Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:
«Розробка методу та програмного засобу для автоматизації ведення складського обліку»

Виконав: студент II курсу
групи ЗПІ-22м спеціальності
121 - Інженерія програмного забезпечення
Сіянко М. А.
(прізвище та ініціали)

Керівник: д.т.н, професор каф. ПЗ
Ліщинська Л. Б.
(прізвище та ініціали)
«15» грудня 2023 р.

Опонент: к.т.н, доцент каф. ЗІ
Дудатьєв А. В.
«15» грудня 2023 р.

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О. Н.
(прізвище та ініціали)
«15» грудня 2023 р.

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти II-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
«19» вересня 2023 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Сіянку Микиті Олександровичу

1. Тема роботи – розробка методу та програмного засобу для автоматизації ведення складського обліку.

Керівник роботи: Ліщинська Людмила Броніславівна, д.т.н., професор кафедри ПЗ, затверджені наказом вищого навчального закладу від «18» вересня 2023 р. № 247.

2. Строк подання студентом роботи
5 грудня 2023 р.

3. Вихідні дані до роботи: графічний режим – TrueColor; система керування базами даних – Microsoft SQL Server; мова запитів – SQL; вхідні дані – дані з кожного вікна такі як: кількість елементів у вікні, новий користувач, нова таблиця; вихідні дані – таблиці з даними їх атрибутів; середовище розробки – IntelliJ IDEA; мова програмування – Java.

4. Зміст текстової частини: вступ; аналіз стану питання та порівняння аналогів; розробка графічного інтерфейсу програми та алгоритмів роботи додатку; розробка програмних компонент додатку; економічна частина; висновки; список використаних джерел; додатки.

5. Перелік ілюстративного матеріалу: титульний слайд; актуальність теми; мета, об'єкт та предмет дослідження; задачі дослідження; наукова новизна;

практична цінність одержаних результатів; порівняльний аналіз аналогів; графічна схема інтерфейсу головного вікна; графічна схема вікна QR-коду; графічна схема вікна товарів; графічна схема мобільного додатку сканера; інтеграція бази даних; тестування програми; апробація матеріалів магістерської кваліфікаційної роботи; фінальний слайд.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Ліщинська Л. Б., д.т.н., проф. кафедри ПЗ	19.09.23	05.12.23
5	Причепа І. В., к.е.н., доцент кафедри ЕПТВМ	28.11.23	01.12.23

7. Дата видачі завдання 19 вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи	Примітка
1	Аналіз стану питання	20.09.2023 – 24.09.2023	Вик.
2	Порівняльний аналіз аналогів	25.09.2023 – 01.10.2023	Вик.
3	Розробка графічного інтерфейсу додатку	02.10.2023 – 12.10.2023	Вик.
4	Розробка алгоритму роботи додатку	13.10.2023 – 23.10.2023	Вик.
5	Розробка програмних компонент додатку	24.10.2023 – 10.11.2023	Вик.
6	Тестування програмного забезпечення	11.11.2023 – 24.11.2023	Вик.
7	Економічна частина	25.11.2023 – 01.12.2023	Вик.

Студент _____ **Сіянко М. О.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Ліщинська Л. Б.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.912.032.26

Сіянко М. О. Розробка методу та програмного засобу для автоматизації ведення складського обліку: магістерська кваліфікаційна робота зі спеціальності 121 – Інженерія програмного забезпечення, освітня програма – Інженерія програмного забезпечення. Вінниця: ВНТУ, 2023, 104 с.

На укр. мові. Бібліогр. :5 розділів, 70 рис., 15 табл., 35 джерел, 5 додатків.

У Магістерській кваліфікаційній роботі проведено детальний аналіз методів для автоматизації складського обліку.

Запропоновано використати QR-коди для отримання інформації про товари що знаходяться на складі.

Було розроблено метод створення QR-кодів на основі даних користувача. Також було розроблено метод сканування QR-кодів для отримання закодованої у них інформації. Розроблено додаток для автоматизації складського обліку. При розробці використано мову програмування Java та технології Intelij IDEA.

Отримані в магістерській кваліфікаційній роботі результати можна використати для побудови додатку для автоматизованого складського обліку.

Ключові слова: автоматизація складського обліку, QR-код, бізнес, сканувати, генерувати, база даних.

ABSTRACT

Siyanko M. O. Development of a method and a software tool for the automation of warehouse accounting: master's thesis on the specialty 121 - software engineering, educational program - software engineering. Vinnytsia: VNTU, 2023. 104 p.

In Ukrainian speech Bibliogr. :5 chapters, 70 figures, 15 tables, 35 sources, 5 appendices.

In the Master's qualification work, a detailed analysis of methods for automating warehouse accounting was carried out.

It is suggested to use QR-codes to obtain information about goods in stock.

A method of creating QR codes based on user data was developed. A method of scanning QR codes to obtain the information embedded in them was also developed. An application for automating warehouse accounting has been developed. The Java programming language and Intelij IDEA technology were used in the development.

The results obtained in the master's qualification work can be used to build an application for automated warehouse accounting.

Keywords: warehouse accounting automation, QR code, business, scan, generate, database of melons.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Сьогодні різноманітні види інформаційних технологій пристосовані для вирішення великого списку проблем з якими може стикнутися людина. Це стосується і систем автоматизації складського обліку, які знаходиться на перетині актуальних викликів сучасного бізнес-середовища та інноваційних підходів до управління інформаційними потоками.

Швидке зростання конкуренції та обмеженість ресурсів у сучасному бізнесі підкреслюють необхідність вдосконалення та ефективного управління процесами складського обліку. Оптимізація цих процесів стає стратегічною важливістю для підприємств, оскільки вона не лише сприяє зниженню витрат, а й підвищує ефективність управління запасами, що, в свою чергу, дозволяє підприємствам більш гнучко реагувати на ринкові зміни.

Вдосконалення цих процесів стає важливою стратегічною задачею, оскільки вони сприяють не лише зниженню витрат, але й підвищенню ефективності управління запасами. Це, у свою чергу, дозволяє підприємствам більш гнучко реагувати на ринкові зміни та забезпечує їхню конкурентоспроможність.

У контексті зростання потреб у підвищенні продуктивності та ефективності управлінських процесів, розробка методу та програмного засобу для автоматизації ведення складського обліку стає важливою задачею. Автоматизація ведення складського обліку не лише оптимізує управління запасами, але й враховує сучасні вимоги до енергоефективності та часу автономної роботи обчислювальних пристроїв.

Сьогодні існує багато систем автоматизованого складського обліку проте більшість запропонованих ними способів вирішення завдань є поверхневими. Більшість подібних рішень представлені додатками створеними для малого або середнього бізнесу. При реалізації даних рішень виходять в основному вузькоспеціалізовані додатки спроба використання яких за межами їх визначеного середовища приводить до невтішних результатів. Схожі рішення для великих

підприємств як правило застарілі та мають у деяких випадках функціонал більш обмежений ніж у системах призначених для малого та середнього бізнесу.

Саме тому важливою задачею є створення методів та програмних засобів для автоматизації ведення складського обліку, оскільки існуючі реалізації мають свої недоліки. Це вимагає розробки нових методів та алгоритмів роботи, щоб забезпечити більш ефективний та оптимізований процес управління складськими ресурсами.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення

Мета та завдання дослідження. Метою дослідження є підвищення ефективності ведення автоматизованого складського обліку.

Основними задачами дослідження є:

- провести аналіз існуючих методів і засобів автоматизації складського обліку;
- розробити метод запису даних у базу даних системи;
- розробити метод створення QR-кодів для розпізнання у системі;
- розробити метод зчитування QR-кодів;
- розробити програмні компоненти та систему автоматизації складського обліку;
- провести тестування додатку.

Об'єкт дослідження – процеси автоматизації складського обліку під час використання додатку.

Предмет дослідження – методи та засоби автоматизації складського обліку.

Методи дослідження. У процесі досліджень використовувались: теорія алгоритмів для розробки алгоритмів і програмних модулів, лінійна алгебра, теорія баз даних для розробки структури бази даних, комп'ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Наукова новизна отриманих результатів.

1. Подальшого розвитку набув метод генерації QR-кодів, в якому, на відміну від існуючих було використано алгоритм Reed-Solomon для швидкого виявлення та виправлення помилок, що дозволило підвищити ефективність зберігання та передачі даних.

2. Подальшого розвитку набув метод зчитування QR-кодів, в якому, на відміну від класичного для підвищення якості зображення було використано медіальний фільтр, що дало можливість підвищити точність зчитування даних.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби для автоматизації складського обліку.

Особистий внесок здобувача. Усі наукові результати, що викладені у магістерській кваліфікаційній роботі, отримано автором самостійно. У наукових працях, опублікованих у співавторстві, автору належать такі результати: сучасні цифрові технології для автоматизованого управління складським обліком [1]; Генерація QR-кодів з використанням алгоритмів корекції помилок BCH та Reed-Solomon [2].

Апробація матеріалів магістерської кваліфікаційної роботи. Основні положення магістерської кваліфікаційної роботи доповідалися на науково-технічній конференції: Міжнародної науково-практичної Інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ» (Вінниця, 2023) «Молодь в науці: дослідження, проблеми, перспективи» (Вінниця 2023).

Публікації. За тематикою дослідження опубліковано 2 наукові праці у збірниках матеріалів конференцій.

1 АНАЛІЗ СТАНУ ПИТАННЯ ТА ПОРІВНЯННЯ АНАЛОГІВ

1.1 Аналіз стану питання

Сучасне бізнес-середовище зазнає значних змін завдяки зростанню обсягів товарів та послуг, покращенню технологій та змін у споживчих звичках. Один із найважливіших аспектів управління бізнесом – ефективне ведення складського обліку. З метою підвищення продуктивності, зменшення витрат та покращення якості обслуговування клієнтів багато підприємств вибирають автоматизовані системи для управління складами.

Сучасний бізнес вимагає відповідати високим стандартам точності, швидкості та надійності під час ведення обліку складських запасів. Потрібно враховувати багато факторів, включаючи обсяги товарів, терміни придатності, споживання та подання на ринку, транспортні витрати та багато інших параметрів. У такому складному середовищі спростити процес ведення обліку та зменшити можливість помилок може лише автоматизація складського обліку [3].

Важливим аспектом автоматизації складського обліку є можливість у реальному часі відстежувати рух товарів на складі, контролювати терміни придатності та оптимізувати запаси. Така система дозволяє підприємствам швидше реагувати на зміни в попиті та постачаннях, що у свою чергу покращує обслуговування клієнтів та зменшує витрати на управління запасами.

Крім того, автоматизовані системи складського обліку дозволяють контролювати терміни придатності товарів. Важливо для підприємств, які працюють із продуктами харчування чи іншими товарами, чутливими до термінів придатності. Системи можуть автоматично сповіщати про наближення термінів придатності та нагадувати про необхідність здійснити ротацію товарів або списати прострочені продукти. Це допомагає уникнути втрат і зберегти якість продукції.

Зазвичай, автоматизація складського обліку допомагає оптимізувати запаси. Системи можуть вести статистику про обороти товарів, попит клієнтів та інші фактори. За допомогою цієї інформації, підприємства можуть приймати

більш обґрунтовані рішення щодо обсягів закупівель, замовлень та зберігання товарів. Це дозволяє підприємствам підтримувати оптимальні рівні запасів, знижувати зайві витрати та оптимізувати процеси поставок.

Зазначена необхідність ефективного ведення складського обліку призвела до розвитку різноманітних програмних рішень та технологій для автоматизації цього процесу. Найбільш сучасні системи включають в себе інтелектуальний аналіз даних, використання штучного інтелекту та можливості інтеграції з іншими бізнес-процесами.

Системи, які включають в себе інтелектуальний аналіз даних вміють прогнозувати потреби в тих чи інших товарах на основі історичних даних та поточних тенденцій. Це допомагає оптимізувати запаси та уникнути надлишків чи недостач.

Використання штучного інтелекту в автоматизації складського обліку робить системи ще більш раціональними. Штучний інтелект допомагає розпізнавати шаблони та аналізувати великі обсяги даних, що важко виконати вручну. Він також може прогнозувати збої та проблеми, дозволяючи вчасно реагувати.

Інтеграція з іншими бізнес-процесами є ще однією перевагою сучасних систем. Дані про стан запасів можуть легко інтегруватися з процесами замовлення, виробництва та поставок. Це дозволяє створювати єдину інформаційну систему для підприємства, яка оптимізує всі процеси та полегшує прийняття рішень.

На сьогоднішній день, успішні підприємства розглядають автоматизацію складського обліку не лише як інструмент оптимізації внутрішніх процесів, а й як засіб досягнення конкурентних переваг на ринку. У разі правильного вибору та впровадження системи автоматизації підприємство може покращити якість обслуговування клієнтів, знизити витрати та бути готовим до подальшого зростання обсягів бізнесу [4].

Зниження витрат - ще одна значуща перевага автоматизації складського обліку. Системи дозволяють ефективно використовувати ресурси та зменшити

зайві витрати, пов'язані з неефективним управлінням запасами. Зокрема, можливість точного ведення обліку запобігає збиткам через прострочені товари та дозволяє оптимізувати запаси згідно з реальним попитом.

Однак, можливість бути готовим до подальшого зростання обсягів бізнесу є можливо найважливішою перевагою автоматизації складського обліку. Підприємства, які вибрали правильну систему та впровадили її належним чином, готові до розширення своєї діяльності без суттєвого збільшення витрат на управління запасами. Це робить їх більш гнучкими та конкурентоспроможними в ринковому середовищі, де швидкість реакції на зміни може визначити переможців.

Отже, автоматизація ведення складського обліку стала необхідністю у сучасному бізнес-середовищі, а також інструментом для досягнення конкурентних переваг. Інноваційні рішення та технології в цій галузі пропонують широкі можливості для оптимізації управління запасами та забезпечення підприємствам гнучкості та ефективності в конкурентному бізнес-середовищі.

1.2 Порівняльний аналіз аналогів

На сьогоднішній день існує безліч рішень для автоматизації складського обліку, розроблених різними виробниками. Деякі з них надають загальні рішення, які підходять для різних видів бізнесу, натомість інші спеціалізуються на певних галузях. Оцінка і вибір оптимального рішення є важливим завданням для підприємств.

Для порівняльного аналізу було обрано наступні програмні продукти:

– УкрСклад;

– Торгсофт;

– Dilovod;

– RemOnline.

Це досить популярні програмні рішення в цій сфері. Їх аналіз дозволить визначити критично-важливий функціонал та найбільш значні недоліки ПЗ для

автоматизації ведення складського обліку, які можна буде використати при розробці власних методів та алгоритмів. Розглянемо більш детально індивідуальні особливості кожного програмного рішення.

УкрСклад – це програмна система для автоматизації управління складським обліком та логістикою підприємств. Ця програма розроблена для оптимізації процесів складського обліку, підвищення ефективності та зниження витрат у сфері управління запасами.

На рисунку 1.1 наведено інтерфейс УкрСклад.

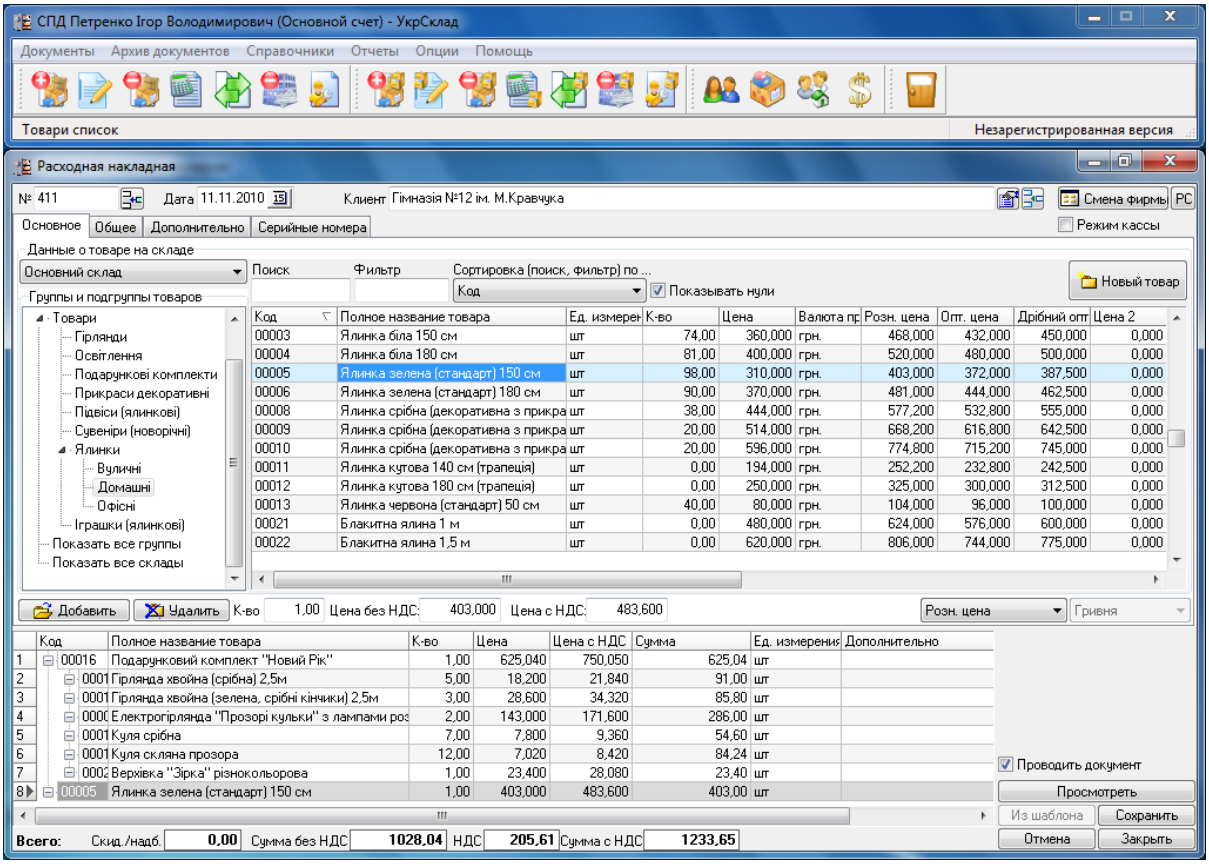

Рисунок 1.1 – Інтерфейс програмного додатку УкрСклад

УкрСклад є досить популярним рішенням для тих хто переходить з 1С та BAS через схожість графічного інтерфейсу. Він надає широкий спектр функцій від аналізу даних і створення звітів до контролю за запасами. Останні версії підтримують мобільні додатки що полегшує ведення обліку на місці. Також присутня підтримка інтеграції з іншими програмами [5]. Проте такий застарілий

зовнішній вигляд та перевантажений інтерфейс можуть відштовхнути деяких клієнтів.

Торгсофт – це програмне рішення для управління бізнесом та складським обліком. Ця програма допомагає підприємствам оптимізуватиз управління запасами, ведення обліку товарів, обробку замовлень і багато інших аспектів бізнесу.

На рисунку 1.2 наведено інтерфейс Торгсофт.

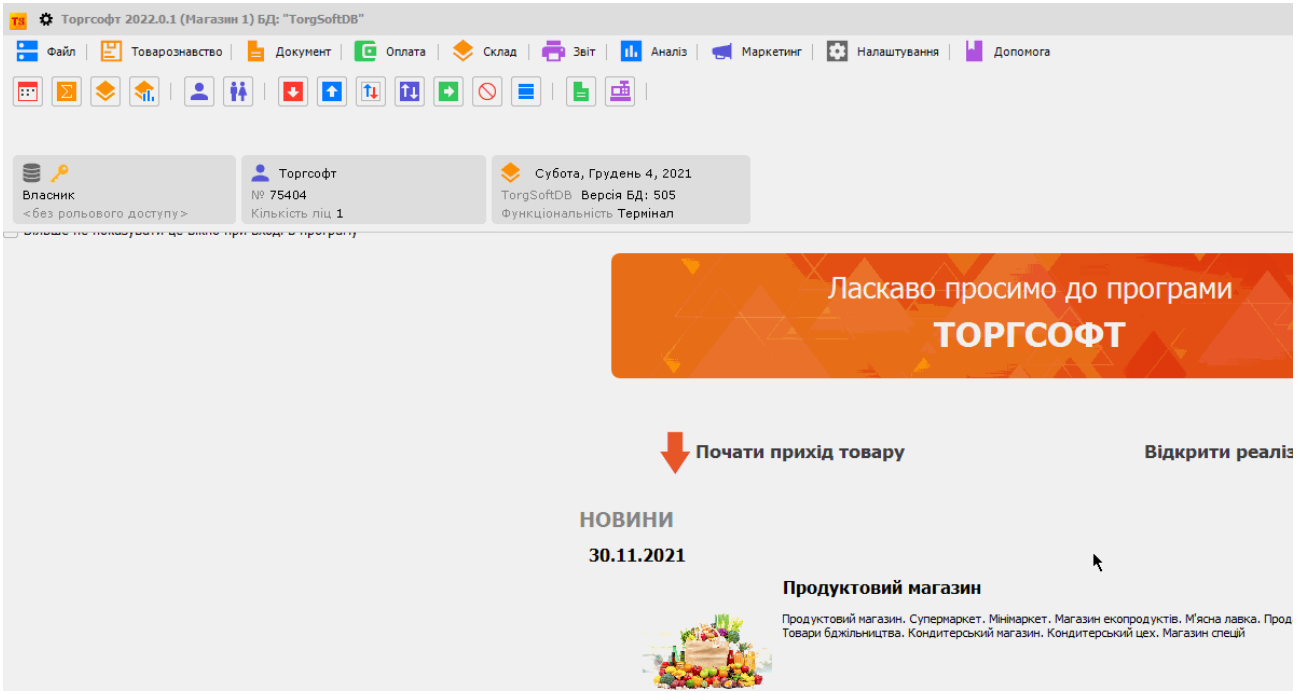

Рисунок 1.2 – Інтерфейс програмного додатку Торгсофт

Торгсофт є корисним рішенням для підприємств, які прагнуть покращити управління запасами, оптимізувати процеси та підвищити ефективність своєї діяльності. Програма надає інструменти для керування складськими операціями в режимі реального часу, що допомагає підприємствам бути більш конкурентоспроможними на ринку [6].

Dilovod – це програмна система для управління бізнесом, включаючи управління складським обліком та логістикою. Ця програма розроблена з метою полегшити процес ведення обліку товарів, замовлень та інших бізнес-процесів.

На рисунку 1.3 наведено інтерфейс Dilovod.

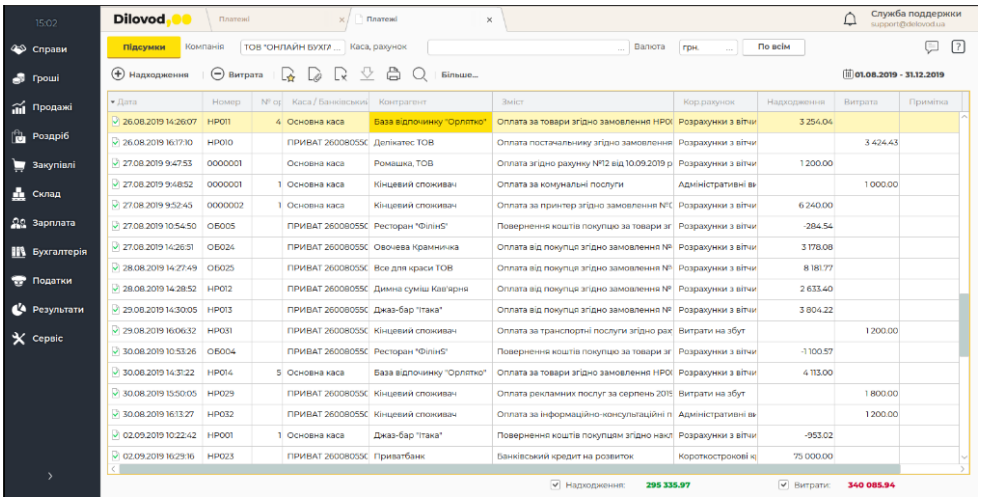

Рисунок 1.3 – Інтерфейс програмного додатку Dilovod

Dilovod допомагає підприємствам покращити управління запасами, оптимізувати процеси та підвищити ефективність своєї діяльності. Програма надає інструменти для керування складськими операціями та забезпечує контроль над багатьма аспектами бізнесу [7].

RemOnline – це програма для автоматизації бізнес-процесів та контролю роботи персоналу. Цей універсальний продукт для організації ефективної роботи виробництва дозволить налагодити дієве управління та отримувати більше прибутку.

На рисунку 1.4 наведено інтерфейс RemOnline.

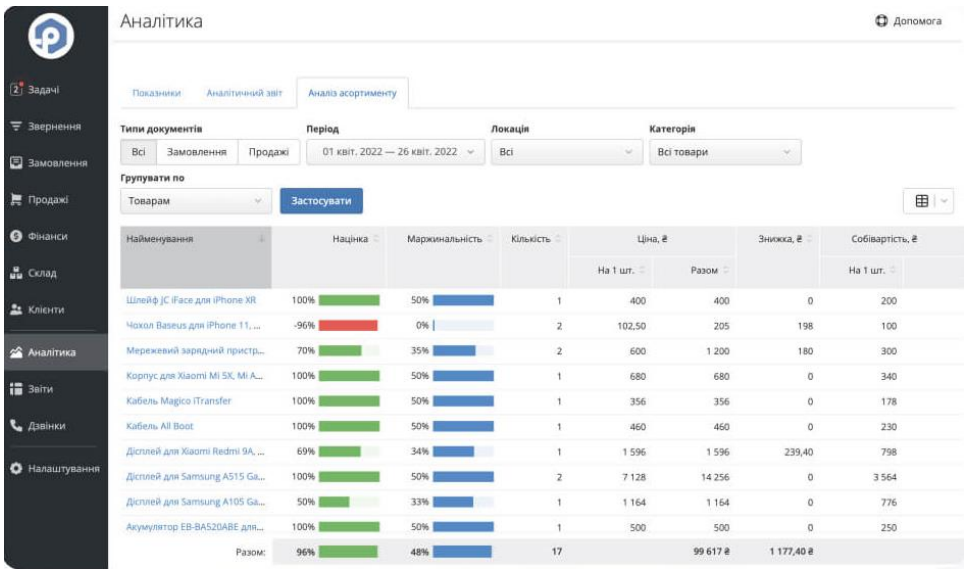

Рисунок 1.4 – Інтерфейс програми RemOnline

RemOnline забезпечує контроль над багатьма аспектами бізнесу завдяки своєму широкому набору інструментів. Також програма дозволяє більш тонко взаємодіяти зі складськими операціями для підвищення ефективності роботи підприємства [8].

Після дослідження усіх аналогів було визначено список критеріїв для їх оцінювання в порівнянні з власним програмним продуктом «Auto-storage». Результати порівняння наведено в таблиці 1.1.

Таблиця 1.1 – Порівняльні характеристики програмних продуктів

Критерій	УкрСклад	Торгсофт	Dilovod	RemOnline	Auto-storage
Масштабність і спроможність масштабуватися	+	+/-	+/-	+	+
Зручність в роботі та інтуїтивний інтерфейс	-	+/-	+	+	+
Можливість роботи офлайн	+	-	-	-	+
Інтеграція з іншими системами	+	+	+	+	+
Сумісність з обладнанням	-	-	-	+	+
Геолокація та маршрутизація	-	-	-	-	-
Підсумок	3	2	2,5	4	5

Розглянемо обрані для оцінювання критерії більш детально. Масштабність та спроможність масштабуватися повноцінно присутня лише у УкрСклад та RemOnline. Інші продукти-аналоги можуть масштабуватися лише із малого до середнього бізнесу.

Всі продукти-аналоги мають дружній та інтуїтивний інтерфейс окрім УкрСклад, інтерфейс якого є перевантаженим. Також взаємодія Торгсофт потребує трохи більш детального вивчення інструкції користувача.

Лише УкрСклад як більш стара програма підтримує роботу офлайн. Всі розглянуті продукти-аналоги мають змогу інтегруватися з іншими системами. Проте сумісність з обладнанням має лише RemOnline. Геолокацію та маршрутизацію не підтримує жоден із розглянутих продуктів.

Таким чином, в підрозділі було детально розглянуто популярні програмні продукти для автоматизації складського обліку, такі як УкрСклад, Торгсофт, Dilovod та RemOnline. Вивчено їх функціонал, переваги та недоліки. Було відібрано ряд критеріїв для порівняння характеристик цих програмних продуктів та потенційної власної розробки під назвою «Auto-storage». Згідно цього аналізу Auto-storage за обраними критеріями отримав 5 балів й він на 30% краще за RemOnline, який отримав 4 бали, та на 40% – за УкрСклад, що отримав 3 бали, а також на 65% краще за Торгсофт та Dilovod, що отримали по 2 та 2,5 бали. Можна зробити висновок, що розробка власного програмного продукту є доцільною та дозволить конкурувати з існуючими аналогами.

1.3 Аналіз методів розв’язання задачі

Масштабованість є бажаною властивістю системи, мережі, або процесу, яка свідчить про здатність системи робити більший обсяг роботи або бути легко розширеною. Наприклад, це може позначати здатність системи до збільшення загальної пропускної спроможності відповідно до підвищеного навантаження, коли додано ресурси. Є два види маштабування програмного забезпечення: горизонтальне та вертикальне [9].

- Вертикальне масштабування, або масштабування вгору, відноситься до процесу розширення і вдосконалення одного сервера або обладнання, щоб підвищити його продуктивність і здатність обробляти більший обсяг даних або навантаження. Вертикальне масштабування використовується для покращення продуктивності і надійності окремого сервера або вузла, а не для додавання нових серверів, як це відбувається у горизонтальному масштабуванні.
- Горизонтальне збільшення масштабу (горизонтальне масштабування) відноситься до розширення системи, додавання нових ресурсів або обладнання, щоб підвищити продуктивність і забезпечити можливість обробки більших обсягів даних або навантаження на існуючу систему. Це один із способів вдосконалення і масштабування інформаційних технологій та комп'ютерних систем у бізнесі.

Питання масштабування системи можна вирішити шляхом вдосконалення апаратної частини програмного продукту.

Сумісність з обладнанням для різних систем може бути вирішена двома способами. Перший спосіб – це модифікація коду програми для інтеграції драйверу для роботи пристрою та пряме або бездротове підключення його до робочої машини, на якій розташована система. Другий метод – це модифікація коду програми для імітування функцій, що мають виконуватись периферійними пристроями. Слід звернути увагу, що не всі функції можливо зімітувати в залежності від пристрою. Також є третій варіант у відриві від перших двох – це використання деяких пристроїв окремо від системи.

Таким чином, в даному розділі було розглянуто існуючі методи для досягнення масштабування системи та способи інтеграції програмного продукту із периферійним обладнанням. На їх основі можна розробити вдосконалені методи для власного програмного продукту.

1.4 Висновки

У першому розділі було розглянуто питання складського обігу та програмного забезпечення для його автоматизації. Наведено причини та приклади застосування такого ПЗ. Було досліджено предметну область, проведено аналіз існуючих аналогів на основі ряду критеріїв, наведено їх переваги й недоліки. Аналіз поточного стану питання та існуючих аналогів довів доцільність розробки власного програмного продукту для автоматизації складського обліку, що зможе надати кінцевому користувачу новий функціонал та виправити виявлені недоліки аналогів. Проведено аналіз існуючих методів для вирішення поставлених задач.

2 РОЗРОБКА СТРУКТУРИ СИСТЕМИ ТА ВДОСКОНАЛЕННЯ МЕТОДІВ ГЕНЕРАЦІЇ ТА ЗЧИТУВАННЯ QR-КОДІВ

2.1 Вдосконалення методу генерації QR-кодів

За останні роки QR-коди стали необхідною частиною сучасної інформаційної технології. Вони використовуються в різноманітних сферах, включаючи маркетинг, логістику, здоров'я та багато інших. Однак, проблема надійності та читання QR-кодів у випадку пошкодження їхньої структури є актуальною. Пошкодження, викликане фізичними або технічними факторами, може призвести до втрати даних та неправильного розпізнавання. Алгоритми корекції помилок мають на меті виправити ці проблеми та забезпечити надійність QR-кодів [10].

Значний обсяг досліджень був приділений питанню корекції помилок в QR-кодах. Роботи таких вчених, як Хемінг, Боуз, Чодурі та Хокенгем, виявилися ключовими для розвитку алгоритмів корекції помилок. Завдяки їх дослідженням було виявлено два найбільш популярних алгоритми для корекції помилок:

- Алгоритм BCH (Bose-Chaudhuri-Hocquenghem);
- Алгоритм Reed-Solomon.

Алгоритм BCH (Bose-Chaudhuri-Hocquenghem) — це алгоритм для корекції помилок у кодах звільнення передачі даних, таких як коди коригування помилок. Цей алгоритм був розроблений в 1959 році Л. Босе, Р. Чаудхурі та А. Хокенгемом, і він став основою для багатьох інших кодів коригування помилок.

Основним завданням кодів коригування помилок є виявлення та виправлення помилок у переданих даних. Код BCH використовується для створення кодів, які можуть виявляти та виправляти певну кількість помилок в переданих бітах [11].

Математично алгоритм BCH базується на теорії поля Галуа. Він використовує поліноми для представлення інформації, а операції виконуються над елементами поля Галуа. Основна ідея полягає в тому, що множення і

додавання відбуваються за певними правилами, які дозволяють здійснювати операції з обмеженим набором значень.

Математична формула, яка описує код BCH, виглядає наступним чином:

$$C(x) = \sum_{i=0}^{n-k} c_i * x^i, \quad (2.1)$$

де: $C(x)$ – поліном коду BCH, n – загальна кількість біт у коді (включаючи коригуючі біти), k – кількість інформаційних біт, c_i – коефіцієнти полінома.

Код BCH дозволяє виявляти та коригувати помилки у переданих даних, що робить його ефективним для застосувань, де важлива надійність передачі інформації, наприклад, у зв'язку та зберіганні даних.

Розглянемо переваги та недоліки алгоритму BCH:

Переваги:

- Високий рівень корекції помилок, що робить його ефективним для систем із значними шумами чи пошкодженням даних;
- Добре пристосований до деяких видів помилок, зокрема до бурштинових помилок.

Недоліки:

- Вимагає більше обчислень та ресурсів для корекції помилок порівняно із Reed-Solomon;
- Може бути менш ефективним у випадках невеликої кількості помилок.

Алгоритм Reed-Solomon (RS) - це інший тип кодування корекції помилок, який використовується для забезпечення цілісності та відновлення даних під час передачі чи зберігання. Цей алгоритм був винайдений Іром Соломоном та Густавом Рідом у 1960 році, і з тих пір він широко використовується в різноманітних сферах, таких як зберігання інформації на компакт-дисках, QR-кодах, бездротовому зв'язку та багатьох інших [12].

Математична основа RS-кодів пов'язана з поліноміальною арифметикою над скінченим полем Галуа. Основна ідея полягає в тому, що всі арифметичні

операції відбуваються за модулем певного простого числа. Важливою частиною RS-кодів є використання поліномів і степеневої арифметики.

Математична формула RS-коду може бути виражена наступним чином:

$$C(x) = (m(x) * x^t) \bmod g(x), \tag{2.2}$$

де: $C(x)$ – поліном коду RS, $m(x)$ – поліном інформації, який виражає оригінальні дані, x^t – додаткові біти коригування, $g(x) -$ поліном – генератор $RS -$ коду.

Основна мета RS-коду - додавати деяку кількість додаткових бітів до оригінальних даних, таким чином, щоб можливо було виявляти та виправляти помилки при передачі або зберіганні інформації. Параметри, такі як кількість додаткових бітів та степінь поліномів, визначаються виходячи з конкретних вимог задачі.

RS-коди відомі своєю високою ефективністю у виправленні помилок та широким спектром застосувань, що робить їх популярними у сучасних системах зберігання та передачі даних.

Розглянемо переваги та недоліки алгоритму Reed-Solomon:

Переваги:

- Добре пристосований до виявлення та корекції невеликої кількості помилок;
- Використовує менше ресурсів порівняно із BCH.

Недоліки:

- Може бути менш ефективним у випадках великої кількості помилок;
- Не так ефективний для бурштинових помилок, як BCH.

Обидва алгоритми чудово підходять для роботи для виправлення помилок при зберігання та передачі даних. Проте алгоритм BCH є більш витратним у плані ресурсів, що також не сприяє його швидкодії порівняно з алгоритмом Reed-Solomon. Тому для виявлення та виправлення помилок під час генерації QR-кодів біло обрано саме алгоритм Reed-Solomon.

Таким чином було проведено аналіз та обрано найбільш підходящий метод для покращення генерації QR-кодів.

2.2 Вдосконалення методу зчитування QR-кодів

QR-коди широко використовуються для зручного обміну інформацією, але якість зображення може суттєво впливати на їхню розпізнаваність. Але використання фільтрів може покращити якість зображення QR-кодів та зробити їх більш придатними для ефективного сканування. Найбільш популярними фільтрами є:

- Гаусівський фільтр;
- Медіальний фільтр.

Гаусівський фільтр — це метод обробки зображень, який використовує гаусівську функцію для розмиття та зменшення шуму на зображенні. Він широко застосовується для поліпшення якості зображень перед подальшою обробкою, такою як розпізнавання об'єктів або визначення контурів [13].

Гаусівський фільтр визначається гаусівською функцією:

$$G(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right), \quad (2.3)$$

де: $G(x,y)$ – значення фільтра в точці (x,y), σ – стандартне відхилення, яке визначає ступінь розмиття. Чим більше σ, тим менше впливу мають віддалені пікселі на центральний піксель.

Функцію $G(x,y)$ можна розглядати як ядро фільтра. Воно має гаусівську форму, концентруючи увагу навколо центрального пікселя. Чим ближче піксель до центру, тим більший вклад він має.

Розглянемо застосування фільтру для сканування QR-кодів:

- Розмиття та зменшення шуму. Гаусівський фільтр корисний для розмиття та зменшення шуму на зображенні, що допомагає покращити якість зображення QR-кодів;

- Підвищення контрастності. Розмиття гаусівським фільтром може допомогти відокремити контури та структуру QR-кодів;
- Адаптація до умов освітлення. Гаусівський фільтр може допомогти розмити або зменшити вплив освітлення, що може допомогти покращити розпізнаваність QR-кодів в різних умовах.

Розглянемо переваги та недоліки гаусівського фільтру.

Переваги:

- Ефективність при розмитті. Гаусівський фільтр ефективно розмиває зображення, зменшуючи деталі та загальний шум;
- Гладкі результати. Результати гаусівського розмиття виглядають гладкими та природними, що може бути корисним для покращення загальної якості зображення;
- Легкість в налаштуванні. Параметри гаусівського фільтру, такі як стандартне відхилення (σ), можна легко налаштувати для досягнення різних ефектів розмиття;
- Ефективність для гладких зображень. Гаусівський фільтр особливо ефективний для роботи з гладкими та низькоконтрастними зображеннями;
- Використання у фронтальних та розподілених обчисленнях. В силу своєї математичної природи, гаусівський фільтр легко впроваджується в апаратне та програмне забезпечення для обчислень.

Недоліки:

- Розмивання різких контурів. Гаусівський фільтр, зокрема при великих значеннях стандартного відхилення (σ), може призводити до розмивання різких контурів та деталей;
- Потреба в підборі параметрів. Вибір оптимального значення стандартного відхилення (σ) впливає на результат розмиття, і його слід підбирати експериментально для конкретного завдання;
- Обчислювальна витратність. Для великих зображень та значень σ гаусівський фільтр може бути витратним з обчислювальної точки зору;

• Можливе розмиття деталей. Якщо зображення містить важливі деталі, гаусівське розмиття може призвести до втрати цих деталей;

• Вибірковість за площиною. Гаусівський фільтр є вибірковим за площиною, що означає, що він впливає на всі пікселі у вказаній області однаково, незалежно від їхнього значення чи структури.

Медіанний фільтр – ефективний метод фільтрації, який використовує медіану для обробки зображень. Цей фільтр особливо корисний для видалення шуму та аномалій, зберігаючи при цьому різкі контури об'єктів на зображенні [14].

Медіанний фільтр використовує медіану у вказаному околі для визначення нового значення пікселя. Формула виглядає так:

$$Imedian(x, y) = median(I(x', y')), \tag{2.4}$$

де: $Imedian(x, y)$ – нове значення пікселя в точці (x, y), $median(\dots)$ – медіана значень у вказаному околі.

Робота медіального фільтру проходить у 4 етапи:

1. Вибір околу. Медіанний фільтр вибирає певний окіл для кожного пікселя. Це може бути квадратне вікно або інша форма залежно від вибору користувача чи специфікацій завдання;

2. Сортування значень. Значення пікселів у вибраному околі сортуються за зростанням;

3. Визначення медіани. Медіана обраного вікна визначається як середнє значення середнього та правого значень (якщо кількість значень парна) або просто як середнє значення центрального значення (якщо кількість значень непарна);

4. Присвоєння нового значення. Отримана медіана стає новим значенням пікселя.

Розглянемо застосування фільтру для сканування QR-кодів:

• Видалення шуму та аномалій. Медіанний фільтр добре працює для видалення різкого, локального шуму та аномалій, що можуть виникати при зйомці зображень QR-кодів;

• Збереження різких контурів. Оскільки медіанний фільтр не впливає на різкі контури, він є ефективним для обробки зображень з чіткими границями QR-кодів;

• Адаптація до освітлення. Медіанний фільтр може бути ефективним для зменшення впливу освітлення на зображення, зберігаючи при цьому важливі деталі.

Розглянемо переваги та недоліки медіального фільтру.

Переваги:

• Видалення шуму та аномалій. Медіанний фільтр ефективно видаляє шум та аномалії на зображенні, особливо в тих випадках, коли шум представляє собою випадкові високі чи низькі значення;

• Збереження різких контурів. Оскільки він використовує медіану, фільтр добре зберігає різкі контури та деталі, не розмиваючи їх;

• Простота та легкість в налаштуванні. Медіанний фільтр дуже простий у реалізації та має мало параметрів для налаштування, що робить його легким у використанні;

• Ефективність при локальному шумі. Виконує добре при роботі з локальним шумом чи аномаліями, оскільки враховує лише значення пікселів у вказаному околі;

• Інваріантність до яскравості. Медіанний фільтр менш чутливий до яскравості зображення порівняно з іншими фільтрами.

Недоліки:

• Обчислювальна витратність. В порівнянні з іншими фільтрами, такими як гаусівський, медіанний фільтр може вимагати більше обчислень, особливо при великому розмірі вибраного вікна;

• Вплив розміру вікна на деталі. Великий розмір вибраного вікна може призвести до втрати деяких деталей та структур, особливо при наявності різких контурів;

• Неможливість видалення глобального шуму. Медіанний фільтр може бути менш ефективним у випадках глобального шуму, де всі пікселі у великому вікні мають аномальні значення;

• Неінваріантність до поворотів та масштабування. Медіанний фільтр може погано працювати в умовах поворотів чи масштабування зображення;

• Залежність від розподілу шуму. Якщо шум на зображенні має складну структуру, медіанний фільтр може бути менш ефективним у порівнянні з іншими фільтрами.

Кожен з представлених фільтрів чудово може вирішити проблему якості зображення при сканування QR-кодів. Але зважаючи на розглянуті сильні та слабкі сторони кожного з них для виконання завдання було обрано медіальний фільтр через простоту його налаштування та меншу чутливість до яскравості зображення.

Таким чином було проведено аналіз та обрано найбільш зручний метод для покращення зчитування QR-кодів.

2.4 Розробка архітектури додатку

Архітектура мікросервісів є підходом до розробки програмного забезпечення, в якому програма розбивається на невеликі та незалежні компоненти, названі мікросервісами. Кожен мікросервіс представляє собою окремий функціональний модуль, який може виконувати конкретні завдання або надавати конкретну послугу. Основні характеристики архітектури мікросервісів включають:

1. Незалежність мікросервісів. Кожен мікросервіс розробляється, випускається та виправляється незалежно від інших. Це дозволяє командам розробників працювати над окремими функціями або сервісами без впливу на інші частини системи;

2. Легкість масштабування. Кожен мікросервіс може бути масштабований окремо, що дозволяє оптимізувати ресурси системи та підтримувати високу доступність лише для необхідних компонентів;

3. Комунікація через API. Мікросервіси спілкуються між собою за допомогою API (інтерфейсів програмування застосунків). Це дозволяє їм ефективно обмінюватися інформацією та послугами;

4. Розподілена розробка. Команди розробників можуть працювати над різними мікросервісами незалежно одна від одної. Це полегшує швидкість розробки та впровадження нового функціоналу;

5. Наявність власної бази даних. Кожен мікросервіс може використовувати свою власну базу даних, що спрощує схему баз даних та зменшує залежність між компонентами системи;

6. Моніторинг та логування. Завдяки незалежності мікросервісів, можливо ефективно відслідковувати та моніторити роботу кожного сервісу окремо, що полегшує виявлення та вирішення проблем;

7. Легка заміна та оновлення. Зміни в одному мікросервісі не повинні впливати на решту системи. Це дозволяє впроваджувати оновлення та виправлення без великих ризиків для всієї системи.

Хоча архітектура мікросервісів пропонує багато переваг, вона також вносить виклики, такі як складність управління великою кількістю сервісів та необхідність ефективного керування комунікацією між ними. Однак при правильному проектуванні та управлінні, мікросервісна архітектура може стати потужним інструментом для розробки складних та високопродуктивних програмних систем.

Монолітна архітектура — це підхід до розробки програмного забезпечення, при якому весь додаток будується як єдина, нероздільна одиниця. У монолітних додатках весь функціонал об'єднаний в одному великому кодовому базисі та запускається в одному процесі. Основні характеристики монолітної архітектури включають:

1. Єдина кодова база. Усі компоненти та функції додатка знаходяться в єдиному кодовому репозиторії. Це робить розробку та управління кодом більш простими;

2. Спільні технології та інструменти. Усі частини додатка використовують спільні технології та інструменти розробки. Це спрощує процес розробки та підтримки;

3. Зручність монолітного впровадження. Додаток встановлюється та запускається як один єдиноцільний модуль, що спрощує деплоймент та масштабування;

4. Загальна база даних. В монолітних додатках зазвичай використовується єдина база даних, яка обслуговує всі компоненти системи;

5. Простота комунікації між компонентами. Оскільки всі частини додатка знаходяться в одному процесі, комунікація між ними може бути виконана дуже ефективно та без значного накладу;

6. Монолітні фреймворки. Зазвичай використовуються монолітні фреймворки, які надають стандартний набір інструментів та практик для розробки.

Хоча монолітна архітектура має свої переваги, такі як простота розробки та управління, вона може стати обмеженням у випадках, коли додаток стає великим і складним. Основні виклики монолітної архітектури включають обмежену масштабованість, складність внесення змін та важкість розділення функціоналу для розробки в розподілених командах.

На відміну від архітектури мікросервісів, де різні компоненти можуть бути розгорнуті та оновлюватися незалежно, монолітна архітектура вимагає розгортання та оновлення всього додатка як єдиної одиниці.

Архітектура додатку для автоматизації складського обліку наведена на рисунку 2.1.

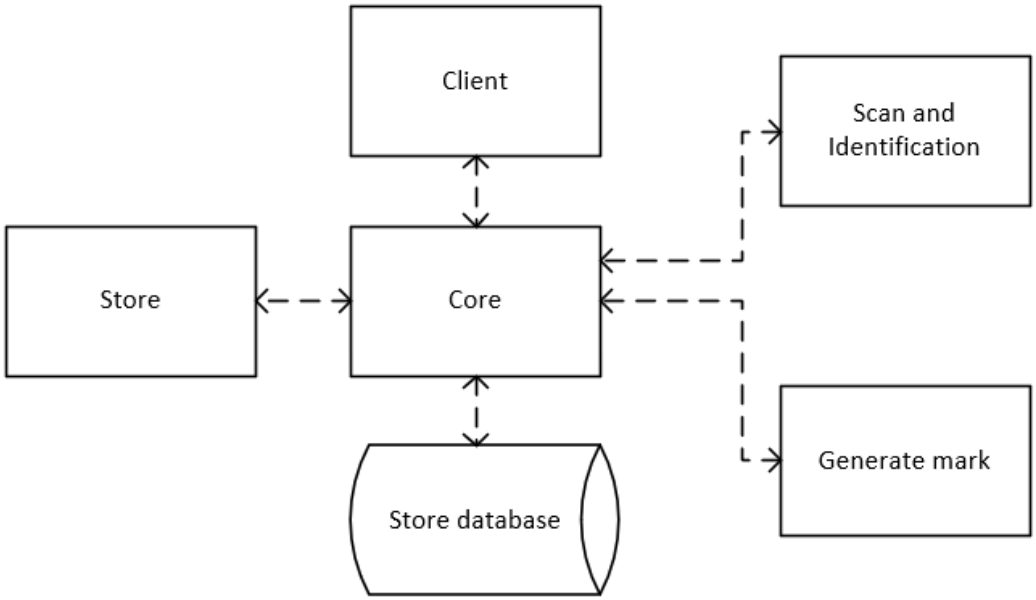

Рисунок 2.1 – Архітектура додатку

Для розробки програмного продукту було обрано архітектуру мікросервісів, що відповідає усім вимогам розробки. Додаток буде складатися з наступних компонентів:

- Client – модуль, що забезпечує графічний інтерфейс для взаємодії з користувачем;
- Store – модуль, що відповідає за ведення обліку товарів на складі;
- Scan and Identification – модуль, що забезпечує зчитування та ідентифікацію міток для користувача;
- Generate mark – модуль, що відповідає за генерацію міток для розпізнавання;
- Store database – модуль, що зберігає дані про товари та забезпечує доступ до даних для інших компонентів через Core;
- Core – основний модуль системи, який забезпечує взаємодію між її частинами.

Таким чином для додатку було обрано архітектуру мікросервісів. Створено схему та описано її елементи.

2.5 Висновки

В другому розділі було удосконалено метод генерації QR-кодів. В результаті чого було проведено аналіз алгоритмів для корекції помилок BCH (Bose-Chaudhuri-Hocquenghem) та Reed-Solomon. Вони обидва чудово підходили для виправлення помилок при зберігання та передачі даних. Проте алгоритм BCH є більш витратним у плані ресурсів, що також не сприяє його швидкодії порівняно з алгоритмом Reed-Solomon. Тому для виявлення та виправлення помилок під час генерації QR-кодів біло обрано саме алгоритм Reed-Solomon. Також було покращено метод для зчитування QR-кодів, для чого було проведено аналіз фільтрів для покращення якості зображення, а саме: Гаусівським та Медіальним. Але зважаючи на розглянуті сильні та слабкі сторони кожного з них для виконання завдання було обрано медіальний фільтр через простоту його налаштування та меншу чутливість до яскравості зображення.

Розроблено архітектуру системи, що використовує мікросервісний метод. Створено схему архітектури додатку та описано її елементи.

3 РОЗРОБКА ПРОГРАМНИХ КОМПОНЕНТІВ ДОДАТКУ

3.1 Розробка структури графічного інтерфейсу веб-додатку

Інтерфейс – це сукупність засобів, методів і правил взаємодії між елементами системи. Є декілька типів інтерфейсів:

- Графічні інтерфейси користувача – тип інтерфейсу, що дозволяє користувачу взаємодіяти з електронними пристроями через графічні зображення та візуальні вказівки, на відміну від текстових інтерфейсів, заснованих на використанні тексту [15];
- Інтерфейси командного рядка – різновид текстового інтерфейсу, в якому комп'ютеру можна дати інструкції лише введенням текстових рядків (команд), також відомий під назвою консоль. Приклад такого інтерфейсу представлений на рисунку [16];
- Веб-інтерфейси – сукупність засобів за допомогою яких користувач взаємодіє з вебсайтом або вебзастосунком через браузер. Даний інтерфейс зображено на рисунку [17].

Врахувавши всі переваги та недоліки вище описаних інтерфейсів, створюваний додаток буде використовуватися для інсталяції на робочі машини, тому веб-інтерфейс не підходить. Також неправильним буде використання інтерфейсу командного рядка, так як додаток буде призначений для широкого кола користувачів. Отже, врахувавши все вище описане, було прийнято рішення що при розробці додатку буде використано графічний інтерфейс користувача.

Інтерфейс користувача складається з декількох вікон. Першим іде вікно авторизації, що зустрічає користувача. Його головною метою є авторизація користувача для початку роботи з додатком. Вікно авторизації включає в себе:

1. Область меню, що містить логотип та кнопки: згорнути, розгорнути та закрити;

2. Область авторизації, в якій знаходяться поля для вводу логіну та паролю. А також кнопки: Exit та Singin;

3. Робоча область, тобто екран самого додатку.

Графічна схема вікна входу зображена на рисунку 3.1.

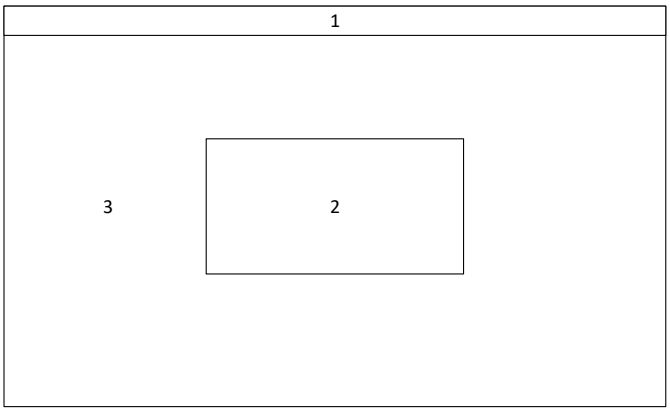

Рисунок 3.1 – Графічна схема інтерфейсу вікна входу

Після авторизації користувача відкривається головне вікно. В ньому знаходиться список таблиць з якими працює користувач. Головне вікно складається з:

1. Область меню, що містить логотип та кнопки: згорнути, розгорнути та закрити;
2. Список таблиць;
3. Кнопка «Delete»;
4. Кнопка «Add»;
5. Кнопка «ОК»;
6. Робоча область, тобто екран самого додатку.

Графічна схема головного вікна входу зображена на рисунку 3.2.

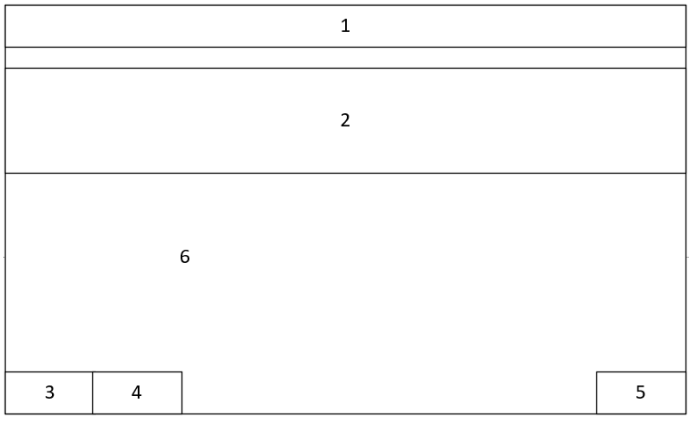

Рисунок 3.2 – Графічна схема головного вікна

Коли користувач обрав таблицю, з якою він буде працювати, відкривається вікно таблиці. В цьому вікні користувач може взаємодіяти з таблицею та її елементами. Вікно таблиці складається з:

1. Область меню, що містить кнопки: згорнути, розгорнути та закрити;
2. Таблицю та її елементи;
3. Кнопка «Delete»;
4. Кнопка «Add»;
5. Кнопка «QR-код»
6. Робоча область, тобто екран самого додатку.

Графічна схема вікна таблиці зображена на рисунку 3.3.

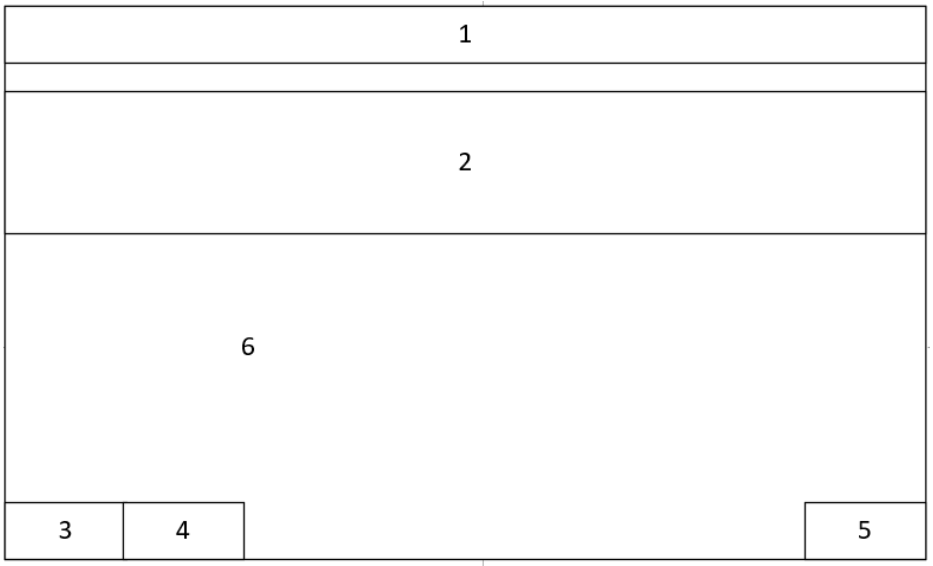

Рисунок 3.3 – Графічна схема вікна таблиці

Якщо користувач натиснув кнопку «QR-код» - перед ним відкриється нове вікно поверх попереднього, у якому і буде згенеровано QR-код. Вікно QR-коду складається з таких елементів як:

1. Кнопка «Close»;
2. Кнопка «Save»;
3. Робоча область де розміщено QR-код.

Графічна схема вікна представлена на рисунку 3.4.

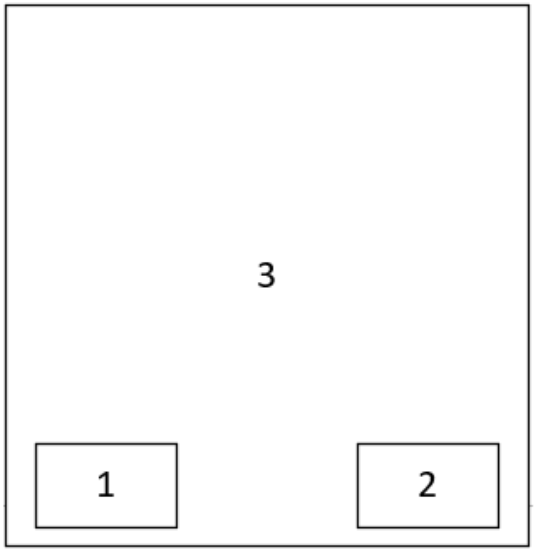

Рисунок 3.4 – Графічна схема вікна QR-коду

Також для працівників складу було розроблено додаток сканер для мобільного телефону, що є сумісним з додатком «Auto-Storage». Додаток сканер містить такі елементи інтерфейсу як:

1. Системна панель телефону;
2. Кнопка «Сканувати»;
3. Віконце камери для сканування;
4. Робоча область, тобто екран додатку.

Графічна схема додатку сканеру показана на рисунку 3.5.

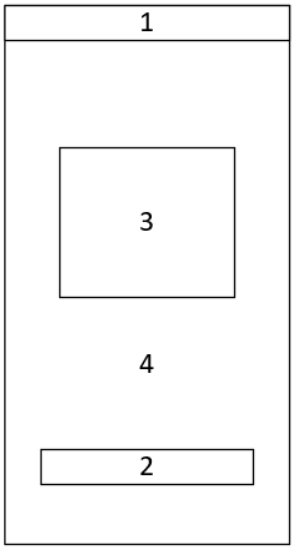

Рисунок 3.5 – Графічна схема додатку сканера

Після того як було відсканово QR-код з’являється вікно, що відображує дані які він зберігав. Вікно з даними має такі елементи:

1. Системна панель телефону;
2. Кнопка «Закрити»;
3. Дані що містив QR-код;
4. Робоча область, тобто екран додатку.

Графічна схема вікна даних QR-коду представлена на рисунку 3.6

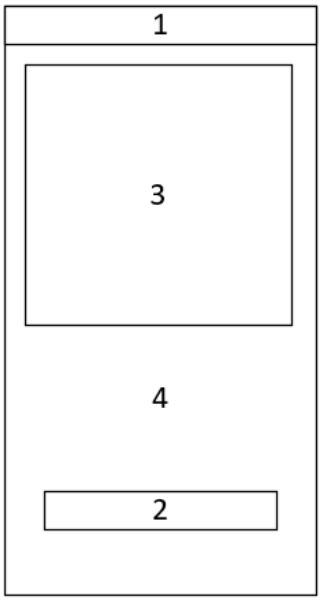

Рисунок 3.6 – Графічна схема вікна даних QR-коду

Отже, в даному розділі було розроблено графічну структуру інтерфейсу для створюваного додатку. Було розглянуто та детально описано структурну схему та призначення деяких елементів графічного інтерфейсу користувача.

3.2 Кольорова гама додатку

Вибір кольорової гами для додатку є ключовим елементом дизайну, оскільки кольори можуть впливати на емоційний стан користувача, визначати легкість використання і створювати враження професіоналізму. Нижче наведено розгорнутий аналіз важливості вибору кольорової гами для додатку:

1. Емоційний вплив. Кольори викликають емоції, і важливо враховувати, які саме асоціації вони можуть викликати у користувачів. Наприклад, теплі

кольори, такі як помаранчевий, можуть створювати позитивні та енергійні враження;

2. Узгодженість з ідентичністю бренду. Кольорова гама повинна відображати ідентичність бренду і підсилювати його цінності. Вибрані кольори повинні бути узгодженими з логотипом та загальною ідентифікацією бренду;

3. Виділення ключових елементів. Використання різних кольорів може допомогти виділити ключові елементи інтерфейсу, такі як кнопки, посилання або важливі повідомлення. Це полегшує орієнтацію користувача і сприяє легшій взаємодії з додатком;

4. Читабельність та доступність. Кольори повинні забезпечувати достатній контраст між фоном та текстом, щоб забезпечити читабельність. Це особливо важливо для людей з обмеженими можливостями або тих, які використовують додаток в умовах обмеженого освітлення;

5. Навігаційна логіка. Кольорова гама може використовуватися для визначення структури додатку та навігаційної логіки. Наприклад, використання специфічних кольорів для певних розділів може полегшити користувачам розуміння інтерфейсу;

6. Відображення контенту. Кольори повинні підтримувати тип контенту, який вони супроводжують. Наприклад, для додатків, які використовують багато мультимедійного контенту, важливо вибрати кольори, які не конфліктують з візуальною інформацією;

7. Консистентність між різними платформами. Якщо додаток доступний на різних платформах, кольорова гама повинна бути консистентною, щоб створити єдиний брендований образ.

Для додатку на колірному колі Іттена було обрано білий та помаранчевий кольори. Колірне коло Іттена представлено на рисунку 3.7.

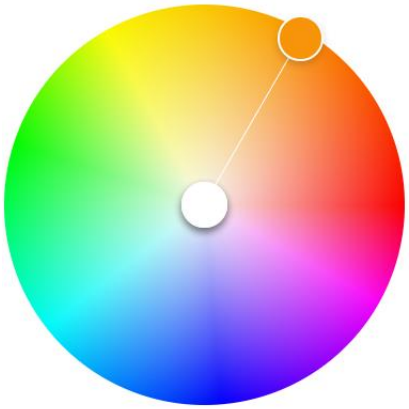

Рисунок 3.7 – Колірне коло Іттена

Розглянемо обрані кольори більш детально.

Білий:

• Чистота і простота. Білий - це колір чистоти і простоти, який може зробити інтерфейс додатку більш зрозумілим та легким для використання;

• Контраст і виділення. Білий використовується для створення контрасту з іншими кольорами. Він дозволяє виділяти текст, зображення та інші елементи інтерфейсу, роблячи їх більш помітними;

• Мінімалізм і сучасність. Білий колір часто асоціюється з мінімалізмом і сучасністю. Його використання може створити враження чистоти і актуальності додатку.

Помаранчевий:

• Енергія та позитивний настрій. Помаранчевий часто асоціюється з енергією, веселим настроєм та позитивом. Використання цього кольору може надати вашому додатку динаміку і заохочувати користувачів до активності;

• Залучення уваги. Помаранчевий - яскравий та виразний колір, який добре привертає увагу. Це може бути корисно для виголошення важливої інформації, визначення кнопок або важливих елементів інтерфейсу;

• Асоціації з інноваціями. Деякі компанії використовують помаранчевий для позначення інновацій і технологічного прогресу. Це може бути важливо, якщо ваш додаток створений для сучасних технологій або новаторських рішень.

Загальний вибір помаранчевого та білого є вдалим, оскільки ці кольори доповнюють один одного, створюючи яскравий та зрозумілий дизайн.

Таким чином було проведено аналіз критеріїв для вибору кольорової гами та основні кольори додатку.

3.3 Проектування бази даних

База даних (БД) - це структурована колекція даних, яка зберігається та організована для ефективного доступу, оновлення та аналізу. Бази даних використовуються для зберігання великого обсягу інформації, такої як тексти, числа, зображення, відео, аудіо та інші типи даних, які структуровані у вигляді таблиць, файлів, документів чи інших форматів.

Бази даних використовуються у різних галузях, включаючи бізнес, науку, освіту, організації, уряд та багато інших сфер. Вони дозволяють організаціям ефективно зберігати, керувати та аналізувати дані, що є критичними для прийняття рішень та виконання різних завдань.

Бази даних можуть бути реляційними (такі, де дані організовані у вигляді таблиць і взаємозв'язані за допомогою ключів) або нереляційними (такі, де дані можуть зберігатися у більш складних структурах, таких як дерева або графи). Вони також можуть бути централізованими або розподіленими, залежно від того, як обробляється доступ до даних.

Бази даних грають важливу роль у сучасному світі, де обробка та аналіз даних стає все більш важливою для багатьох аспектів життя та бізнесу [18].

Загальним способом представлення логічної моделі БД є побудова ER-діаграм (Entity-Relationship — сутність-зв'язок). В цій моделі сутність визначається як дискретний об'єкт, для якого зберігаються елементи даних, а зв'язок описує відношення між двома об'єктами.

Побудуємо ER-діаграму для нашої бази даних.

ER-діаграма для нашої бази даних зображена на рисунку 3.8.

У діаграмі присутні сутності: «Працівник», «Накладна», «Товар», «Склад», «Товар на складі», «Накладна».

Опишемо дані сутності:

– Працівник – це сутність уособлює оператора системи, який буде взаємодіяти з її функціями.

– Накладна – це сутність, що являє собою основний документ при транспортуванні вантажів, який регулює відносини між перевізником, відправником та одержувачем вантажу; оформляє та засвідчує договір перевезення вантажу.

– Накладна – це сутність того самого документу накладної, але вона містить у собі інформацію про кількість та ціну за одиницю товару.

– Товар – це сутність, що представляє собою будь-який товар що може бути придбано або переміщено на склад.

– Склад – це сутність, яка уособлює собою будь-яке приміщення що було позначено як складське.

– Товар на складі – це сутність, що являє собою будь-який товар що може бути розміщений на складі.

Кожна з цих сутностей утворює зв’язки між собою. Давайте розглянемо їх більш детально:

– «Працівник-Накладна» – ця пара утворює між собою зв’язок «один до багатьох», так як у одного працівника може бути декілька накладних.

– «Накладна-Накладна» – дана пара формує зв’язок «один до багатьох», тому що у одній накладній може бути перелічено декілька видів товарів з їх ціною та кількістю.

– «Товар-Накладна» – ця пара створює між собою зв’язок «один до багатьох», через те що один і той самий тип товару може бути вказаний у різних накладних.

– «Товар-Товар на складі» – дана пара утворює зв’язок «один до багатьох», через те що один і той самий тип товару може міститися на декількох складах.

– «Склад-Товар на складі» – ця пара формує зв’язок «один до багатьох», так як на одному складі може міститись декілька одиниць товару.

– «Склад-Накладна» – ця пара створює між собою зв’язок «один до

багатьох», тому що на один склад може бути виписано декілька накладних.

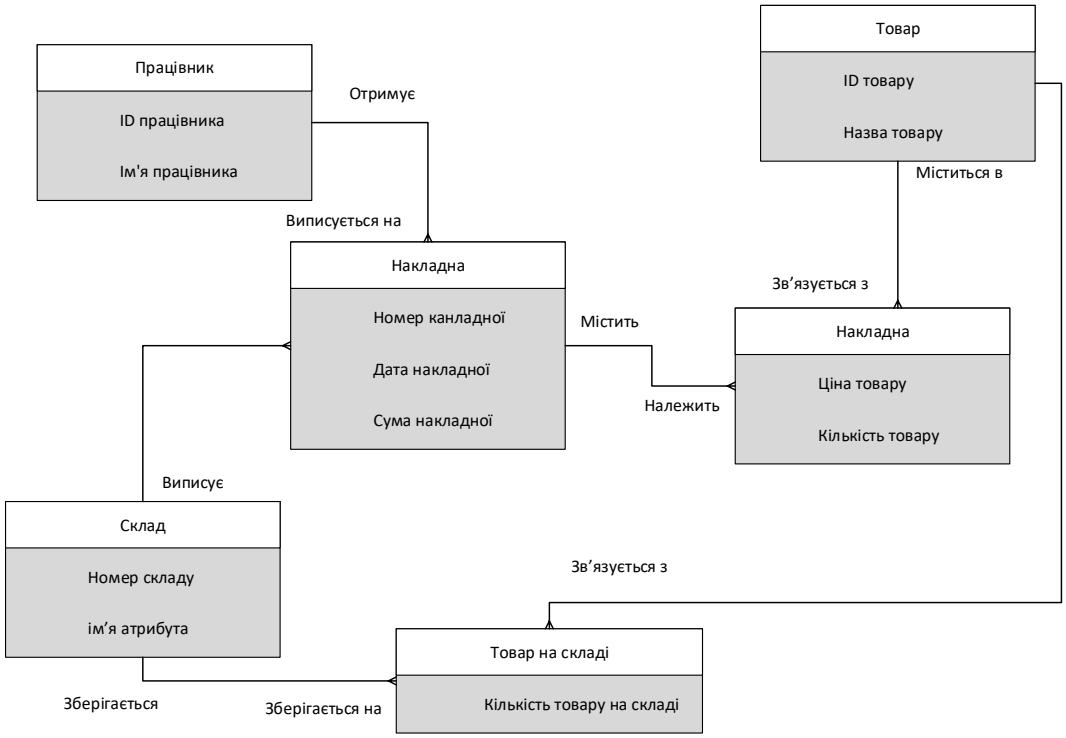

Рисунок 3.8 – ER-діаграма бази даних додатку

Таким чином, в даному підрозділі було розглянуто та розроблено ER-діаграму для бази даних додатку.

3.4 Варіантний аналіз і обґрунтування вибору мови програмування

При виборі мови програмування для розробки додатку, було розглянуто такі мови: C++, Java, C# та Python.

C++ – строго типізована мова програмування, що виникла як покращення мови C у 1979 році. Не дивлячись на те, що мова підтримує різні парадигми програмування, найбільше уваги приділяють саме підтримці об'єктно-орієнтованого програмування [19].

Основні переваги мови C++:

– швидкодія – компілятори для C++ за роки розвитку навчились робити велику кількість низькорівневих оптимізацій, що разом з іншим можливостями цієї мови дає можливість створювати одні з найбільш швидких додатків;

– керованість – ручний контроль над використанням та розподілом пам’яті дозволяє ще більше оптимізувати ПЗ, особливо при розробці для малопотужних пристроїв;

– сумісність з C – при розробці додатків на C++ можна застосовувати й бібліотеки від C, що значно розширює можливості розробника, особливо при роботі з низькорівневими процесами, драйверами, мультимедійними додатками тощо.

Основні недоліки C++:

– безпека роботи з пам’яттю – ручний контроль дає багато можливостей, але так само легко дозволяє створити додаткові проблеми в додатку через необережність або незнання окремих аспектів мови програмування, а «розумні» вказівники з більш нових стандартів лише частково вирішують це питання;

– складність – деякі низькорівневі можливості мови програмування, що C++ отримав від C, є складними для вивчення та застосування, а стандартна бібліотека хоч і є обширною, але менш комфортна для використання в порівнянні з іншими мовами;

– неоднорідність екосистеми – C++ в першу чергу є стандартом, який реалізовують різні компанії та спільноти у вигляді своїх власних компіляторів, тому можуть існувати деякі відмінності в роботі в різних середовищах; також через складність стандартизації відсутні деякі популярні інструменти розробників, як-от менеджери пакетів та залежностей – існують лише сторонні інструменти, які не мають повноцінної позиції на ринку та достатнього поширення в порівнянні з іншими офіційними застосунками.

Java – об'єктно-орієнтована мова програмування, випущена 1995 року компанією «Sun Microsystems» як основний компонент платформи Java. З 2009 року мовою займається компанія «Oracle», яка того року придбала «Sun Microsystems». В офіційній реалізації Java-програми компілюються у байт-код, який при виконанні інтерпретується віртуальною машиною для конкретної платформи [20].

Переваги Java:

– Портативність – Java є портативною мовою, що означає, що програми, написані на Java, можуть запускатися на будь-якому пристрої, який має відповідне виконавче середовище Java (JRE). Це робить Java ідеальною для розробки крос-платформених додатків.

– Об'єктно-орієнтована мова – Java побудована на принципах об'єктно-орієнтованого програмування (ООП), що допомагає створювати структурований та модульний код, покращує його підтримку та розширюється.

– Безпека – Java має вбудовану систему безпеки, яка дозволяє запускати код в безпечному віртуальному середовищі. Це захищає комп'ютери від можливих загроз та вразливостей.

– Збірка сміття – Java використовує автоматичну систему зборки сміття, що дозволяє видалити невикористані об'єкти та пам'ять, що використовується іншими частинами програми.

– Велика спільнота розробників – Java має велику та активну спільноту розробників, яка підтримує та розвиває мову. Це означає, що доступна багата база знань та бібліотек для розв'язання різних завдань.

Недоліки Java:

– Потужність ресурсів – Java може вимагати більшого обсягу пам'яті та потужності процесора порівняно з іншими мовами програмування, що може призвести до менш ефективного використання обладнання.

– Повільність – у деяких випадках Java може бути повільною, особливо у порівнянні з мовами програмування з низькорівневим доступом до апаратного забезпечення.

– Складність інтерфейсів – створення графічних інтерфейсів користувача у Java може бути більш складним у порівнянні з іншими мовами.

– Високий рівень абстракції – вищий рівень абстракції в Java може призвести до меншої ефективності в операціях, де необхідний прямий доступ до апаратного забезпечення.

– Відомі проблеми безпеки – незважаючи на вбудовану систему безпеки, Java була об'єктом відомих вразливостей у минулому.

Мова програмування C# була створена Microsoft у 2000 році в якості конкурента для Java та C++. Маючи подібний синтаксис та ряд зручних можливостей для розробників, як-от автоматичний менеджмент пам'яті та платформу .NET Framework, вона швидко отримала популярність на ОС Windows, а значно пізніше і на інших платформах [21].

Основні переваги мови C#:

– екосистема – сучасна .NET екосистема надає можливість розробляти практично будь-які додатки без використання сторонніх бібліотек, майже все необхідне є в складі офіційних фреймворків;

– баланс продуктивності – додатки на C# рідко можуть досягти ефективності програм на C++, але дружелюбність мови до розробників створює хороше співвідношення між продуктивністю створюваного ПЗ та довжиною циклу розробки;

– підтримка – C# активно розвивається Microsoft та іншими компанія-користувачами, тому регулярні оновлення виходять як для самої мови програмування, так і для її офіційних інструментів, як-от Visual Studio; на відміну від C++ тут відсутній процес стандартизації мови і довгий цикл затвердження змін.

Недоліки C#:

– орієнтація на Windows – довгий час C# та .NET розроблялися під платформи від Microsoft; розробка на інші ОС та пристрої була хоч і можливою, але дуже обмеженою; нові версії .NET орієнтовані на кросплатформенну розробку, але не всі елементи екосистеми зараз повноцінно оновлені для цього, а також залишається велика кількість вже існуючого ПЗ, фреймворків та бібліотек на старих версіях .NET Framework;

– обмеженість сфер застосування – C# дуже добре підходить для створення десктопних та серверних додатків, але погано орієнтований на розробку для вбудованих систем або низькорівневого ПЗ;

– середовище виконання – в C# та ряді інших мов програмування від Microsoft застосовується віртуальна машина Common Language Runtime (CLR),

тому кінцевий код додатку компілюється в проміжний байт-код для виконання на CLR, а не машинні інструкції; така архітектура спрощує запуск додатків на інших платформах та ОС, але може поставити додаткові обмеження по продуктивності й ефективності роботи в окремих сценаріях.

Python отримав свою популярність не через швидкодію або ефективність роботи з пам'яттю – в цих сферах ця мова програмування поступається іншим через інтерпретованість та динамічну типізацію. Код на Python більш простий для сприйняття в порівнянні з С-подібними мовами, а велика кількість синтаксичного цукру та обширна стандартна бібліотека роблять Python зручним у використанні. Гнучкість та доступність цієї мови призвели до її поширення в багатьох сферах – зараз Python використовують навіть для складних математичних моделей або штучного інтелекту [22].

Переваги Python:

– простота – чистий та зрозумілий синтаксис дозволяють використовувати цю мову програмування як розробникам, так і спеціалістам з інших сфер, а можливість інтеграції з іншими мовами дозволяє використовувати практично будь-який інструмент в такому форматі, що призвело до поширення Python за межі звичайних задач по розробці додатків;

– універсальність – обширна стандартна бібліотека Python та велика кількість сторонніх програмних компонентів дозволяють використовувати цю мову для веб-розробки, аналізу даних, машинного навчання або штучного інтелекту, наукових досліджень, автоматизації задач тощо; Python погано підходить лише для задач, що вимагають низькорівневого доступу або високої швидкодії і ефективного використання пам'яті;

– швидкий розвиток – Python розробляється та підтримується спільнотою, тому ця мова програмування регулярно отримує нові версії та виправлення помилок, а також відкриті дані дозволяють іншим командам створювати свої власні інтерпретатори Python з додатковими можливостями.

Основні недоліки Python:

– швидкодія – Python є інтерпретованою мовою програмування і значно поступається в швидкодії C++ та C#, а динамічна типізація та автоматичне керування пам’яттю помітно збільшують використання RAM;

– багатопотоковість – Python застосовує глобальне блокування інтерпретатора (англ. Global Interpreter Lock, GIL) для обмеження доступу окремих виконуваних потоків до інтерпретатора мови, що помітно ускладнює розробку багатопотокових програм та зменшує їх швидкодію;

– портативність – для виконання Python-скриптів необхідно попередньо встановити інтерпретатор та в деяких випадках ще завантажити сторонні бібліотеки, якщо такі застосовувались; цей крок необхідний як розробникам, так і кінцевим користувачам ПЗ, тому розповсюдження результуючого програмного продукту є більш складним процесом.

Для проведення порівняльного аналізу мов програмування було створено таблицю 3.1 в якій було порівняно переваги вище наведених мов.

Таблиця 3.1 – Порівняння мов програмування

Параметр	C++	Java	C#	Python
Швидкість виконання	1	1	1	0
Кросплатформеність	1	1	0.5	0.5
Швидкість макетування та розробки	0	0.5	0.5	1
Динамічна типізація	0	1	0,5	1
Розвиток екосистеми мови	1	1	0.5	1
Сумарний коефіцієнт	3	4.5	3	3.5

Розглянемо більш детально ці критерії, особливості оцінювання та виставлені бали.

Швидкість виконання – усі мови програмування, окрім Python, є компільованими. Додатки, розроблені на C++, C# та Java можуть мати різну

швидкодію через особливості роботи з пам’яттю, але вони все-одно будуть більш ефективними за інтерпретовані скрипти на Python.

Кросплатформеність – цей критерій включає в себе простоту розробки та збірки проектів на різних платформах. З цього боку гірше себе показує C# через потребу встановлення CLR та особливості роботи .NET платформи на сторонніх ОС, а також Python через проблеми з залежностями, що можуть використовувати попередньо скомпільовані компоненти.

Швидкість макетування та розробки – для сучасних комерційних додатків цей параметр може бути особливо важливим, адже дозволяє зробити цикл розробки помітно коротше й раніше віддати клієнту MPV (англ. Minimal Viable Product). Python має більш простий синтаксис, тоді як C++ складніший у використанні й вимагає додаткових знань для коректного управління пам’яттю. C# та Java займають проміжне місце між цими мовами.

Динамічна типізація - динамічна типізація та її реалізація в різних мовах програмування можуть значно впливати на спосіб розробки програм та їхню продуктивність. C++ використовує статичну типізацію за замовчуванням. C# використовує статичну типізацію за замовчуванням, але надає можливість використовувати динамічну типізацію. Java має строго динамічну типізацію, де тип об'єкта визначається в процесі виконання програми. Python має динамічну типізацію, і типи об'єктів визначаються під час виконання.

Розвиток екосистеми мови – C++, Python та Java мають широкий вибір IDE та інших інструментів розробки, а C# частково обмежена в деяких категоріях. C# довгий час була орієнтована лише на розробку для платформ від Microsoft і має якісні офіційні інструменти, як-от Visual Studio, але екосистема для інших ОС почала швидко розвиватися лише після поширення кросплатформних версій .NET.

Згідно таблиці мова програмування Java має деяку перевагу над іншими проаналізованими мовами програмування, такими як C++, C# та Python. Мова Java задовольняє всі необхідні вимоги для успішного створення програмної

реалізації програмного забезпечення тому для створення додатку автоматизації складського обліку було обрано саме її.

Таким чином, в даному підрозділі було розглянуто переваги й недоліки мов програмування C++, C#, Python та Java. Наведено короткі відомості про кожну мову, сформовано таблицю з порівнянням на основі ряду критеріїв та виставлені оцінки. За результатами порівняння мову програмування Java було обрано для розробки програмного продукту.

3.5 Вибір середовища розробки

Інтегроване середовище розробки (Integrated Development Environment, IDE) представляє собою програмне забезпечення, спрямоване на полегшення та оптимізацію робочого процесу розробників. Це централізоване середовище, що надає розробникам різноманітні інструменти для ефективної розробки, тестування та налагодження коду їхніх програм. Зазвичай, воно об'єднує в собі графічний редактор коду, компілятор або інтерпретатор, засоби для налагодження, а також різноманітні інтеграції з системами керування версіями. Інтегроване середовище розробки надає можливість зручно взаємодіяти з усіма цими інструментами через єдиний інтуїтивний графічний інтерфейс, сприяючи таким чином покращенню продуктивності розробників під час створення програмних продуктів.

Для розробки програмного додатку було обрано декілька популярних IDE а саме: IntelliJ IDEA, Visual Studio Code, Eclipse та NetBeans.

Visual Studio Code – розроблений Microsoft, відзначається своєю роллю як редактор коду, а не повноцінним інтегрованим середовищем розробки (IDE). Його базовий функціонал може бути меншим порівняно з деякими іншими IDE, але завдяки відкритому вихідному коду та активній спільноті розробників, VS Code стає ефективним інструментом для роботи з кодом [23].

Однією з ключових переваг Visual Studio Code є його швидкодія та розширюваність. Багатий каталог плагінів дозволяє налаштовувати редактор під конкретні потреби розробника. Це робить VS Code дуже гнучким та адаптованим

до різних вимог розробки. Приклад графічного інтерфейсу наведено на рисунку 3.9.

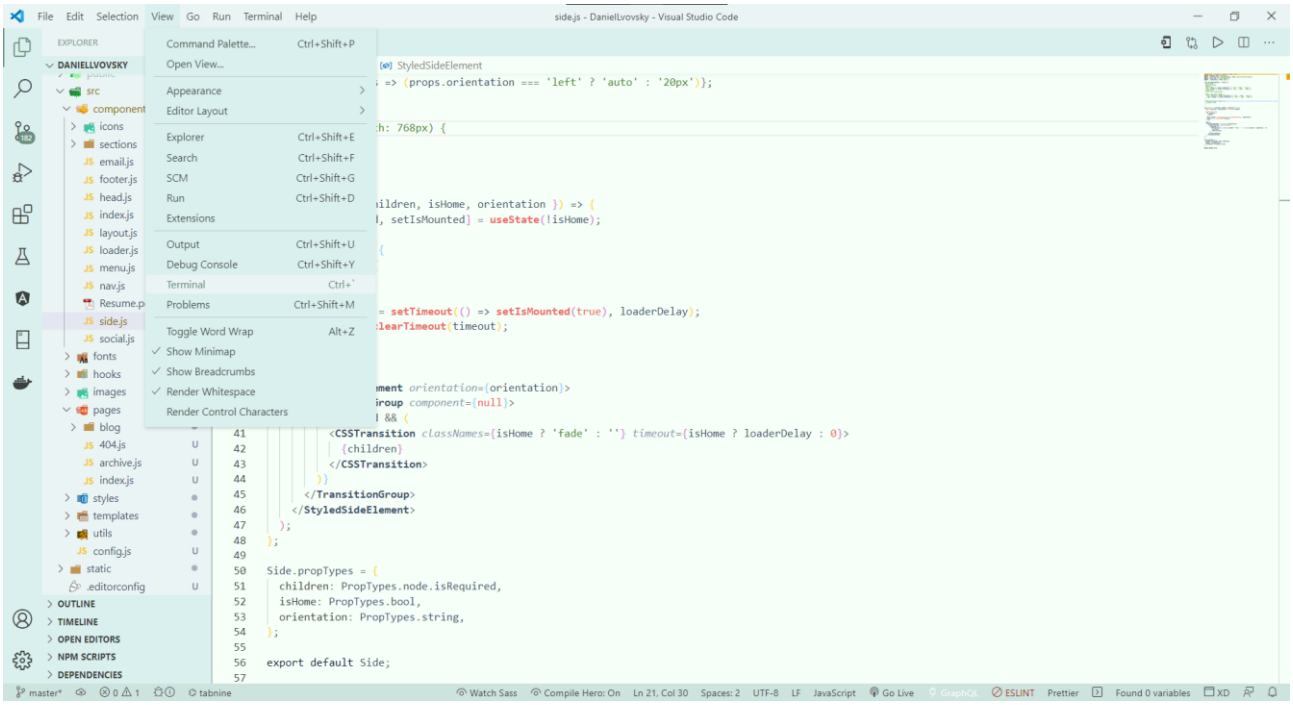

Рисунок 3.9 – Приклад інтерфейсу Visual Studio Code

Однак, навіть з усім своїм функціоналом, Visual Studio Code все ще може відрізнятися від повноцінних IDE. Його менш розвинуті системи рефакторингу та збірки проектів можуть стати основними недоліками, особливо для великих та складних проектів. Тим не менш, для тих, хто цінує швидкість, розширюваність та безкоштовність, Visual Studio Code залишається популярним вибором для розробки з великою спільнотою користувачів та розробників плагінів.

IntelliJ IDEA – розроблена компанією JetBrains, є високофункціональним інтегрованим середовищем розробки (IDE). Вона пропонує різноманітні можливості як у безкоштовному виданні Community, так і у платному виданні Ultimate. IntelliJ IDEA підтримує не лише Java, але й інші мови програмування, такі як Kotlin, Groovy та Scala. Інструмент також вражає глибокою інтеграцією з популярними фреймворками та технологіями [24]. Приклад графічного інтерфейсу наведено на рисунку 3.10.

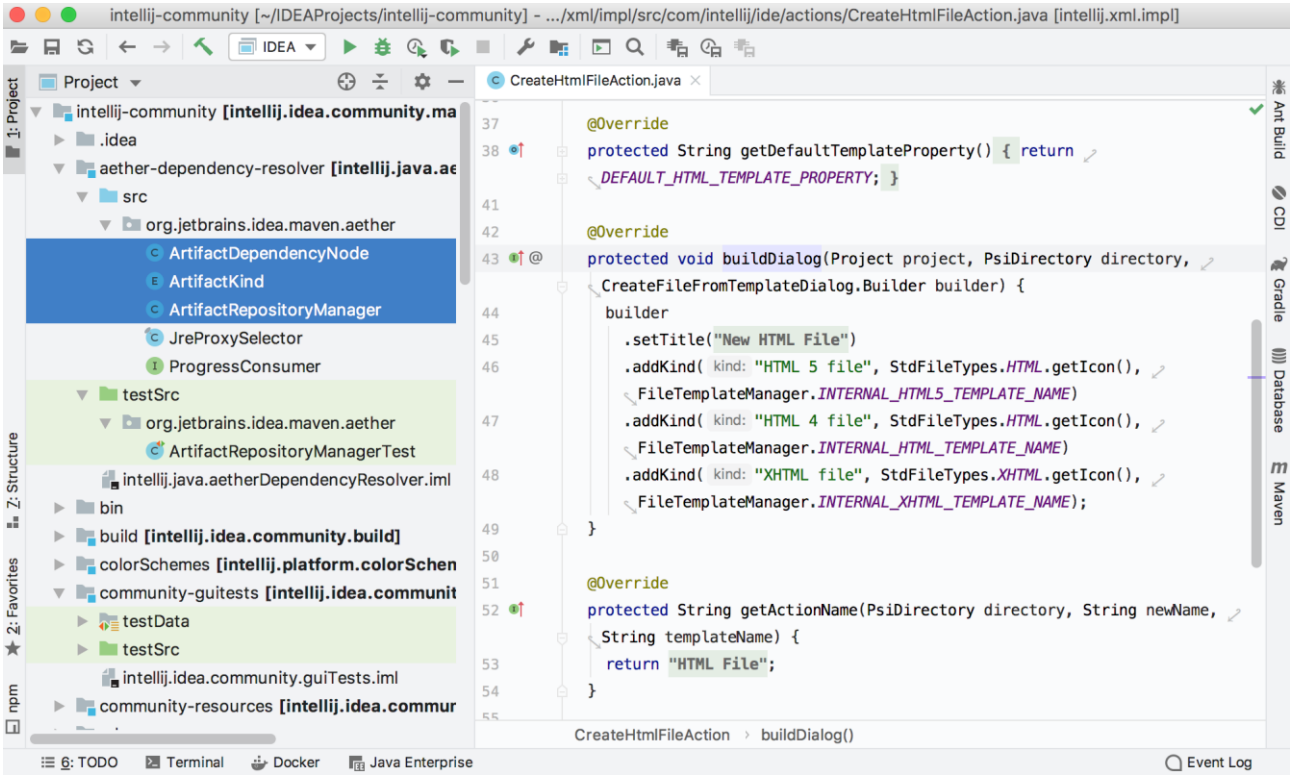

Рисунок 3.10 – Приклад інтерфейсу IntelliJ IDEA

Основними характеристиками IntelliJ IDEA є інтелектуальне автодоповнення коду, аналіз коду на етапі його написання, розширені засоби рефакторингу, інструменти для роботи з базами даних, вбудований термінал та інтеграція з системою контролю версій Git. Крім того, користувачі можуть скористатися різноманіттям плагінів для розширення функціональності згідно з їхніми потребами розробки.

IntelliJ IDEA визначається не лише своєю потужністю та продуктивністю, але й надійністю та зручністю використання, роблячи його однією з переваг для широкого кола розробників.

Eclipse – інтегроване середовище розробки (IDE) з відкритим вихідним кодом, є однією з найстаріших та найбільш розповсюджених платформ для розробки програмного забезпечення. Його тривалий термін існування та велика база користувачів свідчать про його популярність серед розробників [25].

Eclipse вирізняється тим, що підтримує роботу з багатьма мовами програмування. Це дає розробникам можливість легко комбінувати підтримку для різних мов та інші функціональні можливості за допомогою пакетів за

замовчуванням. Крім того, завдяки Eclipse Marketplace, користувачам відкривається безмежна свобода в конфігурації та розширенні можливостей IDE. Приклад графічного інтерфейсу наведено на рисунку 3.11.

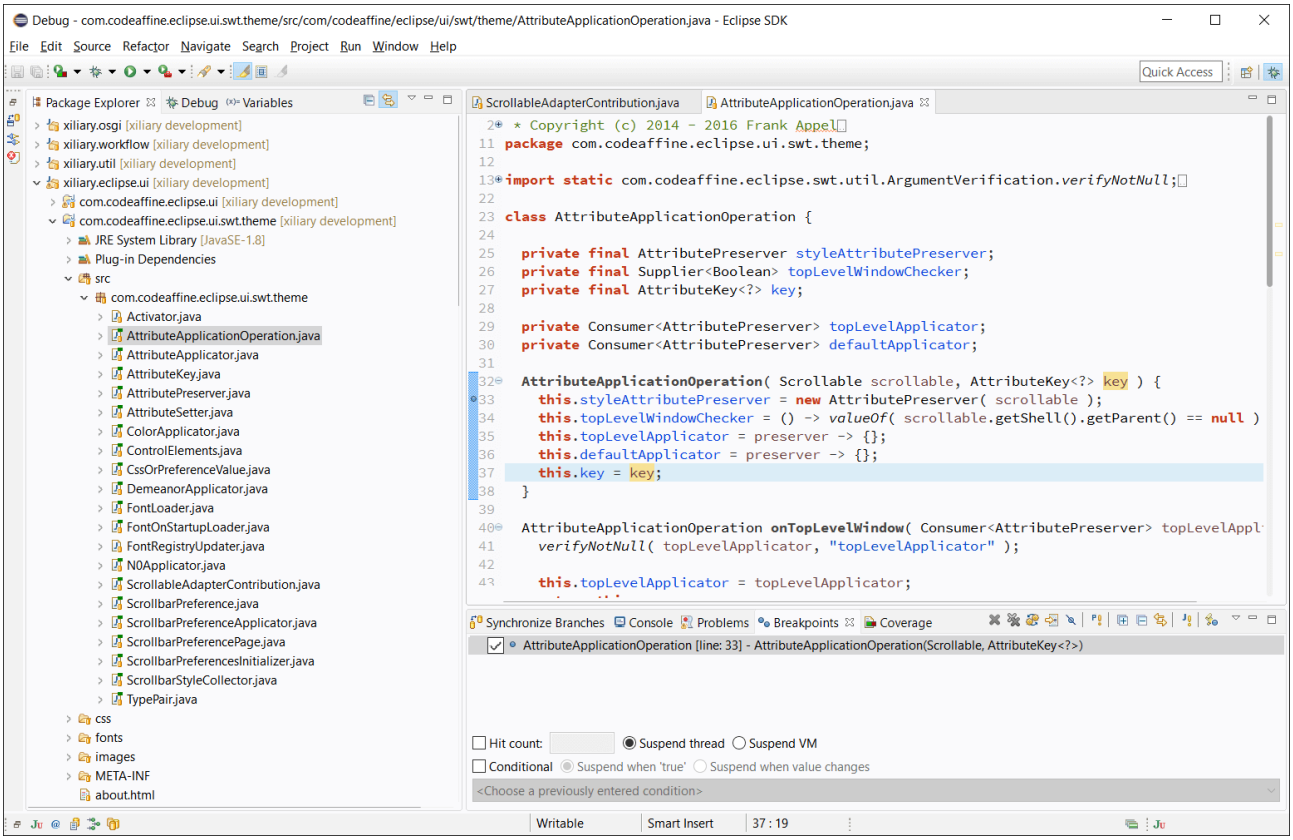

Рисунок 3.11 – Приклад інтерфейсу Eclipse

Основні особливості Eclipse включають автодоповнення коду, обширну документацію API, інтеграцію з системою контролю версій Git, а також величезний ринок плагінів для розширення функціональності. Серед інших важливих елементів варто відзначити середовище розробки плагінів (PDE), яке дозволяє розробникам розширювати можливості Eclipse за допомогою власних плагінів.

NetBeans, спочатку розроблений компанією Sun Microsystems та пізніше придбаний компанією Oracle, є однією з визнаних та популярних інтегрованих середовищ розробки (IDE) з відкритим вихідним кодом для розробників на мові програмування Java. Воно визначається своєю многомовністю, але водночас здатне надати вражаючий функціонал для розробки на Java [26].

Головні особливості NetBeans включають широкий набір шаблонів коду, інтелектуальні підказки щодо програмування, зручну навігацію по проекту, вбудовану підтримку системи управління залежностями Maven, потужний профайлер для аналізу продуктивності коду та інтегрований контроль версій. Приклад графічного інтерфейсу наведено на рисунку 3.12.

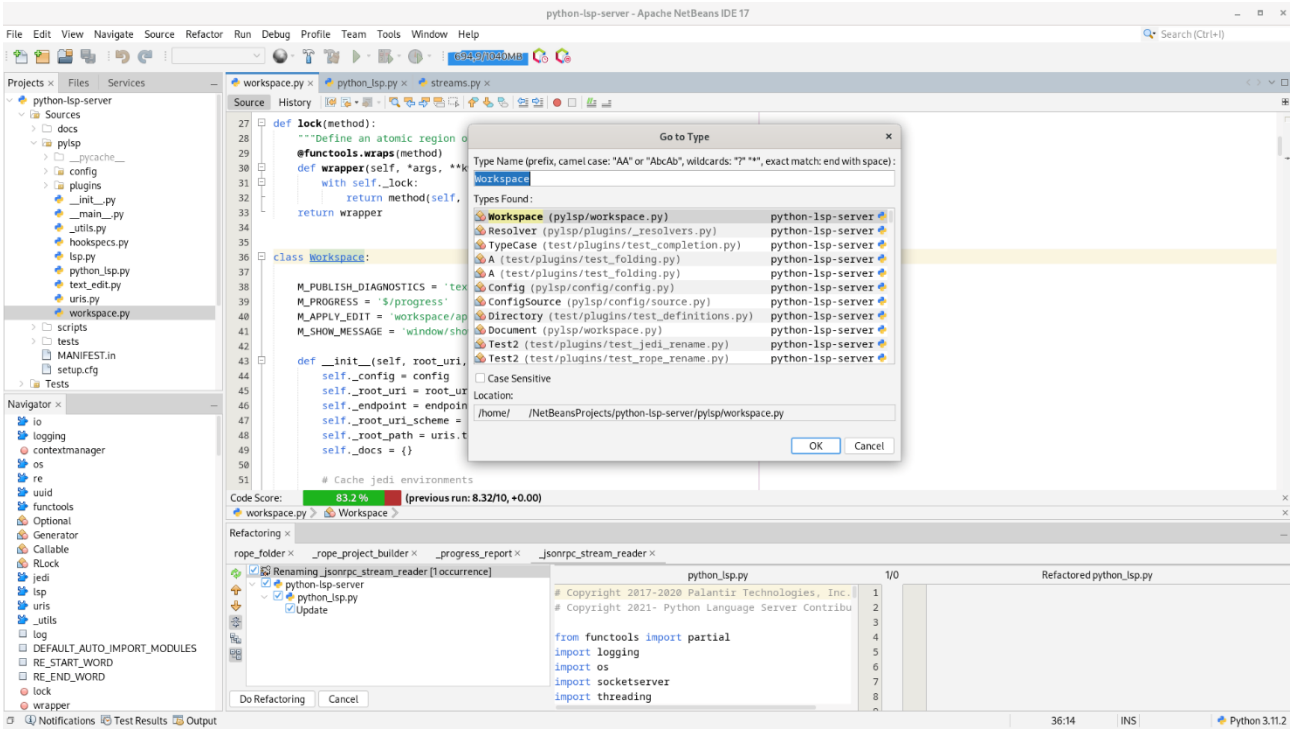

Рисунок 3.12 – Приклад інтерфейсу Eclipse

NetBeans відомий своєю простотою використання та інтуїтивним інтерфейсом, що робить його привабливим вибором для початківців і досвідчених розробників. Із підтримкою важливих інструментів розробки та надійною інтеграцією, NetBeans продовжує залишатися популярним вибором для розробників Java.

Щоб обрати середовище розробки, яке буде задовольняти всі вимоги при розробці програмного продукту, проведемо порівняльний аналіз представлених вище середовищ. Результати порівняння наведено в таблиці 3.2. Розглянемо більш детально категорії оцінювання.

Таблиця 3.2 – Порівняння засобів розробки

Критерій	VS Code	IntelliJ IDEA	Eclipse	NetBeans
Простота інтерфейсу	1	1	0,5	0,5
Розширення (Плагіни)	1	1	0	0,5
Підтримка ОС та платформ	0,5	1	0,5	1
Вартість (Безоплатність)	1	0,5	1	1
Швидкодія та використання ресурсів	0,5	1	0,5	0,5
Результат	4	4,5	2,5	3,5

Завдяки розвитку упродовж багатьох років VS Code та IntelliJ IDEA здобули менш завантажений та більш інтуїтивно зрозумілий інтерфейс. В той час як у Eclipse і NetBeans він все ще залишається перевантаженим.

Всі програмні продукти, окрім Eclipse, мають можливість розширення за допомогою плагінів. Для різних IDE ступінь розширення відрізняється.

За критерієм оцінки підтримки різних операційних системи проходять усі досліджувані продукти.

Усі програмні продукти мають безкоштовну форму розповсюдження. IntelliJ IDEA має також платну версію.

Eclipse, NetBeans та VS Code мають відносно меншу швидкодію порівняно із IntelliJ IDEA при використанні однакової кількості ресурсів.

Таким чином були розглянуті середовища розробки програмного забезпечення: VS Code, IntelliJ IDEA, Eclipse та NetBeans. Проведено аналіз цих чотирьох продуктів за рядом критеріїв, детально розглянуто виставлені оцінки. За результатами порівняння найбільш підходящим середовищем розробки для створення ПЗ для автоматизації складського обліку було обрано IntelliJ IDEA від JetBrains.

3.6 Розробка програмного модуля головно вікна та вікна авторизації

При реалізації головного вікна та вікна авторизації програми було використано бібліотеку для створення графічного інтерфейсу користувача Swing [27].

Бібліотека Swing є частиною Java API і призначена для створення графічного інтерфейсу користувача (GUI) в програмах, написаних на мові програмування Java. Вона надає набір компонентів і класів для створення різноманітних GUI-додатків, включаючи вікна, кнопки, тексти, таблиці і багато інших елементів.

Основні компоненти та можливості бібліотеки Swing включають:

1. Компоненти GUI: Swing надає розширений набір компонентів, таких як кнопки, тексти, таблиці, списки, вкладки і багато інших. Ці компоненти можуть бути використані для побудови різноманітних інтерфейсів;

2. Модель подій: Swing базується на моделі подій, що дозволяє реагувати на різні події, такі як натискання кнопок миші, введення клавіш і т.д. Це робить обробку подій більш гнучкою та зручною;

3. Модель подій для графічних компонентів: ця модель дозволяє відслідковувати події, пов'язані з малюванням і взаємодією з графічними компонентами;

4. Підтримка мінливості (pluggable look-and-feel): Swing дозволяє вибирати різні "look-and-feel" (вигляд та поведінка) для додатка, що дозволяє підлаштовувати вигляд програми під різні операційні системи чи власні дизайнерські вимоги;

5. Модель подій для обробки клавіш: Swing має розвинену модель подій для введення клавіш, що дозволяє зручно обробляти натискання клавіш, комбінації клавіш і події введення тексту;

6. Мережеві можливості: Swing включає підтримку розробки мережевих додатків, що дозволяє створювати програми, які взаємодіють з іншими додатками через мережу;

7. Підтримка перетягування та опускання (Drag and Drop): Swing дозволяє реалізувати функціональність перетягування та опускання між компонентами.

Використання бібліотеки Swing дозволяє розробникам створювати крос-платформені інтерфейси, що виглядають і ведуть себе однаково на різних операційних системах. Бібліотека входить до складу стандартного комплекту поставки Java, що робить її доступною для розробників без додаткового завантаження або встановлення.

Перед тим як користувач побачить головне вікно програми перед ним з’явиться вікно авторизації розміром 300 на 150 пікселів. Вікно авторизації має два поля для вводу даних про логін і пароль користувача та дві кнопки «Exit» і «Singin» для виходу або входу до програми.

Спочатку було підключено бібліотеку Swing та створено заголовок вікна і панель, де будуть розміщені всі інші елементи. За допомогою функції setTitle було призначено заголовок вікна, яким стало слово «Authorization». Після чого створюємо панель для розміщення на ній елементів використавши об’єкт JPanel. Приклад даного коду представлений на рисунку В.1 додатку В.

Наступним кроком було додано поля для вводу даних про логін та пароль користувача та кнопки «Exit» і «Singin». Використовуючи об’єкт JLabel створюємо два поля для вводу тексту з назвами «Username» та «Password» в які користувач пізніше вводитиме свої дані логіну та паролю. Також скориставшись об’єктом JButton створюються дві кнопки навігації «Exit» для виходу та «Singin» для входу у систему. Фрагмент даного коду показано на рисунку В.2 додатку В. Після було додано обробник подій для кнопок. Спочатку за допомогою функції addActionListener було додано обробники подій для двох кнопок для «Exit» вихід із системи, а для «Singin» відповідно вхід. Якщо авторизація була успішною, то використавши функцію showMessageDialog, з’являється невелике віконце із написом «Authorization Successful». Приклад даного коду зображений на рисунку В.3 додатку В.

Коли всі елементи було створено та налаштовано, через метод getContentPane вони були додані на панель, що була розміщена у вікні

авторизації. Також скориставшись методами setSize та setVisible було налаштовано розмір вікна та його видимість. Фрагмент даного коду представлено на рисунку В.4 додатку В.

Після того, як користувач авторизується, перед ним відкриється головне вікно програми. Для реалізації модуля також було використано бібліотеку Swing. По аналогії з вікном авторизації були створені заголовок вікна та панель для розміщення елементів. Використавши клас ImageIcon додаємо на головне вікно логотип програмного додатку. Приклад даного коду представлено на рисунку В.5 додатку В.

Наступним кроком було додано декілька кнопок для взаємодії із користувачем, а саме: «Delete», «Add» та «OK». За прикладом вікна авторизації за допомогою об’єкту JButton створюємо всі три кнопки та обробники для них скориставшись функцією addActionListener. Фрагмент даного коду зображено на рисунку В.6 додатку В.

Після того, як всі елементи було створено та налаштовано, вони були додані на панель, що була розміщена у головному вікні, яке було налаштовано способом подібним до вікна авторизації за допомогою методів setSize та setVisible. Приклад даного коду продемонстровано на рисунку В.7 додатку В.

Таким чином було розроблено модуль головного вікна та вікна авторизації.

3.7 Розробка програмного модуля для інтеграції бази даних

Наступним було створено модуль інтеграції бази даних у програмний додаток. Інтеграція бази даних у програму Java вимагає використання JDBC (Java Database Connectivity) [28].

Java Database Connectivity (JDBC) є стандартним інтерфейсом програмування для забезпечення з'єднання Java-додатків з реляційними базами даних. JDBC дозволяє вам взаємодіяти з базами даних, виконуючи SQL-запити, оновлюючи дані та отримуючи результати запитів. Також, JDBC забезпечує механізми для керування транзакціями та обробки винятків. Цей модуль було використано для підключення до бази даних MySQL і виконання запиту

SELECT. Але слід переконатись що драйвер JDBC встановлено. Інші бази даних (наприклад, PostgreSQL, Oracle, SQLite) вимагатимуть використання відповідних JDBC-драйверів і параметрів підключення.

Основні концепції та можливості JDBC включають:

1. З'єднання (Connection): JDBC дозволяє створювати з'єднання з базою даних за допомогою класу Connection. З'єднання можна встановити з різними видами баз даних, такими як MySQL, Oracle, PostgreSQL і багатьма іншими;

2. Використання SQL-запитів: JDBC дозволяє виконувати SQL-запити до бази даних. Це може бути вставка, оновлення, видалення або вибірка даних з таблиць;

3. Обробка результатів запитів: результати SQL-запитів можна обробляти і отримувати за допомогою об'єктів ResultSet. Це дозволяє отримувати дані з бази даних і подальшу їх обробку в Java-додатку;

4. Підготовлені заявки (Prepared Statements): JDBC підтримує використання підготовлених заявок, які можуть оптимізувати виконання повторюваних SQL-запитів та покращувати безпеку;

5. Транзакції: JDBC дозволяє керувати транзакціями в базі даних. Це включає початок, фіксацію та відкат транзакцій;

6. Метадані (Metadata): JDBC дозволяє отримувати інформацію про базу даних, таку як структура таблиць, типи даних і т. д., за допомогою об'єктів DatabaseMetaData та ResultSetMetaData;

7. Обробка винятків та помилок: JDBC надає засоби для обробки винятків, які можуть виникнути під час взаємодії з базою даних.

Використання JDBC є ключовим для розробки Java-додатків, що вимагають доступу до реляційних баз даних. Для взаємодії з конкретною базою даних, необхідно використовувати конкретний JDBC-драйвер, який забезпечить з'єднання з відповідною базою даних.

Спочатку було підготовано деякі змінні та об’єкти для роботи з базою даних. Змінні являють собою посилання на базу даних, логін та пароль.

Об’єктами значяться з’єднання, заява та результат. Приклад коду можна побачити на рисунку В.8 додатку В.

Наступним кроком було підключення драйверу JDBC для роботи з базою даних, так як це є обов’язковою умовою для початку роботи. Спершу було завантажено драйвер. Далі за допомогою об’єкта connection підключимося до бази даних. І останнім кроком було створення об’єкту statement для виконання запитів. Приклад коду підключення драйверу представлено на рисунку В.9 додатку В.

Після підключення драйверу було виконано запит SELECT у базі даних MySQL. У першу чергу за допомогою класу sqlQuery створюємо запит. Далі за допомогою об’єкта resultSet запит потрапляє на обробку. Після цього створюємо обробник для результату запиту. Він являє собою цикл while, що пропускає через себе значення «id» та «name» та виводить їх на екран. Якщо виводити вже нічого, то виконується процедура закриття ресурсів для завершення роботи. Приклад виконання запиту продемонстровано на рисунку В.10 додатку В.

Таким чином було розроблено програмний модуль для підключення бази даних до додатку.

3.8 Розробка програмного модуля для генерації QR-кодів

Для розробки модуля генерації QR-кодів було використано бібліотеку ZXing (Zebra Crossing) [29].

ZXing (Zebra Crossing) - це відкрита бібліотека для розпізнавання штрих-кодів (QR-кодів, штрих-кодів EAN, UPC і інших форматів) в різних зображеннях. Розроблена на мові програмування Java, ZXing також має порти для інших мов, включаючи JavaScript, Python, Objective-C, PHP і C++.

Основні можливості ZXing включають:

1. Підтримка різних форматів штрих-кодів: ZXing здатна розпізнавати різні формати штрих-кодів, такі як QR-коди, штрих-коди EAN, UPC, Code 128, Aztec, Data Matrix і багато інших;

2. Підтримка зчитування з різних джерел: ZXing може працювати з зображеннями, отриманими з відсканованих документів, фотографій, відеопотоків або навіть зображень, отриманих з веб-камер;

3. Крос-платформеність: основна версія ZXing розроблена на Java, що робить її крос-платформеною і можливою для використання на різних операційних системах;

4. Ліцензія Apache 2.0: ZXing розповсюджується під ліцензією Apache 2.0, що дозволяє використовувати бібліотеку у комерційних та вільних проектах, забезпечуючи при цьому відкритий доступ до вихідних кодів;

5. Розширення функціональності: крім основної функціональності розпізнавання штрих-кодів, ZXing також може генерувати QR-коди та інші формати штрих-кодів;

6. Активна спільнота користувачів та розробників: завдяки своїй популярності, ZXing має активну спільноту, що забезпечує підтримку, оновлення та розвиток бібліотеки.

ZXing використовується у різних проектах для розпізнавання та генерації штрих-кодів у мобільних додатках, веб-сервісах та інших системах, де необхідна робота з штрих-кодами.

Також для корекції помилок при генерації QR-кодів було використано алгоритм Reed-Solomon. Щоб встановити цю бібліотеку, необхідно включити в Maven або Gradle залежності у проекті.

Apache Maven - це інструмент для управління проектами та залежностями в рамках проектів, написаних на мові програмування Java (і не тільки). Maven забезпечує стандартизований спосіб будівництва проектів, управління залежностями, розгортання та документацію [30].

Основні концепції та можливості Apache Maven включають:

1. POM (Project Object Model): проект в Maven описується файлом POM, який є XML-документом і містить інформацію про проект, таку як його ідентифікатор, версія, залежності, плагіни, цілі збірки та інше;

2. Централізоване управління залежностями: Maven автоматично завантажує бібліотеки та інші залежності з централізованого репозиторію, що спрощує управління залежностями та забезпечує їх консистентність;

3. Стандартна структура проекту: Maven рекомендує стандартну структуру проекту, що спрощує взаємодію з іншими розробниками та автоматизує процеси збірки і розгортання;

4. Життєвий цикл збірки проекту: Maven надає стандартний життєвий цикл збірки, який включає такі етапи, як компіляція, тестування, пакування, розгортання та інші;

5. Плагіни: Maven може використовувати різноманітні плагіни для розширення його функціональності або виконання додаткових завдань в рамках проекту;

6. Система зберігання (репозиторій): Maven використовує репозиторії для зберігання залежностей та артефактів проектів, що дозволяє легко ділитися та використовувати код в спільноті розробників;

7. Інтеграція з IDE: Maven інтегрується з багатьма середовищами розробки (Integrated Development Environments - IDEs), що спрощує конфігурування та використання у процесі розробки.

Maven є популярним інструментом в екосистемі Java, і його використання спрощує автоматизацію збірки та управління залежностями для проектів різної складності.

Gradle - це інструмент для автоматизації збірки проектів та управління залежностями, який побудований на принципах конвенції над конфігурацією та використовує мову DSL (Domain-Specific Language) для опису збірки проектів. Gradle підтримує різні мови програмування та технології, хоча його основна популярність припадає на використання в екосистемі Java [31].

Основні концепції та можливості Gradle включають:

1. DSL для опису збірки: Gradle використовує свій власний DSL для опису збірки проекту, що робить конфігурацію більш зрозумілою та ефективною;

2. Засновано на конвенціях: Gradle використовує конвенції для визначення стандартної структури проекту та найбільш поширених налаштувань, що спрощує конфігурацію проекту та сприяє швидшої взаємодії з іншими розробниками;

3. Інкрементальна збірка: Gradle підтримує інкрементальну збірку, що дозволяє перезбирати лише ті частини проекту, які зазнали змін, що прискорює час розробки;

4. Мультипроектні збірки: Gradle дозволяє легко організовувати багатопроектні структури та керувати залежностями між ними;

5. Широкий вибір плагінів: Gradle має широкий вибір плагінів, що дозволяють легко інтегрувати різноманітні технології та фреймворки в проекти.

6. Інтеграція з іншими інструментами: Gradle добре інтегрується з іншими інструментами розробки та системами управління версіями;

7. Зовнішні конфігурації (External Configurations): Gradle дозволяє використовувати зовнішні файли конфігурацій (такі як Gradle Properties або YAML), щоб зручно визначати параметри збірки та налаштування.

Gradle забезпечує гнучкий та потужний механізм для автоматизації збірки проектів та розгортання додатків, роблячи його одним з популярних інструментів у світі розробки програмного забезпечення.

Приклад коду для підключення бібліотеки продемонстровано на рисунку В.11 додатку В.

Після того як були імпортовані всі необхідні залежності було створено клас QRCodeGenerator. У даному класі було створено змінну даних, яка буде кодуватися та змінну шляху, куди буде зберігатися згенерований QR-код. Також були створені два сповіщення – одне буде з’являтися при досягненні успіху, а інше якщо станеться невдача. Фрагмент даного коду представлено на рисунку В.12 додатку В.

Далі було створено приватну статичну змінну generateQRCode яка і буде відповідальна за генерацію QR-коду. У змінній int було вказано розмір

зображення, а у змінній fileType - тип файлу. Приклад даного коду зображено на рисунку В.13 додатку В.

Наступним кроком у змінній generateQRCode було створено цикл для генерації QR-коду, а також перевірки та корекції помилок. Спершу створимо функцію, яка буде відповідати за перевірку помилок при генерації QR-коду. Ініціюємо інтерфейс Map, складовими якого будуть EncodeHintType, в якості ключа та Object – значення і створюємо хеш-таблицю HashMap. Потім у функції hintMap визначаємо тип та рівень корекції, що буде задіяний.

Далі створюємо функцію QrCodeWriter яка і буде генерувати QR-код. У функції створюємо двовимірний бітовий масив BitMatrix у значення якого і вводимо данні для генерації. Після чого клас BufferedImage що за допомогою циклів for і створить необхідне зображення. Фрагмент коду зображено на рисунку В.14 додатку В.

Коли зображення створиться буде використано клас File, де буде вказаний шлях де воно зберігатиметься. Клас ImageIO буде містити параметри зображення. У випадку якщо генерація провалиться, то з’явиться сповіщення про помилку. Приклад даного коду представлено на рисунку В.15 додатку В.

Таким чином було розроблено модуль для генерації QR-коду з корекцією помилок за допомогою алгоритму Reed-Solomon.

3.9 Розробка програмного модуля для зчитування QR-кодів

Останнім було розроблено модуль для зчитування QR-кодів. Цей модуль було розроблено спеціально для працівників складів щоб вони могли дізнатися інформацію про товар на складі за допомогою мобільного телефону. Для сканування QR-коду також було використано бібліотеку ZXing, яка надає можливість розпізнавання QR-кодів та є кросплатформеною.

Спершу було проведено підготовку до читання QR-коду. Створено клас QRCodeScannerActivity який відповідає за роботу сканеру. Та за допомогою класу IntentIntegrator розблокуємо орієнтацію камери і запустимо сам сканер. Приклад коду показано на рисунку В.16 додатку В.

Далі було створено обробник результатів сканування QR-коду. Створимо клас onActivityResult та введемо параметри. Використавши клас IntentResult обробимо результат сканування та отримаємо два повідомлення або про невдачу, або про успішне завершення разом з відсканованими даними. Приклад коду зображено на рисунку В.17 додатку В.

Наступним кроком було використання медіального фільтру для покращення результатів читання QR-коду. Для чого було використано бібліотеку OpenCV [32].

OpenCV (Open Source Computer Vision Library) – це відкрита бібліотека комп'ютерного зору, яка надає інструменти для обробки зображень та комп'ютерного зору в реальному часі. Вона написана на C++ та має інтерфейси для мов програмування, таких як Python, Java, і C#. OpenCV має широкий спектр функціональності, що охоплює розпізнавання обличчя, виявлення об'єктів, обробку зображень, машинне навчання та багато іншого.

Основні можливості та концепції OpenCV включають:

1. Обробка та аналіз зображень: OpenCV дозволяє виконувати різні операції над зображеннями, такі як фільтрація, морфологічні операції, виправлення перспективи, обчислення градієнту та інші;

2. Виявлення та відстеження об'єктів: OpenCV має алгоритми для виявлення об'єктів на зображеннях, а також для відстеження їхнього руху в реальному часі;

3. Розпізнавання обличчя: OpenCV має функції для розпізнавання обличчя, включаючи визначення ключових точок, векторизацію обличчя, ідентифікацію та інші задачі обробки зображень з обличчями;

4. Машинне навчання: OpenCV включає модуль для машинного навчання, який дозволяє використовувати алгоритми класифікації, кластеризації та регресії;

5. Обробка відео та потокове відео: OpenCV дозволяє працювати з відеофайлами та відеопотоками, виконуючи аналіз та обробку кадрів в реальному часі;

6. Інтеграція з різними мовами програмування: OpenCV може бути використана з різними мовами програмування, зокрема C++, Python, Java та C#;

7. Оптимізація для вбудованих систем: OpenCV оптимізована для використання на різних платформах, включаючи вбудовані системи та мобільні пристрої.

OpenCV широко використовується в галузі комп'ютерного зору, робототехніки, медицини, автоматизації та інших сферах, де потрібна обробка зображень та аналіз відео.

Спочатку бібліотеку було активовано та підготовлено до роботи. Було створено клас CameraActivity, який буде відповідати за використання медіального фільтру. Приклад коду продемонстровано на рисунку В.18 додатку В.

Потім було підготовлено клас Mat який буде викликатися камерою для отримання зображень. Далі було створено клас Imgproc у який і було записано параметри для медіального фільтру. Наступним кроком було створено клас onCreate що відповідатиме за запуск OpenCV. При запуску OpenCV ініціалізує та налаштовує камеру для роботи. Фрагмент даного коду представлено на рисунку В.19 додатку В.

Після чого йде обробка результатів використання медіального фільтра по кожному кадру. Також викликається метод onResume який виведе результат із повідомленням про успіх або невдачу. Фрагмент даного коду продемонстровано на рисунку В.20 додатку В.

Таким чином було розроблено модуль для зчитування QR-коду та покращенням якості зображення завдяки використанню медіального фільтра.

3.10 Висновки

В третьому розділі було розроблено структуру графічного інтерфейсу додатку «Auto-storage», а також обрано кольорову гаму, основними кольорами якої стали білий та помаранчевий. Проведено проектування бази даних додатку.

Було проведено аналіз мов програмування C++, C#, Python та Java. Порівняння цих рішень за рядом критеріїв дозволило вибрати мову програмування Java для розробки програмного засобу «Auto-storage».

Також були розглянуті середовища розробки програмного забезпечення як от VS Code, IntelliJ IDEA, Eclipse та NetBeans. За результатами порівняльного аналізу найбільш підходящим середовищем розробки для створення ПЗ для автоматизації складського обліку було обрано IntelliJ IDEA від JetBrains.

Визначено функціонал модуля головного вікна та вікна авторизації та проведено аналіз процесу його розробки.

Визначено функціонал модуля для інтеграції бази даних та проведено аналіз процесу його розробки.

Визначено функціонал модуля для генерації QR-кодів та проведено аналіз процесу його розробки.

Визначено функціонал модуля для зчитування QR-кодів та проведено аналіз процесу його розробки.

4 ТЕСТУВАННЯ ПРОГРАМИ

4.1 Аналіз методів тестування програмного забезпечення

Тестування програмного забезпечення – це процес перевірки відповідності заявлених до продукту вимог до реально реалізованої функціональності. Тестування також включає як процес пошуку помилок та дефектів, так і випробування програмних складових з метою оцінки.

Методи чорної (Black Box) і білої (White Box) скриньки відносяться до динамічних технік тестування, тобто застосовуються в процесі безпосереднього виконання програми [33].

Тестування методом білої скриньки полягає у докладному дослідженні внутрішньої логіки і структури програми, тоді як метод чорної скриньки не вимагає знань про внутрішню структуру додатку. Процес цього методу ґрунтується на аналізі ключових аспектів системи.

Тож розглянемо ці методи детальніше. Тестування методом білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа потоку керування. Для тестування формуються тестові варіанти, в яких:

- Гарантується перевірка всіх незалежних маршрутів програми;
- Знаходяться гілки True, False для всіх логічних рішень;
- Виконуються всі цикли (у межах їхніх кордонів та діапазонів);
- Аналізується правильність внутрішніх структур даних.

Недоліки тестування білої скриньки:

- Кількість незалежних маршрутів може бути дуже велика;
- Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї;
- У програмі можуть бути пропущені деякі маршрути;
- Не можна виявити помилки, поява яких залежить від даних.

До переваг методу можна віднести:

- Кількість помилок мінімально в «центрі» і максимально на «периферії» програми;
- Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо;
- При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних);
- Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Тестування методом чорної скриньки розглядає програмне забезпечення як «чорну скриньку» – аналізуються тільки основні аспекти системи. Переваги даного методу:

- Ефективність для великого сегмента коду;
- Простота сприйняття;
- Перспектива користувача чітко відокремлена від перспективи розробника;
- Більш швидке створення тесту.

Іноді цю техніку називають поведінкове тестування або налагодження методом закритого ящика.

Метод чорної скриньки заснований на специфікаціях, документації, та описах інтерфейсу програмного забезпечення або системи. Крім того, допускається застосування формальних або неформальних моделей, що представляють очікувану поведінку ПЗ.

Таким чином розглянуто базові відомості про тестування ПЗ. Було розглянуто класифікацію видів тестування, для перевірки працездатності додатку обрано метод тестування «чорної скриньки».

4.2 Тестування розробленого програмного продукту

Тестування додатку «Auto-Storage» за методом «чорної скриньки» має на увазі перевірку коректності функціонування додатку при виконанні основних алгоритмів його використання та порівняння фактичного результату перевірки з очікуваним. Подібні алгоритми використання програмного додатку називаються тест-кейси. Для проведення тестування розробленого програмного додатку було розроблено наступні тест-кейси:

Тест-кейс №1 «Перевірка входу в систему»:

1) Відкрити додаток «Auto-Storage»;

2) У полях «Username» та «Password» вікна авторизації ввести дані облікового запису;

3) Натиснути кнопку «Singin».

Очікуваним результатом даного тест-кейсу є авторизація та перехід до головного вікна додатку. Результат виконання тест кейсу представлено на рисунку 4.1.

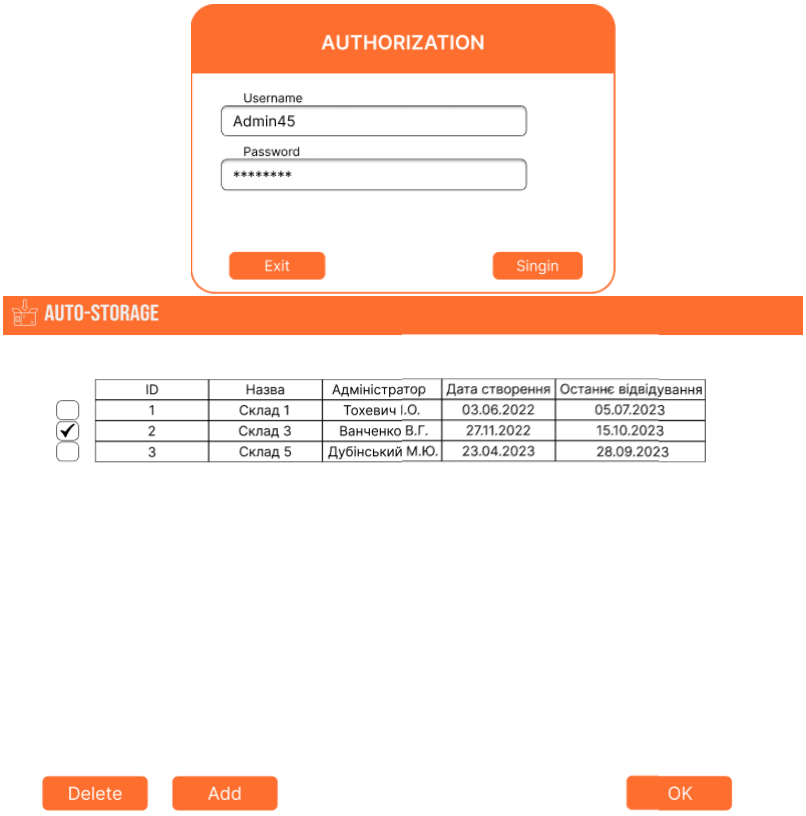

Рисунок 4.1 – Результат виконання тест-кейсу №1

Фактичний результат відповідає очікуваному, таким чином тест-кейс №1 успішно пройдено.

Тест-кейс №2 «Перевірка роботи кнопки «Add»»:

1) Обрати та натиснути копку «Add»;

2) Дочекатися появи діалогового вікна;

3) Додати нову таблицю.

Очікуваним результатом даного тест-кейсу є відкриття діалогового вікна для додання нової таблиці та її створення після вводу даних. Результат виконання тест кейсу показано на рисунку 4.2.

Рисунок 4.2 – Результат виконання тест-кейсу №2

Фактичний результат відповідає очікуваному, отже тест-кейс №2 успішно пройдено.

Тест-кейс №3 «Перевірка роботи кнопки «Delete»»:

1) Обрати та натиснути кнопку «Delete»;

2) Дочекатися появи діалогового вікна;

3) Видалити таблицю.

Очікуваним результатом даного тест-кейсу є відкриття діалогового вікна для видалення таблиці та її видалення після підтвердження. Результат виконання тест кейсу показано на рисунку 4.3.

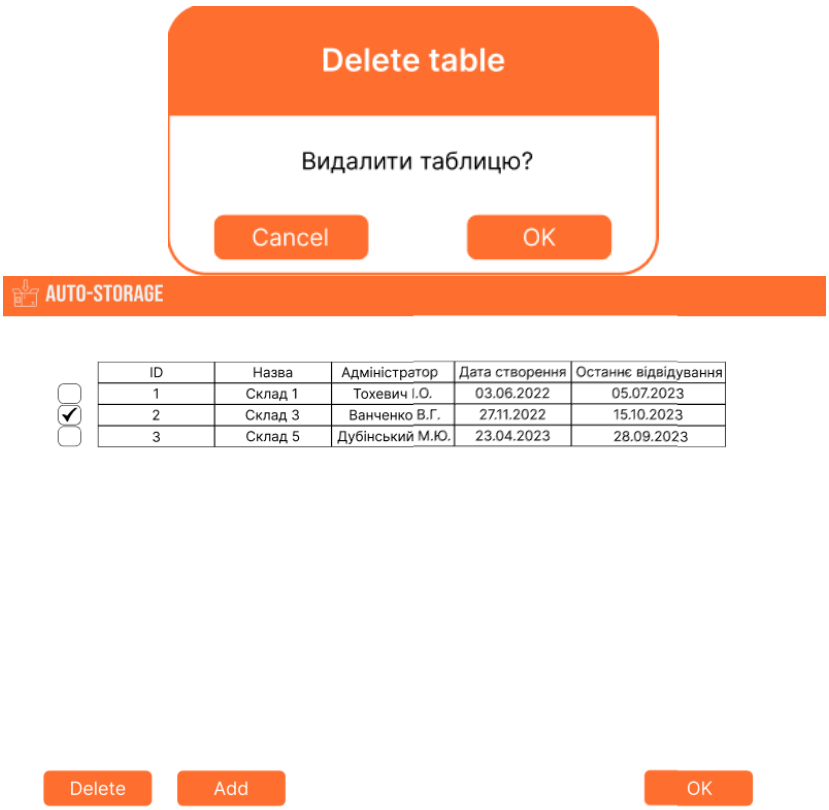

Рисунок 4.3 – Результат виконання тест-кейсу №3

Фактичний результат відповідає очікуваному, отже тест-кейс №3 успішно пройдено.

Тест-кейс №4 «Перевірка роботи кнопки «OK»»:

1) Обрати таблицю;

2) Обрати та натиснути кнопку «OK»;

3) Перейти у вікно таблиці.

Очікуваним результатом даного тест-кейсу є вибір таблиці та перехід у вікно з інформацією про неї після натискання кнопки «OK». Результат виконання тест кейсу показано на рисунку 4.4.

Рисунок 4.4 – Результат виконання тест-кейсу №4

Фактичний результат відповідає очікуваному, отже тест-кейс №4 успішно пройдено.

Тест-кейс №5 «Перевірка роботи кнопки «Add»:

1) Обрати та натиснути копку «Add»;

2) Дочекатися появи діалогового вікна;

3) Додати новий товар.

Очікуваним результатом даного тест-кейсу є відкриття діалогового вікна для додання нового товару та його додання після вводу даних. Результат виконання тест кейсу показано на рисунку 4.5.

Фактичний результат відповідає очікуваному, отже тест-кейс №5 успішно пройдено.

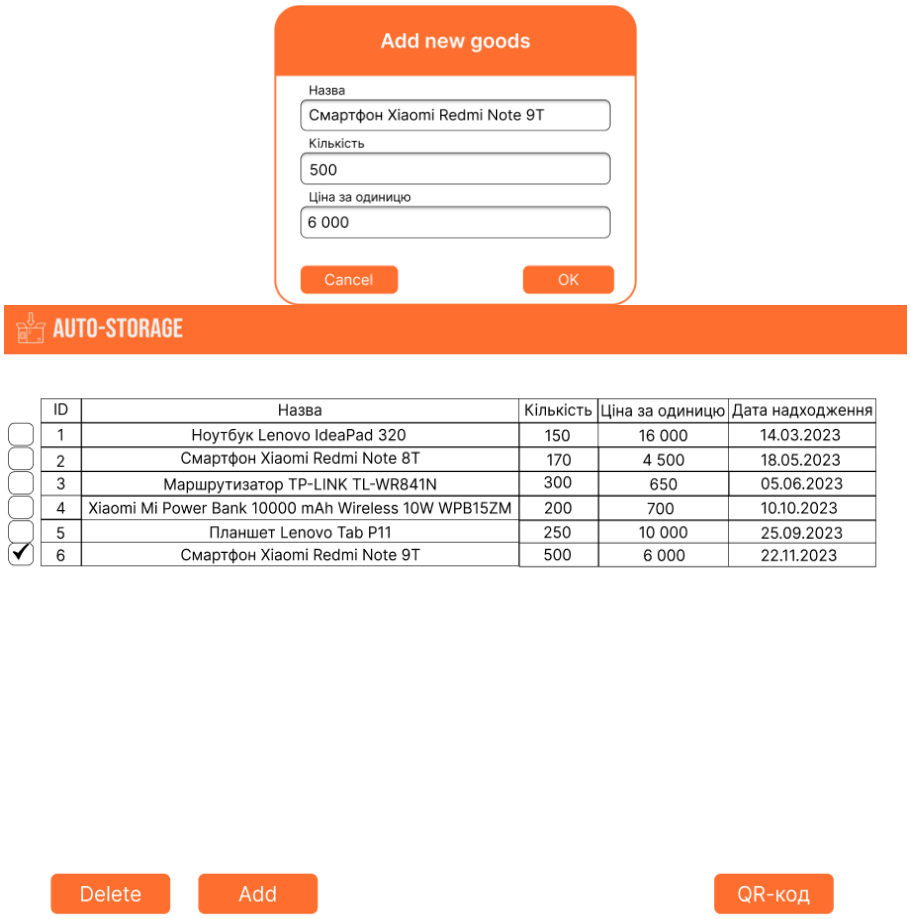

Рисунок 4.5 – Результат виконання тест-кейсу №5

Тест-кейс №6 «Перевірка роботи кнопки «Delete»:

1) Обрати та натиснути кнопку «Delete»;

2) Дочекатися появи діалогового вікна;

3) Видалити товар.

Очікуваним результатом даного тест-кейсу є відкриття діалогового вікна для видалення товару та його видалення після підтвердження. Результат виконання тест кейсу показано на рисунку 4.6.

Фактичний результат відповідає очікуваному, отже тест-кейс №6 успішно пройдено.

Рисунок 4.6 – Результат виконання тест-кейсу №6

Тест-кейс №7 «Перевірка роботи кнопки «QR-код»:

1) Обрати та натиснути кнопку «QR-код»;

2) Підтвердити генерацію QR-коду у діалоговому вікні;

3) Згенерувати QR-код.

Очікуваним результатом даного тест-кейсу є відкриття діалогового вікна для генерації QR-коду та його генерація після підтвердження. Результат виконання тест-кейсу показано на рисунку 4.7.

Фактичний результат відповідає очікуваному, отже тест-кейс №7 успішно пройдено.

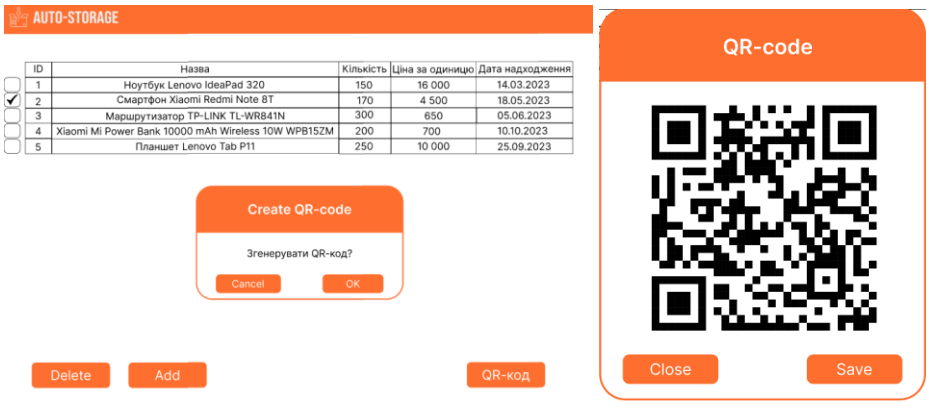

Рисунок 4.7 – Результат виконання тест-кейсу №7

Тест-кейс №8 «Перевірка сканування QR-коду та виведення даних»:

1) Увійти у мобільний додаток сканер;

2) Навести камеру на QR-код та натиснути кнопку «Сканувати»;

3) Перевірити чи виводиться дані з просканованого QR-коду;

4) Закрити додаток.

Очікуваним результатом даного тест кейсу є успішне сканування обраного QR-коду та виведення відсканованих даних на екран. Результат виконання тест кейсу показано на рисунку 4.8.

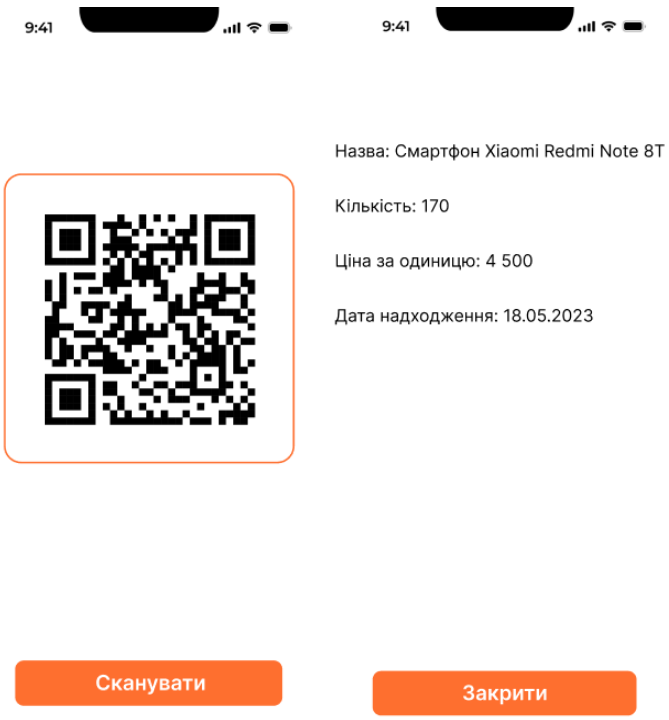

Рисунок 4.8 – Результат виконання тест-кейсу №8

Фактичний результат відповідає очікуваному, отже тест-кейс №8 успішно пройдено.

Таким чином було проведено тестування додатку за допомогою методу «чорної скриньки». При проходженні тест-кейсів помилок чи багів не виявлено. У результаті доведено повну працездатність програмного додатку та його відповідність поставленому технічному завданню.

4.3 Розробка інструкції користувача

Для допомоги у вивчені функцій створюваного програмного додатку було розроблено інструкцію користувача.

Для початку роботи з додатком потрібно запустити файл Auto_Storage.exe, що відкриє перед користувачем вікно авторизації.

Вікно авторизації показане на рисунку 4.9.

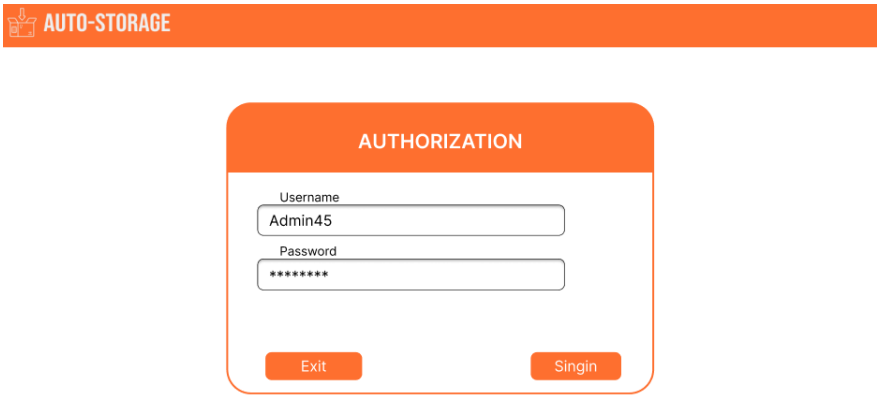

Рисунок 4.9 – Вікно авторизації додатку «Auto-Storage»

У верхньому лівому куті знаходиться логотип додатку «Auto-Storage». Посередині знаходиться вікно авторизації. Якщо користувач хоче увійти до системи він має ввести дані свого облікового запису та натиснути кнопку «Singin».

Після авторизації перед користувачем відкриється головне вікно додатку. Як і у вікні авторизації логотип додатку знаходиться у верхньому лівому куті. Основну частину вікна займає список таблиць що складається з п’яти колонок, а саме:

- ID – порядковий номер таблиці;
- Назва – назва складу;
- Адміністратор – відповідальна за склад особа;
- Дата створення – дата, коли було створено таблицю;
- Останнє відвідування – дата останнього відвідування таблиці.

Також у вікні є три кнопки: «Delete», «Add» та «OK». Їх користувач може використовувати для навігації у додатку. Головне вікно представлено на рисунку 4.10.

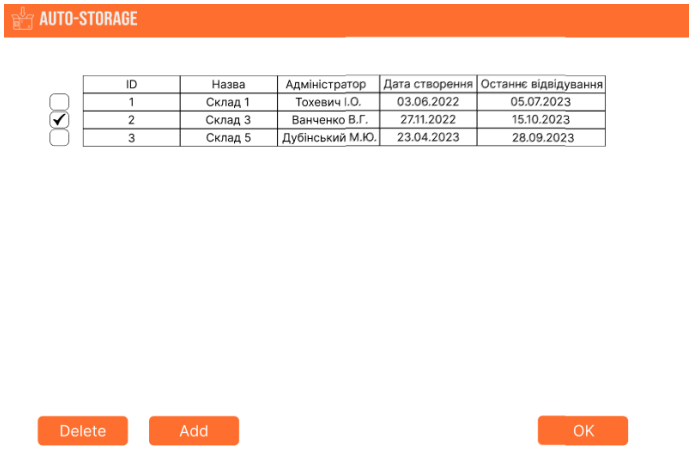

Рисунок 4.10 – Головне вікно додатку

При натисканні кнопки «Add» перед користувачем випливе діалогове вікно, що запропонує йому вписати параметри та створити нову таблицю. Діалогове вікно додання нової таблиці зображено на рисунку 4.11.

Рисунок 4.11 – Діалогове вікно додання нової таблиці

Якщо користувач натисне кнопку «Delete» перед користувачем з’явиться діалогове вікно, яке запросить підтвердження видалення обраної таблиці. Діалогове вікно видалення таблиці зображено на рисунку 4.12.

Рисунок 4.12 – Діалогове вікно видалення таблиці

Після того, як користувач обере таблицю та натиске кнопку «ОК», він перейде до списку товарів, що розміщений у даний таблиці. Вікно списку товарів продемонстровано на рисунку 4.13.

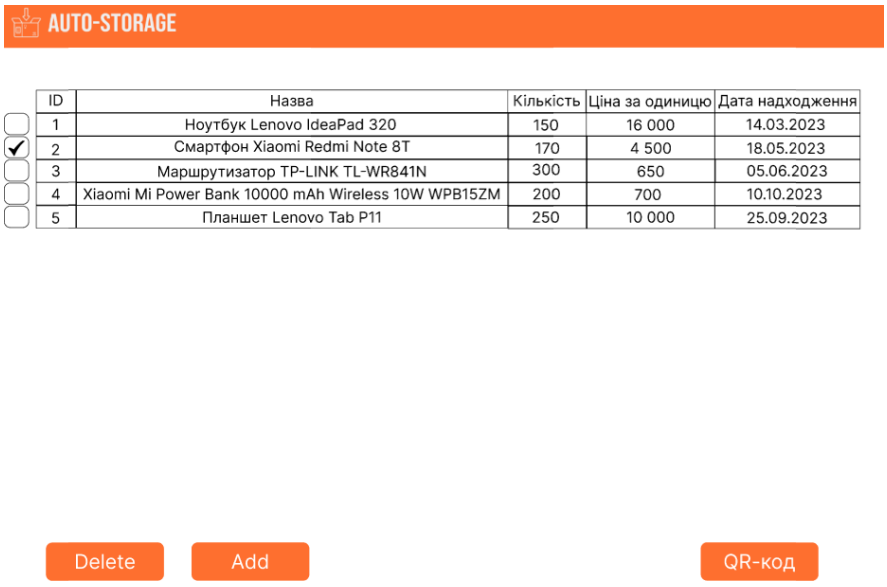

Рисунок 4.13 – Вікно списку товарів

На цьому вікні як і на попередніх у верхньому лівому куті розташований логотип додатку. У самому вікні знаходиться список товарів, що були розміщені в обраній таблиці. Він складається з п’яти колонок:

- ID – порядковий номер товару;
- Назва – назва товару;
- Кількість – кількість одиниць товару що наявні на складі;
- Ціна за одиницю – ціна за одиницю товару;
- Дата надходження – дата надходження товару на склад.

Також у вікні є три кнопки: «Delete», «Add» та «QR-код». Їх користувач може використовувати для навігації у додатку.

При натисканні користувачем кнопки «Add» перед користувачем випливе діалогове вікно з параметрами для додання нового товару. Після заповнення параметрів та підтвердження до списку буде додано новий товар. Діалогове вікно додання нового товару показано на рисунку 4.14.

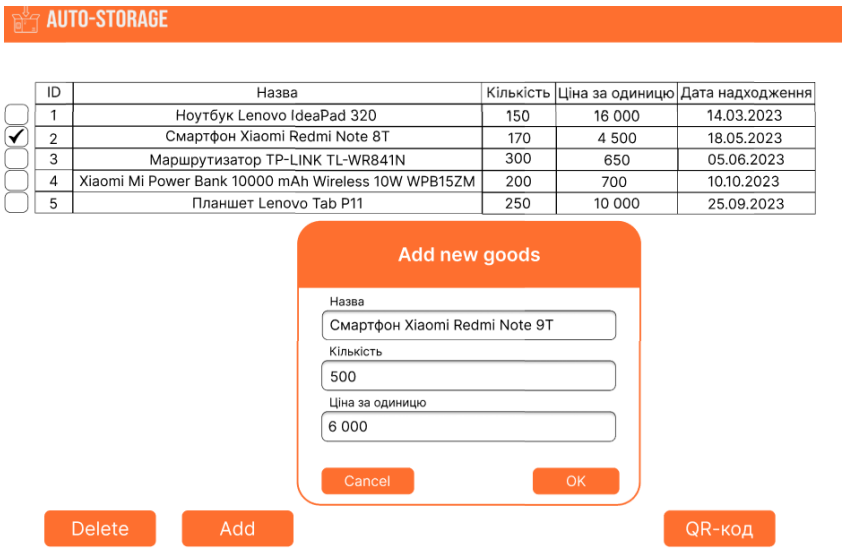

Рисунок 4.14 – Діалогове вікно додання нового товару

Якщо користувач обере товар та натисне кнопку «Delete» перед ним з’явиться діалогове вікно, що запросить підтвердження на видалення товару зі списку. Діалогове вікно видалення товару представлено на рисунку 4.15.

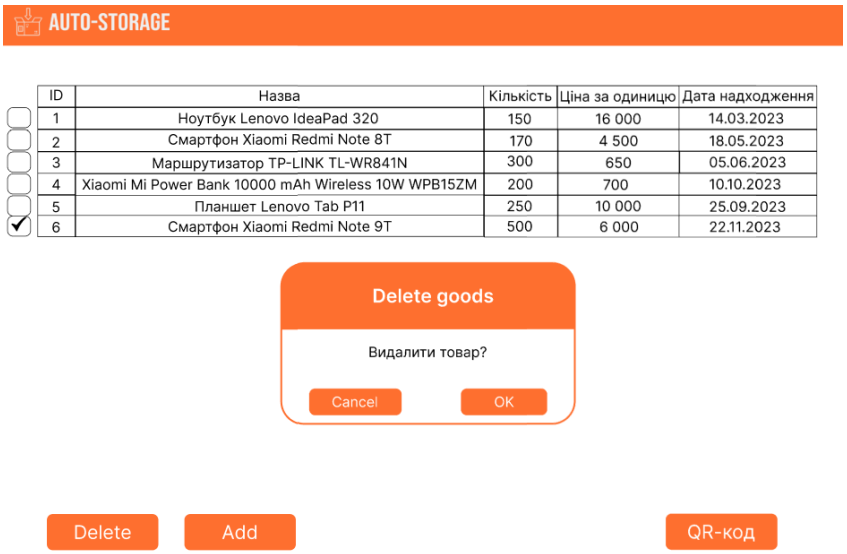

Рисунок 4.15 – Діалогове вікно видалення товару

При натисканні користувачем кнопки «QR-код» перед ним випливе діалогове вікно із запитом чи хоче він згенерувати QR-код. Якщо користувач підтверджує генерацію – на екрані з’явиться вікно із QR-кодом та двома кнопками навігації:

- Close – для закриття вікна;
- Save – для збереження QR-коду.

Вікно із згенерованим QR-кодом зображено на рисунку 4.16.

Рисунок 4.16 – Вікно QR-коду

Для користувачів складу було розроблено спеціальний мобільний додаток сканер, щоб вони могли дізнатися інформацію про товар відсканувавши QR-код.

При запуску додатку на смартфоні відкривається вікно сканеру що містить у собі обмежений прямокутник камери для сканування QR-коду та кнопка «Сканувати», що розпочинає сканування. Вікно сканеру продемонстровано на рисунку 4.17.

Рисунок 4.17 – Вікно сканеру

Після того як QR-код було відсканово, відкривається вікно що містить дані закодовані всередині QR-коду. Там присутня інформація про назву товару його кількість, ціну за одиницю та дату надходження на склад. У цьому вікні також присутня кнопка «Закрити», яка поверне користувача назад до вікна сканеру. Вікно з даними QR-коду показано на рисунку 4.18.

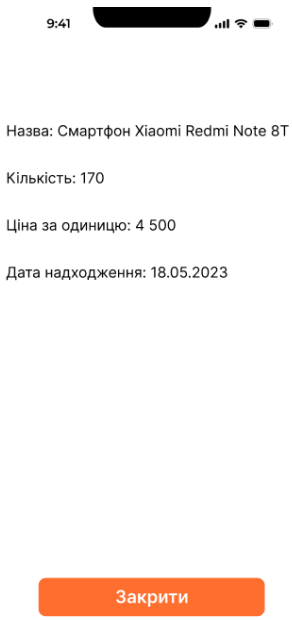

Рисунок 4.18 – Вікно з даними QR-коду

Таким чином було розроблено інструкцію користувача для початку роботи з додатком «Auto-Storage». Було розглянуто всі можливості що надає додаток користувачеві.

4.4 Системні вимоги

Так як розроблений додаток працює на базі робочої машини, то для його використання користувачу потрібно встановити його на свій пристрій. Мінімальна та рекомендована конфігурації персонального комп'ютера на базі операційної системи Windows наведено у таблиці 4.1.

Таблиця 4.1 – Мінімальна та рекомендована конфігурація

Параметр	Мінімальна конфігурація	Рекомендована конфігурація
CPU	Процесор на архітектурі x86-64 або ARM64 з тактовою частотою 1,6 ГГц або більше	Процесор на архітектурі x86-64 або ARM64 з тактовою частотою 2,4 ГГц або більше
RAM	4 ГБ для ОС Windows або MacOS	8 ГБ для ОС Windows або MacOS
GPU	Інтегрований графічний пристрій	Інтегрований графічний пристрій
Накопичувач	HDD, 50ГБ вільного місця	SSD, 100 ГБ вільного місця
ОС	Windows 7	Windows 10

Таким чином було розглянуто мінімальні та рекомендовані системі вимоги для початку роботи із додатком «Auto-Storage».

4.5 Висновки

В четвертому розділі було проведено аналіз методик тестування програмного забезпечення та обрано методику «чорної скриньки», що передбачає перевірку функціональності додатку за допомогою тест-кейсів. Результат тестування програмного додатку довів його відповідність поставленому технічному завданню та повну працездатність. Також було розроблено інструкцію користувача, яка допоможе йому ознайомитися із функціями додатку перед його використанням. Крім того було визначено мінімальну та рекомендовану конфігурації персональних комп’ютерів, які необхідні для правильної та ефективної роботи додатку.

5 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота за темою «Розробка методу та програмного засобу для автоматизації ведення складського обліку» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Розробка методу та програмного засобу для автоматизації ведення складського обліку» є оцінювання науково-технічного рівня та рівня

комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1 [34].

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Продовження таблиці 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці 5.2.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	3	3	3
2. Ринкові переваги (наявність аналогів)	4	3	4
3. Ринкові переваги (ціна продукту)	3	3	3
4. Ринкові переваги (технічні властивості)	3	3	3
5. Ринкові переваги (експлуатаційні витрати)	3	3	3
6. Ринкові перспективи (розмір ринку)	3	3	3
7. Ринкові перспективи (конкуренція)	3	3	3
8. Практична здійсненність (наявність фахівців)	3	3	3
9. Практична здійсненність (наявність фінансів)	3	3	3
10. Практична здійсненність (необхідність нових матеріалів)	3	3	3
11. Практична здійсненність (термін реалізації)	3	4	3
12. Практична здійсненність (розробка документів)	4	3	3
Сума балів	38	37	37
Середньоарифметична сума балів $СБ_c$	**37,3**		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в таблиці 5.3 [35].

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ ,розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методу та програмного засобу для автоматизації ведення складського обліку» становить 37,3 бали, що, відповідно до таблиці 5.3, свідчить

про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вищий середнього).

5.2 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Розробка методу та програмного засобу для автоматизації ведення складського обліку», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.2.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \qquad (5.1)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дні;

T_p – середнє число робочих днів в місяці, T_p=21 день.

$$З_о = 22000{,}00 \cdot 60 / 21 = 62857{,}14 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці 5.4.

Таблиця 5.4 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	22000	1047,62	60	62857,14
Інженер-розробник програмного забезпечення	20000	952,38	58	55238,10
UI/UX-дизайнер	19000	904,76	15	13571,43
Консультант з економічних питань	20000	952,3809524	5	4761,90
Всього				136428,57

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i , \quad (5.2)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} , \quad (5.3)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=6700,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду;

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$T_р$ – середнє число робочих днів в місяці, приблизно $T_р$ = 21 день;

$t_{зм}$ – тривалість зміни, год.

$С_1 = 6700,00 \cdot 1,5 \cdot 1,65 / (21 \cdot 8) = 98,71$ грн.
$З_{р1} = 98,71 \cdot 5,00 = 493,53$грн.

Таблиця 5.5 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Налаштування робочих станцій	5	4	1,5	98,71	493,53
Налаштування локальної мережі для обладнання	10	4	1,5	98,71	987,05
Налаштування і підключення серверу для середовища розробки	15	5	1,7	111,87	1677,99
Всього					3158,57

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.4)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 12%.

$З_{дод} = (136428,57 + 3158,57) \cdot 12 / 100\% = 16750,46$грн.

5.2.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.5)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

$Зн = (136428,57 + 3158,57 + 16750,46) \cdot 22 / 100\% = 34394,272$грн.

5.2.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень.

Витрати на матеріали (M), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj}\,, \qquad (5.6)$$

де H_j – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 ... 1,15*);

B_j – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

$М_1 = 3 \cdot 220{,}00 \cdot 1{,}1 - 0{,}000 \cdot 0{,}00 = 726{,}0$ грн.

Проведені розрахунки зведемо до таблиці 5.6.

Таблиця 5.6 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за од, грн	Норма витрат, од	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір офісний BARVA (A4-500)A4	220	3	0	0	726
Папір для записів Axent Elite White	120	1	0	0	132
Органайзер для паперів	210	2	0	0	462
Канцелярське приладдя (набір офісного працівника)	200	3	0	0	660
Flesh-пам'ять Kingston 16 GB	130	1	0	0	143
Всього					2123

5.2.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Розробка методу та програмного засобу для автоматизації ведення складського обліку» відсутні.

5.2.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \quad , \qquad (5.7)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань устаткування.

$B_{спец} = 25000 \cdot 3 \cdot 1,1 = 82500$ грн.

Отримані результати зведемо до таблиці 5.7.

Таблиця 5.7 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Робоча станція (ПК)	3	25000	82500
Маршрутизатор	1	1800	1980
Сервер для середовища розробки та тестування	1	22000	24200
Всього			108680

5.2.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (5.8)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 1200,00 \cdot 1 \cdot 1,12 = 1344$ *грн.*

Отримані результати зведемо до таблиці 5.8.

Таблиця 5.8 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Операційна система Microsoft Windows 10 Pro	3	14000	47040
Середовище розробки IntelliJ IDEA 2023.1	2	3600	8064
Графічний редактор Adobe Photoshop	1	1200	1344
Всього			56448

5.2.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.9)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (59400{,}00 \cdot 3) / (3 \cdot 12) =$ 4950,00 *грн.*

Проведені розрахунки зведемо до таблиці 5.9.

Таблиця 5.9 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Робоча станція (ПК)	59400	3	3	4950,00
Маршрутизатор	1870	3	3	155,83
Сервер для середовища розробки та тестування	22000	5	3	1100,00
Оргтехніка	6000	4	2	250,00
Всього				5750,00

5.2.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію ($B_е$) розраховуємо за формулою:

$$B_е = \sum_{i=1}^{n} \frac{W_{уі} \cdot t_i \cdot Ц_е \cdot K_{впі}}{\eta_i}, \quad (5.10)$$

де $W_{уі}$ – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е = 7,50$ грн;

$К_{вп і}$ – коефіцієнт, що враховує використання потужності, $К_{вп і} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$В_е = 0,5 \cdot 480,0 \cdot 7,50 \cdot 0,95 / 0,97 = 1762,89 грн.$

Проведені розрахунки зведемо до таблиці 5.10.

Таблиця 5.10 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Блок живлення робочої станції	0,5	480	1762,89
Блок живлення маршрутизатора	0,02	60	8,81
Блок живлення сервера	0,7	60	308,51
Оргтехніка	0,45	20	66,11
Всього			2146,31

5.2.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (5.11)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св} = 20\%$.

$В_{св} = (136428,57 + 3158,57) \cdot 20 / 100\% = 27917,43 грн.$

5.2.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (5.12)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

$B_{сп}$ = (136428,57+ 3158,57) · 30 / 100% = 41876,14грн.

5.2.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.13)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 50%.

$І_в$ = (136428,57+ 3158,57) · 50 / 100% = 69793,57грн.

5.2.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}\ , \qquad (5.14)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 120%.

$B_{нзв} = (136428,57 + 3158,57) \cdot 120 / 100\% = 167\,504,57$ грн.

Витрати на проведення науково-дослідної роботи розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{св} + B_{сп} + I_в + B_{нзв}. \qquad (5.15)$$

$B_{заг}$ = 672970,90 грн.

Загальні витрати $ЗВ$ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \qquad (5.16)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,9.

$ЗВ = 672970,90 / 0,9 = 747745,45$грн.

5.3 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Розробка методу та програмного засобу для автоматизації ведення складського обліку» передбачають комерціалізацію протягом 3-х років реалізації на ринку.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

1-й рік – 100 користувачів;

2-й рік – 300 користувачів;

3-й рік – 350 користувачів.

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 600 користувачів;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 12000 грн;

$\pm\Delta Ц_o$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 2000 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 3-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [36]**:**

$$\Delta П_i = (\pm\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.17)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту. Приймемо ρ =40%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (2000{,}00 \cdot 600{,}00 + 14000{,}00 \cdot 100) \cdot 0{,}83 \cdot 0{,}4 \cdot (1\text{-}0{,}18/100\%) =$ 1092896 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 = (2000{,}00 \cdot 600{,}00 + 14000{,}00 \cdot (100 + 300)) \cdot 0{,}83 \cdot 0{,}4 \cdot (1\text{-}0{,}18/100\%) = 2240436{,}8$ грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (2000{,}00 \cdot 600{,}00 + 140000{,}00 \cdot (100+300+350)) \cdot 0{,}83 \cdot 0{,}4 \cdot (1-0{,}18/100\%) =$ 3579234,4грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (5.18)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=0,15;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП = 1092896/(1+0{,}15)^1 + 2240436{,}8/(1+0{,}15)^2 + 3579234{,}4/(1+0{,}15)^3 = 4997841{,}16$ грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \quad (5.19)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 747745,45грн.

$PV = k_{інв} \cdot ЗВ = 2 \cdot 747745{,}45 = 1495490{,}89$грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{5.20}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 4997841,16грн;

PV – теперішня вартість початкових інвестицій, 1495490,89грн.

$E_{абс} = ПП - PV = 4997841{,}16 - 1495490{,}89 = 3502350{,}27$грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.21}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 3502350,27грн;

PV – теперішня вартість початкових інвестицій, 1495490,89грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 3 роки.

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+3502350{,}27/1495490{,}89)^{1/3} = 0{,}5.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \tag{5.22}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,11;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,18.

$\tau_{мін}$ = 0,11+0,18 = 0,29 < 0,5 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Розробка методу та програмного засобу для автоматизації ведення складського обліку» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \tag{5.23}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,5 = 2 роки.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

5.3 Висновки

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Розробка методу та програмного засобу для автоматизації ведення складського обліку» становить 37,3 бали, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вищий середнього).

Також термін окупності становить 2 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати

потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Розробка методу та програмного засобу для автоматизації ведення складського обліку».

ВИСНОВКИ

У магістерській кваліфікаційній роботі було розроблено методи та програмні засоби для автоматизації ведення складського обліку.

Проведено аналіз сучасних методів для автоматизації складського обліку. Розглянуто відомі програмні рішення, досліджено їх переваги та недоліки, а також виконано порівняння із власним додатком. За результатами порівняння було прийнято рішення про доцільність розробки власного програмного додатку для вирішення наявних проблем. Проведено постановку задачі.

Проведено розробку методів та алгоритмів програмного продукту. Подальшого розвитку набув метод генерації QR-кодів, в якому, на відміну від існуючих, було використано алгоритм Reed-Solomon для швидкого виявлення та виправлення помилок, що дозволило підвищити ефективність зберігання та передачі даних.

Подальшого розвитку набув метод зчитування QR-кодів, в якому, на відміну від класичного, для підвищення якості зображення було використано медіальний фільтр, що дало можливість підвищити точність зчитування даних. Проведено розробку архітектури додатку.

Виконано розробку структури графічного інтерфейсу з детальним описом ключових вікон додатку. Також досліджено та обрано кольорову гаму. Розроблено та спроектовано базу даних програмного додатку. Проведено варіантний аналіз та обґрунтування вибору мови програмування. Вирішено для реалізації поставлених задач використовувати мову Java. Обрано середовище розробки Intelij IDEA, оскільки доступний перелік функцій спрощує розробку, а підтримка інтеграції з системою контролю версій Git допоможе надійно зберігати написаний код. Виконано програмну реалізацію головних компонент додатку.

Проведено аналіз методів тестування програмного забезпечення. За результатами аналізу було обрано метод «чорного ящика» для перевірки працездатності розробленого продукту. Розроблено ряд тестових випадків для

тестування функції ПЗ, перевірка працездатності не виявила недоліків. Розроблено інструкцію користувача та сформовано мінімальну та рекомендовану конфігурації апаратного забезпечення для використання ПЗ.

Отримані в магістерській кваліфікаційній роботі наукові та практичні положення можна застосувати для автоматизації складського обліку.

Згідно з проведеними дослідженнями рівень комерційного потенціалу розробки становить 37,3 бали, що свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього). Це доводить, що програмний продукт є перспективною розробкою. Задачі магістерської кваліфікаційної роботи виконано в повному обсязі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сіянко М. О. Сучасні цифрові технології для автоматизованого управління складським обліком / Микита Олександрович Сіянко, Людмила Броніславівна Ліщинська // Електронні інформаційні ресурси: створення, використання, доступ : Міжнар. науково-практ. Інтернет-конф., Суми/Вінниця, 28–29 листоп. 2023 р. – Суми/Вінниця, 2023. – С. 267.

2. Сіянко М. О. Генерація QR-кодів з використанням алгоритмів корекції помилок BCH та Reed-Solomon [Електронний ресурс] / Микита Олександрович Сіянко, Людмила Броніславівна Ліщинська // LII Науково-технічна конференція факультету інформаційних технологій та комп'ютерної інженерії, Вінниця, 21–23 черв. 2023 р. – Вінниця, 2023. – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2024
14546 (дата звернення: 30.11.2023).

3. M. T. Hompel and T. Schmidt, Warehouse Managmenet - Automation and Organisation of Warehouse and Order Picking Systems. Berlin: Springer Verlag Heidelberg, 2007. 315 p.

4. J. J. Bartholdi and S. T. Hackman, Warehouse & Distribution Science. Georgia Institute of Technology, School of Industrial and Systems Engineering, The Supply Chain and Logistics Institute, August 22 2011, latest release: version 0.95. URL: https://www2.isye.gatech.edu/~jjb/wh/book/editions/wh-sci-0.96.pdf (date of access: 23.09.2023).

5. УкрСклад Універсальний складський облік. URL: https://www.ukrsklad.com/ (date of access: 25.09.2023).

6. Торгсофт. URL: https://torgsoft.ua/price/online/?gad=1&gclid=Cj0KCQjwj5mpBhDJARIsAOVjBdoSYCbciY4OBnVGCKme6jYOH-tvyV797KdtCocV4HON14554dk-ZxYaAjSpEALw_wcB (date of access: 26.09.2023).

7. Онлайн-бухгалтерія Dilovod. URL: https://dilovod.ua/uchet-sklada/?utm_source=google&utm_medium=cpc&utm_campaign=dilovod_PMax_%

D0%A1%D0%BA%D0%BB%D0%B0%D0%B4%D1%81%D0%BA%D0%BE%D0%B9_%D1%83%D1%87%D0%B5%D1%82&gad=1&gclid=Cj0KCQjwj5mpBhDJARIsAOVjBdpeCh0GYAAFCRrMakPRIPV2YrE-q9zqmbAfd9f81DvpzFi5t2JXKiUaAqh2EALw_wcB (date of access: 28.09.2023).

8. Програма складського обліку – RemOnline Україна. URL: https://remonline.ua/features/inventory-management/?url=https://remonline.ua/features/inventory-management/?utm_source=google&utm_medium=cpc&utm_campaign=19615509254&utm_content=646245391403&utm_term=&gclid=Cj0KCQjwj5mpBhDJARIsAOVjBdpMNgxmexYuvtyg_KNLt73aJzTbs-KhIaWxNrOjrkV0JXTumUtT9cUaAjhoEALw_wcB (date of access: 29.09.2023).

9. Вертикальне та горизонтальне масштабування. URL: https://azure.microsoft.com/ru-ru/resources/cloud-computing-dictionary/scaling-out-vs-scaling-up (date of access: 30.09.2023).

10. Todd K. Moon, Wynn C. Stirling, Error Correction Coding: Mathematical Methods and Algorithms. USA: Wiley-Interscience, 2005. 800 p.

11. J. H. van Lint, R. M. Wilson, Introduction to Algebraic Coding Theory. Georgia Institute of Technology, Berlin: Springer, 1998. 248 p.

12. Daniel J. Costello Jr., Stephen A. Yatauro, Cyclic Codes for Error Detection and Correction. Singapore: World Scientific, 2007. 201 p.

13. Scott Stratten, QR Codes Kill Kittens: How to Alienate Customers, Dishearten Employees, and Drive Your Business into the Ground. USA: Wiley, 2013. 208 p.

14. Elwyn R. Berlekamp, Algebraic Coding Theory and Applications. USA: Wiley-Interscience, 2015. 247 p.

15. What is a graphical user interface?. URL: https://www.omnisci.com/technical-glossary/graphical-user-interface (date of access: 24.10.2023).

16. Kowalski R. The CLI Book / Robert Kowalski. Packt Publishing, 2017, 109 p.

17. Bill Scott, Theresa Neil Designing Web Interfaces: Principles and Patterns for Rich Interactions. USA: O'Reilly Media, 2009. 332p.

18. Craig S. Mullins, Database Administration: The Complete Guide to Practices and Procedures. Addison-Wesley Professional, 2002. 736p.

19. Bjarne Stroustrup. The C++ Programming Language, 4th Edition, 2013. 1376 p.

20. Joshua Bloch. Effective Java 3rd Edition, 2017. 392 p.

21. Skeet J. C# in depth: fourth edition. 4th ed. Shelter Island : Manning, 2019. 528 p.

22. Lutz M. Learning python. 5th ed. Sebastopol : O'Reilly Media, 2013. 1643 p.

23. Fan B. VSCode vs Jetbrains – revisited URL: https://www.shade.inc/posts/vscode-vs-jetbrains-revisited-update-sep-2023 (дата звернення: 26.10.2023).

24. IntelliJ IDEA – the Leading Java and Kotlin IDE URL: https://www.jetbrains.com/idea/ (дата звернення: 26.10.2023).

25. About the Eclipse Foundation URL: https://www.eclipse.org/org/ (дата звернення: 26.10.2023).

26. Apache NetBeans Fits the Pieces Together URL: https://netbeans.apache.org/front/main/ (дата звернення: 26.10.2023).

27. javax.swing (Java Platform SE 8). URL: https://docs.oracle.com/javase/8/docs/api/javax/swing/package-summary.html (date of access: 28.10.2023).

28. Java JDBC API. URL: https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/ (date of access: 01.11.2023).

29. zxing/library. URL: https://www.npmjs.com/package/@zxing/library (date of access: 05.11.2023).

30. Maven – Welcome to Apache Maven. URL: https://maven.apache.org/ (date of access: 06.11.2023).

31. Gradle Build Tool. URL: https://gradle.org/ (date of access: 06.11.2023).

32. OpenCV - Open Computer Vision Library. URL: https://opencv.org/ (date of access: 09.11.2023).

33. Види тестування та відмінності між ними. Шпаргалка з тестування. URL: https://qagroup.com.ua/publications/vydy-testuvannya-ta-vidminnosti-mizh-nymy/ (date of access: 17.11.2023).

34. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. Вінниця : ВНТУ, 2021. 42 с.

35. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа. Вінниця : ВНТУ, 2016. 113 с.

ДОДАТКИ

105

Додаток А. Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
"19" вересня 2023 р.

Технічне завдання
на магістерську кваліфікаційну роботу «Розробка методу та програмного засобу для автоматизації ведення складського обліку»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
_____ д.т.н., проф. Л.Б. Ліщинська
"19" 09 2023 р.
Виконав:
_____ студент гр.3ПІ-22м М.О. Сіянко
"19" 09 2023 р.

Вінниця – 2023 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методу та програмного засобу для автоматизації ведення складського обліку».

Галузь застосування - системи складського обліку.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № 247 від 18 вересня 2023 року ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення продуктивності засобів ведення складського обліку за рахунок додання способу шифрування даних у QR-коди та її зчитування працівниками.

Призначення роботи – розробка методів і програмних засобів автоматизації складського обліку.

3 Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Міжнародноа науково-практичноа Інтернет-конференція «Електронні інформаційні ресурси: створення, використання, доступ» (Вінниця, 2023). Сучасні цифрові технології для автоматизованого управління складським обліком.

2. «Молодь в науці: дослідження, проблеми, перспективи» (Вінниця 2023). Генерація QR-кодів з використанням алгоритмів корекції помилок BCH та Reed-Solomon: Науковий Огляд та Перспективи.

3. M. T. Hompel and T. Schmidt, Warehouse Managmenet - Automation and Organisation of Warehouse and Order Picking Systems. Berlin: Springer Verlag Heidelberg, 2007. 315 p.

4. J. J. Bartholdi and S. T. Hackman, Warehouse & Distribution Science.

Georgia Institute of Technology, School of Industrial and Systems Engineering, The Supply Chain and Logistics Institute, August 22 2011, latest release: version 0.95. URL: https://www2.isye.gatech.edu/~jjb/wh/book/editions/wh-sci-0.96.pdf (date of access: 10.10.2023).

4. Технічні вимоги

Графічний режим – TrueColor; система керування базами даних – Microsoft SQL Server; мова запитів – SQL; вхідні дані – дані з кожного вікна такі як: кількість елементів у вікні, новий користувач, нова таблиця; вихідні дані – таблиці з даними їх атрибутів; середовище розробки – IntelliJ IDEA; мова програмування – Java.

5. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

6. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

7. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

8. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз стану питання	20.09.2023 – 24.09.2023
2	Порівняльний аналіз аналогів	25.09.2023 – 01.10.2023
3	Розробка графічного інтерфейсу додатку	02.10.2023 – 12.10.2023
4	Розробка алгоритму роботи додатку	13.10.2023 – 23.10.2023
5	Розробка програмних компонент додатку	24.10.2023 – 10.11.2023
6	Тестування програмного забезпечення	11.11.2023 – 24.11.2023
7	Економічна частина	25.11.2023 – 01.12.2023

9. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи. Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком

109

Додаток Б. Протокол перевірки кваліфікаційної роботи на наявність запозичень

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: «Розробка методу та програмного засобу для автоматизації ведення складського обліку»

Тип роботи: магістерська кваліфікаційна робота

Підрозділ: кафедра програмного забезпечення, ФІТКІ, ЗПІ-22м

Науковий керівник: Лішинська Л.Б.

Unicheck	
Оригінальність	**82.6%**
Схожість	17,4%

Аналіз звіту подібності

1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г.О.

Опис прийнятого рішення: **допустити до захисту**

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Сіянко М.О.

Керівник роботи _____ Ліщинська Л.Б.

Додаток В. Ілюстрації до третього розділу

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class AuthorizationWindow extends JFrame {

    1 usage
    public AuthorizationWindow() {
        // Заголовок вікна
        setTitle("Authorization");

        // Панель для розміщення компонентів
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout( rows: 3,  cols: 2));
```

Рисунок В.1 – Підключення бібліотеки та створення заголовку і панелі для розміщення елементів

```
// Лейбли та тексти для вводу
JLabel usernameLabel = new JLabel( text: "Username:");
JTextField usernameTextField = new JTextField();

JLabel passwordLabel = new JLabel( text: "Password:");
JPasswordField passwordField = new JPasswordField();

// Кнопки
JButton exitButton = new JButton( text: "Exit");
JButton signinButton = new JButton( text: "Signin");
```

Рисунок В.2 – Додання полів для вводу логіну та паролю користувача та кнопок «Exit» і «Singin»

```
// Додавання обробників подій для кнопок
exitButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        System.exit( status: 0); // Закриття програми при натисканні кнопки "Exit"
    }
});

signinButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // Отримання значень з полів для вводу
        String username = usernameTextField.getText();
        char[] password = passwordField.getPassword();

        JOptionPane.showMessageDialog( parentComponent: AuthorizationWindow.this,
                message: "Username: " + username + "\nPassword: " + new String(password),
                title: "Authorization Successful", JOptionPane.INFORMATION_MESSAGE);
    }
});
```

Рисунок В.3 – Обробник подій для кнопок та полів вводу

```
// Додавання панелі на вікно
getContentPane().add(panel);

// Розмір вікна та операция закриття
setSize( width: 300,  height: 150);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

// Встановлення видимості вікна
setVisible(true);
```

Рисунок В.4 – Створення вікна та додання на нього панелі з елементами

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class MainWindow extends JFrame {
    1 usage
    public MainWindow() {
        // Заголовок вікна
        setTitle("Main Window");

        // Зображення логотипу
        ImageIcon logoIcon = new ImageIcon( filename: "D:\\Диплом Магістр\\logo.png");
        JLabel logoLabel = new JLabel(logoIcon);

        // Панель для розміщення компонентів
        JPanel panel = new JPanel();
        panel.setLayout(new FlowLayout(FlowLayout.LEFT));
```

Рисунок В.5 – Підключення бібліотеки та створення заголовку, панелі для розміщення елементів та додання логотипу додатку

```
// Кнопка "Add"
JButton addButton = new JButton( text: "Add");
addButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // Додайте код для обробки події кнопки "Add" тут
        JOptionPane.showMessageDialog( parentComponent: MainWindow.this,  message: "Add button clicked");
    }
});
```

Рисунок В.6 – Додання кнопки для взаємодії із користувачем

```
        // Додавання панелі на вікно
        getContentPane().add(panel);

        // Розмір вікна та операция закриття
        setSize( width: 600,  height: 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Встановлення видимості вікна
        setVisible(true);
```

Рисунок В.7 – Створення вікна та додання на нього панелі з елементами

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class DatabaseIntegration {
    public static void main(String[] args) {
        // Змінні для підключення до бази даних
        String jdbcURL = "jdbc:mysql://localhost:3306/your_database_name";
        String username = "your_username";
        String password = "your_password";

        // Об'єкти для роботи з базою даних
        Connection connection = null;
        Statement statement = null;
        ResultSet resultSet = null;
```

Рисунок В.8 – Підготовка змінних та об’єктів для роботи з базою даних

```java
try {
    // Завантаження драйвера JDBC
    Class.forName( className: "com.mysql.cj.jdbc.Driver");

    // Підключення до бази даних
    connection = DriverManager.getConnection(jdbcURL, username, password);

    // Створення об'єкта Statement для виконання запитів
    statement = connection.createStatement();
```

Рисунок В.9 – Підключення драйверу JDBC

```
        // Виконання SQL-запиту SELECT
        String sqlQuery = "SELECT * FROM your_table_name";
        resultSet = statement.executeQuery(sqlQuery);

        // Обробка результатів запиту
        while (resultSet.next()) {
            int id = resultSet.getInt( columnLabel: "id");
            String name = resultSet.getString( columnLabel: "name");
            // Опрацювання інших полів з бази даних
            System.out.println("ID: " + id + ", Name: " + name);
        }
    } catch (SQLException | ClassNotFoundException e) {
        e.printStackTrace();
    } finally {
        // Закриття ресурсів
        try {
            if (resultSet != null) resultSet.close();
            if (statement != null) statement.close();
            if (connection != null) connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

Рисунок В.10 – Виконання запиту SELECT у базі даних MySQL

```
implementation 'com.google.zxing:core:3.4.1'
implementation 'com.google.zxing:zxing-android:3.4.0'
```

Рисунок В.11 – Підключення бібліотеки ZXing

```
public class QRCodeGenerator {

    public static void main(String[] args) {
        String data = "Hello, QR Code with Reed-Solomon error correction!";
        String filePath = "path/to/qr_code.png";

        try {
            generateQRCode(data, filePath);
            System.out.println("QR Code generated successfully.");
        } catch (IOException e) {
            System.err.println("Error generating QR Code: " + e.getMessage());
        }
    }
```

Рисунок В.12 – Клас «QRCodeGenerator» та сповіщення

```
private static void generateQRCode(String data, String filePath) throws IOException {
        int size = 300; // Розмір QR-коду
        String fileType = "png";
```

Рисунок В.13 – Змінна «generateQRCode»

```
Map<EncodeHintType, Object> hintMap = new HashMap<>();
hintMap.put(EncodeHintType.ERROR_CORRECTION, ErrorCorrectionLevel.H); // Вибір рівня корекції помилок

QRCodeWriter qrCodeWriter = new QRCodeWriter();
try {
        BitMatrix bitMatrix = qrCodeWriter.encode(data, BarcodeFormat.QR_CODE, size, size, hintMap);

        BufferedImage image = new BufferedImage(size, size, BufferedImage.TYPE_INT_RGB);
        for (int x = 0; x < size; x++) {
                for (int y = 0; y < size; y++) {
                        image.setRGB(x, y, bitMatrix.get(x, y) ? 0xFF000000 : 0xFFFFFFFF);
                }
        }
```

Рисунок В.14 – Цикл для генерації QR-коду, а також перевірки та корекції помилок

```
                File qrCodeFile = new File(filePath);
                ImageIO.write(image, fileType, qrCodeFile);
        } catch (WriterException e) {
                throw new IOException("Error generating QR Code", e);
        }
```

Рисунок В.15 – Сповіщення про помилку генерації QR-коду

```
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import com.google.zxing.integration.android.IntentIntegrator;
import com.google.zxing.integration.android.IntentResult;

no usages
public class QRCodeScannerActivity extends AppCompatActivity {

    no usages  2 related problems
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Запускаємо сканер QR-кодів
        IntentIntegrator integrator = new IntentIntegrator(this);
        integrator.setOrientationLocked(false); // Розблокувати орієнтацію
        integrator.initiateScan();
    }
```

Рисунок В.16 – Підготовка до сканування та запуск сканеру

```
no usages
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    // Обробка результатів сканування QR-коду
    IntentResult result = IntentIntegrator.parseActivityResult(requestCode, resultCode, data);
    if (result != null) {
        if (result.getContents() == null) {
            // Сканування скасовано
            Log.d("QRCodeScannerActivity", "Scan cancelled");
        } else {
            // Отримано дані з QR-коду
            String scannedData = result.getContents();
            Log.d("QRCodeScannerActivity", "Scanned data: " + scannedData);
```

Рисунок В.17 – Створено обробник результатів сканування QR-коду.

```
import org.opencv.android.CameraBridgeViewBase.CvCameraViewFrame;
import org.opencv.android.CameraBridgeViewBase.CvCameraViewFrame;
import org.opencv.android.CameraBridgeViewBase.CvCameraViewFrame;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.core.Size;
import org.opencv.imgproc.Imgproc;

no usages
public class CameraActivity extends AppCompatActivity implements CameraBridgeViewBase.CvCameraViewFrame {

    4 usages
    private Mat mIntermediateMat;
```

Рисунок В.18 – Підготовка бібліотеки OpenCV до роботи

```
    @Override
    public Mat rgba() {
        // Викликається камерою для отримання кадру

        // Приклад застосування медіального фільтру
        Imgproc.medianBlur(mIntermediateMat, mIntermediateMat, 3);

        return mIntermediateMat;
    }

    no usages  2 related problems
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_camera);

        // Ініціалізація камери OpenCV
        mOpenCvCameraView = findViewById(R.id.camera_view);
        mOpenCvCameraView.setVisibility(CameraBridgeViewBase.VISIBLE);
        mOpenCvCameraView.setCvCameraViewListener(this);
    }
```

Рисунок В.19 – Підготовка до використання медіального фільтру

```
@Override
public void onCameraFrame(CvCameraViewFrame inputFrame) {
    // Викликається камерою для обробки кожного кадру
    mIntermediateMat = inputFrame.gray();
}

no usages
@Override
public void onResume() {
    super.onResume();
    if (!OpenCVLoader.initDebug()) {
        Log.e(TAG, "OpenCV initialization failed.");
    } else {
        Log.d(TAG, "OpenCV initialization succeeded.");
    }

    mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
}
```

Рисунок В.20 – Обробка результатів використання медіального фільтра

Додаток Г. Лістинг програми

Модуль вікна авторизації

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class AuthorizationWindow extends JFrame {

    public AuthorizationWindow() {
        // Заголовок вікна
        setTitle("Authorization");

        // Панель для розміщення компонентів
        JPanel panel = new JPanel();
        panel.setLayout(new GridLayout(3, 2));

        // Лейбли та тексти для вводу
        JLabel usernameLabel = new JLabel("Username:");
        JTextField usernameTextField = new JTextField();

        JLabel passwordLabel = new JLabel("Password:");
        JPasswordField passwordField = new JPasswordField();

        // Кнопки
        JButton exitButton = new JButton("Exit");
        JButton signinButton = new JButton("Signin");

        // Додавання компонентів на панель
        panel.add(usernameLabel);
        panel.add(usernameTextField);
        panel.add(passwordLabel);
        panel.add(passwordField);
        panel.add(exitButton);
        panel.add(signinButton);

        // Додавання обробників подій для кнопок
        exitButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
```

```
            System.exit(0); // Закриття програми при натисканні кнопки "Exit"
        }
    });

    signinButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            // Отримання значень з полів для вводу
            String username = usernameTextField.getText();
            char[] password = passwordField.getPassword();

            JOptionPane.showMessageDialog(AuthorizationWindow.this,
                    "Username: " + username + "\nPassword: " + new String(password),
                    "Authorization Successful",
JOptionPane.INFORMATION_MESSAGE);
        }
    });

    // Додавання панелі на вікно
    getContentPane().add(panel);

    // Розмір вікна та операция закриття
    setSize(300, 150);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    // Встановлення видимості вікна
    setVisible(true);
  }

  public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            new AuthorizationWindow();
        }
    });
  }
}
```

Модуль головного вікна програми

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class MainWindow extends JFrame {
    public MainWindow() {
        // Заголовок вікна
        setTitle("Main Window");

        // Зображення логотипу
        ImageIcon logoIcon = new ImageIcon("D:\\Диплом Магістр\\logo.png");
        JLabel logoLabel = new JLabel(logoIcon);

        // Панель для розміщення компонентів
        JPanel panel = new JPanel();
        panel.setLayout(new FlowLayout(FlowLayout.LEFT));

        // Кнопка "Delete"
        JButton deleteButton = new JButton("Delete");
        deleteButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Додайте код для обробки події кнопки "Delete" тут
                JOptionPane.showMessageDialog(MainWindow.this, "Delete button
clicked");
            }
        });

        // Кнопка "Add"
        JButton addButton = new JButton("Add");
        addButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Додайте код для обробки події кнопки "Add" тут
                JOptionPane.showMessageDialog(MainWindow.this, "Add button
clicked");
            }
        });

        // Кнопка "OK"
        JButton okButton = new JButton("OK");
        okButton.addActionListener(new ActionListener() {
            @Override
```

```
            public void actionPerformed(ActionEvent e) {
                // Додайте код для обробки події кнопки "OK" тут
                JOptionPane.showMessageDialog(MainWindow.this, "OK button
clicked");
            }
        });

        // Додавання компонентів на панель
        panel.add(logoLabel);
        panel.add(deleteButton);
        panel.add(addButton);
        panel.add(okButton);

        // Додавання панелі на вікно
        getContentPane().add(panel);

        // Розмір вікна та операция закриття
        setSize(600, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Встановлення видимості вікна
        setVisible(true);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                new MainWindow();
            }
        });
    }
}
```

Модуль інтеграції бази даних

```
import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;
```

```
public class DatabaseIntegration {
    public static void main(String[] args) {
        // Змінні для підключення до бази даних
        String jdbcURL = "jdbc:mysql://localhost:3306/your_database_name";
        String username = "your_username";
        String password = "your_password";

        // Об'єкти для роботи з базою даних
        Connection connection = null;
        Statement statement = null;
        ResultSet resultSet = null;

        try {
            // Завантаження драйвера JDBC
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Підключення до бази даних
            connection = DriverManager.getConnection(jdbcURL, username, password);

            // Створення об'єкта Statement для виконання запитів
            statement = connection.createStatement();

            // Виконання SQL-запиту SELECT
            String sqlQuery = "SELECT * FROM your_table_name";
            resultSet = statement.executeQuery(sqlQuery);

            // Обробка результатів запиту
            while (resultSet.next()) {
                int id = resultSet.getInt("id");
                String name = resultSet.getString("name");
```

```
            // Опрацювання інших полів з бази даних
            System.out.println("ID: " + id + ", Name: " + name);
        }
    } catch (SQLException | ClassNotFoundException e) {
        e.printStackTrace();
    } finally {
        // Закриття ресурсів
        try {
            if (resultSet != null) resultSet.close();
            if (statement != null) statement.close();
            if (connection != null) connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Модуль генерації QR-кодів

```
import com.google.zxing.BarcodeFormat;
import com.google.zxing.EncodeHintType;
import com.google.zxing.WriterException;
import com.google.zxing.common.BitMatrix;
import com.google.zxing.qrcode.QRCodeWriter;
import com.google.zxing.qrcode.decoder.ErrorCorrectionLevel;

import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

public class QRCodeGenerator {

    public static void main(String[] args) {
```

```
        String data = "Hello, QR Code with Reed-Solomon error correction!";
        String filePath = "path/to/qr_code.png";

        try {
            generateQRCode(data, filePath);
            System.out.println("QR Code generated successfully.");
        } catch (IOException e) {
            System.err.println("Error generating QR Code: " + e.getMessage());
        }
    }

    private static void generateQRCode(String data, String filePath) throws
IOException {
        int size = 300; // Розмір QR-коду
        String fileType = "png";

        Map<EncodeHintType, Object> hintMap = new HashMap<>();
        hintMap.put(EncodeHintType.ERROR_CORRECTION,
ErrorCorrectionLevel.H); // Вибір рівня корекції помилок

        QRCodeWriter qrCodeWriter = new QRCodeWriter();
        try {
            BitMatrix bitMatrix = qrCodeWriter.encode(data,
BarcodeFormat.QR_CODE, size, size, hintMap);

            BufferedImage image = new BufferedImage(size, size,
BufferedImage.TYPE_INT_RGB);
            for (int x = 0; x < size; x++) {
                for (int y = 0; y < size; y++) {
                    image.setRGB(x, y, bitMatrix.get(x, y) ? 0xFF000000 :
0xFFFFFFFF);
                }
            }

            File qrCodeFile = new File(filePath);
            ImageIO.write(image, fileType, qrCodeFile);
        } catch (WriterException e) {
            throw new IOException("Error generating QR Code", e);
        }
    }
}
```

Модуль сканеру QR-кодів

```
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import com.google.zxing.integration.android.IntentIntegrator;
import com.google.zxing.integration.android.IntentResult;

public class QRCodeScannerActivity extends AppCompatActivity {

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Запускаємо сканер QR-кодів
        IntentIntegrator integrator = new IntentIntegrator(this);
        integrator.setOrientationLocked(false); // Розблокувати орієнтацію
        integrator.initiateScan();
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent
data) {
        super.onActivityResult(requestCode, resultCode, data);

        // Обробка результатів сканування QR-коду
        IntentResult result = IntentIntegrator.parseActivityResult(requestCode,
resultCode, data);
        if (result != null) {
            if (result.getContents() == null) {
                // Сканування скасовано
                Log.d("QRCodeScannerActivity", "Scan cancelled");
            } else {
                // Отримано дані з QR-коду
                String scannedData = result.getContents();
                Log.d("QRCodeScannerActivity", "Scanned data: " + scannedData);

            }
        } else {
            super.onActivityResult(requestCode, resultCode, data);
        }
    }
}
```

Модуль підвищення якості зображення

```
import org.opencv.android.CameraBridgeViewBase.CvCameraViewFrame;
import org.opencv.android.CameraBridgeViewBase.CvCameraViewFrame;
import org.opencv.android.CameraBridgeViewBase.CvCameraViewFrame;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.core.Size;
import org.opencv.imgproc.Imgproc;

public class CameraActivity extends AppCompatActivity implements
CameraBridgeViewBase.CvCameraViewFrame {

    private Mat mIntermediateMat;

    @Override
    public Mat rgba() {
        // Викликається камерою для отримання кадру

        // Приклад застосування медіального фільтру
        Imgproc.medianBlur(mIntermediateMat, mIntermediateMat, 3);

        return mIntermediateMat;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_camera);

        // Ініціалізація камери OpenCV
        mOpenCvCameraView = findViewById(R.id.camera_view);
        mOpenCvCameraView.setVisibility(CameraBridgeViewBase.VISIBLE);
        mOpenCvCameraView.setCvCameraViewListener(this);
    }

    @Override
    public void onCameraFrame(CvCameraViewFrame inputFrame) {
        // Викликається камерою для обробки кожного кадру
        mIntermediateMat = inputFrame.gray();
    }

    @Override
    public void onResume() {
        super.onResume();
        if (!OpenCVLoader.initDebug()) {
```

```
            Log.e(TAG, "OpenCV initialization failed.");
        } else {
            Log.d(TAG, "OpenCV initialization succeeded.");
        }

        mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
    }

    @Override
    public void onPause() {
        super.onPause();
        if (mOpenCvCameraView != null) {
            mOpenCvCameraView.disableView();
        }
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        if (mOpenCvCameraView != null) {
            mOpenCvCameraView.disableView();
        }
     }
}
```

Додаток Д. Ілюстративний матеріал

ІЛЮСТРАТИВНИЙ МАТЕРІАЛ

РОЗРОБКА МЕТОДУ ТА ПРОГРАМНОГО ЗАСОБУ ДЛЯ АВТОМАТИЗАЦІЇ ВЕДЕННЯ СКЛАДСЬКОГО ОБЛІКУ

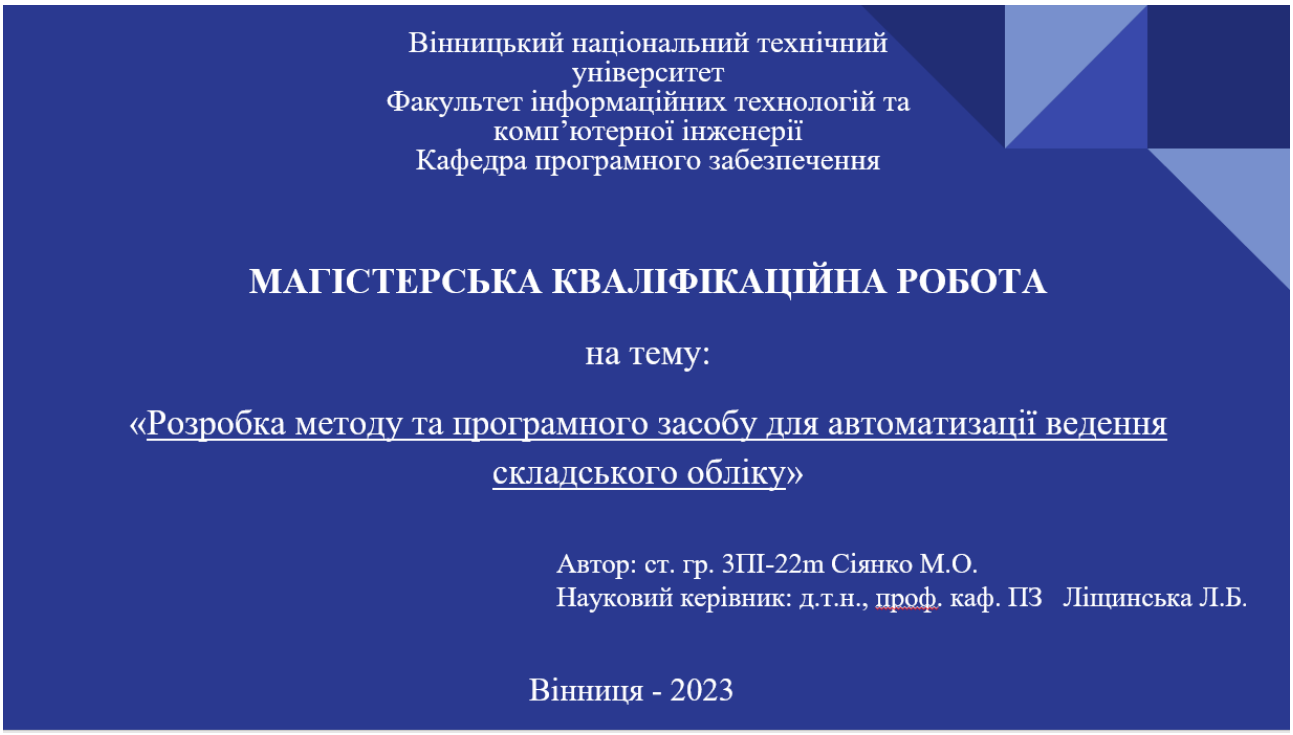

Рисунок В.1 – Титульний слайд

Рисунок В.2 – Актуальність теми

Мета, об’єкт та предмет дослідження

Мета. Метою дослідження є розробка методу та програмного засобу для автоматизації ведення складського обліку

Об’єкт дослідження – процеси автоматизації складського обліку під час використання додатку.

Предмет дослідження – методи та засоби автоматизації складського обліку.

Рисунок В.3 – Мета, об'єкт та предмет дослідження

Задачі дослідження

Основними задачами дослідження є:

- провести аналіз існуючих методів і засобів автоматизації складського обліку;
- розробити метод запису даних у базу даних системи;
- розробити метод створення QR-кодів для розпізнання у системі;
- розробити програмні компоненти та систему автоматизації складського обліку;
- провести тестування додатку.

Рисунок В.4 – Задачі дослідження

Наукова новизна отриманих результатів

- Подальшого розвитку набув метод генерації QR-кодів, в якому, на відміну від існуючих було використано алгоритм Reed-Solomon для швидкого виявлення та виправлення помилок, що дозволило підвищити ефективність зберігання та передачі даних.
- Подальшого розвитку набув метод зчитування QR-кодів, в якому, на відміну від класичного для підвищення якості зображення було використано медіальний фільтр, що дало можливість підвищити точність зчитування даних.

Рисунок В.5 – Наукова новизна

Практична цінність одержаних результатів

Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби для автоматизації складського обліку.

Рисунок В.6 – Практична цінність одержаних результатів

Порівняльний аналіз аналогів

Критерій	УкрСклад	Торгсофт	Dilovod	RemOnline	Auto-storage
Масштабність і спроможність масштабуватися	1	0,5	0,5	1	1
Зручність в роботі та інтуїтивний інтерфейс	0	0,5	1	0	0
Можливість роботи офлайн	1	0	0	1	1
Інтеграція з іншими системами	1	1	1	1	0
Сумісність з обладнанням	0	0	0	1	1
Геолокація та маршрутизація	0	0	0	0	0
Підсумок	3	2	2,5	4	5

Рисунок В.7 – Порівняльний аналіз аналогів

Графічна схема інтерфейсу головного вікна

- Область меню, що містить логотип та кнопки: згорнути, розгорнути та закрити;
- Список таблиць;
- Кнопка «Delete»;
- Кнопка «Add»;
- Кнопка «ОК»;
- Робоча область тобто екран самого додатку.

Рисунок В.8 – Графічна схема інтерфейсу головного вікна

Рисунок В.9 – Графічна схема вікна QR-коду

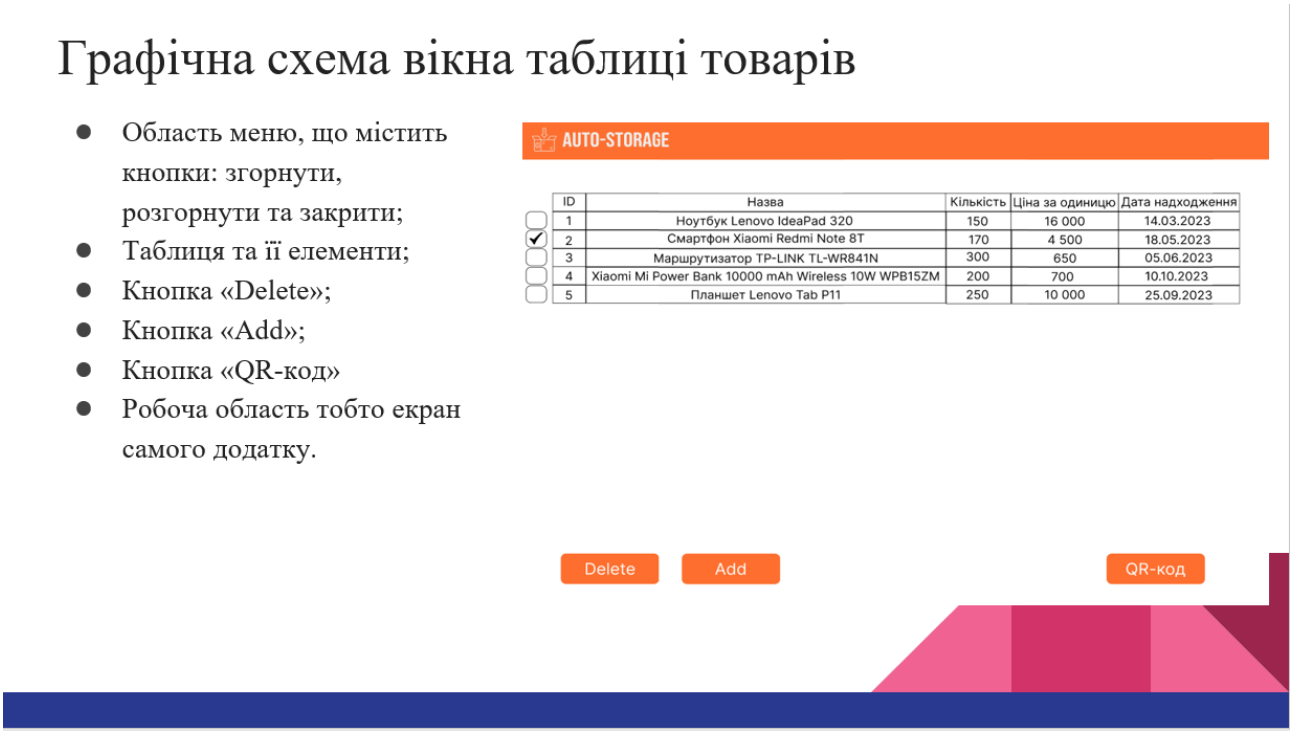

Рисунок В.10 – Графічна схема вікна таблиці товарів

Графічна схема мобільного додатку сканера

- Системна панель телефону;
- Кнопка «Сканувати»;
- Віконце камери для сканування;
- Робоча область тобто екран додатку.

Рисунок В.11 – Графічна схема мобільного додатку сканера

Інтеграція бази даних

- Працівник – це сутність уособлює операторі системи, який буде взаємодіяти з її функціями.
- Накладна – це сутність, що являє собою основний документ при транспортуванні вантажів, який регулює відносини між перевізником, відправником та одержувачем вантажу; оформляє та засвідчує договір перевезення вантажу.
- Накладна – це сутність того самого документу накладної, але вона містить у собі інформацію Про кількість та ціну за одиницю товару.
- Товар – це сутність що представляє собою будь-який товар що може бути придбано або переміщено на склад.
- Склад – це сутність яка уособлює собою будь-яке приміщення що було позначено як складське.
- Товар на складі – це сутність що являє собою будь-який товар що може бути розміщений на складі.

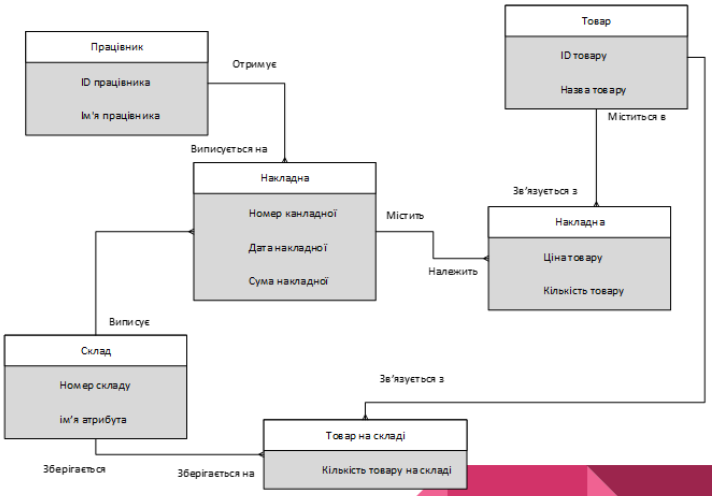

Рисунок В.12 – Інтеграція бази даних

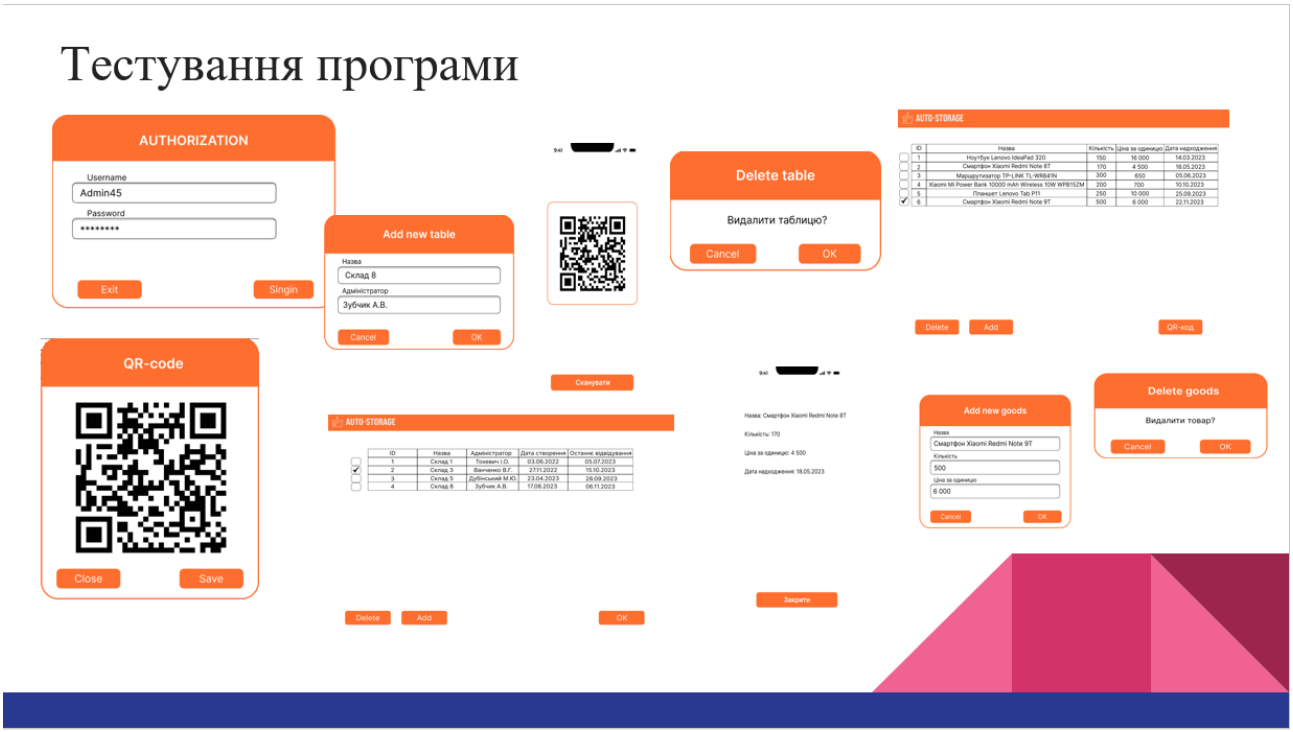

Рисунок В.13 - Тестування програми

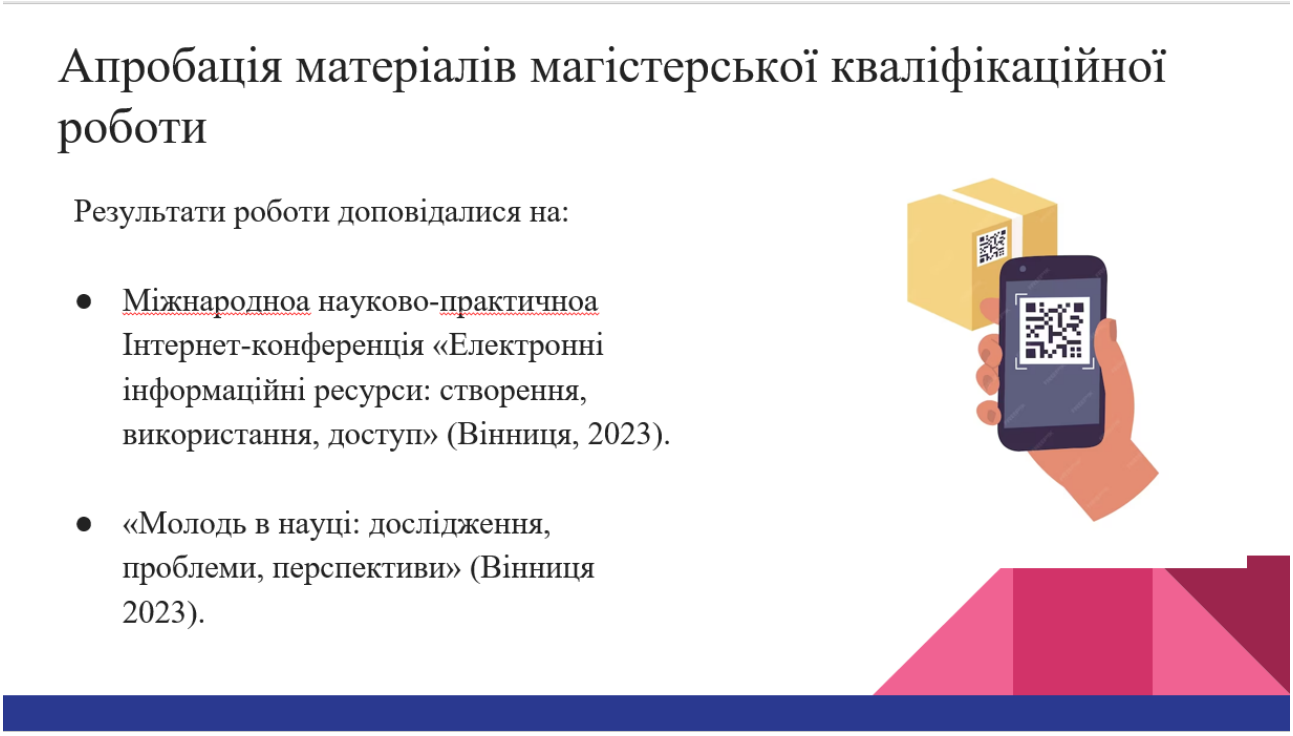

Рисунок В.14 – Апробація матеріалів магістерської кваліфікаційної роботи

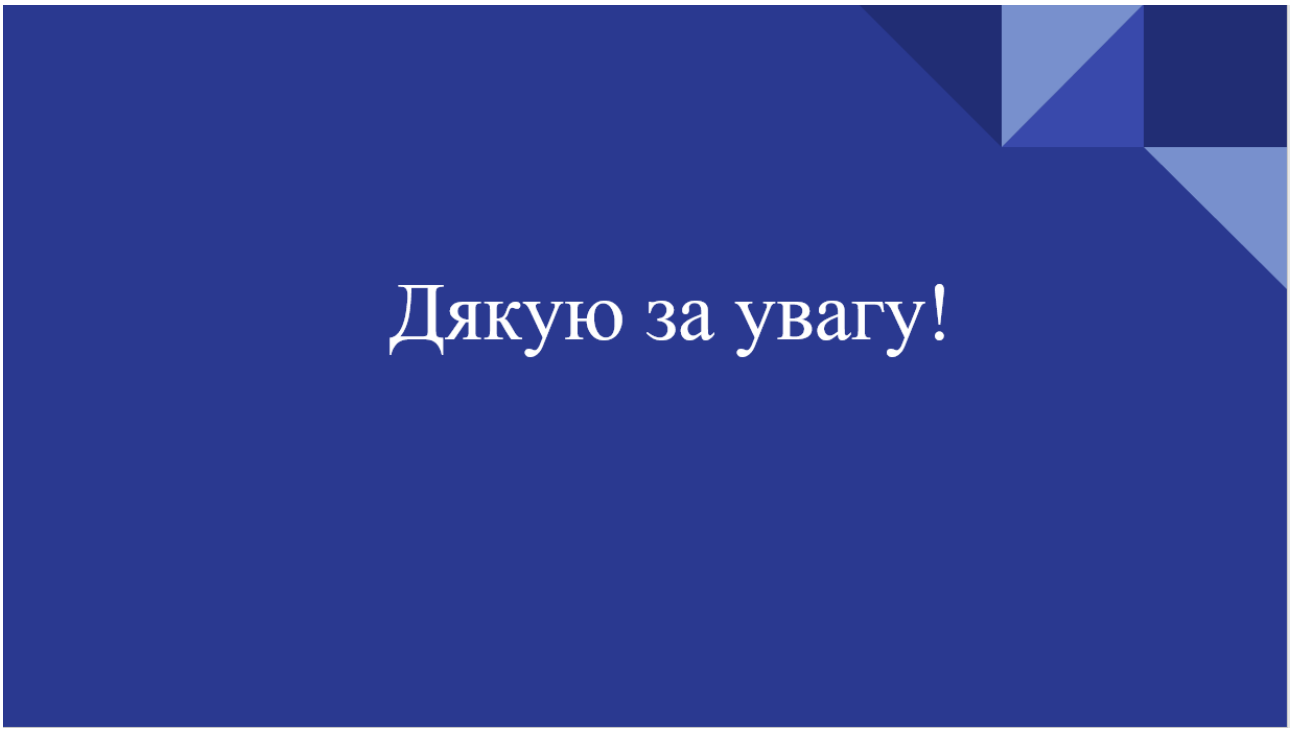

Рисунок В.15 – Фінальний слайд