Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота бакалавра

на тему : «Розроблення автоматизованої інформаційної системи управління інвентарем на підприємстві»

Виконав: студент групи ІПЗ20-2
Спеціальність 121 «Інженерія програмного забезпечення»
Грибок Олексій Валерійович
(прізвище та ініціали)
Керівник к.ф.-м.н., доц. Рудянова Т.М.
(науковий ступінь, вчене звання, прізвище та ініціали)
Рецензент Університет митної справи та фінансів
(місце роботи)
Доцент кафедри кібербезпеки та інформаційних технологій
(посада)
к.т.н., доцент Прокопович- Ткаченко Д.І.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2024

АНОТАЦІЯ

Грибок О.В. Розроблення автоматизованої інформаційної системи управління інвентарем на підприємстві.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавр за спеціальністю 121 «Інженерія програмного забезпечення». – Університет митної справи та фінансів, Дніпро, 2024.

У сучасному світі, насиченому інноваціями та технологічним прогресом, впровадження автоматизованих інформаційних систем (АІС) стає невід'ємною складовою успішного бізнесу у будь-якій галузі. Однією з важливих сфер, де впровадження таких систем має величезне значення, є управління інвентарем.

Розробка автоматизованої інформаційної системи управління інвентарем на підприємстві є процесом, спрямованим на оптимізацію обліку, контролю та управління запасами та оборотними коштами компанії. Система управління інвентарем відіграє ключову роль у забезпеченні безперервності виробничих та комерційних процесів, мінімізації втрат, оптимізації витрат та підвищенні загальної ефективності підприємства. В сучасному динамічному бізнес-середовищі, де кожна деталь має значення, важливо мати точне уявлення про наявність та рух інвентарю, щоб приймати обґрунтовані рішення, спираючись на актуальні дані.

Автоматизована система управління інвентарем забезпечує можливість в реальному часі відстежувати запаси, контролювати їхній рух, оптимізовувати замовлення та розподіл ресурсів, а також мінімізувати ризики, пов'язані з надлишками або нестачами товарів.

Ключові слова: автоматизована інформаційна система, АІС, управління інвентарем, мова програмування Python, tkinter, ttkthemes, мова SQL, база даних SQLite, модуль sqlite3, datetime, statistics.

ABSTRACT

Hrybok O.V. Development of an automated information system for managing inventory at the enterprise.

Qualification of work to obtain a bachelor's degree in specialty 121 «Software Security Engineering». – University of Customs and Finance, Dnipro, 2024.

In today's world, saturated with innovation and technological progress, the development of automated information systems (AIS) has become an invisible source of successful business for everyone. One of the important areas where the implementation of such systems is of great importance is inventory management.

Development of an automated information system for managing inventory at the enterprise and a process aimed at optimizing the environment, controlling and managing inventories and the company's working capital. The inventory management system plays a key role in ensuring the continuity of production and commercial processes, minimizing costs, optimizing waste and increasing the efficiency of production reception. In today's dynamic business environment, where every detail is important, it is important to accurately indicate the availability of inventory in order to make informed decisions based on current data.

An automated inventory management system ensures the ability to maintain inventory in real time, control inventory flow, optimize the procurement and distribution of resources, and also minimize risks associated with excess or stacks of goods.

Key words: automated information system, AIS, inventory management, Python language programming, tkinter, ttkthemes, SQL language, SQLite database, sqlite3 module, datetime, statistics.

ЗМІСТ

ВСТУП

У сучасний час, коли інновації та технологічний прогрес стають невід'ємною частиною бізнесу, впровадження автоматизованих інформаційних систем (АІС) стає необхідністю для ефективного управління підприємством у будь-якій галузі. Однією з ключових сфер, де впровадження таких систем має величезне значення, є управління інвентарем. Розробка автоматизованої інформаційної системи управління інвентарем на підприємстві – це процес, спрямований на оптимізацію обліку, контролю та управління запасами й оборотними коштами компанії. Система управління інвентарем відіграє ключову роль у забезпеченні безперервності виробничих і комерційних процесів, мінімізації втрат, оптимізації витрат і підвищенні загальної ефективності підприємства.

У сучасному динамічному бізнес-середовищі, де кожна деталь має значення, необхідно мати точне уявлення про наявність і переміщення інвентарю, щоб приймати обґрунтовані рішення, спираючись на актуальні дані.

Автоматизована система управління інвентарем забезпечує можливість у режимі реального часу відстежувати запаси, контролювати їхній рух, оптимізувати замовлення і розподіл ресурсів, а також мінімізувати ризики, пов'язані з надлишками або нестачами товарів.

Ключові переваги розробки та впровадження подібної системи включають в себе підвищення операційної ефективності, скорочення витрат на складські запаси, поліпшення обслуговування клієнтів за рахунок більш точного і швидкого виконання замовлень, а також можливість оперативно реагувати на зміни попиту і ринкової ситуації.

Мета роботи полягає у створенні ефективного інструменту, що дасть змогу оптимізувати облік, контроль та управління запасами товарів і матеріальними ресурсами компанії.

Для досягнення поставленої мети необхідно вирішити наступні основні

завдання:

1) спочатку необхідно провести ретельний аналіз бізнес-процесів і потреб підприємства в управлінні запасами. Це дозволить виявити основні вимоги до функціоналу та можливості системи управління інвентарем;

2) на основі виявлених вимог необхідно розробити архітектуру системи управління інвентарем, визначити основні компоненти та модулі системи, а також спроектувати інтерфейси взаємодії між ними;

3) наступним завданням є вибір найбільш підходящих технологій та інструментів для реалізації системи управління інвентарем, з огляду на особливості підприємства, його інфраструктуру та вимоги до системи;

4) на наступному етапі необхідно розробити програмне забезпечення для системи управління інвентарем, включно зі створенням бази даних, реалізацією функціональності обліку запасів, контролю переміщення товарів, аналітики даних та інших необхідних функцій;

5) після розробки необхідно провести тестування системи управління інвентарем для перевірки її працездатності, надійності та відповідності вимогам. Це включає в себе функціональне тестування, тестування продуктивності, а також тестування на міцність і безпеку.

Об'єктом дослідження є процес управління інвентарем на підприємстві. Цей процес включає в себе облік, контроль і управління запасами товарів і матеріальними ресурсами компанії. У рамках дослідження розглядається весь ланцюжок дій, починаючи від надходження товарів на склад, їхнього зберігання та переміщення, закінчуючи відвантаженням товарів клієнтам або використанням у виробничих процесах.

Предметом дослідження є розробка автоматизованої інформаційної системи управління інвентарем на підприємстві. Ця система представляє собою комплекс програмно-апаратних засобів, які забезпечують автоматизацію процесів обліку, контролю та управління запасами товарів і матеріальними ресурсами компанії.

Методи дослідження в даному випадку можуть включати як теоретичні,

так і практичні підходи, спрямовані на розробку та впровадження автоматизованої інформаційної системи управління інвентарем на підприємстві. Наприклад, дослідження наявних теоретичних підходів, методів і моделей управління інвентарем, а також аналіз актуальної літератури та наукових статей за цією темою, або проведення експертних оцінок за участю фахівців у галузі управління запасами та інформаційних технологій, а також проведення інтерв'ю зі співробітниками підприємства для виявлення їхніх потреб і вимог до системи. Також доцільно розглянути використання методів проектування і моделювання для розроблення архітектури системи управління інвентарем, визначення її функціональності та структури.

Практична значимість роботи з розроблення автоматизованої інформаційної системи управління інвентарем на підприємстві проявляється у наступних механізмах:

1) розробка і впровадження такої системи дозволяє підприємству оптимізувати процеси управління запасами, що призводить до підвищення ефективності бізнесу, зниження операційних витрат і поліпшення фінансових показників;

2) завдяки більш точному обліку і контролю запасів, підприємство може забезпечувати більш швидке і точне виконання замовлень, що призводить до поліпшення рівня обслуговування клієнтів і підвищення їхньої задоволеності;

3) автоматизована система управління інвентарем допомагає оперативно виявляти й усувати потенційні ризики, пов'язані з надлишками або нестачами товарів, а також контролювати їхнє переміщення та використання, що дає змогу мінімізувати втрати та збитки;

4) впровадження сучасних інформаційних технологій, таких як автоматизована система управління інвентарем, допомагає підприємству бути гнучкішим і адаптивнішим до змін ринкового середовища, що сприяє підвищенню його конкурентоспроможності;

5) автоматизація рутинних операцій, пов'язаних з управлінням інвентарем, знижує необхідність вручну виконувати безліч операцій, що

дозволяє працівникам підприємства ефективніше використовувати свій час і ресурси.

Загалом розробка і впровадження автоматизованої інформаційної системи управління інвентарем на підприємстві забезпечує низку практичних вигод, серед яких збільшення ефективності бізнесу, поліпшення обслуговування клієнтів, зниження ризиків і втрат, а також підвищення конкурентоспроможності на ринку.

Кваліфікаційна робота складається зі вступу, 3-х розділів, висновків, використаних джерел з 19 найменувань, 1 додатка.

Обсяг роботи 72 сторінки, містить 41 рисунок.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Аналіз публікацій щодо існуючих автоматизованих інформаційних систем управління інвентарем на підприємстві

Конкретні автоматизовані інформаційні системи (АІС) управління інвентарем на підприємстві можуть варіюватися залежно від галузі, масштабу підприємства та його потреб, але найбільш помітні з них можна представити наступним чином:

1) SAP ERP (Enterprise Resource Planning) [1]. Одна з найпоширеніших і найпотужніших платформ для управління підприємством. У рамках SAP ERP є модуль управління інвентарем, який дозволяє контролювати запаси, замовлення на поставку, облік руху товарів тощо (рис. 1.1);

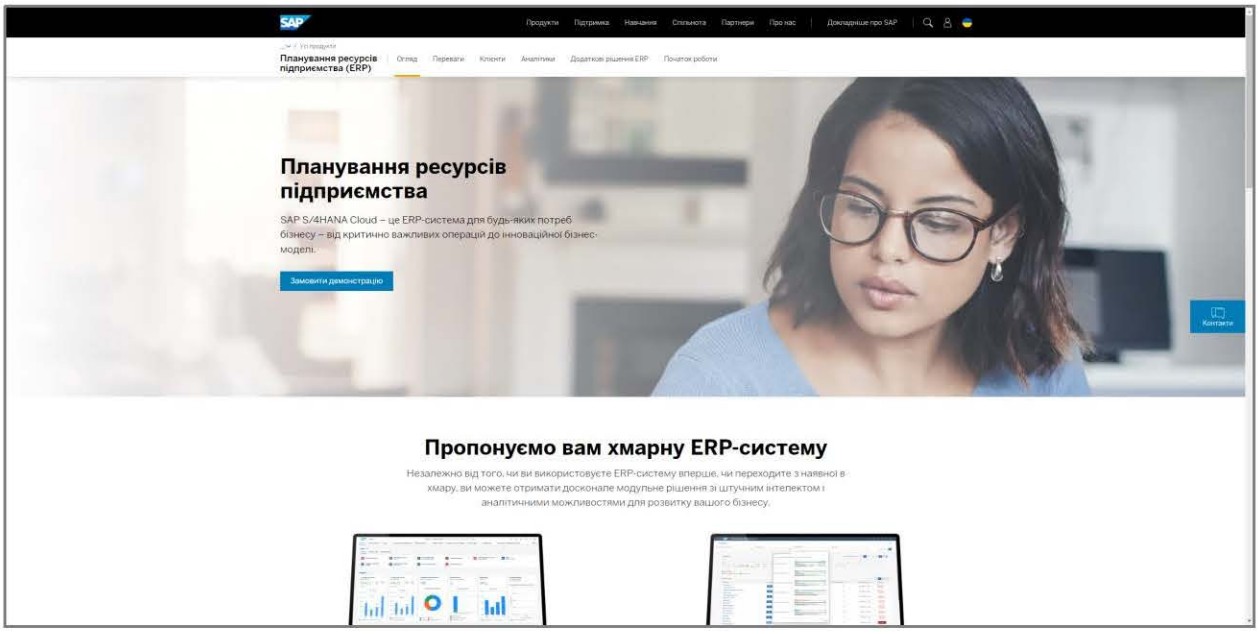

Рисунок 1.1 – Інтерфейс SAP ERP

2) Oracle Inventory Management. Oracle пропонує свою систему управління інвентарем у складі Oracle Supply Chain Management, що дозволяє оптимізувати запаси, управляти замовленнями та забезпечувати ефективне

планування виробництва (рис. 1.2) [2];

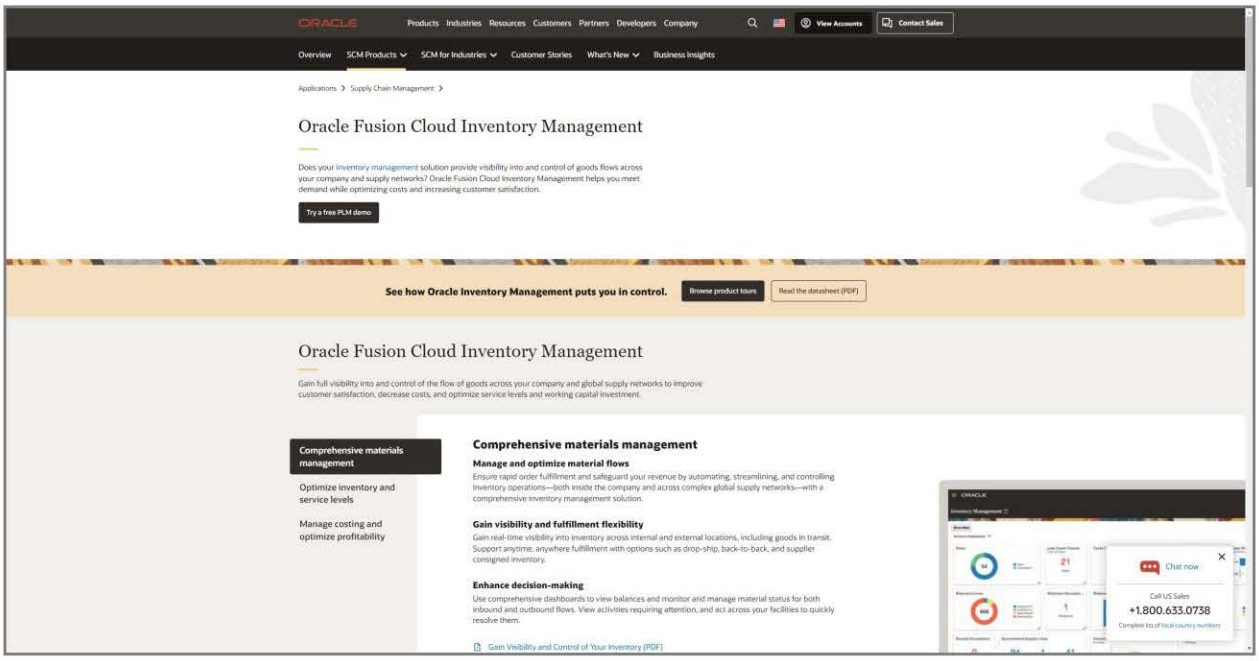

Рисунок 1.2 – Інтерфейс Oracle Inventory Management

3) Microsoft Dynamics 365 for Supply Chain Management. Це хмарне рішення, що пропонує повний цикл управління інвентарем, починаючи від обліку товарів на складі і закінчуючи управлінням поставками і прогнозуванням попиту (рис. 1.3) [3];

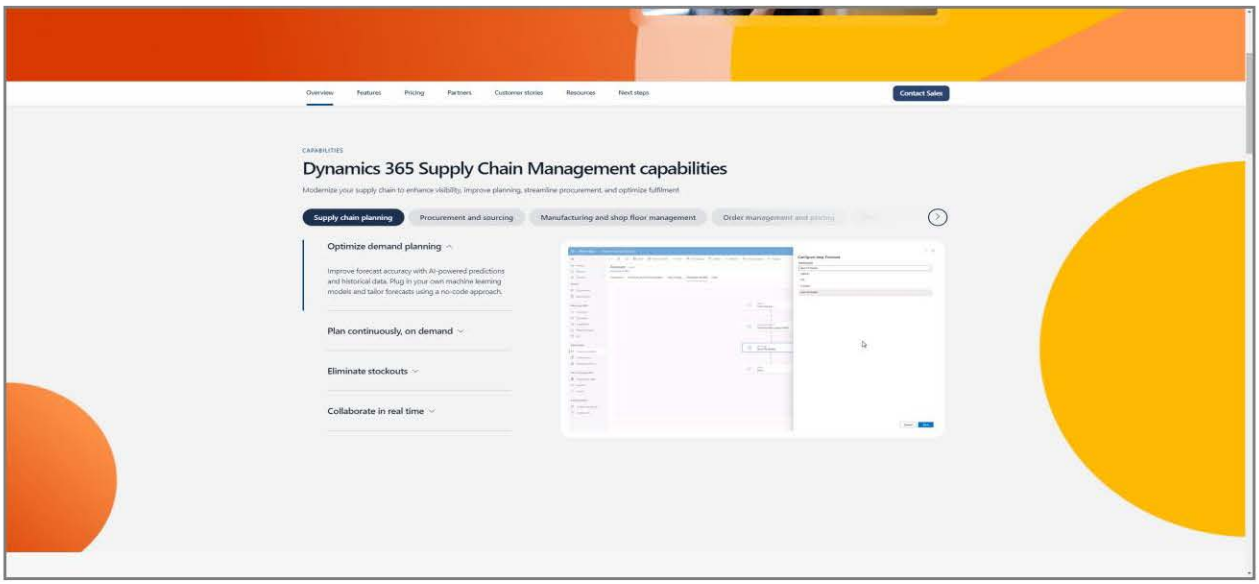

Рисунок 1.3 – Інтерфейс Microsoft Dynamics 365 for Supply Chain Management

4) IBM Sterling Inventory Control Tower. Ця система призначена для моніторингу та управління інвентарем у реальному часі з використанням аналітики даних для ухвалення рішень щодо оптимізації запасів і поставок (рис. 1.4) [4];

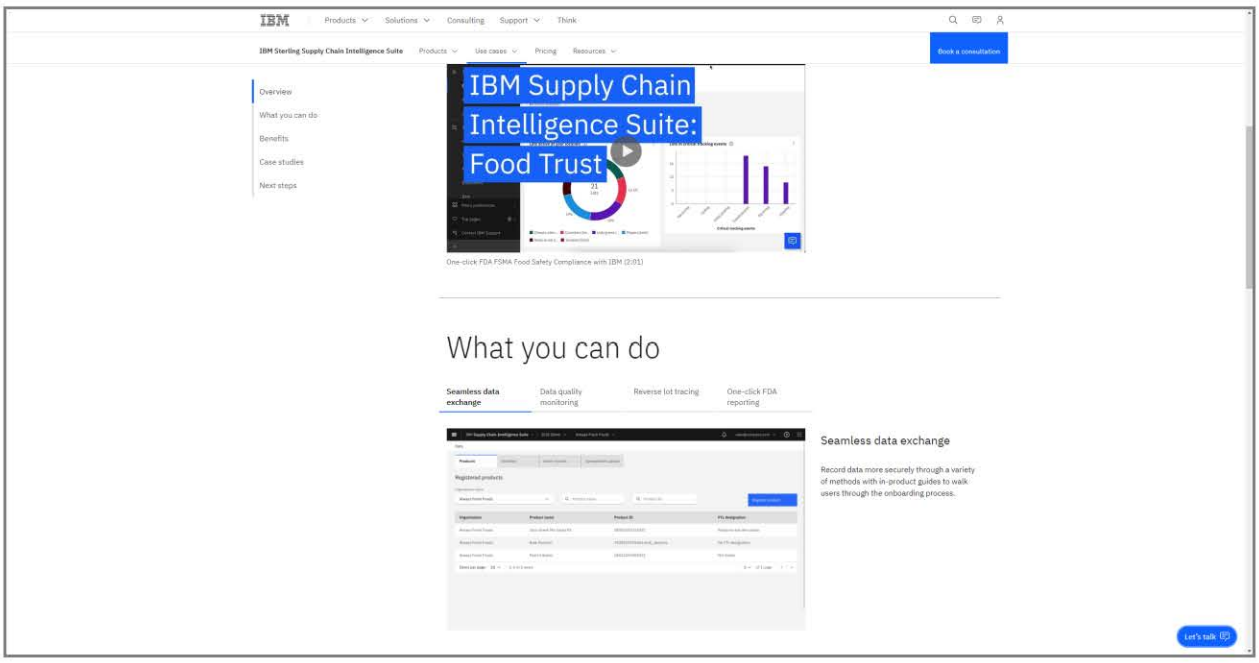

Рисунок 1.4 – Інтерфейс IBM Sterling Inventory Control Tower

5) Zoho Inventory. Якщо підприємство малого або середнього розміру, то система управління інвентарем від Zoho може бути хорошим вибором. Вона пропонує функції обліку запасів, відстеження замовлень та управління поставками (рис. 1.5) [5];

6) Quickbooks Commerce. Хмарна платформа, яка дозволяє керувати запасами, замовленнями, поставками та відстежувати процеси в реальному часі. Вона також інтегрується з різними електронними майданчиками для електронної комерції (рис. 1.6) [6];

7) Fishbowl Inventory. Ця система призначена для малих і середніх підприємств і пропонує широкий спектр функцій, включно з управлінням запасами, відстеженням продажів, виробничим плануванням тощо (рис. 1.7) [7];

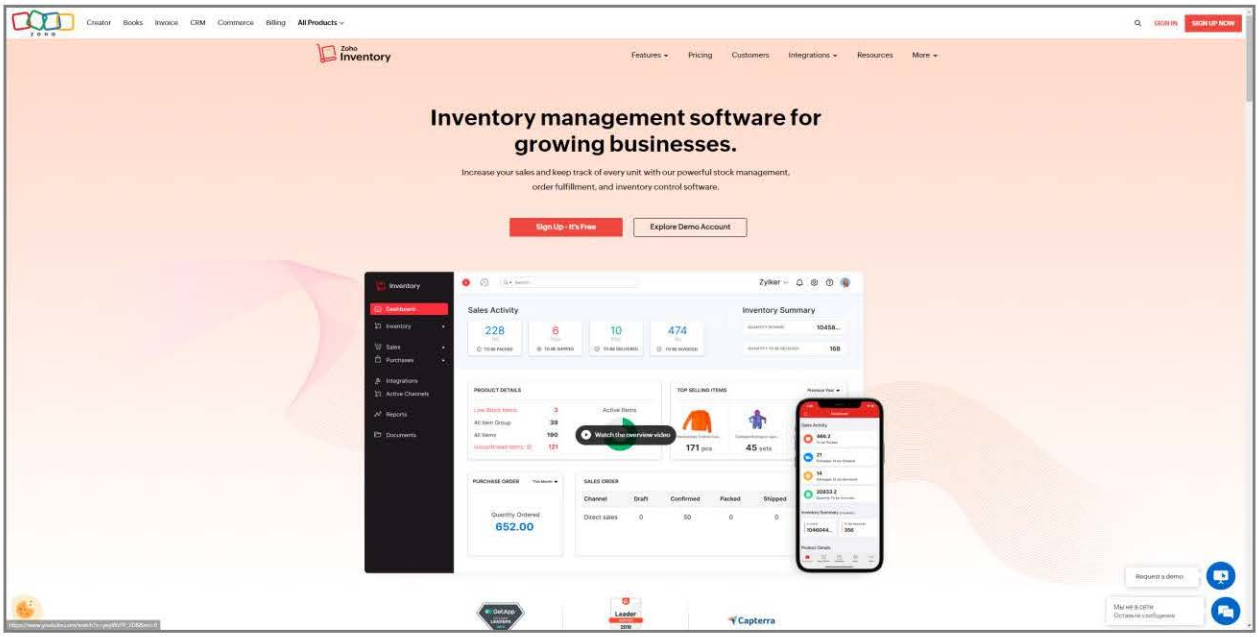

Рисунок 1.5 – Інтерфейс Zoho Inventory

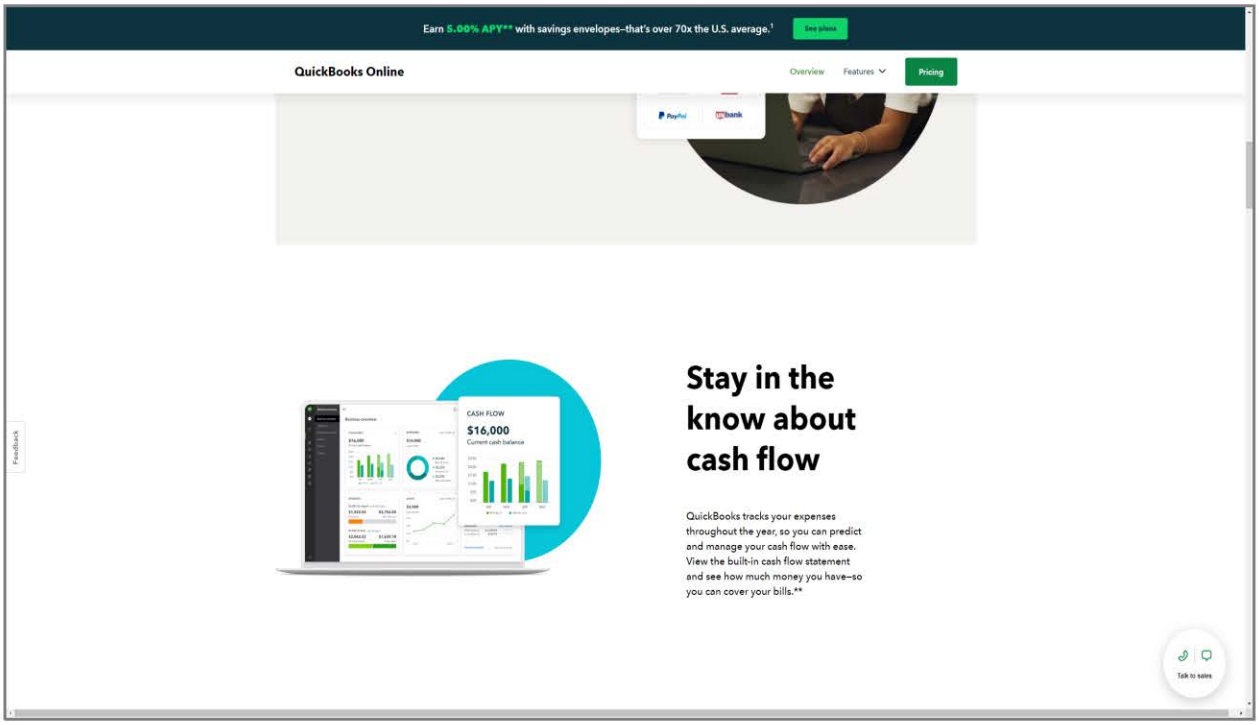

Рисунок 1.6 – Інтерфейс Quickbooks Commerce

8) Tradegecko. Ще одне хмарне рішення, яке допомагає керувати запасами, замовленнями і поставками. Воно також пропонує інструменти для аналізу даних і прогнозування попиту;

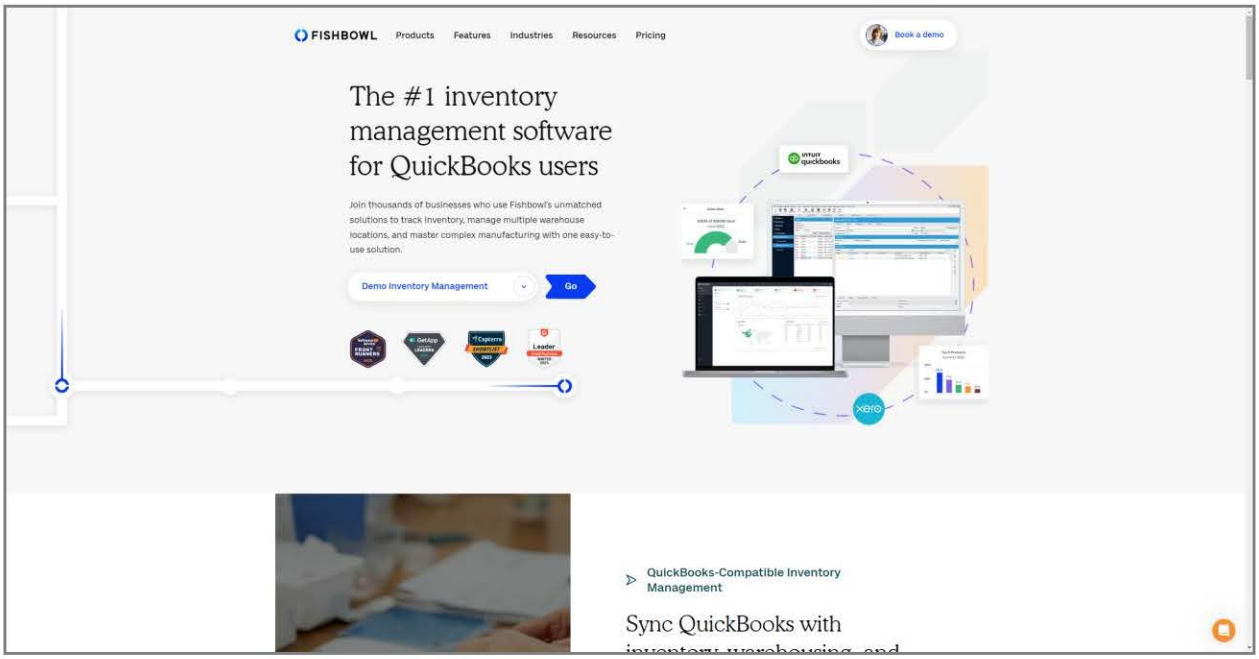

Рисунок 1.7 – Інтерфейс Fishbowl Inventory

9) Wasp Inventory Control. Ця система орієнтована на малий і середній бізнес і надає інструменти для обліку та відстеження запасів, управління замовленнями та інвентаризації (рис. 1.8) [8];

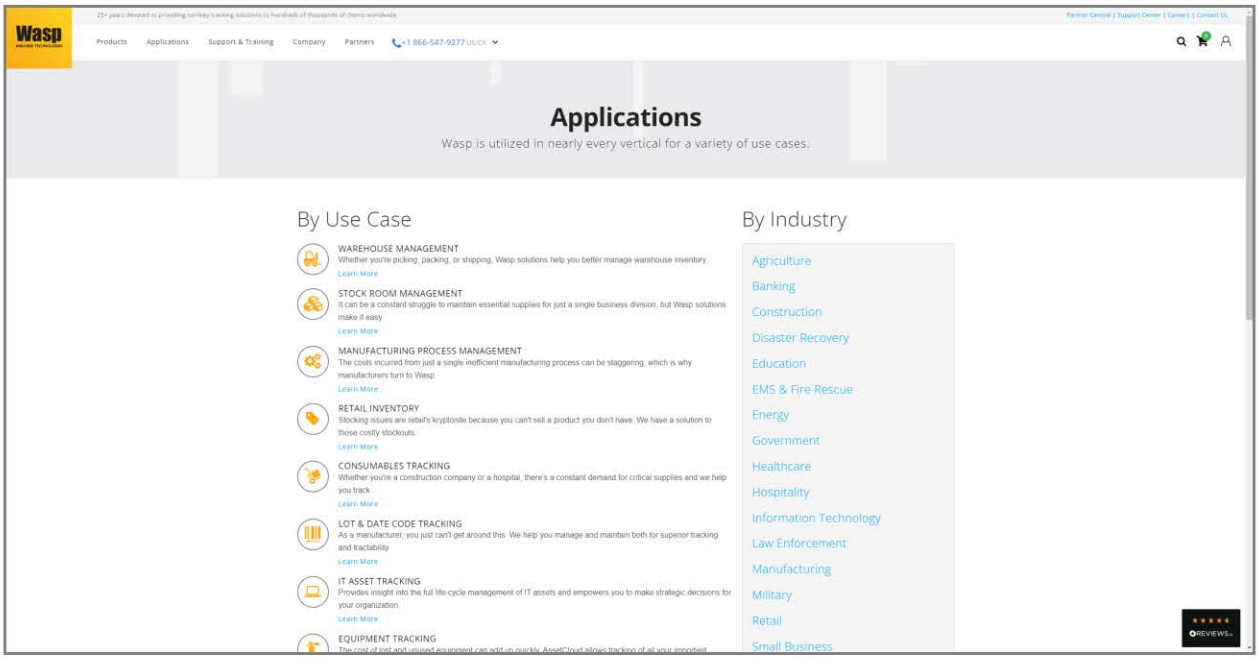

Рисунок 1.8 – Інтерфейс Wasp Inventory Control

10) Inflow Inventory. Це програмне забезпечення для управління

інвентарем, яке включає функції обліку запасів, відстеження серійних номерів і партій, а також управління замовленнями і поставками (рис. 1.9) [9].

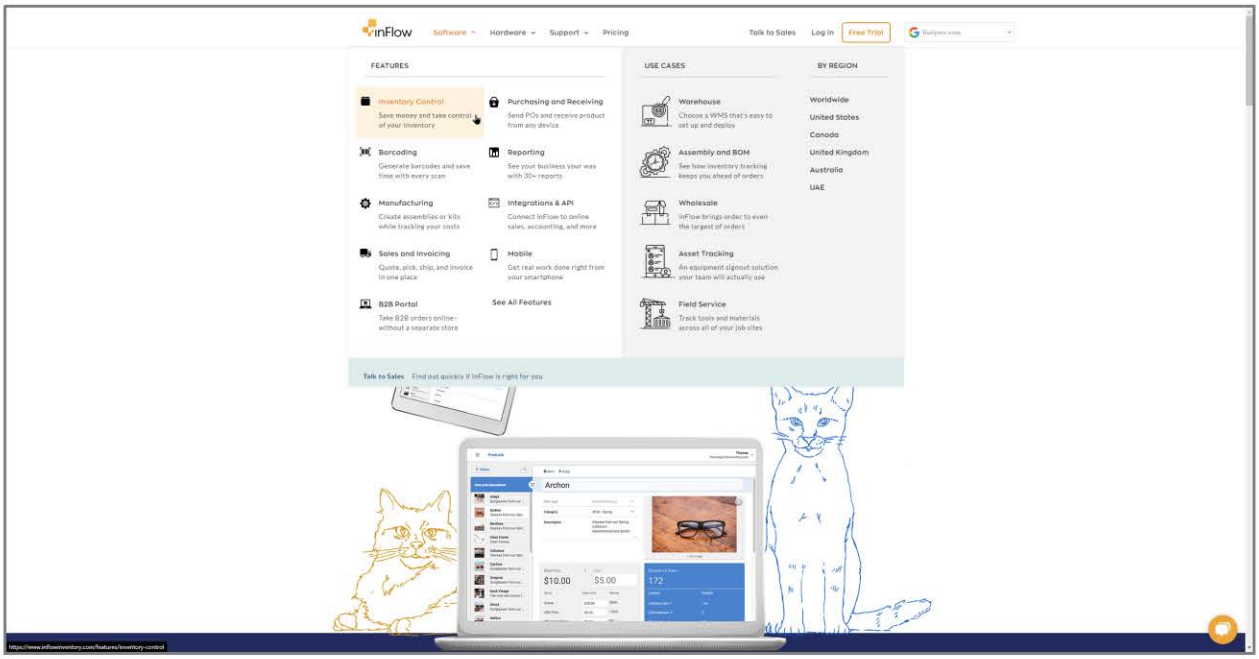

Рисунок 1.9 – Інтерфейс Inflow Inventory

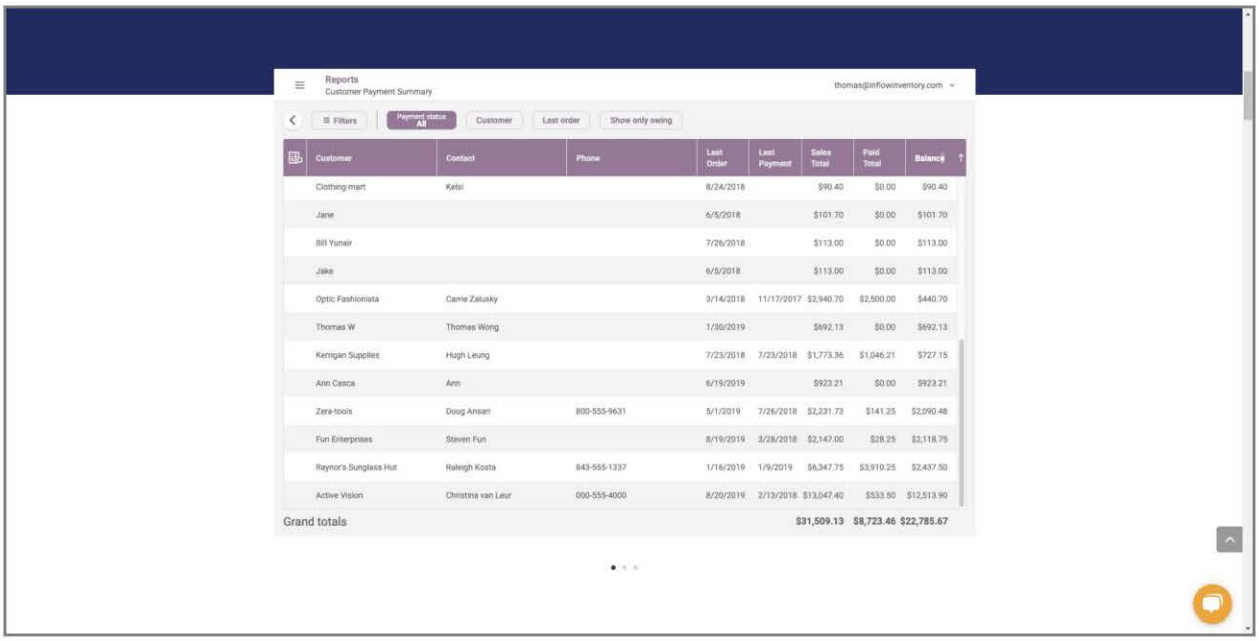

Рисунок 1.10 – Інтерфейс Inflow Inventory

Під час вибору відповідної АІС управління інвентарем важливо враховувати специфіку бізнесу, бюджет, вимоги до функціональності та

зручності використання. Також важливо провести аналіз інтеграційних можливостей з іншими системами, які вже використовуються на підприємстві, щоб забезпечити ефективну взаємодію між різними процесами та підрозділами.

1.2 Аналіз методів розробки АІС

Розробка автоматизованої інформаційної системи (АІС) управління інвентарем на підприємстві – це той процес, який вимагає детального аналізу потреб бізнесу, проєктування ефективної архітектури системи, розроблення функціоналу та його подальшої реалізації, тестування і впровадження.

Для того, аби розробити таку систему детально і якісно, необхідно пройти через кілька етапів, кожен з яких має свої особливості та вимоги. Першим і, мабуть, одним із найважливіших етапів розроблення АІС управління інвентарем є аналіз бізнес-процесів і потреб підприємства. Цей етап дає змогу зрозуміти, які саме функції та можливості має надавати система, щоб найефективніше керувати інвентарем. У рамках аналізу проводиться вивчення поточних процесів управління запасами, їхніх проблем і вузьких місць, а також визначення вимог до нової системи. Наприклад, підприємство може мати певні особливості в обліку запасів (наприклад, облік за серійними номерами або партіями), специфічні вимоги до звітності або інтеграцію з іншими системами.

Наступним етапом є проєктування архітектури системи. На цьому етапі визначаються основні компоненти та модулі системи, їхні взаємозв'язки та взаємодія. Важливо врахувати, що система управління інвентарем може містити в собі не тільки вебінтерфейс для користувачів, а й різноманітні служби інтеграції, автоматизовані процеси, а також базу даних для зберігання інформації про товари, замовлення, постачання тощо. Необхідно також забезпечити масштабованість і гнучкість системи, щоб вона могла адаптуватися до мінливих потреб бізнесу.

Після проектування архітектури слідує етап розробки функціоналу системи. На цьому етапі створюється програмний код, що реалізує певні функції та можливості, необхідні для управління інвентарем. Це може охоплювати розробку користувацького інтерфейсу, бізнес-логіки застосунку, а також інтеграцію із зовнішніми системами і службами. Важливо враховувати вимоги до безпеки та захисту даних під час розроблення функціоналу, особливо якщо система оброблятиме конфіденційну інформацію про товари, клієнтів і замовлення.

Паралельно з розробкою функціоналу необхідно проводити тестування системи. На цьому етапі перевіряється працездатність і коректність роботи всіх компонентів системи, а також її відповідність вимогам і очікуванням користувачів. Тестування може проводитися як автоматизованими засобами, так і вручну, з використанням різних сценаріїв використання системи. Необхідно виявити і виправити всі помилки та недоліки до того, як систему буде впроваджено в реальну експлуатацію.

І останній етап – це впровадження системи та навчання користувачів. На цьому етапі систему розгортають на робочих серверах підприємства, дані переносять зі старих систем, якщо такі є, і користувачі проходять навчання щодо роботи з новою системою.

Важливо надати користувачам достатньо інформації та інструкцій для того, щоб вони могли ефективно використовувати всі можливості системи і досягти максимальної продуктивності.

Таким чином, розробка автоматизованої інформаційної системи управління інвентарем на підприємстві – це процес, що потребує уважного аналізу бізнес-процесів, проєктування ефективної архітектури системи, розробки функціоналу, тестування та впровадження.

Необхідно враховувати потреби та вимоги бізнесу, забезпечувати безпеку та захист даних, а також навчати користувачів для ефективного використання системи.

1.3 Висновки до першого розділу. Постановка завдань дослідження

Мета роботи полягає у створенні ефективного інструменту, що дасть змогу оптимізувати облік, контроль та управління запасами товарів і матеріальними ресурсами компанії.

Для досягнення поставленої мети необхідно вирішити наступні основні завдання:

1) спочатку необхідно провести ретельний аналіз бізнес-процесів і потреб підприємства в управлінні запасами. Це дозволить виявити основні вимоги до функціоналу та можливості системи управління інвентарем;

2) на основі виявлених вимог необхідно розробити архітектуру системи управління інвентарем, визначити основні компоненти та модулі системи, а також спроектувати інтерфейси взаємодії між ними;

3) наступним завданням є вибір найбільш підходящих технологій та інструментів для реалізації системи управління інвентарем, з огляду на особливості підприємства, його інфраструктуру та вимоги до системи;

4) на наступному етапі необхідно розробити програмне забезпечення для системи управління інвентарем, включно зі створенням бази даних, реалізацією функціональності обліку запасів, контролю переміщення товарів, аналітики даних та інших необхідних функцій;

5) після розробки необхідно провести тестування системи управління інвентарем для перевірки її працездатності, надійності та відповідності вимогам. Це включає в себе функціональне тестування, тестування продуктивності, а також тестування на міцність і безпеку.

Методи дослідження в даному випадку можуть включати як теоретичні, так і практичні підходи, спрямовані на розробку та впровадження автоматизованої інформаційної системи управління інвентарем на підприємстві.

Аналіз публікацій щодо існуючих автоматизованих інформаційних систем управління інвентарем на підприємстві дозволяє отримати огляд

існуючих рішень і технологій у цій галузі. Цей аналіз виявляє основні тренди, проблеми та виклики, з якими стикаються підприємства при впровадженні таких систем.

Аналіз методів розробки автоматизованих інформаційних систем надає інформацію про різні підходи до створення систем управління інвентарем. Це включає в себе розгляд методологій розроблення, використовуваних технологій, інструментів і технік, а також приклади найкращих практик і рекомендацій.

Висновки до розділу підкреслюють ключові аспекти дослідження, узагальнюють основні результати аналізу публікацій і методів розроблення, а також формулюють постановку завдань і цілі подальшого дослідження.

Все вищеперераховане в комплексі дозволяє зрозуміти, які аспекти будуть розглянуті далі в роботі та які питання будуть вирішуватися.

Таким чином, поточний розділ є важливим етапом дослідження, який забезпечує фундаментальну базу знань для подальшого аналізу та розроблення автоматизованої інформаційної системи управління інвентарем на підприємстві.

РОЗДІЛ 2 АНАЛІЗ ЗАСОБІВ РЕАЛІЗАЦІЇ АВТОМАТИЗОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ІНТВЕНТАРЕМ НА ПІДПРИЄМСТВІ

2.1 Вибір програмних засобів для реалізації проекту

Для реалізації проєкту доцільно використати наступні програмні засоби та технології:

1) мова програмування Python, яка використовується для розробки програми. Python є дуже популярною та потужною мовою програмування з багатьма бібліотеками та інструментами для розробки різних програм;

2) Tkinter є стандартною бібліотекою для створення графічних інтерфейсів користувача (GUI) у Python. Вона включає в себе набір інструментів для створення вікон, кнопок, полів вводу та інших елементів інтерфейсу;

3) база даних SQLite – це вбудована СУБД, яка використовується для зберігання даних програми;

4) Matplotlib – це бібліотека для візуалізації даних у Python;

5) ttkthemes – це бібліотека, яка надає різноманітні теми для віджетів Tkinter;

6) datetime – це модуль Python, який дозволяє працювати з датами та часом.

Усі перелічені програмні засоби та технології дозволять створити функціональну та зручну у використанні програму для управління запасами.

2.2 Мова програмування Python

Мова програмування Python є універсальною мовою загального призначення, яка швидко завоювала популярність завдяки своїй простоті, універсальності та читабельності коду. Розроблений у 1990-х роках Гвідо ван

Россумом, Python сьогодні використовується в найрізноманітніших галузях, від веброзробки та машинного навчання до наукових досліджень і автоматизації завдань [10].

Python відомий своїм лаконічним синтаксисом, що нагадує природну мову. Він підходить для широкого спектра завдань, включно з:

1) веброзробкою. Django і Flask – популярні фреймворки Python для створення вебдодатків і сайтів;

2) аналізом даних. NumPy, Pandas і SciPy – бібліотеки Python, що широко використовуються для обробки та аналізу даних;

3) машинним навчанням. TensorFlow і PyTorch – бібліотеки Python, які є лідерами в галузі машинного навчання та штучного інтелекту;

4) системним адмініструванням. Ansible і Fabric – інструменти Python для автоматизації завдань адміністрування систем;

5) розробкою ігор. PyGame – бібліотека Python для створення 2D-ігор.

Python-код, як правило, легко читається і зрозумілий іншим розробникам. Це робить його ідеальним вибором для спільних проектів і полегшує підтримку коду в майбутньому.

Python має багату стандартну бібліотеку, яка містить модулі для роботи з файлами, мережею, датами, часом, регулярними виразами тощо. Існує величезна кількість сторонніх бібліотек Python для найрізноманітніших завдань, від вебскрапінгу та обробки природної мови до створення графіків і візуалізації даних. Також Python працює на різних операційних системах, включно з Windows, macOS і Linux [10].

Синтаксис Python простий і лаконічний, що робить його доступним для початківців. Замість фігурних дужок Python використовує відступи для визначення блоків коду. Це робить код більш читабельним і зрозумілим.

На рисунку 2.1 наведений синтаксис мови програмування Python, на висвітлено особливості його використання.

```python
import tkinter as tk
from tkinter import messagebox, simpledialog, filedialog, ttk
import sqlite3
from datetime import datetime
import csv
import statistics
import matplotlib.pyplot as plt
from ttkthemes import ThemedStyle


class InventoryApp:
    def __init__(self, master):
        self.master = master
        self.master.title("Система управління запасами")
        self.master.geometry("1570x600")  # Встановлення розміру вікна

        self.db_connection = sqlite3.connect('inventory.db')
        self.create_tables()

        self.style = ThemedStyle(self.master)
        self.style.theme_use("arc")

        self.create_widgets()


    def create_tables(self):
        cursor = self.db_connection.cursor()
        cursor.execute('''CREATE TABLE IF NOT EXISTS categories
                          (id INTEGER PRIMARY KEY, name TEXT)''')
        cursor.execute('''CREATE TABLE IF NOT EXISTS inventory
                          (id INTEGER PRIMARY KEY, item TEXT, category_id INTEGER, quantity INTEGER, last_updated TEXT,
                          FOREIGN KEY(category_id) REFERENCES categories(id))''')
        self.db_connection.commit()


    def create_widgets(self):
        self.menu_bar = tk.Menu(self.master)

        self.categories_menu = tk.Menu(self.menu_bar, tearoff=0)
        self.categories_menu.add_command(label="Переглянути категорії", command=self.view_categories)
        self.categories_menu.add_command(label="Додати категорію", command=self.add_category)
        self.categories_menu.add_command(label="Оновити категорію", command=self.update_category)
        self.categories_menu.add_command(label="Видалити категорію", command=self.delete_category)
        self.menu_bar.add_cascade(label="Категорії", menu=self.categories_menu)

        self.master.config(menu=self.menu_bar)

        self.item_label = tk.Label(self.master, text="Товар:")
        self.item_label.grid(row=0, column=0, padx=10, pady=5, sticky="E")

        self.item_entry = ttk.Entry(self.master)
        self.item_entry.grid(row=0, column=1, padx=10, pady=5)

        self.category_label = tk.Label(self.master, text="Категорія:")
        self.category_label.grid(row=0, column=2, padx=10, pady=5, sticky="E")
```

Рисунок 2.1 – Синтаксис мови Python

Python підтримує широкий спектр операторів, включно з арифметичними, порівняльними, логічними та операторами присвоювання. Змінні не потрібно оголошувати заздалегідь. Тип змінної визначається автоматично за значенням, яке їй присвоюється. Python також підтримує різні структури даних, такі як списки, кортежі, словники та множини. Ще мова підтримує різні керуючі оператори, такі як оператори if, else, for і while.

Функції – це блоки коду, які виконують певне завдання. Функції можуть приймати аргументи та повертати значення. Слід зауважити, що Python – це об'єктно-орієнтована мова програмування, тобто вона підтримує створення класів та об'єктів. Класи – це шаблони для створення об'єктів, які мають свої власні атрибути та методи.

Python використовується в різних наукових галузях, таких як фізика, хімія, біологія та інженерія. Ще він може використовуватись для навчання програмування в школах та університетах.

В цілому Python – одна з найпростіших мов програмування для вивчення, що робить її ідеальним вибором.

2.3 Бібліотека Tkinter

Python Tkinter – це бібліотека, яка надає інструменти для створення графічного інтерфейсу користувача (GUI) у додатках Python.

Tkinter є стандартним набором інструментів для створення GUI в Python і ґрунтується на бібліотеці Tk, яка надає графічні елементи та методи їхнього керування. Використовуючи Tkinter, існує можливість створювати різноманітні інтерфейси, включаючи вікна, кнопки, текстові поля, фрейми і багато іншого [12, 13].

Основним компонентом у Tkinter є віджет (widget). Віджети представляють собою різні елементи керування, такі як кнопки, текстові поля, мітки тощо. Ці віджети можуть бути організовані на графічному вікні в різних конфігураціях з використанням менеджера геометрії Tkinter, такого як grid, pack або place. Одним із ключових понять у Tkinter є вікно (window). Вікно представляє собою контейнер, у якому розміщуються всі віджети. Вікно може бути головним (root window) або дочірнім. Головне вікно є основним вікном програми, а дочірні вікна можуть бути створені для відображення додаткових діалогів або підпорядкованих інтерфейсів [14].

Tkinter надає безліч різних віджетів для створення інтерактивних

додатків. Деякі з найбільш поширених віджетів включають в себе [12-14]:

1) button (Кнопка). Представляє собою віджет, який дозволяє користувачеві виконати певну дію під час натискання. Кнопки часто використовуються для запуску функцій або команд;

2) label (Мітка). Представляє собою текстовий елемент, який використовується для відображення статичного тексту або зображень на графічному інтерфейсі;

3) entry (Текстове поле). Представляє собою віджет, який дозволяє користувачеві вводити текст із клавіатури. Воно може бути використано для отримання інформації від користувача;

4) frame (Фрейм). Є контейнером для інших віджетів. Він використовується для організації та групування віджетів разом;

5) canvas (Полотно). Представляє собою область, на якій можна малювати графічні об'єкти, такі як лінії, кола, прямокутники тощо;

6) listbox (Список). Представляє собою віджет, який дозволяє користувачеві обирати один або кілька елементів зі списку;

7) menu (Меню). Представляє собою віджет, який надає навігаційні опції для програми. Меню можуть містити підменю, команди та різні віджети.

Приклад створення порожнього вікна Tkinter наведений на рисунку 2.2.

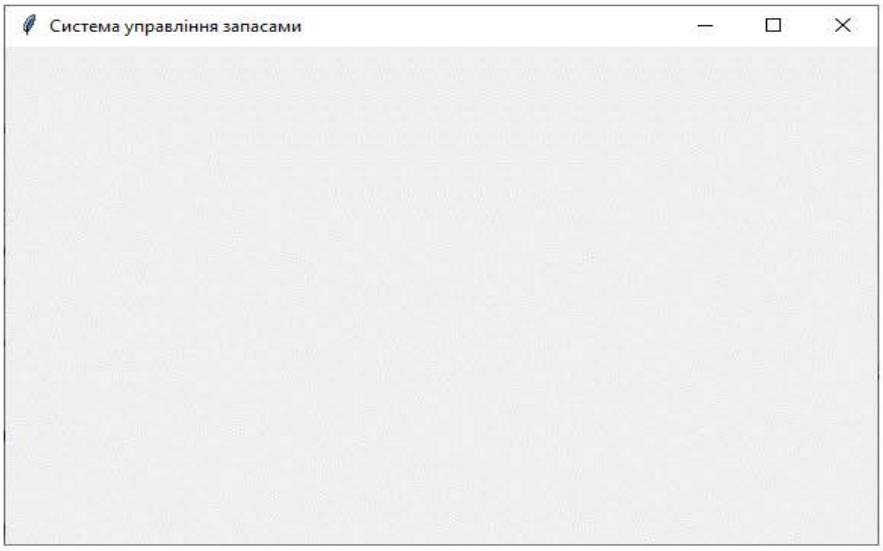

Рисунок 2.2 – Порожнє вікно Tkinter

На рисунку 2.3 міститься програмний код, який демонструє приклад ініціалізації простого застосунку з підтримкою графічного інтерфейсу на Tkinter. Рисунок 2.4 наводить приклад розміщення віджетів на вікно.

```
import tkinter as tk
from tkinter import messagebox, simpledialog, filedialog, ttk
import sqlite3
from datetime import datetime
import csv
import statistics
import matplotlib.pyplot as plt
from ttkthemes import ThemedStyle


class InventoryApp:
    def __init__(self, master):
        self.master = master
        self.master.title("Система управління запасами")
        self.master.geometry("1570x600")  # Встановлення розміру вікна

        self.db_connection = sqlite3.connect('inventory.db')
        self.create_tables()

        self.style = ThemedStyle(self.master)
        self.style.theme_use("arc")

        self.create_widgets()

```

Рисунок 2.3 – Ініціалізація порожнього вікна Tkinter

Tkinter також надає можливості для оформлення та налаштування віджетів, такі як зміна кольору, розміру та шрифту тексту, додавання зображень і багато іншого [12, 13].

Одним зі способів організації віджетів на графічному інтерфейсі є використання менеджерів геометрії. Tkinter надає три основні менеджери геометрії:

– grid;

– pack;

– place.

```
def create_widgets(self):
    self.menu_bar = tk.Menu(self.master)

    self.categories_menu = tk.Menu(self.menu_bar, tearoff=0)
    self.categories_menu.add_command(label="Переглянути категорії", command=self.view_categories)
    self.categories_menu.add_command(label="Додати категорію", command=self.add_category)
    self.categories_menu.add_command(label="Оновити категорію", command=self.update_category)
    self.categories_menu.add_command(label="Видалити категорію", command=self.delete_category)
    self.menu_bar.add_cascade(label="Категорії", menu=self.categories_menu)

    self.master.config(menu=self.menu_bar)

    self.item_label = tk.Label(self.master, text="Товар:")
    self.item_label.grid(row=0, column=0, padx=10, pady=5, sticky="E")

    self.item_entry = ttk.Entry(self.master)
    self.item_entry.grid(row=0, column=1, padx=10, pady=5)

    self.category_label = tk.Label(self.master, text="Категорія:")
    self.category_label.grid(row=0, column=2, padx=10, pady=5, sticky="E")

    self.category_combobox = ttk.Combobox(self.master, state="readonly")
    self.category_combobox.grid(row=0, column=3, padx=10, pady=5)

    self.load_categories()

    self.quantity_label = tk.Label(self.master, text="Кількість:")
    self.quantity_label.grid(row=0, column=4, padx=10, pady=5, sticky="E")

    self.quantity_entry = ttk.Entry(self.master)
    self.quantity_entry.grid(row=0, column=5, padx=10, pady=5)

    self.add_button = ttk.Button(self.master, text="Додати товар", command=self.add_item)
    self.add_button.grid(row=0, column=6, padx=10, pady=5, sticky="WE")

    self.update_button = ttk.Button(self.master, text="Оновити кількість", command=self.update_quantity)
    self.update_button.grid(row=0, column=7, padx=10, pady=5, sticky="WE")

    self.remove_button = ttk.Button(self.master, text="Видалити товар", command=self.remove_item)
    self.remove_button.grid(row=0, column=8, padx=10, pady=5, sticky="WE")

    self.search_label = tk.Label(self.master, text="Пошук:")
    self.search_label.grid(row=1, column=0, padx=10, pady=5, sticky="E")
```

Рисунок 2.4 – Приклад розміщення віджетів на вікно

Менеджер grid вирівнює віджети у вигляді сітки, менеджер pack упаковує віджети в батьківський контейнер, а менеджер place розміщує віджети на заданих координатах.

Подієва модель Tkinter дає змогу додаткам реагувати на дії користувача, такі як натискання клавіш, кліки миші та інші події.

Для опрацювання подій у Tkinter використовуються обробники подій (event handlers), що зв’язуються з певними подіями та виконують певні дії в разі їх виникнення. Tkinter також підтримує можливість створення багатопотокових додатків, що дозволяє виконувати тривалі операції у фоновому режимі, не блокуючи користувацький інтерфейс.

Завдяки своїй простоті та гнучкості, Tkinter широко використовується для створення різних типів додатків, включаючи ігри, утиліти, освітні програми та багато іншого [12-14].

Tkinter також є платформонезалежним, тобто додатки, створені з його використанням, працюватимуть на різних операційних системах, включно з Windows, macOS і Linux.

2.4 Мова запитів до бази даних SQL

SQL (Structured Query Language) – це спеціалізована мова програмування, яка використовується для роботи з реляційними базами даних.

Вона надає можливість створення, зміни, управління та вилучення даних з баз даних, а також виконання різних операцій, таких як створення таблиць, вставка, оновлення та видалення записів, агрегація даних та багато іншого.

В основі SQL лежить концепція реляційної моделі даних, розроблена Едгаром Коддом у 1970-х роках [15]. Реляційна модель даних подає дані у вигляді таблиць, що складаються з рядків (записів) та стовпців (полів). Кожна таблиця представляє окрему сутність, а кожен рядок таблиці представляє конкретний екземпляр цієї сутності. Одним із ключових елементів SQL є мова запитів (query language), яка дозволяє користувачеві виконувати операції з даними в базі даних.

Мова запитів SQL складається з різних типів операторів та виразів, які дозволяють формувати запити до бази даних та отримувати необхідну інформацію.

Основні типи операторів SQL включають [15]:

1) оператор SELECT використовується для вилучення даних з таблиці або кількох таблиць бази даних. Він дозволяє вибрати певні стовпці для виведення, фільтрувати дані з використанням умов WHERE, сортувати дані за допомогою ORDER BY та об'єднувати дані з кількох таблиць за допомогою оператора JOIN;

2) оператор INSERT дозволяє додати нові записи до таблиці бази даних. Він дозволяє вказати значення для кожного стовпця запису, що вставляється, або вставити дані з іншої таблиці або запиту;

3) оператор UPDATE використовується для оновлення існуючих записів у таблиці. Він дозволяє змінити значення стовпців для вибраних рядків за допомогою умови WHERE;

4) оператор DELETE використовується для видалення одного або декількох записів з таблиці бази даних. Він дозволяє видалити всі рядки з таблиці або лише ті, що відповідають певним умовам;

5) оператор CREATE TABLE використовується для створення нової таблиці у базі даних. Він визначає структуру таблиці, включаючи назви стовпців, їх типи даних та будь-які обмеження цілісності даних;

6) оператор ALTER TABLE використовується зміни структури існуючої таблиці. Він дозволяє додавати нові стовпці, змінювати типи даних стовпців, видаляти стовпці тощо;

7) оператор DROP TABLE використовується для видалення таблиці з бази даних. Він видаляє всі дані та структуру таблиці, тому його слід використовувати з обережністю.

Мова SQL також надає можливості для виконання різних операцій над даними, таких як агрегація, угруповання, об'єднання та сортування.

Оператори агрегації, такі як SUM, AVG, MIN та MAX, дозволяють обчислювати суму, середнє значення, мінімальне та максимальне значення стовпця даних.

Оператори GROUP BY та HAVING використовуються для групування даних та застосування умов до груп.

Оператори JOIN дозволяють об'єднувати дані з кількох таблиць за певними умовами.

Завдяки своїй сучасності та гнучкості, SQL широко використовується в різних галузях інформаційних технологій, таких як веброзробка, аналітика даних, бізнес-аналіз, фінанси, медицина та багато іншого. Він є стандартною

мовою запитів для роботи з реляційними базами даних та є необхідним інструментом для розробників, адміністраторів баз даних та аналітиків даних.

2.5 SQLite та sqlite3 в Python

SQLite – це система управління реляційними базами даних (СУБД), що вбудовується, яка є відкритою і безкоштовною. Вона призначена для роботи з невеликими та середніми базами даних, виключно легка у використанні та не потребує окремого сервера для своєї роботи [15, 16].

На рисунках 2.5 та 2.6 наведений синтаксис модуля sqlite3 у Python для взаємодії з СУБД SQLite.

```
def create_tables(self):
    cursor = self.db_connection.cursor()
    cursor.execute('''CREATE TABLE IF NOT EXISTS categories
                      (id INTEGER PRIMARY KEY, name TEXT)''')
    cursor.execute('''CREATE TABLE IF NOT EXISTS inventory
                      (id INTEGER PRIMARY KEY, item TEXT, category_id INTEGER, quantity INTEGER, last_updated TEXT,
                      FOREIGN KEY(category_id) REFERENCES categories(id))''')
    self.db_connection.commit()
```

Рисунок 2.5 – Синтаксис sqlite3 для створення таблиць

```
def delete_category(self):
    cursor = self.db_connection.cursor()
    cursor.execute("SELECT name FROM categories")
    categories = [row[0] for row in cursor.fetchall()]
    selected_category = simpledialog.askstring("Видалити категорію", "Виберіть категорію для видалення:", initialvalue=categories[0])

    if selected_category:
        confirm = messagebox.askyesno("Підтвердження", f"Ви впевнені, що хочете видалити категорію '{selected_category}'?")

        if confirm:
            cursor.execute("DELETE FROM categories WHERE name = ?", (selected_category,))
            cursor.execute("DELETE FROM inventory WHERE category_id = (SELECT id FROM categories WHERE name = ?)", (selected_category,))
            self.db_connection.commit()
            self.load_categories()
            self.load_inventory()


def add_item(self):
    item = self.item_entry.get()
    category = self.category_combobox.get()
    quantity = self.quantity_entry.get()

    if item and category and quantity:
        try:
            quantity = int(quantity)
            now = datetime.now()
            formatted_date = now.strftime("%Y-%m-%d %H:%M:%S")
            cursor = self.db_connection.cursor()
            cursor.execute("INSERT INTO inventory (item, category_id, quantity, last_updated) VALUES (?, (SELECT id FROM categories WHERE name = ?), ?, ?)", (item, category, quantity, formatted_date))
            self.db_connection.commit()
            self.load_inventory()
            self.item_entry.delete(0, tk.END)
            self.quantity_entry.delete(0, tk.END)
        except ValueError:
            messagebox.showerror("Помилка", "Будь ласка, введіть дійсну кількість.")
    else:
        messagebox.showerror("Помилка", "Будь ласка, заповніть всі поля.")
```

Рисунок 2.6 – Синтаксис sqlite3 для видалення категорії товарів та додавання нових товарів

SQLite зберігає всю базу даних (структури, таблиці, індекси та дані) в одному файлі на диску, що робить її зручною для впровадження у додатки та управління даними локально. Однією з ключових переваг SQLite є його простота у використанні та підтримка багатьох операційних систем, включаючи Windows, macOS, Linux. Вона надає всі основні функції реляційних баз даних, такі як підтримка SQL, транзакції, індекси, обмеження цілісності даних та багато іншого, при цьому залишаючись легковагою та швидкою [16].

У Python для роботи з базами даних SQLite використовується модуль sqlite3. Цей модуль надає API для взаємодії з базами даних SQLite із Python, дозволяючи виконувати запити, отримувати результати та керувати транзакціями. Загалом же СУБД SQLite вбудована у стандартну бібліотеку Python, що робить її доступною без необхідності встановлення додаткових пакетів.

Модуль sqlite3 дозволяє створювати та керувати базами даних SQLite, виконувати SQL-запити, отримувати результати запитів у вигляді об'єктів Python, керувати транзакціями, створювати та змінювати таблиці, індекси та інші об'єкти бази даних. Він забезпечує зручний та гнучкий спосіб роботи з даними у додатках Python.

Однією з особливостей sqlite3 є його інтеграція з мовою Python. Він дозволяє використовувати параметризовані запити для захисту від SQL-ін'єкцій, а також надає можливість роботи з даними у форматі Python, що спрощує та прискорює розробку програм.

Для початку роботи з SQLite у Python спочатку необхідно підключитися до бази даних. Це робиться за допомогою функції connect(), яка приймає ім'я файлу бази даних як аргумент. Якщо файл не існує, його буде створено автоматично. Після підключення до бази даних можна створювати таблиці, виконувати SQL-запити та керувати даними [16].

Для виконання SQL-запитів у SQLite з використанням sqlite3 в Python

використовується метод execute(). Цей метод приймає рядок SQL-запиту як аргумент та виконує його на базі даних. Результати запиту можна отримати за допомогою методу fetchone() для отримання одного рядка результату або fetchall() для всіх рядків результату. Однією з особливостей SQLite є підтримка транзакцій.

Транзакції дозволяють групувати декілька операцій бази даних в одну логічну одиницю роботи та забезпечують цілісність даних. Python транзакції можуть бути керовані за допомогою методів commit() для підтвердження змін і rollback() для відкату змін. SQLite також підтримує створення індексів для прискорення виконання запитів до бази даних.

Індекси можуть бути створені з використанням оператора CREATE INDEX та прискорюють виконання запитів до таблиць з великим обсягом даних.

Python модуль sqlite3 також надає підтримку параметризованих запитів. Параметризовані запити дозволяють передавати параметри до SQL-запитів, що забезпечує захист від SQL-ін'єкцій та покращує продуктивність виконання запитів.

Отже, використання SQLite з sqlite3 у Python забезпечує зручний та ефективний спосіб роботи з базами даних у додатках Python. SQLite є відмінним вибором для додатків, яким потрібна легковажна та проста у використанні база даних, а модуль sqlite3 забезпечує зручний інтерфейс для роботи з нею.

2.6 Стилізація та бібліотека ttkthemes

Стилізація графічного інтерфейсу користувача (GUI) відіграє важливу роль у створенні приємного та естетичного візуального враження для користувачів. Завдяки можливостям стилізації існує можливість надавати своїм додаткам унікальний зовнішній вигляд, а також забезпечувати відповідність корпоративному стилю або дизайну [17].

В цьому контексті бібліотека ttkthemes є додатковим розширенням до стандартної бібліотеки Tkinter в Python, яка надає різні теми і стилі для віджетів Tkinter.

Tkinter сама по собі надає деякі стандартні стилі віджетів, але ttkthemes розширює цей функціонал, додаючи більше можливостей для стилізації.

Однією з ключових особливостей ttkthemes є надання різних тем оформлення, які можуть бути легко застосовані до віджетів Tkinter. Ці теми включають різні колірні схеми, шрифти, межі та інші стильові елементи, які допомагають створювати сучасні та стильні інтерфейси. Використання ttkthemes у додатках Python з Tkinter дозволяє створювати інтерфейси з більш просунутою стилізацією, яка може бути легко налаштована та адаптована під конкретні вимоги проекту. Це дозволяє створювати програми з унікальним та професійним зовнішнім виглядом, який привертає увагу користувачів та покращує їх враження від продукту. Одним із прикладів того, як ttkthemes може бути використана, є додавання кнопок, кадрів, полів введення та інших віджетів специфічних колірних схем і стилів, які відображають атмосферу програми або корпоративні кольори компанії. Це може допомогти зробити інтерфейс більш узгодженим та впізнаваним для користувачів [17].

Крім надання готових тем оформлення, ttkthemes також дозволяє створювати власні стилі користувача, що дозволяє забезпечити велику гнучкість в налаштуванні зовнішнього вигляду додатків. Це дозволяє створювати унікальні та індивідуальні інтерфейси, що відображають стиль та імідж бренду.

Використання ttkthemes може значно прискорити процес розробки інтерфейсів, оскільки власноруч не потрібно буде створювати стилі з нуля, а може скористатися готовими темами або адаптувати їх під свої потреби. Це також допомагає знизити витрати на розробку та забезпечує більш високу якість та консистентність інтерфейсу.

В цілому, ttkthemes є сучасним інструментом для стилізації графічного інтерфейсу в додатках Python з використанням Tkinter. Його гнучкі

можливості дозволяють створювати професійні та привабливі інтерфейси, які покращують користувальницький досвід та підвищують цінність продукту.

2.7 Модуль дати та часу datetime

Модуль datetime Python надає великі можливості для роботи з датами і часом. Цей модуль є частиною стандартної бібліотеки Python і є сучасним інструментом для роботи з датами, часом, дельтами часу тощо.

Концепцією модуля datetime є створення об'єктів, що становлять дату і час. Ці об'єкти можуть бути створені за допомогою різних конструкторів, які дозволяють вказати рік, місяць, день, годину, хвилину, секунду та мікросекунду [18].

Таким чином, можна створювати об'єкти, які представляють будь-яку дату та час у заданих межах. Крім того, модуль datetime надає можливості для роботи з поточним часом та датою. Наприклад, можна отримати поточну дату та час за допомогою функції datetime.now(). Це зручно для реєстрації часу виконання операцій, роботи з логами або інших сценаріїв, де потрібен поточний час.

Ще однією корисною можливістю модуля datetime є робота з дельтами часу. Дельта часу є різницею між двома моментами часу і може бути використана для додавання або віднімання певної кількості часу з об'єкта datetime. Наприклад, можна створити дельту часу, що становить 7 днів, і додати її до поточної дати для отримання дати через тиждень [18].

Модуль datetime також забезпечує можливості для форматування та розбору рядків, що представляють дату та час. Це дозволяє перетворювати рядки в об'єкти datetime і навпаки. Форматування та розбір рядків важливі для роботи з даними, що надходять із зовнішніх джерел, таких як введення користувача або дані з файлів.

Також модуль datetime надає безліч методів та операцій для роботи з об'єктами дати та часу. Наприклад, можна виконувати арифметичні операції з

об'єктами datetime, порівнювати їх між собою, витягувати компоненти дати та часу (наприклад, рік, місяць, день, година тощо). Ці методи та операції забезпечують широкі можливості для роботи з датами та часом у Python.

Модуль datetime також підтримує роботу з часовими поясами та переходом на літній час. Це важливо для додатків, які працюють у різних часових поясах або потребують точного обліку часу, включаючи перехід на літній та зимовий час [18].

Однією з основних переваг модуля datetime є його зручність та гнучкість. Він надає безліч методів та операцій для роботи з датами та часом, дозволяючи створювати складні та точні сценарії роботи з часом. Завдяки цим можливостям модуль datetime є невід'ємною частиною розробки додатків, пов'язаних з обробкою часу та дати в Python.

2.8 Модуль statistics у Python

Модуль statistics в Python є необхідним інструментом для роботи з різними статистичними операціями та обчисленнями. Він включає безліч функцій для аналізу даних, обчислення різних метрик і характеристик розподілів. У цьому модулі реалізовані методи обчислення середнього значення, медіани, моди, стандартного відхилення, і навіть інших важливих параметрів [19].

Середнє значення (або середнє арифметичне) є сумою всіх значень у вибірці, поділену на кількість цих значень. Для обчислення середнього значення в модулі statistics використовується функція mean(). Крім середнього значення, важливою статистичною характеристикою медіана.

Медіана є значенням, яке ділить впорядкований список навпіл. Якщо кількість значень у вибірці непарна, медіана буде середнім значенням цієї вибірки після впорядкування. Якщо кількість значень парна, то медіана буде середнім арифметичним двох середніх елементів після впорядкування. Функція median() у модулі statistics дозволяє обчислити медіану.

Далі, модуль statistics надає можливість обчислення моди – значення, яке найчастіше зустрічається у вибірці. Якщо у вибірці кілька значень зустрічаються однакова кількість разів і ця кількість більша, ніж для будь-якого іншого значення, то таких значень може бути кілька, і всі вони вважаються модою.

Функція mode() повертає список мод у порядку зменшення частоти. Стандартне відхилення є мірою розкиду даних щодо середнього значення. Воно показує, наскільки значення вибірці відрізняються від середнього значення. Чим більше стандартне відхилення, тим більший розкид даних. Для обчислення використовується функція stdev() [19].

Додатково, в модулі statistics доступні функції для обчислення інших важливих характеристик розподілів, таких як дисперсія (variance), квантил (quantiles), середнє геометричне (geometric_mean), середнє гармонійне (harmonic_mean), а також різні сумарні статистики, такі як сума елементів (sum), сума квадратів (sumsq), сума добутку пар (pairwise) та ін.

Перевага використання модуля statistics полягає в його простоті та зручності. Функції цього модуля можуть бути застосовані до різних типів даних, таких як списки, кортежі або навіть об'єкти, що ітеруються. Завдяки цій універсальності існує можливість легко проводити аналіз даних незалежно від їх формату та розміру. Крім того, модуль statistics надає високу точність обчислень, що особливо важливо під час роботи з великими обсягами даних або при виконанні складних статистичних операцій [19]. Це забезпечується використанням числових методів з високою точністю, що дозволяє уникнути помилок заокруглення або втрати точності під час виконання обчислень.

2.9 Висновки до другого розділу

Отже, Python – це універсальна мова програмування, проста у вивченні та використанні, з широким спектром бібліотек для роботи з даними, GUI і веброзробкою. Tkinter є стандартною бібліотекою Python для створення

графічних інтерфейсів користувача (GUI), інтуїтивно зрозуміла і проста в реалізації.

Потужна мова запитів до баз даних SQL дозволяє легко додавати, змінювати та витягувати дані з інвентарної системи. SQLite – легка й автономна реляційна система керування базами даних (СУБД), яка ідеально підходить для вбудованих систем, мобільних додатків і проєктів, де важлива простота та портативність. В той же час модуль Python sqlite3 підходить для роботи з базами даних SQLite, що дозволяє створювати, читати, оновлювати і видаляти дані в базах даних SQLite.

Бібліотека Python для створення тем для Tkinter ttkthemes доомагає змінити зовнішній вигляд віджетів Tkinter, роблячи їх привабливішими та сучаснішими. Модуль datetime у Python призначений для роботи з датами, часом і форматами дат, що дозволяє маніпулювати датами, обчислювати тимчасові інтервали, форматувати дати в різні рядки тощо.

Для реалізації автоматизованої інформаційної системи управління інвентарем на підприємстві рекомендується використовувати стек технологій Python, Tkinter, SQL, SQLite, sqlite3, ttkthemes і datetime. Цей стек технологій забезпечує гнучкість, масштабованість і простоту використання, що робить його ідеальним вибором для розробки ефективної та надійної системи управління інвентарем.

Перед початком розробки системи необхідно провести ретельне планування та аналіз вимог. Важливо забезпечити безпеку і конфіденційність даних інвентарної системи. Необхідно регулярно тестувати й оновлювати систему, щоб вона відповідала мінливим потребам підприємства.

РОЗДІЛ 3 РОЗРОБКА АВТОМАТИЗОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ІНВЕНТАРЕМ НА ПІДПРИЄМСТВІ

3.1 Постановка задачі

Метою процесу розроблення є створення програми для ефективного управління запасами товарів на підприємстві. Програма має надавати можливість додавання, оновлення, видалення та перегляду товарів та їх кількості, а також аналізувати статистику та візуалізувати дані у вигляді графіків.

3.1.1 Функціональні вимоги

Виділяються наступні функціональні вимоги:

1) додавання нових товарів у базу даних разом із зазначенням категорії та кількості;

2) оновлення кількості наявних товарів;

3) видалення товарів з бази даних;

4) перегляд списку наявних товарів та їх кількості;

5) можливість фільтрації товарів за категорією або кількістю;

6) експорт та імпорт даних у форматі CSV;

7) виведення статистичних показників, таких як середнє значення, максимальне та мінімальне значення кількості товарів.

3.1.2 Нефункціональні вимоги

Виділяються наступні нефункціональні вимоги:

1) інтуїтивно зрозумілий та зручний інтерфейс користувача;

2) ефективна робота програми з великою кількістю записів в базі даних;

3) надійна та безпечна робота з даними.

В цілому постановка задачі визначає основні функції та вимоги до програми управління інвентарем, які необхідно реалізувати для досягнення поставленої мети.

3.2 Архітектура проекту

Архітектура програми управління інвентарем може бути розділена на кілька компонентів, які взаємодіють між собою для забезпечення функціональності програми:

1) користувацький інтерфейс (GUI). Головне вікно програми – це вікно, яке містить всі елементи інтерфейсу, такі як кнопки, поля вводу, список товарів та інші елементи. Віджети Tkinter використовуються для створення різних елементів інтерфейсу, таких як кнопки, поля вводу, список тощо;

2) логіка додатку. Модуль управління інвентарем містить класи та функції, які відповідають за взаємодію з базою даних та обробку даних;

3) SQLite база даних. Використовується для зберігання інформації про товари, категорії та їх кількість;

4) модуль експорту/імпорту містить функції для експорту та імпорту даних у форматі CSV;

5) комунікація між інтерфейсом та логікою. Обробники подій відповідають за виклик відповідних функцій логіки додатку при взаємодії користувача з інтерфейсом;

6) комунікація з користувачем. Повідомлення використовуються для відображення повідомлень користувачеві (наприклад, про помилки або успішні операції). Діалогові вікна використовуються для отримання введених користувачем даних або підтвердження дій;

7) модулі сторонніх бібліотек. Matplotlib використовується для побудови графіків та візуалізації статистичних даних. Ttkthemes використовується для зміни вигляду елементів інтерфейсу.

Наведена архітектура проєкту дозволяє розділити логіку програми на

окремі компоненти, що спрощує розробку, тестування та підтримку програмного забезпечення. Кожен компонент виконує свої визначені завдання, а взаємодія між компонентами забезпечує необхідну функціональність програми.

3.3 Структура проекту

Структура програми управління інвентарем може бути організована наступним чином:

1) головний файл програми містить точку входу в програму (функцію main), яка створює головне вікно програми та інші необхідні об'єкти. Ініціалізує головне вікно та запускає головний цикл подій;

2) модуль GUI містить класи та функції, які відповідають за створення та оновлення елементів графічного інтерфейсу користувача (GUI). Він включає в себе створення вікон, кнопок, полів вводу, списків тощо;

3) модуль управління даними містить класи та функції, які відповідають за взаємодію з базою даних та обробку даних: функції для додавання, оновлення, видалення та перегляду товарів та їх кількості;

4) файл бази даних SQLite, який містить таблиці для зберігання даних про товари, категорії та їх кількість;

5) модуль експорту/імпорту даних містить функції для експорту та імпорту даних у форматі CSV. Він включає в себе функції для збереження даних у файл CSV та завантаження даних з файлу CSV до бази даних;

6) модуль візуалізації даних містить функції для побудови графіків та візуалізації статистичних даних. Використовується для створення графіків на основі даних, збережених у базі даних;

7) модулі сторонніх бібліотек містять бібліотеки, такі як Matplotlib та ttkthemes, які використовуються для створення графічного інтерфейсу та візуалізації даних.

Описана структура дозволяє розділити програму на логічні компоненти,

що полегшує розробку, тестування та підтримку програмного забезпечення. Кожен модуль виконує певні функції та має чітко визначену відповідальність, що сприяє підтримці та розширенню програми у майбутньому.

3.4 База даних

База даних для програми управління інвентарем містить таблиці для зберігання інформації про товари, категорії та їх кількість.

Таблиця категорій товарів «categories» містить наступні поля:

– id (INTEGER, PRIMARY KEY). Унікальний ідентифікатор категорії товарів;

– name (TEXT). Назва категорії товарів.

Таблиця інвентарю товарів «inventory» містить наступні поля:

– id (INTEGER, PRIMARY KEY). Унікальний ідентифікатор товару;

– item (TEXT). Назва товару;

– category_id (INTEGER, FOREIGN KEY). Ідентифікатор категорії, зовнішній ключ, який посилається на таблицю «categories»;

– quantity (INTEGER). Кількість товару на складі;

– last_updated (TEXT). Дата та час останнього оновлення кількості товару.

Ця структура бази даних дозволяє зберігати інформацію про товари, їх кількість та категорії. Таблиця «categories» дозволяє розділити товари на категорії для кращого організації та пошуку. Таблиця «inventory» містить інформацію про кожен товар, його категорію, кількість та дату останнього оновлення. Завдяки цій структурі бази даних можна ефективно зберігати, оновлювати та аналізувати інформацію про інвентар на підприємстві.

3.5 Проектування користувацького інтерфейсу

Завдяки проектуванню користувацького інтерфейсу важливо

забезпечити зручність користування програмою та доступність всіх необхідних функцій. Отже, користувацький інтерфейс складається з наступних складових:

1) головне вікно програми. Містить головне меню зі списком доступних опцій, таких як «Додати товар», «Оновити кількість», «Видалити товар», «Переглянути статистику», «Експорт/Імпорт даних» та «Вийти». Також включає в себе список товарів разом з їх кількістю та категорією;

2) додавання, оновлення та видалення товарів. Для додавання нового товару користувачу слід заповнити поля для назви товару, категорії та кількості, після чого натиснути кнопку «Додати». Для оновлення кількості наявного товару користувач може вибрати товар зі списку, ввести нову кількість та натиснути кнопку «Оновити». Для видалення товару користувач може вибрати товар зі списку та натиснути кнопку «Видалити»;

3) перегляд статистики та візуалізація даних. Кнопка «Переглянути статистику» відкриває вікно зі статистичними показниками, такими як середня кількість, максимальна та мінімальна кількість товарів. Кнопка «Побудувати графік» відкриває вікно з графіком, що відображає кількість товарів у кожній категорії;

4) експорт та імпорт даних. Кнопки «Експорт даних» та «Імпорт даних» дозволяють користувачу зберегти поточний стан бази даних у файл CSV або завантажити дані з файлу CSV до бази даних;

5) додаткові функції. Додаткові опції, такі як можливість фільтрації товарів за категорією або кількістю, можуть бути додані для полегшення навігації користувача.

3.6 Процес роботи програмного забезпечення

Процес роботи програмного додатка для управління інвентарем може бути описаний покроковим чином:

1) запуск програмного додатка. Користувач запускає програмне додаток

інвентаризації, яке відображає головне вікно зі списком товарів та доступних опцій;

2) взаємодія з головним меню. Користувач може обрати необхідну опцію з головного меню, таку як «Додати товар», «Оновити кількість», «Видалити товар», «Переглянути статистику», «Експорт/Імпорт даних» або «Вийти»;

3) додавання нового товару. Користувач обирає опцію «Додати товар» з головного меню. Відкривається вікно для введення назви товару, вибору категорії та вказання кількості. Після введення всіх даних користувач натискає кнопку «Додати», інформація про товар додається до бази даних, а список товарів оновлюється;

4) оновлення кількості товару. Користувач обирає товар зі списку та обирає опцію «Оновити кількість» з головного меню. Відкривається вікно для введення нової кількості товару. Після введення нової кількості користувач натискає кнопку «Оновити», інформація про товар оновлюється у базі даних;

5) видалення товару. Користувач обирає товар зі списку та обирає опцію «Видалити товар» з головного меню. Обраний товар видаляється з бази даних, а список товарів оновлюється;

6) перегляд статистики та візуалізація даних. Користувач обирає опцію «Переглянути статистику» з головного меню. Відкривається вікно зі статистичними показниками, такими як середня кількість, максимальна та мінімальна кількість товарів. Користувач може також обрати опцію «Побудувати графік» для відображення графіка кількості товарів у кожній категорії;

7) експорт та імпорт даних. Користувач може обрати опцію «Експорт даних» або «Імпорт даних» з головного меню для збереження поточного стану бази даних у файл CSV або завантаження даних з файлу CSV до бази даних;

8) вихід з програми. Користувач обирає опцію «Вийти» з головного меню або закриває головне вікно програми, що завершує роботу програмного додатка.

3.7 Процес тестування

Тестування програмного додатка для управління інвентарем вручну включає в себе перевірку різних функціональностей та переконання в тому, що вони працюють правильно.

Загальний вигляд розробленого програмного забезпечення наведений на рисунку 3.1.

У застосунку передбачена валідація даних, результати якої відображені на рисунках 3.2-3.4.

Також у застосунку присутнє меню, список пунктів якого представлений на рисунку 3.5.

Для додавання нової категорії використовується відповідний пункт у меню. Інтерфейс цього вікна наведений на рисунку 3.6, а результат додавання нової категорії та її вибір з випадаючого списку у головному вікні – на рисунку 3.7.

Результат додавання нового товару для раніше створеної категорії товарів наведений на рисунку 3.8.

Для оновлення категорії товарів використовується відповідний пункт меню (рис. 3.5). Інтерфейс цього вікна представлений на рисунку 3.9.

На рисунку 3.10 наведений приклад заповнення даних про нову назву категорії товарів, а на рисунку 3.11 – результат оновлення обраної категорії.

Якщо товар необхідно видалити, необхідно скористатися відповідною можливістю у головному вікні (рис. 3.1). Перш ніж товар буде видалено, користувачеві необхідно буде підтвердити (рис. 3.12) намір видалення товару. Результат видалення товару після підтвердження наведений на рисунку 3.13.

У головному вікні застосунку також передбачена можливість для пошуку товарів та додаткової можливості використання фільтрів для цієї ж дії. Результат пошуку товару наведений на рисунку 3.14, а результат використання фільтрів – на рисунку 3.15.

Для вже доданих товарів можна переглянути статистику, скориставшись

відповідною кнопкою у головному вікні (рис. 3.1). Результат виведення статистики на екран наведений на рисунку 3.16. Так само можна переглянути і графік за категоріями товарів. Вікно графіка наведено на рисунку 3.17.

Якщо необхідно зберегти базу даних у вигляді копії, така можливість теж передбачена. Для цього необхідно натиснути відповідну кнопку у головному вікні (рис. 3.1). Сповіщення про успішне створення копії бази даних наведене на рисунку 3.18. Результатом створення копії є відповідний файл бази даних у файловій системі (рис. 3.19).

Для іншого формату зберігання даних передбачена можливість імпорту та експорту з CSV формату (рис. 3.20). Результат експорту даних у цей формат наведений на рисунку 3.21.

Категорію товарів, як і самі товари, можна видалити. Вікно для вибору категорії для її подальшого видалення наведено на рисунку 3.22. Перед тим, як видалити категорію, необхідно буде підтвердити намір виконання цієї дії (рис. 3.23). Результат видалення категорії товарів наведений на рисунку 3.24.

Результат імпорту з файлу CSV формату наведений на рисунку 3.25.

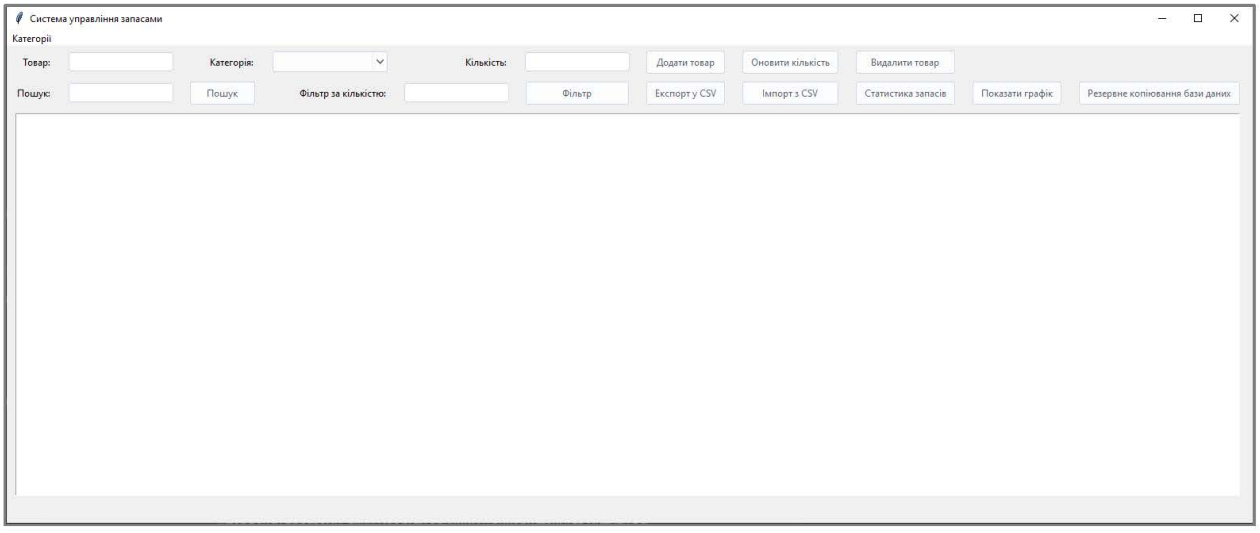

Рисунок 3.1 – Інтерфейс програмного застосунку для управління інвентарем на підприємстві

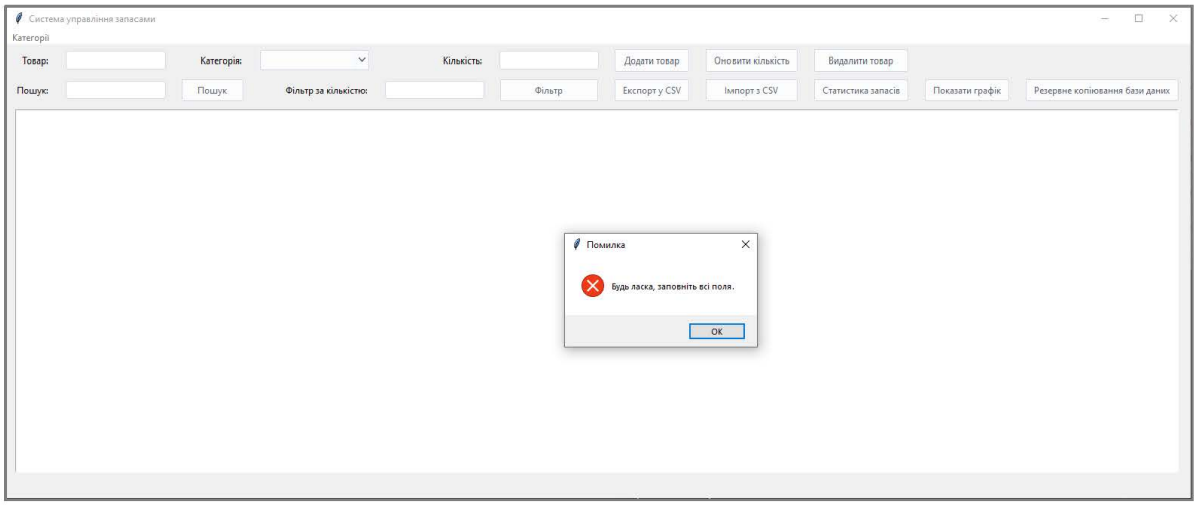

Рисунок 3.2 – Валідація даних та сповіщення при помилку

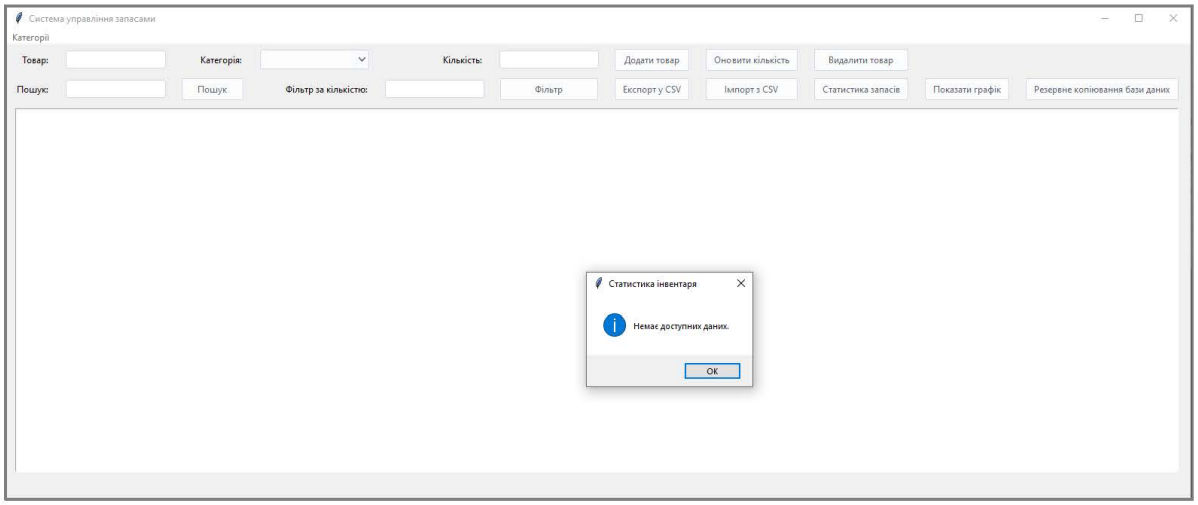

Рисунок 3.3 – Валідація даних та сповіщення про попередження

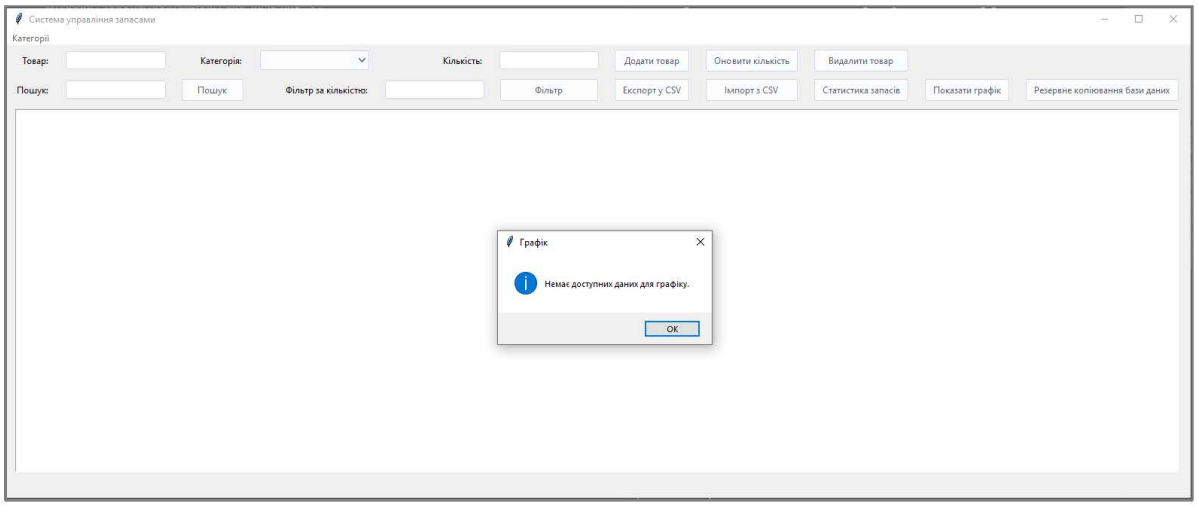

Рисунок 3.4 – Валідація даних та сповіщення про попередження

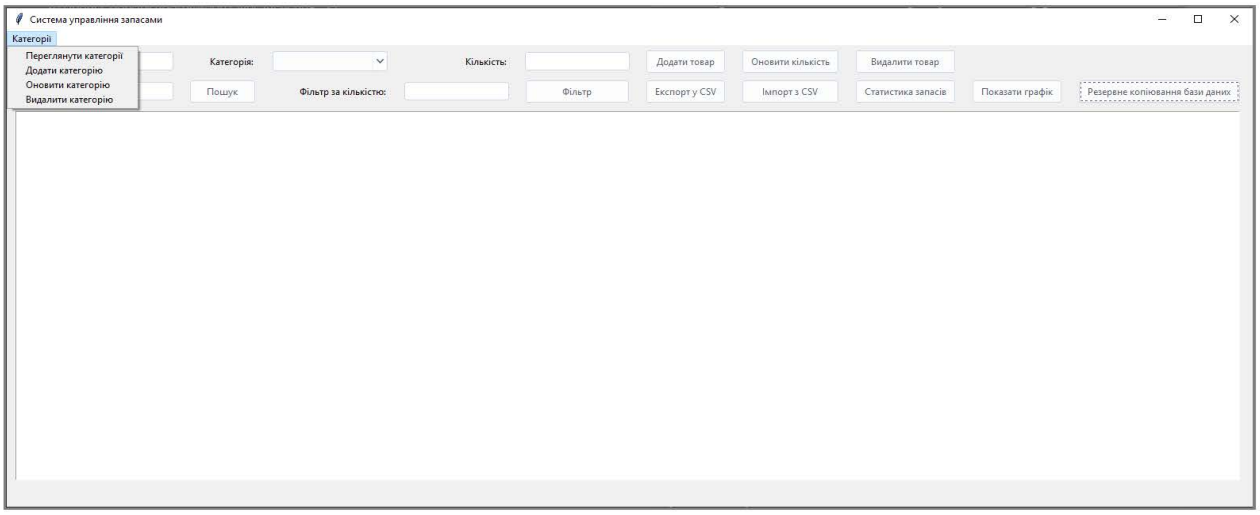

Рисунок 3.5 – Список пунктів меню

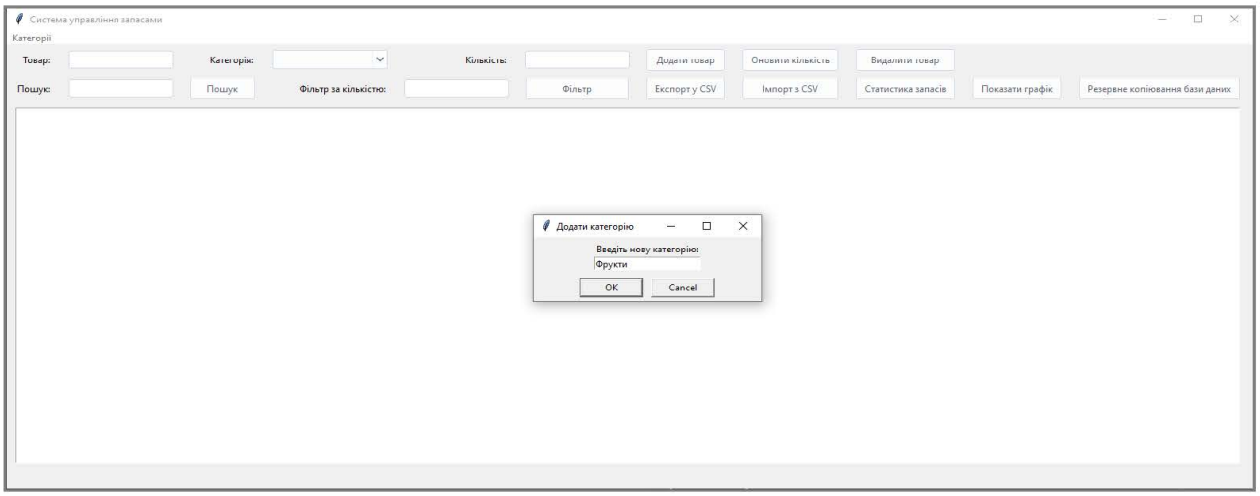

Рисунок 3.6 – Вікно додавання нової категорії

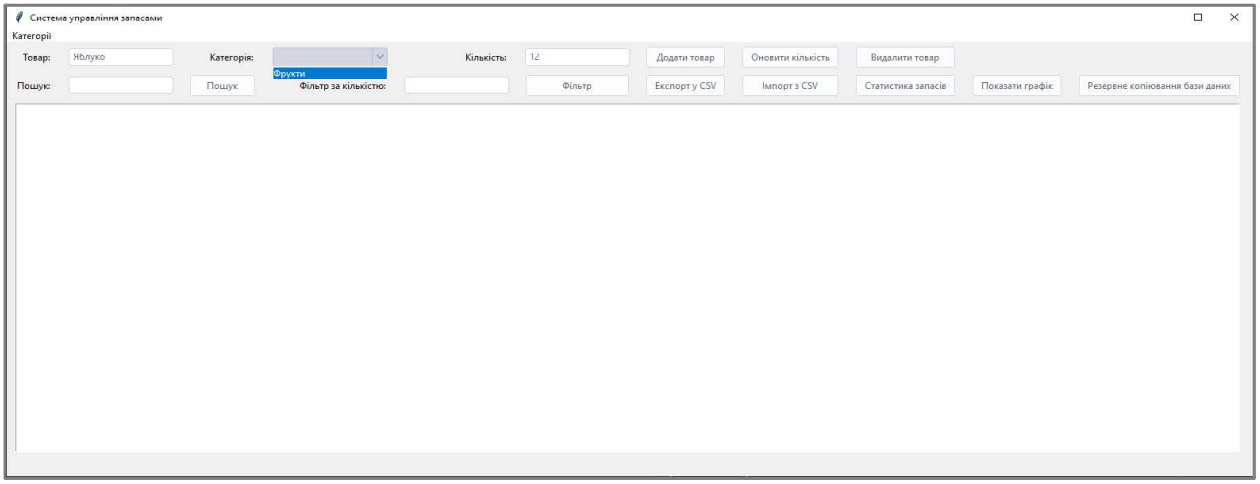

Рисунок 3.7 – Результат додавання нової категорії та її вибір з випадаючого списку

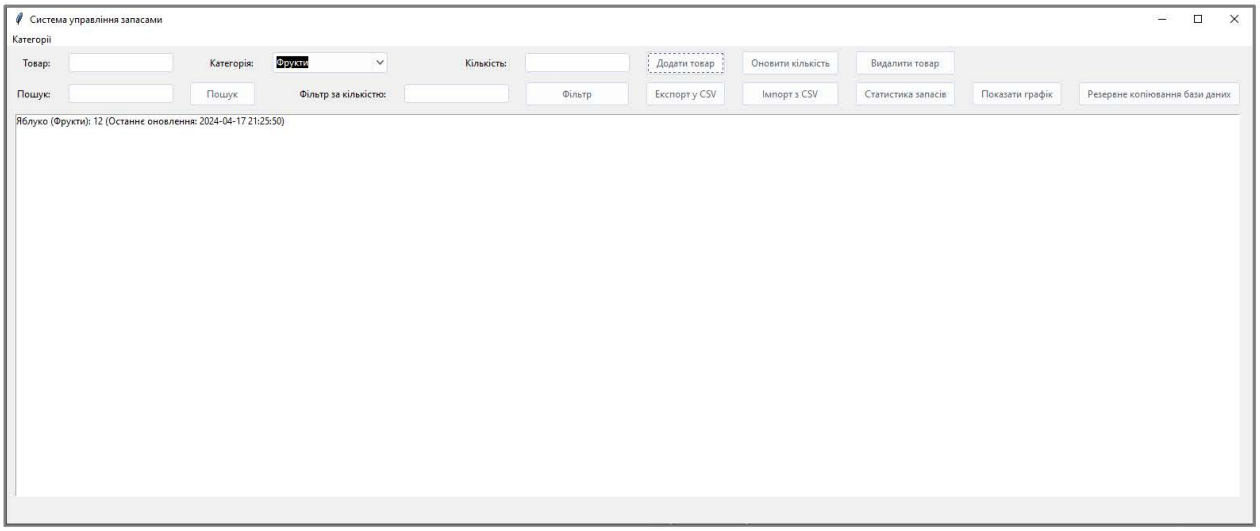

Рисунок 3.8 – Результат додавання нового товару для раніше створеної категорії товарів

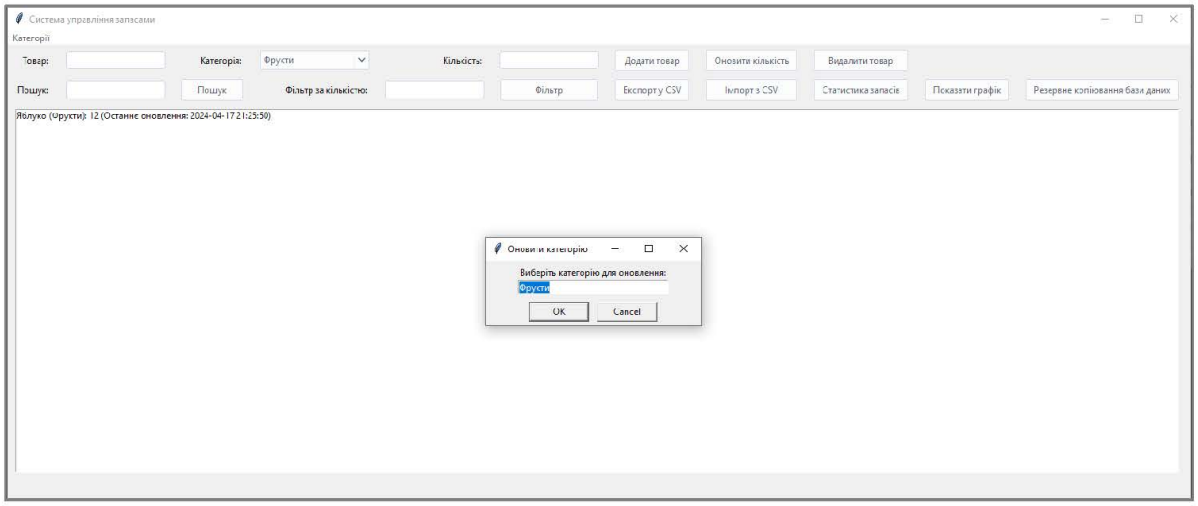

Рисунок 3.9 – Вікно для оновлення категорії та її назви

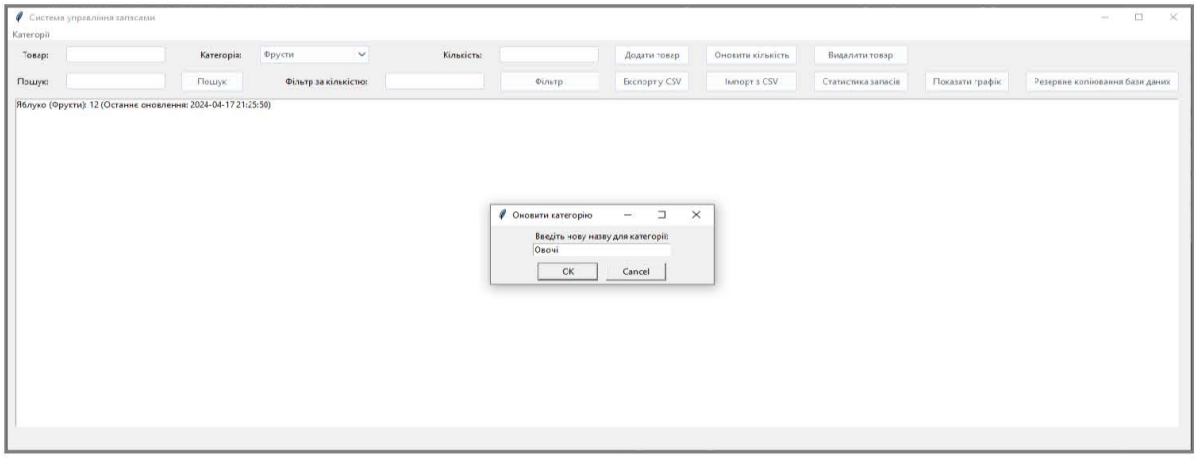

Рисунок 3.10 – Заповнення даних про нову назву категорії

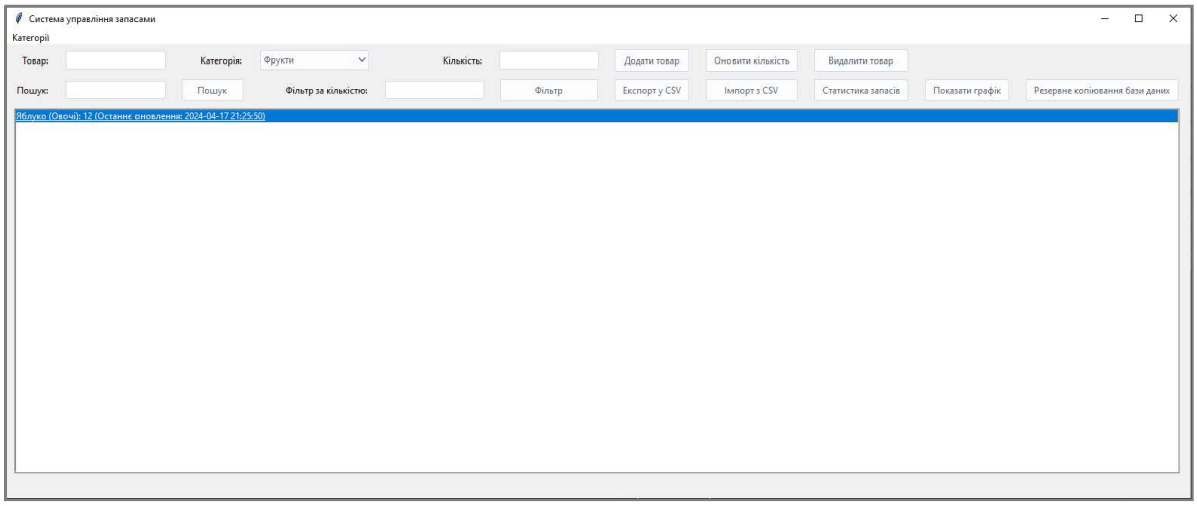

Рисунок 3.11 – Результат оновлення категорії

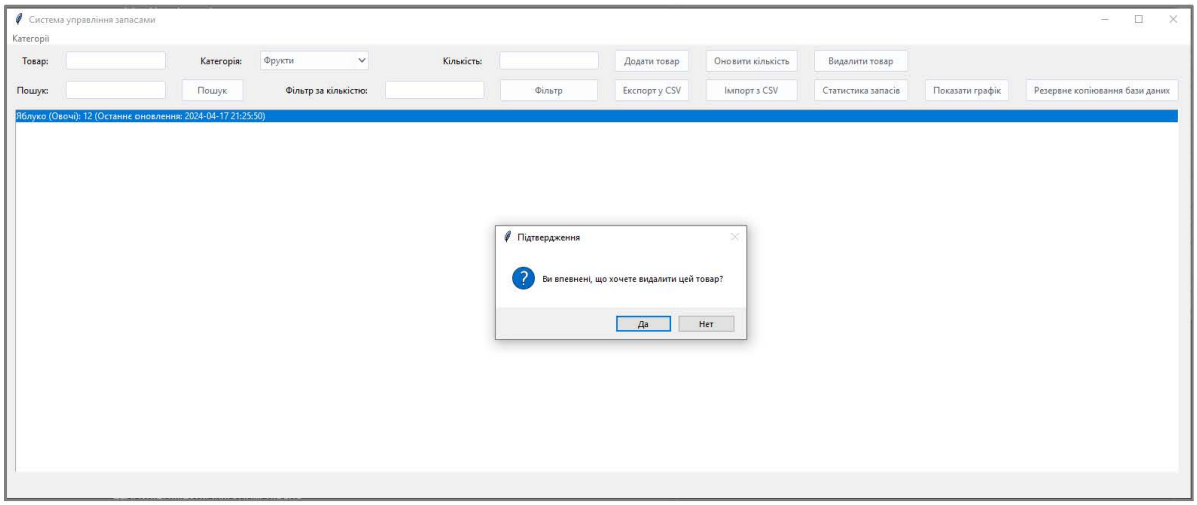

Рисунок 3.12 – Попередження про видалення товару

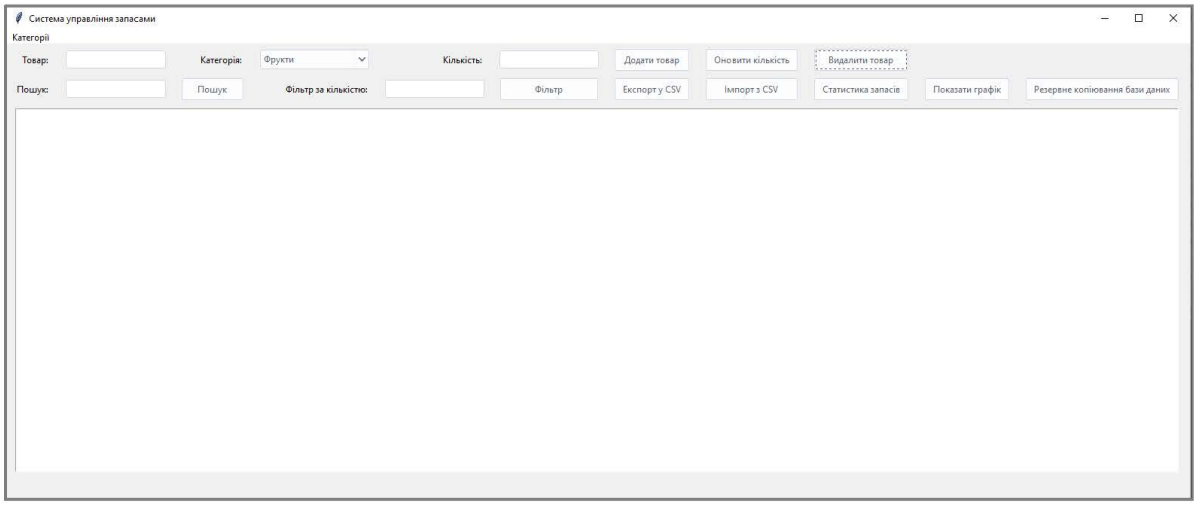

Рисунок 3.13 – Результат видалення товару

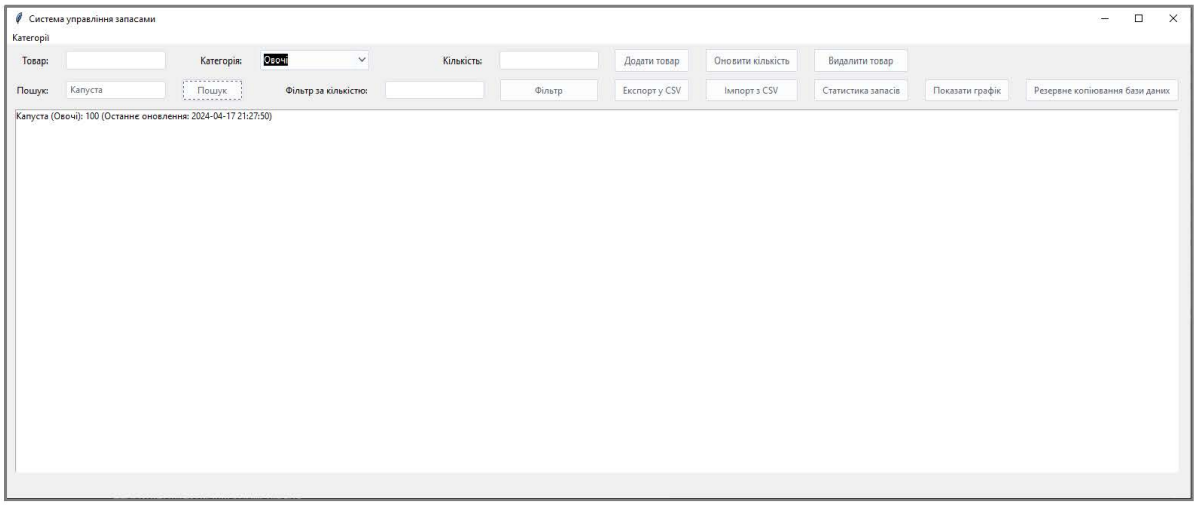

Рисунок 3.14 – Результат пошуку товару

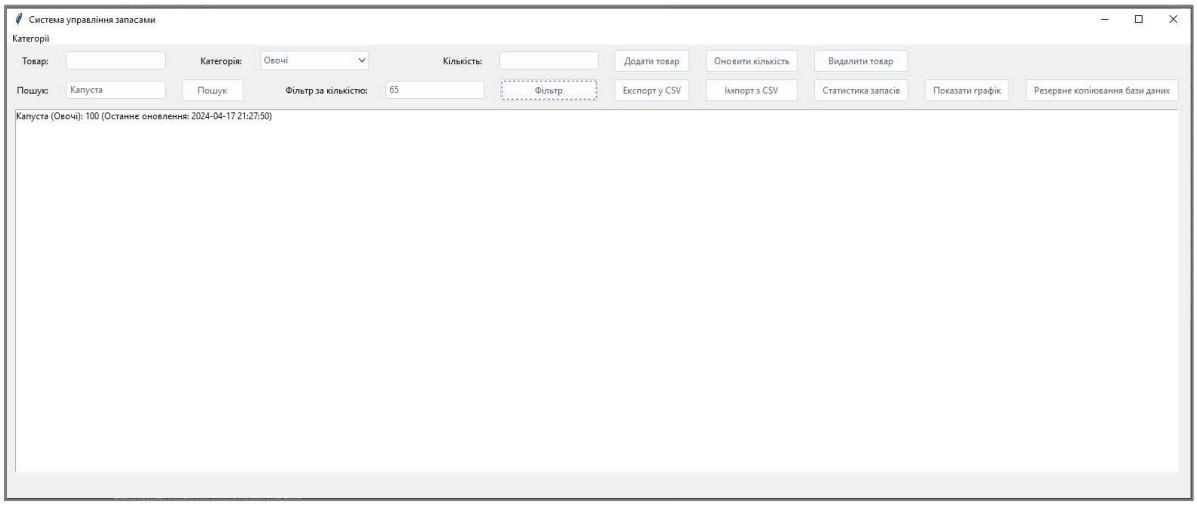

Рисунок 3.15 – Результат використання фільтрів для пошуку

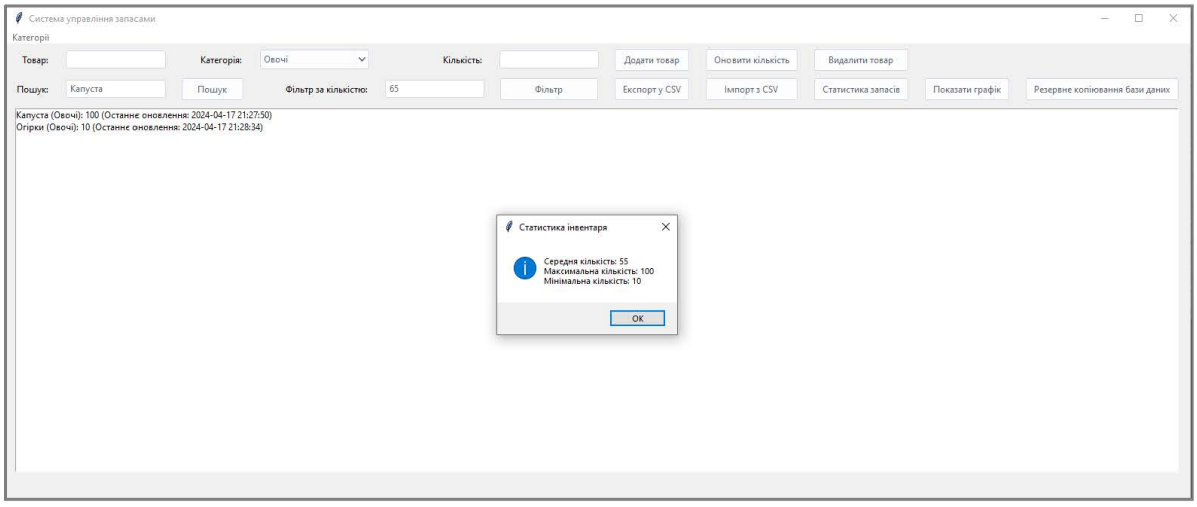

Рисунок 3.16 – Перегляд статистики по товарам

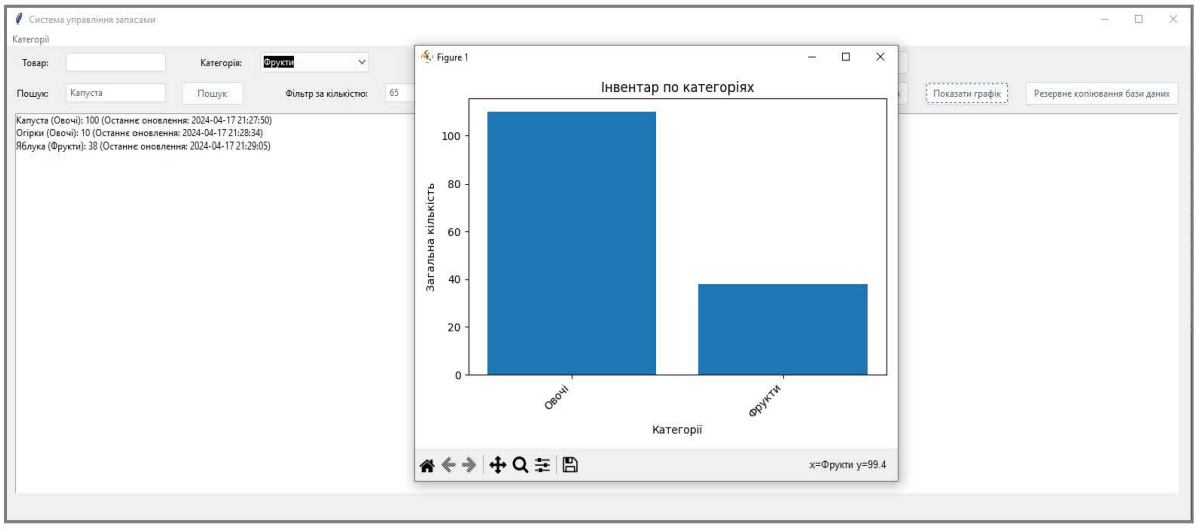

Рисунок 3.17 – Перегляд графіка за категоріями товарів

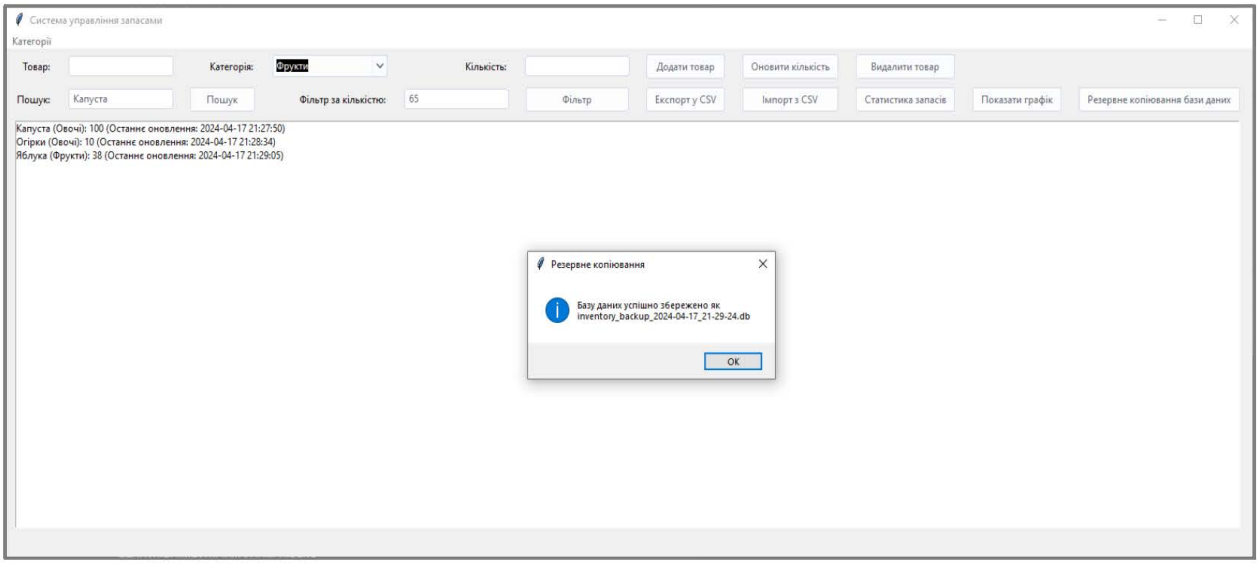

Рисунок 3.18 – Результат зберігання копії бази даних застосунку

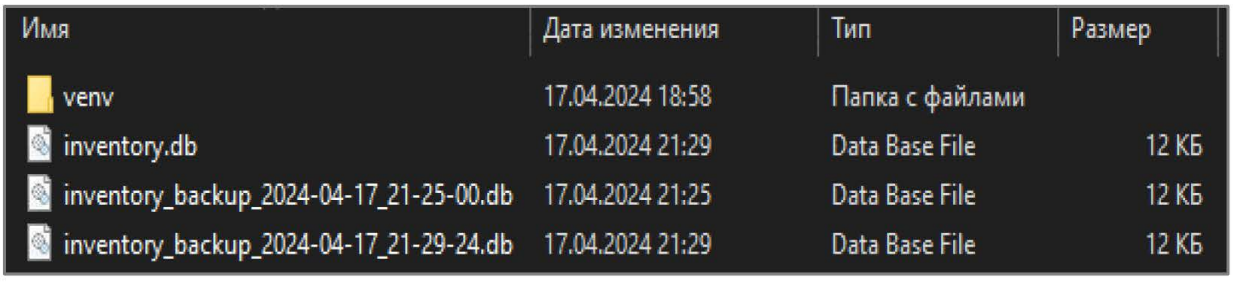

Рисунок 3.19 – Копія бази даних застосунку у файловій системі

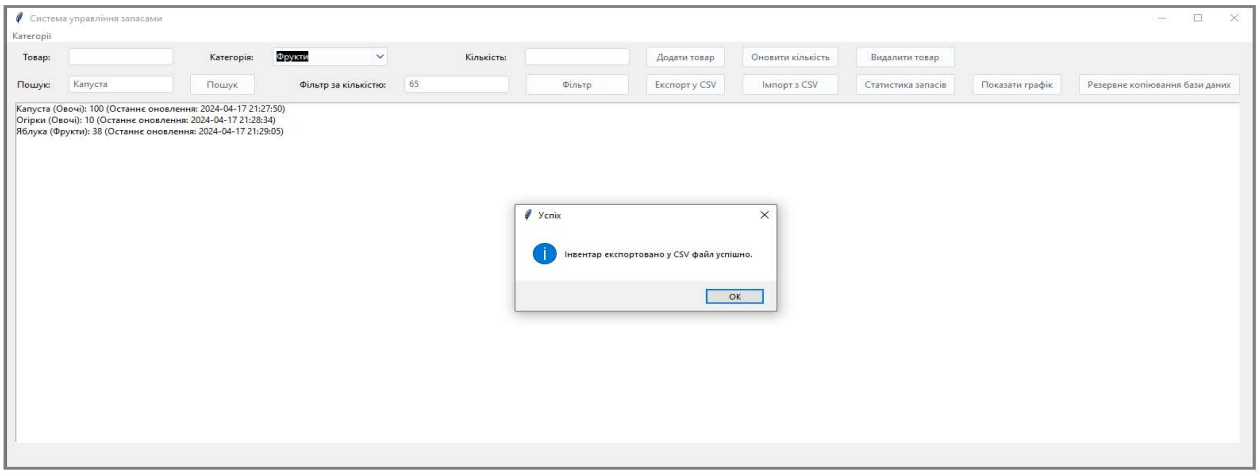

Рисунок 3.20 – Експорт даних у CSV формат

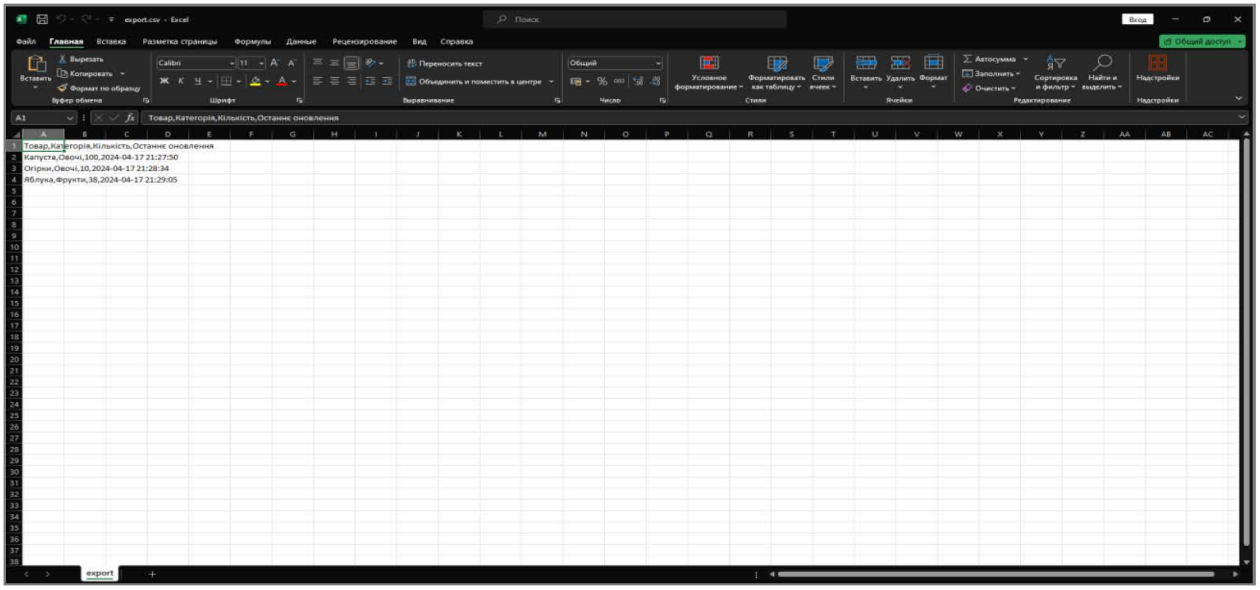

Рисунок 3.21 – Результат експорту даних у формат CSV

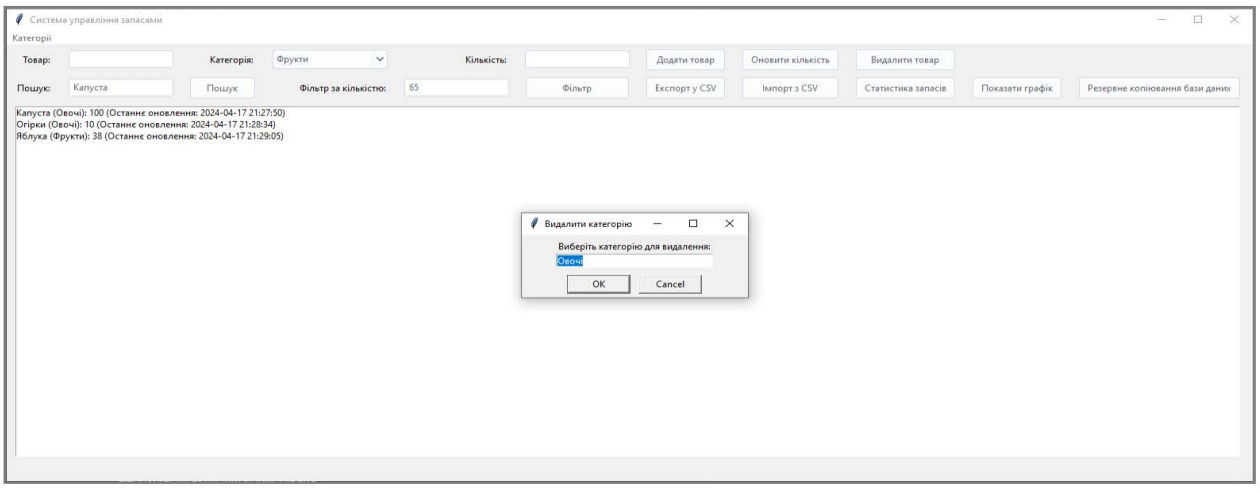

Рисунок 3.22 – Вікно для вибору категорії для її подальшого видалення

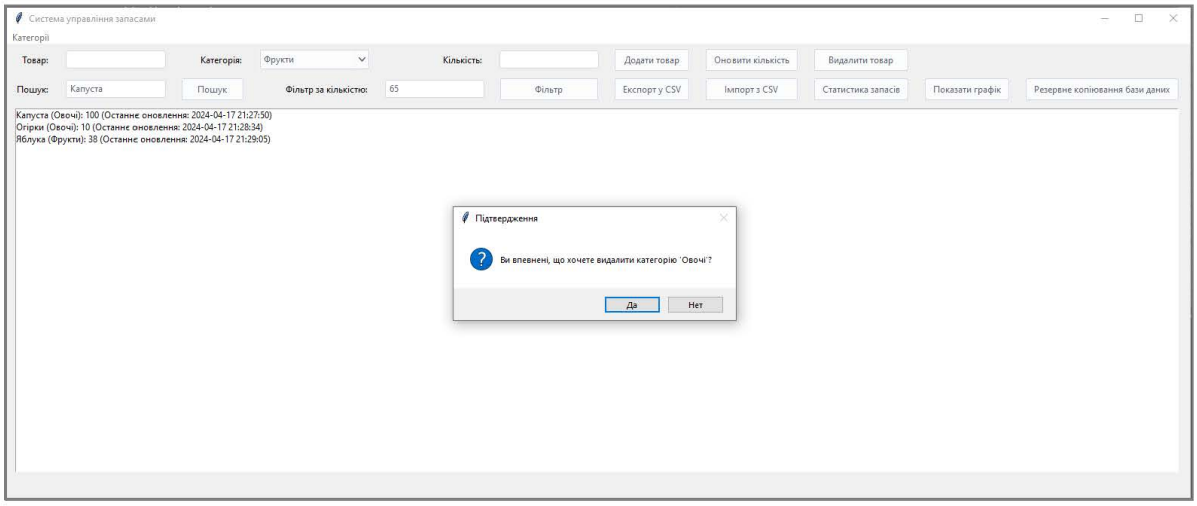

Рисунок 3.23 – Підтвердження про видалення категорії товарів

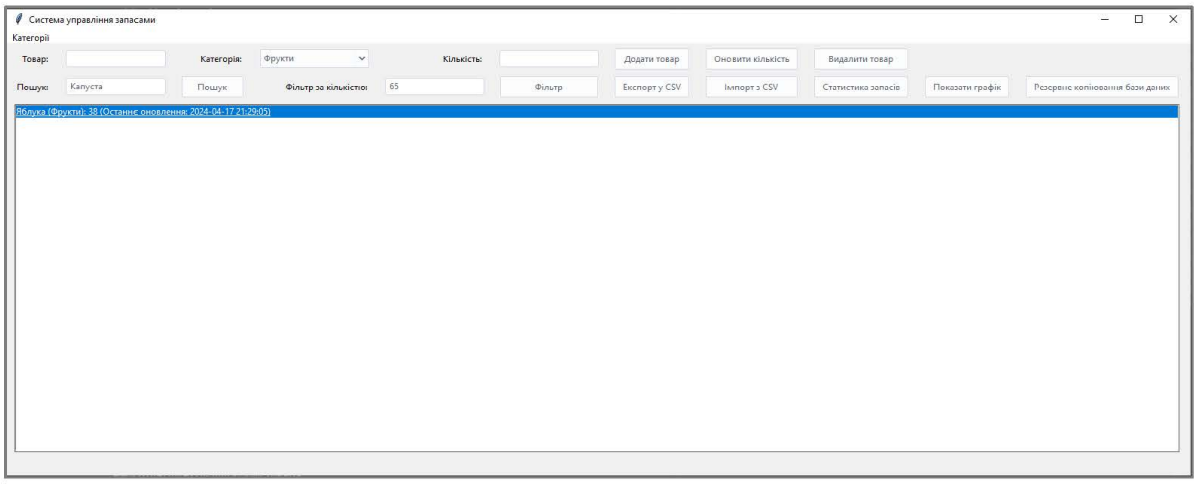

Рисунок 3.24 – Результат видалення категорії товарів

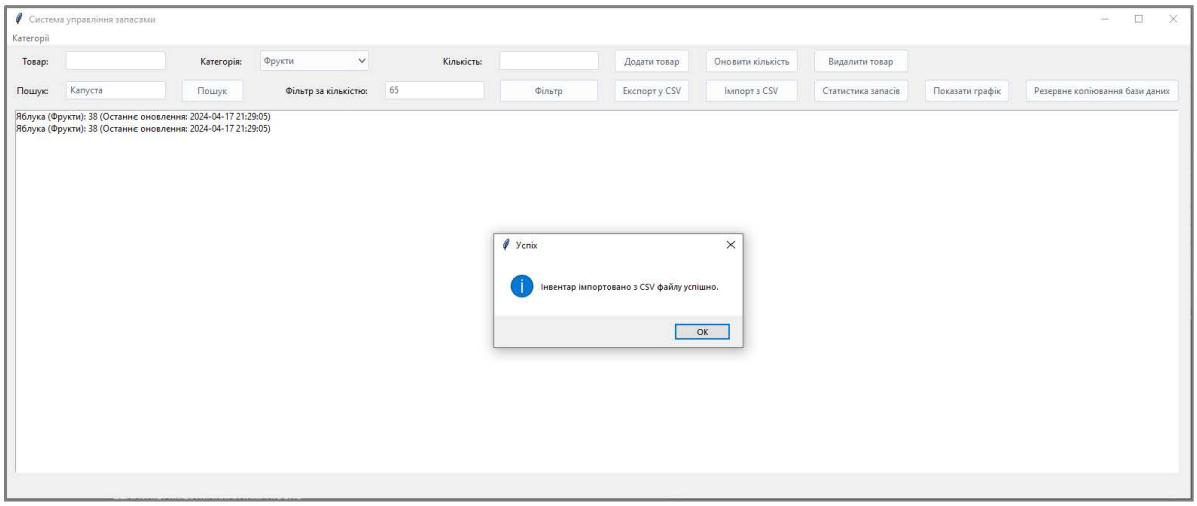

Рисунок 3.25 – Результат імпорту з CSV файлу

3.8 Висновки до третього розділу

Визначено цілі та завдання проекту, які полягають у створенні програмного забезпечення для управління інвентарем на підприємстві. Описана загальна структура програми, яка включає різні компоненти, такі як користувацький інтерфейс, логіка додатка, база даних та інші.

Детально описано, як різні модулі та компоненти програми організовані та взаємодіють між собою для забезпечення функціональності програми. Описана структура бази даних, яка використовується для зберігання інформації про товари, категорії та їх кількість. Розглянуто розробку інтерфейсу програми, який має бути зручним у використанні та містити всі необхідні функції для користувача.

Описано послідовність дій користувача під час використання програмного додатка та функціональність, яка реалізована для кожної операції. Подано методи та кроки для тестування програмного додатка вручну, щоб перевірити правильність його роботи та функціональність

ВИСНОВКИ

Мета роботи полягає у створенні ефективного інструменту, що дало змогу оптимізувати облік, контроль та управління запасами товарів і матеріальними ресурсами компанії.

Об'єктом дослідження був процес управління інвентарем на підприємстві. Цей процес включав в себе облік, контроль і управління запасами товарів і матеріальними ресурсами компанії. У рамках дослідження розглядався весь ланцюжок дій, починаючи від надходження товарів на склад, їхнього зберігання та переміщення, закінчуючи відвантаженням товарів клієнтам або використанням у виробничих процесах.

Предметом дослідження була розробка автоматизованої інформаційної системи управління інвентарем на підприємстві. Ця система представляє собою комплекс програмно-апаратних засобів, які забезпечують автоматизацію процесів обліку, контролю та управління запасами товарів і матеріальними ресурсами компанії.

В рамках аналізу виявлено основні завдання та цілі дослідження. Описано різні методи та підходи до розробки автоматизованих інформаційних систем управління інвентарем. Проаналізовано різноманітні програмні інструменти та технології, обрані для реалізації проекту.

Представлені етапи та методи розробки автоматизованої інформаційної системи управління інвентарем, включаючи постановку завдання, проектування архітектури та структури системи, розробку бази даних, проектування інтерфейсу користувача, процес роботи програмного забезпечення, тестування та впровадження.

Зроблені висновки щодо кожного розділу дослідження, узагальнюються результати проведеного аналізу та розробки. Формулюються рекомендації щодо подальшого розвитку та вдосконалення системи управління інвентарем.

Таким чином, робота є комплексним аналізом та розробкою автоматизованої інформаційної системи управління інвентарем на

підприємстві, що має практичну значимість для покращення процесів управління ресурсами та підвищення ефективності бізнесу.

Кваліфікаційна робота виконана у відповідності до стандарту спеціальності 121 «Інженерія програмного забезпечення» і демонструє володіння такими компетентностями як:

– здатність ідентифікувати, класифікувати та формулювати вимоги до програмного забезпечення;

– здатність брати участь у проектуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його структури, поведінки та процесів функціонування;

– здатність розробляти архітектури, модулі та компоненти програмних систем;

– здатність обґрунтовано обирати та освоювати інструментарій з розробки та супроводження програмного забезпечення;

– здатність до алгоритмічного та логічного мислення тощо.

Серед програмних результатів, визначених стандартом, кваліфікаційна робота реалізовує наступні:

– аналізувати, цілеспрямовано шукати і вибирати необхідні для вирішення професійних завдань інформаційно-довідникові ресурси і знання з урахуванням сучасних досягнень науки і техніки;

– сміти розробляти людино-машинний інтерфейс;

– знати та вміти використовувати методи та засоби збору, формулювання та аналізу вимог до програмного забезпечення;

– проводити передпроектне обстеження предметної області, системний аналіз об'єкта проектування;

– мотивовано обирати мови програмування та технології розробки для розв’язання завдань створення і супроводження програмного забезпечення;

– знати та вміти застосовувати інформаційні технології обробки, зберігання та передачі даних;

– вміти документувати та презентувати результати розробки програмного забезпечення тощо.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. SAP. URL: https://www.sap.com/ukraine/products/erp.html

2. Oracle. URL: https://www.oracle.com/scm/inventory-management/

3. Microsoft Dynamics 365 for Supply Chain Management. URL: https://www.microsoft.com/en-US/dynamics-365/products/supply-chain-management

4. IBM Sterling Inventory Control Tower. URL: https://www.ibm.com/products/supply-chain-intelligence-suite/food-safety

5. Zoho. URL: https://www.zoho.com/inventory/

6. Quickbooks. URL: https://quickbooks.intuit.com/accounting/track-expenses/

7. Fishbowl Inventory. URL: https://www.fishbowlinventory.com/

8. Wasp Inventory Control. URL: https://www.waspbarcode.com/applications

9. Inflow inventory. URL: https://www.inflowinventory.com/features/inventory-control

10. Python Documentation. URL: https://www.python.org/

11. Tkinter Documentation. URL: https://docs.python.org/uk/3/library/tkinter.html

12. Moore A. D. Python GUI Programming with Tkinter: Design and Build Functional and User-Friendly GUI Applications, 2nd Edition. Packt Publishing, Limited, 2021.

13. Elder J. TKinter Widget Quick Reference Guide. Codemy.com, 2022.

14. Zhao A. SQL Pocket Guide: A Guide to SQL Usage. O'Reilly Media, Incorporated, 2021. 336 с.

15. SQL QuickStart Guide: The Simplified Beginner's Guide to Managing, Analyzing, and Manipulating Data With SQL (Coding & Programming - QuickStart Guides).

16. SQLite Documentation. URL: https://www.sqlite.org/

17. Ttkthemes Documentation. URL: https://ttkthemes.readthedocs.io/en/latest/

18. Datetime module Documentation. URL: https://docs.python.org/uk/3/library/datetime.html

19. Statistics module Documentation. URL: https://docs.python.org/uk/3/library/statistics.html

ДОДАТОК А

ЛІСТИНГ ПРОГРАМНОГО КОДУ

```
import tkinter as tk
from tkinter import messagebox, simpledialog, filedialog, ttk
import sqlite3
from datetime import datetime
import csv
import statistics
import matplotlib.pyplot as plt
from ttkthemes import ThemedStyle

class InventoryApp:
    def __init__(self, master):
        self.master = master
        self.master.title("Система управління запасами")
        self.master.geometry("1570x600")  # Встановлення розміру вікна

        self.db_connection = sqlite3.connect('inventory.db')
        self.create_tables()

        self.style = ThemedStyle(self.master)
        self.style.theme_use("arc")

        self.create_widgets()


    def create_tables(self):
        cursor = self.db_connection.cursor()
```

```
        cursor.execute('''CREATE TABLE IF NOT EXISTS categories
                          (id INTEGER PRIMARY KEY, name TEXT)''')
        cursor.execute('''CREATE TABLE IF NOT EXISTS inventory
                        (id INTEGER PRIMARY KEY, item TEXT, category_id INTEGER,
quantity INTEGER, last_updated TEXT,
                         FOREIGN KEY(category_id) REFERENCES categories(id))''')
        self.db_connection.commit()


    def create_widgets(self):
        self.menu_bar = tk.Menu(self.master)

        self.categories_menu = tk.Menu(self.menu_bar, tearoff=0)
                self.categories_menu.add_command(label="Переглянути категорії",
command=self.view_categories)
                    self.categories_menu.add_command(label="Додати категорію",
command=self.add_category)
                  self.categories_menu.add_command(label="Оновити категорію",
command=self.update_category)
                 self.categories_menu.add_command(label="Видалити категорію",
command=self.delete_category)
        self.menu_bar.add_cascade(label="Категорії", menu=self.categories_menu)

        self.master.config(menu=self.menu_bar)

        self.item_label = tk.Label(self.master, text="Товар:")
        self.item_label.grid(row=0, column=0, padx=10, pady=5, sticky="E")

        self.item_entry = ttk.Entry(self.master)
        self.item_entry.grid(row=0, column=1, padx=10, pady=5)
```

```
        self.category_label = tk.Label(self.master, text="Категорія:")
        self.category_label.grid(row=0, column=2, padx=10, pady=5, sticky="E")

        self.category_combobox = ttk.Combobox(self.master, state="readonly")
        self.category_combobox.grid(row=0, column=3, padx=10, pady=5)

        self.load_categories()

        self.quantity_label = tk.Label(self.master, text="Кількість:")
        self.quantity_label.grid(row=0, column=4, padx=10, pady=5, sticky="E")

        self.quantity_entry = ttk.Entry(self.master)
        self.quantity_entry.grid(row=0, column=5, padx=10, pady=5)

        self.add_button = ttk.Button(self.master, text="Додати товар",
command=self.add_item)
        self.add_button.grid(row=0, column=6, padx=10, pady=5, sticky="WE")

        self.update_button = ttk.Button(self.master, text="Оновити кількість",
command=self.update_quantity)
        self.update_button.grid(row=0, column=7, padx=10, pady=5, sticky="WE")

        self.remove_button = ttk.Button(self.master, text="Видалити товар",
command=self.remove_item)
        self.remove_button.grid(row=0, column=8, padx=10, pady=5, sticky="WE")

        self.search_label = tk.Label(self.master, text="Пошук:")
        self.search_label.grid(row=1, column=0, padx=10, pady=5, sticky="E")
```

```
        self.search_entry = ttk.Entry(self.master)
        self.search_entry.grid(row=1, column=1, padx=10, pady=5)

        self.search_button = ttk.Button(self.master, text="Пошук",
command=self.search_item)
        self.search_button.grid(row=1, column=2, padx=10, pady=5, sticky="WE")

        self.filter_label = tk.Label(self.master, text="Фільтр за кількістю:")
        self.filter_label.grid(row=1, column=3, padx=10, pady=5, sticky="E")

        self.filter_entry = ttk.Entry(self.master)
        self.filter_entry.grid(row=1, column=4, padx=10, pady=5)

        self.filter_button = ttk.Button(self.master, text="Фільтр",
command=self.filter_items)
        self.filter_button.grid(row=1, column=5, padx=10, pady=5, sticky="WE")

        self.export_button = ttk.Button(self.master, text="Експорт у CSV",
command=self.export_csv)
        self.export_button.grid(row=1, column=6, padx=10, pady=5, sticky="WE")

        self.import_button = ttk.Button(self.master, text="Імпорт з CSV",
command=self.import_csv)
        self.import_button.grid(row=1, column=7, padx=10, pady=5, sticky="WE")

        self.statistics_button = ttk.Button(self.master, text="Статистика запасів",
command=self.show_statistics)
        self.statistics_button.grid(row=1, column=8, padx=10, pady=5, sticky="WE")

        self.graph_button = ttk.Button(self.master, text="Показати графік",
```

```
command=self.show_graph)
        self.graph_button.grid(row=1, column=9, padx=10, pady=5, sticky="WE")

        self.backup_button = ttk.Button(self.master, text="Резервне копіювання бази даних", command=self.backup_database)
        self.backup_button.grid(row=1, column=10, padx=10, pady=5, sticky="WE")

        self.inventory_list = tk.Listbox(self.master, height=30)
        self.inventory_list.grid(row=2, column=0, columnspan=11, padx=10, pady=5, sticky="NSEW")

        self.load_inventory()


    def load_categories(self):
        cursor = self.db_connection.cursor()
        cursor.execute("SELECT name FROM categories")
        categories = [row[0] for row in cursor.fetchall()]
        self.category_combobox['values'] = categories


    def view_categories(self):
        cursor = self.db_connection.cursor()
        cursor.execute("SELECT name FROM categories")
        categories = [row[0] for row in cursor.fetchall()]
        messagebox.showinfo("Категорії", "\n".join(categories))


    def add_category(self):
         new_category = simpledialog.askstring("Додати категорію", "Введіть нову
```

```
категорію:")

        if new_category:
            cursor = self.db_connection.cursor()
                cursor.execute("INSERT INTO categories (name) VALUES (?)",
(new_category,))
            self.db_connection.commit()
            self.load_categories()



    def update_category(self):
        cursor = self.db_connection.cursor()
        cursor.execute("SELECT name FROM categories")
        categories = [row[0] for row in cursor.fetchall()]
        selected_category = simpledialog.askstring("Оновити категорію", "Виберіть
категорію для оновлення:", initialvalue=categories[0])

        if selected_category:
            new_name = simpledialog.askstring("Оновити категорію", "Введіть нову
назву для категорії:", initialvalue=selected_category)

            if new_name:
                cursor.execute("UPDATE categories SET name = ? WHERE name = ?",
(new_name, selected_category))
                self.db_connection.commit()
                self.load_categories()



    def delete_category(self):
        cursor = self.db_connection.cursor()
```

```
        cursor.execute("SELECT name FROM categories")
        categories = [row[0] for row in cursor.fetchall()]
       selected_category = simpledialog.askstring("Видалити категорію", "Виберіть категорію для видалення:", initialvalue=categories[0])

        if selected_category:
            confirm = messagebox.askyesno("Підтвердження", f"Ви впевнені, що хочете видалити категорію '{selected_category}'?")

            if confirm:
                cursor.execute("DELETE FROM categories WHERE name = ?", (selected_category,))
                cursor.execute("DELETE FROM inventory WHERE category_id = (SELECT id FROM categories WHERE name = ?)", (selected_category,))
                self.db_connection.commit()
                self.load_categories()
                self.load_inventory()


    def add_item(self):
        item = self.item_entry.get()
        category = self.category_combobox.get()
        quantity = self.quantity_entry.get()

        if item and category and quantity:
            try:
                quantity = int(quantity)
                now = datetime.now()
                formatted_date = now.strftime("%Y-%m-%d %H:%M:%S")
                cursor = self.db_connection.cursor()
```

```
            cursor.execute("INSERT INTO inventory (item, category_id, quantity, last_updated) VALUES (?, (SELECT id FROM categories WHERE name = ?), ?, ?)", (item, category, quantity, formatted_date))
            self.db_connection.commit()
            self.load_inventory()
            self.item_entry.delete(0, tk.END)
            self.quantity_entry.delete(0, tk.END)
        except ValueError:
            messagebox.showerror("Помилка", "Будь ласка, введіть дійсну кількість.")
    else:
        messagebox.showerror("Помилка", "Будь ласка, заповніть всі поля.")


def update_quantity(self):
    selected_item = self.inventory_list.curselection()

    if selected_item:
        new_quantity = simpledialog.askinteger("Оновити кількість", "Введіть нову кількість:")

        if new_quantity is not None:
            item_id = selected_item[0] + 1
            now = datetime.now()
            formatted_date = now.strftime("%Y-%m-%d %H:%M:%S")
            cursor = self.db_connection.cursor()
            cursor.execute("UPDATE inventory SET quantity = ?, last_updated = ? WHERE id = ?", (new_quantity, formatted_date, item_id))
            self.db_connection.commit()
            self.load_inventory()
```

```
        else:
            messagebox.showerror("Помилка", "Будь ласка, виберіть товар для оновлення.")


    def remove_item(self):
        selected_item = self.inventory_list.curselection()

        if selected_item:
            confirm = messagebox.askyesno("Підтвердження", "Ви впевнені, що хочете видалити цей товар?")

            if confirm:
                item_id = selected_item[0] + 1
                cursor = self.db_connection.cursor()
                cursor.execute("DELETE FROM inventory WHERE id = ?", (item_id,))
                self.db_connection.commit()
                self.load_inventory()
        else:
            messagebox.showerror("Помилка", "Будь ласка, виберіть товар для видалення.")


    def search_item(self):
        search_term = self.search_entry.get()

        if search_term:
            cursor = self.db_connection.cursor()
            cursor.execute("SELECT i.item, c.name, i.quantity, i.last_updated FROM inventory i JOIN categories c ON i.category_id = c.id WHERE i.item LIKE ?", ('%'
```

```
+ search_term + '%',))
            rows = cursor.fetchall()
            self.inventory_list.delete(0, tk.END)

            for row in rows:
                    self.inventory_list.insert(tk.END, f"{row[0]} ({row[1]}): {row[2]} (Останнє оновлення: {row[3]})")
        else:
            self.load_inventory()


    def filter_items(self):
        filter_quantity = self.filter_entry.get()

        if filter_quantity:
            try:
                filter_quantity = int(filter_quantity)
                cursor = self.db_connection.cursor()
                cursor.execute("SELECT i.item, c.name, i.quantity, i.last_updated FROM inventory i JOIN categories c ON i.category_id = c.id WHERE i.quantity >= ?", (filter_quantity,))
                rows = cursor.fetchall()
                self.inventory_list.delete(0, tk.END)

                for row in rows:
                    self.inventory_list.insert(tk.END, f"{row[0]} ({row[1]}): {row[2]} (Останнє оновлення: {row[3]})")
            except ValueError:
                    messagebox.showerror("Помилка", "Будь ласка, введіть дійсну кількість.")
```

```
        else:
            self.load_inventory()


    def export_csv(self):
                filename = filedialog.asksaveasfilename(defaultextension=".csv",
filetypes=[("CSV файли", "*.csv")])

        if filename:
            with open(filename, 'w', newline='') as file:
                writer = csv.writer(file)
                        writer.writerow(["Товар", "Категорія", "Кількість", "Останнє
оновлення"])
                cursor = self.db_connection.cursor()
                cursor.execute("SELECT i.item, c.name, i.quantity, i.last_updated FROM
inventory i JOIN categories c ON i.category_id = c.id")
                rows = cursor.fetchall()

                for row in rows:
                    writer.writerow([row[0], row[1], row[2], row[3]])

                messagebox.showinfo("Успіх", "Інвентар експортовано у CSV файл
успішно.")


    def import_csv(self):
        filename = filedialog.askopenfilename(filetypes=[("CSV файли", "*.csv")])

        if filename:
            with open(filename, 'r') as file:
```

```
            reader = csv.reader(file)
            next(reader)  # Пропустити заголовковий рядок

            for row in reader:
                item, category, quantity, last_updated = row
                cursor = self.db_connection.cursor()
                cursor.execute("INSERT INTO inventory (item, category_id, quantity, last_updated) VALUES (?, (SELECT id FROM categories WHERE name = ?), ?, ?)", (item, category, int(quantity), last_updated))
                self.db_connection.commit()

        self.load_inventory()
            messagebox.showinfo("Успіх", "Інвентар імпортовано з CSV файлу успішно.")


    def show_statistics(self):
        cursor = self.db_connection.cursor()
        cursor.execute("SELECT quantity FROM inventory")
        quantities = [row[0] for row in cursor.fetchall()]

        if quantities:
            average_quantity = statistics.mean(quantities)
            max_quantity = max(quantities)
            min_quantity = min(quantities)
                messagebox.showinfo("Статистика інвентаря", f"Середня кількість: {average_quantity}\nМаксимальна кількість: {max_quantity}\nМінімальна кількість: {min_quantity}")
        else:
                messagebox.showinfo("Статистика інвентаря", "Немає доступних
```

даних.")

```
    def show_graph(self):
        cursor = self.db_connection.cursor()
         cursor.execute("SELECT c.name, SUM(i.quantity) FROM inventory i JOIN
categories c ON i.category_id = c.id GROUP BY c.name")
        data = cursor.fetchall()

        if data:
            categories = [row[0] for row in data]
            quantities = [row[1] for row in data]
            plt.bar(categories, quantities)
            plt.xlabel('Категорії')
            plt.ylabel('Загальна кількість')
            plt.title('Інвентар по категоріях')
            plt.xticks(rotation=45, ha='right')
            plt.tight_layout()
            plt.show()
        else:
            messagebox.showinfo("Графік", "Немає доступних даних для графіку.")


    def backup_database(self):
        now = datetime.now()
        formatted_date = now.strftime("%Y-%m-%d_%H-%M-%S")
        backup_filename = f"inventory_backup_{formatted_date}.db"

        try:
            self.db_connection.close()
```

```
            import shutil
            shutil.copyfile('inventory.db', backup_filename)
            self.db_connection = sqlite3.connect('inventory.db')
              messagebox.showinfo("Резервне копіювання", f"Базу даних успішно збережено як {backup_filename}")

  except Exception as e:
            messagebox.showerror("Помилка", f"Не вдалося зробити резервну копію бази даних: {str(e)}")


    def load_inventory(self):
        self.inventory_list.delete(0, tk.END)
        cursor = self.db_connection.cursor()
          cursor.execute("SELECT i.item, c.name, i.quantity, i.last_updated FROM inventory i JOIN categories c ON i.category_id = c.id")
        rows = cursor.fetchall()

        for row in rows:
          self.inventory_list.insert(tk.END, f"{row[0]} ({row[1]}): {row[2]} (Останнє оновлення: {row[3]})")

def main():
    root = tk.Tk()
    app = InventoryApp(root)
    root.mainloop()

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