Центральноукраїнський національний технічний університет
Центр заочної та дистанційної освіти
Кафедра кібербезпеки та програмного забезпечення

“Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 20 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві на основі методів штучного інтелекту”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-22Мз
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Михайленко Г.В.
«____» _____ 20 р.

Керівник проекту
кандидат технічних наук, ст. викладач
_____ Буравченко К.О.
«____» _____ 20 р.
Рецензент _____
_____

м.Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Центр заочної та дистанційної освіти*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань 12 "Інформаційні технології"

Спеціальність *122* "Комп'ютерні науки"

Освітньо-професійна (освітньо-наукова) програма "Комп'ютерні науки"

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2023 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Михайленку Георгію Васильовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та програмна реалізація системи автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві на основі методів штучного інтелекту
2. Керівник роботи Буравченко Костянтин Олегович, канд. техн. наук
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу №37-13 від 04.08.2023 року

3. Строк подання студентом роботи до захисту 10.12.2023 р.
4. Мета та завдання випускної кваліфікаційної роботи Метою роботи є дослідження та програмна реалізація системи автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві на основі методів штучного інтелекту
5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.
2. Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Наукова новизна
7. Економічна ефективність розробленої програми.
8. Заходи з охорони праці та техніки безпеки.
9. Висновки.

6. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	1 аркуш
Структурна схема системи	1 аркуш
Функціональна схема системи	1 аркуш
Блок-схема алгоритму роботи додатку	2 аркуша
Діаграма процесів	1 аркуш
Показники економічної ефективності	1 аркуш

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2023	14.11.2023
Охорона праці	Оришака О.В.	06.10.2023	16.11.2023

8. Дата видачі завдання « 6 » вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2023 р.	
3.	Розробка моделі компонента	20.10.2023 р.	
4.	Розробка структур даних	25.10.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2023 р.	
6.	Програмування алгоритмів	10.11.2023 р.	
7.	Розрахунок економічної ефективності	13.11.2023 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2023 р.	
9.	Оформлення ПЗ	17.11.2023 р.	
10.	Попередній захист роботи	10.12.2023 р.	

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника
_____ Буравченко К.О.
(прізвище та ініціали)
Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача
_____ Михайленко Г.В.
(прізвище та ініціали)

АНОТАЦІЯ

Михайленко Г.В. Дослідження та програмна реалізація системи автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві на основі методів штучного інтелекту. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі магістра розроблена система автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві на основі методів штучного інтелекту.

Метою розробки є дослідження та програмна реалізація системи автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві на основі методів штучного інтелекту.

Об'єктом дослідження є процес автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві.

Предметом дослідження є методи штучного інтелекту для автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві.

Методи дослідження базуються на методах штучного інтелекту, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи — програмна реалізація системи автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві на основі методів штучного інтелекту.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows 10/11. Програму розроблено в середовищі Python.

Ключові слова: комп'ютерні науки, штучний інтелект, автоматизація

ABSTRACT

Mykhailenko H.V. Research and software implementation of the system of automated inventory management and service maintenance at the enterprise based on artificial intelligence methods. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this master's thesis, a system of automated inventory management and service maintenance at the enterprise was developed based on artificial intelligence methods.

The goal of the development is the research and software implementation of the system of automated inventory management and service maintenance at the enterprise based on artificial intelligence methods.

The object of the study is the process of automated inventory management and service maintenance at the enterprise.

The subject of the study is artificial intelligence methods for automated inventory management and service maintenance at the enterprise.

Research methods are based on artificial intelligence methods, methods of mathematical statistics, and methods of software development.

The result of the work is the software implementation of the system of automated inventory management and service maintenance at the enterprise based on artificial intelligence methods.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS. The program was developed in the Python environment.

Keywords: computer science, artificial intelligence, automation

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БД — база даних

ОС — операційна система

СКБД — система керування базами даних

ШІ — штучний інтелект

ВСТУП

Актуальність теми. Актуальність даної теми випускної кваліфікаційної роботи полягає в необхідності покращення ефективності та конкурентоспроможності підприємств, що спеціалізуються на продажу, ремонті комп'ютерної техніки та збиранні комп'ютерів на замовлення. Завдання полягає в створенні і впровадженні системи автоматизованого управління інвентарем та сервісним обслуговуванням на основі методів штучного інтелекту, що дозволить оптимізувати процеси управління та підвищити якість обслуговування клієнтів. Реалізація цього проекту є важливою для підтримання конкурентоспроможності підприємства та задоволення потреб сучасного ринку комп'ютерної техніки.

Мета і задачі дослідження. Метою даної випускної кваліфікаційної роботи є створення та програмна реалізація системи автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві на основі методів штучного інтелекту.

Для досягнення цієї мети передбачені такі задачі дослідження:

– Розгляд і аналіз сучасних методів та технологій управління інвентарем на підприємствах, що спеціалізуються на продажу, ремонті комп'ютерної техніки та збиранні комп'ютерів на замовлення.

– Розробка алгоритмів та методів для автоматизованого контролю за інвентарем та сервісним обслуговуванням з використанням штучного інтелекту.

– Створення програмної реалізації системи автоматизованого управління інвентарем та сервісним обслуговуванням.

– Тестування та експериментальна перевірка розробленої системи на підприємстві.

– Оцінка ефективності та визначення впливу системи на показники управління і обслуговування.

Ці задачі спрямовані на створення системи, яка сприятиме оптимізації управління і підвищенню якості обслуговування на підприємстві, що спеціалізується на продажу, ремонті комп'ютерної техніки та збиранні комп'ютерів на замовлення, з використанням передових методів штучного інтелекту.

Об'єктом дослідження є процес автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві.

Предметом дослідження є методи штучного інтелекту для автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві.

Методи дослідження базуються на методах штучного інтелекту, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна одержаних результатів. Наукова новизна отриманих результатів полягає в наступному:

1. Використано методи штучного інтелекту. Алгоритми враховують кількість товарів, які були замовлені і надають рекомендації щодо подальших дій підприємства.

2. Покращено ефективність та точність. Завдяки використанню методів ШІ, система може аналізувати великий обсяг даних та приймати рішення в реальному часі.

Практичне значення одержаних результатів. Отримані результати та розроблена система автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві на основі методів штучного інтелекту мають велике практичне значення і можуть бути використані наступними способами:

– Підвищення ефективності підприємства — розроблена система дозволяє оптимізувати управління інвентарем та сервісним обслуговуванням, що призводить до підвищення продуктивності та зниження витрат. Це може значно покращити фінансові показники підприємства.

– Покращення обслуговування клієнтів — завдяки системі можна планувати та надавати послуги клієнтам більш якісно та оперативно, що сприяє задоволеності клієнтів та збільшенню їх лояльності.

– Оптимізація управління запасами — система допомагає підприємству управляти запасами інвентарю більш ефективно, уникати зайвих запасів та зменшувати витрати на їх зберігання.

– Впровадження сучасних технологій — розроблена система використовує сучасні методи штучного інтелекту, що дозволяє підприємству бути на крок попереду у конкурентному середовищі.

– Можливість масштабування — система розроблена з урахуванням можливості масштабування та адаптації до різних типів підприємств, що робить її універсальним інструментом для вдосконалення управління на різних рівнях.

Отже, розроблена система має велике практичне застосування та готова до впровадження на підприємствах, що спеціалізуються на продажу, ремонті комп’ютерної техніки та збиранні комп’ютерів на замовлення, та може значно покращити їхню ефективність та конкурентоспроможність.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Призначення системи автоматизованого управління інвентарем та сервісним обслуговуванням, розробленої в рамках даної магістерської роботи, полягає у впровадженні інноваційного підходу до управління ресурсами та обслуговуванням на підприємстві, що спеціалізується на продажу, ремонті комп'ютерної техніки та збиранні комп'ютерів на замовлення.

Основні завдання та функції системи включають в себе наступне:

– Моніторинг інвентарю — система забезпечує постійний моніторинг наявності різних видів інвентарю на підприємстві, включаючи комп'ютерну техніку, комплектуючі, програмне забезпечення та інші ресурси.

– Планування обслуговування — система дозволяє автоматично планувати обслуговування та технічне обслуговування обладнання, враховуючи технічні характеристики та історію використання.

– Оптимізація запасів — система допомагає управляти запасами інвентарю, забезпечуючи їх належну рівновагу між недостачею та надлишком, що дозволяє зменшити витрати на зберігання.

– Прогнозування потреб — використовуючи методи штучного інтелекту, система допомагає прогнозувати майбутні потреби у ресурсах та обслуговуванні, що спрямоване на попередження можливих проблем.

– Аналітика та звітність — система забезпечує можливість аналізу даних про інвентар, підтримує створення звітів та надає корисну інформацію для прийняття управлінських рішень.

– Підвищення якості обслуговування — штучний інтелект в системі допомагає підприємству реагувати на проблеми та виправляти їх до їхнього виникнення, що сприяє підвищенню якості обслуговування клієнтів.

– Масштабованість — система розроблена з урахуванням можливості масштабування та адаптації до різних типів підприємств, незалежно від їхнього розміру та специфіки.

Призначення системи полягає у впровадженні сучасних технологій для оптимізації управління ресурсами та покращення обслуговування клієнтів на підприємстві.

1.2 Область застосування

Область застосування розробленої системи автоматизованого управління інвентарем та сервісним обслуговуванням на основі методів штучного інтелекту є досить широкою і охоплює різні сфери діяльності, де існує потреба у ефективному управлінні ресурсами та обслуговуванні. До основних областей застосування входять:

– Інформаційні та технічні компанії — система може бути використана для управління комп'ютерною технікою, програмним забезпеченням та запасами комплектуючих. Вона допоможе підприємствам підтримувати інвентар під контролем, планувати обслуговування обладнання та прогнозувати потреби.

– Роздрібна торгівля — для роздрібних магазинів, особливо тих, що спеціалізуються на продажу комп'ютерної техніки, система дозволить ефективно керувати запасами, стежити за рухом товару та оптимізувати процеси замовлення товарів.

– Сервісні компанії — сервісні підприємства, які надають послуги з ремонту та технічного обслуговування, зможуть використовувати систему для планування обслуговування клієнтів, керування запасами необхідних запчастин та ресурсами.

– Виробничі підприємства — для підприємств, які виробляють комп'ютерну техніку або комплектуючі, система допоможе забезпечити ефективний контроль за процесом виробництва, запасами сировини та готової продукції.

– Офіси та установи — навіть офіси та установи, де використовується комп'ютерна техніка та офісне обладнання, можуть вигідно впровадити систему для контролю за інвентарем, планування технічного обслуговування та витратами на ресурси.

– Онлайн-роздріб — інтернет-магазини та електронні платформи, що продають комп'ютерну техніку, можуть використовувати систему для керування складами, відстеження замовлень та прогнозування попиту.

– Освітні заклади — у навчальних закладах, де велика кількість комп’ютерної техніки та обладнання, система допоможе ефективно управляти активами, проводити технічне обслуговування та відслідковувати стан інвентарю.

– Медичні установи — у медичних закладах система може бути використана для керування обладнанням, ліками та медичними приладами, забезпечуючи їхню належну доступність.

В цілому, система має потенціал застосування в будь-якій галузі, де важливо забезпечити оптимальне управління інвентарем і покращити якість обслуговування, спираючись на сучасні методи штучного інтелекту.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень з профілю теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Існуючими рішеннями з профілю теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є:

GLPi (Gestionnaire Libre de Parc Informatique)

GLPi (Gestionnaire Libre de Parc Informatique) є відкритою системою управління IT-активами та сервісами. Ця програма призначена для ефективного управління та моніторингу IT-інфраструктури організації. Основні характеристики та можливості програми GLPi:

– Управління активами — GLPi дозволяє вести реєстр всіх активів, таких як комп'ютери, сервери, монітори, принтери, програмне забезпечення тощо. Є можливість додавати, видаляти та оновлювати записи про активи.

– Управління сервісами — GLPi допомагає створювати та відстежувати заявки на обслуговування, вирішувати проблеми та надавати підтримку користувачам. Можна призначати завдання, встановлювати терміни виконання та слідкувати за статусами заявок.

– Інтеграція з активами — GLPi може автоматично виявляти активи в мережі та додавати їх до реєстру. Це полегшує ведення актуального списку активів.

– Управління запитами на закупівлю — програма дозволяє створювати запити на закупівлю нового обладнання або програмного забезпечення, встановлювати бюджети та відстежувати статуси замовлень.

– Розширені звіти — GLPi надає можливість створювати різноманітні звіти та аналізи щодо активів, сервісів, витрат та інших аспектів управління IT-інфраструктурою.

– Модульність та розширення — GLPi можна легко розширити за допомогою плагінів та додаткових модулів, що дозволяє налаштовувати систему під конкретні потреби організації.

– Відкритий код — GLPi є вільним програмним забезпеченням з відкритим вихідним кодом, що робить його доступним для розробки та модифікації власними силами або залученням спільноти розробників.

Переваги:

– безкоштовнсть і відкритість;

– інтуїтивний інтерфейс користувача;

– можливість налаштування та надійність;

– легке керування запитами.

Недоліки:

– застарілий інтерфейс;

– низька швидкодія;

– складність налаштування.

GLPi є потужним інструментом для організацій, які потребують ефективного управління своєю IT-інфраструктурою, включаючи великі підприємства, установи та організації різного розміру. Він допомагає знижувати витрати, покращувати продуктивність та забезпечувати більший контроль над IT-ресурсами.

Snipe-IT

Snipe-IT — це відкрита система управління активами та інвентаризацією, призначена для організацій, які шукають ефективний спосіб відстежувати та керувати фізичними та програмними активами в їхній IT-інфраструктурі. Можливості Snipe-IT:

– Управління активами — Snipe-IT дозволяє зберігати інформацію про різні типи активів, включаючи комп’ютери, монітори, сервери, принтери, програмне забезпечення, аксесуари та інше. Кожен актив має свій унікальний ідентифікатор та докладну інформацію, таку як модель, серійний номер, стан, історію обслуговування і багато іншого.

– Інвентаризація — Snipe-IT допомагає створювати актуальну інвентаризацію обладнання та програмного забезпечення в реальному часі. Ви можете швидко виявляти, додавати та вилучати активи.

– Управління користувачами і ролями — програма надає можливість призначати користувачам різні ролі та права доступу до системи в залежності від їхніх функцій та відповідальностей.

– Управління гарантіями та обслуговуванням — Ви можете відстежувати гарантійні та післягарантійні періоди для кожного активу, планувати обслуговування та ремонти.

– Інтеграція з штрих-кодами та QR-кодами — Snipe-IT дозволяє використовувати штрих-коди та QR-коди для швидкого сканування та ідентифікації активів.

– Звіти та аналітика — Програма надає багатий набір звітів і аналітичних інструментів для аналізу даних про активи, витрати та інші аспекти управління активами.

– Мультиплатформеність і підтримка різних мов — Snipe-IT підтримується на різних платформах і доступний в різних мовах, що дозволяє використовувати його для глобальних організацій.

– Відкритий код і спільнота розробників — як вільне програмне забезпечення з відкритим вихідним кодом, Snipe-IT піддається розробці та розширенню спільнотою користувачів та розробників.

Переваги:

– безкоштовність і відкритість;

– інтуїтивний інтерфейс користувача;

– легкість відстеження витрат;

– наявність усіх необхідних вкладок у меню продуктів.

Недоліки:

– складність встановлення;

– відсутність інтеграцій з іншими сервісами.

Snipe-IT допомагає організаціям більш ефективно управляти своєю IT-інфраструктурою, підвищувати продуктивність та зменшувати витрати, шляхом оптимізації використання активів і забезпечення точного обліку.

Ralph (Ralph Asset Management System)

Ralph (Ralph Asset Management System) — це відкрита система управління активами та інвентаризацією, яка спеціалізується на управлінні інформацією про IT-активи, зокрема, обладнання, програмне забезпечення та інші ресурси, використовувані в інформаційних технологіях. Основні характеристики та можливості програми Ralph:

– Управління активами — Ralph дозволяє створювати докладну інвентаризацію фізичних та програмних активів. Це включає в себе сервери, комп'ютери, монітори, принтери, маршрутизатори, програмне забезпечення та інше обладнання.

– Сканування активів — програма надає можливість автоматично сканувати мережу для виявлення нових активів та оновлення інформації про них, що полегшує процес інвентаризації.

– Управління гарантіями та обслуговуванням — Ralph дозволяє відстежувати гарантійні та післягарантійні обслуговування для активів, заплановані ремонти та обслуговування.

– Управління закупівлями і бюджетуванням — Ви можете вести облік замовлень, бюджетів та закупівель обладнання та програмного забезпечення.

– Спеціалізовані звіти — Ralph надає розширений набір звітів і аналітичних інструментів для аналізу даних про активи, витрати та інші аспекти управління активами.

– Інтеграція з іншими системами — програма підтримує інтеграцію з іншими системами, такими як системи моніторингу, служби заявок та інші інструменти, що сприяє автоматизації процесів управління активами.

– Мультиплатформеність і розширення — Ralph може працювати на різних платформах та підтримує розширення для налаштування системи під конкретні потреби організації.

– Відкритий код — як вільне програмне забезпечення з відкритим вихідним кодом, Ralph дозволяє розробникам та спільноті вносити зміни та вдосконалення в програму.

Переваги:

– безкоштовність і відкритість.

Недоліки:

– відсутність комерційної підтримки;

– низька активність спільноти;

– мало інформації на інших джерелах.

Ralph допомагає організаціям більш ефективно управляти своєю IT-інфраструктурою, зменшувати витрати, оптимізувати використання активів та забезпечувати точний облік ресурсів. Ця програма особливо корисна для великих компаній та організацій, де активи IT відіграють важливу роль в щоденних операціях.

Lansweeper

Lansweeper — це інтегрована система інвентаризації та управління активами, розроблена для моніторингу, аудиту та управління IT-інфраструктурою організацій. Ця програма спроєктована для допомоги адміністраторам мережі та відділам IT у веденні актуального обліку та керуванні фізичними та програмними активами. Основні характеристики та можливості програми Lansweeper:

– Інвентаризація активів — Lansweeper автоматично сканує всю мережу та збирає інформацію про фізичні та програмні активи, включаючи комп’ютери, сервери, монітори, принтери, маршрутизатори, програмне забезпечення і багато іншого.

– Сканування активів за допомогою агентів і без агентів — Ви можете вибирати між різними методами сканування, включаючи встановлення агентів на пристроях або безпечне зовнішнє сканування.

– Управління обладнанням та програмним забезпеченням — програма дозволяє докладно відстежувати характеристики обладнання, включаючи серійні номери, конфігурацію, стан обладнання та версії програмного забезпечення.

– Управління гарантіями та обслуговуванням — Lansweeper надає інформацію про гарантійні та післягарантійні обслуговування активів та надає можливість створювати регулярні завдання обслуговування.

– Повідомлення та моніторинг стану — програма дозволяє налаштовувати сповіщення про події, такі як закінчення гарантії, низький рівень запасів тощо.

– Звіти і аналітика — Lansweeper надає багатий набір звітів та аналітичних інструментів для аналізу інформації про активи, витрати та інші аспекти управління активами.

– Інтеграція з іншими системами — програма має API, що дозволяє інтегрувати її з іншими системами моніторингу, заявок, обліку часу тощо.

– Віддалений доступ— Ви можете отримувати доступ до даних Lansweeper з будь-якого пристрою за допомогою веб-інтерфейсу.

Переваги:

– зручність встановлення та використання;

– може визначати обладнання у локальній мережі;

– якісна технічна підтримка.

Недоліки:

– пропрієтарна ліцензія;

– не завжди визначає все обладнання.

Lansweeper є потужним інструментом для організацій будь-якого розміру, які потребують точного обліку та керування своєю IT-інфраструктурою. Вона допомагає підвищити ефективність управління активами, зменшити витрати і забезпечити безпеку та надійність мережі.

Spiceworks

Spiceworks — це безкоштовна інтегрована платформа для управління IT-ресурсами та надання технічної підтримки, розроблена для IT-адміністраторів та професіоналів у сфері обслуговування. Ця програма має низку інструментів і функціональності, які спрощують керування IT-інфраструктурою та підтримку користувачів.

Ключові особливості та можливості програми Spiceworks:

– Інвентаризація активів — Spiceworks автоматично сканує вашу мережу для ідентифікації фізичних та програмних активів, таких як комп'ютери, сервери, маршрутизатори, принтери, програмне забезпечення тощо. Ви можете відстежувати їхню конфігурацію та зміни.

– Моніторинг мережі — програма надає можливість відстежувати стан мережевих пристроїв, виявляти відмови та проблеми, а також отримувати сповіщення про них.

– Управління заявками і допомогою користувачам — Spiceworks має систему керування заявками, що дозволяє користувачам надсилати запити на допомогу та адміністраторам вирішувати їх. Ви також можете створювати планові завдання та слідкувати за статусом обслуговування.

– Звіти і аналітика — програма надає засоби для створення різноманітних звітів та аналізу даних про вашу IT-інфраструктуру, що допомагає приймати рішення.

– Управління гарантіями та оновленнями — Spiceworks відстежує гарантійні та післягарантійні періоди для обладнання та дозволяє вам вчасно реагувати на оновлення та усунення недоліків.

– Інтеграція з різними інструментами — програма має можливість інтеграції з іншими системами, такими як Active Directory, Google Workspace (раніше G Suite), Microsoft 365 і багатьма іншими.

– Спільнота користувачів — Spiceworks має велику активну спільноту користувачів, яка надає підтримку та спільну роботу над рішеннями проблем.

Переваги:

– зручність встановлення;

– наявність мобільного додатка.

Недоліки:

– пропрієтарна ліцензія;

– наявність реклами.

Spiceworks є корисним інструментом для малих та середніх підприємств, організацій та IT-спеціалістів, які шукають швидкий і ефективний спосіб управління IT-інфраструктурою та надання технічної підтримки. Більшість функціональності доступна безкоштовно, що робить Spiceworks привабливим вибором для багатьох організацій.

Network Inventory Advisor

Network Inventory Advisor — це програмне забезпечення для інвентаризації мережевих ресурсів та управління інвентарем комп'ютерів і пристроїв в корпоративних мережах. Ця програма допомагає IT-адміністраторам та управлінцям зберігати актуальну та точну інформацію про обладнання і програмне забезпечення на мережі, спрощуючи процес управління та моніторингу. Докладніше про можливості та характеристики Network Inventory Advisor:

– Інвентаризація обладнання — програма автоматично сканує мережу для виявлення комп'ютерів, серверів, маршрутизаторів, принтерів, моніторів і іншого обладнання. Вона збирає докладну інформацію про кожен актив, включаючи модель, серійний номер, конфігурацію та багато іншого.

– Сканування програмного забезпечення — Network Inventory Advisor також аналізує встановлене програмне забезпечення на комп’ютерах та інших пристроях, надаючи перелік встановлених програм і їхні версії.

– Моніторинг змін — програма відстежує зміни в обладнанні та програмному забезпеченні та надає звіти про них, що дозволяє вчасно реагувати на оновлення, виходи з ладу або незадокументовані зміни.

– Звіти і аналітика — Network Inventory Advisor надає широкий спектр звітів і аналітичних інструментів для аналізу даних про активи, допомагаючи приймати рішення щодо обслуговування, оновлень і резервування.

– Розгалуження та фільтрація даних — Ви можете організовувати інформацію за допомогою різних фільтрів та категорій, що спрощує пошук інформації про конкретні активи.

– Безпека даних — для захисту конфіденційної інформації, програма має можливість авторизації та обмеження доступу для користувачів.

– Мультиплатформеність — Network Inventory Advisor сумісний з різними операційними системами і має версії для Windows і macOS.

Переваги:

– зручність використання;

– може визначати обладнання у локальній мережі.

Недоліки:

– пропрієтарна ліцензія;

– відсутність оновлень протягом останніх декількох років.

Це програмне забезпечення допомагає організаціям зберігати докладну та актуальну інформацію про свою IT-інфраструктуру, зменшуючи час і ресурси, які витрачаються на ручну інвентаризацію та моніторинг.

Service Creatio

Service Creatio — це інтегрована платформа для автоматизації та управління процесами обслуговування клієнтів (CRM) і управління сервісом (FSM). Ця програма розроблена для підвищення ефективності обслуговування клієнтів, оптимізації роботи з клієнтами та покращення оперативного управління послугами та ресурсами. Ключові можливості та характеристики програми Service Creatio:

– CRM-функції — Service Creatio надає багато інструментів для керування взаємодією з клієнтами, включаючи ведення історії спілкування, керування контактами та відстеження обліку клієнтів.

– Управління обслуговуванням — програма дозволяє створювати, відстежувати та призначати завдання на обслуговування клієнтів та ремонт обладнання. Вона також надає інструменти для планування маршрутів, розкладів та контролю над запасами.

– Автоматизація процесів — Service Creatio дозволяє автоматизувати багато рутинних операцій та процесів, що покращує продуктивність та точність роботи.

– Управління замовленнями — Ви можете створювати та відстежувати замовлення клієнтів, виконувати операції опрацювання замовлень та відстежувати їх статус.

– Звіти і аналітика — програма надає широкий спектр аналітичних інструментів та звітів для відстеження ефективності обслуговування клієнтів та оптимізації бізнес-процесів.

– Інтеграція — Service Creatio може інтегруватися з іншими системами та сервісами, такими як системи електронної пошти, телефонія, платіжні системи та інші.

– Портал клієнта — Ви можете надавати доступ клієнтам до порталу для відстеження статусу їхніх замовлень, запитів та обслуговування.

Service Creatio спрямований на підвищення якості обслуговування клієнтів та оптимізацію роботи з ними. Ця програма широко використовується в різних галузях, включаючи послуги, обслуговування обладнання, IT-обслуговування, фінансові послуги і багато інших.

ServiceNow

ServiceNow — це інтегрована платформа для управління послугами та цифрового робочого середовища, яка дозволяє організаціям автоматизувати та оптимізувати різні процеси, включаючи управління IT-сервісами, обслуговування клієнтів, управління проєктами, управління ризиками тощо. ServiceNow використовується великими корпораціями, урядовими установами і організаціями всіх розмірів для покращення продуктивності та забезпечення високого рівня обслуговування. Докладніше про можливості та характеристики програми ServiceNow:

– Управління IT-сервісами — ServiceNow допомагає організаціям ефективно керувати усіма аспектами IT-інфраструктури, включаючи сервіси, обладнання, програмне забезпечення та зміни. Це включає в себе створення та відстеження заявок на обслуговування, розгортання нового обладнання та оновлення програмного забезпечення.

– Управління обслуговуванням клієнтів (Customer Service Management) — платформа надає інструменти для забезпечення високої якості обслуговування клієнтів, включаючи системи заявок, онлайн-чати, бази знань та аналітику задоволеності клієнтів.

– Управління робочими процесами — ServiceNow дозволяє організаціям створювати, відстежувати та оптимізовувати бізнес-процеси, включаючи проєкти, операції та багато інших аспектів діяльності.

– Управління ризиками та забезпечення дотримання — платформа допомагає виявляти та керувати ризиками, а також забезпечує відповідність з правилами та регуляторними вимогами.

– Спільна робота і колаборація — ServiceNow надає інструменти для спільної роботи команд та внутрішнього обміну інформацією.

– Аналітика та звіти — платформа надає розширені можливості аналізу та створення звітів для прийняття обґрунтованих рішень.

– Інтеграція з іншими системами — ServiceNow може легко інтегруватися з іншими системами, що дозволяє об'єднати дані та процеси з різних джерел.

ServiceNow допомагає організаціям раціоналізувати та автоматизувати багато бізнес-процесів, що допомагає покращити продуктивність, знизити витрати і підвищити задоволеність клієнтів.

Total Network Inventory

Total Network Inventory — це програмне забезпечення для інвентаризації мережевих ресурсів та управління інвентарем комп'ютерів і пристроїв у корпоративних мережах. Ця програма дозволяє адміністраторам мережі та IT-професіоналам отримувати докладну інформацію про обладнання та програмне забезпечення, які використовуються в організації, для забезпечення ефективного управління та підтримки. Основні можливості та характеристики програми Total Network Inventory:

– Інвентаризація обладнання — Total Network Inventory автоматично сканує вашу мережу для виявлення комп'ютерів, серверів, маршрутизаторів,

принтерів і іншого обладнання. Вона збирає інформацію про обладнання, таку як модель, серійний номер, конфігурацію та статус.

– Сканування програмного забезпечення — програма аналізує встановлене програмне забезпечення на комп'ютерах і пристроях, надаючи перелік встановлених програм і їхні версії.

– Збереження інформації — отримана інформація про обладнання та програмне забезпечення зберігається в централізованій базі даних, що дозволяє легко відстежувати зміни та зберігати історію.

– Планування інвентаризації — Ви можете налаштовувати регулярні автоматичні сканування мережі для оновлення інформації про обладнання та програмне забезпечення.

– Звіти і аналітика — Total Network Inventory надає інструменти для створення різноманітних звітів і аналізу даних про вашу IT-інфраструктуру, що допомагає приймати рішення щодо обслуговування, оновлень і резервування.

– Безпека даних — для захисту конфіденційної інформації, програма надає можливість авторизації та обмеження доступу для користувачів.

Total Network Inventory допомагає зберігати актуальну та точну інформацію про вашу IT-інфраструктуру, що полегшує управління ресурсами та прийняття обґрунтованих рішень щодо обслуговування та розвитку вашої мережі.

EMCO Network Inventory

EMCO Network Inventory — це програмне забезпечення для інвентаризації і управління мережевими ресурсами та комп'ютерами у корпоративних мережах. Ця програма дозволяє адміністраторам мережі та IT-спеціалістам отримувати докладну інформацію про обладнання, програмне забезпечення та конфігурації в усіх пристроях в мережі для забезпечення ефективного управління та підтримки. Ключові можливості та характеристики програми EMCO Network Inventory:

– Інвентаризація обладнання — EMCO Network Inventory автоматично сканує вашу мережу для виявлення комп'ютерів, серверів, маршрутизаторів, принтерів та іншого обладнання. Вона збирає інформацію про обладнання, таку як модель, серійний номер, конфігурацію та статус.

– Сканування програмного забезпечення — програма аналізує встановлене програмне забезпечення на комп'ютерах і пристроях, надаючи перелік встановлених програм і їхні версії.

– Інформація про конфігурацію — EMCO Network Inventory дозволяє вам відстежувати конфігурацію пристроїв, включаючи характеристики процесора, обсяг оперативної пам'яті, жорсткого диску і багато інших параметрів.

– Відстеження змін — програма відстежує зміни в обладнанні та програмному забезпеченні і надає звіти про них, що допомагає вчасно реагувати на оновлення, виходи з ладу або незадокументовані зміни.

– Звіти і аналітика — EMCO Network Inventory надає інструменти для створення різних звітів та аналізу даних про вашу IT-інфраструктуру, допомагаючи приймати рішення щодо обслуговування, оновлень і резервування.

– Безпека даних — для захисту конфіденційної інформації, програма надає можливість авторизації та обмеження доступу для користувачів.

EMCO Network Inventory спрямований на підтримку актуальної та точної інформації про вашу мережу, що полегшує управління ресурсами та підтримку. Вона може бути корисною для великих і середніх організацій, які прагнуть оптимізувати управління мережею та ресурсами.

Обґрунтування необхідності розробки системи

Необхідність розробки системи автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві на основі методів штучного інтелекту (далі - система) відображається в ряді факторів, і зокрема, в аналізі існуючого програмного забезпечення на цю тему. Нижче представлені аргументи, що обґрунтовують необхідність розробки такої системи:

– Відсутність універсального рішення — наразі, існуюче програмне забезпечення, призначене для управління інвентарем та сервісним обслуговуванням, має свої обмеження і не завжди може задовольнити потреби конкретних підприємств, особливо тих, що спеціалізуються на продажу, ремонті комп’ютерної техніки та збиранні комп’ютерів на замовлення. Зазвичай, існуюче

програмне забезпечення не здатне забезпечити високу ступінь індивідуалізації для вирішення конкретних завдань підприємства.

– Швидкі та динамічні зміни у галузі — галузь комп'ютерної техніки і інформаційних технологій швидко розвивається, та вимагає постійного адаптування до нових вимог і технологій. Існуюче програмне забезпечення може бути застарілим та неспроможним ефективно впроваджувати інновації, в той час як власна система може бути розроблена з урахуванням сучасних підходів і технологій.

– Специфічні потреби підприємства — кожне підприємство має свої унікальні особливості та вимоги до управління інвентарем та сервісним обслуговуванням. Зазвичай, готові програмні продукти надають обмежені можливості для налаштування під конкретне підприємство. Розробка власної системи дозволить врахувати всі особливості та потреби підприємства.

– Ефективність і економічні вигоди — розробка власної системи може призвести до більш ефективного управління інвентарем та сервісним обслуговуванням, що в свою чергу призведе до зниження витрат, оптимізації процесів, підвищення якості обслуговування і, як результат, до збільшення прибутку підприємства.

– Розвиток сучасних технологій — використання методів штучного інтелекту в системах управління інвентарем та сервісним обслуговуванням є актуальним і дозволяє підприємствам підвищити ефективність, зробити прогнози та приймати стратегічні рішення.

Таким чином, наявність існуючого програмного забезпечення не завжди може задовольнити потреби підприємств у галузі продажу, ремонті комп’ютерної техніки та збиранні комп’ютерів на замовлення. Розробка власної системи на основі методів штучного інтелекту є доцільною для забезпечення більш точного, ефективного та індивідуалізованого управління інвентарем та сервісним обслуговуванням на підприємстві.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для побудови системи були використані наступні засоби:

Python

За допомогою цієї мови програмування створюється основна частина програмного продукту.

Дана мова програмування обрана з наступних причин:

– Простота та читабельність коду — Python має чистий і легко зрозумілий синтаксис, що робить його ідеальним вибором для початківців та професіоналів. Код на Python зазвичай коротший і більш зрозумілий, що полегшує спільну роботу над проєктами.

– Велика спільнота та багата екосистема — Python має велику та активну спільноту розробників, що означає, що завжди можливо знайти відповіді на свої питання, документацію та сторонні бібліотеки для вирішення різноманітних завдань.

– Можливості для різних видів розробки — Python використовується для веб-розробки, наукових обчислень, штучного інтелекту, аналізу даних, автоматизації завдань і багатьох інших областей. Ця універсальність робить його дуже потужним і гнучким інструментом.

– Переносимість — Python доступний для різних операційних систем, і код, написаний на Python, може бути перенесений з однієї платформи на іншу без значних зусиль.

– Безкоштовність та відкритий код — Python є вільним програмним забезпеченням, що означає, що його можна використовувати безкоштовно.

Flask

За допомогою цього фреймворку створюється основна частина програмного продукту.

Даний фреймворк був обраний з наступних причин:

– Простота використання — Flask має дуже простий та інтуїтивний синтаксис. Розробка веб-додатків на ньому вимагає менше коду порівняно з іншими фреймворками. Це особливо корисно для початківців.

– Мінімалізм — Flask — це «мікро»-фреймворк, що означає, що він надає лише необхідні базові функції. Ви можете додавати розширення (наприклад, SQLAlchemy для роботи з базою даних) за потреби, що робить його дуже гнучким.

– Підтримка розширень — Flask має широкий спектр розширень, які допомагають розвивати додаток, додаючи новий функціонал, такий як аутентифікація, обробка форм, робота з базою даних і багато іншого.

– RESTful підтримка — Flask легко інтегрується з RESTful API, що дозволяє створювати веб-служби та інші додатки, які використовують HTTP-запити для взаємодії.

– Добра документація — Flask має високоякісну документацію та активну спільноту користувачів. Ви знайдете багато прикладів і ресурсів для навчання.

– Спільнота і підтримка — Flask має велику та активну спільноту, яка готова допомогти з питаннями та проблемами. Є також багато розширень, розроблених спільнотою.

– Підтримка WSGI — Flask використовує WSGI (Web Server Gateway Interface), що дозволяє легко інтегрувати його з різними веб-серверами та хостинговими платформами.

– Можливість розгортання — Flask можна легко розгорнути на хмарних платформах, таких як Heroku, і на серверах з використанням різних веб-серверів, таких як Gunicorn або uWSGI.

SQLite

Ця програма використовується для зберігання даних, що оброблюються програмним продуктом, який розроблюється.

Даний рушій баз даних був обраний з наступних причин:

– Простота використання — SQLite — це легковаговий рушій баз даних, який не вимагає окремого сервера та налаштувань. Це робить його дуже простим у встановленні та використанні, і він підходить для проєктів будь-якого рівня складності.

– Наявність коду для роботи з SQLite у стандартній бібліотеці мови програмування Python — не потрібно встановлювати сторонні бібліотеки для початку роботи.

– Ефективність — SQLite добре працює для проєктів з невеликим обсягом даних або простими запитами. Він не вимагає значних ресурсів системи та добре підходить для мобільних додатків та інших вбудованих систем з обмеженими ресурсами.

– Транзакції та надійність — SQLite підтримує транзакції, що дозволяє забезпечити цілісність даних та виконання групових операцій з базою даних. Він також має механізми відновлення даних у випадку аварій та відмов, що робить його надійним.

– Підтримка для багатьох мов програмування — SQLite має бібліотеки для багатьох мов програмування, включаючи Python, Java, C++, інші. Це дозволяє використовувати SQLite в різних проєктах та на різних платформах.

draw.io/diagrams.net

Ця програма використовується для створення схем під час виконання наукової практики.

Дана програма була обрана з наступних причин:

– Безкоштовність та вільне програмне забезпечення — draw.io є вільним програмним забезпеченням з відкритим вихідним кодом, що означає, що його можна використовувати безкоштовно. Це особливо корисно для невеликих проєктів, де може бути обмежений бюджет на придбання програмного забезпечення.

– Веб-додаток — draw.io доступний як веб-додаток, що означає, що можливо створювати та редагувати діаграми без необхідності встановлення

додаткового програмного забезпечення на своєму комп'ютері. Можливо працювати з ним з будь-якого пристрою, який має доступ до Інтернету.

– Розширені можливості — draw.io підтримує створення різних видів діаграм, включаючи блок-схеми, організаційні діаграми, діаграми потоку даних, схеми баз даних і багато інших. Він також має багатий набір форм та елементів для створення професійних діаграм.

– Зручний інтерфейс — draw.io має інтуїтивний і зручний інтерфейс, що дозволяє швидко створювати та редагувати діаграми. Інструменти розташовані логічно і легко доступні, що дозволяє зосередитися на самому процесі створення діаграм, а не на вивченні складних інструментів.

– Можливість імпорту та експорту — draw.io підтримує імпорт та експорт діаграм в різних форматах, включаючи PDF, PNG, SVG та інші. Це дозволяє легко обмінюватися діаграмами з іншими користувачами та інтегрувати їх у свої проєкти.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на кваліфікаційну магістерську роботу, реалізації підлягає система автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві на основі методів штучного інтелекту.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві на основі методів штучного інтелекту. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

У цьому розділі буде надано докладний опис функціонування системи автоматизованого управління інвентарем та сервісним обслуговуванням. Система реалізована на основі методів штучного інтелекту та спрямована на оптимізацію процесів управління інвентарем та покращення сервісного обслуговування клієнтів.

Огляд функціоналу системи

Система автоматизованого управління інвентарем та сервісним обслуговуванням (далі — система) розроблена для покращення ефективності та точності управління запасами, обслуговуванням клієнтів та ремонтом комп'ютерної техніки. Основні функціональні можливості системи включають:

1) моніторинг запасів інвентарю:

а) ведення актуального обліку кількості комп'ютерної техніки та компонентів;

б) автоматичне оновлення даних про запаси на основі надходження нового обладнання та продажу старого;

2) управління замовленнями — приймання та обробка замовлень від клієнтів на збірку ПК, ремонт, або придбання обладнання;

3) прогнозування необхідності запасів — використання методів штучного інтелекту для прогнозування кількості необхідного обладнання та компонентів в майбутньому;

4) облік історії обслуговування — ведення історії обслуговування обладнання для підвищення якості та швидкості ремонту;

5) звітність та аналітика:

а) генерація звітів щодо запасів, продажів, ремонтів та інших аспектів діяльності підприємства;

б) аналіз даних для виявлення тенденцій та вдосконалення управлінських рішень;

6) інтерфейс користувача:

Система має інтуїтивний інтерфейс користувача, що дозволяє персоналу підприємства легко взаємодіяти з системою. Основні компоненти інтерфейсу включають в себе:

а) панель керування — централізований доступ до основних функцій системи, включаючи перегляд запасів, обробку замовлень та аналітику;

б) форми введення даних — забезпечення можливості внесення даних щодо нових товарів та замовлень;

в) звітність і аналітика — можливість створення та перегляду звітів та аналітики у зручному графічному форматі.

Процес роботи з системою

Процес роботи з системою передбачає наступні кроки:

1) автентифікація: користувач вводить свої ім’я та пароль, які мають міститися у базі даних. Дані порівнюються з тими, що містяться у базі даних, та доступ надається лише у випадку збігу.

2) реєстрація товарів: персонал вносить дані про товари, які перебувають на складі. Вносяться наступні дані:

а) Назва виробника.

б) Назва товару.

в) Ціна.

г) Кількість на складі.

д) Опис (якщо є).

3) моніторинг та управління запасами: система автоматично моніторить стан запасів та вказує, коли необхідно провести замовлення нового обладнання.

4) реєстрація замовлень: після внесення у систему товарів, фахівці можуть вносити інформацію про замовлення. Вносяться наступні дані:

а) Дата.

б) Замовник (якщо відомий).

в) Вартість ремонту (якщо у замовленні був ремонт обладнання).

г) Знижка.

д) Статус замовлення.

е) Додаткові дані (якщо є).

ж) Замовлені товари та їх кількість.

5) Звітність і аналітика: система генерує звіти та аналітичну інформацію, яку використовують для прийняття управлінських рішень.

Ця система допомагає підприємству підвищити продуктивність, знизити витрати та підвищити якість обслуговування клієнтів, завдяки автоматизації та використанню методів штучного інтелекту для прийняття оптимальних управлінських рішень.

Система, що реалізується, містить такі складові:

1. Інтерфейс користувача для введення інформації у базу даних і зчитування введеної інформації.

2. База даних, що складається з наступних таблиць, що позначені на рисунку 3.1:

2.1. Адміністратори (admins):

2.1.1. Ім’я (username).

2.1.2. Хеш-сума пароля (password).

2.2. Товари (products):

2.2.1. Виробник (vendor).

2.2.2. Назва (name).

2.2.3. Ціна (price).

2.2.4. Кількість (amount).

2.2.5. Опис (description).

2.3. Замовлення (orders):

2.3.1. Дата (date).

2.3.2. Замовник (customer).

2.3.3. Вартість ремонту (repair_cost).

2.3.4. Знижка (discount).

2.3.5. Статус замовлення (status).

2.3.6. Додаткові дані (details).

2.4. Замовлені товари (ordered_products):

2.4.1. Замовлення (order).

2.4.2. Товар (product).

2.4.3. Кількість (amount).

3. Інтерфейс користувача, який надає рекомендації щодо пріорітетів діяльності підприємства відповідно до внесеної інформації.

Під час взаємодії із системою користувач вносить інформацію щодо товарів, які є на підприємстві, та замовлень, які надходять. Після того, як інформація внесена, система може надати наступну інформацію:

– Перелік товарів із найбільшим попитом.

– Перелік товарів із найменшим попитом.

– Перелік товарів, яких не вистачає.

– Перелік найкращих клієнтів.

– Замовлення, що потребують дій.

Використання штучного інтелекту

У системі було використано штучний інтелект, який приймає рішення на основі заздалегідь визначених правил:

– Товарами з найбільшим попитом за певний відрізок часу вважаються ті, загальна кількість яких у замовленнях за даний час була найбільшою.

– Товарами з найменшим попитом за певний відрізок часу аналогічно вважаються ті, загальна кількість яких у замовленнях була найменшою.

– Найкращими замовниками за певний відрізок часу вважаються ті, які зробили замовлень на найбільшу суму за даний час.

– Товарами з недостатньою кількістю вважаються ті, кількість яких менша за 50.

– Необробленими замовленнями вважаються ті, статус яких «Прийнято».

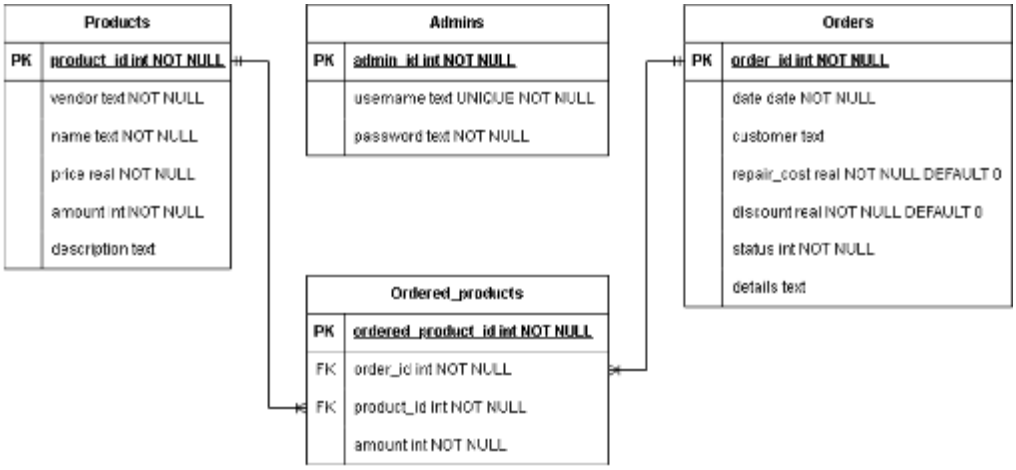

Рисунок 3.1 — Схема бази даних

3.2 Розробка структурної схеми

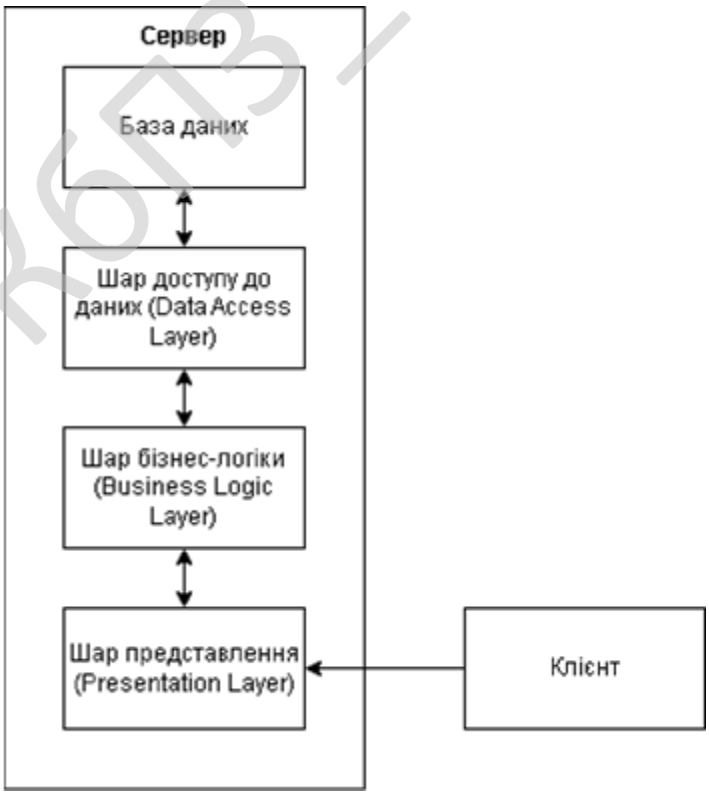

Рисунок 3.2 — Структурна схема системи

Для реалізації проєкту була використана тришарова клієнт-серверна архітектура. Тришарова клієнт-серверна архітектура — це підхід до розробки програмного забезпечення, в якому програма розділяється на три основні компоненти або шари, які взаємодіють один з одним для забезпечення функціональності системи. Ця архітектура використовується для поліпшення масштабованості, підтримки одночасного доступу користувачів та забезпечення модульності програмного забезпечення.

Опис кожного з трьох шарів тришарової клієнт-серверної архітектури:

Шар представлення (Presentation Layer)

У тришаровій архітектурі шар представлення (Presentation Layer) відповідає за представлення інформації користувачам та взаємодію з ними. Цей шар виконує наступні завдання:

– Представлення даних — шар представлення відповідає за те, щоб інформація була відображена користувачам у зручній і зрозумілій формі. Це може включати веб-сторінки, додатки для настільних комп'ютерів, мобільні додатки, графічний інтерфейс користувача (GUI) тощо.

– Взаємодія з користувачем — шар представлення забезпечує можливість користувачам взаємодіяти з системою. Це включає в себе обробку введених даних, відправку запитів до інших шарів (наприклад, шару бізнес-логіки) та отримання відповідей.

– Валідація та перевірка даних — шар представлення проводить валідацію та перевірку даних, які користувач вводить. Це допомагає запобігти введенню помилкових даних у систему.

– Управління сесією — шар представлення відстежує сесії користувачів, щоб забезпечити їм доступ до облікових записів та даних. Він також може включати можливості входу в систему, виходу із системи та управління правами доступу користувачів.

– Локалізація та міжнародна підтримка — шар представлення дозволяє адаптувати систему до різних мов, регіональних налаштувань та культурних особливостей.

– Відображення помилок та сповіщення користувачів: При виникненні помилок або несподіваних ситуацій шар представлення повинен інформувати користувачів про це та надавати можливість виправити ситуацію або звернутися за підтримкою.

– Зберігання стану інтерфейсу — шар представлення може зберігати стан інтерфейсу користувача, щоб забезпечити послідовність операцій та відновлення попередніх станів після перезавантаження сторінки або додатку.

– Кешування та оптимізація швидкодії — шар представлення може використовувати кешування для збереження попередньо завантажених даних та оптимізації швидкодії інтерфейсу.

Шар представлення грає важливу роль у взаємодії користувачів з системою і забезпечує їм доступ до функціональності, яку надає програма, у зручній та ефективній формі.

Шар бізнес-логіки (Business Logic Layer)

У тришаровій архітектурі шар бізнес-логіки (Business Logic Layer) відіграє ключову роль у виконанні бізнес-процесів та опрацюванні логіки додатку. Цей шар містить в собі всі правила, операції та опрацювання даних, які потрібні для досягнення бізнес-цілей додатку. Основні аспекти та функції шару бізнес-логіки:

– Обробка бізнес-логіки — шар бізнес-логіки визначає, які дії повинні бути виконані для досягнення бізнес-цілей додатку. Він включає в себе операції обробки даних, розрахунки, перевірку дозволів та правила бізнес-процесів.

– Правила валідації даних — шар бізнес-логіки встановлює правила валідації для введених даних. Це включає перевірку на коректність та цілісність даних, а також перевірку на відповідність бізнес-правилам.

– Робота з шаром доступу до даних — шар бізнес-логіки взаємодіє з шаром доступу до даних, забезпечуючи операції з даними (створення, зчитування, оновлення, видалення).

– Робота з послугами та інтеграціями — він також включає в себе взаємодію з іншими послугами, системами або сторонніми додатками, які можуть бути потрібні для роботи додатку.

– Бізнес-правила та обмеження — шар бізнес-логіки визначає бізнес-правила та обмеження, які регулюють доступ до функціональності додатку та взаємодію користувачів.

– Безпека — він відповідає за захист конфіденційності та цілісності даних, а також забезпечує контроль доступу та автентифікацію користувачів.

– Логування та моніторинг — шар бізнес-логіки може включати в себе функціональність для реєстрації подій, логування та моніторингу роботи додатку.

Шар бізнес-логіки відділений від інших шарів (наприклад, шару представлення та шару доступу до даних), що дозволяє підтримувати незалежність бізнес-логіки від інтерфейсів користувача та змін в структурі бази даних. Це полегшує розширення та підтримку додатку, а також сприяє більшій чіткості та стабільності системи.

Шар доступу до даних (Data Access Layer)

У тришаровій архітектурі шар доступу до даних (Data Access Layer) відіграє важливу роль у взаємодії між бізнес-логікою додатку та базою даних. Його основним завданням є забезпечення доступу до даних, зчитування та запис даних в базу даних. Ось основні аспекти та функції шару доступу до даних:

– Підключення до бази даних — шар доступу до даних забезпечує підключення до бази даних, включаючи в себе встановлення з'єднання та автентифікацію.

– Створення, зміна та видалення даних — він забезпечує можливість створення, зміни та видалення записів в базі даних згідно з бізнес-логікою додатку.

– Зчитування даних — шар доступу до даних дозволяє зчитувати інформацію з бази даних, виконувати запити та отримувати дані для подальшого використання в додатку.

– Виконання SQL-запитів — він включає можливість виконання SQL-запитів для опрацювання даних.

– Валідація даних — шар доступу до даних може проводити попередню валідацію даних перед їх відправленням до бази даних.

– Забезпечення безпеки — він гарантує безпеку взаємодії з базою даних, включаючи правила автентифікації, авторизації та захист від атак на базу даних.

– Взаємодія з ORM — шар доступу до даних може взаємодіяти з об'єктно-реляційними відомствами (ORM) для спрощення операцій з даними та мапування об'єктів додатку на структуру бази даних.

– Кешування даних — деякі системи шару доступу до даних підтримують кешування даних для покращення продуктивності та зниження навантаження на базу даних.

– Транзакції — шар доступу до даних забезпечує можливість роботи з транзакціями.

Цей шар відокремлений від інших шарів (бізнес-логіки та шару представлення), що дозволяє підтримувати незалежність від бази даних та легко змінювати систему у випадку зміни бази даних або ORM. Він також дозволяє керувати операціями з даними, щоб забезпечити цілісність інформації в базі даних.

Тришарова клієнт-серверна архітектура дозволяє забезпечити чітку розділеність обов'язків між шарами, що полегшує розробку, підтримку та масштабування системи. Кожен шар може розвиватися незалежно від інших, що робить архітектуру більш гнучкою та підходить для великих проектів.

3.3 Розробка функціональної схеми

Рисунок 3.3 — Функціональна схема системи

У реалізованому проєкті наявні чотири основних функціональних компонента:

1) Інтерфейс користувача

Даний компонент відповідає за відображення інформації, яка є результатом роботи кожного з блоків та призначена для повідомлення користувачу. За допомогою інтерфейсу користувача можлива взаємодія з усіма іншими компонентами системи.

2) Блок обробки товарів

Блок обробки товарів в системі відповідає за управління інформацією про товари, включаючи додавання, зчитування, оновлення та видалення товарів. Опис основних операцій та компонентів цього блоку:

а) Додавання товарів:

Користувач може додавати нові товари в систему. Параметри, що вводяться, можуть включати виробника, назву товару, кількість на складі, ціну та опис товару.

б) Зчитування товарів:

Користувач може переглядати список всіх товарів, які знаходяться в системі. Інтерфейс користувача надає можливість фільтрування та сортування товарів за різними параметрами.

в) Оновлення товарів:

Користувач може внести зміни в інформацію про існуючі товари. Зазвичай, він може змінювати виробника, назву товару, кількість на складі, ціну та опис.

г) Видалення товарів:

Користувач може видаляти товари, які більше не продаються або не доступні.

д) Організація даних:

Блок обробки товарів має структуру даних, що відображає таблицю товарів в базі даних. Кожен запис в таблиці відповідає окремому товару та включає поля для виробника, назви, кількості на складі, ціни та опису.

е) Зв'язок з іншими блоками:

Цей блок взаємодіє з іншими компонентами системи, такими як блок обробки замовлень та блок статистики. Наприклад, у замовленнях можуть бути посилання на конкретні товари.

3) Блок обробки замовлень

Блок обробки замовлень в системі відповідає за управління інформацією про замовлення, включаючи додавання, зчитування, оновлення та видалення замовлень. Опис основних операцій та компонентів цього блоку:

а) Додавання замовлень:

Користувач може додавати нові замовлення в систему. Параметри, що вводяться, можуть включати дату замовлення, інформацію про замовника, вартість ремонту, знижку, статус замовлення, опис та перелік замовлених товарів.

б) Зчитування замовлень:

Користувач може переглядати список всіх замовлень, які знаходяться в системі. Інтерфейс користувача надає можливість фільтрування та сортування замовлень за різними параметрами, такими як статус або дата замовлення.

в) Оновлення замовлень:

Користувач може внести зміни в інформацію про існуючі замовлення. Зазвичай, він може змінювати дату замовлення, інформацію про замовника, вартість ремонту, знижку, статус замовлення, опис та перелік замовлених товарів.

г) Видалення замовлень:

Користувач може видаляти замовлення, які більше не актуальні або не потрібні.

д) Організація даних:

Блок обробки замовлень має структуру даних, що відображає таблицю замовлень в базі даних. Кожен запис в таблиці відповідає окремому замовленню та включає поля для дати замовлення, інформації про замовника, вартості ремонту, знижки, статусу, опису та переліку замовлених товарів.

е) Зв'язок з іншими блоками:

Цей блок взаємодіє з іншими компонентами системи, такими як блок обробки товарів та блок статистики. Наприклад, замовлення може посилатися на певні товари, а також надавати інформацію щодо найкращих замовників.

Блок обробки замовлень дозволяє ефективно управляти процесом обробки замовлень, зберігаючи всю необхідну інформацію та дозволяючи користувачам швидко отримувати необхідну інформацію.

4) Блок статистики

Блок статистики в системі відповідає за створення інформаційних звітів на основі даних про товари та замовлення. Цей блок надає користувачам можливість отримувати наступну інформацію:

а) Перелік товарів із найбільшим попитом:

Система аналізує дані про замовлення та товари, визначає, які товари були замовлені найчастіше, та створює список товарів із найбільшим попитом. Це може допомогти підприємству визначити, які товари є найбільш популярними серед клієнтів.

б) Перелік товарів із найменшим попитом:

Аналогічно, система аналізує дані про товари та замовлення, щоб визначити, які товари не були популярними серед клієнтів. Це може допомогти підприємству вирішити питання про відновлення чи видалення цих товарів.

в) Перелік товарів, яких не вистачає:

Система аналізує залишки товарів на складі та активні замовлення, і визначає, яких товарів не вистачає. Ця інформація допомагає управлінцям вчасно здійснити замовлення на товари, яких бракує.

г) Перелік найкращих клієнтів:

Система аналізує дані про замовлення, ідентифікує клієнтів, які роблять найбільше замовлень або здійснюють найвищі витрати. Вона створює перелік найкращих клієнтів, що може використовуватися для нагород та рекламних акцій.

д) Замовлення, що потребують дій:

Система аналізує статуси замовлень і визначає, які замовлення потребують оброблення. Ця інформація допомагає управлінцям вчасно реагувати на потреби клієнтів та підтримувати якість обслуговування.

е) Відображення результатів:

Блок статистики забезпечує відображення зібраної інформації у зручному графічному форматі, такому як таблиці, графіки або звіти. Користувачі можуть переглядати ці звіти через веб-інтерфейс або завантажувати їх у різні формати.

Даний блок дозволяє підприємству здійснювати аналіз та приймати управлінські рішення на основі даних, що збираються в системі. Це допомагає оптимізувати бізнес-процеси, виходячи з реальних потреб та змін на ринку.

3.4 Розробка діаграми процесів

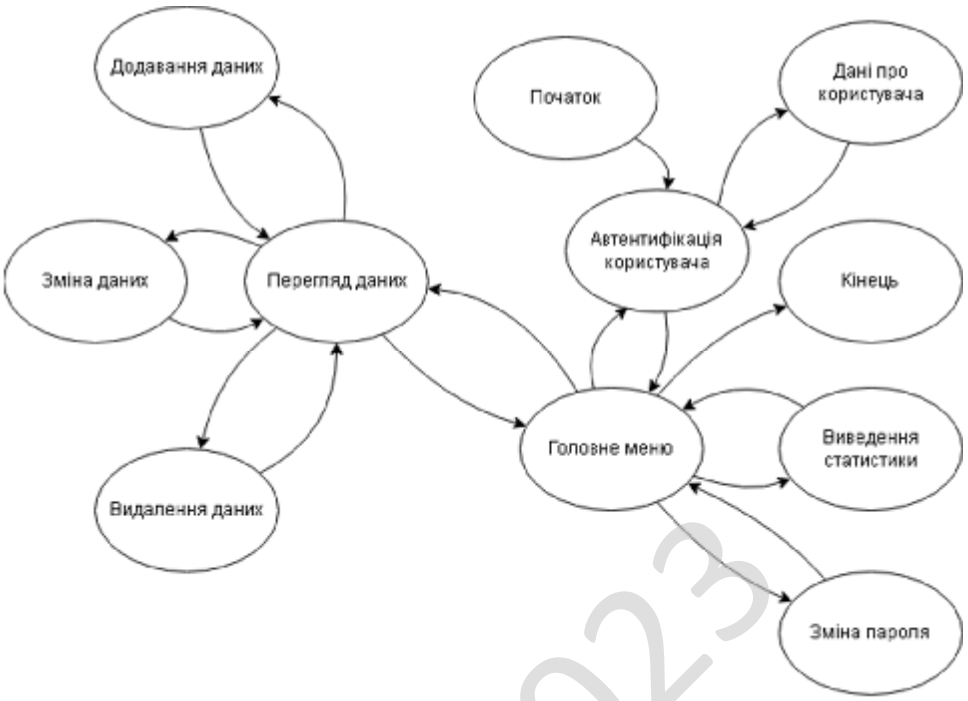

Рисунок 3.4 — Діаграма процесів

У реалізованій системі відбуваються наступні процеси:

– Автентифікація користувача — для уникнення доступу сторонніх осіб до конфіденційних даних підприємства, перед подальшим використанням системи користувач має надати ім’я свого облікового запису та пароль.

– Доступ до головного меню — з головного меню користувач може отримати доступ до усіх інших розділів системи.

– Зміна пароля — у випадку підозри на компрометацію пароля, користувач може у будь-який момент його змінити.

– Перегляд даних про товари та замовлення.

– Додавання даних про товари та замовлення.

– Зміна даних про товари та замовлення.

– Видалення даних про товари та замовлення.

– Виведення статистики на основі інформації про товари та замовлення, що міститься у системі.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Розробка блок–схем та опис алгоритмів функціонування системи

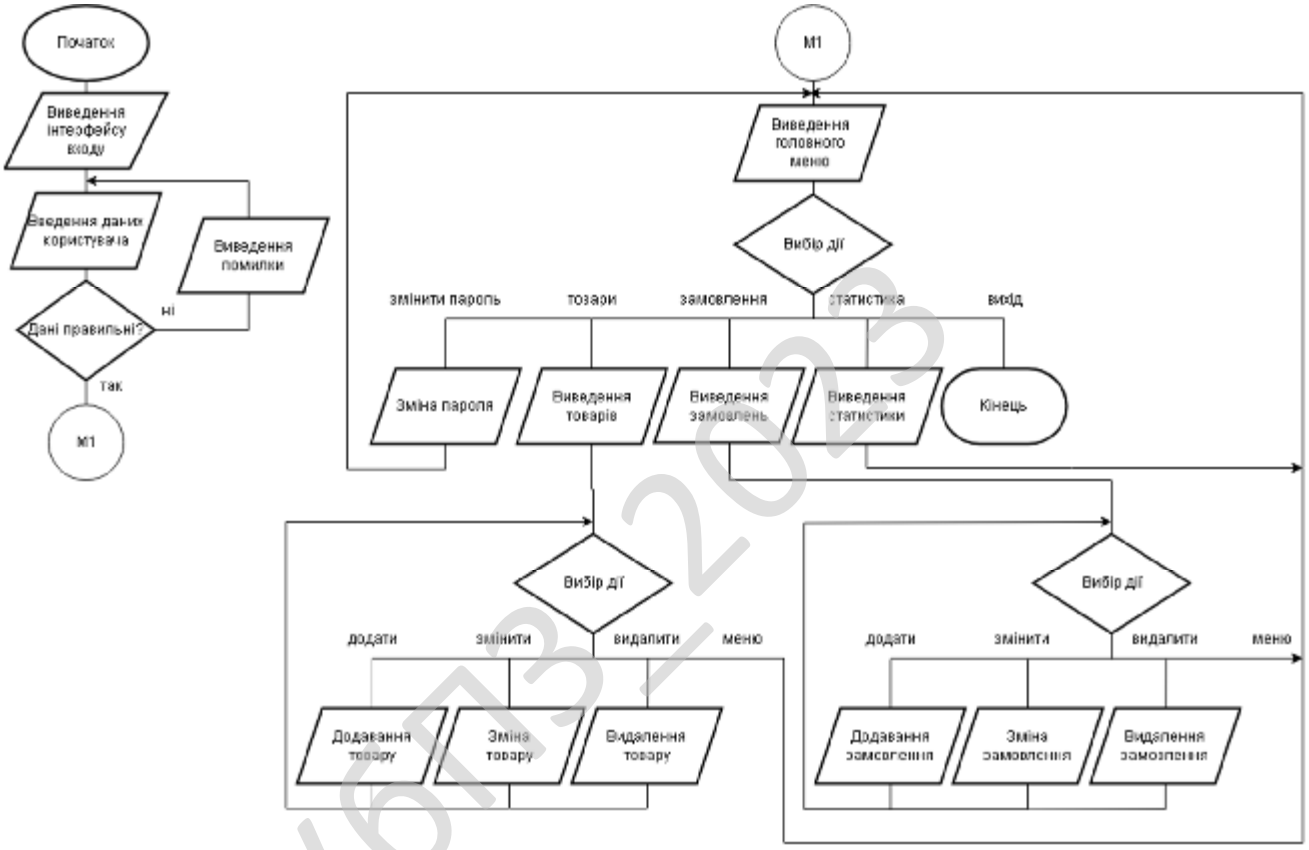

Рисунок 4.1 — Блок-схема основної програми

Блок-схема алгоритму роботи основної програми зображена на рисунку 4.1.

Після відкриття програми, користувач бачить інтерфейс входу. Даний інтерфейс необхідний для того, щоб до системи не могли отримати доступ сторонні особи, які мають доступ до мережі, у якій система працює. Код реалізації інтерфейсу входу:

Шаблон HTML-сторінки входу:

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="utf-8">
    <title>Вхід - Випускна кваліфікаційна робота</title>
</head>
<body>
    {% if error %}
    <p>{{ error }}</p>
    {% endif %}
    <form method="POST">
        <label for="username">Ім’я: </label><br>
        <input type="text" id="username" name="username" required><br>
        <label for="password">Пароль: </label><br>
        <input type="password" id="password" name="password" required><br>
        <button id="submit" type="submit">Увійти</button>
    </form>

    <p><a href="copyright">Авторське право</a></p>
</body>
</html>
```

Код входу в шарі представлення:

```
"""Вхід"""

from flask import Blueprint, render_template, redirect, request, session,
url_for

from BLL.auth import auth, check_account

from PL.app import app

login_bp = Blueprint("login", __name__)


@app.route("/login", methods=["GET", "POST"])
def login():
    """Вхід"""
    if check_account(session.get("account")):
        return redirect(url_for("index"))

    error = None
```

```
        if request.method == "POST":
            username = request.form.get("username")
            password = request.form.get("password")
            account = auth(username, password)

            if account:
                session["account"] = account
                return redirect(url_for("index"))
            error = "Неправильні дані."
        return render_template("login.html", error=error)
```

Код входу в шарі бізнес-логіки:

```
    """У даному файлі міститься функція, яка перевіряє, чи авторизований
адміністратор"""

    import bcrypt

    from DAL.auth import user_id_check, username_check
    from DAL.auth import change_password as dal_change_password


    def auth(username, password):
        """Перевіряє на дійсність дані, вказані під час авторизації
        Повертає номер облікового запису у випадку успішної авторизації, інакше
повертає False
        """
        row = username_check(username)
        if row:
            if bcrypt.checkpw(password.encode(), row[1]):
                return row[0]
        return False


    def change_password(user_id, form):
        """Змінює пароль
        Повертає True, якщо пароль змінено та False у разі помилки"""

        if not form.get("new"):
            return False

        if form.get("new") != form.get("confirmation"):
            return False
```

```
            row = user_id_check(user_id)
            if row:
                if not bcrypt.checkpw(form.get("current").encode(), row[1]):
                    return False

            dal_change_password(user_id, bcrypt.hashpw(form["new"].encode(),
bcrypt.gensalt()))
            return True


        def check_account(user_id):
            """Перевіряє, чи авторизований адміністратор
            Повертає True, якщо авторизований, та False, якщо не авторизований"""
            if user_id_check(user_id):
                return True
            return False
```

Код входу в шарі доступу до даних:

```
        """Автентифікація"""

        from DAL.connection import connection


        def user_id_check(user_id):
            """Перевіряє, чи існує обліковий запис із вказаним номером"""
            cursor = connection.cursor()
            cursor.execute("SELECT id, password FROM admins WHERE id = ?",
(user_id,))
            return cursor.fetchone()


        def change_password(user_id, password_hash):
            """Змінює пароль"""
            connection.execute(
                "UPDATE admins SET password = ? WHERE `id` = ?",
                (password_hash, user_id),
            )
            connection.commit()


        def username_check(username):
            """Перевіряє, чи існує обліковий запис із вказаним іменем"""
            cursor = connection.cursor()
```

```
        cursor.execute("SELECT id, password FROM admins WHERE username = ?",
(username,))
        return cursor.fetchone()
```

Якщо дані, уведені користувачем, відсутні у базі даних, виводиться помилка та користувач має можливість повторити спробу. Якщо ж автентифікація пройшла успішно, користувач отримує доступ до головного меню, звідки він може отримати доступ до інших частин програми.

HTML-сторінка головного меню:

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="utf-8">
    <title>Головна сторінка - Випускна кваліфікаційна робота</title>
</head>
<body>
    <p><a href="logout">Вихід</a></p><br>

    <p><a href="products">Товари</a></p>
    <p><a href="orders">Замовлення</a></p><br>

    <p><a href="stats">Статистика</a></p>

    <p><a href="password">Зміна пароля</a></p>

    <p><a href="copyright">Авторське право</a></p>
</body>
</html>
```

Натискання кнопки виходу повертає користувача на сторінку входу. Вона необхідна для захисту від несанкціонованого доступу до системи під час відсутності співробітника.

Код виходу в шарі представлення:

```
"""Вихід"""

from flask import Blueprint, redirect, session, url_for

from BLL.auth import check_account

from PL.app import app
```

```
logout_bp = Blueprint("logout", __name__)


@app.route("/logout", methods=["GET"])
def logout():
    """Вихід"""
    if check_account(session["account"]):
        session.pop("account")
    return redirect(url_for("index"))
```

Натискання кнопки товарів виводить список товарів, що містяться у системі. На сторінці товарів є можливість створити нові товари, переглянути існуючі, відредагувати чи видалити їх.

Шаблон HTML-сторінки списку товарів:

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="/static/style.css">
    <title>Товари - Випускна кваліфікаційна робота</title>
</head>
<body>
    {{product_list|safe}}
    <a href="products/create">Додати</a>
</body>
</html>
```

Шаблон HTML-сторінки редагування товару:

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="/static/style.css">
    <title>Товар - Випускна кваліфікаційна робота</title>
</head>
<body>
    {% if error %}
    <p>{{ error }}</p>
    {% endif %}
    <form method="POST">
        <label for="vendor">Виробник: </label><br>
        <input type="text" id="vendor" name="vendor" value="{{ vendor }}"
required><br>
```

```
                <label for="name">Назва: </label><br>
                <input type="text" id="name" name="name" value="{{ name }}"
required><br>
                <label for="price">Ціна: </label><br>
                <input type="number" id="price" name="price" value="{{ price }}"
min="0" step="0.01" required><br>
                <label for="amount">Кількість: </label><br>
                <input type="number" id="amount" name="amount" value="{{ amount }}"
min="0" step="1" required><br>
                <label for="description">Опис: </label><br>
                <textarea id="description" name="description">{{ description
}}</textarea><br>
                <button id="submit" type="submit">Записати</button>
            </form>
        </body>
        </html>
```

Код товарів у шарі представлення:

```
        """Список товарів"""

        from flask import Blueprint, render_template, redirect, request, session,
url_for

        from BLL.auth import check_account
        from BLL.products import create_product as bll_create_product
        from BLL.products import read_product as bll_read_product
        from BLL.products import read_products as bll_read_products
        from BLL.products import update_product as bll_update_product
        from BLL.products import delete_product as bll_delete_product

        from PL.app import app

        products_bp = Blueprint("products", __name__)


        @app.route("/products/create", methods=["GET", "POST"])
        def create_product():
            """Додавання нового товару"""
            if not check_account(session.get("account")):
                return redirect(url_for("login"))

            error = None

            if request.method == "POST":
```

```
            error = bll_create_product(request.form)
            if not error:
                return redirect(url_for("products"))

        return render_template("edit_product.html", error=error)


    @app.route("/products", methods=["GET"])
    def products():
        """Список товарів"""
        if not check_account(session.get("account")):
            return redirect(url_for("login"))

        return render_template("products.html",
product_list=bll_read_products())


    @app.route("/products/update/<product_id>", methods=["GET", "POST"])
    def update_product(product_id):
        """Редагування товару"""
        if not check_account(session.get("account")):
            return redirect(url_for("login"))

        product = bll_read_product(product_id)
        if not product:
            return redirect(url_for("products"))

        error = None

        if request.method == "POST":
            error = bll_update_product(product_id, request.form)
            if not error:
                return redirect(url_for("products"))

        return render_template(
            "edit_product.html",
            vendor=product[0],
            name=product[1],
            price=product[2],
            amount=product[3],
            description=product[4],
            error=error,
        )
```

```
@app.route("/products/delete/<product_id>", methods=["GET"])
def delete_product(product_id):
    """Видалення товару"""
    if not check_account(session.get("account")):
        return redirect(url_for("login"))

    bll_delete_product(product_id)
    return redirect(url_for("products"))
```

Код товарів у шарі бізнес-логіки:

```
"""Операції над списком товарів"""

from BLL.numbers import is_numeric

from DAL.products import create_product as dal_create_product
from DAL.products import read_product as dal_read_product
from DAL.products import read_products as dal_read_products
from DAL.products import read_products_amount as dal_read_products_amount
from DAL.products import read_products_demand as dal_read_products_demand
from DAL.products import update_product as dal_update_product
from DAL.products import delete_product as dal_delete_product


def validate_product(form):
    """Перевіряє дані товару на коректність
    Повертає True, якщо коректні, та False, якщо некоректні"""
    vendor = form.get("vendor")
    name = form.get("name")
    price = form.get("price")
    amount = form.get("amount")

    if (vendor is None or name is None or price is None or amount is None)
or (
        not is_numeric(price)
        or not amount.isdigit()
        or float(price) < 0
        or int(amount) < 0
    ):
        return False
    return True
```

```
        def create_product(form):
            """Додає новий товар
            Повертає помилку, якщо вона є"""
            if not validate_product(form):
                return "Некоректні дані"
            dal_create_product(
                form.get("vendor"),
                form.get("name"),
                form.get("price"),
                form.get("amount"),
                form.get("description"),
            )
            return None


        def read_product(product_id):
            """Отримує один товар"""
            return dal_read_product(product_id)


        def read_products(max_amount=0, demand_sort=0, date_from=None,
date_to=None):
            """Отримує список усіх товарів, або товарів, яких залишилося менше
певної кількості,
            або товарів за попитом за певний проміжок часу або за увесь час роботи
програми"""
            if max_amount and demand_sort:
                raise ValueError("Одночасно можливо використовувати лише один вид
налаштувань.")

            table = "<table>\n"
            table += """        <tr>
                    <th>Номер</th>
                    <th>Виробник</th>
                    <th>Назва</th>
                    <th>Ціна</th>
                    <th>Кількість</th>
                    <th>Опис</th>
                    """
            if demand_sort:
                table += """<th>Замовлено одиниць</th>
                """
```

```
            table += """<th>Зміна</th>
                    <th>Видалення</th>
                </tr>\n"""
            if max_amount:
                products = dal_read_products_amount(max_amount)
            elif demand_sort == 1:
                products = dal_read_products_demand(date_from=date_from,
date_to=date_to)
            elif demand_sort == 2:
                products = dal_read_products_demand(
                    reverse=True, date_from=date_from, date_to=date_to
                )
            else:
                products = dal_read_products()
            for product in products:
                table += f"""       <tr>
                    <td>{product[0]}</td>
                    <td>{product[1]}</td>
                    <td>{product[2]}</td>
                    <td>{product[3]}</td>
                    <td>{product[4]}</td>
                    <td>{product[5]}</td>
                    """
                if demand_sort:
                    table += f"""<td>{product[6]}</td>
                    """
                table += f"""<td><a
href="products/update/{product[0]}">Змінити</a></td>
                    <td><a href="products/delete/{product[0]}">Видалити</a></td>
                </tr>\n"""

            table += "    </table>"
            return table


        def update_product(product_id, form):
            """Додає новий товар
            Повертає помилку, якщо вона є"""
            if not validate_product(form):
                return "Некоректні дані"
            dal_update_product(
                product_id,
                form.get("vendor"),
```

```
                form.get("name"),
                form.get("price"),
                form.get("amount"),
                form.get("description"),
            )
            return None


        def delete_product(product_id):
            """Видаляє товар"""
            return dal_delete_product(product_id)
```

Код товарів у шарі доступу до даних:

```
        """Операції над списком товарів"""

        from DAL.connection import connection


        def create_product(vendor, name, price, amount, description):
            """Створює товар"""
            connection.execute(
                "INSERT INTO products(vendor, name, price, amount, description)
VALUES(?, ?, ?, ?, ?)",
                (vendor, name, price, amount, description),
            )
            connection.commit()


        def read_product(product_id):
            """Отримує один товар"""
            cursor = connection.cursor()
            cursor.execute(
                "SELECT vendor, name, price, amount, description FROM products
WHERE id = ?",
                (product_id,),
            )
            return cursor.fetchone()


        def read_products():
            """Отримує список усіх товарів"""
            cursor = connection.cursor()
            cursor.execute("SELECT * FROM products")
            return cursor.fetchall()
```

```
def read_products_amount(amount):
    """Отримує список товарів, яких залишилося менше певної кількості"""
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM products WHERE amount < ?", (amount,))
    return cursor.fetchall()


def read_products_demand(reverse=False, date_from=None, date_to=None):
    """Отримує дані щодо попиту на товари"""
    cursor = connection.cursor()
    cursor.execute(
        f"""SELECT p.*, COALESCE(SUM(op.amount), 0) AS total_amount
        FROM products AS p
        LEFT JOIN ordered_products AS op ON p.id = op.product
        LEFT JOIN orders AS o ON op.`order` = o.id
        {"WHERE o.date BETWEEN ? AND ?" if date_from and date_to else ""}
        GROUP BY p.id
        ORDER BY total_amount {"ASC" if reverse else "DESC"}
        LIMIT 10;""",
        (date_from, date_to) if date_from and date_to else (),
    )
    return cursor.fetchall()


def update_product(product_id, vendor, name, price, amount, description):
    """Змінює дані товару"""
    connection.execute(
        """UPDATE products SET vendor = ?, name = ?, price = ?,
        amount = ?, description = ? WHERE id = ?""",
        (vendor, name, price, amount, description, product_id),
    )
    connection.commit()


def decrease_amount(product_id, amount):
    """Зменшує кількість товару"""
    connection.execute(
        "UPDATE products SET amount = amount - ? WHERE id = ?",
        (amount, product_id),
    )
    connection.commit()
```

```
def delete_product(product_id):
    """Видаляє товар"""
    connection.execute(
        "DELETE FROM products WHERE id = ?",
        (product_id,),
    )
    connection.commit()
```

Натискання кнопки замовлень виводить список замовлень, що містяться у системі. На сторінці замовлень є можливість створити нові замовлення, переглянути існуючі, відредагувати чи видалити їх. Також можна переглянути список товарів у замовленні та встановити кількість різних товарів.

Шаблон HTML-сторінки перегляду замовлень:

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="/static/style.css">
    <title>Замовлення - Випускна кваліфікаційна робота</title>
</head>
<body>
    {{order_list|safe}}
    <a href="orders/create">Додати</a>
</body>
</html>
```

Шаблон HTML-сторінки редагування замовлення:

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="/static/style.css">
    <title>Замовлення - Випускна кваліфікаційна робота</title>
</head>
<body>
    {% if error %}
    <p>{{ error }}</p>
    {% endif %}
    <form method="POST">
        <label for="date">Дата: </label><br>
```

```
                <input type="date" id="date" name="date" value="{{ date }}"
required><br>
                <label for="customer">Замовник: </label><br>
                <input type="text" id="customer" name="customer" value="{{ customer
}}"><br>
                <label for="repair_cost">Вартість ремонту: </label><br>
                <input type="number" id="repair_cost" name="repair_cost" value="{{
repair_cost }}" min="0" step="0.01" required><br>
                <label for="discount">Знижка: </label><br>
                <input type="number" id="discount" name="discount" value="{{
discount }}" min="0" step="0.01" required><br>
                <label for="status">Статус: </label><br>
                <select name="status" id="status" required>
                    <option value="">Оберіть значення</option>
        {{statuses|safe}}
                </select><br>
                <label for="details">Деталі: </label><br>
                <textarea id="details" name="details">{{ details }}</textarea><br>
                <button id="submit" type="submit">Записати</button>
            </form>
        </body>
        </html>
```

Шаблон HTML-сторінки перегляду товарів у замовленні:

```
        <!DOCTYPE html>
        <html lang="uk">
        <head>
            <meta charset="utf-8">
            <link rel="stylesheet" href="/static/style.css">
            <title>Товари у замовленні - Випускна кваліфікаційна робота</title>
        </head>
        <body>
            {{ordered_products|safe}}
            {% if error %}
            <p>{{ error }}</p>
            {% endif %}
            <form method="POST">
                <label for="product">Номер товару: </label><br>
                <input type="number" id="product" name="product" min="1" step="1"
required><br>
                <label for="amount">Кількість: </label><br>
                <input type="number" id="amount" name="amount" min="0" step="1"
required><br>
                <input type="checkbox" id="change_amount" name="change_amount">
```

```
            <label for="change_amount">Змінювати кількість</label><br>
            <button id="submit" type="submit">Записати</button>
        </form>
    </body>
    </html>
```

Код замовлень у шарі представлення:

```
    """Список замовлень"""

    from flask import Blueprint, render_template, redirect, request, session,
url_for

    from BLL.auth import check_account
    from BLL.orders import create_order as bll_create_order
    from BLL.orders import read_order as bll_read_order
    from BLL.orders import read_orders as bll_read_orders
    from BLL.orders import read_statuses as bll_read_statuses
    from BLL.orders import update_order as bll_update_order
    from BLL.orders import delete_order as bll_delete_order
    from BLL.orders import read_ordered_products as bll_read_ordered_products
    from BLL.orders import update_ordered_product as bll_update_ordered_product

    from PL.app import app

    orders_bp = Blueprint("orders", __name__)


    @app.route("/orders/create", methods=["GET", "POST"])
    def create_order():
        """Додавання нового замовлення"""
        if not check_account(session.get("account")):
            return redirect(url_for("login"))

        error = None

        if request.method == "POST":
            error = bll_create_order(request.form)
            if not error:
                return redirect(url_for("orders"))

        return render_template(
            "edit_order.html", error=error, discount=0,
statuses=bll_read_statuses()
        )
```

```
@app.route("/orders", methods=["GET"])
def orders():
    """Список замовлень"""
    if not check_account(session.get("account")):
        return redirect(url_for("login"))

    order_list = bll_read_orders()
    return render_template("orders.html", order_list=order_list)


@app.route("/orders/update/<order_id>", methods=["GET", "POST"])
def edit_order(order_id):
    """Редагування замовлення"""
    if not check_account(session.get("account")):
        return redirect(url_for("login"))

    order = bll_read_order(order_id)
    if not order:
        return redirect(url_for("orders"))

    error = None

    if request.method == "POST":
        error = bll_update_order(order_id, request.form)
        if not error:
            return redirect(url_for("orders"))

    return render_template(
        "edit_order.html",
        date=order[0],
        customer=order[1],
        repair_cost=order[2],
        discount=order[3],
        details=order[5],
        statuses=bll_read_statuses(order[4]),
        error=error,
    )


@app.route("/orders/delete/<order_id>", methods=["GET"])
def delete_order(order_id):
```

```
    """Видалення замовлення"""
    if not check_account(session.get("account")):
        return redirect(url_for("login"))

    bll_delete_order(order_id)
    return redirect(url_for("orders"))


@app.route("/orders/products/<order_id>", methods=["GET", "POST"])
def ordered_products(order_id):
    """Товари у замовленні"""
    if not check_account(session.get("account")):
        return redirect(url_for("login"))

    order = bll_read_order(order_id)
    if not order:
        return redirect(url_for("orders"))

    error = None

    if request.method == "POST":
        error = bll_update_ordered_product(order_id, request.form)

    return render_template(
        "ordered_products.html",
        ordered_products=bll_read_ordered_products(order_id),
        error=error,
    )
```

Код замовлень у шарі бізнес-логіки:

```
"""Операції над списком замовлень"""

from BLL.dates import validate_date
from BLL.numbers import is_numeric

from DAL.orders import create_order as dal_create_order
from DAL.orders import read_order as dal_read_order
from DAL.orders import read_orders as dal_read_orders
from DAL.orders import read_new_orders as dal_read_new_orders
from DAL.orders import update_order as dal_update_order
from DAL.orders import delete_order as dal_delete_order

from DAL.orders import create_ordered_product as dal_create_ordered_product
from DAL.orders import read_ordered_product as dal_read_ordered_product
```

```
        from DAL.orders import read_ordered_products as dal_read_ordered_products
        from DAL.orders import update_ordered_product as dal_update_ordered_product
        from DAL.orders import delete_ordered_product as dal_delete_ordered_product

        from DAL.orders import read_best_customers as dal_read_best_customers

        from DAL.products import read_product as dal_read_product
        from DAL.products import decrease_amount as dal_decrease_amount


        statuses = (
            "Прийнято",
            "Очікується оплата",
            "Очікується доставка",
            "Виконано",
            "Повернено",
            "Скасовано",
        )


        def validate_order(form):
            """Перевіряє дані замовлення на коректність"""
            date = form.get("date")
            repair_cost = form.get("repair_cost")
            discount = form.get("discount")
            status = form.get("status")

            if (
                (date is None or repair_cost is None or discount is None or status
is None)
                or not (is_numeric(repair_cost) or float(repair_cost) < 0)
                or not (is_numeric(discount) or float(discount) < 0)
                or not status.isdigit()
                or not validate_date(date)
            ):
                return False

            status = int(status)

            if status < 0 or status >= len(statuses):
                return False

            return True
```

```
def create_order(form):
    """Додає нове замовлення"""
    if not validate_order(form):
        return "Некоректні дані"
    dal_create_order(
        form.get("date"),
        form.get("customer"),
        form.get("repair_cost"),
        form.get("discount"),
        form.get("status"),
        form.get("details"),
    )
    return None


def read_order(order_id):
    """Отримує одне замовлення"""
    return dal_read_order(order_id)


def read_orders(only_new=False):
    """Отримує список усіх або лише нових замовлень"""
    table = "<table>\n"
    table += """        <tr>
            <th>Номер</th>
            <th>Дата</th>
            <th>Замовник</th>
            <th>Вартість ремонту</th>
            <th>Вартість товарів</th>
            <th>Знижка</th>
            <th>До сплати</th>
            <th>Статус</th>
            <th>Деталі</th>
            <th>Товари</th>
            <th>Зміна</th>
            <th>Видалення</th>
        </tr>\n"""
    orders = dal_read_new_orders() if only_new else dal_read_orders()

    for order in orders:
        product_cost = read_products_cost(order[0])
```

```
            table += f"""        <tr>
                <td>{order[0]}</td>
                <td>{order[1]}</td>
                <td>{order[2]}</td>
                <td>{order[3]}</td>
                <td>{product_cost}</td>
                <td>{order[4]}</td>
                <td>{order[3]+product_cost-order[4]}
                <td>{statuses[order[5]]}</td>
                <td>{order[6]}</td>
                <td><a href="orders/products/{order[0]}">Товари</a></td>
                <td><a href="orders/update/{order[0]}">Змінити</a></td>
                <td><a href="orders/delete/{order[0]}">Видалити</a></td>
            </tr>\n"""

        table += "    </table>"
        return table


    def update_order(order_id, form):
        """Оновлює замовлення"""
        if not validate_order(form):
            return "Некоректні дані"
        dal_update_order(
            order_id,
            form.get("date"),
            form.get("customer"),
            form.get("repair_cost"),
            form.get("discount"),
            form.get("status"),
            form.get("details"),
        )
        return None


    def delete_order(order_id):
        """Видаляє замовлення"""
        return dal_delete_order(order_id)


    def read_statuses(selected=None):
        """Повертає список статусів у HTML-вигляді"""
        response = ""
```

```
    for idx, status in enumerate(statuses):
        response += f'        <option value="{idx}"'
        if idx == selected:
            response += " selected"
        response += f">{status}</option>\n"
    return response


def validate_ordered_product(form):
    """Перевіряє дані товару у замовленні на коректність"""
    product = form.get("product")
    amount = form.get("amount")

    if not (is_numeric(product) or is_numeric(amount)):
        return False

    product = int(product)
    amount = int(amount)
    product_data = dal_read_product(product)

    if (product < 0 or amount < 0) or not product_data:
        return False

    if form.get("change_amount") and product_data[3] < amount:
        return False

    return True


def read_products_cost(order):
    """Отримує вартість товарів замовлення"""
    total = 0.0
    entries = dal_read_ordered_products(order)
    for entry in entries:
        product = dal_read_product(entry[0])
        total += product[2] * entry[1]
    return total


def read_ordered_products(order):
    """Отримує список товарів у замовленні"""
    total = 0.0
    table = "<table>\n"
```

```
        table += """        <tr>
                <th>Номер товару</th>
                <th>Виробник</th>
                <th>Назва</th>
                <th>Ціна</th>
                <th>Кількість</th>
                <th>Вартість</th>
            </tr>\n"""
        entries = dal_read_ordered_products(order)
        for entry in entries:
            product = dal_read_product(entry[0])
            cost = product[2] * entry[1]
            table += f"""        <tr>
                <td>{entry[0]}</td>
                <td>{product[0]}</td>
                <td>{product[1]}</td>
                <td>{product[2]}</td>
                <td>{entry[1]}</td>
                <td>{cost}</td>
            </tr>\n"""
            total += cost

        table += f"""        <tr>
                <td colspan="5">Всього за товари:</td>
                <td>{total}</td>
            </tr>\n"""

        table += "    </table>"
        return table


    def update_ordered_product(order, form):
        """Змінює кількість товару у замовленні
        Повертає помилку, якщо вона є"""
        if not validate_ordered_product(form):
            return "Некоректні дані"

        product = int(form.get("product"))
        amount = int(form.get("amount"))

        ordered_product = dal_read_ordered_product(order, product)

        if form.get("change_amount"):
```

```
            dal_decrease_amount(product, (amount - ordered_product[1]) if
ordered_product else amount)

            if amount == 0:
                if ordered_product:
                    dal_delete_ordered_product(order, product)
            else:
                if not dal_read_ordered_product(order, product):
                    dal_create_ordered_product(order, product, amount)
                else:
                    dal_update_ordered_product(order, product, amount)
            return None


        def read_best_customers(date_from=None, date_to=None):
            """Отримує список найкращих замовників"""
            table = "<table>\n"
            table += """        <tr>
                    <th>Замовник</th>
                    <th>Сума</th>
                </tr>\n"""
            customers = dal_read_best_customers(date_from=date_from,
date_to=date_to)

            for customer in customers:
                table += f"""      <tr>
                    <td>{customer[0]}</td>
                    <td>{customer[1]}</td>
                </tr>\n"""

            table += "    </table>"
            return table
```

Код замовлень у шарі доступу до даних:

```
        """Операції над списком замовлень"""

        from DAL.connection import connection


        def create_order(date, customer, repair_cost, discount, status, details):
            """Створює замовлення"""
            connection.execute(
                """INSERT   INTO   orders(date,   customer,   repair_cost,   discount,
status, details)
```

```
                VALUES(?, ?, ?, ?, ?, ?)""",
                (date, customer, repair_cost, discount, status, details),
            )
            connection.commit()


        def read_order(order_id):
            """Отримує одне замовлення"""
            cursor = connection.cursor()
            cursor.execute(
                "SELECT date, customer, repair_cost, discount, status, details FROM
orders WHERE id = ?",
                (order_id,),
            )
            return cursor.fetchone()


        def read_orders():
            """Отримує список усіх замовлень"""
            cursor = connection.cursor()
            cursor.execute("SELECT * FROM orders")
            return cursor.fetchall()


        def read_new_orders():
            """Отримує список нових замовлень"""
            cursor = connection.cursor()
            cursor.execute("SELECT * FROM orders WHERE status = 0")
            return cursor.fetchall()


        def read_best_customers(date_from=None, date_to=None):
            """Отримує список найкращих замовників"""
            cursor = connection.cursor()
            cursor.execute(
                f"""SELECT o.customer,
                SUM(COALESCE((p.price * op.amount), 0) + o.repair_cost -
o.discount) AS total_order_cost
                FROM orders AS o
                LEFT JOIN ordered_products AS op ON o.id = op.`order`
                LEFT JOIN products AS p ON op.product = p.id
                WHERE o.customer IS NOT NULL and o.customer != ''
                {"AND o.date BETWEEN ? AND ?" if date_from and date_to else ""}
```

```
            GROUP BY o.customer
            ORDER BY total_order_cost DESC
            LIMIT 10;""",
            (date_from, date_to) if date_from and date_to else (),
        )
        return cursor.fetchall()


    def update_order(order_id, date, customer, repair_cost, discount, status,
details):
        """Змінює дані замовлення"""
        connection.execute(
            """UPDATE orders SET date = ?, customer = ?, repair_cost = ?,
            discount = ?, status = ?, details = ? WHERE id = ?""",
            (date, customer, repair_cost, discount, status, details, order_id),
        )
        connection.commit()


    def delete_order(order_id):
        """Видаляє замовлення"""
        connection.execute(
            "DELETE FROM orders WHERE id = ?",
            (order_id,),
        )
        connection.commit()


    def create_ordered_product(order, product, amount):
        """Створює товар у замовленні"""
        connection.execute(
            "INSERT INTO ordered_products(`order`, `product`, `amount`)
VALUES(?, ?, ?)",
            (order, product, amount),
        )
        connection.commit()


    def read_ordered_product(order, product):
        """Отримує один товар у замовленні"""
        cursor = connection.cursor()
        cursor.execute(
```

```
            "SELECT product, amount FROM ordered_products WHERE `order` = ? AND
`product` = ?",
            (order, product),
        )
        return cursor.fetchone()


    def read_ordered_products(order):
        """Отримує список товарів у замовленні"""
        cursor = connection.cursor()
        cursor.execute(
            "SELECT product, amount FROM ordered_products WHERE `order` = ?",
            (order,),
        )
        return cursor.fetchall()


    def update_ordered_product(order, product, amount):
        """Оновлює дані про товар у замовленні"""
        connection.execute(
            "UPDATE ordered_products SET amount = ? WHERE `order` = ? AND
`product` = ?",
            (amount, order, product),
        )
        connection.commit()


    def delete_ordered_product(order, product):
        """Видаляє дані про товар у замовленні"""
        connection.execute(
            "DELETE FROM ordered_products WHERE `order` = ? AND `product` = ?",
            (order, product),
        )
        connection.commit()
```

Натискання кнопки статистики виводить інформацію про статистику, що повертається блоками замовлень та товарів.

Рисунок 4.2 — Блок-схема блоку статистики

Блок-схема блоку статистики зображена на рисунку 4.2.

Шаблон HTML-сторінки статистики:

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="/static/style.css">
    <title>Статистика - Випускна кваліфікаційна робота</title>
</head>
<body>
    {{period}}
```

```
            <form method="POST">
                <label for="date_from">Від: </label><br>
                <input type="date" id="date_from" name="date_from"><br>
                <label for="date_to">До: </label><br>
                <input type="date" id="date_to" name="date_to"><br>
                <button id="submit" type="submit">Підтвердити</button>
            </form>
            <p>10 товарів із найбільшим попитом:</p>
        {{high_demand|safe}}
            <p>10 товарів із найменшим попитом:</p>
        {{low_demand|safe}}
            <p>10 найкращих замовників:</p>
        {{best_customers|safe}}
            <p>Перелік товарів, яких залишилося менше 50:</p>
        {{low_amount|safe}}
            <p>Необроблені замовлення:</p>
        {{new_orders|safe}}
        </body>
        </html>
```

Код статистики в шарі представлення:

```
        """Статистика"""

        from flask import Blueprint, render_template, redirect, request, session,
url_for

        from BLL.auth import check_account
        from BLL.orders import read_orders as bll_read_orders
        from BLL.orders import read_best_customers as bll_read_best_customers
        from BLL.products import read_products as bll_read_products

        from BLL.dates import validate_dates as bll_validate_dates

        from PL.app import app

        stats_bp = Blueprint("stats", __name__)


        @app.route("/stats", methods=["GET", "POST"])
        def stats():
            """Статистика"""
            if not check_account(session.get("account")):
                return redirect(url_for("login"))
```

```
            dated = False

            low_amount = bll_read_products(max_amount=50)
            new_orders = bll_read_orders(only_new=True)

            if request.method == "POST":
                date_from = request.form.get("date_from")
                date_to = request.form.get("date_to")
                if bll_validate_dates(date_from, date_to):
                    dated = True

            if dated:
                high_demand = bll_read_products(
                    demand_sort=1, date_from=date_from, date_to=date_to
                )
                low_demand = bll_read_products(
                    demand_sort=2, date_from=date_from, date_to=date_to
                )
                best_customers = bll_read_best_customers(date_from=date_from,
date_to=date_to)
            else:
                high_demand = bll_read_products(demand_sort=1)
                low_demand = bll_read_products(demand_sort=2)
                best_customers = bll_read_best_customers()

            return render_template(
                "stats.html",
                period=f"Відображення даних з {date_from} по {date_to}"
                if dated
                else "Відображення загальних даних",
                high_demand=high_demand,
                low_demand=low_demand,
                best_customers=best_customers,
                low_amount=low_amount,
                new_orders=new_orders,
            )
```

Натискання кнопки авторського права виводить інформацію про автора даної програми.

HTML-сторінка авторського права:

```
<!DOCTYPE html>
<html lang="uk">
<head>
```

```
            <meta charset="utf-8">
            <title>Авторське право - Випускна кваліфікаційна робота</title>
        </head>
        <body>
            <p>МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ<br>
            Центральноукраїнський національний технічний університет<br>
            Кафедра кібербезпеки та програмного забезпечення</p>

            <p>Тема: Дослідження та програмна реалізація системи автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві на основі методів штучного інтелекту<br>
            Спеціальність: 122 «Комп’ютерні науки»<br>
            Освітній ступінь: магістр<br>
            Виконав: Михайленко Георгій Васильович<br>
            Науковий керівник: Буравченко Костянтин Олегович<br>
            Кропивницький - 2023</p>

        </body>
        </html>
```


4.2 Захист розробленого програмного забезпечення

У реалізованому програмному забезпеченні використовується алгоритм хешування bcrypt.

Bcrypt — це алгоритм хешування паролів, який призначений для забезпечення безпеки паролів шляхом ускладнення атаки повним перебором або з використанням райдужних таблиць. Bcrypt використовує адаптивну функцію хешування, яка включає в себе обчислення хешу з додатковими параметрами, такими як «сіль» (salt) і «робочий фактор» (work factor).

Загальний опис алгоритму хешування Bcrypt:

1. Генерація «солі» (Salt)

Сіль — це випадкова величина, яка генерується для кожного пароля окремо. Вона додається до паролю перед хешуванням і зберігається разом з хешем. Сіль гарантує, що однакові паролі будуть мати різні хеші.

2. Обчислення хешу

Пароль разом із сіллю обробляється через функцію хешування, яка базується на ключі (Blowfish cipher). При цьому використовується ітеративний підхід, де пароль хешується багатократно з використанням обраного робочого фактора.

3. Робочий фактор (Work Factor)

Робочий фактор визначає кількість ітерацій, які будуть виконані для обчислення хешу. Він контролює витрату обчислювальних ресурсів і може бути налаштований так, щоб збільшити складність атак брутфорса. Зазвичай використовується значення від 10 до 12, і воно збільшується з плином часу з огляду на збільшення потужності обчислювальної техніки.

4. Зберігання солі та хешу

Сіль і отриманий хеш зберігаються разом у базі даних. При верифікації пароля, сіль витягається з бази даних і використовується для переобчислення хешу на введеному паролі. Потім порівнюється обчислений хеш зі збереженим у базі даних. Якщо вони співпадають, пароль вважається правильним.

Bcrypt є популярним алгоритмом хешування паролів, оскільки він надійно захищає паролі від атак повним перебором та з використанням райдужних таблиць, і дозволяє гнучке налаштування рівня безпеки за допомогою робочого фактора.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Після відкриття програми, користувач бачить вікно входу до системи:

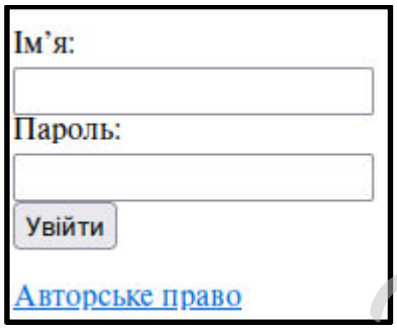

Рисунок 5.1 — вхід до системи

Якщо внесені правильні дані, відкривається головне меню:

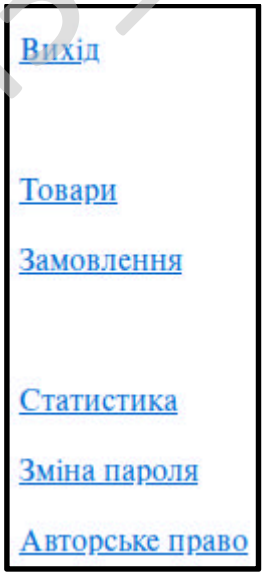

Рисунок 5.2 — головне меню програми

З головного меню користувач може виконати наступні дії:

– Вийти з системи.

– Відкрити список товарів.

– Відкрити список замовлень.

– Відкрити меню статистики.

– Змінити пароль.

– Переглянути інформацію щодо авторського права.

При натисканні кнопки виходу з системи, користувач знову бачить вікно входу у систему з рисунку 5.1.

При натисканні кнопки списку товарів, користувач бачить вікно товарів:

Номер	Виробник	Назва	Ціна	Кількість	Опис	Зміна	Видалення
1	Philips	Монітор Philips 221V8/00	3579.0	500	Монітор Philips лінійки V із широким кутом огляду пропонує перегляд без обмежень і є чудовим надбанням із необхідними функціями. Технологія Adaptive-Sync забезпечує бездоганний перегляд без розривів зображення. А завдяки таким функціям, як режим LowBlue та відсутність мерехтіння, очі не втомлюються під час роботи.	Змінити	Видалити
2	HP	Ноутбук HP 15s-eq2289nw	16499.0	100	Ноутбук HP Laptop — повнофункціональний ноутбук, обладнаний усіма потрібними функціями для виконання різних завдань і роз'ємами для швидкого під'єднання додаткових пристроїв. HP Laptop вирізняється вигідною ціною й дає змогу завжди залишатися на зв'язку. Міцний корпус ноутбука надійно захищає його від пошкоджень у разі перенесення.	Змінити	Видалити
3	Vinga	Корпус Vinga CS112B	879.0	25	Корпус CS112B розроблений для доукомплектування персональних комп'ютерів середньої потужності. Внутрішній простір моделі сегментируется на 4 слоти PCI, а також 2 зовнішніх відсіки і 4 внутрішніх. У тильній, фронтальної та бокової частинах конструкції передбачені місця для установки додаткових кулерів діаметром 8 і 12 см. Вивід гарячого повітря за межі системного блоку здійснюється через вентиляційні решітки, створені методом перфорирування панелей. Стінки і внутрішні деталі корпуса виготовлені з високоякісної листової сталі товщиною 0.4 мм.	Змінити	Видалити

Додати

Рисунок 5.3 — список товарів

У списку товарів користувач може натиснути кнопку додавання нового товару чи зміни існуючого, або видалити товар. При натисканні кнопки додавання чи зміни товару, відкривається вікно редагування.

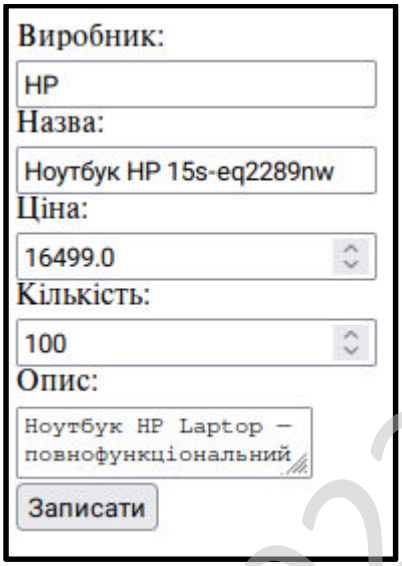

Рисунок 5.4 — вікно редагування товару

Якщо у головному меню користувач натискає на кнопку замовлень, він бачить список та статус замовлень:

Номер	Дата	Замовник	Вартість ремонту	Вартість товарів	Знижка	До сплати	Статус	Деталі	Товари	Зміна	Видалення
1	2023-11-19	Гавриленко Юрій Антонович	2000.0	3579.0	300.0	5279.0	Прийнято	Пришвидшити виконання.	Товари	Змінити	Видалити
2	2023-11-21	Сушко Владислав Семенович	1500.0	0.0	100.0	1400.0	Виконано	Виконати в першу чергу.	Товари	Змінити	Видалити
3	2023-11-22	Бойко Григорій Петрович	2000.0	879.0	0.0	2879.0	Прийнято	Зателефонувати 2023-11-23.	Товари	Змінити	Видалити

Додати

Рисунок 5.5 — список замовлень

При натисканні на кнопку списку товарів у замовленні, користувач бачить перелік товарів та їхньої кількості, а також може внести зміни.

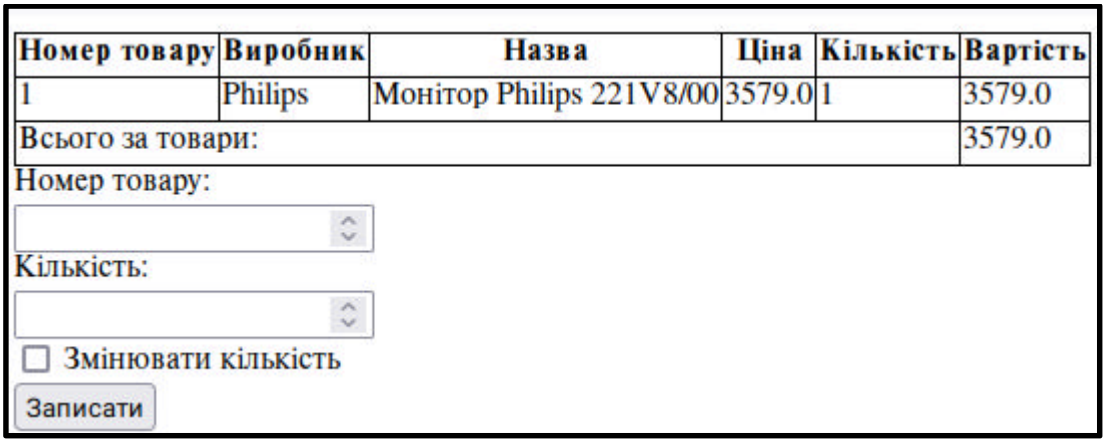

Рисунок 5.6 — список товарів у замовленні

Якщо була натиснута кнопка редагування або створення замовлення, користувач бачить вікно редагування замовлення:

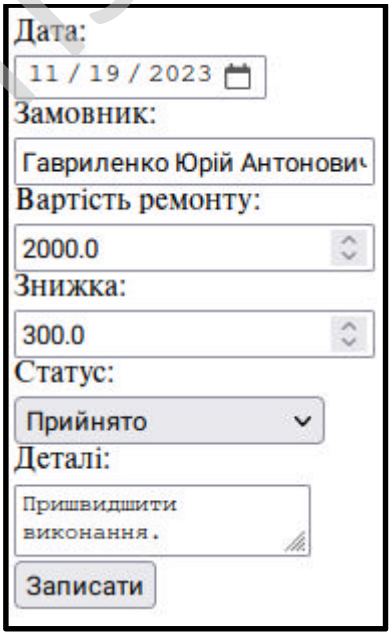

Рисунок 5.7 — вікно редагування замовлення

Якщо користувач натискає на кнопку статистики, він бачить статистику за увесь час, або за обраний період:

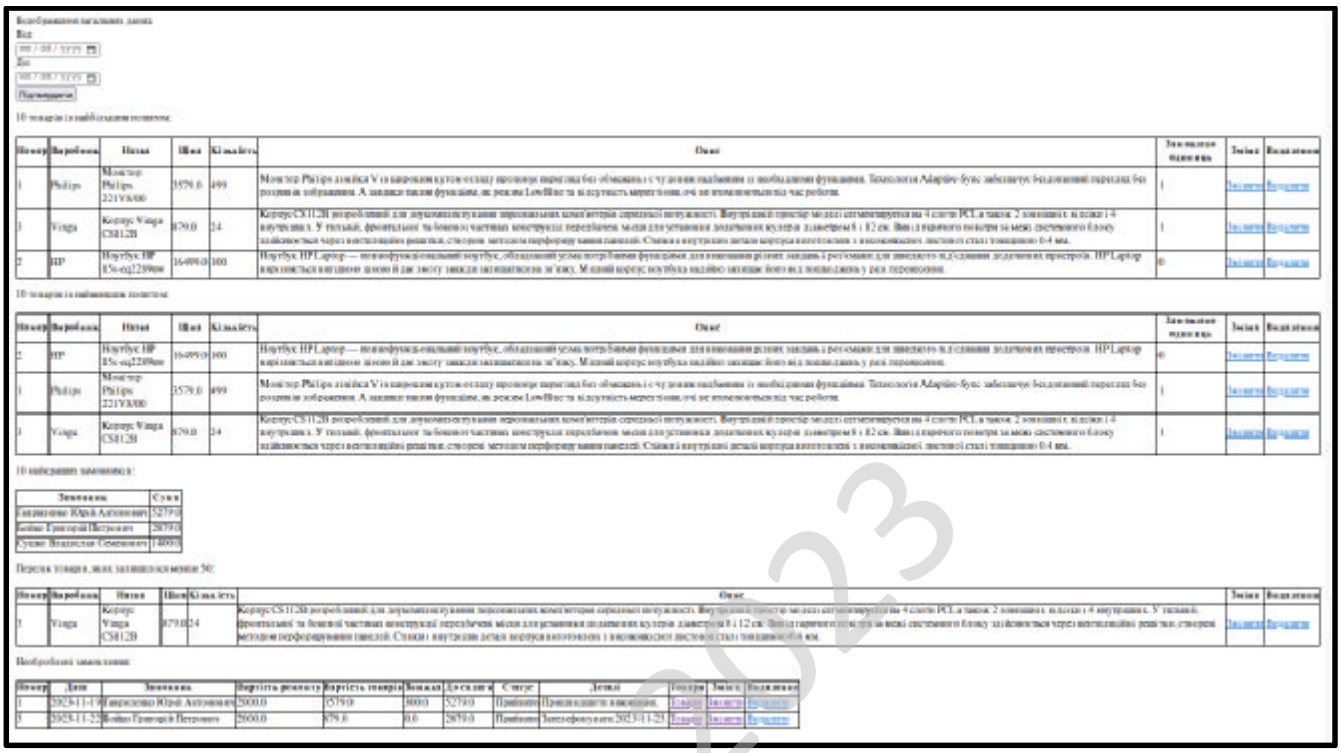

Рисунок 5.8 — вікно статистики

При натисканні на кнопку зміни пароля, відображається вікно зміни пароля:

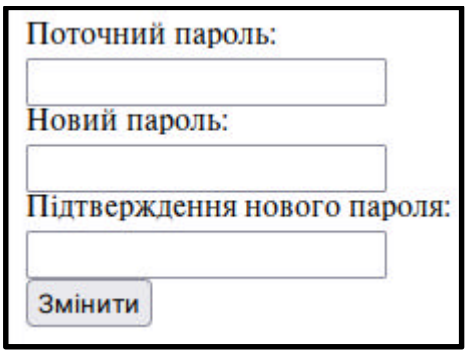

Рисунок 5.9 — вікно зміни пароля

При натисканні на кнопку авторського права, відображається інформація щодо авторства програми:

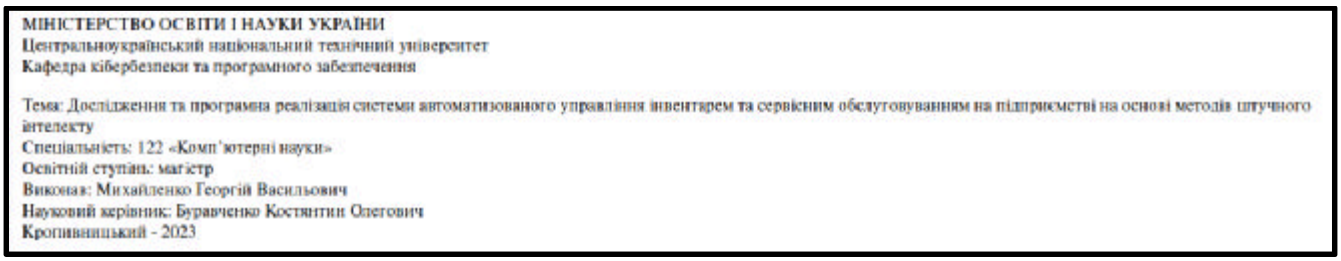

Рисунок 5.10 — вікно авторського права

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблена система автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві на основі методів штучного інтелекту.

Метою розробки є дослідження та програмна реалізація системи автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві на основі методів штучного інтелекту.

Об'єктом дослідження є процес автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві.

Предметом дослідження є методи штучного інтелекту для автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві.

Методи дослідження базуються на методах штучного інтелекту, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна одержаних результатів. Наукова новизна отриманих результатів полягає в наступному:

1. Використано методи штучного інтелекту. Алгоритми враховують кількість товарів, які були замовлені і надають рекомендації щодо подальших дій підприємства.

2. Покращено ефективність та точність. Завдяки використанню методів ШІ, система може аналізувати великий обсяг даних та приймати рішення в реальному часі.

7 ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми дипломного проекту

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 24 дні (один місяць).

В магістерській роботі було проведено дослідження та виконана програмна реалізація системи автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві на основі методів штучного інтелекту.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп’ютері баз даних;

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	20
3. Запланований термін розробки, днів	Fpq	24 (1 місяць)
4. Група задачі підсистеми управління (1-6)	–	3
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2
7. Кількість макетів вхідної інформації	–	7

Продовження таблиці 7.1

Показники	Позначення	Характеристика або величина
1	2	3
8. Кількість форм вихідної інформації.	–	8
9. Мова програмування (1-6)	–	6
10. Попередній досвід (1-6)	–	2
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	2
13. Рівень спрацьованості колективу (1-6)	–	1
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	4
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	3
17. Складність кінцевого програмного продукту (1-6)	–	2
18. Необхідний рівень забезпечення повторного використання (1-6)	–	1
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	1
20. Вимоги до швидкодії ПП (1-6)	–	2
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	1
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	3
25. Постійність складу команди розробників (1-6)	–	2

Продовження таблиці 7.1

Показники	Позначення	Характеристика або величина
1	2	3
26. Досвід розробки додатків (1-6)	–	3
27. Досвід роботи з обчислювальною платформою (1-6)	–	2
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	1
29. Досвід роботи з програмними інструментами розробки (1-6)	–	2
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	1
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	17000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках

МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \text{ Size}^B, \tag{7.1}$$

де: A – коефіцієнт Боема, $A = 2{,}45$;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i, \tag{7.2}$$

де: Wi – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(3{,}24+3{,}64+3{,}38+4{,}94+2{,}73) = 1{,}028.$$

$$T_{ном} = 2{,}45 \cdot 1{,}3^{1{,}028} = 3{,}2 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПV_j, \tag{7.3}$$

де: $ПV_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 3{,}2 \cdot (1{,}15 \cdot 1 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}89 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1 \cdot 1{,}1 \cdot 1 \times$$
$$\times 1{,}12 \cdot 1{,}22 \cdot 1{,}12 \cdot 1{,}25 \cdot 1{,}1) = 6{,}44 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3CT_{уточн}^{0{,}33+0{,}2(B-1{,}01)}S, \qquad (7.4)$$

де: C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4). Для мов програмування, що не зазначені у переліку – 2,75;

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%.

$$T_{РП} = 0{,}3 \cdot 2{,}75 \cdot 6{,}44^{0{,}33+0{,}2(1{,}028-1{,}01)} \cdot 100 = 154 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	14	Д7
Робочий проект	100	Ф 7.1-7.4

Продовження таблиці 7.2

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Впровадження	21	Д13
Всього	154	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \quad (7.5)$$

де: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні

$$Ч = \frac{154 \cdot 1}{24-3} = 7{,}3 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт протягом року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	385	12	4620	77
Монітор	160	12	1920	32
Клавіатура	140	12	1680	28
Маніпулятор «мишка»	30	12	360	6
Принтер матричний	185	1	185	3
Принтер лазерний	355	2	710	12
Принтер струминний	300	1	300	5
Сканер	155	2	310	5
Концентратор–маршрутизатор	155	2	310	5
Кабельні господарства ЛОМ на 1 м. п.	2,5	100	250	4
Кабельне господарство електромережі	48	50	2400	40
Копіювальний апарат	285	2	570	10
Усього за рік:			$З_ч$	227

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків не повинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{mic}}{1,2}, \quad (7.6)$$

$$Ф_{др}^{с} = \frac{227 \cdot 2}{1,2} = 378,3 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф^{с}_{др}}{F_{др} \cdot T_{зм}}, \quad (7.7)$$

$$Ч_{ел} = 378{,}3/(24 \cdot 8) = 2 \text{ ставки.}$$

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп’ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2019, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN, PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв’язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв’язку між віддаленими об’єктами, ЛОМ	0,5	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Забезпечення цілодобової роботи зв’язку клієнтів до мережі Інтернет	1	0,5
Всього		4	
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Продовження таблиці 7.4.

Посада	Вид роботи	Час	К-ть штатних одиниць
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (IT-менеджер)	1	16000	16000
Продакт-менеджер	0,25	13138	3284,5
Інженер-програміст	7,3	15000	109500
Інженер-електронщик	2	10000	20000
Інженер-системотехнік	0,25	10000	2500
Адміністратор мережі	0,5	10000	5000
Системний програміст	0,25	10000	2500
Дизайнер WEB	0,25	10000	2500
Інженер-верстальник	0,25	10000	2500
Бухгалтер-економіст	0,25	12500	3125
Всього за період розробки	$R_{сп}$ = 12,3	-	$Ф_{роб}$ = 166909,5

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{166909,5}{12,3 \cdot 24} = 565 \text{ грн.}$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць;

$S_у$ – питома площа на одне робоче місце, $м^2$;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пектораль» (м. Кіровоград, вул. Глинки 16) ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 500…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 36 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 18000 грн./м^2. На кожне робоче місце у середньому потрібно $8 м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 18000 = 1152000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 115200 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$I_{нв} = R^1_{сп} \cdot Ц_м, \qquad (7.10)$$

де: $Ц_м$ – ціна меблів для одного робочого місця, грн.

$$I_{нв} = 8 \cdot 3500 = 28000 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми supercomp.kiev.ua за 08.11.23 – джерело https://supercomp.kiev.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		48024
Системний блок		13894
Процесор	AMD Ryzen 5 4500 (100-000000644MPK) AM4, 6 ядер, 12 потоків, 3.6 GHz, 4.1 GHz, TDP - 65 Вт, 7nm, L1: 384KB, L2: 3MB, L3: 8MB	-
Системна плата	ASUS PRIME A320M-K AM4, DDR4, 3200 MHz, LAN - 1 Гбіт/с, D-Sub (VGA), HDMI, 1 x M.2, 4 x Sata 6.0 Gb/s, Micro-ATX	-
Відеокарта	Radeon R7 350 2Gb Afox (AFR7350-2048D5H4-V3) PCI-Express 3.0, 2 ГБ, GDDR5, 128 Bit	-
Жорсткий диск	SSD 2.5" 256GB Mibrand (MI2.5SSD/CA256GBST) 256 GB, 3D TLC NAND, 2.5", SATA III (6Gb/s)	-
Оперативна пам’ять	DDR4 8GB 3200 MHz Dato (DT8G4DLDND32) DDR4, 8 ГБ, В наборі – 1 шт	-
DVD-привод	-	-
Корпус	Vinga CS105B-450W, Miditower, ATX, Micro - ATX, Mini - ITX, PSU - 450 Вт	-
Кардридер внутрішній	Transcend TS-RDF8K2 USB 3.1	-
інше	Клавіатура, мишка	Подару-нок

Продовження таблиці 7.6.

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Монітор	Монітор BenQ GL2780 Black	5702
Принтер лазерний	Canon LBP-6030B (8468B006)	6612
Принтер струминний	Canon PIXMA TS704 з WI-FI (3109C027)	4544
Сканер	Epson Perfection V37	5132
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	8436
Пристрій безперебійного живлення	Powercom BNT-600AP USB	3704

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 - Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість ,шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	9	48024	11448,9	443664,9
Принтер лаз.	2	6612	540	13764
Принтер струм.	1	4544	550	5094
Сканери	1	5132	280	5412
Копіюв. апарат	1	8436	596,5	9032,5
Всього	–	–	–	476967,4

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1152000	-	-
2. Передавальні пристрої	115200	-	-
Всього по групі	1267200	5	63360
Група 4			
3. Обчислювальна техніка	476967,4	-	-
Всього по групі	476967,4	40	190786,96
Нематеріальні активи			
4. Нематеріальні активи	17000	10	1700
Група 5, 6			
5. Вимірювальні пристрої	9031	25	2257,75
6. Транспортні засоби	121875	20	24375
7. Господарський інвентар	31500	25	7875
Всього по групі	162406	-	34507,75
Разом	$К_р$ = 1923573,4		$А_р$ = 290354,71

Примітка: вартість автомобіля ГАЗ Газель взята по даним електронного ресурсу http://www.avtopoisk.ua, що враховуючи курс 36 складає 121875 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot T_{пз}}{N_e}, \quad (7.11)$$

де: N_e – кількість екземплярів програм, шт.

$$З_о = 565 \cdot 154/20 = 4351 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot H_q \cdot 0{,}01, \quad (7.12)$$

де: H_q – норматив додаткової зарплати, %.

$$З_д = 4351 \cdot 10 \cdot 0{,}01 = 435 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $H_c = 22\%$ від суми основної та додаткової зарплати:

$$С_{оц} = 0{,}01 \cdot H_c (З_о + З_д), \quad (7.13)$$

де: H_c – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 22(4351+435) = 1053 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $Н_г$ = 15% від основної зарплати:

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 435 \cdot 15 \cdot 0{,}01 = 65 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.;

$З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.;

$З_{М3}$ – вартість фарби, картриджів, тонеру, грн.;

$N_е$ – кількість екземплярів програм, шт.

Згідно виданих викладачем норм приймаємо 0,5 пачки паперу на місяць розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$ = 200 грн., визначаємо вартість паперу за період розробки $N_м$ = 1 міс:

$$З_{М1} = Ц_п \cdot N_м. \quad (7.16)$$

$$З_{М1} = 0{,}5 \cdot 200 \cdot 1 = 100 \text{ грн.}$$

Згідно норм до вартості запам'ятовуючих пристроїв входить вартість CD дисків за потребою та одного DVD диска для збереження резервної копії програми:

$$З_{М2} = \sum Ц_д, \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 36 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 46 грн./шт.

$$З_{М2} = 46 \cdot 1 = 46 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з., \tag{7.18}$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon LBP-6030B – 574 грн.; картридж для Canon PIXMA TS704 – 558 грн.; відновлення картриджу для Canon i-SENSYS MF217W – 570 грн.

$$З_{М3} = 574+558+570 = 1702 \text{ грн.}$$
$$З_М = (100 + 46 + 1702)/1 = 1848 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \tag{7.19}$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 4351 \cdot 15 \cdot 0{,}01 = 653 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 17 прим.):

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \tag{7.20}$$

де: $A_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$A_м = (290355 \cdot 1)/(20 \cdot 12) = 1210 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн.
1	2	3
1. Основна зарплата виконавців	$З_о$	4351
2. Додаткова зарплата виконавців	$З_д$	435
3. Відрахування на соціальні потреби	$С_{оц}$	1053
4. Загальногосподарські витрати	$Г_{осп}$	65
5. Витрати на матеріали	$З_М$	1848
6. Освоєння нових операційних систем, мов програмування	$О_п$	653
7. Амортизація основних фондів	$А_м$	1210
8. Повна собівартість програмного забезпечення	$С_п$	9615
9. Плановий прибуток	$П_р$	4807,5
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	14422,5
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	2884,5
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	17307

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$C_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$C_п = 4351+435+1053+65+1848+653+1210 = 9615 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності ($P_п$) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 50%.

$$П_р = 0{,}01 \cdot P_п \cdot C_п, \quad (7.22)$$

де: $P_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 9615 = 4807{,}5 \text{ грн.}$$

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

Об’єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах.

Таблиця 7.10 – Розрахунок об’єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	17307
Всього капітальних витрат	–	17307

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми протягом року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування	$З_р$	26572	11810
2. Витрати на електроенергію	$З_{ел}$	205	137
3. Витрати на амортизацію	$З_{ам}$	0	4326,75
Всього витрат за рік	$І$	26777	16273,75

Витрати на обслуговування системи:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_с), \qquad (7.23)$$

де: $Т_р$ – кількість годин обслуговування за рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 180 годин на рік до 80 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\ баз} = 180 \cdot 110 \cdot 1{,}1 \cdot 1{,}22 = 26572\ грн,$$

до:

$$З_{р\ нов} = 80 \cdot 110 \cdot 1{,}1 \cdot 1{,}22 = 11810\ грн.$$

Витрати на електроенергію визначаються з урахуванням споживаємої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \quad (7.24)$$

$$З_{ел\ баз} = 0{,}45 \cdot 120 \cdot 3{,}8 = 205 \text{ грн.}$$

$$З_{ел\ нов} = 0{,}45 \cdot 80 \cdot 3{,}8 = 137 \text{ грн.}$$

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн., за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	17307	–	4326,75
Всього відрахувань	-	–	17307	–	4326,75

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.24)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$E_в = (14423 - 9615) \cdot 20 - (0{,}05 \cdot 1267200 + 0{,}4 \cdot 476967 + \\ + 0{,}2 \cdot 121875 + 0{,}25 \cdot 40531 + 0{,}1 \cdot 17000) = 71964 \text{ грн}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_p^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.25)$$

де: K_p^* – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_е = \frac{656374}{(14423 - 9615) \cdot 20 \cdot 12/1} = 0{,}5 \text{ років.}$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	20
2. Повна собівартість розробленої програми	Грн.	9615
3. Ціна розробленої програми	Грн.	14423
4. Плановий прибуток від реалізації розробленої програми	Грн.	4808
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1923573
7. Загальний прибуток від реалізації програмної продукції	Грн.	96160
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	71964
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	0,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	17307
11.Величина економічного ефекту у користувача програмної продукції	Грн.	6176
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	1,65

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \qquad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (26777\text{-}16274)\text{-}0{,}25 \cdot 17307 = 6176 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \quad (7.28)$$

$$T_{\text{сп}} = \frac{17307}{26777 - 16274} = 1{,}65 \text{ року.}$$

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Забезпечення охорони праці та техніки безпеки є надзвичайно важливою складовою будь-якого підприємства. У зв'язку зі швидкими змінами в галузі інформаційних технологій та підвищеною складністю систем, пов'язаних з управлінням інвентарем та сервісним обслуговуванням, забезпечення техніки безпеки стає ще актуальнішим завданням.

Спроєктована система автоматизованого управління інвентарем та сервісним обслуговуванням, заснована на методах штучного інтелекту, передбачає впровадження нових процесів та технологій на підприємстві. Ця система може відзначитися значними перевагами, але також створює нові технічні та організаційні ризики.

У зв'язку з цим, даний розділ присвячений аналізу, оцінці та розробці необхідних заходів для забезпечення безпеки персоналу, нормального функціонування обладнання та програмного забезпечення нашої системи. Передбачення потенційних ризиків і впровадження заходів з їх запобігання є обов’язковою умовою для успішної реалізації цього проєкту та забезпечення безпеки всіх учасників.

8.2 Аналіз умов праці на робочому місці фахівців

Розглянемо умови праці у приміщенні, в якому працюють фахівці. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 — Розміри приміщення

Найменування	Значення, м
Ширина	4
Довжина	6
Висота	3

Таблиця 8.2 — Площа та об’єм приміщень на одного працівника*

Геометрична характеристика	Одиниця виміру	Нормативне значення	Фактичне значення
Площа, S	$м^2$	не менше 6.0	8
Об’єм, V	$м^2$	не менше 20.0	24

*Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин).

У зазначеному приміщенні працює троє людей. За даними, що наведені у таблицях 8.1 та 8.2, можна зробити висновок, що площа та об’єм приміщення у розрахунку на одне робоче місце програміста відповідають нормативним вимогам ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин»[52].

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Відповідно до Санітарних норм мікроклімату виробничих приміщень ДСН 3.3.6.042-99[53], у даному приміщенні виконується робота категорії Іа, тобто працівники витрачають до 120 ккал/год.

Вологість повітря в приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Таблиця 8.3 — оптимальні та фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Вологість, %	Швидкість повітря, м/с	Температура, °С	Вологість, %	Швидкість повітря, м/с
Холодна	22—24	40—60	0,1	22—23	40—55	0,1
Тепла	23—25	50—70	0,1	24—25	50—65	0,11

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

8.3 Пропозиції щодо підвищення працездатності ІТ-фахівців

У процесі впровадження системи на підприємстві, велике значення надається працездатності та психофізичному стану ІТ-фахівців. Далі наводяться конкретні пропозиції та рекомендації з метою підвищення працездатності:

1) Організація регулярних перерв.

Забезпечення регулярних перерв у роботі є ключовим аспектом підвищення продуктивності та зменшення стресу серед ІТ-фахівців. Пропонується встановити гнучкий графік роботи та забезпечити короткі перерви між робочими циклами для відновлення концентрації та психофізичного стану персоналу.

2) Організація тренінгів з управління стресом.

Враховуючи високий рівень відповідальності та ризику, пов’язаного з роботою з новою системою, пропонується проведення навчань з управління стресом та підвищення психологічної стійкості серед ІТ-фахівців.

3) Створення зон відпочинку та релаксації.

Рекомендується створити спеціальні зони відпочинку та релаксації на робочому місці для відновлення фізичного та психологічного стану.

8.4 Розрахункова частина

Розрахуємо занулення глухозаземленої нейтралі трансформатора приміщення, в якому працюють ІТ-фахівці.

Початкові дані:

- Загальна потужність: P = 5 кВт;
- Кількість електродвигунів: m = 10;
- Потужність освітлювальних приладів: P_o = 3 кВт;
- Довжина магістрального кабеля: L_M = 70 м;
- Довжина розгалудження: L = 22 м;
- Лінійна напруга U = 380 В;
- Фазна напруга $U_ф$ = 220 В.

Визначаємо силу номінального струму електроустановки:

$$I_{\text{ном}} = I_{max} = \frac{P \cdot 1000}{\sqrt{3} \cdot U_{\text{л}} \cdot cos\varphi} \tag{8.1}$$

$$I_{\text{ном}} = I_{max} = \frac{5 \cdot 1000}{\sqrt{3} \cdot 380 \cdot 0{,}85} \approx 8{,}9 \text{ А}$$

де Р − номінальна сумарна потужність електроприладів, кВт;

$U_{\text{л}}$ − лінійна напруга, В;

cosφ − коефіцієнт потужності, приймається в залежності від типу електрообладнання в межах 0,8..0,87.

Визначаємо силу пускового струму електродвигуна:

$$I_{\text{пус}} = 5 \cdot I_{\text{ном}} \tag{8.2}$$

$$I_{\text{пус}} = 5 \cdot 8{,}9 = 44{,}5 \text{ А}$$

Визначаємо номінальну силу струму апарата захисту:

$$\text{I}_{\text{н}} = \frac{\text{I}_{\text{пус}}}{\beta} \tag{8.3}$$

$$\text{I}_{\text{н}} = \frac{44{,}5}{2{,}5} = 17{,}8 \text{ А}$$

де β — коефіцієнт пуску електродвигуна − для легких умов пуску − 2,5..3.

Вибираємо запобіжник ПН 2-100 з плавкою вставкою $\text{I}_{\text{ном}}$ = 50 А. Визначаємо найменше допустиме по умовам спрацьовування захисту значення сили струму короткого замикання:

$$I_{\text{ктіп}} = \text{I}_{\text{н}} \cdot K \tag{8.4}$$

$$I_{\text{ктіп}} = 50 \cdot 3 = 150 \text{ А}$$

де $I_н$ — номінальний струм апарата захисту;

К — коефіцієнт надійності.

Знаходимо переріз провода або кабеля розгалуження з умови допустимого нагрівання:

$$I_{доп} = \frac{I_{вст}}{\alpha} \tag{8.5}$$

$$I_{доп} = \frac{50}{3} \approx 16{,}6 \text{ А}$$

де α — коефіцієнт відповідності, який залежить від умов прокладання і нагляду за мережею : $\alpha = 3$ – для промисливих мереж.

Вибираємо площу перерізу 10 мм^2 (Sφ) при числі проводів і = 4, кабель АВВБ розташований у повітрі. Визначаємо максимальний робочий струм:

$$I_{роб} = I_{max} =$$

$$= K_О \sum_{1}^{n} K_З \cdot I_{ном} = K_О \left(K_З \frac{P \cdot 1000}{\sqrt{3} \cdot U_л \cdot cos\varphi} \cdot m + K_З \frac{P_О \cdot 1000}{\sqrt{3} \cdot U_л \cdot cos\varphi} \right) \tag{8.6}$$

$$I_{роб} = I_{max} \approx$$

$$\approx 0{,}75 \left(0{,}85 \frac{5 \cdot 1000}{1{,}73 \cdot 380 \cdot 0{,}85} \cdot 10 + 0{,}85 \frac{3 \cdot 1000}{1{,}73 \cdot 380 \cdot 0{,}85} \right) \approx$$

$$\approx 60{,}4 \text{ А}$$

де $К_О$ — коефіцієнт одночасності роботи групи електроприймачів;

$К_О$ = 0.7…0.8; Кз = 0.8…0.9;

$К_З$ – коефіцієнт завантажених електродвигунів;

$Р_О$ = 3 кВт — потужність освітлювальної мережі.

Визначається струм короткочасного перевантаження магістрального кабеля:

$$I_{\text{пер}} = K_{\text{О}} \sum_{1}^{n-1} (K_{\text{З}} \cdot I_{\text{ном}}) + I_{\text{пус}} = \tag{8.7}$$

$$= K_{\text{О}} \left(K_{\text{З}} \frac{P \cdot 1000}{\sqrt{3} \cdot U_{\text{л}} \cdot cos\varphi} \cdot n + K_{\text{З}} \frac{P_{\text{О}} \cdot 1000}{\sqrt{3} \cdot U_{\text{л}} \cdot cos\varphi} \right) + I_{\text{пус}}$$

$$I_{\text{пер}} \approx 0{,}75 \left(0{,}85 \frac{5 \cdot 1000}{1{,}73 \cdot 380 \cdot 0{,}85} \cdot 9 + 0{,}85 \frac{30 \cdot 1000}{1{,}73 \cdot 380 \cdot 0{,}85} \right) + 44{,}5 \approx$$

$$\approx 130{,}0643 \text{ А}$$

Струм спрацювання електромагнітного розчеплювача додатково перевіряємо по максимальному струму перевантаження лінії:

$$I_{\text{спр}} \geq 1{,}25 \cdot I_{\text{пер}} \tag{8.8}$$

$$I_{\text{спр}} \geq 1{,}25 \cdot 130{,}0643 \approx 162{,}5804 \text{ А}$$

Приймаємо $I_{\text{спр}}$ = 162,5804 А. Вимикач : А3714Б.

Вибираємо площу перерізу Sφ магістрального кабеля (провідника) по Ідоп. Sφ = 70 mm^2 , − кабель АВРГ прокладений в землі, і = 3 (число проводів). Вибрану площу перерізу перевіряємо для автоматів з електромагнітним розчеплювачем:

$$I_{\text{доп}} \geq \frac{I_{\text{спр}}}{4{,}5} \tag{8.9}$$

$$I_{\text{доп}} \geq \frac{162}{4{,}5} = 36 \text{ А}$$

Проводимо узгодження з номінальним струмом автомата:

$$I_{\text{доп}} = \frac{I_{\text{спр}}}{3} \tag{8.10}$$

$$I_{\text{доп}} = \frac{162}{3} = 54 \text{ А}$$

Значення 36 і 54 А. менше ніж I_{max} = 60,4 А., значить площа перерізу кабеля вибрана правильно. Визначаємо потужність трансформатора:

$$N_{\text{тр}} = \frac{K_{\text{п}} \cdot P_{\text{ном}}}{cos\varphi} \tag{8.11}$$

$$N_{\text{тр}} = \frac{0{,}7 \cdot 53}{0{,}8} = 46{,}375 \text{ кВ} \cdot \text{А}$$

де $P_{ном}$ − сумарна потужність електроприймачів, кВт;

cosφ − середній коефіцієнт потужності електроприймачів (0,8);

$K_п$ − коефіцієнт попиту (0,7).

Одержане значення потужності трансформатора округляємо до ближчого стандартного значення. Визначаємо опір трансформатора Z_T. Вибираємо трансформатор на 40 кВА (Z_T = 0,562 Ом). визначаємо орієнтовно площу перерізу провідника. Для магістрального кабеля:

$$Sn_1 \geq 0{,}5 \cdot S\varphi \tag{8.12}$$

$$Sn_1 \geq 0{,}5 \cdot 70 = 35\text{мм}^2$$

Визначаємо для розгалуження:

$$Sn_2 \geq 0{,}5 \cdot 10 = 5\text{мм}^2$$

Округляємо ці значення до найближчих більших 35 мм^2 (Sn_1), і 6 мм^2 (Sn_2). Визначаємо активний і індуктивний опір фазного і нульового захисного провідників на дільницях 1 і 2:

$$R_{\Phi} = \rho \frac{L_{\text{М}}}{S\phi_1} + \rho \frac{L}{S\phi_2}$$

(8.13)

$$R_{\Phi} = 0{,}028\frac{70}{70} + 0{,}028\frac{22}{10} = 0{,}0896 \text{ Ом}$$

$$R_n = \rho\frac{L_{\text{М}}}{Sn_1} + \rho\frac{L}{Sn_2} \tag{8.14}$$

$$R_{\Phi} = 0{,}028\frac{70}{35} + 0{,}028\frac{22}{6} \approx 0{,}1586 \text{ Ом}$$

Для окремо проложених нульових провідників иого приимають рівним 0,6 Ом/км. При прокладці кабелем, або в сталєвих трубах індуктивним опором нехтують.

Знаходимо дійсне значення (модуль) струма однофазного короткого замикання:

$$I_{\text{кр}} = \frac{U_{\text{ф}}}{\frac{Z_{\text{Т}}}{3} + \sqrt{\left(R_{\text{ф}} + R_n\right)^2 + (X\text{ф} + Xn + X'n)^2}} \tag{8.15}$$

$$I_{\text{кр}} = \frac{220}{\frac{0{,}562}{3} + \sqrt{(0{,}0896 + 0{,}1586)^2}} \approx 505{,}13 \text{ А}$$

Визначення максимальноі напруги на корпусі обладнання відносно землі при замиканні фази на корпус.

$$U_{\text{к}max} = I_{\text{кр}} \cdot Z_{\text{н}} \leq U_{\text{дан.д.}} \tag{8.16}$$

$$U_{\text{к}max} = 505{,}13 \cdot 0{,}1586 \approx 80{,}11 \text{ В} > 36 \text{ В}$$

де $U_{\text{дан.д.}}$ — це допустима напруга, що нормується по ГОСТ 12.1.038-82. При часі дії більше 1 с. приймається 36 В.

$Z_{\text{н}}$ — повний опір нульового провідника;

Умова не виконується. Необхідно збільшити перерізи Sn_1 та Sn_2 до $S\text{ф}_1$ та $S\text{ф}_2$ та зробити перерахунок:

$$R_n = 0{,}028\frac{70}{70} + 0{,}028\frac{22}{10} = 0{,}0896 \text{ Ом}$$

$$I_{\text{кр}} = \frac{220}{\frac{0{,}562}{3} + \sqrt{(0{,}0896 + 0{,}0896)^2}} = 600{,}21 \text{ А}$$

$$U_{\text{к}max} = 600{,}21 \cdot 0{,}0896 \approx 53{,}78 \text{ В} > 36 \text{ В}$$

Умова не виконується, необхідно або замінити запобіжник з плавкою вставкою на автоматичний вимикач із струмовим реле, що дає можливість зменшити час замикання на корпус і підвищити допустиму напругу на корпусі або застосувати повторне заземлення нульового захисного провідника. Повторне заземлення нульового захисного провідника:

$$R_n = \frac{U_{\text{доп}} \cdot R_0}{I_{\text{кр}} \cdot Z_{\text{н}} - U_{\text{доп}}} \tag{8.17}$$

$$R_n = \frac{36 \cdot 4}{600{,}21 \cdot 0{,}0896 - 36} \approx 8{,}09 \text{ Ом}$$

8.5 Висновки

У рамках даного розділу було проведено аналіз питань охорони праці та техніки безпеки на підприємстві.

На основі аналізу було виявлено наступні ключові висновки:

– Ефективне управління ризиками важливе для зменшення можливостей виникнення аварійних ситуацій.

– Охорона праці та техніка безпеки повинні бути постійно оновлюваними та покращуваними відповідно до змін у виробничих процесах та обладнанні.

– Забезпечення охорони праці та техніки безпеки сприяє збереженню здоров'я та життя працівників, а також зменшенню збитків для підприємства внаслідок можливих аварій та надзвичайних ситуацій.

9 ОСНОВНІ ВИСНОВКИ

В результаті проведеного дослідження та реалізації системи автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві на основі методів штучного інтелекту виявлено ряд ключових висновків, які визначають ефективність та практичність запропонованого рішення.

– Ефективність системи автоматизованого управління.

Розроблена система демонструє високий рівень ефективності управління інвентарем на підприємстві. Використання методів штучного інтелекту дозволяє оптимізувати процеси управління запасами та підтримувати необхідний рівень інвентаризації.

– Підвищення точності та швидкості обслуговування.

Система сприяє підвищенню точності та швидкості обслуговування клієнтів завдяки автоматизованій обробці замовлень та ефективному управлінню інформацією про наявність товарів на складі.

– Оптимізація запасів інвентарю.

Впровадження системи сприяє оптимізації запасів інвентарю, уникненню надмірності та недостачі товарів, що призводить до зниження витрат та підвищення рентабельності підприємства.

– Зростання рентабельності підприємства.

Результати роботи системи підтверджують зростання рентабельності підприємства завдяки ефективному управлінню інвентарем, покращенню обслуговування клієнтів та оптимізації бізнес-процесів.

– Важливість використання методів штучного інтелекту.

Дослідження підтверджує важливість використання методів штучного інтелекту в управлінні інвентарем. Це дозволяє системі адаптуватися до змін у попиті, робити точні прогнози та максимально ефективно використовувати ресурси.

Узагальнюючи вищезазначене, можна стверджувати, що розроблена система є перспективною та може стати важливим інструментом для оптимізації бізнес-процесів підприємства.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Правила оформлення графічних матеріалів у електронному вигляді (у вигляді слайдів) : додаток : метод. рекоменд. до виконання й захисту випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти : для студентів спец. 122 “ Комп'ютерні науки ” / [уклад. : О. А. Смірнов, В. С. Гермак, Є. В. Мелешко [та ін.] ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т, каф. кібербезпеки та прог. забезпеч. - Кропивницький : ЦНТУ, 2023. - 12 с.

2. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» Системи управління, навігації та зв'язку, 2022, No 3(69). С. 93-98. 2022. (Фахове видання. Категорія «Б»)

3. Методичні вказівки до виконання й захисту магістерської роботи : для студ. спец. 122 “Комп’ютерні науки” / [уклад. : О. А. Смірнов, В. С. Гермак, Є. В. Мелешко [та ін.] ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т, каф. кібербезпеки та прог. забезпеч. – Кропивницький : ЦНТУ, 2023. – 74 с.

4. Кавун С. В. Системи штучного інтелекту. Навчальний посібник. / Кавун С.В., Смірнов О.А., Сорбат І.В., Мелешко Є.В., Коваленко О.В. – Кіровоград: Вид. КНТУ, 2013. – 336 с. (укр. мовою)

5. Методи та системи штучного інтелекту: Навчальний посібник для студентів напряму підготовки 6.050101 «Комп’ютерні науки» / Уклад. : А.С. Савченко, О. О. Синельніков. – К. : НАУ, 2017. – 190 с.

6. Комп’ютерні системи штучного інтелекту : метод. вказ. до викон. лаб. робіт студ. ден. та заоч. форми навч. спец. 123 "Комп’ютерна інженерія" та 122 "Комп’ютерні науки та інформаційні технології" / М-во освіти і науки України, Кіровоград. нац. техн. ун-т, каф. програмування та захисту інформації; [укл. Є. В. Мелешко]. - Кіровоград : КНТУ, 2016. - 61 с.

7. Лівшиц Д. Інвентаризація. Практичний посібник / Давид Лівшиц. – Київ: Центр учбової літератури, 2019. – 140 с.

8. Коваленко, А. С. Розробка структури бази даних для обліку технічного стану елементів інтегрованої інформаційної системи з урахуванням вимог споживачів інформації / А. С. Коваленко, О. А. Смірнов, О. В. Коваленко // Системи обробки інформації. — Харків : ХУПС, 2015. — № 1. — С. 75-79.

9. Коваленко А.С. Підсистема технічної діагностики для автоматизації процесів керування в інтегрованих інформаційних системах / А.С. Коваленко , О.А.Смірнов, О.В. Коваленко // Системи озброєння і військова техніка.– Х.: ХУПС, 2014. – No 1(37). – С. 126-129.

10. "Штучний інтелект." Вікіпедія. 22 лис 2023, 05:54 UTC. 22 лис 2023, 14:53<https://uk.wikipedia.org/w/index.php?title=%D0%A8%D1%82%D1%83%D1%87%D0%BD%D0%B8%D0%B9_%D1%96%D0%BD%D1%82%D0%B5%D0%BB%D0%B5%D0%BA%D1%82&oldid=40975202>.

11. Логістичне управління запасами: навчально-методичний комплекс дисципліни [Електронний ресурс] : навч. посіб. для студ. спеціальності 073 «Менеджмент» / КПІ ім. Ігоря Сікорського ; уклад.: І.С. Луценко. – Електронні текстові дані (1 файл: 1, 96 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2021. – 69 с.

12. Методичні рекомендації до виконання розділу "Економічна ефективність розробленої програми" випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти для студентів спеціальностей: 123 “Комп’ютерна інженерія” та 122 “Комп’ютерні науки” - Кропивницький: ЦНТУ, 2022, 75 с.

13. Т.В. Смірнова, О.М. Дрєєв, О.А. Смірнов, «Формування абстрактних експертних систем на основі досліджень відомих експертних систем», XXI Міжнародний науково-практичний семінар «Комбінаторні конфігурації та їх застосування», м. Кропивницький, 17-18 травня, 2019, с. 143-147.

14. Програмування на мові Python : метод. вказівки до викон. лаб. робіт студ. денної та заочної форми навч. спец. 123 "Комп’ютерна інженерія", 125

"Кібербезпека" / уклад. Є. В. Мелешко ; М-во освіти і науки України, Центральноукр. нац. техн. ун-т, каф. програмування та захисту інформації. - Кропивницький : ЦНТУ, 2017. - 58 с.

15. Фратавчан В.Г., Фратавчан Т.М., Лукашів Т.О., Літвінчук Ю.А., Методи та системи штучного інтелекту: навчальний посібник. Чернівці: ЧНУ, 2023, – 114 с.

16. Експертні системи : метод. вказівки до викон. лаб. робіт студ. денної та заочної форми навч. спец. 123 "Комп’ютерна інженерія", 122 "Комп’ютерні науки та інформаційні технології", 125 "Кібербезпека" / уклад. В. Д. Хох, Є. В. Мелешко, О. М. Дрєєв ; М-во освіти і науки України, Кіровоград. нац. техн. ун-т, каф. програмування та захисту інформації. - Кіровоград : КНТУ, 2016. - 35 с.

17. Коваленко А.С. Структура системи технічної діагностики інтегрованих інформаційних систем / А.С. Коваленко, О.А. Смірнов, О.В. Коваленко // Збірник наукових праць Кіровоградського національного технічного університету / техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. – Кіровоград: Вид-во КНТУ, 2014. – Вип. 27. – С. 245-251.

18. Коваленко А.С. Основні складові та функції системи технічної діагностики інтегрованих інформаційних систем / Коваленко А.С. // Інформаційні технології та комп’ютерна інженерія: наук.-практ. конф., 4 груд. 2014 р., м. Кіровоград: зб. тез доп. – Кіровоград: КНТУ, 2014. – С. 236.

19. Штучний інтелект. Частина 2. Методичні вказівки до виконання лабораторних робіт студентами денної та заочної форми навчання спеціальностей 122 "Комп’ютерні науки" та 123 "Комп’ютерна інженерія" / Укл.: Є.В. Мелешко – Кропивницький: ЦНТУ, 2022. – 43 с.

20. Ярмоленко Л. І. УДОСКОНАЛЕННЯ СИСТЕМИ УПРАВЛІННЯ ЗАПАСАМИ ДИСТРИБ’ЮТОРСЬКОЇ ФІРМИ / Л. І. Ярмоленко, Т. В. Чумак. // Академічний огляд. – 2017. – №1. – С. 83–91.

21. Дюбко, Г. Ф. Нейронечітка модель і програмний комплекс формування баз знань експертних систем / Г. Ф. Дюбко, О. А. Смірнов, В. В.

Вакулін // Збірник наукових праць Кіровоградського національного технічного університету. Техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. - Кіровоград: КНТУ, 2008. - Вип. 20. - С. 298-305.

22. Albayrak Ünal, Ö., Erkayman, B. & Usanmaz, B. Applications of Artificial Intelligence in Inventory Management: A Systematic Review of the Literature. Arch Computat Methods Eng 30, 2605–2625 (2023). https://doi.org/10.1007/s11831-022-09879-5

23. Lutz M. Learning Python, 5th Edition / Mark Lutz. – Sebastopol: O'Reilly Media, 2013. – 1594 с.

24. Основи програмування. Python. Частина 1 [Електронний ресурс]: підручник для студ. спеціальності 122 "Комп'ютерні науки", спеціалізації "Інформаційні технології в біології та медицині" / А. В. Яковенко ; КПІ ім. Ігоря Сікорського. – Електронні текстові данні (1 файл: 1,59 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2018. – 195 с.

25. Moroney L. AI and Machine Learning for On-Device Development: A Programmer's Guide. 1st Ed. / Laurence Moroney. – Sebastopol: O'Reilly Media, 2021. – 300 с.

26. Relph G. Inventory Management: Advanced Methods for Managing Inventory within Business Systems 1st Edition / G. Relph, C. Milner. – London: Kogan Page, 2016. – 280 с.

27. Lakshmanan V. Practical Machine Learning for Computer Vision: End-to-End Machine Learning for Images 1st Edition / V. Lakshmanan, M. Görner, R. Gillard. – Sebastopol: O'Reilly Media, 2021. – 480 с.

28. Алєксєєв, Д. С. Штучні інтелектуальні системи в гнучкому автоматизованому виробництві / Д. С. Алєксєєв, А. В. Татаров // Наукові записки : зб. наук. пр. - Кіровоград : КНТУ, 2012. - Вип. 12, ч. 1. - С. 15-16.

29. Методичні рекомендації до виконання лабораторних робіт з дисципліни «Web-програмування» для студентів денної та заочної форми навчання спеціальностей 123 «Комп'ютерна інженерія», 125 «Кібербезпека» та

122 «Комп’ютерні науки» / [уклад. : Є. В. Мелешко, В. В. Босько, Л. В. Константинова] ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. - Кропивницький : ЦНТУ, 2023. - 87 с.

30. Web-програмування. Частина 1 (frontend) : навч. посіб. / В. В. Босько, Л. В. Константинова, К. М. Марченко, О. С. Улічев ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. - Кропивницький : ЦНТУ, 2022. - 208 с.

31. Локазюк В.М. Надійність, контроль, діагностика і модернізація ПК: Посібн. / В.М. Локазюк, Ю.Г. Савченко. – К.: Видавничий центр «Академія», 2004. – 376 с.

32. Безпека інформаційних технологій : метод. рекомендації до виконання лабораторних робіт для студент. денної форми навч. галузі 12 "Інформаційні технології" / [уклад. : О. А. Смірнов, К. О. Буравченко, Т. В. Смірнова та ін.] ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т., каф. кібербезпеки та програм. забезпеч. / Кропивницький : ЦНТУ, 2023. - 48 с.

33. Вступ до кібербезпеки : метод. рекомендації до виконання лабораторних робіт для студент. денної форми навчання галузі 12 "Інформаційні технології" / [уклад. : О. А. Смірнов, К. О. Буравченко, Т. В. Смірнова та ін.] ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т., каф. кібербезпеки та програм. забезпечення. - Кропивницький : ЦНТУ, 2022. - 155 с.

34. Методичні рекомендації до виконання лабораторних робіт з навчальної дисципліни «Бази даних» (1 частина) : для студ. денної та заочної форми навч. за спец. : 123 «Комп’ютерна інженерія», 125 «Кібербезпека» / [уклад. : В. В. Босько, Л. В. Константинова] ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т, каф. кібербезпеки та програмного забезпечення. - Кропивницький : ЦНТУ, 2020. - 77 с.

35. Сучасні інформаційні технології та програмне забезпечення комп’ютерних систем : зб. тез доп. : всеукр. наук.-практ. конф. студ. та магістр., 27-29 бер. 2013 року, Кіровоград / М-во освіти і науки, молоді та спорту України, Кіровоград. нац. техн. ун-т, Нац. техн. ун-т “Харківський політехнічний інститут” та ін. ; [відп. за вип. : О. П. Доренський]. - Кіровоград : КОД, 2013. - 208 с.

36. Ertaul L. Implementation and Performance Analysis of PBKDF2, Bcrypt, Scrypt Algorithms / Ertaul Levent – CSU East Bay, Hayward, CA, USA., 2016. – 7 с.

37. Чорний, Є. О. Вдосконалення менеджменту бізнес-процесів на підприємстві : кваліфікаційна магістерська робота : спец. 073 «Менеджмент» ОПП «Менеджмент бізнес-організацій» / наук. кер. О. М. Гуцалюк ; Центральноукраїн. нац. техн. ун-т. - Кропивницький : ЦНТУ, 2022. – 76 с.

38. Методичні рекомендації до виконання лабораторних робіт з навчальної дисципліни «Організація баз даних» (2 частина) : для студ. денної та заочної форми навч. за спец. : 123 «Комп'ютерна інженерія», 125 «Кібербезпека» / [уклад. : В. В. Босько, Л. В. Константинова] ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т, каф. кібербезпеки та програмного забезпечення. - Кропивницький : ЦНТУ, 2020. - 60 с.

39. Web технології в управлінні та проектуванні. Ч. 1: метод. вказ. для студ. спеціальностей 151 "Автоматизація та комп'ютерно-інтегровані технології" спеціалізації "Комп'ютеризовані системи управління та автоматика" , "Комп'ютеризовані та робототехнічні системи" ; 123 "Комп'ютерна інженерія" спеціалізація "Комп'ютерні системи та мережі" / [уклад. В. В. Босько, Ю. М. Пархоменко]; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький : ЦНТУ, 2017. – 76 с.

40. Шевченко, Д. Г. Життєвий цикл розробки програмного забезпечення / Д. Г. Шевченко, О. Л. Левошко // Сучасні інформаційні технології та програмне забезпечення комп'ютерних систем : Всеукр. студ. наук.-практ. семінар, 21–23 бер. 2012 р., м. Кіровоград : зб. тез доп. / М-во освіти і науки, молоді та спорту України, Кіровоград. нац. техн. ун-т. — Кіровоград: КОД, 2012. — С. 74–75.

41. Коваль, В. О. Захист персональних даних в Інтернет / В. О. Коваль, Л. В. Константинова // Інформаційна безпека та комп'ютерні технології : зб. тез доп. : II Міжнар. наук.-практ. конф., 20-22 квіт. 2017 року, м. Кропивницький. – Кропивницький : ЦНТУ, 2017. – С 51–52.

42. Комп'ютерна інженерія і кібербезпека: досягнення та інновації : матеріали II Всеукр. наук.-практ. конф. здобувачів вищої освіти й молодих

учених, м. Кропивницький, 25–27 листоп. 2020 р. / М–во освіти і науки України, Держ. наук. установа “Інститут модернізації змісту освіти”, Центральноукраїн. нац. техн. ун–т ; [відп. за вип. О. П. Доренський]. — Кропивницький : ЦНТУ, 2020. — 147 с.

43. Інформаційна безпека в комп’ютерних мережах : навч. посіб. / О. А. Смірнов, О. К. Конопліцька-Слободенюк, С. А. Смірнов [та ін.] ; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. - Кропивницький : Лисенко В.Ф., 2020. – 295 с.

44. Каплун, В. А. Захист програмного забезпечення. Частина 2 : навчальний посібник / В. А. Каплун, О. В. Дмитришин, Ю. В. Баришев – Вінниця : ВНТУ, 2014. – 105 с.

45. Gang Cheng and Nirwan Ansari. A New Heuristics For Finding The Delay Constrained Least Cost Path // IEEE GLOBECOM – 2003, P. 3711-3715.

46. Якименко І.З. // Опорний конспект лекцій з дисципліни «Безпека програм та даних», для студентів спеціальності «Кібербезпека». – Тернопіль, 2007. – 50 с.

47. Хрущ Л.З. Економіка програмного забезпечення : навчальний посібник. Івано-Франківськ : ЛІК, 2018. 103 с.

48. Закон України «Про захист інформації в інформаційно-телекомунікаційних системах» [Електронний ресурс]. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2594-15

49. Методичні рекомендації до виконання розділу "Заходи з охорони праці та техніки безпеки" випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти для здобувачів вищої освіти спеціальності 122 “Комп’ютерні науки” [укл. О.В. Оришака, К.М. Марченко]; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. - Кропивницький : ЦНТУ, 2022. – 20 с.

50. Основи охорони праці. Навчальний посібник / укладачі: О.В. Оришака, Г.П. Горбачова, К.М. Марченко − Кропивницький: 2022. − 175 с.: іл.

51. Охорона праці. Ч. 2. Занулення : метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ−сумісного типу / [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака, А. Е. Солових, С. Е. Катеринич] ; Мін-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. - 2−ге вид., перероб. та доп. - Кропивницький : ЦНТУ, 2019. - 27 с.

52. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-007-98. - Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98 (дата звернення: 11.11.2023).

53. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. - Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99

54. Економіка й організація діяльності об‘єднань підприємств : метод. вказ. до вивч. курсу для студ. спец. 7.050107 Економіка підприємства / [уклад.: Н. П. Сисоліна, Л. В. Вдовиченко, Г. В. Савеленко] ; М-во освіти і науки України, Кіровоград. нац. техн. ун-т. – Кіровоград : Код, 2010. – 36 с.

Додаток А

(обов’язковий)

Технічне завдання

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві на основі методів штучного інтелекту.

2 Підстава для розробки

Підставою для розробки служить завдання на випуску кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (наказ № 37-13 від 04.08.2023 р.).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем є програмна реалізація системи автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві на основі методів штучного інтелекту.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є розробки, які ведуться кафедрою кібербезпеки та програмного забезпечення і стосовні до теми бібліографічні джерела.

5 Технічні вимоги

5.1 Вміст проекту

Складовими розробки є:

– аналіз існуючих систем автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві на предмет їхньої відповідності сучасним вимогам;

– вибір і обґрунтування методики побудови системи і засобів її реалізації;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві на основі методів штучного інтелекту;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4. Вимоги до архітектури

Компонент, що розробляється повинен використовувати архітектуру клієнт/сервер, реляційні БД і об'єктно-орієнтовані засоби розробки.

5.5 Вимоги до надійності

Компонент повинен використати існуючі угоди по стандартним викликам процедур, функцій, засобів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Автоматизовані робочі місця користувачів системи повинні задовольняти наступним умовам експлуатації:

- температура повітря: 18-22 °С;
- відносна вологість повітря при 20 °С до 80%;
- атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Компонент повинен бути реалізований на ПЕОМ типу IBM PC в операційному середовищі Windows 10 і орієнтований на сумісні з цією

платформою зовнішні пристрої, мережне обладнання і прикладне програмне забезпечення.

5.8 Вимоги до інформаційної та програмної сумісності

Сумістність програмного забезпечення повинна бути забезпечена за рахунок його реалізації засобами об'єктно-орієнтованої СУБД, працюючої під управлінням ОС Windows 10 (рекомендується СУБД Mysql).

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена в вигляді опису структури даних, схем і описів алгоритмів, інструкції користувача, а також текстів вхідних модулів програмного забезпечення в відповідності з ЄСПД.

7 Економічні вимоги

1. Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.
2. Виконати розрахунок кошторису витрат на реалізацію прийнятого варіанту з урахуванням розцінок на 1 жовтня 2023 р.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинен бути розглянутий аналіз умов праці на робочому місці програміста.

9 Перелік документів, що розробляються

- Наукова новизна — 1 аркуш.
- Структурна схема системи — 1 аркуш.
- Функціональна схема системи — 1 аркуш.
- Діаграма процесів — 1 аркуш.
- Блок-схема алгоритму роботи програми — 2 аркуша.
- Показники економічної ефективності — 1 аркуш.
- Пояснювальна записка — 126 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

10.3 Розробка функціональних схем, блок-схем алгоритмів роботи програмного забезпечення компоненту.

10.4 Побудова схем взаємодії структур даних.

10.5 Створення прототипу компоненту. Створення програмного продукту.

10.6 Відлагодження компоненту, аналіз отриманих результатів.

10.7 Робота над питаннями охорони праці і техніки безпеки.

10.8 Розрахунки по техніко-економічному обґрунтуванню (остаточні).

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 02.12.2023 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 11.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи

за другим (магістерським) рівнем вищої освіти

_____ Буравченко К.О.

Дослідження та програмна реалізація системи автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві на основі методів штучного інтелекту

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 52

Літера: РП

Кропивницький — 2023 року

Основна програма

Файл main.py

```
"""Основний файл проєкту"""

from PL.pl import main

main()
```

Шар представлення

Файл PL/app.py — об’єкт застосунку

```
"""Об'єкт застосунку"""

from flask import Flask

app = Flask(__name__)
```

Файл PL/copyright.py — авторське право

```
"""Авторське право"""

from flask import Blueprint, render_template

from PL.app import app

copyright_bp = Blueprint("copyright", __name__)


@app.route("/copyright", methods=["GET"])
def copyright_info():
    """Авторське право"""
    return render_template("copyright.html")
```

Файл PL/index.py — головна сторінка

```
"""Головне меню"""

from flask import Blueprint, redirect, render_template, session, url_for

from BLL.auth import check_account

from PL.app import app

index_bp = Blueprint("index", __name__)


@app.route("/", methods=["GET"])
def index():
    """Головне меню"""
    if not check_account(session.get("account")):
        return redirect(url_for("login"))

    return render_template("index.html")
```

Файл PL/login.py — вхід

```
"""Вхід"""

from flask import Blueprint, render_template, redirect, request, session, url_for

from BLL.auth import auth, check_account

from PL.app import app

login_bp = Blueprint("login", __name__)


@app.route("/login", methods=["GET", "POST"])
def login():
    """Вхід"""
    if check_account(session.get("account")):
        return redirect(url_for("index"))

    error = None

    if request.method == "POST":
        username = request.form.get("username")
        password = request.form.get("password")
        account = auth(username, password)

        if account:
            session["account"] = account
            return redirect(url_for("index"))
        error = "Неправильні дані."
    return render_template("login.html", error=error)
```

Файл PL/logout.py — вихід

```
"""Вихід"""

from flask import Blueprint, redirect, session, url_for

from BLL.auth import check_account

from PL.app import app

logout_bp = Blueprint("logout", __name__)


@app.route("/logout", methods=["GET"])
def logout():
    """Вихід"""
    if check_account(session["account"]):
        session.pop("account")
    return redirect(url_for("index"))
```

Файл PL/orders.py — замовлення

```
"""Список замовлень"""

from flask import Blueprint, render_template, redirect, request, session, url_for

from BLL.auth import check_account
from BLL.orders import create_order as bll_create_order
from BLL.orders import read_order as bll_read_order
from BLL.orders import read_orders as bll_read_orders
from BLL.orders import read_statuses as bll_read_statuses
from BLL.orders import update_order as bll_update_order
from BLL.orders import delete_order as bll_delete_order
from BLL.orders import read_ordered_products as bll_read_ordered_products
from BLL.orders import update_ordered_product as bll_update_ordered_product

from PL.app import app

orders_bp = Blueprint("orders", __name__)


@app.route("/orders/create", methods=["GET", "POST"])
def create_order():
    """Додавання нового замовлення"""
    if not check_account(session.get("account")):
        return redirect(url_for("login"))

    error = None

    if request.method == "POST":
        error = bll_create_order(request.form)
        if not error:
            return redirect(url_for("orders"))

    return render_template(
        "edit_order.html", error=error, discount=0, statuses=bll_read_statuses()
    )


@app.route("/orders", methods=["GET"])
def orders():
    """Список замовлень"""
    if not check_account(session.get("account")):
        return redirect(url_for("login"))
```

```
    order_list = bll_read_orders()
    return render_template("orders.html", order_list=order_list)


@app.route("/orders/update/<order_id>", methods=["GET", "POST"])
def edit_order(order_id):
    """Редагування замовлення"""
    if not check_account(session.get("account")):
        return redirect(url_for("login"))

    order = bll_read_order(order_id)
    if not order:
        return redirect(url_for("orders"))

    error = None

    if request.method == "POST":
        error = bll_update_order(order_id, request.form)
        if not error:
            return redirect(url_for("orders"))

    return render_template(
        "edit_order.html",
        date=order[0],
        customer=order[1],
        repair_cost=order[2],
        discount=order[3],
        details=order[5],
        statuses=bll_read_statuses(order[4]),
        error=error,
    )


@app.route("/orders/delete/<order_id>", methods=["GET"])
def delete_order(order_id):
    """Видалення замовлення"""
    if not check_account(session.get("account")):
        return redirect(url_for("login"))

    bll_delete_order(order_id)
    return redirect(url_for("orders"))
```

```
@app.route("/orders/products/<order_id>", methods=["GET", "POST"])
def ordered_products(order_id):
    """Товари у замовленні"""
    if not check_account(session.get("account")):
        return redirect(url_for("login"))

    order = bll_read_order(order_id)
    if not order:
        return redirect(url_for("orders"))

    error = None

    if request.method == "POST":
        error = bll_update_ordered_product(order_id, request.form)

    return render_template(
        "ordered_products.html",
        ordered_products=bll_read_ordered_products(order_id),
        error=error,
    )
```

Файл PL/password.py — зміна пароля

```
"""Зміна пароля"""

from flask import Blueprint, render_template, redirect, request, session, url_for

from BLL.auth import check_account, change_password

from PL.app import app

password_bp = Blueprint("password", __name__)


@app.route("/password", methods=["GET", "POST"])
def password():
    """Зміна пароля"""
    if not check_account(session.get("account")):
        return redirect(url_for("index"))

    error = None

    if request.method == "POST":
        if change_password(session.get("account"), request.form):
            return redirect(url_for("index"))
        error = "Неправильні дані."
    return render_template("password.html", error=error)
```

Файл PL/pl.py — основний файл шару представлення

```
"""Основний файл проєкту"""

import secrets

from BLL.init import init

from PL.app import app
from PL.copyright import copyright_bp
from PL.index import index_bp
from PL.login import login_bp
from PL.logout import logout_bp
from PL.orders import orders_bp
from PL.password import password_bp
from PL.products import products_bp
from PL.stats import stats_bp


def main():
    """Створює застосунок Flask"""

    init()

    app.secret_key = secrets.token_hex(32)
    app.config["SESSION_TYPE"] = "memcached"
    app.jinja_env.trim_blocks = True
    app.jinja_env.lstrip_blocks = True

    app.register_blueprint(copyright_bp)
    app.register_blueprint(index_bp)
    app.register_blueprint(login_bp)
    app.register_blueprint(logout_bp)
    app.register_blueprint(orders_bp)
    app.register_blueprint(password_bp)
    app.register_blueprint(products_bp)
    app.register_blueprint(stats_bp)

    app.run()
```

Файл PL/products.py — товари

```
"""Список товарів"""

from flask import Blueprint, render_template, redirect, request, session, url_for

from BLL.auth import check_account
from BLL.products import create_product as bll_create_product
from BLL.products import read_product as bll_read_product
from BLL.products import read_products as bll_read_products
from BLL.products import update_product as bll_update_product
from BLL.products import delete_product as bll_delete_product

from PL.app import app

products_bp = Blueprint("products", __name__)


@app.route("/products/create", methods=["GET", "POST"])
def create_product():
    """Додавання нового товару"""
    if not check_account(session.get("account")):
        return redirect(url_for("login"))

    error = None

    if request.method == "POST":
        error = bll_create_product(request.form)
        if not error:
            return redirect(url_for("products"))

    return render_template("edit_product.html", error=error)


@app.route("/products", methods=["GET"])
def products():
    """Список товарів"""
    if not check_account(session.get("account")):
        return redirect(url_for("login"))

    return render_template("products.html", product_list=bll_read_products())


@app.route("/products/update/<product_id>", methods=["GET", "POST"])
```

```
def update_product(product_id):
    """Редагування товару"""
    if not check_account(session.get("account")):
        return redirect(url_for("login"))

    product = bll_read_product(product_id)
    if not product:
        return redirect(url_for("products"))

    error = None

    if request.method == "POST":
        error = bll_update_product(product_id, request.form)
        if not error:
            return redirect(url_for("products"))

    return render_template(
        "edit_product.html",
        vendor=product[0],
        name=product[1],
        price=product[2],
        amount=product[3],
        description=product[4],
        error=error,
    )


@app.route("/products/delete/<product_id>", methods=["GET"])
def delete_product(product_id):
    """Видалення товару"""
    if not check_account(session.get("account")):
        return redirect(url_for("login"))

    bll_delete_product(product_id)
    return redirect(url_for("products"))
```

Файл PL/stats.py — статистика

```
"""Статистика"""

from flask import Blueprint, render_template, redirect, request, session, url_for

from BLL.auth import check_account
from BLL.orders import read_orders as bll_read_orders
from BLL.orders import read_best_customers as bll_read_best_customers
from BLL.products import read_products as bll_read_products

from BLL.dates import validate_dates as bll_validate_dates

from PL.app import app

stats_bp = Blueprint("stats", __name__)


@app.route("/stats", methods=["GET", "POST"])
def stats():
    """Статистика"""
    if not check_account(session.get("account")):
        return redirect(url_for("login"))

    dated = False

    low_amount = bll_read_products(max_amount=50)
    new_orders = bll_read_orders(only_new=True)

    if request.method == "POST":
        date_from = request.form.get("date_from")
        date_to = request.form.get("date_to")
        if bll_validate_dates(date_from, date_to):
            dated = True

    if dated:
        high_demand = bll_read_products(
            demand_sort=1, date_from=date_from, date_to=date_to
        )
        low_demand = bll_read_products(
            demand_sort=2, date_from=date_from, date_to=date_to
        )
        best_customers = bll_read_best_customers(date_from=date_from,
date_to=date_to)
```

```
else:
    high_demand = bll_read_products(demand_sort=1)
    low_demand = bll_read_products(demand_sort=2)
    best_customers = bll_read_best_customers()

return render_template(
    "stats.html",
    period=f"Відображення даних з {date_from} по {date_to}"
    if dated
    else "Відображення загальних даних",
    high_demand=high_demand,
    low_demand=low_demand,
    best_customers=best_customers,
    low_amount=low_amount,
    new_orders=new_orders,
)
```

Файл PL/static/style.css — стиль сторінок

```
table, th, td {
    border: 1px solid black;
    border-collapse: collapse;
}
```

Файл PL/templates/copyright.html — шаблон сторінки авторського права

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="utf-8">
    <title>Авторське право - Випускна кваліфікаційна робота</title>
</head>
<body>
    <p>МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ<br>
    Центральноукраїнський національний технічний університет<br>
    Кафедра кібербезпеки та програмного забезпечення</p>

    <p>Тема: Дослідження та програмна реалізація системи автоматизованого управління інвентарем та сервісним обслуговуванням на підприємстві на основі методів штучного інтелекту<br>
    Спеціальність: 122 «Комп’ютерні науки»<br>
    Освітній ступінь: магістр<br>
    Виконав: Михайленко Георгій Васильович<br>
    Науковий керівник: Буравченко Костянтин Олегович<br>
    Кропивницький - 2023</p>

</body>
</html>
```

Файл PL/templates/edit_order.py — шаблон сторінки редагування замовлення

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="/static/style.css">
    <title>Замовлення - Випускна кваліфікаційна робота</title>
</head>
<body>
    {% if error %}
    <p>{{ error }}</p>
    {% endif %}
    <form method="POST">
        <label for="date">Дата: </label><br>
        <input type="date" id="date" name="date" value="{{ date }}" required><br>
        <label for="customer">Замовник: </label><br>
        <input type="text" id="customer" name="customer" value="{{ customer
}}"><br>
        <label for="repair_cost">Вартість ремонту: </label><br>
        <input type="number" id="repair_cost" name="repair_cost" value="{{
repair_cost }}" min="0" step="0.01" required><br>
        <label for="discount">Знижка: </label><br>
        <input type="number" id="discount" name="discount" value="{{ discount }}"
min="0" step="0.01" required><br>
        <label for="status">Статус: </label><br>
        <select name="status" id="status" required>
            <option value="">Оберіть значення</option>
{{statuses|safe}}
        </select><br>
        <label for="details">Деталі: </label><br>
        <textarea id="details" name="details">{{ details }}</textarea><br>
        <button id="submit" type="submit">Записати</button>
    </form>
</body>
</html>
```

Файл PL/templates/edit_product.html — шаблон сторінки редагування товару

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="/static/style.css">
    <title>Товар - Випускна кваліфікаційна робота</title>
</head>
<body>
    {% if error %}
    <p>{{ error }}</p>
    {% endif %}
    <form method="POST">
        <label for="vendor">Виробник: </label><br>
        <input type="text" id="vendor" name="vendor" value="{{ vendor }}"
required><br>
        <label for="name">Назва: </label><br>
        <input type="text" id="name" name="name" value="{{ name }}" required><br>
        <label for="price">Ціна: </label><br>
        <input type="number" id="price" name="price" value="{{ price }}" min="0"
step="0.01" required><br>
        <label for="amount">Кількість: </label><br>
        <input type="number" id="amount" name="amount" value="{{ amount }}"
min="0" step="1" required><br>
        <label for="description">Опис: </label><br>
        <textarea id="description" name="description">{{ description
}}</textarea><br>
        <button id="submit" type="submit">Записати</button>
    </form>
</body>
</html>
```

Файл PL/templates/index.html — головна сторінка

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="utf-8">
    <title>Головна сторінка - Випускна кваліфікаційна робота</title>
</head>
<body>
    <p><a href="logout">Вихід</a></p><br>

    <p><a href="products">Товари</a></p>
    <p><a href="orders">Замовлення</a></p><br>

    <p><a href="stats">Статистика</a></p>

    <p><a href="password">Зміна пароля</a></p>

    <p><a href="copyright">Авторське право</a></p>
</body>
</html>
```

Файл PL/templates/login.html — шаблон сторінки входу

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="utf-8">
    <title>Вхід - Випускна кваліфікаційна робота</title>
</head>
<body>
    {% if error %}
    <p>{{ error }}</p>
    {% endif %}
    <form method="POST">
        <label for="username">Ім’я: </label><br>
        <input type="text" id="username" name="username" required><br>
        <label for="password">Пароль: </label><br>
        <input type="password" id="password" name="password" required><br>
        <button id="submit" type="submit">Увійти</button>
    </form>

    <p><a href="copyright">Авторське право</a></p>
</body>
</html>
```

Файл PL/templates/ordered_products.html — шаблон сторінки списку товарів у замовленні

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="/static/style.css">
    <title>Товари у замовленні - Випускна кваліфікаційна робота</title>
</head>
<body>
    {{ordered_products|safe}}
    {% if error %}
    <p>{{ error }}</p>
    {% endif %}
    <form method="POST">
        <label for="product">Номер товару: </label><br>
        <input type="number" id="product" name="product" min="1" step="1"
required><br>
        <label for="amount">Кількість: </label><br>
        <input type="number" id="amount" name="amount" min="0" step="1"
required><br>
        <input type="checkbox" id="change_amount" name="change_amount">
        <label for="change_amount">Змінювати кількість</label><br>
        <button id="submit" type="submit">Записати</button>
    </form>
</body>
</html>
```

Файл PL/templates/orders.html — шаблон сторінки списку замовлень

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="/static/style.css">
    <title>Замовлення - Випускна кваліфікаційна робота</title>
</head>
<body>
    {{order_list|safe}}
    <a href="orders/create">Додати</a>
</body>
</html>
```

Файл PL/templates/password.html — шаблон сторінки зміни пароля

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="utf-8">
    <title>Зміна пароля - Випускна кваліфікаційна робота</title>
</head>
<body>
    {% if error %}
    <p>{{ error }}</p>
    {% endif %}
    <form method="POST">
        <label for="current">Поточний пароль: </label><br>
        <input type="password" id="current" name="current" required><br>
        <label for="new">Новий пароль: </label><br>
        <input type="password" id="new" name="new" required><br>
        <label for="confirmation">Підтвердження нового пароля: </label><br>
        <input type="password" id="confirmation" name="confirmation" required><br>
        <button id="submit" type="submit">Змінити</button>
    </form>
</body>
</html>
```

Файл PL/templates/products.html — шаблон сторінки списку товарів

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="/static/style.css">
    <title>Товари - Випускна кваліфікаційна робота</title>
</head>
<body>
    {{product_list|safe}}
    <a href="products/create">Додати</a>
</body>
</html>
```

Файл PL/templates/stats.html — шаблон сторінки статистики

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="/static/style.css">
    <title>Статистика - Випускна кваліфікаційна робота</title>
</head>
<body>
    {{period}}
    <form method="POST">
        <label for="date_from">Від: </label><br>
        <input type="date" id="date_from" name="date_from"><br>
        <label for="date_to">До: </label><br>
        <input type="date" id="date_to" name="date_to"><br>
        <button id="submit" type="submit">Підтвердити</button>
    </form>
    <p>10 товарів із найбільшим попитом:</p>
{{high_demand|safe}}
    <p>10 товарів із найменшим попитом:</p>
{{low_demand|safe}}
    <p>10 найкращих замовників:</p>
{{best_customers|safe}}
    <p>Перелік товарів, яких залишилося менше 50:</p>
{{low_amount|safe}}
    <p>Необроблені замовлення:</p>
{{new_orders|safe}}
</body>
</html>
```

Шар бізнес-логіки

Файл BLL/auth.py — автентифікація

```
"""У даному файлі міститься функція, яка перевіряє, чи авторизований
адміністратор"""

import bcrypt

from DAL.auth import user_id_check, username_check
from DAL.auth import change_password as dal_change_password


def auth(username, password):
    """Перевіряє на дійсність дані, вказані під час авторизації
    Повертає номер облікового запису у випадку успішної авторизації, інакше
повертає False
    """
    row = username_check(username)
    if row:
        if bcrypt.checkpw(password.encode(), row[1]):
            return row[0]
    return False


def change_password(user_id, form):
    """Змінює пароль
    Повертає True, якщо пароль змінено та False у разі помилки"""

    if not form.get("new"):
        return False

    if form.get("new") != form.get("confirmation"):
        return False

    row = user_id_check(user_id)
    if row:
        if not bcrypt.checkpw(form.get("current").encode(), row[1]):
            return False

    dal_change_password(user_id, bcrypt.hashpw(form["new"].encode(),
bcrypt.gensalt()))
    return True
```

```
def check_account(user_id):
    """Перевіряє, чи авторизований адміністратор
    Повертає True, якщо авторизований, та False, якщо не авторизований"""
    if user_id_check(user_id):
        return True
    return False
```

Файл BLL/dates.py — операції з датами

```
"""Операції з датами"""

from datetime import date as dt_date


def validate_date(date):
    """Перевіряє на правильність одну дату
    Повертає True якщо правильна, та False якщо неправильна"""
    try:
        dt_date.fromisoformat(date)
        return True
    except ValueError:
        return False


def validate_dates(date_from, date_to):
    """Перевіряє на правильність проміжок із двох дат
    Повертає True якщо правильний, та False якщо неправильний"""
    try:
        date_from = dt_date.fromisoformat(date_from)
        date_to = dt_date.fromisoformat(date_to)
    except ValueError:
        return False

    if date_from <= date_to:
        return True
    return False
```

Файл BLL/init.py — передача запиту на ініціалізацію бази даних

```
"""Ініціалізація бази даних"""

import bcrypt

from DAL.init import init as dal_init


def init():
    """Передає у DAL запит на створення бази даних"""
    dal_init()
```

Файл BLL/numbers.py — операції над числами

```
"""Операції над числами"""


def is_numeric(string: str) -> bool:
    """Перевіряє, чи є рядок числом
    Якщо є, повертає True, інакше False"""
    try:
        float(string)
        return True
    except ValueError:
        return False
```

Файл BLL/orders.py — операції над замовленнями

```
"""Операції над списком замовлень"""

from BLL.dates import validate_date
from BLL.numbers import is_numeric

from DAL.orders import create_order as dal_create_order
from DAL.orders import read_order as dal_read_order
from DAL.orders import read_orders as dal_read_orders
from DAL.orders import read_new_orders as dal_read_new_orders
from DAL.orders import update_order as dal_update_order
from DAL.orders import delete_order as dal_delete_order

from DAL.orders import create_ordered_product as dal_create_ordered_product
from DAL.orders import read_ordered_product as dal_read_ordered_product
from DAL.orders import read_ordered_products as dal_read_ordered_products
from DAL.orders import update_ordered_product as dal_update_ordered_product
from DAL.orders import delete_ordered_product as dal_delete_ordered_product

from DAL.orders import read_best_customers as dal_read_best_customers

from DAL.products import read_product as dal_read_product
from DAL.products import decrease_amount as dal_decrease_amount


statuses = (
    "Прийнято",
    "Очікується оплата",
    "Очікується доставка",
    "Виконано",
    "Повернено",
    "Скасовано",
)


def validate_order(form):
    """Перевіряє дані замовлення на коректність"""
    date = form.get("date")
    repair_cost = form.get("repair_cost")
    discount = form.get("discount")
    status = form.get("status")

    if (
```

```
        (date is None or repair_cost is None or discount is None or status is
None)
        or not (is_numeric(repair_cost) or float(repair_cost) < 0)
        or not (is_numeric(discount) or float(discount) < 0)
        or not status.isdigit()
        or not validate_date(date)
    ):
        return False

    status = int(status)

    if status < 0 or status >= len(statuses):
        return False

    return True


def create_order(form):
    """Додає нове замовлення"""
    if not validate_order(form):
        return "Некоректні дані"
    dal_create_order(
        form.get("date"),
        form.get("customer"),
        form.get("repair_cost"),
        form.get("discount"),
        form.get("status"),
        form.get("details"),
    )
    return None


def read_order(order_id):
    """Отримує одне замовлення"""
    return dal_read_order(order_id)


def read_orders(only_new=False):
    """Отримує список усіх або лише нових замовлень"""
    table = "<table>\n"
    table += """         <tr>
            <th>Номер</th>
            <th>Дата</th>
```

```
            <th>Замовник</th>
            <th>Вартість ремонту</th>
            <th>Вартість товарів</th>
            <th>Знижка</th>
            <th>До сплати</th>
            <th>Статус</th>
            <th>Деталі</th>
            <th>Товари</th>
            <th>Зміна</th>
            <th>Видалення</th>
        </tr>\n"""
    orders = dal_read_new_orders() if only_new else dal_read_orders()

    for order in orders:
        product_cost = read_products_cost(order[0])
        table += f"""        <tr>
            <td>{order[0]}</td>
            <td>{order[1]}</td>
            <td>{order[2]}</td>
            <td>{order[3]}</td>
            <td>{product_cost}</td>
            <td>{order[4]}</td>
            <td>{order[3]+product_cost-order[4]}
            <td>{statuses[order[5]]}</td>
            <td>{order[6]}</td>
            <td><a href="orders/products/{order[0]}">Товари</a></td>
            <td><a href="orders/update/{order[0]}">Змінити</a></td>
            <td><a href="orders/delete/{order[0]}">Видалити</a></td>
        </tr>\n"""

    table += "    </table>"
    return table


def update_order(order_id, form):
    """Оновлює замовлення"""
    if not validate_order(form):
        return "Некоректні дані"
    dal_update_order(
        order_id,
        form.get("date"),
        form.get("customer"),
        form.get("repair_cost"),
```

```
            form.get("discount"),
            form.get("status"),
            form.get("details"),
        )
        return None


def delete_order(order_id):
    """Видаляє замовлення"""
    return dal_delete_order(order_id)


def read_statuses(selected=None):
    """Повертає список статусів у HTML-вигляді"""
    response = ""
    for idx, status in enumerate(statuses):
        response += f'        <option value="{idx}"'
        if idx == selected:
            response += " selected"
        response += f">{status}</option>\n"
    return response


def validate_ordered_product(form):
    """Перевіряє дані товару у замовленні на коректність"""
    product = form.get("product")
    amount = form.get("amount")

    if not (is_numeric(product) or is_numeric(amount)):
        return False

    product = int(product)
    amount = int(amount)
    product_data = dal_read_product(product)

    if (product < 0 or amount < 0) or not product_data:
        return False

    if form.get("change_amount") and product_data[3] < amount:
        return False

    return True
```

```
def read_products_cost(order):
    """Отримує вартість товарів замовлення"""
    total = 0.0
    entries = dal_read_ordered_products(order)
    for entry in entries:
        product = dal_read_product(entry[0])
        total += product[2] * entry[1]
    return total


def read_ordered_products(order):
    """Отримує список товарів у замовленні"""
    total = 0.0
    table = "<table>\n"
    table += """        <tr>
            <th>Номер товару</th>
            <th>Виробник</th>
            <th>Назва</th>
            <th>Ціна</th>
            <th>Кількість</th>
            <th>Вартість</th>
        </tr>\n"""
    entries = dal_read_ordered_products(order)
    for entry in entries:
        product = dal_read_product(entry[0])
        cost = product[2] * entry[1]
        table += f"""        <tr>
            <td>{entry[0]}</td>
            <td>{product[0]}</td>
            <td>{product[1]}</td>
            <td>{product[2]}</td>
            <td>{entry[1]}</td>
            <td>{cost}</td>
        </tr>\n"""
        total += cost

    table += f"""        <tr>
            <td colspan="5">Всього за товари:</td>
            <td>{total}</td>
        </tr>\n"""

    table += "    </table>"
```

```
    return table


def update_ordered_product(order, form):
    """Змінює кількість товару у замовленні
    Повертає помилку, якщо вона є"""
    if not validate_ordered_product(form):
        return "Некоректні дані"

    product = int(form.get("product"))
    amount = int(form.get("amount"))

    ordered_product = dal_read_ordered_product(order, product)

    if form.get("change_amount"):
        dal_decrease_amount(product, (amount - ordered_product[1]) if
ordered_product else amount)

    if amount == 0:
        if ordered_product:
            dal_delete_ordered_product(order, product)
    else:
        if not dal_read_ordered_product(order, product):
            dal_create_ordered_product(order, product, amount)
        else:
            dal_update_ordered_product(order, product, amount)
    return None


def read_best_customers(date_from=None, date_to=None):
    """Отримує список найкращих замовників"""
    table = "<table>\n"
    table += """        <tr>
            <th>Замовник</th>
            <th>Сума</th>
        </tr>\n"""
    customers = dal_read_best_customers(date_from=date_from, date_to=date_to)

    for customer in customers:
        table += f"""       <tr>
            <td>{customer[0]}</td>
            <td>{customer[1]}</td>
        </tr>\n"""
```

```
table += "    </table>"
return table
```

Файл BLL/products.py — операції над товарами

```
"""Операції над списком товарів"""

from BLL.numbers import is_numeric

from DAL.products import create_product as dal_create_product
from DAL.products import read_product as dal_read_product
from DAL.products import read_products as dal_read_products
from DAL.products import read_products_amount as dal_read_products_amount
from DAL.products import read_products_demand as dal_read_products_demand
from DAL.products import update_product as dal_update_product
from DAL.products import delete_product as dal_delete_product


def validate_product(form):
    """Перевіряє дані товару на коректність
    Повертає True, якщо коректні, та False, якщо некоректні"""
    vendor = form.get("vendor")
    name = form.get("name")
    price = form.get("price")
    amount = form.get("amount")

    if (vendor is None or name is None or price is None or amount is None) or (
        not is_numeric(price)
        or not amount.isdigit()
        or float(price) < 0
        or int(amount) < 0
    ):
        return False
    return True


def create_product(form):
    """Додає новий товар
    Повертає помилку, якщо вона є"""
    if not validate_product(form):
        return "Некоректні дані"
    dal_create_product(
        form.get("vendor"),
        form.get("name"),
        form.get("price"),
        form.get("amount"),
        form.get("description"),
```

```
    )
    return None


def read_product(product_id):
    """Отримує один товар"""
    return dal_read_product(product_id)


def read_products(max_amount=0, demand_sort=0, date_from=None, date_to=None):
    """Отримує список усіх товарів, або товарів, яких залишилося менше певної кількості,
    або товарів за попитом за певний проміжок часу або за увесь час роботи програми"""
    if max_amount and demand_sort:
        raise ValueError("Одночасно можливо використовувати лише один вид налаштувань.")

    table = "<table>\n"
    table += """        <tr>
            <th>Номер</th>
            <th>Виробник</th>
            <th>Назва</th>
            <th>Ціна</th>
            <th>Кількість</th>
            <th>Опис</th>
            """
    if demand_sort:
        table += """<th>Замовлено одиниць</th>
        """

    table += """<th>Зміна</th>
            <th>Видалення</th>
        </tr>\n"""
    if max_amount:
        products = dal_read_products_amount(max_amount)
    elif demand_sort == 1:
        products = dal_read_products_demand(date_from=date_from, date_to=date_to)
    elif demand_sort == 2:
        products = dal_read_products_demand(
            reverse=True, date_from=date_from, date_to=date_to
        )
    else:
```

```
        products = dal_read_products()
    for product in products:
        table += f"""       <tr>
            <td>{product[0]}</td>
            <td>{product[1]}</td>
            <td>{product[2]}</td>
            <td>{product[3]}</td>
            <td>{product[4]}</td>
            <td>{product[5]}</td>
            """
        if demand_sort:
            table += f"""<td>{product[6]}</td>
            """
        table += f"""<td><a href="products/update/{product[0]}">Змінити</a></td>
            <td><a href="products/delete/{product[0]}">Видалити</a></td>
        </tr>\n"""

    table += "    </table>"
    return table


def update_product(product_id, form):
    """Додає новий товар
    Повертає помилку, якщо вона є"""
    if not validate_product(form):
        return "Некоректні дані"
    dal_update_product(
        product_id,
        form.get("vendor"),
        form.get("name"),
        form.get("price"),
        form.get("amount"),
        form.get("description"),
    )
    return None


def delete_product(product_id):
    """Видаляє товар"""
    return dal_delete_product(product_id)
```

Шар доступу до даних

Файл DAL/auth.py — автентифікація

```
"""Автентифікація"""

from DAL.connection import connection


def user_id_check(user_id):
    """Перевіряє, чи існує обліковий запис із вказаним номером"""
    cursor = connection.cursor()
    cursor.execute("SELECT id, password FROM admins WHERE id = ?", (user_id,))
    return cursor.fetchone()


def change_password(user_id, password_hash):
    """Змінює пароль"""
    connection.execute(
        "UPDATE admins SET password = ? WHERE `id` = ?",
        (password_hash, user_id),
    )
    connection.commit()


def username_check(username):
    """Перевіряє, чи існує обліковий запис із вказаним іменем"""
    cursor = connection.cursor()
    cursor.execute("SELECT id, password FROM admins WHERE username = ?",
(username,))
    return cursor.fetchone()
```

Файл DAL/connection.py — з’єднання з базою даних

```
"""Встановлення з'єднання з базою даних"""

import sqlite3

connection = sqlite3.connect("database.db", check_same_thread=False)
```

Файл DAL/init.py — ініціалізація бази даних

```
"""Ініціалізація бази даних"""

from DAL.connection import connection


def init_tables():
    """Створює таблиці у базі даних"""
    connection.execute(
        "CREATE TABLE IF NOT EXISTS admins (`id` integer PRIMARY KEY, `username` text UNIQUE NOT NULL, `password` text NOT NULL)"
    )
    connection.execute(
        "CREATE TABLE IF NOT EXISTS products (`id` integer PRIMARY KEY, `vendor` text NOT NULL, `name` text NOT NULL, `price` real NOT NULL, `amount` integer NOT NULL, `description` text)"
    )
    connection.execute(
        "CREATE TABLE IF NOT EXISTS orders (`id` integer PRIMARY KEY, `date` date NOT NULL DEFAULT CURRENT_TIMESTAMP, `customer` text, `repair_cost` real NOT NULL DEFAULT 0, `discount` real NOT NULL DEFAULT 0, `status` integer NOT NULL DEFAULT 0, `details` text)"
    )
    connection.execute(
        "CREATE TABLE IF NOT EXISTS ordered_products (`id` integer PRIMARY KEY, `order` integer REFERENCES orders(id) ON DELETE CASCADE, `product` integer REFERENCES products(id) ON DELETE CASCADE, `amount` integer NOT NULL)"
    )
    connection.commit()


def init_admin():
    """Створює обліковий запис адміністратора
    за замовчуванням"""
    password = (
        b"$2b$12$5klXD73Uakl3VyQJIXGfRe5YK.QDdU2JRbI1NtnDdmDXI8fK22CvO"  # "admin"
    )
    connection.execute(
        'INSERT OR IGNORE INTO admins(username, password) VALUES("admin", ?)',
        (password,),
    )
    connection.commit()
```

```
def init():
    """Створює базу даних, якщо вона не існує, вмикає foreign keys"""
    init_tables()
    init_admin()
    connection.execute("PRAGMA foreign_keys = ON")
```

Файл DAL/orders.py — операції над замовленнями

```
"""Операції над списком замовлень"""

from DAL.connection import connection


def create_order(date, customer, repair_cost, discount, status, details):
    """Створює замовлення"""
    connection.execute(
        """INSERT INTO orders(date, customer, repair_cost, discount, status,
details)
        VALUES(?, ?, ?, ?, ?, ?)""",
        (date, customer, repair_cost, discount, status, details),
    )
    connection.commit()


def read_order(order_id):
    """Отримує одне замовлення"""
    cursor = connection.cursor()
    cursor.execute(
        "SELECT date, customer, repair_cost, discount, status, details FROM orders
WHERE id = ?",
        (order_id,),
    )
    return cursor.fetchone()


def read_orders():
    """Отримує список усіх замовлень"""
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM orders")
    return cursor.fetchall()


def read_new_orders():
    """Отримує список нових замовлень"""
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM orders WHERE status = 0")
    return cursor.fetchall()


def read_best_customers(date_from=None, date_to=None):
```

```
    """Отримує список найкращих замовників"""
    cursor = connection.cursor()
    cursor.execute(
        f"""SELECT o.customer,
        SUM(COALESCE((p.price * op.amount), 0) + o.repair_cost - o.discount) AS
total_order_cost
        FROM orders AS o
        LEFT JOIN ordered_products AS op ON o.id = op.`order`
        LEFT JOIN products AS p ON op.product = p.id
        WHERE o.customer IS NOT NULL and o.customer != ''
        {"AND o.date BETWEEN ? AND ?" if date_from and date_to else ""}
        GROUP BY o.customer
        ORDER BY total_order_cost DESC
        LIMIT 10;""",
        (date_from, date_to) if date_from and date_to else (),
    )
    return cursor.fetchall()


def update_order(order_id, date, customer, repair_cost, discount, status,
details):
    """Змінює дані замовлення"""
    connection.execute(
        """UPDATE orders SET date = ?, customer = ?, repair_cost = ?,
        discount = ?, status = ?, details = ? WHERE id = ?""",
        (date, customer, repair_cost, discount, status, details, order_id),
    )
    connection.commit()


def delete_order(order_id):
    """Видаляє замовлення"""
    connection.execute(
        "DELETE FROM orders WHERE id = ?",
        (order_id,),
    )
    connection.commit()


def create_ordered_product(order, product, amount):
    """Створює товар у замовленні"""
    connection.execute(
```

```
        "INSERT INTO ordered_products(`order`, `product`, `amount`) VALUES(?, ?,
?)",
        (order, product, amount),
    )
    connection.commit()


def read_ordered_product(order, product):
    """Отримує один товар у замовленні"""
    cursor = connection.cursor()
    cursor.execute(
        "SELECT product, amount FROM ordered_products WHERE `order` = ? AND
`product` = ?",
        (order, product),
    )
    return cursor.fetchone()


def read_ordered_products(order):
    """Отримує список товарів у замовленні""
    cursor = connection.cursor()
    cursor.execute(
        "SELECT product, amount FROM ordered_products WHERE `order` = ?",
        (order,),
    )
    return cursor.fetchall()


def update_ordered_product(order, product, amount):
    """Оновлює дані про товар у замовленні"""
    connection.execute(
        "UPDATE ordered_products SET amount = ? WHERE `order` = ? AND `product` =
?",
        (amount, order, product),
    )
    connection.commit()


def delete_ordered_product(order, product):
    """Видаляє дані про товар у замовленні"""
    connection.execute(
        "DELETE FROM ordered_products WHERE `order` = ? AND `product` = ?",
        (order, product),
```

```
)
connection.commit()
```

Файл DAL/products.py — операції над товарами

```
"""Операції над списком товарів"""

from DAL.connection import connection


def create_product(vendor, name, price, amount, description):
    """Створює товар"""
    connection.execute(
        "INSERT INTO products(vendor, name, price, amount, description) VALUES(?,
?, ?, ?, ?)",
        (vendor, name, price, amount, description),
    )
    connection.commit()


def read_product(product_id):
    """Отримує один товар"""
    cursor = connection.cursor()
    cursor.execute(
        "SELECT vendor, name, price, amount, description FROM products WHERE id =
?",
        (product_id,),
    )
    return cursor.fetchone()


def read_products():
    """Отримує список усіх товарів"""
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM products")
    return cursor.fetchall()


def read_products_amount(amount):
    """Отримує список товарів, яких залишилося менше певної кількості"""
    cursor = connection.cursor()
    cursor.execute("SELECT * FROM products WHERE amount < ?", (amount,))
    return cursor.fetchall()


def read_products_demand(reverse=False, date_from=None, date_to=None):
    """Отримує дані щодо попиту на товари"""
```

```
    cursor = connection.cursor()
    cursor.execute(
        f"""SELECT p.*, COALESCE(SUM(op.amount), 0) AS total_amount
        FROM products AS p
        LEFT JOIN ordered_products AS op ON p.id = op.product
        LEFT JOIN orders AS o ON op.`order` = o.id
        {"WHERE o.date BETWEEN ? AND ?" if date_from and date_to else ""}
        GROUP BY p.id
        ORDER BY total_amount {"ASC" if reverse else "DESC"}
        LIMIT 10;""",
        (date_from, date_to) if date_from and date_to else (),
    )
    return cursor.fetchall()


def update_product(product_id, vendor, name, price, amount, description):
    """Змінює дані товару"""
    connection.execute(
        """UPDATE products SET vendor = ?, name = ?, price = ?,
        amount = ?, description = ? WHERE id = ?""",
        (vendor, name, price, amount, description, product_id),
    )
    connection.commit()


def decrease_amount(product_id, amount):
    """Зменшує кількість товару"""
    connection.execute(
        "UPDATE products SET amount = amount - ? WHERE id = ?",
        (amount, product_id),
    )
    connection.commit()


def delete_product(product_id):
    """Видаляє товар"""
    connection.execute(
        "DELETE FROM products WHERE id = ?",
        (product_id,),
    )
    connection.commit()
```