Прикарпатський національний університет імені Василя Стефаника
Факультет математики та інформатики
Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНА ДИПЛОМНА РОБОТА

Тема: «Автоматизована система управління складом меблевої продукції»

Виконав: студент IV курсу гр. ІСТ41 ОР
бакалавр
спеціальності 126 "Інформаційні системи та технології"
Галахов Максим Віталійович

Науковий керівник: к.т.н., доц. Превисокова Наталія Володимирівна

Рецензент: к.т.н. Горєлов В.О

м. Івано-Франківськ – 2024

Анотація

Дана дипломна робота присвячена розробці та впровадженню автоматизованої системи управління складськими запасами для меблевої промисловості. Основна мета проекту - спрощення та оптимізація управління запасами меблів на складі за допомогою сучасних інформаційних технологій.

Система побудована за клієнт-серверною архітектурою, що включає серверну частину для управління базою даних та клієнтську частину для взаємодії з користувачем через графічний інтерфейс. Для реалізації бази даних обрана SQLite, яка забезпечує простоту в користуванні та достатню функціональність для зберігання інформації про товари .

Інтерфейс користувача створений за допомогою бібліотеки Tkinter, що дозволяє реалізувати зручний і інтуїтивно зрозумілий графічний інтерфейс. Основні функції системи включають додавання, редагування та видалення товарів, пошук та сортування за різними критеріями, а також обчислення загальної вартості вибраних товарів .

Проведене тестування показало, що система ефективно виконує всі заявлені функції та значно спрощує процес управління складськими запасами меблів. Завдяки своїй простоті і доступності, розроблена система може бути корисною для малого та середнього бізнесу, що працює у сфері виробництва та продажу меблів .

Таким чином, дипломна робота демонструє успішну реалізацію автоматизованої системи управління складом, яка може бути впроваджена на реальних підприємствах для підвищення ефективності їхньої діяльності .

Abstract

This thesis is dedicated to the development and implementation of an automated inventory management system for the furniture industry. The primary goal of the project is to simplify and optimize the management of furniture stock in a warehouse using modern information technologies.

The system is built using a client-server architecture, which includes a server part for managing the database and a client part for user interaction through a graphical interface. SQLite was chosen for the database implementation, ensuring ease of use and sufficient functionality for storing product information .

The user interface is created using the Tkinter library, which allows for the implementation of a convenient and intuitive graphical interface. The main functions of the system include adding, editing, and deleting products, searching and sorting by various criteria, and calculating the total cost of selected products .

Testing has shown that the system effectively performs all declared functions and significantly simplifies the process of managing furniture stock. Due to its simplicity and accessibility, the developed system can be useful for small and medium-sized businesses engaged in furniture production and sales .

Thus, the thesis demonstrates the successful implementation of an automated warehouse management system that can be deployed in real enterprises to increase the efficiency of their operations .

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 8
ВСТУП 9
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 12
1.1 Огляд літератури за темою 12
1.2 Постанова задачі 14
1.3 Аналіз аналогічних рішень задачі 16
Висновки 17
РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ 19
2.1 Обгрунтування вибору технологій для реалізації програми 19
2.2 Архітектура програмного продукту 20
2.3 Модуль логіки бізнес-процесів 25
2.4 Опис вхідних та вихідних даних 27
Висновки до Розділу 2 30
РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ 32
3.1 Опис програмної реалізації 32
3.1.2 Створення графічного інтерфейсу користувача (GUI) 34
3.1.3 Реалізація основних функцій 38
3.1.4 Тестування та перевірка системи 39
3.2 Використання програмного продукту 39
ВИСНОВКИ 43
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 44
ДОДАТКИ 46

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

1. GUI - Graphical User Interface (Графічний Інтерфейс Користувача)
2. DB - Database (База Даних)
3. CRUD - Create, Read, Update, Delete (Створення, Читання, Оновлення, Видалення)
4. SQL - Structured Query Language (Структурована Мова Запитів)
5. IDE - Integrated Development Environment (Інтегроване Середовище Розробки)
6. API - Application Programming Interface (Інтерфейс Програмування Додатків)
7. JSON - JavaScript Object Notation (Формат Обміну Даними)
8. CSV - Comma-Separated Values (Значення, Розділені Комами)
9. MVC - Model-View-Controller (Модель-Представлення-Контролер)
10. HTTP - HyperText Transfer Protocol (Протокол Передачі Гіпертексту)
11. TCP/IP - Transmission Control Protocol/Internet Protocol (Протокол Управління Передачею/Інтернет Протокол)
12. OS - Operating System (Операційна Система)
13. RAM - Random Access Memory (Оперативна Пам'ять)
14. ROM - Read-Only Memory (Постійна Пам'ять)
15. UX - User Experience (Користувацький Досвід)
16. Tkinter - бібліотека для створення графічного інтерфейсу в Python
17. SQLite - легка реляційна база даних, яка вбудовується у додаток
18. Python - мова програмування високого рівня, яка використовується для розробки системи
19. PIP - Python Package Installer (Інсталятор Пакетів Python)
20. PEP - Python Enhancement Proposal (Пропозиція Поліпшення Python)

Опис компонентів програми:

- База Даних (DB) - зберігає інформацію про товари, включаючи їхні коди, назви, ціни, кількість, колір, розмір, форму, компанію, упаковку та бренд.
- Графічний Інтерфейс Користувача (GUI) - забезпечує взаємодію користувача з програмою через візуальні елементи, такі як кнопки, поля введення та таблиці.
- Функціональні Компоненти (CRUD) - забезпечують основні операції з даними: створення, читання, оновлення та видалення записів у базі даних.
- Формати Даних (JSON, CSV) - використовуються для зберігання та обміну даними між різними компонентами системи.

ВСТУП

Автоматизація процесів управління складськими запасами є ключовою складовою ефективного функціонування підприємств у різних галузях, включаючи меблеву промисловість. В даній дипломній роботі представлено розробку системи "Автоматизований меблевий склад", яка призначена для спрощення та оптимізації управління запасами меблів на складі .

Система побудована за клієнт-серверною архітектурою і включає серверну частину для управління базою даних та клієнтську частину для взаємодії з користувачем через графічний інтерфейс. Для реалізації бази даних використано SQLite, що забезпечує простоту в користуванні та достатню функціональність для зберігання інформації про товари .

Інтерфейс користувача створено за допомогою бібліотеки Tkinter, яка дозволяє реалізувати зручний і інтуїтивно зрозумілий графічний інтерфейс. Основні функції системи включають додавання, редагування та видалення товарів, пошук та сортування за різними критеріями, а також обчислення загальної вартості вибраних товарів .

Проведене тестування показало, що система ефективно виконує всі заявлені функції та значно спрощує процес управління складськими запасами меблів. Завдяки своїй простоті і доступності, розроблена система може бути корисною для малого та середнього бізнесу, що працює у сфері виробництва та продажу меблів .

Таким чином, дипломна робота демонструє успішну реалізацію автоматизованої системи управління складом, яка може бути впроваджена на реальних підприємствах для підвищення ефективності їхньої діяльності .

Об'єкт дослідження – управління складськими процесами, яка забезпечує ефективне функціонування складу меблевої компанії. Процес обчислення рейтингу в різних сферах діяльності.

Предмет дослідження – методики та алгоритми обчислення рейтингу, їх аналіз, порівняння та удосконалення.

Мета дослідження

Метою даного дослідження є розробка та впровадження автоматизованої системи управління складськими запасами для меблевої промисловості, що дозволить оптимізувати процеси зберігання, обліку та обробки товарів на складі, підвищити ефективність управління запасами та знизити операційні витрати.

Постанова задач для виконання цієї мети

1. Аналіз існуючих рішень та вимог
    - Провести огляд існуючих автоматизованих систем управління складом, вивчити їхні можливості та обмеження.
    - Визначити основні вимоги до системи, враховуючи специфіку меблевої промисловості та потреби потенційних користувачів.
2. Розробка концепції та архітектури системи
    - Розробити концептуальну модель автоматизованої системи управління складом.
    - Визначити архітектуру системи, обравши відповідну клієнт-серверну модель.
    - Обрати інструменти та технології для реалізації системи, включаючи базу даних SQLite та бібліотеку Tkinter.
3. Реалізація серверної частини системи
    - Розробити структуру бази даних для зберігання інформації про товари.

- Реалізувати серверну частину для управління базою даних, включаючи функції додавання, редагування, видалення та пошуку товарів.

- Надати рекомендації щодо подальшого розвитку та вдосконалення системи, враховуючи отриманий досвід та результати дослідження.

Ці задачі спрямовані на досягнення головної мети дослідження – створення ефективної автоматизованої системи управління складськими запасами для меблевої промисловості.

Досягнення поставленої мети дозволить значно оптимізувати процеси управління складом, знизити операційні витрати, покращити точність обліку товарів і підвищити загальну продуктивність складських операцій у меблевій компанії.

Стан наукової розробки. Науково-дослідна робота базується на вивченні та аналізі наукової літератури, статей, та досліджень в автоматизований склад. Актуальність та значимість проблеми автоматизований склад підтверджується численними публікаціями в наукових журналах та матеріалах конференцій.

Методи дослідження включають аналіз та синтез, порівняльний аналіз, експериментальну перевірку та математичне моделювання.

Практичне значення одержаних результатів полягає в можливості застосування розробленого автоматизованого складу в меблевих компаній для збільшення ефективності та зручності робочих процесів.

Апробація результатів. Результати дослідження були представлені на наукових конференціях та семінарах.

Структура роботи. Дипломна робота складається з вступу, трьох розділів, висновків, списку використаних джерел та додатків. У першому розділі проводиться аналіз предметної області, у другому – проектування програмного

продукту, у третьому – його програмна реалізація. Робота містить таблиці та рисунки, які ілюструють основні аспекти дослідження.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд літератури за темою

Автоматизація складських процесів є важливою складовою ефективного управління підприємством, особливо у галузі виробництва та торгівлі меблями. Сучасні системи управління складом (WMS) дозволяють автоматизувати процеси зберігання, обробки та обліку товарів, що сприяє підвищенню

продуктивності та зниженню операційних витрат. За даними досліджень, впровадження WMS може зменшити витрати на складування до 30% та підвищити точність обліку товарів до 99% .

Використання інформаційних технологій для управління складом дозволяє забезпечити високу швидкість та точність обробки даних, що є особливо важливим у меблевій промисловості, де асортимент товарів є досить різноманітним. Сучасні WMS системи використовують RFID-технології, бар-коди та інші інструменти для автоматичної ідентифікації та відстеження товарів, що значно знижує ймовірність помилок при обліку запасів .

Однією з ключових переваг автоматизованих систем управління складом є можливість інтеграції з іншими інформаційними системами підприємства, такими як ERP (Enterprise Resource Planning) та CRM (Customer Relationship Management). Це дозволяє забезпечити єдиний інформаційний простір для управління всіма бізнес-процесами, що сприяє підвищенню ефективності роботи підприємства в цілому .

Дослідження показують, що впровадження автоматизованих систем управління складом особливо корисне для малих та середніх підприємств, які мають обмежені ресурси для управління складськими запасами. Використання таких систем дозволяє оптимізувати складські процеси, зменшити кількість необхідного персоналу та підвищити швидкість обслуговування клієнтів, що є критичним фактором для успіху в умовах жорсткої конкуренції .

Крім того, сучасні WMS системи підтримують функції аналітики та звітності, що дозволяє керівникам підприємств отримувати актуальні дані про стан складських запасів, аналізувати ефективність роботи складу та приймати обґрунтовані управлінські рішення. Це сприяє підвищенню прозорості бізнес-процесів та покращенню контролю за діяльністю підприємства .

Таким чином, огляд літератури свідчить про високу ефективність автоматизованих систем управління складом у різних галузях, включаючи меблеву промисловість. Впровадження таких систем дозволяє значно підвищити продуктивність, точність обліку та оптимізувати витрати на управління складськими запасами, що є важливим фактором для успішного функціонування підприємства на сучасному ринку .

1.2 Постанова задачі

Розробка системи "Автоматизований меблевий склад" вимагає чіткого визначення мети та задач, які потрібно вирішити для досягнення оптимальної функціональності та ефективності системи. Основна мета цього дослідження полягає в створенні програмного забезпечення, яке забезпечить автоматизацію процесів управління складськими запасами меблів, зменшить витрати на обслуговування складу та підвищить точність обліку товарів .

Першою задачею є аналіз існуючих систем управління складом та визначення ключових функціональних вимог до нової системи. Це включає дослідження сучасних технологій автоматизації складських процесів, таких як використання бар-кодів, RFID-технологій та інтеграція з ERP системами. Аналіз літератури та існуючих рішень на ринку дозволяє визначити оптимальні підходи до розробки та впровадження системи управління складом .

Другою важливою задачею є проектування архітектури системи, яка включає серверну частину для управління базою даних та клієнтську частину для взаємодії з користувачем через графічний інтерфейс. Вибір відповідних інструментів та технологій для реалізації бази даних, таких як SQLite, є критично важливим для забезпечення надійності та ефективності системи .

Третя задача полягає в розробці зручного та інтуїтивно зрозумілого інтерфейсу користувача за допомогою бібліотеки Tkinter. Це включає створення функціоналу для додавання, редагування та видалення товарів, а також пошуку

та сортування за різними критеріями. Важливо забезпечити, щоб інтерфейс був максимально простим у використанні для співробітників складу, що сприятиме швидкому освоєнню системи .

Четвертою задачею є впровадження функцій аналітики та звітності, які дозволять керівникам підприємств отримувати актуальні дані про стан складських запасів, аналізувати ефективність роботи складу та приймати обґрунтовані управлінські рішення. Це включає розробку модулів для генерації звітів та візуалізації даних, що дозволить підвищити прозорість бізнес-процесів .

Остання задача полягає в тестуванні та впровадженні системи на реальних підприємствах. Проведення комплексного тестування допоможе виявити та виправити можливі недоліки системи, а також адаптувати її до конкретних потреб користувачів. Впровадження системи в реальних умовах дозволить оцінити її ефективність та корисність для підприємств меблевої промисловості .

1.3 Аналіз аналогічних рішень задачі

Аналіз існуючих рішень у сфері автоматизації складських процесів дозволяє виявити сильні та слабкі сторони різних підходів і технологій, що використовуються. Це є важливим етапом у процесі розробки власної системи, оскільки дає змогу уникнути поширених помилок та інтегрувати перевірені на практиці методи.

Одним із найбільш відомих і широко використовуваних рішень є SAP Extended Warehouse Management (SAP EWM). Ця система забезпечує повний контроль над усіма процесами складу, включаючи прийом товарів, їх зберігання, комплектування замовлень та відправку. SAP EWM дозволяє інтегрувати управління складом з іншими бізнес-процесами підприємства, такими як планування виробництва та управління запасами. Однак висока вартість впровадження та складність налаштувань можуть бути значними бар'єрами для малого та середнього бізнесу .

Іншим популярним рішенням є Oracle Warehouse Management Cloud (WMS), яке надає функції управління складом через хмарні технології. Це дозволяє знизити витрати на IT-інфраструктуру та забезпечує гнучкість у масштабуванні системи. Oracle WMS включає інструменти для автоматизації основних складських операцій, аналітики та звітності, що допомагає покращити ефективність управління складом. Однак для підприємств, які не готові до повного переходу на хмарні сервіси, це може бути не найкращим вибором .

Варто також відзначити рішення від компанії Infor - Infor CloudSuite WMS. Ця система пропонує широкий спектр функцій для управління складськими операціями, включаючи інтеграцію з ERP системами, підтримку мобільних пристроїв та можливості для оптимізації складських процесів. Infor CloudSuite WMS підходить для різних типів підприємств, але її впровадження також може вимагати значних фінансових та часових ресурсів .

Серед рішень для малого та середнього бізнесу варто виділити Zoho Inventory. Ця система пропонує основні функції для управління складом, включаючи облік товарів, управління замовленнями та аналітику. Zoho Inventory інтегрується з іншими продуктами Zoho, що забезпечує цілісність бізнес-процесів. Вартість цього рішення є значно нижчою порівняно з великими ERP системами, що робить його доступним для невеликих підприємств .

Ще одним цікавим рішенням є Fishbowl Inventory, яке спеціалізується на інтеграції з QuickBooks для малого та середнього бізнесу. Fishbowl Inventory надає функції управління запасами, виробництвом та замовленнями, що дозволяє підприємствам ефективно контролювати свої складські процеси. Проте, залежність від QuickBooks може бути обмеженням для підприємств, які використовують інші бухгалтерські системи .

Також варто розглянути OpenWMS, систему з відкритим кодом, яка пропонує базові функції управління складом без значних витрат на ліцензування. OpenWMS є гнучким рішенням, яке можна налаштувати відповідно до потреб конкретного підприємства. Однак, використання цієї системи може вимагати додаткових ресурсів на налаштування та підтримку .

На основі аналізу існуючих рішень можна зробити висновок, що жодна з них не є універсальною та має свої переваги й недоліки. Розробка власної системи "Автоматизований меблевий склад" дозволяє врахувати специфічні потреби меблевого бізнесу та оптимізувати процеси управління складом з урахуванням кращих практик та технологій. Це включає інтеграцію з існуючими бізнес-системами, використання сучасних інструментів для автоматизації та аналітики, а також забезпечення гнучкості та масштабованості системи.

Висновки

У рамках першого розділу проведено аналіз сучасного стану автоматизації управління складськими процесами у меблевій промисловості. Оглянуто різноманітні джерела літератури та аналогічні рішення, що використовуються в інших галузях.

За результатами огляду літератури виявлено, що автоматизація управління складом є актуальною проблемою для підприємств меблевої промисловості. Існуючі системи мають свої переваги та обмеження, що вимагає розробки спеціалізованої системи для цієї галузі.

Постановка задачі полягає у створенні системи "Автоматизований меблевий склад", яка забезпечить ефективне управління запасами меблів на складі. Основні функції системи включають додавання, редагування та видалення товарів, пошук та сортування за різними критеріями, а також обчислення загальної вартості вибраних товарів.

Аналіз аналогічних рішень показав, що існуючі системи, такі як SAP EWM, Oracle WMS, Infor CloudSuite WMS та інші, мають свої переваги, але не завжди відповідають потребам малого та середнього бізнесу в меблевій галузі. Розробка власної системи дозволить врахувати специфічні вимоги цієї галузі та забезпечити оптимальні рішення для управління складом.

Отже, на основі проведеного аналізу можна зробити висновок, що розробка системи "Автоматизований меблевий склад" є актуальною та доцільною задачею, яка вирішує проблеми управління складськими процесами у меблевій промисловості.

РОЗДІЛ 2. ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

2.1 Обгрунтування вибору технологій для реалізації програми

Для розробки системи "Автоматизований меблевий склад" було обрано кілька ключових технологій: Python, SQLite та бібліотека Tkinter. Ці технології були обрані з урахуванням їх переваг, зручності використання та відповідності вимогам проекту.

Python був обраний як основна мова програмування для розробки системи завдяки своїй простоті, зручності та широким можливостям. Це високорівнева мова програмування, яка підтримує кілька парадигм, включаючи об'єктно-орієнтоване, процедурне та функціональне програмування .

Python надає багатий набір бібліотек та модулів, що дозволяє швидко розробляти складні додатки. Наприклад, для побудови графічного інтерфейсу користувача використовується бібліотека Tkinter, яка є стандартною бібліотекою для створення GUI-додатків у Python .

Окрім цього, Python має активну спільноту розробників, яка забезпечує підтримку та розвиток мови. Це дозволяє швидко знаходити рішення для різних проблем та отримувати допомогу у разі потреби. Завдяки своїй популярності, Python також має велику кількість документації та навчальних матеріалів, що спрощує процес навчання та розробки .

SQLite був обраний як система керування базами даних (СКБД) для проекту завдяки своїй простоті, легкості вбудовування та високій продуктивності. SQLite - це вбудована СКБД, яка зберігає всі дані у одному файлі, що робить її ідеальною для невеликих та середніх додатків, таких як "Автоматизований меблевий склад" .

SQLite не вимагає налаштування сервера та адміністрування, що значно спрощує процес розгортання та експлуатації системи. Вона також підтримує стандарт SQL, що дозволяє використовувати всі можливості реляційних баз даних для зберігання та обробки даних .

Крім того, SQLite є дуже швидкою та ефективною СКБД. Вона забезпечує високу швидкість виконання запитів та обробки даних, що є важливим фактором для забезпечення продуктивності системи .

Для розробки графічного інтерфейсу користувача було обрано бібліотеку Tkinter. Tkinter є стандартною бібліотекою для створення GUI-додатків у Python, яка забезпечує простоту та зручність у використанні .

Tkinter дозволяє швидко створювати інтуїтивно зрозумілий та зручний інтерфейс користувача. Вона підтримує широкий набір віджетів, таких як кнопки, текстові поля, таблиці та інші елементи управління,

2.2 Архітектура програмного продукту

Архітектура програмного продукту "Автоматизований меблевий склад" визначає його структурні компоненти, зв'язки між ними, а також принципи та підходи, за якими ці компоненти взаємодіють. У цьому розділі буде детально розглянуто основні аспекти архітектури системи, включаючи її компоненти, модулі та їх взаємодію.

Клієнт-серверна архітектура

Система "Автоматизований меблевий склад" побудована за клієнт-серверною архітектурою, яка забезпечує розподіл обчислювальних ресурсів між сервером і клієнтом. Це дозволяє підвищити продуктивність системи та забезпечити гнучкість в її масштабуванні.

Серверна частина відповідає за управління базою даних, зберігання та обробку даних. Вона реалізована за допомогою SQLite, що забезпечує ефективне зберігання інформації про товари на складі. Сервер також відповідає за виконання SQL-запитів, що надходять від клієнтської частини.

Клієнтська частина забезпечує взаємодію користувача з системою через графічний інтерфейс. Вона реалізована за допомогою бібліотеки Tkinter, яка надає зручні інструменти для створення інтуїтивно зрозумілих інтерфейсів користувача.

Компоненти системи

Система складається з наступних основних компонентів:

1. Інтерфейс користувача - реалізований за допомогою Tkinter. Він надає користувачу можливість взаємодіяти з системою через різні елементи управління, такі як кнопки, текстові поля, таблиці тощо.

2. Модуль управління базою даних - відповідає за взаємодію з базою даних SQLite. Він забезпечує виконання основних операцій з базою даних, таких як додавання, редагування, видалення та пошук даних.

3. Модуль логіки бізнес-процесів - реалізує основні бізнес-процеси системи, такі як обробка даних, виконання запитів, обчислення загальної вартості товарів тощо.

Взаємодія компонентів

Взаємодія між компонентами системи відбувається наступним чином:

1. Користувач взаємодіє з інтерфейсом користувача, вводячи дані та викликаючи різні функції системи.

2. Інтерфейс користувача передає запити до модуля логіки бізнес-процесів, який обробляє ці запити та виконує необхідні дії.

3. Для виконання операцій з базою даних модуль логіки бізнес-процесів взаємодіє з модулем управління базою даних, який забезпечує виконання відповідних SQL-запитів.

4. Результати виконання запитів повертаються до інтерфейсу користувача, де вони відображаються користувачу у зручному вигляді.

Модулі та їх функціональність

Основні модулі системи та їх функціональність описані нижче:

1. Інтерфейс користувача:

    - Введення даних про товари (код, назва, ціна, кількість, колір, розмір, форма, компанія, пакування, бренд).
    - Пошук та сортування товарів за різними критеріями.
    - Відображення списку товарів у вигляді таблиці.
    - Додавання, редагування та видалення товарів.
    - Обчислення загальної вартості вибраних товарів.

2. Модуль управління базою даних:

    - Підключення до бази даних SQLite.
    - Виконання SQL-запитів для додавання, редагування, видалення та пошуку даних.
    - Забезпечення цілісності та безпеки даних.

3. Модуль логіки бізнес-процесів:

    - Обробка запитів від інтерфейсу користувача.

- Взаємодія з модулем управління базою даних для виконання операцій з даними.
- Виконання обчислень та інших бізнес-логічних операцій.

Діаграма архітектури

Для наочності можна представити діаграму архітектури системи, яка показує взаємодію між її компонентами.

Архітектура системи "Автоматизований меблевий склад" базується на клієнт-серверній моделі, що забезпечує ефективну взаємодію між її компонентами. Використання Python, SQLite та Tkinter дозволяє створити надійну, продуктивну та зручну систему для управління складськими запасами меблів. Обрані технології та архітектурні рішення забезпечують гнучкість, масштабованість та легкість у використанні, що є ключовими факторами для успішної реалізації проекту.

Опис компонентів програми

Програмний продукт "Автоматизований меблевий склад" складається з кількох основних компонентів, кожен з яких виконує певні функції для забезпечення загальної працездатності системи. У цьому розділі буде детально розглянуто кожен з компонентів програми, їх функціональність та взаємодію між собою.

1. Інтерфейс користувача

   Інтерфейс користувача є основним компонентом, через який відбувається взаємодія користувача з системою. Він реалізований за допомогою бібліотеки Tkinter, що дозволяє створювати зручні та інтуїтивно зрозумілі графічні інтерфейси.

   Основні функції інтерфейсу користувача:

- Введення даних про товари: Користувач має можливість вводити інформацію про товари, такі як код, назва, ціна, кількість, колір, розмір, форма, компанія, пакування, бренд.

- Відображення списку товарів: Всі товари відображаються у вигляді таблиці, що дозволяє користувачу легко переглядати інформацію про них.
- Пошук та сортування товарів: Користувач може здійснювати пошук та сортування товарів за різними критеріями, такими як код, назва, ціна тощо.
- Додавання, редагування та видалення товарів: Інтерфейс надає можливість додавання нових товарів, редагування існуючих та видалення непотрібних товарів.
- Обчислення загальної вартості товарів: Користувач може обрати кілька товарів та обчислити їх загальну вартість на основі ціни та кількості.

2. Модуль управління базою даних

Модуль управління базою даних відповідає за взаємодію з базою даних SQLite. Він забезпечує виконання основних операцій з базою даних, таких як додавання, редагування, видалення та пошук даних.

Основні функції модуля управління базою даних:

- Підключення до бази даних: Модуль забезпечує підключення до бази даних SQLite, де зберігається вся інформація про товари.
- Виконання SQL-запитів: Модуль виконує SQL-запити для додавання нових товарів, редагування інформації про існуючі товари, видалення товарів та пошуку даних у базі.
- Забезпечення цілісності та безпеки даних: Модуль гарантує, що всі операції з базою даних виконуються коректно, з дотриманням всіх правил цілісності даних.

2.3 Модуль логіки бізнес-процесів

Модуль логіки бізнес-процесів реалізує основні бізнес-процеси системи, такі як обробка даних, виконання запитів та обчислення загальної вартості товарів.

Основні функції модуля логіки бізнес-процесів:

- Обробка запитів від інтерфейсу користувача: Модуль отримує запити від інтерфейсу користувача та виконує необхідні дії для їх обробки.
- Взаємодія з модулем управління базою даних: Модуль передає запити до модуля управління базою даних та отримує результати їх виконання.
- Виконання обчислень та інших бізнес-логічних операцій: Модуль виконує всі необхідні обчислення, такі як обчислення загальної вартості товарів, та інші бізнес-логічні операції.

Взаємодія компонентів

Взаємодія між компонентами системи відбувається наступним чином:

1. Інтерфейс користувача: Користувач взаємодіє з інтерфейсом користувача, вводячи дані та викликаючи різні функції системи. Інтерфейс користувача передає запити до модуля логіки бізнес-процесів.
2. Модуль логіки бізнес-процесів: Модуль логіки бізнес-процесів обробляє запити від інтерфейсу користувача та передає їх до модуля управління базою даних для виконання необхідних операцій. Після виконання операцій модуль логіки бізнес-процесів отримує результати та передає їх назад до інтерфейсу користувача.
3. Модуль управління базою даних: Модуль управління базою даних виконує SQL-запити для додавання, редагування, видалення та пошуку даних у базі даних. Він забезпечує цілісність та безпеку даних, а також передає результати виконання запитів до модуля логіки бізнес-процесів.

2.4 Опис вхідних та вихідних даних

Програмний продукт "Автоматизований меблевий склад" працює з різноманітними типами даних, що включають інформацію про товари, запити користувачів та результати обробки цих запитів. Вхідні та вихідні дані відіграють ключову роль у функціонуванні системи, забезпечуючи її здатність здійснювати операції додавання, редагування, видалення, пошуку та сортування товарів, а також обчислення загальної вартості товарів.

Вхідні дані

Вхідні дані — це інформація, яку вводить користувач або отримує система від інших компонентів для подальшої обробки. Вхідні дані в нашій системі включають:

1. Дані про товари:
    - Код товару (code): Унікальний ідентифікатор товару, що використовується для його однозначної ідентифікації у системі.
    - Назва товару (name): Текстовий рядок, що описує найменування товару.
    - Ціна (price): Вартість одиниці товару.
    - Кількість (quantity): Кількість одиниць товару, наявних на складі.
    - Колір (color): Текстовий рядок, що описує колір товару.
    - Розмір (size): Текстовий рядок або числове значення, що описує розмір товару.

- Форма (shape): Текстовий рядок, що описує форму товару.
- Компанія (company): Назва компанії-виробника або постачальника товару.
- Пакування (package): Тип пакування товару.
- Бренд (brand): Бренд або марка товару.

2. Запити користувача:

- Додавання товару: Запит, що містить дані про новий товар, який потрібно додати до бази даних.
- Редагування товару: Запит на зміну існуючої інформації про товар.
- Видалення товару: Запит на видалення товару з бази даних за його унікальним кодом.
- Пошук товару: Запит на пошук товару за певними критеріями, такими як код, назва, ціна тощо.
- Сортування товарів: Запит на сортування списку товарів за певними атрибутами.
- Обчислення загальної вартості: Запит на обчислення загальної вартості вибраних товарів.

Вихідні дані

Вихідні дані — це результати обробки вхідних даних, які система відображає користувачу або використовує для подальших обчислень. Вихідні дані в нашій системі включають:

1. Список товарів:

- Відображення списку всіх товарів, що зберігаються на складі, у вигляді таблиці. Кожен запис у таблиці містить усі атрибути товару: код, назва, ціна, кількість, колір, розмір, форма, компанія, пакування, бренд.

2. Результати пошуку та сортування:

    - Система відображає список товарів, що відповідають критеріям пошуку, або відсортований список товарів відповідно до обраного критерію.

3. Підтвердження операцій:

    - Повідомлення про успішне додавання, редагування або видалення товару. Це можуть бути спливаючі вікна або текстові повідомлення в інтерфейсі користувача.

4. Обчислена загальна вартість:

    - Результат обчислення загальної вартості вибраних товарів, що відображається користувачу. Це значення отримується шляхом множення ціни кожного товару на його кількість та сумування отриманих результатів.

Висновки до Розділу 2

У другому розділі дипломної роботи було детально розглянуто процес проектування програмного продукту "Автоматизований меблевий склад". Було обґрунтовано вибір технологій для реалізації програми, визначено архітектуру програмного продукту та описано компоненти, з яких складається система, а також вхідні та вихідні дані, з якими вона працює.

По-перше, було обґрунтовано вибір мови програмування Python та бібліотеки Tkinter для створення графічного інтерфейсу користувача. Python був обраний через його простоту у використанні, розширені можливості для роботи з базами даних, а також активну спільноту розробників та велику кількість бібліотек, що значно прискорює процес розробки програмного забезпечення. Tkinter забезпечує зручність у створенні графічних інтерфейсів, що дозволяє користувачам легко взаємодіяти з системою.

По-друге, було визначено архітектуру системи, яка побудована за клієнт-серверним принципом. Серверна частина відповідає за управління базою даних, тоді як клієнтська частина забезпечує взаємодію з користувачем через графічний інтерфейс. Така архітектура забезпечує масштабованість та гнучкість системи, дозволяючи легко додавати нові функції та адаптуватися до змін у вимогах користувачів.

По-третє, описано компоненти системи, що включають модулі для додавання, редагування, видалення товарів, пошуку та сортування, а також обчислення загальної вартості вибраних товарів. Кожен компонент виконує специфічну роль, що забезпечує чіткість та структурованість системи.

Нарешті, детально описано вхідні та вихідні дані, з якими працює система. Вхідні дані включають інформацію про товари та запити користувачів на різні операції, тоді як вихідні дані представлені у вигляді результатів цих операцій, таких як відображення списку товарів, результати пошуку та сортування, підтвердження операцій та обчислена загальна вартість товарів. Правильне оброблення вхідних та вихідних даних є критично важливим для забезпечення коректної роботи системи та високого рівня задоволеності користувачів.

Таким чином, у другому розділі було створено міцну основу для подальшої реалізації та впровадження системи "Автоматизований меблевий склад", що дозволить значно спростити та оптимізувати процеси управління складськими запасами у меблевій промисловості.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Опис програмної реалізації

У цьому розділі детально розглянуто процес реалізації програмного забезпечення "Автоматизований меблевий склад", що включає створення бази даних, розробку графічного інтерфейсу користувача (GUI), а також реалізацію основних функцій системи, таких як додавання, редагування, видалення товарів, пошук та сортування, і обчислення загальної вартості вибраних товарів.

3.1.1 База даних SQLite

Для створення бази даних у програмі "Автоматизований меблевий склад" було обрано SQLite, що є вбудованою легковаго рішення для зберігання структурованих даних. SQLite було вибрано через його простоту, швидкість та ефективність у використанні, а також через його здатність працювати без необхідності окремого серверу баз даних, що полегшує розгортання програми.

У програмі створена одна таблиця з назвою products, яка містить інформацію про товари на складі. Кожен товар характеризується рядком даних, що включає унікальний код, назву, ціну, кількість, колір, розмір, форму, компанію-виробника, тип пакування та бренд товару.

Структура таблиці products:

code (INTEGER): унікальний ідентифікатор товару;

name (TEXT): назва товару;

price (REAL): ціна товару;

quantity (INTEGER): кількість товару;

color (TEXT): колір товару;

size (TEXT): розмір товару;

shape (TEXT): форма товару;

company (TEXT): компанія-виробник;

package (TEXT): тип пакування товару;

brand (TEXT): бренд товару.

Колонка code була обрана як первинний ключ (PRIMARY KEY), що забезпечує унікальність ідентифікаторів товарів і дозволяє ефективно виконувати операції пошуку та сортування. Використання текстових полів для деяких атрибутів (наприклад, колір, розмір, форма, компанія, тип пакування, бренд) дозволяє зберігати різноманітну інформацію та дозволяє користувачам здійснювати різні запити до бази даних.

Така структура бази даних забезпечує можливість ефективного управління складськими запасами, швидкої інтеграції нових товарів та зручного взаємодії з користувачем через графічний інтерфейс програми.3.1.2 Створення графічного інтерфейсу користувача (GUI)

Для забезпечення зручної взаємодії користувача з системою було обрано бібліотеку Tkinter. Tkinter є стандартною бібліотекою для створення GUI в мові програмування Python і забезпечує широкий набір елементів інтерфейсу, таких як кнопки, текстові поля, таблиці та інші віджети.

Головне вікно програми містить таблицю, в якій відображаються всі товари, а також набір кнопок для виконання основних операцій. Таблиця налаштовується таким чином, щоб її колонки автоматично підлаштовувалися під розміри вікна, забезпечуючи зручне відображення даних.

Для створення бази даних та таблиці products у програмі використовується мова запитів SQL (Structured Query Language). При запуску програми спочатку перевіряється наявність бази даних. Якщо база даних відсутня, то вона автоматично створюється за допомогою наступного SQL запиту:

```
CREATE TABLE IF NOT EXISTS products (
    code INTEGER PRIMARY KEY,
    name TEXT,
    price REAL,
    quantity INTEGER,
```

```
    color TEXT,
    size TEXT,
    shape TEXT,
    company TEXT,
    package TEXT,
    brand TEXT
);
```

Цей запит створює таблицю products з необхідними полями. Параметр code виступає як первинний ключ (PRIMARY KEY), що забезпечує унікальність ідентифікаторів товарів. Колонки name, price, quantity, color, size, shape, company, package, та brand призначені для зберігання відповідних атрибутів товарів.

Після створення таблиці база даних готова до зберігання інформації про товари. До цієї бази даних можна звертатися через SQL-запити для додавання, редагування, видалення, пошуку та сортування товарів, а також для виконання різних аналітичних операцій.

SQLite було обрано для цієї програми через його простоту у використанні та легкість вбудовання в програмне забезпечення. Він не потребує окремого серверу баз даних та може працювати локально на комп'ютері користувача. Такий підхід спрощує розгортання програми та забезпечує швидкий доступ до даних навіть при великій кількості записів.

3.1.2 Створення графічного інтерфейсу користувача (GUI)

Для створення графічного інтерфейсу користувача (GUI) використовувалася бібліотека Tkinter, яка є стандартним інструментом для розробки GUI в Python. Tkinter була обрана через свою простоту в освоєнні, стабільність та широкі можливості для створення різноманітних віконних програм.

1. Вікно програми: Головне вікно програми містить меню з основними функціями програми (додавання, редагування, видалення, пошук, сортування, обчислення загальної вартості) та поле для відображення списку товарів.

2. Функціональні кнопки: Для кожної основної операції (додавання, редагування, видалення) створюються відповідні кнопки у головному вікні програми. При кліку на ці кнопки виконуються відповідні функції.

3. Вікна для введення даних: При додаванні нового товару або редагуванні існуючого відкриваються додаткові вікна для введення атрибутів товару. В цих вікнах користувач вводить необхідну інформацію, яка потім зберігається у базі даних.

4. Поле для відображення списку товарів: У головному вікні програми розміщується поле, в якому відображається список товарів з бази даних. Цей список може бути відсортований та оновлюється після кожної операції з даними.

5. Меню з основними функціями: Меню програми містить основні функції, такі як додавання, редагування, видалення, пошук, сортування, обчислення загальної вартості. Користувач може вибрати потрібну функцію з цього меню для виконання певної операції.

6. Інформаційні вікна: Додатково створюються вікна для відображення інформації про результати виконання операцій або повідомлення про помилки. Ці вікна використовуються для взаємодії з користувачем та надання зрозумілої зворотної інформації. Пошукове вікно: Для зручного пошуку товарів за різними критеріями створюється окреме вікно, де користувач може ввести параметри пошуку, такі як назва товару, код, ціна тощо. Після введення критеріїв пошуку програма відображає результати у відповідному полі.

7. Список вибраних товарів: Для обчислення загальної вартості вибраних товарів створюється додаткове вікно, де користувач може вибрати товари, які йому потрібно врахувати. Після вибору програма обчислює суму та відображає її користувачеві.

8. Коректура розмірів елементів: Всі елементи інтерфейсу відповідають розмірам вікна програми, що забезпечує оптимальне використання простору та зручну роботу з програмою на різних пристроях та дисплеях.

9. Адаптивність: Графічний інтерфейс програми адаптується до різних роздільних здатностей екранів, що дозволяє користувачам з різними пристроями та налаштуваннями використовувати програму без проблем.

10. Зручний дизайн: Елементи інтерфейсу створені з урахуванням простоти та зручності використання. Вони мають зрозумілі підказки та інструкції для користувача, що сприяє зниженню часу на ознайомлення з програмою.

11. Модульність: Кожен компонент графічного інтерфейсу розроблений як окремий модуль, що дозволяє легко модифікувати або розширювати програму в майбутньому без необхідності переробки всього коду.

12. Кросплатформенність: Tkinter є кросплатформовою бібліотекою, тому розроблений за її допомогою графічний інтерфейс може працювати на різних операційних системах, включаючи Windows, macOS та Linux.

13. В результаті, створений графічний інтерфейс користувача забезпечує зручну та ефективну взаємодію з програмою "Автоматизований меблевий склад". Він має всі необхідні функції для управління товарами на складі та дозволяє користувачам легко виконувати потрібні операції з меблями.

3.1.3 Реалізація основних функцій

Додавання товару

Функція додавання товару реалізована через форму введення, що дозволяє користувачу ввести всі необхідні дані про товар. Після заповнення форми користувач натискає кнопку "Додати", і новий товар зберігається в базі даних. При цьому таблиця в головному вікні оновлюється автоматично, щоб відобразити новий запис.

Редагування товару

Редагування товару здійснюється через подвійну натискання на рядок таблиці або через натискання на кнопку "Редагувати". Відкривається форма введення з попередньо заповненими даними обраного товару, що дозволяє користувачу внести необхідні зміни. Після редагування даних користувач натискає кнопку "Зберегти", і зміни зберігаються в базі даних.

Видалення товару

Видалення товару здійснюється через натискання на кнопку "Видалити" після вибору товару в таблиці. Користувачеві пропонується підтвердити видалення, щоб уникнути випадкового видалення даних. Після підтвердження товар видаляється з бази даних, а таблиця оновлюється.

Пошук та сортування товарів

Функції пошуку та сортування дозволяють користувачеві швидко знаходити та упорядковувати товари за різними критеріями. Пошук може здійснюватися за всіма атрибутами товарів, такими як код, назва, ціна, кількість, колір, розмір, форма, компанія, пакування та бренд. Це забезпечує високу гнучкість у використанні системи. Сортування товарів дозволяє упорядковувати їх за будь-яким з доступних атрибутів.

Обчислення загальної вартості вибраних товарів

Ця функція дозволяє користувачам вибрати кілька товарів зі списку та обчислити їх загальну вартість. Функція множить кількість кожного вибраного товару на його ціну та підсумовує результати. Це корисно для розрахунку вартості замовлень або для оцінки загальної вартості товарів на складі.

3.1.4 Тестування та перевірка системи

Тестування системи є критично важливим етапом розробки, що забезпечує надійність та коректність роботи програмного забезпечення. Було проведено ретельне тестування всіх функцій системи, включаючи додавання, редагування та видалення товарів, пошук та сортування, а також обчислення загальної вартості вибраних товарів.

Кожна функція була протестована з різними наборами даних, щоб переконатися у її стабільності та коректності. Наприклад, функція додавання товарів була перевірена на можливість введення некоректних даних, таких як від'ємні значення або нечислові символи в полях, де очікуються числа. Функція пошуку була протестована на здатність знаходити товари за різними критеріями, а функція сортування – на коректне упорядкування товарів за всіма доступними атрибутами.

3.2 Використання програмного продукту

Реалізація основних функцій програмного продукту "Автоматизований меблевий склад" включає в себе широкий спектр операцій з управлінням товарами на складі. Нижче подано опис основних функцій програми:

1. Функція додавання товарів є однією з основних операцій. Користувач має можливість додавати нові товари до бази даних, заповнюючи відповідні поля у формі.

Після вибору опції "Додати товар" користувачу відкривається форма, де він може ввести всю необхідну інформацію про новий товар. Серед основних полів для заповнення можуть бути:

1. Назва товару: Користувач вводить назву товару, щоб ідентифікувати його в системі.

2. Унікальний код товару: Код, який однозначно ідентифікує товар в системі, допомагає унікально відокремлювати товари один від одного.

3. Ціна товару: Вартість одиниці товару.

4. Кількість: Кількість одиниць товару, які додаються на склад.

5. Колір, розмір, форма тощо: Інші характеристики товару, які можуть бути важливими для управління та пошуку.

Крім того, можуть бути додаткові поля для введення інформації, такі як виробник, постачальник, дата придбання тощо, в залежності від конкретних потреб системи.

Після введення всієї необхідної інформації користувач натискає кнопку "Додати", і новий товар додається до бази даних. Система може також проводити перевірку коректності введених даних, щоб уникнути помилок та забезпечити консистентність даних у базі.

2. Функція редагування товарів дозволяє користувачеві вносити зміни до існуючої інформації про товари в базі даних. Це важлива операція, яка дозволяє підтримувати актуальність та точність даних про товари на складі.

Під час вибору опції "Редагувати товар" користувачу відображається список існуючих товарів, які він може відредагувати. Після вибору конкретного товару відкривається форма, де користувач може переглянути та змінити всі його атрибути.

У формі редагування можуть бути представлені всі атрибути товару, включаючи назву, код, ціну, кількість, колір, розмір, форму та інші характеристики. Користувач може змінити будь-яке поле за необхідності.

Після внесення потрібних змін користувач натискає кнопку "Зберегти", і відредаговані дані зберігаються в базі даних. Система може також проводити перевірку на коректність введених даних та їх відповідність обмеженням, що встановлені для кожного атрибуту.

Важливою функціональністю є також можливість видалення товарів. Користувач може вибрати товар, який він бажає видалити, і після підтвердження операції вибраний товар видаляється з бази даних.

Ця функція дозволяє користувачам легко управляти асортиментом товарів на складі, забезпечуючи актуальну та достовірну інформацію про наявні товари.

3. Видалення товарів: Для видалення товару користувач вибирає запис про товар і натискає кнопку "Видалити". Після підтвердження операції товар видаляється з бази даних.

4. Функція пошуку товарів дозволяє користувачам знаходити певні товари за різними критеріями швидко та ефективно. Це важлива операція, яка допомагає знаходити потрібні товари серед великої кількості інформації про складський асортимент

У програмі реалізовано можливість пошуку за різними атрибутами товару, такими як назва, код, ціна, кількість, колір, розмір, форма, виробник тощо. Користувач може обирати один або кілька критеріїв пошуку для точного визначення потрібного товару.

Після введення параметрів пошуку користувач натискає кнопку "Пошук", і система аналізує базу даних, щоб знайти товари, які відповідають заданим критеріям. Результати пошуку відображаються у вигляді списку товарів, які задовольняють введені критерії.

Користувач може переглядати детальну інформацію про знайдені товари, таку як їх атрибути та характеристики, а також візуально оцінювати, наскільки вони відповідають його потребам.

Ця функція дозволяє ефективно виконувати пошук необхідних товарів на складі, що робить процес управління запасами меблів більш зручним та продуктивним.

5. Функція сортування дозволяє користувачам впорядковувати список товарів за певними критеріями для зручності навігації та аналізу асортименту.

У програмі реалізовано можливість сортування товарів за різними параметрами, такими як назва, ціна, кількість, колір, розмір, форма тощо. Користувач може обирати один з доступних критеріїв сортування та вказувати напрямок сортування: за зростанням або за спаданням.

Після вибору параметрів сортування користувач активує відповідну функцію, і система проводить впорядкування товарів у відповідності з вибраними критеріями. Результати сортування відображаються у вигляді відсортованого списку товарів, де перший елемент відповідає найменшому або найбільшому значенню вибраного параметра.

Ця функція надає користувачам зручний спосіб впорядкувати складський асортимент за їхніми потребами та вимогами. Вона допомагає покращити організацію та швидкість доступу до інформації про товари, сприяючи ефективному управлінню запасами меблів.

6. Розрахунок загальної вартості в системі "Автоматизований меблевий склад" виконується шляхом обчислення суми вартості всіх вибраних користувачем товарів на основі їхньої ціни та кількості. Формула для обчислення загальної вартості товарів має вигляд:

Загальна вартість=∑*i*=1*n*(ціна*i*×кількість*i*)Загальна вартість=∑*i*=1*n*(ціна*i* ×кількість*i*)

Де:

- *nn* - кількість товарів;
- ціна*i*ціна*i* - ціна одиниці товару *ii*;
- кількість*i*кількість*i* - кількість одиниць товару *ii*.

Для обчислення загальної вартості програма пройде по кожному вибраному користувачем товару, перемножить його ціну на кількість, а потім додаст результати обчислень для всіх товарів. Таким чином, отримається загальна вартість у вибраній валюті або одиницях виміру.

Ця функція дозволяє користувачам швидко та зручно обчислити вартість свого замовлення або вибраних товарів, спрощуючи процес розрахунків та допомагаючи контролювати витрати.

7. Повернення до початкового списку: Під час пошуку або сортування товарів користувач може легко повернутися до початкового списку товарів, натиснувши відповідну кнопку "Повернутися назад".

Ці функції дозволяють користувачеві зручно управляти товарами на меблевому складі, швидко знаходити необхідну інформацію та ефективно виконувати різноманітні операції з товарами.

ВИСНОВКИ

У ході даної дипломної роботи було розроблено та детально описано систему "Автоматизований меблевий склад". Починаючи з аналізу предметної області, де було вивчено літературні джерела та проведено аналіз аналогічних рішень, а завершуючи програмною реалізацією системи, було досягнуто наступних результатів.

У процесі аналізу було виявлено актуальність проблеми управління складськими запасами в меблевій промисловості та обґрунтовано необхідність автоматизації цих процесів. Також були проаналізовані існуючі рішення та виявлені їхні переваги та недоліки.

У другому розділі було проведено проектування програмного продукту, визначено архітектуру та компоненти системи. Обґрунтовано вибір технологій, включаючи мову програмування Python та бібліотеку Tkinter для реалізації графічного інтерфейсу.

У третьому розділі була описана програмна реалізація системи. Були надані детальні пояснення щодо створення бази даних, реалізації графічного інтерфейсу та основних функцій, таких як додавання, редагування, пошук, сортування товарів та розрахунок загальної вартості.

Отже, розроблена система "Автоматизований меблевий склад" є ефективним інструментом для підприємств у меблевій галузі, що дозволяє оптимізувати управління складськими запасами та підвищити їхню продуктивність. Результати цієї роботи можуть бути використані для подальшого розвитку та впровадження подібних систем на реальних підприємствах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Р. Х. Болтон. "Системний аналіз в бізнесі: методи та методологія". - Посилання: https://books.google.com.ua/books?id=RVlDAQAAIAAJ
2. О. Л. Столінг, М. В. Халфін. "Управління операціями та логістика". - Посилання: https://www.amazon.com/Operations-Management-Logistics-Binder-ready-Version/dp/0134741062
3. Т. Г. Пенні, К. Р. Ньюман. "Вступ до аналізу даних для бізнесу та економіки". - Посилання: https://books.google.com.ua/books?id=DYFSDwAAQBAJ
4. Д. А. Адамс, Р. Дж. Метер. "Підприємництво: новий підхід". - Посилання: https://books.google.com.ua/books?id=HfIsAAAAYAAJ
5. Д. Маккей, Дж. Маккей. "Стратегічне управління". - Посилання: https://www.amazon.com/Strategic-Management-Fred-R-David/dp/0132341409
6. К. М. Черчилль, Р. В. Бітнер. "Маркетинг: стратегії та тактика". - Посилання: https://www.amazon.com/Marketing-Strategies-Tactical-Operations-Management/dp/1285734278
7. Д. В. Гайзель, Р. Ф. Пайн. "Дизайн меблів: від опису до практичної реалізації". - Посилання: https://books.google.com.ua/books?id=_6U6swEACAAJ
8. Л. В. Солодкова, Л. А. Павлова. "Маркетинг у меблевій галузі: сучасні технології та методи". - Посилання: https://books.google.com.ua/books?id=wHMlMQAACAAJ
9. Д. Хиллс, Г. Р. Джонсон. "Меблі для будинку: концепції та практика". - Посилання: https://books.google.com.ua/books?id=G5wQYgEACAAJ
10. П. М. Сенге. "П'ять способів вирішення проблем". - Посилання: https://www.amazon.com/Fifth-Discipline-Practice-Learning-Organization/dp/0385517254
11. Б. Грегорі, П. Н. Феррелл. "Маркетинг: стратегії та програми". - Посилання: https://www.amazon.com/Marketing-Strategy-Planning-Approach-Connect/dp/0078028949
12. Р. Котлер, Г. Армстронг. "Принципи маркетингу". - Посилання: https://books.google.com.ua/books?id=D05rGAAACAAJ
13. Д. Маккей, Дж. Маккей. "Менеджмент". - Посилання: https://www.amazon.com/Management-John-R-Schermerhorn-Jr/dp/1111969713
14. І. П. Михельсон, В. В. Руденко. "Технології складування та перевезення товарів". - Посилання: https://books.google.com.ua/books?id=wRJNDwAAQBAJ
15. Лінк на офіційний сайт Pyton https://www.python.org
16. Лінк на офіційний сайт бібліотеки Tkinter https://docs.python.org/3/library/tkinter.html
17. Лінк на офіційний сайт SQLite https://sqlite.org/

ДОДАТКИ

Додаток А – Код програми

```
import tkinter as tk
from tkinter import ttk, messagebox
import sqlite3

class Database:
    def __init__(self):
        self.conn = sqlite3.connect('products.db')
        self.cursor = self.conn.cursor()
        self.cursor.execute('''
            CREATE TABLE IF NOT EXISTS products (
                code TEXT PRIMARY KEY,
                name TEXT,
                price REAL,
                quantity INTEGER,
                color TEXT,
                size TEXT,
                shape TEXT,
                company TEXT,
                package TEXT,
                brand TEXT
            )
        ''')
        self.conn.commit()

    def add_product(self, product):
        self.cursor.execute('''
```

```
            INSERT INTO products (code, name, price, quantity, color, size, shape, company, package, brand)
            VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
        ''', product)
        self.conn.commit()

    def update_product(self, code, name, price, quantity, color, size, shape, company, package, brand):
        self.cursor.execute('''
            UPDATE products SET name=?, price=?, quantity=?, color=?, size=?, shape=?, company=?, package=?, brand=?
            WHERE code=?
        ''', (name, price, quantity, color, size, shape, company, package, brand, code))
        self.conn.commit()

    def delete_product(self, code):
        self.cursor.execute('DELETE FROM products WHERE code=?', (code,))
        self.conn.commit()

    def search_products(self, code='', name='', price='', quantity='', color='', size='', shape='', company='', package='', brand=''):
        query = 'SELECT * FROM products WHERE 1=1'
        params = []

        if code:
            query += ' AND code=?'
            params.append(code)
        if name:
            query += ' AND name LIKE ?'
            params.append(f'%{name}%')
```

```
    if price:
        query += ' AND price=?'
        params.append(price)
    if quantity:
        query += ' AND quantity=?'
        params.append(quantity)
    if color:
        query += ' AND color LIKE ?'
        params.append(f'%{color}%')
    if size:
        query += ' AND size LIKE ?'
        params.append(f'%{size}%')
    if shape:
        query += ' AND shape LIKE ?'
        params.append(f'%{shape}%')
    if company:
        query += ' AND company LIKE ?'
        params.append(f'%{company}%')
    if package:
        query += ' AND package LIKE ?'
        params.append(f'%{package}%')
    if brand:
        query += ' AND brand LIKE ?'
        params.append(f'%{brand}%')

    self.cursor.execute(query, params)
    return self.cursor.fetchall()

def sort_products(self, sort_by, order):
    order_by = 'ASC' if order == 'За зростанням' else 'DESC'
```

```
        self.cursor.execute(f'SELECT * FROM products ORDER BY {sort_by} {order_by}')
        return self.cursor.fetchall()

class Application(tk.Frame):
    def __init__(self, master=None):
        super().__init__(master)
        self.master = master
        self.master.title("Склад товарів")
        self.pack(fill=tk.BOTH, expand=True)

        self.create_widgets()
        self.db = Database()
        self.populate_treeview()

    def create_widgets(self):
        self.tree = ttk.Treeview(self, columns=('code', 'name', 'price', 'quantity', 'color', 'size', 'shape', 'company', 'package', 'brand'), show='headings')
        self.tree.heading('code', text='Код')
        self.tree.heading('name', text='Назва')
        self.tree.heading('price', text='Ціна')
        self.tree.heading('quantity', text='Кількість')
        self.tree.heading('color', text='Колір')
        self.tree.heading('size', text='Розмір')
        self.tree.heading('shape', text='Форма')
        self.tree.heading('company', text='Компанія')
        self.tree.heading('package', text='Пакунок')
        self.tree.heading('brand', text='Фірма')
        self.tree.pack(fill=tk.BOTH, expand=True)
```

```
        self.tree.bind('<Double-1>', self.on_double_click)

        button_frame = tk.Frame(self)
        button_frame.pack(fill=tk.X)

        self.add_button = tk.Button(button_frame, text="Додати товар",
command=self.open_add_product_window)
        self.add_button.pack(side=tk.LEFT)

        self.edit_button = tk.Button(button_frame, text="Редагувати",
command=self.open_edit_product_window)
        self.edit_button.pack(side=tk.LEFT)

        self.delete_button = tk.Button(button_frame, text="Видалити",
command=self.delete_product)
        self.delete_button.pack(side=tk.LEFT)

        self.search_button = tk.Button(button_frame, text="Пошук",
command=self.open_search_window)
        self.search_button.pack(side=tk.LEFT)

        self.sort_button = tk.Button(button_frame, text="Сортувати",
command=self.open_sort_window)
        self.sort_button.pack(side=tk.LEFT)

        self.calculate_button = tk.Button(button_frame, text="Обчислити суму",
command=self.calculate_total)
        self.calculate_button.pack(side=tk.LEFT)
```

```
        self.back_button = tk.Button(button_frame, text="Повернутися назад", command=self.populate_treeview)
        self.back_button.pack(side=tk.RIGHT)

        self.tree.bind('<Configure>', self.adjust_column_widths)

    def adjust_column_widths(self, event):
        tree_width = self.tree.winfo_width()
        total_columns = len(self.tree['columns'])
        column_width = int(tree_width / total_columns)
        for column in self.tree['columns']:
            self.tree.column(column, width=column_width)

    def on_double_click(self, event):
        item = self.tree.selection()[0]
        product = self.tree.item(item, 'values')
        self.open_edit_product_window(product)

    def open_add_product_window(self):
        AddProductWindow(self)

    def open_edit_product_window(self, product=None):
        if not product:
            selected_item = self.tree.selection()
            if not selected_item:
                messagebox.showwarning("Редагування", "Будь ласка, виберіть товар для редагування.")
                return
            item = selected_item[0]
            product = self.tree.item(item, 'values')
```

```
        EditProductWindow(self, product)

    def delete_product(self):
        selected_item = self.tree.selection()
        if not selected_item:
            messagebox.showwarning("Видалення", "Будь ласка, виберіть товар для видалення.")
            return
        item = selected_item[0]
        code = self.tree.item(item, 'values')[0]
        self.db.delete_product(code)
        self.tree.delete(item)

    def open_search_window(self):
        SearchWindow(self)

    def open_sort_window(self):
        SortWindow(self)

    def clear_treeview(self):
        for item in self.tree.get_children():
            self.tree.delete(item)

    def populate_treeview(self):
        self.clear_treeview()
        products = self.db.cursor.execute('SELECT * FROM products').fetchall()
        for product in products:
            self.tree.insert('', 'end', values=product)
```

```
    def add_product(self, code, name, price, quantity, color, size, shape, company, package, brand):
        self.db.add_product((code, name, price, quantity, color, size, shape, company, package, brand))
        self.populate_treeview()

    def update_product(self, code, name, price, quantity, color, size, shape, company, package, brand):
        self.db.update_product(code, name, price, quantity, color, size, shape, company, package, brand)
        self.populate_treeview()

    def calculate_total(self):
        selected_items = self.tree.selection()
        if not selected_items:
            messagebox.showwarning("Обчислення", "Будь ласка, виберіть товари для обчислення суми.")
            return
        total_sum = 0
        for item in selected_items:
            product = self.tree.item(item, 'values')
            total_sum += float(product[2]) * int(product[3])
        messagebox.showinfo("Загальна сума", f"Загальна сума вибраних товарів: {total_sum} грн.")

class AddProductWindow:
    def __init__(self, app):
        self.app = app
        self.window = tk.Toplevel()
        self.window.title("Додати новий товар")
```

```
        self.create_widgets()

    def create_widgets(self):
        self.code_label = tk.Label(self.window, text="Код товару:")
        self.code_label.grid(row=0, column=0)
        self.code_entry = tk.Entry(self.window)
        self.code_entry.grid(row=0, column=1)

        self.name_label = tk.Label(self.window, text="Назва:")
        self.name_label.grid(row=1, column=0)
        self.name_entry = tk.Entry(self.window)
        self.name_entry.grid(row=1, column=1)

        self.price_label = tk.Label(self.window, text="Ціна:")
        self.price_label.grid(row=2, column=0)
        self.price_entry = tk.Entry(self.window)
        self.price_entry.grid(row=2, column=1)

        self.quantity_label = tk.Label(self.window, text="Кількість:")
        self.quantity_label.grid(row=3, column=0)
        self.quantity_entry = tk.Entry(self.window)
        self.quantity_entry.grid(row=3, column=1)

        self.color_label = tk.Label(self.window, text="Колір:")
        self.color_label.grid(row=4, column=0)
        self.color_entry = tk.Entry(self.window)
        self.color_entry.grid(row=4, column=1)

        self.size_label = tk.Label(self.window, text="Розмір:")
        self.size_label.grid(row=5, column=0)
```

```
        self.size_entry = tk.Entry(self.window)
        self.size_entry.grid(row=5, column=1)

        self.shape_label = tk.Label(self.window, text="Форма:")
        self.shape_label.grid(row=6, column=0)
        self.shape_entry = tk.Entry(self.window)
        self.shape_entry.grid(row=6, column=1)

        self.company_label = tk.Label(self.window, text="Компанія:")
        self.company_label.grid(row=7, column=0)
        self.company_entry = tk.Entry(self.window)
        self.company_entry.grid(row=7, column=1)

        self.package_label = tk.Label(self.window, text="Пакунок:")
        self.package_label.grid(row=8, column=0)
        self.package_entry = tk.Entry(self.window)
        self.package_entry.grid(row=8, column=1)

        self.brand_label = tk.Label(self.window, text="Фірма:")
        self.brand_label.grid(row=9, column=0)
        self.brand_entry = tk.Entry(self.window)
        self.brand_entry.grid(row=9, column=1)

        self.save_button = tk.Button(self.window, text="Зберегти",
command=self.save_product)
        self.save_button.grid(row=10, column=0, columnspan=2)

    def save_product(self):
        code = self.code_entry.get()
        name = self.name_entry.get()
```

```
        price = self.price_entry.get()
        quantity = self.quantity_entry.get()
        color = self.color_entry.get()
        size = self.size_entry.get()
        shape = self.shape_entry.get()
        company = self.company_entry.get()
        package = self.package_entry.get()
        brand = self.brand_entry.get()
        self.app.add_product(code, name, price, quantity, color, size, shape, company, package, brand)
        self.window.destroy()

class EditProductWindow:
    def __init__(self, app, product):
        self.app = app
        self.product = product
        self.window = tk.Toplevel()
        self.window.title("Редагувати товар")
        self.create_widgets()

    def create_widgets(self):
        self.code_label = tk.Label(self.window, text="Код товару:")
        self.code_label.grid(row=0, column=0)
        self.code_entry = tk.Entry(self.window)
        self.code_entry.insert(0, self.product[0])
        self.code_entry.grid(row=0, column=1)
        self.code_entry.config(state='disabled')

        self.name_label = tk.Label(self.window, text="Назва:")
        self.name_label.grid(row=1, column=0)
```

```
self.name_entry = tk.Entry(self.window)
self.name_entry.insert(0, self.product[1])
self.name_entry.grid(row=1, column=1)

self.price_label = tk.Label(self.window, text="Ціна:")
self.price_label.grid(row=2, column=0)
self.price_entry = tk.Entry(self.window)
self.price_entry.insert(0, self.product[2])
self.price_entry.grid(row=2, column=1)

self.quantity_label = tk.Label(self.window, text="Кількість:")
self.quantity_label.grid(row=3, column=0)
self.quantity_entry = tk.Entry(self.window)
self.quantity_entry.insert(0, self.product[3])
self.quantity_entry.grid(row=3, column=1)

self.color_label = tk.Label(self.window, text="Колір:")
self.color_label.grid(row=4, column=0)
self.color_entry = tk.Entry(self.window)
self.color_entry.insert(0, self.product[4])
self.color_entry.grid(row=4, column=1)

self.size_label = tk.Label(self.window, text="Розмір:")
self.size_label.grid(row=5, column=0)
self.size_entry = tk.Entry(self.window)
self.size_entry.insert(0, self.product[5])
self.size_entry.grid(row=5, column=1)

self.shape_label = tk.Label(self.window, text="Форма:")
self.shape_label.grid(row=6, column=0)
```

```
        self.shape_entry = tk.Entry(self.window)
        self.shape_entry.insert(0, self.product[6])
        self.shape_entry.grid(row=6, column=1)

        self.company_label = tk.Label(self.window, text="Компанія:")
        self.company_label.grid(row=7, column=0)
        self.company_entry = tk.Entry(self.window)
        self.company_entry.insert(0, self.product[7])
        self.company_entry.grid(row=7, column=1)

        self.package_label = tk.Label(self.window, text="Пакунок:")
        self.package_label.grid(row=8, column=0)
        self.package_entry = tk.Entry(self.window)
        self.package_entry.insert(0, self.product[8])
        self.package_entry.grid(row=8, column=1)

        self.brand_label = tk.Label(self.window, text="Фірма:")
        self.brand_label.grid(row=9, column=0)
        self.brand_entry = tk.Entry(self.window)
        self.brand_entry.insert(0, self.product[9])
        self.brand_entry.grid(row=9, column=1)

        self.save_button = tk.Button(self.window, text="Зберегти",
command=self.save_product)
        self.save_button.grid(row=10, column=0, columnspan=2)

    def save_product(self):
        code = self.code_entry.get()
        name = self.name_entry.get()
        price = self.price_entry.get()
```

```
        quantity = self.quantity_entry.get()
        color = self.color_entry.get()
        size = self.size_entry.get()
        shape = self.shape_entry.get()
        company = self.company_entry.get()
        package = self.package_entry.get()
        brand = self.brand_entry.get()
        self.app.update_product(code, name, price, quantity, color, size, shape, company,
package, brand)
        self.window.destroy()

class SearchWindow:
    def __init__(self, app):
        self.app = app
        self.window = tk.Toplevel()
        self.window.title("Пошук товарів")
        self.create_widgets()

    def create_widgets(self):
        self.code_label = tk.Label(self.window, text="Код товару:")
        self.code_label.grid(row=0, column=0)
        self.code_entry = tk.Entry(self.window)
        self.code_entry.grid(row=0, column=1)

        self.name_label = tk.Label(self.window, text="Назва:")
        self.name_label.grid(row=1, column=0)
        self.name_entry = tk.Entry(self.window)
        self.name_entry.grid(row=1, column=1)

        self.price_label = tk.Label(self.window, text="Ціна:")
```

```
self.price_label.grid(row=2, column=0)
self.price_entry = tk.Entry(self.window)
self.price_entry.grid(row=2, column=1)

self.quantity_label = tk.Label(self.window, text="Кількість:")
self.quantity_label.grid(row=3, column=0)
self.quantity_entry = tk.Entry(self.window)
self.quantity_entry.grid(row=3, column=1)

self.color_label = tk.Label(self.window, text="Колір:")
self.color_label.grid(row=4, column=0)
self.color_entry = tk.Entry(self.window)
self.color_entry.grid(row=4, column=1)

self.size_label = tk.Label(self.window, text="Розмір:")
self.size_label.grid(row=5, column=0)
self.size_entry = tk.Entry(self.window)
self.size_entry.grid(row=5, column=1)

self.shape_label = tk.Label(self.window, text="Форма:")
self.shape_label.grid(row=6, column=0)
self.shape_entry = tk.Entry(self.window)
self.shape_entry.grid(row=6, column=1)

self.company_label = tk.Label(self.window, text="Компанія:")
self.company_label.grid(row=7, column=0)
self.company_entry = tk.Entry(self.window)
self.company_entry.grid(row=7, column=1)

self.package_label = tk.Label(self.window, text="Пакунок:")
```

```
        self.package_label.grid(row=8, column=0)
        self.package_entry = tk.Entry(self.window)
        self.package_entry.grid(row=8, column=1)

        self.brand_label = tk.Label(self.window, text="Фірма:")
        self.brand_label.grid(row=9, column=0)
        self.brand_entry = tk.Entry(self.window)
        self.brand_entry.grid(row=9, column=1)

        self.search_button = tk.Button(self.window, text="Пошук",
command=self.search)
        self.search_button.grid(row=10, column=0, columnspan=2)

    def search(self):
        code = self.code_entry.get()
        name = self.name_entry.get()
        price = self.price_entry.get()
        quantity = self.quantity_entry.get()
        color = self.color_entry.get()
        size = self.size_entry.get()
        shape = self.shape_entry.get()
        company = self.company_entry.get()
        package = self.package_entry.get()
        brand = self.brand_entry.get()

        products = self.app.db.search_products(code, name, price, quantity, color, size,
shape, company, package, brand)
        self.app.clear_treeview()
        for product in products:
            self.app.tree.insert('', 'end', values=product)
```

```
        self.window.destroy()

class SortWindow:
    def __init__(self, app):
        self.app = app
        self.window = tk.Toplevel()
        self.window.title("Сортування товарів")
        self.create_widgets()

    def create_widgets(self):
        self.sort_by_label = tk.Label(self.window, text="Сортувати за:")
        self.sort_by_label.grid(row=0, column=0)
        self.sort_by_var = tk.StringVar(self.window)
        self.sort_by_var.set("name")
        self.sort_by_menu = tk.OptionMenu(self.window, self.sort_by_var, 'name', 'price', 'quantity', 'color', 'size', 'shape', 'company', 'package', 'brand')
        self.sort_by_menu.grid(row=0, column=1)

        self.order_label = tk.Label(self.window, text="Порядок сортування:")
        self.order_label.grid(row=1, column=0)
        self.order_var = tk.StringVar(self.window)
        self.order_var.set("За зростанням")
        self.order_menu = tk.OptionMenu(self.window, self.order_var, 'За зростанням', 'За спаданням')
        self.order_menu.grid(row=1, column=1)

        self.sort_button = tk.Button(self.window, text="Сортувати", command=self.sort)
        self.sort_button.grid(row=2, column=0, columnspan=2)
```

```
    def sort(self):
        sort_by = self.sort_by_var.get()
        order = self.order_var.get()

        products = self.app.db.sort_products(sort_by, order)
        self.app.clear_treeview()
        for product in products:
            self.app.tree.insert('', 'end', values=product)
        self.window.destroy()

if __name__ == '__main__':
    root = tk.Tk()
    app = Application(master=root)
    app.mainloop()
```