Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Інформаційна система керування комерційним складським приміщенням

Виконав студент 4 курсу, групи 2КІ-21б
спеціальності 123 — « Комп'ютерна інженерія »
освітня програма — « Комп'ютерна інженерія »
Рябоконь К. М.
Керівник к.т.н., доц. каф. ОТ
Дудник О. В.
Рецензент к.т.н., доц. каф. ПЗ
Чехместрук Р. Ю.

Допущено до захисту
Зав. кафедри ОТ
д.т.н., проф. Азаров О. Д.
"____"_____2025р.

ВНТУ 2025

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

Освітньо—кваліфікаційний рівень бакалавр

Галузь знань — 12 Інформаційні технології

Спеціальність — 123 Комп'ютерна інженерія

Освітня програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри обчислювальної техніки

д.т.н. проф. Азаров О.Д.

"21" березня 2025 р.

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Рябоконю Костянтину Михайловичу

1 Тема роботи «Інформаційна система керування комерційним складським приміщенням» керівник роботи Дудник Олександр Вікторович к.т.н., доцент, затверджено наказом вищого навчального закладу від 20 березня 2025 року № 97

2 Строк подання студентом роботи 13.05.2025р.

3 Вихідні дані до роботи: стандарти, монографії, підручники та наукові статті по темі, існуюче програмне забезпечення, яке стосується теми бакалаврської кваліфікаційної роботи.

4 Зміст розрахунково-пояснювальної записки: вступ, аналіз підходів до розробки віконних додатків, розробка структури та алгоритмів роботи органайзеру, програмна реалізація та тестування програми, висновки.

5 Перелік ілюстративного матеріалу: відображення сторінки клієнтів, відображення сторінки стелажів, сторінки товарів, зображення аналітики.

6 Консультанти розділів бакалаврської кваліфікаційної роботи.

Таблиця 1— Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата		
		завдання видав	завдання прийняв	Підпис
1-3	Дудник О.В. к.т.н., доцент каф. ОТ	21.03.25	13.05.25	
Нормоконтроль	ас. доц.каф. ОТ. Швець С.І	14.05.25	30.05.25	

7 Дата видачі завдання 21.03.2025р.

8 Календарний план виконання БКР приведений в таблиці 2.

Таблиця 2 — Календарний план

№	Назва та зміст етапу	Термін виконання		Приміт
		Початок	Закінчкення	
1	Визначення напрямку бакалаврської роботи, формулювання теми	21.03.25	30.05.25	
2	Аналіз предметної області обраної теми	21.03.25	30.05.25	
3	Розробка алгоритму роботи	31.03.25	13.04.25	
4	Розробка програмної частини	14.04.25	27.04.25	
5	Тестування програмної частини	14.04.25	27.04.25	
6	Оформлення пояснювальної записки та ілюстративного матеріалу	28.04.25	11.05.25	
7	Перевірка якості виконання бакалаврської кваліфікаційної роботи та усунення недоліків	13.05.25	13.05.25	

Студент Рябоконь К.М.

Керівник к.т.н., доц. Дудник О.В

АНОТАЦІЯ

УДК 004.77

Рябоконь К.М. Інформаційна система керування комерційним складським приміщенням.Кваліфікаційна робота зі спеціальності 123 – комп'ютерна інженерія, освітня програма – комп'ютерна інженерія. Вінниця: ВНТУ, 2025. 79 с.,

На укр.мові. Бібліогр.: 23 назв,рис., 6 табл., 4 дод.

У комплексній бакалаврській роботі спроектованно та розроблено інформаційну систему управління складським приміщенням.Для розробки було використано мову програмування Python із залученням бібліотеки PyQt5 та бази даних SQLite для збереження інформації

Робота містить опис вимог, структур класів, алгорти, програмну реалізацію та тестування

Ключові слова: облік, клієнт, стелаж, управління, PyQt5, GUI, python.

ABSTRACT

Ryabokon K.M. Information System for Managing a Commercial Warehouse Facility. Qualification Thesis in the Specialty 123 – Computer Engineering, Educational Program – Computer Engineering. Vinnytsia: VNTU, 2025. 70 pages, 18 figures, 6 tables, 3 appendices, 20 references.

During the course of the work, an information system for managing a warehouse facility was designed and developed. The development was carried out using the Python programming language with the PyQt5 library and the SQLite database for data storage.

The thesis includes a description of the system requirements, class structures, algorithms, software implementation, and testing.

Keywords: accounting, client, rack, management, PyQt5, GUI, Python.

.

ЗМІСТ

ВСТУП

У сучасних умовах стрімкого розвитку логістики та електронної комерції питання ефективного управління складськими приміщеннями набуває особливої **актуальності**. Комерційні склади виконують ключову роль у забезпеченні безперервності постачання товарів, що вимагає чіткої організації, контролю і швидкого реагування на зміни в попиті. У цьому контексті виникає потреба в розробці інформаційних систем, здатних оптимізувати роботу складу, підвищити її ефективність і знизити витрати.

Інформаційна система управління складом є сукупністю програмних та апаратних засобів, що забезпечують автоматизацію основних логістичних процесів: облік товарів, контроль залишків, обробку замовлень, планування поставок тощо. Такі системи дозволяють уникнути людських помилок, прискорюють обробку інформації та забезпечують прозорість усіх операцій.

Особливої ваги інформаційна система набуває для комерційного складського приміщення, яке обслуговує велику кількість клієнтів, партнерів та товарних груп. Забезпечення високого рівня сервісу вимагає точності, надійності й адаптивності управлінських рішень. Тому впровадження інформаційної системи є не лише вимогою часу, а й важливим конкурентним інструментом.

Метою цієї роботи є розробка інформаційної системи, здатної ефективно управляти процесами на комерційному складі. Така система повинна забезпечувати реальний контроль за товарними залишками, автоматизований а також підтримку аналітики.

Завдання дослідження:

— дослідити існуючі підходи до автоматизації складського обліку в комерційній сфері;

— визначити функціональні вимоги до інформаційної системи управління складом;

— розробити структуру та описати основні компоненти інформаційної системи.

Об’єктом дослідження є процеси управління комерційним складським приміщенням підприємства.

Предметом дослідження є інформаційна система, яка забезпечує автоматизоване управління складськими операціями та логістичними процесами.

Методи дослідження бакалаврської роботи: у роботі використано принципи об’єктно-орієнтованого програмування мовою Python для реалізації підходу.

1 АНАЛІЗ ПРОБЛЕМ І РІШЕНЬ СКЛАДСЬКОГО ОБЛІКУ

1.1 Аналіз існуючих систем складського обліку

Серед великого розмаїття програм розглянемо наступні три: ТоргСофт, УкрСклад та myWarehouse. В таблиці 1.1. показано порівняння цих трьох додатків.

Таблиця 1.1. Порівняння ТоргСофт, УкрСклад та myWarehouse

Параметр	ТоргСофт	УкрСклад	myWarehouse
Платформа	Windows (десктоп), онлайн через браузер, мобільний додаток	Windows (десктоп)	Android (мобільний додаток)
Тип ліцензії	Безстрокова, оренда, розстрочка	Безстрокова	Безкоштовна (з рекламою)
Ціна	Торгсофт-Старт: від 6 990 грн Торгсофт-Ультра: від 14 990 грн Оренда: від 1 090 грн/міс.	УкрСклад: 995 грн (перше місце), 500 грн (додаткове) УкрСклад Про: 1 595 грн (перше місце), 800 грн (додаткове)	Безкоштовно
Тестовий період	30 днів	45 днів	Не вказано
Підтримка	2 години безкоштовної техпідтримки при покупці; подальша підтримка — платна	Безкоштовна підтримка протягом 3 років після покупки	Через Google Play Store
Основний складський функціонал	Облік товарів і залишків Інвентаризація Внутрішні переміщення Місця зберігання Списання, повернення Синхронізація з інтернет-магазином	Облік товарів і залишків Інвентаризація Внутрішні переміщення Списання, повернення Робота з фіскальними реєстраторами (в Про-версії)	Облік товарів і залишків Інвентаризація Сканування штрих-кодів Підтримка Bluetooth-принтерів
Додаткові можливості	Синхронізація з Новою Поштою Робота з POS-терміналами SMS-розсилки Інтеграція з Binotel Мобільний додаток	Синхронізація з Новою Поштою (в Про-версії) Робота з POS-терміналами (в Про-версії) SMS-розсилки (в Про-версії) Інтеграція з Binotel (в Про-версії)	Підтримка Bluetooth-принтерів Сканування штрих-кодів Мобільний доступ

ТоргСофт — це програмне забезпечення, яке працює на платформі Windows, а також має версії для браузерів і мобільних пристроїв. Воно пропонується в трьох варіантах ліцензії, вартість яких варіюється від 6990 до 14990 грн, залежно від обраного функціоналу та необхідних можливостей. Важливою особливістю «ТоргСофт» є широкий спектр функцій, орієнтованих на бізнес, зокрема, облік товарів та інвентаризація. Завдяки інтеграції з інтернет-магазинами та можливості синхронізації з кур'єрськими службами, такими як Нова Пошта, програма дозволяє забезпечити зручний контроль за товарообігом і процесами доставки. Це рішення є популярним серед підприємців, які займаються торгівлею в режимі онлайн і потребують ефективного інструменту для управління своїм складом і комерційною діяльністю.Приклад застосунку показано на рисунку 1.1.

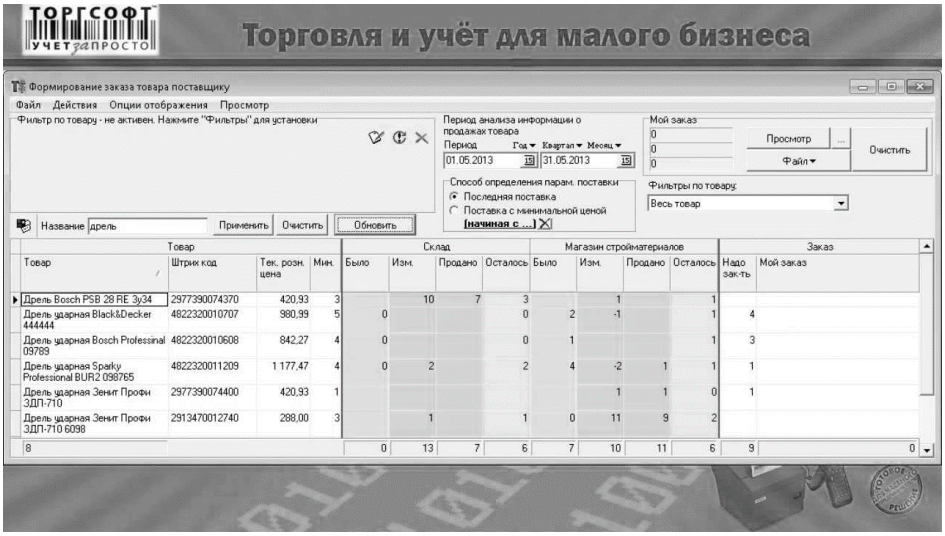

Рисунок 1.1 — Програма ТоргСофт

УкрСклад – це ще одне програмне забезпечення для ведення складського обліку, яке доступне тільки на платформі Windows. Воно пропонується за безстроковою ліцензією від 995 грн. Основними перевагами «УкрСклад» є простота використання та зручний інтерфейс, що забезпечує ефективну роботу з основними складськими функціями, такими як облік товарів, їх переміщення та зберігання. У додатковій Про-версії доступні розширені функції, які можуть бути корисними для великих підприємств або для тих, хто потребує більш детального налаштування обліку. Завдяки доступності безстрокової ліцензії, «УкрСклад» є

вигідним варіантом для бізнесів, які не хочуть регулярно витрачати кошти на оновлення або продовження ліцензії.Програма показана на рисунку 1.2

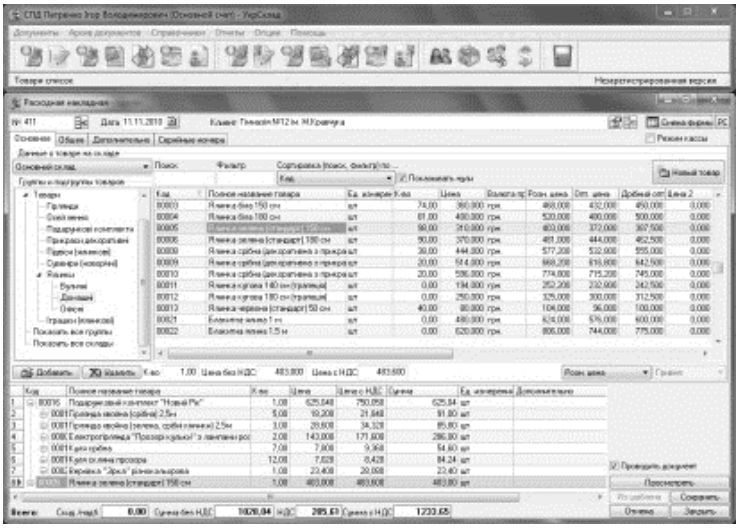

Рисунок 1.2 — Програма УкрСклад

MyWarehouse – це безкоштовне мобільне рішення для ведення складського обліку, яке доступне на платформі Android. Додаток підтримує Bluetooth-принтери та дозволяє сканувати штрих-коди, що робить його зручним для малих підприємств або підприємців, які працюють в роздрібній торгівлі. Хоча програма не має такого широкого функціоналу, як «ТоргСофт» або «УкрСклад», вона є хорошим варіантом для тих, хто шукає простий і доступний інструмент для ведення обліку на мобільних пристроях. Вона ідеально підходить для невеликих складських приміщень або точок продажу, де немає необхідності в складних функціях, але важлива мобільність і зручність у використанні.Застосунок показано на рисунку 1.3

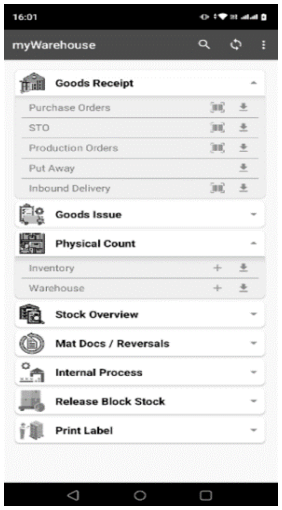

Рисунок 1.3 — Додаток MyWarehouse

1.2 Аналіз проблем системи складського обліку

У сучасних умовах, коли ринок вимагає високої швидкості обслуговування та прозорості логістичних операцій, управління складом стикається з численними викликами. Ефективна організація складського господарства є ключовою складовою успішної діяльності підприємств, особливо в комерційному секторі, де швидкість обробки замовлень і точність контролю залишків мають вирішальне значення. До основних проблем, що виникають при традиційній моделі управління складом, належать:

— високий ризик помилок при ручному введенні даних — невірний запис партій товарів, неправильне оновлення кількості залишків, дублювання записів у журналах;

— складність у відстеженні товару — без централізованої системи важко ідентифікувати місцезнаходження одиниці продукції або її історію руху;

— зниження оперативності обробки замовлень — відсутність автоматизації призводить до затримок при прийомі, відвантаженні та інвентаризації товару;

— недостатня аналітична підтримка — керівництво не має доступу до статистичних даних у режимі реального часу, що ускладнює планування закупівель або оцінку ефективності складу;

— високі витрати на обслуговування — велика кількість операційних процесів виконуються вручну, що вимагає значних трудових ресурсів;

— проблеми інтеграції з іншими підсистемами підприємства — склад працює ізольовано, що ускладнює взаємодію з бухгалтерією, логістикою, CRM.

Усі ці фактори знижують загальну ефективність складської логістики, сприяють накопиченню надлишкових товарів або, навпаки, дефіциту, що веде до втрат або невдоволення клієнтів.

1.3. Огляд існуючих мов програмування

На даний час існує кілька мов програмування з допомогою яких можна

реалізувати віконний додаток, до них відносяться:

— C/C++;

— C#;

— Java;

— Python.

Також є багато інших мов програмування з допомогою яких можна реалізувати віконний додаток проте вони або мають вужчу спеціалізацію або погано підтримуються. Тож розглянемо чотири основні мови.

C++ — це мова програмування загального призначення, створена Б’ярном Страуструпом у 1980-х роках як розширення мови C. Вона зберегла низькорівневу ефективність C, але при цьому додала механізми об’єктно-орієнтованого програмування, такі як класи, наслідування, інкапсуляція та поліморфізм. Саме це зробило C++ популярною мовою для розробки великих програмних систем, які вимагають ефективного використання ресурсів та гнучкості [1].

C++ підтримує кілька парадигм програмування, зокрема процедурне, об’єктно-орієнтоване та шаблонне програмування. Завдяки шаблонам можна створювати універсальні алгоритми та структури даних, які працюють з будь-якими типами даних. Стандартна бібліотека шаблонів (STL) надає готові до використання контейнери, ітератори та алгоритми, що значно спрощують розробку.

Мова активно розвивається. Починаючи з C++11, були додані нові можливості, такі як лямбда-вирази, розумні вказівники, покращена підтримка багатопоточності, auto-типи та інші зручності для сучасного програмування. Кожен новий стандарт робить мову більш виразною, безпечнішою та ефективнішою.

C++ широко використовується в системному програмуванні, ігровій індустрії, розробці драйверів, вбудованих систем, фінансових систем, CAD-програм та багатьох інших галузях. Її гнучкість і потужність дозволяють реалізовувати як низькорівневі компоненти, так і високорівневу логіку в межах одного середовища[3,4].

C++ являється мовою для розробки високопродуктивних програм, що разом з використанням бібліотеки Qt дозволяє широко контролювати ресурси системи.Розробка GUI та інтеграція до бази даних у данній мові програмування виявилась надмірно тяжкою.Саме через це C++ не було обрано мовою для програмування даного застосунку На лістингу 1.1. додаток в показано код C++ для вирішення квадратного рівняння.

C# (C-Sharp) — це сучасна об’єктно-орієнтована мова програмування, розроблена компанією Microsoft у межах платформи .NET. Вона була створена під керівництвом Андерса Гейлсберга й уперше представлена у 2000 році як частина ініціативи .NET Framework. C# поєднує в собі зручність мов високого рівня та строгість типізації, що дозволяє писати надійний, масштабований і безпечний код.

Мова C# має чітко визначену структуру та синтаксис, близький до C++ і Java. Вона підтримує всі основні принципи об’єктно-орієнтованого програмування — інкапсуляцію, спадкування та поліморфізм. Завдяки цьому C# підходить як для новачків, так і для досвідчених програмістів, яким важливо дотримуватись структурного підходу до розробки [5].

Особливістю C# є глибока інтеграція з екосистемою Microsoft. Розробники можуть створювати настільні програми (через Windows Forms або WPF), вебдодатки (через ASP.NET), мобільні застосунки (через Xamarin або MAUI) та хмарні сервіси (через Azure), використовуючи один і той самий синтаксис і підхід. Це робить C# універсальним інструментом у корпоративному програмуванні [6].

Із кожною новою версією C# розширюється. Наприклад, у версії 6 було додано зручні конструкції, такі як інтерполяція рядків, у C# 7 — шаблони й кортежі, у C# 8 — асинхронні потоки та Nullable Reference Types, які допомагають уникати NullReferenceException. Версія C# 9 додала record-типи для зручного створення незмінних об’єктів, а C# 10 продовжила оптимізувати синтаксис і читабельність коду.

Мова підтримує автоматичне керування пам’яттю через збирач сміття, що значно зменшує кількість помилок, пов’язаних із ресурсами. Крім того, C# має

сильну систему типів, що виявляє багато помилок ще на етапі компіляції. Це підвищує безпеку коду й полегшує його підтримку в довготривалих проектах [7].

Завдяки широкому набору бібліотек та API, C# дозволяє швидко реалізовувати складні функції без написання великої кількості коду. Наприклад, через LINQ можна ефективно працювати з колекціями даних, а через Entity Framework — взаємодіяти з базами даних без прямого написання SQL-запитів.

Середовище розробки Visual Studio, також створене Microsoft, пропонує потужні інструменти для розробки на C#: автозаповнення, підказки, відлагодження, профілювання й інтеграцію з системами контролю версій. Це зменшує поріг входження для новачків і підвищує продуктивність досвідчених розробників [8].

На сьогодні C# залишається однією з найпопулярніших мов програмування у світі. Її активно використовують у бізнесі, освіті, ігровій індустрії (зокрема через движок Unity) та наукових проєктах. З огляду на постійний розвиток .NET-платформи та відкритість C#, ця мова продовжує залишатися актуальною для широкого спектру завдань.Проте врахоіуючи потрби в кросплатформеності програми, C# не був основним вибором

На лістингу Г.2. додаток в показано код C# для вирішення квадратного рівняння.

Java — це мова програмування загального призначення, яка була створена компанією Sun Microsystems у 1995 році під керівництвом Джеймса Гослінга. Основною ідеєю Java є принцип «Write Once, Run Anywhere» — тобто програма, написана на Java, може виконуватись на будь-якій платформі, де встановлена віртуальна машина Java (JVM). Саме ця незалежність від операційної системи зробила Java однією з найпоширеніших мов у світі [9]. Java є строго типізованою об’єктно-орієнтованою мовою [10]. Всі об’єкти створюються на основі класів, а механізми спадкування, інкапсуляції та поліморфізму закладені в основу її структури. Також у мові реалізовано автоматичне керування пам’яттю — збирач сміття JVM самостійно звільняє ресурси, які більше не використовуються. Це спрощує розробку та зменшує ризик помилок, пов’язаних із пам’яттю [11].

Мова Java має велику стандартну бібліотеку, яка охоплює роботу з колекціями, файлами, мережею, багатопоточністю, базами даних, інтерфейсами користувача тощо. Вона активно застосовується у веброзробці (через сервлети, Spring), в Android-розробці, корпоративних системах, банківському ПЗ, наукових і освітніх проєктах.

З роками Java зазнала багатьох змін. Починаючи з Java 8, були додані лямбда-вирази, функціональні інтерфейси та потоки (Streams), що значно розширило функціональність мови і зробило її більш виразною. У пізніших версіях з’явилися нові типи даних, спрощення синтаксису (var, switch-вирази), а також покращена продуктивність JVM [12].

Java підтримує багатопоточність на високому рівні, що дозволяє розробляти складні паралельні системи. Крім того, завдяки платформі Java EE (нині Jakarta EE) Java широко використовується для створення масштабованих серверних застосунків з високою надійністю та безпекою.

Інструменти розробки для Java включають популярні IDE, такі як IntelliJ IDEA, Eclipse і NetBeans. Вони забезпечують повноцінне середовище для створення, тестування, відлагодження та розгортання програм. Велика спільнота, багаті документації та приклади значно полегшують навчання мови й вирішення практичних задач.

Java — це мова з довгою історією та стабільною архітектурою, яка продовжує активно розвиватись. Вона залишається популярною серед розробників завдяки надійності, масштабованості, кросплатформеності та величезному екосередовищу, що охоплює практично всі сфери програмування.

Мова Java являється більш стабільною та кросплатформенної а ніж попередні мови програмування завдяки своїм бібліотекам.Java може забезпечувати легке зєднання з базами даних та підримує ООП.Візальна частина а саме створення інерфейсу вимагає більших зусиль а ніж в інших мовах програмування

На лістингу Г.3. додаток в показано код Java для вирішення квадратного рівняння.

Python — це високорівнева мова програмування, створена Ґвідо ван

Россумом у 1991 році. Вона була розроблена з акцентом на читабельність коду та простоту синтаксису, що робить її однією з найзручніших для навчання. Основною філософією Python є принцип «зрозуміло краще, ніж заплутано», що відображено навіть у його офіційному маніфесті — "The Zen of Python" [13].

Мова підтримує кілька парадигм програмування: об’єктно-орієнтовану, процедурну та функціональну. Python дозволяє писати як невеликі скрипти, так і складні системи. Завдяки своїй гнучкості та багатій стандартній бібліотеці, Python широко використовується в різних сферах: від автоматизації завдань до складного наукового моделювання.

Однією з головних переваг Python є його лаконічний синтаксис [14]. Замість складних конструкцій, як у мовах C-подібного типу, у Python застосовуються прості інструкції та відступи для структурування коду. Це дозволяє швидше писати програми й легше їх підтримувати, особливо в командній роботі.

Python має величезну екосистему бібліотек і фреймворків. У галузі науки та аналізу даних популярні NumPy, Pandas, Matplotlib, SciPy. У машинному навчанні використовуються TensorFlow, PyTorch, Scikit-learn. Для веброзробки активно застосовуються Django та Flask. Завдяки цим інструментам Python став стандартом де-факто в аналітиці, дослідженнях та стартапах [15].

Важливою особливістю мови є її інтерпретованість: код виконується рядок за рядком, без попередньої компіляції. Це зручно для тестування та налагодження, однак у масштабних проєктах може позначатися на швидкодії. Тим не менш, розширення на C/C++ або JIT-компілятори, як-от PyPy, допомагають частково вирішити це обмеження.

Python також має активну спільноту, яка постійно підтримує та розвиває мову. Завдяки відкритому коду та прозорому процесу ухвалення змін, Python еволюціонує відповідно до потреб розробників. Нові версії мови регулярно включають корисні нововведення, зберігаючи сумісність і стабільність.

Мову підтримують найпопулярніші середовища розробки: PyCharm, VS Code, Jupyter Notebook. Вони забезпечують автодоповнення, інтеграцію з системами керування версіями, відлагодження та профілювання. Для новачків

існує багато інтерактивних платформ для навчання, що робить Python доступним навіть без технічного бекграунду [16].

Важливою особливістю мови є її інтерпретованість: код виконується рядок за рядком, без попередньої компіляції. Це зручно для тестування та налагодження, однак у масштабних проєктах може позначатися на швидкодії. Тим не менш, розширення на C/C++ або JIT-компілятори, як-от PyPy, допомагають частково вирішити це обмеження.

Мову підтримують найпопулярніші середовища розробки: PyCharm, VS Code, Jupyter Notebook. Вони забезпечують автодоповнення, інтеграцію з системами керування версіями, відлагодження та профілювання. Для новачків існує багато інтерактивних платформ для навчання, що робить Python доступним навіть без технічного бекграунду [16].

При використанні бібліотеки PyQt5 розробка гаффічного інтерфейсу являється максимально бистрою та зручною.Бібліотека PyQt5 дозволяє створювати сучасні кросплатформенні застосунки та забезпечує коспонування графічних елементів, а завдяки вбудованому модулю sqlite3 що дозволяє спростити роботу з базою даних SQLite.

На лістингу Г.4. додаток в показано код Python для вирішення квадратного рівняння.

Як можна зробити висновок з лістингів Г.1-Г.4 код написаний на пайтоні найкомпактніший. Крім того він не потребує компілятора для виконання і це ще кілька причин для вибору його як мови реалізації даної роботи.

1.4 Вибір IDE для розробки

Існує кілька середовищ розробки для мови програмування Python. Вибір якої є важливим етапом при розробці.Зручний інтрфейс для написання, налагодження та тестування є невід’ємною частиною роботи програміста.Інтеграція з іншими з іншими інструментами та зручність у підтримці коду є невід’ємною частиною роботи.Visual Studio Code,Pycharm, Jupyter Notebook саме ці середовища є популярними на данний момент тому далі розглянемо кожну і виберему кращу для

розробки данної системи

1.4.1 Огляд Visual Studio Code

Visual Studio Code (VS Code) — це легке, але потужне середовище розробки коду, створене компанією Microsoft. Воно є безкоштовним, з відкритим вихідним кодом і підтримує кросплатформену роботу — доступне для Windows, Linux та macOS. Завдяки своїй гнучкості VS Code набув величезної популярності серед програмістів різного рівня, від студентів до професійних розробників.

Однією з головних переваг VS Code є його розширюваність. Із базовою установкою середовище доволі мінімалістичне, але користувач може встановити тисячі розширень з офіційного marketplace. Доступні плагіни для різних мов програмування (Python, C++, Java, C#, JavaScript тощо), засобів форматування, відлагодження, роботи з базами даних, а також інтеграції з хмарними сервісами та системами контролю версій.

Інтерфейс VS Code побудований довкола простоти та ефективності. Він підтримує розділення редакторів, вкладки, інтелектуальне автодоповнення коду (IntelliSense), підсвічування синтаксису, пошук по проєкту, керування файлами та багато іншого. Навіть при роботі з великими проєктами середовище залишається швидким і стабільним.

Середовище має вбудований термінал, що дозволяє працювати з Git, запускати команди системи або скрипти без переключення в інші програми. Крім того, воно має потужну інтеграцію з Git і GitHub: підтримується перегляд змін, коміти, злиття, робота з гілками — усе через графічний інтерфейс.

Для відлагодження VS Code підтримує breakpoints, перегляд змінних, виклик стеку та інші типові інструменти дебагінгу. Після встановлення відповідного розширення користувач може налагоджувати код на Python, Node.js, C++, .NET, Java тощо. Усе це робить редактор універсальним для фронтенд- і бекенд-розробників.

Незважаючи на легку вагу, VS Code підтримує багато функцій повноцінної IDE. Це включає підтримку проєктів, конфігурацій запуску, тестування коду, а

також підтримку віддаленої розробки через SSH або WSL. Редактор також має режим Live Share, який дозволяє кільком розробникам спільно працювати над кодом у реальному часі.

VS Code постійно оновлюється, отримуючи нові можливості, виправлення помилок та оптимізації. Завдяки активній спільноті, зручному API для розробки розширень і відкритості проєкту, середовище зберігає свою актуальність та адаптується до потреб сучасної розробки.

На рисунку 1.4 показано інтерфейс VS Code.

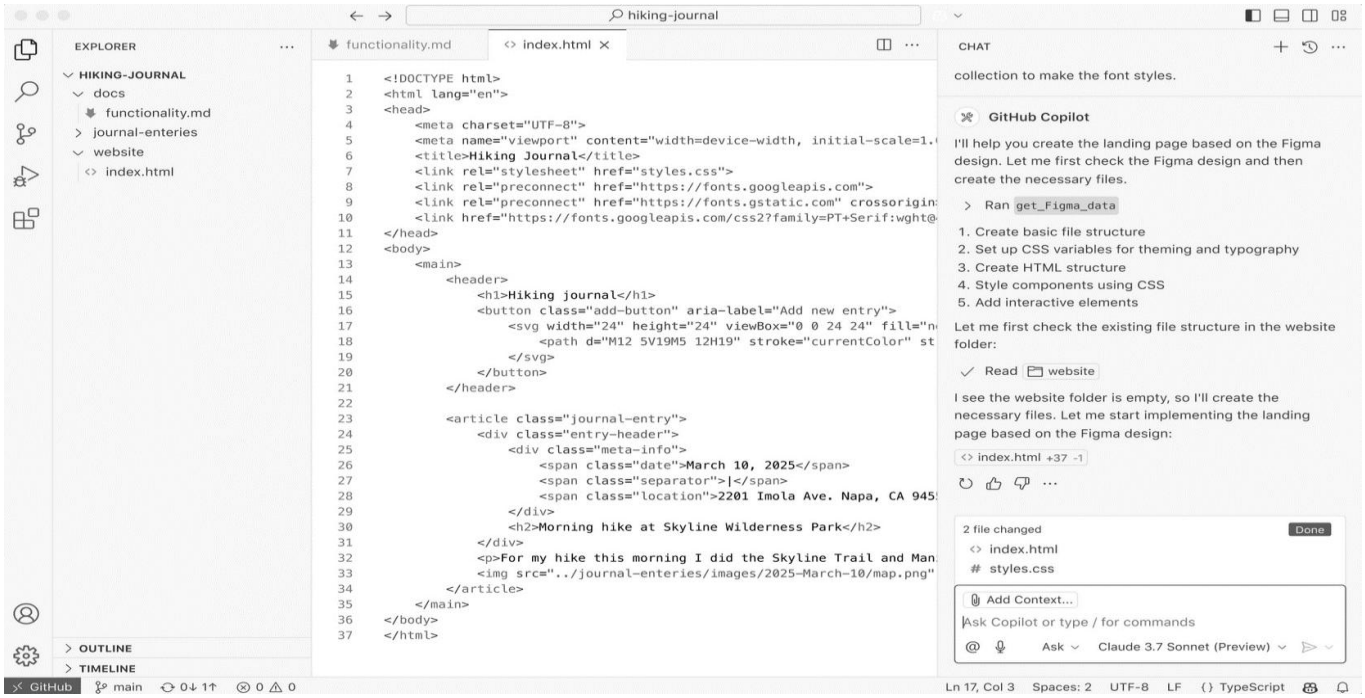

Рисунок 1.4 — Visual studio Code

1.4.2 Огляд PyCharm

PyCharm є одним з найпопулярніших інтегрованих середовищ розробки (IDE) для мови програмування Python. Розроблений компанією JetBrains, PyCharm надає потужний набір інструментів, що значно полегшують процес розробки, тестування та налагодження Python-кодів. Його функціонал включає розширене автодоповнення коду, рефакторинг, а також потужний аналіз коду, який дозволяє швидко знаходити помилки та пропонувати варіанти їх виправлення.

Однією з основних переваг PyCharm є вбудована підтримка різних

фреймворків і бібліотек для Python, таких як Django, Flask, Pyramid та інших. Завдяки інтеграції з цими фреймворками, розробник може значно швидше налаштувати середовище для роботи з ними, не вимагаючи додаткових налаштувань. Крім того, PyCharm має вбудовану підтримку для роботи з системами керування версіями, такими як Git, що дозволяє ефективно працювати в команді та управляти кодом.

Одним з важливих аспектів PyCharm є його потужні інструменти для налагодження коду. IDE дозволяє запускати програми в режимі налагодження, що дозволяє покроково перевіряти виконання коду, перевіряти значення змінних та навіть змінювати їх під час виконання програми. Це значно спрощує процес пошуку помилок і покращує ефективність роботи розробника.

PyCharm також має вбудований тестовий інтерфейс, що дозволяє запускати юніт-тести для перевірки правильності коду. Це забезпечує зручну інтеграцію з популярними бібліотеками для тестування, такими як pytest і unittest. Зокрема, підтримка різних методів тестування дозволяє зберігати високий рівень якості коду та знижувати ймовірність виникнення помилок у продакшн-середовищі.

Інтеграція з Docker та віртуальними середовищами, такими як venv і conda, є ще однією важливою перевагою PyCharm. Це дозволяє легко створювати ізольовані середовища для запуску та тестування Python-додатків без необхідності глобального встановлення залежностей. Таке налаштування значно спрощує процес роботи з проектами, що мають специфічні вимоги до середовища.

Інтерфейс PyCharm дуже гнучкий і зручний у використанні. Розробник може налаштувати вигляд середовища відповідно до своїх уподобань, змінюючи теми, розкладку вікон, гарячі клавіші та багато іншого. Це дозволяє значно підвищити продуктивність і зручність роботи в IDE, зменшуючи час, витрачений на пошук необхідних інструментів.

PyCharm також підтримує роботу з базами даних, що дозволяє розробникам безпосередньо підключатися до різних СУБД (наприклад, PostgreSQL, MySQL, SQLite). Це дозволяє виконувати запити та управляти даними без необхідності використовувати окремі інструменти для роботи з базами, що є зручним у багатьох

сценаріях розробки.

Окрім того, PyCharm має вбудовану підтримку для створення графічних інтерфейсів користувача, що дозволяє швидко розробляти настільні додатки за допомогою бібліотек, таких як Tkinter чи PyQt. Це особливо корисно для тих, хто працює з додатками, що мають візуальну частину, оскільки PyCharm дозволяє інтегрувати дизайнерські та програмні компоненти в єдину робоче середовище.

На рисунку 1.5 показано інтерфейс PyCharm.

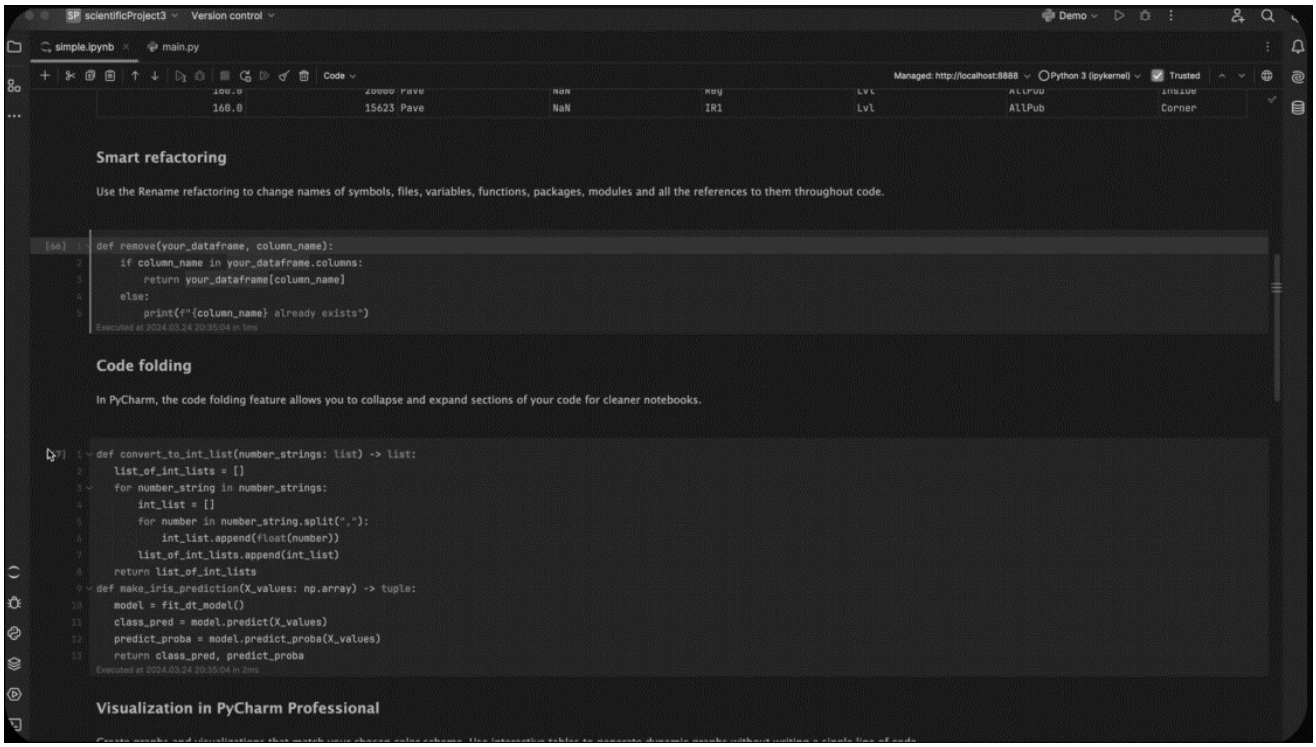

Рисунок 1.5 — PyCharm

1.4.3 Огляд Jupyter Notebook

Jupyter Notebook — це інтерактивне середовище для розробки, яке широко використовується для наукових обчислень, аналізу даних і машинного навчання. Це відкритий інструмент, який дозволяє створювати та обмінюватися документами, що містять живий код, рівняння, візуалізації та текстові пояснення. Він підтримує багато мов програмування, однак найчастіше використовується з Python. Завдяки такій гнучкості, Jupyter Notebook є популярним серед дослідників і розробників у різних галузях, таких як статистика, математика, наука про дані та

машинне навчання.

Однією з основних переваг Jupyter Notebook є можливість поєднувати виконуваний код з документованим контекстом, що робить його ідеальним для створення навчальних матеріалів, звітів та дослідницьких статей. У межах одного ноутбука можна розміщувати текст (Markdown), математичні рівняння, графіки та інтерактивні елементи, що значно покращує подання та аналіз даних.

Jupyter підтримує відразу кілька ядрових мов програмування, серед яких Python, R, Julia та інші. Однак найбільш поширеним є використання Python, оскільки Jupyter Notebook ідеально інтегрується з бібліотеками для аналізу даних, такими як NumPy, pandas, Matplotlib, Seaborn і SciPy. Це дозволяє зручно виконувати обчислення, проводити аналіз даних та створювати візуалізації прямо в середовищі ноутбука.

Процес роботи з Jupyter Notebook простий і зручний: кожен блок коду може бути виконаний окремо, що дозволяє розробникам тестувати та відлагоджувати код поетапно, не запускаючи всю програму. Це особливо корисно при обробці великих наборів даних або виконанні складних обчислень, коли потрібно отримати результати в реальному часі.

Для збереження роботи, Jupyter Notebook використовує власний формат файлів .ipynb, який зберігає не лише код, а й всі візуалізації, графіки та текстову інформацію. Такі файли можуть бути відкриті та виконані на будь-якому комп'ютері, на якому встановлено Jupyter, що робить їх зручними для спільної роботи та обміну результатами.

У Jupyter Notebook також є можливість інтеграції з іншими інструментами для виконання більш складних завдань, такими як Apache Spark для аналізу великих даних або TensorFlow і Keras для машинного навчання. Це дозволяє користувачам розширювати можливості ноутбука, працюючи з передовими технологіями без необхідності виходити за межі середовища.

Однією з цікавих можливостей є інтерактивні віджети, які дозволяють користувачам взаємодіяти з даними в реальному часі. Це можуть бути слайдери, кнопки та інші елементи, що дозволяють маніпулювати параметрами та миттєво

бачити результат. Такі елементи є особливо корисними при демонстраціях та презентаціях.

Незважаючи на всі переваги, Jupyter Notebook має і свої обмеження, зокрема при роботі з великими проектами, де потрібна більш складна структура коду. У таких випадках рекомендується використовувати інші інструменти для розробки, але для аналітичних задач і швидкого прототипування Jupyter Notebook є одним із найкращих варіантів.

На рисунку 1.6 показано інтерфейс Jupyter Notebook.

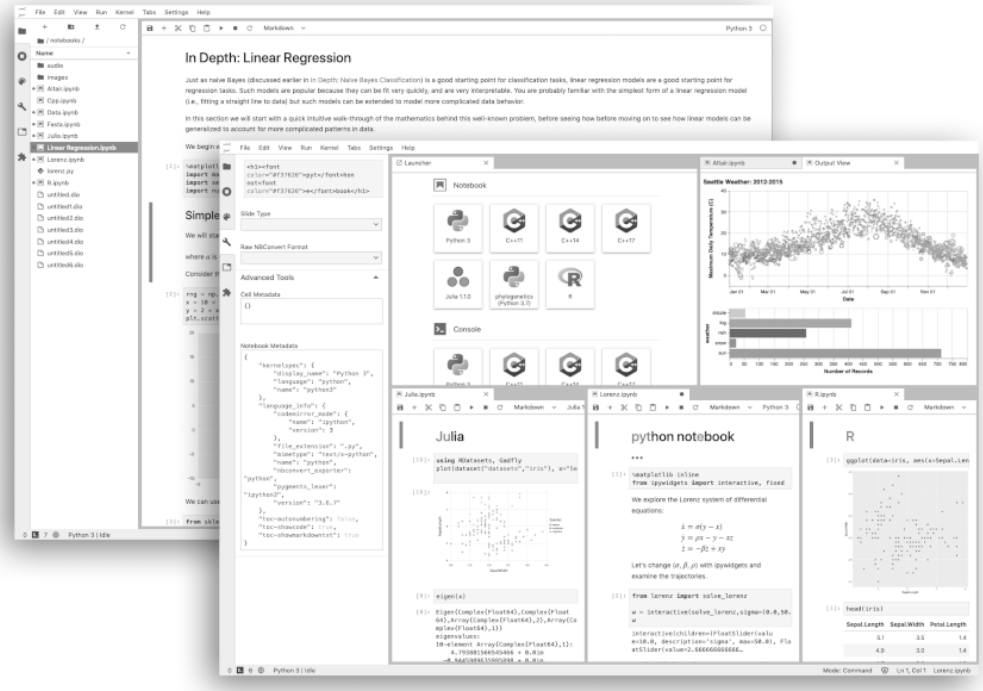

Рисунок 1.6 — Jupyter Notebook

1.5 Поняття ООП

Об’єктно-орієнтоване програмування (ООП) є методологією розробки програмного забезпечення, що базується на представленні сутностей предметної області у вигляді об’єктів. Кожен об’єкт є екземпляром певної абстрактної структури — класу, який визначає набір атрибутів (властивостей) та методів (функцій), що формують інтерфейс об’єкта. Такий підхід дозволяє моделювати складні системи, ізолювати функціональні блоки й спростити підтримку коду за рахунок чіткої структуризації.

Клас виконує роль шаблону або специфікації, за якою створюються об’єкти з однаковою структурою, але з унікальними значеннями стану. Об’єкти мають власний набір даних і можуть взаємодіяти між собою через методи. Цей механізм реалізує концепцію представлення системи у вигляді сукупності співпрацюючих компонентів з автономною поведінкою.

Основоположними принципами ООП є інкапсуляція, наслідування, поліморфізм і абстрагування. Інкапсуляція забезпечує приховування внутрішньої реалізації класу та доступ до нього через публічні методи. Це дозволяє обмежити пряме маніпулювання станом об’єкта ззовні, що сприяє підвищенню надійності та контрольованості змін.

Наслідування — це механізм формування нових типів на основі вже існуючих, коли підклас (нащадок) отримує набір властивостей і методів базового класу.Данний функціонал було застосованно за допомогою

Лістинг 1.1 — Наслідування

```
class ClientDialog(BaseDialog):
  def __init__(...):
      ...
```

Це забезпечує повторне використання логіки та створення ієрархій типів із можливістю розширення функціональності або перевизначення поведінки.

Поліморфізм реалізується як здатність викликати методи однаково через спільний інтерфейс, незалежно від конкретного типу об’єкта. Він може бути статичним (перевантаження методів) або динамічним (перевизначення методів у спадкових класах).Під час виконаня завдання цей метод був реалізовано за допомогою такої частини коду

Лісинг 1.2 – Наслідування в коді

```
def create_card(self, data, type):
   if type == "client":
      ...
   elif type == "product":
      ...
```

Це дозволяє реалізовувати узагальнені алгоритми без прив’язки до конкретної реалізації.

Інкапсуляція за допомогою структури класів надає змогу приховати внутрішні методи та реалізацію, та надає лиш публічні методи для взаємодіїї з ними.Даний метод було описано в коді за допомогою такого методу

Лістинг 1.3 — Інкапсуляція в коді

```
def add_shelf(self, name, capacity):
    self.db.execute_query(
        "INSERT INTO shelves (name, capacity, status, tenant) VALUES (?, ?, 'Вільний', '')",
        (name, capacity),
        commit=True
    )
```

Абстрагування дає змогу виділити суттєві характеристики об’єкта та приховати другорядні деталі реалізації. Воно досягається шляхом використання абстрактних класів або інтерфейсів, які задають контракт, обов’язковий для реалізації в похідних структурах. Це підвищує модульність і полегшує тестування компонентів.

Застосування ООП є доцільним у випадках складних, масштабованих систем, де необхідна чітка декомпозиція, повторне використання коду та адаптивність до змін. Воно широко використовується в мовах, таких як Python, Java, C++, C# та ін., які мають відповідні синтаксичні засоби для підтримки об’єктної моделі.

У Python реалізація ООП здійснюється за допомогою визначення класів за допомогою ключового слова class, створення екземплярів через виклик конструктора, використання спеціальних методів (наприклад, __init__, __str__, __repr__) і механізмів спадкування. Мова підтримує як класичне, так і множинне спадкування, а також дозволяє створювати абстрактні базові класи через модуль abc.

Загалом, ООП є потужною парадигмою для створення розширюваних та підтримуваних програмних рішень, де структура системи проектується навколо об'єктів, які інкапсулюють стан і поведінку та взаємодіють відповідно до визначених протоколів.

На лістингах Г.5-Г.8 що в додатку Г показано основні принципи ООП мовою python

1.6 Оцінка ефективності складу

У межах дослідження було проведено аналіз ефективності традиційних підходів до управління складським приміщенням, зокрема тих, що базуються на ручному введенні даних, використанні паперових або електронних таблиць, та несистематизованих підходах до звітності. Такий підхід, попри свою простоту, вимагає значних затрат часу на кожну рутинну операцію та є вразливим до людського фактору.

Згідно з результатами дослідження, додавання нового клієнта вручну зазвичай займає 2–3 хвилини, оновлення інформації про товар або залишки — 3–5 хвилин, а формування звітності, особливо коли потрібно звести фінансові або логістичні показники, може тривати до 10–15 хвилин. Такі часові витрати є неприйнятними у швидкому комерційному середовищі, де важлива оперативність прийняття рішень та гнучкість логістики.

На етапі тестування розробленої інформаційної системи було здійснено вимірювання часу виконання аналогічних дій в середовищі програми. Виявлено, що додавання клієнта, оновлення товарної інформації та створення звітів у додатку займає в середньому від 1 до 2 хвилин у сукупності, завдяки автоматизованому інтерфейсу, структурованому зберіганню даних у базі SQLite та можливості швидкого доступу до них через графічний інтерфейс.

Таке порівняння демонструє прискорення рутинних операцій у 7–10 разів. Це не лише скорочує загальні часові витрати персоналу, а й мінімізує ризики помилок, які можуть виникати при ручному введенні даних. Крім того, автоматичне формування звітів і візуалізація аналітики в режимі реального часу значно покращує прозорість управління та дає можливість оперативно реагувати на зміни в товарообігу або заповненості складу.

Висока продуктивність системи, підтверджена результатами тестування, дає підстави вважати розроблену інформаційну систему ефективним інструментом для оптимізації складських процесів. Вона має потенціал до масштабування та

адаптації під потреби різних комерційних підприємств, що підтверджує доцільність подальшого її впровадження у виробничу діяльність та розвитку у напрямку розширення функціоналу.

2 РОЗРОБКА СТРУКТУРИ РОБОТИ ІС СКЛАДСЬКОГО ОБЛІКУ

2.1 Розробка інтерфейсу користувача

Для створення користувацького інтерфейсу було використано бібліотеку PyQt5. Це набір обгорток для мови програмування Python, що дозволяє створювати графічні інтерфейси користувача на базі бібліотеки Qt5. Цей інструмент забезпечує гнучкий та потужний спосіб створення віконних додатків, які можуть працювати на різних операційних системах без необхідності змінювати код.

Основною ідеєю PyQt5 є використання об'єктно-орієнтованого підходу до побудови інтерфейсів, де кожен елемент, такий як кнопка, напис чи поле вводу, представлений у вигляді об'єкта певного класу. Розробник створює ці об'єкти, розміщує їх у вікні і задає взаємодію між ними.

PyQt5 має потужну систему сигналів і слотів, яка дозволяє легко реагувати на події, наприклад, натискання кнопки або зміну тексту. Коли користувач виконує дію, відповідний сигнал надсилається об'єкту, який обробляє цю подію за допомогою функції-обробника.

Однією з важливих переваг PyQt5 є те, що він дозволяє використовувати як ручне створення інтерфейсу в коді, так і візуальне проектування за допомогою Qt Designer. Цей графічний інструмент дозволяє розробнику створювати макети інтерфейсу мишкою, а потім імпортувати їх у Python-код.

Завдяки PyQt5 можна створювати як прості вікна з кількома кнопками, так і складні багатовіконні додатки з меню, вкладками, діалогами, таблицями та іншими функціональними елементами. Бібліотека також підтримує стилізацію інтерфейсу через CSS-подібні стилі, що дозволяє гнучко налаштовувати зовнішній вигляд додатку.

PyQt5 має широку документацію і велике співтовариство розробників, тому знайти приклади або вирішення типових проблем досить просто. Це робить його хорошим вибором для тих, хто хоче створювати якісні графічні застосунки на Python.

Програми, створені за допомогою PyQt5, можуть бути упаковані в

інсталяційні файли для Windows, Linux або macOS з допомогою таких інструментів як PyInstaller чи cx_Freeze. Це дозволяє легко поширювати додатки без необхідності встановлювати Python на комп’ютері користувача.

Незважаючи на те, що PyQt5 безкоштовний для використання у відкритому програмному забезпеченні, для комерційного використання може знадобитися придбання ліцензії, оскільки він розповсюджується за подвійною ліцензією — GPL або комерційною.

Було реалізовано ініціалізуючу заставку показану на рисунку 2.1.

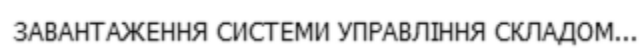

Рисунок 2.1 — Ініціалізуюча заставка

Після чого відкривається головне вікно програми з п’ятьма вкладками.

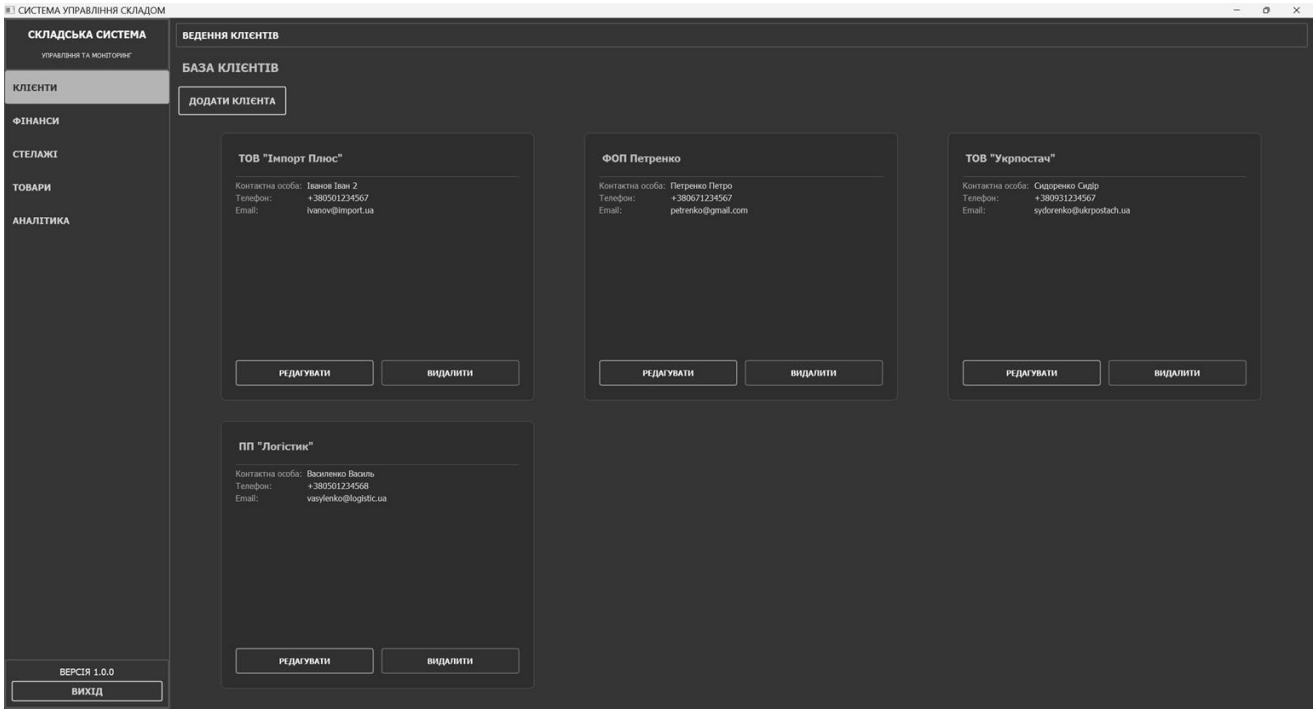

Рисунок 2.2 — Головне вікно програми

На вкладках розташована інформація про клієнтів, фінанси, стелажі, товари та вкладка аналітики. Розглянемо їх детальніше.

Вкладка клієнтів містить дані про клієнтів. Передбачені CRUD операції. Сам

інтерфейс реалізовано з використанням алаптиву, тобто елементи підлаштовуються під розмір вікна див рисунок 2.3.

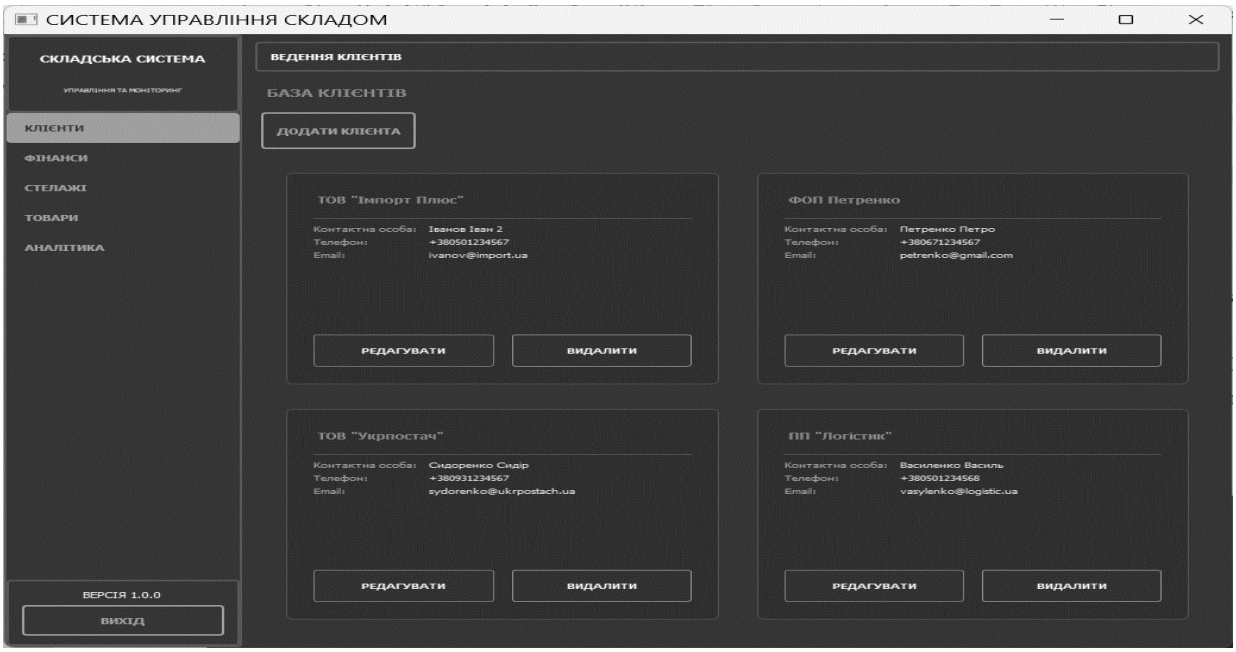

Рисунок 2.3 — Головне вікно програми

Даний підхід реалізовано і в інших вкладках. На рисунок 2.4 показана вкладка фінансів.

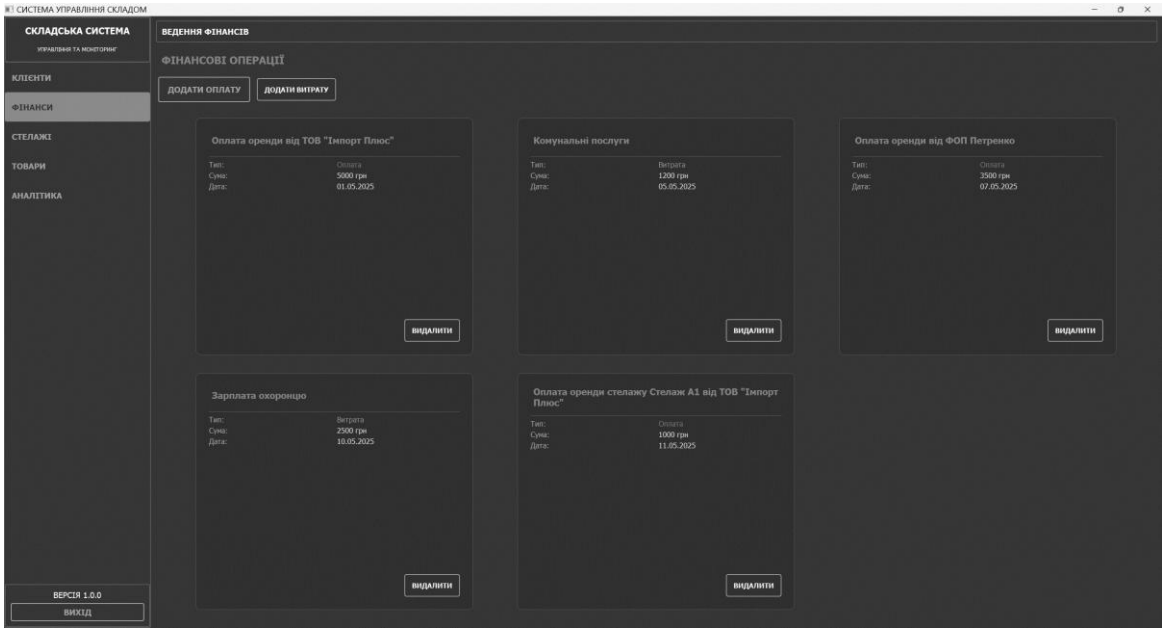

Рисунок 2.4 — Фінансові записи

У порівнянні з клієнтами фінансовий запис можна видалити. На рисунку 2.5 показана вкладка стелажів.

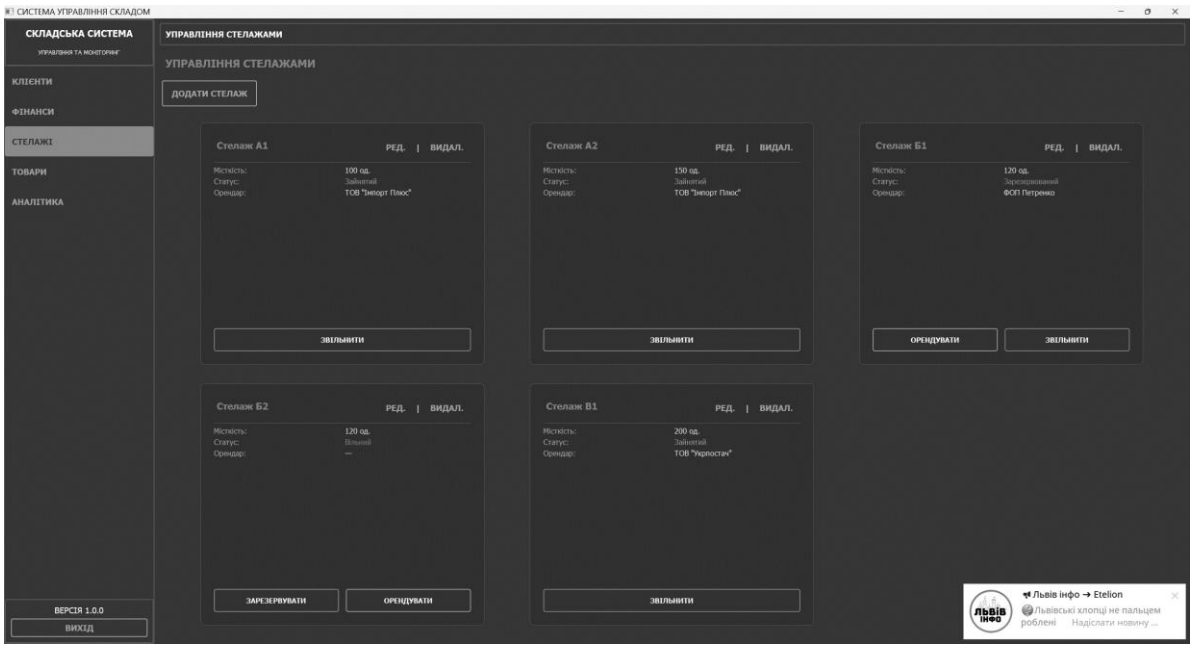

Рисунок 2.5 — Стелажі

На рисунку 2.6 показано вкладку товарів.

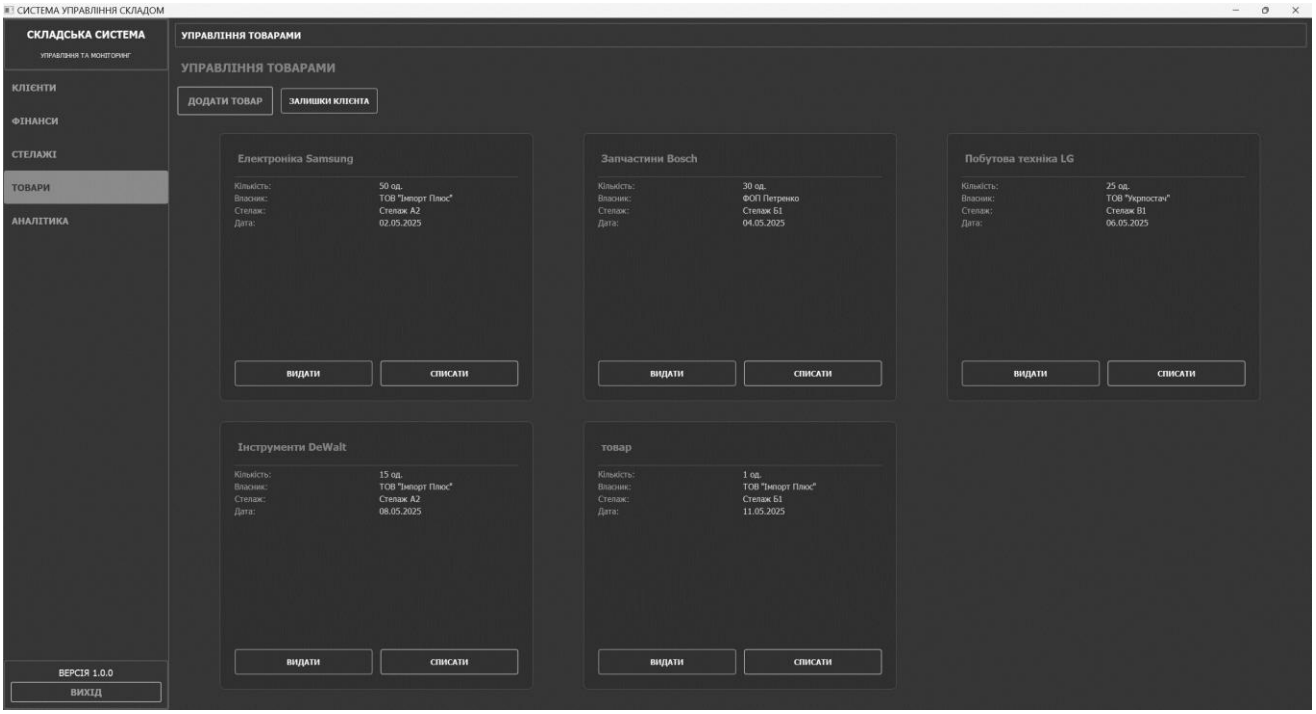

Рисунок 2.6 — Стелажі

Також з цієї вкладки можна згенерувати звіт про залишки товарів див рисунок 2.7.

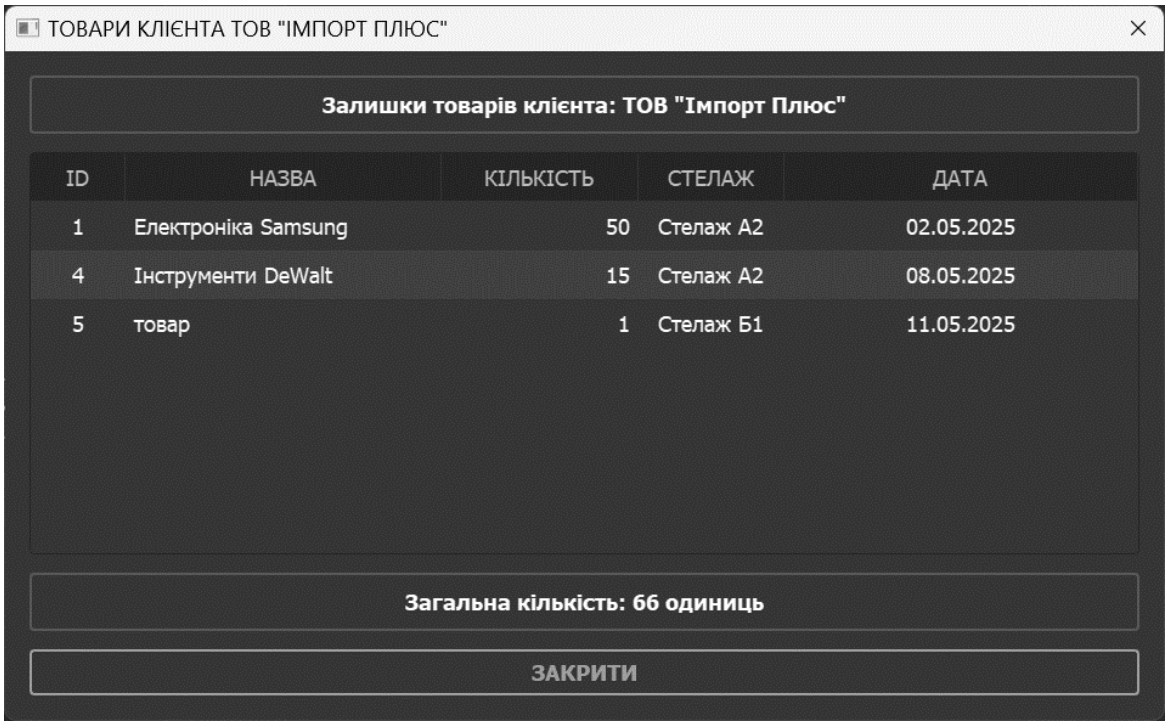

Рисунок 2.7 — Звіт про залишки товарів

Про вкладку аналітики варто розповісти детальніше. Там є можливість створити балансовий звіт, звіт про зайнятість стелажів, фін. звіт та звіт про клієнтів. Ці три звіти можуть бути у вигляді діаграми, таблиці чи деталізації. На рисунках 2.8-2.10 показано різні комбінації звітів та способів подання

Рисунок 2.8 — Балансовий звіт у формі діаграми

На рисунку 2.9. показано балансовий звіт у формі діаграми. В історії діаграми вказано суми та відсотки витрат та доходів. Також ці дані є у легенді діаграми. При перемиканні режиму на відображення діаграми сама діаграма з’являється з ефектом анімації задля зосередження уваги.

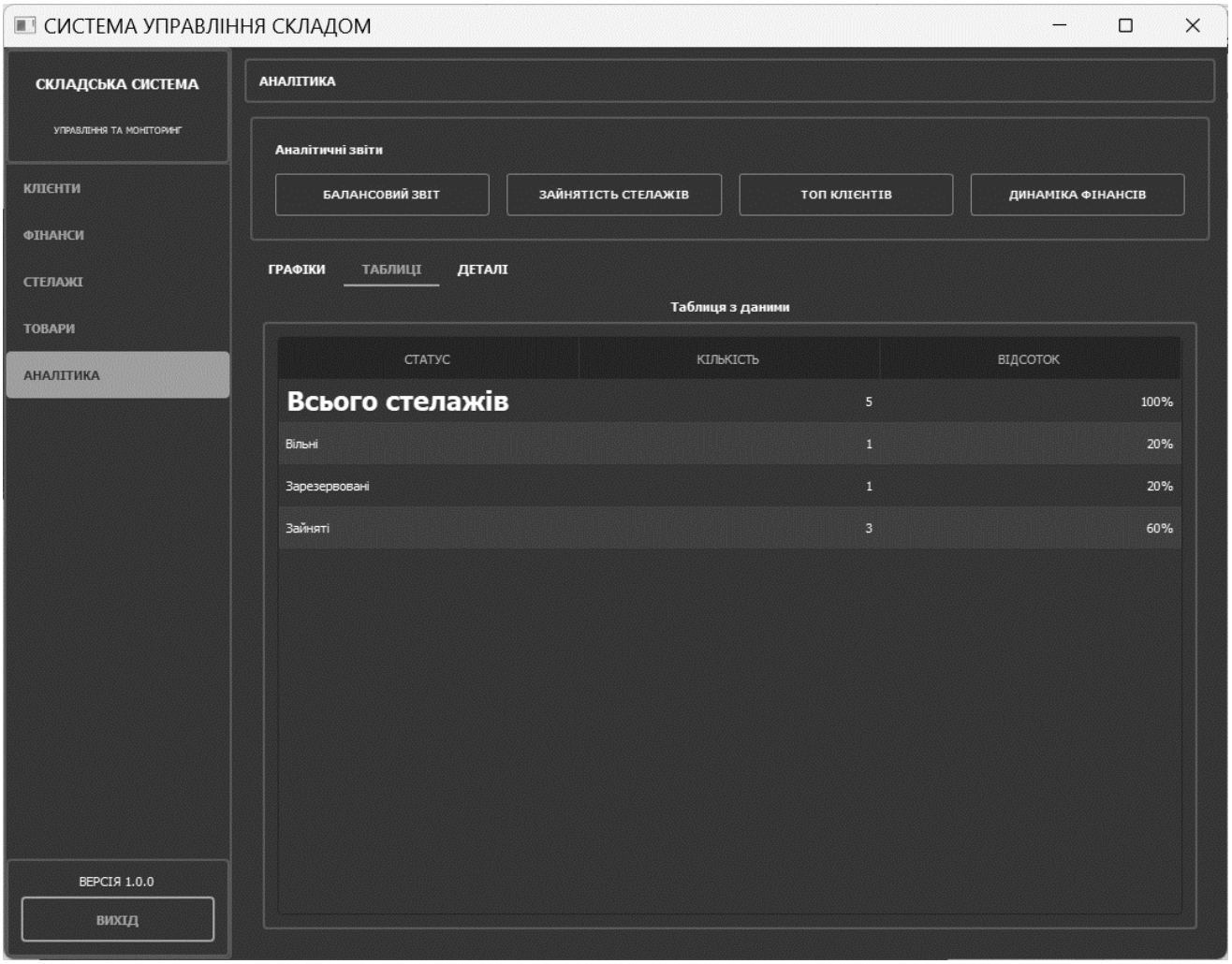

Рисунок 2.9 — Звіт про зайнятість стелажів у формі таблиці

На рисунку 2.9 показано звіт про зайнятість стелажів у формі таблиці. Є поля статусу, кількості та відсоткового відношення вільних, зарезервованих та зайнятий стелажів.

На рисунку 2.10 показано звіт про фінанси у формі деталізації. Там розміщена таблиця з фін записами та фін статистика як підбиття балансу. Дана таблиця буде дуже корисна бухгалтерам.

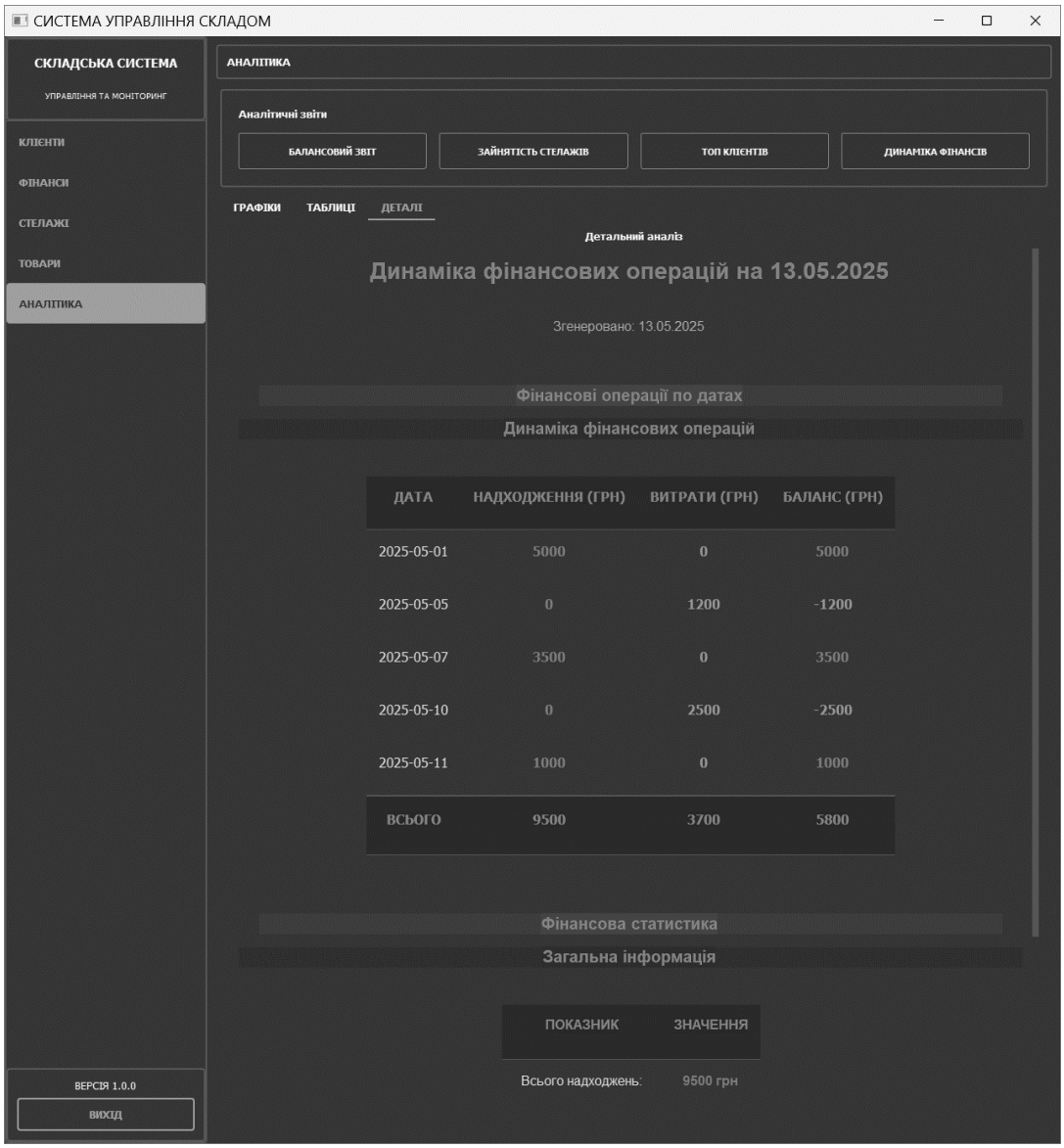

Рисунок 2.10 — Звіт про динаміку фінансів у формі деталізації

2.2 Використані технології.

При реалізації програми було використано мову програмування Python та середовище розробки Visual Studio Code. Також використано PyQt5 - для розробки UI та систему управління БД SQLite для зберігання інформації про клієнтів, товари, стелажі та фінансові операції. Також використані бібліотека Qt Material та компонент Qt Charts. А тепер детальніше.

Python 3.11 є високорівневою об'єктно-орієнтованою мовою програмування, що відзначається збалансованим поєднанням читабельності, простоти синтаксису та широких можливостей для розширення. Його гнучка структура дозволяє

створювати як прості утиліти, так і комплексні програмні рішення, підтримуючи як процедурний, так і об'єктно-орієнтований стиль програмування. Така універсальність робить мову привабливою для розробників різного рівня, від початківців до професіоналів у сфері наукових досліджень, бізнес-аналізу та інженерії.

Однією з ключових переваг Python є висока швидкість розробки програмного забезпечення. Завдяки багатій стандартній бібліотеці та активній спільноті, яка підтримує тисячі відкритих пакетів, розробник може сконцентруватися на логіці проєкту замість рутинної реалізації базового функціоналу. Це особливо важливо в умовах обмежених строків або коли необхідно оперативно створити прототип продукту чи провести експериментальну перевірку ідеї.

Величезна екосистема Python дозволяє легко інтегрувати різні компоненти — від вебфреймворків до бібліотек машинного навчання чи роботи з базами даних. Така відкритість до інтеграцій розширює сферу застосування мови: її однаково ефективно використовують у настільному програмуванні, автоматизації, мобільних додатках, хмарних обчисленнях і багатьох інших галузях. Python 3.11 не лише зберігає сумісність із попередніми версіями, але й демонструє зростаючу продуктивність завдяки вдосконаленому інтерпретатору.

Для створення сучасних графічних інтерфейсів з використанням Python часто застосовується PyQt5 — набір бібліотек, що забезпечує повний доступ до функціональності Qt Framework. Цей фреймворк дає змогу розробляти кросплатформені інтерфейси користувача з високим рівнем інтерактивності та широкими можливостями налаштування. Завдяки PyQt5 можна створювати як прості форми, так і складні віконні додатки з меню, діалогами, віджетами та динамічними елементами керування.

PyQt5 відзначається своєю стабільністю, гнучкістю у проєктуванні компонентів та широким інструментарієм для роботи з сигналами, подіями та візуальними елементами. Цей фреймворк підтримує також модульність, що дозволяє структурувати великі програми за принципами MVC або MVP. Особливо цінним є те, що PyQt5 зберігає повну функціональність Qt, включно з можливістю

доступу до анімації, мережевих з'єднань, OpenGL, потоків і баз даних, що значно підвищує гнучкість проєкту.

SQLite у свою чергу виступає як зручна, вбудована реляційна база даних, що легко інтегрується з Python через стандартний модуль sqlite3 або відповідні ORM-бібліотеки. Її легковажність, відсутність потреби у сервері та простота у використанні роблять її ідеальним варіантом для зберігання структурованої інформації в настільних або локальних програмах.

Завдяки використанню SQL-запитів та широкому набору індексів, SQLite забезпечує достатню продуктивність для більшості невеликих та середніх застосунків. Її формат зберігання у вигляді одного файлу спрощує резервне копіювання, переносимість та забезпечення безпеки даних. Крім того, її сумісність із PyQt5 дозволяє напряму з’єднувати інтерфейс із базою даних, реалізуючи зручну інтеракцію користувача з даними.

Для надання графічному інтерфейсу сучасного вигляду, бібліотека Qt Material пропонує набір стилів, що реалізують концепцію Material Design від Google. Вона дозволяє змінити вигляд стандартних віджетів PyQt5 на більш естетично привабливі, відповідаючи сучасним очікуванням користувачів щодо дизайну. Це стосується як палітри кольорів, так і форм елементів, анімацій, ефектів натискання чи спливаючих повідомлень.

Qt Material не лише покращує зовнішній вигляд програми, але й сприяє підвищенню зручності використання інтерфейсу. Уніфікація елементів управління дозволяє створити послідовний, інтуїтивно зрозумілий користувацький досвід, що особливо важливо при розробці систем для щоденного використання або в комерційних програмних продуктах. Таким чином, поєднання PyQt5 та Qt Material дає змогу досягти балансу між функціональністю та візуальною привабливістю.

Для аналітичних завдань і побудови звітів бібліотека Qt Charts виступає ефективним рішенням для візуалізації даних. Вона дозволяє створювати лінійні графіки, гістограми, кругові діаграми, а також комбіновані варіанти з підтримкою масштабування, навігації та взаємодії з користувачем. Візуальні елементи можуть бути інтегровані безпосередньо в основне вікно застосунку, забезпечуючи швидкий

доступ до ключової інформації.

Qt Charts відрізняється високим рівнем інтеграції з іншими модулями Qt, що дозволяє динамічно оновлювати графіки залежно від змін у базі даних або взаємодії користувача. Наприклад, при оновленні фінансових даних у SQLite можна миттєво оновити аналітичні графіки без перезавантаження інтерфейсу. Це значно полегшує аналіз інформації та прийняття рішень на основі візуалізованих показників.

2.3 Проектування класів.

У даному проекті було реалізовано кілька класів, розглянемо їх детальніше.

WarehouseDatabase це клас для роботи з базою даних складу. В ньому описані функції створення таблиць БД, створення зразка даних для демо та сама функція створення запиту до БД.

Також було реалізовано ряд класів для роботи з графічним інтерфейсом серед них AnalyticsPage, ClientsPage, FinanciesPage, ProductsPage та AnalyticsPage. Ці класи описують поведінку графічного інтерфейсу на дії користувача.

Також розроблені класи CardComponents та UIComponents. Ці два класи відповідають за створення універсальної карточки в UI та універсального компонента відповідно.

Клас BaseDialog та його дочірні класи ShelfDialog, ClientDialog, FinanceDialog, SelectClientDialog, ProductDialog, IssueProductDialog та ClientProductsReportDialog відповідають за діалогові вікна які необхідні для редагування та додавання сутностей через інтерфейс користувача.

Клас FlowLayout реалізовано для адаптивного розміщення карток. Саме він забезпечує адаптивність інтерфейсу.

Клас WarehouseData це клас де міститься вся бізнес логіка проекту. У ньому логіка додавання та видалення сутностей, їх зміна та інші CRUD операції.

Клас WarehouseManagementSystem це основний клас UI. Він відповідає за основну форму програми та її логіку обробки дій користувача.

Структура класів проекту показана на рисунку 2.9.

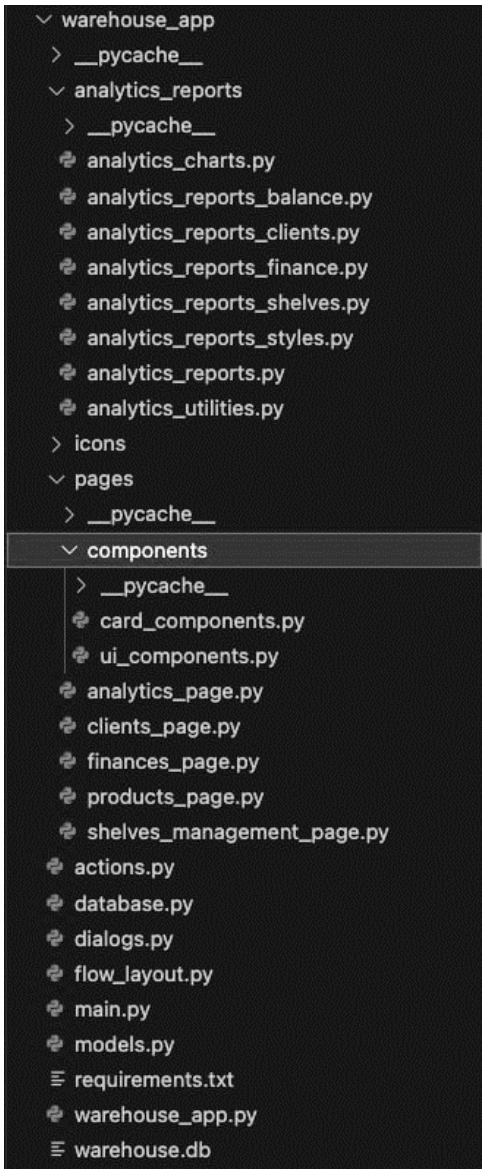

Рисунок 2.9 — Дерево проекту

2.4 Розробка структури даних

Одним із ключових етапів створення інформаційної системи є формування її структурної основи, що полягає в проєктуванні схеми зберігання та обробки даних. На цьому етапі визначаються основні сутності, які використовуються в межах системи, а також формуються зв’язки між ними, що дозволяє забезпечити логічну цілісність та узгодженість всієї інформаційної моделі.

Структура даних має бути достатньо гнучкою для подальшого розширення, а також відповідати реальним процесам, які відбуваються у середовищі складського обліку. Важливою умовою стало забезпечення прозорої взаємодії між компонентами системи, зокрема тих, які відповідають за клієнтів, товарні позиції, місця зберігання, а також фінансову аналітику. Ці компоненти повинні мати чітко окреслену функціональність, водночас не створюючи надмірної залежності між собою, що сприяє підтримуваності коду та підвищенню надійності системи в цілому.

Саме тому структура бази даних містить такі ключові слова як Clients – зберігає інформацію про клієнта а саме його контактні дані ПІБ та історію взаємодії.Products та Shelves описують структури товарів та стелажів їх назви ідентифікатори в товарів описує їх кількість ціну та відповідне місце на стелажах.Finance – записує фінансові операції, а саме дату коли вона була проведенна суми та тип транзакції

Побудова моделі даних супроводжувалася врахуванням типових ситуацій, що виникають при використанні програм у реальних умовах комерційного складу. Тому розроблена структура дозволяє зберігати всю необхідну інформацію у зручному форматі та забезпечує її цілісність навіть при інтенсивній взаємодії користувача з інтерфейсом системи.

2.5 Логіка побудови звітів

Під час функціонування будь-якої інформаційної системи, що оперує великим обсягом даних, зростає потреба у формуванні підсумкової інформації для подальшого аналізу. В нашому випадку важливим завданням стало створення такої логіки побудови звітів, яка б давала змогу швидко та наочно відображати найбільш релевантну інформацію про стан складського приміщення, його фінанси та активність клієнтів.

Звіти виконують роль інструменту аналітичного мислення – вони перетворюють сухі числові значення на зручні для сприйняття графіки, таблиці та текстові викладки. У своїй основі така логіка побудована на динамічній взаємодії

між поточним станом бази даних та візуальними елементами програми, які зчитують зміни майже в реальному часі. Це дозволяє уникати необхідності ручного оновлення або повторного запуску компонентів інтерфейсу, що в свою чергу значно підвищує зручність користування системою.

Особливу увагу було приділено гнучкості відображення звітів. Адже в різних ситуаціях користувач може потребувати як загального огляду, так і деталізованого представлення певної категорії інформації. Тому реалізовано механізми, що дозволяють формувати звіти у кількох форматах — від графічної візуалізації до текстової деталізації з підрахунками. Це створює умови для широкого спектра аналітичних підходів і допомагає керівникам швидше приймати обґрунтовані рішення, спираючись на актуальні дані.

2.6 Архітектура застосунка

Архітектура розробленої інформаційної системи складського обліку базується на концепції чіткої структурованості та модульності, що дозволяє спростити супровід, масштабування та подальший розвиток проєкту. Ключовою архітектурною моделлю було обрано патерн Model–View–Controller (MVC), який забезпечує логічне розділення частин програми відповідно до їхніх функціональних обов’язків.

У межах цієї архітектури модель (Model) відповідає за усю бізнес-логіку застосунку та взаємодію з базою даних. Саме тут реалізується збереження, обробка та передача інформації. Усі ключові дії з даними — зчитування, оновлення, додавання та видалення — виконуються за допомогою класу WarehouseData, який абстрагує взаємодію з реляційною базою даних SQLite. Це дозволяє забезпечити ефективну та прозору роботу з даними без дублювання логіки у різних частинах коду.

Представлення (View) у застосунку реалізовано за допомогою бібліотеки PyQt5. Візуальні компоненти винесені у відповідні окремі модулі, кожен із яких відповідає за окрему частину інтерфейсу. Інтерфейс побудований на основі вкладок, де кожна відображає певний аспект системи: клієнти, фінанси, товари,

стелажі, аналітика. Вікна розроблені з урахуванням адаптивності – компоненти автоматично змінюють розмір відповідно до ширини вікна, що забезпечується реалізацією кастомного компонента FlowLayout.

Уся взаємодія між інтерфейсом користувача та логікою програми контролюється через контролер (Controller), реалізований у вигляді класу WarehouseManagementSystem. Саме цей компонент обробляє події, ініційовані користувачем, координує відповідні зміни в даних та оновлює інтерфейс відповідно до нових значень. Таким чином, користувач взаємодіє виключно з візуальними елементами, у той час як зміни відбуваються на рівні моделі без прямого доступу до даних.

Окремо варто відзначити модулі діалогових вікон, які реалізовані у вигляді спадкових класів, що базуються на загальному батьківському класі BaseDialog. Вони забезпечують зручну та уніфіковану взаємодію з користувачем під час додавання або редагування записів. Це значно підвищує зручність роботи з системою та покращує користувацький досвід.

Завдяки такій архітектурі програмний продукт характеризується високою гнучкістю, структурованістю та зручністю у підтримці. Вона також забезпечує кросплатформену сумісність і дозволяє швидко адаптувати застосунок до змін у вимогах без необхідності повної переробки його логіки.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ З РЕКОМЕНДАЦІЯМИ ПО ВИКОРИСТАННЮ ІС

3.1 Реалізація програми.

Розглянемо основні методи програми. Для забезпечення структурованості та підтримуваності проєкту була реалізована модульна архітектура, яка чітко розділяє логіку програми, графічний інтерфейс користувача та взаємодію з базою даних Мабуть найперше варто звернутися до методів класу WarehouseData. На лістингу 3.1. показано ініціалізацію БД

Лістинг 3.1. Ініціалізація БД

```
def __init__(self):
    # Ініціалізація бази даних
    self.db = WarehouseDatabase()
    # Ініціалізація зразкових даних, якщо база порожня
    self.db.init_sample_data()
```

На лістингу 3.2. показано метод додавання стелажів.

Лістинг 3.2. Додавання стелажів

```
def add_shelf(self, name, capacity):
    """Додавання нового стелажу"""
    self.db.execute_query(
        "INSERT INTO shelves (name, capacity, status, tenant) VALUES (?, ?, 'Вільний', '')",
        (name, capacity),
        commit=True
    )
    return self.db.execute_query(
        "SELECT id FROM shelves WHERE name = ?",
        (name,),
        fetchone=True
    )[0]
```

Як бачимо бізнес логіка тісно пов’язана зі сховищем даних, а саме з базою даних sqlite.

Також варто розглянути функції отримання стелажу за ід та назвою, див

лістинг 3.3.

Лістинг 3.3. Отримання стелажу за ід та назвою.

```
def get_shelf_by_id(self, shelf_id):
    """Отримання стелажу за ID"""
    return self.db.execute_query(
        "SELECT id, name, capacity, status, tenant FROM shelves WHERE id =
?",
        (shelf_id,),
        fetchone=True
    )
  def get_shelf_by_name(self, name):
    """Отримання стелажу за назвою"""
    return self.db.execute_query(
        "SELECT id, name, capacity, status, tenant FROM shelves WHERE name
= ?",
        (name,),
        fetchone=True
    )
```

На лістингу 3.4. показано отримання всіх стелажів з БД.

Лістинг 3.4. Отримання всіх стелажів

```
  def shelves(self):
    """Отримання всіх стелажів"""
    return self.db.execute_query(
        "SELECT id, name, capacity, status, tenant FROM shelves",
        fetchall=True
    )
```

Також варто розглянути функцію generate_shelves_report_html з класу ShelvesReport яка показана на лістингу 3.5.

Лістинг 3.5. generate_shelves_report_html

```
def generate_shelves_report_html(self, shelf_stats):
    """Генерація HTML звіту для аналізу зайнятості стелажів"""
    today = QDate.currentDate().toString("dd.MM.yyyy")
    html = self.get_report_header(f"Аналіз зайнятості стелажів на {today}")
    html += self.get_section_divider("Статистика стелажів")
…
    html += self.get_card_html(
        "Загальна інформація",
        f"""
```

```
        <table class="data-table">
          <thead>
            <tr>
              <th>Показник</th>
              <th>Кількість</th>
              <th>Відсоток</th>
            </tr>
          </thead>
          <tbody>
            <tr class="highlighted-row">
              <td>Загальна кількість стелажів</td>
              <td>{total_shelves}</td>
              <td>100%</td>
            </tr>

…
          </tbody>
        </table>
        """,
    )
```

Як бачимо дана функція оперує html розміткою яку потім перетворює в елементи на формі. Цей підхід суттєво спрощує реалізацію інтерфейсу користувача і разом з тим сам інтерфейс є зручним та комфортним

Також розглянемо функцію створення у класі ClientDialog який відповідає за форму редагування та створення нового клієнта. Вона показана на лістингу 3.6.

Лістинг 3.6. ClientDialog

```
    def __init__(self, parent=None, name="", contact_person="", phone="", email=""):
        super().__init__(parent, "Клієнт")
        layout = QFormLayout(self)
        layout.setSpacing(15)
        layout.setContentsMargins(20, 20, 20, 20)
        # Заголовок
        title_label = QLabel("Інформація про клієнта")
        title_label.setObjectName("dialogTitle")
        title_label.setFont(QFont("Roboto", 16))
        layout.addRow(title_label)

        # Email
        self.email_edit = QLineEdit(email)
```

```
        self.email_edit.setPlaceholderText("example@example.com")
        layout.addRow("Email:", self.email_edit)
    …
        # Кнопки
        buttons = QDialogButtonBox(QDialogButtonBox.Ok | QDialogButtonBox.Cancel)
        buttons.button(QDialogButtonBox.Ok).setText("Зберегти")
        buttons.button(QDialogButtonBox.Cancel).setText("Скасувати")
        buttons.accepted.connect(self.accept)
        buttons.rejected.connect(self.reject)
        layout.addRow(buttons)
```

Як бачимо з лістингу 3.6 підхід трішки змінився у проектуванні інтерфейсу. Попри те що тут все ще використовується html, використовується обгортка у виді класу QFormLayout який допомагає розміщувати елементи на формі в більш зручному форматі. Це суттєво спрощує проектування інтерфейсу.

Розглянемо клас UIComponents. Цей клас містить опис універсальних компонентів інтерфейсу таких як кнопка, картка, шрифт та поле з інфо див. лістинг Г.9 Блок обробки даних (WarehouseData): відповідає за всі CRUD-операції, зокрема додавання, редагування та видалення записів клієнтів, товарів, фінансових операцій та даних про стелажі. Реалізовано кешування даних у пам’яті для зменшення кількості звернень до бази.

Інтерфейс користувача (UI-класи): побудований із використанням PyQt5. Кожна вкладка програми реалізована у вигляді окремого класу (наприклад, ClientsPage, ProductsPage, AnalyticsPage), що дозволяє легко вносити зміни або додавати нові модулі.

Компонент діалогів: реалізовано як базовий клас BaseDialog із похідними класами (ClientDialog, FinanceDialog тощо), що значно зменшує повторення коду та полегшує розширення функціоналу.

Адаптивне розміщення елементів: за допомогою класу FlowLayout забезпечується автоматична підгонка UI-елементів до розміру вікна, що покращує користувацький досвід.

Графічна візуалізація даних: модуль AnalyticsPage використовує Qt Charts для побудови кругових діаграм, таблиць і деталізованих звітів. Передбачено три

режими подання: діаграма, таблиця та деталізація з можливістю перемикання в реальному часі.

Ці методи є конструкторами для створення значної кількості варіацій шаблонних кнопок таких як кнопка для додавання, кнопка для видалення, кнопка для редагування і так далі. Аналогічно з написами, типами шрифтів та фреймами.

Так для шрифтів передбачено жирний та курсив.

А для інфополів передбачені налаштування адаптивності такі як розтягування по вертикалі та горизонталі, зміна розміру шрифтів, розтягування заголовку, перенесення слів і так далі.

Для карток по аналогії передбачено типу картки такі як: картка клієнта, картка полиці, картка фін. запису, картка товару. А також встановлення політики для адаптивності через pyqt5.

У класі CardComponents є методи для створення типів карток такі як

— create_client_card;

— create_shelf_card;

— create_product_card;

— create_finance_card.

Розглянемо детальніше створення картки клієнта.

Її можна розбити на три блоки створення представлення, прив’язка даних та прив’язка кнопок. На лістингах Г.9 – Г.11 показано ці три умовні блоки

Як бачимо з лістингів вище це доволі велика функція, проте вона має вирішальне значення оскільки пов’язує UI та бізнес логіку. Крім цієї є ще три аналогічні функції для товарів, фінансових записів та полиць. Разом цей блок функцій розташований у файлі card_components.py

Таким чином на лістингах що вище описано основні та цікаві моменти що стосуються програмної реалізації проекту. І попри те що сторонньому спостерігачу може здатися що все якось не те і не так дана система працює швидко та без програмних помилок, не залежить від операційної системи і в цілому попри свою відносну компактність виконує поставлену задачу.

3.2 Тестування програми

Ручне тестування — це процес перевірки роботи програмного забезпечення без використання автоматизованих інструментів. Тестувальник взаємодіє з програмою так, як це робив би звичайний користувач: відкриває сторінки, натискає кнопки, вводить дані та відстежує поведінку системи. Основна мета — виявити помилки, які можуть вплинути на функціональність, зручність або безпеку додатку. Цей тип тестування є особливо цінним на ранніх етапах розробки або коли потрібно оцінити загальний користувацький досвід.

Незважаючи на те, що ручне тестування потребує більше часу порівняно з автоматизованим, воно дозволяє гнучко адаптуватися до змін у продукті та краще помічати нетипові ситуації, які важко передбачити заздалегідь. Людський фактор у цьому процесі часто стає перевагою, оскільки тестувальник може помічати неочевидні деталі, які залишаються поза увагою автоматичних скриптів. Саме тому ручне тестування і досі залишається невід’ємною частиною якісного забезпечення програмних продуктів.

Крім того, ручне тестування часто використовується для проведення тестування юзабіліті, яке дозволяє оцінити інтуїтивність інтерфейсу, логіку навігації та загальне задоволення користувача від взаємодії з продуктом. Завдяки живому сприйняттю тестувальника можна швидко виявити моменти, які викликають плутанину або дискомфорт у користувачів, що допомагає зробити програму не лише технічно правильною, але й приємною у використанні.

Ручне тестування дасть змогу переконатися у стабільності роботи інтерфейсу та коректності виконання основних сценаріїв. Було перевірено взаємодію між різними модулями системи, зокрема додавання, редагування та видалення записів. Особливу увагу приділено перевірці правильності відображення даних після виконання операцій. Усі дії виконувались послідовно відповідно до заздалегідь підготовлених тест-кейсів.

В рамках тестування програми було проведено ряд тестів в ручному режимі, в таблиці 3.1 показані деякі з них.

Таблиця 3.1. Тесткейси ручного тестування

№	Дія	Очікуваний результат	Фактичний результат	Пройдено
1	Запустити програму Натиснути кнопку “Додати стелаж” Ввести назву стелажу та місткість Натиснути кнопку “Зберегти”	Повідомлення про успішне додавання стелажу, новий стелаж відображається на вкладці стелажів	Повідомлення про успішне додавання стелажу, новий стелаж відображається на вкладці стелажів	пройдено
2	Натиснути кнопку ”Зарезервувати” Обрати клієнта Натиснути кнопку “Вибрати”	Повідомлення про успішне резервування стелажу, стелаж відображається на вкладці стелажів як зарезервований	Повідомлення про успішне резервування стелажу, стелаж відображається на вкладці стелажів як зарезервований	
3	Натиснути кнопку ”Орендувати” Обрати клієнта Натиснути кнопку “Вибрати” Вказати дату, суму та опис Натиснути кнопку “Зберегти”	Повідомлення про успішне орендування стелажу, стелаж відображається на вкладці стелажів як орендований Створився фін запис на вказану суму	Повідомлення про успішне орендування стелажу, стелаж відображається на вкладці стелажів як орендований Створився фін запис на вказану суму	
4	Натиснути кнопку ”Звільнити”	Запит підтвердження дії	Запит підтвердження дії	
5	Запит підтвердження дії Підтвердити	Повідомлення про успішне виконання	Повідомлення про успішне виконання	

Схожих тесткейсів було виконано більше 80 штук. Більшість пройшли успішно. Котрі не пройшли успішно з першого разу були внесені в список на повторне проходження і пройшли в наступній ітерації тестування. Після кожної ітерації тестування проводилися зміни в коді задля виправлення помилок. Цей підхід забезпечив створення програмного продукту без помилок або з несуттєвими помилками які не мають значного впливу на користувацький досвід.

Також було перевіренно програму, що до її не доцільного чи не типового використання та подальшого реагування програми на дії щодо неї.Завдяки цьому було виявленно ситуації за яких система може отримувати навантаженн та як вона з нми справиться.Таке тестування дало змогу отримати уявлення що до стійкості системи на людські помилки.

Таблиця 3.3.— Тест-кейси невірного тестування

№	Дія	Очікуваний результат	Фактичний результат	Пройдено
1	Додавання клієнта з порожнім іменем	Виводиться повідомлення про помилку, запис не додається	Виводиться повідомлення про помилку, запис не додається	пройдено
2	Відсутній інтернет-зв’язок при запуску	Додаток запускається, проте оновлення та база данних не підтягуються	Додаток запустився проте оновлення данних не відбулось	Пройдено
3	Додавання нульової кількості товару на стелаж	Додаток не дає змоги вибрати нульову кількість під час заповнення	Додаток не дає змоги вибрати нульову кількість під час заповнення та поверне в поле попереднє цифрове значення	Пройдено

Приклади пройденних текст кейсів наведенно у вигляді рисунків.Перший випадок додавання стелажа зображенно на рисунку 3.1

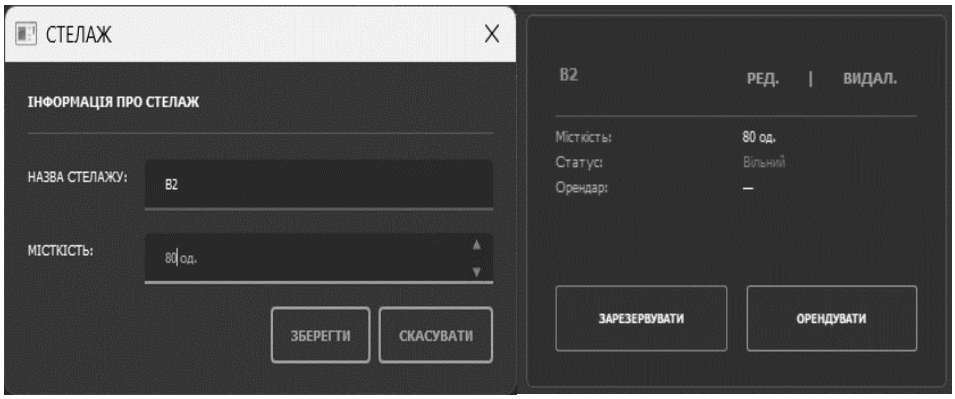

Рисунок 3.1 — Позитивне тестування

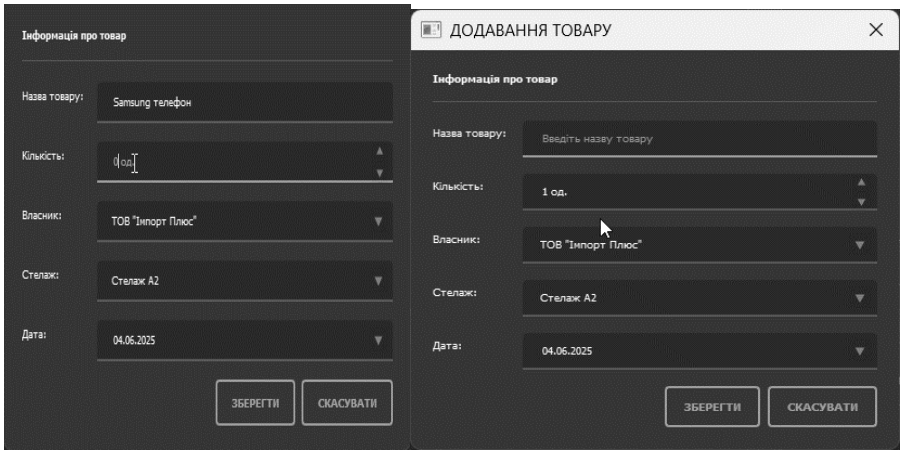

Рисунок 3.2 — Невірне виконання вимог

3.3 Компіляція та запуск програми

Для компіляції програми потрібно встановити python. Для цього необхідно скачати дистрибутив з офіційного сайту пайтон https://www.python.org/downloads/release/python-3110/ та встановити його на ПК.

Для компіляції програми потрібно перейти в папку з програмним кодом та викликати командний рядок від імені адміністратора. Опісля потрібно встановити pip якщо він не був встановлений, для цього потрібно виконати команду pip install pyinstaller

Після цього необхідно виконати команду pyinstaller --onefile --clean --noconsole .\main.py результат на рисунку 3.2.

Рисунок 3.3 — Виконання команди pyinstaller --onefile --clean --noconsole .\main.py

Розглянемо детальніше ці команди.

pyinstaller — це сама утиліта, яка упаковує Python-програму у виконуваний файл.

--onefile — створює один єдиний .exe файл замість папки з кількома файлами. Усі залежності (модулі, бібліотеки тощо) упаковуються всередину одного файлу.

--clean — перед створенням виконуваного файлу видаляє всі тимчасові файли, які могли залишитися після попередніх збірок. Це допомагає уникнути конфліктів і забезпечує "чисту" компіляцію.

--noconsole — вимикає запуск консолі під час виконання програми.

.\main.py — шлях до файлу, який потрібно зібрати. У цьому випадку, це файл main.py, розташований у поточній директорії.

Як бачимо в результаті виконання команди в консолі з’являється лог виконання, з очікуваним Build complete! і в результаті у папці dist з’являється файл main.exe та файл з БД warehouse.db.

3.4 Використання ІС

Для використання програми потрібно запустити файл main.exe після цього з’явиться повідомлення про завантаження та доволі швидко з’явиться головна форма програми див. рисунок 3.4.

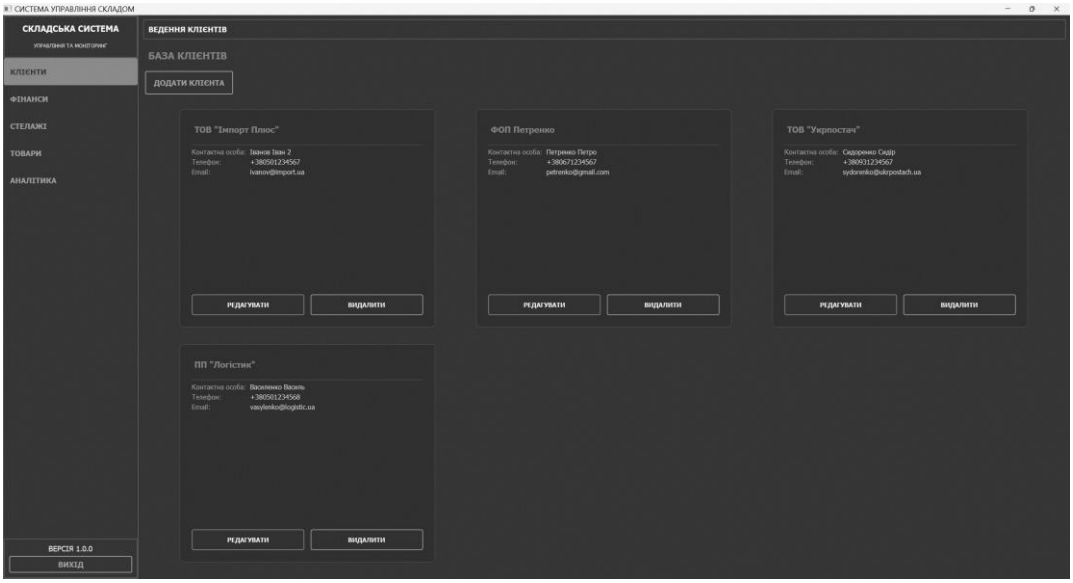

Рисунок 3.4 — Головне вікно програми

З цієї вкладки можна керувати клієнтами, інтерфейс максимально інтуітивний. Є можливість додавання, видалення та редагування даних про клієнтів.

У вкладці фінансів все аналогічно за вийнятком того що є можливість не лише добавити дохід, але й добавити витрату див. рисунок 3.5.

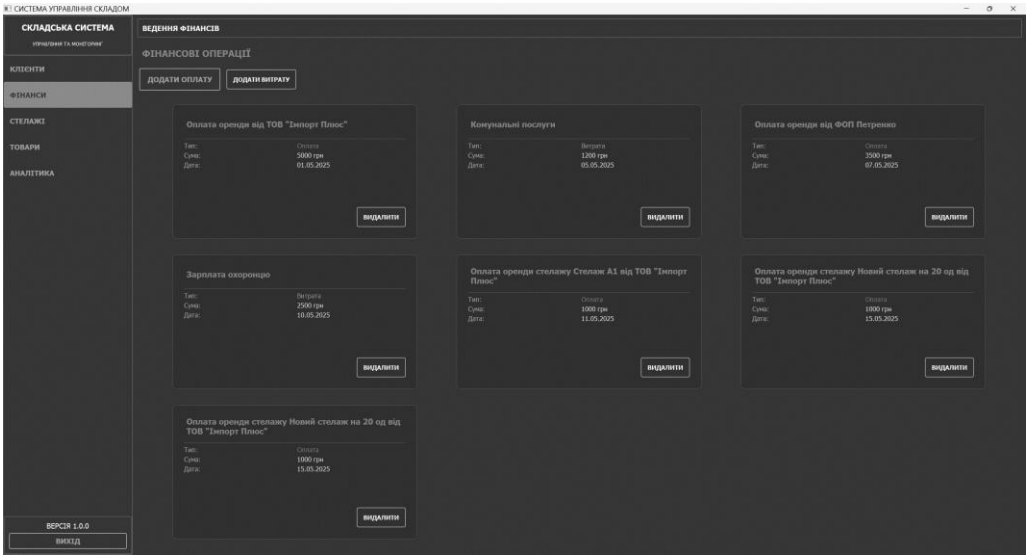

Рисунок 3.5— Вкладка фінансів

Також фінансовий запис може створитися при здачі стелажів в оренду.

На рисунок 3.6 показана вкладка стелажів. На ній чудово видно що є кнопки

додавання, редагування та видаленя стелажів аналогічні як і у попередніх вкладках, проте також є кнопки звільнити, орендувати та зарезервувати. Ці кнопки змінюють статус стелажа.

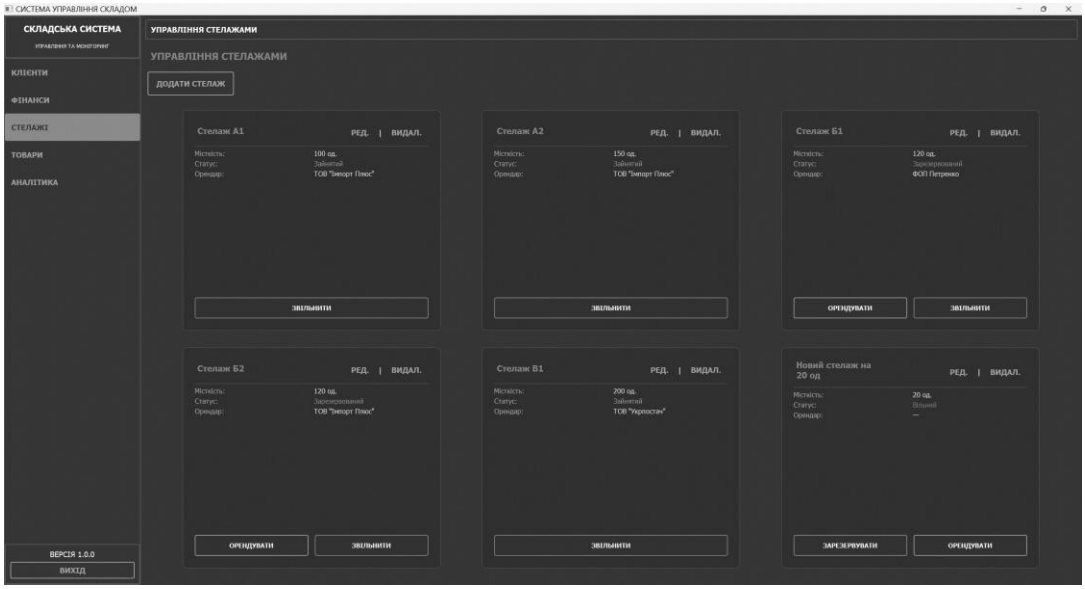

Рисунок 3.6— Вкладка стелажів

При резерві з’являється форма вибору клієнта

А опісля виринає сповіщення де вказано назву стелажу та його місткість та інформацію про клієнта який його зарезервував. Це дуже зручно якщо в процесі резервування менеджеру довелося відволіктися на іншого клієнта.

При оренді стелажа з’являється форма, проте опісля вибору клієнта з’являється ще одна форма Якщо натиснути на кнопку зберегти, з’явиться сповіщення схоже на те що проте це не все. Оренда стелажів це основна продукція комерційного складу, тобто потрібно ще прийняти оплату. А отже з’явиться фінансовий запис про дохід від оренди стелажу. Також ці дані з’являться в балансовому звіті

Також у вкладці аналітики на вкладці з графіками є анімація появи графіків

Підсумовуючи все вищесказане варто відмітити що програма є зручною та зрозумілою з точки зору користувача. Новим користувачам не потрібне довге навчання та адаптація, достатньо кількох хвилин з більш досвідченим співробітником і можна працювати самостійно.

Також програма є кросплатформною, достатньо лише встановленого пайтона для виконання чи компіляції у виконавчий файл як показано вище на рпикладі ОС Windows.

3.5 Рекомендації щодо використання

Для ефективного впровадження та повноцінного використання розробленої інформаційної системи керування комерційним складським приміщенням важливо дотримуватись ряду практичних рекомендацій. Передусім, ключову роль відіграє належна підготовка персоналу. Співробітники, які працюватимуть із системою, повинні пройти попереднє навчання або інструктаж, що дозволить уникнути помилок при введенні даних та забезпечить коректну взаємодію з усіма функціональними модулями системи.

Не менш важливим є організація процесу резервного копіювання даних. Враховуючи, що система зберігає важливу інформацію про запаси, переміщення товарів, орендарів тощо, втрата цих даних може мати суттєві наслідки для підприємства. Тому доцільно реалізувати автоматизовану систему створення резервних копій з можливістю швидкого відновлення.

Також варто передбачити регулярний технічний супровід та моніторинг продуктивності системи. Це дозволить своєчасно виявляти та усувати можливі збої, а також адаптувати систему до зростаючих навантажень. У разі виявлення сповільнень або нестабільної роботи доцільно провести оптимізацію коду або бази даних.

З метою забезпечення інформаційної безпеки слід постійно оновлювати програмні компоненти системи — як серверну частину, так і клієнтський інтерфейс. Актуальні версії зазвичай містять покращення безпеки та стабільності, що зменшує ризик несанкціонованого доступу чи збоїв.

Крім того, у випадку, якщо підприємство вже використовує інші програмні засоби — наприклад, для бухгалтерського обліку чи логістики — доцільно розглянути можливість інтеграції нової системи з уже існуючими. Це дозволить

автоматизувати процеси документообігу, обліку товарів і полегшить обмін даними між підрозділами.

Загалом, систематичний аналіз ефективності використання, відгуки користувачів та своєчасна адаптація функціоналу до потреб підприємства сприятимуть довготривалому й результативному використанню інформаційної системи.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було детально проаналізовано проблематику складського обліку, зокрема труднощі, пов’язані з неефективною організацією зберігання, обробки та контролю товарів. Сучасні підходи до автоматизації дозволяють значно оптимізувати ці процеси, тому актуальність теми дослідження не викликає сумнівів.

На початковому етапі досліджено завдання та мету проведено аналіз предметної області та визначенно значимість проблеми що стосується комерційних приміщень, що дозволило надалі вибрати найдоцільнішу для реалізації поставленої задачі мову програмування.

У першому розділі ми розглянули найбільш популярні мови та середовища програмування та виявили їх переваги та недоліки.

Пояснення понять об’єктно-орієнтованого програмування стало важливою частиною теоретичної підготовки, оскільки саме парадигма ООП лягла в основу побудови логіки програми. Це забезпечило структурованість та масштабованість коду, що є критично важливим у подальшому розвитку системи.

Окрему увагу приділено огляду середовищ розробки. Було розглянуто такі популярні IDE як Visual Studio Code, PyCharm та Jupyter Notebook. Кожне з них має свої особливості, однак у процесі реалізації обрано Visual Studio Code.

Значна частина роботи була присвячена аналізу існуючих систем складського обліку. Це дало змогу не лише виявити їхні недоліки, а й врахувати ці аспекти при розробці власного рішення. Отримана інформація лягла в основу постановки вимог до створюваної інформаційної системи.

У другому розділі було розроблено структуру та алгоритми, що забезпечують роботу складської системи. Зокрема, реалізовано модулі для обліку товарів, взаємодії з користувачем та обробки даних. Було враховано зручність інтерфейсу, швидкодію та простоту навігації.Велику увагу було приділенно адаптивному інтерфейсу та використанню технологій

Розробка інтерфейсу користувача проходила із урахуванням принципів UX-дизайну, що забезпечує зрозумілий і логічний процес роботи з системою. Це дозволяє зменшити час навчання персоналу та мінімізувати помилки у введенні даних.

У реалізації використовувались бібліотека PyQt5, що гарантують гнучкість, масштабованість і безпеку розробленої програми. Компонентна структура дозволяє легко оновлювати або додавати нові функціональні можливості в майбутньому.

Особлива увага приділялася проектуванню класів, що відповідають за основні функціональні елементи системи. В результаті вдалося досягти логічної ієрархії та мінімізації дублювання коду.

У розділі три було проведенно реалізацію програми а саме здійсненно розробку, кодування, налагоджування та тестування системи, що дало змогу виявити низку недоліків, які були оперативно усуненні та система була доведенна до стабільної версії.

Тестування програми продемонструвало її працездатність у передбачених сценаріях, а також показало високий рівень точності обліку даних. Зручність користування підтверджено в результаті імітації реального користувацького досвіду та представлено у вигляді лістингів, діаграм та зразків виводу програми . Що були представленні у додатках.

У підсумку, створена інформаційна система для складського обліку є ефективним інструментом автоматизації та обробки ключових складських приміщень.Завдяки цьому загальна продуктивність продукту виросла в 7-10 разів в порівнянні з аналогічними моделями, що підтверджує її доцільність у використанні в реальних умовах

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Страуструп Б. Мова програмування C++. Основи / Б'ярне Страуструп ; пер. з англ. – Київ : Діалектика, 2014. – 640 с.

2. Липпманн С.Е., Лажой Ж., Мусман Б. C++. Основи та практика програмування / Стенлі Б. Липпманн, Жозе Лажой, Барбара Мусман ; пер. з англ. – Київ : Видавництво «Професіонал», 2013. – 944 с.

3. Шилдт Г. C++: повний курс / Герберт Шилдт ; пер. з англ. – Київ : Видавнича група BHV, 2010. – 960 с.

4. Гроссмейєр Д. C++ для початківців / Дірк Гроссмейєр ; пер. з нім. – Харків : Фактор, 2006. – 384 с.

5. Троелсен Е., Іокі А. C# 9.0 та платформа .NET 5: довідник програміста / Ендрю Троелсен, Філіп Джепікс ; пер. з англ. – Харків : Видавництво "Фабула", 2021. – 1040 с.

6. Шилдт Г. C# 8.0. Повний посібник / Герберт Шилдт ; пер. з англ. – Київ : Діалектика, 2020. – 832 с.

7. Нагель К. Visual C# 2019 і платформа .NET. Розробка додатків / Карл Нагель ; пер. з англ. – Київ : Діалектика, 2019. – 1056 с.

8. Ліберман Б. Microsoft Visual C# .NET. Біблія користувача / Бредлі Ліберман, Річард Йохансен ; пер. з англ. – Київ : BHV, 2004. – 912 с.

9. Шилдт Г. Java. Повний курс / Герберт Шилдт ; пер. з англ. – Київ : Діалектика, 2021. – 1248 с.

10. Хорстманн К.С. Core Java. Том 1. Основи / Кей С. Хорстманн ; пер. з англ. – Київ : Видавництво "Фабула", 2020. – 928 с.

11. Еккель Б. Філософія Java / Брюс Еккель ; пер. з англ. – Санкт-Петербург : Пітер, 2018. – 1168 с.

12. Блінн Дж. Java. Біблія користувача / Джон Блінн ; пер. з англ. – Київ : BHV, 2010. – 896 с.

13. Свейгарт А. Автоматизація рутинних завдань з Python / Ал Свейгарт ;

пер. з англ. – Київ : Видавництво "Фабула", 2021. – 592 с.

14. Лутц М. Вивчаємо Python / Марк Лутц ; пер. з англ. – Київ : Діалектика, 2020. – 1600 с.

15. Рамальо Л. Python. Керівництво для професіоналів / Лучіано Рамальо ; пер. з англ. – Київ : Видавнича група "Лінк", 2019. – 770 с.

16. Матлофф Н. Вивчення Python: практичний підхід / Норм Матлофф ; пер. з англ. – Харків : Видавництво "Фактор", 2018. – 496 с.

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

д.т.н., проф. Азаров О.Д.

"____"_____2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи

" Інформаційна система керування комерційним складським приміщенням"

08-54.БКР.030.00.000.ТЗ

Науковий керівник: доцент к.т.н.

_____ Дудник О. В.

Студент групи 2КІ-21б

_____ Рябоконь К.М.

ВНТУ 2025

1 Підстава для виконання бакалаврської кваліфікаційної роботи (БКР)

1.1 Важливим є актуальність дослідження у напрямку кваліфікаційної роботи, яка обумовлена тим, що в даний час в умовах стрімкого розвитку логістики та електронної комерції питання ефективного управління складськими приміщеннями набуває особливої актуальності. При цьому коло завдань, що потребують вирішення, безперервно розширюється. Поряд з такими «базовими» завданнями, як додавання клієнта, все більший інтерес викликають і більш складні – створення звітів про залишки;

1.2 Наказ про затвердження теми БКР.

2 Мета БКР і призначення розробки

2.1 Мета роботи — є розробка або аналіз інформаційної системи, здатної ефективно управляти процесами на комерційному складі;

2.2 Призначення розробки — ІС складського обліку.

3 Вихідні дані для виконання БКР

3.1 Проведення аналізу існуючих систем складського обліку;

3.2 Розробка структури ІС;

3.3 На основі структурних та функціональних схем здійснено розрахунок та моделювання елементів ІС.

4 Вимоги до виконання БКР

Головна вимога — реалізувати інформаційну систему управління комерційним складським приміщенням.

5 Етапи БКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 — Етапи БКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Огляд і аналіз існуючих методів та рішень проектування органайзерів	21.03.25	30.03.25	Аналітичний огляд літературних джерел, задачі досліджень, розділ 1 ПЗ
2	Дослідження методів розробки віконних додатків	31.03.25	13.04.25	Розділ 2
3	Проектування ІС	14.04.25	27.04.25	Розділ 3
4	Оформлення пояснювальної записки, графічного матеріалу і презентації	28.04.25	11.05.25	Пояснювальна записка, графічний матеріал і презентація
5	Підготовка супроводжуючих документів, їх підписування, проходження нормоконтролю та тесту на плагіат	28.04.25	11.05.25	Оформлені документи

6 Матеріали, що подаються до захисту БКР

До захисту подаються: пояснювальна записка БКР, графічні і ілюстративні матеріали, протокол попереднього захисту БКР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до БКР українською та іноземною мовами, довідка про відповідність оформлення БКР діючим вимогам.

7 Порядок контролю виконання та захисту БКР

Виконання етапів графічної та розрахункової документації БКР контролюється науковим керівником згідно зі встановленими термінами. Захист БКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання БКР

8.1 При оформлюванні БКР використовуються:

— ДСТУ 3008 : 2015 «Звіти в сфері науки і техніки. Структура та правила

оформлювання»;

— ДСТУ 8302 : 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104-2006 «Единa система конструкторської документації. Основні написи»;

— Методичні вказівки. Кафедра обчислювальної техніки 2022;

— документами на які посилаються у вище вказаних.

8.2 Порядок виконання БКР викладено в «Положення про кваліфікаційні роботи на другому (бакалаврську) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21».

ДОДАТОК Б

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПРОГРАМИ

```
from PyQt5.QtWidgets import (
    QMainWindow,
    QWidget,
    QVBoxLayout,
    QHBoxLayout,
    QPushButton,
    QLabel,
    QStackedWidget,
    QFrame,
    QSplitter,
    QSizePolicy,
    QApplication,
)
from PyQt5.QtCore import Qt, QSize, QEvent
from PyQt5.QtGui import QFont, QResizeEvent

from models import WarehouseData
from pages.clients_page import ClientsPage
from pages.finances_page import FinancesPage
from pages.products_page import ProductsPage
from pages.shelves_management_page import ShelvesManagementPage
from pages.analytics_page import AnalyticsPage
from actions import WarehouseActions


class WarehouseManagementSystem(QMainWindow):
    """Головний клас програми для управління складом"""
 def __init__(self):

def setup_ui(self):

def setup_menu(self):

def create_pages(self):

 def switch_page(self):

def populate_tables(self):

def update_scale_factor(self, width, height):

def update_font_sizes(self):

def apply_global_styles(self):

 def resizeEvent(self, event):
 def eventFilter(self, obj, event):
```

```
from PyQt5.QtWidgets import QMessageBox
from dialogs import (
    ShelfDialog,
    ClientDialog,
    FinanceDialog,
    SelectClientDialog,
    ProductDialog,
    IssueProductDialog,
    ClientProductsReportDialog,
)
class WarehouseActions:
    """Клас для обробки дій корисуноктувача"""

    def __init__(self, main_window, data):
        self.main_window = main_window
        self.data = data

    def refresh_all_tables(self):
        """Оновлення всіх таблиць після змін в даних"""
        self.main_window.populate_tables()
    # Методи для роботи зі стелажами
    def add_shelf(self):
        """Додавання нового стелажу"""
        dialog = ShelfDialog(self.main_window)
        if dialog.exec_():
            name, capacity = dialog.get_data()
            try:
                self.data.add_shelf(name, capacity)
                self.refresh_all_tables()
                QMessageBox.information(
                    self.main_window, "Успіх", f"Стелаж {name} успішно додано!"
                )
            except Exception as e:
                QMessageBox.critical(
                    self.main_window,
                    "Помилка",
                    f"Не вдалося додати стелаж: {str(e)}"
                )
…..
class WarehouseDatabase:
    """Клас для роботи з базою даних складу"""
    def __init__(self, db_path="warehouse.db"):
        """Ініціалізація бази даних"""
        self.db_path = db_path
        self.create_tables()
    def create_tables(self):
        """Створення таблиць в базі даних, якщо вони не існують"""
        connection = sqlite3.connect(self.db_path)
        cursor = connection.cursor()
…...
```

ДОДАТОК В

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: ІНФОРМАЦІЙНА СИСТЕМА КЕРУВАННЯ КОМЕРЦІЙНИМ СКЛАДСЬКИМ ПРИМІЩЕННЯМ

Тип роботи: _____ бакалаврська кваліфікаційна робота _____
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ _____кафедра обчислювальної техніки_____
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism ___3____%

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____Азаров О.Д., зав. каф. ОТ _____ _____
(прізвище, ініціали, посада) (підпис)

____Крупельницький Л.В., доц. каф ОТ_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ ___Захарченко С.М.____
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ _____Дудник О.В_____
(підпис) (прізвище, ініціали, посада)

Здобувач _____ _____Рябоконь К.М_____
(підпис) (прізвище, ініціали)

ДОДАТОК Г
ЛІСТИНГИ

Лістинг 1.1. Вирішення квадратного рівняння на C++

```
#include <iostream>
#include <cmath>
int main() {
    double a, b, c;
    std::cout << "Введіть коефіцієнти a, b, c: ";
    std::cin >> a >> b >> c;
    double D = b * b - 4 * a * c;
    if (D > 0) {
        double x1 = (-b + sqrt(D)) / (2 * a);
        double x2 = (-b - sqrt(D)) / (2 * a);
        std::cout << "Корені: " << x1 << " і " << x2 << "\n";
    } else if (D == 0) {
        double x = -b / (2 * a);
        std::cout << "Один корінь: " << x << "\n";
    } else {
        std::cout << "Дійсних коренів немає.\n";
    }    return 0;
}
```

Лістинг 1.2. Вирішення квадратного рівняння на C#

```
using System;
class Program {
    static void Main() {
        Console.Write("Введіть коефіцієнти a, b, c: ");
        double a = Convert.ToDouble(Console.ReadLine());
        double b = Convert.ToDouble(Console.ReadLine());
        double c = Convert.ToDouble(Console.ReadLine());
        double D = b * b - 4 * a * c;
        if (D > 0) {
            double x1 = (-b + Math.Sqrt(D)) / (2 * a);
            double x2 = (-b - Math.Sqrt(D)) / (2 * a);
            Console.WriteLine($"Корені: {x1} і {x2}");
        } else if (D == 0) {
            double x = -b / (2 * a);
            Console.WriteLine($"Один корінь: {x}");
        } else {
            Console.WriteLine("Дійсних коренів немає.");
        }
    }
}
```

Лістинг 1.3. Вирішення квадратного рівняння на Java

```
import java.util.Scanner;
public class QuadraticSolver {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Введіть коефіцієнти a, b, c: ");
        double a = scanner.nextDouble();
        double b = scanner.nextDouble();
        double c = scanner.nextDouble();
        double D = b * b - 4 * a * c;
        if (D > 0) {
            double x1 = (-b + Math.sqrt(D)) / (2 * a);
            double x2 = (-b - Math.sqrt(D)) / (2 * a);
            System.out.println("Корені: " + x1 + " і " + x2);
        } else if (D == 0) {
            double x = -b / (2 * a);
            System.out.println("Один корінь: " + x);
        } else {
            System.out.println("Дійсних коренів немає.");
        }
    }
}
```

Лістинг 1.4. Вирішення квадратного рівняння на Python

```
import math
a = float(input("Введіть коефіцієнт a: "))
b = float(input("Введіть коефіцієнт b: "))
c = float(input("Введіть коефіцієнт c: "))
D = b ** 2 - 4 * a * c
if D > 0:
    x1 = (-b + math.sqrt(D)) / (2 * a)
    x2 = (-b - math.sqrt(D)) / (2 * a)
    print(f"Корені: {x1} і {x2}")
elif D == 0:
    x = -b / (2 * a)
    print(f"Один корінь: {x}")
else:
    print("Дійсних коренів немає.")
```

Лістинг 1.5. Інкапсуляція

```
class BankAccount:
    def __init__(self, balance):
        self.__balance = balance  # приватне поле
    def deposit(self, amount):
        if amount > 0:
            self.__balance += amount
    def withdraw(self, amount):
        if 0 < amount <= self.__balance:
            self.__balance -= amount
    def get_balance(self):
        return self.__balance
account = BankAccount(1000)
account.deposit(500)
account.withdraw(200)
print(account.get_balance())  # 1300
```

Лістинг 1.6. Наслідування

```
class Animal:
    def speak(self):
        return "Some sound"
class Dog(Animal):
    def speak(self):
        return "Woof"
class Cat(Animal):
    def speak(self):
        return "Meow"
dog = Dog()
cat = Cat()
print(dog.speak())  # Woof
print(cat.speak())  # Meow
```

Лістинг 1.7. Поліморфізм

```
def make_sound(animal):
    print(animal.speak())
animals = [Dog(), Cat()]
for a in animals:
    make_sound(a)  # викликає відповідну реалізацію speak()
```

Лістинг 1.8. Абстракція

```
from abc import ABC, abstractmethod
class Shape(ABC):
    @abstractmethod
    def area(self):
        pass
class Rectangle(Shape):
    def __init__(self, width, height):
        self.width = width
        self.height = height
    def area(self):
        return self.width * self.height
rect = Rectangle(10, 5)
print(rect.area())  # 50

car.drive()
```

Лістинг 3.7 Клас UIComponents

```
    @staticmethod
    def create_material_button(text, action, button_type="default"):
    @staticmethod
    def add_info_field(info_layout, row, title, value, value_style=None):
@staticmethod
    def create_card_frame(min_width=300, min_height=220, max_width=None, max_height=None, card_type="default"):
    @staticmethod
    def get_font(family="Roboto", size=14, bold=False, italic=False):
```

Лістинг 3.8 Створення представлення

```
# Створення картки
        card = UIComponents.create_card_frame(card_type="client")
        card.setProperty("client_id", client_id)
        card.setSizePolicy(QSizePolicy.Expanding, QSizePolicy.Preferred)
         # Створення основного лейауту для картки
        card_layout = QVBoxLayout(card)
        card_layout.setContentsMargins(15, 15, 15, 15)
        # Назва клієнта
        name_label = QLabel(name)
        name_label.setObjectName("nameLabel")
        name_label.setWordWrap(True)  # Дозволяємо перенесення тексту
        card_layout.addWidget(name_label)
        # Лінія розділення
        separator = QFrame()
        separator.setFrameShape(QFrame.HLine)
        separator.setFixedHeight(1)
        separator.setObjectName("separator")
        card_layout.addWidget(separator)

        # Інформація про клієнта
        info_layout = QGridLayout()
        info_layout.setVerticalSpacing(5)
        info_layout.setHorizontalSpacing(10)
        card_layout.addLayout(info_layout)
```

Лістинг 3.9 Прив’язка даних

```
# Контактна особа
UIComponents.add_info_field(info_layout, 0, "Контактна особа:", contact_person)
# Телефон
UIComponents.add_info_field(info_layout, 1, "Телефон:", phone)
# Email
UIComponents.add_info_field(info_layout, 2, "Email:", email)
card_layout.addStretch()
```

Лістинг 3.10 Прив’язка кнопок

```
  buttons_layout = QHBoxLayout()
buttons_layout.setSpacing(15)
edit_button = QPushButton("РЕДАГУВАТИ")
edit_button.setObjectName("editButton")
edit_button.setProperty("client_id", client_id)
edit_button.clicked.connect(edit_action)
edit_button.setMinimumWidth(120)
delete_button = QPushButton("ВИДАЛИТИ")
delete_button.setObjectName("deleteButton")
delete_button.setProperty("client_id", client_id)
delete_button.clicked.connect(remove_action)
delete_button.setMinimumWidth(120)
buttons_layout.addWidget(edit_button)
buttons_layout.addWidget(delete_button)
card_layout.addLayout(buttons_layout)
```