Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра математики

Магістерська робота

освітній супінь – магістр

на тему: «**СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ В ЗАДАЧАХ КЕРУВАННЯ ВИПАДКОВИМИ ПОТОКАМИ В МЕРЕЖАХ**»

Виконав: студентка 2-го року навчання,
спеціальності 124 Системний аналіз

Керімова Рафіга Мугамат кизи

Керівник: Чорней Р. К.,
кандидат фізико-математичних наук,
доцент

Рецензент_____

Магістерська робота захищена
з оцінкою «_____»

Секретар ЕК _____
«____» _____20__ р.

Київ – 2024

Зміст

Вступ 3

Розділ 1. Теоретичні відомості 5

1.1. Модель EOQ (економічний розмір замовлення) 7

або модель Харріса, чи модель Уілсона без дефіциту

1.2. Модель оптимального розміру замовлення з дефіцитом 9

Розділ 2. Аналіз проблематики 14

2.1. Основні проблеми управління запасами 14

2.2. Вплив невірного управління запасами на бізнес 15

2.3. Сучасні підходи до управління запасами 15

Розділ 3. Розробка системи керування запасами 17

Розділ 4. Програмна реалізація 19

Розділ 5. Реалізація обраних моделей керування запасами 31

Розділ 6. Використання методу симуляцій для аналізу роботи моделей 38

Висновок 42

Список літератури 44

Інтернет джерела 45

Додатки 46

ВСТУП

В сучасних умовах, коли випадковість та нестабільність стають невід'ємною частиною бізнес-процесів, розробка ефективних стратегій керування стає критично важливою для успішної діяльності підприємств. Оптимальне управління запасами забезпечує не тільки стабільність постачань, але й мінімізацію витрат і підвищення загальної ефективності бізнесу.
Актуальність даної теми посилюється потребою спростити задачу ведення бізнесу та полегшити її для малого бізнесу, особливо в умовах економічної нестабільності та війни.

Метою даної магістерської роботи є аналіз та дослідження товарних запасів за допомогою моделей керування запасами а також розробка та імплементаця системи керування запасами для вибраного магазину, яка дозволить оптимізувати процеси управління запасами, знизити витрати та підвищити рівень обслуговування клієнтів.
Об'єктом дослідження є система керування запасами в невеликому магазині непродовольчих товарів, агро-хімії та товарів для садівництва і городництва в селі.
Предметом дослідження є моделі керування запасами з використанням сучасних програмних рішень на базі мови програмування Python.

Для досягнення поставленої мети були сформульовані наступні завдання дослідження:

1. Аналіз поточного стану управління запасами в обраному магазині.
2. Вивчення теоретичних основ оптимального управління запасами.
3. Розробка програмного забезпечення для аналізу запасів на основі мови програмування Python.
4. Тестування розробленої системи на практиці та оцінка її ефективності.

5. Використання моделей управління запасами на даних, отриманих з програмного застосунку.
6. Використання методу симуляцій для аналізу роботи моделей.

При підготовці до роботи було проведено аналіз наукової літератури, що стосується теми управління запасами, створення продуктів для обліку та оптимізації. Дослідження базується на методологічних принципах управління запасами та використанні сучасних інструментів програмування для реалізації розроблених алгоритмів.

У контексті сучасної економічної ситуації в Україні, де малі підприємства є основою економіки, важливо розробляти інструменти та методики, які можуть допомогти в підтримці та розвитку малого бізнесу. Оптимальне управління запасами може стати вирішальним фактором для багатьох підприємств, дозволяючи їм ефективно конкурувати на ринку та забезпечувати стабільність у важкі часи.[21]

В результаті проведеного дослідження, можливо адаптувати та використати отримані нові знання та методики в управлінні запасами в різних сферах економіки. Практична значущість полягає в тому, що розроблена система може бути використана реальними підприємствами для покращення їхньої ефективності та конкурентоспроможності на ринку.[2]

Дана магістерська робота має на меті внести вклад у вивчення систем підтримки прийняття рішень в задачах керування випадковими мережами, з урахуванням специфіки малого бізнесу в Україні в умовах економічної нестабільності та військового стану. Реалізація цього дослідження може сприяти розвитку ефективних інструментів та методик управління запасами, які будуть корисні для підтримки та розвитку малого бізнесу в умовах сучасних викликів.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ВІДОМОСТІ

В даному розділі буде наведено визначення поняття «запас», інформацію про причини необхідності утримання запасів та відображено розглянуті в роботі моделі керування запасами.

Запаси - активи, які:

- утримуються для подальшого продажу (розподілу, передачі) за умов звичайної господарської діяльності;

- перебувають у процесі виробництва з метою подальшого продажу продукту виробництва;

- утримуються для споживання під час виробництва продукції, виконання робіт та надання послуг, а також управління підприємством. (Про затвердження Національного положення (стандарту) бухгалтерського обліку). [22]

З'ясуємо, для чого підприємство має на меті зосереджувати запаси та зберігати їх:

- забезпечення якісного і продуктивного обслуговування клієнтів;[17]

- страхування від ризиків при постачанні;

- мінімізація витрат при зростi цін на товари;

- отримання дисконту при об'ємному замовленні;

- зменшення логістичних витрат.

Тому, незважаючи на витрати, пов'язані із утриманням товарних запасів, через ряд перелічених чинників, підприємцям вигідніше створювати певний запас товарів.

Наразі нам відомо декілька концепцій управління запасами:

1) Максимізація запасів – найбільш застаріла концепція, в основу якої покладено необґрунтоване накопичення максимального рівня товарних запасів. Дана концепція не бере до уваги надлишок запасів, роблячи акцент на задоволенні попиту та утворенні надлишку без аналізу продажів.
2) Оптимізація запасів – модель упорядкування рівня запасів з метою мінімізації витрат на їх зберігання. Наразі дана концепція є найуживанішою.
3) Мінімізація запасів – модель, яка завдяки зваженому підходу до управління запасами зводить їх кількість до мінімуму. [3]

В політиці керування торгівельними об'єктами запаси допомагають відобразити попит на певні товари з часом та виділити популярні товари. За допомогою накопичення запасів можливо уникнути дефіциту товару між регулярними постачаннями та, проаналізувавши попит на товари, визначити необхідну частоту замовлень на окремі групи товарів.

Товарні запаси є головним ресурсом об'єктів торгівлі. Управління товарними запасами є важливою складовою для успішної діяльності торговельних компаній. Від якості управління цим ресурсом залежать показники прибутковості та ліквідності підприємства. У багатьох магазинах є чітко встановлені правила обслуговування клієнтів та викладання товарів, але відсутня загальна політика щодо управління товарними запасами. Це призводить до ситуації, коли деякі магазини мають недостатню кількість популярних товарів, тоді як інші мають їх надлишок, що зберігається на складі.

На прикладі розглянутого об'єкту (магазину в селі) а також конкуруючих підприємст можна наочно помітити недоліки в управлінні. Магазин може мати перевагу серед конкурентів, реалізуючи спеціалізовані товари, закриваючи тим самим нішу необхідних товарів, проте за неправильної організації виникає надлишок непопулярних товарів, чи таких, які втратили популярність з настанням іншого сезону. Також, навпаки, в період пікового ажіотажу, виникає проблема в

нестачі актуальних позицій. Однак система управління запасами може допомогти виявити ці проблеми та ефективно планувати закупівлі товарів на основі попиту, забезпечуючи їх швидку реалізацію та покращення ефективності управління запасами.

Управління товарними запасами включає ряд заходів, спрямованих на максимізацію продажів при мінімізації витрат, пов'язаних з утриманням запасів. Для покращення логістичних процесів підприємства повинні раціонально управляти своїми товарними запасами, зокрема, удосконалювати систему складського зберігання товарів для зменшення втрат. Управління запасами включає в себе регулярність поповнення запасів та правильно розраховані обсяги закупівель . Важливо збалансувати широкий асортимент товарів для задоволення потреб покупців з оптимізацією витрат на зберігання запасів і забезпеченням швидкої обробки замовлень.

Для впорядкування системи запасів використовують різноманітні моделі управління. Це можуть бути однокомпонентні (детерміновані) моделі – моделі, в яких розглядається певний вид товару або багатокомпонентні (стохастичні).[20]

В роботі розглянуто дві основні моделі керування запасами:

1. **Модель EOQ (економічний розмір замовлення), або модель Харріса, чи модель Уілсона без дефіциту.**

   Подвійне авторство у назві моделі зумовлене тим, що дана модель оптимального розміру завмовлення була вперше запропонована Фордом Уітманом Харрісом у 1913 році, як стаття в журналі адресована менеджерам виробничої сфери. Проте, широкого розголосу дана модель набула саме публікації у статті Уілсона в 1934 році. У даній статті Уілсон розглянув модель Харріса та підкреслив важливість балансу між витратами пов'язаними із розміщенням замовлення та витратами на його зберігання для отримання оптимального розміру замовлення. [19]

Дану модель ще називають найпростішою моделлю оптимального розміру замовлення, вона є детермінованою. Припускаємо, що:

- темп попиту на товар є відомим та сталим;
- отримання замовлення відбувається миттєво;
- закупівельна ціна не змінюється залежно від розміру замовлення;
- дефіцит виключений;
- витрати на здійснення замовлення не залежать від його розміру.

Формули для розрахунку EOQ:

Вхідні дані включають : D- попит на продукцію за час планування, S – витрати на оформлення одного замовлення (вартість товару + вартість транспортування та доставки), H- вартість зберігання запасів, T – період планування. [20]

Вихідні дані: Q – величина замовлення, , F – загальні витрати на керування запасами.

Витрати на оформлення замовлення за період T:

$$S(Q) = \frac{D}{Q} * S \qquad (1)$$

Середні витрати на утримання запасів між замовленнями:

$$H(Q) = \frac{Q}{2} * H \qquad (2)$$

Маємо, сукупні витрати :

$$F(Q) = \frac{D}{Q} * S + \frac{Q}{2} * H \rightarrow min \qquad (3)$$

Диференціюємо функцію сукупних витрат по Q та прирівняємо до 0. Отримаємо оптимальний розмір замовлення:

$$0 = -\frac{D}{Q^2} * S + \frac{1}{2} * H \quad (4)$$

$$Q^2 = -\frac{2*D*S}{H} \quad (5)$$

$$Q = \sqrt{\frac{2DS}{H}} \quad (6)$$

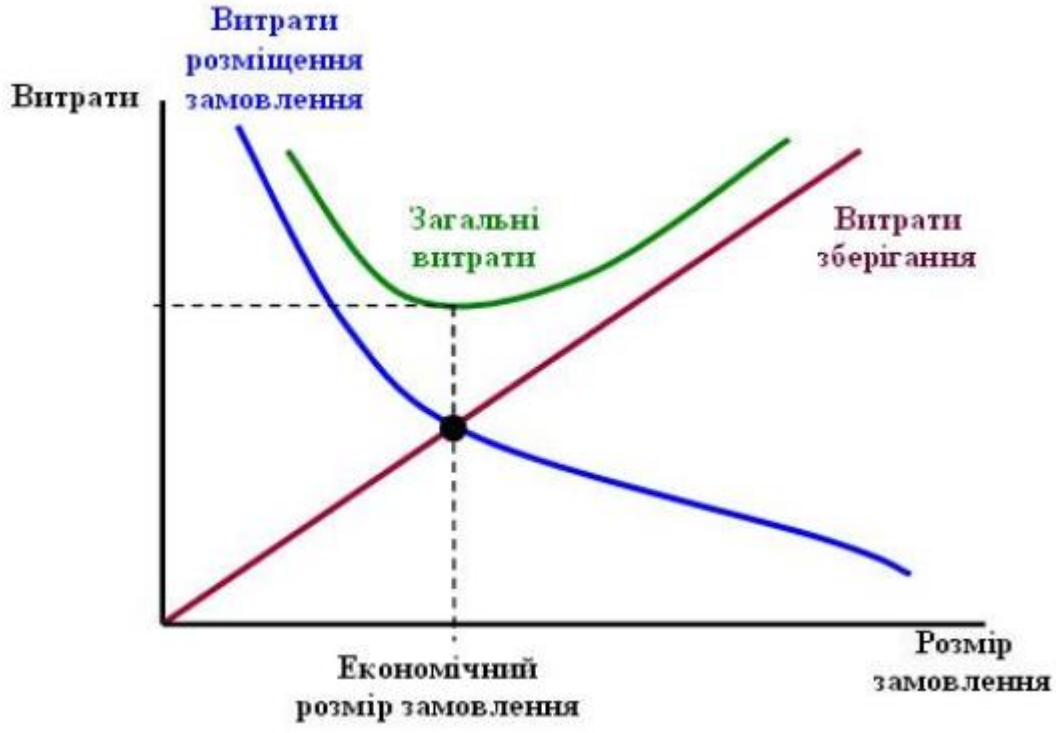
Оптимальне число замовлень за визначений період:

$$N = \frac{D}{Q} \quad (7)$$

Оптимальний час між замовленнями:

$$t = \frac{Q}{d} = \frac{T}{N} \quad (8)$$

Отже, оптимальна партія поставки є такою, щоб значення функції сукупних витрат було мінімальним. Графічна зображення даної моделі буде наступним [20]:



Оптимальним рішенням є розмір замовлення, при якому сума витрат на зберігання та витрат на замовлення мінімальна протягом розглянутого періоду. [20]

2. **Модель оптимального розміру замовлення з дефіцитом**

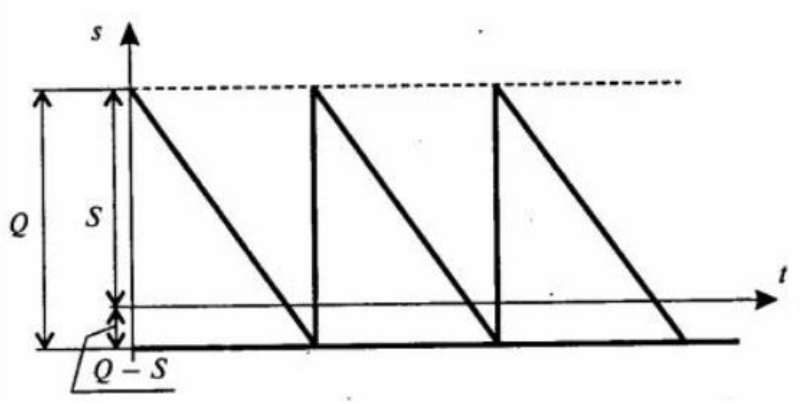
   Дана модель передбачає, що темп попиту на товар відомий (відстежується у списку очікування), час виконання замовлень є відомим (визначимо його як сталу величину). Також припустимо, що закупівельна ціна не залежить від розміру замовлення (не надається знижка в залежності від кількості замовлених товарів). [20]

Вхідні дані: D- попит на продукцію за час планування, S – витрати на оформлення одного замовлення (вартість товару + вартість транспортування та доставки), H- вартість зберігання запасів, Y - витрати дефіциту.

Вихідними даними є: Q – величина замовлення, T- період між замовленнями (кількість днів), H_0 – точка замовлення (кількість товару на складі для закриття попиту, при якій необхідно здійснити замовлення) та F - загальні витрати.

Оптимальне рішення в даній моделі являє собою оптимальний розмір замовлення такий, щоб мінімізувати сукупні витрати, які містять окрім витрат на зберігання та витрат на здійснення замовлення ще додатково витрати на дефіцит.[20]

Динаміку зміни кількості продуктів на складі можна побачити на графіку [20]:



Формули для розрахунку:

Маємо, витрати на оформлення замовлення за період T:

$$S(Q) = \frac{D}{Q} * S \qquad (7)$$

Середні витрати на утримання запасів між замовленнями:

$$H(Q) = \frac{S^2}{2Q} * H \qquad (7)$$

Витрати за рахунок дефіциту:

$$Y(Q) = \frac{(Q-S)^2}{2Q} * Y \qquad (8)$$

Маємо, сукупні витрати :

$$F(Q) = \frac{D}{Q} * S + \frac{S^2}{2Q} * H + \frac{(Q-S)^2}{2Q} * Y \to min \qquad (9)$$

Отримаємо оптимальний розмір замовлення:

$$Q = \sqrt{\frac{2DS}{H} * \frac{Y+H}{Y}} \qquad (10)$$

Оптимальний максимально можливий розмір замовлення:

$$s = \sqrt{\frac{2DS}{H} * \frac{Y}{Y+H}} \qquad (11)$$

Точка відновлення запасів:

$$H_0 = D * T \qquad (12)$$

Та оптимальний максимально можливий дефіцит: [20]

$$Y_{\text{макс}} = Q - S \qquad (13)$$

Для ефективного використання моделі важливо мати достатні дані про попит на товар, витрати на оформлення замовлення, вартість утримання запасів та час між замовленнями. Аналізуючи результати моделей, ми зможемо приймати обґрунтовані рішення щодо управління запасами та оптимізації витрат відповідно до конкретних умов та потреб розглянутого підприємства а також, порівнявши результати обох моделей, зможемо визначити доцільність використання кожної з них в даному об'єкті та можливі шляхи модифікації вхідних даних для зміни результатів.

Модель оптимального розміру замовлення з дефіцитом має як переваги, так і недоліки в порівнянні з іншими методами управління запасами.

Переваги:

1. **Облік дефіциту:** Ця модель дозволяє враховувати витрати на дефіцит, що є важливим для підприємств, де недопустимий дефіцит товарів може значно вплинути на репутацію та фінансові показники. Вона дозволяє прорахувати витрати, пов'язані з нестачею запасів, та порівняти їх з витратами на зберігання даних запасів, в разі відсутності попиту.
2. **Точність:** Використання математичних формул дозволяє точно розрахувати оптимальні параметри замовлення та мінімізувати загальні витрати.
3. **Гнучкість:** Модель може бути адаптована під різні умови попиту та витрат, що робить її універсальною для різних типів підприємств.

Недоліки:

1. **Вимоги до даних:** Модель вимагає точних даних про попит, витрати на оформлення замовлення, утримання запасів та дефіцит, що може бути складно забезпечити на практиці. Дані про дефіцит мають бути прораховані з високою точністю, недостатньо враховувати кількість товару, чітко мають бути враховані суми вартості товару, на який виник дефіцит.

2. **Складність розрахунків:** Для малого бізнесу або підприємств без належної технічної підтримки складність розрахунків може стати бар'єром для використання моделі.
3. **Відсутність знижок:** Модель не враховує можливі знижки при закупівлі великих партій товарів, що може бути важливим фактором для деяких підприємств.

Порівнюючи з моделлю EOQ без дефіциту, модель з дефіцитом є більш комплексною та точнішою у ситуаціях, де можливий дефіцит має значний вплив. Однак, для підприємств, де дефіцит не є критичним або де існують знижки на великі замовлення, модель EOQ без дефіциту може бути більш доцільною.

РОЗДІЛ 2. АНАЛІЗ ПРОБЛЕМАТИКИ

Сучасний ринок характеризується високою невизначеністю, змінністю попиту та постійною динамікою конкурентного середовища. В таких умовах питання ефективного управління запасами стає критично важливим для забезпечення стабільності та конкурентоспроможності бізнесу.

На прикладі ситуації в невеликому селі Чернігівської області я намагаюсь в своїй роботі провести аналіз ринку та, дослідивши потреби споживачів, використати дані аналізу попиту для використання моделей управління запасами та коректної розробки додатку, який допоможе малому бізнесу.

Буде розглянуто основний об'єкт – непродовольчий магазин, який в асортименті також має товари агро-хімії, садове приладдя, посадковий матеріал рослин тощо. З іншого боку – буде проведено порівняння цінової політики, розглянуто попит на асортимент товарів даного магазину та враховано закупівельні можливості.

2.1. Основні проблеми управління запасами

Основні проблеми, з якими стикаються підприємства при управлінні запасами, включають:

- **Недостача або перевищення запасів**: Неправильно розрахований рівень запасів може призвести до втрат від нереалізованих товарів або втрат покупців через нестачу товарів.

  На прикладі розглянутого об'єкту – було втрачено близько 20% покупців через несвоєчасний аналіз потреб споживачів та не врахування прогнозів погодних умов (важливо для реалізації посадкового матеріалу).

  І, навпаки, перенасичення нерентабельних товарів, через надмірно лояльне ставлення до постачальників та помилковий аналіз продукції.

- **Високі витрати на утримання запасів**: Підтримка великих запасів може призвести до високих витрат на зберігання, обслуговування та інші

управлінські витрати.

Прикладом випадку даної проблеми на об'єкті стало надмірне замовлення агро-продукції наприкінці сезону, коли відчутно знизився попит на товари даного сегменту. Як результат – втрата частини посадкового матеріалу через неможливість підтримувати повною мірою належні умови зберігання у зимній період.

- **Сезонність та непередбачуваність попиту**: Сезонні варіації та непередбачуваний попит можуть ускладнити прогнозування та планування запасів. Необхідне врахування багатьох факторів (потреби споживачів, аналіз асортименту конкурентів, погодні умови, ситуації пов'язані з воєнними діями в регіоні чи околицях тощо).

2.2. Вплив невірного управління запасами на бізнес

Неправильне управління запасами може мати серйозний негативний вплив на бізнес. Серед основних наслідків можна виділити:

- **Втрати від нереалізованих товарів**: Неоптимальне управління запасами може призвести до великих втрат від товарів, які не були реалізовані вчасно.
- **Погіршення іміджу підприємства**: Нестача товарів може негативно вплинути на репутацію підприємства перед клієнтами. Дефіцитом може скористатись конкурентна організація та перехопити частину потенційних клієнтів на хвилі попиту.
- **Високі операційні витрати**: Неправильне управління запасами призводить до збільшення витрат на зберігання, логістику та обробку замовлень.

2.3. Сучасні підходи до управління запасами

Сучасні підходи до управління запасами орієнтовані на використання сучасних технологій та аналітичних інструментів для прогнозування попиту, оптимізації запасів та підвищення ефективності управління. До основних підходів належать:

- **Just-In-Time (JIT)**: Метод, який передбачає доставку матеріалів на виробництво саме в момент їх необхідності, мінімізуючи таким чином запаси на складі.
- **Vendor Managed Inventory (VMI)**: Метод, при якому постачальник відповідає за управління запасами підприємства, що дозволяє оптимізувати рівень запасів та підвищувати ефективність логістичних процесів.
- **Data Analytics та Machine Learning**: Використання аналітичних інструментів та машинного навчання для прогнозування попиту, виявлення патернів та оптимізації стратегій управління запасами.

Проблематика управління запасами є важливою та актуальною для сучасного бізнесу. Ефективне управління запасами вимагає комплексного підходу, включаючи використання сучасних технологій, аналітичних методів та оптимальних стратегій управління, що спрямовані на максимізацію прибутку та зниження витрат.

РОЗДІЛ 3. РОЗРОБКА СИСТЕМИ КЕРУВАННЯ ЗАПАСАМИ

З метою збору і структуризації даних про наявний товар, прорахунку попиту на товари за фіксований час та отримання наочних даних для аналізу моделей керування, було прийнято рішення розробити програмний застосунок – «Система керування запасами».

Заплановано, що даний застосунок буде перенесений та доступний на мобільних додатках, щоб розширити коло користувачів та надати їм можливість для покращення системи розподілу та планування товарів.

Визначення вимог та функціональних можливостей системи

Під час розробки системи було враховано ряд важливих вимог та функціональних можливостей.

Вимоги до системи включають:

- Автоматизація процесу моніторингу та контролю за запасами.
- Можливість прогнозування попиту на товари.
- Інтеграція з іншими системами управління, такими як фінансова звітність, логістика та облік.
- Гнучкість та масштабованість для адаптації до змінних ринкових умов.

Заплановані функціональні можливості системи:

- Аналіз та візуалізація даних про запаси в режимі реального часу.
- Генерація звітів та аналітичних матеріалів для прийняття рішень.
- Автоматичне поповнення запасів на основі попиту та історичних даних.
- Моніторинг та оповіщення про критичні рівні запасів.

В даній роботі було вирішено розробити програму за допомогою мови програмування Python з графічним інтерфейсом для користувача.

Програма містить наступний функціонал:

- надає можливість приймати за базу даних – список наявних в магазині товарів, якщо такий існує;
- додавати в список нові товари;
- аналізувати запити на замовлення нових товарів від користувачів;
- аналізувати попит на товари та прогнозувати обсяг наступних замовлень на основі аналізу;
- сповіщати про необхідність актуалізації даних (зміну кількості товарів на складі при повторній закупівлі тощо).

РОЗДІЛ 4. ПРОГРАМНА РЕАЛІЗАЦІЯ

Програма розроблена з урахуванням простоти та ефективності управління товарними запасами. Вона надає користувачеві інтерфейс для додавання нових товарів, оновлення існуючих кількостей товарів та видалення товарів з бази даних. Основні принципи роботи полягають у:

1. **Модульності:** Кожна основна функція програми (додавання товару, оновлення кількості, видалення товару) реалізована як окрема функція, що дозволяє легко масштабувати та модифікувати програму в майбутньому.
2. **Гнучкості:** Програма надає користувачеві можливість легко додавати, оновлювати та видаляти товари без необхідності втручання в код програми.

Функціональні можливості:

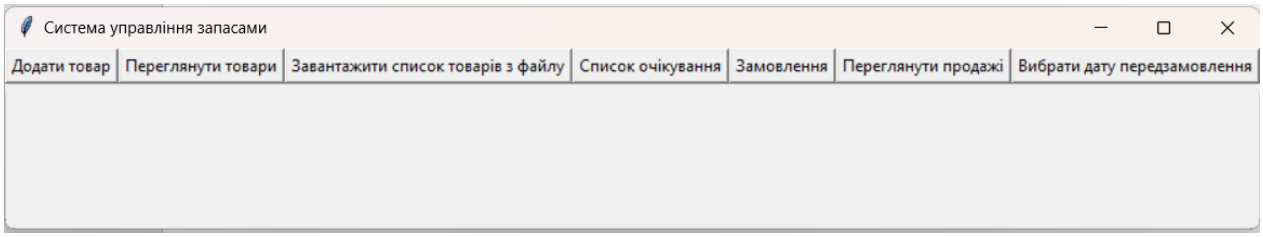
1. **Додавання товарів:** Користувач може ввести назву, кількість та ціну товару у відповідні поля. Після натискання кнопки "Додати товар", інформація про новий товар зберігається у базі даних.
2. **Перегляд товарів:** Програма автоматично завантажує список всіх доступних товарів із бази даних та відображає їх у відповідному графічному інтерфейсі. Для кожного товару показується його назва, кількість та ціна.
3. **Оновлення кількості товару:** Користувач може збільшити кількість певного товару, вказавши додаткову кількість, яку необхідно додати до існуючого запасу. Програма автоматично оновлює дані в базі даних.
4. **Видалення товару:** Користувач може видалити непотрібний товар з бази даних, використовуючи відповідну кнопку поблизу інформації про товар.
5. **Формування списку очікування:** Користувач може додавати в список очікування товари, на які надходять запити від клієнтів.

6. **Формування списку замовлень:** Користувач може додавати в список товари, які плануються до майбутніх закупівель.
7. **Автоматичне додавання товарів до списку замовлень:** Користувачу доступний функціонал, за допомогою якого товар зі списку очікування, перейшовши визначену межу запитів, буде запропонований до переміщення у список замовлень.
8. **Завантаження товарів з файлу:** Користувач може додати в базу даних товари з нових закупівель, завантаживши файл-excel з переліком товарів.

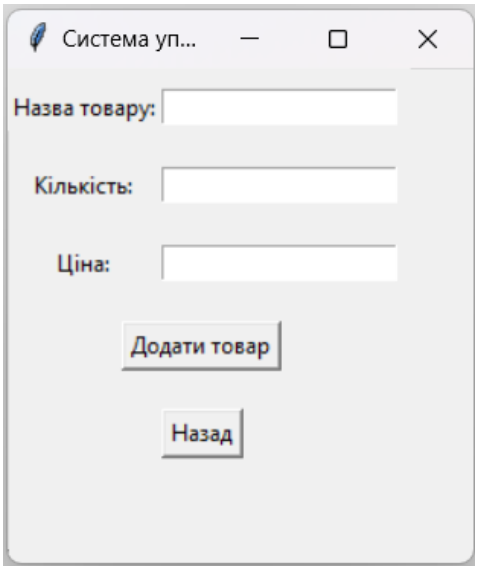
Вигляд застосунку для користувача:

1. При запуску програми перед користувачем з'являється загальне меню вибору (буде частково розширене найближчим часом) із можливістю вибіру наступної дії.

На ньому наявні розділи «Додати товар», «Переглянути товари», «Завантажити список товарів з файлу», «Список очікування», «Замовлення», «Переглянути продажі» та «Вибрати дату передзамовлення».

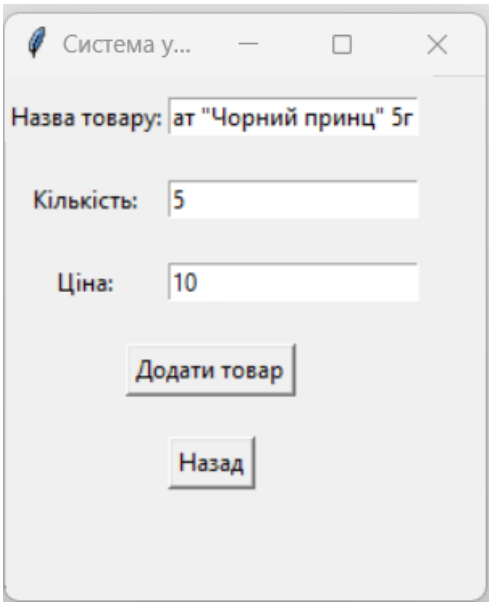


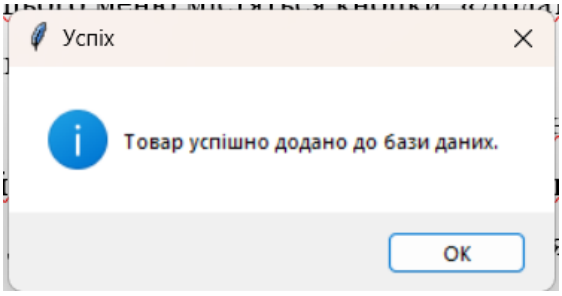
2. При виборі кнопки «Додати товар», користувачу стає доступним під-меню, де верхня частина робочого поля – рядки для введення інформації про товар, який необхідно додати в базу.



- В нижній частині цього меню містяться кнопки: «Додати товар» та кнопка «Назад».

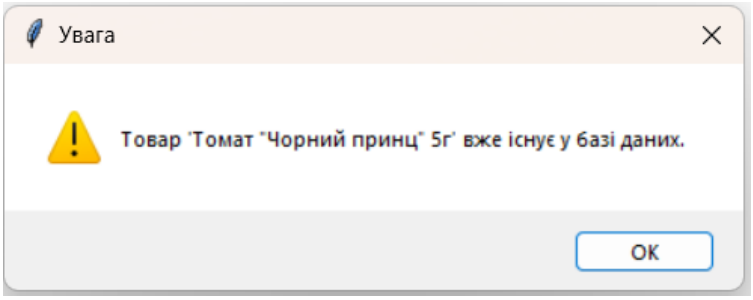
3. При правильному введенні даних у верхній частині та натисканні «Додати товар» користувачу надійде сповіщення про успішне включення нового товару до бази даних, товар буде додано та робочі поля очистяться для можливості вписання нового товару.





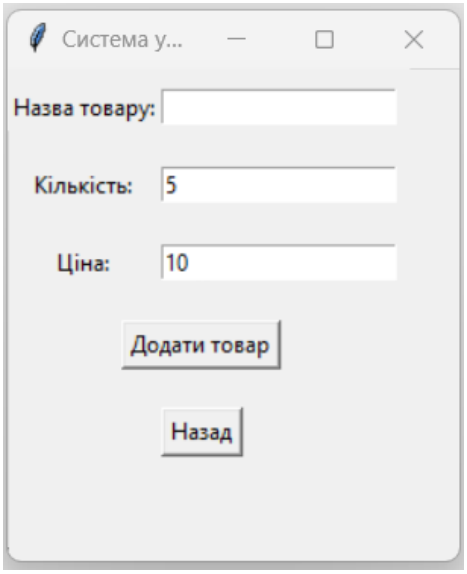
- Якщо товар вже існує в базі та користувач намагатиметься його повторно додати з такою ж ціною, то програма видасть сповіщення, що даний товар

вже існує . Якщо була отримана нова партія товару і необхідно оновити кількість, користувач зможе зробити це в розділі «Список товарів»

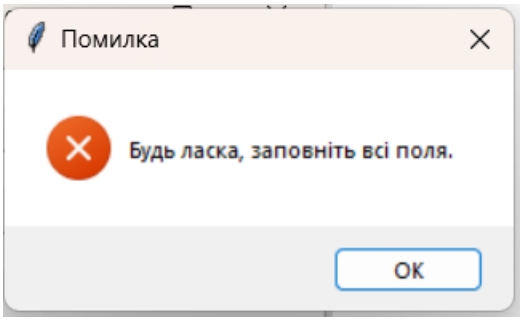


При цьому, поля вводу не будуть очищені, щоб користувач мав змогу редагувати дані в разі, якщо помилково було введено ціну чи назву.

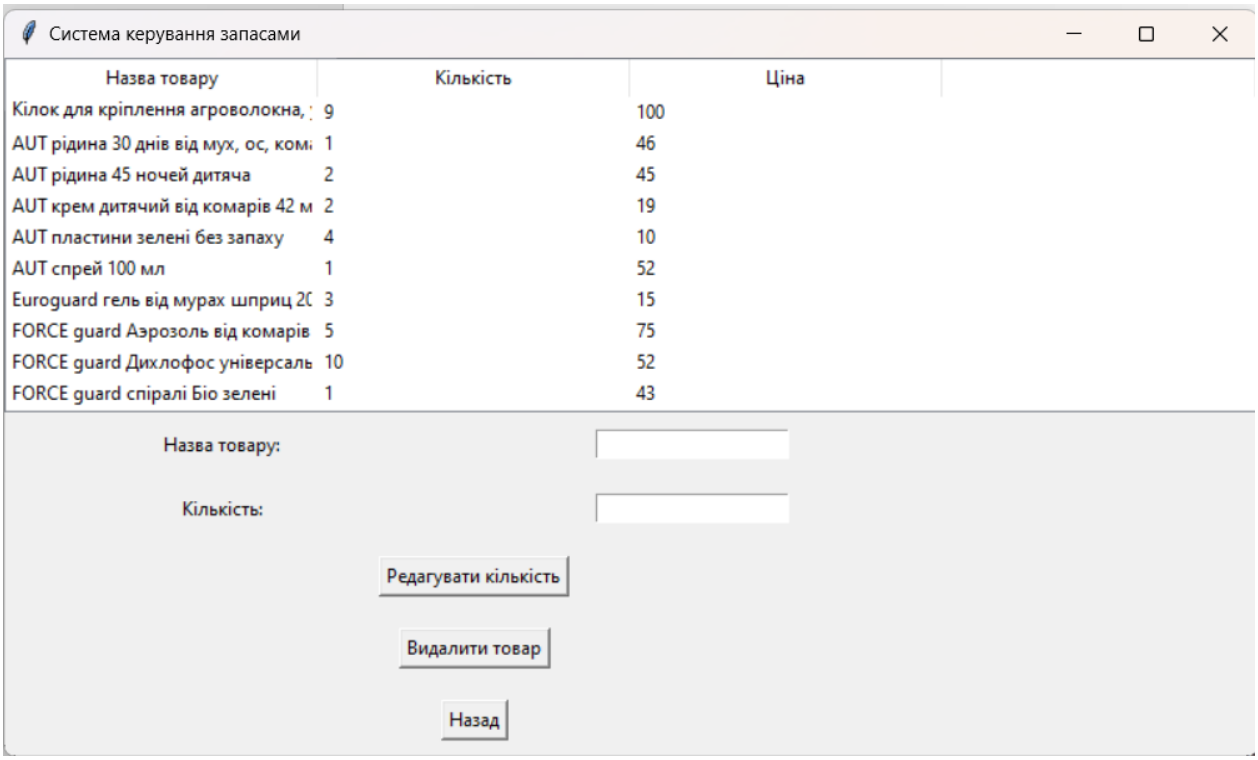
- Якщо поля будуть не коректно заповнені чи будуть відсутні дані в якомусь полі – приклад вводу:



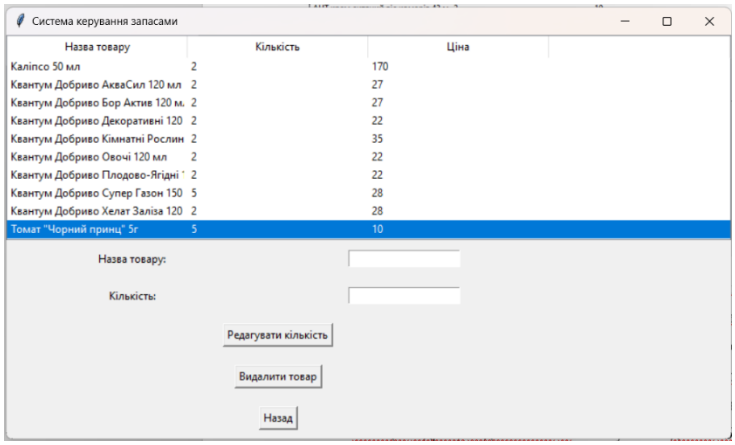
то користувачу, при спробі додати товар, натиснувши на однойменну кнопку в нижній частині, надійде відповідне сповіщення про помилку.

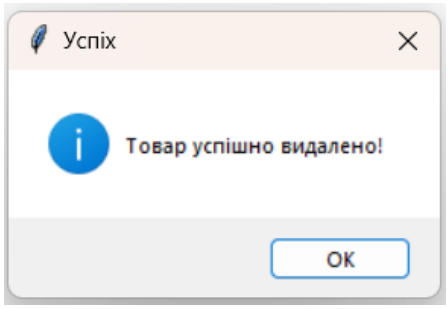


4. Якщо товар вже існує в базі даних і користувач хоче внести відомості про нову поставку без зміни ціни – необхідно зайти в розділ «Список товарів» та, обравши товар і вписавши нову кількість, натиснути на відповідну кнопку (буде описано далі).
5. За допомогою кнопки «Назад» користувач має змогу повернутись в загальне меню.
6. Кнопка «Переглянути товари» продемонструє користувачу список товарів з бази даних

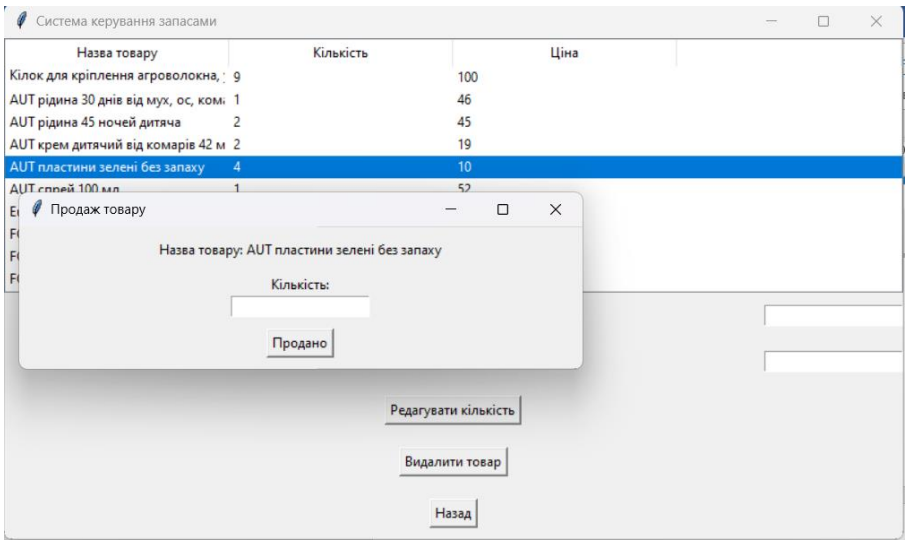


- Реалізована можливість видалення товару з бази даних для користувача напряму із застосунку. Виділивши рядок з необхідним товаром, натиснувши на нього один раз, користувач може обрати кнопку «Видалити товар» внизу робочого поля. Обрана позиція відразу видаляється зі списку і база даних зберігається в оновленому вигляді, як в програмі, так і окремо в файлі excel для зручної звітності.



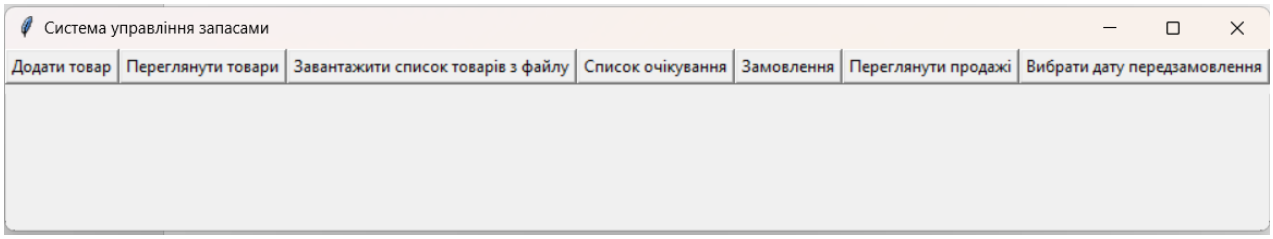


- Додаткова функція для аналізу продажів та побудування якісної моделі управління запасами – відслідковування проданих товарів.
  Вона реалізована подвійним кліком на позицію. Двічі натиснувши на рядок товару, користувачу демонструється вікно, де відображається назва товару і є можливість ввести кількість товару, яку було продано.

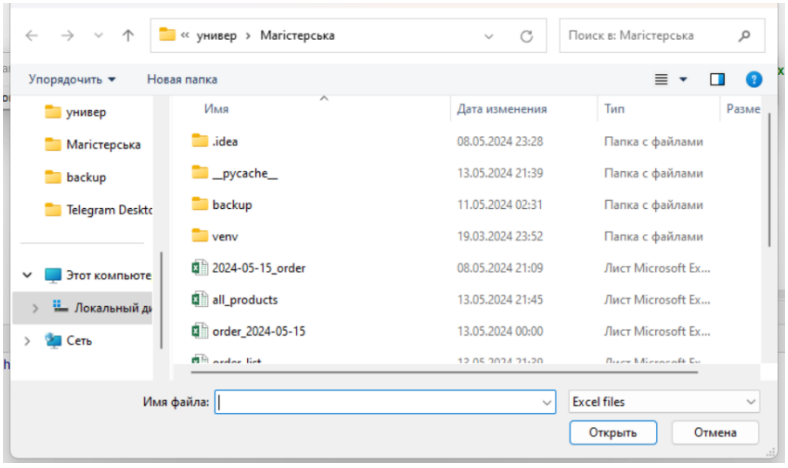


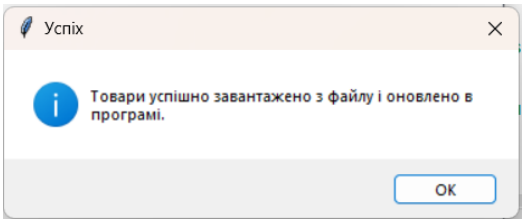
- Далі, натиснувши кнопку «Продано», кількість товару в системі буде оновлена та збережена, а товар у вказаній кількості буде відправлений до проданих товарів.

7. Кнопка «Завантажити список товарів з файлу» була додана для можливості швидко оновити асортимент товарів, якщо відбулась об’ємна закупівля і не доцільно буде додавати кожен товар вручну окремо.

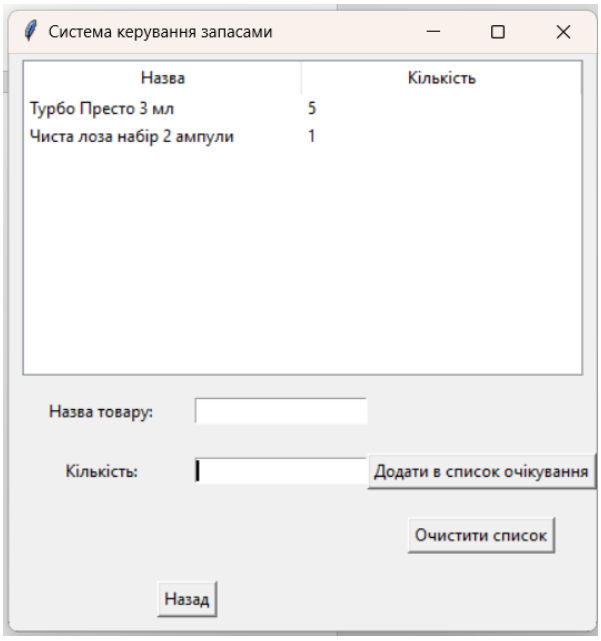


- Під час вибору цієї кнопки перед вами з’являється вікно вибору файлу для завантаження. При виборі файлу та завантаженні товарів з нього до бази даних, користувач також отримує сповіщення.



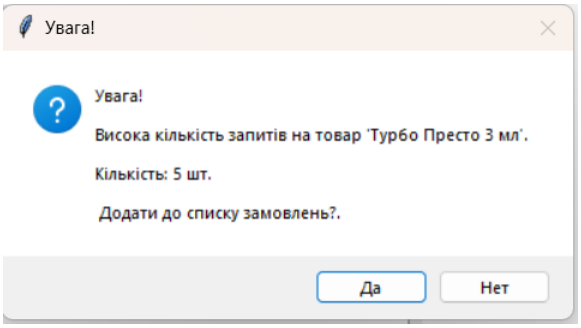


8. Кнопка «Список очікування» - розділ, який створений для збереження запитів клієнтів в список. За цим списком користувач зможе зрозуміти попит на товари, зберегти передзамовлення від клієнтів та, в разі успішного аналізу, перевести товари зі списку очікування автоматично до списку замовлень.



- При виборі кнопки «Список очікування», користувачу стає доступним під-меню, де верхня частина вікна програми– збережений список очікування на основі запиту користувачів, а нижня частина вікна - рядки для введення інформації про товар та активні кнопки: «Додати в список очікування», «Очистити список» та «Назад».
- Після додавання введеного товару в список очікування, поля для вводу очищуються для зручності користування.
- Наразі реалізований алгоритм надсилає користувачу сповіщення, щоразу при першому відкритті списку очікування, якщо певний товар має високу кількість запитів від клієнтів (3 і вище).

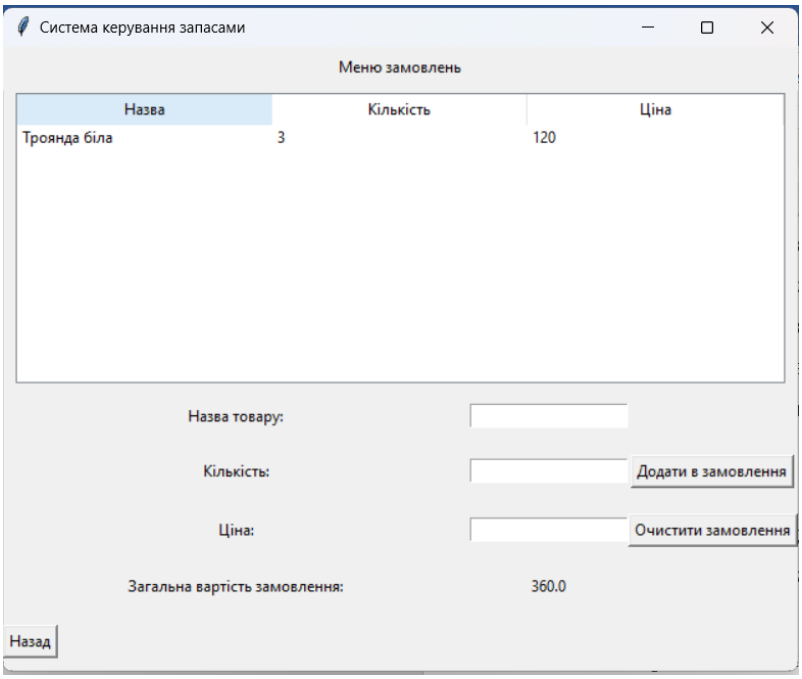
  Товари аналізуються окремо та почергово, тому далі користувач має можливість вибору дії в даному сповіщенні. Можна обрати – додати товар в список замовлень, чи залишити в списку очікування.



Обравши відповідну кнопку, користувач зможе автоматично перенести товар зі списку очікування до списку замовлень.

Паралельно, позиція з цим товаром видалиться зі списку очікування.

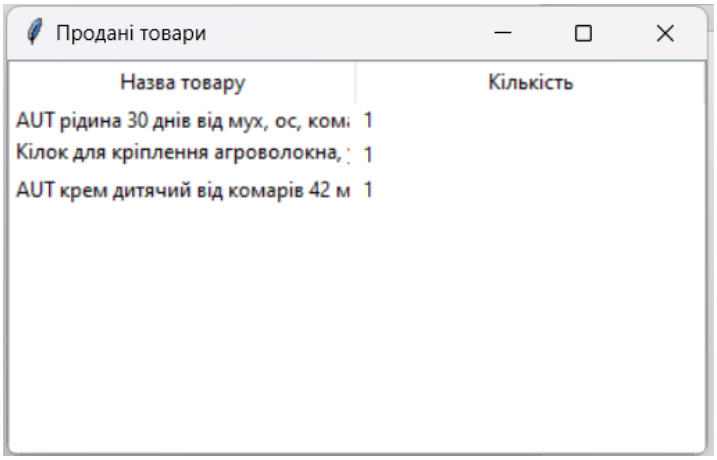
9. Кнопка «Замовлення» відкриває користувачу розділ із списком товарів, які необхідно придбати при найближчій закупівлі.



- Вікно програми даного розділу умовно поділено на дві частини. У верхній частині – поля для вводу. Нижче – кнопки: «Додати в замовлення», «Очистити замовлення» та «Назад». В нижній частині відображається загальна вартість замовлення на основі попередніх цін на товари, які вписав користувач при формуванні замовлення. Зручний функціонал для розрахунку витрат на майбутню закупівлю.

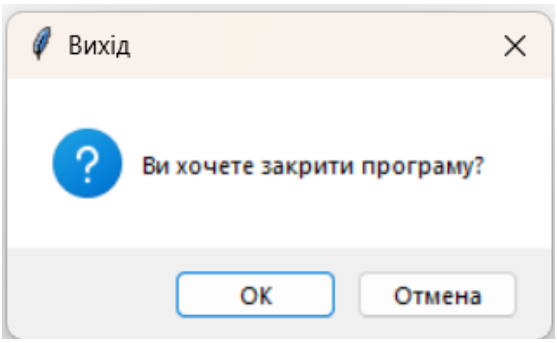
- При введені даних про товар у полях для вводу, та вибору кнопки «Додати товар», даний товар буде додано до списку в нижній частині а поля для вводу будуть очищені.
- Кнопка «Очистити замовлення» видаляє всі товари зі списку у вікні програми та у файлі списку замовлень. Зручно в разі, якщо користувач здійснив замовлення та бажає очистити відразу весь список для повторного наповнення.

10. Кнопка «Переглянути продажі» створена для перегляду користувачем списку проданих товарів. У даному розділі відображається назва товару та кількість, яка була продана.



- До даного списку товар переміщується користувачем з розділу «Список товарів»

11. З турботою про користувача, для зручності було додано запит перед виходом з програми



Використані програмні компоненти:

- **Tkinter:** Ця бібліотека використовується для створення графічного інтерфейсу користувача. Вона дозволяє легко розміщувати кнопки, поля вводу та інші елементи на вікні програми.
- **SQLite (через модуль database):** Для зберігання та управління базою даних товарів використовується SQLite. Це легкий та компактний реляційний двигун баз даних, який ідеально підходить для таких завдань.

Реалізація:

- **clear_window():** Ця функція використовується для очищення вікна Tkinter перед відображенням нового інтерфейсу. Вона перебирає всі дочірні елементи головного вікна і видаляє їх.
- **save_products_to_excel(products, file_name):** Функція для збереження списку товарів у файл Excel з вказаним ім'ям.
- **load_products_from_excel(file_name):** Функція для завантаження списку товарів з файлу Excel.
- **show_main_toolbar():** Функція створює головне меню програми з двома основними кнопками: "Додати товар" та "Переглянути товари".
- **show_add_product_toolbar():** Ця функція генерує інтерфейс для додавання нового товару.
- **delete_product_from_tree(tree, event):** Функція для видалення товару з бази даних та відповідного вузла з дерева відображення.
- **show_product_list():** Ця функція відображає список усіх товарів, що зберігаються в базі даних. Для кожного товару виводиться інформація про назву, кількість та ціну, а також кнопка для його видалення.

- **add_product_to_db(entry_name, entry_quantity, entry_price):** Функція отримує дані з полів введення та додає новий товар до бази даних через модуль **database**.
- **update_product_quantity(entry_name, entry_quantity, entry_price):** Ця функція відповідає за оновлення кількості існуючого товару. Вона використовує ім'я товару для пошуку в базі даних, потім додає нову кількість до існуючої і оновлює запис в базі даних.
- **show_waiting_list():** Функція для відображення списку очікування товарів та інтерфейсу додавання нового товару до цього списку.
- **add_to_waiting_list(tree, entry_name, entry_quantity):** Функція для додавання товару до списку очікування.
- **save_waiting_list_to_excel():** Функція для збереження списку очікування у файл Excel.
- **show_order_menu():** Функція для відображення списку замовлень та інтерфейсу додавання нового товару до цього списку.
- **add_to_order(entry_name, entry_quantity):** Функція для додавання товару до списку замовлень.
- **save_order_list_to_excel():** Функція для збереження списку замовлень у файл Excel.
- **delete_and_refresh(product_id):** Функція видаляє вказаний товар з бази даних за його ідентифікатором та оновлює відображення списку товарів.

Цей код є базовим функціоналом системи керування запасами і може бути розширений або модифікований відповідно до конкретних потреб користувача. Він забезпечує простий і ефективний спосіб управління товарними запасами для малих та середніх підприємств.

РОЗДІЛ 5. РЕАЛІЗАЦІЯ МОДЕЛЕЙ КЕРУВАННЯ ЗАПАСАМИ

Щоб визначити, яка модель краще підходить для розглянутого бізнесу, спробуємо порівняти їх ефективність на практиці, використовуючи реальні дані.

Спочатку в ході виконання роботи було зібрано необхідні для аналізу дані та створено загальну базу даних наявності товару.

1) Було створено зібрано список продажів за період 30 днів.

2) В даному списку продажів було вказано товари, кількість товарів, що були продані за 30 днів, роздрібну ціну продажу а також прораховано середній попит за найпростішою формулою:

$$\text{Середній попит} = \frac{\text{Кількість проданих товарів} * \text{Вартість}}{\text{Кількість днів (30)}}$$

3) Об'єктом дослідження є невеликий магазин у селі, тому відомо, що час доставки товарів - раз у тиждень, тобто час між поставками сталий і становить 7 днів.

4) Постачальники не стягують оплату за замовлення та його доставку. Витрати магазину на утримання товарів становлять комунальні платежі, заробітна плата найманого працівника, страховий платіж та дрібні витрати на господарські товари.

Обчислимо загальні витрати на утримання запасів за місяць, враховуючи всі вищезазначені витрати:

1. Оплата за комунальні послуги:
    - У весняно-літній період: 900 грн/місяць
    - У зимовий-осінній період: 1800 грн/місяць
2. Заробітна плата найманому працівнику: 4500 грн/місяць
3. Витрати на упакувальні матеріали, засоби індивідуального захисту та товари для підтримки чистоти: 800 грн/місяць

4. Витрати на страхування: 83,33 грн/місяць (1000 грн/рік)

Тепер складемо всі ці витрати:

- у весняно-літній період: 900 грн (комунальні послуги) + 4500 грн (заробітна плата) + 800 грн (упакувальні матеріали, засоби індивідуального захисту, товари для чистоти) + 83,33 грн (страхування) = 6283,33 грн/місяць

- у зимово-осінній період: 1800 грн (комунальні послуги) + 4500 грн (заробітна плата) + 800 грн (упакувальні матеріали, засоби індивідуального захисту, товари для чистоти) + 83,33 грн (страхування) = 7183,33 грн/місяць

Отже, загальні витрати на утримання запасів у магазині складають близько 6283,33 грн/місяць у весняно-літній період та 7183,33 грн/місяць у зимово-осінній період.

1) Найпростіша модель (без дефіциту)

Почнемо із застосування першої моделі EOQ (економічної партії замовлення) – без дефіциту.
Використаємо формули, зазначені в теоретичній частині та обрахуємо оптимальне замовлення для кожного товару.

Ми вже маємо середній попит за місяць (D) для кожного товару, що підрахований нами у файлі продажів, витрати на замовлення (S).

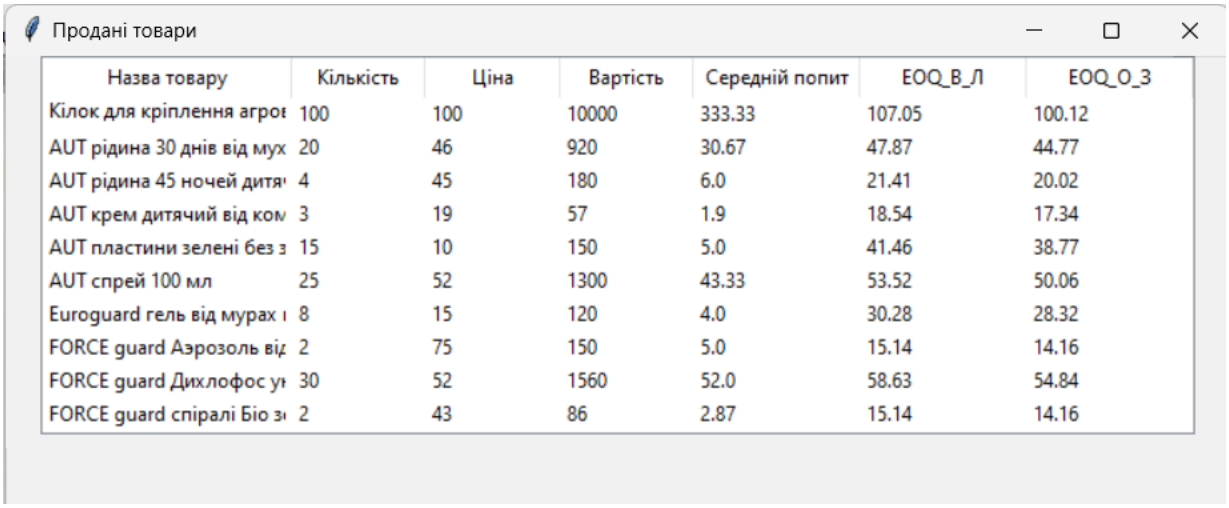
Обчислимо витрати на утримання одиниці товару на складі (H):

$$H_{\text{в,л}} = \frac{6283{,}33 \text{ грн}}{3000} = 5{,}550644876$$

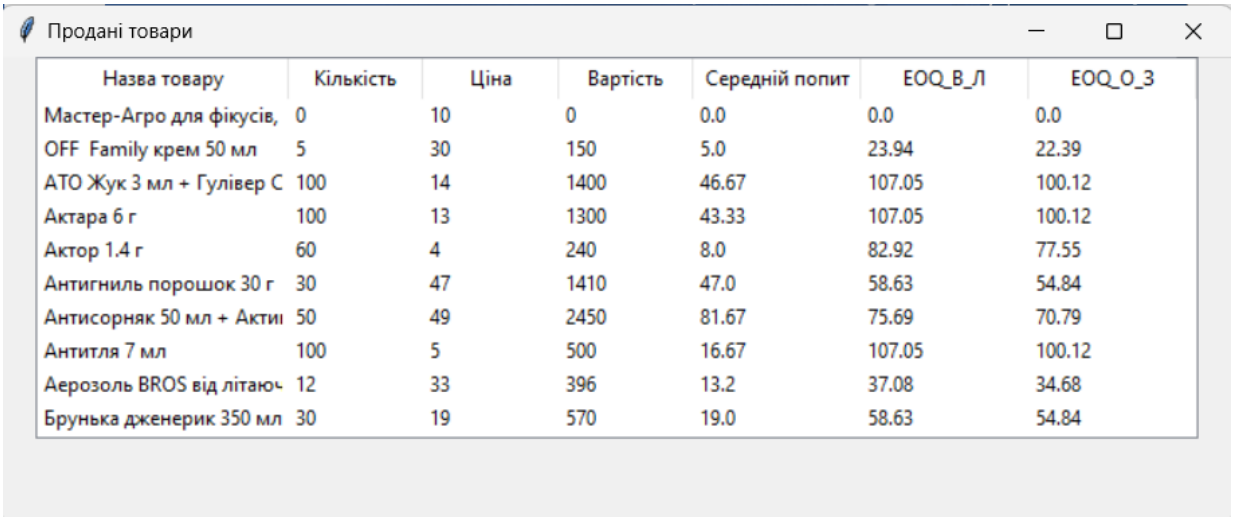
$$H_{\text{о,з}} = \frac{7183{,}33 \text{ грн}}{3000} = 6{,}345698$$

Де 3000 – загальна кількість одиниць товарів, що може вміщуватись на складі.

Підставимо дані в формулу та виконаємо обчислення в програмі. Результати представлені нижче:



Назва товару	Кількість	Ціна	Вартість	Середній попит	EOQ_В_Л	EOQ_О_З
Кілок для кріплення агрос	100	100	10000	333.33	107.05	100.12
AUT рідина 30 днів від мух	20	46	920	30.67	47.87	44.77
AUT рідина 45 ночей дитяч	4	45	180	6.0	21.41	20.02
AUT крем дитячий від ком	3	19	57	1.9	18.54	17.34
AUT пластини зелені без з	15	10	150	5.0	41.46	38.77
AUT спрей 100 мл	25	52	1300	43.33	53.52	50.06
Euroguard гель від мурах і	8	15	120	4.0	30.28	28.32
FORCE guard Аэрозоль від	2	75	150	5.0	15.14	14.16
FORCE guard Дихлофос ун	30	52	1560	52.0	58.63	54.84
FORCE guard спіралі Біо зі	2	43	86	2.87	15.14	14.16



Назва товару	Кількість	Ціна	Вартість	Середній попит	EOQ_В_Л	EOQ_О_З
Мастер-Агро для фікусів,	0	10	0	0.0	0.0	0.0
OFF Family крем 50 мл	5	30	150	5.0	23.94	22.39
АТО Жук 3 мл + Гулівер С	100	14	1400	46.67	107.05	100.12
Актара 6 г	100	13	1300	43.33	107.05	100.12
Актор 1.4 г	60	4	240	8.0	82.92	77.55
Антигниль порошок 30 г	30	47	1410	47.0	58.63	54.84
Антисорняк 50 мл + Акти	50	49	2450	81.67	75.69	70.79
Антитля 7 мл	100	5	500	16.67	107.05	100.12
Аерозоль BROS від літаюч	12	33	396	13.2	37.08	34.68
Брунька дженерик 350 мл	30	19	570	19.0	58.63	54.84

Після розрахунків, можемо побачити, що для багатьох продуктів значення $EOQ_{\text{в,л}}$ (весна-літо) та $EOQ_{\text{о,з}}$(осінь-зима) відрізняються. Це свідчить про те, що оптимальні розміри партії замовлення відрізняються в залежності від сезону.

Наприклад, продукти, які продаються краще влітку, можуть мати більші значення $EOQ_{\text{в,л}}$, оскільки попит на них зазвичай збільшується у цей період**.** Навпаки, для продуктів, популярних взимку, $EOQ_{\text{о,з}}$ може бути більшим, оскільки попит на них зазвичай зростає взимку. Також ця різниця помітна, враховуючи ріст витрат на утримання товарів в осінньо-зимовий період навідміну від весняно-літнього.

Також можемо помітити, що оптимальний розмір замовлення не є цілим числом майже скрізь. Для детального аналізу моделі та розгляду можливих варіантів зміни

вхідних даних, оберемо 2 товари зі списку, та детально розглянемо. Розрахунки окремо проведемо в Ексель.

Найменування	Продано за 30 днів D	Ціна за од	Вартість	Середній попит	EOQ_в_л	EOQ_о_з	Загальні витрати F_в_л	Загальні витрати F_о_з
Актара 6 г	100	13	1300	43,33333	107,0462	100,116	224,2022	240,2592
Бак емальований 16л	1	600	600	20	10,70462	10,0116	22,42022	24,02592

Використаємо формули, вказані в теоретичній частині.

Розпочнемо з товару №1. Вихідні дані :

Обсяг попиту за 30 днів = 100 шт;
витрати на розміщення замовлення візьмемо = 120 грн (витрати на доставку будь-якої партії Новою поштою);
витрати на зберігання в весняно-літній період становлять $H_{в,л} = 2{,}094443333 \approx 2{,}1$ грн, а в осінньо-зимовий період - $H_{о,з} = 2{,}394443333 \approx 2{,}4$ грн.

Кількість днів, за які велось спостереження = 30 днів. Час поставки товару – 7 днів.

Обчислимо оптимальний розмір замовлення даного товару:

$$Q_{в,л} = \sqrt{\frac{2DS}{H}} = \sqrt{\frac{2*100*120}{2{,}1}} = 106{,}9045 \approx 107$$

$$Q_{о,з} = \sqrt{\frac{2DS}{H}} = \sqrt{\frac{2*100*120}{2{,}4}} = 100$$

Оптимальне число замовлень за 30 днів:

$$N_1 = \frac{D}{Q} = \frac{100}{107} = 0{,}934597$$

$$N_2 = \frac{D}{Q} = \frac{100}{100} = 1$$

Так як за 30 днів було продано 100 одиниць товару, то в добу було продано $\frac{100}{30} = 3{,}3333$, що становить попит в день. Так як поновлення запасів може

відбуватись не раніше, ніж раз за 7 днів, то мінімальний рівень запасу, при якому потрібно здійснити замовлення (момент відновлення) складе: $3{,}3333 * 7 = 23{,}333$.

Мінімальні сукупні витрати при цьому будуть рівні:

$$F_{\text{в,л}} = \frac{D}{Q} * S + \frac{Q}{2} * H = \frac{100}{107} * 120 + \frac{107}{2} * 2{,}1 = 224{,}499 \approx 225 \text{ грн}$$

$$F_{\text{о,з}} = \frac{D}{Q} * S + \frac{Q}{2} * H = \frac{100}{100} * 120 + \frac{100}{2} * 2{,}4 = 240 \text{ грн}$$

Здійснимо прорахунок для товару №2:

Найменування	Продано за 30 днів D	Ціна за од	Вартість	Середній попит	EOQ_в_л	EOQ_о_з	Загальні витрати F_в_л	Загальні витрати F_о_з
Актара 6 г	100	13	1300	3,333333	107	100	224	240
Бак емальований 16л	1	600	600	0,033333	11	10	22	24

Формулою підрахунку оптимального часу між замовленнями було знехтувано і параметр t не був взятий до уваги у випадку даного економічного об'єкту, так як невеликі партії замовлень при обрахунку дали б нам велику величину проміжку між замовленнями і наступні результати були б менш «реалістичні». Даним параметром довелось знехтувати.

2) Модель оптимального розміру замовлення з дефіцитом

Візьмемо ті ж товари, як і в попередній моделі, використаємо ті ж вхідні дані, та припустимо, що нам відомий втрачений прибуток, який пов’язаний з нестачею (дефіцитом) товару. (Даний дефіцит в ситуації магазину, що розглядається в задачі, можна буде відслідкувати в розділі «Список очікування», вписуючи запити клієнтів на товар, якого нема в наявності). Нехай розмір такої втрати становить :
30*13 грн = 390 грн в місяць за товар №1. Використаємо цю інформацію, та спробуємо визначити, чи варто нам вдаватись до моделі з дефіцитом.

Обчислимо оптимальний розмір замовлення даного товару:

$$Q_{\text{в,л}} = \sqrt{\frac{2DS}{H} * \frac{Y+H}{Y}} = \sqrt{\frac{2*100*120}{2,1} * \frac{390+2,1}{390}} = 107,19193 \approx 107$$

$$Q_{\text{о,з}} = \sqrt{\frac{2DS}{H} * \frac{Y+H}{Y}} = \sqrt{\frac{2*100*120}{2,4} * \frac{390+2,4}{390}} = 100,30722 \approx 100$$

Оптимальний максимально можливий розмір замовлення:

$$s_1 = \sqrt{\frac{2DS}{H} * \frac{Y}{Y+H}} = \sqrt{\frac{2*100*120}{2,1} * \frac{390}{390+2,1}} = 106,617 \approx 107$$

$$s_2 = \sqrt{\frac{2DS}{H} * \frac{Y}{Y+H}} = \sqrt{\frac{2*100*120}{2,4} * \frac{390}{390+2,4}} = 99,6937 \approx 100$$

Мінімальні сукупні витрати при цьому будуть рівні:

$$F_{\text{в,л}} = \frac{D}{Q} * S + \frac{S^2}{2Q} * H + \frac{(Q-S)^2}{2Q} * Y = \frac{100}{107} * 120 + \frac{120^2}{2*107} * 2,1 + \frac{169}{2*107} * 390 =$$

$$= 544 \text{ грн}$$

$$F_{\text{о,з}} = \frac{100}{100} * 120 + \frac{120^2}{2*100} * 2,4 + \frac{400}{2*100} * 390 = 1062,149 \approx 1062 \text{ грн}$$

Як бачимо, у випадку розглянутого товару №1, при найпростішій моделі розмір оптимального замовлення дорівнює визначеному оптимальному замовленню в моделі із дефіцитом. Проте, модель замовлення з дефіцитом в кінцевому результаті приносить більше сукупних витрат. Прорахуємо дані для другого товару:

Найменування	Продано за 30 днів D	Ціна за од	Вартість	Середній попит	EOQ_в_л	EOQ_о_з	Загальні витрати F_в_л	Загальні витрати F_о_з
Актара 6 г	100	13	1300	43,33333	107	100	544	1062
Бак емальований 16л	1	600	600	20	11	10	335696	361663

Бачимо, що в ситуації із другим товаром модель з дефіцитом не принесла бажаного результату, а навпаки, в рази збільшила сукупні витрати, тому її використання стало вкрай нераціональним. Спробуємо видозмінити наші витрати і розглянемо випадок,

коли витрати на формування замовлення є мінімальними.

В першому випадку, розглянемо варіант, коли витрати на формування замовлення рівні 10 грн і перевіримо зміни оптимального замовлення та сукупних витрат:

Актара 6 г	100	13	1300	3,333333	31	29	30012	2449
Бак емальований 16л	1	600	600	0,033333	3	3	4686	5236

Можемо помітити, що в порівнянні з попереднім варіантом сукупні витрати товару №2 зменшились, в той час як витрати товару №1 значно зросли.

Спробуємо звести витрати на формування замовлення до мінімуму і встановимо це значення рівне 1. Відслідкуємо зміни:

Актара 6 г	100	13	1300	3,333333	10	9	1551	1425
Бак емальований 16л	1	600	600	0,033333	1	1	2	5

Як бачимо, в товарі з низьким попитом та високою собівартістю модель з дефіцитом мінімізує наші витрати лише в тому випадку, коли витрати на формування замовлення також зводяться до мінімуму. Товар №1, попит якого порівняно високий, навпаки – отримує збільшення сукупнх витрат, в разі зниження суми витрат на формування замовлення.

Порівнюючи результати $EOQ_{в,л}$ (весна-літо) та $EOQ_{о,з}$(осінь-зима) двох моделей, бачимо, що оцінка оптимальної кількості товару, яку слід замовляти майже не змінилась. Проте, загальні витрати при використанні дрегої моделі (з врахуванням дефіциту) значно зросли. В такому випадку, оптимізація товарів за першою схемою допоможе користувачу ефективніше керувати запасами, ніж оптимізація за другою.

РОЗДІЛ 6. ВИКОРИСТАННЯ МЕТОДУ СИМУЛЯЦІЙ ДЛЯ АНАЛІЗУ РОБОТИ МОДЕЛЕЙ

Короткий опис методу симуляцій

Метод симуляцій - це процес використання комп'ютерного моделювання для аналізу поведінки системи в різних умовах та стратегіях. У контексті управління запасами, симуляція дозволяє вам досліджувати різні параметри і стратегії управління запасами та аналізувати їх вплив на ефективність підприємства.

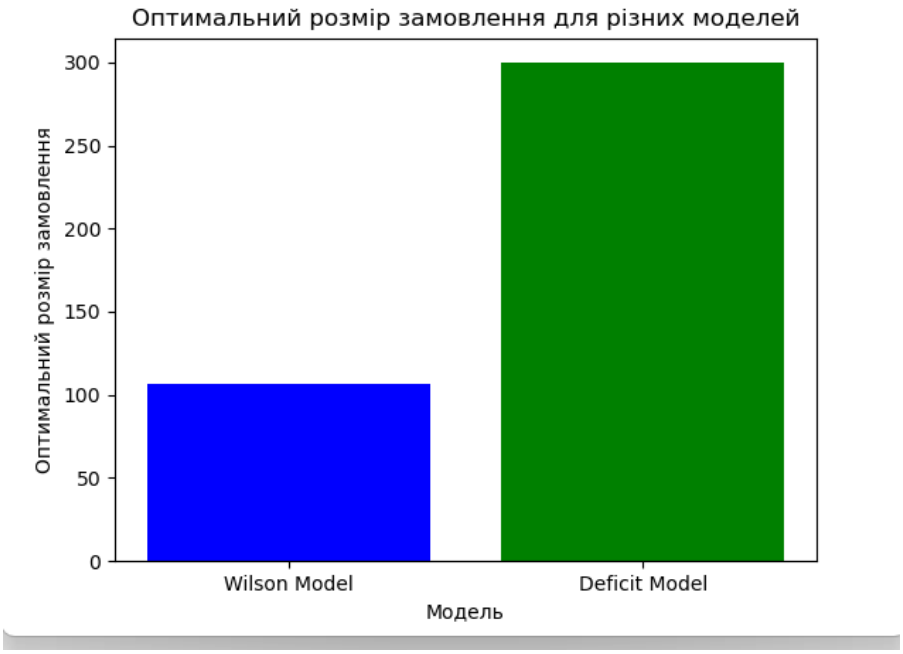
Основні кроки методу симуляцій управління запасами включають:

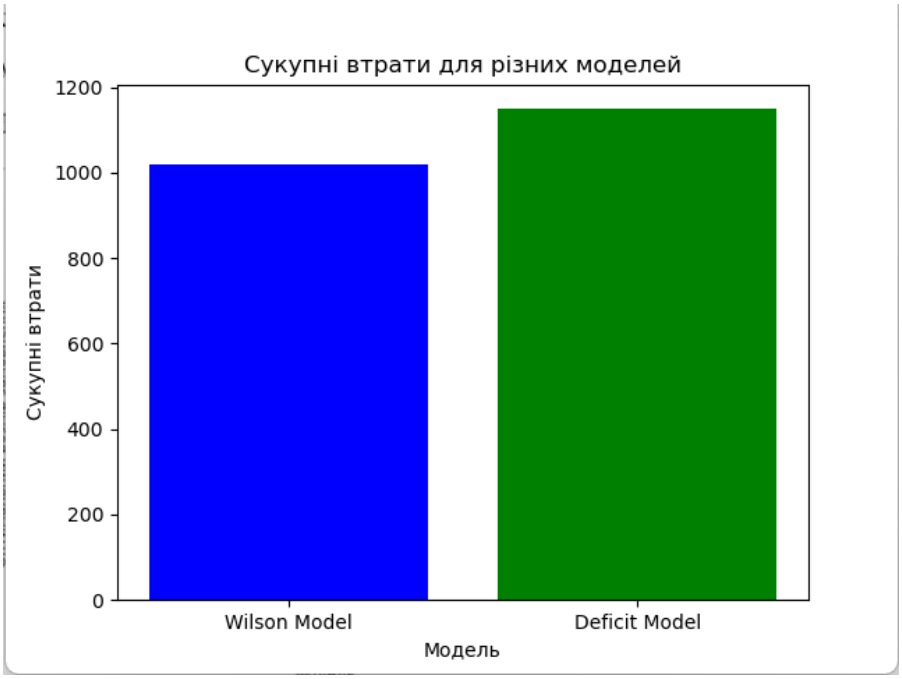
1. **Визначення моделі**: Спочатку потрібно визначити математичну модель системи управління запасами, яка описує взаємодію між різними параметрами, такими як розмір замовлення, час між замовленнями, рівень сервісу клієнтів та інші.
2. **Вибір параметрів та стратегій**: Вибрати параметри моделі, які потрібно досліджувати, такі як розмір замовлення, час між замовленнями, рівень сервісу клієнтів тощо. Також обрати різні стратегії управління запасами, які ви потрібно порівняти, наприклад, зміна рівня сервісу клієнтів, оптимізація розмірів замовлень або зміна частоти замовлень.
3. **Симуляція**: Виконати симуляцію кожної стратегії, використовуючи визначені параметри та модель.

Використання методу симуляцій

Використаємо метод симуляцій для наочного порівняння обраних нами моделей. Далі, дані, отримані в ході роботи моделей під час першої симуляції, проаналізуємо зміни після роботи моделей та використаємо модифіковані дані для другого прогону методу.

Перший прогін:

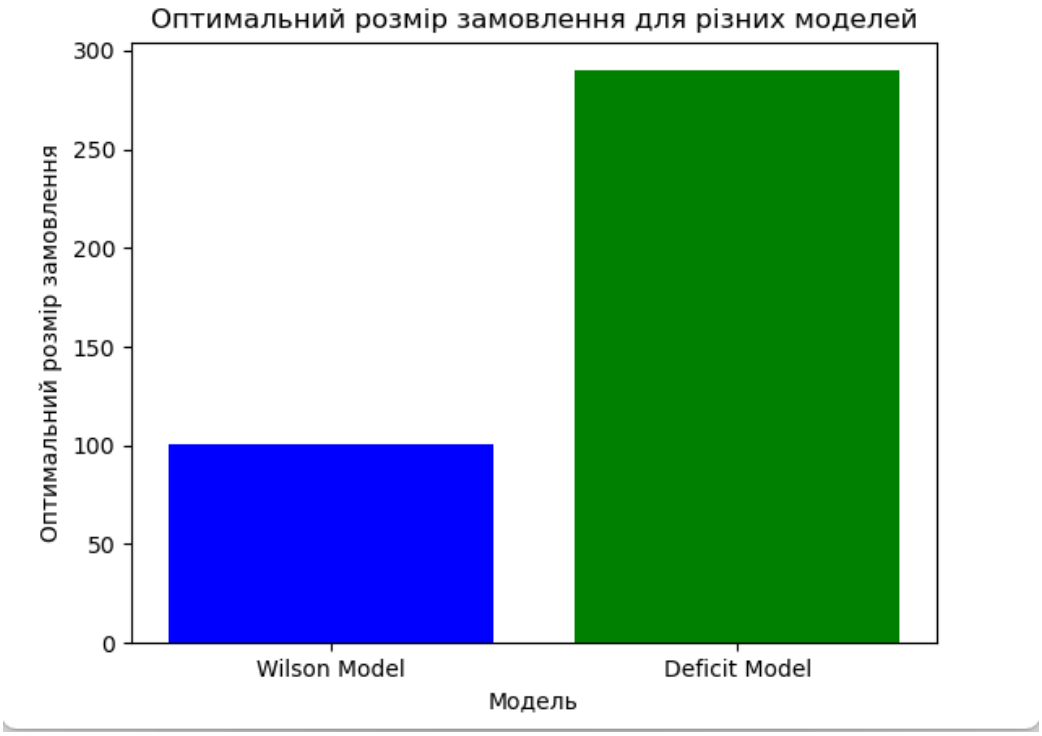


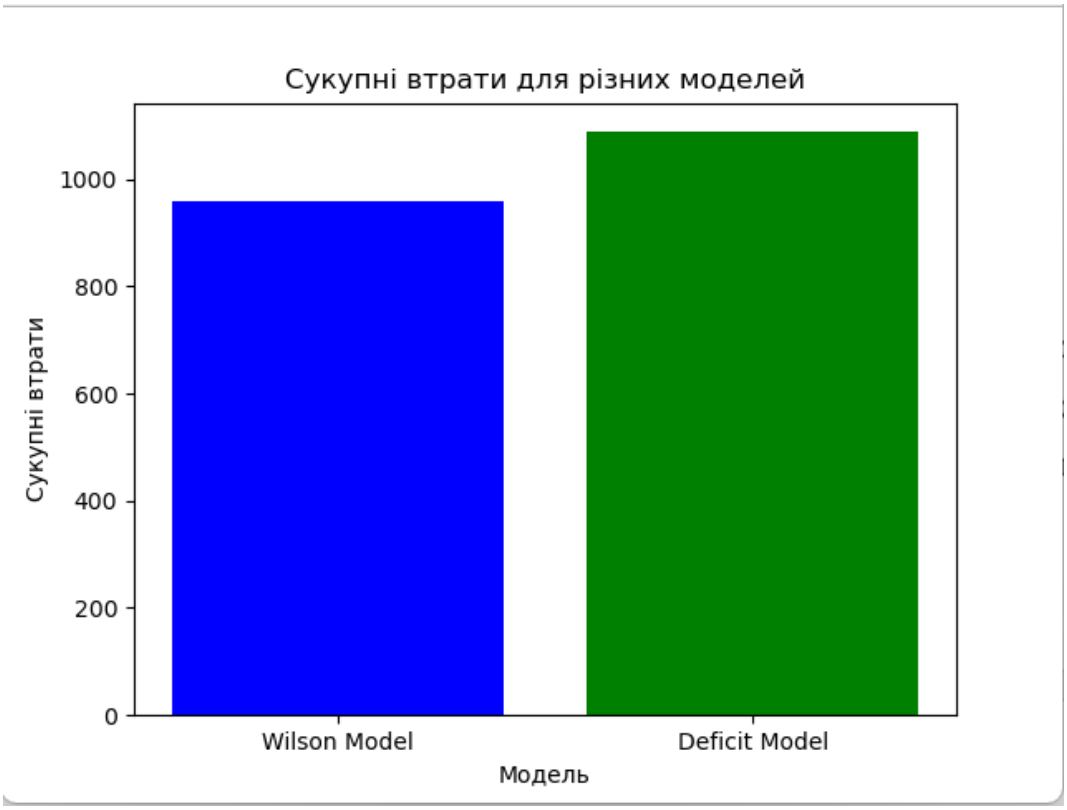


В першому прогоні методу було використано вихідні дані товару №1 та формули з теоретичної частини. Додатково було вказано таку змінну, як «Рівень сервісу клієнтів». Можемо побачити, що сукупні витрати моделі з дефіцитом значно вищі за сукупні витрати простішої моделі (без дефіциту), як і при попередніх обрахуваннях.

Спробуємо використати дані, які отримались в результаті першого прогону та дещо модифікуємо вихідні дані, виходячи з отриманих результатів.
При обчисленнях було обраховану оптимальну кількість замовлень за період, що розглядається – 1. Тобто, після обчислень пропонується робити замовлення не кожні 7 днів в місяці, а лише 1 раз на місяць. Використаємо ці дані і змінимо час між замовленнями з 7 днів на 30 днів. Також, зменшиться сума витрат на оформлення замовлення, так як транспортування більшої кількості замовлень стягує більшу оплату. Відповідно, коли замовлення здійснювалось кожні 7 днів – оплата на оформлення замовлення становила 120 грн. Понизимо її до 100 грн.

Для другої моделі (з дефіцитом) понизимо втрати від дефіциту, так як партія замовлення збільшилась і дефіцит прогнозовано меншим. З 390 понизимо до 350 грн. Тепер на оновлених даних, з використанням отриманого розміру оптимального замовлення проведемо повторний прогін:





Бачимо, що при повторному прогоні на оновлених даних перша модель все ще має відчутну перевагу при порівнянні сукупних витрат, тому робимо висновок, що модель з дефіцитом є менш успішною при керування запасами даного магазину.

Проте, на загальній картині спостерігаєтсья зниження сукупних витрат в результаті використання даних, запропонованих за результатами першого відпрацювання моделей.

ВИСНОВОК

Результатом даної магістерської роботи було створення корисного програмного застосунку з графічним інтерфейсом для користувача. Дана програма стане інструментом обліку товарів магазину, допоможе користувачу відслідковувати рівень запасів, зміну цін, слідкувати за запитами клієнтів та формувати замовлення. Також, на основі зібраної інформації з даного застосунку, було досліджено використання двох моделей оптимізації управління запасами з метою зниження витрат та оптимізації процесів управління запасами.

Було розглянуто найпростішу модель EOQ (економічний розмір замовлення) та її модифікацію – модель оптимального розміру замовлення з дефіцитом.

Аналізуючи результати досліджень, було встановлено, що модель з дефіцитом демонструє більшу гнучкість і точність в управлінні запасами, оскільки вона дозволяє врахувати втрату дефіциту та є складнішою, за рахунок більшої кількості врахованих факторів. Проте, при аналізі результатів використання даної моделі, було виявлено її очевидний програш, в порівнянні з простішою моделлю.

На відміну від моделі без врахування дефіциту, друга модель дала гірші результати при роботі вперше і при повторному прогоні з використанням методу симуляцій.

Перша модель, в порівнянні з другою, показала менші сукупні втрати, що було метою роботи моделей керування запасами. Також, при використанні даних які отримались при роботі моделі вперше, в порівнянні з другою моделлю вона повторно показала кращий результат і покращилась порівняно з власними результатами при першому використанні.

Функціональне наповнення даної програми можна вдосконалювати. Запланована робота також над графічним інтерфейсом для покращення користувацького інтерфейсу в програмі.

Можливі функціональні оновлення:

- В розділі «Завантажити список товарів з файлу» планується покращена можливість відстежування дублювань. Наразі програма не додає дубльовані позиції до бази, якщо такі є в файлі. Оновлення передбачає можливість аналізу позиції та виведення сповіщення, в якому користувачу буде запропонований вибір: збільшити кількість товару на складі, якщо така позиція вже є, оновити ціну, якщо відбулось підняття, чи додати товар окремо якщо відбулась зміна ціни.
- Покращення програми може відбутися за рахунок розподілу товарів даного магазину на типи. Це допоможе користувачу швидше обробляти списки замовлень в разі, якщо різні постачальники достявляють різні типи товарів.
- Програму буде в реальному часі підв’язано до календаря, з можливістю обрати дні, в які магазин має здійснити передзамовлення. За день до дати користувач отримуватиме сповіщення про необхідність звернути увагу на список замовлень певного типу товарів та актуалізувати його.
- Заплановано перенесення додатку до мобільної версії для доступності користувачам незалежно від апаратного забезпечення.

СПИСОК ЛІТЕРАТУРИ

1. Adams, C.R., Early, M.P. Principles of Horticulture. – Butterworth-Heinemann, 2004. – 432 с.
2. Ash, M., Ash, I. Agrochemicals: Preparation and Mode of Action. – CRC Press, 2018. – 550 с.
3. Barnard, F.L. Agribusiness Management. – McGraw-Hill Education, 2019. – 560 с.
4. McKinney, W. Python for Data Analysis. – O'Reilly Media, 2017. – 544 с.
5. Raschka, S., Mirjalili, V. Python Machine Learning. – Packt Publishing, 2017. – 454 с.
6. Wallace, T., Stahl, R.A. Effective Inventory Management. – Chapman & Hall, 2004. – 320 с.
7. Гребінь, В.В. Управління запасами в сучасних умовах господарювання. – Київ: КНЕУ, 2017. – 312 с.
8. Коваленко, В.В. Системи управління запасами. – Київ: Центр учбової літератури, 2019. – 328 с.
9. Проценко І. Ю. Оптимальні стратегії для багатопродуктових моделей керування запасами [Текст] : дис. канд. фіз.-мат. наук : 01.05.04 / Ірина Юріївна Проценко – Київ, 2016, – 127 с.
10. Семенова, Л.І., Литвиненко, О.В. Основи управління запасами. – Київ: КНЕУ, 2015. – 288 с.
11. Ткаченко, В.М. Організація та управління підприємствами АПК. – Київ: КНЕУ, 2016. – 272 с.
12. Шевченко, І.О. Теорія і практика управління запасами. – Київ: Видавництво "Старого Лева", 2017. – 240 с.

ІНТЕРНЕТ ДЖЕРЕЛА

13. http://www.agrosvit.info/pdf/3_2016/9.pdf
14. https://ekmair.ukma.edu.ua/home
15. https://uk.wikipedia.org/wiki/Економічний_розмір_замовлення
16. https://uk.wikipedia.org/wiki/Управління_товарними_запасами
17. https://ela.kpi.ua/server/api/core/bitstreams/31cf409b-4ce6-4a15-a943-bb342925d892/content
18. https://lib.chmnu.edu.ua/pdf/naukpraci/computer/2014/250-238-14.pdf
19. http://dspace.oneu.edu.ua/jspui/bitstream/123456789/522/1/%D0%90%D1%80%D1%85%D1%96%D0%BF%D0%BE%D0%B2%20%D0%A1.%D0%92.%20%D0%9C%D0%BE%D0%B4%D0%B8%D1%84%D1%96%D0%BA%D0%B0%D1%86%D1%96%D1%8F%20%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%96%20%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%96%D0%BD%D0%BD%D1%8F%20%D0%B7%D0%B0%D0%BF%D0%B0%D1%81%D0%B0%D0%BC%D0%B8%20%D0%A5%D0%B0%D1%80%D1%80%D1%96%D1%81%D0%B0-%D0%A3%D1%96%D0%BB%D1%81%D0%BE%D0%BD%D0%B0.pdf
20. https://studfile.net/preview/5163126/page:6/
21. http://www.economy.nayka.com.ua/?op=1&z=4299
22. https://zakon.rada.gov.ua/laws/show/z0751-99#Text

ДОДАТКИ

Код додатку для обліку товарів

main.py

```
import tkinter as tk
from tkinter import messagebox
from interface import show_main_toolbar  # Змінено з "from interface import show_main_toolbar"
from database import create_table, close_connection

def main():
    try:
        create_table()  # Створюємо таблицю при запуску програми
        root = tk.Tk()
        root.title('Система керування запасами')
        show_main_toolbar()  # Передаємо вікно root у функцію show_main_toolbar
        root.protocol("WM_DELETE_WINDOW", lambda: on_closing(root))
        root.mainloop()
    except Exception as e:
        messagebox.showerror("Error", f"An error occurred: {e}")

def on_closing(root):
    if messagebox.askokcancel("Вихід", "Ви хочете закрити програму?"):
        close_connection()  # Закриваємо з'єднання з базою даних перед виходом з програми
        root.destroy()

if __name__ == "__main__":
    main()
```

database.py

```
import sqlite3

def create_table():
    connection = sqlite3.connect('inventory.db')
    cursor = connection.cursor()
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS products (
            id INTEGER PRIMARY KEY,
            name TEXT NOT NULL,
            quantity INTEGER NOT NULL,
            price REAL NOT NULL
        )
    ''')
    connection.commit()
    connection.close()

def add_product(name, quantity, price):
    connection = sqlite3.connect('inventory.db')
    cursor = connection.cursor()
    cursor.execute('INSERT INTO products (name, quantity, price) VALUES (?, ?, ?)', (name, quantity, price))
    connection.commit()
    connection.close()
```

```
def check_product(name):
    connection = sqlite3.connect('inventory.db')
    cursor = connection.cursor()
    cursor.execute('SELECT * FROM products WHERE name=?', (name,))
    product = cursor.fetchone()
    connection.close()
    return product

def update_quantity(name, new_quantity):
    connection = sqlite3.connect('inventory.db')
    cursor = connection.cursor()
    cursor.execute('UPDATE products SET quantity=? WHERE name=?', (new_quantity, name))
    connection.commit()
    connection.close()


def get_product_list():
    connection = sqlite3.connect('inventory.db')
    cursor = connection.cursor()
    cursor.execute('SELECT * FROM products')
    products = cursor.fetchall()
    connection.close()
    return products

def delete_product(id):
    connection = sqlite3.connect('inventory.db')
    cursor = connection.cursor()
    cursor.execute('DELETE FROM products WHERE id=?', (id,))
    connection.commit()
    connection.close()
```

data_handler.py

```
from database import get_product_list

def display_products():
    products = get_product_list()  # Замінено з get_all_products() на get_product_list()
    if products:
        for product in products:
            print(product)
    else:
        print("No products available")

def main():
    display_products()

if __name__ == "__main__":
    main()
```

interface.py

```
import tkinter as tk
from tkinter import ttk, messagebox, filedialog, simpledialog
from datetime import datetime, timedelta
import openpyxl
import os
import tkcalendar
import pickle
import sys
import math
from database import add_product, check_product, update_quantity, get_product_list, delete_product

waiting_list = []
order_list = []
product_notification_shown = False
waiting_list_notification_shown = False
all_products = []

def clear_window():
    for widget in root.winfo_children():
        widget.destroy()

def save_products_to_excel(products, file_name):
    wb = openpyxl.Workbook()
    ws = wb.active
    for product in products:
        ws.append(product)
    wb.save(file_name)

def load_products_from_excel(file_name):
    if os.path.exists(file_name):
        try:
            wb = openpyxl.load_workbook(file_name)
            sheet = wb.active
            new_products = []

            for row in sheet.iter_rows(values_only=True):
                new_products.append(row)

            return new_products
        except Exception as e:
            messagebox.showerror("Error", f"An error occurred while loading products: {str(e)}")
    else:
        return []

def load_lists_from_files():
    global all_products, waiting_list
    all_products = load_products_from_excel("all_products.xlsx")
    waiting_list = load_products_from_excel("waiting_list.xlsx")

def show_main_toolbar():
    global root

    if not root:
        root = tk.Tk()
        root.title('Система управління запасами')
```

```
        root.protocol("WM_DELETE_WINDOW", on_closing)
    else:
        # Очищаємо вміст вікна перед відображенням нового вмісту
        clear_window()

    load_lists_from_files()
    load_order_list_from_excel()

    button_add = tk.Button(root, text='Додати товар', command=show_add_product_toolbar)
    button_add.grid(row=0, column=0)

    button_view = tk.Button(root, text='Переглянути товари', command=show_product_list)
    button_view.grid(row=0, column=1)

    button_load_from_excel = tk.Button(root, text='Завантажити список товарів з файлу',
command=lambda:load_products_from_file_and_update_list())
    button_load_from_excel.grid(row=0, column=2)

    button_waiting_list = tk.Button(root, text='Список очікування', command=show_waiting_list)
    button_waiting_list.grid(row=0, column=3)

    button_order_menu = tk.Button(root, text='Замовлення', command=show_order_menu)
    button_order_menu.grid(row=0, column=4)

    button_view_sales = tk.Button(root, text="Переглянути продажі", command=show_sold_products)
    button_view_sales.grid(row=0, column=5)

    button_choose_order_date = tk.Button(root, text='Вибрати дату передзамовлення', command=choose_order_date)
    button_choose_order_date.grid(row=0, column=6)

    critical_products_count = sum(1 for product in all_products if int(product[2]) <= 3)
    if critical_products_count > 0:
        messagebox.showinfo("Увага", f"Кількість товарів з критичним рівнем запасу: {critical_products_count}")

    root.mainloop()

def load_products_from_excel_and_show():
    root.title('Список очікування')
    products = load_products_from_excel("waiting_list.xlsx")
    global waiting_list
    waiting_list = products
    show_waiting_list()

def load_products_from_file_and_update_list():
    file_path = tk.filedialog.askopenfilename(filetypes=[("Excel files", "*.xlsx"), ("All files", "*.*")])
    if file_path:
        new_products = load_products_from_excel(file_path)
        if new_products:
            all_products.extend(new_products)
            save_products_to_excel(all_products, "all_products.xlsx")
            show_product_list()
            messagebox.showinfo("Успіх", "Товари успішно завантажено з файлу і оновлено в програмі.")

def show_add_product_toolbar():
    clear_window()

    label_name = tk.Label(root, text='Назва товару:')
    label_name.grid(row=1, column=0, pady=10)
```

```
    entry_name = tk.Entry(root)
    entry_name.grid(row=1, column=1)

    label_quantity = tk.Label(root, text='Кількість:')
    label_quantity.grid(row=2, column=0, pady=10)
    entry_quantity = tk.Entry(root)
    entry_quantity.grid(row=2, column=1)

    label_price = tk.Label(root, text='Ціна:')
    label_price.grid(row=3, column=0, pady=10)
    entry_price = tk.Entry(root)
    entry_price.grid(row=3, column=1)

    button_add = tk.Button(root, text='Додати товар', command=lambda: add_product_to_db(entry_name, entry_quantity,
entry_price))
    button_add.grid(row=4, column=0, columnspan=2, pady=10)

    button_back = tk.Button(root, text='Назад', command=show_main_toolbar)
    button_back.grid(row=7, column=0, columnspan=2, pady=10)

def add_to_waiting_list(tree, entry_name, entry_quantity):
    name = entry_name.get()
    quantity = entry_quantity.get()

    if not name or not quantity:
        messagebox.showerror("Помилка", "Будь ласка, заповніть всі поля.")
        return

    tree.insert("", "end", values=(name, quantity))
    global waiting_list
    waiting_list.append((name, quantity))
    entry_name.delete(0, tk.END)
    entry_quantity.delete(0, tk.END)

    save_products_to_excel(waiting_list, "waiting_list.xlsx")

    check_waiting_list(tree)

def save_waiting_list_to_excel():
    save_products_to_excel(waiting_list, "waiting_list.xlsx")

def edit_product_quantity(tree, entry_quantity):
    selected_item = tree.selection()
    if selected_item:
        quantity = entry_quantity.get()
        if quantity.isdigit():
            new_quantity = int(quantity)
            product_name = tree.item(selected_item, "values")[0]
            for index, product in enumerate(all_products):
                if product[1] == product_name:
                    all_products[index] = (product[0], product_name, new_quantity, product[3])
                    update_quantity(product_name, new_quantity)
                    break
            tree.item(selected_item,
                      values=(product_name, new_quantity, tree.item(selected_item, "values")[2], "Видалити"))
            messagebox.showinfo("Успіх", "Кількість товару успішно оновлено!")
            save_products_to_excel(all_products, "all_products.xlsx")
        else:
```

```
            messagebox.showerror("Помилка", "Будь ласка, введіть ціле число для кількості.")
    else:
        messagebox.showerror("Помилка", "Будь ласка, виберіть товар для редагування.")


def show_product_sale_window(tree):
    selected_item = tree.selection()
    if selected_item:
        product_name = tree.item(selected_item, "values")[0]
        quantity = tree.item(selected_item, "values")[1]
        sale_window = tk.Toplevel(root)
        sale_window.title("Продаж товару")

        label_name = tk.Label(sale_window, text=f"Назва товару: {product_name}")
        label_name.pack(pady=10)

        label_quantity = tk.Label(sale_window, text="Кількість:")
        label_quantity.pack()
        entry_quantity = tk.Entry(sale_window)
        entry_quantity.pack()

        button_sold = tk.Button(sale_window, text="Продано",command=lambda: mark_as_sold(tree, selected_item,
product_name, entry_quantity.get(), sale_window))
        button_sold.pack(pady=10)

        sale_window.grab_set()

def mark_as_sold(tree, selected_item, product_name, sold_quantity, sale_window):
    if sold_quantity.isdigit():
        sold_quantity = int(sold_quantity)
        for index, product in enumerate(all_products):
            if product[0] == product_name:
                remaining_quantity = int(product[1]) - sold_quantity
                all_products[index] = (product_name, remaining_quantity, product[2])
                update_quantity(product_name, remaining_quantity)
                break
        tree.item(selected_item, values=(product_name, remaining_quantity, tree.item(selected_item, "values")[2],
"Продано"))
        save_products_to_excel(all_products, "all_products.xlsx")
        save_sold_product(product_name, sold_quantity)  # Функція для збереження проданих товарів у файлі Excel
        sale_window.destroy()
    else:
        messagebox.showerror("Помилка", "Будь ласка, введіть ціле число для кількості проданих товарів.")

def save_sold_product(product_name, sold_quantity):
    # Збереження проданих товарів у файлі Excel
    wb = openpyxl.load_workbook("sold_products.xlsx")
    ws = wb.active
    ws.append((product_name, sold_quantity))
    wb.save("sold_products.xlsx")


def show_sold_products():
    # Функція для відображення списку проданих товарів
    sold_products = load_products_from_excel("sold_products.xlsx")
    if sold_products:
        sold_window = tk.Toplevel(root)
        sold_window.title("Продані товари")
```

```
        tree = ttk.Treeview(sold_window,columns=("Name", "Quantity", "Price", "Value", "Average Demand",
"EOQ_B_Л", "EOQ_O_З", "S(t)", "RQ", "F(RQ,t)"), show="headings")
        tree.heading("Name", text="Назва товару")
        tree.heading("Quantity", text="Кількість")
        tree.heading("Price", text="Ціна")
        tree.heading("Value", text="Вартість")
        tree.heading("Average Demand", text="Середній попит")
        tree.heading("EOQ_B_Л", text="EOQ_B_Л")
        tree.heading("EOQ_O_З", text="EOQ_O_З")
        tree.heading("S(t)", text="S(t)")
        tree.heading("RQ", text="RQ")
        tree.heading("F(RQ,t)", text="F(RQ,t)")

        tree.column("Name", width=150)  # Звузити стовпець "Назва товару"
        tree.column("Quantity", width=80)
        tree.column("Price", width=80)
        tree.column("Value", width=80)
        tree.column("Average Demand", width=100)
        tree.column("EOQ_B_Л", width=100)
        tree.column("EOQ_O_З", width=100)
        tree.column("S(t)", width=100)
        tree.column("RQ", width=100)
        tree.column("F(RQ,t)", width=100)

    for product in sold_products:
            name, quantity = product[0], product[1]
            # Перевірка, чи є ціна в списку проданих товарів
            try:
                price = product[2]
            except IndexError:
                price = None
            if not price:
                # Якщо ціни немає, виводимо запит на введення
                price = tk.simpledialog.askfloat("Введіть ціну", f"Введіть ціну для товару '{name}':")
                if price is None:
                    continue  # Якщо користувач натиснув "Скасувати", перейти до наступного товару
            value = quantity * price  # Обчислення вартості
            average_demand = round(value / 30, 2)  # Обчислення середнього попиту, округлення до 2 знаків після коми
            EOQ_V_L = round(math.sqrt(2 * quantity * price / 5.50645), 2)
            EOQ_O_Z = round(math.sqrt(2 * quantity * price / 6.345698), 2)
            RQ = round(math.sqrt((2 * quantity * 120) / (5.50645 * 7)), 2)
            if RQ != 0:
                S_t = round(quantity / RQ, 2)
                F_RQ_t = round(((quantity * S_t) / RQ) + ((RQ / 2) * 5.550644876 * 7), 2)
            else:
                S_t = 0
                F_RQ_t = 0
            tree.insert("", "end", values=(name, quantity, price, value, average_demand,EOQ_V_L, EOQ_O_Z, S_t, RQ,
F_RQ_t))
            tree.pack()

    else:
        messagebox.showinfo("Інформація", "Немає проданих товарів.")

def show_price(tree):
    # Функція для відображення ціни товарів
    price_window = tk.Toplevel(root)
    price_window.title("Ціна товарів")
```

```
    sold_products = load_products_from_excel("sold_products.xlsx")
    if sold_products:
        price_tree = ttk.Treeview(price_window, columns=("Name", "Price"), show="headings")
        price_tree.heading("Name", text="Назва товару")
        price_tree.heading("Price", text="Ціна")
        for product in sold_products:
            name, quantity = product[0], product[1]
            # Перевірка, чи є ціна товару
            try:
                price = product[2]
            except IndexError:
                price = None
            if not price:
                # Якщо ціни немає, виводимо запит на введення
                price = tk.simpledialog.askfloat("Введіть ціну", f"Введіть ціну для товару '{name}':")
                if price is None:
                    continue  # Якщо користувач натиснув "Скасувати", перейти до наступного товару
            tree.set(product, "Price", price)
            price_tree.insert("", "end", values=(name, price))
        price_tree.pack()
    else:
        messagebox.showinfo("Інформація", "Немає товарів.")

def show_product_list():
    clear_window()

    tree = ttk.Treeview(root, columns=("name", "quantity", "price", "action"), show="headings")
    tree.heading("name", text="Назва товару")
    tree.heading("quantity", text="Кількість")
    tree.heading("price", text="Ціна")
    tree.grid(row=0, column=0, columnspan=3)

    for product in all_products:
        tree.insert("", "end", values=product)

    label_name = tk.Label(root, text='Назва товару:')
    label_name.grid(row=1, column=0, pady=10)
    entry_name = tk.Entry(root)
    entry_name.grid(row=1, column=1)

    label_quantity = tk.Label(root, text='Кількість:')
    label_quantity.grid(row=2, column=0, pady=10)
    entry_quantity = tk.Entry(root)
    entry_quantity.grid(row=2, column=1)

    button_edit = tk.Button(root, text='Редагувати кількість', command=lambda: edit_product_quantity(tree,
entry_quantity))
    button_edit.grid(row=3, column=0, columnspan=2, pady=10)

    button_delete = tk.Button(root, text='Видалити товар', command=lambda: delete_product_from_db(tree))
    button_delete.grid(row=4, column=0, columnspan=2, pady=10)

    button_back = tk.Button(root, text='Назад', command=show_main_toolbar)
    button_back.grid(row=5, column=0, columnspan=2, pady=10)

    tree.bind("<Double-1>", lambda event: show_product_sale_window(tree))

def check_product_notification():
```

```
    for product in all_products:
        if int(product[2]) <= 3:
            messagebox.showinfo("Увага", f"Кількість товару '{product[1]}' наближається до критичного рівня.")

def delete_product_from_tree(tree, event):
    item_id = tree.identify_row(event.y)
    if item_id:
        values = tree.item(item_id, "values")
        if values and event.x > 610:
            product_id = check_product(values[0])[0]
            delete_product(product_id)
            tree.delete(item_id)
            save_products_to_excel(get_product_list(), "all_products.xlsx")

def add_product_to_db(entry_name, entry_quantity, entry_price):
    name = entry_name.get()
    quantity = entry_quantity.get()
    price = entry_price.get()

    if not name or not quantity or not price:
        messagebox.showerror("Помилка", "Будь ласка, заповніть всі поля.")
        return

    existing_product = check_product(name)
    if existing_product:
        messagebox.showwarning("Увага", f"Товар '{name}' вже існує у базі даних.")
        return

    try:
        add_product(name, quantity, price)
        messagebox.showinfo("Успіх", "Товар успішно додано до бази даних.")
        entry_name.delete(0, tk.END)
        entry_quantity.delete(0, tk.END)
        entry_price.delete(0, tk.END)
        save_products_to_excel(get_product_list(), "all_products.xlsx")
    except Exception as e:
        messagebox.showerror("Помилка", f"Не вдалося додати товар до бази даних: {e}")

def update_product_quantity(entry_name, entry_quantity, entry_price):
    name = entry_name.get()
    quantity = entry_quantity.get()

    if not name or not quantity:
        messagebox.showerror("Помилка", "Будь ласка, заповніть всі поля.")
        return

    existing_product = check_product(name)
    if not existing_product:
        messagebox.showerror("Помилка", "Товар не знайдено в базі даних.")
        return

    try:
        current_quantity = int(existing_product[2])
        new_quantity = int(quantity)
        existing_product[2] = str(new_quantity)
        update_quantity(name, new_quantity)
        messagebox.showinfo("Успіх", "Кількість успішно оновлено!")
        save_products_to_excel(get_product_list(), "all_products.xlsx")
```

```
    except ValueError:
        messagebox.showerror("Помилка", "Введіть коректну кількість товару.")
    except Exception as e:
        messagebox.showerror("Помилка", f"Не вдалося оновити кількість товару: {e}")

    entry_name.delete(0, tk.END)
    entry_quantity.delete(0, tk.END)
    entry_price.delete(0, tk.END)

    show_waiting_list()

def delete_product_from_db(tree):
    selected_item = tree.selection()
    if selected_item:
        product_name = tree.item(selected_item, "values")[0]
        for index, product in enumerate(all_products):
            if product[1] == product_name:
                all_products.pop(index)
                delete_product(product_name)
                break
        tree.delete(selected_item)
        messagebox.showinfo("Успіх", "Товар успішно видалено!")
        save_products_to_excel(all_products, "all_products.xlsx")

def save_order_list_to_excel():
    save_products_to_excel(order_list, "order_list.xlsx")

def show_waiting_list():
    global waiting_list, waiting_list_notification_shown
    # Очистити вікно
    clear_window()

    tree = ttk.Treeview(root, columns=("Name", "Quantity"), show="headings")
    tree.heading("Name", text="Назва")
    tree.heading("Quantity", text="Кількість")

    for product in waiting_list:
        tree.insert("", "end", values=product)

    tree.grid(row=1, column=0, columnspan=3, padx=10, pady=5, sticky="nsew")

    label_name = tk.Label(root, text='Назва товару:')
    label_name.grid(row=2, column=0, pady=10)
    entry_name = tk.Entry(root)
    entry_name.grid(row=2, column=1)

    label_quantity = tk.Label(root, text='Кількість:')
    label_quantity.grid(row=3, column=0, pady=10)
    entry_quantity = tk.Entry(root)
    entry_quantity.grid(row=3, column=1)

    button_add_to_waiting_list = tk.Button(root, text='Додати в список очікування', command=lambda:
add_to_waiting_list(tree, entry_name, entry_quantity))
    button_add_to_waiting_list.grid(row=3, column=2, pady=10)

    button_clear_waiting_list = tk.Button(root, text='Очистити список', command=lambda: clear_waiting_list(tree))
    button_clear_waiting_list.grid(row=4, column=2, pady=10)
```

```
    button_back = tk.Button(root, text='Назад', command=show_main_toolbar)
    button_back.grid(row=5, column=0, columnspan=2, pady=10)

    root.grid_rowconfigure(1, weight=1)
    root.grid_columnconfigure(0, weight=1)

    if not waiting_list_notification_shown:
        check_waiting_list(tree)
        waiting_list_notification_shown = True

def clear_waiting_list(tree):
    global waiting_list
    waiting_list = []
    tree.delete(*tree.get_children())  # Очищаємо вміст дерева
    save_waiting_list_to_excel()  # Зберігаємо зміни у файл

def check_waiting_list(tree):
    for product in waiting_list:
        if int(product[1]) >= 3:
            message = f"Увага!\n\nВисока кількість запитів на товар '{product[0]}'.\n\nКількість: {product[1]} шт.\n\n
Додати до списку замовлень?."
            response = messagebox.askyesno("Увага!", message)
            if response:
                add_product_to_order(product)
                waiting_list.remove(product)
                save_waiting_list_to_excel()
                tree.delete(*tree.get_children())
                for product in waiting_list:
                    tree.insert("", "end", values=product)

def load_order_list_from_excel():
    global order_list
    order_list = load_products_from_excel("order_list.xlsx")

def save_order_list_to_excel():
    save_products_to_excel(order_list, "order_list.xlsx")

def add_product_to_order(product):
    order_list.append(product)
    save_order_list_to_excel()

def clear_order_list(tree):
    global order_list
    order_list = []
    tree.delete(*tree.get_children())
    save_products_to_excel(order_list, "order_list.xlsx")

def calculate_total_order_price():
    total_price = 0
    for item in order_list:
        if len(item) >= 3:
            total_price += int(item[1]) * float(item[2])
    return total_price

def show_order_menu():
    clear_window()

    label_title = tk.Label(root, text='Меню замовлень')
```

```
    label_title.grid(row=0, column=0, columnspan=3, padx=10, pady=5)

    tree = ttk.Treeview(root, columns=("Name", "Quantity", "Price"), show="headings")
    tree.heading("Name", text="Назва")
    tree.heading("Quantity", text="Кількість")
    tree.heading("Price", text="Ціна")

    for product in order_list:
        tree.insert("", "end", values=product)

    tree.grid(row=1, column=0, columnspan=3, padx=10, pady=5, sticky="nsew")

    label_name = tk.Label(root, text='Назва товару:')
    label_name.grid(row=2, column=0, pady=10)
    entry_name = tk.Entry(root)
    entry_name.grid(row=2, column=1)

    label_quantity = tk.Label(root, text='Кількість:')
    label_quantity.grid(row=3, column=0, pady=10)
    entry_quantity = tk.Entry(root)
    entry_quantity.grid(row=3, column=1)

    label_price = tk.Label(root, text='Ціна:')
    label_price.grid(row=4, column=0, pady=10)
    entry_price = tk.Entry(root)
    entry_price.grid(row=4, column=1)

    button_add_to_order_list = tk.Button(root, text='Додати в замовлення', command=lambda: add_to_order_list(tree,
entry_name, entry_quantity, entry_price))
    button_add_to_order_list.grid(row=3, column=2, pady=10)

    button_clear_order_list = tk.Button(root, text='Очистити замовлення', command=lambda: clear_order_list(tree))
    button_clear_order_list.grid(row=4, column=2, pady=10)

    label_total_price = tk.Label(root, text='Загальна вартість замовлення:')
    label_total_price.grid(row=5, column=0, pady=10)
    label_total_price_value = tk.Label(root, text=str(calculate_total_order_price()))
    label_total_price_value.grid(row=5, column=1, pady=10)

    button_back = tk.Button(root, text='Назад', command=show_main_toolbar)
    button_back.grid(row=6, column=0, columnspan=3, pady=10, sticky='w')

    root.grid_rowconfigure(1, weight=1)
    root.grid_columnconfigure(0, weight=1)

def add_to_order_list(tree, entry_name, entry_quantity, entry_price):
    name = entry_name.get()
    quantity = entry_quantity.get()
    price = entry_price.get()

    if not name or not quantity or not price:
        messagebox.showerror("Помилка", "Будь ласка, заповніть всі поля.")
        return

    tree.insert("", "end", values=(name, quantity, price))
    product = (name, quantity, price)  # Зберігаємо товар у вигляді кортежу
    add_product_to_order(product)
```

```
    entry_name.delete(0, tk.END)
    entry_quantity.delete(0, tk.END)
    entry_price.delete(0, tk.END)

    save_order_list_to_excel()
    show_order_list()

def show_order_list():
    label_title = tk.Label(root, text='Список замовлення')
    label_title.grid(row=6, column=0, columnspan=3, padx=10, pady=5)

    tree = ttk.Treeview(root, columns=("Name", "Quantity"), show="headings")
    tree.heading("Name", text="Назва")
    tree.heading("Quantity", text="Кількість")

    for product in order_list:
        tree.insert("", "end", values=product)

    tree.grid(row=7, column=0, columnspan=3, padx=10, pady=5, sticky="nsew")

def choose_order_date():
    top = tk.Toplevel(root)

    cal = tkcalendar.Calendar(top, selectmode='day', date_pattern='yyyy-mm-dd')
    cal.pack(pady=20)

    def on_date_selected():
        chosen_date = cal.get_date()
        top.destroy()
        create_order(chosen_date)

    button_ok = tk.Button(top, text="OK", command=on_date_selected)
    button_ok.pack(pady=20)

def create_order(chosen_date):
    order_file_name = f"order_{chosen_date}.xlsx"
    save_products_to_excel(order_list, order_file_name)
    messagebox.showinfo("Успіх", f"Замовлення на дату {chosen_date} успішно створено та збережено у файлі
{order_file_name}")


def on_closing():
    global root
    if messagebox.askokcancel("Вихід", "Ви хочете закрити програму?"):
        # Зберегти дані перед закриттям
        save_waiting_list_to_excel()
        save_order_list_to_excel()

        # Збереження списку очікування та списку замовлень у файл
        with open("waiting_list.pkl", "wb") as f:
            pickle.dump(waiting_list, f)
        with open("order_list.pkl", "wb") as f:
            pickle.dump(order_list, f)

        # Приховати вікно, не знищуючи
        root.withdraw()

        # Завершити роботу програми
```

```
    sys.exit(0)

# Інтерфейс
root = tk.Tk()
root.title('Система керування запасами')
root.protocol("WM_DELETE_WINDOW", on_closing)

# Показати основний інтерфейс
show_main_toolbar()

root.mainloop()
```

Метод симуляцій

1 прогін

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as st

# Вхідні дані для моделі Уілсона
D = 100  # Обсяг попиту за період
S = 120  # Витрати на розміщення замовлення
H = 2.1  # Витрати на зберігання одиниці товару на складі

# Вхідні дані для моделі з дефіцитом
L = 390  # Втрата від дефіциту

# Вхідні дані для розрахунку рівня сервісу клієнтів
TBO = 7  # Час між замовленнями (у днях)
TC = 2   # Час обробки замовлення (у днях)
ICCp = 2.1 # Вартість утримання одиниці товару на складі (доларів на одиницю)
SL = 0.95 # Рівень сервісу клієнтів (у відсотках)

# Коефіцієнт стандартної нормальної розподілу для рівня сервісу 95%
Z = st.norm.ppf(SL)

# Функція для обчислення розміру замовлення за моделлю Уілсона
def wilson_model(D, S, H):
    Q = np.sqrt((2 * D * S) / H)
    return Q

# Функція для обчислення розміру замовлення за моделлю з дефіцитом
def deficit_model(D, S, H, L):
    Q = np.sqrt((2 * D * S) / H) + np.sqrt((2 * D * L) / H)
    return Q

# Обчислення розміру замовлення для обох моделей
Q_wilson = wilson_model(D, S, H)
Q_deficit = deficit_model(D, S, H, L)

# Обчислення сукупних втрат
def total_losses(Q, D, S, H, L):
    SS = np.sqrt((2 * D * L) / H)  # Розмір безпечного запасу
    TI = (Q / 2) + SS  # Загальний обсяг запасів
```

```
    ICC = TI * H  # Витрати на утримання запасів
    OC = (D / Q) * S  # Витрати на замовлення
    total_cost = ICC + OC + L  # Загальні витрати
    return total_cost

# Обчислення сукупних втрат для кожної моделі
total_losses_wilson = total_losses(Q_wilson, D, S, H, L)
total_losses_deficit = total_losses(Q_deficit, D, S, H, L)

# Візуалізація результатів
models = ['Wilson Model', 'Deficit Model']
sizes = [Q_wilson, Q_deficit]
# Візуалізація результатів сукупних втрат
losses = [total_losses_wilson, total_losses_deficit]

plt.bar(models, sizes, color=['blue', 'green'])
plt.xlabel('Модель')
plt.ylabel('Оптимальний розмір замовлення')
plt.title('Оптимальний розмір замовлення для різних моделей')
plt.show()

plt.bar(models, losses, color=['blue', 'green'])
plt.xlabel('Модель')
plt.ylabel('Сукупні втрати')
plt.title('Сукупні втрати для різних моделей')
plt.show()
```

2 прогін

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as st

# Вхідні дані для моделі Уілсона
D = 107  # Обсяг попиту за період
S = 100  # Витрати на розміщення замовлення
H = 2.1  # Витрати на зберігання одиниці товару на складі

# Вхідні дані для моделі з дефіцитом
L = 350  # Втрата від дефіциту

# Вхідні дані для розрахунку рівня сервісу клієнтів
TBO = 30  # Час між замовленнями (у днях)
TC = 2   # Час обробки замовлення (у днях)
ICCp = 2.1 # Вартість утримання одиниці товару на складі (доларів на одиницю)
SL = 0.96 # Рівень сервісу клієнтів (у відсотках)

# Коефіцієнт стандартної нормальної розподілу для рівня сервісу 95%
Z = st.norm.ppf(SL)

# Функція для обчислення розміру замовлення за моделлю Уілсона
def wilson_model(D, S, H):
    Q = np.sqrt((2 * D * S) / H)
    return Q

# Функція для обчислення розміру замовлення за моделлю з дефіцитом
```

```
def deficit_model(D, S, H, L):
    Q = np.sqrt((2 * D * S) / H) + np.sqrt((2 * D * L) / H)
    return Q

# Обчислення розміру замовлення для обох моделей
Q_wilson = wilson_model(D, S, H)
Q_deficit = deficit_model(D, S, H, L)

# Обчислення сукупних втрат
def total_losses(Q, D, S, H, L):
    SS = np.sqrt((2 * D * L) / H)  # Розмір безпечного запасу
    TI = (Q / 2) + SS  # Загальний обсяг запасів
    ICC = TI * H  # Витрати на утримання запасів
    OC = (D / Q) * S  # Витрати на замовлення
    total_cost = ICC + OC + L  # Загальні витрати
    return total_cost

# Обчислення сукупних втрат для кожної моделі
total_losses_wilson = total_losses(Q_wilson, D, S, H, L)
total_losses_deficit = total_losses(Q_deficit, D, S, H, L)

# Візуалізація результатів
models = ['Wilson Model', 'Deficit Model']
sizes = [Q_wilson, Q_deficit]
# Візуалізація результатів сукупних втрат
losses = [total_losses_wilson, total_losses_deficit]

plt.bar(models, sizes, color=['blue', 'green'])
plt.xlabel('Модель')
plt.ylabel('Оптимальний розмір замовлення')
plt.title('Оптимальний розмір замовлення для різних моделей')
plt.show()

plt.bar(models, losses, color=['blue', 'green'])
plt.xlabel('Модель')
plt.ylabel('Сукупні втрати')
plt.title('Сукупні втрати для різних моделей')
plt.show()
```