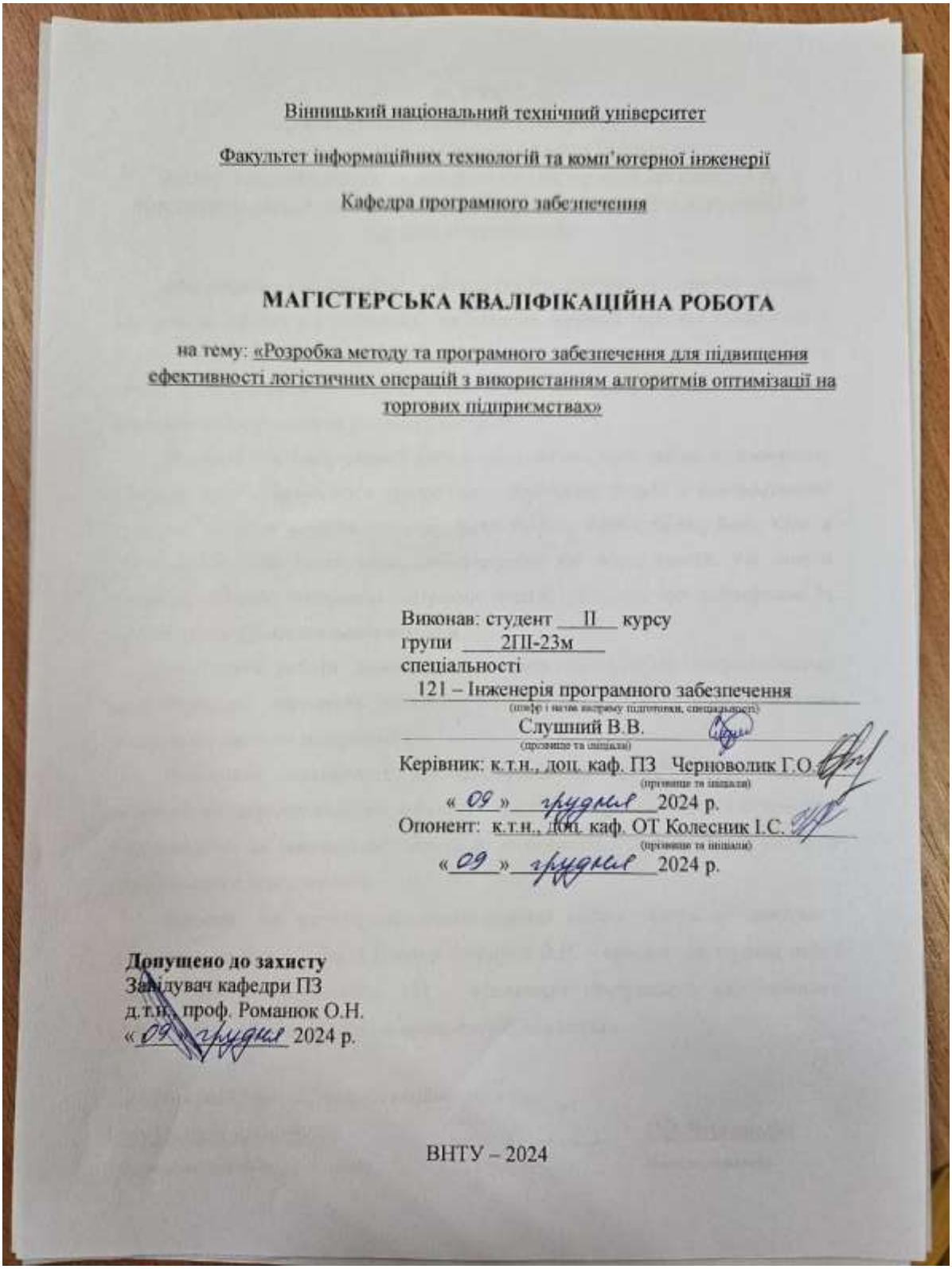


Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка методу та програмного забезпечення для підвищення ефективності логістичних операцій з використанням алгоритмів оптимізації на торгових підприємствах»

Виконав: студент II курсу
групи 2ПІ-23м
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Слушний В.В.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Черноволик Г.О.
(прізвище та ініціали)
«09» грудня 2024 р.
Опонент: к.т.н., доц. каф. ОТ Колесник І.С.
(прізвище та ініціали)
«09» грудня 2024 р.

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О.Н.
«09» грудня 2024 р.

ВНТУ – 2024

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти II-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
«18» вересня 2024 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Слушному Василю Віталійовичу

1. Тема роботи – Розробка методу та програмного забезпечення для підвищення ефективності логістичних операцій з використанням алгоритмів оптимізації на торгових підприємствах.

Керівник роботи: Черноволик Галина Олександрівна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від « 17 » вересня 2024 р. № 310.

2. Строк подання студентом роботи
3 грудня 2024 р.

3. Вихідні дані до роботи: середовища розробки Intellij Idea та Sublim Text, мови програмування Java, Kotlin, JavaScrip, лінійне програмування, генетичні алгоритми.

4. Зміст розрахунково-пояснювальної записки: вступ, обґрунтування вибору методу аналізу, моделі та методи підвищення ефективності логістичних операцій, запропоновані метод та модель підвищення ефективності логістичних операцій на торгових підприємствах, розробка та впровадження програмних модулів, особливості та етапи тестування систем з управління криптовалютними операціями, економічна частина, висновки.

5. Перелік графічного матеріалу: вступ, актуальність теми, аналіз задачі необхідності аналізу основних підходів, запропоновані метод та модель підвищення ефективності логістичних операцій на торгових підприємствах,

розробка та впровадження програмних модулів, особливості та етапи тестування систем з управління криптовалютними операціями, економічна частина, висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Черноволик Г. О., к.т.н., доцент кафедри ПЗ	19.09.2024	02.12.2024
5	Причепа І. В., к.е.н., доцент кафедри ЕПВМ	15.11.2024	30.11.2024

7. Дата видачі завдання 19 вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз задачі дослідження напрямків моделей генетичного програмування для виконання задач логістики	20.09.2024-28.09.2024	вик.
2	Дослідження моделей та методів ефективності логістичних операцій	29.03.2024-11.10.2024	вик.
3	Розробка та впровадження програмних модулів	12.10.2024-24.10.2024	вик.
4	Особливості та тестування етапів тестування систем з управління криптовалютними операціями	24.10.2024-15.11.2024	вик.
5	Економічна частина	15.11.24-30.11.24	вик.

Студент _____ **Слушний В.В.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Черноволик Г.О.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 658.7:004.78

Слушний В. В. Розробка методу та програмного забезпечення для підвищення ефективності логістичних операцій з використанням алгоритмів оптимізації на торгових підприємствах : магістерська кваліфікаційна робота зі спеціальності 121 Інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2024. 108 с.

На укр. мові. Бібліогр. : 36 назв ; рис. : 18; табл. 18.

У магістерській кваліфікаційній роботі розглядались особливості використання оптимізаційних алгоритмів управління логістичними ланцюгами постачання. Для цього, були визначені основні напрямки моделей генетичного програмування для виконання задач логістики. Також, під час аналізу проводилась порівняльна характеристика моделей оптимізації логістичних процесів з використанням нейронних мереж. На основі проведення порівняльних характеристик моделей та методів підвищення ефективності логістичних операцій були запропоновані метод та модель підвищення ефективності логістичних операцій на торгових підприємствах. Також в процесі роботи були створені функціональні вимоги та розробка основних модулів програмного забезпечення та проведені їх тестування. Для цього, були визначені етапи проведення тестування систем з управління криптовалютними логістичними операціями та наведено контрольний приклад використання логістичних операцій на торгівельних підприємствах.

Ключові слова: логістичні операції, алгоритми оптимізації, моделі генетичного програмування, оптимізація управління запасами, розробка та тестування програмного забезпечення.

ABSTRACT

Slushnyi V.V. Development of a method and software to improve the efficiency of logistics operations using optimization algorithms in retail enterprises : master's qualification thesis on the specialty 121 Software engineering, educational program - software engineering. Vinnytsia: VNTU, 2024. 108 pages.

In Ukrainian speech Bibliogr. : 36 titles; Fig. : 18; table 18.

The master's thesis considered the features of using optimization algorithms in managing logistics supply chains. The main directions of genetic programming models for performing logistics tasks were determined for this. Also, a comparative characteristic of models for optimizing logistics processes using neural networks was carried out during the analysis. Based on the comparative characteristics of models and methods for increasing the efficiency of logistics operations, a method and model for increasing the efficiency of logistics operations at trading enterprises were proposed. Functional requirements were created during work, and the leading software modules were developed and tested. For this, the stages of systems testing for managing cryptocurrency logistics operations were determined, and a control example of the use of logistics operations at trading enterprises was given.

Keywords: logistics operations, optimization algorithms, genetic programming models, inventory management optimization, software development and testing.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. У сучасному діловому світі торгові підприємства значною мірою покладаються на ефективні логістичні операції, щоб забезпечити своєчасну доставку товарів і послуг. Тому, ефективне управління логістикою має вирішальне значення для збереження конкурентних переваг, зниження операційних витрат і забезпечення своєчасного надання послуг. Однак традиційні підходи часто не відповідають вимогам сучасних торговельних підприємств через їх нездатність ефективно керувати великомасштабними даними або адаптуватися до мінливих умов ринку. Також, багатьом підприємствам потрібна допомога в оптимізації цих операцій [1,2]. Нехтування основними положеннями логістичних операцій призводить до неефективності у діях підприємств, наприклад збільшення транспортних витрат, затримок у ланцюжку постачання та неоптимального використання ресурсів. Ця неефективність негативно впливає на загальну прибутковість підприємств і конкурентоспроможність на ринку праці. Внаслідок цього настає швидке зростання помилок в роботі торговельних підприємств і дедалі складніший ланцюжок постачання створюють значний тиск на логістичні операції, вимагаючи від компаній пошуку нових стратегій для оптимізації своїх процесів.

Рішення цих проблем полягають у використанні наукової методології дослідження, яка передбачає багатоетапний підхід, що поєднує теоретичну та практичну складові. Перший етап повинен бути зосереджений на розробці алгоритмів оптимізації з урахуванням конкретних потреб логістичних операцій на підприємствах торгівлі. Ці алгоритми базуються на моделях лінійного програмування для оптимізації розподілу ресурсів і евристичних алгоритмах для вирішення складних проблем маршрутизації та планування [1].

Таким чином, для досягнення поставлених цілей у дослідженні необхідно використовувати поєднання методологій математичного моделювання та програмної інженерії. Тому, основний підхід у роботах повинен передбачати розробку оптимізаційної моделі за допомогою методів лінійного програмування (LP) і змішаного цілочисельного програмування (MIP). Ці методи добре

підходять для вирішення складних логістичних проблем, які включають численні змінні рішення та обмеження, такі як мінімізація витрат на транспортування та складування, оптимізація графіків доставки та забезпечення ефективного розподілу ресурсів. Лінійне програмування вибрано тому, що воно ефективно вирішує проблеми розподілу, коли такі ресурси, як транспортні засоби, персонал і ємності для зберігання, повинні бути оптимально розподілені. З іншого боку, евристичні алгоритми, використовуються для завдань маршрутизації та планування, які включають великі набори даних і динамічні змінні. Тому актуальною є розробка методів і засобів для підвищення ефективності логістичних операцій з використанням алгоритмів оптимізації на торгових підприємствах.

Мета та завдання дослідження:

Метою роботи є підвищення ефективності логістичних операцій на підприємствах торгівлі шляхом розробки програмного забезпечення з використанням алгоритмів оптимізації, що дозволить покращити процеси логістики та ефективно розподілити ресурси підприємства.

У відповідності до поставленої мети потрібно виконати такі **завдання**:

- обґрунтувати вибір методів аналізу основних підходів до оптимізації логістичних операцій на підприємствах торгівлі;

- розробити метод підвищення ефективності логістичних операцій на торгових підприємствах;

- розробити метод інтеграції засобів оптимізації в систему розрахунку логістичних операцій;

- визначити функціональні вимоги до основних програмних модулів розробленого програмного засобу;

- розробити програмний засіб системи;

- провести тестування програмних модулів на модельних та реальних даних.

Об’єктом дослідження є процес розробки програмного забезпечення для підвищення ефективності логістичних операцій на підприємствах торгівлі.

Предметом дослідження є методи та засоби розробки програмного забезпечення для підвищення ефективності логістичних операцій із використанням алгоритмів оптимізації.

Методи дослідження. У процесі дослідження використовувались:

- методи оптимізації (лінійне програмування, евристичні алгоритми) для вирішення проблем оптимізації, методи системного аналізу;
- методи комп'ютерного моделювання для створення складних нелінійних зв'язків;
- методи порівняльного аналізу для визначення переваг та недоліків алгоритмів оптимізації;
- методи статистичної обробки даних для виявлення прихованих взаємозв’язків та закономірностей різних факторів.

Наукова новизна одержаних результатів:

1. Подальшого розвитку отримав метод інтеграції засобів оптимізації в систему розрахунку логістичних операцій, який, на відміну від існуючих, поєднує можливість оптимізації за евристичними алгоритмами і можливість реалізації з використанням лінійного програмування, що дозволяє враховувати змінність параметрів у часі, таких як маршрутизація, управління запасами та розподіл ресурсів, і підвищити ефективність логістичних процесів.

2. Подальшого розвитку отримав метод оптимізації логістичних операцій, який, на відміну від існуючих, базується на комбінованому використанні методів мінімізації витрат на транспортування та зберігання, що дозволяє надавати підприємствам комплексний інструмент для ефективного управління всіма етапами логістики та забезпечує більш точне моделювання логістичних процесів зі скороченням витрат на логістику.

Практична цінність отриманих результатів. Практичне значення отриманих результатів полягає в тому, що розроблені в магістерській кваліфікаційній роботі методи та програмні засоби дозволять торговим підприємствам у реальному часі ефективно керувати логістичними процесами, включаючи маршрутизацію, управління запасами та розподіл ресурсів.

Особистий внесок здобувача. У магістерській кваліфікаційній роботі усі результати дослідження здобуті автором даної роботи самостійно. У роботі [1], опублікованій у співавторстві, автору належить розробка методу та програмних засобів для підвищення ефективності логістичних операцій на торгових підприємствах.

Апробація матеріалів магістерської кваліфікаційної роботи. Основні положення магістерської кваліфікаційної роботи доповідалися та обговорювалися на Всеукраїнській науково-технічній конференції молодих вчених, аспірантів та студентів "Комп'ютерні ігри і мультимедіа як інноваційний підхід до комунікації - 2024", (Одеса, 2024).

Публікації. Основні результати досліджень опубліковано в науковій праці [1] у матеріалах Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів "Комп'ютерні ігри і мультимедіа як інноваційний підхід до комунікації - 2024", (Одеса, 2024).

Структура та обсяг роботи. Магістерська кваліфікаційна робота складається зі вступу, п’яти розділів, висновків, списку використаних джерел, що містить 36 найменувань, 4 додатка. Робота містить 18 ілюстрацій, 18 таблиць.

1 ОБГРУНТУВАННЯ ВИБОРУ МЕТОДУ АНАЛІЗУ

1.1. Аналіз особливостей використання оптимізаційних алгоритмів управлінні логістичними ланцюгами постачання

У сучасних умовах розвитку торгівлі та електронної комерції ефективне управління логістичними ланцюгами постачання є ключовим чинником для досягнення конкурентних переваг торгових підприємств. Логістичні процеси включають планування, координацію та контроль руху товарів і інформації між постачальниками, виробниками, дистриб'юторами та кінцевими споживачами. Впровадження алгоритмів оптимізації в логістиці дозволяє значно підвищити продуктивність операцій, зменшити витрати на транспортування, зберігання, а також оптимізувати управління запасами [1,2].

Основні особливості використання оптимізаційних алгоритмів полягають в оптимізаційних процесах (рис. 1.1).

Рисунок 1.1 – Основні особливості використання оптимізаційних алгоритмів

Оптимізаційні алгоритми дозволяють вирішувати складні задачі управління логістикою, враховуючи велику кількість факторів, таких як мінливість попиту, складність транспортних маршрутів, умови зберігання товарів, час доставки, тощо. Наведено основні особливості використання оптимізаційних алгоритмів в

управлінні логістичними ланцюгами постачання на торгових підприємствах (рис. 1.1).

Алгоритми оптимізації маршрутів, такі як генетичні алгоритми (ГА) або алгоритми мурашиних колоній, використовуються для знаходження оптимальних шляхів транспортування товарів. Це дозволяє знизити витрати на перевезення, мінімізувати час доставки і збільшити рівень обслуговування клієнтів (табл. 1.1).

Алгоритми маршрутизації відіграють важливу роль в управлінні логістичними процесами на торгових підприємствах, оскільки вони дозволяють оптимізувати шляхи транспортування товарів від постачальників до кінцевих споживачів. Основна задача таких алгоритмів - знаходження найбільш ефективного маршруту для доставки товарів, який мінімізує витрати на транспортування, зменшує час виконання замовлень та максимізує ефективність використання транспортних засобів [3-5].

Таблиця 1.1 – Основні особливості використання оптимізаційних алгоритмів

Категорія	Опис
Алгоритми маршрутизації	Алгоритми, такі як генетичні алгоритми або алгоритми мурашиних колоній, для оптимізації шляхів транспортування товарів.
Оптимізація управління запасами	Стохастичні та детерміновані методи оптимізації для управління запасами з урахуванням ймовірнісних факторів попиту.
Алгоритми розподілу ресурсів	Алгоритми лінійного програмування та теорії черг для оптимізації використання транспортних засобів, складів та робочої сили.
Аналіз даних і прогностичні алгоритми	Алгоритми на базі ШІ та машинного навчання для прогнозування попиту та адаптації до змін ринку на основі історичних даних.
Розподіл вантажів і управління запасами	Оптимізаційні методи для визначення стратегій розподілу вантажів і управління складськими запасами для мінімізації витрат.

Алгоритм найближчого сусіда (Nearest Neighbor) є простим і ефективним жадібний алгоритм, який працює за принципом вибору наступного пункту маршруту, що знаходиться на найменшій відстані від поточного місцезнаходження. Хоча цей метод не гарантує глобально оптимального рішення, він часто використовується як початкове рішення для більш складних алгоритмів [4,5].

Метод динамічного програмування розв’язує складні задачі маршрутизації шляхом розбиття їх на простіші підзадачі. Такий підхід особливо корисний для задач із високою кількістю змінних і обмежень. Він дозволяє ефективно знаходити оптимальні рішення, проте вимагає значних обчислювальних ресурсів для зберігання та обробки проміжних результатів.

Переваги використання алгоритмів маршрутизації полягаютьв в оптимізації витрат на транспортування.

Правильний вибір маршруту дозволяє значно знизити витрати на доставку товарів, скорочуючи відстань і час перевезення [6-8].

Така оптимізація маршрутів дозволяє доставляти товари в найкоротші терміни, що підвищує рівень задоволеності клієнтів. Алгоритми маршрутизації дозволяють ефективно розподілити транспортні засоби і робочу силу, мінімізуючи простої та підвищуючи ефективність роботи підприємства.

Моделі управління запасами часто враховують ймовірнісні фактори попиту та строків постачання. Оптимізація управління запасами є однією з ключових складових успішного функціонування торгових підприємств. Вона полягає у визначенні найкращого рівня запасів, який забезпечує задоволення попиту, мінімізуючи при цьому витрати на зберігання, закупівлю та транспортування товарів. Ефективне управління запасами дозволяє уникнути дефіциту продукції, знизити витрати на надлишкові запаси та підвищити загальну рентабельність бізнесу [6].

Модель управління запасами за системою замовлень "з фіксованою кількістю" (Continuous Review System) Ця модель передбачає постійний

моніторинг рівня запасів, і замовлення нового товару здійснюється тоді, коли кількість продукції на складі знижується до заздалегідь визначеного рівня — так званого рівня перезамовлення. При досягненні цього рівня формується замовлення на фіксовану кількість товару. Цей підхід дозволяє своєчасно поповнювати запаси та уникати ситуацій дефіциту, особливо при нестабільному попиті [6-8].

Запас безпеки (Safety Stock) Запас безпеки — це додатковий рівень запасів, який зберігається на складі для покриття можливих коливань попиту або затримок у поставках. Оптимізація управління запасом безпеки дозволяє мінімізувати ризик дефіциту продукції, зберігаючи при цьому витрати на зберігання на прийнятному рівні. Розрахунок запасу безпеки враховує статистичні дані про попит і строк поставки, а також імовірнісні моделі, які дозволяють оцінити ризики непередбачених ситуацій.

Моделі управління запасами при змінному попиті В реальних умовах попит на продукцію рідко залишається постійним, що вимагає використання складніших моделей управління запасами. До таких моделей належать стохастичні моделі, які враховують випадкові зміни попиту і строків постачання. Вони дозволяють розраховувати ймовірності різних сценаріїв розвитку подій та відповідним чином коригувати стратегію управління запасами.

Переваги оптимізації управління запасами полягають у зменшені витрат на зберігання. Оптимізація рівня запасів дозволяє уникнути зайвих витрат на утримання великої кількості товарів на складі. Правильне управління запасами забезпечує наявність необхідної продукції у потрібний час, що сприяє підвищенню рівня задоволеності клієнтів. Оптимізація запасів дозволяє уникати ситуацій, коли товари відсутні на складі в момент, коли вони необхідні для виконання замовлень. Ефективне управління запасами сприяє поліпшенню роботи всієї логістичної системи підприємства, забезпечуючи безперервний процес постачання та зменшуючи затримки.

Лінійне програмування, що поширено використовується, є математичним методом, який використовується для вирішення задач оптимізації, де цільова

функція і обмеження виражені у вигляді лінійних рівнянь. Це один з найпоширеніших методів для розподілу ресурсів в логістиці. Лінійне програмування дозволяє знаходити оптимальний розподіл ресурсів для мінімізації витрат або максимізації прибутку з урахуванням обмежень, таких як доступність транспортних засобів, обсяги складу, час доставки тощо.

Формулювання задачі лінійного програмування включає:

- цільову функцію, яка визначає мету оптимізації (наприклад, мінімізація загальних витрат).

- обмеження, що відображають наявні ресурси і обмеження (наприклад, кількість доступних транспортних засобів, обсяги товарів на складі). Лінійне програмування може вирішувати задачі такого типу за допомогою симплекс-методу або інших алгоритмів.

Алгоритми теорії черг застосовуються для моделювання і оптимізації процесів, де ресурси (наприклад, транспортні засоби, склади) використовуються спільно для обслуговування кількох потоків замовлень. Вони дозволяють визначити найкращі стратегії для розподілу ресурсів у реальному часі, мінімізуючи затримки і простої. Задачі теорії черг можуть вирішуватися як у детермінованих, так і в стохастичних умовах, що дозволяє враховувати випадкові зміни попиту і доступності ресурсів. Тому, жадібні алгоритми (Greedy algorithms) використовуються для вирішення задач розподілу ресурсів за принципом поступового вибору найкращого варіанту на кожному кроці. Жадібні алгоритми можуть бути ефективними для розподілу обмежених ресурсів в умовах, коли рішення необхідно приймати швидко. Наприклад, в логістиці це може означати розподіл транспорту або складів для виконання найнагальніших замовлень [7,8].

Алгоритм імітації відпалу (Simulated Annealing) використовується для вирішення задач розподілу ресурсів в умовах складної багатоваріантної системи. Він базується на метафорі процесу охолодження металів, коли поступове зниження температури дозволяє знайти оптимальне рішення. Цей алгоритм

корисний у випадках, коли необхідно знайти глобальне оптимальне рішення, але існує ризик потрапляння в локальний мінімум.

Алгоритм імітації відпалу може використовуватися для оптимізації великих логістичних систем, де необхідно знайти найкраще рішення з багатьох можливих варіантів розподілу ресурсів. Основна перевага цього методу — його здатність уникати локальних мінімумів і шукати глобальне оптимальне рішення.

Алгоритми штучного інтелекту (ШІ) та машинного навчання (ML) використовуються для автоматизації процесів розподілу ресурсів і підвищення ефективності управління складними логістичними операціями. Машинне навчання дозволяє аналізувати великі масиви даних і прогнозувати майбутні потреби в ресурсах, базуючись на історичних даних. Алгоритми ШІ можуть автоматично приймати рішення про розподіл ресурсів на основі поточних умов і вимог. Однією з переваг застосування таких алгоритмів є їхня здатність постійно адаптуватися до змін у попиті, доступності ресурсів та інших факторів. Це дозволяє підприємствам бути більш гнучкими і ефективними у прийнятті рішень[8-10].

Переваги використання алгоритмів розподілу ресурсів полягаютьв ефективному використанні ресурсів. Алгоритми дозволяють максимально раціонально розподіляти обмежені ресурси, мінімізуючи простої та нераціональне використання. Правильний розподіл ресурсів дозволяє знизити операційні витрати, такі як витрати на транспорт, зберігання або робочу силу.

Алгоритми розподілу ресурсів є невід'ємною частиною управління логістичними процесами на торгових підприємствах. Використання таких методів, як лінійне програмування, жадібні алгоритми, динамічне програмування і алгоритми машинного навчання, дозволяє підвищити ефективність роботи, зменшити витрати і забезпечити конкурентоспроможність підприємства [9].

Основні концепції аналізу даних і прогностичних алгоритмів полягає у зборі і попередньої обробки даних. Тому, перед тим, як використовувати прогностичні алгоритми, необхідно зібрати та підготувати дані. Це можуть бути дані про

продажі, попит на товари, історія замовлень, рівні запасів, інформація про транспортні операції та інші показники. Попередня обробка даних включає очищення, нормалізацію, трансформацію та видалення аномальних значень, що може спотворити результати аналізу.

Для отримання коректних прогнозів дані мають бути повними та точними. Неякісні або неповні дані можуть призвести до хибних результатів і, відповідно, до неправильних управлінських рішень. При цьому, аналіз часових рядів є важливою складовою прогнозування попиту і логістичних операцій. Часові ряди відображають зміни попиту, продажів чи інших показників у часі. Прогностичні моделі, засновані на часових рядах, дозволяють передбачати майбутні тенденції та коливання попиту на основі попередніх даних.

Прогнозування попиту є одним із ключових застосувань прогностичних алгоритмів у логістиці. На основі історичних даних і поточних ринкових тенденцій підприємства можуть оцінити майбутній попит на свої товари та послуги. Це дозволяє їм планувати закупівлі, виробництво і управління запасами таким чином, щоб уникати надлишкових запасів або дефіциту товарів.

Однією з популярних моделей для прогнозування попиту є експоненційне згладжування (Exponential Smoothing). Ця модель надає більшу вагу недавнім даним, що дозволяє краще адаптувати прогноз до змін на ринку. Використання експоненційного згладжування є корисним для прогнозування короткострокових змін у попиті. Для цього досить часто, використовується кластеризація — це метод машинного навчання, який використовується для групування схожих об'єктів у кластери. У логістиці та управлінні запасами кластеризація може бути корисною для сегментації клієнтів або товарів на основі їх поведінки чи характеристик [11-13].

Аналіз асоціативних правил Аналіз асоціативних правил використовується для виявлення закономірностей у великих масивах даних. Наприклад, в логістиці можна використовувати цей метод для визначення товарів, які часто купуються разом. Це дозволяє оптимізувати управління запасами, формуючи більш ефективні стратегії для підготовки до попиту на конкретні групи товарів.

Переваги прогностичних алгоритмів та аналізу даних полягає у поліпшення точності прогнозів. Використання машинного навчання та аналізу часових рядів дозволяє отримувати точні прогнози попиту та операційних потреб підприємства.

Таким чином, аналіз даних і прогностичні алгоритми є потужними інструментами для підвищення ефективності управління логістикою і запасами на торгових підприємствах. Використання таких алгоритмів, як ARIMA, нейронні мережі, експоненційне згладжування та кластеризація, дозволяє забезпечити точне прогнозування попиту, покращити управління запасами та оптимізувати логістичні процеси. Інтеграція прогностичних моделей у бізнес-процеси підприємства дозволяє підвищити гнучкість і конкурентоспроможність на сучасному ринку.

Такі оптимізаційні методи застосовуються для визначення найкращих стратегій розподілу вантажів між різними складами і торговими точками. Використання таких алгоритмів сприяє більш точному управлінню складськими запасами, що дозволяє знижувати витрати на зберігання і уникати втрат через надмірні запаси або дефіцит [12-14].

Основні принципи розподілу вантажів полягають в оптимізації маршрутів, де одним з ключових завдань у розподілі вантажів є вибір оптимальних маршрутів для доставки товарів. Для цього використовуються різні алгоритми маршрутизації, такі як алгоритми мурашиних колоній або генетичні алгоритми, які допомагають знаходити найбільш економічно вигідні та швидкі маршрути для транспортування вантажів. Оптимізація маршрутів дозволяє скоротити витрати на паливо, зменшити час перевезення та підвищити ефективність використання транспортних засобів.

Консолідація вантажів полягає в об’єднанні декількох невеликих вантажів у один великий, щоб зменшити кількість транспортних рейсів і знизити загальні витрати на транспортування. Це особливо ефективно для підприємств, які доставляють товари в різні регіони або клієнтам із меншими обсягами замовлень.

Консолідація дозволяє більш ефективно використовувати транспортні засоби, знижуючи витрати на логістику.

Ефективне управління транспортними засобами включає розподіл автомобілів, вантажівок чи іншого транспорту для виконання перевезень. Це забезпечує баланс між кількістю транспортних засобів і обсягом вантажу, що транспортується. Використання систем управління транспортом (TMS) дозволяє автоматизувати процес розподілу вантажів і вибору оптимальних транспортних засобів для доставки.

Крос-докінг — це логістична техніка, що передбачає пряме переміщення вантажу з одного транспортного засобу на інший без необхідності тривалого зберігання на складі. Це дозволяє зменшити витрати на зберігання і прискорити доставку. Крос-докінг є ефективним методом для торгових підприємств, які прагнуть забезпечити швидкий обіг товарів і зменшити час їх доставки.

Запас безпеки — це додатковий рівень запасів, що утримується для захисту від несподіваних коливань попиту або затримок у постачанні. Оптимізація запасу безпеки дозволяє підприємству забезпечити безперервне постачання товарів і уникнути втрат від дефіциту, але при цьому мінімізувати витрати на зберігання[12-14].

Модель управління запасами з фіксованою кількістю замовлення (Continuous Review System) передбачає постійний контроль за рівнем запасів, і нове замовлення здійснюється тоді, коли кількість товару досягає заздалегідь визначеного рівня перезамовлення. Ця модель дозволяє підприємствам постійно контролювати запаси і своєчасно поповнювати їх. Модель управління запасами з фіксованим інтервалом замовлення (Periodic Review System) передбачає перевірку рівня запасів через регулярні проміжки часу і формування замовлення на поповнення запасів до певного рівня. Такий підхід дозволяє зменшити частоту перевірки запасів і знизити адміністративні витрати.

Переваги ефективного розподілу вантажів та управління складськими запасами полагають в зменшені витрат на транспортування і зберігання. Оптимізація розподілу вантажів і управління запасами дозволяє знизити загальні

витрати на логістичні операції. Правильний розподіл транспортних засобів і ефективне управління запасами сприяють зменшенню простоїв і покращенню використання складських площ.

Використання моделей управління запасами дозволяє передбачати та компенсувати непередбачені зміни у попиті чи строках постачання.

Тому, оптимізаційні алгоритми відіграють критично важливу роль в управлінні логістичними ланцюгами постачання на торгових підприємствах, дозволяючи суттєво підвищити ефективність процесів та знизити операційні витрати. Основні переваги включають покращення маршрутизації, управління запасами, розподілу ресурсів і прогнозування попиту.

Для успішної імплементації таких алгоритмів необхідно забезпечити доступ до високоякісних даних, а також інвестувати в обчислювальні потужності та персонал з відповідними компетенціями. Це дозволить підприємствам ефективно адаптуватися до мінливих ринкових умов і забезпечити стабільність та надійність логістичних операцій [12-14].

Таким чином, впровадження оптимізаційних алгоритмів є стратегічно важливим кроком для сучасних торгових підприємств, які прагнуть залишатися конкурентоспроможними в умовах глобалізованої економіки.

1.2. Аналіз основних напрямків моделей генетичного програмування для виконання задач логістики

Підвищення ефективності логістичних операцій на торгових підприємствах є важливим аспектом управління ресурсами, оскільки логістика безпосередньо впливає на швидкість обробки замовлень, зменшення витрат та підвищення рівня задоволеності клієнтів. Одним із найбільш перспективних методів оптимізації є використання генетичних алгоритмів, що дозволяють знаходити оптимальні рішення в складних багатовимірних середовищах. Це особливо актуально для великих торгових підприємств, які мають справу з великими обсягами даних та процесів [12,14-16]. Огляд моделей генетичного програмування для логістики повинен включати базові напрямки (рис. 1.2).

Рисунок 1.2 – Огляд основних напрямків моделей генетичного програмування для логістики

Генетичні алгоритми (ГА) є частиною еволюційних обчислювальних методів, що використовуються для вирішення оптимізаційних завдань. Вони імітують процес природного відбору та є ефективними для знаходження оптимальних рішень в умовах великих обчислювальних складностей. Зокрема, ГА можуть бути застосовані до оптимізації логістичних операцій, включаючи такі завдання, як маршрутизація транспорту, управління запасами, планування перевезень та мінімізація логістичних витрат.

Тому, одним із найважливіших завдань логістики є оптимізація маршрутів доставки товарів. В умовах торгових підприємств це особливо актуально для забезпечення своєчасної доставки та зниження витрат на паливо. Генетичні алгоритми дозволяють знаходити рішення для складних маршрутів, що враховують численні обмеження, такі як:

- ємність транспортних засобів;
- часові вікна доставки;
- вартість транспортування;
- географічні обмеження.

Модель, що заснована на генетичному алгоритмі (табл. 1.2) для маршрутизації транспортних засобів, передбачає генерацію початкової популяції можливих рішень (маршрутів), еволюцію цих рішень через селекцію, кросовер та мутацію, і, нарешті, вибір оптимального рішення.

Таблиця 1.2 – Опис задач для моделей генетичного програмування для логістики

Модель	Опис задачі
Маршрутизація транспортних засобів	Оптимізація маршрутів доставки з урахуванням ємності транспортних засобів, часових вікон та витрат на транспортування.
Управління запасами	Оптимізація рівня запасів і термінів постачання для мінімізації витрат на зберігання і транспортування.
Планування перевезень	Оптимізація графіків і маршрутів перевезень з урахуванням наявних ресурсів та потреб клієнтів.

На торгових підприємствах важливо забезпечити безперервну наявність товарів, водночас мінімізуючи витрати на зберігання. Генетичні алгоритми можуть допомогти оптимізувати процес управління запасами, визначаючи оптимальний рівень замовлень та розміри партій поставок. Вони можуть використовуватися для визначення оптимальної кількості товарів для замовлення. Тому, мінімізації вартості зберігання і транспортних витрат. ГА допомагають управляти запасами з урахуванням сезонних коливань, попиту та пропозиції, забезпечуючи баланс між витратами на зберігання і вчасністю постачання [15,16].

1.3. Варіантний аналіз аналогів та інструментів розробки

Щоб підкреслити переваги запропонованого рішення, важливо порівняти його з існуючими аналогами, такими як SAP SCM, Oracle Transportation Management, LogiNext Mile та Manhattan Associates TMS.

SAP SCM є комплексною системою для управління ланцюгами постачання, яка пропонує широкі можливості з оптимізації маршрутів, управління запасами та прогнозування попиту. Її головними перевагами є інтеграція з ERP-системами та висока надійність, але основним недоліком залишається висока вартість впровадження та тривалий період навчання персоналу. У порівнянні з SAP SCM, запропоноване рішення є значно доступнішим, оскільки орієнтоване на середній бізнес і не вимагає великих витрат на впровадження.

Oracle Transportation Management спеціалізується на управлінні транспортом і маршрутизації, зосереджуючи увагу на інтеграції з іншими модулями Oracle. Попри високу масштабованість і функціональність, ця система має складний процес налаштування і високі вимоги до інфраструктури. Запропоноване рішення вирізняється простотою налаштування та зручністю використання, що робить його доступним для невеликих і середніх торгових підприємств.

LogiNext Mile орієнтована на автоматизацію процесів доставки та планування маршрутів із підтримкою інтеграції з GPS. Її перевагами є швидка інтеграція та сучасний інтерфейс, однак обмежені функції прогнозування та висока залежність від точності вхідних даних є суттєвими недоліками. У цьому аспекті запропоноване рішення забезпечує точність прогнозів завдяки використанню історичних даних та алгоритмів машинного навчання.

Manhattan Associates TMS пропонує високу функціональність для великих підприємств із можливістю оптимізації складів і управління транспортними операціями. Однак ця система не підходить для малого та середнього бізнесу через складність та високу вартість. Натомість розроблене рішення адаптоване для середніх торгових підприємств із швидким впровадженням і мінімальними вимогами до ресурсів.

Таким чином, запропоноване рішення є інноваційним та доступним інструментом, який включає оптимізацію маршрутів, управління запасами, розподіл ресурсів і вантажів, а також аналіз даних і прогнозування. Воно

вирізняється своєю модульною архітектурою, що дозволяє адаптувати функціонал під конкретні потреби підприємства, і сучасними алгоритмами, які забезпечують високу ефективність та економічність.

Проведемо порівняння середовищ розробки

IntelliJ IDEA. IntelliJ IDEA — це одна з найпотужніших інтегрованих середовищ розробки (IDE), розроблених для створення програмного забезпечення різними мовами, зокрема Java, Kotlin, Python та іншими.

Серед переваг IntelliJ IDEA виділимо:

- широкий функціонал: підтримку автодоповнення коду, рефакторинг, інтеграцію з системами контролю версій (Git);
- модульність – можливість додавання плагінів для роботи з алгоритмами оптимізації (наприклад, бібліотеки для Python чи Java);
- інтеграцю з Maven/Gradle – спрощує керування залежностями проекту;
- інструменти для аналізу продуктивності, які допомагають оптимізувати програму;
- потребу у навчанні для новачків.

Результати порівняння Intellej Idea та його аналогів наведені у таблиці 1.3.

Sublime Text. Sublime Text — текстовий редактор, що підтримує багато мов програмування і є легким у використанні.

Серед переваг Sublime Text виділимо:

- швидкодію, яка забезпечується завдяки низьким вимогам до апаратного забезпечення;
- простоту використання завдяки інтуїтивноому інтерфейсу;
- гнучкість та великий набір плагінів, які можна встановити через Package Control (наприклад, інтеграція з Git, синтаксис для Python чи Java);
- кросплатформеність – доступність для Windows, macOS, Linux;
- низьку вартість (безкоштовний доступ з обмеженим функціоналом або ліцензія за разову оплату).

Серед недоліків Sublime Text виділимо:

- відсутність вбудованого компілятора;

Серед інших засобів можна назвати:

- Eclipse – альтернативна IDE для Java з багатим функціоналом, але менш інтуїтивним інтерфейсом;
- PyCharm – потужна IDE для Python, проте специфічна для роботи тільки з однією мовою;
- Visual Studio Code – багатофункціональний редактор, але потребує часу для налаштування плагінів під конкретні задачі.

Таблиця 1.3 – Порівняння IntelliJ IDEA та його аналогів

Критерій	IntelliJ IDEA	Eclipse	Visual Studio Code
Основна мова програмування	+ (Java, Kotlin)	+ (Java)	+ (Java)
Інтерфейс користувача	+ (зручний)	-	+ (сучасний)
Швидкодія	+	+	-
Інтеграція з DevOps	+	+	+
Рефакторинг коду	+ (розвинений)	- (середній)	+ (через плагіни)

Таким чином, згідно з наведеними вище даними, Intellij Idea має вищий сумарний коефіцієнт, ніж Eclipse, Visual Studio Code. Тому IntelliJ IDEA найкраще підходить для поставленого завдання.

Результати порівняння Sublime Text та його аналогів зведено у таблицю1.4.

Таблиця 1.4 – Порівняльна таблиця Sublime Text та його аналогів

Критерій	Sublime Text	Notepad++	Atom
Тип інструменту	+	+	+
Швидкодія	+	+	-

Продовження таблиці 1.4

Критерій	Sublime Text	Notepad++	Atom
Модульність (плагіни)	+ (багато)	-	+ (багато)
Підтримувані мови	+ (різні)	+ (різні)	+ (різні)
Інтерфейс користувача	+ (сучасний)	-	+ (сучасний)
Робота з великими файлами	+	+	-

Таким чином, згідно з наведеними вище даними, Sublime Text має вищий сумарний коефіцієнт, ніж Notepad++, Atom. Тому Sublime Text найкраще підходить для поставленого завдання.

1.4. Постановка задач дослідження

Магістерська робота повинна бути присвячена розробці методу та програмного забезпечення на основі алгоритмів оптимізації. Для цього потрібно виконати такі завдання: обґрунтувати вибір методів аналізу основних підходів до оптимізації логістичних операцій на підприємствах торгівлі; виконати порівняльну характеристику існуючих методів та алгоритмів оптимізації в логістиці; розробити модель оптимізації логістичних операцій із мінімізацією витрат на транспортування та зберігання; визначити проблеми впровадження алгоритмів оптимізації у типових сценаріях логістики; узагальнити характеристику методів аналізу та моделювання логістичних процесів; запропонувати метод інтеграції засобів оптимізації в систему розрахунку логістичних операцій; розробити програмний засіб для оптимізації логістичних операцій; визначити функціональні вимоги до основних програмних модулів розробленого програмного засобу; реалізувати основні програмні модулі для

виконання оптимізаційних розрахунків; провести тестування програмних модулів на модельних та реальних даних; оцінити ефективність впровадженого програмного засобу для оптимізації логістичних операцій.

1.5. Висновки

У першому розділі обґрунтовано вибір методу аналізу, де показані особливості використання оптимізаційних алгоритмів управлінні логістичними ланцюгами постачання, до яких належали, базові алгоритми маршрутизації, оптимізація управління запасами, алгоритми розподілу ресурсів, аналіз даних і прогностичні алгоритми, розподіл вантажів і управління складськими запасами. Алгоритми маршрутизації, такі як генетичні алгоритми або алгоритми мурашиних колоній є важливими для оптимізації шляхів транспортування товарів. Оптимізація управління запасами полягала у використанні стохастичних та детермінованих методів досліджень для управління запасами з урахуванням ймовірнісних факторів попиту. Алгоритми розподілу ресурсів зазвичай використовують алгоритми лінійного програмування та теорії черг для оптимізації використання транспортних засобів, складів та робочої сили. Аналіз даних і прогностичні алгоритми особливо потрібні під час використання ШІ та машинного навчання для прогнозування попиту та адаптації до змін ринку на основі історичних даних. Для розподілу вантажів і управління складськими запасами використовуються оптимізаційні методи для визначення стратегій розподілу вантажів і управління складськими запасами для мінімізації витрат.

Також розглянуто основні напрямки моделей генетичного програмування для виконання задач логістики, де основними напрямками розгляду були маршрутизація транспортних засобів, управління запасами та планування перевезень.

Маршрутизація транспортних засобів головним чином використовувалась для оптимізації маршрутів доставки з урахуванням ємності транспортних засобів, часових вікон та витрат на транспортування. Управління запасами складали визначення різних рівнів запасів і термінів постачання для мінімізації

витрат на зберігання і транспортування.. Також, в цьому розділі розглядались переваги та недоліки маршрутизація транспортних засобів, управління запасами та планування перевезень.

Порівняльна характеристика моделей оптимізації логістичних процесів з використанням нейронних мереж дозволила визначити переваги та недоліки моделі оптимізації у нейронних мережах. При цьому визначено, що для оптимізації складських площ або планування маршрутів доставки з використанням просторових даних найкраще підходять конволюційні нейронні мережі.

2 РОЗРОБКА МОДЕЛІ ТА МЕТОДУ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ЛОГІСТИЧНИХ ОПЕРАЦІЙ

2.1. Постановка задачі підвищення ефективності логістичних операцій з використанням алгоритмів оптимізації на торгових підприємствах

Сучасні торгові підприємства функціонують у надзвичайно динамічному середовищі, де ефективність логістичних операцій є критичним фактором для забезпечення конкурентоспроможності. Логістика охоплює широкий спектр процесів, включаючи управління запасами, планування маршрутів, складські операції та координацію ланцюга постачання. З урахуванням високої складності та взаємозалежності цих процесів, зростає потреба в інноваційних методах оптимізації, які можуть ефективно вирішувати завдання в умовах невизначеності та великої кількості змінних.

Постановка задачі даного дослідження полягає у розробці та реалізації комплексного рішення для підвищення ефективності логістичних операцій на торгових підприємствах шляхом інтеграції генетичних алгоритмів та методів лінійного програмування.

Запропоноване комплексне рішення передбачає інноваційне поєднання двох потужних методів оптимізації наступним чином.

Генетичні алгоритми мають бути використані для попереднього пошуку рішень у складних і багатопараметричних задачах, таких як управління запасами ресурсів (валюти або цінних паперів) в умовах невизначеності. Ці алгоритми мають дозволяти досліджувати великий простір рішень і знаходити глобальні оптимуми, що є особливо важливим у складних логістичних операціях [14].

Таким чином, інтеграція генетичних алгоритмів та лінійного програмування дозволяє вирішувати складні задачі оптимізації в логістичних операціях шляхом комбінування глобального пошуку рішень (генетичні алгоритми) з точною оптимізацією (лінійне програмування). Цей підхід дозволяє торговим підприємствам значно знижувати витрати, пов’язані з

транспортуванням, управлінням запасами, складськими операціями та іншими логістичними процесами, що забезпечує їм конкурентні переваги на ринку.

Підвищити ефективність управління ресурсами завдяки точному розподілу ресурсів і швидкому реагуванню на зміни ринкових умов.

Інтеграція генетичних алгоритмів та лінійного програмування може суттєво підвищити ефективність управління ресурсами завдяки поєднанню двох ключових особливостей: здатності генетичних алгоритмів швидко реагувати на зміни в умовах невизначеності та гнучко адаптуватися до нових ситуацій, а також здатності лінійного програмування забезпечувати точний і оптимальний розподіл ресурсів в умовах визначеності. Розглянемо детальніше, як це відбувається [16].

Розподіл ресурсів на торгових підприємствах, зокрема таких як фінансові, матеріальні та людські ресурси, є важливим фактором для досягнення ефективності та конкурентоспроможності. Тому, ресурси повинні бути розподілені так, щоб максимізувати прибуток, мінімізувати витрати та забезпечити задоволення попиту.

Методи лінійного програмування дозволяють точно розподілити ресурси, враховуючи всі обмеження та вимоги. Наприклад, підприємство може мати обмеження на кількість наявних фінансових ресурсів, обмежену кількість товарів на складі або обмеження на час роботи персоналу. Лінійне програмування дозволяє оптимально розподілити ці ресурси для досягнення цільової функції (наприклад, максимізації прибутку або мінімізації витрат).

Наприклад, підприємство, що займається логістикою, може використовувати лінійне програмування для оптимального розподілу транспортних засобів між різними маршрутами, враховуючи обмеження на обсяги вантажу, час доставки та витрати на паливо.

Ринкові умови можуть швидко змінюватися, і підприємства повинні вміти швидко адаптувати свої ресурси до нових умов, щоб залишатися конкурентоспроможними. Наприклад, зміна попиту на продукцію, коливання цін

на ринку або зміни в доступності ресурсів можуть вимагати негайного перегляду поточних стратегій.

Така інтеграція генетичних алгоритмів і лінійного програмування дозволяє скористатися перевагами обох методів для досягнення високої ефективності управління ресурсами. Генетичні алгоритми можуть використовуватися для швидкої адаптації до змін ринкових умов, знаходячи нові можливі стратегії розподілу ресурсів у реальному часі.

Лінійне програмування забезпечує математичну точність і оптимальність у розподілі ресурсів, що особливо важливо для досягнення мінімальних витрат і максимального прибутку. Така інтеграція дозволяє скоротити час, необхідний для прийняття рішень, оскільки генетичні алгоритми швидко знаходять оптимальні варіанти, а лінійне програмування забезпечує їхню точну реалізацію [17,18].

Наприклад, у кризових умовах, коли ринок переживає значні зміни, інтеграція генетичних алгоритмів та лінійного програмування може допомогти підприємству швидко адаптувати свій бізнес, перенаправляючи ресурси в найбільш прибуткові або критичні напрямки, водночас зберігаючи точність і ефективність у їхньому використанні.

Таким чином, інтеграція генетичних алгоритмів та лінійного програмування підвищує ефективність управління ресурсами завдяки поєднанню здатності генетичних алгоритмів швидко реагувати на змінні ринкові умови з точністю та оптимальністю, які забезпечує лінійне програмування. Це дозволяє підприємствам ефективніше розподіляти свої ресурси, скорочуючи витрати і підвищуючи прибутковість, навіть у складних і нестабільних ринкових умовах.

Логістичні процеси на торгових підприємствах часто піддаються впливу різних факторів невизначеності, таких як коливання попиту, зміни в умовах постачання, перебої в роботі транспорту, та інші непередбачувані події. Такі умови створюють ризики, що можуть призвести до затримок, підвищення витрат або навіть втрат у бізнесі.

Генетичні алгоритми відрізняються своєю здатністю ефективно працювати в умовах невизначеності завдяки їхній еволюційній природі. Вони моделюють процеси природного відбору, де найкращі рішення зберігаються і поліпшуються на кожному кроці, адаптуючись до змін навколишнього середовища. Це дозволяє швидко знаходити рішення, які є оптимальними навіть у складних і мінливих умовах.

Наприклад, у разі раптових змін у попиті на товар, генетичний алгоритм може адаптувати логістичні процеси підприємства, пропонуючи нові маршрути доставки або зміни в управлінні запасами, щоб уникнути дефіциту або надлишкових витрат.

Генетичні алгоритми можуть швидко адаптуватися до нових умов, що виникають внаслідок змін на ринку або зовнішніх факторів. Вони можуть переглядати попередні рішення і знаходити нові шляхи оптимізації, що дозволяє знижувати ризики, пов’язані з логістичними процесами.

Наприклад, якщо певний постачальник не може виконати замовлення через проблеми з транспортом, генетичний алгоритм може швидко запропонувати альтернативні маршрути або постачальників, мінімізуючи ризики затримок.

Після того як генетичний алгоритм знаходить нові можливі рішення, методи лінійного програмування можуть бути застосовані для подальшої точкової оптимізації і мінімізації ризиків. Це дозволяє забезпечити математично точне розподілення ресурсів, що враховує всі нові умови.

Інтеграція генетичних алгоритмів і лінійного програмування також дозволяє одночасно використовувати переваги адаптивності та гнучкості генетичних алгоритмів і точність та оптимальність лінійного програмування.

Генетичні алгоритми можуть реагувати на зміни у реальному часі, забезпечуючи адаптацію логістичних процесів до нових умов, що знижує ризики збоїв і перебоїв. Лінійне програмування може точно розподілити ресурси на основі нових умов, мінімізуючи витрати та забезпечуючи надійність логістичних операцій.

До переваг такої інтеграції відносяться наступні чинники:

- гнучкість у прийнятті рішень, де генетичні алгоритми забезпечують гнучкість у прийнятті рішень, дозволяючи швидко реагувати на зміни в умовах невизначеності;

- точність і оптимальність, де лінійне програмування забезпечує точне виконання стратегій, розроблених генетичними алгоритмами, що підвищує надійність логістичних процесів;

- зниження ризиків, існує інтеграція цих методів, що дозволяє знизити ризики, пов'язані з невизначеністю, такими як затримки, підвищення витрат або збитки через непередбачувані події.

Наприклад, у разі форс-мажорних обставин, таких як природні катастрофи, інтеграція генетичних алгоритмів і лінійного програмування може допомогти підприємству швидко перебудувати свої логістичні операції, перенаправивши ресурси на альтернативні маршрути або постачальників, тим самим мінімізуючи втрати і забезпечуючи безперервність бізнесу.

Тому, інтеграція генетичних алгоритмів і лінійного програмування мінімізує ризики та підвищує надійність логістичних процесів за рахунок здатності генетичних алгоритмів адаптуватися до змінних умов невизначеності та забезпечувати гнучкість у прийнятті рішень, тоді як лінійне програмування забезпечує точність та оптимальність у виконанні цих рішень. Цей підхід дозволяє підприємствам ефективніше управляти ризиками та підтримувати надійність своїх логістичних операцій навіть у складних та непередбачуваних умовах [17-19].

Таким чином, дана постановка задачі пропонує комплексне та науково обґрунтоване рішення для підвищення ефективності логістичних операцій на торгових підприємствах. Інтеграція генетичних алгоритмів та лінійного програмування дозволить досягти значних покращень у управлінні логістичними процесами, що є критично важливим у сучасних умовах конкуренції на ринку.

2.2. Розробка методу інтеграції засобів оптимізації в систему розрахунку логістичних операцій

У сучасних умовах стрімкого розвитку цифрових технологій, зокрема в галузі малого та середнього бізнесу, ефективне управління ресурсами на торгових підприємствах з використанням математичних моделей стає ключовим фактором успіху. З огляду на постійне збільшення обсягів операцій купівлі-продажу зростає необхідність в оптимізації логістичних операцій, що забезпечують безперебійне функціонування підприємств. У цьому контексті використання алгоритмів оптимізації є одним із найбільш перспективних підходів для підвищення ефективності управління торгових операцій.

Такі алгоритми оптимізації знаходять широке застосування у різних сферах економіки та промисловості, зокрема у логістичних процесах на торгових підприємствах. Застосування таких алгоритмів дозволяє знизити витрати на логістику, оптимізувати маршрути постачання та скоротити час обробки замовлень, що є критичним для підприємств, які працюють з криптовалютами.

Тому торгові ресурси характеризуються зміною параметрів у часі, що створює додаткові виклики для управління. Впровадження алгоритмів оптимізації може сприяти більш раціональному розподілу ресурсів, ефективнішому плануванню закупівель та продажів, а також зниженню ризиків, пов'язаних з коливаннями курсів.

Методи лінійного програмування, у свою чергу, забезпечують оптимальний розподіл ресурсів між різними торговими підприємствами, з урахуванням обмежених можливостей кожного з них. Запропонований підхід дозволяє підвищити швидкість прийняття рішень та забезпечити гнучкість у процесі управління ресурсами.

Для підвищення ефективності логістичних операцій на торгових підприємствах, що працюють з цінними паперами або валютами, пропонується використання комбінованого підходу на основі генетичних алгоритмів та методів лінійного програмування. Генетичні алгоритми дозволяють ефективно вирішувати завдання оптимізації в умовах невизначеності, враховуючи численні

параметри, такі як курс валюти, обсяги попиту, витрати на здійснення операцій конвертування тощо [20, 21].

Для багатовимірних задач використовується побудова математичної моделі за допомогою симплекс-методу. Симплекс-метод є найпоширенішим і ефективним методом для розв'язання задач лінійного програмування з більш ніж двома змінними. Цей алгоритм починається з однієї з вершин допустимої області (базового рішення) і покроково переміщується до інших вершин, кожного разу покращуючи значення цільової функції, доки не буде досягнуто оптимальне рішення.

Метод пошуку оптимального рішення.

1.Ініціалізація. Введення даних: Вводяться початкові дані задачі, які включають.

2. Генерація першого рішення. Алгоритм генерує перше можливе рішення, яке задовольняє заданим обмеженням. Спосіб генерації залежить від конкретної задачі і може бути випадковим, детермінованим або комбінованим.

3. Оцінка рішення. Обчислюється значення цільової функції (критерію оптимальності) для поточного рішення.

4. Порівняння з найкращим рішенням. Порівнюється значення цільової функції поточного рішення з найкращим значенням, знайденим раніше.Якщо поточне рішення краще, то воно стає новим найкращим рішенням.

5. Генерація наступного рішення. Алгоритм генерує наступне можливе рішення. Спосіб генерації може бути різним і залежить від структури множини можливих рішень.

6. Перевірка на завершення. Перевіряється умова завершення алгоритму.

7. Перехід до кроку 3. Якщо умова завершення не виконана, алгоритм повертається до кроку 3 для оцінки нового рішення.

8. Виведення результату.

Блок-схему алгоритму роботи методу продемонстровано на рисунку 2.1

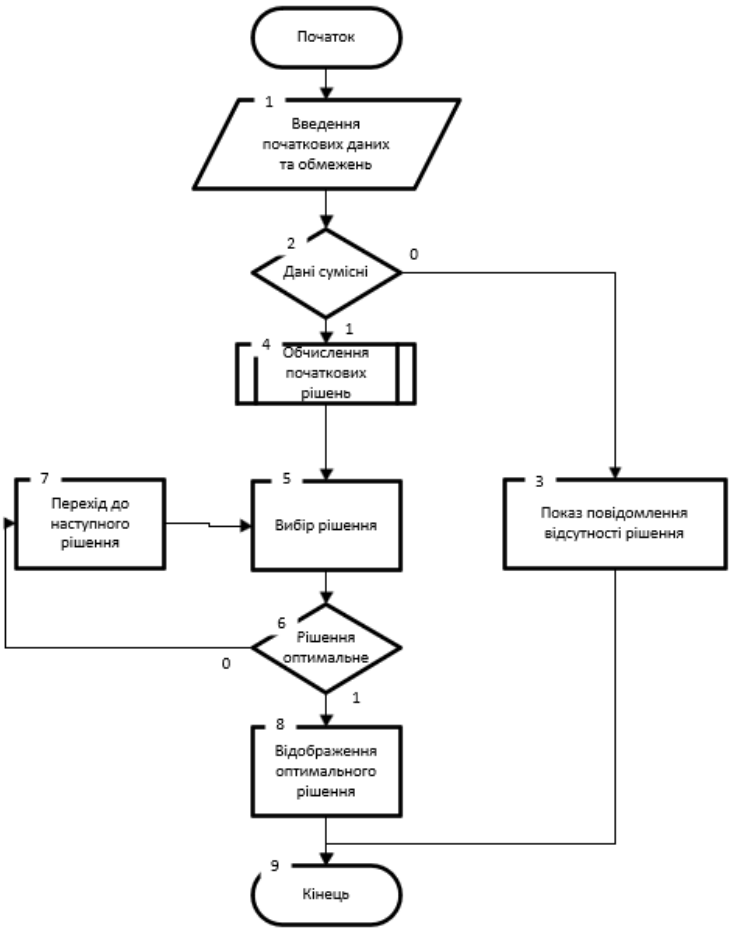


Рисунок 2.1 – Блок-схема пошуку оптимального рішення за методом інтеграції засобів оптимізації в систему розрахунку логістичних операцій

2.3 Метод підвищення ефективності логістичних операцій на основі генетичних алгоритмів та лінійного програмування

Метод підвищення ефективності логістичних операцій на основі генетичних алгоритмів та лінійного програмування передбачає комбіноване використанні методів мінімізації витрат на транспортування та зберігання, що дозволяє надавати підприємствам комплексний інструмент для ефективного управління всіма етапами логістики та забезпечує більш точне моделювання логістичних процесів зі скороченням витрат на логістику.

Метод підвищення ефективності логістичних операцій включає послідовність дій:

1. Введення початкових даних. На цьому етапі вводяться всі необхідні дані для роботи алгоритму, такі як:

- Ціль оптимізації (мінімізація витрат, максимізація прибутку тощо).
- Критерії оцінки рішень.
- Історичні дані, які будуть використані для навчання алгоритму.

2. Аналіз введених даних. Виконується попередній аналіз даних для виявлення закономірностей, трендів та інших важливих характеристик.

3. Ініціалізація генетичного алгоритму. Створюється початкова популяція випадкових рішень, які будуть служити основою для подальшої еволюції.

4. Оцінка працездатності. Кожному рішенню з поточної популяції присвоюється значення придатності (fitness), яке відображає наскільки добре це рішення відповідає поставленій меті.

5. Умова завершення виконана. Перевіряється, чи виконано умову завершення алгоритму. Це може бути досягнення заданої кількості ітерацій, знаходження рішення з достатньо високою придатністю або інша умова.

6. Генерація нових рішень. Якщо умова завершення не виконана, то виконується генерація нових рішень шляхом застосування генетичних операцій (селекція, кросинговер, мутація) до поточної популяції.

7. Моделювання взаємозв'язків між параметрами. Цей крок може бути присутній, якщо в задачі є взаємозалежність між різними параметрами. Моделювання дозволяє врахувати ці взаємозв'язки при оцінці рішень.

8. Вибір найкращого варіанта. З усіх розглянутих рішень вибирається рішення з найвищою придатністю.

9. Демонстрація найкращого рішення. Виводиться інформація про найкраще знайдене рішення.

10. Кінець. Метод завершує свою роботу.

Блок-схему алгоритму роботи методу наведено на рисунку 2.3

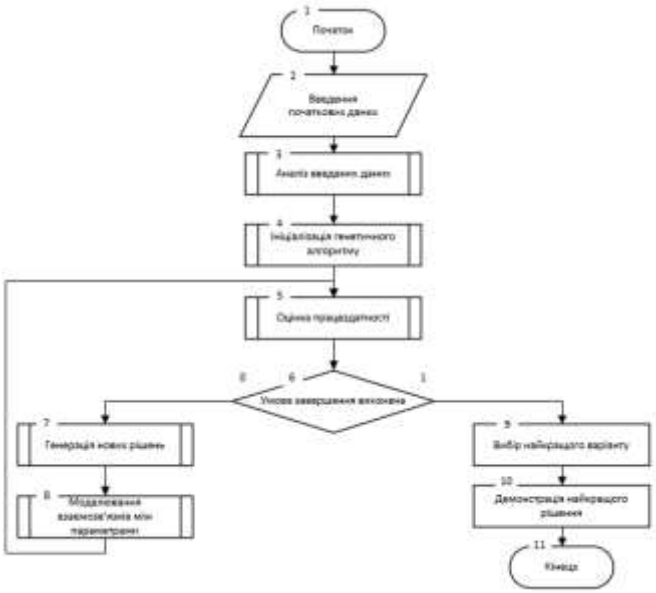


Рисунок 2.3 – Блок-схема алгоритму підвищення ефективності логістичних операцій за допомогою генетичного алгоритму

Таким чином, методи лінійного програмування є ефективними для вирішення задач розподілу ресурсів, оптимізації витрат та інших задач завдяки їх здатності точно та швидко вирішувати задачі з лінійною структурою, враховуючи всі необхідні обмеження. Вони дозволяють отримувати оптимальні рішення, що легко інтерпретуються і можуть бути застосовані в управлінні підприємством, що є особливо важливим у складних економічних умовах та при обмеженості ресурсів.

2.4. Висновки

У другому розділі розглядались моделі та методи підвищення ефективності логістичних операцій. Для цього виконувалась постанока задачі підвищення ефективності логістичних операцій з використанням алгоритмів оптимізації на торгових підприємствах. Визначено, що інтеграція генетичних алгоритмів та лінійного програмування дозволяє вирішувати складні задачі оптимізації в логістичних операціях шляхом комбінування глобального пошуку рішень (генетичні алгоритми) з точною оптимізацією (лінійне програмування). Цей підхід дозволяє торговим підприємствам значно знижувати витрати, пов’язані з

транспортуванням, управлінням запасами, складськими операціями та іншими логістичними процесами, що забезпечує їм конкурентні переваги на ринку.

Запропонований метод підвищення ефективності логістичних операцій на торгових підприємствах, де розглянуто впровадження алгоритмів оптимізації, який є ефективним засобом підвищення ефективності логістичних операцій на торгових підприємствах, що працюють з валютами та цінними паперами. Це дозволяє не лише знизити витрати та покращити якість обслуговування, але й мінімізувати ризики, пов'язані з волатильністю валютних ринків. При цьому, може рекомендуватись подальше вдосконалення методів оптимізації з метою адаптації до швидко змінюваних умов ринку та розширення їх застосування на інші сфери діяльності підприємств. Такі подальші дослідження можуть бути спрямовані на розробку інтегрованих систем управління криптовалютними ресурсами, що включають алгоритми машинного навчання для прогнозування ринкових тенденцій та розширення можливостей адаптивного управління в реальному часі.

Запропоновано метод підвищення ефективності логістичних операцій на торгових підприємствах мета якого полягає в підвищенні ефективності логістичних операцій на основі використання оптимізації, генетичних алгоритмів та лінійного програмування. Сутність підвищення ефективності логістичних операцій оптимізації на основі генетичних алгоритмів полягала в обробці невизначеності, багатопараметричній оптимізації, підвищення адаптивності та оптимізація витрат. В загальному випадку, оптимізація валютних витрат за допомогою генетичних алгоритмів дозволяє підприємствам ефективніше управляти своїми фінансами, мінімізуючи витрати, пов'язані з конвертацією валют, вибором платформ та хеджуванням валютних ризиків. Ці алгоритми забезпечують гнучкість та швидкість у прийнятті рішень, що є критично важливим у швидкозмінюваному середовищі валютних ринків та валютної торгівлі. Підвищення ефективності логістичних операцій за допомогою генетичних алгоритмів забезпечує підприємствам конкурентні переваги на ринку шляхом більш раціонального управління ресурсами та адаптації до мінливих умов.

3 РОЗРОБКА ТА ВПРОВАДЖЕННЯ ПРОГРАМНИХ МОДУЛІВ

3.1. Формування функціональних вимог та розробка основних програмних модулів

В умовах зростаючої конкуренції та постійного розвитку ринку торгові підприємства стикаються з необхідністю підвищення ефективності управління своїми логістичними операціями [2-24]. Використання оптимізаційних алгоритмів є ключовим інструментом для вирішення цих задач, особливо у сфері управління криптовалютними ресурсами, де гнучкість і швидкість процесів мають вирішальне значення. Даний підрозділ присвячений аналізу та розробці програми, що дозволить підвищити ефективність логістичних операцій за допомогою використання сучасних алгоритмів оптимізації. Дана програма складається з модулів, що показані на рис. 3.1. Розглянемо функціональність цих модулі більш ретельно.

Рисунок 3.1 – Модульна структура програмного забезпечення з оптимізації процесів на торгових підприємствах за допомогою алгоритмів оптимізації логістичних операцій

Розробка програми для оптимізації управління логістичними ланцюгами постачання на торгових підприємствах повинна базуватися на інтеграції кількох ключових модулів.

Кожен із цих модулів покликаний вирішити специфічні задачі логістичного управління та підвищити загальну ефективність процесів. Тому, ретельно розглянемо кожен з них.

Маршрутизація є основною складовою логістичних операцій, яка забезпечує ефективний транспорт товарів від постачальників до споживачів. Базові алгоритми маршрутизації, такі як алгоритми Дейкстри та Беллмана-Форда, використовуються для побудови оптимальних маршрутів з урахуванням відстані, часу та вартості доставки.

Використання евристичних підходів, таких як генетичні алгоритми та алгоритм мурашиних колоній, дозволяє знаходити рішення для складних логістичних задач, де точні методи є надмірно ресурсоємними.

Модуль базових алгоритмів маршрутизації відіграє ключову роль в оптимізації логістичних операцій на торгових підприємствах, оскільки саме від ефективної маршрутизації залежить швидкість та вартість доставки товарів. Основна функціональність цього модуля полягає у побудові найкоротших і найефективніших маршрутів для транспортування товарів між різними точками (складами, магазинами, клієнтами) з урахуванням різних обмежень та критеріїв оптимізації [24-25].

Основною функцією модуля є пошук найкоротших маршрутів між точками постачання та доставки.

Для цього використовуються класичні алгоритми пошуку найкоротших шляхів, такі як:

- алгоритм Дейкстри, що використовується для знаходження найкоротшого шляху в графі з не негативними вагами. Це дозволяє ефективно визначати мінімальну відстань або найменший час для виконання доставки;

- алгоритм Беллмана-Форда – розширює можливості Дейкстри і дозволяє працювати з графами, що мають негативні ваги, тобто у випадках, коли є знижки на певні маршрути або інші специфічні умови.

Також, ще цей модуль додатково дозволяє враховувати додаткові фактори при виборі оптимального маршруту, такі як:

- пропускна здатність доріг;

- транспортні витрати (паливо, оплата доріг тощо);

- час доставки (включно з очікуваними затримками, наприклад, через трафік);

- наявність проміжних зупинок для оптимального завантаження або розвантаження.

Даний модуль підтримує маршрутизацію для кількох транспортних засобів, розподіляючи завдання так, щоб мінімізувати загальні витрати або максимізувати ефективність використання ресурсів.

Це може включати розв'язання задачі про транспортні шляхи (Vehicle Routing Problem, VRP), де необхідно обрати оптимальні маршрути для кількох автомобілів з урахуванням їх вантажопідйомності та доступного часу.

Модуль дозволяє використовувати різні алгоритми в залежності від складності задачі. Наприклад, якщо задача має численні обмеження або змінні, можна застосувати евристичні алгоритми:

Генетичні алгоритми – імітують процес еволюції для пошуку наближеного оптимального рішення.

Блок-схема роботи генетичного алгоритму продемонстрована на рисунку 3.2.

Алгоритм мурашиних колоній – імітує поведінку мурах у пошуку найкоротшого шляху до джерела їжі, що дозволяє вирішувати складні задачі маршрутизації з великим числом змінних.

Рисунок 3.2 – блок-схема роботи генетичного алгоритму

У сучасних умовах маршрутизація повинна бути гнучкою та адаптуватися до змін, таких як аварії на дорогах або несприятливі погодні умови. Модуль базових алгоритмів маршрутизації дозволяє інтегрувати дані в реальному часі (наприклад, від систем GPS або дорожніх сенсорів) для оновлення маршрутів і миттєвої оптимізації. Наприклад, торгове підприємство, яке має кілька складів та постачає товари до роздрібних точок по всьому місту, може використовувати цей модуль для щоденної оптимізації маршрутів. Залежно від розташування клієнтів, завантаженості доріг та часу доставки, модуль може вибрати найкращі маршрути для кожного транспорту, що дозволить знизити витрати на пальне та прискорити доставку товарів [26].

Таким чином, модуль базових алгоритмів маршрутизації забезпечує ефективну роботу транспортної логістики торгових підприємств. Його використання дозволяє знижувати витрати, скорочувати час виконання замовлень і підвищувати загальну продуктивність транспортних операцій за рахунок оптимальних рішень щодо маршрутів і ресурсів.

Оптимізація управління запасами включає аналіз і прогнозування потреб у ресурсах на основі поточних даних про продажі та запаси. Тому, задача полягає в мінімізації витрат на зберігання та доставку запасів при одночасному забезпеченні своєчасної поставки товарів клієнтам. Для цього використовуються методи динамічного програмування та стохастичні алгоритми. Важливим інструментом у цьому процесі є прогнозування попиту на основі історичних даних і сезонних тенденцій, що дозволяє уникати дефіциту або надлишкових запасів.

Звідси, модуль оптимізації управління запасами є важливою частиною логістичних операцій на торгових підприємствах. Основною функцією цього модуля є забезпечення ефективного управління рівнем запасів на складах, що включає прогнозування попиту, мінімізацію витрат на зберігання, а також забезпечення своєчасної доставки товарів клієнтам. Оптимізація управління запасами є критичною для запобігання як дефіциту товарів, так і надмірному накопиченню, що може призвести до додаткових витрат [27, 28].

Тому, однією з основних функцій модуля є прогнозування попиту на основі аналізу історичних даних, сезонних трендів, змін на ринку та поведінки споживачів. Для цього використовуються різні методи прогнозування, такі як:

- часові ряди для виявлення сезонних коливань і тенденцій;

- регресійні моделі для виявлення зв'язку між попитом та зовнішніми факторами, такими як ціна, конкуренція або макроекономічні показники;

- методи машинного навчання, що дозволяють враховувати широкий спектр змінних та створювати більш точні прогнози, що особливо важливо в умовах високої варіативності попиту.

Модуль оптимізації управління запасами дозволяє мінімізувати витрати на зберігання товарів шляхом оптимізації кількості запасів, які необхідно зберігати на складах. Це завдання вирішується шляхом балансування між витратами на зберігання (оренда складів, витрати на обслуговування) і витратами, пов'язаними з поповненням запасів (транспортні витрати, витрати на замовлення). Використовуються такі підходи як:

- моделі економічного замовлення, де визначають оптимальний розмір замовлення, що мінімізує сукупні витрати на зберігання та замовлення товарів;

- моделі запасів з фіксованими інтервалами, що дозволяють оптимізувати графік замовлень для того, щоб товари поповнювалися з певними інтервалами, що зменшує ризик виникнення дефіциту.

Модуль забезпечує можливість постійного моніторингу рівня запасів у реальному часі, що дозволяє підприємствам швидко реагувати на зміни в попиті або поставках. Сучасні системи управління запасами інтегровані з програмами для автоматизації складів, що забезпечує точне відстеження запасів, оптимальне розташування товарів на складі та мінімізацію ручних помилок.

Цей модуль допомагає запобігти ситуаціям, коли товари стають недоступними для замовлення (дефіцит) або зберігаються занадто довго, що призводить до втрати їхньої вартості (надлишок). Оптимізація здійснюється за допомогою таких стратегій:

- безпечний рівень запасів, де встановлення мінімального рівня запасів, при якому необхідно автоматично запускати замовлення для поповнення;

- ABC-аналіз – поділ товарів на категорії за важливістю (A – найцінніші, B – середньої важливості, C – найменш цінні) для визначення пріоритетів у поповненні запасів;

- алгоритми коригування запасів, де існує динамічне коригування рівнів запасів на основі зміни попиту та поточних умов ринку.

Модуль оптимізації управління запасами може автоматизувати процес поповнення запасів шляхом інтеграції з системами постачальників. Це дозволяє мінімізувати затримки в постачаннях, забезпечити стабільний потік товарів і

знизити ризики, пов'язані з людськими помилками. Автоматичне поповнення базується на встановлених порогах рівня запасів і прогнозах попиту.

Наприклад, для торгового підприємства, яке регулярно постачає товари до мережі магазинів, модуль оптимізації управління запасами дозволяє підтримувати оптимальний рівень товарів на складах. Підприємство може використовувати цей модуль для прогнозування попиту на певні товари в період святкових розпродажів і автоматичного коригування запасів для запобігання дефіциту популярних товарів або перевантаженості складами продукцією, на яку попит може різко знизитися після свят [29-31].

Таким чином, функціональність модуля оптимізації управління запасами спрямована на мінімізацію витрат на зберігання та забезпечення безперервного потоку товарів у системі логістики торгових підприємств. Це досягається за рахунок точного прогнозування попиту, ефективного управління ризиками та автоматизації процесу поповнення запасів.

Відомо, що розподіл ресурсів є критично важливим завданням для забезпечення рівномірного навантаження на складські приміщення та транспортні засоби. Оптимізація цього процесу зменшує витрати на логістику та підвищує швидкість виконання замовлень. Застосування методів математичного програмування, зокрема лінійного та нелінійного програмування, дозволяє оптимізувати розподіл ресурсів з урахуванням обмежень на обсяг, час та вартість ресурсів.

Модуль алгоритмів розподілу ресурсів є важливою складовою оптимізації логістичних операцій на торгових підприємствах. Його основна мета полягає в тому, щоб забезпечити ефективне використання доступних ресурсів, таких як транспортні засоби, складські потужності, робоча сила та фінансові ресурси. Ефективний розподіл цих ресурсів дозволяє мінімізувати витрати, уникати затримок у постачанні та підвищити продуктивність логістичних операцій.

Один із основних викликів для торгових підприємств полягає у правильному розподілі транспортних засобів між різними маршрутами та замовленнями. Модуль алгоритмів розподілу ресурсів дозволяє вирішувати

задачі розподілу транспортних одиниць за допомогою методів лінійного програмування та евристичних алгоритмів, таких як:

- задача транспортних шляхів (Vehicle Routing Problem, VRP), де існує класична задача оптимізації, яка допомагає мінімізувати загальну відстань, час чи вартість доставки шляхом вибору найефективніших маршрутів для кожного транспорту;

- метаевристичні методи (генетичні алгоритми, алгоритми рою частинок) використовуються для розв'язання складніших варіантів задачі VRP, де необхідно враховувати обмеження на час роботи, вантажопідйомність та інші логістичні фактори.

Модуль також дозволяє оптимізувати розподіл людських ресурсів, таких як вантажники, водії та інші співробітники. За допомогою методів оптимізації можна автоматизувати процес планування робочих змін, з урахуванням робочих годин, доступності персоналу та необхідності в ресурсах на різних етапах логістичної операції. Цей підхід допомагає уникати простоїв та перевантаженості працівників.

Діаграму діяльності модуля наведено на рисунку 3.3.

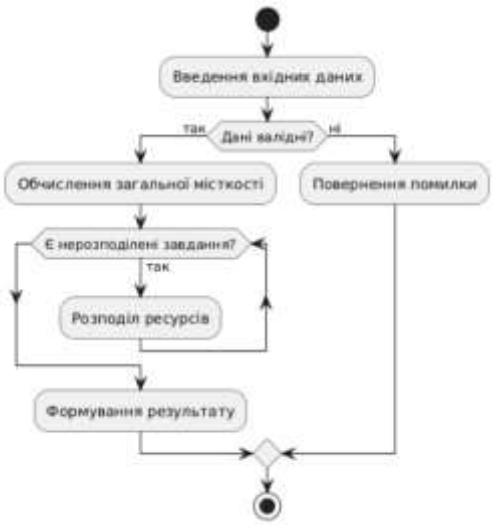


Рисунок 3.3. – Діаграма діяльності модуля алгоритмів розподілу ресурсів

Ефективне використання складських площ також є важливою частиною розподілу ресурсів. Модуль забезпечує оптимальне планування місць для зберігання товарів із врахуванням таких факторів, як:

- частота замовлень де товари, які користуються найбільшим попитом, розташовуються у найбільш доступних місцях для зменшення часу на завантаження/розвантаження;

- обмеження на обсяг, де обмеження складських площ з урахуванням об'ємів вантажів і типу зберігання;

- тривалість зберігання, де оптимізація ротації товарів, щоб зберігати лише необхідний рівень запасів та мінімізувати витрати на довготривале зберігання.

Модуль також може включати управління фінансовими ресурсами, зокрема розподіл витрат на різні етапи логістичних операцій. Алгоритми дозволяють оптимізувати розподіл бюджету на транспорт, складування, закупівлі та інші логістичні активності таким чином, щоб досягти максимального рівня ефективності при мінімальних витратах.

Аналіз великих обсягів даних у логістичних операціях дозволяє виявити приховані закономірності та прогнози, що сприяють підвищенню ефективності планування. Використання машинного навчання та нейронних мереж допомагає створювати прогностичні моделі, які враховують зміни ринкових умов, сезонні коливання попиту та інші чинники. Це дозволяє підприємствам гнучко реагувати на зміни попиту та своєчасно коригувати свої логістичні процеси.

Модуль аналізу даних і прогностичних алгоритмів є однією з ключових складових сучасних логістичних систем на торгових підприємствах. Його основне завдання полягає в обробці великих обсягів даних (Big Data), виявленні прихованих закономірностей, побудові прогнозів і забезпеченні підтримки прийняття рішень для підвищення ефективності логістичних операцій. Завдяки використанню методів аналізу даних і прогнозування підприємства можуть оперативно адаптуватися до змін на ринку та підвищити якість своїх бізнес-процесів.

Основою для ефективного аналізу є наявність якісних даних. Модуль аналізу даних виконує функції збору інформації з різних джерел, таких як системи управління складськими запасами (WMS), транспортні засоби (GPS), продажі та замовлення, ринкові умови, зовнішні джерела даних (наприклад, погодні умови). Дані можуть бути неструктурованими або структурованими, тому модуль виконує попередню обробку для очищення та нормалізації даних, щоб забезпечити їх придатність для подальшого аналізу.

За допомогою методів статистичного аналізу та машинного навчання модуль дозволяє виявляти приховані закономірності та взаємозв'язки між різними факторами. Це може включати:

- аналіз залежності попиту на товари від зовнішніх факторів (сезонність, погодні умови, економічні показники);

- кореляція між продажами та змінами цін;

- виявлення трендів у поведінці споживачів на основі історичних даних продажів.

Прогностичні алгоритми є ключовим компонентом цього модуля і використовуються для прогнозування попиту на товари, а також для прогнозування обсягів постачання та потреб у ресурсах. Серед основних алгоритмів прогнозування, які використовуються, можна виділити:

- лінійні та нелінійні регресійні моделі, де застосовуються передбачення кількісних змін попиту залежно від ключових факторів (ціна, сезонність, економічні умови);

- моделі часових рядів, , де існує можливість передбачати майбутні зміни попиту на основі історичних даних. Це може бути корисно для прогнозування продажів у святкові періоди або сезонні піки попиту;

- алгоритми машинного навчання (наприклад, нейронні мережі, дерева рішень), де існує можливість моделювати складні взаємозв'язки між багатьма факторами та створювати більш точні прогнози;

- оптимізацію логістичних процесів на основі прогнозів.

Прогнозування дозволяє оптимізувати логістичні операції на всіх етапах – від планування запасів до управління доставкою. Завдяки точним прогнозам підприємства можуть:

- заздалегідь підготувати достатню кількість запасів для задоволення попиту;

- оптимізувати транспортні маршрути на основі передбачуваного обсягу замовлень;

- забезпечити ефективне розміщення товарів на складах з урахуванням майбутніх потреб.

Модуль аналізу даних також забезпечує підтримку прийняття рішень у режимі реального часу. Використовуючи поточні дані з різних джерел, підприємства можуть оперативно реагувати на зміни в ринкових умовах, наприклад, зміни попиту, затримки в доставці або зміни у витратах на транспорт. Прогностичні алгоритми допомагають передбачати можливі проблеми та рекомендувати найкращі варіанти рішень для зниження ризиків.

Аналіз даних також дозволяє оцінити ефективність логістичних процесів, виявити слабкі місця та підвищити продуктивність. Наприклад, модуль може:

- аналізувати час виконання замовлень та виявляти затримки на певних етапах;

- оцінювати витрати на логістичні операції з метою їх оптимізації;

- прогнозувати потенційні ризики (затримки поставок, збої в транспорті) на основі історичних даних і зовнішніх факторів.

Модуль забезпечує функціональність для візуалізації отриманих даних і прогнозів у зручному для користувача форматі. Це дозволяє менеджерам логістичних операцій приймати обґрунтовані рішення на основі графіків, діаграм та звітів, що в реальному часі відображають поточний стан логістики, показники ефективності та прогнозовані зміни.

Таким чином, модуль аналізу даних і прогностичних алгоритмів є невід'ємною частиною сучасних систем управління логістикою на торгових підприємствах. Завдяки аналізу великих обсягів даних і побудові точних

прогнозів, підприємства можуть підвищити ефективність своїх операцій, знизити витрати та забезпечити своєчасну доставку товарів відповідно до попиту. Діаграму діяльності цього модуля наведено на рисунку 3.4.

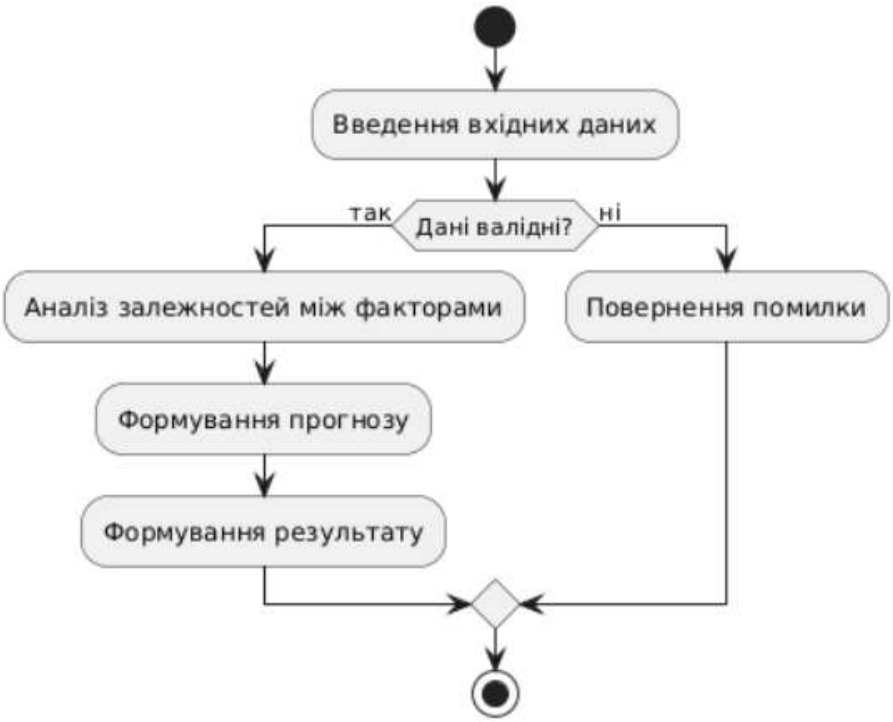


Рисунок 3.4. – Діаграма діяльності модуля аналізу даних

Модуль розподілу вантажів і управління складськими запасами є важливою складовою логістичних операцій на торгових підприємствах, оскільки він забезпечує ефективне зберігання товарів і своєчасну доставку вантажів. Мета цього модуля полягає в оптимізації управління складськими запасами, їх розміщенням, а також у раціональному розподілі вантажів між транспортними засобами для забезпечення своєчасної доставки при мінімальних витратах.

Модуль управління складськими запасами дозволяє ефективно контролювати та оптимізувати кількість товарів, які зберігаються на складах, забезпечуючи їх постійне поповнення та мінімізацію надлишкових запасів. Основні функції:

- моніторинг рівнів запасів у реальному часі, що забезпечується постійне відстеження поточного рівня запасів на складах, що дозволяє вчасно поповнювати запаси або перерозподіляти товари між різними складами;

- планування запасів, де модуль використовує прогностичні алгоритми для передбачення потреб у товарах на основі аналізу попиту та тенденцій на ринку;

- оптимізація розташування товарів на складі, де товари розміщуються з урахуванням їх попиту, частоти використання та логістичних вимог, що дозволяє мінімізувати час і витрати на їх обробку.

Для оптимізації управління запасами модуль застосовує різні моделі:

- EOQ (Economic Order Quantity), де є модель оптимального розміру замовлення, яка допомагає мінімізувати витрати на замовлення та зберігання товарів;

- ABC-аналіз, де є класифікація товарів на основі їх значущості, що дозволяє підприємству зосередити свої зусилля на управлінні найбільш важливими товарами (категорія A), водночас знижуючи контроль за менш важливими (категорія C);

Модуль розподілу вантажів забезпечує оптимізацію процесу транспортування товарів, включаючи вибір транспортних засобів, маршрутизацію та завантаження. Тому, існують такі основні функції як:

- оптимальне завантаження транспортних засобів – розподіл вантажів по транспортних одиницях здійснюється так, щоб максимізувати використання простору та мінімізувати кількість рейсів;

- алгоритми маршрутизації, де є модуль, що використовує різні алгоритми для побудови оптимальних маршрутів доставки (наприклад, алгоритм Дейкстри або генетичні алгоритми), що дозволяє скоротити час доставки та витрати на транспорт;

- оптимізація вантажних перевезень, що застосовуються алгоритми, які дозволяють вибрати найкращі варіанти перевезення з урахуванням вартості, термінів доставки, типу вантажу та доступності транспортних засобів.

Одна з функцій модуля полягає в оптимізації тривалості зберігання товарів. Для цього використовуються алгоритми ротації товарів, такі як FIFO (First In, First Out) або LIFO (Last In, First Out), які допомагають ефективно керувати запасами і зменшувати ризики втрати товарів через їх застарілість або псування.

На основі історичних даних про попит та продажі модуль прогнозує необхідний рівень запасів для майбутніх періодів. Це дозволяє уникати ситуацій з дефіцитом товарів або надмірним накопиченням на складах. Алгоритми прогнозування враховують сезонні коливання попиту, маркетингові кампанії та зміни ринкових умов.

Модуль також може включати функції управління зворотною логістикою, які стосуються обробки повернень товарів, їх ремонту або перерозподілу. Це дозволяє ефективно організовувати процеси, пов'язані з поверненням товарів на склад і їх подальшим обігом.

Наприклад, для торгового підприємства, яке займається продажем побутової техніки, модуль управління складськими запасами та розподілу вантажів дозволить:

- оптимізувати кількість товарів на складах, щоб уникати дефіциту або надлишкових запасів;

- ефективно організовувати доставку товарів до магазинів або безпосередньо до клієнтів через оптимізацію маршрутів та розподіл вантажів між транспортними засобами;

- впровадити систему моніторингу продуктивності складів для виявлення можливих затримок та оптимізації складських операцій.

Таким чином, модуль розподілу вантажів і управління складськими запасами забезпечує ефективну організацію логістичних процесів на торгових підприємствах. Його використання дозволяє мінімізувати витрати на зберігання, скоротити час доставки, забезпечити раціональне використання транспортних засобів та оптимізувати роботу складів. Діаграма класів для цього модуля наведена на рисунку 3.5.

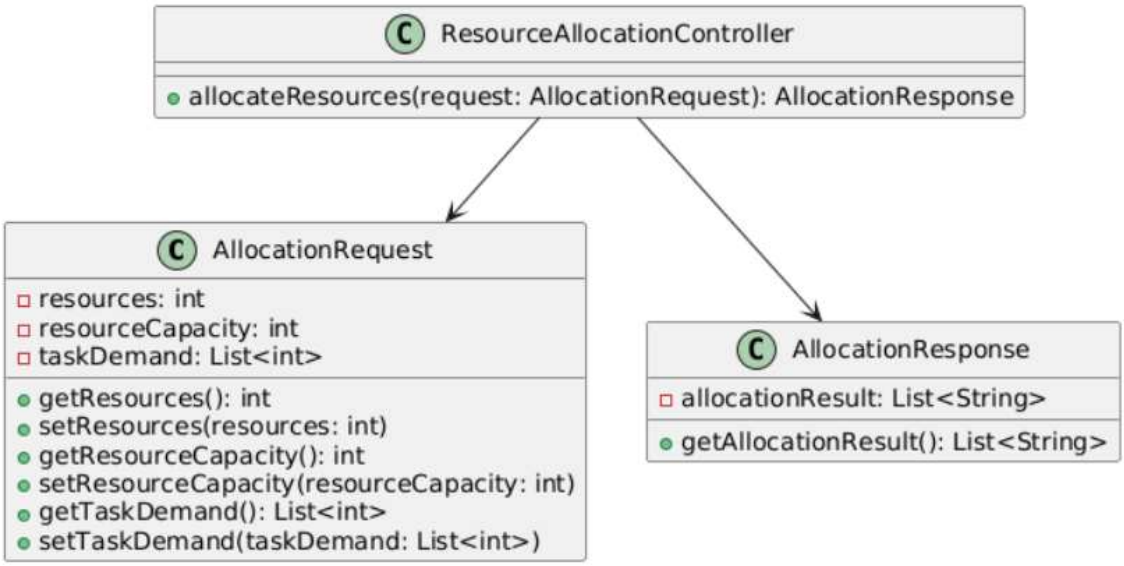


Рисунок 3.5 – Діаграма класів модуля розподілу ресурсів

Таким чином, запровадження програми з використанням оптимізаційних алгоритмів у логістичних процесах на торгових підприємствах дозволяє значно підвищити ефективність операцій, скоротити витрати та підвищити швидкість доставки товарів.

Рекомендації щодо подальшого розвитку включають інтеграцію з сучасними системами керування запасами та використання хмарних технологій для аналізу великих обсягів даних у реальному часі.

Використання машинного навчання та глибокого аналізу даних дозволить покращити прогностичні можливості систем та забезпечити гнучкість у управлінні ресурсами.

3.2. Розробка програмних засобів системи

Модуль оптимізації управління запасами є критично важливим для ефективного функціонування логістичних процесів на торгових підприємствах. Його мета полягає в тому, щоб оптимізувати кількість запасів на складах, забезпечити безперервність постачання, мінімізувати витрати на зберігання і

транспортування товарів, а також запобігти дефіциту або надмірному накопиченню товарів. Обробка вхідних даних є ключовим етапом, від якого залежить ефективність роботи модуля.

Базові принципи роботи модуля оптимізації управління запасами у зборі вхідних даних на основі алгоритму (рис. 3.6.) Цей модуль оптимізації управління запасами працює з великим масивом вхідних даних, які надходять з різних джерел, таких як:

- дані про продажі, де існують історичні дані про продажі товарів у різних торгових точках або через онлайн-канали. Ці дані можуть містити інформацію про кількість проданих товарів, сезонні коливання та поведінкові тенденції клієнтів;

- дані про запаси на складах, де існує інформація про поточний стан запасів, включаючи кількість товарів, їх розташування на складах, швидкість їх використання або ротацію;

- дані про постачальників, де є інформація про постачальників товарів, включаючи час доставки, мінімальні обсяги замовлень, надійність та умови контрактів;

- дані про логістичні витрати – витрати на зберігання, транспортування, пакування, страхування та обробку товарів;

- дані про ринкові умови, де існують зовнішні фактори, такі як зміни на ринку, поведінка конкурентів, економічні умови, що можуть впливати на попит на товари.

Алгоритм оптимізації управління запасами:

1. Збір даних:

- На цьому етапі збираються всі необхідні дані для аналізу, такі як:
  - Історія продажів.
  - Поточні залишки на складі.
  - Інформація про постачальників.
  - Витрати на зберігання і транспортування.
  - Дані про ринок і конкурентів.

2. Перевірка якості даних:

- Зібрані дані перевіряються на повноту, точність і узгодженість.
- Якщо дані містять помилки або пропуски, виконується їх очищення.

Попередня обробка даних:

- Дані приводяться до єдиного формату, здійснюється їх нормалізація та стандартизація.
- Можливе виконання додаткових перетворень даних, таких як фільтрація, агрегація або розрахунок нових показників.

Прогнозування попиту:

- На основі історичних даних про продажі будується модель для прогнозування майбутнього попиту.
- Оцінюється точність прогнозу.

Перевірка прогнозу:

- Перевіряється, чи прогнозований рівень попиту знаходиться в допустимих межах. Якщо ні, то необхідно переглянути модель прогнозування або додаткові фактори, що впливають на попит.

Розрахунок ключових показників:

- Розраховуються ключові показники ефективності управління запасами, такі як:
  - Середній рівень запасу
  - Швидкість обертання запасів
  - Рівень обслуговування замовлень
  - Витрати на зберігання

3. Оптимізація запасів:

- На основі розрахованих показників і прогнозу попиту визначається оптимальний рівень запасів для кожного товару.
- Можна використовувати різні методи оптимізації, такі як моделі економічного замовлення партії (EOQ), моделі з випадковим попитом тощо.

4. Перевірка витрат на зберігання:

- Перевіряється, чи витрати на зберігання запасів відповідають встановленим нормам. Якщо витрати занадто високі, необхідно скоригувати рівень запасів.

5. Корекція стратегії закупівель або продовження:

- Якщо витрати на зберігання в нормі і прогноз попиту точний, то стратегія закупівель продовжується.
- Якщо витрати занадто високі або прогноз неточний, то необхідно скоригувати стратегію закупівель, змінивши розміри замовлень, частоту поставок або інших параметрів.

Алгоритм продемонстровано на рисунку 3.6.

Після збору даних модуль виконує їх попередню обробку, що включає:

- очищення даних, де існує видалення дублікатів, пропущених або некоректних значень;

- нормалізація даних, де є приведення даних до єдиного формату для їх коректного аналізу. Наприклад, дані про запаси можуть бути представлені в різних одиницях вимірювання (штучні, вагові, об'ємні показники), тому їх необхідно уніфікувати.

Інтеграція даних – об'єднання даних з різних джерел для створення єдиної інформаційної бази, яка дозволяє здійснювати точний аналіз і прогнозування.

Після обробки даних модуль визначає ключові показники, які використовуються для оптимізації управління запасами. До таких показників можуть належати:

- рівень запасів, що позначає поточну кількість товарів на складах;

- коефіцієнт обороту запасів, що позначає показник, який відображає, як швидко запаси товарів обертаються за певний період часу.

- точка замовлення (reorder point), що позначає рівень запасів, при якому необхідно зробити нове замовлення товару для поповнення запасів;

- економічний розмір замовлення (EOQ), що позначає оптимальний обсяг замовлення, який мінімізує сукупні витрати на зберігання та замовлення товарів.

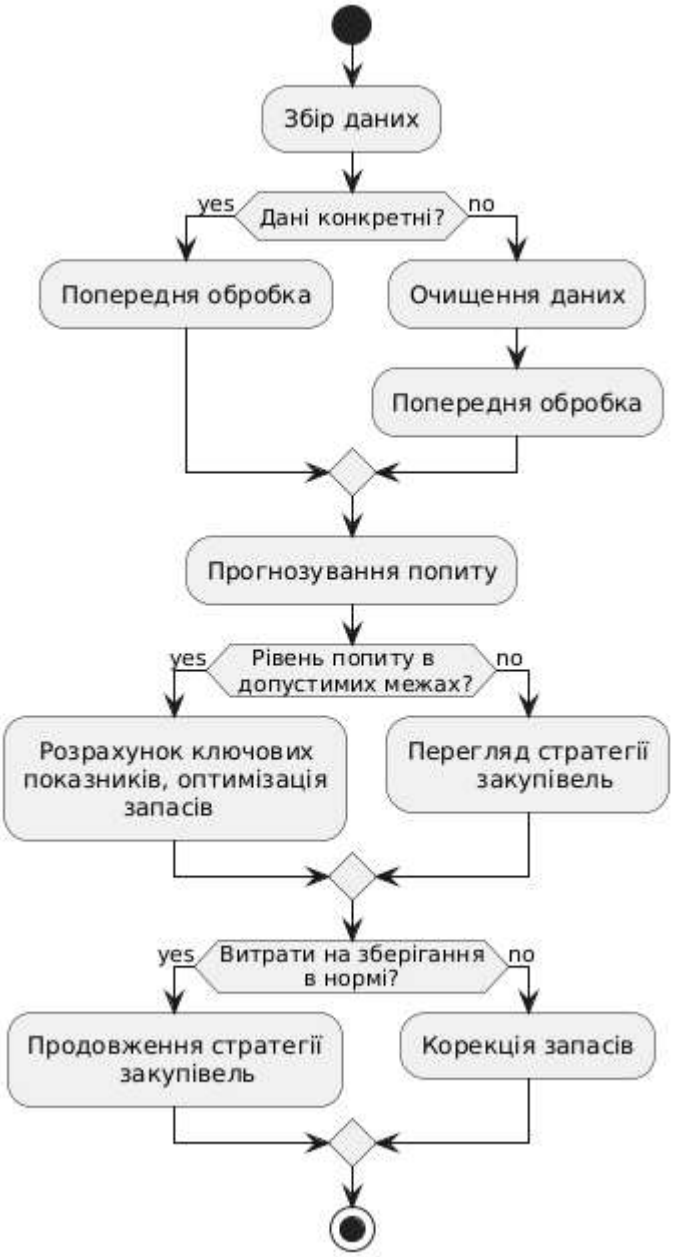


Рисунок 3.6 – Алгоритм роботи модуля оптимізації управління запасами у вигляді UML діаграми активності

Один із ключових етапів обробки вхідних даних полягає в аналізі та прогнозуванні попиту на основі історичних даних і ринкових умов. Модуль використовує різні алгоритми для побудови точних прогнозів попиту:

- часові ряди, що дозволяють виявляти сезонні коливання попиту та довгострокові тренди;

- регресійні моделі, що аналізують залежність попиту від різних факторів, таких як ціни, економічні показники або маркетингові активності;

- методи машинного навчання, що дозволяють створювати моделі прогнозування з використанням великих обсягів даних і врахуванням численних змінних, таких як погодні умови, поведінка конкурентів тощо.

Важливим аспектом обробки даних є оцінка витрат на управління запасами та прогнозування можливих ризиків, що пов'язані з дефіцитом або надлишком товарів наступним чином:

- оцінка витрат на зберігання, де модуль аналізує витрати на оренду складів, зберігання товарів, обслуговування складських систем, страхування тощо;

- оцінка витрат на замовлення, де є витрати на транспортування товарів від постачальників, витрати на оформлення замовлень, оплата митних зборів та інші витрати, пов'язані із замовленням товарів;

- прогнозування ризиків, де модуль аналізує можливість виникнення дефіциту товарів через затримки у постачанні, а також ризик надлишкового накопичення запасів, яке може призвести до збільшення витрат на зберігання або втрати вартості товарів через їх псування чи застарілість.

Таким чином, обробка вхідних даних у модулі оптимізації управління запасами є критично важливим етапом, який включає збір, нормалізацію та аналіз даних з різних джерел. Завдяки цьому модуль забезпечує точне прогнозування попиту, оцінку витрат та ризиків, а також розробку оптимальних стратегій управління запасами, що дозволяє підприємствам мінімізувати витрати, уникнути дефіциту або надлишку товарів і підвищити ефективність логістичних операцій. Далі, розглянемо приклад обробки вхідних даних для модуля оптимізації управління запасами з веб-інтерфейсом на основі HTML, CSS та JavaScript (рис. 3.7).

Рисунок 3.7 – Скріншот прикладу веб-інтерфейсу модуля оптимізації управління запасами

Метою цього прикладу є створення веб-інтерфейсу для модуля оптимізації управління запасами, який дозволить користувачу вводити вхідні дані, такі як поточний рівень запасів, історичні дані про продажі, мінімальні та максимальні рівні запасів, а також економічні параметри. На основі цих даних модуль повинен обробити інформацію і видати рекомендації щодо оптимального розміру замовлення та періодичності поповнення запасів.

Цей веб-інтерфейс складається з трьох основних частин:

- HTML для структури веб-сторінки і полів введення даних;

- CSS для стилізації інтерфейсу, щоб зробити його зручним і зрозумілим для користувача;

- JavaScript для обробки вхідних даних і виконання розрахунків.

Опис HTML-коду складається з поля введення, що дозволяє користувачу вводити необхідні вхідні дані, такі як поточний рівень запасів, середньоденний

рівень продажів, час доставки від постачальника, мінімальні та максимальні рівні запасів, а також економічні параметри (вартість замовлення і зберігання). Кнопка "Розрахувати оптимальне замовлення" запускає обробку даних через JavaScript.

Опис CSS-коду складається з інтерфейсу, що оформлено за допомогою простих стилів, що роблять його зручним і привабливим для користувачів. При цьому, форми й кнопки стилізовані для полегшення взаємодії.

Діаграма послідовності (рис 3.8.) наочно демонструє, як різні компоненти системи взаємодіють між собою в часі. Давайте детальніше розглянемо кожен етап взаємодії, який відбувається в процесі розрахунку оптимального маршруту:

1.Ініціалізація запиту користувачем:

- Користувач вводить необхідні дані (початкова точка, кінцева точка, вага вантажу тощо) у веб-форму.
- Веб-браузер збирає введені дані та формує HTTP-запит (зазвичай POST-запит) до веб-сервера.

2.Передача запиту на веб-сервер:

- Веб-браузер відправляє сформований HTTP-запит на веб-сервер.
- Веб-сервер приймає цей запит і витягує з нього необхідні дані.

3. Передача даних на бекенд-сервер:

- Веб-сервер передає отримані від користувача дані на бекенд-сервер для подальшої обробки. Ця передача може здійснюватися різними протоколами (наприклад, HTTP, gRPC).

4. Запит до бази даних:

- Бекенд-сервер аналізує отримані дані і формує запит до бази даних. Цей запит може бути SQL-запитом, якщо використовується реляційна база даних, або запитом на іншій мові, якщо використовується інший тип бази даних.
- База даних отримує запит і виконує його, повертаючи результати пошуку (наприклад, список можливих маршрутів, їхню вартість, час в дорозі).

5. Обробка даних і розрахунок маршруту:

- Бекенд-сервер отримує результати від бази даних і виконує необхідні обчислення для визначення оптимального маршруту. Ці обчислення можуть

включати в себе алгоритми пошуку найкоротшого шляху, врахування обмежень, оптимізацію за критеріями вартості, часу тощо.

6. Формування відповіді:

- Після виконання всіх розрахунків, бекенд-сервер формує відповідь, яка містить інформацію про оптимальний маршрут. Ця відповідь може бути представлена у вигляді JSON, XML або іншого структурованого формату.

7. Передача відповіді на веб-сервер:

- Бекенд-сервер передає сформовану відповідь на веб-сервер.

8. Відображення результату користувачу:

- Веб-сервер отримує відповідь від бекенд-сервера і передає її веб-браузеру.
- Веб-браузер парсить отримані дані і відображає їх користувачу у зручному вигляді (наприклад, на мапі, у вигляді таблиці або графіка).

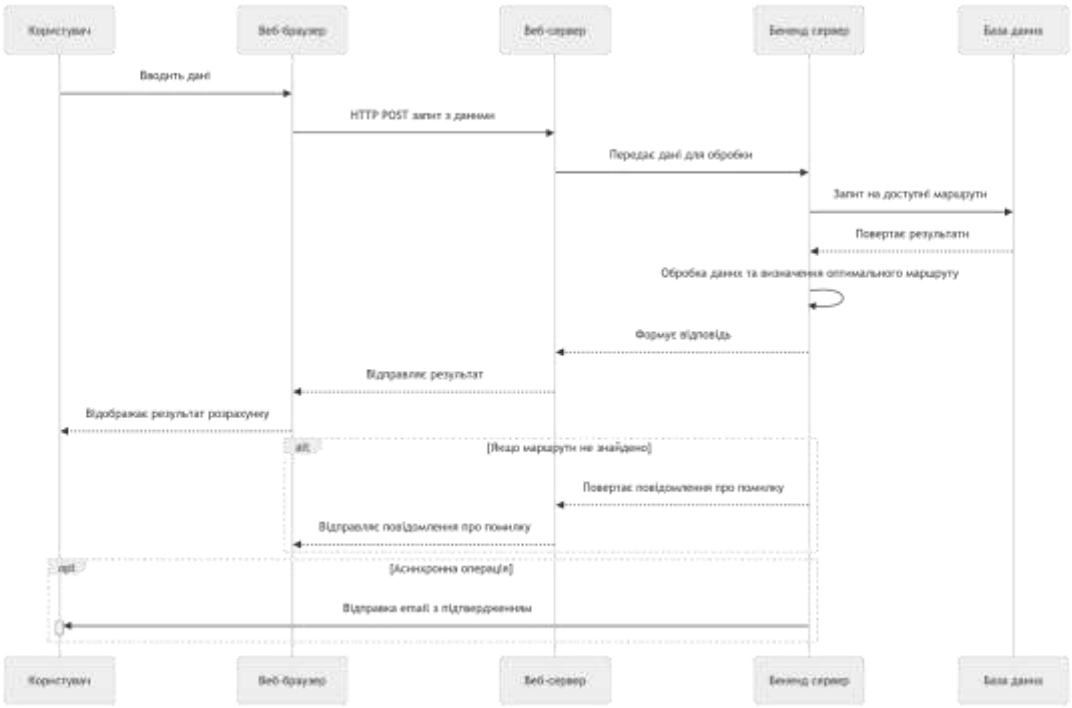


Рисунок 3.8 – Діаграма послідовності взаємодії користувача із застосунком

JavaScript-код був призначений для обробки вхідних даних і розрахунку оптимального замовлення, де значення вводяться користувачем у відповідні поля форми, і через JavaScript вони перетворюються на числові значення.

Під час перевірки валідності користувач отримує повідомлення, якщо дані введено некоректно.

Розрахунок оптимального замовлення використовується в якості класичної моделі EOQ (Economic Order Quantity), яка розраховує оптимальний обсяг замовлення на основі вартості замовлення, витрат на зберігання та середньоденного рівня продажів.

Розрахунок оптимального замовлення використовується в якості класичної моделі EOQ (Economic Order Quantity), яка розраховує оптимальний обсяг замовлення на основі вартості замовлення, витрат на зберігання та середньоденного рівня продажів.

Виведення висновків виконуються після розрахунку оптимального замовлення, де результати значень відображається на сторінці.

Таким чином, цей приклад демонструє, як можна використовувати HTML, CSS і JavaScript для побудови веб-інтерфейсу для модуля оптимізації управління запасами. Вхідні дані, зібрані через форму, обробляються за допомогою JavaScript для розрахунку оптимального розміру замовлення та точки замовлення. Такий інтерфейс можна використовувати для інтерактивної роботи з модулями оптимізації в реальних умовах торгових підприємств.

Backend для модуля оптимізації управління запасами необхідний для обробки вхідних даних, розрахунку оптимального розміру замовлення та точки замовлення на сервері, а також для інтеграції з базою даних або іншими системами управління запасами. Будемо використовувати Java або Kotlin як серверну мову програмування з використанням фреймворків Spring Boot (для Java) і Ktor (для Kotlin).

У роботі впровадження Backend цього модуля відбувалось на Java з використанням Spring Boot. Для цього підготовувався проект Spring Boot, що обробляв HTTP-запити для розрахунку оптимального замовлення. В цьому

випадку використовувався REST API для взаємодії з frontend. Створення Spring Boot проекту відбувалось на основі файл pom.xml для середовища Maven.

Також використовувався веб-контролер який приймав POST-запити від клієнта з даними про рівень продажів, час доставки, мінімальний рівень запасів, витрати на замовлення і зберігання. Контролер обробляє ці дані та обчислює оптимальний розмір замовлення за допомогою моделі EOQ.

Запит та відповідь надсилаються у форматі JSON через HTTP-запит і отримуються на стороні клієнта через API у вигляді JSON-відповіді.

Також була частина Backend, яка була реалізована на Kotlin з використанням Ktor.

Ktor – це асинхронний веб-фреймворк на Kotlin, який дозволяє легко створювати веб-додатки. У цьому випадку використовувався Ktor для обробки POST-запитів на сервер. Для цього, API отримував JSON-запит, який містив вхідні дані для обчислення оптимального замовлення, і повертав результат у форматі JSON.

Також використовувався веб-контролер який приймав POST-запити від клієнта з даними про рівень продажів, час доставки, мінімальний рівень запасів, витрати на замовлення і зберігання. Контролер обробляє ці дані та обчислює оптимальний розмір замовлення за допомогою моделі EOQ.

Розглянемо роботу цього контролера InventoryController.java (знаходиться в додатку В), що підтримує обробку даних для моделі EOQ (Economic Order Quantity) більш ретельно. Цей клас InventoryController реалізує REST-контролер для роботи з модулем управління запасами. Він обробляє HTTP-запити, пов'язані з розрахунком оптимального замовлення та точки повторного замовлення (reorder point) за допомогою моделі EOQ. При цьому використовуються наступні компоненти.

1. Анотації контролера:

- @RestController, який вказує на те, що цей клас є REST-контролером, і всі методи автоматично повертають дані у форматі JSON;

- @RequestMapping("/api/inventory"), який встановлює базовий URL для всіх ендпоінтів цього контролера, що починаються з /api/inventory.

2. Ендпоінт POST /api/inventory/calculateOrder:

- @PostMapping("/calculateOrder"), який обробляє POST-запити, що надсилаються на шлях /api/inventory/calculateOrder.

3. Логіка методу calculateOrder.

Метод виконує наступні дії:

Отримання параметрів:

- дані з JSON-запиту розпаковуються в об'єкт InventoryRequest.

- параметри:

averageSales, leadTime, minStock, orderCost, holdingCost.

Розрахунки:

Потреба під час доставки (demandDuringLeadTime):

Обчислюється кількість товару (або надходження - для криптовалюти), необхідного протягом часу доставки:

demandDuringLeadTime = averageSales × leadTime

demandDuringLeadTime=averageSales×leadTime

Точка повторного замовлення (reorderPoint):

Визначається як сума потреби під час доставки і мінімального запасу:

reorderPoint = demandDuringLeadTime + minStock

reorderPoint=demandDuringLeadTime+minStock

Оптимальне замовлення (optimalOrder):

Формування відповіді - результати розрахунків повертаються у вигляді об'єкта InventoryResponse, який містить:

optimalOrder: оптимальне замовлення.

reorderPoint: точка повторного замовлення.

4. Повернення JSON-відповіді

Відповідь формується у форматі JSON на основі об'єкта InventoryResponse. Наприклад:

{

"optimalOrder": 71.0,

"reorderPoint": 550.0

}

Також була частина Backend, яка була реалізована на Kotlin з використанням Ktor.

Ktor – це асинхронний веб-фреймворк на Kotlin, який дозволяє легко створювати веб-додатки. У цьому випадку використовувався Ktor для обробки POST-запитів на сервер. Для цього, API отримував JSON-запит, який містив вхідні дані для обчислення оптимального замовлення, і повертав результат у форматі JSON.

Після розрахунку backend повертав клієнту результат у форматі JSON, де було вказано оптимальний обсяг замовлення при якій потрібно робити нове замовлення.

Таким чином, у розробленому програмному засобу, частина якого розміщена у додатку В представлено два підходи для реалізації backend: з використанням Java (Spring Boot) та Kotlin (Ktor). Обидва варіанти дозволяли обробляти вхідні дані з frontend, виконувати розрахунок оптимального замовлення, і повертати результат клієнту через REST API.

3.3. Висновки

У третьому розділі виконувалась розробка та впровадження програмних модулів, визначались функціональні вимоги. Була розроблена модульна структура програмного забезпечення з оптимізації процесів на торгових підприємствах за допомогою алгоритмів оптимізації логістичних операцій до якого входили модулі базових алгоритмів маршрутизації, оптимізації управління запасами, алгоритми розподілу ресурсів, аналізу даних і прогностичні алгоритми, модулі розподілу вантажів і управління складськими запасами.

У результаті розробки цих модулів було виконано запровадження програми з використанням оптимізаційних алгоритмів у логістичних процесах на торгових підприємствах, що дозволяло значно підвищити ефективність

операцій, скоротити витрати та підвищити швидкість доставки товарів. При цьому, перспективними напрямками розвитку вважались інтеграція з сучасними системами керування запасами та використання хмарних технологій для аналізу великих обсягів даних у реальному часі. Використання машинного навчання та глибокого аналізу даних дозволятимуть покращити прогностичні можливості систем та забезпечити гнучкість у управлінні ресурсами.

Також було розглянуті основні принципи роботи модуля оптимізації управління запасами, де представлено алгоритм роботи модуля у вигляді UML діаграми активності. Частина програмного коду модуля розміщена у додатку В, де представлено два підходи для реалізації backend: з використанням Java (Spring Boot) та Kotlin (Ktor). Обидва варіанти дозволяли обробляти вхідні дані з frontend, виконувати розрахунок оптимального замовлення, і повертати результат клієнту через REST API.

4 ТЕСТУВАННЯ СИСТЕМ З УПРАВЛІННЯ КРИПТОВАЛЮТНИМИ ОПЕРАЦІЯМИ

4.1. Аналіз особливостей тестування систем з управління криптовалютними логістичними операціями

Особливості тестування систем управління криптовалютними логістичними операціями на торговельних підприємствах зосереджуються на кількох ключових аспектах (рис. 4.1).

Рисунок 4.1 - Особливості тестування систем управління криптовалютними логістичними операціями у торговельних підприємствах

Під час тестування необхідно перевірити швидкість та надійність криптовалютних платежів між учасниками логістичного ланцюга. Основною метою є забезпечення миттєвого проведення транзакцій з мінімальними витратами та без посередників. Для цього використовують симуляцію високих навантажень транзакцій, щоб перевірити стійкість та швидкість обробки операцій під час пікових періодів (табл. 4.1). При цьому, важливо враховувати

швидкість транзакцій, де необхідно оцінювати, наскільки швидко криптовалютні платежі проходять між учасниками логістичного ланцюга. Це важливо для забезпечення безперервності бізнес-процесів та уникнення затримок у доставці товарів чи послуг.

Таблиця 4.1 - Опис основних аспектів тестування систем управління криптовалютними логістичними операціями

Аспекти тестування	Опис
Перевірка ефективності фінансових транзакцій	Перевірка швидкості та надійності криптовалютних платежів між учасниками логістичного ланцюга, зокрема у періоди пікових навантажень.
Тестування смарт-контрактів	Тестування умов виконання смарт-контрактів та їх безпеки для мінімізації ризиків втручання та забезпечення автоматичного виконання угод.
Відстеження вантажів	Тестування можливості відстеження вантажів у реальному часі з використанням блокчейн-технології, включаючи сценарії зі збоями в зв'язку або втратою даних.
Оцінка управління ризиками	Моделювання сценаріїв різких змін курсів криптовалют та оцінка системи на здатність до хеджування валютних ризиків.
Автоматизація процесів	Перевірка здатності систем до автоматизації обробки платежів, контрактів та відстеження товарів, а також її інтеграції з іншими системами підприємства.

Під час надійності обробки транзакцій перевіряється стійкість системи до помилок або збоїв під час проведення платежів, включаючи здатність відновлювати транзакції у разі технічних проблем.

Для тестування смарт-контрактів необхідно переконатися, що смарт-контракти функціонують належним чином, автоматизуючи виконання угод за визначеними умовами. Це включає тестування сценаріїв із різними умовами

виконання, що дозволяє мінімізувати ризик людських помилок та гарантувати, що договірні зобов’язання виконуються при настанні відповідних подій. Особливу увагу слід приділити перевірці безпеки смарт-контрактів, щоб уникнути несанкціонованого втручання або зміни умов контракту.

Тестування смарт-контрактів полягає у перевірці їх функціональності, безпеки та надійності в автоматизованому виконанні умов договорів. Основні етапи та завдання тестування включають як перевірка правильності виконання умов контракту. В цьому випадку, тестування полягає в симуляції різних сценаріїв виконання контракту, щоб переконатися, що смарт-контракт автоматично виконує задані умови при настанні певних подій. Наприклад, контракт може автоматично переказувати кошти після отримання товару або виконання послуги.

Тестування також включає перевірку на вразливості, які можуть призвести до несанкціонованих змін умов або виконання контракту. Наприклад, проводиться аналіз захисту від зовнішніх атак, таких як "переповнення чисел" або "перехоплення даних", щоб запобігти шахрайським діям. Якщо проводиться тестування на стійкість до помилок, то перевіряється здатність смарт-контрактів коректно обробляти помилки або збої в мережі. Наприклад, чи зберігаються всі дані та чи продовжує контракт виконуватися після відновлення зв'язку, чи правильним чином обробляються виняткові ситуації.

Перевірка правильності взаємодії з іншими контрактами та системами проводиться наступним чином. Оскільки смарт-контракти можуть взаємодіяти з іншими контрактами або зовнішніми системами, важливо переконатися, що такі взаємодії виконуються коректно і не викликають конфліктів або помилок у логіці контракту. Для тестування продуктивності та ефективності оцінюється, наскільки швидко смарт-контракти можуть виконувати свої функції у реальних умовах, особливо при великому навантаженні. Це включає перевірку швидкості виконання контрактів та обробки великої кількості запитів одночасно.

Тестування смарт-контрактів дозволяє гарантувати, що вони будуть працювати відповідно до очікувань, забезпечуючи автоматизацію угод і

знижуючи ризики людських помилок або шахрайства в логістичних операціях торгових підприємств.

У процесі тестування критично важливо перевірити можливість відстеження всіх логістичних операцій у реальному часі за допомогою блокчейн-технології. Тестові сценарії мають включати випадки збоїв у зв'язку або втрати даних, що дасть змогу оцінити стабільність системи та її здатність забезпечувати безперервний доступ до інформації про переміщення товарів.

Реальне відстеження вантажів у режимі реального часу відбувається завдяки інтеграції з IoT-пристроями (наприклад, GPS-трекерами), можна точно визначати місцезнаходження вантажу на будь-якому етапі транспортування. Блокчейн-технологія дозволяє фіксувати всі ці дані та робити їх доступними для всіх зацікавлених сторін у режимі реального часу.

Оскільки всі дії з вантажем фіксуються на блокчейні, це значно зменшує ризик шахрайства або втрат вантажу. Незмінність записів унеможливлює підробку даних або приховування фактів затримки, пошкодження чи втрати товарів. Відстеження вантажів у реальному часі дозволяє ефективно керувати запасами. Торговельні підприємства можуть своєчасно планувати поставки та розподіл товарів, що дозволяє оптимізувати логістичні процеси та зменшити витрати на зберігання.

Під час відстеження вантажів може бути впроваджено автоматичну систему сповіщень, яка повідомляє зацікавлені сторони про ключові події — наприклад, коли вантаж досягає певної точки або коли виникають затримки.

Таким чином, відстеження вантажів за допомогою блокчейн-технології дозволяє значно підвищити ефективність і прозорість логістичних операцій на торгових підприємствах. Ця технологія забезпечує доступ до достовірної інформації в режимі реального часу, сприяє підвищенню довіри між партнерами та покращує управління запасами.

Волатильність криптовалют вимагає особливих підходів до управління ризиками. Системи мають бути протестовані на здатність виконувати хеджування валютних коливань, особливо у випадках міжнародних операцій. Це

тестування включає моделювання сценаріїв різких змін курсів криптовалют та оцінку відповідних механізмів реагування, таких як автоматичне коригування валютних запасів або активація захисних механізмів.

Для мінімізації ризиків, пов’язаних із волатильністю криптовалют, торговельні підприємства можуть застосовувати стратегії хеджування. Це передбачає використання фінансових інструментів для захисту від коливань курсів валют, наприклад, через укладання ф’ючерсних контрактів або використання деривативів. Оцінка ефективності цих стратегій також є важливою частиною управління ризиками. Використання криптовалют вимагає ретельного управління кіберризиками. Підприємства повинні забезпечити безпеку своїх криптовалютних гаманців і транзакцій, щоб запобігти кібератакам або шахрайським діям. Оцінка ризиків включає перевірку захисту системи, оцінку можливих точок уразливості та впровадження заходів для мінімізації ризику несанкціонованого доступу.

Оскільки в логістичних операціях можуть використовуватися смарт-контракти, важливо оцінити ризики, пов’язані з їх функціонуванням. Це передбачає перевірку правильності написання смарт-контрактів, їх стійкість до помилок або збоїв, а також можливість змін або атак на їхній код.

Внаслідок того, що криптовалюти можуть відрізнятися низькою ліквідністю порівняно з традиційними валютами, то важливо оцінити можливість швидкого конвертування активів у готівку або інші форми, необхідні для підтримки поточних операцій підприємства. Оцінка ризику ліквідності полягає у визначенні, наскільки легко і швидко можна реалізувати криптовалюту для здійснення платежів або купівлі активів без значних фінансових втрат.

Автоматизація процесів управління запасами полягає у використанні програмних систем, які відстежують рівень запасів у реальному часі і автоматично формують замовлення на поповнення при досягненні мінімального рівня товару на складах. Це дозволяє підприємствам зменшити витрати на зберігання товарів, оптимізувати закупівлі та забезпечити безперебійне постачання. Така інтеграція програмних рішень для автоматизації управління

ризиками передбачає використання алгоритмів, які аналізують ринкові дані та зміни курсів криптовалют для автоматичного хеджування ризиків. Це дозволяє миттєво реагувати на волатильність ринків та уникати втрат без потреби у постійній участі людей.

Завдяки блокчейн-технології, всі документи та угоди, пов'язані з логістичними операціями, можуть автоматично реєструватися та зберігатися у незмінному вигляді. Це спрощує процеси перевірки угод, виключає необхідність ручної перевірки та гарантує достовірність даних.

Таким чином, автоматизація процесів у криптовалютних логістичних операціях дозволяє значно підвищити ефективність, знизити витрати та ризики, а також забезпечити прозорість і надійність операцій. Використання таких інструментів, як смарт-контракти, блокчейн і криптовалютні платежі, дозволяє торговельним підприємствам мінімізувати ручну працю та швидко реагувати на зміни в умовах ринку.

Тестування систем управління криптовалютними логістичними операціями є багатогранним процесом, який вимагає перевірки на різних рівнях: від транзакцій до смарт-контрактів та відстеження вантажів. Для досягнення високої ефективності логістичних операцій на торговельних підприємствах рекомендується використовувати комплексні методи тестування, які охоплюють симуляцію реальних умов експлуатації системи. Важливим аспектом є перевірка здатності системи адаптуватися до змін у криптовалютних ринках та забезпечувати надійність операцій у реальному часі. Таке тестування повинно також враховувати майбутні масштаби операцій та здатність систем до інтеграції з новими технологіями, що можуть з'явитися на ринку.

4.2. Тестування модулів методом чорної скриньки

1. Модуль оптимізації управління запасами:

- Забезпечує розрахунок оптимального розміру замовлення (EOQ) та точки замовлення для уникнення дефіциту або надмірних запасів.

- Використовується для прогнозування та управління рівнями запасів на основі вхідних параметрів.

Ключові функції:

- Обчислення оптимального розміру замовлення на основі вартості замовлення та зберігання.
- Розрахунок точки замовлення для вчасного поповнення запасів.

Основні перевірки під час тестування:

- Коректність розрахунків EOQ та точки замовлення.
- Валідність вхідних даних (неприпустимі значення, наприклад, негативні або нульові).
- Обробка помилок для некоректних даних.

Результати тестування наведено в таблиці 4.2 та на рисунку 4.2

Таблиця 4.2 – Результат тестування модуля оптимізації управління запасами

Тест-кейс	Вхідні дані	Очікуваний результат	Результат тесту
1	currentStock = 50, averageSales = 10, leadTime = 2, minStock = 20, orderCost = 50, holdingCost = 1	Оптимальний розмір замовлення: 31.62, точка замовлення: 40.00	Успішно
2	currentStock = -10, averageSales = 10, leadTime = 2, minStock = 20, orderCost = 50, holdingCost = 1	Повідомлення про помилку (некоректний рівень запасів).	Успішно
3	currentStock = 100, averageSales = 0, leadTime = 2, minStock	Повідомлення про помилку (середній рівень продажів не може бути нульовим).	Успішно

	= 20, orderCost = 50, holdingCost = 1		

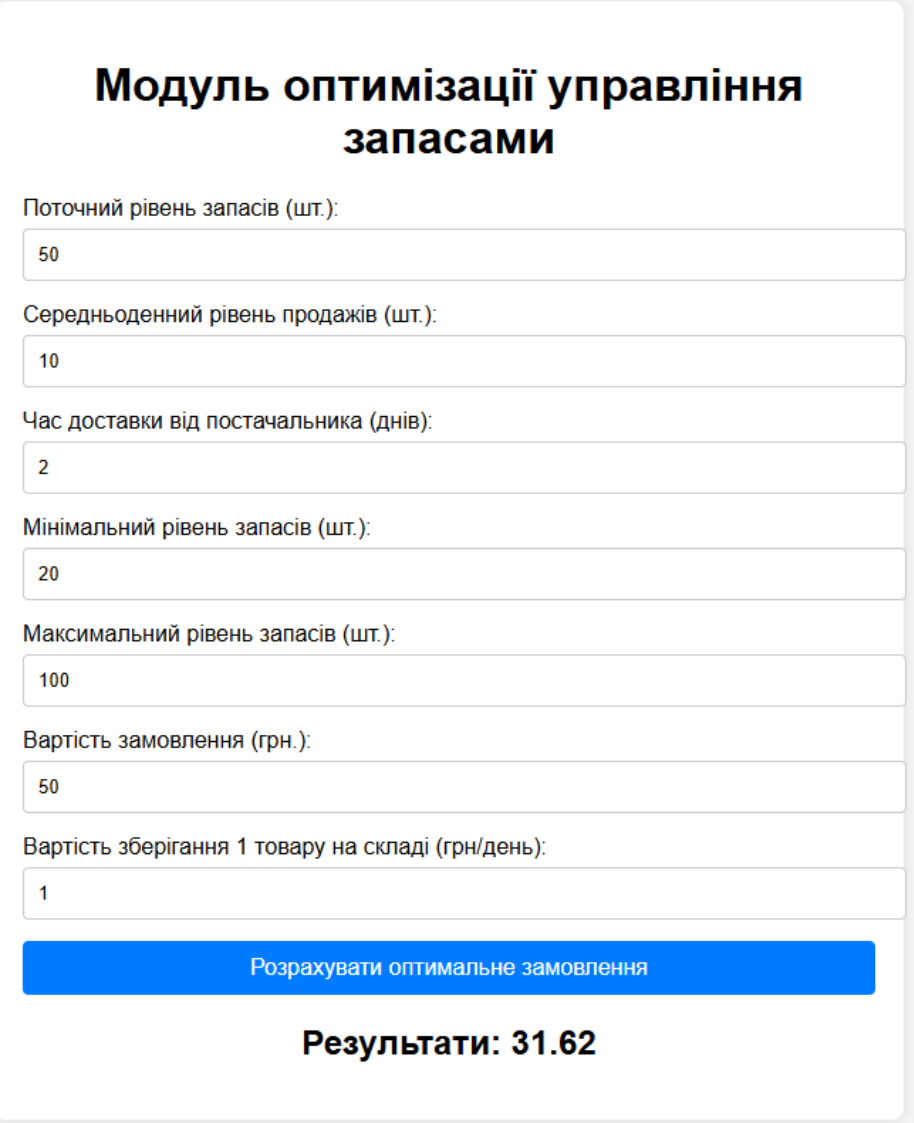


Рисунок 4.2 – Результат тестування модуля оптимізації управління запасами

2. Модуль алгоритмів розподілу ресурсів

Відповідальність:

- Забезпечує розподіл доступних ресурсів між завданнями на основі їх попиту та доступної місткості.
- Оптимізує використання ресурсів, запобігаючи надмірному або недостатньому виділенню.

Ключові функції:

- Розрахунок доступної місткості.
- Розподіл ресурсів між завданнями на основі їх попиту.

- Генерація результатів розподілу.

Основні перевірки під час тестування:

- Коректний розподіл ресурсів між завданнями.
- Валідність вхідних даних (наприклад, позитивні значення ресурсів та попиту).
- Обробка випадків нестачі ресурсів.

Результати тестування наведено в таблиці 4.3 та на рисунку 4.3

Таблиця 4.3 – Результат тестування модуля алгоритмів розподілу ресурсів

Тест-кейс	Вхідні дані	Очікуваний результат	Результат тесту
1	resources = 2, resourceCapacity = 10, taskDemand = [5, 10]	Завдання 1: 5 ресурсів, Завдання 2: 10 ресурсів.	Успішно
2	resources = 1, resourceCapacity = 5, taskDemand = [5, 10]	Завдання 1: 5 ресурсів, Завдання 2: 0 ресурсів (недостатньо ресурсів).	Успішно
3	resources = 2, resourceCapacity = -5, taskDemand = [5, 10]	Повідомлення про помилку (некоректна місткість ресурсу).	Успішно

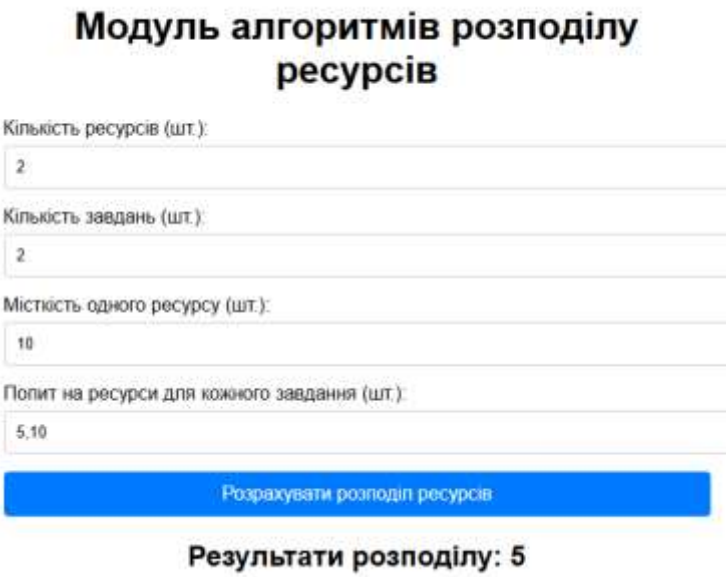


Рисунок 4.3 – Результат тестування модуля алгоритмів розподілу ресурсів

3. Модуль аналізу даних та прогнозування

Відповідальність:

- Виконує аналіз історичних даних і генерує прогнози на основі заданих факторів та горизонтів прогнозування.
- Використовується для прийняття рішень, пов’язаних із попитом, запасами та ресурсами.

Ключові функції:

- Аналіз залежностей між вхідними даними та заданими факторами.
- Формування прогнозів на основі історичних даних.

Основні перевірки під час тестування:

- Коректність прогнозів (наприклад, кількість прогнозованих значень відповідає горизонту).
- Валідність вхідних даних (історичні дані, значення факторів).
- Обробка помилок, коли дані відсутні або не відповідають очікуваним форматам.

Результати тестування наведено в таблиці 4.4 та на рисунку 4.4

Таблиця 4.4 – Результат тестування модуля аналізу даних та прогнозування

Тест-кейс	Вхідні дані	Очікуваний результат	Результат тесту
1	historicalData = [100, 200, 150], factors = {"trend": 0.1}, timeHorizon = 3	Прогноз повертає 3 значення, враховуючи тренд.	Успішно
2	historicalData = [], factors = {"trend": 0.1}, timeHorizon = 3	Повідомлення про помилку (немає історичних даних).	Успішно
3	historicalData = [100, 200, 150], factors = {"trend": 0.1}, timeHorizon = -1	Повідомлення про помилку (горизонт прогнозу має бути додатнім).	Успішно

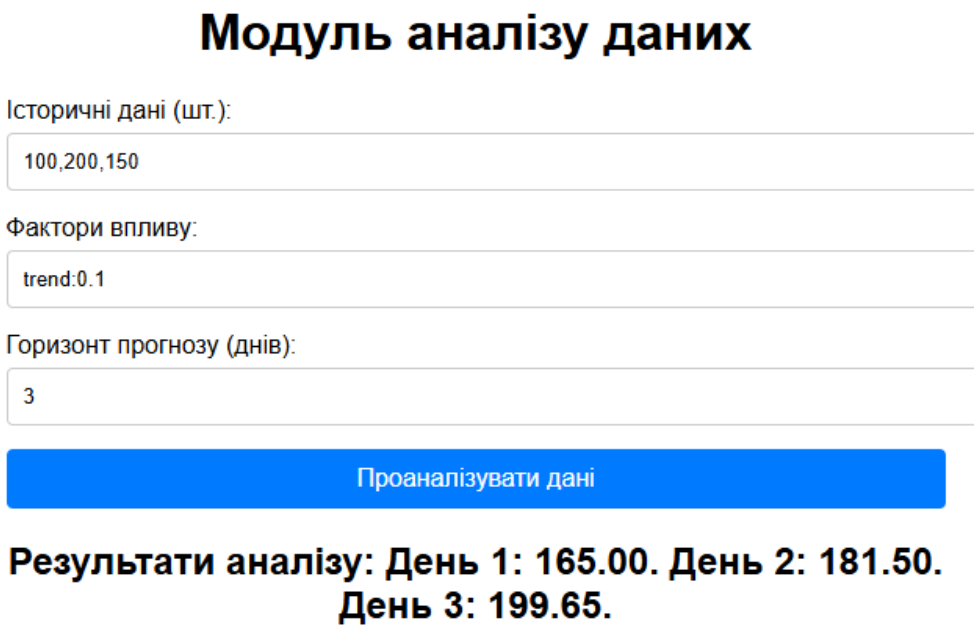


Рисунок 4.4 – Результат тестування модуля аналізу даних та прогнозування

4. Модуль розподілу вантажів

Відповідальність:

- Забезпечує оптимальний розподіл вантажів між транспортними засобами, враховуючи їх місткість і кількість.

- Використовується для планування перевезень і зменшення транспортних витрат.

Ключові функції:

- Розрахунок доступної місткості транспорту.
- Планування розподілу вантажів між транспортними засобами.
- Генерація плану розподілу.

Основні перевірки під час тестування:

- Коректний розподіл вантажів між транспортними засобами.
- Валідність вхідних даних (наприклад, позитивні значення для місткості та кількості транспортних засобів).
- Обробка випадків, коли транспорту недостатньо для перевезення всього вантажу.

Результати тестування наведено в таблиці 4.5 та на рисунку 4.5

Таблиця 4.5- Результат тестування модуля розподілу вантажів

Тест-кейс	Вхідні дані	Очікуваний результат	Результат тесту
1	cargoList = [100, 200], vehicleCapacity = 150, vehicleCount = 2	План розподілу повертається коректно.	Успішно
2	cargoList = [100, 200], vehicleCapacity = 100, vehicleCount = 1	Повідомлення про неможливість розподілу (недостатньо транспорту).	Успішно
3	cargoList = [], vehicleCapacity = 100, vehicleCount = 2	Повідомлення про помилку (немає вантажів для розподілу).	Успішно

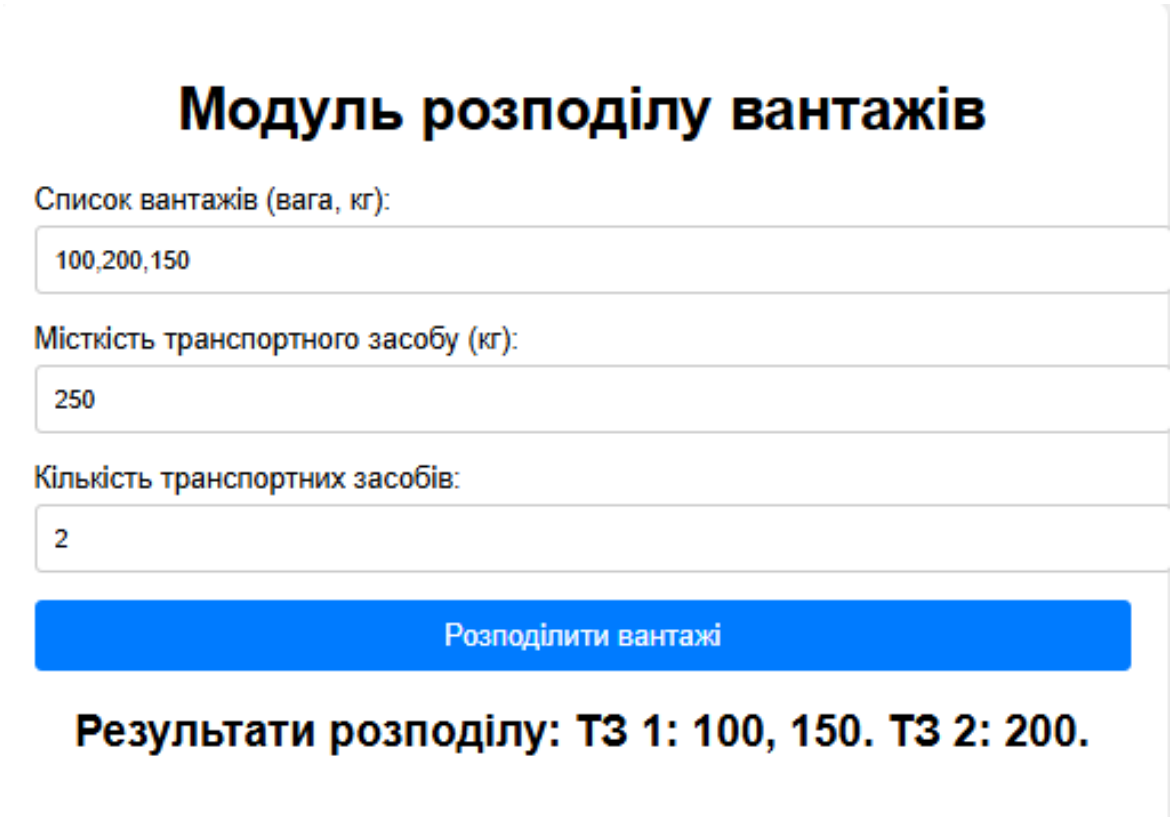


Рисунок 4.5 – Результат тестування модуля розподілу вантажів

4.3. Використання логістичних операцій на торгівельних підприємствах

Управління логістичними процесами на торгових підприємствах має вирішальне значення для забезпечення ефективного функціонування бізнесу. Одним з ключових завдань є оптимізація логістичних витрат, яка дозволяє зменшити фінансові втрати та підвищити рентабельність операцій. Впровадження алгоритмів оптимізації, таких як генетичне програмування, нейронні мережі, алгоритми управління маршрутами та оптимізація запасів, дозволяє досягти значного покращення ефективності. У даному контрольному прикладі розглянемо порівняльну характеристику та основні категорії логістичних витрат та способи їх оптимізації.

Транспортування є важливою складовою логістики, оскільки воно забезпечує переміщення товарів між виробниками, складами та кінцевими споживачами. Витрати включають оплату пального, обслуговування транспортних засобів, оплату праці водіїв та амортизаційні витрати на

транспортні засоби. Оптимізація маршрутів з використанням базових алгоритмів, таких як алгоритм найближчого сусіда або генетичні алгоритми, дозволяє мінімізувати час доставки та скоротити витрати на паливо і технічне обслуговування [32].

Ефективне управління складськими приміщеннями включає підтримку необхідного рівня запасів при мінімальних витратах. Витрати на зберігання включають оплату за оренду складів, енергоспоживання, обслуговування приміщень, охорону та витрати на управління запасами. Впровадження нейронних мереж для прогнозування попиту на товари дозволяє оптимізувати рівні запасів і уникати надмірних витрат на зберігання продукції.

Логістичні витрати включають адміністративні витрати, пов'язані з прийомом, обробкою та виконанням замовлень. Ці витрати включають використання інформаційних систем, оплату праці персоналу, який займається обробкою замовлень. Використання алгоритмів прогнозування та аналізу даних дозволяє автоматизувати процеси управління замовленнями, підвищити точність прогнозів і зменшити адміністративні витрати.

Витрати на управління запасами включають фінансові витрати, пов'язані з замороженим капіталом, страхуванням запасів, а також ризики морального зносу товарів. Оптимізація управління запасами за допомогою алгоритмів лінійного програмування та генетичних алгоритмів дозволяє підтримувати оптимальний рівень запасів, зменшуючи фінансове навантаження та ризики.

Також, логістичні операції супроводжуються значними фінансовими транзакціями, зокрема банківськими комісіями, обміном валют, оплатою послуг фінансових посередників тощо. Впровадження криптовалютних технологій для здійснення транзакцій дозволяє зменшити комісійні витрати і забезпечити більш швидкі та безпечні платежі.

З метою порівняння розглянемо перший випадок використання логістичних операцій на торгових підприємствах стандартними методами. Так, у класичному підході до управління логістичними операціями на торгових підприємствах оптимізація досягається за рахунок використання стандартних

методів та інструментів управління ланцюгом постачань. Цей підхід базується на впровадженні інформаційних систем, плануванні маршрутів та прогнозуванні попиту за допомогою:

- оптимізацію маршрутів та розкладів транспортування, де існує використання алгоритмів оптимізації, таких як методи лінійного програмування або класичні алгоритми пошуку найкоротших шляхів, дозволяє зменшити витрати на паливо та час доставки, що безпосередньо впливає на загальні транспортні витрати;

- покращення управління запасами, де за допомогою прогнозних моделей управління запасами, торговельні підприємства можуть точніше оцінювати попит на продукцію, знижуючи витрати на зберігання та мінімізуючи ризики надлишкових або недостатніх запасів;

- автоматизація процесів обробки замовлень, де інтегровані інформаційні системи виконують прийом та обробку замовлень, що зменшує адміністративні витрати, час обробки та ймовірність помилок, пов'язаних з ручною обробкою даних;

- впровадження криптовалют для фінансових транзакцій, де використання криптовалют як платіжного засобу може знизити комісійні витрати, що пов'язані з традиційними фінансовими операціями, та прискорити процес платежів між партнерами, особливо на міжнародному рівні.

- інтеграція блокчейн-технологій, де використання блокчейну підвищує прозорість та безпеку логістичних операцій, знижуючи витрати на аудит та зменшуючи ризики шахрайства або помилок в управлінні ланцюгом постачань.

Таким чином, логістичні витрати, пов'язані з класичним підходом, можуть бути тільки знижені за рахунок оптимізації процесів транспортування, управління запасами, автоматизації та інновацій у фінансових операціях.

У другому випадку будемо використовувати нейромережевий метод та генетичне програмування, де як показали проведені досліджнення, основні заходи знижують логістичні витрати. Це відбувається внаслідок того, що сучасні торгові підприємства можуть досягати ще більшого зниження логістичних

витрат шляхом впровадження нейромережевих методів та алгоритмів генетичного програмування. Це підходи, які дозволяють адаптивно оптимізувати процеси в реальному часі на основі аналізу великих обсягів даних. Основні переваги використання нейромережевого методу та генетичного програмування включають:

- оптимізація маршрутів доставки, де використання алгоритмів генетичного програмування та нейромереж для аналізу змінних, таких як трафік, погодні умови та дорожні умови, дозволяють мінімізувати витрати на паливо та час транспортування, надаючи адаптивні та точні маршрути;

- швидка обробка платежів перевізникам через криптовалюти, де використання смарт-контрактів та криптовалют дозволяє знизити затримки в обробці фінансових операцій, покращуючи грошовий потік та зменшуючи витрати, пов'язані з традиційними фінансовими послугами;

- зменшення помилок та втрат товарів надає можливість інтегрувати блокчейн-технології для відстеження товарів у реальному часі дозволяє уникнути втрат та помилок під час транспортування. Це зменшує необхідність у повторних доставках, що значно знижує загальні логістичні витрати;

Перевага у використанні іноваційного методу (другого підходу) та отримання збільшення ефективності. пояснюється інтеграцією нейромережевих методів і генетичного програмування, які дозволяють суттєво оптимізувати витрати за рахунок багатьох ключових аспектів.

Таким чином, у другому прикладі використання нейромережевих методів та алгоритмів генетичного програмування має безпосередній вплив на кілька ключових аспектів оптимізації логістичних операцій. Ці технології використовуються для покращення процесів через аналіз великих обсягів даних, підвищення точності прогнозів і адаптивного управління ресурсами. Це відбувається внаслідок виконання наступних дій.

Нейронні мережі можуть бути використані для прогнозування дорожніх умов, трафіку, погоди та інших зовнішніх факторів, що впливають на транспортування товарів. Мережі навчаються на великій кількості історичних

даних і можуть передбачати оптимальні маршрути доставки в реальному часі, знижуючи витрати на паливо та час перевезення.

Алгоритми генетичного програмування шукають оптимальні рішення шляхом імітації еволюційних процесів. В логістиці вони можуть використовуватися для постійного покращення планування маршрутів, враховуючи велику кількість змінних. Ці алгоритми вибирають найбільш оптимальні шляхи серед тисяч можливих варіантів на основі багатьох критеріїв, таких як мінімальні витрати на транспортування та час.

Використання нейромереж може допомогти у відстеженні грошових потоків і прогнозуванні фінансових потреб для забезпечення своєчасної оплати перевізникам. Це сприяє кращому управлінню коштами підприємства.

Генетичні алгоритми можуть використовуватися для оптимізації платіжних процесів, обираючи найвигідніші моменти для проведення фінансових транзакцій, знижуючи комісії та затримки при роботі з криптовалютами.

Нейромережі можуть також аналізувати дані про попередні втрати, пошкодження або збої в доставці товарів, допомагаючи виявляти закономірності та попереджувати можливі ризики в майбутньому. Це дозволяє уникнути повторних доставок та знижує витрати на повернення товарів.

Генетичні алгоритми використовуються для оптимізації управління запасами, зменшуючи вартість замороженого капіталу та запобігаючи моральному старінню товарів. Алгоритми можуть пропонувати оптимальні рішення щодо обсягів закупівель, частоти поставок та мінімізації втрат.

Таким чином, у другому прикладі нейромережеві методи та генетичне програмування активно використовуються для адаптивного управління логістичними операціями, забезпечуючи точність прогнозування, зниження витрат та підвищення ефективності всіх етапів процесу. В загальному випадку, використання нейромережевих технологій та генетичного програмування забезпечує не тільки зниження витрат, але й підвищення загальної ефективності

логістичних процесів, забезпечуючи адаптивність та гнучкість в управлінні криптовалютними операціями.

Розглянемо тестування сервісу за допомогою API з використанням HTTP-запитів у форматі JSON через консоль в операційній системі Linux. Для цього, створюємо JSON-файл із вхідними даними для розрахунку EOQ, наприклад, request.json:

```
{
  "average_sales": 50,
  "order_cost": 200,
  "holding_cost": 2
}
```

Далі, надсилаємо запит в консолі за допомогою curl:

```
curl -X POST http://127.0.0.1:5000/api/calculate_eoq \
-H "Content-Type: application/json" \
-d @request.json
```

Після цього, отримуємо JSON-відповідь від сервера наступного складу:

```
{
  "status": "success",
  "optimal_order": 70.71,
  "input_data": {
    "average_sales": 50,
    "order_cost": 200,
    "holding_cost": 2
  }
}
```

Таким чином, можно повторити тестування запитами із різними даними. Наприклад:

echo '{"average_sales": 100, "order_cost": 150, "holding_cost": 3}' > request.json

*curl -X POST http://127.0.0.1:5000/api/calculate_eoq *

*-H "Content-Type: application/json" *

-d @request.json

Така схема дозволяє перевірити коректність роботи API через консоль і забезпечує інтерактивне тестування сервісу з розрахунком EOQ (Economic Order Quantity).

Було проведено тестування для такого сценарію:

- Середньоденний рівень продажів: 50 шт.
- Вартість замовлення: 200 грн.
- Вартість зберігання 1 товару: 2 грн./день

Перетворення даних:

- Річний попит (D): 50 шт/день * 365 днів/рік = 18250 шт/рік

Тестовий сценарій:

1. Введення даних: Вводимо в систему розрахунку EOQ наступні значення:
   - Середньоденний рівень продажів: 50
   - Вартість замовлення: 200
   - Вартість зберігання 1 товару: 2
2. Запуск розрахунку.
3. Перевірка результату.

Очікуваний результат. Система повинна видати результат, який збігається з нашим розрахунком (з урахуванням можливої похибки округлення).

Реальний результат. Було отримано позитивний : система видає результат, близький до 70.71 шт. Це підтверджує коректність роботи алгоритму розрахунку (рисунок 4.6).

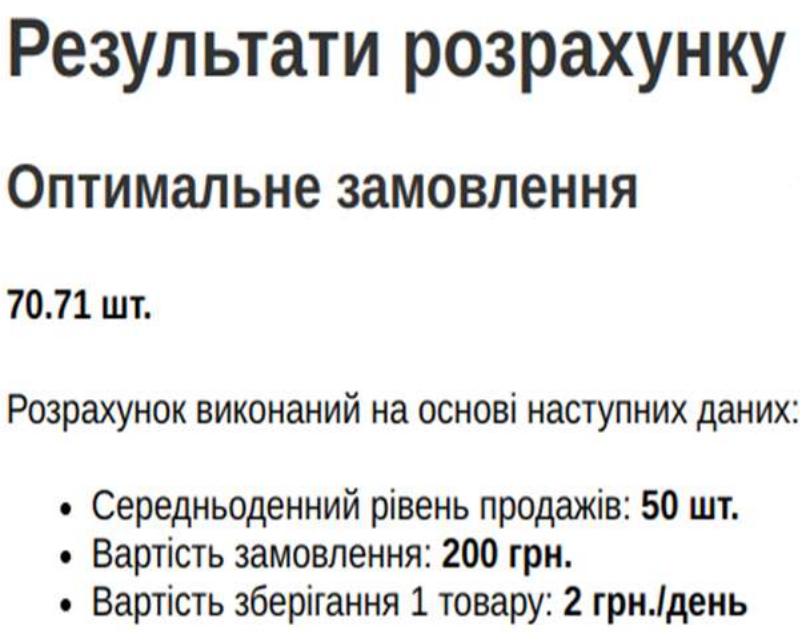


Рисунок 4.6 – Результат тестування

4.4. Висновки

У четвертому розділі були розглянуті особливості та основні етапи тестування систем з управління криптовалютними операціями на основі логістичними операцій. Такі особливості тестування систем управління криптовалютними логістичними операціями на торговельних підприємствах включали кілька ключових аспектах до яких відносились перевірка ефективності фінансових транзакцій, тестування смарт-контрактів, відстеження вантажів, оцінка управління ризиками та автоматизація процесів. В результаті досліджень було визначено, що тестування систем управління криптовалютними логістичними операціями є багатогранним процесом, який вимагає перевірки на різних рівнях: від транзакцій до смарт-контрактів та відстеження вантажів. Тому, для досягнення високої ефективності логістичних операцій на торговельних підприємствах рекомендується використовувати комплексні методи тестування, які охоплюють симуляцію реальних умов експлуатації системи.

До основних етапів тестування систем управління криптовалютними логістичними операціями на торговельних підприємствах належали підготовчий етап, пілотне тестування, оцінка управління ризиками, тестування кібербезпеки, навчання персоналу та підготовка до впровадження, фінальне впровадження та масштабування.

Було доведено, що етапи тестування систем управління криптовалютними логістичними операціями повинні враховувати пілотні проекти, оцінку ризиків, кібербезпеку, підготовку персоналу та фінальне впровадження. Це забезпечить оптимізацію процесів, підвищить ефективність і захистить активи торгових підприємств під час використання криптовалют у логістиці.

Контрольний приклад використання логістичних операцій на торгівельних підприємствах дозволив визначити, що використання нейромережевих технологій та генетичного програмування забезпечує не тільки зниження витрат, але й підвищення загальної ефективності логістичних процесів, забезпечуючи адаптивність та гнучкість в управлінні криптовалютними операціями.

5 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота за темою «Розробка методу та програмного забезпечення для підвищення ефективності логістичних операцій з використанням алгоритмів оптимізації на торгових підприємствах» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Розробка методу та програмного забезпечення для підвищення ефективності логістичних операцій з використанням алгоритмів оптимізації на

торгових підприємствах» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1 [33].

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)				
0	1	2	3	4
Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів

Продовження таблиці 5.1

Бали (за 5-ти бальною шкалою)				
0	1	2	3	4
Практична здійсненність				
Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	4	4
2. Ринкові переваги (наявність аналогів)	4	4	4
3. Ринкові переваги (ціна продукту)	3	3	3
4. Ринкові переваги (технічні властивості)	4	3	3
5. Ринкові переваги (експлуатаційні витрати)	3	3	3
6. Ринкові перспективи (розмір ринку)	3	4	3
7. Ринкові перспективи (конкуренція)	3	3	4
8. Практична здійсненність (наявність фахівців)	3	3	3
9. Практична здійсненність (наявність фінансів)	3	3	3
10. Практична здійсненність (необхідність нових матеріалів)	3	3	3
11. Практична здійсненність (термін реалізації)	3	4	4
12. Практична здійсненність (розробка документів)	4	3	3
Сума балів	40	40	40
Середньоарифметична сума балів СБс	40		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3 [33].

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методу та програмного забезпечення для підвищення ефективності логістичних операцій з використанням алгоритмів оптимізації на торгових підприємствах» становить 40 балів, що, відповідно до таблиці 5.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вищий середнього).

5.2 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Розробка методу та програмного забезпечення для підвищення ефективності логістичних операцій з використанням алгоритмів оптимізації на торгових підприємствах», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.2.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними

ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників (Зо) розраховуємо у відповідності до посадових окладів працівників, за формулою 5.1:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (5.1)$$

де k – кількість посад дослідників залучених до процесу досліджень;

Мпі – місячний посадовий оклад конкретного дослідника, грн;

ti – число днів роботи конкретного дослідника, дні;

Тр – середнє число робочих днів в місяці, Тр=21 день.

Зо = 40000,00 · 60 / 21 = 114285,71 грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	40000,00	1904,76	60,00	114285,71
Інженер-розробник програмного забезпечення	25000,00	1190,48	60,00	71428,57
Консультант-аналітик	22000,00	1047,62	15,00	15714,29
Консультант з економічних питань	20000,00	952,38	15,00	14285,71
Всього				215714,29

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників (Зр) за відповідними найменуваннями робіт НДР розраховуємо за формулою 5.2.

$$З_p = \sum_{i=1}^{n} C_i \cdot t_i \quad , \tag{5.2}$$

де Сі – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

ti – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду Сі можна визначити за формулою 5.3:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} \quad , \tag{5.3}$$

де ММ – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо Мм=8000,00 грн;

Кі – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду [33]

Кс – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань.

tзм – тривалість зміни, год.

С1 = 8000,00 · 1,1 · 1,65 / (21 · 8) = 86,43 грн.

Зр1 = 86,43 · 8,00 = 691,43 грн.

Таблиця 5.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Встановлення допоміжного	8	2	1,1	86,43	691,43

офісного обладнання					

Продовження таблиці 5.5

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Монтаж робочого місця розробника	12	2	1,1	86,43	1037,14
Інсталяція програмного забезпечення	5	5	1,7	133,57	667,86
Інші допоміжні роботи	10	3	1,1	86,43	864,29
Всього					3260,71

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою 5.4:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.4)$$

де Ндод – норма нарахування додаткової заробітної плати. Приймемо 12%.

Здод = (215714,29+ 3260,71) · 12 / 100% = 26277,00 грн.

5.2.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою 5.5:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.5)$$

де Нзп – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (215714,29+ 3260,71+ 26277,00) · 22 / 100% = 53955,44 грн.

5.2.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень.

Витрати на матеріали (М), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою 5.6:

$$M = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (5.6)$$

де Нj – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

Цj – вартість матеріалу j-го найменування, грн/кг;

Кj – коефіцієнт транспортних витрат, (Кj = 1,1 … 1,15);

Вj – маса відходів j-го найменування, кг;

Цвj – вартість відходів j-го найменування, грн/кг.

М1 = 2 · 200,00 · 1,1 - 0,000 · 0,00 = 440,0 грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за од, грн	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Офісний папір	200	2	0	0	440

Папір для записів	170	1	0	0	187

Продовження таблиці 5.6

Найменування матеріалу, марка, тип, сорт	Ціна за од, грн	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Flesh-пам'ять Kingston 32 GB	160	2	0	0	352
Набір офісного працівника JOBMAX, с наповненням (16 предметів)	200	2	0	0	440
Всього					1419

5.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі (Кв), які використовують при проведенні НДР на тему «Розробка методу та програмного забезпечення для підвищення ефективності логістичних операцій з використанням алгоритмів оптимізації на торгових підприємствах» відсутні.

5.2.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Витрати на спецустаткування, які використовують при проведенні НДР на тему «Розробка методу та програмного забезпечення для підвищення ефективності логістичних операцій з використанням алгоритмів оптимізації на торгових підприємствах» відсутні.

5.2.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою 5.7:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \quad , \qquad (5.7)$$

де Ціпрг – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

Кі – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (Кі = 1,10…1,12);

k – кількість найменувань програмних засобів.

Впрг = 4000 · 1 · 1,12 = 4480 грн.

Отримані результати зведемо до таблиці:

Таблиця 5.7 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Sublime text	1	4000	4480
Всього			4480

Intellij Idea є безкоштовною для використання.

5.2.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою 5.8:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.8)$$

де Цб – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

tвик – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

Тв – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

Аобл = (60000,00 · 3) / (5 · 12) = 3000,00 грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.8 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Ноутбук (2 шт)	60000	5	3	3000,00
Wifi-роутер	1000	3	3	83,33
Sublime text	6053,6	3	3	504,47
Приміщення лабораторії	280000	20	3	3500,00
Всього				7087,80

5.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (Ве) розраховуємо за формулою 5.9:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{впі}}{\eta_i}, \quad (5.9)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

ti – тривалість роботи обладнання на етапі дослідження, год;

Це – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо Це = 11,0 грн;

Квпі – коефіцієнт, що враховує використання потужності, Квпі <1;

ηі – коефіцієнт корисної дії обладнання, ηі<1.

Ве = 0,065 · 480,0 · 11,0 · 0,95 / 0,97 = 336,12 грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 5.9 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Ноутбук 1	0,065	480	336,12
Ноутбук 2	0,065	480	336,12
Wifi-роутер	0,01	60	6,46
Робоче місце дослідника	0,15	480	775,67
Всього			1454,38

5.2.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою 5.10:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (5.10)$$

де Нсв – норма нарахування за статтею «Службові відрядження», приймемо Нсв = 20%.

Всв = (215714,29+ 3260,71) · 20 / 100% = 43795,00 грн.

5.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою 5.11:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (5.11)$$

де Нсп – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо Нсп= 30%.

Всп = (215714,29+ 3260,71) · 30 / 100% = 65692,50 грн.

5.2.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою 5.12:

$$I_{в} = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.12)$$

де Нів – норма нарахування за статтею «Інші витрати», приймемо Нів = 50%.

Ів = (215714,29+ 3260,71) · 50 / 100% = 109487,50 грн.

5.2.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою 5.13:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.13)$$

де Ннзв – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо Ннзв = 100%.

Внзв = (215714,29+ 3260,71) · 100 / 100% = 218 975,00 грн.

Витрати на проведення науково-дослідної роботи розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (5.14)$$

Взаг= 751598,62 грн.

Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (5.15)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,9.

ЗВ = 751598,62 / 0,9 = 835109,58 грн.

5.3 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Розробка методу та програмного забезпечення для підвищення ефективності логістичних операцій з використанням алгоритмів оптимізації на торгових підприємствах» передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

1-й рік – 1000 користувачів;

2-й рік – 700 користувачів;

3-й рік – 500 користувачів.

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 3000 користувачів;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 5000 грн;

$\pm\Delta Ц_о$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 500 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою 5.14 [33]:

$$\Delta П_i = (\pm \Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.16)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2024 році ставка податку на додану вартість складає 20%, а коефіцієнт λ=0,8333;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо ρ=45%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2024 році ϑ=18%;

Збільшення чистого прибутку 1-го року:

$\Delta П_1$ =(3000,00·500,00+550,00·1000)·0,83·0,45·(1-0,18/100%)= 2151639 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2$ =(3000,00·500,00+550,00·(1000+700))·0,83·0,45·(1-0,18/100%)=3335040,45 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3$ =(3000,00·500,00+550,00·(1000+700+500))·0,83·0,45·(1-0,18/100%)= 4180327,2 грн.

Приведена вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки 5.17:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.17)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,3;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП =$$

2151639/(1+0,3)1+3335040,45/(1+0,3)2+4180327,2/(1+0,3)3=5531247,02 грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки (формула 5.18):

$$PV = k_{інв} \cdot ЗВ, \tag{5.18}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$ =2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 835109,58 грн.

$PV = k_{інв} \cdot ЗВ$ = 2 · 835109,58 = 1670219,159 грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме (формула 5.19):

$$E_{абс} = ПП - PV \tag{5.19}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 5531247,02 грн;

PV – теперішня вартість початкових інвестицій, 1670219,159 грн.

$E_{абс} = ПП - PV$ = 5531247,02 - 1670219,159 = 3861027,86 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки (5.20):

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.20)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 3861027,86 грн;

PV – теперішня вартість початкових інвестицій, 1670219,159 грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1$ = (1+3861027,86/1670219,159)1/3= 0,49.

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$ (формула 5.20):

$$\tau_{мін} = d + f, \quad (5.21)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d =0,11;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,18.

$\tau_{мін}$ = 0,11+0,18 = 0,29 < 0,49 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Розробка методу та програмного забезпечення для підвищення ефективності логістичних операцій з використанням алгоритмів оптимізації на торгових підприємствах» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки (формула 5.22):

$$T_{ок} = \frac{1}{E_в}, \tag{5.22}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,49= 2,04 року.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

5.3 Висновки

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методу та програмного забезпечення для підвищення ефективності логістичних операцій з використанням алгоритмів оптимізації на торгових підприємствах» становить 40 балів, що, свідчить про комерційну

важливість проведення даних досліджень (рівень комерційного потенціалу розробки вищий середнього).

Також термін окупності становить 2,04 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Розробка методу та програмного забезпечення для підвищення ефективності логістичних операцій з використанням алгоритмів оптимізації на торгових підприємствах».

ВИСНОВКИ

У магістерській кваліфікаційній роботі розроблено метод інтеграції засобів оптимізації в систему розрахунку логістичних операцій, який поєднує можливість оптимізації за евристичними алгоритмами і можливість реалізації з використанням лінійного програмування, що дозволяє враховувати змінність параметрів у часі, таких як маршрутизація, управління запасами та розподіл ресурсів, і підвищити ефективність логістичних процесів.

Розроблено метод оптимізації логістичних операцій, який базується на комбінованому використанні методів мінімізації витрат на транспортування та зберігання, що дозволяє надавати підприємствам комплексний інструмент для ефективного управління всіма етапами логістики та забезпечує більш точне моделювання логістичних процесів зі скороченням витрат на логістику.

Також було виконано:

- обґрунтовано вибір методів аналізу основних підходів до оптимізації логістичних операцій на підприємствах торгівлі;
- розроблено метод підвищення ефективності логістичних операцій на торгових підприємствах;
- розроблено метод інтеграції засобів оптимізації в систему розрахунку логістичних операцій;
- визначено функціональні вимоги до основних програмних модулів розробленого програмного засобу;
- розроблено програмний засіб системи;
- проведено тестування роботи програми.

Магістерська кваліфікаційна робота оформлена відповідно до вказівок [35-36].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Slushnyi Vasyl, Khoshaba Oleksandr. Development of Optimization Algorithms and Software for Enhancing Efficiency in Logistics Operations at Trade Enterprises /Матеріали IV конференції Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів. "Комп'ютерні ігри і мультимедіа як інноваційний підхід до комунікації - 2024". Одеса, 26-27 вересня 2024 р. - Одеса, Видавництво ОНТУ, 2024 р. - с.192-194.

2. Григорак М.І. Інформаційні технології в логістиці: рек. до виконання розрахунк. роботи: навч. посіб. для здобувачів ступеня бакалавра за освіт. програмою «Логістика» спец. 073 «Менеджмент». Київ: КПІ ім. Ігоря Сікорського, 2023. - 65 с.

3. Ольхова М. В. Оптимізація логістичних процесів: конспект лекцій для студентів другого (магістерського) рівня вищої освіти денної і заочної форм навчання спеціальності / М. В. Ольхова ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2021. – 75 с.

4. Логістика майбутнього: ефективні рішення для торгівлі [Електронний ресурс] : тези доп. Міжнар. наук.-практ. інтернет-конф. (Київ, 20 квіт. 2023 р.) / відп. ред. Н. Б. Ільченко. – Київ : Держ. торг.-екон. ун-т, 2023. – 239 с.

5. Савченко Ю.В., Олійник О.М., Савченко О.Ю. Управління дистрибуційними мережами: навч. посібник. Київ: Центр навчальної літератури, 2019. - 320 с.

6. Закон України «Про валюту і валютні операції» від 21.06.2018 р. № 2473-VIII. [Електронний ресурс]. – Режим доступу: https://zakon.rada.gov.ua (дата звернення: 10.08.2024).

7. Закон України «Про внесення змін до Закону України «Про бухгалтерський облік та фінансову звітність в Україні» щодо удосконалення

деяких положень» № 2164-VIII від 05 жовтня 2017 р. [Електронний ресурс]. − Режим доступу: http:// www.zakon.rada.gov.ua. (дата звернення: 12.08.2024)

8. Закон України «Про зовнішньоекономічну діяльність» від 16.04.2001 (із змінами і доповненнями) // Відомості Верховної Ради України. – 1991. – № 29. [Електронний ресурс] – Режим доступу: https:// zakon2.rada.gov.ua (дата звернення: 17.08.2024)

9. Колодізєва Т. О. Методичне забезпечення оцінки ефективності логістичної діяльності підприємств: монографія / Т. О. Колодізєва, Г. Р. Руденко. – Х.: Вид. ХНЕУ, 2022. – 292 с.

10. Колодка Я.В. Особливості та оптимізація логістичних систем підприємств сільськогосподарської галузі / Я.В. Колодка // Інноваційна економіка. – 2024. – №2. – с.131-136

11. Савіна Н. Б. Інвестування у логістичні системи: монографія / Н.Б. Савіна; Нац. ун-т «Львівська політехніка». – Львів: Вид-во Львівської політехніки, 2023. – 328 с.26

12. Сметаніна А. В. Взаємодія служби логістики із суміжними структурними підрозділами / А. В. Сметаніна // Управління розвитком. – 2024. – № 1. – С. 143-145.

13. Солодка О. В. Реінжиніринг логістичних бізнес-процесів як спосіб їх вдосконалення / О. В. Солодка // Вісник НУ «Львівська політехніка». – 2020. – № 2. – С. 21–23.

14. Струтинська Ірина. Проблема визначення класу логістичного центру на вітчизняному ринку логістичної нерухомості / Ірина Струтинська //Актуальні проблеми економіки. – 2023. – № 6. – С. 211–219.

16. Тридід О. М., Таньков К. М. Логістичний менеджмент: навчальний посібник / За ред. проф., д-ра екон. наук О. М. Тридіда. – Х.: ВД«ІНЖЕК», 2022. – 224 с.

17. Шевчун М. Б. Особливості управління логістичними процесами на торговельних підприємствах / М. Б. Шевчун // Сталий розвиток економіки. – 2023. – № 3. – С. 353-356.

18. Хаджинова, О. В. Логістична стратегія управління витратами великого багатопрофільного промислового підприємства: автореф. дис. канд. екон. наук: спец. 08.06.01 „Економіка, організація і управління підприємствами” / О. В. Хаджинова. – Донецьк, 2019. – 23 с.

19. Чухрай Н. Логістичне обслуговування : підручник для вузів / Н.Чухрай ; М-во освіти і науки України. – Л. : Львів. політехніка, 2022. – 292 с.

20. Шевців Л. Ю. Логістичні витрати підприємства: формування та оцінювання: [монографія] / Л. Ю. Шевців, І. Петецький. – Львів: НУ «Львівська політехніка», 2021. – 244 с.

21. Шевців Л. Ю. Логістичні витрати підприємства: формування та оцінювання: [монографія] / Л. Ю. Шевців, І. Петецький. – Львів: НУ «Львівська політехніка», 2021. – 244 с.

22. Окландер М.А. Логістика: підручник. Київ: Центр учбової літератури, 2018. 346 с.

23. CSCMP Supply Chain Management Definitions and Glossary. [Електронний ресурс] – Режим доступу: https://cscmp.org/CSCMP/Educate/SCM_Definitions_and_Glossary_of_Terms.aspx (дата звернення: 07.09.2024).

24. Oskarsson B. Total Cost Analysis in Logistics - Practical Execution, Learning, and Teaching in Higher Education: [Електронний ресурс] – Режим доступу: https://www.divaportal.org/smash/get/diva2:1367153/FULLTEXT01.pdf (дата звернення: 07.09.2024).

25. Warehousing and distribution network design from a Third-Party Logistics (3PL) company perspective: веб-сайт. [Електронний ресурс] – Режим доступу:

https://www.sciencedirect.com/science/article/pii/ S2405896322022200 (дата звернення: 10.09.2024).

26. Савченко Ю.В., Олійник О.М., Савченко О.Ю. Управління дистрибуційними мережами: навч. посібник. Київ: Центр навчальної літератури, 2019. 320 с.

27. Олефіренко О.М. Теоретичні основи визначення інструментів та каналів збуту продукції промислових підприємств: веб-сайт. [Електронний ресурс] – Режим доступу: URL: http://www.economy.nayka.com.ua/ ?op=1&z=5848 (дата звернення: 10.09.2024).

28. Коніщева Н.Й. Управління логістичною діяльністю промислових підприємств / Н.Й. Коніщева, Н.В. Трушкі на. 2023. № 1 (27). С.114–124.

29. Клімова І.Г. Проблеми та передумови використання логістики в Україні / І.Г. Клімова // Держава та регіони. 2024. № 3. С. 143–147.

30. Паласюк Б. Логістичне управління підприємством: сутність і основні принципи / Б. Паласюк // Галицький економічний вісник. 2022. № 3(36). С. 166–170.

31. Крикавський Є.В. Логістика. Основи теорії: підруч. для ВНЗ / Є.В. Крикавський. Нац. унт. «Львівська політехніка»;. Л.: ІнтелектЗахід, 2024. 414с.

32. Шандрівська О.Є. Логістичний менеджмент. Теоретичні основию/ О.Є. Шандрівська, В.В. Кузяк, Н.І. Хтей. Львів: Львівська політехніка, 2022. 195 с.

33. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. Вінниця : ВНТУ, 2021. 42 с.

34. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа. Вінниця : ВНТУ, 2016. 113 с.

35. Положення про кваліфікаційну роботу на другому (вищому) рівні вищої освіти. Розробники: А.О. Семенов, Л.П. Громова, Т.В. Макарова, О.В.Сердюк. Вінниця: ВНТУ, 2021. – 60 с.

36. Методичні вказівки до виконання магістерської кваліфікаційної роботи для студентів спеціальності 121 «Інженерія програмного забезпечення» / уклад.: О. Н. Романюк, Г. О. Черноволик. – Вінниця : ВНТУ, 2022. – 50 с.

ДОДАТКИ

114

Додаток А. Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
"_19_" _вересня_____2024 р.

Технічне завдання
на магістерську кваліфікаційну роботу «Розробка методу та програмного забезпечення для підвищення ефективності логістичних операцій з використанням алгоритмів оптимізації на торгових підприємствах» за спеціальністю
121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
_____ к.т.н., доцент О.Г. Черноволик
"_19_" _вересня_____2024 р.
Виконав:
_____студент гр. 2ПІ-22м В.В. Слушний
"_19_" _вересня_____2024 р.

Вінниця – 2024 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методу та програмного забезпечення для підвищення ефективності логістичних операцій з використанням алгоритмів оптимізації на торгових підприємствах».

Галузь застосування – програмне забезпечення логістичних операцій з використанням алгоритмів оптимізації на торгових підприємствах.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № 310 від 17 вересня 2024 року ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення ефективності логістичних операцій на підприємствах торгівлі шляхом розробки програмного забезпечення з використанням алгоритмів оптимізації, що дозволить покращити процеси логістики та ефективно розподілити ресурси підприємства.

Призначення роботи – розробка методів і засобів підвищення ефективності логістичних операцій на підприємствах торгівлі.

4 Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Slushnyi Vasyl, Khoshaba Oleksandr. Development of Optimization Algorithms and Software for Enhancing Efficiency in Logistics Operations at Trade Enterprises /Матеріали IV конференції Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів. "Комп'ютерні ігри і мультимедіа як інноваційний підхід до комунікації - 2024". Одеса, 26-27 вересня 2024 р. - Одеса, Видавництво ОНТУ, 2024 р. - с.192-194.

2. Григорак М.І. Інформаційні технології в логістиці: рек. до виконання розрахунк. роботи: навч. посіб. для здобувачів ступеня бакалавра за освіт. програмою «Логістика» спец. 073 «Менеджмент». Київ: КПІ ім. Ігоря Сікорського, 2023. - 65 с.

3. Ольхова М. В. Оптимізація логістичних процесів: конспект лекцій для студентів другого (магістерського) рівня вищої освіти денної і заочної форм навчання спеціальності / М. В. Ольхова ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2021. – 75 с.

4. Логістика майбутнього: ефективні рішення для торгівлі [Електронний ресурс] : тези доп. Міжнар. наук.-практ. інтернет-конф. (Київ, 20 квіт. 2023 р.) / відп. ред. Н. Б. Ільченко. – Київ : Держ. торг.-екон. ун-т, 2023. – 239 с.

5. Савченко Ю.В., Олійник О.М., Савченко О.Ю. Управління дистрибуційними мережами: навч. посібник. Київ: Центр навчальної літератури, 2019. - 320 с.

5. Технічні вимоги

Для використання методу точки безбитковості: постійні витрати на логістичні операції не менше 10 000 грн (FC, в грн), змінні витрати на одиницю доставки не менше 50 грн (VC, в грн), вартість доставки одиниці товару не менше 100 грн (SP, в грн), кількість одиниць товару для доставки за один цикл не менше 200 одиниць (N, в шт), загальна кількість товарів, які необхідно доставити для забезпечення покриття всіх витрат, не менше 2 000 одиниць (Q, в шт), точка беззбитковості у грошовому вираженні не менше 200 000 грн (BEP, в грн), середній час виконання доставки одного замовлення не більше 48 годин (LT, у годинах), максимальна допустима кількість транспортних засобів у роботі за один цикл не менше 5 одиниць (VT, в шт).

6. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз задачі дослідження напрямків моделей генетичного програмування для виконання задач логістики	20.09.2024-28.09.2024
2	Дослідження моделей та методів ефективності логістичних операцій	29.03.2024-11.10.2024
3	Розробка та впровадження програмних модулів	12.10.2024-24.10.2024
4	Особливості та тестування етапів тестування систем з управління криптовалютними операціями	24.10.2024-15.11.2024
5	Економічна частина	15.11.24-30.11.24

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

118

Додаток Б – Протокол перевірки магістерської кваліфікаційної роботи на наявність текстових запозичень

ПРОТОКОЛ ПЕРЕВІРКИ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка методу та програмного забезпечення для підвищення ефективності логістичних операцій з використанням алгоритмів оптимізації на торгових підприємствах.

Тип роботи: магістерська кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, 2ПІ-23м

Науковий керівник: к.т.н. доц. Черноволик Г. О.

Turnitin	
Оригінальність	94%
Схожість	6%

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений з повним звітом подібності, який був згенерований Системою щодо роботи "Розробка методу та програмного забезпечення для підвищення якості персоналізованого навчання учнів у приватних закладах освіти".

Автор роботи _____ Слушний В.В.

Опис прийнятого рішення: **допустити до захисту**

Особа, відповідальна за перевірку _____ Черноволик Г.О.
(підпис) (прізвище, ініціали)

Експерт (за потреби посада) _____ _____
(підпис) (прізвище, ініціали)

Додаток В. Лістинг програми

HTML-код для вхідних даних модуля оптимізації управління запасами

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Модуль оптимізації управління запасами</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>

    <div class="container">
        <h1>Модуль оптимізації управління запасами</h1>

        <form id="inventoryForm">
            <div class="form-group">
                <label for="currentStock">Поточний рівень запасів (шт.):</label>
                <input type="number" id="currentStock" required>
            </div>

            <div class="form-group">
                <label for="averageSales">Середньоденний рівень продажів
(шт.):</label>
                <input type="number" id="averageSales" required>
            </div>

            <div class="form-group">
```

```
                <label for="leadTime">Час доставки від постачальника
(днів):</label>
                <input type="number" id="leadTime" required>
            </div>

            <div class="form-group">
                <label for="minStock">Мінімальний рівень запасів (шт.):</label>
                <input type="number" id="minStock" required>
            </div>

            <div class="form-group">
                <label for="maxStock">Максимальний рівень запасів
(шт.):</label>
                <input type="number" id="maxStock" required>
            </div>

            <div class="form-group">
                <label for="orderCost">Вартість замовлення (грн.):</label>
                <input type="number" id="orderCost" required>
            </div>

            <div class="form-group">
                <label for="holdingCost">Вартість зберігання 1 товару на складі
(грн/день):</label>
                <input type="number" id="holdingCost" required>
            </div>

            <button type="button" onclick="calculateOrder()">Розрахувати
оптимальне замовлення</button>
        </form>
```

```
        <div id="result">
            <h2>Результати:</h2>
            <p id="optimalOrder"></p>
        </div>
    </div>

    <script src="script.js"></script>
</body>
</html>
```

CSS-код для стилізації модуля оптимізації

```
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    padding: 20px;
}

.container {
    max-width: 600px;
    margin: auto;
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

h1, h2 {
    text-align: center;
```

```
}

.form-group {
    margin-bottom: 15px;
}

label {
    display: block;
    margin-bottom: 5px;
}

input {
    width: 100%;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 4px;
}

button {
    width: 100%;
    padding: 10px;
    background-color: #007bff;
    border: none;
    color: white;
    font-size: 16px;
    border-radius: 4px;
    cursor: pointer;
}

button:hover {
```

```
        background-color: #0056b3;
    }

    #result {
        margin-top: 20px;
        text-align: center;
    }

    #optimalOrder {
        font-weight: bold;
    }
```

JavaScript-код для обробки вхідних даних і розрахунку оптимального замовлення модуля оптимізації

```
    function calculateOrder() {
        // Отримання даних з полів форми
        const currentStock =
parseFloat(document.getElementById('currentStock').value);
        const averageSales =
parseFloat(document.getElementById('averageSales').value);
        const leadTime = parseFloat(document.getElementById('leadTime').value);
        const minStock = parseFloat(document.getElementById('minStock').value);
        const maxStock = parseFloat(document.getElementById('maxStock').value);
        const orderCost = parseFloat(document.getElementById('orderCost').value);
        const holdingCost =
parseFloat(document.getElementById('holdingCost').value);

        // Перевірка на валідність даних
        if (isNaN(currentStock) || isNaN(averageSales) || isNaN(leadTime) ||
isNaN(minStock) || isNaN(maxStock) || isNaN(orderCost) || isNaN(holdingCost)) {
```

```
            alert("Будь ласка, заповніть всі поля коректно.");
            return;
        }

        // Розрахунок оптимального замовлення (модель EOQ)
        const demandDuringLeadTime = averageSales * leadTime;
        const reorderPoint = demandDuringLeadTime + minStock;
        const optimalOrder = Math.sqrt((2 * orderCost * averageSales) /
holdingCost);

        // Виведення результатів
        document.getElementById('optimalOrder').innerHTML = `
            Оптимальний розмір замовлення: ${optimalOrder.toFixed(2)} шт.<br>
            Точка замовлення: ${reorderPoint.toFixed(2)} шт.
        `;
    }
```

Файл pom.xml Spring Boot проекту модуля оптимізації

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
    </dependency>
</dependencies>
```

Контролер InventoryController.java модуля оптимізації:

```
package org.logistics.inventory;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api/inventory")
public class InventoryController {

    @PostMapping("/calculateOrder")
    public InventoryResponse calculateOrder(@RequestBody InventoryRequest request) {
        // Отримуємо дані з запиту
        double averageSales = request.getAverageSales();
        double leadTime = request.getLeadTime();
        double minStock = request.getMinStock();
        double orderCost = request.getOrderCost();
        double holdingCost = request.getHoldingCost();

        // Розрахунок оптимального замовлення (модель EOQ)
        double demandDuringLeadTime = averageSales * leadTime;
        double reorderPoint = demandDuringLeadTime + minStock;
        double optimalOrder = Math.sqrt((2 * orderCost * averageSales) / holdingCost);

        // Повертаємо результат
        return new InventoryResponse(optimalOrder, reorderPoint);
    }
}
```

Модель запиту InventoryRequest.java модуля оптимізації:

```
package org.logistics.inventory;
```

```
public class InventoryRequest {
    private double averageSales;
    private double leadTime;
    private double minStock;
    private double orderCost;
    private double holdingCost;

    // Геттери та сеттери
    public double getAverageSales() { return averageSales; }
    public void setAverageSales(double averageSales) { this.averageSales =
averageSales; }
    public double getLeadTime() { return leadTime; }
    public void setLeadTime(double leadTime) { this.leadTime = leadTime; }
    public double getMinStock() { return minStock; }
    public void setMinStock(double minStock) { this.minStock = minStock; }
    public double getOrderCost() { return orderCost; }
    public void setOrderCost(double orderCost) { this.orderCost = orderCost; }
    public double getHoldingCost() { return holdingCost; }
    public void setHoldingCost(double holdingCost) { this.holdingCost =
holdingCost; }
}
```

Модель відповіді InventoryResponse.java модуля оптимізації:

```
package org.logistics.inventory;
public class InventoryResponse {
    private double optimalOrder;
    private double reorderPoint;

    public InventoryResponse(double optimalOrder, double reorderPoint) {
        this.optimalOrder = optimalOrder;
```

```
        this.reorderPoint = reorderPoint;
    }

    // Геттери
    public double getOptimalOrder() { return optimalOrder; }
    public double getReorderPoint() { return reorderPoint; }
}
```

Backend на Kotlin з використанням Ktor, створення проекту на Kotlin з Ktor, підготовка файлу build.gradle.kts:

```
dependencies {
    implementation("io.ktor:ktor-server-core:2.0.0")
    implementation("io.ktor:ktor-server-netty:2.0.0")
    implementation("io.ktor:ktor-serialization:2.0.0")
    implementation("ch.qos.logback:logback-classic:1.2.6")
}
```

Створення контролера в Ktor, основний файл Application.kt:

```
package org.logistics.inventory
import io.ktor.application.*
import io.ktor.features.ContentNegotiation
import io.ktor.http.*
import io.ktor.request.*
import io.ktor.response.*
import io.ktor.routing.*
import io.ktor.serialization.*
import io.ktor.server.engine.embeddedServer
import io.ktor.server.netty.Netty

fun main() {
```

```
        embeddedServer(Netty, port = 8080) {
            install(ContentNegotiation) {
                json()
            }
            routing {
                post("/api/inventory/calculateOrder") {
                    val request = call.receive<InventoryRequest>()

                    // Отримуємо дані та виконуємо розрахунок
                    val averageSales = request.averageSales
                    val leadTime = request.leadTime
                    val minStock = request.minStock
                    val orderCost = request.orderCost
                    val holdingCost = request.holdingCost

                    // Розрахунок оптимального замовлення (модель EOQ)
                    val demandDuringLeadTime = averageSales * leadTime
                    val reorderPoint = demandDuringLeadTime + minStock
                    val optimalOrder = kotlin.math.sqrt((2 * orderCost * averageSales) /
holdingCost)

                    // Відповідь з результатом
                    call.respond(HttpStatusCode.OK, InventoryResponse(optimalOrder,
reorderPoint))
                }
            }
        }.start(wait = true)
    }
```

Модель запиту InventoryRequest.kt:

```
package org.logistics.inventory
import kotlinx.serialization.Serializable

@Serializable
data class InventoryRequest(
    val averageSales: Double,
    val leadTime: Double,
    val minStock: Double,
    val orderCost: Double,
    val holdingCost: Double
)
```

Модель відповіді InventoryResponse.kt:

```
package org.logistics.inventory
import kotlinx.serialization.Serializable

@Serializable
data class InventoryResponse(
    val optimalOrder: Double,
    val reorderPoint: Double
)

public class AllocationResponse {
    private List<String> allocationResult;

    public AllocationResponse(List<String> allocationResult) {
        this.allocationResult = allocationResult;
    }

    public List<String> getAllocationResult() {
```

```
        return allocationResult;
    }
}
```

HTML-код для вхідних даних модуля алгоритмів розподілу ресурсів

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Алгоритми розподілу ресурсів</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        <h1>Модуль алгоритмів розподілу ресурсів</h1>
        <form id="resourceForm">
            <div class="form-group">
                <label for="resources">Кількість ресурсів (шт.):</label>
                <input type="number" id="resources" required>
            </div>

            <div class="form-group">
                <label for="tasks">Кількість завдань (шт.):</label>
                <input type="number" id="tasks" required>
            </div>

            <div class="form-group">
                <label for="resourceCapacity">Місткість одного ресурсу
(шт.):</label>
```

```
            <input type="number" id="resourceCapacity" required>
        </div>

        <div class="form-group">
            <label for="taskDemand">Попит на ресурси для кожного завдання (шт.):</label>
            <input type="text" id="taskDemand" placeholder="Введіть через кому, напр.: 10, 20, 15" required>
        </div>

        <button type="button" onclick="calculateAllocation()">Розрахувати розподіл ресурсів</button>
    </form>

    <div id="result">
        <h2>Результати розподілу:</h2>
        <p id="allocationResult"></p>
    </div>
</div>

<script src="script.js"></script>
</body>
</html>
```

CSS-код для стилізації веб-інтерфейсу модуля алгоритмів розподілу ресурсів

```
body {
    font-family: Arial, sans-serif;
    background-color: #f0f0f0;
    margin: 0;
```

```
    padding: 20px;
}

.container {
    max-width: 600px;
    margin: auto;
    background-color: white;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

h1, h2 {
    text-align: center;
}

.form-group {
    margin-bottom: 15px;
}

label {
    display: block;
    margin-bottom: 5px;
}

input {
    width: 100%;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 5px;
```

```
}

button {
    width: 100%;
    padding: 10px;
    background-color: #007bff;
    border: none;
    color: white;
    font-size: 16px;
    border-radius: 5px;
    cursor: pointer;
}

button:hover {
    background-color: #0056b3;
}

#result {
    margin-top: 20px;
    text-align: center;
}

#allocationResult {
    font-weight: bold;
}
```

JavaScript-код для обробки вхідних даних і розрахунку оптимального розподілу модуля алгоритмів розподілу ресурсів

```
function calculateAllocation() {
    // Отримуємо значення полів форми
```

```
        const resources = parseInt(document.getElementById('resources').value);
        const tasks = parseInt(document.getElementById('tasks').value);
        const resourceCapacity =
parseFloat(document.getElementById('resourceCapacity').value);
        const taskDemandInput = document.getElementById('taskDemand').value;

        // Перетворення рядка попиту на масив чисел
        const taskDemand = taskDemandInput.split(',').map(Number);

        // Перевірка валідності даних
        if (taskDemand.length !== tasks) {
            alert("Кількість завдань і попиту на ресурси повинні збігатися.");
            return;
        }

        // Ініціалізуємо результати розподілу
        let totalCapacity = resources * resourceCapacity;
        let allocationResult = [];

        for (let i = 0; i < tasks; i++) {
            if (totalCapacity >= taskDemand[i]) {
                allocationResult.push(`Завдання ${i + 1}: ${taskDemand[i]}
ресурсів`);
                totalCapacity -= taskDemand[i];
            } else {
                allocationResult.push(`Завдання ${i + 1}: ${totalCapacity} ресурсів
(недостатньо)`);
                totalCapacity = 0;
            }
        }
```

```
        // Виведення результатів
        document.getElementById('allocationResult').innerHTML =
allocationResult.join('<br>');
    }
```

Файл pom.xml модуля алгоритмів розподілу ресурсів

```
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
        </dependency>
    </dependencies>
```

Контролер ResourceAllocationController.java модуля алгоритмів розподілу ресурсів

```
    package org.logistics.resourceallocation;
    import org.springframework.web.bind.annotation.*;
    import java.util.ArrayList;
    import java.util.List;

    @RestController
    @RequestMapping("/api/resources")
    public class ResourceAllocationController {

        @PostMapping("/allocate")
```

```
    public AllocationResponse allocateResources(@RequestBody
AllocationRequest request) {
        int resources = request.getResources();
        int resourceCapacity = request.getResourceCapacity();
        List<Integer> taskDemand = request.getTaskDemand();

        int totalCapacity = resources * resourceCapacity;
        List<String> allocationResult = new ArrayList<>();

        for (int i = 0; i < taskDemand.size(); i++) {
            int demand = taskDemand.get(i);
            if (totalCapacity >= demand) {
                allocationResult.add("Завдання " + (i + 1) + ": " + demand + "
ресурсів");
                totalCapacity -= demand;
            } else {
                allocationResult.add("Завдання " + (i + 1) + ": " + totalCapacity + "
ресурсів (недостатньо)");
                totalCapacity = 0;
            }
        }

        return new AllocationResponse(allocationResult);
    }
}
```

Модель запиту AllocationRequest.java модуля алгоритмів розподілу ресурсів

```
package org.logistics.resourceallocation;
import java.util.List;
```

```
public class AllocationRequest {
    private int resources;
    private int resourceCapacity;
    private List<Integer> taskDemand;

    public int getResources() {
        return resources;
    }

    public void setResources(int resources) {
        this.resources = resources;
    }

    public int getResourceCapacity() {
        return resourceCapacity;
    }

    public void setResourceCapacity(int resourceCapacity) {
        this.resourceCapacity = resourceCapacity;
    }

    public List<Integer> getTaskDemand() {
        return taskDemand;
    }

    public void setTaskDemand(List<Integer> taskDemand) {
        this.taskDemand = taskDemand;
    }
}
```

Модель відповіді AllocationResponse.java модуля алгоритмів розподілу ресурсів

```
package org.logistics.resourceallocation;
import java.util.List;

public class AllocationResponse {
    private List<String> allocationResult;

    public AllocationResponse(List<String> allocationResult) {
        this.allocationResult = allocationResult;
    }

    public List<String> getAllocationResult() {
        return allocationResult;
    }
}
```

HTML-код для вхідних даних модуля оптимізації управління запасами

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Модуль оптимізації управління запасами</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>

    <div class="container">
        <h1>Модуль оптимізації управління запасами</h1>

        <form id="inventoryForm">
            <div class="form-group">
                <label for="currentStock">Поточний рівень запасів (шт.):</label>
                <input type="number" id="currentStock" required>
            </div>

            <div class="form-group">
                <label for="averageSales">Середньоденний рівень продажів
(шт.):</label>
                <input type="number" id="averageSales" required>
            </div>

            <div class="form-group">
                <label for="leadTime">Час доставки від постачальника
(днів):</label>
                <input type="number" id="leadTime" required>
```

```
            </div>

            <div class="form-group">
                <label for="minStock">Мінімальний рівень запасів (шт.):</label>
                <input type="number" id="minStock" required>
            </div>

            <div class="form-group">
                <label for="maxStock">Максимальний рівень запасів
(шт.):</label>
                <input type="number" id="maxStock" required>
            </div>

            <div class="form-group">
                <label for="orderCost">Вартість замовлення (грн.):</label>
                <input type="number" id="orderCost" required>
            </div>

            <div class="form-group">
                <label for="holdingCost">Вартість зберігання 1 товару на складі
(грн/день):</label>
                <input type="number" id="holdingCost" required>
            </div>

            <button type="button" onclick="calculateOrder()">Розрахувати
оптимальне замовлення</button>
        </form>

        <div id="result">
            <h2>Результати:</h2>
```

```
            <p id="optimalOrder"></p>
        </div>
    </div>

    <script src="script.js"></script>
</body>
</html>
```

CSS-код для стилізації модуля оптимізації

```
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    padding: 20px;
}

.container {
    max-width: 600px;
    margin: auto;
    background: white;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

h1, h2 {
    text-align: center;
}

.form-group {
```

```
    margin-bottom: 15px;
}

label {
    display: block;
    margin-bottom: 5px;
}

input {
    width: 100%;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 4px;
}

button {
    width: 100%;
    padding: 10px;
    background-color: #007bff;
    border: none;
    color: white;
    font-size: 16px;
    border-radius: 4px;
    cursor: pointer;
}

button:hover {
    background-color: #0056b3;
}
```

```
    #result {
        margin-top: 20px;
        text-align: center;
    }

    #optimalOrder {
        font-weight: bold;
    }
```

JavaScript-код для обробки вхідних даних і розрахунку оптимального замовлення модуля оптимізації

```
    function calculateOrder() {
        // Отримання даних з полів форми
        const currentStock =
parseFloat(document.getElementById('currentStock').value);
        const averageSales =
parseFloat(document.getElementById('averageSales').value);
        const leadTime = parseFloat(document.getElementById('leadTime').value);
        const minStock = parseFloat(document.getElementById('minStock').value);
        const maxStock = parseFloat(document.getElementById('maxStock').value);
        const orderCost = parseFloat(document.getElementById('orderCost').value);
        const holdingCost =
parseFloat(document.getElementById('holdingCost').value);

        // Перевірка на валідність даних
        if (isNaN(currentStock) || isNaN(averageSales) || isNaN(leadTime) ||
isNaN(minStock) || isNaN(maxStock) || isNaN(orderCost) || isNaN(holdingCost)) {
            alert("Будь ласка, заповніть всі поля коректно.");
            return;
        }
```

```
    // Розрахунок оптимального замовлення (модель EOQ)
    const demandDuringLeadTime = averageSales * leadTime;
    const reorderPoint = demandDuringLeadTime + minStock;
    const optimalOrder = Math.sqrt((2 * orderCost * averageSales) /
holdingCost);

    // Виведення результатів
    document.getElementById('optimalOrder').innerHTML = `
      Оптимальний розмір замовлення: ${optimalOrder.toFixed(2)} шт.<br>
      Точка замовлення: ${reorderPoint.toFixed(2)} шт.
    `;
}
```

Файл pom.xml Spring Boot проекту модуля оптимізації

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
  </dependency>
</dependencies>
```

Контролер InventoryController.java модуля оптимізації:

```
package com.example.inventory;
import org.springframework.web.bind.annotation.*;
```

```
@RestController
@RequestMapping("/api/inventory")
public class InventoryController {

    @PostMapping("/calculateOrder")
    public InventoryResponse calculateOrder(@RequestBody InventoryRequest request) {
        // Отримуємо дані з запиту
        double averageSales = request.getAverageSales();
        double leadTime = request.getLeadTime();
        double minStock = request.getMinStock();
        double orderCost = request.getOrderCost();
        double holdingCost = request.getHoldingCost();

        // Розрахунок оптимального замовлення (модель EOQ)
        double demandDuringLeadTime = averageSales * leadTime;
        double reorderPoint = demandDuringLeadTime + minStock;
        double optimalOrder = Math.sqrt((2 * orderCost * averageSales) / holdingCost);

        // Повертаємо результат
        return new InventoryResponse(optimalOrder, reorderPoint);
    }
}
```

Модель запиту InventoryRequest.java модуля оптимізації:

```
package com.example.inventory;
public class InventoryRequest {
    private double averageSales;
    private double leadTime;
```

```
    private double minStock;
    private double orderCost;
    private double holdingCost;

    // Геттери та сеттери
    public double getAverageSales() { return averageSales; }
    public void setAverageSales(double averageSales) { this.averageSales =
averageSales; }
    public double getLeadTime() { return leadTime; }
    public void setLeadTime(double leadTime) { this.leadTime = leadTime; }
    public double getMinStock() { return minStock; }
    public void setMinStock(double minStock) { this.minStock = minStock; }
    public double getOrderCost() { return orderCost; }
    public void setOrderCost(double orderCost) { this.orderCost = orderCost; }
    public double getHoldingCost() { return holdingCost; }
    public void setHoldingCost(double holdingCost) { this.holdingCost =
holdingCost; }
}
```

Модель відповіді InventoryResponse.java модуля оптимізації:

```
package com.example.inventory;
public class InventoryResponse {
    private double optimalOrder;
    private double reorderPoint;

    public InventoryResponse(double optimalOrder, double reorderPoint) {
        this.optimalOrder = optimalOrder;
        this.reorderPoint = reorderPoint;
    }
```

```
    // Геттери
    public double getOptimalOrder() { return optimalOrder; }
    public double getReorderPoint() { return reorderPoint; }
}
```

Backend на Kotlin з використанням Ktor, створення проекту на Kotlin з Ktor, підготовка файлу build.gradle.kts:

```
dependencies {
    implementation("io.ktor:ktor-server-core:2.0.0")
    implementation("io.ktor:ktor-server-netty:2.0.0")
    implementation("io.ktor:ktor-serialization:2.0.0")
    implementation("ch.qos.logback:logback-classic:1.2.6")
}
```

Створення контролера в Ktor, основний файл Application.kt:

```
package com.example.inventory
import io.ktor.application.*
import io.ktor.features.ContentNegotiation
import io.ktor.http.*
import io.ktor.request.*
import io.ktor.response.*
import io.ktor.routing.*
import io.ktor.serialization.*
import io.ktor.server.engine.embeddedServer
import io.ktor.server.netty.Netty

fun main() {
    embeddedServer(Netty, port = 8080) {
        install(ContentNegotiation) {
            json()
```

```
        }
        routing {
            post("/api/inventory/calculateOrder") {
                val request = call.receive<InventoryRequest>()

                // Отримуємо дані та виконуємо розрахунок
                val averageSales = request.averageSales
                val leadTime = request.leadTime
                val minStock = request.minStock
                val orderCost = request.orderCost
                val holdingCost = request.holdingCost

                // Розрахунок оптимального замовлення (модель EOQ)
                val demandDuringLeadTime = averageSales * leadTime
                val reorderPoint = demandDuringLeadTime + minStock
                val optimalOrder = kotlin.math.sqrt((2 * orderCost * averageSales) /
holdingCost)

                // Відповідь з результатом
                call.respond(HttpStatusCode.OK, InventoryResponse(optimalOrder,
reorderPoint))
            }
        }
    }.start(wait = true)
}
```

Модель запиту InventoryRequest.kt:

```
package com.example.inventory
import kotlinx.serialization.Serializable
```

```
@Serializable
data class InventoryRequest(
    val averageSales: Double,
    val leadTime: Double,
    val minStock: Double,
    val orderCost: Double,
    val holdingCost: Double
)
```

Модель відповіді InventoryResponse.kt:

```
package com.example.inventory
import kotlinx.serialization.Serializable

@Serializable
data class InventoryResponse(
    val optimalOrder: Double,
    val reorderPoint: Double
)
```

```
public class AllocationResponse {
    private List<String> allocationResult;

    public AllocationResponse(List<String> allocationResult) {
        this.allocationResult = allocationResult;
    }

    public List<String> getAllocationResult() {
        return allocationResult;
    }
}
```

HTML-код для вхідних даних модуля алгоритмів розподілу ресурсів

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Алгоритми розподілу ресурсів</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        <h1>Модуль алгоритмів розподілу ресурсів</h1>
        <form id="resourceForm">
            <div class="form-group">
                <label for="resources">Кількість ресурсів (шт.):</label>
                <input type="number" id="resources" required>
            </div>

            <div class="form-group">
                <label for="tasks">Кількість завдань (шт.):</label>
                <input type="number" id="tasks" required>
            </div>

            <div class="form-group">
                <label for="resourceCapacity">Місткість одного ресурсу
(шт.):</label>
                <input type="number" id="resourceCapacity" required>
            </div>
```

```
            <div class="form-group">
                <label for="taskDemand">Попит на ресурси для кожного завдання (шт.):</label>
                <input type="text" id="taskDemand" placeholder="Введіть через кому, напр.: 10, 20, 15" required>
            </div>

            <button type="button" onclick="calculateAllocation()">Розрахувати розподіл ресурсів</button>
        </form>

        <div id="result">
            <h2>Результати розподілу:</h2>
            <p id="allocationResult"></p>
        </div>
    </div>

    <script src="script.js"></script>
</body>
</html>
```

CSS-код для стилізації веб-інтерфейсу модуля алгоритмів розподілу ресурсів

```
body {
    font-family: Arial, sans-serif;
    background-color: #f0f0f0;
    margin: 0;
    padding: 20px;
}
```

```
.container {
    max-width: 600px;
    margin: auto;
    background-color: white;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

h1, h2 {
    text-align: center;
}

.form-group {
    margin-bottom: 15px;
}

label {
    display: block;
    margin-bottom: 5px;
}

input {
    width: 100%;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 5px;
}

button {
```

```
    width: 100%;
    padding: 10px;
    background-color: #007bff;
    border: none;
    color: white;
    font-size: 16px;
    border-radius: 5px;
    cursor: pointer;
}

button:hover {
    background-color: #0056b3;
}

#result {
    margin-top: 20px;
    text-align: center;
}

#allocationResult {
    font-weight: bold;
}
```

JavaScript-код для обробки вхідних даних і розрахунку оптимального розподілу модуля алгоритмів розподілу ресурсів

```
function calculateAllocation() {
    // Отримуємо значення полів форми
    const resources = parseInt(document.getElementById('resources').value);
    const tasks = parseInt(document.getElementById('tasks').value);
```

```
        const resourceCapacity =
parseFloat(document.getElementById('resourceCapacity').value);
        const taskDemandInput = document.getElementById('taskDemand').value;

        // Перетворення рядка попиту на масив чисел
        const taskDemand = taskDemandInput.split(',').map(Number);

        // Перевірка валідності даних
        if (taskDemand.length !== tasks) {
            alert("Кількість завдань і попиту на ресурси повинні збігатися.");
            return;
        }

        // Ініціалізуємо результати розподілу
        let totalCapacity = resources * resourceCapacity;
        let allocationResult = [];

        for (let i = 0; i < tasks; i++) {
            if (totalCapacity >= taskDemand[i]) {
                allocationResult.push(`Завдання ${i + 1}: ${taskDemand[i]}
ресурсів`);
                totalCapacity -= taskDemand[i];
            } else {
                allocationResult.push(`Завдання ${i + 1}: ${totalCapacity} ресурсів
(недостатньо)`);
                totalCapacity = 0;
            }
        }

        // Виведення результатів
```

```
        document.getElementById('allocationResult').innerHTML =
allocationResult.join('<br>');
    }
```

Файл pom.xml модуля алгоритмів розподілу ресурсів

```
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
        </dependency>
    </dependencies>
```

Контролер ResourceAllocationController.java модуля алгоритмів розподілу ресурсів

```
    package com.example.resourceallocation;
    import org.springframework.web.bind.annotation.*;
    import java.util.ArrayList;
    import java.util.List;

    @RestController
    @RequestMapping("/api/resources")
    public class ResourceAllocationController {

        @PostMapping("/allocate")
        public AllocationResponse allocateResources(@RequestBody
AllocationRequest request) {
```

```
            int resources = request.getResources();
            int resourceCapacity = request.getResourceCapacity();
            List<Integer> taskDemand = request.getTaskDemand();

            int totalCapacity = resources * resourceCapacity;
            List<String> allocationResult = new ArrayList<>();

            for (int i = 0; i < taskDemand.size(); i++) {
                int demand = taskDemand.get(i);
                if (totalCapacity >= demand) {
                    allocationResult.add("Завдання " + (i + 1) + ": " + demand + "
ресурсів");
                    totalCapacity -= demand;
                } else {
                    allocationResult.add("Завдання " + (i + 1) + ": " + totalCapacity + "
ресурсів (недостатньо)");
                    totalCapacity = 0;
                }
            }

            return new AllocationResponse(allocationResult);
        }
    }
```

Модель запиту AllocationRequest.java модуля алгоритмів розподілу ресурсів

```
package com.example.resourceallocation;
import java.util.List;

public class AllocationRequest {
```

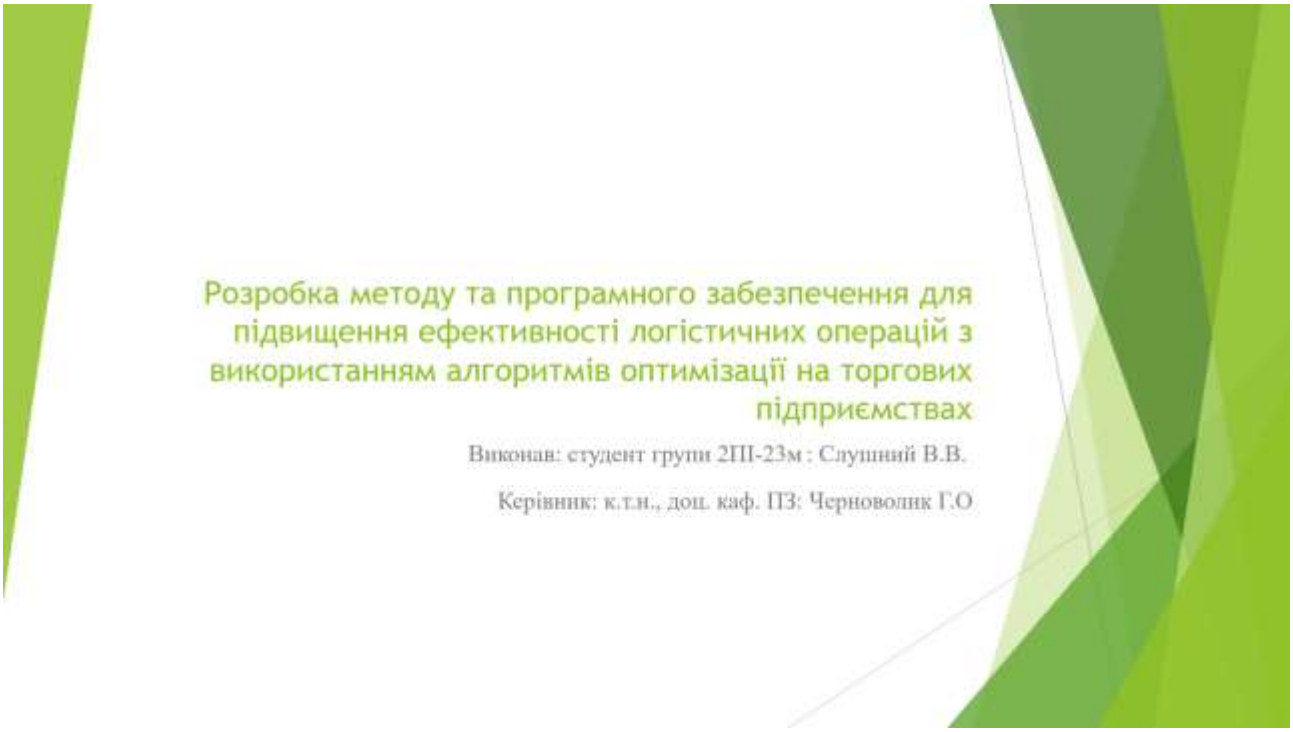
```
    private int resources;
    private int resourceCapacity;
    private List<Integer> taskDemand;

    public int getResources() {
        return resources;
    }

    public void setResources(int resources) {
        this.resources = resources;
    }

    public int getResourceCapacity() {
        return resourceCapacity;
    }

    public void setResourceCapacity(int resourceCapacity) {
        this.resourceCapacity = resourceCapacity;
    }

    public List<Integer> getTaskDemand() {
        return taskDemand;
    }

    public void setTaskDemand(List<Integer> taskDemand) {
        this.taskDemand = taskDemand;
    }
}
```

Модель відповіді AllocationResponse.java модуля алгоритмів розподілу ресурсів

```
package com.example.resourceallocation;
import java.util.List;

public class AllocationResponse {
    private List<String> allocationResult;

    public AllocationResponse(List<String> allocationResult) {
        this.allocationResult = allocationResult;
    }

    public List<String> getAllocationResult() {
        return allocationResult;
    }
}
```

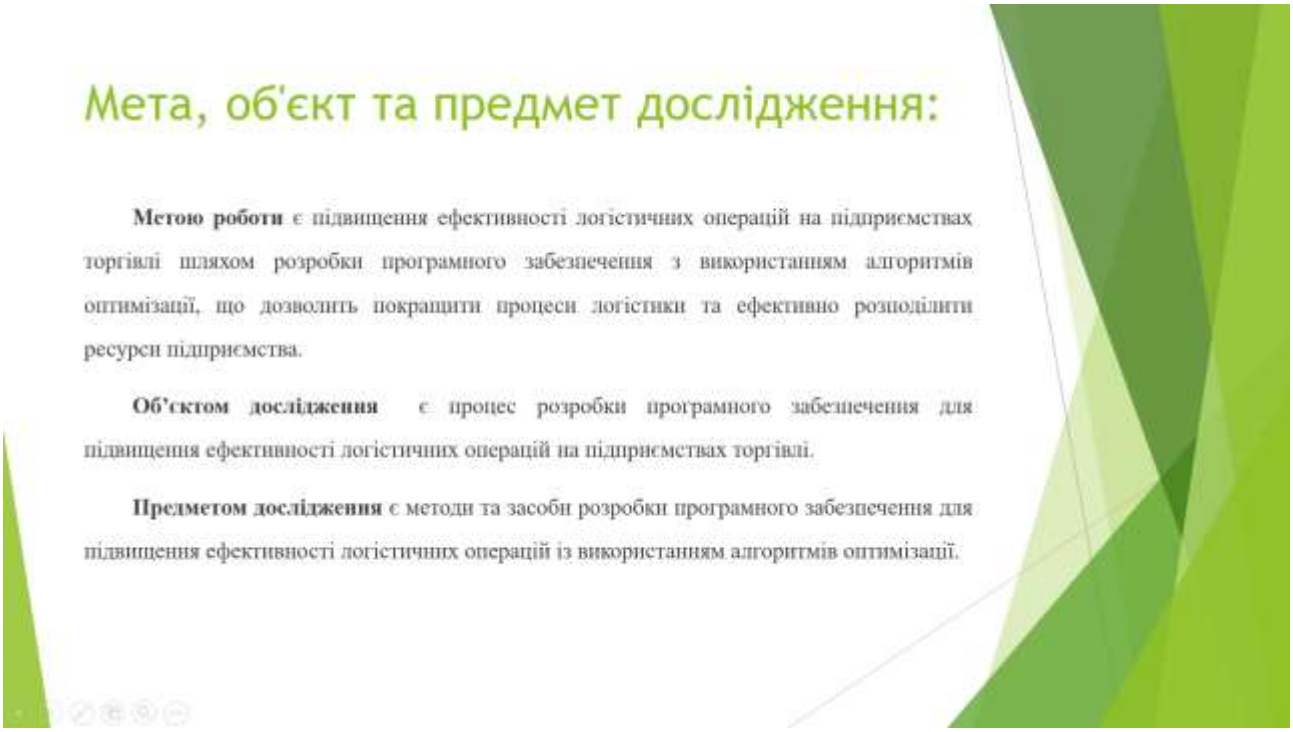
Додаток Г. Ілюстративна частина

ІЛЮСТРАТИВНА ЧАСТИНА
РОЗРОБКА МЕТОДУ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ЛОГІСТИЧНИХ ОПЕРАЦІЙ З ВИКОРИСТАННЯМ АЛГОРИТМІВ ОПТИМІЗАЦІЇ НА ТОРГОВИХ ПІДПРИЄМСТВАХ



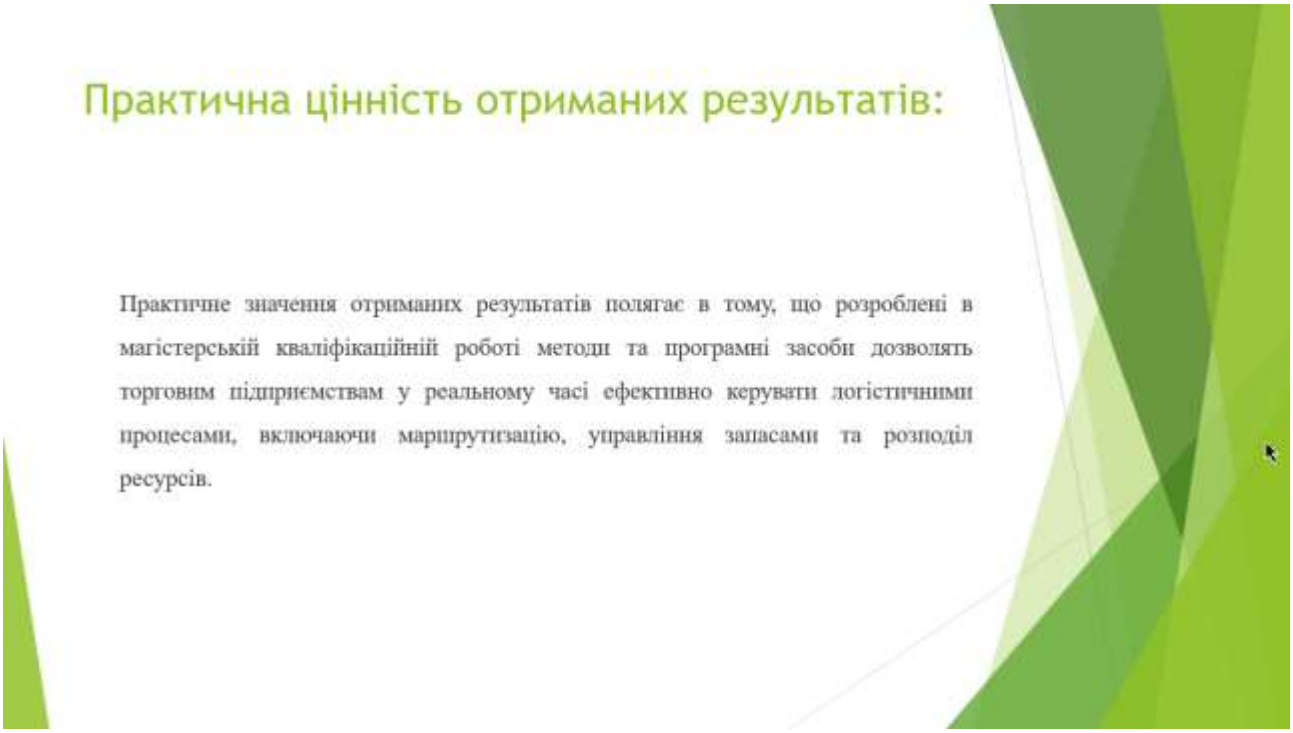
Додаток Г.1 – Титульний слайд

Додато Г.2 – Актуальність теми

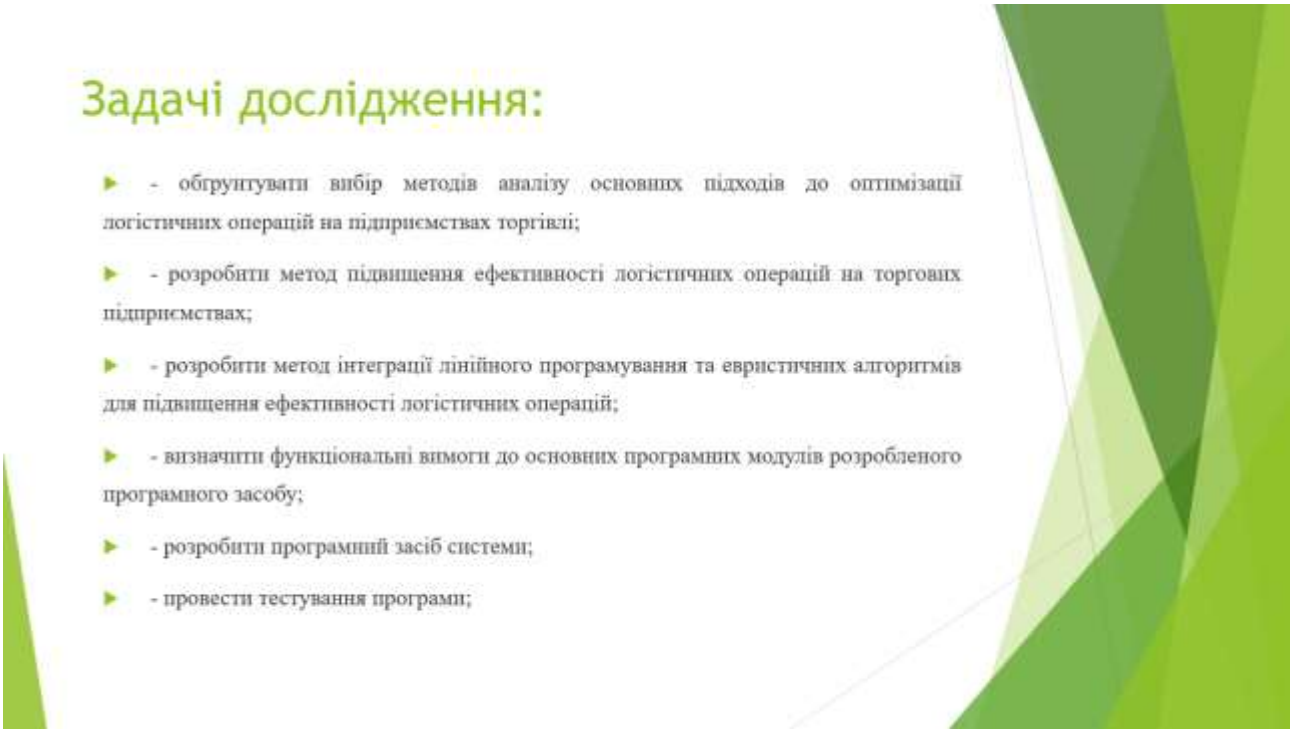


Додаток Г.3 – Мета, об’єкт та предмет дослідження

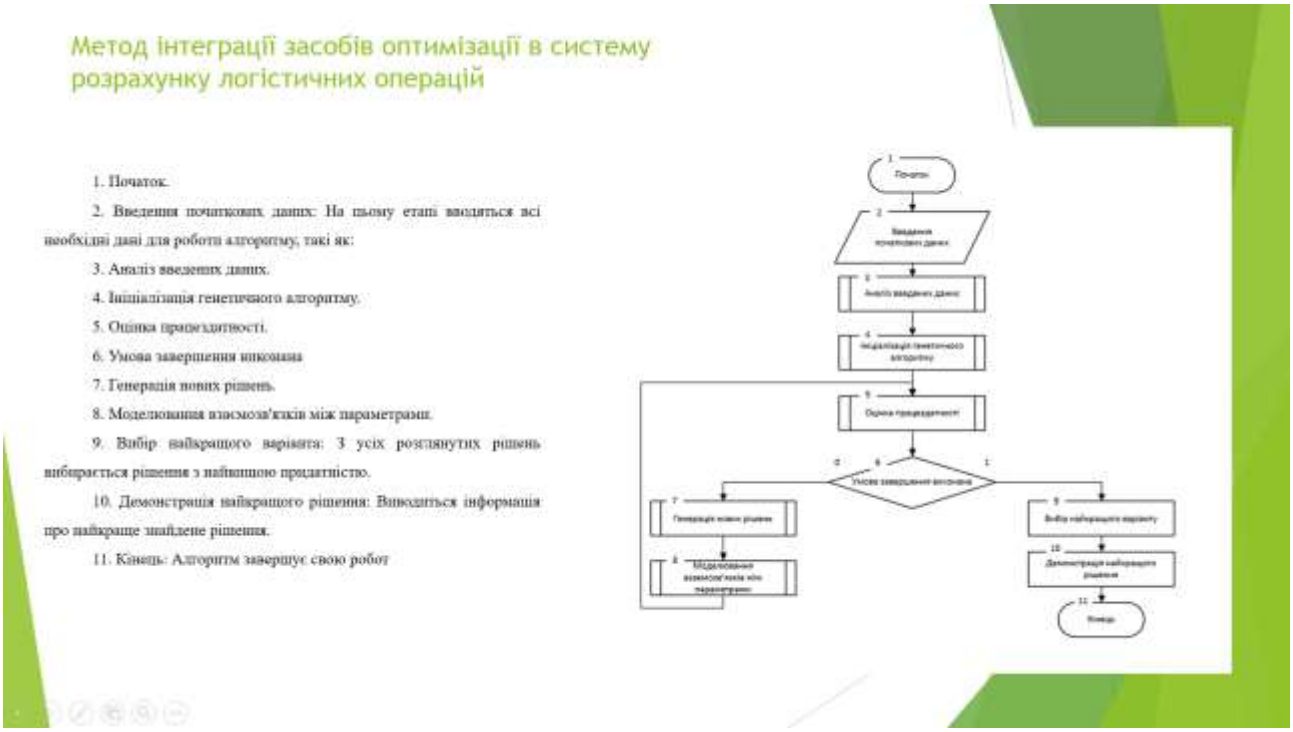
Додаток Г.4 – Наукова новизна одержаних результатів



Додаток Г.5 – Практична цінність отриманих результатів



Додаток Г.6 – Задачі дослідження

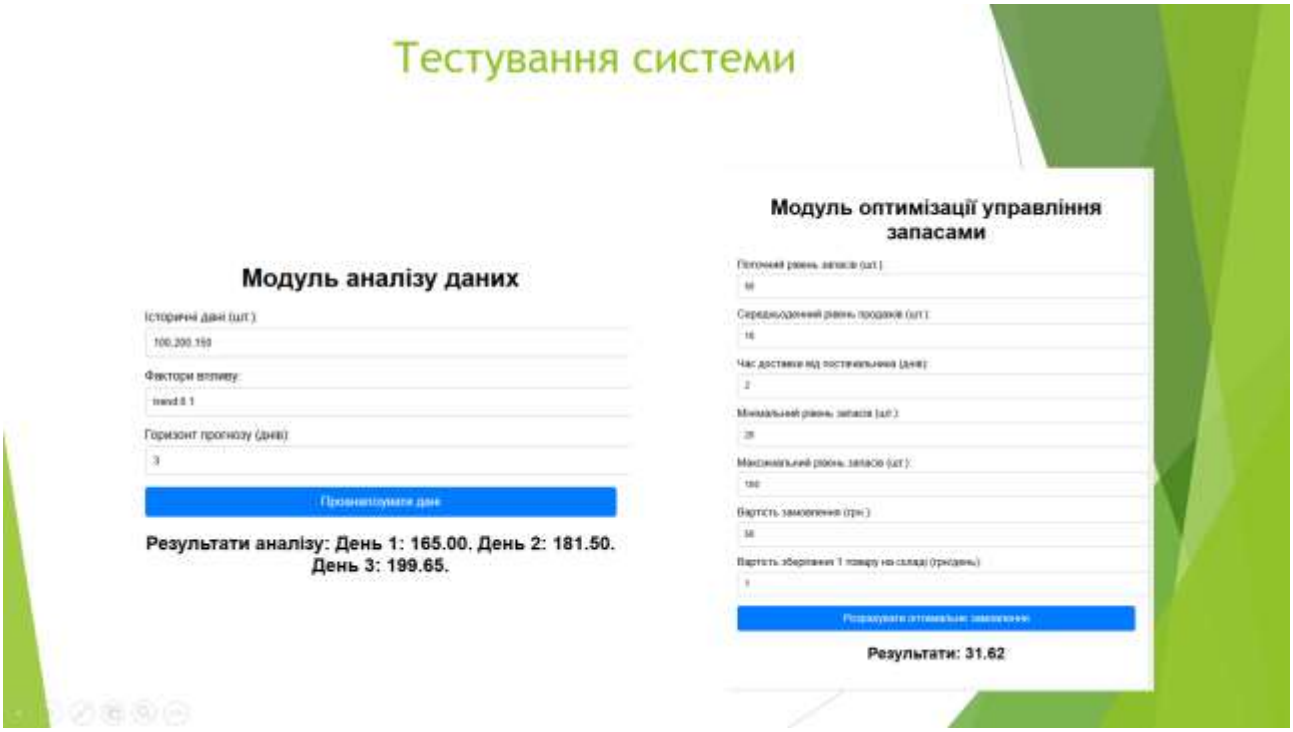


Додаток Г.7 – Метод інтеграції засобів оптимізації в систему розрахунку логістичних операцій

Додаток Г.8 – Метод підвищення ефективності логістичних операцій на основі генетичних алгоритмів та лінійного програмування

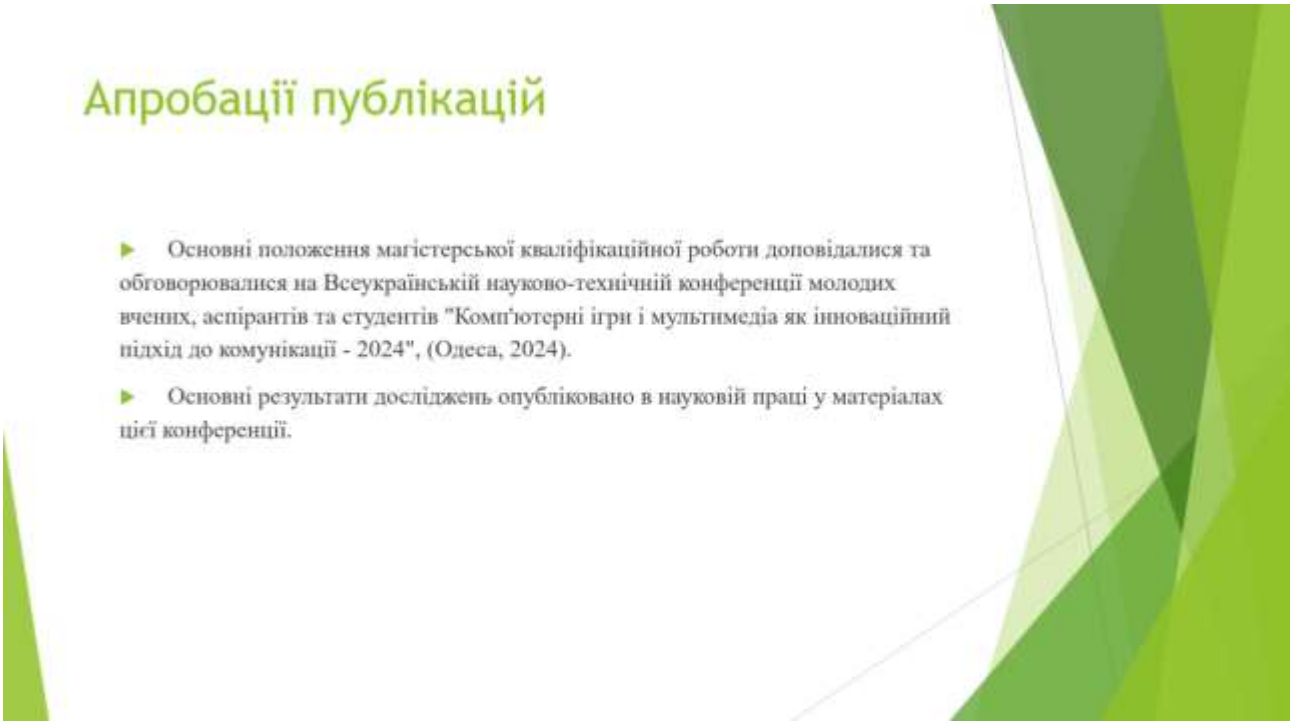
Додаток Г.9 – Алгоритм роботи модуля оптимізації

Додаток Г.10 – Діаграма послідовності взаємодії користувача із застосунком



Додаток Г.11 – Тестування модуля аналізу даних та модуля оптимізації

Додаток Г.12 – Висновки



Додаток Г.13 – Апробація і публікації