Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

Навчально-науковий інститут деревообробних та
комп'ютерних технологій і дизайну
(повне найменування інституту, назва факультету (відділення))

Кафедра інформаційних систем та комп'ютерного моделювання
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до дипломної роботи
ОКР – бакалавр
(освітньо-кваліфікаційний рівень)

на тему: *Розроблення клієнт-серверної інформаційної системи складського обліку товарів (FrontEnd).*

Виконав: студент 4 курсу групи ІСТ-41
спеціальності
126 " Інформаційні системи та технології "
(шифр і назва напряму підготовки, спеціальності)

Вітка О.В.
(прізвище та ініціали)

Керівник Сторожук О.Л.
(прізвище та ініціали)

Рецензент Крошний І.М.
(прізвище та ініціали)

Львів – 2023

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ деревообробних та комп'ютерних технологій і дизайну
Кафедра інформаційних систем та комп'ютерного моделювання
Рівень вищої освіти перший (бакалаврський)
Спеціальність 126 " Інформаційні системи та технології"
(шифр і назва)

ЗАТВЕРДЖУЮ
В.о.завідувача кафедри
_____ Сторожук О.Л.
" 12 " 06 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Вітка Олегові Володимировичеві
(прізвище, ім'я, по батькові)

1. Тема роботи *«Розроблення клієнт-серверної інформаційної системи складського обліку товарів (FrontEnd)»*
керівник роботи *Сторожук Олександр Леонідович, к.т.н., доцент*,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від *"21"11 2022 року № С-521*
2. Термін подання студентом роботи *12.06.2023р.*
3. Вихідні дані до роботи: Розробити програмний застосунок клієнт-серверної інформаційної системи складського обліку товарів. Застосунок має мати зручний та комфортний інтерфейс для користувача цієї програми. Для реалізації роботи використати мову програмування "Python", а також бібліотеки PyQt5, та інші.
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити): *1) Стан проблемної області; 2) Інформаційне та математичне забезпечення; 3) Програмне та технічне забезпечення.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація до диплому
6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання *23 листопада 2022 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1.	*Огляд літературних даних.*	*23.11.2022р. – 21.12.2022р.*	виконано
2.	*Розділ 1. Стан проблемної області.*	*21.12.2022р. – 05.01.2023р.*	виконано
3.	*Розділ 2. Інформаційне та математичне забезпечення.*	*05.01.2023р. – 02.02.2023р.*	виконано
4.	*Розділ 3. Програмне та технічне забезпечення.*	*02.02.2023р. – 12.05.2023р.*	виконано
9.	*Аналіз отриманих результатів та написання висновків. Оформлення дипломної роботи.*	*12.05.2023р. – 06.06.2023р.*	виконано
10.	*Здача пояснювальної записки на перевірку керівнику, виправлення помилок та здача роботи рецензенту.*	*12.06.2023 р.*	виконано

Студент (підпис) *Вітка О.В.* (прізвище та ініціали)

Керівник роботи (підпис) *Сторожук О.Л.* (прізвище та ініціали)

РЕФЕРАТ

Дипломна робота містить *30* сторінок пояснювальної записки, *37* рисунків, *15* джерел та 1 додаток.

Дипломна робота присвячена розробці клієнт-серверної інформаційної системи складського обліку товарів (FrontEnd). Для виконання поставленої задачі було використано бібліотеку PyQt5, на базі мови програмування Python. Дана бібліотека дозволяє виводити на екран і робити графічне представлення коду на цій мові програмування. Також було використано ряд бібліотек, таких як: sys, csv, os, sqlite3, PIL, tkinter, re. Серед функціоналу передбачено додавання, видалення, а також редагування товарів у системі. Графічний інтерфейс було виконано враховуючи вимоги користувачів, для його зручності.

Ключові слова: Python, PyQt5, sys, csv, os, sqlite3, PIL, tkinter, re, бібліотеки, інтерфейс, програмний застосунок.

ABSTRACT

Thesis contains *30* pages of explanatory note, *37* drawings, *15* sources and *1* appendix.

The thesis is devoted to the development of a client-server information system for warehouse accounting of goods (FrontEnd). To accomplish this task, we mainly used the PyQt5 library, based on the Python programming language. This library allows you to display and graphically represent the code in this programming language. A number of libraries were also used, such as: sys, csv, os, sqlite3, PIL, tkinter, re. The functionality includes adding, deleting, and editing products in the system. The graphical interface was made to meet all the requirements of the user, for his convenience.

Keywords: Python, PyQt5, sys, csv, os, sqlite3, PIL, tkinter, re, libraries, interface, software application.

ТЕХНІЧНЕ ЗАВДАННЯ

Розробити клієнт-серверну інформаційну систему складського обліку товарів (FrontEnd). Ознайомитись із специфікою даної сфери для розробки якісної програми. При розробці використати мову програмування Python і бібліотеку PyQt5. Передбачити додавання, видалення, а також редагування товарів через бібліотеку sqlite3. Інтерфейс створити враховуючи результати аналізу вимог працівників цієї сфери.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface, інтерфейс програмування застосунків;

DB – Database, база даних;

GUI – Graphical User Interface, графічний інтерфейс користувача;

IoT – Internet of Things, інтернет речей;

SQL – Structured Query Language, мова структурованих запитів до баз даних;

ACID – Atomicity, Consistency, Isolation, Durability, властивості транзакцій баз даних;

ООП – об'єктно-орієнтоване програмування;

IDE – Integrated Development Environment, інтегроване середовище розробки.

ВСТУП

Відвідуючи магазини у яких присутній склад деякі покупці задаються питанням, як організована робота у складських приміщеннях і яким чином ведеться складський облік? Розпитуючи працівників, а також приймаючи у цьому участь, було виявлено, що складський облік, або взагалі не ведеться, або ведеться за допомогою “важких” і незрозумілих програм. Розробка власної програми складського обліку дозволить не тільки керівним посадам зручно контролювати, та відслідковувати товари у цій зоні, а і дасть працівникам можливість зручно комунікувати із складом.

Складське приміщення – це сердце магазину, без нього не можливий ні один магазин і ні один бізнес. Якщо все буде структуровано, буде легша робота на складі, а також буде більш ефективніше розподілений час для комірників.

Об’єктом дослідження є реалізація інформаційної системи обліку товарів на серверній частині sqlite3.

Метою роботи є розробка зручної програми для обліку товарів.

Предметом дослідження є реалізація можливості роботи з обліком товарів.

Практичне значення – розроблена інформаційна система полегшує роботу працівникам складу та дозволяє вести складський облік товарів.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Огляд проблемної області.

Складський облік є важливою складовою ефективного управління логістичними процесами в компаніях, що займаються зберіганням і розподілом товарів. Однак, в цій сфері існують деякі проблеми, які можуть впливати на ефективність та точність обліку. Нижче описані деякі з них [7]:

1. Недостатня автоматизація: Багато складів все ще оперують застарілими методами ручного обліку, що викликає велику кількість помилок. Відсутність автоматизованих систем складського обліку може призводити до неточностей у веденні журналів, інвентаризаційних актів та інших документів.

2. Неправильне ідентифікування та маркування товарів: Недостатня якість або відсутність маркування на товарах може призвести до помилок у процесі отримання, зберігання та відвантаження товарів. Це може впливати на точність обліку запасів і спричиняти проблеми з виявленням втрат або крадіжок.

3. Незручні системи планування: Відсутність ефективних систем планування та прогнозування попиту може призвести до зайвих запасів або недостачі товарів на складі. Це може впливати на затрати компанії і покупців, а також на загальну продуктивність складських процесів.

Враховуючи ці проблеми, важливо вдосконалювати системи складського обліку шляхом впровадження сучасних технологій автоматизації, використання надійних систем ідентифікації та маркування, ефективного планування та прогнозування попиту, а також забезпечення безпеки та контролю на складах.

1.2. Актуальність розроблення застосунку.

Розробка програми для складського обліку товарів має велику актуальність у сучасному бізнес-середовищі. Ось кілька причин, чому це важливо [8]:

1. Ефективне управління запасами: Програма для складського обліку товарів дозволяє точно відстежувати рух товарів на складі, від отримання до відвантаження. Це допомагає уникнути надлишків або недостачі товарів, оптимізувати запаси і забезпечити, що товари завжди доступні в потрібний час.
2. Точність інвентаризації: Ручний облік запасів може призводити до помилок і неточностей, особливо в разі великих обсягів товарів. Розробка програми для складського обліку дозволяє автоматизувати процес підрахунку запасів і забезпечити більш точні результати інвентаризації.
3. Забезпечення ефективного управління простором складу: Програма для складського обліку дозволяє планувати оптимальне розміщення товарів на складі, враховуючи їх характеристики та частоту попиту. Це допомагає ефективно використовувати простір складу та скорочувати час на пошук та видачу товарів.
4. Вдосконалення процесів замовлення та поставок: Розробка програми для складського обліку дозволяє автоматизувати процеси замовлення товарів, моніторингу поставок та відстеження їх стану. Це допомагає покращити комунікацію з постачальниками, скоротити час на замовлення та отримання товарів і запобігти запізненням або невідповідностям.
5. Забезпечення точності та надійності даних: Програма для складського обліку дозволяє зберігати всю інформацію про товари та їх рух в одній централізованій базі даних. Це дозволяє уникнути втрати або помилкового введення даних, забезпечує легкий доступ до інформації та поліпшує аналітичні можливості для прийняття управлінських рішень.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1. Python

Python - це високорівнева, інтерпретована, загального призначення мова програмування. Ось деякі характеристики та особливості мови програмування Python [9]:

1. **Синтаксис і зрозумілість:** Синтаксис Python є простим і зрозумілим. Він наближений до природної мови, що робить код більш читабельним і зрозумілим для розробників. Це сприяє швидкому розвитку програм і зменшує кількість помилок.
2. **Широкий спектр застосувань:** Python має велику базу бібліотек і фреймворків, що робить його універсальним і придатним для розв'язання різноманітних задач. Він використовується веб-розробкою, наукових обчисленнях, штучному інтелекті, аналізі даних, автоматизації завдань і багатьох інших галузях.
3. **Крос-платформенність:** Python є крос-платформною мовою, що означає, що програми, написані на Python, можуть запускатися на різних операційних системах, таких як Windows, macOS, Linux і багатьох інших. Це робить його універсальним і зручним для розробки програм, які мають бути доступні на різних платформах.
4. **Об'єктно-орієнтоване програмування:** Python підтримує об'єктно-орієнтоване програмування (ООП), що дозволяє розробникам створювати класи, об'єкти, успадкування та інші ООП-концепції. Це сприяє модульності, повторному використанню коду і полегшує підтримку та розвиток програм [2,5].

2.2. PyQt5

PyQt5 є потужною бібліотекою для розробки графічного інтерфейсу користувача (GUI) в мові програмування Python. Вона базується на функціях та класах бібліотеки Qt, що робить її варіантом для створення крос-платформених додатків зі зручним та професійним виглядом.

Ось деякі особливості та переваги PyQt5 [10]:

1. **Повна функціональність Qt:** PyQt5 надає доступ до повного функціоналу Qt, що включає широкий набір віджетів, класів та інструментів для побудови різноманітних GUI-додатків. Ви можете створювати вікна, кнопки, поля введення, таблиці, меню, діалогові вікна та багато іншого.
2. **Крос-платформеність:** PyQt5 дозволяє розробляти крос-платформені додатки, які працюють на різних операційних системах, таких як Windows, macOS, Linux і багатьох інших. Це дозволяє вам створювати один код, який працюватиме на різних платформах без необхідності внесення змін.
3. **Дизайнер Qt:** PyQt5 постачається зі вбудованим інструментом дизайну Qt DEsignEr, який дозволяє вам створювати GUI за допомогою візуального інтерфейсу. Ви можете перетягувати та розміщувати віджети, налаштовувати їх властивості та підключати сигнали і слоти, щоб створити потрібний інтерфейс.
4. **Велика спільнота та підтримка:** PyQt5 має велику спільноту розробників, яка надає допомогу, документацію, приклади та відповіді на питання. Це значно спрощує вивчення та використання бібліотеки.
5. **Інтеграція зі стандартними інструментами Python:** PyQt5 легко інтегрується з іншими бібліотеками та інструментами Python, такими як NumPy, pandas, Matplotlib і багато інших. Ви можете використовувати всю потужність Python для обробки даних, наукових обчислень та

інших завдань, а PyQt5 дозволить вам візуалізувати результати у зручному GUI [1,6].

2.3. Sqlite3

Бібліотека sqlitE3 є вбудованою бібліотекою мови програмування Python, яка надає простий спосіб взаємодії з базами даних SQLitE. SQLitE є легким, серверним базованим файловим СУБД, який не вимагає окремого серверу та налаштування [11].

Основні особливості SQLite3 [12]:

- працює без необхідності встановлення окремого сервера баз даних або налаштування складних конфігурацій. Вона просто зберігає дані в локальних файлах, що спрощує використання і розгортання.
- доступна для різних платформ, включаючи Windows, macOS, Linux та інші. Це дозволяє розробникам використовувати SQLite3 на різних пристроях та операційних системах.
- має дуже маленький обсяг, що робить його ідеальним вибором для обмежених ресурсів, таких як мобільні пристрої або вбудовані системи.
- використовує стандартний набір мови SQL для взаємодії з базою даних. Це означає, що розробники можуть використовувати розширені запити, оператори і функції SQL для роботи з даними.
- підтримує транзакції ACID (Atomicity, Consistency, Isolation, Durability), що забезпечує консистентність та надійність даних. Ви можете виконувати групу операцій як одну транзакцію, забезпечуючи атомарність та цілісність даних [13].
- SQLite3 є популярним вибором для розробки мобільних додатків, таких як Android- та iOS-додатки. Вона забезпечує зручне зберігання

та управління локальними даними, такими як налаштування, користувацькі дані та кеш.

- SQLite3 також може бути використана в веб-розробці для локального зберігання даних на серверах. Вона може бути використана для зберігання конфігурацій, сеансової інформації, кешу та інших даних, які не вимагають масштабного сервера баз даних.
- часто використовується в вбудованих системах, таких як малий комп'ютер Raspberry Pi або вбудовані системи IoT. Вона забезпечує можливість зберігати та управляти дані на самому пристрої без необхідності звертатися до зовнішнього сервера.
- використовується для тестування та навчання. Легкість у використанні та зручність, вбудована підтримка, робить її прекрасним інструментом для експериментів, розробки прототипів та навчання баз даних.

Враховуючи ці переваги, SQLite3 стає частим вибором для розробників, які шукають просту та ефективну реляційну базу даних для своїх проектів.

РОЗДІЛ 3. ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Послідовна розробка інтерфейсу програми

Для розробки було використано бібліотеку PyQt5, а саме такі її функції як: QtCorE, QtWidgEts, QtGui, QtSql.

PyQt - це набір інструментів графічного інтерфейсу віджетів, це інтерфейс Python до Qt, однієї з найпотужніших і найпопулярніших крос-платформених бібліотек GUI, PyQt розроблений компанією RivErBank Computing Ltd. Остання версія PyQt доступна на офіційному сайті RivErBankcomputing.com [14].

PyQt API-це набір модулів, що містять набір класів та функцій: Модуль QtCore містить функції неграфічного інтерфейсу для маніпулювання файлами, каталогами тощо, тоді як модуль QtGui містить усі графічні елементи керування. Модуль QtGui містить усі графічні елементи керування. Крім того, є модулі для роботи з XML (QtXml), SVG (QtSvg) та SQL (QtSql).

QtCorE - базовий клас безграфічного інтерфейсу,який використовується іншими модулями QtWidgEts - клас для створення класичних десктопних інтерфейсів користувача QtGui-компонент графічного інтерфейсу користувача QtSql-класдля інтеграції з базами даних за допомогою SQL [15].

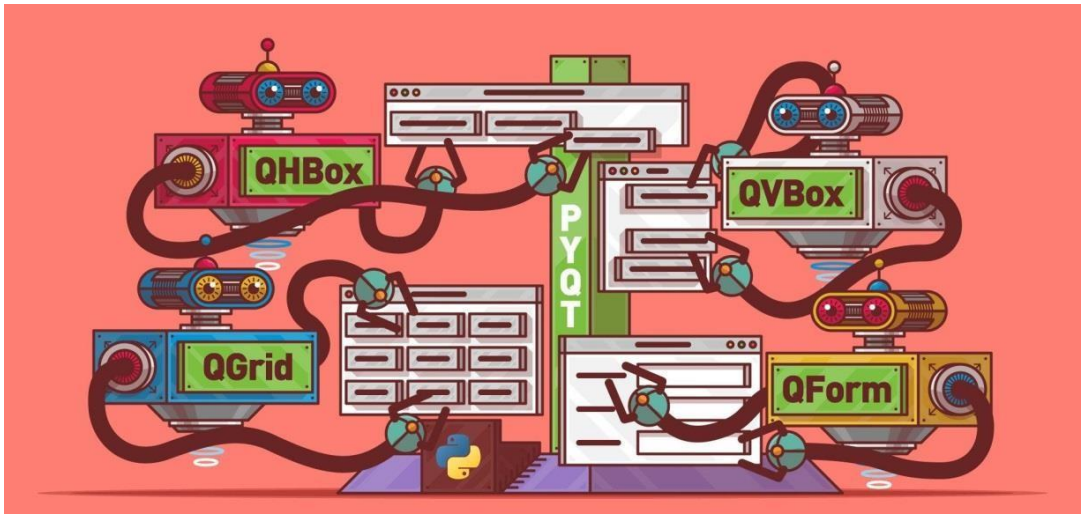

Рис. 3.1. Варіації класів у бібліотеки PyQt5

Qt Version	State
Qt 4.8.7	End-of-Life (as of December 2015)
Qt 5.12	End-of-Life (as of December 2021)
Qt 5.15	Long Term Support Release
Qt 6.2	Long Term Support Release
Qt 6.5	Long Term Support Release
Qt 6.6	Feature Development

Рис. 3.2. Версії Qt

Версій Qt є достатньо багато, за свій не дуже короткий час він пережив декілька версій, з яких на даний час най стабільнішою є PtQt5, яка працює зі всіма доступними класами і не тільки.

Головна сторінка застосунку виглядає наступним чином:

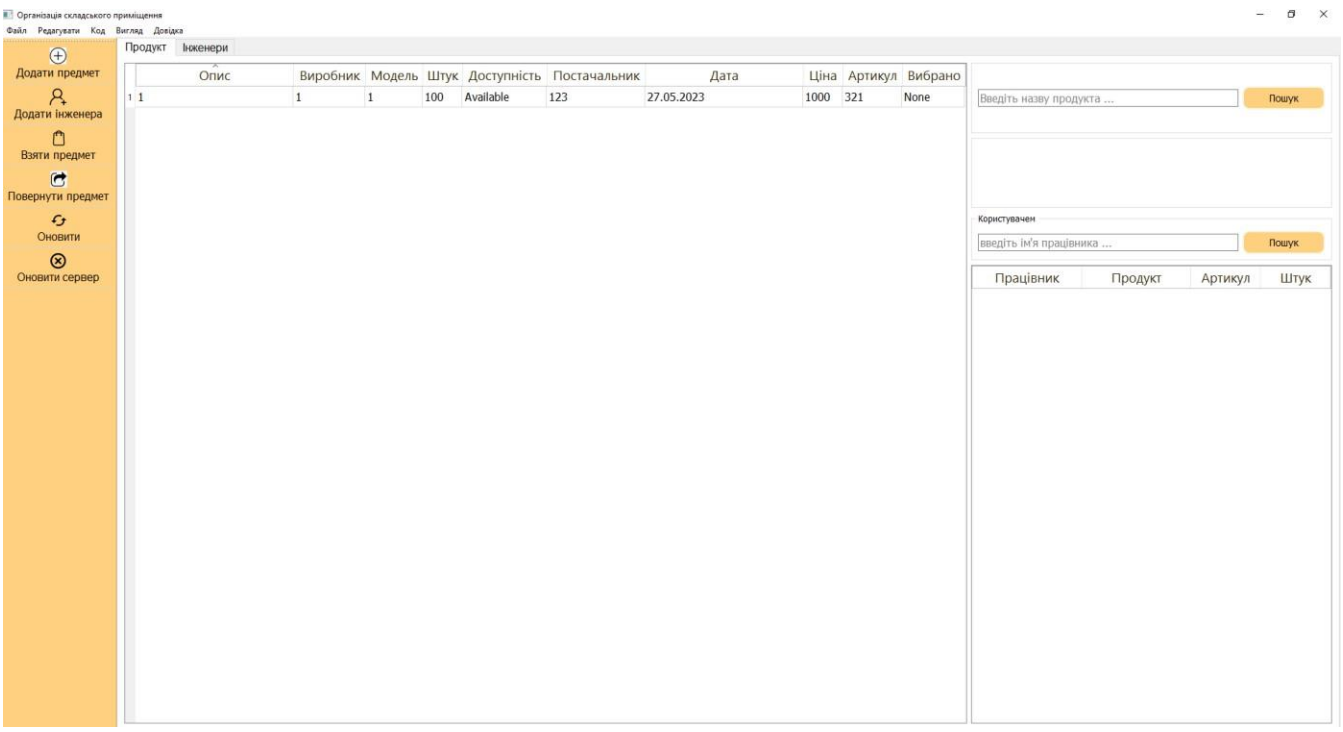

Рис. 3.3. Головна сторінка

Програмний код для цієї сторінки:

```
31  class Main(QMainWindow):
32      somedata = 10
33
34
35      def __init__(self):
36          super().__init__()
37          self.setWindowTitle("Організація складського приміщення")
38          self.setWindowIcon(QIcon("src/icons/main_win_logo.jpg"))
39          self.setGeometry(100, 200, 1000, 786)
40
```

Рис. 3.4. Титульна інформація головної сторінки

У цьому коді описано титульна назва програми, а також логотип і стартова геометрія самого вікна програми. Клас Qmain Window виконує роботу головного меню програми.

```
51
52      def about_programm1(self):
53          msg = QMessageBox()
54          msg.setWindowTitle("Про програму")
55          msg.setText("Вітаю")
56          msg.setIcon(QMessageBox.information)
57          msg.setInformativeText("111")
58
```

Рис. 3.5. Про програму

На цьому рисунку представлено за допомогою QMEssagEBox модальне діалогове вікно для інформування користувача або для запитання користувачеві та отримання відповіді.

```
59
60      def ui(self):
61          self.toolBar()
62          self.mainMenu()
63          self.tabWidget()
64          self.widgets()
65          self.layouts()
66          self.displayProducts()
67          self.displayMembers()
```

Рис. 3.6. Опис перемінних ui частини

Тут прописано всі перемінні які будуть використовуватись у побудові інтерфейсу.

```
69      def toolBar(self):
70          self.tb = self.addToolBar("Панель інструментів")
71          self.tb.setToolButtonStyle(Qt.ToolButtonTextUnderIcon)
72          self.tb.setStyleSheet("QToolBar {background-color:#ffd080;}")
73
74          # Tool Bar Buttons
75          self.addProductBtn = QAction(QIcon('src/icons/new_item.png'), "Додати предмет", self)
76          self.addProductBtn.triggered.connect(self.add_product_btn)
77          self.addProductBtn.setFont(QFont("Liberation Mono", 12))
78          self.tb.addAction(self.addProductBtn)
79          self.tb.addSeparator()
80
81
82          self.addMember = QAction(QIcon('src/icons/new_member.png'), "Додати інженера", self)
83          self.addMember.triggered.connect(self.add_member)
84          self.addMember.setFont(QFont("Liberation Mono",12))
85          self.tb.addAction(self.addMember)
86          self.tb.addSeparator()
87
88          self.sellProduct = QAction(QIcon('src/icons/take_item.png'), "Взяти предмет", self)
89          self.sellProduct.setFont(QFont("Liberation Mono", 12))
90          self.sellProduct.triggered.connect(self.sell_product)
91          self.tb.addAction(self.sellProduct)
92          self.tb.addSeparator()
93
94          self.returnProduct = QAction(QIcon('src/icons/return.png'), "Повернути предмет", self)
95          self.returnProduct.setFont(QFont("Liberation Mono", 12))
96          self.returnProduct.triggered.connect(self.return_product)
97          self.tb.addAction(self.returnProduct)
98          self.tb.addSeparator()
99
100         self.refreshTb = QAction(QIcon('src/icons/upgrade_btn.png'), "Оновити", self)
101         self.refreshTb.setFont(QFont("Arial", 12))
102         self.refreshTb.triggered.connect(self.refresh_all_tables)
103         self.tb.addAction(self.refreshTb)
104         self.tb.addSeparator()
105
106         self.update_to_DB = QAction(QIcon(update_db_img), update_DB_text, self)
107         self.update_to_DB.setFont(QFont("Arial", 12))
108         self.update_to_DB.triggered.connect(self.update_DB_database)
109         self.tb.addAction(self.update_to_DB)
110         self.tb.addSeparator()
```

Рис. 3.7. Опис лівої части навігації програми

У dEf toolbar прописані всі головні кнопки навігації, а саме панель інструментів, такі як “Додати предмет”, “Додати інженера”, “Взяти предмет”, “Повернути предмет”, “Оновити”. Клас QStylEOptionToolButton використовується для опису параметрів для малювання кнопки навігаційної частини.

Клас QAction - це абстракція користувацьких команд, які можна додавати до різних компонентів інтерфейсу користувача. Клас Qicon надає масштабовані іконки в різних режимах і станах. Клас QFont визначає вимоги до шрифту, що використовується для виведення тексту.

```
112     def mainMenu(self):
113         self.menubar = self.menuBar()
114         file_bar = self.menubar.addMenu("Файл")
115         edit_bar = self.menubar.addMenu("Редагувати")
116         code_bar = self.menubar.addMenu("Код")
117         view_bar = self.menubar.addMenu("Вигляд")
118         help_bar = self.menubar.addMenu("Довідка")
119 
120         # Sub menu items
121         save_to_excel = QAction("Зберегти як...", self)
122         file_bar.addAction(save_to_excel)
123         save_to_excel.triggered.connect(self.save_to_EXCEL_file)
124 
125         open_file = QAction("Відкрити", self)
126         open_file.setIcon(QIcon("src/icons/ds.png"))
127         file_bar.addAction(open_file)
128 
129         exit_file = QAction("Вихід", self)
130         exit_file.setIcon(QIcon("src/icons/cancel.png"))
131         exit_file.triggered.connect(self.exit_programm_func)
132         file_bar.addAction(exit_file)
133 
134         # Code Bar
135         check_statistic = QAction("Таблиця транзакцій", self)
136         check_statistic.triggered.connect(self.check_staticctics)
137         code_bar.addAction(check_statistic)
138 
139         clear_items_table = QAction("Очистити всі таблиці", self)
140         clear_items_table.triggered.connect(self.clear_all_items)
141         code_bar.addAction(clear_items_table)
142 
143         # Help_bar
144         about_prg = QAction("Про що", self)
145         about_prg.triggered.connect(self.about_programm1)
146         help_bar.addAction(about_prg)
147 
148         # View Bar
149         full_screen = QAction("Повноекранний режим", self)
150         full_screen.triggered.connect(self.enter_full_screen_mode)
151         view_bar.addAction(full_screen)
152 
```

Рис. 3.8. Опис верхньої части навігації програми

Тут використовуються всі ті класи які були описані вище, а також прописано додаткові елемнти цієї навігації.

```
153
154     def tabWidget(self):
155         font = QFont("Arial",12,10,False)
156         self.tabs = QTabWidget()
157         self.tabs.setStyleSheet(style.qTabWidgetStyle())
158         self.tabs.blockSignals(True)
159         self.tabs.currentChanged.connect(self.tabChanged)
160         self.setCentralWidget(self.tabs)
161         self.tabs1 = QWidget()
162         self.tabs2 = QWidget()
163         self.tabs.addTab(self.tabs1, "Продукт")
164         self.tabs.addTab(self.tabs2, "Інженери")
165         self.tabs.setFont(font)
166         self.tabs.setStyleSheet("QTabWidget {background-color:#90AFC5;}")
167
```

Рис. 3.9. Опис навігації по сторінках програми

У головній сторінці є 2 сторінки які використовуються, це “Продукт”, а також “Інженери”. Тут використовано QWidgEt, QFont, А також QTabWidgEt, який надає декілька віджетів з вкладками.

```
168     def widgets(self):
169         # [Tab_1] Widgets
170         # Main Left Layout Widgets
171         self.productsTable = QTableWidget()
172 
173 
174         self.productsTable.setColumnCount(11)
175         self.productsTable.setColumnHidden(0, True)
176         self.productsTable.setHorizontalHeaderItem(0, QTableWidgetItem("Id Продукту"))
177         self.productsTable.setHorizontalHeaderItem(1, QTableWidgetItem("Опис"))
178         self.productsTable.setHorizontalHeaderItem(2, QTableWidgetItem("Виробник"))
179         self.productsTable.setHorizontalHeaderItem(3, QTableWidgetItem("Модель"))
180         self.productsTable.setHorizontalHeaderItem(4, QTableWidgetItem("Штук"))
181         self.productsTable.setHorizontalHeaderItem(5, QTableWidgetItem("Доступність"))
182         self.productsTable.setHorizontalHeaderItem(6, QTableWidgetItem("Постачальник"))
183         self.productsTable.setHorizontalHeaderItem(7, QTableWidgetItem("Дата"))
184         self.productsTable.setHorizontalHeaderItem(8, QTableWidgetItem("Ціна"))
185         self.productsTable.setHorizontalHeaderItem(9, QTableWidgetItem("Артикул"))
186         self.productsTable.setHorizontalHeaderItem(10, QTableWidgetItem("Вибрано"))
187         #self.productsTable.setHorizontalHeaderItem(11, QTableWidgetItem("Some"))
188 
189         self.productsTable.horizontalHeader().setSectionResizeMode(1, QHeaderView.Stretch)
190         self.productsTable.horizontalHeader().setSectionResizeMode(2, QHeaderView.ResizeToContents)
191         self.productsTable.horizontalHeader().setSectionResizeMode(3, QHeaderView.ResizeToContents)
192         self.productsTable.horizontalHeader().setSectionResizeMode(4, QHeaderView.ResizeToContents)
193         self.productsTable.horizontalHeader().setSectionResizeMode(5, QHeaderView.ResizeToContents)
194         self.productsTable.horizontalHeader().setSectionResizeMode(6, QHeaderView.ResizeToContents)
195         self.productsTable.horizontalHeader().setSectionResizeMode(7, QHeaderView.Stretch)
196 
197 
198         self.productsTable.doubleClicked.connect(self.selected_product) #TODO double click event
199         self.productsTable.cellClicked.connect(self.sell_picked_clicked)
200         self.productsTable.setContextMenuPolicy(Qt.ActionsContextMenu)
201         self.quitAction = QAction("Добавити продукт", self)
202         self.quitAction.triggered.connect(self.add_product_btn)
203         self.productsTable.addAction(self.quitAction)
204 
205         self.productsTable.horizontalHeader().setStyleSheet(style.horizontalHeaderView())
206         self.productsTable.setStyleSheet(style.forQTabWidget())
```

Рис. 3.10. Опис “Продукт”

Використовуються всі ті самі класи що і у попередніх рисунках, прописано за допомогою QTabWidgEtItEm, а також клас QHEadErViEw надає рядки заголовків та стовпці заголовків для представлень елементів.

```
        # TODO Second Table
        self.productsTable2 = QTableWidget()
        self.productsTable2.setColumnCount(5)
        self.productsTable2.setColumnHidden(0, True)
        self.productsTable2.setHorizontalHeaderItem(0, QTableWidgetItem("id"))
        self.productsTable2.setHorizontalHeaderItem(1, QTableWidgetItem("Працівник"))
        self.productsTable2.setHorizontalHeaderItem(2, QTableWidgetItem("Продукт"))
        self.productsTable2.setHorizontalHeaderItem(3, QTableWidgetItem("Артикул"))
        self.productsTable2.setHorizontalHeaderItem(4, QTableWidgetItem("Штук"))
        self.productsTable2.horizontalHeader().setSectionResizeMode(1, QHeaderView.Stretch)
        self.productsTable2.horizontalHeader().setSectionResizeMode(2, QHeaderView.Stretch)
        self.productsTable2.horizontalHeader().setStyleSheet(style.horizontalHeaderView())
        self.productsTable2.setStyleSheet(style.forQTabWidget())
        self.productsTable2.setSortingEnabled(True)
        # Right Top Layout Widgets
        self.searchText = QLabel("Пошук")
        self.searchEntry = QLineEdit()
        self.searchEntry.setPlaceholderText("Введіть назву продукта ...")
        self.searchEntry.setStyleSheet('QLineEdit{border-color: #A3C1DA; font-size: 12pt; font: Liberation Mono}')
        self.searchBtn = QPushButton("Пошук")
        self.searchBtn.setStyleSheet(style.search_btn_style())
        self.searchBtn.clicked.connect(self.search_products_btn)
```

Рис. 3.11. Опис “Інженери”

Тут прописано всі поля які використовуються на цій сторінці.

```
        # Right Bottom layout Widgets
        self.searchEntryPickedByEmployee = QLineEdit()
        self.searchEntryPickedByEmployee.setPlaceholderText("введіть ім'я працівника ...")
        self.searchEntryPickedByEmployee.setStyleSheet('QLineEdit{border-color: #A3C1DA; font-size: 12pt; font: Liberation Mono}')
        self.searchByEmployee = QPushButton("Пошук")
        self.searchByEmployee.setStyleSheet(style.search_btn_style())
        self.searchByEmployee.clicked.connect(self.search_products_by_emmployee_btn)

        # [Tab_2] Widgets
        self.membersTableWidgets = QTableWidget()
        self.membersTableWidgets.horizontalHeader().setStyleSheet(style.horizontalHeaderView())
        self.membersTableWidgets.setStyleSheet(style.forQTabWidget())
        self.membersTableWidgets.setColumnCount(4)
        self.membersTableWidgets.setHorizontalHeaderItem(0, QTableWidgetItem("Користувач Id"))
        self.membersTableWidgets.setHorizontalHeaderItem(1, QTableWidgetItem("Користувач Ім'я"))
        self.membersTableWidgets.setHorizontalHeaderItem(2, QTableWidgetItem("Користувач Позиція"))
        self.membersTableWidgets.setHorizontalHeaderItem(3, QTableWidgetItem("Користувач ID"))
        self.membersTableWidgets.horizontalHeader().setSectionResizeMode(1, QHeaderView.Stretch)
        self.membersTableWidgets.horizontalHeader().setSectionResizeMode(2, QHeaderView.Stretch)
        self.membersTableWidgets.horizontalHeader().setSectionResizeMode(3, QHeaderView.Stretch)
        self.membersTableWidgets.doubleClicked.connect(self.selected_member)
        self.memberSearchText = QLabel("Пошук користувачів")
        self.memberSearchEntry = QLineEdit()
        self.memberSearchEntry.setPlaceholderText("ВВедіть назву ...")
        self.membersearchBtn = QPushButton("Пошук")
        self.membersearchBtn.clicked.connect(self.search_members_btn)
        self.membersTableWidgets.setContextMenuPolicy(Qt.ActionsContextMenu)
        self.addMemberSmall = QAction("Добавити користувача", self)
        self.addMemberSmall.triggered.connect(self.add_member)
        self.membersTableWidgets.addAction(self.addMemberSmall)
```

Рис. 3.12. Права частина програми

У цій частині коду прописано всі кнопки, а також поля для вводу інформації для пошуку, а також і інших маніпуляцій з програмою.

```
714  class DisplayMember(QWidget):
715      def __init__(self, main):
716          super().__init__()
717          self.setWindowTitle("Деталі користувача")
718          self.setWindowIcon(QIcon("src/icons/add_emp.png"))
719          self.setGeometry(550, 250, 350, 600)
720          self.setFixedSize(700, 300)
721          self.main = main
722          self.ui()
723          self.show()
```

Рис. 3.13. Деталі користувача

Тепер розглянемо детальніше кожну із сторінок:

1. **Додати предмет:** Ця кнопка додає предмет до наявної бази даних зразу виводить у поле “Продукт”

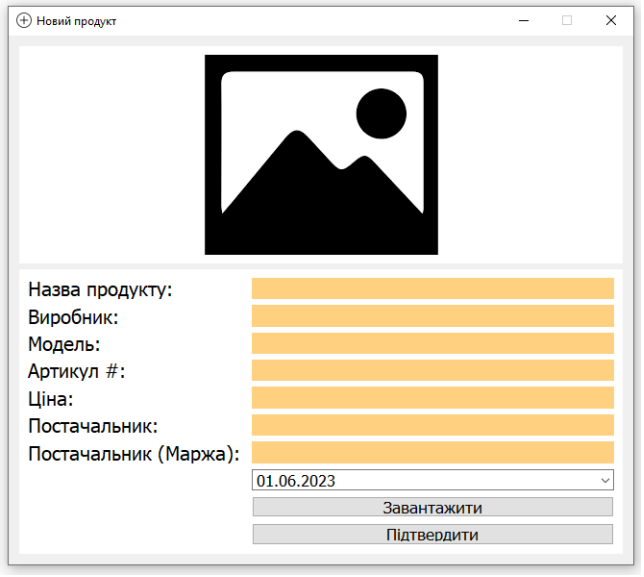

Рис. 3.14. Екран добавлення продукту

Тут можна записувати в поля різні дані, для майбутньої зручності самого додатку, це для пошуку товару, видалення і т.д., для будь яких маніпуляцій з цим додатком.

Програмний код саме цього вікна виглядає ось так:

```
15
16   con = sqlite3.connect("db_database/main_database.db")
17   cur = con.cursor()
```

Рис. 3.15. Підключення до БД

Тут було використано бібліотеку sqlitE3 для підключення до загальної бази даних.

```
19    defaulImg = 'src/icons/upload_new_img.png'
20    real_img_name = ''
```

Рис. 3.16. Завантаження іконки

```
23  class AddProduct(QWidget):
24      background = True
25
26      def __init__(self, main):
27          super().__init__()
28          self.setWindowTitle("Новий продукт")
29          self.setWindowIcon(QIcon('src/icons/new_item.png'))
30          self.setGeometry(650, 300, 650, 550)
31          self.setFixedSize(self.size())
32          self.main = main
33          self.snippingTool = SnippingWidget(self)
34          self.ui()
35          self.show()
```

Рис. 3.17. Титульна частина сторінки

Для титульної інформації було використано функцію бібліотеки, а саме QWidgEt, за допомогою якої можна добавляти різні частини програми.

```
40
41      def widgets(self):
42          # Top Layout
43          self.addProductImg = QLabel()
44          self.addProductImg.setPixmap(QPixmap('src/icons/upload_new_img.png'))
45          #self.addProductImg.mouseDoubleClickEvent = self.some_stuff
46          #self.titleText = QLabel("Add Product")
47          self.addProductImg.setAlignment(Qt.AlignCenter)
48          #self.addProductImg.mousePressEvent = self.snipping_tool
49          # Bottom Layout
50          self.descriptionEntry = QLineEdit()
51          self.descriptionEntry.setStyleSheet(style.search_btn_style_2())
52          self.manufacturerEntry = QLineEdit()
53          self.supplier = QLineEdit()
54          self.supplier.setStyleSheet(style.search_btn_style_2())
55          self.manufacturerEntry.setStyleSheet(style.search_btn_style_2())
56          self.nameEntry = QLineEdit()
57          self.nameEntry.setStyleSheet(style.search_btn_style_2())
58          #self.nameEntry.setPlaceholderText("Enter name of product ...")
59          self.manufactuterEntry = QLineEdit()
60          self.manufactuterEntry.setStyleSheet(style.search_btn_style_2())
61          #self.manufactuterEntry.setPlaceholderText("Enter name of manufacturer ...")
62          self.PoNumber = QLineEdit()
63          self.PoNumber.setStyleSheet(style.search_btn_style_2())
64
65          self.priceEntry = QLineEdit()
66          self.priceEntry.setStyleSheet(style.search_btn_style_2())
67          #self.priceEntry.setPlaceholderText("Enter price of product ...")
68          self.quotaEntry = QLineEdit()
69          self.quotaEntry.setStyleSheet(style.search_btn_style_2())
70
71          self.timeAndDateEdit = QDateEdit()
72          self.timeAndDateEdit.setDateTime(QDateTime.currentDateTime())
73          self.timeAndDateEdit.setCalendarPopup(True)
74          self.timeAndDateEdit.setStyleSheet("QDateEdit{font-size: 13pt; font: Liberation Mono}")
75          #self.timeAndDateEdit.setStyleSheet(style.time_edit())
76          #self.quotaEntry.setPlaceholderText("Enter quota of product ...")
77          self.uploadBtn = QPushButton("Завантажити")
78          self.uploadBtn.setStyleSheet("QPushButton{font-size: 13pt; font: Liberation Mono}")
79          self.uploadBtn.clicked.connect(self.upload_img_btn)
80          self.submitBtn = QPushButton("Підтвердити")
81          self.submitBtn.setStyleSheet("QPushButton{font-size: 13pt; font: Liberation Mono}")
82          self.submitBtn.clicked.connect(self.add_product)
83
```

Рис. 3.18. Введеня у форму (backEnd)

На цьому рисунку продемонстровано всю частину вводу у форму. Віджет QLabEl забезпечує відображення тексту або зображення, щоб краще відображувати ці функції потрібно використовувати srtPixmap (QPixmap). Qt.AlignCEntEr - центрує горизонтально в доступному просторі. QlinEEdit – це текстовий редактор. Клас QDatEEdit надає віджет для редагування дат на основі QDatETimEdit. Віджет QPushButton відповідає за командну кнопку, тобто за нажимання кнопок у самій програмі.

```
84      def layouts(self):
85
86          self.mainLayout = QVBoxLayout()
87          self.topLayout = QHBoxLayout()
88          self.bottomLayout = QFormLayout()
89          self.topFrame = QFrame()
90          self.topFrame.setStyleSheet(style.sell_product_top_frame())
91          self.bottomFrame = QFrame()
92          self.bottomFrame.setStyleSheet(style.sell_product_bottom_frame())
93
94          # Add widgets
95          self.topLayout.addWidget(self.addProductImg)
96          #self.topLayout.addWidget(self.titleText)
97          self.topFrame.setLayout(self.topLayout)
98
99          # Widgets of Form layout
100         self.bottomLayout.addRow(QLabel("Назва продукту: "), self.descriptionEntry)
101         self.bottomLayout.addRow(QLabel("Виробник: "), self.manufacturerEntry)
102         self.bottomLayout.addRow(QLabel("Модель: "), self.nameEntry)
103         self.bottomLayout.addRow(QLabel("Артикул #: "), self.PoNumber)
104         self.bottomLayout.addRow(QLabel("Ціна: "), self.priceEntry)
105         self.bottomLayout.addRow(QLabel("Постачальник: "), self.supplier)
106         self.bottomLayout.addRow(QLabel("Постачальник (Маржа): "), self.quotaEntry)
107         self.bottomLayout.addRow(QLabel(""), self.timeAndDateEdit)
108         self.bottomLayout.addRow(QLabel(""), self.uploadBtn)
109         self.bottomLayout.addRow(QLabel(""), self.submitBtn)
110         self.bottomFrame.setLayout(self.bottomLayout)
111
112         self.mainLayout.addWidget(self.topFrame)
113         self.mainLayout.addWidget(self.bottomFrame)
114
115         self.setLayout(self.mainLayout)
116
```

Рис. 3.19. Введеня у форму (frontEnd)

Наведений код відповідає за введеня у форму, а саме візуальної частини. Тут описана за допомогою QLabEl назви стрічок які потрібно заповнити для комунікації з програмою

2. **Додати інженера:** Ця кнопка додає нового інженера до наявної бази даних зразу виводить у поле “Інженери”

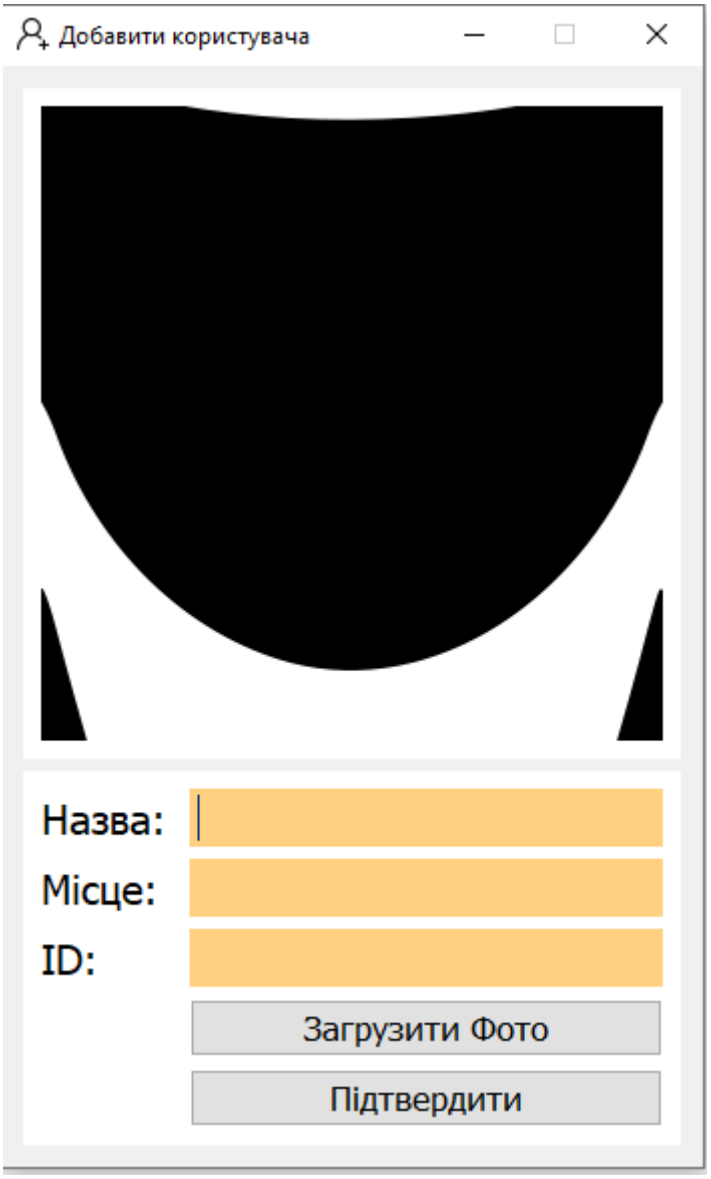

Рис. 3.20. Екран добавлення нового інженера.

На цій сторінці можливий ввід “Назва”, “Місце”, і “ID”, нового інженера, також присутні 2 кнопки, “Загрузити фото” для завантаження фотографії, або аватару нового інженера і “Підтвердити”, для підтвердження введеної інформації і добавлення нового інженера.

Ось програмний код цієї сторінки:

```
class AddMember(QWidget):
    def __init__(self, main):
        super().__init__()
        self.setWindowTitle("Добавити користувача")
        self.setWindowIcon(QIcon('src/icons/new_member.png'))
        self.setGeometry(650, 300, 350, 550)
        self.setFixedSize(self.size())
        self.empl_dc = dict()
        self.main = main
        self.chek_empl()
        self.ui()
        self.show()
```

Рис. 3.21. Титульна інформація користувача.

У цьому розділі коду прописана титульна інформація, разом із фотографією (іконкою) цієї сторінки, а також прописана його початкова позиція на екрані.

```
    def widgets(self):
        # Widgets of top layout
        self.addMemberImg = QLabel()
        self.addMemberImg.setPixmap(QPixmap('src/icons/add_engineer.png'))
        self.addMemberImg.setAlignment(Qt.AlignCenter)
        #self.titleText = QLabel("Add Member")
        #self.titleText.setAlignment(Qt.AlignCenter)

        # Widgets of bottom layout
        self.nameEntry = QLineEdit()
        self.nameEntry.setStyleSheet(style.search_btn_style_2())
        #self.nameEntry.setPlaceholderText("Enter name of member ...")
        self.surnameEntry = QLineEdit()
        self.surnameEntry.setStyleSheet(style.search_btn_style_2())
        #self.surnameEntry.setPlaceholderText("Enter surname of member ...")
        self.phonenumberEntry = QLineEdit()
        self.phonenumberEntry.setStyleSheet(style.search_btn_style_2())
        #self.phonenumberEntry.setPlaceholderText("Enter phone number of member ...")
        self.uploadBtn = QPushButton("Загрузити Фото")
        self.uploadBtn.setStyleSheet("QPushButton{font-size: 13pt; font: Liberation Mono}")
        self.uploadBtn.clicked.connect(self.upload_img_btn)
        self.submitBtn = QPushButton("Підтвердити")
        self.submitBtn.setStyleSheet("QPushButton{font-size: 13pt; font: Liberation Mono}")
        self.submitBtn.clicked.connect(self.add_member_btn)
```

Рис. 3.22. Кнопки цієї сторінки.

Тут прописано початкову (стандартну) фотографію яка іде як загальна для всіх, за допомогою QLabEl і його під класу QPixmap, а також центрувати по

горизонту за допомогою Qt.AlignCEntEr. Також за допомогою QPushButton записано назву кнопки, а також його стиль.

```
58
59      def layouts(self):
60          self.mainLayout = QVBoxLayout()
61          self.topLayout = QVBoxLayout()
62          self.bottomLayout = QFormLayout()
63          self.topFrame = QFrame()
64          self.topFrame.setStyleSheet(style.sell_product_top_frame())
65          self.bottomFrame = QFrame()
66          self.bottomFrame.setStyleSheet(style.sell_product_bottom_frame())
67
68          # Add widgets
69          #self.topLayout.addWidget(self.titleText)
70          self.topLayout.addWidget(self.addMemberImg)
71          self.topFrame.setLayout(self.topLayout)
72          self.bottomLayout.addRow(QLabel("Назва: "), self.nameEntry)
73          self.bottomLayout.addRow(QLabel("Місце: "), self.surnameEntry)
74          self.bottomLayout.addRow(QLabel("ID: "), self.phonenumberEntry)
75          self.bottomLayout.addRow(QLabel(""), self.uploadBtn)
76          self.bottomLayout.addRow(QLabel(""), self.submitBtn)
77          self.bottomFrame.setLayout(self.bottomLayout)
78
79          self.mainLayout.addWidget(self.topFrame)
80          self.mainLayout.addWidget(self.bottomFrame)
81
```

Рис. 3.23. Поля сторінки.

Тут описано які саме поля будуть доступні на цій сторінці, за допомогою QLabel.

3. **Авторизація:** сторінка авторизація випливає тоді коли користувач хоче зробити маніпуляції з базою даних, будь зміна її, чи видалення.

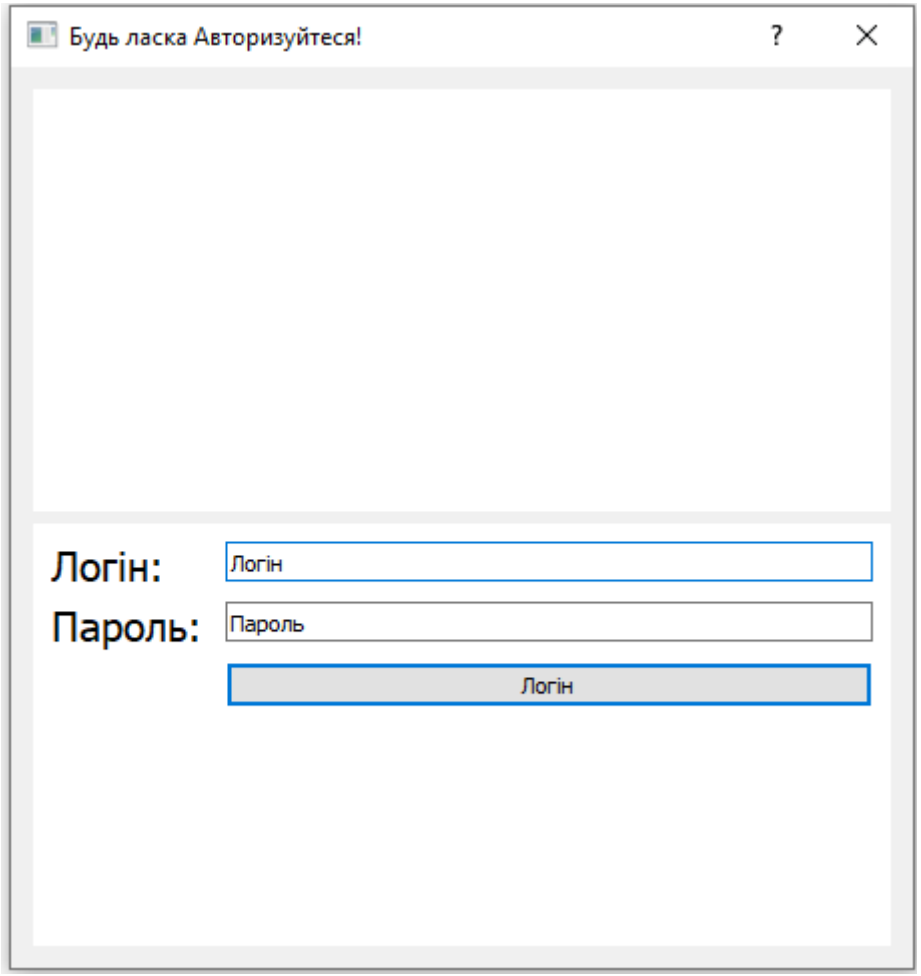

Рис. 3.24. Сторінка авторизації.

На цій сторінці є 2 поля вводу інформації, а саме “Логін”, та “Пароль” і кнопка “Логін”, так як реєстрація нового інженера відбувається в самій програмі.

```
7   class PasswordInitializer(QDialog):
8       def __init__(self, parent=None):
9           super(PasswordInitializer, self).__init__(parent)
10          self.setWindowTitle("Будь ласка Авторизуйтеся!")
11          self.setWindowIcon(QIcon('src/icons/power_module.png'))
12          self.setGeometry(650, 300, 450, 450)
13          self.setFixedSize(self.size())
14
```

Рис. 3.25. Титульна інформація.

Титульна інформація, а також фотографія самої сторінки і позиція сторінки.

```
21
22      def widgets(self):
23          self.textName = QLineEdit(self)
24          self.textName.setText("Логін")
25          self.textPass = QLineEdit(self)
26          self.textPass.setText("Пароль")
27          self.btnLogin = QPushButton("Логін", self)
28          self.btnLogin.clicked.connect(self.handlePassword)
29          self.addProductImg = QLabel()
30          self.addProductImg.setPixmap(QPixmap('src/icons/main_win_logo.jpg'))
31          self.addProductImg.setAlignment(Qt.AlignCenter)
32
```

Рис. 3.26. Загальна інформація на сторінці.

QLinEEdit – за допомогою цього, як було вже вище описано, ми прописуємо які в нас будуть поля, а також за допомогою QPushButton, ми описуємо нашу кнопку.

```
57
58      def handlePassword(self):
59          if (self.textName.text() == 'Логін' and self.textPass.text() == 'Пароль'):
60              self.accept()
61          else:
62              QMessageBox.warning(self, 'Помилка!', 'Неправильний користувач або пароль!')
63
```

Рис. 3.27. Помилка.

На цьому коді описано опис помилки і як саме вона буде показано користувачу, а саме буде перевірка по таким полям як Логін, та Пароль, за допомогою QMEssagEBox – ми якраз і виводимо на екран потрібну нам інформацію.

А саме його властивість .warning видасть нам помилку Увага, а саме те щоб користувач звернув увагу на цей тип помилки і зробив певні дії для його вирішення і не використання цієї помилки в подальшому використанні програми, цей опис помилки буде показуватись кожен раз як тільки буде у цьому потреба.

3.2. Тестування програми

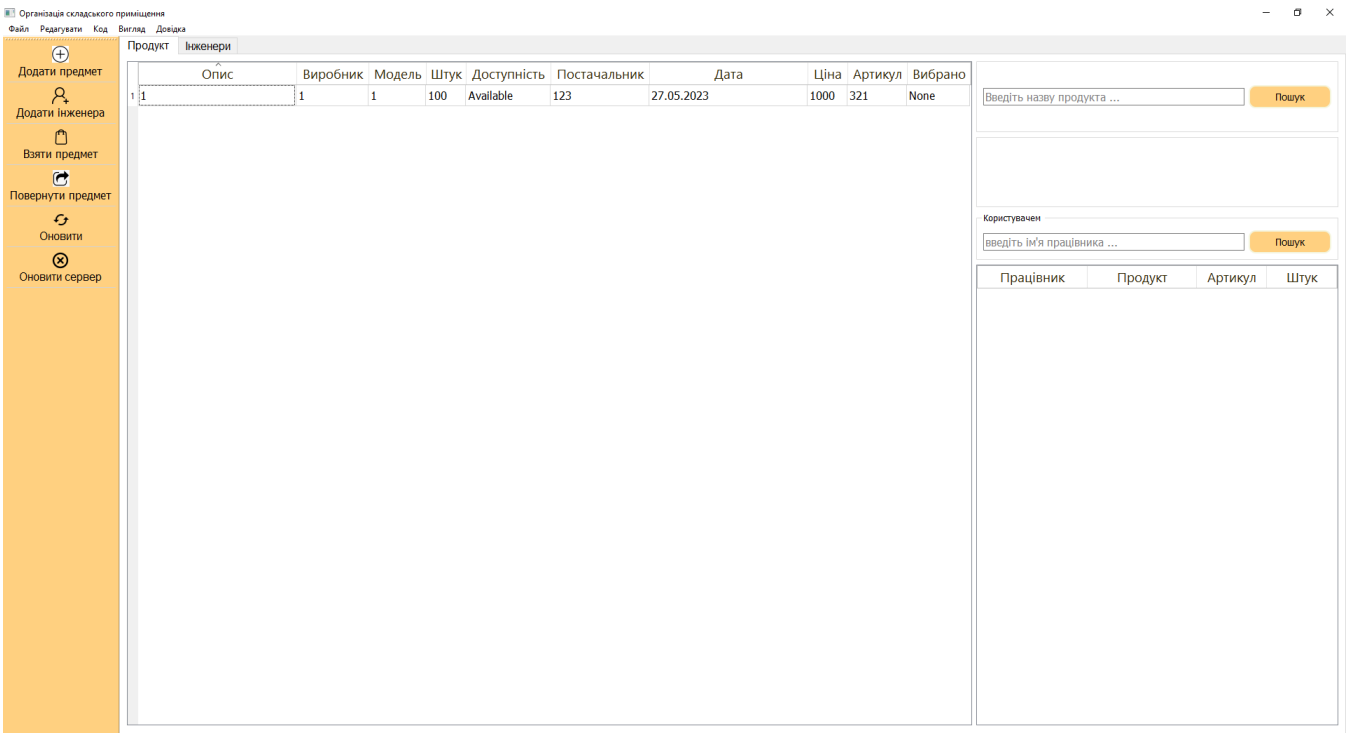

Рис. 3.28. Головна сторінка.

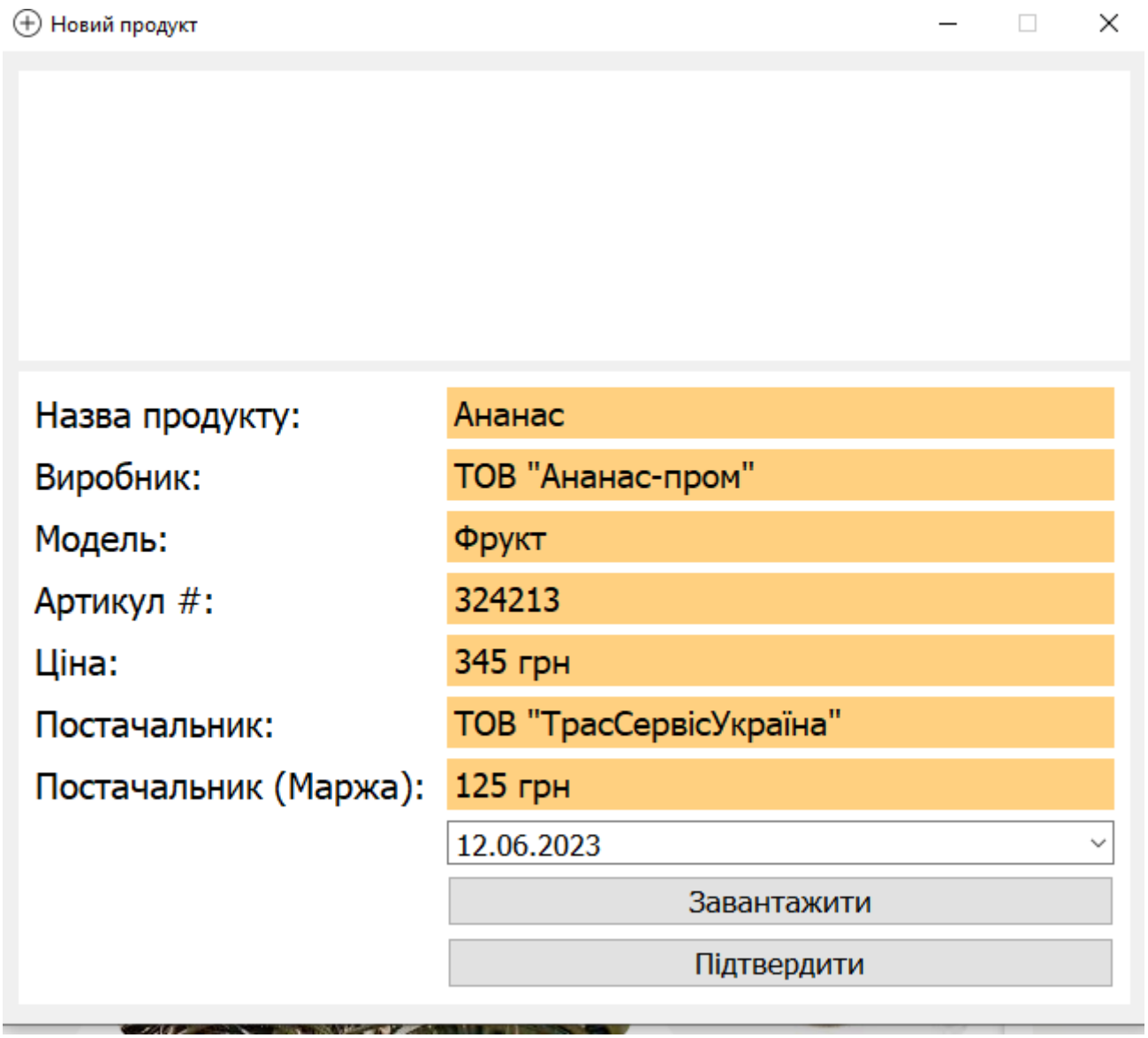

Рис. 3.29. Новий продукт.

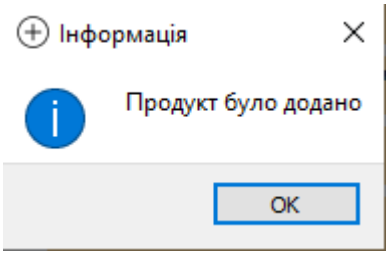

Рис. 3.30. Інформація про товар.

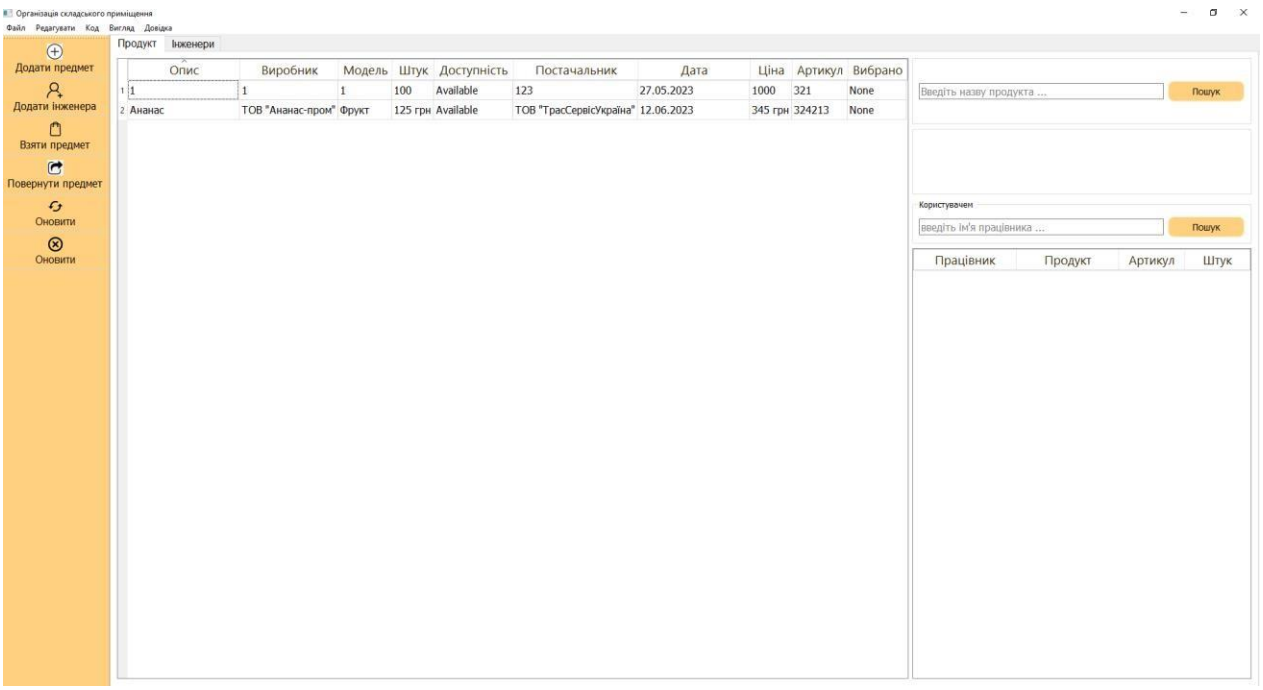

Рис. 3.31. Оновлена головна сторінка.

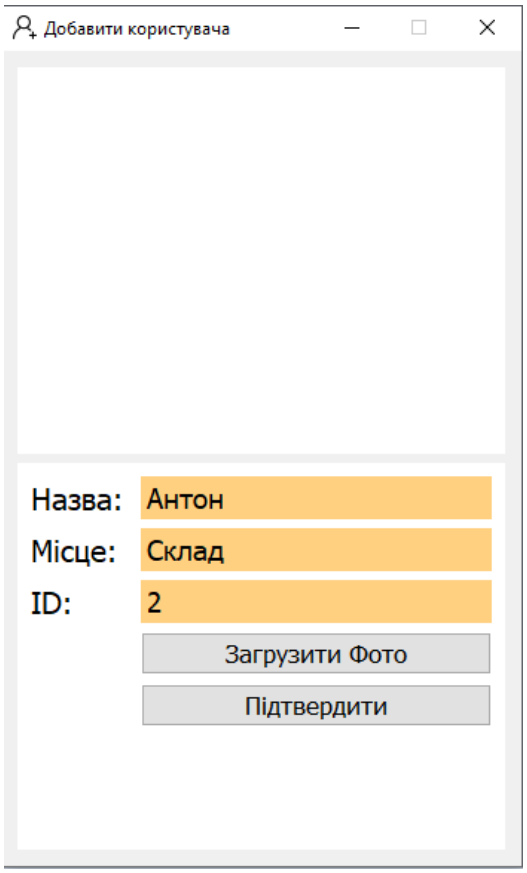

Рис. 3.32. Добавлення користувача.

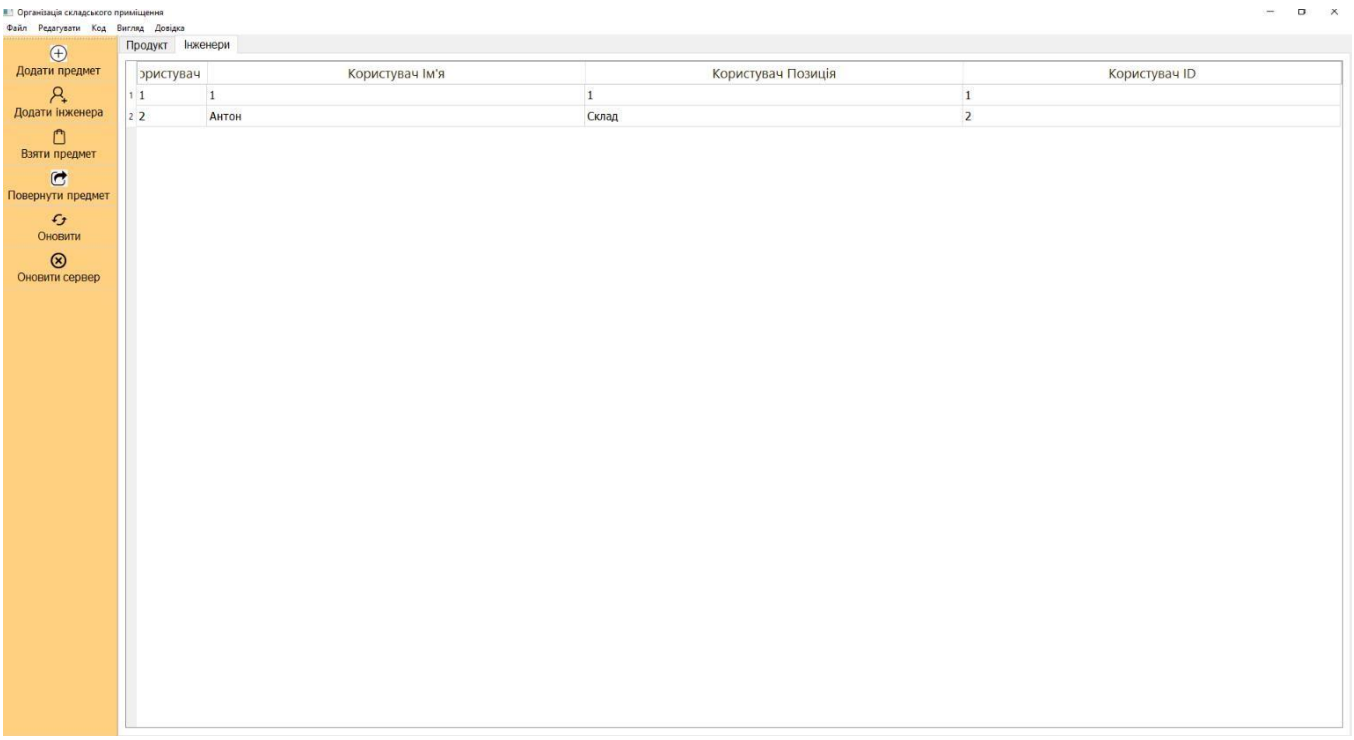

Рис. 3.33. Оновлена інформація на сторінці Інженери.

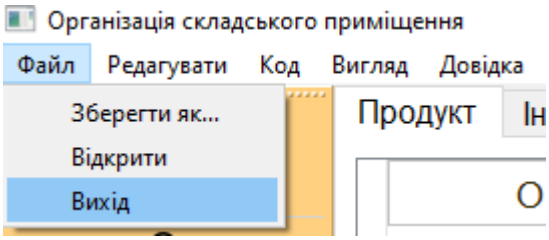

Рис. 3.34. Навігація “Файл”.

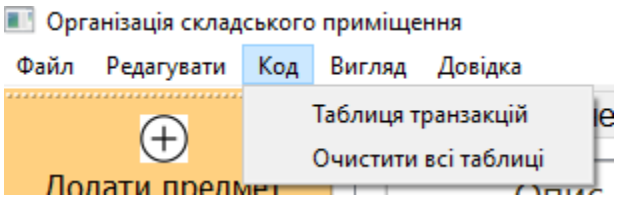

Рис. 3.35. Навігація “Код”.

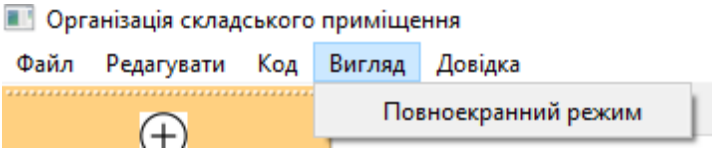

Рис. 3.36. Навігація “Вигляд”.

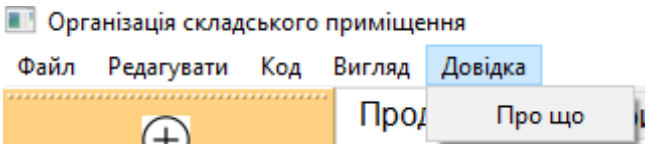

Рис. 3.37. Навігація “Довідка”.

ВИСНОВКИ

У дипломній роботі розроблено клієнт-серверну інформаційну систему складського обліку товарів (FrontEnd). Під час реалізації було використано бібліотеку PyQt5, а також класи для неї. Базу даних було створено за допомогою sqlitE3. У цій програмі реалізовано додавання, видалення, та редагування продуктів, а також інженерів які ці продукти добавляють. Реалізовано пошук по базі даних. Інтерфейс було розроблено на основі аналізу пропозицій цільової аудиторії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Joshua Willman: Beginning PyQt: A Hands-on Approach to GUI Programming with PyQt6, Paperback, 2022.
2. Michael Herrmann: Python and Qt: The Best Parts, March 2022.
3. PyQt5 - Quick Guide [Електроний ресурс] - Режим доступу до ресурсу. tutorialspoint.com/pyqt5/pyqt5_quick_guidE.htm - доступний станом на 12.06.2023.
4. Qt Documentation [Електроний ресурс] - Режим доступу до ресурсу. https://doc.qt.io/ - доступний станом на 12.06.2023.
5. Python Documentation [Електроний ресурс] - Режим доступу до ресурсу. https://www.python.org/ - доступний станом на 12.06.2023.
6. PyQt Documentation [Електроний ресурс] - Режим доступу до ресурсу. https://wiki.python.org/moin/PyQt - доступний станом на 12.06.2023.
7. Richards G. Warehouse Management: A Complete Guide to Improving Efficiency and Minimizing Costs in the Modern Warehouse, 4th ed, Kogan Page, 2019.
8. Emmett S. Excellence in Warehouse Management: How to Minimise Costs and Maximise Value, Kogan Page, 2021.
9. Downey A. Think Python: How to Think Like a Computer Scientis*t*, 2nd ed, O'Reilly Media, 2020.
10. Summerfield M. Rapid GUI Programming with Python and Qt, Apress, 2020.
11. Beaulieu A. Learning SQL: Generate, Manipulate, and Retrieve Data, 3rd ed, O'Reilly Media, 2020.
12. Sweigart A. Automate the Boring Stuff with Python, 2nd ed, No Starch Press, 2020.
13. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems, O'Reilly Media, 2022.
14. Lee Z. Hands-On GUI Programming with Python, Packt Publishing, 2021.
15. Fitzpatrick M. Create GUI Applications with Python & Qt, 2023.

ДОДАТКИ

```
dEf sEarch_box_stylE():
    rEturn """
                QGroupBox{
                    background-color:#ffd080;
                    font:15pt TimEs Bold;
                    color:whitE;
                    bordEr:2px solid gray;
                    bordEr-radius:15px
                }
    """


dEf list_box_stylE():
    rEturn """
                QGroupBox{
                    background-color:#ffd080;
                    font:20pt TimEs Bold;
                    color:whitE;
                    bordEr:2px solid gray;
                    bordEr-radius:15px
                }
    """

dEf sEarch_btn_stylE():
    rEturn """
                QPushButton{
                    background-color:#ffd080;
                    bordEr-stylE:solid;
                    bordEr-width:2px;
                    bordEr-radius:10px;
                    bordEr-color:bEigE;
                    font:14px;
                    padding:6px;
                    min-width:6Em;
                    font-family:LibEration Mono;
                }
    """

dEf simplE_btn_stylE():
    rEturn """
                QPushButton{

                    font-family:GEorgia;
```

```
                }
    """
```

39

```
dEf mEmbErsPicsStylE():
    rEturn """
                    QLabEl{
                        background-color:#F0F8FF;
                        bordEr-stylE:solid;
                        bordEr-radius:20px;
                        bordEr-color:bEigE;
                        font:12px;
                        padding:6px;
                        min-width:6Em;
                        font-family:Arial;
                    }
        """


dEf sEarch_btn_stylE_2():
    rEturn """
                    QLinEEdit{
                        background-color:#ffd080;
                        bordEr-stylE:solid;

                        bordEr-color:bEigE;
                        font:19px;
                        padding:2px;
                        min-width:2Em;
                        font-family:LibEration Mono;
                    }
        """


dEf qLinEEditREd():
    rEturn """
                    QLinEEdit{
                        background-color:#ffd080;
                        bordEr-stylE:solid;

                        bordEr-color:rEd;
                        font:12px;
                        padding:6px;
                        min-width:6Em;
                        font-family:Arial;

                        bordEr-width:1px;
                        bordEr-radius:1px;
```

```
                }
        """
```

40

```
dEf timE_Edit():
    rEturn """
                    QDatEEdit{
                        background-color:#ffd080;
                        bordEr-stylE:solid;
                        bordEr-width:2px;
                        bordEr-radius:10px;
                        bordEr-color:bEigE;
                        font:12px;
                        padding:6px;
                        min-width:6Em;
                        font-family:GEorgia;
                    }
            """

dEf sEarch_btn_stylE_3():
    rEturn """
                QComboBox{
                    background-color:#ffd080;
                    bordEr-stylE:outsEt;
                    bordEr-color:bEigE;
                    font:12px;
                    padding:6px;
                    min-width:6Em;
                    font-family:GEorgia;
                }
        """

dEf list_btn_stylE():
    rEturn """
              QPushButton{
                  background-color:#ffd080;
                  bordEr-stylE:outsEt;
                  bordEr-width:2px;
                  bordEr-radius:10px;
                  bordEr-color:bEigE;
                  font:12px;
                  padding:6px;
                  min-width:6Em;
              }
    """
```

```
dEf product_bottom_framE():
    rEturn """
                QFramE{
                    font:15pt TimEs Bold;
                    background-color:whitE;

                }
```

41

```
    """


dEf product_top_framE():
    rEturn """
                QFramE{
                    font:15pt TimEs Bold;
                    background-color:whitE;

                }
    """


dEf mEmbEr_top_framE():
    rEturn """
                QFramE{
                    font:12pt TimEs Bold;
                    background-color:whitE;
                }
    """


dEf mEmbEr_bottom_framE():
    rEturn """
                QFramE{
                    font:12pt TimEs Bold;
                    background-color:whitE;
                }
    """


dEf sEll_product_top_framE():
    rEturn """
                QFramE{
                    font:12pt TimEs Bold;
                    background-color:whitE;
                }
    """


dEf sEll_product_bottom_framE():
```

```
    rEturn """
                QFramE{
                    font:15pt TimEs Bold;
                    background-color:whitE;
                }
    """


dEf confirm_product_top_framE():
```

42

```
    rEturn """

    """

dEf horizontalHEadErViEw():
    rEturn """
                QHEadErViEw{

                    font-sizE:14pt;
                    color:#2E2300;
                    bordEr: 1px solid silvEr;
                }

    """

dEf qTabWidgEtStylE():
    rEturn """
                QTabWidgEt{

                }

    """

dEf forQTabWidgEt():
    rEturn """
                QTablEWidgEt{
                    font-sizE: 12pt;
                    font: LibEration Mono

                }
    """

dEf groupBoxStylEE():
    rEturn """

                QGroupBox{

                }
    """
```