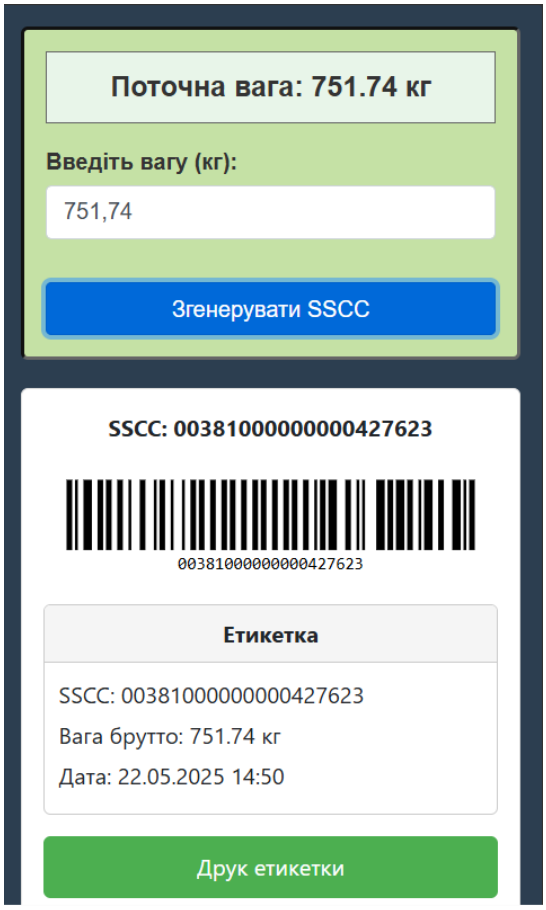

Рис. 21 Згенерований штрих-код\SSCC.

Також присутній шаблон друку стікеру.

Рис. 22 Шаблон стікеру.

5. Приймання через ТЗД

Переходимо у вікно «ТЗД» -> «Емулятор ТЗД». Бачимо інтерфейс ТЗД.

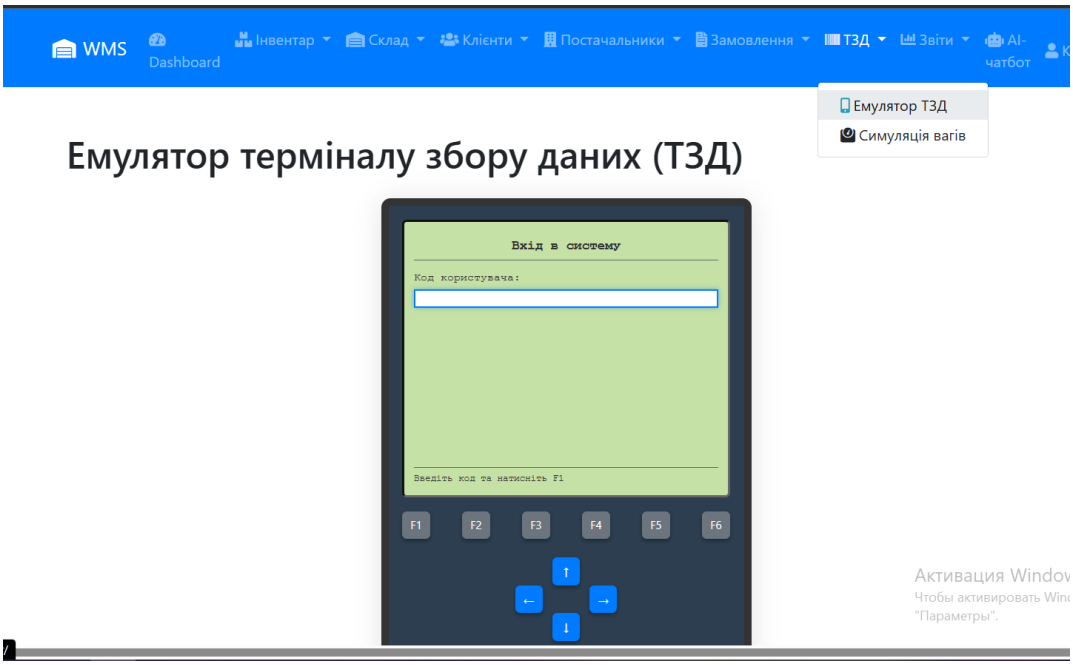

Рис. 23 Вікно входу «Емулятор ТЗД».

Якщо вводимо неіснуючий логін, бачимо помилку.

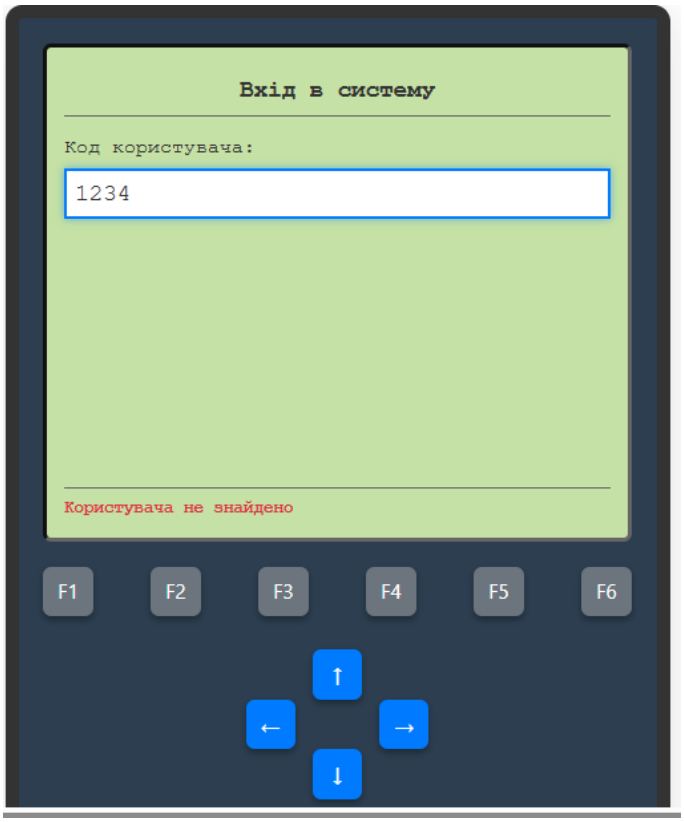

Рис. 24 Вікно входу «Емулятор ТЗД». Неіснуючий логін.

Вносимо існуючи логін, тиснемо F1, як результат заходимо у систему.

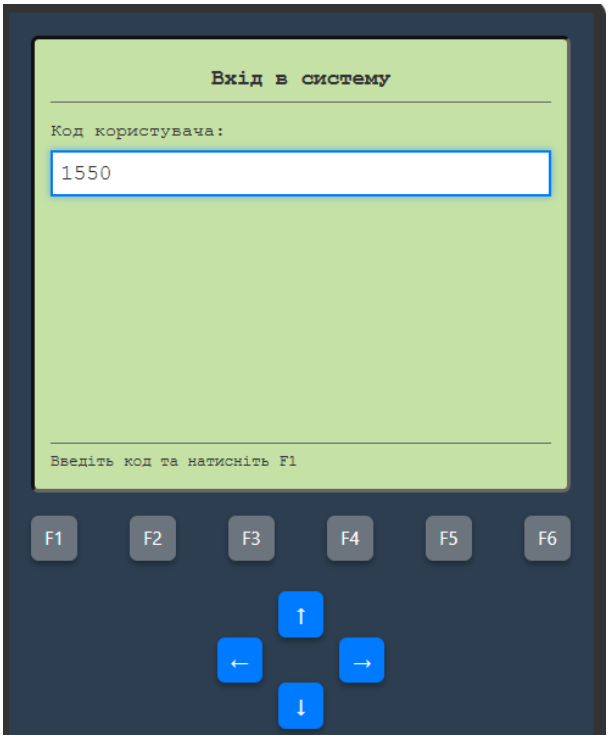

Рис. 25 Вікно входу «Емулятор ТЗД». Існуючий логін.

Бачимо меню вибору дій зі сканером. Обираємо «Прийом».

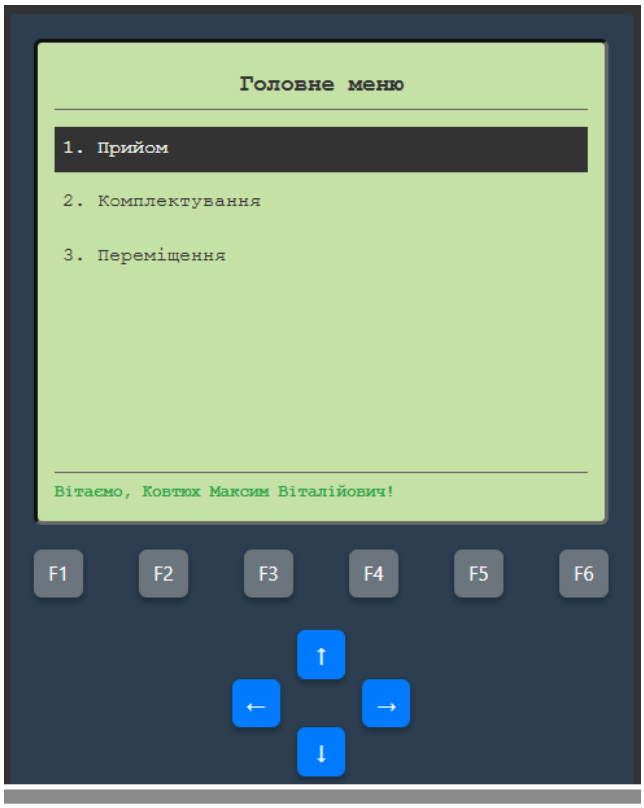

Рис. 26 «Емулятор ТЗД». Меню дій.

Відкриється вікно вводу даних. Вводимо необхідні дані та тиснемо F1.

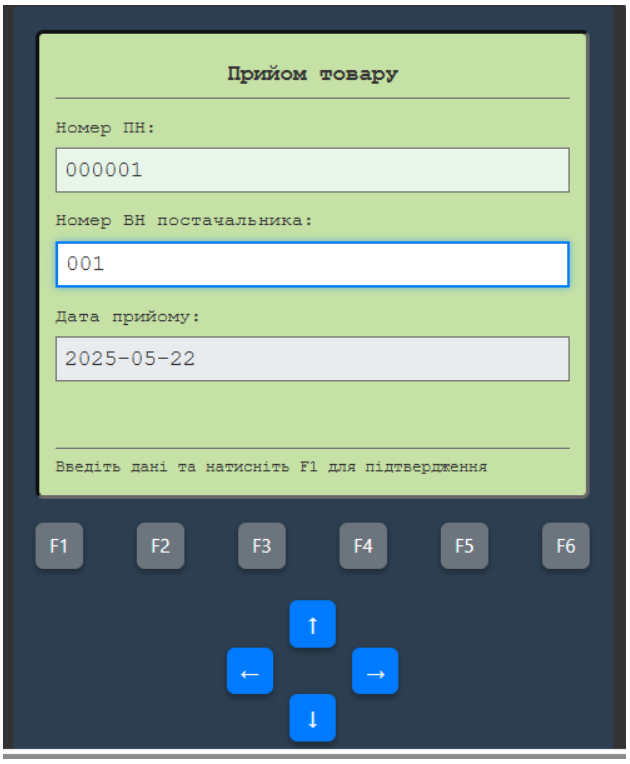

Рис. 27 Початкове вікно прийому.

Потрапляємо на вікно «Введення SSCC», вводимо SSCC, згенерований з симуляції вагів та тиснемо F1.

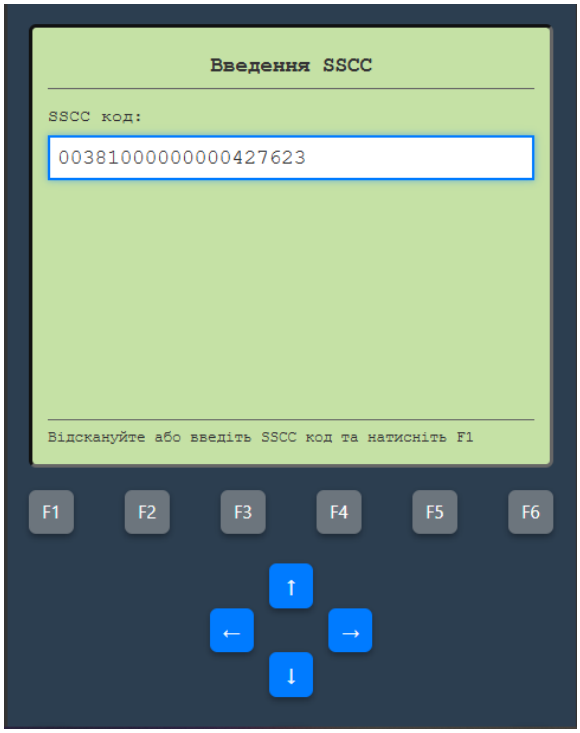

Рис. 28 Вікно «Введення SSCC».

Потрапляємо на вікно вибору артикулу, що буде прийнятий. Кнопкою F1 підтверджуємо вибір.

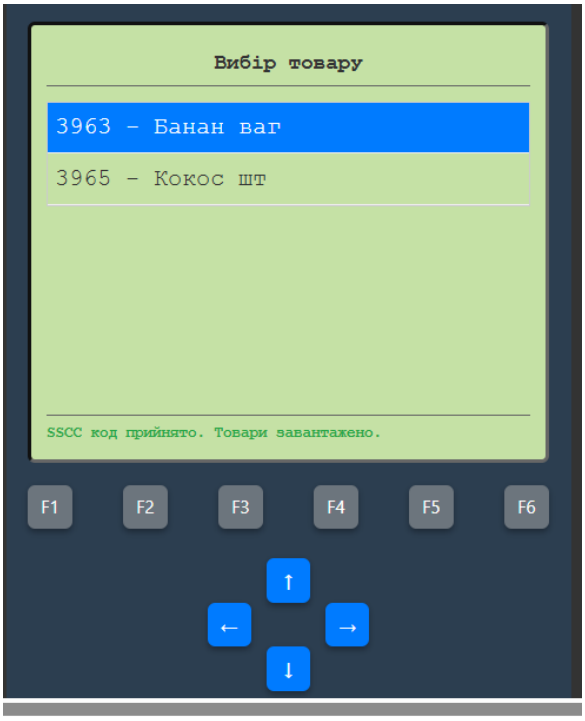

Рис. 29 Вікно «Вибір товару».

Обравши артикул, переходимо до вікна, де вводимо кількість ящиків на піддоні, вагу самого ящика та вагу піддона, для розрахунку ваги нетто. Та тиснемо F1, нас повертає на вікно «Введення SSCC». Вводимо наступний SSCC та продовжуємо приймання.

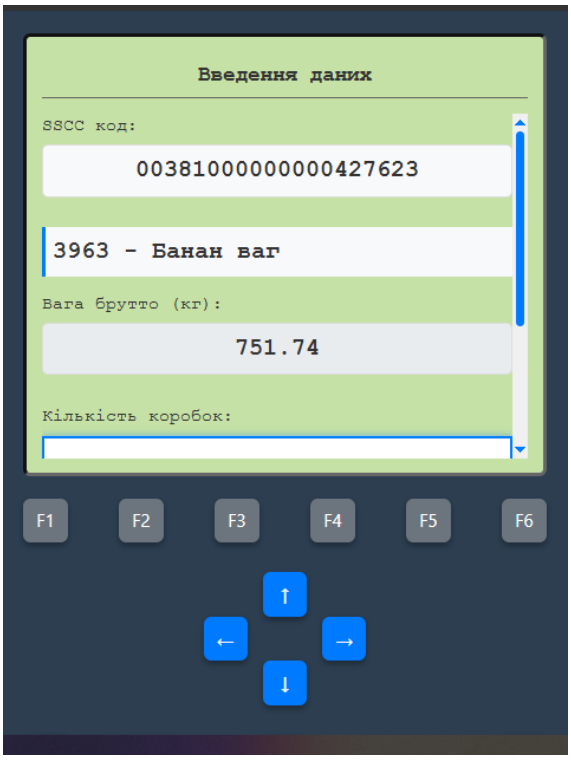

Рис. 30 Вікно «Введення даних»

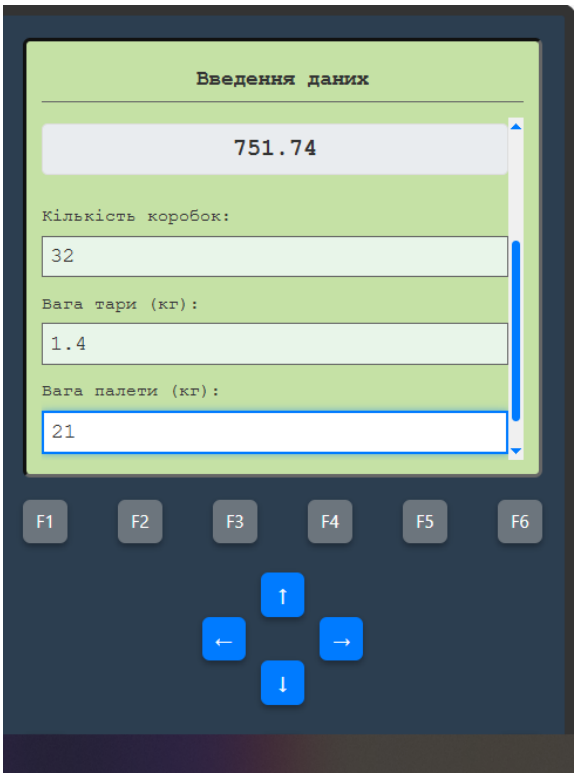

Рис. 31 Продовження вікна «Введення даних».

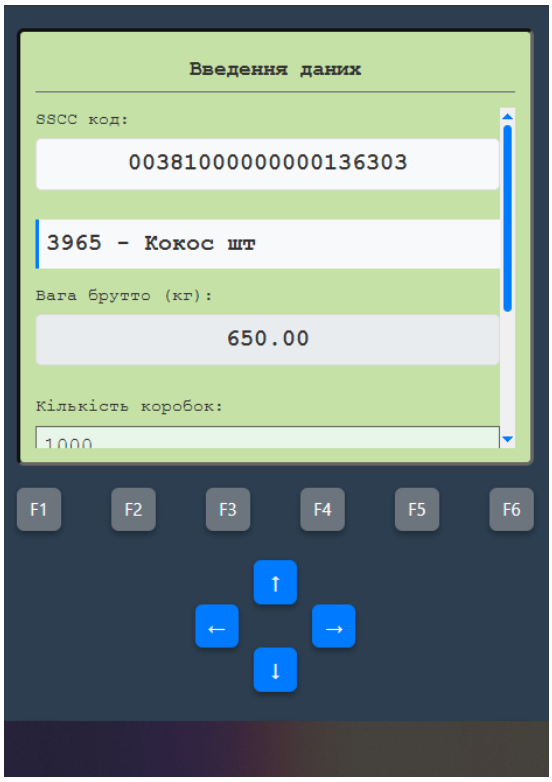

Рис. 32 Продовження вікна «Введення даних».

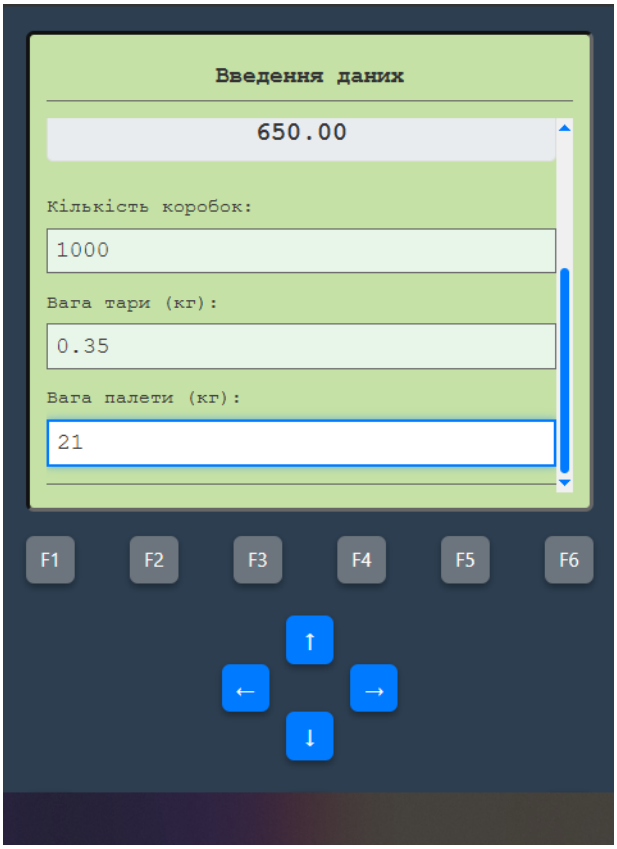

Рис. 33 Продовження вікна «Введення даних».

Закінчивши приймання, у вікні «Введення SSCC» тиснемо кнопку F3 підтвердження прийому. Та ще раз F3 для підтвердження, щоб уникнути випадкове натискання при прийманні.

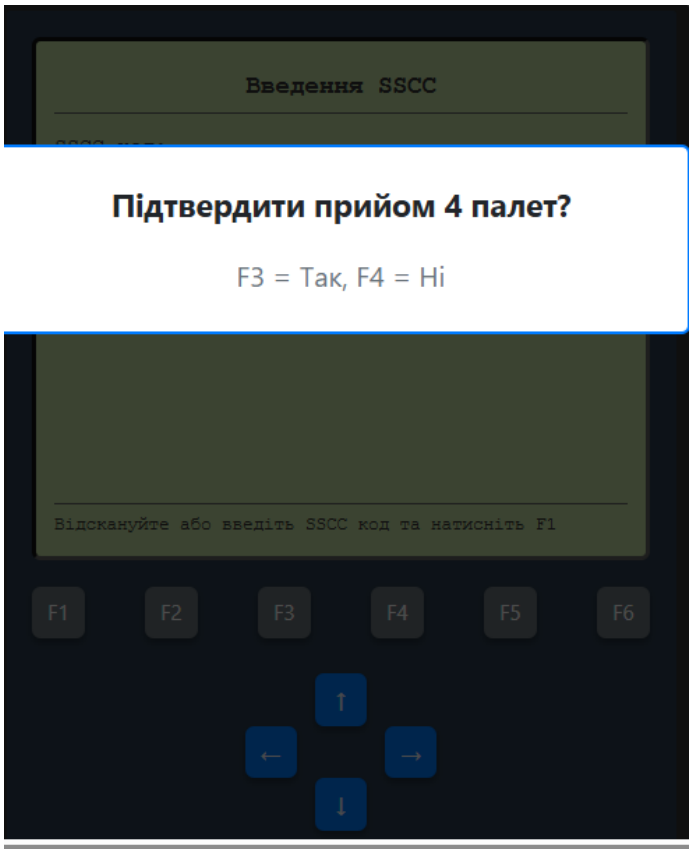

Рис. 34 Завершення приймання.

6. Рахунок-фактура

Закінчивши приймання, створюється рахунок-фактура, згідно якого проходить оплата постачальникам за доставлений товар. У списку замовлень, тиснемо кнопку «Рахунок-фактура» необхідного замовлення.

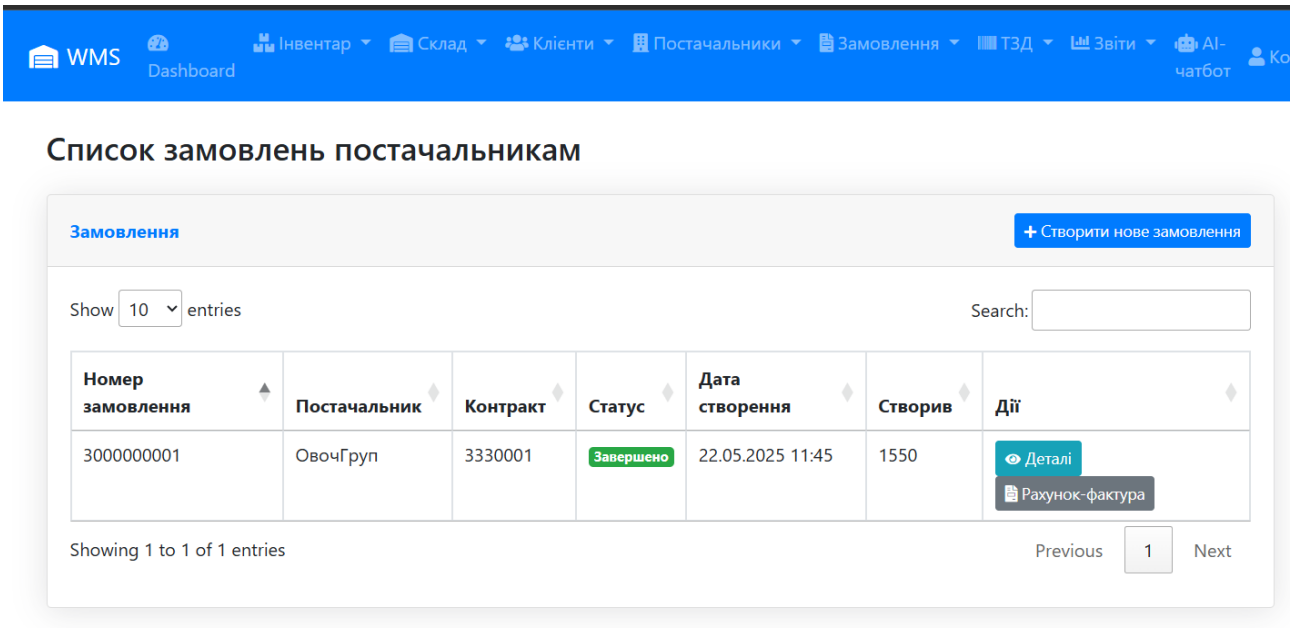

Рис. 35 Рахунок-фактуру згенеровано.

Бачимо всі необхідні дані та маємо можливіть надрукувати.

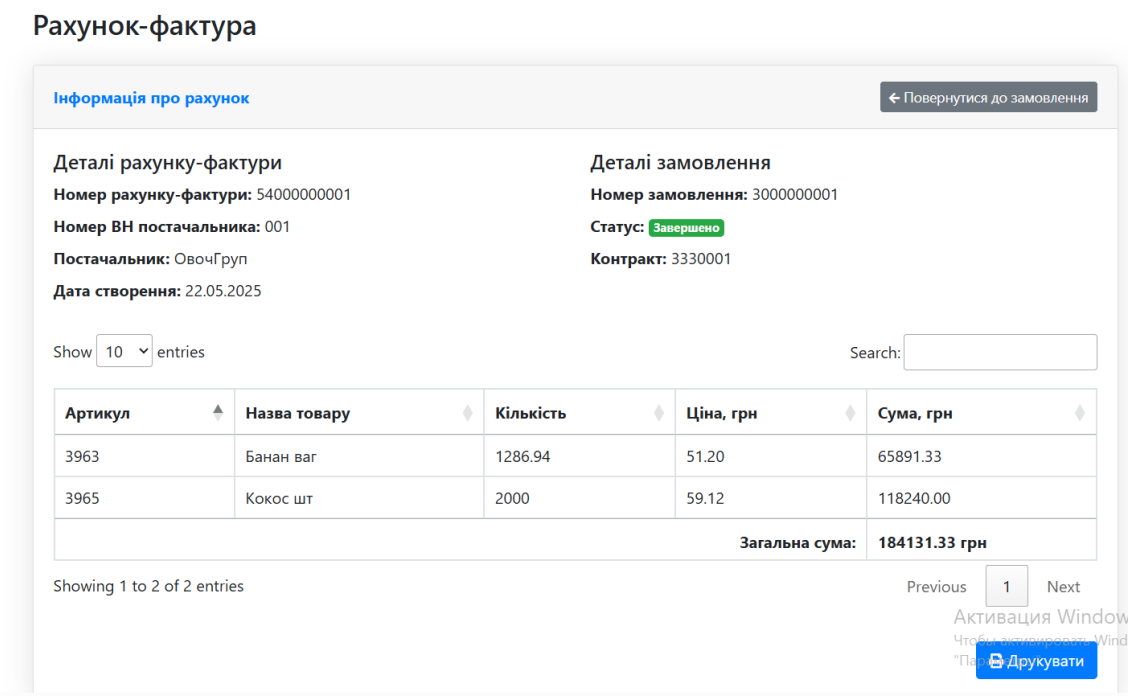

Рис. 36 Рахунок-фактура.

Тиснемо кнопку «Друкувати» та переходимо на вікно друку.

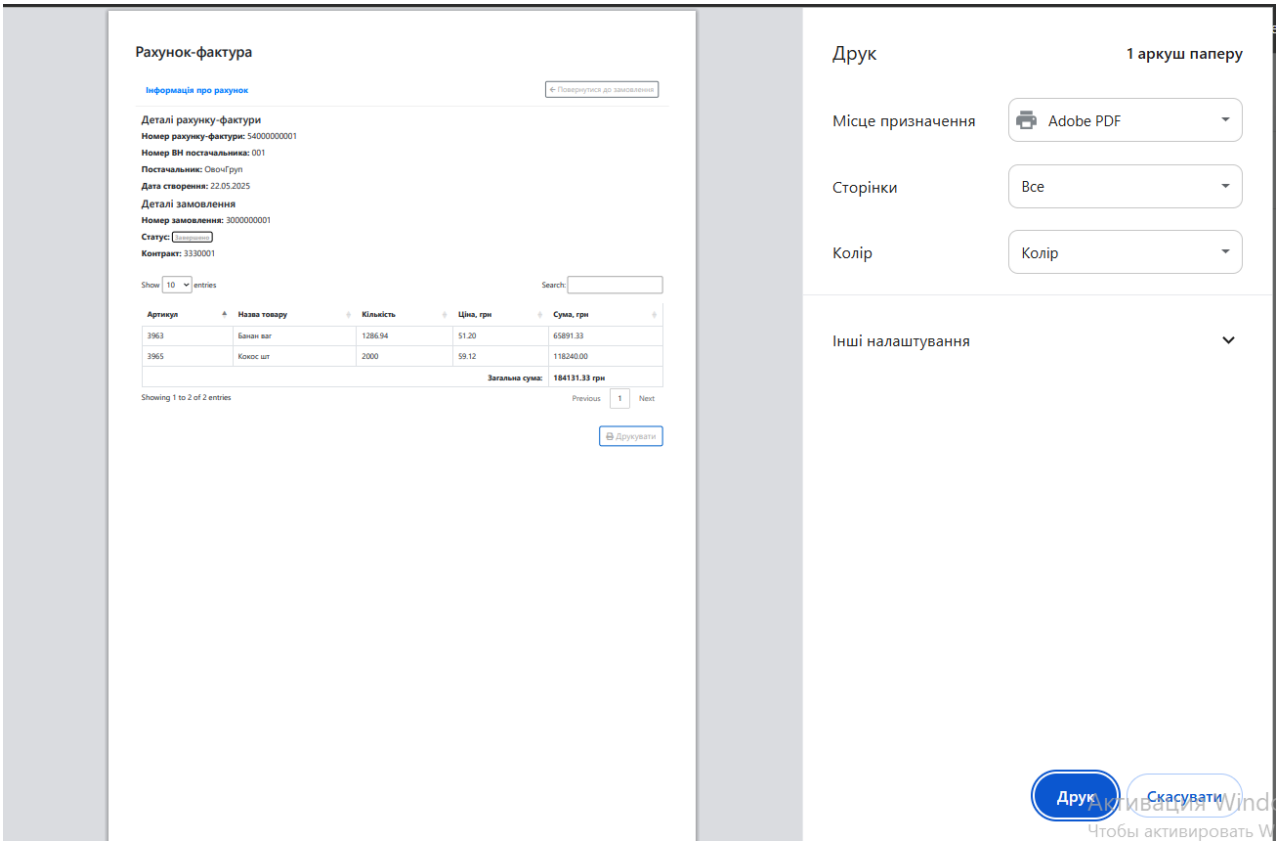

Рис. 37 Друк рахунку-фактури.

7. Поселення товару

Підтвердивши приймання, товар поміщається спочатку у місце відбору, потім на місця зберігання.

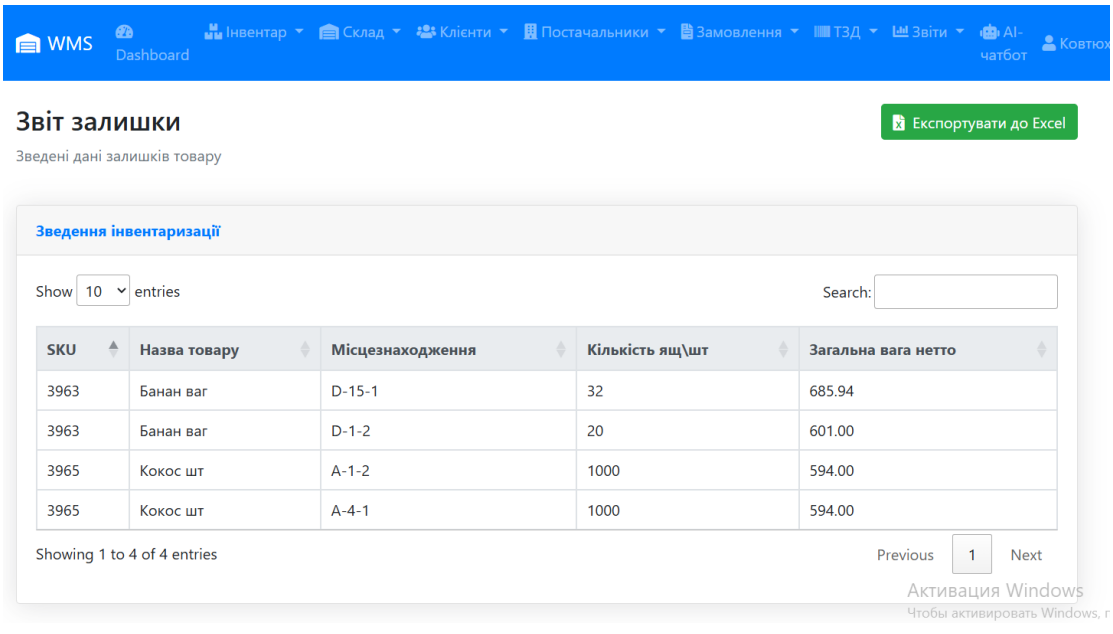

Рис. 38 Поселення товару.

8. Створення клієнта\замовлення клієнта

Переходимо до вкладки створення замовлення (Рис. 15), та обираємо «Створити замовлення клієнту». Потрапляємо на вікно вибору клієнту, та вибору товарних позицій.

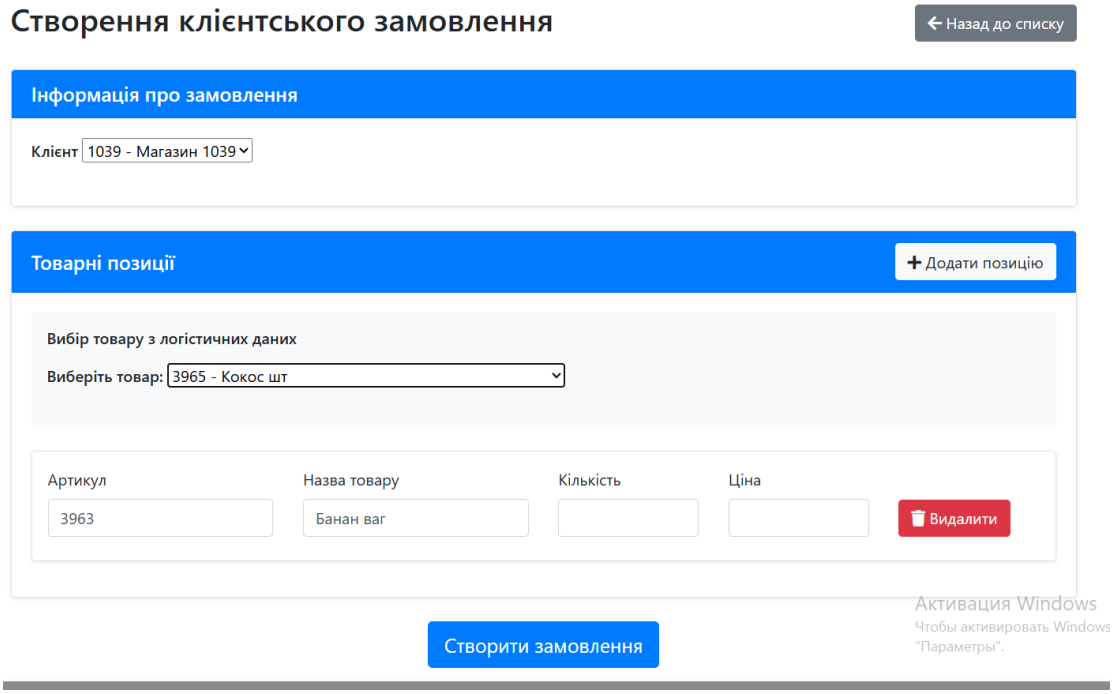

Рис. 39 Вікно введення даних замовлення клієнту.

Можемо вносити артикула як вручну, так і вибрати зі списку.

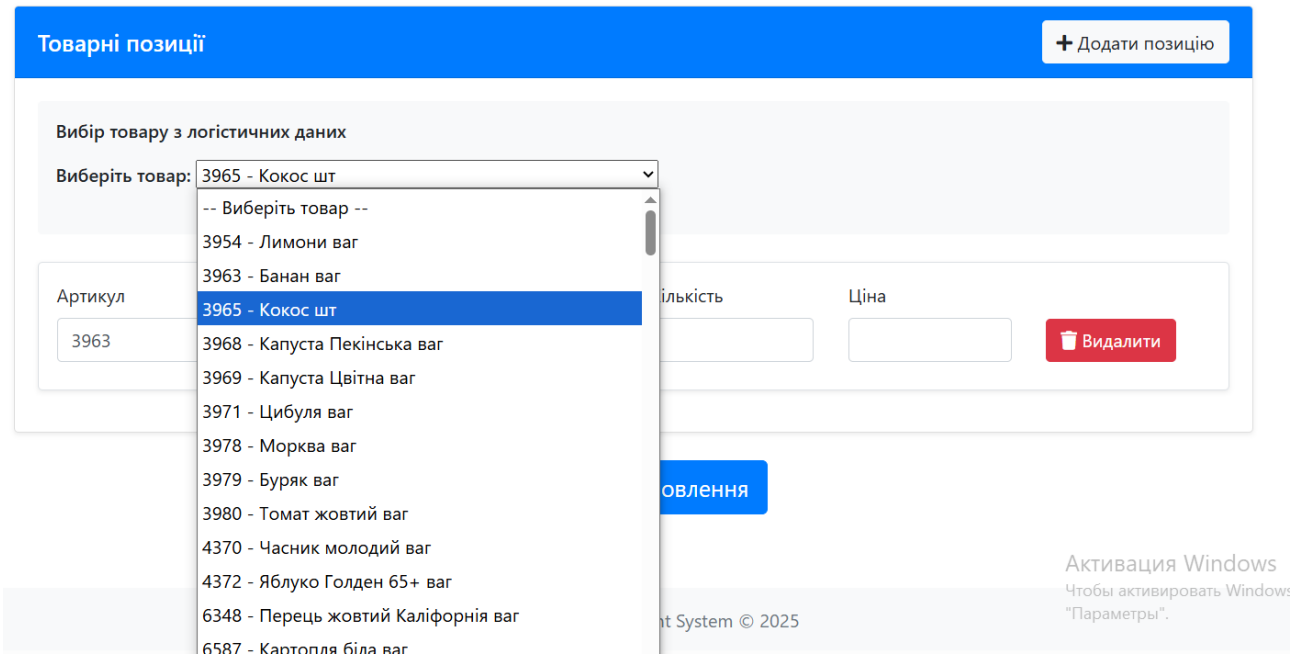

Рис. 40. Вибір артикулів.

Внісши дані, тиснемо кнопку «Створити замовлення». Для вагового товару, замовлення формуються у кілограмах, для штучного - у штуках.

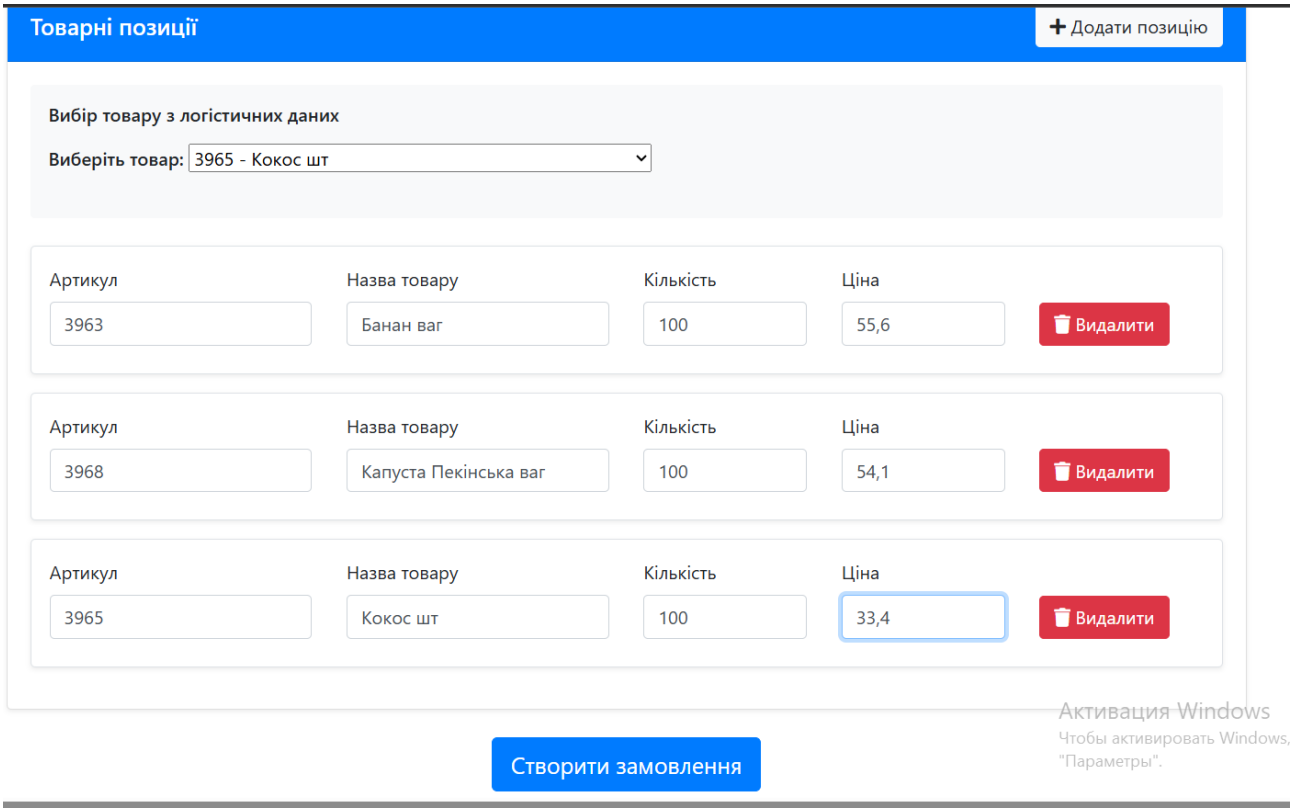

Рис. 41 Створюємо замовлення.

Замовлення створено, однак наразі статус «Створено», для відбору необхідно, ще обробити замовлення. Тиснемо кнопку «Деталі».

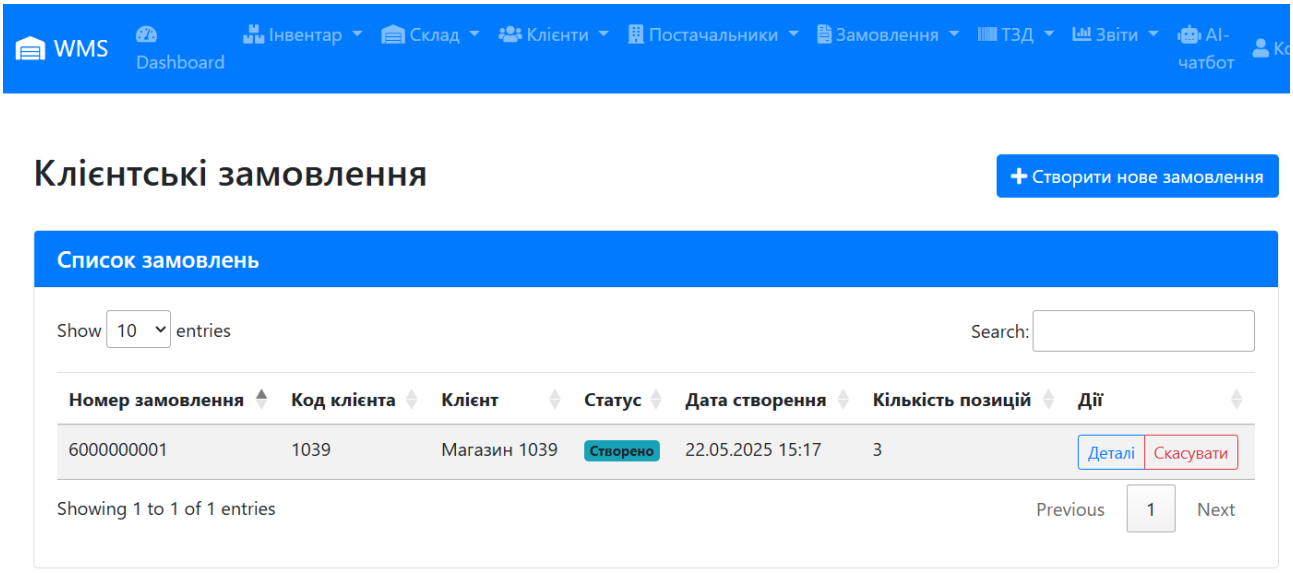

Рис. 42 Замовлення створено.

9. Обробка замовлення\генерація лотів

У деталях замовлення можемо ще раз переглянути артикула та кількість. Тиснемо кнопку «Обробити замовлення».

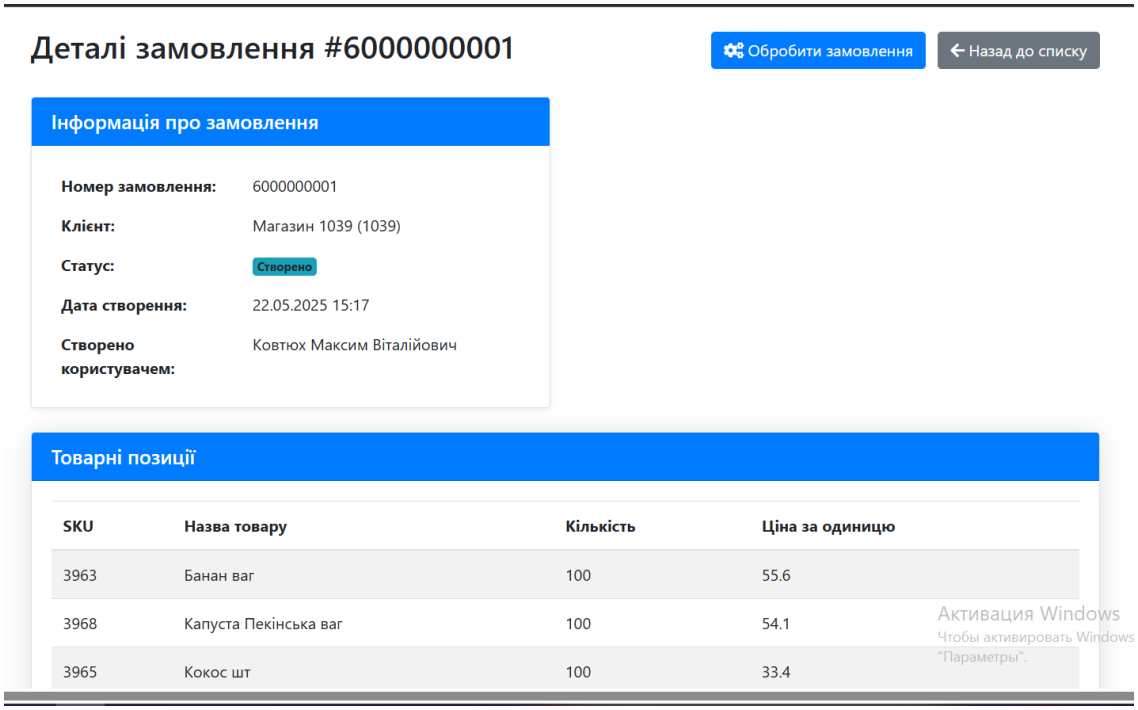

Рис. 43 Деталі замовлення.

Обробивши замовлення, воно змінює статус.

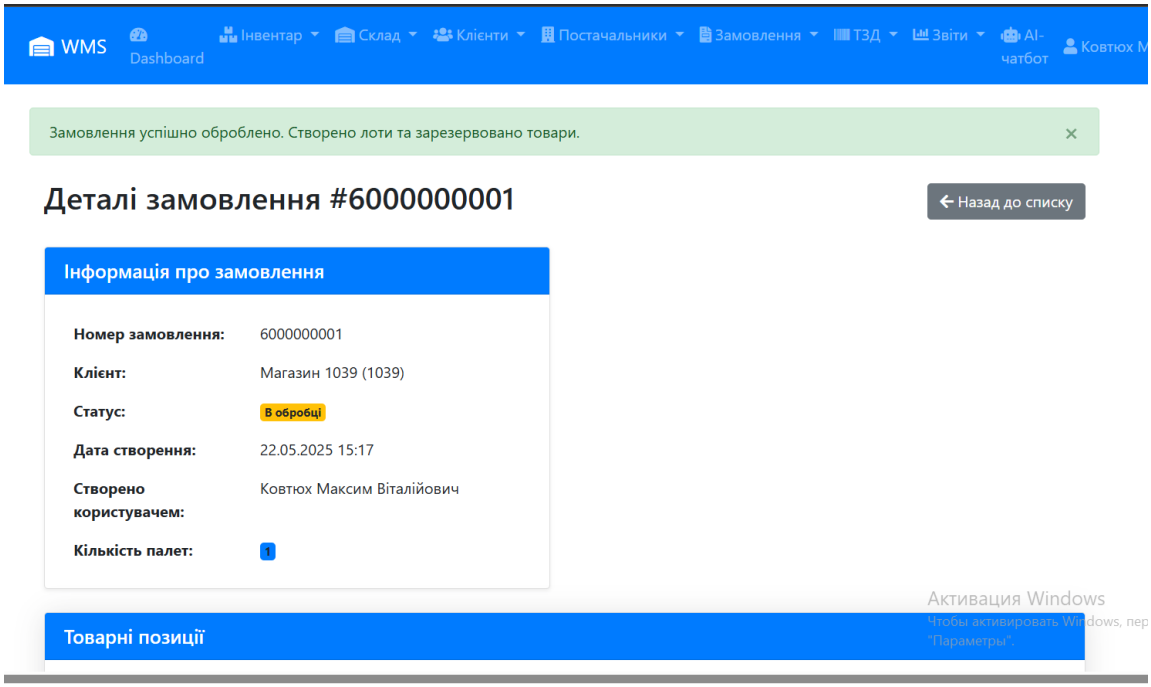

Рис. 44 Зміна статусу замовлення.

Лоти, що будуть відбиратись формуються, згідно об’єму. Кількість вагового товару, що замовлене стає кількістю ящиків, що необхідно відібрати.

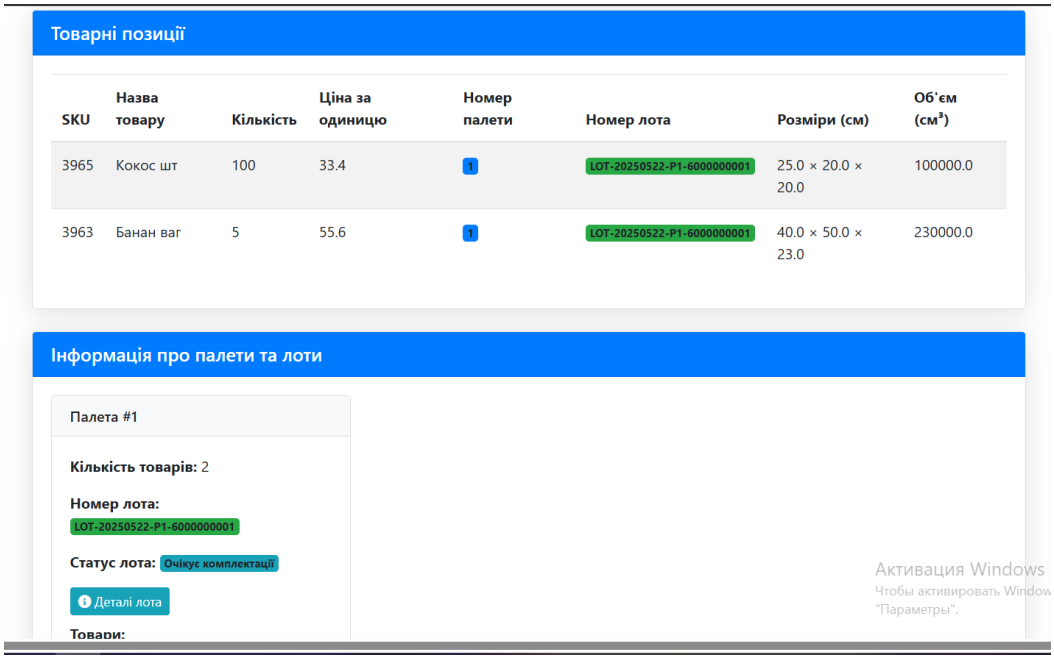

Рис. 45 Лоти сформовано.

10.Комплектування через ТЗД

Заходимо у «Емулятор ТЗД», вводимо логін. Та на головному меню обираємо комплектування.

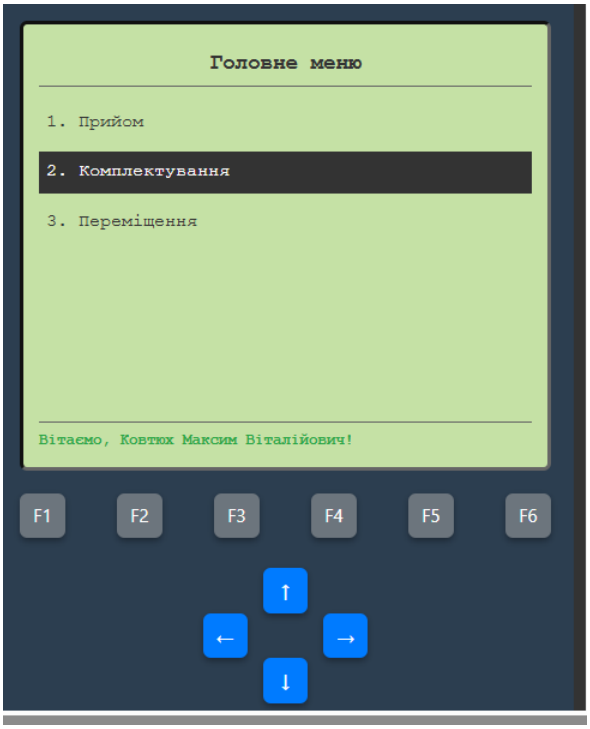

Рис. 46 Головне меню ТЗД.

Якщо лот один, автоматично заходимо у нього. Якщо лотів декілька, можливо обрати. У вікні введення кількості бачимо, артикул, назву товару, кількість необхідного товару необхідно та місце відбору, де даний товар стоїть. Вводимо кількість та тиснемо F1.

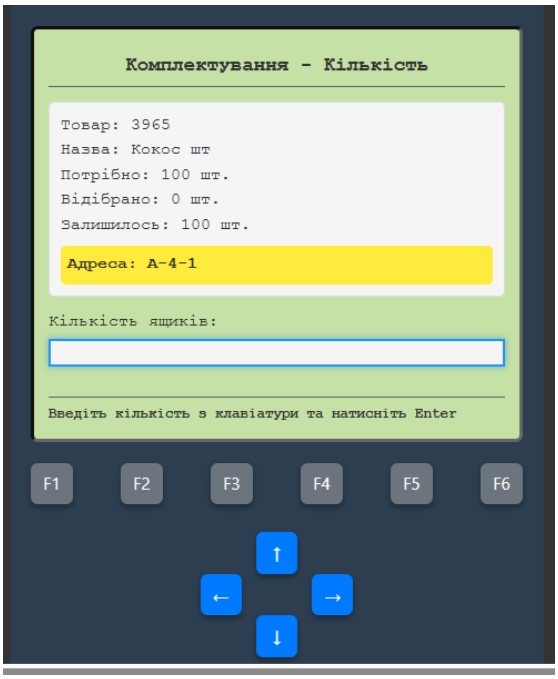

Рис. 47 Вікно відбору одного артикулу.

Отримуємо повідомлення, що товар успішно відібрано. Тиснемо F1 – переходимо до наступного

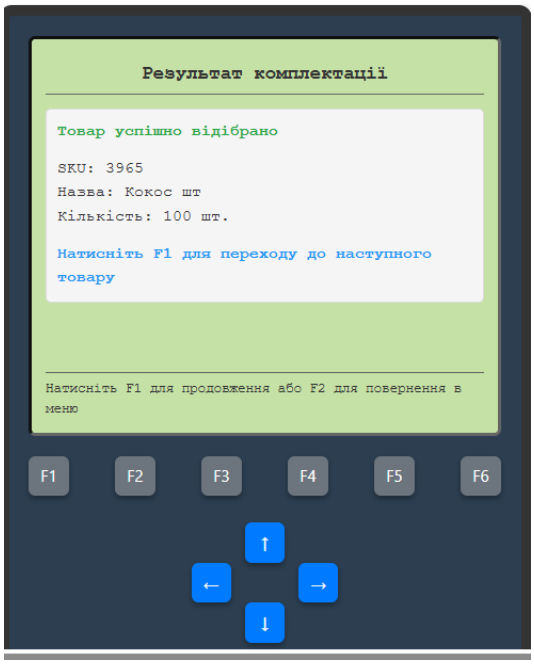

Рис. 48 Успішний відбір 1\2.

Якщо кількість не введена, чи введено більше, ніж потрібно – лот не закриється.

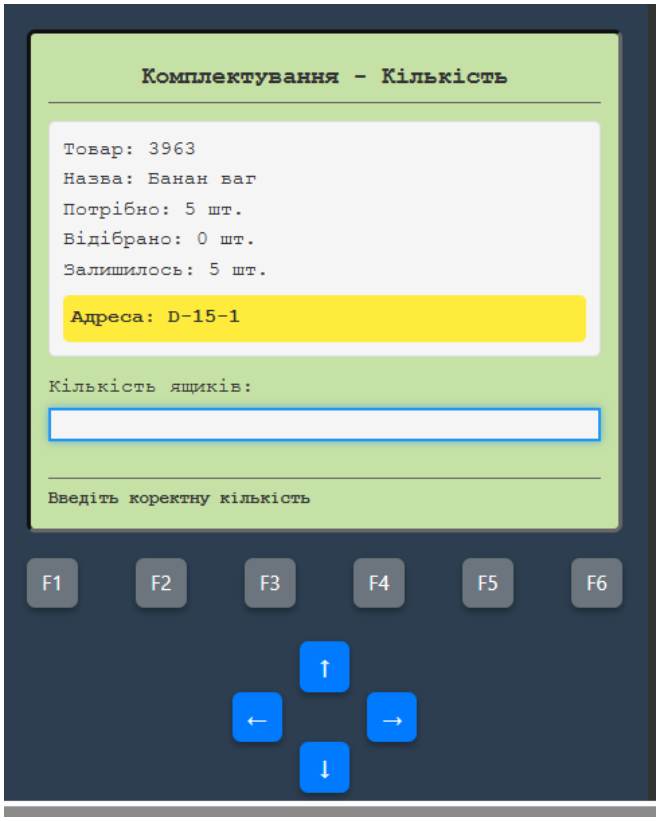

Рис. 49 Пусте поле кількості.

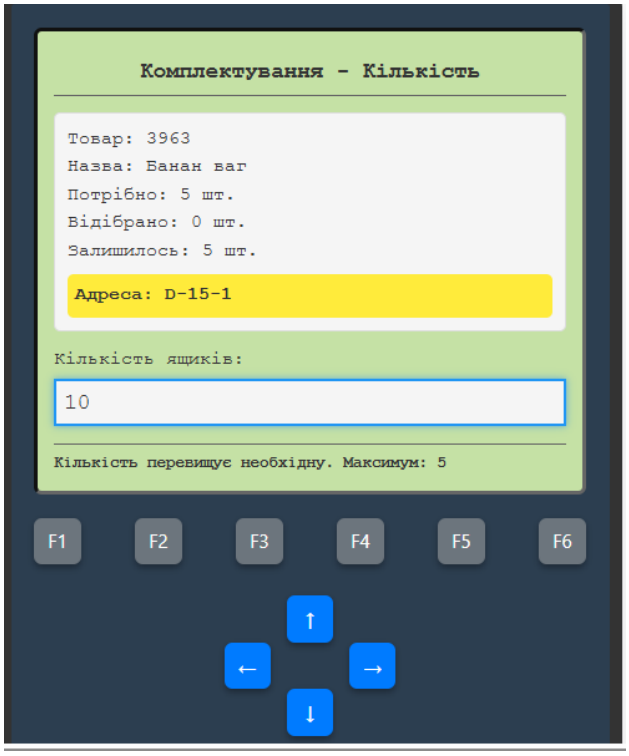

Рис. 50 Кількість більша, ніж потрібно.

Якщо артикул останній у лоті бачимо надпис «Це останній товар у лоті».

Тиснемо F1 , щоб завершити лот.

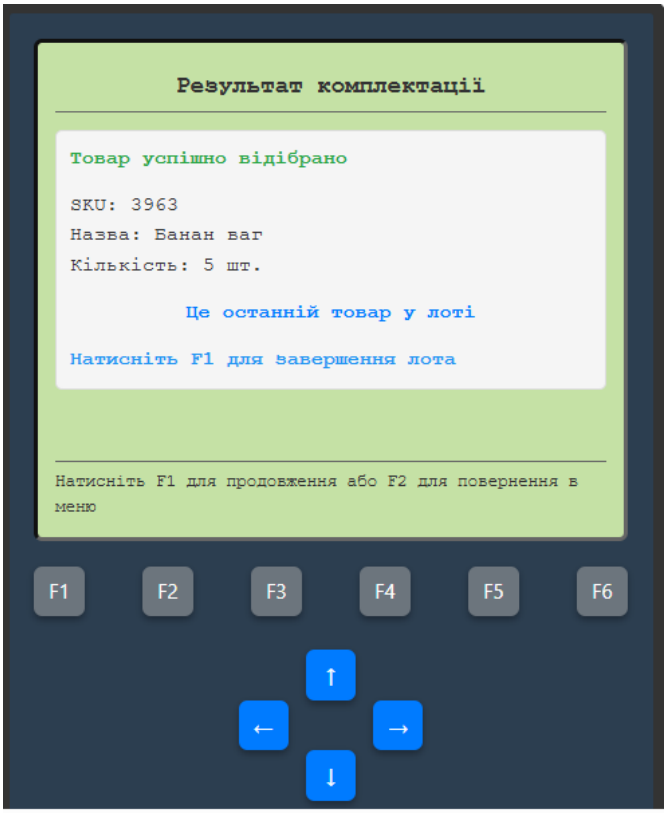

Рис. 51 Товар відібрано.

Товар відібрано, лот завершено, щоб повернутись до вибору лотів – тиснемо F2.

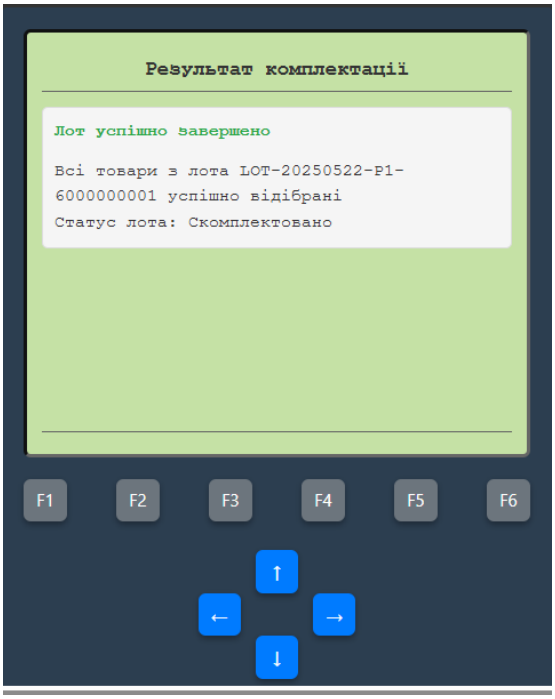

Рис. 52 Кінець лоту.

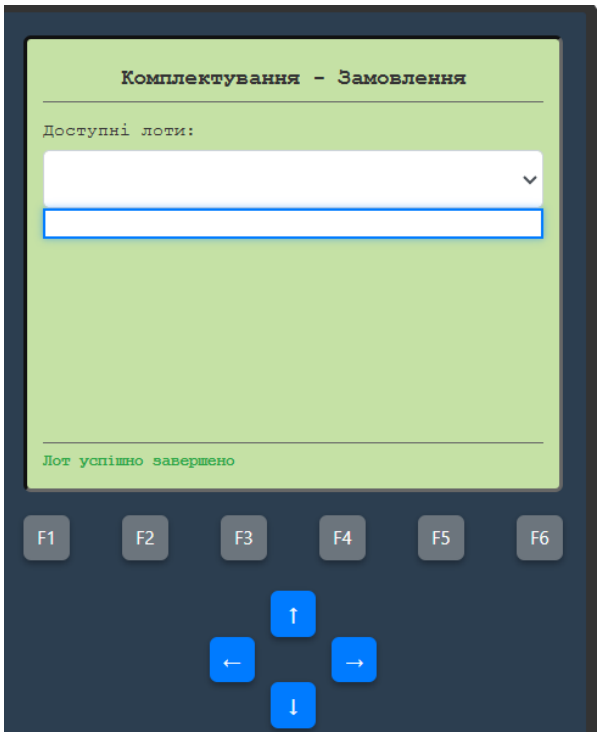

Рис. 53 Повернення у вікно вибору лотів.

Статус лота змінюється на «Зібрано». У деталях лота можливо переглянути дані про відбір.

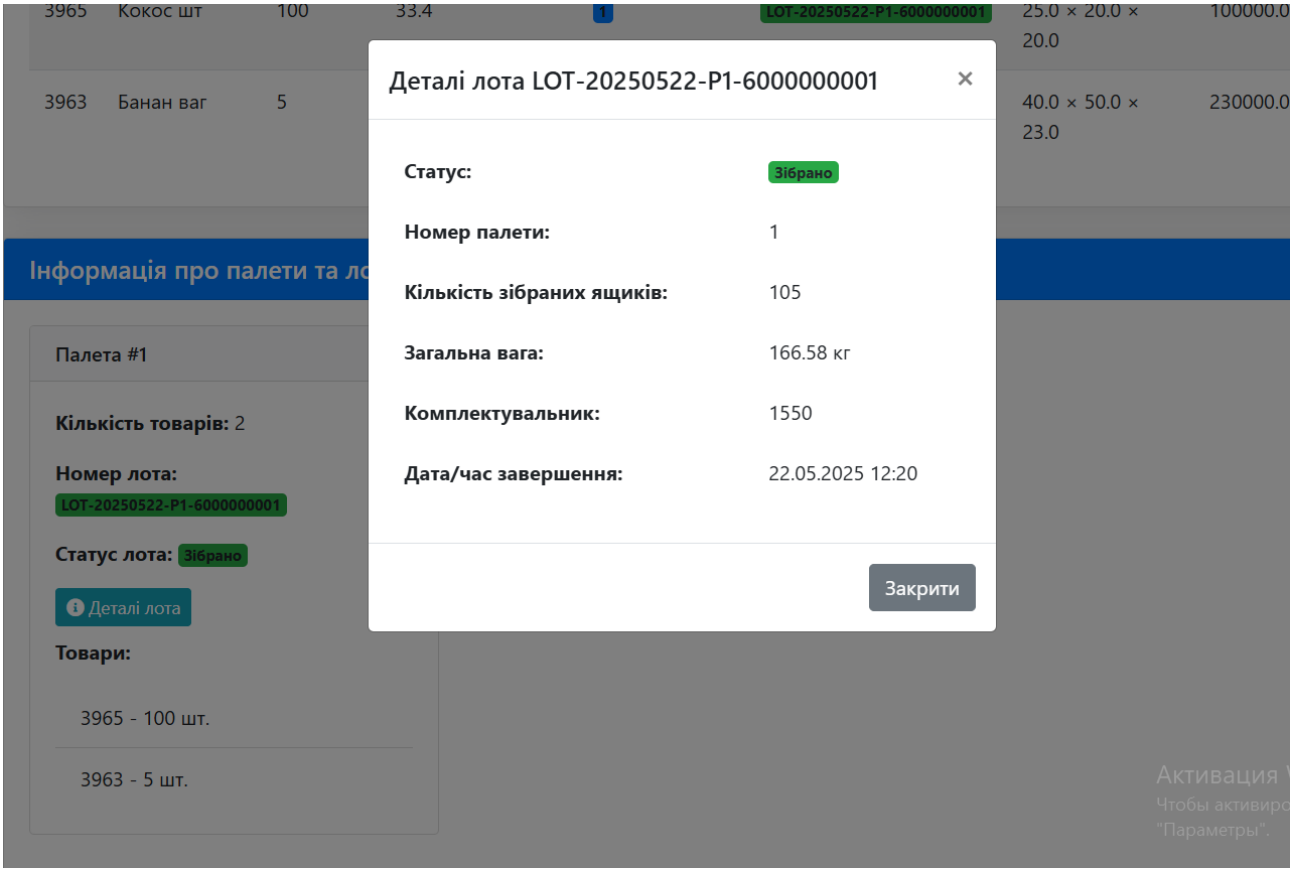

Рис. 54 Деталі зібраного лоту.

11.Відвантаження замовлення

Зібравши всі лоти у замовлені його статус змінюється на «Упаковано», та з’являється можливість відвантажити замовлення з подальшою генерацією видаткової накладної.

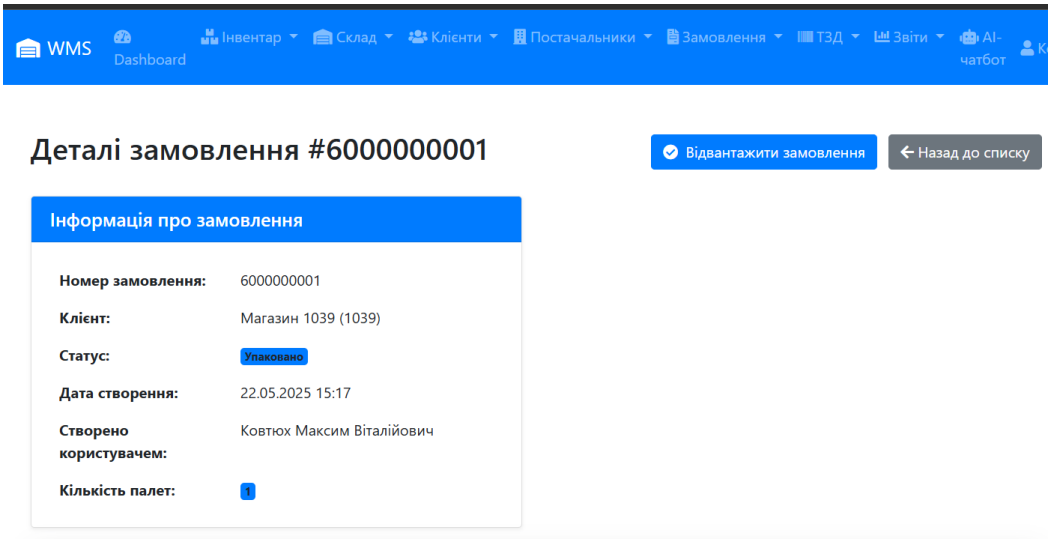

Рис. 55 Деталі зібраного замовлення.

Відвантаживши замовлення на залишки клієнту потрапляє відібрана кількість, що можливо переглянути натиснувши «Переглянути залишки товару». З’являється кнопка «Видаткова накладна».

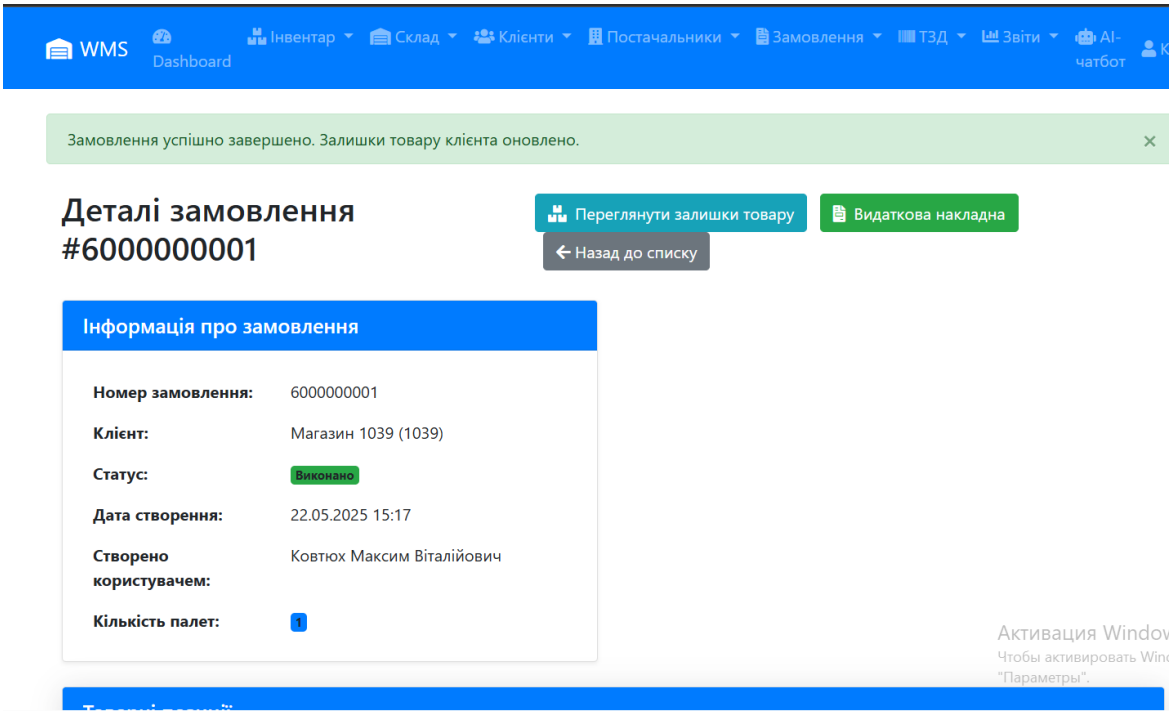

Рис. 56 Замовлення відвантажено.

Видаткова накладна містить всі необхідні дані для подальшого приймання товару клієнтом.

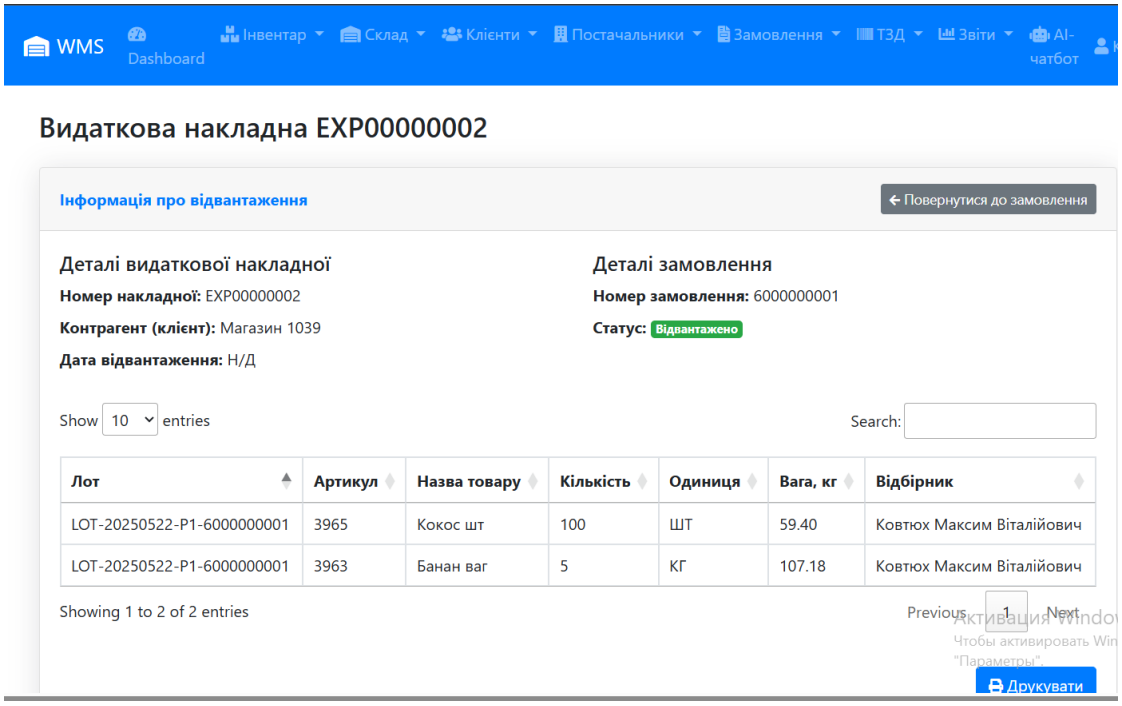

Рис. 57 Видаткова накладна

Видаткову накладку, як і рахунок-фактури можливо надрукувати.

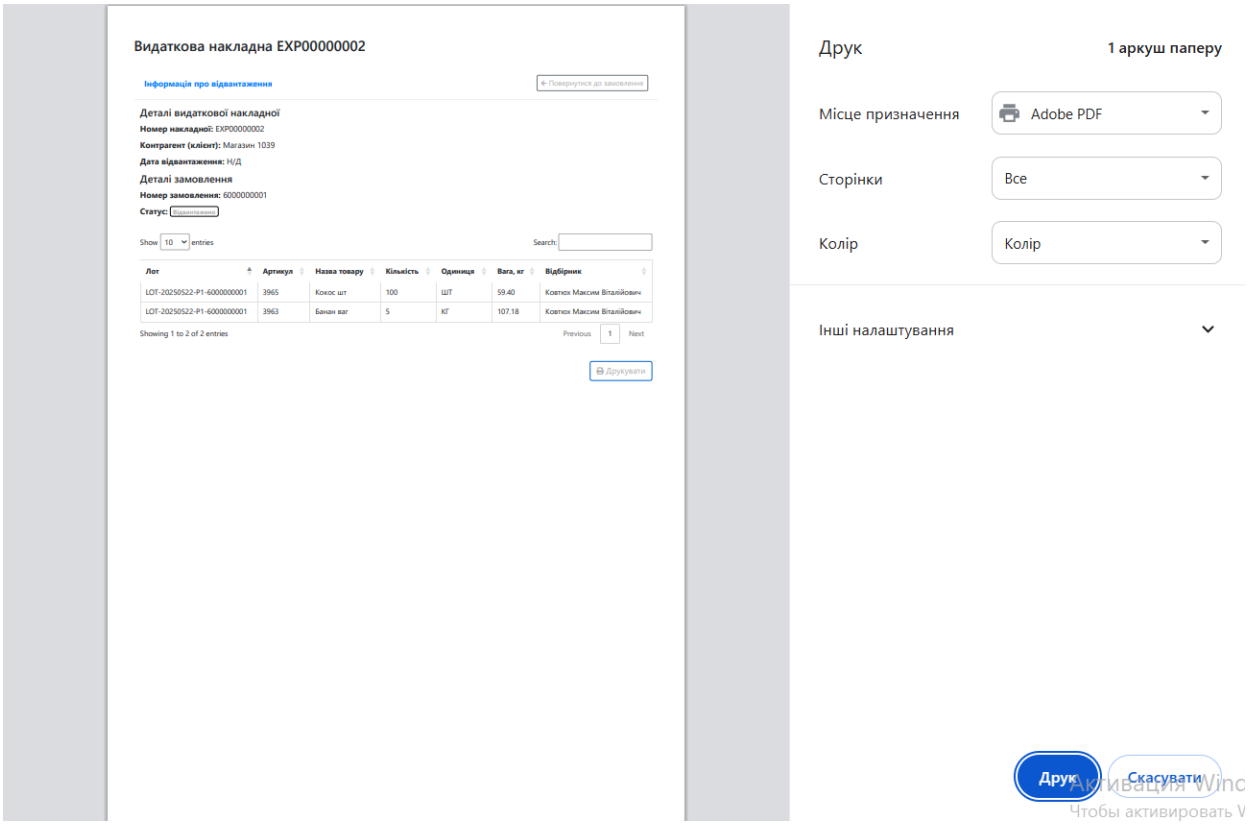

Рис. 58 Друкування видаткової накладної.

Відібраний товар потрапляє клієнту на залишки.

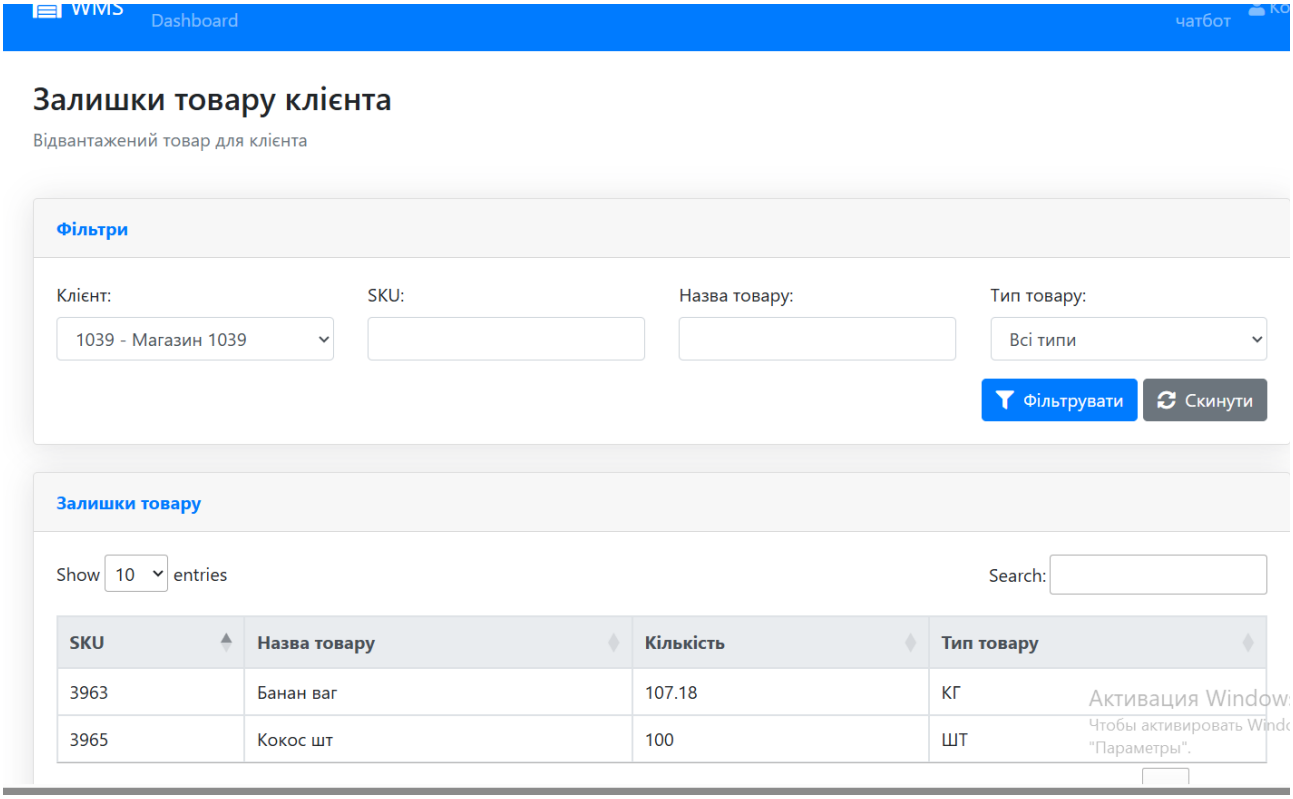

Рис. 59 Залишки клієнту.

У процесі відбору з наших залишків віднімається відібрана кількість.

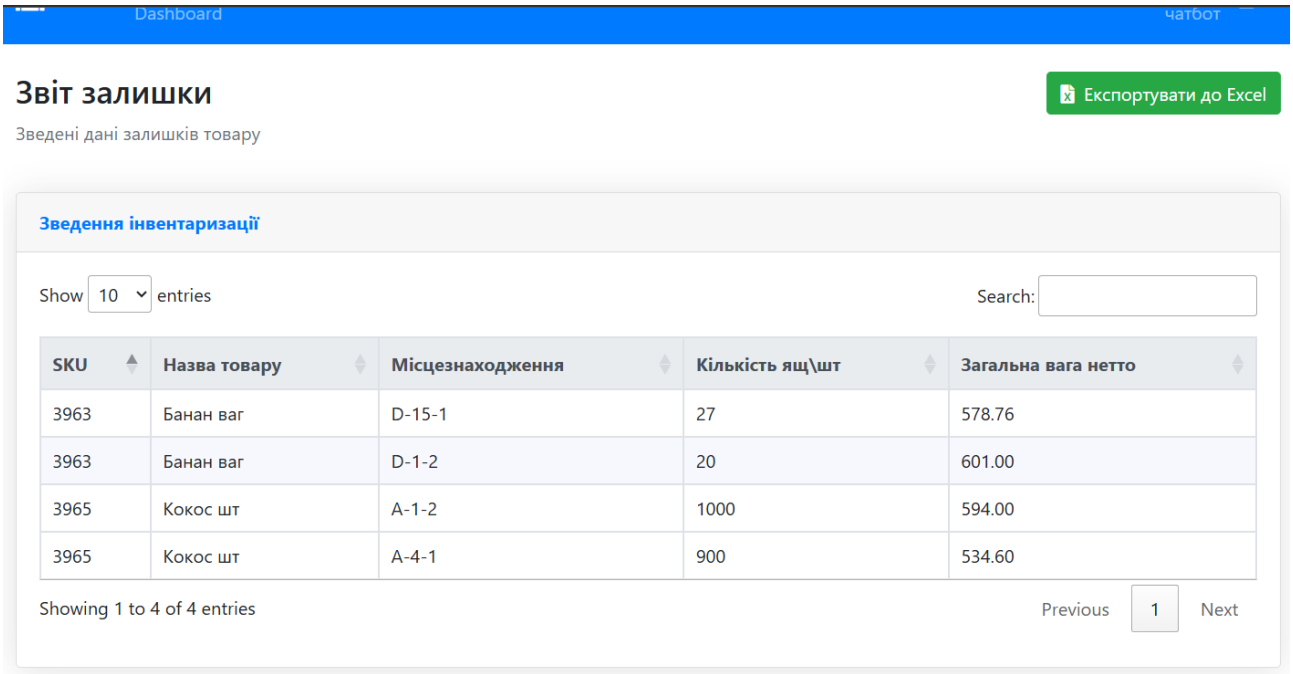

Рис. 60 Власні залишки після відбору.

Отже, відбір працює коректно.

Створимо ще одне замовлення на кількість значно більшу, ніж на залишках.

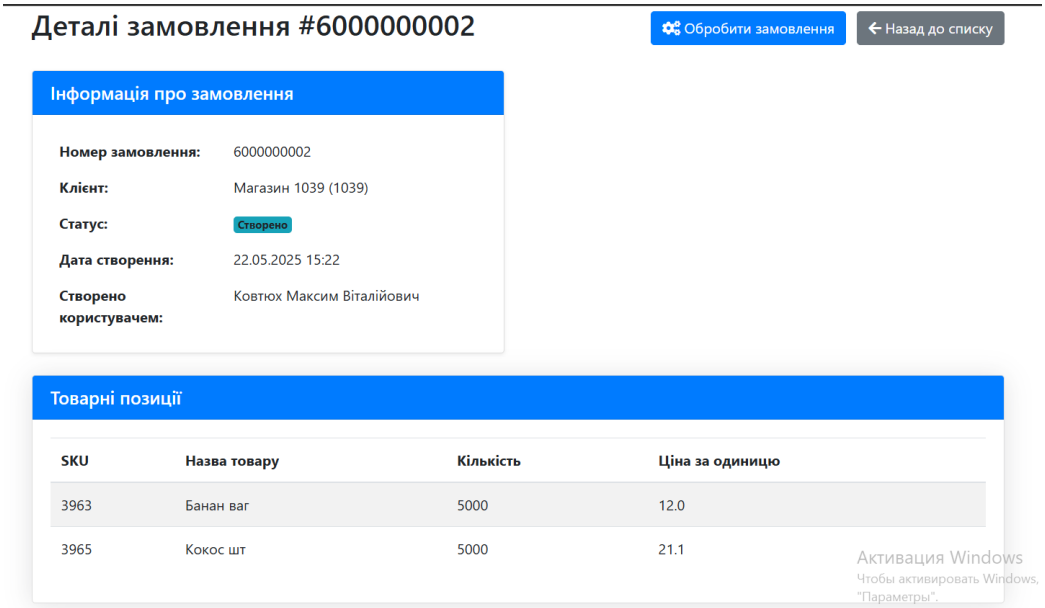

Рис. 61 Замовлення на клієнта.

12.Переміщення

Якщо під час обробки замовлення на місці відбору недостатньо товару, створюються переміщення із зберігання на місце відбору, а в деталях замовлення це помічається

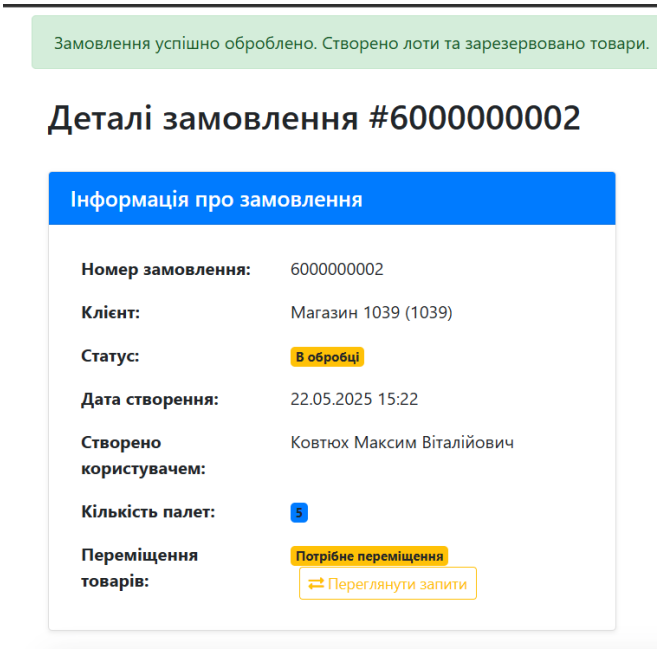

Рис. 62 Деталі нового замовлення.

Якщо замовлена кількість товару більша, ніж є на залишку, обробляється та резервується для замовлення лише, те є.

Рис. 63 Створені лоти.

Переходимо на вікно переміщення, та бачимо всі необхідні дані про палету, що потрібно поповнити у відбір. Кнопкою «Підтвердити», товар переміщається з зберігання у місце відбору.

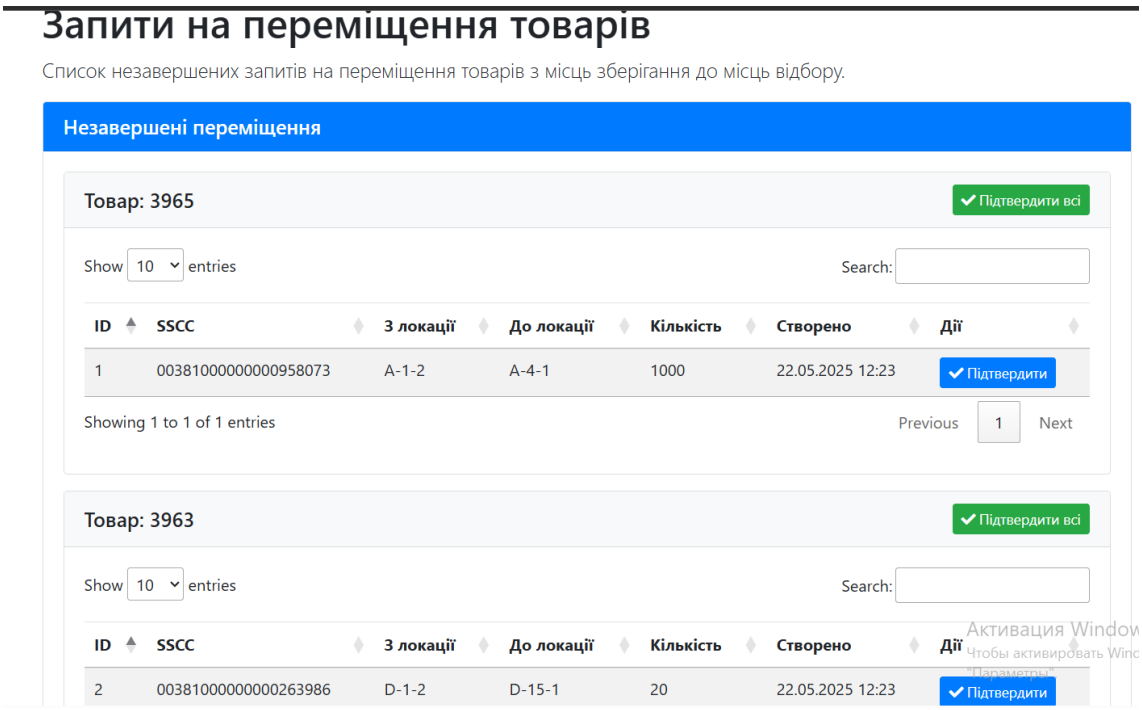

Рис. 64 Запити на переміщення товарів.

Підтвердивши, отримуємо повідомлення, що успішно підтверджено.

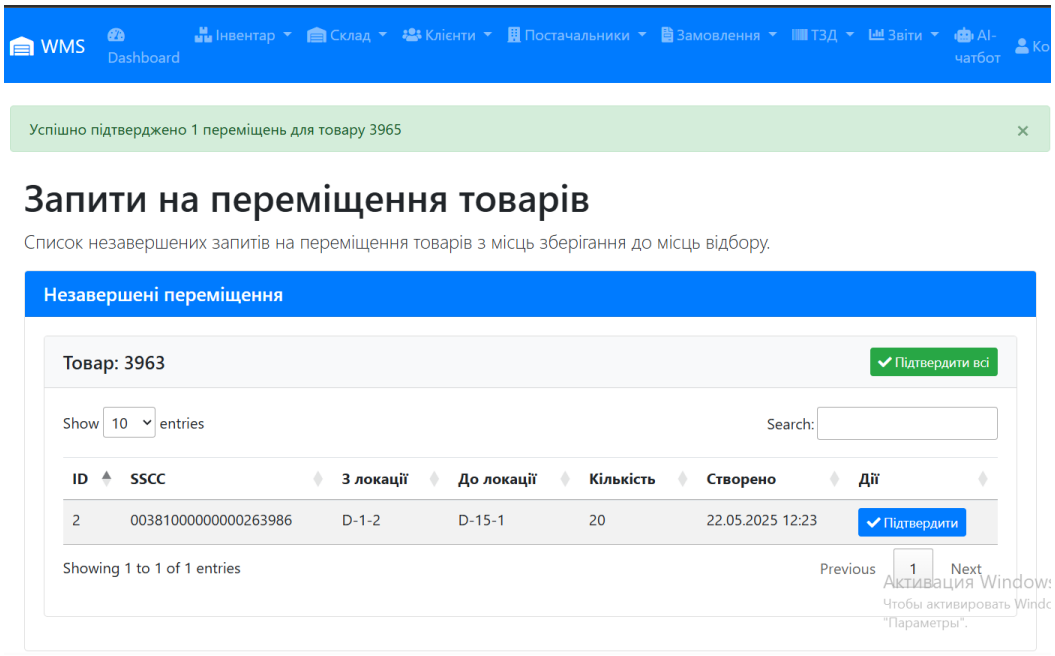

Рис. 65 Успішне переміщення (1/2)

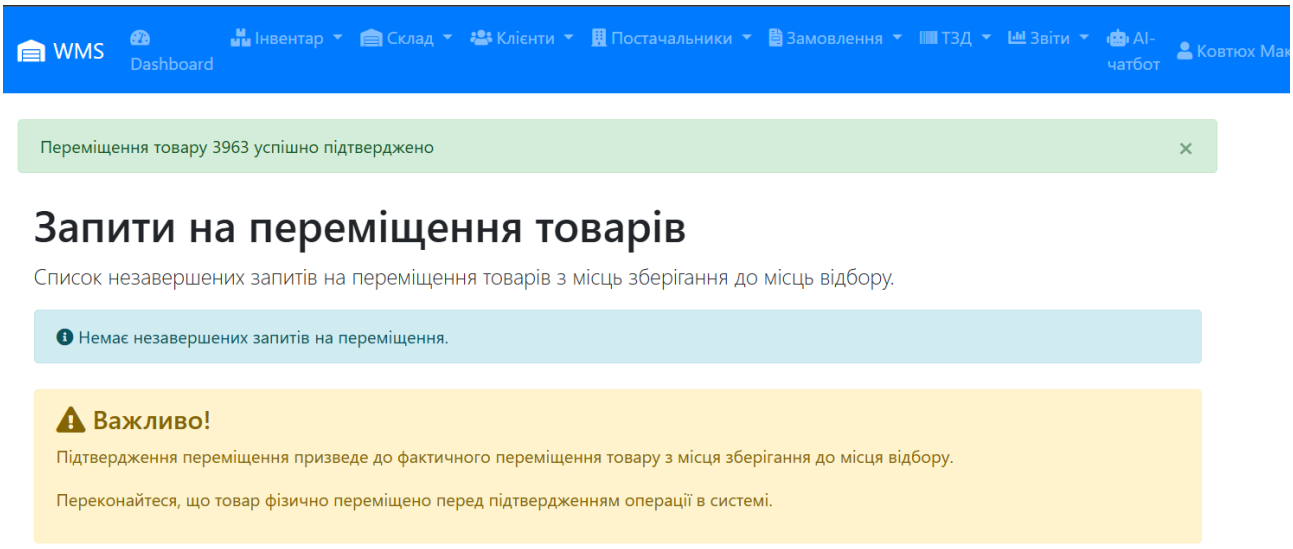

Рис. 66 Успішне переміщення (2/2)

На власних залишках також бачимо, що товар переміщено.

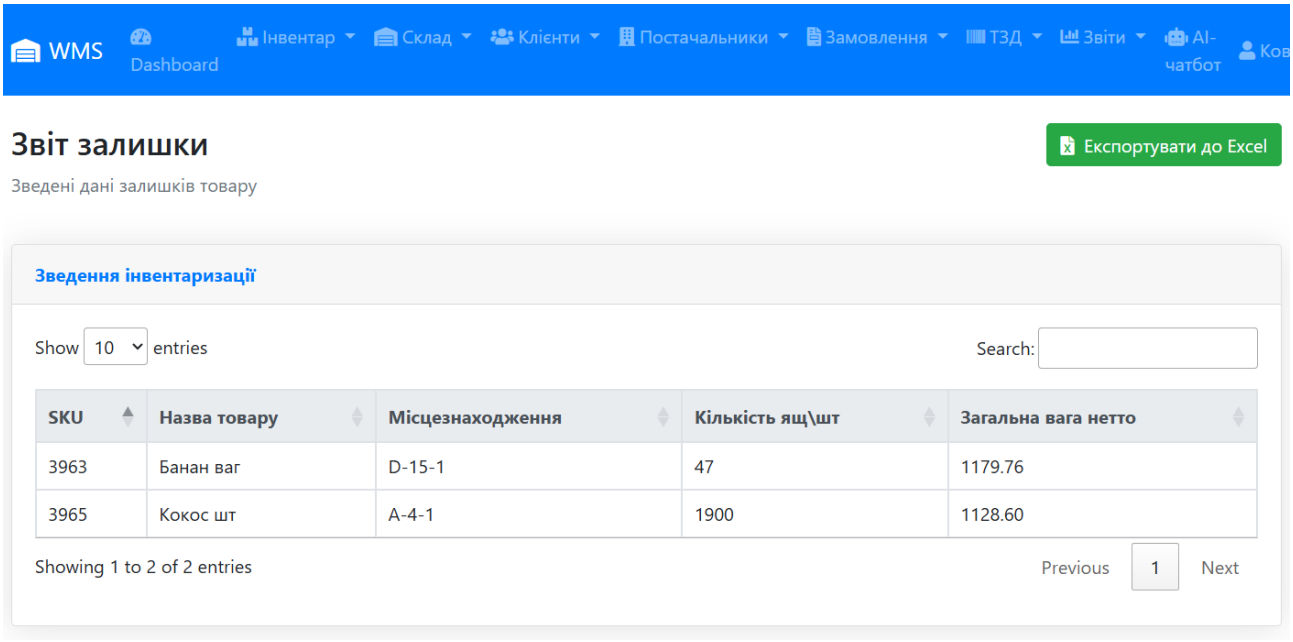

Рис. 67 Залишки після переміщення.

Відбираємо та відвантажуємо товар.

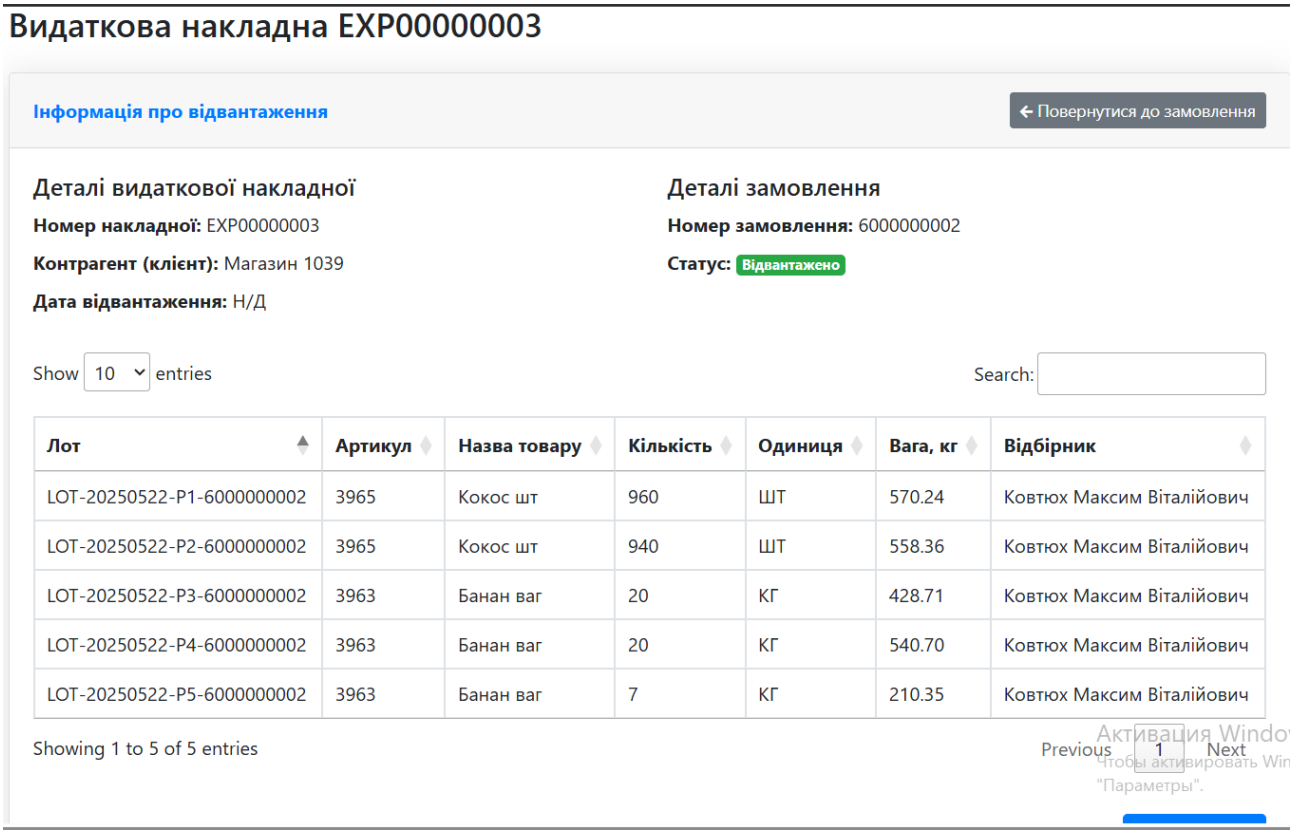

Рис. 68 Видаткова накладна.

Всі залишки товару виїхали.

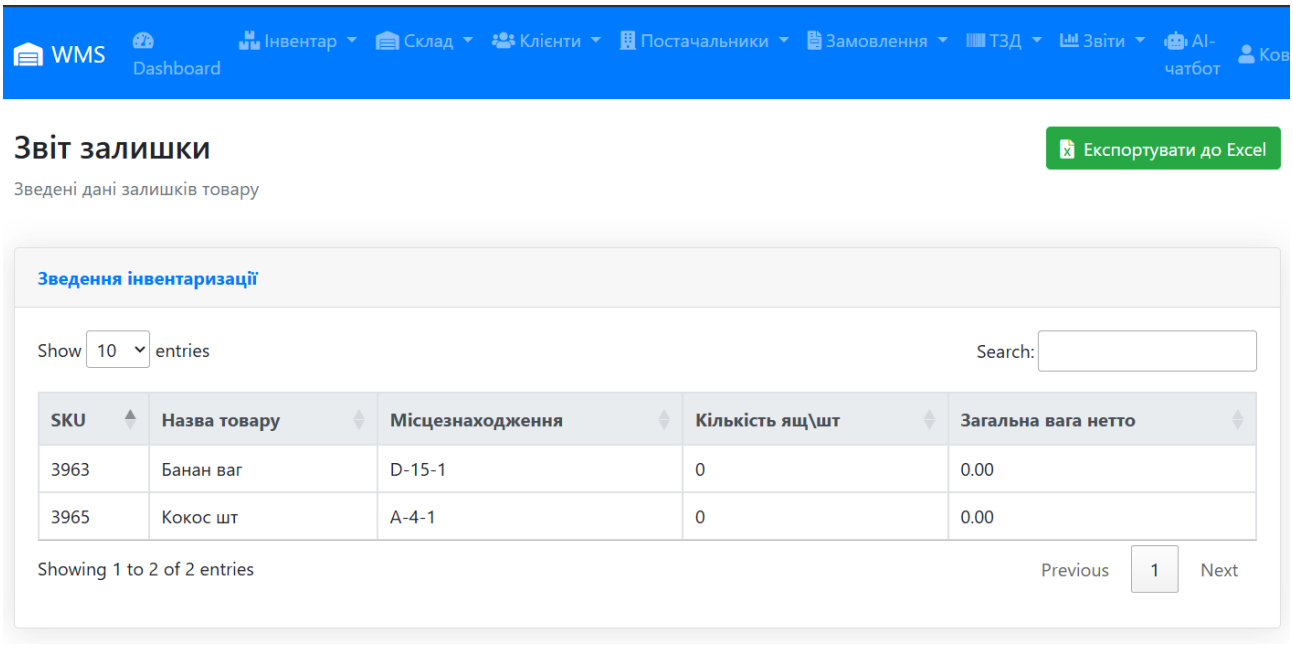

Рис. 69 Залишки товару відсутні.

13.Чат-бот

Перевіримо чат-бота на декількох запитах.

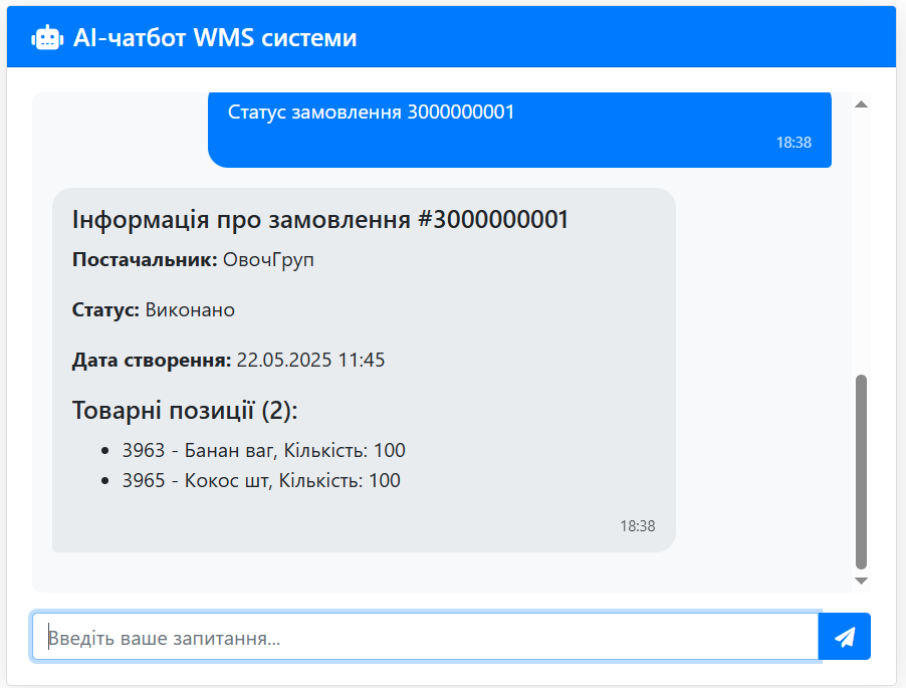

Рис. 70 Запит чат-боту (1/3)

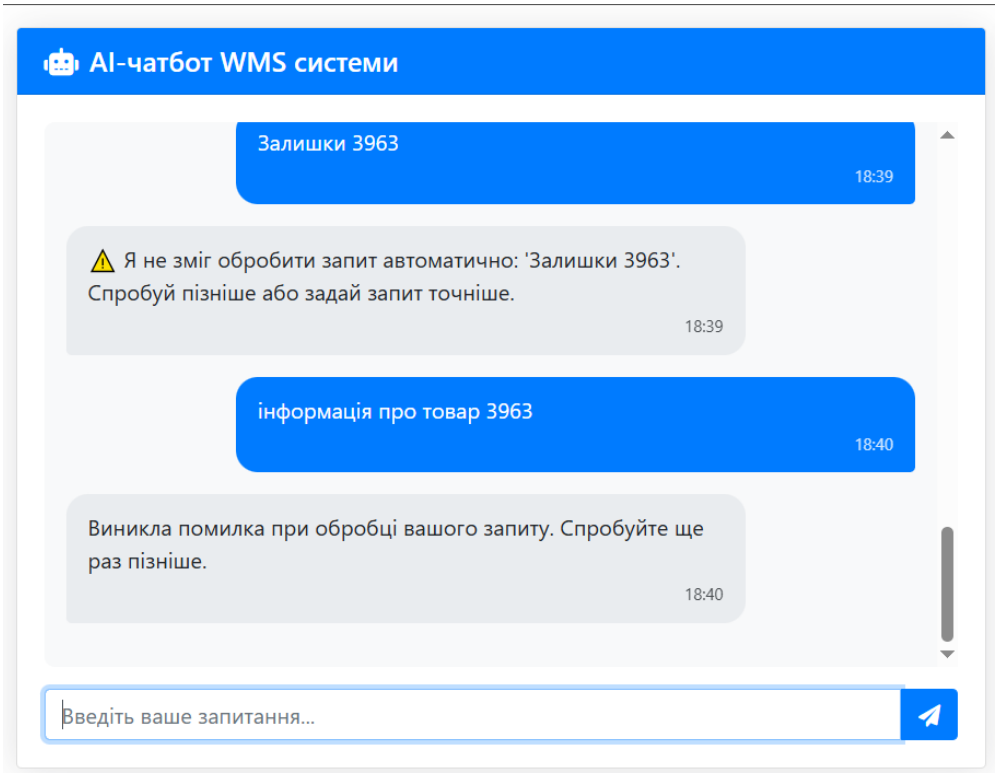

Рис. 71 Запит чат-боту (2/3)

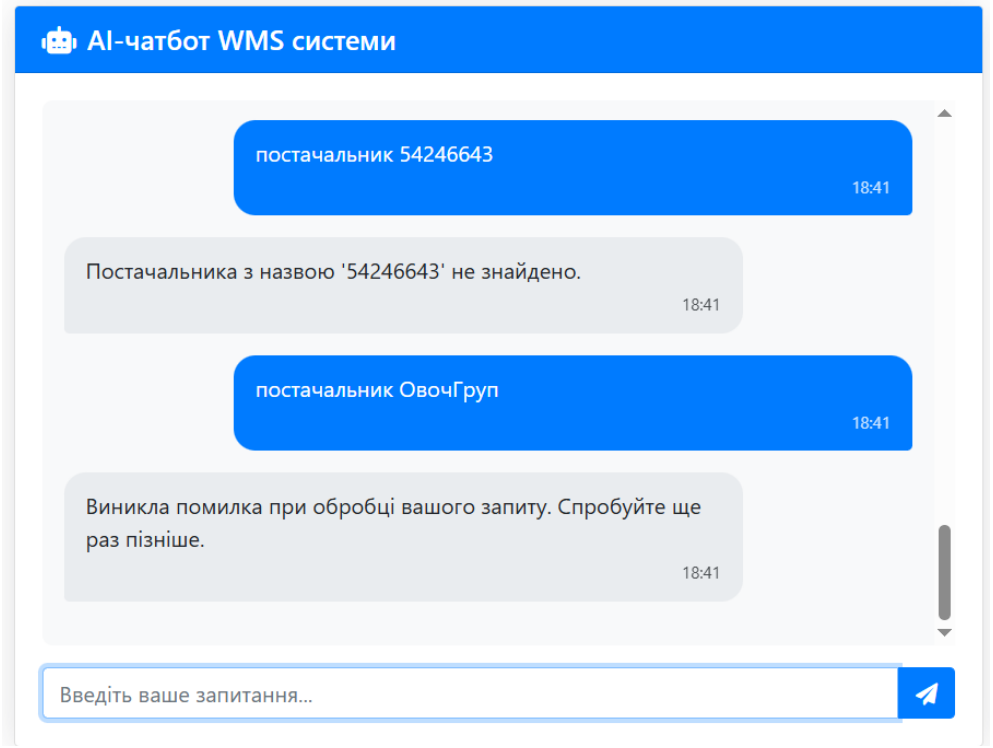

Рис. 72 Запит чат-боту (3/3)

Отже, можна зробити висновок, що чат бот працює не достатньо коректно. Необхідно точніше розробити функції отримання команд, щоб навіть оффлайн-бот був корисним та допомагав.

Додаток Б. Діаграми.

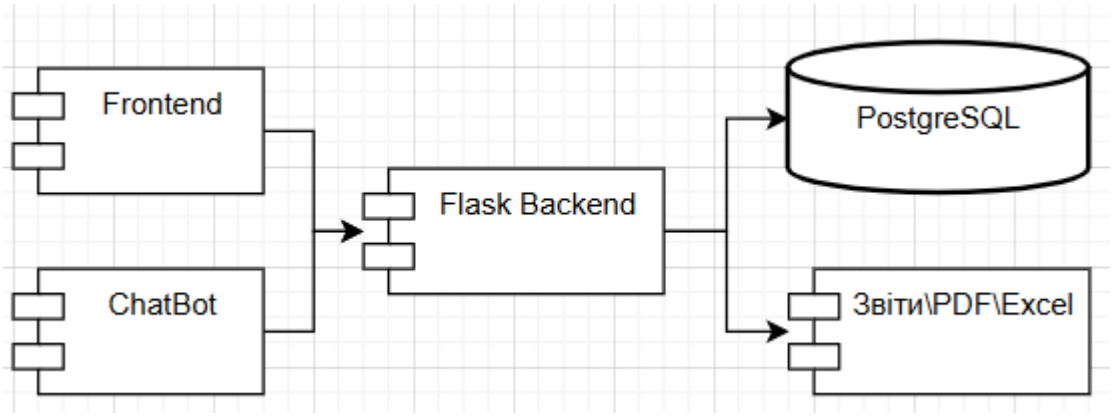

Рис. 73 UML-діаграма компонентів.

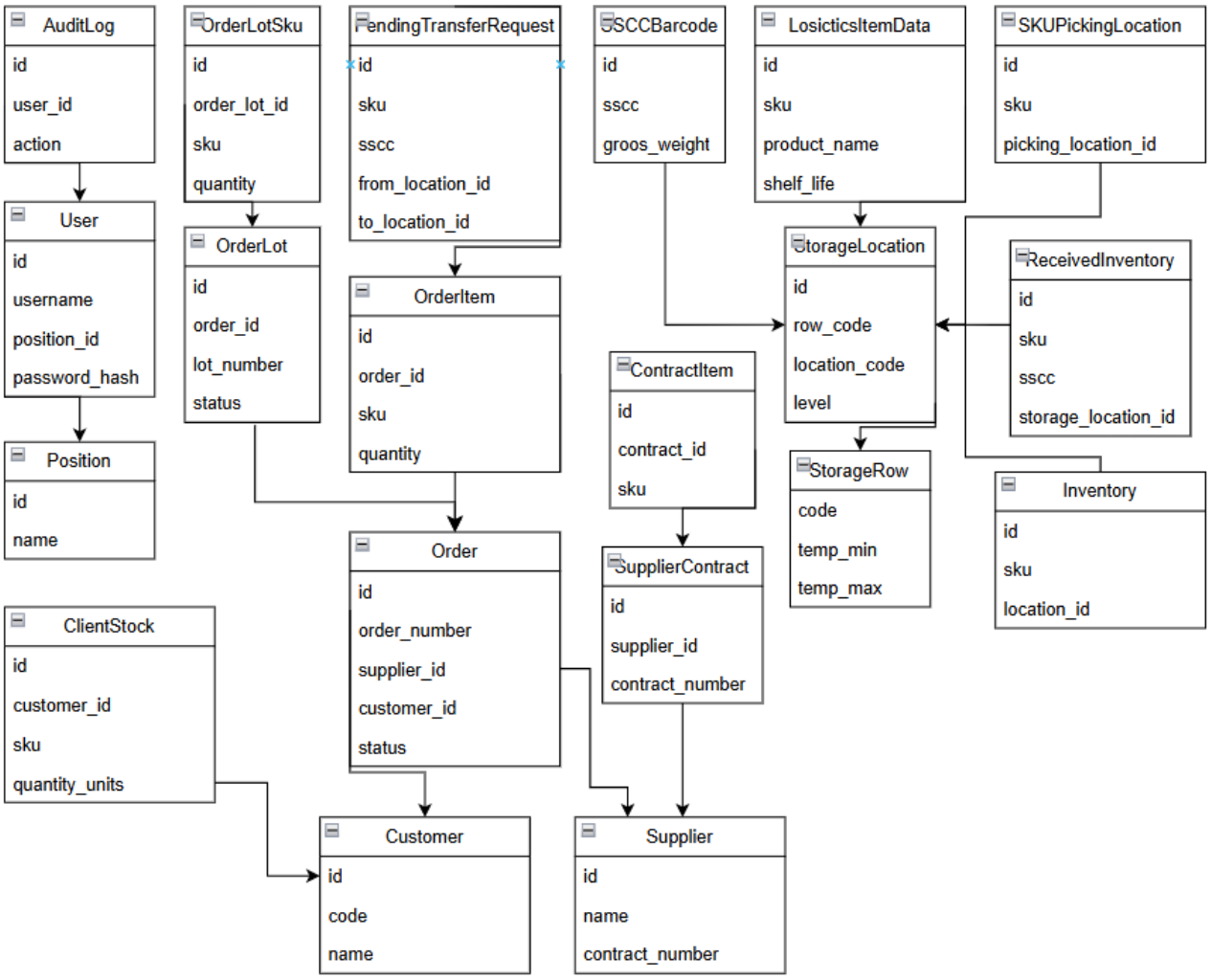

Рис. 74 UML-діаграма класів

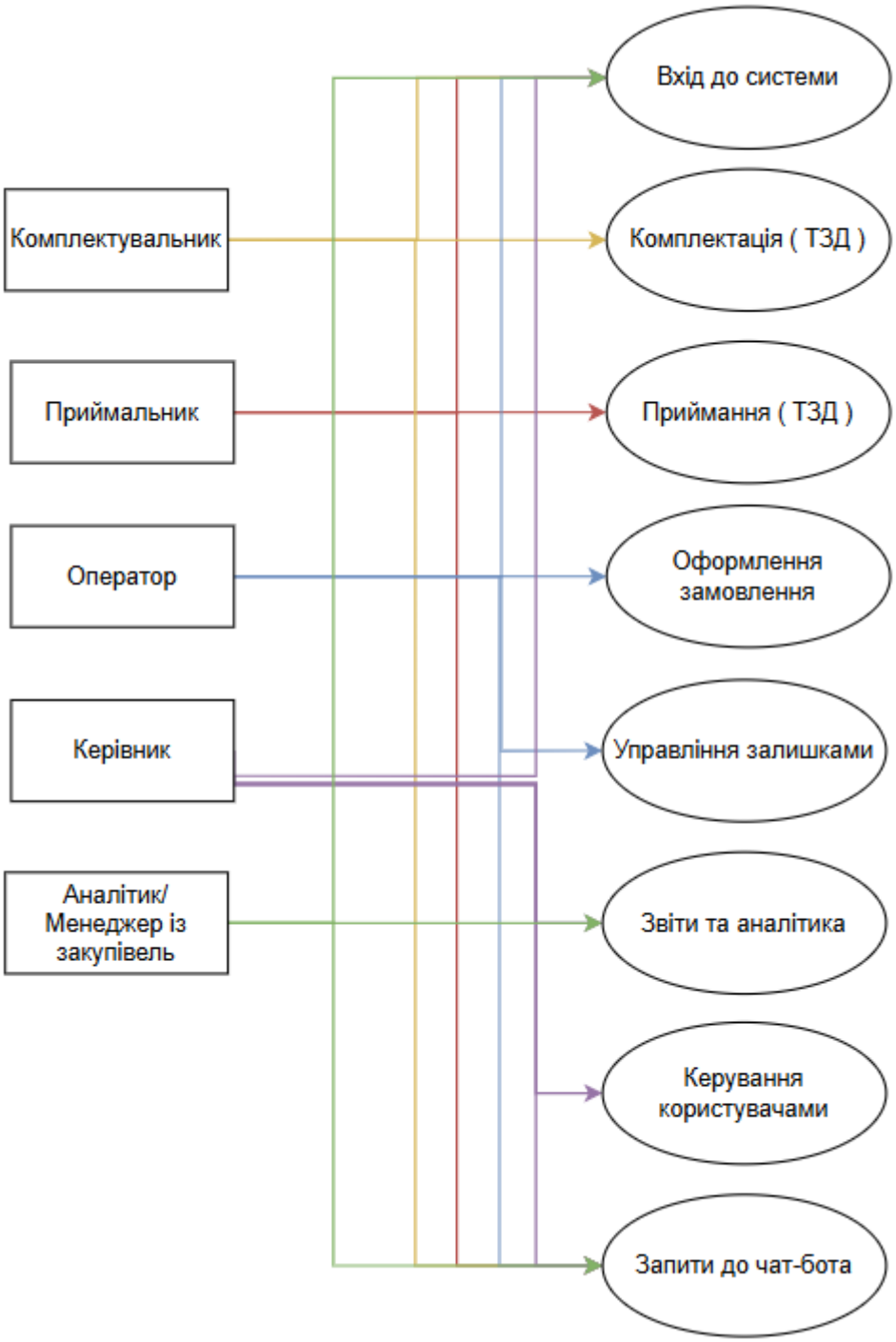

Рис. 75 UML-діаграма прецедентів

Додаток В. Основні фрагменти програмного коду.

1. Жадібний алгоритм розподілу товарів по палетах.

```
def split_items_by_volume(self):
        """
        Розбиває товари по палетах з урахуванням спільного об'єму (жадібний алгоритм).
        Товари з різними SKU можуть бути на одній палеті, якщо вміщаються по об'єму.
        Ураховує кратність упаковки для штучного товару та доступність товару на
складі.
        """
        from models import LogisticsItemData, OrderItem
        from models.inventory import Inventory

        PALLET_VOLUME = 960000
        pallets = []  # Список палет: [{'number': int, 'volume': float, 'items': []}]
        next_pallet_number = 1
        new_items = []
        skipped_items = []

        for item in self.items:
            # Перевіряємо загальний доступний залишок по всіх рівнях складу
            total_available = Inventory.get_total_available_all_levels(item.sku) if
hasattr(Inventory, 'get_total_available_all_levels') else
Inventory.get_total_available(item.sku)

            # Якщо товару немає в наявності, пропускаємо його
            if total_available <= 0:
                skipped_items.append({
                    'sku': item.sku,
                    'product_name': item.product_name,
                    'requested': item.quantity
                })
                print(f"Товар {item.sku} ({item.product_name}) пропущено: немає в
наявності")
                continue

            # Обмежуємо кількість до фактично доступної
            if total_available < item.quantity:
                print(f"Кількість товару {item.sku} обмежена до {total_available} (було
{item.quantity})")
                item.quantity = total_available

            logistics = LogisticsItemData.query.filter_by(sku=item.sku).first()
            if not logistics:
```

```
                continue

            # Якщо товар штучний, розрахунок іде на ящики (враховується кратність)
            multiplicity = logistics.count or 1  # кратність упаковки (напр., 10 шт = 1
ящик)

            # Перевіряємо, чи це ваговий товар (або з прапорця, або з типу упаковки)
            is_weight_based = item.is_weight_based or logistics.packaging_unit_type ==
"КГ"  # "КГ" або "ШТ"

            # Встановлюємо прапорець для подальшого використання
            item.is_weight_based = is_weight_based

            # Об’єм одиниці для вагового товару — просто упаковка; для штучного — ящик
            unit_volume = logistics.length * logistics.width * logistics.height
            if is_weight_based:
                boxes_remaining = item.quantity  # для вагових товарів: 1 шт = 1 "ящик"
            else:
                boxes_remaining = math.ceil(item.quantity / multiplicity)  # кількість
ящиків

            while boxes_remaining > 0:
                placed = False
                for pallet in pallets:
                    if pallet['volume'] + unit_volume <= PALLET_VOLUME:
                        qty_fit = min(
                            boxes_remaining,
                            int((PALLET_VOLUME - pallet['volume']) // unit_volume)
                        )
                        if qty_fit > 0:
                            actual_qty = qty_fit if is_weight_based else qty_fit *
multiplicity

                            new_item = OrderItem(
                                sku=item.sku,
                                product_name=item.product_name,
                                quantity=actual_qty,
                                unit_price=item.unit_price,
                                pallet_number=pallet['number'],
                                length_cm=logistics.length,
                                width_cm=logistics.width,
                                height_cm=logistics.height,
                                is_weight_based=is_weight_based
                            )
                            pallet['volume'] += qty_fit * unit_volume
```

```
                            pallet['items'].append(new_item)
                            new_items.append(new_item)
                            boxes_remaining -= qty_fit
                            placed = True
                        break

                if not placed:
                    qty_fit = min(boxes_remaining, int(PALLET_VOLUME // unit_volume))
or 1
                    actual_qty = qty_fit if is_weight_based else qty_fit * multiplicity
                    new_item = OrderItem(
                        sku=item.sku,
                        product_name=item.product_name,
                        quantity=actual_qty,
                        unit_price=item.unit_price,
                        pallet_number=next_pallet_number,
                        length_cm=logistics.length,
                        width_cm=logistics.width,
                        height_cm=logistics.height,
                        is_weight_based=is_weight_based
                    )
                    pallets.append({
                        'number': next_pallet_number,
                        'volume': qty_fit * unit_volume,
                        'items': [new_item]
                    })
                    new_items.append(new_item)
                    boxes_remaining -= qty_fit
                    next_pallet_number += 1

        # Виводимо інформацію про пропущені товари
        if skipped_items:
            print(f"Пропущено {len(skipped_items)} товарів через відсутність на
складі:")
            for item in skipped_items:
                print(f"  - {item['sku']} ({item['product_name']}): запитано
{item['requested']}")

        self.items.clear()
        self.items.extend(new_items)
        self.pallets_count = len(pallets)
        db.session.commit()
        return True
```

2. Перетворення ваги у ящики, при обробці замовлення.

```
def adapt_weight_based_items(self):
        """Адаптивна обробка вагових товарів (packaging_unit_type == 'КГ')
        Конвертує вагу в приблизну кількість ящиків на основі середньої ваги"""
        from models import LogisticsItemData
        from models.inventory import Inventory
        from models.received_inventory import ReceivedInventory
        import math

        for item in self.items:
            # Отримуємо логістичні дані
            logistics_item = LogisticsItemData.query.filter_by(sku=item.sku).first()
            if not logistics_item or logistics_item.packaging_unit_type != 'КГ':
                continue

            # Отримуємо всі прийняті палети для SKU
            received_items = ReceivedInventory.query.filter_by(sku=item.sku).all()

            total_weight = sum(p.net_weight for p in received_items if p.net_weight)
            total_boxes = sum(p.box_count for p in received_items if p.box_count)

            if total_boxes == 0 or total_weight == 0:
                continue  # Уникаємо ділення на нуль

            avg_weight_per_box = total_weight / total_boxes

            requested_weight = item.quantity  # у КГ
            estimated_box_count = math.ceil(requested_weight / avg_weight_per_box)

            # Зберігаємо оригінальну запитану вагу для подальшого використання
            item.original_requested_quantity = item.quantity

            # Конвертуємо вагу в кількість ящиків
            item.quantity = estimated_box_count
            item.is_weight_based = True

            # Перевіряємо загальний доступний залишок по всіх рівнях складу
            total_available = Inventory.get_total_available_all_levels(item.sku)

            # Якщо доступно менше, ніж запитано, обмежуємо кількість
            if total_available < estimated_box_count:
                print(f"Для товару {item.sku} ({item.product_name}) доступно лише
{total_available} з {estimated_box_count} необхідних ящиків")
```

```
                item.quantity = total_available

        db.session.commit()
```

3. Комплексна обробка замовлення

```
def process_order(self):
        """Обробляє замовлення: розраховує палети, створює лоти, резервує товари.
        Обробляє лише ту кількість товару, яка реально доступна на складі.
        Створює лоти незалежно від підтвердження переміщення."""
        try:
            # Крок 0: Адаптивна обробка вагових товарів
            self.adapt_weight_based_items()

            # Крок 1: Розрахунок кількості палет
            self.calculate_pallets()

            # Крок 2: Розподіл товарів по палетах з урахуванням доступності
            self.split_items_by_volume()

            # Крок 3: Резервування товарів та отримання списку оброблених товарів
            processed_items = self.reserve_inventory()

            # Якщо немає жодного обробленого товару, повертаємо помилку
            if not processed_items:
                print(f"Замовлення {self.order_number} не може бути оброблене: немає
доступних товарів")
                return False

            # Крок 4: Створення лотів для всіх товарів, незалежно від підтвердження
переміщення
            # Змінено: створюємо лоти навіть якщо товар знаходиться на верхніх рівнях і
потребує переміщення
            self.create_lots()

            # Крок 5: Оновлення статусу замовлення
            self.status = 'processing'
            db.session.commit()

            # Виводимо інформацію про оброблене замовлення
            print(f"Замовлення {self.order_number} оброблено. Оброблено
{len(processed_items)} товарів.")
            if len(processed_items) < len(self.items):
```

```
                print(f"Увага! {len(self.items) - len(processed_items)} товарів не
оброблено через недостатню кількість на складі.")

            return True
        except Exception as e:
            db.session.rollback()
            print(f"Помилка обробки замовлення: {str(e)}")
            raise e
```

4. Резервування і розрізервування інвентарю

```
def reserve(self, quantity):
        """Резервує вказану кількість товару"""
        if quantity <= 0:
            raise ValueError("Кількість для резервування має бути більше нуля")

        if self.available_quantity < quantity:
            raise ValueError(f"Недостатньо товару для резервування. Доступно:
{self.available_quantity}, запитано: {quantity}")

        self.reserved_quantity += quantity
        db.session.commit()
        return True

    def unreserve(self, quantity):
        """Скасовує резервування вказаної кількості товару"""
        if quantity <= 0:
            raise ValueError("Кількість для скасування резервування має бути більше
нуля")

        if self.reserved_quantity < quantity:
            raise ValueError(f"Неможливо скасувати резервування. Зарезервовано:
{self.reserved_quantity}, запитано: {quantity}")

        self.reserved_quantity -= quantity
        db.session.commit()
        return True

    @classmethod
    def unreserve_by_sku(cls, sku, quantity):
        """Скасовує резервування вказаної кількості товару за SKU"""
        if quantity <= 0:
            raise ValueError("Кількість для скасування резервування має бути більше
нуля")
```

```
        # Знаходимо всі записи за SKU
        items = cls.query.filter_by(sku=sku).order_by(cls.created_at).all()

        # Перевіряємо, чи достатньо зарезервовано
        total_reserved = sum(item.reserved_quantity for item in items)
        if total_reserved < quantity:
            raise ValueError(f"Неможливо скасувати резервування. Всього зарезервовано:
{total_reserved}, запитано: {quantity}")

        # Скасовуємо резервування
        remaining = quantity
        for item in items:
            if item.reserved_quantity > 0:
                to_unreserve = min(item.reserved_quantity, remaining)
                item.reserved_quantity -= to_unreserve
                remaining -= to_unreserve

                if remaining <= 0:
                    break

        db.session.commit()
        return True

    def can_reserve(self, quantity):
        """Перевіряє, чи можна зарезервувати вказану кількість товару"""
        return self.available_quantity >= quantity

    @classmethod
    def can_reserve_total(cls, sku, quantity):
        """Перевіряє, чи можна зарезервувати вказану кількість товару за SKU"""
        available = cls.get_total_available(sku)
        return available >= quantity

    @classmethod
    def reserve_by_sku(cls, sku, quantity):
        """Резервує вказану кількість товару за SKU"""
        if quantity <= 0:
            raise ValueError("Кількість для резервування має бути більше нуля")

        available = cls.get_total_available(sku)
        if available < quantity:
            raise ValueError(f"Недостатньо товару для резервування. Доступно:
{available}, запитано: {quantity}")
```

```
        # Знаходимо всі записи за SKU і резервуємо потрібну кількість
        items = cls.query.filter_by(sku=sku).order_by(cls.created_at).all()
        remaining = quantity

        for item in items:
            if item.available_quantity > 0:
                to_reserve = min(item.available_quantity, remaining)
                item.reserved_quantity += to_reserve
                remaining -= to_reserve

                if remaining <= 0:
                    break

        db.session.commit()
        return True
```

5. Перевірка температурної сумісності

```
def check_item_compatibility(self, item):
        """Перевірка, чи сумісний товарний елемент із цим місцем розташування на основі його логістичних даних"""
        from models import LogisticsItemData

        # Якщо товар не має артикула, ми не можемо перевірити сумісність
        if not item or not item.sku:
            return True

        # Шукаємо логістичні дані для цього артикулу
        logistics_data = LogisticsItemData.query.filter_by(sku=item.sku).first()

        # Якщо немає логістичних даних або вимог до температури, вважайте сумісним
        if not logistics_data or logistics_data.temperature_range is None:
            return True

        # Перетворити окреме значення температури в діапазон (±2°C)
        item_temp = logistics_data.temperature_range
        min_temp = item_temp - 2
        max_temp = item_temp + 2

        # Перевірка, чи сумісний температурний діапазон
        return self.is_temperature_compatible(min_temp, max_temp)
```

6. Алгоритм комплектації

```
def perform_picking(order_id, sku, requested_quantity, user_id, lot_number):
    """
    Виконує відбір товару для замовлення

    Args:
        order_id: ID замовлення
        sku: Артикул товару
        requested_quantity: Запитана кількість
        user_id: ID користувача, який виконує відбір

    Returns:
        dict: Результат операції з інформацією про відбір
    """
    from models.received_inventory import ReceivedInventory
    from models.storage import StorageLocation
    from models.order import Order, OrderItem
    from models import LogisticsItemData
    from models.pending_transfer import PendingTransferRequest
    import math

    # Перевіряємо наявність замовлення
    order = Order.query.get(order_id)
    if not order or order.status != 'assembling':
        return {
            'success': False,
            'message': 'Замовлення не знайдено або не в статусі комплектації'
        }

    # Перевіряємо наявність товару в замовленні
    order_item = OrderItem.query.filter_by(order_id=order_id, sku=sku,
lot_number=lot_number).first()
    if not order_item:
        return {
            'success': False,
            'message': 'Товар не знайдено в замовленні'
        }

    # Перевіряємо наявність незавершених запитів на переміщення для цього SKU
    pending_transfers = PendingTransferRequest.get_pending_transfers_for_sku(sku)
    if pending_transfers:
        return {
            'success': False,
```

```
            'message': f'Для товару {sku} є незавершені запити на переміщення.
Зачекайте на їх підтвердження перед відбором.',
            'pending_transfers': len(pending_transfers)
        }

    # Отримуємо дані про логістику товару
    logistics = LogisticsItemData.query.filter_by(sku=sku).first()
    if not logistics:
        return {
            'success': False,
            'message': 'Логістичні дані для товару не знайдено'
        }

    # Використовуємо безпосередньо запитану кількість
    target_box_count = requested_quantity

    # Знаходимо товар в комірках комплектації (level = 1)
    # Спочатку знаходимо SSCC, які вже відібрані для інших замовлень

    # Тепер знаходимо доступний інвентар, виключаючи вже відібрані SSCC
    picking_inventory = db.session.query(ReceivedInventory).join(
        StorageLocation, ReceivedInventory.storage_location_id == StorageLocation.id
    ).filter(
        ReceivedInventory.sku == sku,
        StorageLocation.level == '1',
        ReceivedInventory.box_count > 0,
        # Виключаємо SSCC, які вже відібрані для інших замовлень
    ).order_by(ReceivedInventory.expiry_date).all()

    # Виводимо діагностичну інформацію
    print(f"Знайдено {len(picking_inventory)} доступних позицій для SKU {sku} в
комірках комплектації")

    # Перевіряємо загальну доступну кількість
    available_boxes = sum(item.box_count for item in picking_inventory)

    # Якщо недостатньо товару в комірках комплектації, перевіряємо наявність запитів на
переміщення
    if available_boxes < target_box_count:
        from models.inventory_management import check_picking_availability
        from models.pending_transfer import PendingTransferRequest

        # Перевіряємо наявність товару та створюємо запити на переміщення при
необхідності
```

```
        availability_result = check_picking_availability(sku, target_box_count)

        # Перевіряємо, чи є незавершені запити на переміщення
        pending_transfers = PendingTransferRequest.get_pending_transfers_for_sku(sku)

        if pending_transfers:
            return {
                'success': False,
                'message': f'Для товару {sku} є незавершені запити на переміщення.
Зачекайте на їх підтвердження.',
                'pending_transfers': len(pending_transfers)
            }

        # Якщо потрібне переміщення, повертаємо повідомлення про необхідність
підтвердження
        if availability_result.get('requires_transfer', False):
            return {
                'success': False,
                'message': f'Недостатньо товару в місцях відбору.
{availability_result.get("message", "")}',
                'requires_transfer': True,
                'available': availability_result.get('available', 0),
                'required': availability_result.get('required', target_box_count)
            }

        # Оновлюємо доступну кількість
        available_boxes = availability_result.get('available', available_boxes)

    # Якщо доступна кількість менша за запитану, відбираємо те, що є
    actual_box_count = min(target_box_count, available_boxes)

    # Якщо немає доступних ящиків взагалі, повертаємо помилку
    if actual_box_count == 0:
        return {
            'success': False,
            'message': 'Немає доступного товару для відбору'
        }

    # Виконуємо відбір товару
    remaining_to_pick = actual_box_count
    picked_items = []

    # Зберігаємо інформацію про запитану кількість для логування
    requested_box_count = target_box_count
```

```
    total_picked_weight = 0
    total_picked_boxes = 0

    for item in picking_inventory:
        if remaining_to_pick <= 0:
            break

        pick_count = min(remaining_to_pick, item.box_count)

        # Розраховуємо вагу відібраного товару
        calculated_weight = (item.net_weight / item.box_count) * pick_count if
item.box_count > 0 else 0

        # Зменшуємо кількість товару в комірці
        item.box_count -= pick_count
        item.net_weight -= calculated_weight

        # Зберігаємо інформацію про відбір
        location_code =
StorageLocation.query.get(item.storage_location_id).location_code

        picking_item = OrderPickingItem(
            order_id=order_id,
            sku=sku,
            sscc=item.sscc,
            picked_box_count=pick_count,
            calculated_weight=calculated_weight,
            actual_quantity=calculated_weight if logistics.packaging_unit_type == 'КГ'
else pick_count,
            packaging_unit_type=logistics.packaging_unit_type,
            picked_by_user_id=user_id,
            storage_location=location_code,
            lot_number=lot_number
        )

        # Виводимо діагностичну інформацію про створений запис
        print(f"Створено запис відбору: SKU={sku}, lot_number={lot_number},
кількість={pick_count}")

        db.session.add(picking_item)
        # Зберігаємо зміни одразу, щоб уникнути проблем з транзакціями
        db.session.flush()
        picked_items.append({
            'sscc': item.sscc,
```

```
            'location': location_code,
            'box_count': pick_count,
            'weight': calculated_weight
        })

        total_picked_weight += calculated_weight
        total_picked_boxes += pick_count
        remaining_to_pick -= pick_count

    # Розраховуємо фактичну кількість відібраного товару
    actual_picked_quantity = total_picked_boxes

    # Якщо не вдалося відібрати жодного ящика, повертаємо помилку
    if total_picked_boxes == 0:
        return {
            'success': False,
            'message': 'Не вдалося відібрати жодного ящика товару'
        }

    # Перевіряємо чи всі товари в замовленні зібрані
    all_items_picked = True
    underpicked = False

    # Отримуємо тільки товари з поточного лота
    lot_items = [item for item in order.items if item.lot_number == lot_number]
    print(f"Перевіряємо товари лота {lot_number}, знайдено {len(lot_items)} товарів")

    for item in lot_items:
        # Підраховуємо загальну кількість відібраних ящиків для цього SKU в цьому лоті
        picked_count =
db.session.query(db.func.sum(OrderPickingItem.picked_box_count)).filter(
            OrderPickingItem.order_id == order_id,
            OrderPickingItem.sku == item.sku,
            OrderPickingItem.lot_number == lot_number
        ).scalar() or 0

        print(f"Перевірка товару {item.sku} в лоті {lot_number}: потрібно
{item.quantity}, відібрано {picked_count}")

        # Якщо відібрано менше ніж потрібно, то не всі товари зібрані
        if picked_count < item.quantity:
            all_items_picked = False
            # Якщо це поточний товар і відібрано менше ніж запитано, фіксуємо недобір
            if item.sku == sku and actual_box_count < requested_box_count:
```

```
                underpicked = True
            break

    # Перевіряємо всі лоти цього замовлення
    all_lots = OrderLot.query.filter_by(order_id=order_id).all()
    all_lots_packed = all(lot.status == 'packed' for lot in all_lots)

    if all_lots_packed:
        order.status = 'packed'
    db.session.add(order)

    # Виводимо діагностичну інформацію перед збереженням змін
    print(f"Зберігаємо зміни в базі даних. Відібрано {total_picked_boxes} ящиків товару
{sku} для лота {lot_number}")

    # Зберігаємо всі зміни в базі даних
    db.session.commit()

    # Перевіряємо, чи зберігся запис в базі даних
    verification =
db.session.query(db.func.sum(OrderPickingItem.picked_box_count)).filter(
        OrderPickingItem.order_id == order_id,
        OrderPickingItem.sku == sku,
        OrderPickingItem.lot_number == lot_number
    ).scalar() or 0

    print(f"Перевірка після збереження: для SKU {sku} в лоті {lot_number} відібрано
{verification} ящиків")

    all_lot_items = OrderItem.query.filter_by(order_id=order_id,
lot_number=lot_number).all()
    lot_fully_picked = True
    for item in all_lot_items:
        picked_count =
db.session.query(db.func.sum(OrderPickingItem.picked_box_count)).filter(
            OrderPickingItem.order_id == order_id,
            OrderPickingItem.sku == item.sku,
            OrderPickingItem.lot_number == lot_number
        ).scalar() or 0
        if picked_count < item.quantity:
            lot_fully_picked = False
            break

    if lot_fully_picked:
```

```
        lot = OrderLot.query.filter_by(order_id=order_id,
lot_number=lot_number).first()
        if lot:
            lot.status = 'packed'
            lot.completed_at = datetime.utcnow()

            for item in all_lot_items:
                picked_count =
db.session.query(db.func.sum(OrderPickingItem.picked_box_count)).filter(
                    OrderPickingItem.order_id == order_id,
                    OrderPickingItem.sku == item.sku,
                    OrderPickingItem.lot_number == lot_number
                ).scalar() or 0

                total_weight =
db.session.query(db.func.sum(OrderPickingItem.calculated_weight)).filter(
                    OrderPickingItem.order_id == order_id,
                    OrderPickingItem.sku == item.sku,
                    OrderPickingItem.lot_number == lot_number
                ).scalar() or 0

                logistics = LogisticsItemData.query.filter_by(sku=item.sku).first()
                product_name = item.product_name or (logistics.product_name if
logistics else item.sku)

                db.session.add(OrderLotSKU(
                    order_lot_id=lot.id,
                    sku=item.sku,
                    product_name=product_name,
                    quantity=picked_count,
                    weight=total_weight
                ))

            lot.box_count = sum(s.quantity for s in lot.sku_details)
            lot.total_weight = sum(s.weight for s in lot.sku_details)

            db.session.add(lot)
            db.session.commit()

    return {
        'success': True,
        'message': 'Товар успішно відібрано' + (' (частково)' if underpicked else ''),
        'picked_items': picked_items,
```

```
        'all_items_picked': all_items_picked,
        'requested_quantity': requested_quantity,
        'actual_quantity': actual_picked_quantity,
        'underpicked': underpicked,
        'is_last_item': lot_fully_picked
    }
```

7. Генерація SSCC за стандартом GS1

```
def generate_sscc():
    """ Згенерувати код SSCC та етикетку для вагової станції"""
    data = request.get_json()
    weight = data.get('weight')

    if not weight:
        return jsonify({'success': False, 'message': 'Вага не вказана'})

    try:
       Розбираємо вагу, щоб переконатися, що це дійсне число
        weight_value = float(weight)
        weight_formatted = f"{weight_value:.2f}"

        # Генерація SSCC коду
        company_prefix = '00381000000000'
        serial_number = random.randint(10000, 99999)
        sscc_without_check = f"{company_prefix}{serial_number}"

        # Обчислення контрольної цифри (алгоритм GS1)
        sum_value = 0
        for i, digit in enumerate(sscc_without_check):
            sum_value += int(digit) * (3 if i % 2 == 0 else 1)
        check_digit = (10 - (sum_value % 10)) % 10

        # Повний SSCC
        sscc = f"{sscc_without_check}{check_digit}"

        # Отримати поточну дату та час
        now = datetime.now()
        date_str = now.strftime('%d.%m.%Y')
        time_str = now.strftime('%H:%M')
        date_time_str = f"{date_str} {time_str}"

        # Збереження SSCC та вагу в базі даних
        new_sscc = SSCCBarcode(
```

```
            sscc=sscc,
            gross_weight=weight_value
        )
        db.session.add(new_sscc)
        db.session.commit()

        return jsonify({
            'success': True,
            'sscc': sscc,
            'weight': weight_formatted,
            'date_time': date_time_str
        })

    except ValueError:
        return jsonify({'success': False, 'message': 'Некоректне значення ваги'})
```

8. Початок приймання

```
def start_receiving():
    """Розпочати процес приймання замовлення"""
    data = request.get_json()

    if not data or 'invoice_number' not in data:
        return jsonify({
            'success': False,
            'message': 'Неповні дані'
        }), 400

    invoice_number = data['invoice_number']
    invoice_number_supplier = data.get('invoice_number_supplier', '')
    receiving_date = data.get('receiving_date', datetime.now().strftime('%Y-%m-%d'))

    # Знайти замовлення за номером
    order_item = OrderItem.query.filter_by(invoice_number=invoice_number).first()
    if not order_item:
        return jsonify({
            'success': False,
            'message': f'Замовлення з номером накладної {invoice_number} не знайдено'
        }), 404

    order = order_item.order

    # Перевірка статусу
    if order.status == 'created':
```

```
        order.status = 'processing'
        order.started_by_user_id = current_user.id  # фіксуємо хто почав обробку
        order.updated_at = datetime.now()

    elif order.status == 'processing':
        if order.started_by_user_id != current_user.id:
            return jsonify({
                'success': False,
                'message': f"Замовлення {order.order_number} вже приймається іншим
користувачем."
            }), 403  # Заборонено
    # Якщо current_user — той самий, хто вже обробляє, то дозволяємо

    else:
        return jsonify({
            'success': False,
            'message': f"Замовлення {order.order_number} має статус '{order.status}' і
не може бути оброблене."
        }), 400


    if invoice_number_supplier:
        order.invoice_number_supplier = invoice_number_supplier

    db.session.commit()

    return jsonify({
        'success': True,
        'message': f'Процес прийому для замовлення {order.order_number} розпочато',
        'order': {
            'id': order.id,
            'invoice_number': invoice_number,
            'order_number': order.order_number,
            'supplier_name': order.supplier.name if order.supplier else 'Unknown'
        }
    })
```

СПИСОК ВИКОРИСТАНОЇ ЛИТЕРАТУРИ

1. What is a WMS system? Ukrainian Intelligent Systems. URL: https://uislab.com/uk/chto-takoe-wms-sistema/ (date of access: 22.05.2025)
2. What is a ERP. SAP. URL: https://www.sap.com/ukraine/products/erp.html (date of access: 22.05.2025)
3. Economic Order Quantity (EOQ): How to optimize purchasing logistics in the network? Datawiz Inc. URL: https://datawiz.io/uk/blog/economic-order-quantity-eoq-how-to-optimize-chain-purchasing-logistics (date of access: 22.05.2025)
4. Kochubei Dmytro V. Evaluation of the Effect of the Introduction of Logistics Information Systems, 2014. URL: https://mail.business-inform.net/export_pdf/business-inform-2014-6_0-pages-228_232.pdf (date of access: 22.05.2025)
5. Christopher, M. Logistics and Supply Chain Management. Pearson Education Limited, 2016.
6. Nahmias, S. Production and Operations Analysis. Waveland Press, 2013.
7. Just-in-Time Manufacturing. Encyclopaedia Britannica. URL: https://www.britannica.com/topic/just-in-time-manufacturing (date of access: 22.05.2025)
8. Технологія "Vendor-managed Inventory - VMI" URL: https://stud.com.ua/49103/logistika/tehnologiya_vendor_managed_inventory (date of access: 22.05.2025)
9. ABC та XYZ аналіз у Торгсофт. URL: https://torgsoft.ua/articles/gid-po-torgsoft/abc-i-xyz-analiz-v-torgsoft/ (date of access: 22.05.2025)
10. What is AI inventory management? IBM Supply Chain Blog. URL: https://www.ibm.com/think/topics/ai-inventory-management (date of access: 22.05.2025)
11. ERP WMS Integration Explained: Benefits, Methods & Challenges. DCKAP. URL: https://www.dckap.com/blog/erp-wms-integration/ (date of access: 22.05.2025)

12. Data Collection Terminals in Warehousing. Zebra Technologies. URL: https://www.zebra.com/ (date of access: 22.05.2025)
13. Best Practices For Integrating Warehouse Management With ERP. DCKAP. URL: https://www.dckap.com/commerce/blog/best-practices-for-integrating-warehouse-management-with-erp/ (date of access: 22.05.2025)
14. ERP Integration: Strategy, Benefits and Best Practices. NetSuite. URL: https://www.netsuite.com/portal/resource/articles/erp/erp-integration-strategy.shtml (date of access: 22.05.2025)
15. Portable Data Collection Terminal (PDT) Definition. Speed Commerce. URL: https://www.speedcommerce.com/what-is/portable-data-collection-terminal/ (date of access: 22.05.2025)
16. Tableau Overview – Business Intelligence and Analytics Platform URL: https://www.tableau.com/ (date of access: 22.05.2025)
17. IT-enterprise – Supply Chain Management, SCM URL: https://www.it.ua/knowledge-base/technology-innovation/supply-chain-management-scm (date of access: 22.05.2025)
18. Освіта.Юа – Логістичні ланцюги: розподільчі центри. URL: https://osvita.ua/vnz/reports/management/14578/ (date of access: 22.05.2025)
19. Unleashed – Inventory Management Systems. URL: https://www.unleashedsoftware.com/inventory-management-guide/inventory-management-systems/ (date of access: 22.05.2025)
20. IT-enterprise – Business Intelligence, BI. URL: https://www.it.ua/knowledge-base/technology-innovation/business-intelligence-bi (date of access: 22.05.2025)
21. DOU – Типи баз даних: особливості, відмінності та приклади. URL: https://dou.ua/lenta/articles/types-of-databases/ (date of access: 22.05.2025)
22. Astera – SQL vs. NoSQL: 5 Main Differences. URL: https://www.astera.com/knowledge-center/sql-vs-nosql/ (date of access: 22.05.2025)
23. Iterator15 – Про ТЗД термінали збору даних у питаннях та відповідях. URL: https://www.iterator.com.ua/ua/poleznye-materialy/231-pro-tzd-terminali-zboru-

danikh-u-pitannyakh-ta-vidpovidyakh?srsltid=AfmBOoriDDkOjCrkrbMjXV_EK-HSwJNxuLRxQj0SFz0OobAFoWLu2eZZ (date of access: 22.05.2025)

24. Choi, T.-M., Wallace, S.W., & Wang, Y. (2018). Big Data Analytics in Operations Management. Production and Operations Management, 27(10), 1868–1889. URL: https://journals.sagepub.com/doi/10.1111/poms.12838 (date of access: 22.05.2025)
25. Amazon Web Services (AWS). Implementing Amazon Forecast in the retail industry: A journey from POC to production. URL: https://aws.amazon.com/ru/blogs/machine-learning/tag/demand-forecasting/ (date of access: 22.05.2025)
26. Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The M4 Competition: Results, findings, conclusion and way forward. International Journal of Forecasting, 36(1), 54–74. URL: https://www.sciencedirect.com/science/article/pii/S0169207019301128?via%3Dihub (date of access: 22.05.2025)
27. Zoho Inventory – Reorder Point – Definition, Formula & Importance. URL: https://www.zoho.com/inventory/academy/inventory-management/what-is-a-reorder-point.html (date of access: 22.05.2025)
28. Zoho Inventory – What is Safety Stock? URL: https://www.zoho.com/inventory/academy/inventory-management/what-is-safety-stock.html (date of access: 22.05.2025)