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 програмного забезпечення для автоматизації роботи магазину будівельних матеріалів»

Виконав: студент групи ІПЗ20-1
Спеціальність 121 «Інженерія програмного забезпечення»
Перебора Данило Олегович
(прізвище та ініціали)
Керівник к.ф.-м.н., доц. Фірсов О.Д.
(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

Перебора Д. О. Розроблення програмного забезпечення для автоматизації роботи магазину будівельних матеріалів.

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

Роздрібні магазини будівельних матеріалів активно шукають можливості для постійного покращення ефективності та задоволення потреб своїх клієнтів. Розробка програмного забезпечення для автоматизації роботи таких магазинів стає нагальною потребою, що відповідає викликам сучасного ринку та підвищує конкурентоспроможність бізнесу.

Магазини будівельних матеріалів відіграють важливу роль у будівельній галузі, забезпечуючи необхідними ресурсами як професійних будівельників, так і звичайних споживачів. Проте, зі зростанням конкуренції та обсягів інформації та замовлень, управління такими магазинами стає все складнішим і потребує системного підходу.

У цьому контексті розробка програмного забезпечення для автоматизації роботи магазину будівельних матеріалів стає не лише інноваційним кроком, але й стратегічною необхідністю. Ефективне програмне рішення дозволяє оптимізувати процеси управління складом, обліку товарів, обробки замовлень, взаємодії з постачальниками та обслуговування клієнтів.

Метою поточної роботи є розробка програмного забезпечення, спеціально адаптованого для автоматизації роботи магазину будівельних матеріалів.

Ключові слова: мова програмування Python, PyQt5, база даних SQLite, Qt Designer, QPixmap.

ABSTRACT

Perebora D. O. Development of software for automating the work of a building materials store.

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

Building materials retailers actively seek opportunities to continuously improve efficiency and meet the needs of their customers. The development of software for automating the operation of such stores becomes an urgent need that meets the challenges of the modern market and increases business competitiveness.

Building materials stores play an important role in the construction industry, providing the necessary resources for both professional builders and ordinary consumers. However, with the growth of competition and the volume of information and orders, the management of such stores becomes more and more complex and requires a systematic approach.

In this context, the development of software to automate the work of a building materials store becomes not only an innovative step, but also a strategic necessity. An effective software solution allows you to optimize the processes of warehouse management, product accounting, order processing, interaction with suppliers and customer service.

The goal of the current work is the development of software specially adapted for automating the work of a building materials store.

Keywords: Python programming language, PyQt5, SQLite database, Qt Designer, QPixmap.

ЗМІСТ

ВСТУП

Роздрібні магазини будівельних матеріалів не залишаються осторонь, прагнучи постійного поліпшення ефективності та задоволення потреб своїх клієнтів. Розробка програмного забезпечення для автоматизації роботи таких магазинів стає необхідністю, відповідаючи на виклики сучасного ринку та підвищуючи конкурентоспроможність бізнесу.

Магазини будівельних матеріалів є важливою ланкою в інфраструктурі будівельної галузі, забезпечуючи необхідними ресурсами як професійних будівельників, так і звичайних споживачів. Проте, зі зростанням конкуренції та збільшенням обсягів інформації та замовлень, управління такими магазинами стає дедалі складнішим і потребує системного підходу. У цьому контексті розробка програмного забезпечення для автоматизації роботи магазину будівельних матеріалів є не тільки інноваційним кроком, а й стратегічною необхідністю.

Ефективне програмне рішення дозволяє оптимізувати процеси управління складом, обліку товарів, обробки замовлень, взаємодії з постачальниками та обслуговування клієнтів. Крім цього, таке програмне забезпечення сприяє підвищенню точності прогнозування попиту, скороченню часу на виконання замовлень, поліпшення якості обслуговування та підвищення рівня задоволеності клієнтів.

Важливим аспектом розробки програмного забезпечення для магазину будівельних матеріалів є облік специфічних вимог даної галузі. Це включає не тільки облік особливостей складського обліку та логістики, але й адаптацію під різні види будівельних матеріалів, особливості їх зберігання та транспортування, а також облік специфічних клієнтських запитів. Нарешті, слід зазначити, що розробка програмного забезпечення для автоматизації роботи магазину будівельних матеріалів як збільшує ефективність бізнес-процесів, а й сприяє зростанню прибутків і зміцненню позицій компанії над ринком. Це інвестиція в майбутнє, яка дозволяє магазинам будівельних

матеріалів успішно адаптуватися до умов ринку, що змінюються, і залишатися конкурентоспроможними в довгостроковій перспективі.

Метою даної роботи є розробка програмного забезпечення, спеціально адаптованого для автоматизації роботи магазину будівельних матеріалів. Це програмне рішення буде спрямоване на оптимізацію та покращення всіх ключових аспектів бізнес-процесів магазину, включаючи управління складом, облік товарів, обробку замовлень, взаємодію з постачальниками та обслуговування клієнтів.

Для досягнення поставленої мети – розробки програмного забезпечення для автоматизації роботи магазину будівельних матеріалів перед роботою ставляться такі завдання:

1) аналіз потреб та вимог. Проведення аналізу бізнес-процесів магазину будівельних матеріалів для виявлення основних потреб та вимог до програмного забезпечення. Це включає вивчення процесів управління складом, обробки замовлень, взаємодії з постачальниками і обслуговування клієнтів;

2) визначення функціональних вимог. На основі аналізу потреб визначення конкретних функцій та можливостей, які має надавати програмне забезпечення. Це може включати управління запасами, обробку замовлень, генерацію звітів і аналітику, а також інструменти для поліпшення обслуговування клієнтів;

3) проектування архітектури. Розробка архітектури програмного рішення, яка забезпечуватиме його ефективну роботу, масштабованість та надійність. Це включає вибір відповідних технологій, структурування функціональності і визначення взаємозв'язків між компонентами системи;

4) розробка та тестування. Фази розробки програмного забезпечення, що включають написання коду, створення інтерфейсів, реалізацію функціональності та проведення тестування для забезпечення якості та надійності роботи програми.

Кожне із цих завдань відіграє важливу роль у забезпеченні успішної розробки та впровадження програмного забезпечення для автоматизації роботи магазину будівельних матеріалів.

Об'єктом дослідження у цьому контексті є процес автоматизації роботи магазину будівельних матеріалів з використанням програмного забезпечення. Це включає вивчення всіх аспектів бізнес-процесів цього магазину, починаючи з управління запасами і закінчуючи обслуговуванням клієнтів.

Предметом дослідження у цьому контексті є розробка програмного забезпечення для автоматизації роботи магазину будівельних матеріалів. Це включає конкретні технології, методики та інструменти, які будуть використовуватися для створення програмного рішення.

Кваліфікаційна робота складається зі вступу, 3-х розділів, висновків, використаних джерел з 14 найменувань, 1-го додатка.

Обсяг роботи складається з 50 сторінок основного тексту з 71 сторінки кваліфікаційної роботи та 16 рисунків.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Аналіз публікацій щодо існуючих ПЗ для автоматизації роботи магазинів

У світі будівництва та торгівлі будівельними матеріалами сучасні технології відіграють ключову роль в оптимізації бізнес-процесів, підвищенні ефективності та покращенні обслуговування клієнтів. Різні програми для автоматизації роботи магазинів будівельних матеріалів надають широкий спектр інструментів для управління складом, обліку товарів, обробки замовлень, взаємодії з клієнтами та багато іншого.

Одним з потужних інструментів для автоматизації роботи магазину будівельних матеріалів є Retail Pro (рис. 1.1). Це програмне забезпечення має розширені можливості з управління багатьма аспектами бізнесу, включаючи інвентаризацію, облік продажів, управління персоналом та аналітику [1]. Однією з основних переваг Retail Pro є його масштабованість та гнучкість, що дозволяє використовувати його як для невеликих магазинів, так і для великих роздрібних мереж. Важливим аспектом автоматизації роботи магазину будівельних матеріалів є ефективне керування складом.

У цій галузі виділяється додаток «СкладОК», спеціально розроблений для обліку товарів на складі та оптимізації логістичних процесів. За допомогою СкладОК можна відстежувати надходження та відвантаження товарів, контролювати залишки на складі, а також оптимізувати реалізацію товарів з урахуванням їх характеристик та попиту.

Для ефективного управління клієнтськими відносинами магазину будівельних матеріалів часто застосовуються CRM-системи. Необхідно також згадати платформи електронної комерції, які можуть бути використані для продажу будівельних матеріалів в інтернеті. Однією з найпопулярніших платформ у цій галузі є «Shopify». Цей вебдодаток надає можливості для

створення онлайн-магазину з мінімальними витратами та технічними навичками. Shopify (рис. 1.2) включає інструменти з управління каталогом товарів, обробки замовлень, налаштування способів оплати та доставки [2].

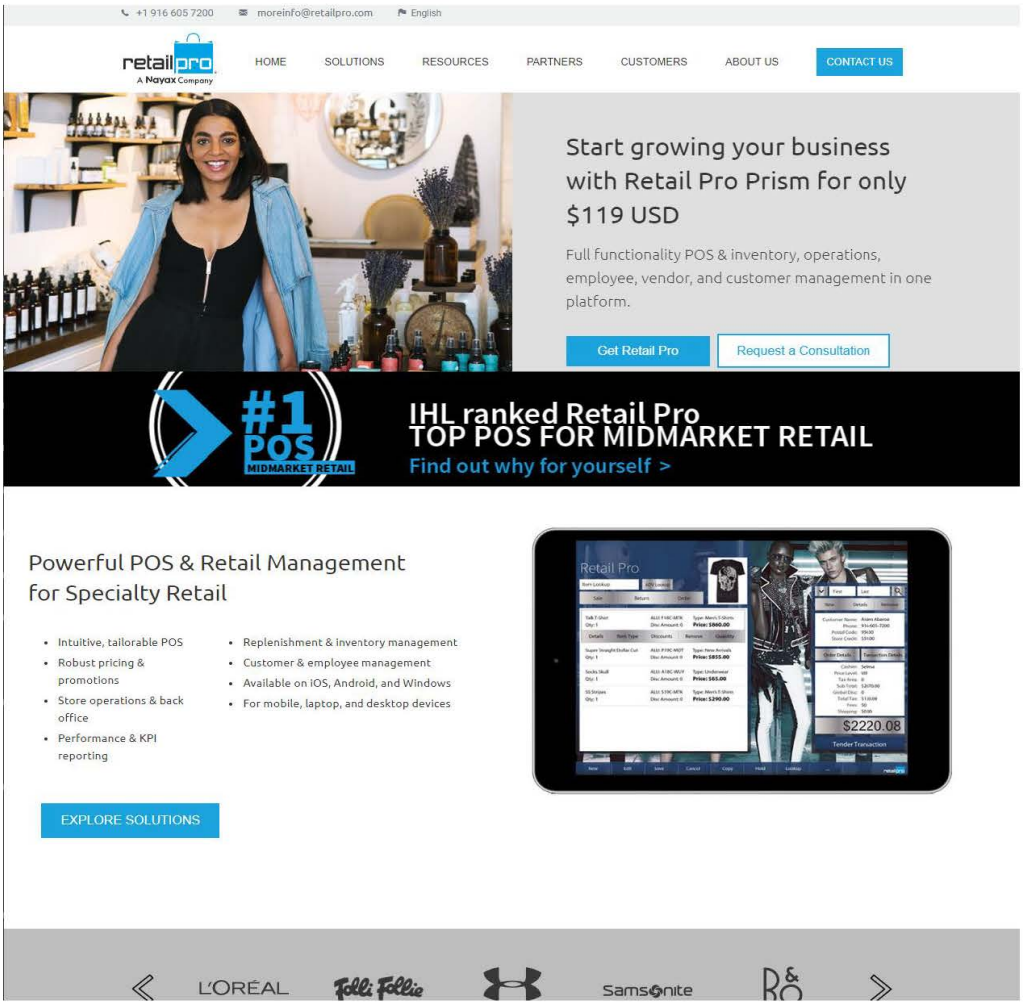

Рисунок 1.1 – Інтерфейс Retail Pro

Крім згаданих вище додатків існує ще кілька цікавих рішень, які можуть бути корисні для автоматизації роботи магазину будівельних матеріалів. ERPNext (рис. 1.3) – це відкрите програмне забезпечення, що надає функціональність ERP (Enterprise Resource Planning) [3]. Воно включає модулі для управління складом, обліку фінансів, продажів, закупівель і виробництва. ERPNext може бути налаштований під конкретні потреби бізнесу та інтегрований з іншими системами.

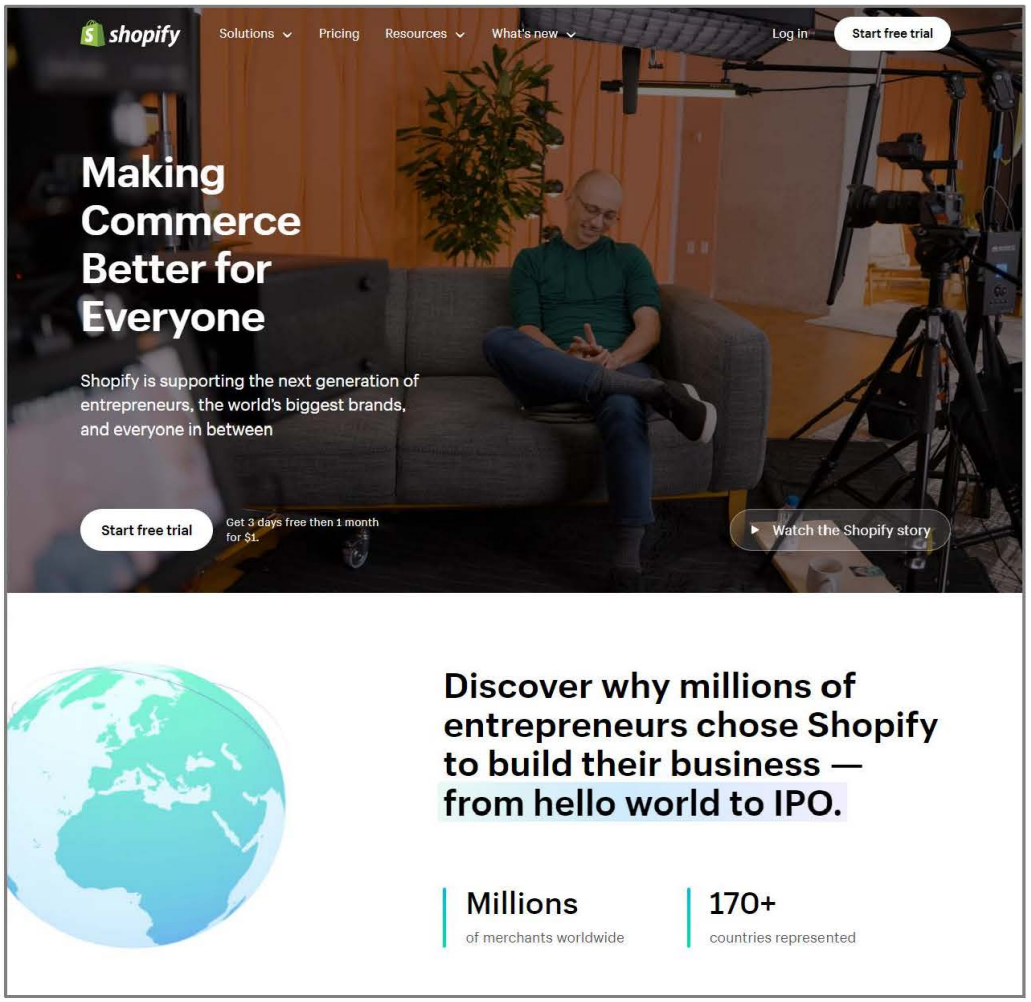

Рисунок 1.2 – Інтерфейс Shopify

WooCommerce – це популярний плагін для WordPress, який перетворює сайт на цій платформі на повнофункціональний інтернет-магазин. Хоча WooCommerce (рис. 1.4) в першу чергу орієнтований на електронну комерцію, його також можна використовувати для керування фізичним магазином, особливо, якщо він має онлайн-присутність [4].

QuickBooks Point of Sale – це програмне забезпечення для керування точкою продажу. Воно дозволяє керувати продажами, інвентаризацією, замовленнями, а також веде облік клієнтів та надає звіти про продаж. QuickBooks Point of Sale інтегрується з іншими продуктами QuickBooks, що забезпечує синхронізацію даних між різними бізнес-процесами.

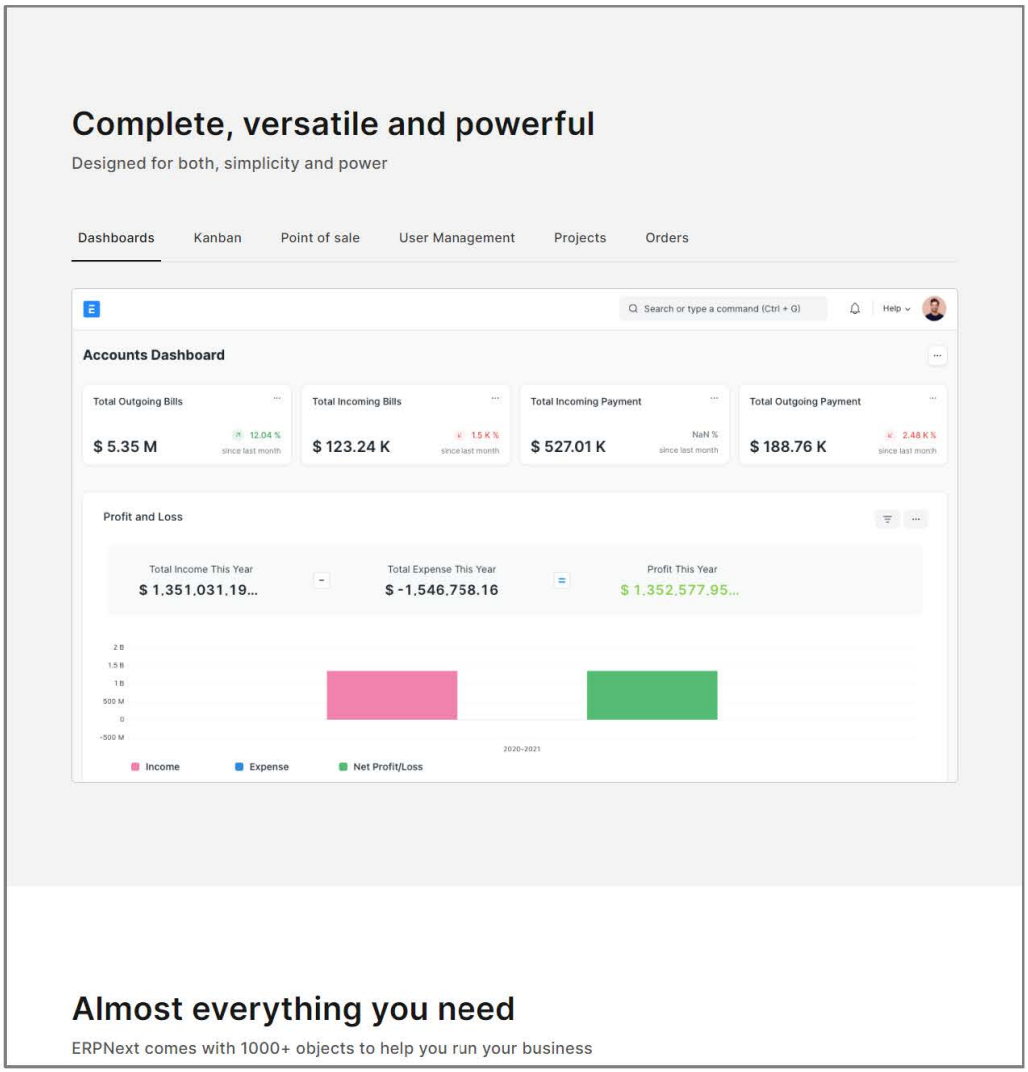

Рисунок 1.3 – Інтерфейс ERPNext

Square Point of Sale – це програма для керування точкою продажів, яка працює на смартфонах та планшетах. Воно надає інструменти для обробки платежів, обліку продажу та інвентаризації. Square також пропонує додаткові послуги, такі як створення інтернет-магазину, маркетингові інструменти та аналітику.

Lightspeed Retail – це хмарне програмне забезпечення для управління роздрібним бізнесом. Воно включає модулі для обліку товарів, продажів, клієнтів, замовлень і звітності. Lightspeed Retail також інтегрується з різними платіжними системами, електронними комерційними платформами та іншими сервісами. від потреб та цілей магазину будівельних матеріалів.

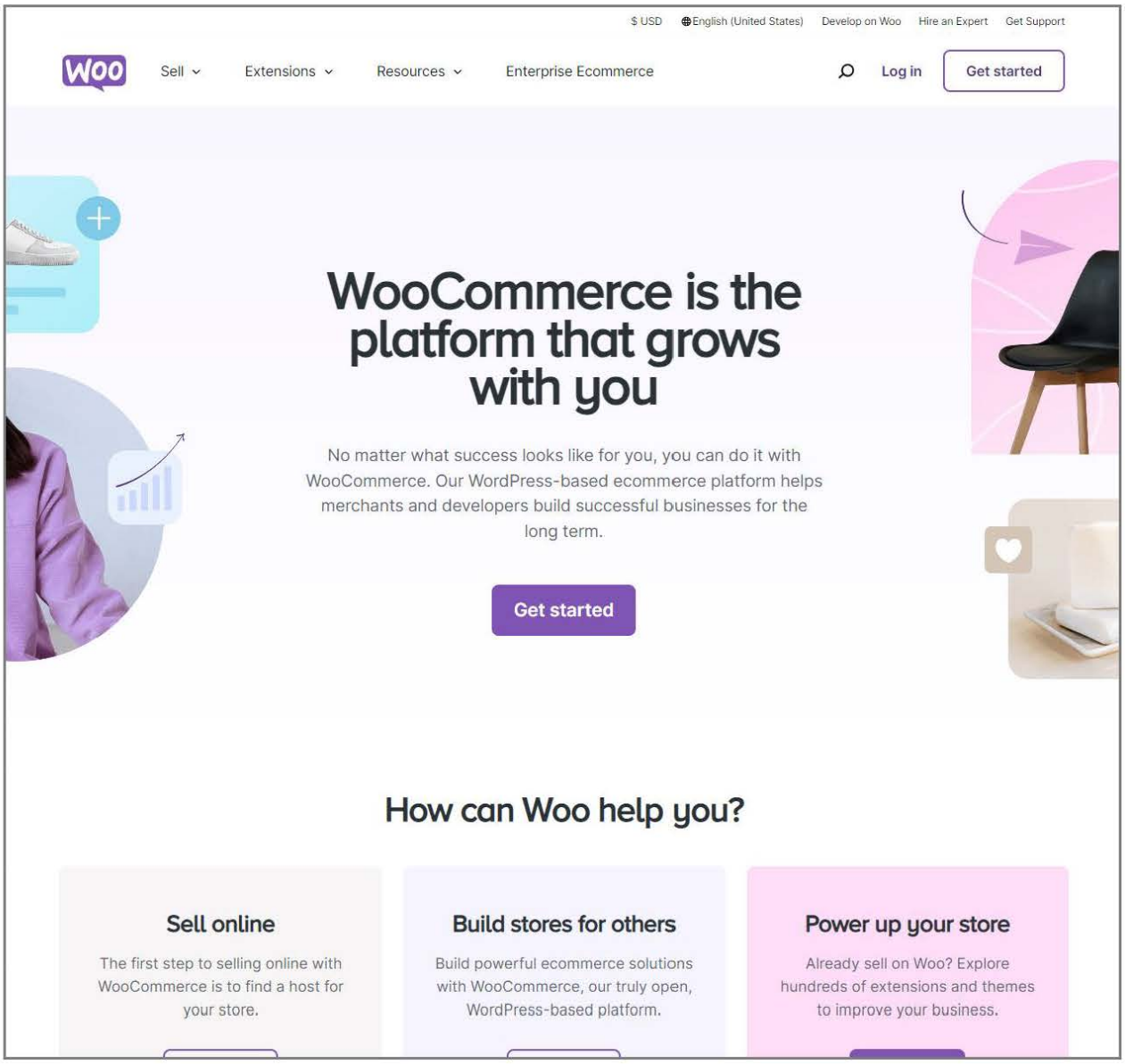

Рисунок 1.4 – Інтерфейс WooCommerce

Кожна з цих програм має свої особливості та переваги, тому вибір конкретного рішення залежить. Однак у будь-якому випадку автоматизація з використанням сучасних технологій допоможе підвищити ефективність бізнесу та покращити обслуговування клієнтів.

1.2 Аналіз методів розробки

Для створення програми для автоматизації роботи магазину будівельних матеріалів потрібне поглиблене розуміння як бізнес-процесів самого магазину,

так і технологічних інструментів, які можна використовувати для їх оптимізації.

Веброзробка: вебпрограми стають все більш популярними в бізнес-сфері завдяки своїй доступності та зручності використання. Для створення програми для магазину будівельних матеріалів можна використовувати мови програмування, такі як JavaScript, Python, або PHP у поєднанні з фреймворками, такими як React, Angular або Vue.js для розробки фронтенду, а також фреймворки для бекенда, такі як Node.js, Django або Flask. Ці інструменти дозволяють створювати інтерактивні та масштабовані вебпрограми.

Мобільна розробка: враховуючи поширеність мобільних пристроїв, включаючи смартфони та планшети, розробка мобільного додатка також може бути ключовим аспектом. Для створення мобільного додатка для магазину будівельних матеріалів можна використовувати мови програмування Java або Kotlin для платформ Android та Swift для iOS. Також можна скористатися фреймворками та інструментами, такими як React Native або Flutter, які дозволяють створювати кросплатформові мобільні програми із загальним кодом для різних платформ.

Бази даних: ефективне зберігання та управління даними відіграє критичну роль у розробці програми для автоматизації роботи магазину будівельних матеріалів. Для цього можуть бути використані реляційні бази даних, такі як MySQL, PostgreSQL або Microsoft SQL Server, а також бази даних NoSQL, такі як MongoDB або Firebase. Вибір конкретної бази даних залежить від вимог до масштабованості, продуктивності та структури даних програми.

Хмарні технології: використання хмарних технологій дозволяє створювати гнучкі та масштабовані програми, які легко можуть адаптуватися до потреб бізнесу, що змінюються. Популярні хмарні платформи, такі як Amazon Web Services (AWS), Microsoft Azure або Google Cloud Platform,

надають широкий спектр інструментів та сервісів для хостингу, зберігання даних, масштабування та моніторингу програм.

Інтеграція та API: для забезпечення взаємодії між різними компонентами системи, такими як облік товарів, обробка замовлень та керування складом, необхідна інтеграція із зовнішніми сервісами та API.

Багато популярних програм та сервісів надають API для інтеграції з ними, що дозволяє створювати гнучкі та розширювані системи.

Штучний інтелект та аналітика даних: використання технологій штучного інтелекту та аналітики даних може допомогти оптимізувати бізнес-процеси та приймати більш поінформовані рішення. Наприклад, алгоритми машинного навчання можуть допомогти прогнозувати попит на товари, оптимізувати ціноутворення та управління запасами, а аналітика даних дозволяє аналізувати продажі, поведінку клієнтів та інші ключові показники ефективності.

Безпека: безпека даних є важливим аспектом розробки будь-якої програми, особливо при роботі з конфіденційною інформацією про клієнтів та замовлення. Для забезпечення безпеки програми необхідно реалізувати механізми автентифікації та авторизації, захист даних у спокої та передачі, а також механізми виявлення та запобігання атак. Розробка програми для автоматизації роботи магазину будівельних матеріалів потребує комплексного підходу та використання різних технологій та інструментів. Важливо враховувати потреби та особливості конкретного бізнесу, а також стежити за останніми тенденціями в галузі інформаційних технологій, щоб створити ефективне та конкурентоспроможне рішення.

Крім технологій, описаних раніше, існують інші аспекти, які слід враховувати при розробці додатка для автоматизації роботи магазину будівельних матеріалів.

Інтерфейс користувача (UI) та досвід користувача (UX): створення зручного та інтуїтивно зрозумілого інтерфейсу користувача є ключовим фактором успішності будь-якої програми. Важливо враховувати потреби та

переваги користувачів магазину будівельних матеріалів при розробці дизайну інтерфейсу та досвіду користувача. Це включає зручну навігацію, зрозумілі та інтуїтивно зрозумілі елементи управління, а також адаптивний дизайн для забезпечення коректного відображення на різних пристроях та екранах.

Система керування версіями: використання системи керування версіями, такої як Git, дозволяє ефективно керувати кодовою базою програми, відстежувати зміни, керувати гілками розробки та співпрацювати з іншими розробниками. Це допомагає забезпечити цілісність та стабільність коду при розробці та підтримці програми.

Тестування та налагодження: якісне тестування та налагодження є невід'ємною частиною процесу розробки будь-якої програми. Це включає функціональне тестування для перевірки правильності роботи різних функцій і модулів, а також тестування продуктивності, безпеки та сумісності. Регулярне тестування допомагає виявити та виправити помилки та проблеми до їх потрапляння в продакшн.

Документація та навчання: добре структурована та докладна документація відіграє важливу роль у підтримці та розвитку програми. Це включає в себе технічну документацію, що описує архітектуру, функціональність і API додатки, а також документацію користувача, що пояснює, як користуватися додатком.

Також важливо надати навчання та підтримку користувачам при впровадженні та використанні програми.

Масштабованість та гнучкість: при розробці програми для магазину будівельних матеріалів необхідно враховувати можливість масштабування та гнучкості системи. Це дозволяє застосуванню адаптуватися до потреб і обсягів роботи, що змінюються, а також забезпечує можливість додавання нових функцій і модулів у майбутньому.

Дотримання правових та регуляторних вимог: важливо враховувати дотримання різних правових та регуляторних вимог при розробці програми, особливо щодо захисту даних та конфіденційності. Це включає дотримання

законодавства про захист персональних даних (наприклад, GDPR), а також інших галузевих нормативних вимог.

Оновлення та підтримка: після випуску програми в продакшн важливо забезпечити його регулярне оновлення та підтримку, включаючи виправлення помилок, додавання нових функцій та оновлення безпеки. Регулярні оновлення допомагають підтримувати програму в актуальному стані та забезпечувати безпеку та продуктивність.

Всі ці аспекти відіграють важливу роль у розробці програми для автоматизації роботи магазину будівельних матеріалів. Успішне впровадження такого додатку потребує комплексного підходу, врахування потреб бізнесу та кінцевих користувачів, а також використання сучасних технологій та кращих практик розробки програмного забезпечення.

1.3 Висновки до першого розділу. Постановка завдань дослідження

Метою даної є розробка програмного забезпечення, спеціально адаптованого для автоматизації роботи магазину будівельних матеріалів. Це програмне рішення буде спрямоване на оптимізацію та покращення всіх ключових аспектів бізнес-процесів магазину, включаючи управління складом, облік товарів, обробку замовлень, взаємодію з постачальниками та обслуговування клієнтів.

Для досягнення поставленої мети – розробки програмного забезпечення для автоматизації роботи магазину будівельних матеріалів перед роботою ставляться такі завдання:

1) аналіз потреб та вимог. Проведення аналізу бізнес-процесів магазину будівельних матеріалів для виявлення основних потреб та вимог до програмного забезпечення. Це включає вивчення процесів управління складом, обробки замовлень, взаємодії з постачальниками і обслуговування клієнтів;

2) визначення функціональних вимог. На основі аналізу потреб визначення конкретних функцій та можливостей, які має надавати програмне забезпечення. Це може включати управління запасами, обробку замовлень, генерацію звітів і аналітику, а також інструменти для поліпшення обслуговування клієнтів;

3) проектування архітектури. Розробка архітектури програмного рішення, яка забезпечуватиме його ефективну роботу, масштабованість та надійність. Це включає вибір відповідних технологій, структурування функціональності і визначення взаємозв'язків між компонентами системи;

4) розробка та тестування. Фази розробки програмного забезпечення, що включають написання коду, створення інтерфейсів, реалізацію функціональності та проведення.

Проведено аналіз публікацій, пов'язаних із існуючим програмним забезпеченням (ПЗ) для автоматизації роботи магазинів будівельних матеріалів. Це дозволило отримати огляд сучасних рішень у цій галузі, їх переваги та недоліки. Виконано аналіз методів розробки, який включав у собі розгляд різних підходів до створення програмного забезпечення, таких як веброзробка, мобільна розробка, використання різних мов програмування та фреймворків.

Після проведення аналізу у першому розділі зроблені висновки та сформульовано постановку завдань дослідження. Це включає в себе визначення потреб та вимог до програмного забезпечення для магазинів будівельних матеріалів, а також визначення напрямків подальшого дослідження та розробки.

Отже, розділ 1 створює базову основу для подальшої роботи, дозволяючи визначити потреби та вимоги до програмного забезпечення, а також розглянути різні методи розробки, які можуть бути використані в процесі розробки додатка для автоматизації роботи магазинів будівельних матеріалів.

РОЗДІЛ 2 АНАЛІЗ ЗАСОБІВ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ МАГАЗИНУ БУДІВЕЛЬНИХ МАТЕРІАЛІВ

2.1 Вибір програмних засобів для реалізації проекту

Для реалізації цього програмного застосунку було використано наступні програмні засоби:

1) python. Основна мова програмування, на якій було написано програму. Python – це високорівнева мова програмування з простим синтаксисом, що робить його ідеальним для швидкої розробки;

2) PyQt5 – це набір інструментів для розробки програмного забезпечення з графічним інтерфейсом користувача (GUI) на Python. Використовується для створення вікон, кнопок, полів введення та інших елементів інтерфейсу;

3) SQLite – це легковага вбудовувана база даних, яка використовується для зберігання та організації даних програми. Вона ідеально підходить для невеликих або середніх проектів, таких як цей, оскільки не вимагає окремого сервера та може зберігати дані в одному файлі;

4) Qt Designer – це графічний редактор для створення та редагування інтерфейсів користувача PyQt. Його можна використовувати для швидкої побудови GUI за допомогою перетягування та розміщення елементів;

5) QPixmap – це клас для роботи з зображеннями у PyQt. Використовується для завантаження та відображення зображень в інтерфейсі.

Ці програмні засоби дозволили створити програму з графічним інтерфейсом, яка може додавати, видаляти, редагувати та продавати товари в магазині будівельних матеріалів, а також відображати статистику і завантажувати зображення для кожного товару.

2.2 Мова програмування Python

Python — це інтерпретована, високорівнева мова програмування загального призначення, розроблена наприкінці 1980-х років Гвідо ван Россумом і випущена в 1991 році. Він поєднує в собі простоту із сильною системою типів та динамічним управлінням пам'яттю, що робить його популярним вибором для розробників на всіх рівнях досвіду [5].

Однією з ключових особливостей Python є його читабельний синтаксис, який робить код більш зрозумілим та легким для підтримки. Python підтримує різні парадигми програмування, включаючи процедурне, об'єктно-орієнтоване та функціональне програмування, що робить його універсальним інструментом для вирішення широкого кола завдань.

Однією з найбільш помітних переваг Python є його широка стандартна бібліотека, яка включає модулі для роботи з файлами, мережами, базами даних, регулярними виразами і багатьма іншими [5]. Це робить Python дуже продуктивним інструментом, оскільки багато завдань можна виконати за допомогою вбудованих засобів, мінімізуючи необхідність у сторонніх бібліотеках.

Python також відомий своєю розширюваністю та активною спільнотою розробників. Існує безліч сторонніх бібліотек і фреймворків, які розширюють функціональність Python для різних цілей, починаючи від веброзробки та машинного навчання, і закінчуючи науковими обчисленнями та ігровою розробкою. Наприклад, бібліотеки такі як NumPy, Pandas, TensorFlow, Django та Flask широко використовуються у відповідних областях.

Одним з основних принципів Python є «ясність краще, ніж неясність», що означає, що код повинен бути написаний так, щоб його легко розуміли інші розробники. Це дозволяє скоротити час розробки, спростити підтримку коду та зробити його надійнішим.

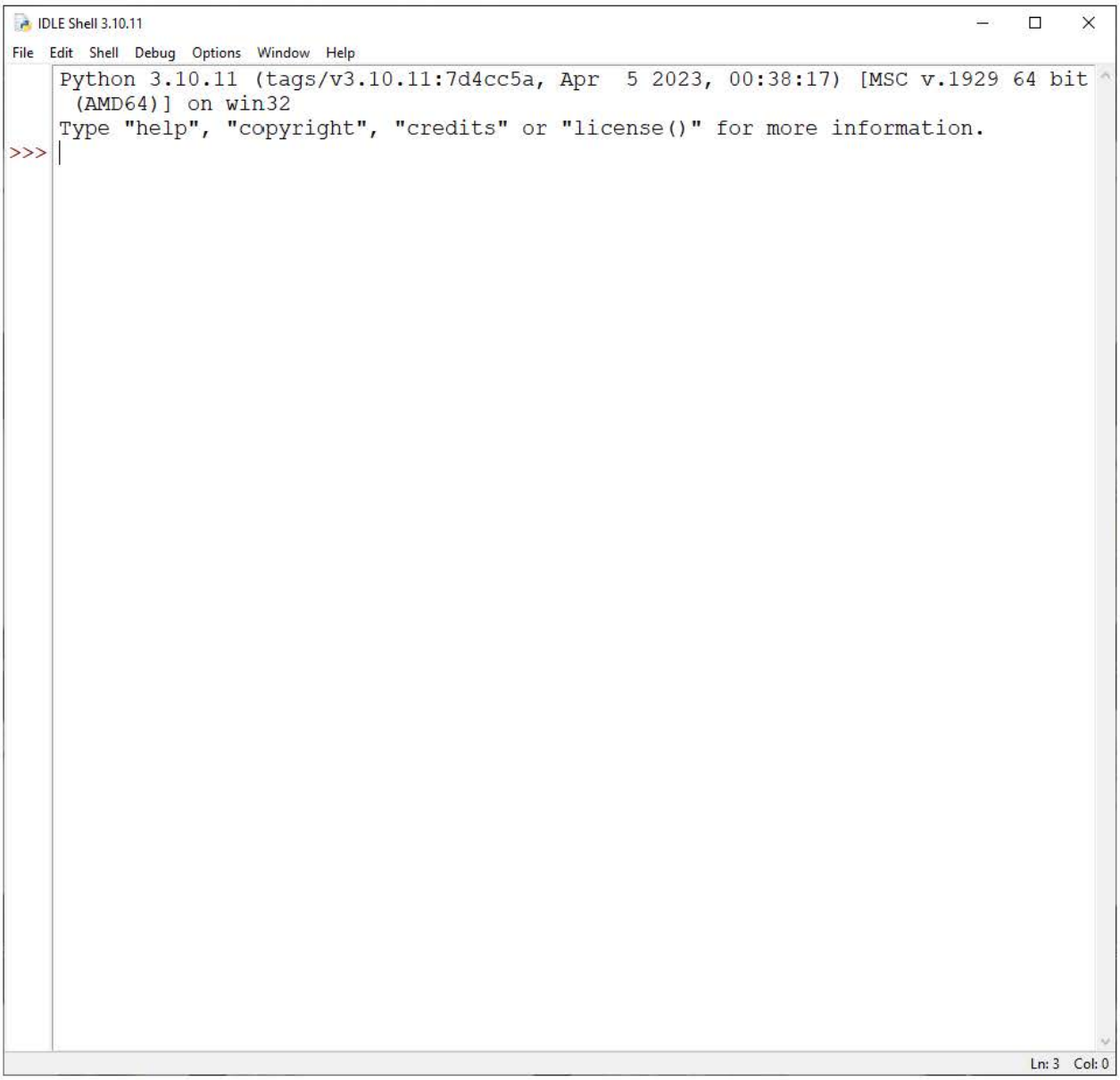

Рисунок 2.1 – Інтерфейс середовища Python IDLE

Python також активно використовується в освіті завдяки своїй простоті та доступності. Багато навчальних закладів використовують Python для навчання основ програмування, оскільки він дозволяє швидко досягти результатів та стимулює студентів до подальшого вивчення інформатики. В цілому, Python – це потужна та гнучка мова програмування, яка знаходить широке застосування у багатьох галузях, від веброзробки до наукових досліджень. Його простота, читабельність і широка екосистема роблять його однією з найпопулярніших і найпопулярніших мов програмування у світі.

2.3 PyQt5 бібліотека

PyQt5 – це бібліотека для створення графічного інтерфейсу користувача (GUI) мовою Python. Вона надає доступ до багатьох віджетів, елементів керування та функцій для створення інтерактивних програм за допомогою PyQt5.

Завдяки своїй потужності та гнучкості, PyQt5 стала популярним інструментом для розробки програм з графічним інтерфейсом у Python. Основним компонентом PyQt5 є модуль QtWidgets, який надає класи для створення різних віджетів, таких як кнопки, поля введення, списки та багато іншого. За допомогою цих віджетів можна створювати різноманітні інтерфейси користувача, від простих діалогових вікон до складних додатків з безліччю елементів управління. Однією з ключових переваг PyQt5 є його інтеграція з Qt Designer – графічним інструментом для створення інтерфейсів.

Qt Designer дозволяє візуально проектувати інтерфейси користувача, додавати віджети і встановлювати властивості без необхідності написання коду. Потім створений інтерфейс можна експортувати до файлу .ui, який можна завантажити і використовувати в PyQt5.

Крім того, PyQt5 має широкі можливості кастомізації та стилізації інтерфейсів. При цьому можна налаштовувати зовнішній вигляд віджетів, використовуючи каскадні таблиці стилів (CSS) або навіть створюючи власні кастомні віджети. Це дозволяє створювати унікальні та стильні інтерфейси користувача, які відповідають вимогам проекту або корпоративному брендингу.

Крім того, PyQt5 має широку функціональність для обробки подій та взаємодії з користувачем. Існує можливість прив'язувати обробники подій до різних віджетів, реагувати на дії користувача та оновлювати інтерфейс у реальному часі. Це робить PyQt5 сучасним інструментом для створення інтерактивних програм, таких як ігри, веббраузери або інструменти аналізу даних.

Програми, створені за допомогою PyQt5, можуть запускатися на різних операційних системах, включаючи Windows, macOS та Linux, без зміни вихідного коду. Це робить PyQt5 ідеальним вибором для розробки кросплатформових додатків, які мають працювати на різних пристроях та платформах. Крім того, PyQt5 надає доступ до безлічі додаткових модулів та інструментів для розширення функціональності додатків.

Наприклад, QtSql дозволяє працювати з базами даних, а QtWebEngine дозволяє вбудовувати веббраузери до програм PyQt5. Це робить PyQt5 повноцінним інструментом для розробки широкого спектру програм, від маленьких утиліт до великих корпоративних програм.

Незважаючи на всі переваги, є деякі недоліки при використанні PyQt5. По-перше, PyQt5 є комерційним продуктом, і для комерційного використання може знадобитися придбання ліцензії. По-друге, PyQt5 має великий розмір, що може призвести до збільшення розміру додатків, що випускаються.

В цілому PyQt5 – це потужний і гнучкий інструмент для створення графічного інтерфейсу користувача в Python. Він надає широкі можливості для створення красивих, інтерактивних та кросплатформових додатків, які відповідають найвищим стандартам продуктивності та функціональності.

2.4 Мова програмування SQL

SQL або мова структурованих запитів є стандартною мовою для роботи з реляційними базами даних. Цей потужний інструмент використовується для управління даними, що зберігаються в базах даних, виконання запитів до цих даних, модифікації структури бази даних та багато іншого.

Розуміння SQL є ключовою навичкою для розробників, аналітиків даних, адміністраторів баз даних та інших фахівців, які працюють із даними. Однією з основних особливостей SQL є його декларативний характер. Замість того, щоб вказувати точні кроки для виконання операції, розробник описує

бажаний результат, і СУБД сама визначає оптимальний спосіб виконання запиту. Це дозволяє зосередитись на меті запиту, а не на тому, як її досягти.

SQL підтримує кілька основних операцій, включаючи SELECT, INSERT, UPDATE та DELETE. Оператор SELECT використовується для вилучення даних із бази даних відповідно до заданих умов. Оператори INSERT, UPDATE та DELETE дозволяють додавати, оновлювати та видаляти дані відповідно. Крім того, SQL також підтримує операції створення та зміни структури бази даних, такі як CREATE TABLE, ALTER TABLE та DROP TABLE.

Таблиці є структурованими наборами даних, що складаються з рядків і стовпців. Кожен рядок таблиці є записом, а кожен стовпець – атрибут або поле запису. SQL дозволяє створювати, змінювати та видаляти таблиці, визначати їх структуру та зв'язки між ними. SQL також підтримує використання умов, функцій та агрегатних функцій для фільтрації, сортування та аналізу даних. Умови дозволяють задавати критерії для вибірки даних, наприклад, вибрати всі записи, які відповідають певній умові.

Функції дозволяють перетворювати дані або виконувати обчислення на них, наприклад обчислити суму або середнє значення стовпця. Агрегатні функції дозволяють аналізувати дані групи, наприклад, обчислити суму чи середнє значення кожної групи записів. SQL також підтримує використання індексів підвищення продуктивності запитів.

Індекси – це структури даних, які прискорюють пошук записів у таблиці, попередньо сортуючи дані щодо певного стовпця або комбінації стовпців. Індекси дозволяють СУБД швидко знаходити записи, що задовольняють заданим умовам, що значно покращує продуктивність запитів.

Загалом SQL є потужною та універсальною мовою для роботи з даними в реляційних базах даних. Він надає широкий набір інструментів для виконання різноманітних операцій, від вилучення та модифікації даних до адміністрування баз даних та оптимізації запитів. Розуміння SQL є ключовою навичкою для ефективної роботи з даними та вирішення різноманітних завдань у галузі інформаційних технологій.

2.5 База даних SQLite

SQLite – це компактна СУБД, що вбудовується, яка забезпечує легкий і ефективний спосіб роботи з реляційними базами даних.

Він відрізняється від великих реляційних баз даних, таких як MySQL або PostgreSQL, тим, що не вимагає окремого сервера і керується безпосередньо з програми, в якій він вбудований. SQLite є популярним вибором для систем, мобільних додатків, а також для невеликих і середніх вебдодатків. Однією з ключових переваг SQLite є його простота використання та встановлення. SQLite не вимагає складної установки або конфігурації сервера, що робить його легким у використанні для розробників. Для створення бази даних SQLite досить просто створити новий файл бази даних та почати працювати з нею за допомогою SQL-запитів.

Ще однією перевагою SQLite є його переносимість та кросплатформність. Файли бази даних SQLite можуть бути легко переміщені між різними операційними системами та архітектурами, що робить їх ідеальним вибором для розробки платформних додатків. SQLite підтримується на багатьох операційних системах, включаючи Windows, macOS, Linux та багато інших. Крім того, SQLite має високу продуктивність та ефективність роботи з даними.

SQLite використовує простий та ефективний двигун бази даних, який забезпечує швидке виконання SQL-запитів та ефективне використання ресурсів системи. SQLite також забезпечує надійне зберігання даних та підтримує транзакції для забезпечення цілісності даних.

Важливою особливістю SQLite є підтримка повного набору стандартних SQL-функцій та операторів. Розробники можуть використовувати знайомий SQL-синтаксис для виконання запитів до даних, створення таблиць, індексів та тригерів, а також для виконання інших операцій із базою даних.

SQLite також підтримує розширений набір функцій, включаючи агрегатні функції, оператори та функції рядків, дат та чисел. Однією з

ключових переваг SQLite є його низьке споживання ресурсів. SQLite має невеликий розмір бінарного файлу та низькі вимоги до пам'яті та процесора, що робить його ідеальним вибором для вбудованих систем, мобільних пристроїв та інших обмежених ресурсів середовищ.

Однак, незважаючи на всі свої переваги, SQLite має деякі обмеження. Одним з них є недостатня масштабованість для великих додатків з високими вимогами до продуктивності та масштабованості.

SQLite не підтримує розрахований на багато користувачів доступ до бази даних, що робить його неспроможним для використання у великих мережних додатках з великою кількістю одночасних користувачів.

Тим не менш, SQLite залишається популярним вибором для широкого кола додатків завдяки своїй простоті використання, високої продуктивності та ефективності роботи з даними.

Він використовується в різних областях, включаючи мобільне програмування, системи, аналіз даних і багато інших. SQLite продовжує розвиватися та покращуватися, забезпечуючи потужний та надійний інструмент для роботи з даними.

2.6 База даних MongoDB

MongoDB – це популярна документоорієнтована база даних, яка відрізняється від традиційних реляційних СУБД тим, що використовує формат зберігання даних у вигляді документів замість таблиць [6].

Вона надає гнучку та масштабовану модель даних, а також широкий набір функцій для роботи з даними. Однією з ключових особливостей MongoDB є гнучка схема даних. На відміну від реляційних баз даних, де структура даних визначається заздалегідь і суворо відповідає схемі таблиці,

MongoDB дані зберігаються у вигляді документів у форматі JSON або BSON, що дозволяє легко додавати, змінювати і видаляти поля в документі без необхідності зміни схеми бази даних. Це робить MongoDB ідеальним вибором

для проектів, де структура даних може змінюватися або потрібна гнучка модель даних [6]. Ще однією важливою особливістю MongoDB є її масштабованість.

MongoDB надає можливість горизонтального масштабування, що дозволяє розподілити навантаження на кілька серверів та забезпечити високу продуктивність і доступність навіть за дуже великих обсягів даних та високих навантажень. Це MongoDB ідеальним вибором для проектів з великими обсягами даних або високими вимогами до продуктивності. MongoDB також має широкий набір функцій для роботи з даними.

Вона підтримує безліч операцій CRUD (Create, Read, Update, Delete) для роботи з документами, а також потужні запити, агрегації та індекси для ефективного доступу до даних. MongoDB також надає можливості для безпеки даних, реплікації, резервного копіювання та відновлення даних, що робить її повноцінним рішенням для широкого кола завдань.

Важливою особливістю MongoDB є її розширюваність та екосистема. MongoDB надає API та інструменти для роботи з даними різних мов програмування, а також інтеграцію з різними фреймворками та інструментами розробки. Крім того, MongoDB має активну спільноту розробників і велику документацію, що робить її легкою в освоєнні та використанні для нових проектів [6].

Однією з переваг MongoDB є її підтримка для роботи з неструктурованими даними та різними типами даних, включаючи текст, географічні дані, зображення та багато іншого. Це робить MongoDB універсальним інструментом для зберігання та обробки різноманітних даних, що дозволяє використовувати її у різних галузях, від веброзробки до аналізу даних та машинного навчання.

Однак, незважаючи на всі свої переваги, MongoDB також має деякі недоліки. Один з них – відсутність транзакцій у деяких версіях MongoDB, що може створювати проблеми при забезпеченні цілісності даних у деяких сценаріях використання. Крім того, MongoDB потребує додаткових ресурсів

для забезпечення високої продуктивності та доступності при масштабуванні, що може збільшити витрати на інфраструктуру [6].

В цілому, MongoDB є потужним і гнучким рішенням для зберігання і обробки даних. Вона надає широкий набір функцій і можливостей для роботи з даними, а також має високу продуктивність і масштабованість. MongoDB підходить для широкого кола завдань, від невеликих вебдодатків до великих корпоративних систем, і продовжує розвиватися та покращуватись, забезпечуючи при цьому надійний інструмент для роботи з даними.

2.7 База даних PostgreSQL

PostgreSQL – це потужна об'єктно-реляційна система управління базами даних (СУБД), яка надає широкий набір можливостей для зберігання, управління та обробки даних [7-9].

Ця СУБД відрізняється від інших рішень завдяки своїй розширюваності, масштабованості та підтримці стандартів. Однією з ключових особливостей PostgreSQL є повна підтримка стандарту SQL. PostgreSQL надає багатий та потужний набір SQL-функцій та операторів для роботи з даними, що робить її ідеальним вибором для розробників, які звикли до використання стандартного SQL. Завдяки цьому перенесення додатків між різними СУБД стає простіше і менш витратним.

Ще однією перевагою PostgreSQL є його розширюваність та екосистема. PostgreSQL надає широкий набір вбудованих функцій і типів даних, а також можливість створення розширень і функцій користувача на мові програмування PL/pgSQL. Це дозволяє розширювати функціональність PostgreSQL та адаптувати її під свої потреби та вимоги проекту.

PostgreSQL також відрізняється від інших СУБД своєю підтримкою розширених можливостей для роботи з даними. Вона підтримує безліч типів даних, включаючи географічні дані, JSON та XML, що робить її універсальним інструментом для зберігання та обробки різноманітних даних. Крім того,

PostgreSQL надає можливості для роботи з великими обсягами даних, партиціонування таблиць, реплікації та кластеризації, що робить її ідеальним вибором для великих та високонавантажених додатків [8].

Важливою особливістю PostgreSQL є її надійність та відмовостійкість. PostgreSQL забезпечує механізми для забезпечення цілісності даних, транзакційної підтримки та механізмів відновлення після збоїв, що робить її надійним рішенням для критично важливих програм та систем. Крім того, PostgreSQL надає можливості для резервного копіювання та відновлення даних, що дозволяє забезпечити безпечне зберігання даних та захист від втрати.

Однак, незважаючи на всі свої переваги, PostgreSQL має деякі недоліки. Один з них – це високе навантаження на ресурси системи при виконанні складних запитів чи операцій із великими обсягами даних. PostgreSQL вимагає достатньої кількості пам'яті та процесорних ресурсів для забезпечення високої продуктивності та чуйності додатків.

Крім того, PostgreSQL вимагає досвідчених адміністраторів баз даних для налаштування та оптимізації продуктивності, що може створювати додаткові витрати та складності для організацій [8]. В цілому, PostgreSQL є потужним і гнучким рішенням для зберігання та обробки даних. Вона надає широкий набір функцій та можливостей для роботи з даними, а також має високу надійність та відмовостійкість. PostgreSQL підходить для широкого кола завдань, від маленьких вебдодатків до великих корпоративних систем, і продовжує розвиватися та покращуватись, забезпечуючи розробникам надійний інструмент для роботи з даними.

2.8 Qt Designer

Qt Designer – це графічний інструмент, що надається фреймворком Qt для створення інтерфейсів додатків. Він дозволяє візуально проектувати та

налаштовувати графічні елементи інтерфейсу без необхідності написання коду вручну [10, 11].

Qt Designer інтегрується з іншими інструментами Qt, такими як Qt Creator, що робить процес створення інтерфейсів більш ефективним та зручним. Однією з основних особливостей Qt Designer є його інтуїтивно зрозумілий інтерфейс користувача. У Qt Designer реалізовані інтуїтивно зрозумілі елементи управління та інструменти, такі як палітра віджетів, меню контексту та панелі інструментів, що робить процес створення інтерфейсів легким та приємним.

Існує можливість додавати, переміщувати та налаштовувати віджети за допомогою миші та клавіатури, що дозволяє швидко створювати та налаштовувати інтерфейси програм. Qt Designer надає доступ до великого набору певних віджетів та елементів керування, таких як кнопки, поля введення, списки, таблиці, зображення та багато іншого [10].

Розробники можуть вибирати з цього набору віджетів і додавати їх на форму інтерфейсу, а потім налаштовувати їх властивості та зовнішній вигляд за допомогою властивостей та стилів. Це дозволяє створювати різноманітні і стильні інтерфейси користувача, які відповідають вимогам проекту або брендингу.

Однією з ключових особливостей Qt Designer є його підтримка механізму макетів (layouts), який дозволяє створювати адаптивні та масштабовані інтерфейси.

Механізм макетів автоматично розподіляє та вирівнює віджети на формі інтерфейсу залежно від розмірів вікна програми, що забезпечує коректне відображення інтерфейсу на різних пристроях та роздільній здатності екрану. Це робить Qt Designer ідеальним інструментом для створення кроссплатформенних додатків, які повинні працювати на різних пристроях та операційних системах. Крім того, Qt Designer має широкі можливості кастомізації та розширення [11].

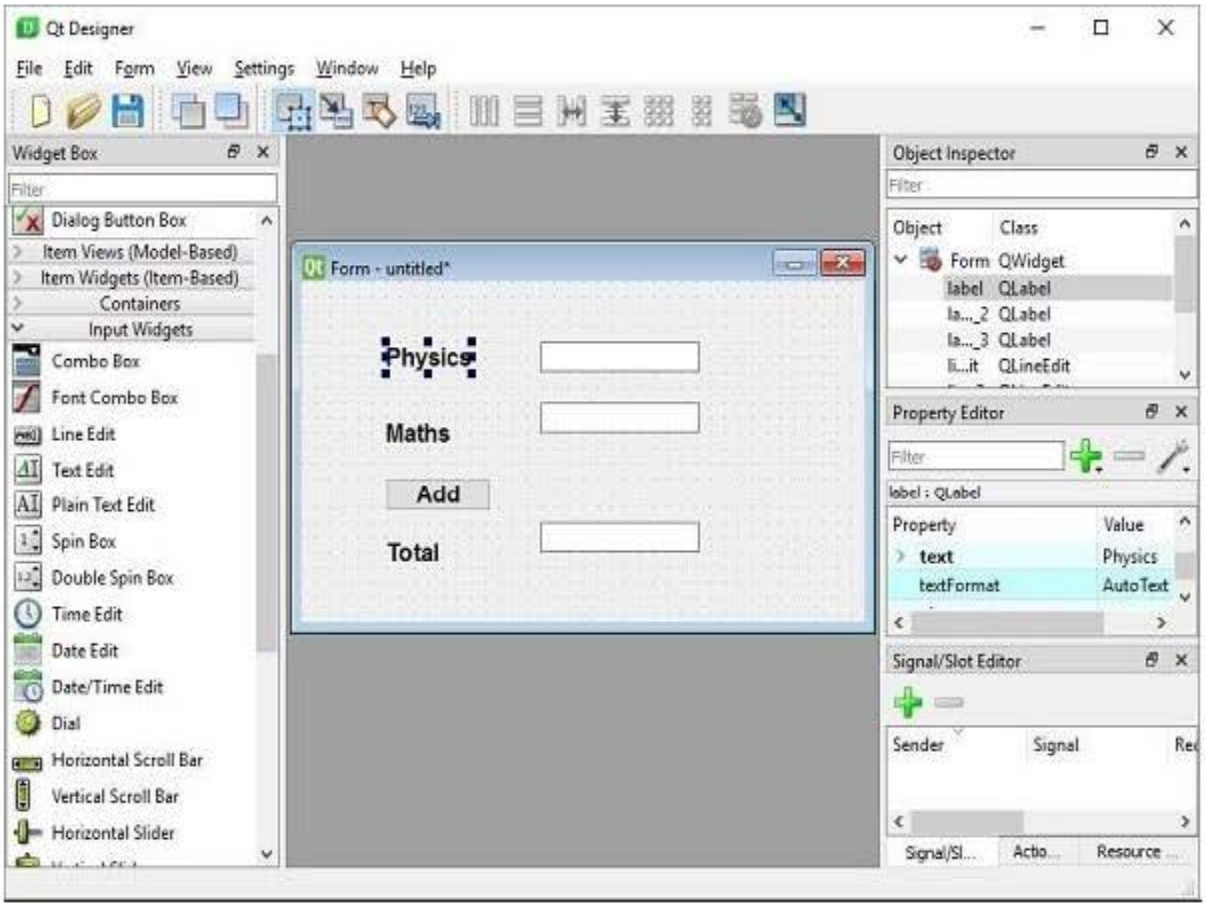

Рисунок 2.2 – Інтерфейс Qt Designer

При цьому можна створювати власні віджети і елементи управління за допомогою механізму віджетів (custom widgets) і інтегрувати їх в Qt Designer для подальшого використання. Це дозволяє створювати унікальні та спеціалізовані елементи управління, які відповідають вимогам проекту або специфічним потребам програми. Qt Designer також має широкі можливості взаємодії з іншими інструментами та технологіями.

Він інтегрується з іншими інструментами Qt, такими, як Qt Creator, що забезпечує єдиний робочий процес розробки додатків на Qt. Крім того, Qt Designer підтримує експорт та імпорт форм інтерфейсу у різних форматах, таких як XML, що дозволяє спільно працювати над проектами та обмінюватися інтерфейсами між різними інструментами та розробниками.

В цілому, Qt Designer – це потужний і зручний інструмент для створення інтерфейсів додатків на Qt. Він надає розробникам широкі можливості для

створення красивих, функціональних та адаптивних інтерфейсів, а також спрощує процес розробки додатків на Qt, роблячи його більш ефективним та продуктивним.

2.9 QPixmap

QPixmap – це клас у бібліотеці Qt, який надає можливості для роботи із зображеннями [12].

Він є частиною модуля QtGui та використовується для завантаження, відображення та маніпулювання растровими зображеннями у додатках, створених з використанням фреймворку Qt.

Однією з основних можливостей QPixmap є завантаження зображень із файлів різних форматів, таких як PNG, JPEG, BMP та інших. QPixmap надає зручний інтерфейс для завантаження зображень з файлового сховища та створення об'єктів зображень, які можуть бути використані в програмі для відображення на екрані або в інтерфейсі користувача.

Крім того, QPixmap дозволяє створювати зображення програмним шляхом за допомогою різних методів і функцій. Існує можливість створювати порожні зображення певного розміру і формату, заповнювати їх кольором, малювати на них різні фігури та лінії, а також наносити текст і зображення за допомогою QPainter, що робить QPixmap потужним інструментом для створення графічних елементів та інтерфейсів. Однією з ключових особливостей QPixmap є його підтримка альфа-каналу та прозорості. QPixmap дозволяє створювати зображення із альфа-каналом, який визначає прозорість пікселів зображення.

Це дозволяє створювати зображення з прозорими областями, які можуть бути відображені поверх інших елементів інтерфейсу або фону, що дає можливість створювати складні та креативні інтерфейси користувача. Ще однією важливою особливістю QPixmap є його підтримка масштабування та перетворення зображень. QPixmap надає методи для зміни розміру

зображення, зміни його пропорцій, обертання, відображення та інших перетворень, що дозволяє легко маніпулювати зображеннями у додатку та адаптувати їх під різні вимоги та умови відображення [12].

Qt також надає потужні засоби роботи з QPixmap в контексті графічного інтерфейсу користувача. Разом з іншими класами та компонентами бібліотеки QtGui, такими як QWidget, QLabel, QGraphicsView та іншими, QPixmap може бути використаний для створення складних та інтерактивних графічних програм, які надають користувачеві можливість взаємодіяти із зображеннями та виконання різних дій з ними.

Нарешті, важливо відзначити, що QPixmap має гарну продуктивність і оптимізацію. Qt використовує ефективні алгоритми та структури даних для роботи із зображеннями, що дозволяє забезпечити швидке завантаження, відображення та маніпулювання зображеннями навіть при роботі з великими обсягами даних або високих дозволів.

Це робить QPixmap відповідним вибором для створення графічно інтенсивних програм з високими вимогами до продуктивності та якості візуалізації.

2.10 QMessageBox

QMessageBox – це один із найпоширеніших класів у бібліотеці Qt, призначений для відображення діалогових вікон із повідомленнями та інформацією для користувачів у додатках із графічним інтерфейсом [13].

Цей клас надає зручний та простий спосіб додатків взаємодіяти з користувачами, надаючи їм інформацію, попередження, запити підтвердження та інші види повідомлень. Одним з ключових аспектів роботи з QMessageBox є його гнучкість та настроюваність. Цей клас дозволяє створювати різні типи повідомлень, включаючи інформаційні, попереджувальні, критичні та запитальні.

Кожен тип повідомлення має свої унікальні стилі та іконки, що дозволяє користувачеві швидко зрозуміти його зміст та прийняти відповідні дії. За допомогою QMessageBox можна відображати текстові повідомлення, а також додаткові кнопки для взаємодії користувача, такі як «ОК», «Скасувати», «Так», «Ні» та інші [13]. Це дозволяє програмам надавати користувачеві вибір та управління залежно від ситуації, що підвищує зручність використання та функціональність програми.

Одним із найпоширеніших застосувань QMessageBox є відображення попереджень та помилок, наприклад, при некоректному введенні даних або виникненні непередбаченої ситуації в додатку. Це дозволяє користувачеві швидко помітити проблему та вжити необхідних заходів для її вирішення.

Крім того, QMessageBox може використовуватися для відображення інформаційних повідомлень, наприклад, для повідомлення користувача про успішне виконання операції або надання інструкцій щодо використання програми. Це допомагає покращити загальний досвід користувача та зробити взаємодію з програмою більш прозорою та інтуїтивно зрозумілою.

Однією з особливостей QMessageBox є його можливість взаємодіяти з подіями користувача та зворотними викликами. Це дозволяє розробникам виконувати різні дії в залежності від вибору користувача, наприклад, виконувати певні операції, натиснувши кнопку «Так» або «Скасувати» [13].

Такий підхід забезпечує більш гнучке та потужне управління поведінкою програми та її взаємодією з користувачем. Важливо відзначити, що QMessageBox є частиною більшої бібліотеки Qt і може бути інтегрований з іншими компонентами та функціональними можливостями Qt для створення більш складних і потужних програм з графічним інтерфейсом. Однак навіть у найпростіших програмах QMessageBox залишається незамінним інструментом для взаємодії з користувачами та надання їм інформації та управління.

2.11 QInputDialog

QInputDialog – це клас у бібліотеці Qt, який надає зручний спосіб для створення діалогових вікон із запитом інформації у користувача [14].

Ці діалоги можуть бути використані для отримання різних типів даних, таких як текст, числа, вибір елемента зі списку та інші. Одним із найпоширеніших способів використання QInputDialog є запит текстової інформації у користувача.

За допомогою методу getText() можна створити діалогове вікно, де користувач може ввести текст. Це може бути корисним, наприклад, для запиту імені користувача, коментаря або інших даних. QInputDialog також надає можливість запиту цифрової інформації у користувача. Для цього можна використовувати методи getInt() або getDouble(), які створюють діалогові вікна для введення цілих чи дробових чисел відповідно. Це може бути корисним, наприклад, при запиті віку користувача, кількості елементів або інших числових значень.

Крім того, QInputDialog дозволяє створювати діалогові вікна для вибору елемента зі списку [14]. За допомогою методу getItem() можна створити вікно, в якому користувач може вибрати один із запропонованих елементів. Це може бути корисним, наприклад, при виборі опції з визначеного списку або категорії. Іншим цікавим способом використання QInputDialog є створення діалогового вікна для вибору кольору. За допомогою методу getColor() можна створити вікно, в якому користувач може вибрати колір із палітри або ввести його значення вручну.

Це може бути корисним, наприклад, при налаштуванні колірної схеми програми або виборі кольору для малювання. Однією з ключових переваг використання QInputDialog є його простота та зручність у використанні. За допомогою нього розробники можуть легко додавати можливість введення даних до своїх програм без необхідності створення власних діалогових вікон або обробки введення вручну [14]. Крім того, QInputDialog надає різні

параметри та налаштування для налаштування зовнішнього вигляду та поведінки діалогових вікон.

Це дозволяє адаптувати їх під конкретні потреби програми та забезпечити більш зручний та інтуїтивно зрозумілий інтерфейс для користувачів. Таким чином, QInputDialog є потужним та зручним інструментом для створення діалогових вікон із запитом інформації у користувача у додатках на основі Qt. Він надає простий та ефективний спосіб взаємодії з користувачами та отримання необхідних даних для роботи програми.

2.12 Висновки до другого розділу

Розділ починається з обговорення вибору програмних засобів для автоматизації роботи магазину будівельних матеріалів. Це вказує на важливість правильного вибору технологій для успішної реалізації проекту. Багато інструментів, таких як PyQt5, працюють мовою програмування Python. Це свідчить про широке використання Python у розробці програмного забезпечення та його популярність серед розробників.

PyQt5 розглядається в розділі як одна з ключових бібліотек для створення графічного інтерфейсу користувача. Її вибір обумовлений широкими можливостями, гнучкістю та простотою у використанні.

Розділ також включає огляд мови SQL та декількох типів баз даних, включаючи SQLite, MongoDB та PostgreSQL. Це вказує на необхідність зберігання та управління даними в рамках проекту, а також різноманітність рішень для цієї мети.

Qt Designer та QPixmap обговорюються як інструменти для створення графічного інтерфейсу та роботи із зображеннями відповідно. Їхня згадка свідчить про необхідність візуального створення інтерфейсів та роботи із зображеннями в рамках проекту.

РОЗДІЛ 3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ МАГАЗИНУ БУДІВЕЛЬНИХ МАТЕРІАЛІВ

3.1 Постановка задачі

Задача полягає у розробці програмного засобу для автоматизації роботи магазину будівельних матеріалів. Програмний засіб повинен мати графічний інтерфейс користувача (GUI) і забезпечувати наступні функціональні можливості:

1) додавання товарів. Користувач може ввести назву товару, кількість, ціну, категорію та опис, а також завантажити зображення товару. Ці дані зберігаються у базі даних;

2) видалення товарів. Користувач може вибрати товар зі списку і видалити його. Після видалення товар повинен бути видалений із бази даних;

3) редагування товарів. Користувач може вибрати товар зі списку для редагування та змінити його кількість та ціну. Оновлені дані повинні зберігатися у базі даних;

4) продаж товарів. Користувач може вибрати товар для продажу, вказати кількість та ціну продажу. Після продажу кількість товару у базі даних має бути оновлена, а також повинен бути створений запис про продаж;

5) відображення статистики. Програма повинна відображати статистику, таку як мінімальна, максимальна та загальна кількість товарів у магазині;

6) фільтрація та сортування товарів. Користувач може фільтрувати та сортувати товари за назвою, категорією та іншими параметрами.

Завдання полягає в розробці програмного засобу, який надасть ефективний і зручний інтерфейс для управління інвентарем магазину будівельних матеріалів, дозволяючи ефективно виконувати операції додавання, видалення, редагування та продажу товарів, а також відображення статистики.

3.2 Аналіз архітектури

Архітектура програми складається з різних компонентів, які співпрацюють між собою для досягнення функціональності програми.

Основні компоненти архітектури включають:

1) графічний інтерфейс користувача (GUI). Головне вікно програми та інші візуальні елементи, такі як кнопки, поля введення, таблиці тощо. Реалізовано за допомогою PyQt5, який надає засоби для створення та управління GUI;

2) база даних SQLite. Використовується для зберігання та управління даними про товари в магазині, а також про продажі товарів. Включає таблиці для товарів та продажів, а також відповідні зв'язки між ними;

3) логіка програми. Включає класи та методи для виконання операцій додавання, видалення, редагування та продажу товарів. Організовано в клас InventoryApp, який містить реалізації цих методів;

4) взаємодія між компонентами. GUI і логіка програми взаємодіють через сигнали та слоти PyQt5. Логіка програми звертається до бази даних для зберігання та отримання даних;

5) відображення статистики. Використовується для відображення мінімальної, максимальної та загальної кількості товарів у магазині. Оновлюється на основі даних, які зберігаються в базі даних;

6) фільтрація та сортування товарів. Реалізована за допомогою методів для фільтрації та сортування даних, які взаємодіють з інтерфейсом користувача.

Усі перераховані компоненти працюють разом, щоб забезпечити користувачеві можливість ефективно управляти інвентарем магазину будівельних матеріалів через зручний інтерфейс користувача.

3.3 Аналіз структури

Структура програми може бути організована наступним чином:

1) основний файл програми. Включає в себе виклик головного вікна програми та налаштування основних параметрів;

2) головне вікно програми. Містить головний інтерфейс користувача з різними елементами керування. Відповідає за відображення даних та реакцію на взаємодію користувача;

3) функції та методи для обробки подій. Включають в себе методи для додавання, видалення, редагування та продажу товарів. Також містять методи для фільтрації, сортування та відображення статистики;

4) клас для роботи з базою даних. Містить методи для підключення до бази даних, виконання sql-запитів та отримання даних;

5) класи для роботи з графічним інтерфейсом. Включають в себе класи для створення елементів інтерфейсу, таких як кнопки, поля введення, таблиці тощо. Містять методи для відображення та оновлення інтерфейсу;

6) модуль для взаємодії зображень. Містить функції для завантаження та відображення зображень товарів;

7) додаткові модулі та бібліотеки. Включають в себе модулі для роботи з базою даних SQLite, PyQt5 для створення графічного інтерфейсу, а також інші необхідні бібліотеки.

Така структура дозволяє розділити функціональність програми на окремі компоненти, які легко управляти та модифікувати. Кожен компонент відповідає за свою частину функціональності, що спрощує розробку та підтримку програмного забезпечення.

3.4 Огляд бази даних

База даних для цієї програми – це SQLite база даних, яка є вбудованою і не вимагає окремого сервера. SQLite – це легка, ефективна та проста у

використанні реляційна база даних, яка ідеально підходить для невеликих та середніх проектів. Структура бази даних включає наступні таблиці:

1) таблиця «products»:

– id: Унікальний ідентифікатор товару (integer, primary key);

– name: Назва товару (text);

– quantity: Кількість товару в наявності (integer);

– price: Ціна товару (real);

– category: Категорія товару (text);

– description: Опис товару (text);

– image_path: Шлях до зображення товару (text).

2) таблиця «sales»:

– id: Унікальний ідентифікатор продажу (integer, primary key);

– product_id: Ідентифікатор товару, який був проданий (зовнішній ключ, пов'язаний з колонкою id таблиці «products»);

– quantity: Кількість проданого товару (integer);

– price: Ціна продажу товару (real);

– date: Дата продажу (timestamp або text).

Ці таблиці дозволяють зберігати дані про товари в магазині, а також записи про продажі цих товарів. Реляційні зв'язки між таблицями дозволяють зберігати зв'язану інформацію та легко виконувати операції з об'єднанням та фільтрацією даних. Інформація у цих таблицях може бути збережена, оновлена, видалена та використана програмою для відображення, додавання, редагування та продажу товарів у магазині.

3.5 Інтерфейс користувача

Проектування інтерфейсу користувача в цьому додатку включає в себе розробку зручного та ефективного інтерфейсу, який дозволяє користувачеві легко взаємодіяти з програмою.

Основні принципи проектування інтерфейсу користувача включають у себе:

1) простота та зрозумілість. Інтерфейс повинен бути легким для розуміння та використання користувачем. Меню та елементи управління повинні бути чіткими та легко доступними;

2) коректність та відповідність завданням. Інтерфейс повинен відповідати завданням програми, забезпечуючи користувачеві можливість виконувати всі необхідні операції;

3) ергономіка та доступність. Кожен елемент інтерфейсу повинен бути легко доступний та зручний для використання. Розташування елементів та їх організація повинні сприяти швидкій та зручній роботі з програмою;

4) відповідність дизайну та стилю. Дизайн інтерфейсу повинен відповідати загальному стилю програми та бути приємним для користувача;

5) функціональність та можливості. Інтерфейс повинен забезпечувати доступ до всіх функціональних можливостей програми, таких як додавання, видалення, редагування та продаж товарів, а також відображення статистики.

Інтерфейс користувача містить такі елементи, як:

– головне меню з розділами для різних функцій програми;

– кнопки для додавання, видалення, редагування та продажу товарів;

– таблиця для відображення списку товарів;

– поля для введення даних про товари;

– поля для введення кількості та ціни при продажу товарів;

– елементи для фільтрації та сортування списку товарів;

– вікно для відображення зображень товарів.

Під час проектування інтерфейсу користувача важливо врахувати потреби та зручність для кінцевих користувачів, щоб забезпечити оптимальний досвід використання програми.

3.6 Процес роботи ПЗ

Програмне застосування для автоматизації роботи магазину будівельних матеріалів працює за наступним процесом:

1) запуск програми. Користувач запускає програму на своєму комп'ютері або іншому пристрої;

2) відображення головного вікна. Після запуску програма відображає головне вікно з графічним інтерфейсом користувача;

3) завантаження даних. При запуску програма завантажує дані з бази даних про існуючі товари в магазині;

4) відображення списку товарів. Завантажені дані відображаються у вигляді списку товарів у таблиці на головному вікні програми;

5) виконання операцій. Користувач може виконувати різні операції над товарами, такі як додавання нових товарів, видалення існуючих, редагування даних та продаж товарів;

6) взаємодія з інтерфейсом. Користувач може взаємодіяти з програмою через різні елементи інтерфейсу, такі як кнопки, поля введення, таблиці та меню;

7) оновлення даних у базі даних. Після виконання операцій над товарами дані оновлюються у базі даних, щоб зберегти зміни;

8) відображення статистики. Програма автоматично оновлює статистичні дані про кількість товарів у магазині та відображає їх на головному вікні;

9) завершення роботи. Користувач може закрити програму, після чого робота з нею завершиться.

Процес роботи програмного застосування забезпечує користувачеві зручний і ефективний інструмент для управління інвентарем магазину будівельних матеріалів, дозволяючи швидко виконувати необхідні операції та контролювати стан товарів.

3.7 Тестування ПЗ

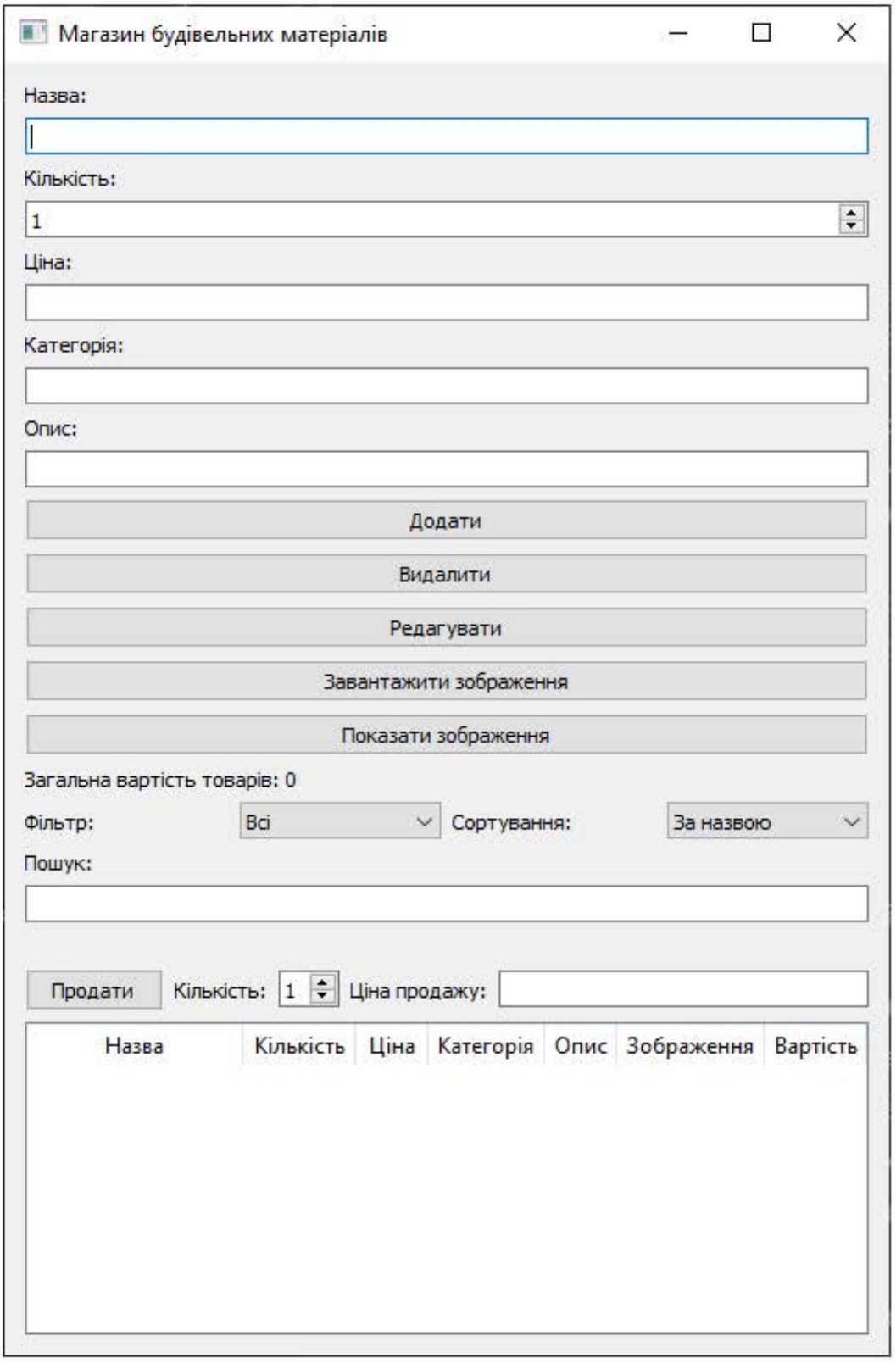

Рисунок 3.1 – Інтерфейс головного вікна розробленого ПЗ

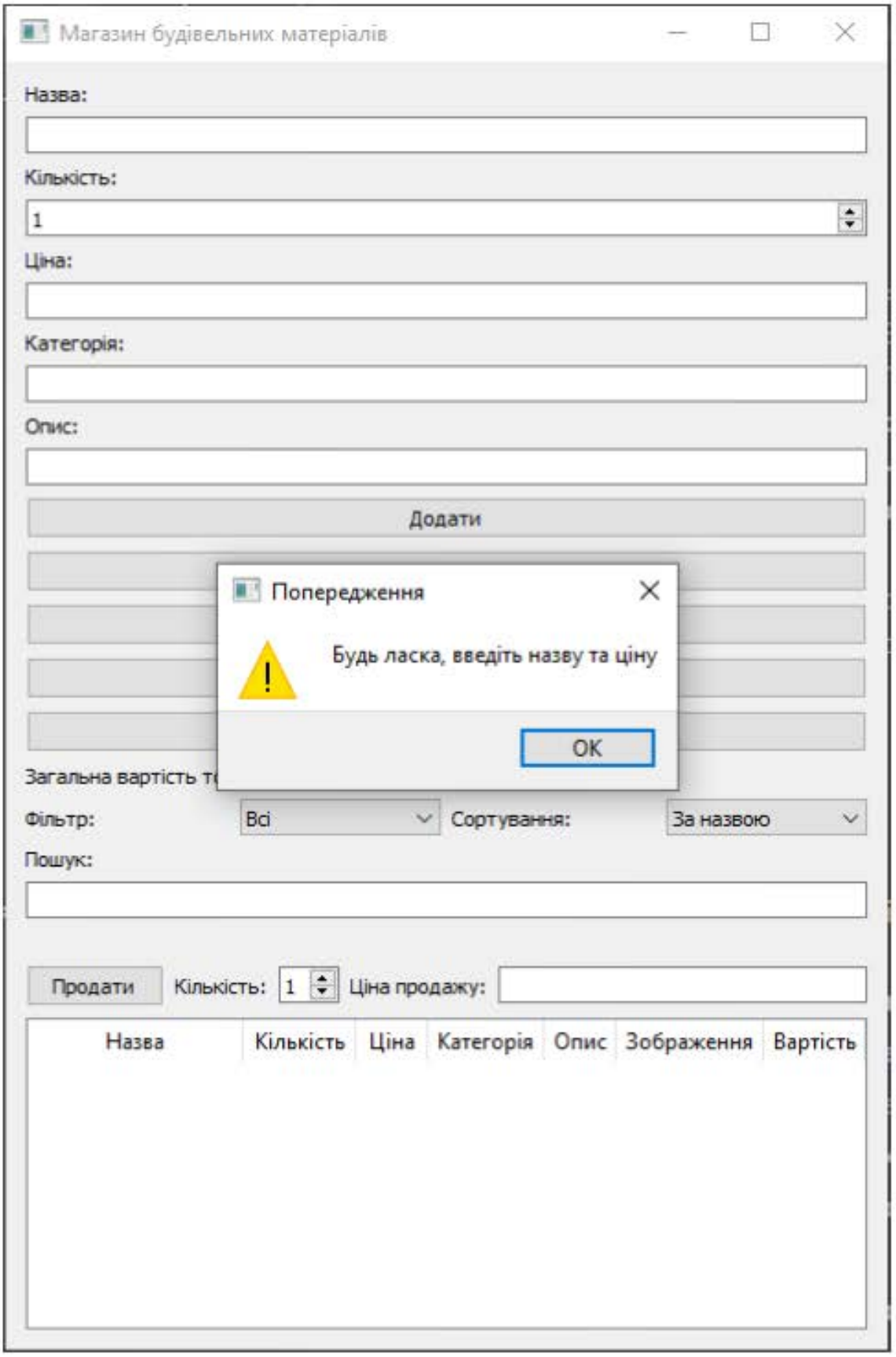

Рисунок 3.2 – Валідація та відповідне попередження про відсутність заповнених даних

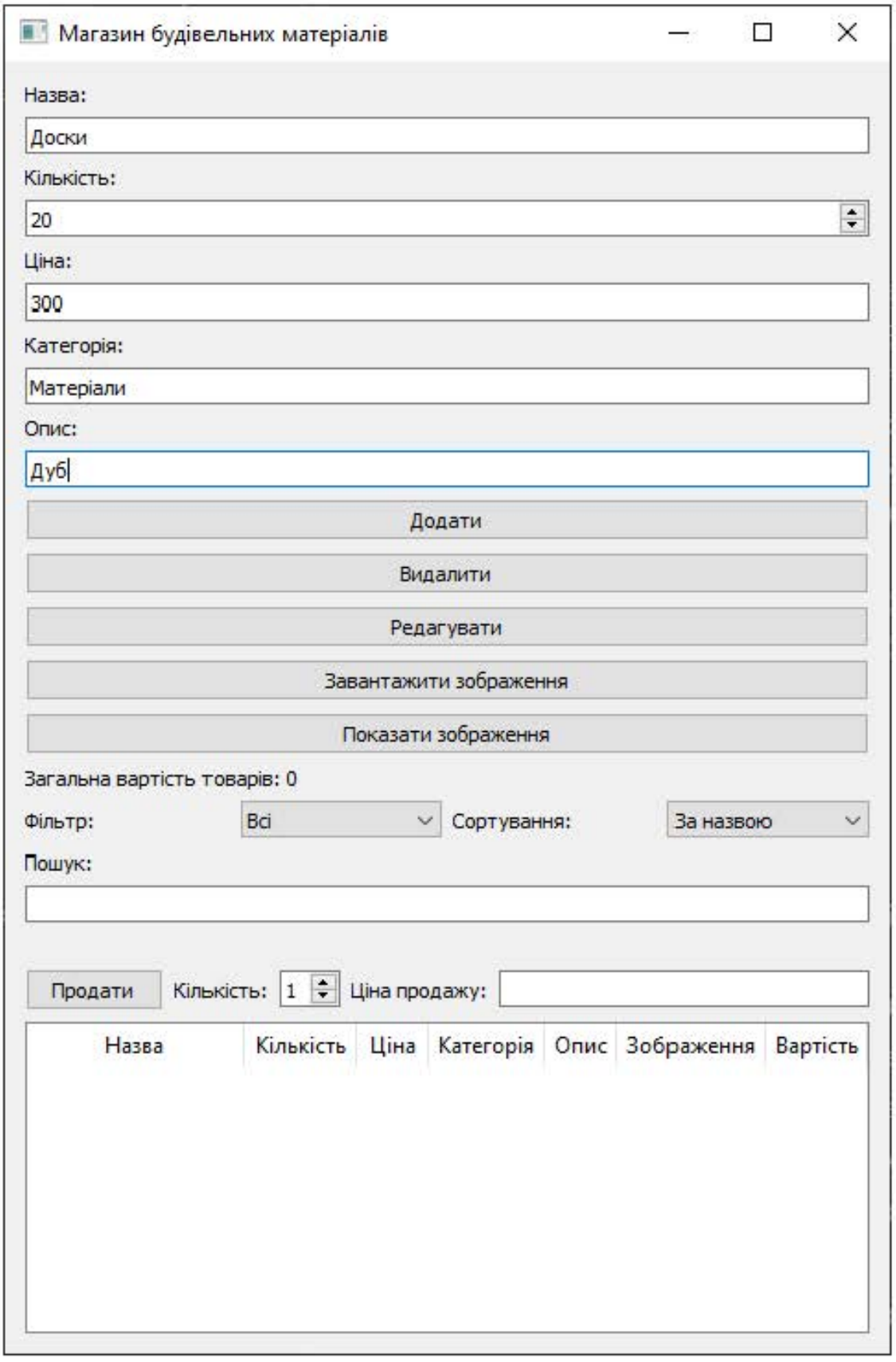

Рисунок 3.3 – Приклад заповнення даними

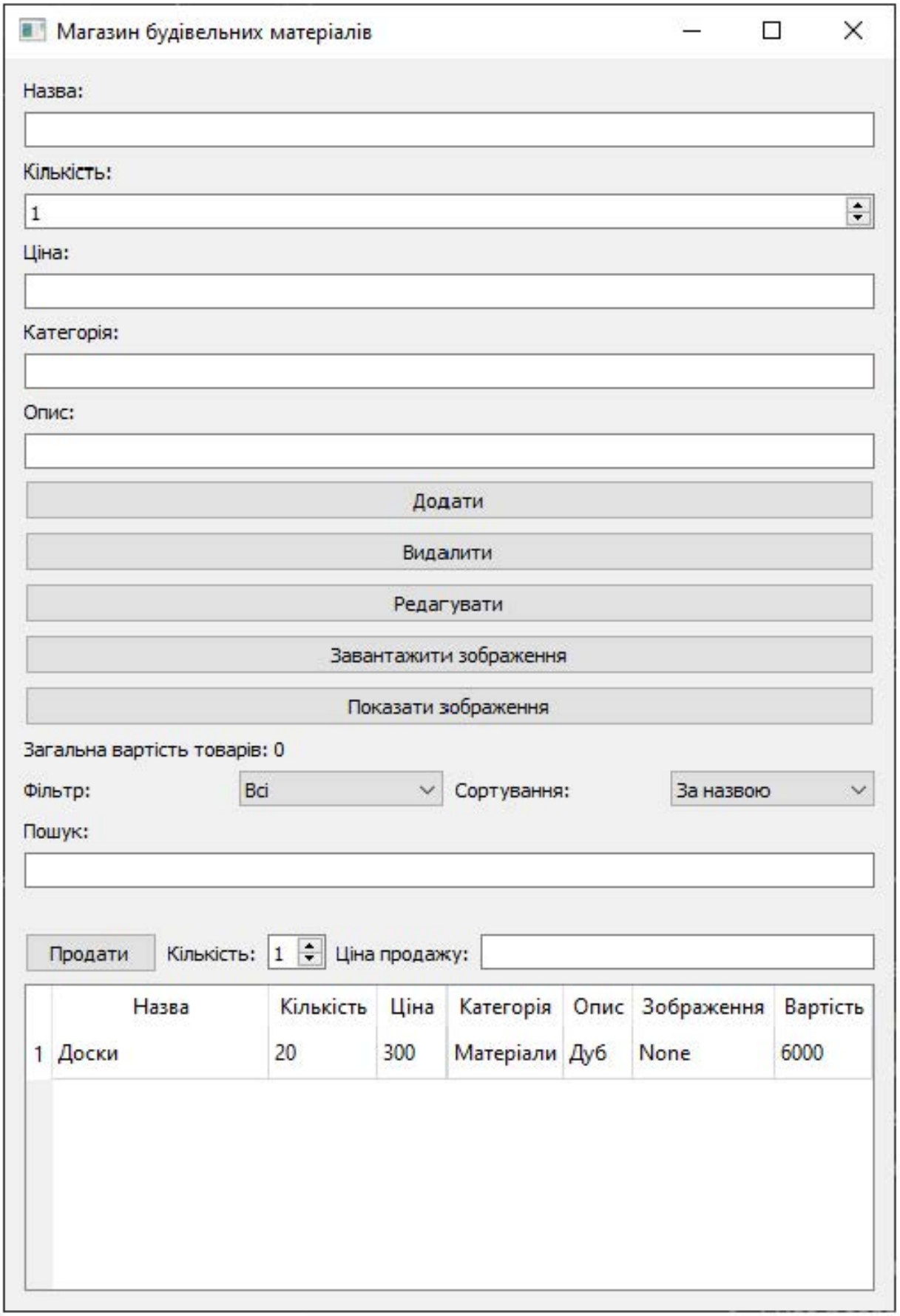

Рисунок 3.4 – Результат додавання нового товару

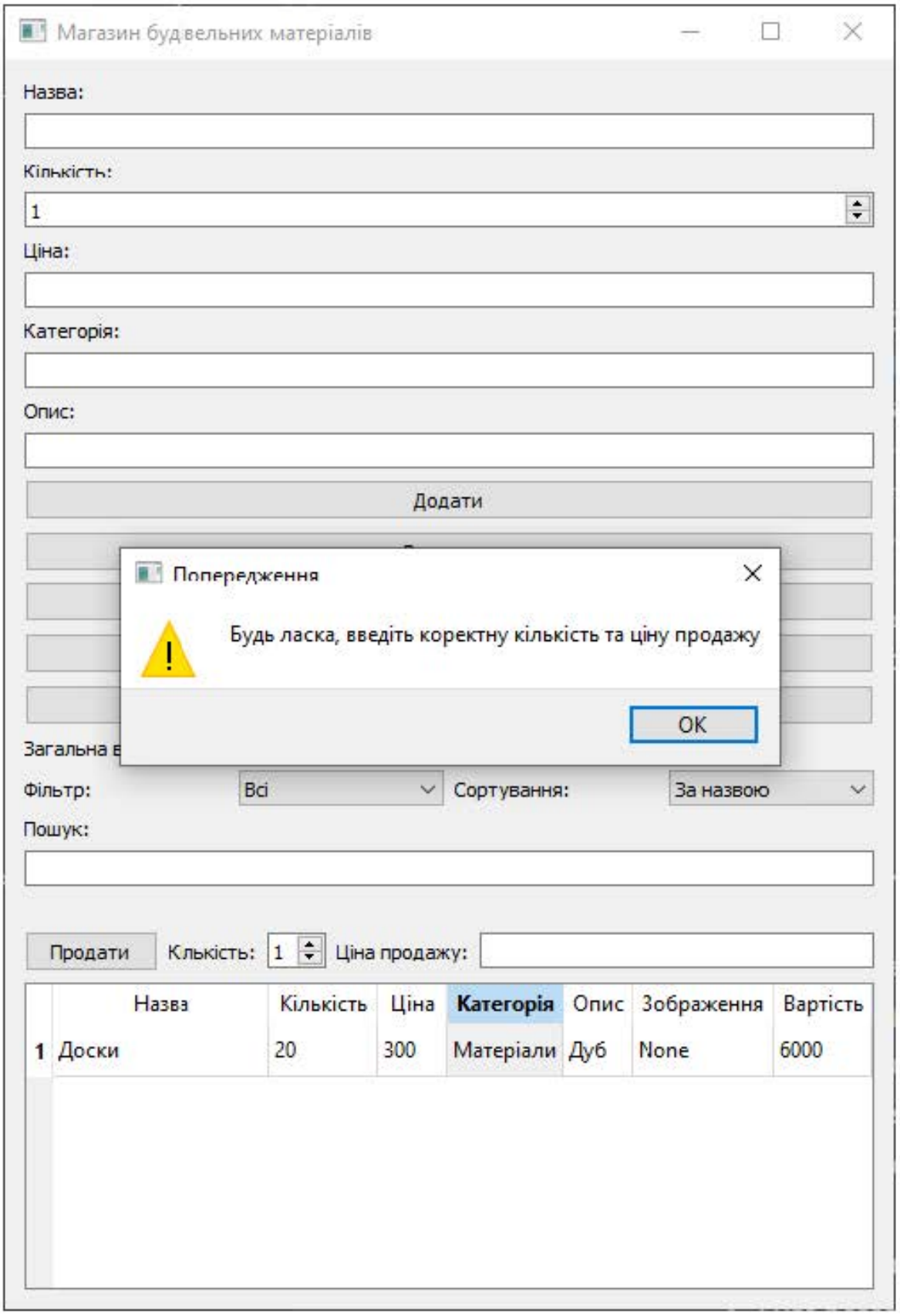

Рисунок 3.5 – Механізми валідації

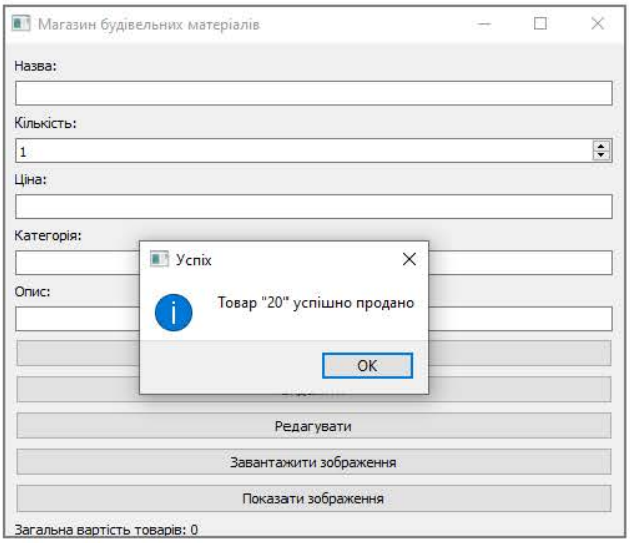

Рисунок 3.6 – Результат успішної продажи товару

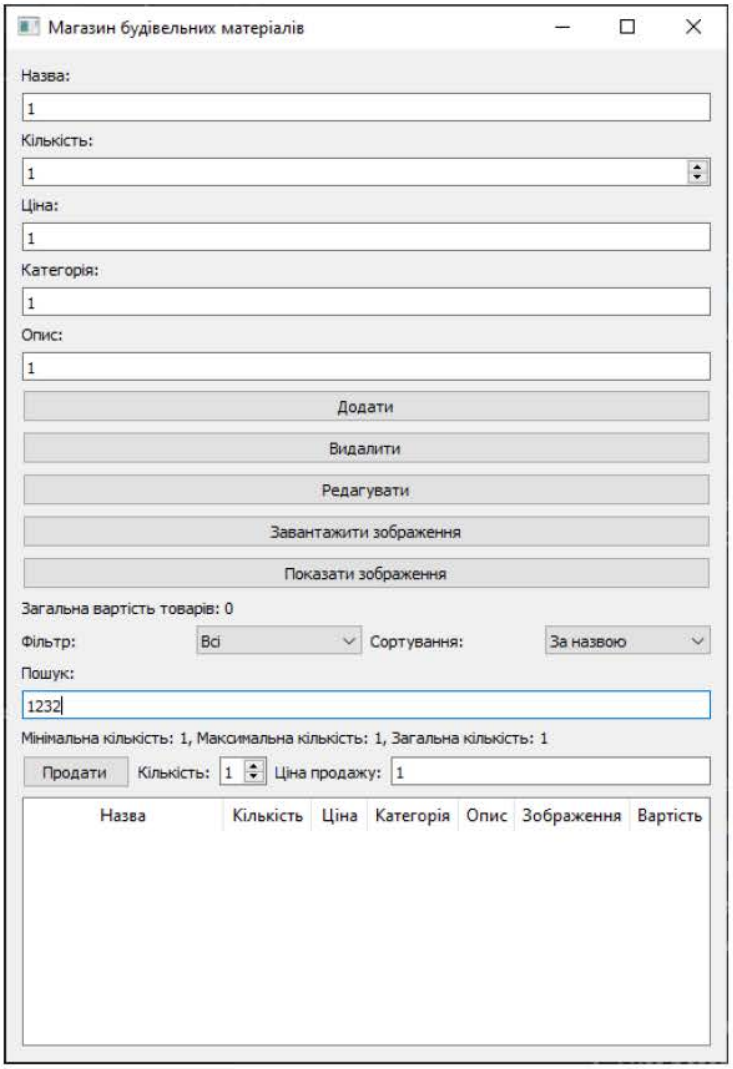

Рисунок 3.7 – Механізми пошуку

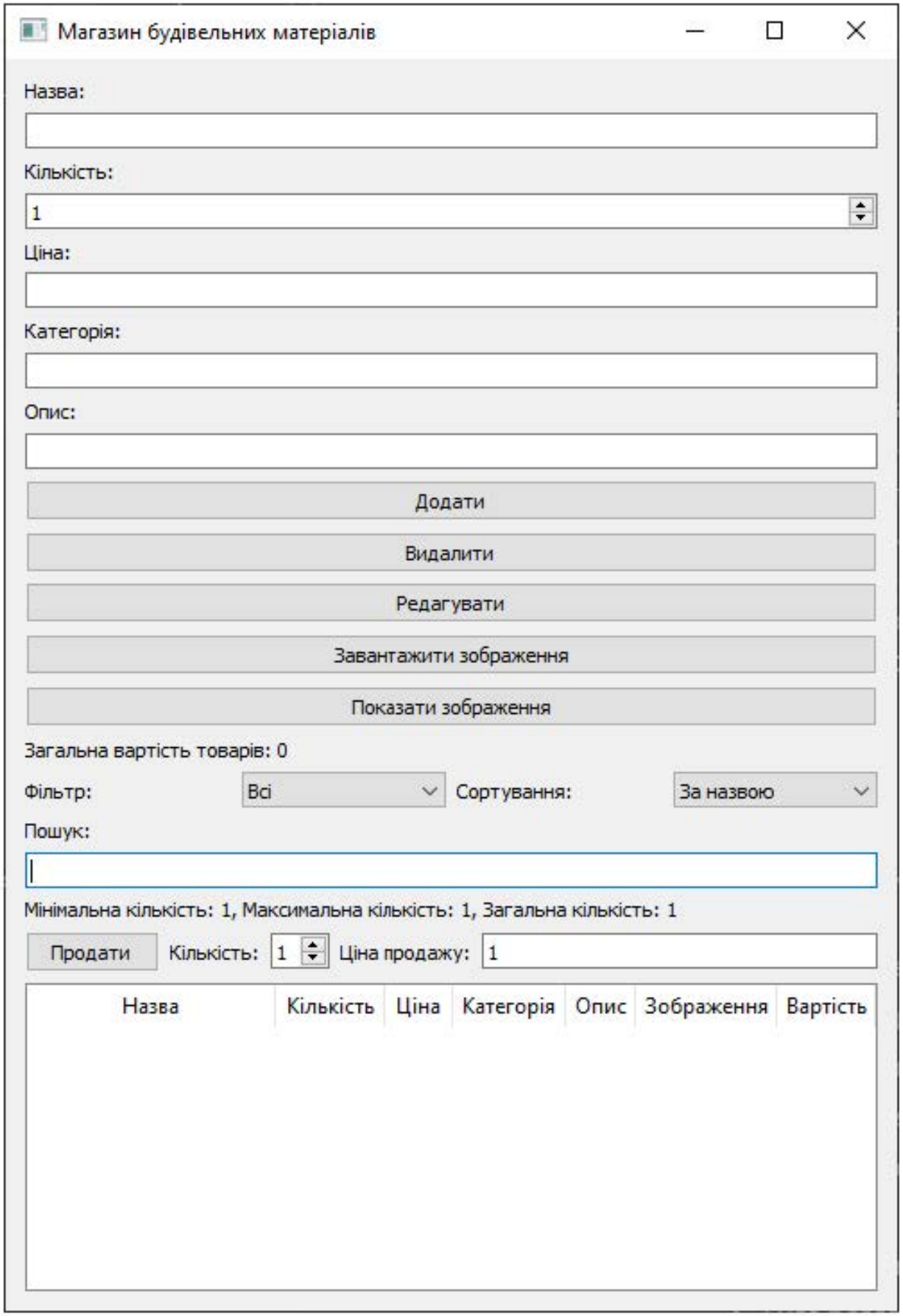

Рисунок 3.8 – Результат видалення даних

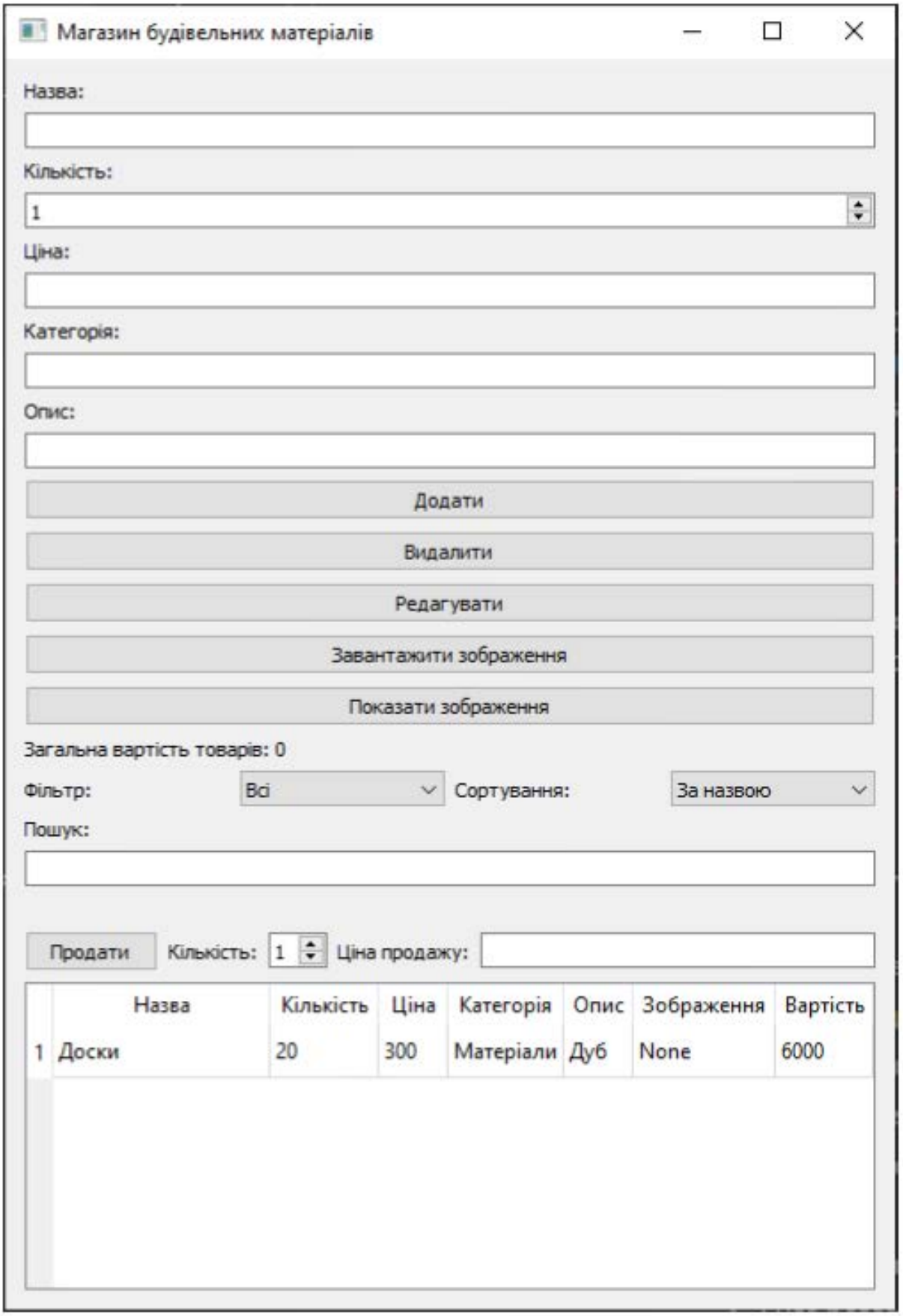

Рисунок 3.9 – Результат використання фільтра

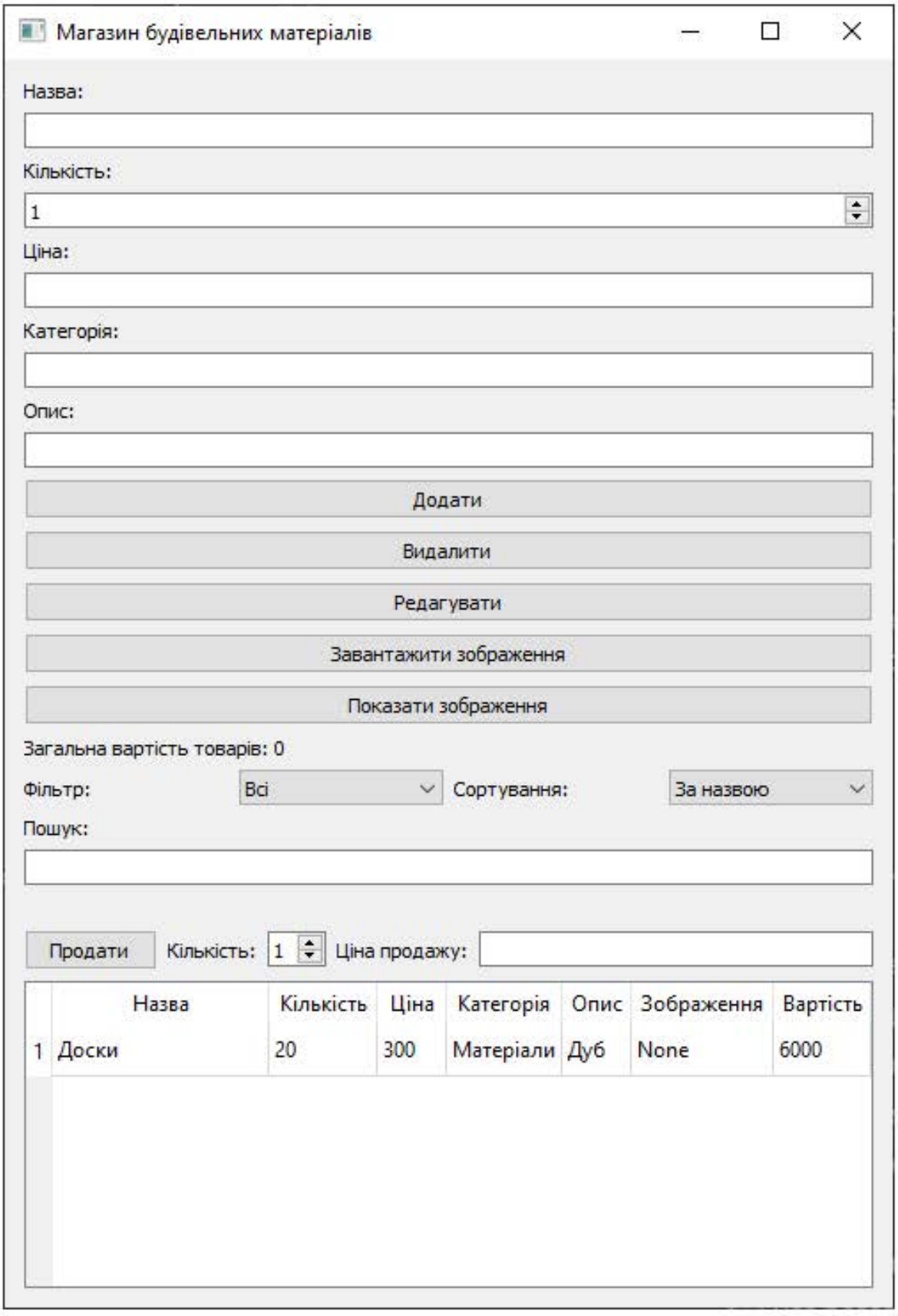

Рисунок 3.10 – Результат використання сортування

3.8 Висновки до третього розділу

В рамках цього розділу були сформульовані вимоги до програмного забезпечення для автоматизації роботи магазину будівельних матеріалів. Було проведено аналіз архітектури програми, яка включає компоненти такі, як графічний інтерфейс, база даних та логіка програми.

Була розглянута структура програми, включаючи різні класи та методи для роботи з даними, графічним інтерфейсом та базою даних. Була надана інформація про структуру бази даних, включаючи таблиці для товарів та продажів, а також опис колонок цих таблиць.

Було описано проектування інтерфейсу користувача, включаючи розміщення елементів, їх функціональність та зручність використання.

Було описано послідовність дій та операцій, які виконує програмне застосування під час своєї роботи, включаючи завантаження даних, взаємодію з користувачем та оновлення бази даних.

В цілому, розділ надає повний огляд розробленого програмного забезпечення для автоматизації роботи магазину будівельних матеріалів, включаючи архітектуру, структуру, базу даних, інтерфейс користувача та процес його роботи.

ВИСНОВКИ

Метою даної була розробка програмного забезпечення, спеціально адаптованого для автоматизації роботи магазину будівельних матеріалів. Це програмне рішення буде спрямоване на оптимізацію та покращення всіх ключових аспектів бізнес-процесів магазину, включаючи управління складом, облік товарів, обробку замовлень, взаємодію з постачальниками та обслуговування клієнтів.

Об'єктом дослідження у цьому контексті був процес автоматизації роботи магазину будівельних матеріалів з використанням програмного забезпечення. Це включає вивчення всіх аспектів бізнес-процесів цього магазину, починаючи з управління запасами і закінчуючи обслуговуванням клієнтів.

Предметом дослідження у цьому контексті була розробка програмного забезпечення для автоматизації роботи магазину будівельних матеріалів. Це включає конкретні технології, методики та інструменти, які будуть використовуватися для створення програмного рішення.

Дослідження предметної галузі дозволило оцінити існуючі програмні продукти для автоматизації роботи магазинів будівельних матеріалів. Цей аналіз допоміг визначити переваги та недоліки існуючих систем, а також виявити прогалини, які може заповнити програмне забезпечення, що розробляється.

Розгляд різних методів та підходів до розробки програмного забезпечення дало уявлення про найбільш ефективні способи створення програмного рішення для автоматизації роботи магазину будівельних матеріалів. Цей аналіз дозволив вибрати оптимальні інструменти та технології для реалізації проекту.

Аналіз засобів реалізації програмного забезпечення для автоматизації роботи магазину будівельних матеріалів дозволив визначити основні компоненти та технології, які будуть використовуватись у розробці. Це

включає вибір мов програмування, баз даних, бібліотек та інструментів для створення інтерфейсу користувача.

Описаний процес розробки програмного забезпечення для автоматизації роботи магазину будівельних матеріалів включає постановку завдання, аналіз архітектури та структури, розробку інтерфейсу користувача, процес роботи системи, тестування та формування висновків. Розроблене програмне рішення повинне задовольняти поставленим вимогам та завданням.

Кожен розділ роботи містить висновки, що ґрунтуються на проведених аналізах та результатах розробки. Вони дозволяють зробити узагальнені висновки про процес дослідження та розробки програмного забезпечення для автоматизації роботи магазину будівельних матеріалів, а також оцінити досягнення поставлених цілей та завдань.

В цілому, робота є комплексним аналізом та розробкою програмного забезпечення, спрямованого на підвищення ефективності та покращення процесів управління магазином будівельних матеріалів.

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 формулювати та забезпечувати вимоги щодо якості програмного забезпечення у відповідності з вимогами замовника, технічним завданням та стандартами;

– здатність дотримуватися специфікацій, стандартів, правил і рекомендацій в професійній галузі при реалізації процесів життєвого циклу;

– володіння знаннями про інформаційні моделі даних, здатність створювати програмне забезпечення для зберігання, видобування та опрацювання даних;

– здатність застосовувати фундаментальні і міждисциплінарні знання для успішного розв'язання завдань інженерії програмного забезпечення;

– 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;

– вибирати вихідні дані для проектування, керуючись формальними методами опису вимог та моделювання;

– застосовувати на практиці ефективні підходи щодо проектування програмного забезпечення;

– знати і застосовувати методи розробки алгоритмів, конструювання програмного забезпечення та структур даних і 10 знань;

– 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. Retailpro. URL: https://www.retailpro.com/

2. Shopify. URL: https://www.shopify.com/

3. Erpnext. URL: https://erpnext.com/

4. Woocommerce. URL: https://woocommerce.com/

5. Python. URL: https://python.org/

6. Mongodb. URL: https://www.mongodb.com/

7. Postgresql. URL: https://www.postgresql.org/

8. Campoli M. F. PostgreSQL for DBA: PostgreSQL 12. Independently Published, 2020. 395 p.

9. Martin S. PostgreSQL. Independently Published, 2017.

10. Qt Designer Documentation. URL: https://doc.qt.io/qt-6/qtdesigner-manual.html

11. Willman J. M. Creating GUIs with Qt Designer. Beginning PyQt. Berkeley, CA, 2020. P. 165–203. URL: https://doi.org/10.1007/978-1-4842-5857-6_7

12. QPixmap Class Documentation. URL: https://doc.qt.io/qt-6/qpixmap.html

13. QMessageBox Class Documentation. URL: https://doc.qt.io/qt-6/qmessagebox.html

14. QInputDialog Class Documentation. URL: https://doc.qt.io/qt-6/qinputdialog.html

ДОДАТОК А

ЛІСТИНГ ПРОГРАМНОГО КОДУ

```
import sys
import json
import sqlite3
from PyQt5.QtWidgets import QApplication, QWidget, QVBoxLayout, QHBoxLayout, QLabel, QLineEdit, QPushButton, QMessageBox, QTableWidget, QTableWidgetItem, QHeaderView, QSpinBox, QComboBox, QFileDialog
from PyQt5.QtGui import QPixmap


class InventoryApp(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle('Магазин будівельних матеріалів')
        self.setGeometry(100, 100, 1000, 500)

        self.init_ui()
        self.create_database()
        self.load_data()

    def init_ui(self):
        # Створення віджетів
        self.name_label = QLabel('Назва:')
        self.name_input = QLineEdit()
        self.quantity_label = QLabel('Кількість:')
        self.quantity_input = QSpinBox()
        self.quantity_input.setMinimum(1)
```

```
self.price_label = QLabel('Ціна:')
self.price_input = QLineEdit()
self.category_label = QLabel('Категорія:')
self.category_input = QLineEdit()
self.description_label = QLabel('Опис:')
self.description_input = QLineEdit()
self.add_button = QPushButton('Додати')
self.delete_button = QPushButton('Видалити')
self.edit_button = QPushButton('Редагувати')
self.load_image_button = QPushButton('Завантажити зображення')
self.show_image_button = QPushButton('Показати зображення')
self.total_label = QLabel('Загальна вартість товарів: 0')
self.filter_label = QLabel('Фільтр:')
self.filter_combo = QComboBox()
self.filter_combo.addItems(['Всі', 'Назва', 'Категорія'])
self.sort_label = QLabel('Сортування:')
self.sort_combo = QComboBox()
self.sort_combo.addItems(['За назвою', 'За кількістю'])
self.search_label = QLabel('Пошук:')
self.search_input = QLineEdit()
self.statistics_label = QLabel()


# Додавання кнопок для продажу
self.sell_button = QPushButton('Продати')
self.sell_quantity_label = QLabel('Кількість:')
self.sell_quantity_input = QSpinBox()
self.sell_quantity_input.setMinimum(1)
self.sell_price_label = QLabel('Ціна продажу:')
self.sell_price_input = QLineEdit()
```

```
        self.table = QTableWidget()


        # Налаштування таблиці
        self.table.setColumnCount(7)
        self.table.setHorizontalHeaderLabels(['Назва', 'Кількість', 'Ціна', 'Категорія',
'Опис', 'Зображення', 'Вартість'])
        header = self.table.horizontalHeader()
        header.setSectionResizeMode(0, QHeaderView.Stretch)
        header.setSectionResizeMode(1, QHeaderView.ResizeToContents)
        header.setSectionResizeMode(2, QHeaderView.ResizeToContents)
        header.setSectionResizeMode(3, QHeaderView.ResizeToContents)
        header.setSectionResizeMode(4, QHeaderView.ResizeToContents)
        header.setSectionResizeMode(5, QHeaderView.ResizeToContents)
        header.setSectionResizeMode(6, QHeaderView.ResizeToContents)


        # Розміщення віджетів на екрані
        form_layout = QVBoxLayout()
        form_layout.addWidget(self.name_label)
        form_layout.addWidget(self.name_input)
        form_layout.addWidget(self.quantity_label)
        form_layout.addWidget(self.quantity_input)
        form_layout.addWidget(self.price_label)
        form_layout.addWidget(self.price_input)
        form_layout.addWidget(self.category_label)
        form_layout.addWidget(self.category_input)
        form_layout.addWidget(self.description_label)
        form_layout.addWidget(self.description_input)
```

```
form_layout.addWidget(self.add_button)
form_layout.addWidget(self.delete_button)
form_layout.addWidget(self.edit_button)
form_layout.addWidget(self.load_image_button)
form_layout.addWidget(self.show_image_button)
form_layout.addWidget(self.total_label)


form_layout_buttons = QHBoxLayout()
form_layout_buttons.addWidget(self.filter_label)
form_layout_buttons.addWidget(self.filter_combo)
form_layout_buttons.addWidget(self.sort_label)
form_layout_buttons.addWidget(self.sort_combo)
form_layout.addLayout(form_layout_buttons)


form_layout.addWidget(self.search_label)
form_layout.addWidget(self.search_input)
form_layout.addWidget(self.statistics_label)

# Додавання віджетів для продажу
form_layout_sell = QHBoxLayout()
form_layout_sell.addWidget(self.sell_button)
form_layout_sell.addWidget(self.sell_quantity_label)
form_layout_sell.addWidget(self.sell_quantity_input)
form_layout_sell.addWidget(self.sell_price_label)
form_layout_sell.addWidget(self.sell_price_input)
form_layout.addLayout(form_layout_sell)
```

```
    layout = QVBoxLayout()
    layout.addLayout(form_layout)
    layout.addWidget(self.table)

    self.setLayout(layout)


    # Підключення функцій до кнопок
    self.add_button.clicked.connect(self.add_item)
    self.delete_button.clicked.connect(self.delete_item)
    self.edit_button.clicked.connect(self.edit_item)
    self.load_image_button.clicked.connect(self.load_image)
    self.show_image_button.clicked.connect(self.show_image)
    self.filter_combo.currentIndexChanged.connect(self.filter_items)
    self.sort_combo.currentIndexChanged.connect(self.sort_items)
    self.search_input.textChanged.connect(self.search_items)


    # Підключення функції продажу до кнопки
    self.sell_button.clicked.connect(self.sell_item)


def create_database(self):
    try:
        connection = sqlite3.connect('inventory.db')
        cursor = connection.cursor()
        cursor.execute('''
            CREATE TABLE IF NOT EXISTS products (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                name TEXT NOT NULL,
```

```
            quantity INTEGER NOT NULL,
            price REAL NOT NULL,
            category TEXT NOT NULL,
            description TEXT NOT NULL,
            image_path TEXT
        )
    ''')
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS sales (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            product_id INTEGER NOT NULL,
            quantity INTEGER NOT NULL,
            price REAL NOT NULL,
            date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
            FOREIGN KEY (product_id) REFERENCES products(id)
        )
    ''')
    connection.commit()
    connection.close()
except sqlite3.Error as error:
    print('Помилка при створенні бази даних:', error)


def add_item(self):
    name = self.name_input.text()
    quantity = self.quantity_input.value()
    price = self.price_input.text()
    category = self.category_input.text()
    description = self.description_input.text()
```

```
        # Перевірка наявності введених даних
        if not name or not price:
            QMessageBox.warning(self, 'Попередження', 'Будь ласка, введіть назву та ціну')
            return

        try:
            connection = sqlite3.connect('inventory.db')
            cursor = connection.cursor()
            cursor.execute('''
                INSERT INTO products (name, quantity, price, category, description)
                VALUES (?, ?, ?, ?, ?)
            ''', (name, quantity, price, category, description))
            connection.commit()
            connection.close()
        except sqlite3.Error as error:
            print('Помилка при додаванні товару:', error)
        else:
            self.load_data()
            self.update_statistics()


    def delete_item(self):
        selected_row = self.table.currentRow()
        if selected_row >= 0:
            item_id = self.table.item(selected_row, 0).text()
            try:
                connection = sqlite3.connect('inventory.db')
                cursor = connection.cursor()
```

```
            cursor.execute('DELETE FROM products WHERE id = ?', (item_id,))
            connection.commit()
            connection.close()
        except sqlite3.Error as error:
            print('Помилка при видаленні товару:', error)
        else:
            self.load_data()
            self.update_statistics()


    def edit_item(self):
        selected_row = self.table.currentRow()
        if selected_row >= 0:
            item_id = self.table.item(selected_row, 0).text()
            new_quantity = self.quantity_input.value()
            new_price = self.price_input.text()


            if new_quantity <= 0 or not new_price:
                QMessageBox.warning(self, 'Попередження', 'Будь ласка, введіть коректну кількість та ціну')
                return


            try:
                connection = sqlite3.connect('inventory.db')
                cursor = connection.cursor()
                cursor.execute('''
                    UPDATE products
                    SET quantity = ?, price = ?, category = ?, description = ?
```

```
                WHERE id = ?
                        ''', (new_quantity, new_price, self.category_input.text(),
self.description_input.text(), item_id))
            connection.commit()
            connection.close()
        except sqlite3.Error as error:
            print('Помилка при редагуванні товару:', error)
        else:
            self.load_data()
            self.update_statistics()


    def load_image(self):
        selected_row = self.table.currentRow()
        if selected_row >= 0:
            file_path, _ = QFileDialog.getOpenFileName(self, 'Обрати зображення', '',
'Зображення (*.png *.jpg *.jpeg)')
            if file_path:
                item_id = self.table.item(selected_row, 0).text()
                try:
                    connection = sqlite3.connect('inventory.db')
                    cursor = connection.cursor()
                    cursor.execute('UPDATE products SET image_path = ? WHERE id =
?', (file_path, item_id))
                    connection.commit()
                    connection.close()
                except sqlite3.Error as error:
                    print('Помилка при завантаженні зображення:', error)
                else:
                    self.load_data()
```

```
    def show_image(self):
        selected_row = self.table.currentRow()
        if selected_row >= 0:
            image_path = self.table.item(selected_row, 5).text()
            if image_path:
                pixmap = QPixmap(image_path)
                if not pixmap.isNull():
                    pixmap = pixmap.scaledToWidth(300)
                     QMessageBox.information(self, 'Зображення', '', QMessageBox.Yes,
pixmap)


    def filter_items(self):
        filter_text = self.filter_combo.currentText()
        if filter_text == 'Всі':
            self.load_data()
        elif filter_text == 'Назва':
            self.search_items()
        elif filter_text == 'Категорія':
            self.search_items()


    def sort_items(self):
        sort_text = self.sort_combo.currentText()
        if sort_text == 'За назвою':
            self.table.sortItems(0)
        elif sort_text == 'За кількістю':
            self.table.sortItems(1)
```

```
    def search_items(self):
        search_text = self.search_input.text()
        filter_text = self.filter_combo.currentText()
        try:
            connection = sqlite3.connect('inventory.db')
            cursor = connection.cursor()
            if filter_text == 'Назва':
                cursor.execute('SELECT * FROM products WHERE name LIKE ?', ('%'
+ search_text + '%',))
            elif filter_text == 'Категорія':
                cursor.execute('SELECT * FROM products WHERE category LIKE ?',
('%' + search_text + '%',))
            data = cursor.fetchall()
            connection.close()
            self.populate_table(data)
        except sqlite3.Error as error:
            print('Помилка при пошуку товарів:', error)


    def sell_item(self):
        selected_row = self.table.currentRow()
        if selected_row >= 0:
            item_id = self.table.item(selected_row, 0).text()
            item_name = self.table.item(selected_row, 1).text()
            item_quantity = self.sell_quantity_input.value()
            item_price = self.sell_price_input.text()
            current_quantity = int(self.table.item(selected_row, 2).text())
```

```
        if item_quantity <= 0 or item_quantity > current_quantity or not item_price:
            QMessageBox.warning(self, 'Попередження', 'Будь ласка, введіть коректну кількість та ціну продажу')
            return


        try:
            connection = sqlite3.connect('inventory.db')
            cursor = connection.cursor()
            cursor.execute('''
                UPDATE products
                SET quantity = ?
                WHERE id = ?
            ''', (current_quantity - item_quantity, item_id))
            cursor.execute('''
                INSERT INTO sales (product_id, quantity, price)
                VALUES (?, ?, ?)
            ''', (item_id, item_quantity, item_price))
            connection.commit()
            connection.close()
        except sqlite3.Error as error:
            print('Помилка при продажу товару:', error)
        else:
            self.load_data()
            QMessageBox.information(self, 'Успіх', f'Товар "{item_name}" успішно продано')


    def populate_table(self, data):
```

```
        self.table.setRowCount(0)
        for row_number, row_data in enumerate(data):
            self.table.insertRow(row_number)
            for column_number, column_data in enumerate(row_data):
                                    self.table.setItem(row_number,    column_number,
QTableWidgetItem(str(column_data)))


    def update_statistics(self):
        if self.table.rowCount() > 0:
                    quantities = [int(self.table.item(row, 1).text()) for row in
range(self.table.rowCount())]
            min_quantity = min(quantities)
            max_quantity = max(quantities)
            total_quantity = sum(quantities)
                self.statistics_label.setText(f'Мінімальна кількість: {min_quantity},
Максимальна кількість: {max_quantity}, Загальна кількість: {total_quantity}')
        else:
            self.statistics_label.setText('')


    def load_data(self):
        try:
            connection = sqlite3.connect('inventory.db')
            cursor = connection.cursor()
            cursor.execute('SELECT * FROM products')
            data = cursor.fetchall()
            connection.close()
            self.populate_table(data)
        except sqlite3.Error as error:
```

```
            print('Помилка при завантаженні даних:', error)


    def closeEvent(self, event):
        event.accept()


if __name__ == '__main__':
    app = QApplication(sys.argv)
    window = InventoryApp()
    window.show()
    sys.exit(app.exec_())
```