Державний торговельно-економічний університет

Кафедра комп’ютерних наук та інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
на тему:

«Проектування і програмана розробка автоматизованої системи управління запасами в торгівлі»

Студента 2 курсу, 4м групи
спеціальності
122 «Комп’ютерні науки» _____ *підпис студента* Постоленко Олександр Олександрович

Науковий керівник
Кандидат технічних наук, доцент, професор _____ *підпис керівника* Паращак Олексій Миколайович

Гарант освітньої програми
доктор фізико-математичних наук, професор _____ *підпис керівника* Пурський Олег Іванович

Київ 2023

Державний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем
Спеціальність 122 «Комп'ютерні науки»
Освітня програма «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
«9» грудня 2022р.

Завдання
на випускну кваліфікаційну роботу студенту

(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи
«Проектування і розробка автоматизованої системи управління запасами в торгівлі»
Затверджена наказом ректора від *«06» грудня 2022 р. № 3284*

2. Строк здачі студентом закінченої роботи 0*2 грудня 2023 року*

3. Цільова установка та вихідні дані до роботи
Мета роботи: розробка автоматизованої системи управління запасами в торгівлі .
Об'єкт дослідження: процеси системи управління запасами в торгівлі.
Предмет дослідження: вимоги , методи та технології покращення автоматизації системи управління запасами в торгівлі.

4. Перелік графічного матеріалу_____
_____
_____
_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Паращак О.М.	09.12.2022 р.	09.12.2022 р.
2	Паращак О.М.	06.05.2023 р.	14.06.2023 р.
3	Паращак О.М.	04.09.2023 р.	09.11.2023 р.

6. Зміст випускного кваліфікаційної роботи (перелік питань за кожним розділом)
ЗМІСТ
ВСТУП .....

7. Календарний план виконання роботи

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*01.11.2022*	*01.11.2022*
2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*	*09.12.2022*	*09.12.2022*
3	*Вступ*	*01.05.2023*	*01.05.2023*
4	*РОЗДІЛ 1. Аналіз предметної області*	*14.06.2023*	*14.06.2023*
5	*Підготовка статті у збірник наукових статей магістрів*	*20.06.2023*	*20.06.2023*
6	*РОЗДІЛ 2. Проектування системи управління в торгівлі*	*08.09.2023*	*08.09.2023*
7	*РОЗДІЛ 3. Програмна реалізація системи управління в торгівлі*	*20.10.2023*	*20.10.2023*
8	*РОЗДІЛ 4. Тестування та дослідна експлуатація*	*25.10.2023*	*25.10.2023*
9	*Висновки*	*02.11.2023*	*02.11.2023*
10	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	*22.11.2023*	*22.11.2023*
11	*Попередній захист випускної*	*29.11.2023*	*29.11.2023*

	кваліфікаційної роботи		
12	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*	*04.12.2023*	*04.12.2023*
13	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	*06.12.2023*	*06.12.2023*
14	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «9» грудня 2022 р.

9. Керівник випускного кваліфікаційного проекту Паращак О.М.
(прізвище, ініціали, підпис)

10. Гарант освітньої програми Паращак О.М.
(прізвище, ініціали, підпис)

11. Завдання прийняв до виконання студент Постоленко О.О.
(прізвище, ініціали, підпис)

12. Відгук керівника випускної кваліфікаційної роботи

____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____
____

Керівник випускної кваліфікаційної роботи

____
(підпис, дата)

13. Висновок про випускну кваліфікаційну роботу

Випускна кваліфікаційна робота студента __________________
(прізвище, ініціали)

може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми____________________ Пурський О.І.____
(підпис, прізвище, ініціали)

Завідувач кафедри ___________________________ Пурський О.І.____
(підпис, прізвище, ініціали)

«_____»________________2023 р.

Анотація

У випускній кваліфікаційній роботі була здійснена розробка та реалізація автоматизованої системи управління запасами в торгівлі з метою поліпшення ефективності процесів управління та оптимізації витрат. Було визначено вимоги до автоматизованої системи управління запасами та розроблено перелік функціональних вимог, які повинна задовольняти система, враховуючи особливості торгівельного бізнесу та потреби користувачів. Створено автоматизовану програму для підприємства з підтримкою бази данних , зручним та одночасно простим інтерфейсом .

Anotation

In the final qualification work, the development and implementation of an automated inventory management system in trade was carried out in order to improve the efficiency of management processes and optimise costs. The requirements for an automated inventory management system were determined and a list of functional requirements was developed that the system must meet, taking into accou In the final qualification work, the development and implementation of an automated inventory management system in trade was carried out in order to improve the efficiency of management processes and optimise costs. The requirements for an automated inventory management system were determined and a list of functional requirements was developed that the system must meet, taking into account the specifics of the trading business and the needs of users. An automated application for the enterprise with database support, user-friendly and at the same time simple interface was created.

ЗМІСТ

Вступ

Управління запасами є частиною підприємницької діяльності будь- якого підприємства на товарному ринку і відрізняється тим, що не охоплює сам процес надання послуги.

Ефективне управління матеріальними запасами дозволяє знизити тривалість виробничого і операційного циклу, зменшити поточні затрати на їх зберігання, вивільнити з поточного господарського обороту частину фінансових коштів, реінвестуючи їх в інші активи. Підприємствам, незважаючи на численні відхилення в постачальницько–збутової діяльності, необхідно дотримуватися певної системи управління запасами, щоб уникнути хаотичності і невизначеності в забезпеченні процесу забезпечення необхідними торгівельними ресурсами. Для цього підприємствам необхідна певна методика проектування системи управління запасами.

У цілому, торгівельна діяльність включає закупівельну та збутову види діяльності. Керівництво підприємства прагне придбати ресурси чи надати послуги у відповідності зі своїми власними інтересами. Завдання, які ставить перед підприємством ринок, зводиться до необхідності створити якісний товар і вигідно його реалізувати.

Проблема раціонального управління запасами ще більше ускладнюється, коли розглядаються запасні частини для виробництва. Це є додатковими запасами, які, хоча можуть зменшити фінансові результати через заморожування коштів, забезпечують стійкість та ліквідність підприємства.

Сучасні умови вимагають від підприємств застосування сучасних обчислювальних систем та програмних продуктів для підвищення якості та ефективності управління постачанням.. На жаль, більшість наявних програм не надають повного спектру функцій для розрахунку оптимальних запасів та управлінням їх в торгівлі, що суттєво обмежує їхню ефективність.

Тому, на сучасному етапі, актуальність дослідження та розробки систем управління запасами, які враховують особливості управління запасами в торгівлі, стає критично важливою.

1. РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ПІДТРИМКИ ПРОЦЕСІВ УПРАВЛІННЯ ЗАПАСАМИ В ТОРГІВЛІ

1.1. Коротка характеристика об'єкту дослідження

ПП «Давос» працює на ринку України з травня 2007 року. Весь цей час підприємство розвивалося і продовжує розвиватися, перетворюючись на стабільну і ефективну організацію, здатну конкурувати на українському ринку в області продажу будівельних матеріалів, орієнтованої на вимоги клієнтів та високу якість продукції.

За весь час роботи торговельне підприємство ПП «Давос» зарекомендувало себе як надійний партнер, стабільне у фінансовому відношенні підприємство.

Підприємство почало свою діяльність у лютому 2007 року. 12 лютого 2007 року отримала свідоцтво про постановку на облік в податковій службі з присвоєнням ідентифікаційного номера платника податків. Свою офіційну комерційну діяльність підприємство розпочало з травня 2007 року з занесення до єдиного державного реєстру юридичних осіб відповідної діяльності. Підприємство утворено як оптово-роздрібна торгова мережа.

Підприємство зареєстровано за адресую: м. Харків, вул. Новгородська, 87. Основною метою діяльності ПП «Давос», згідно статуту, є отримання прибутку, задоволення потреб покупців, розширення в регіоні, створення нових робочих місць, скорочення безробіття, розвиток соціальної структури міста та області.

Основним завданням діяльності є задоволення потреби населення в товарах, продукції, роботах, послугах. Основним предметом діяльності ПП «Давос» є торгівля будівельними матеріалами.

Додатковими видами діяльності компанії є:

неспеціалізована оптова торгівля;

діяльність посередників у торгівлі будівельними матеріалами.

ПП «Давос», крім перерахованого вище, може займатися окремими видами діяльності, перелік яких визначається законом, за умови надання спеціального дозволу (ліцензії) на організацію даного виду діяльності.

Організаційно-правова форма підприємства – приватне підприємство. ПП «Давос» утворено повністю як приватне підприємство. Підприємство є

юридичною особою, має самостійний баланс, печатку, штампи, бланки зі своїм найменуванням, зареєстрований у встановленому порядку товарний знак.

Підприємство міцно закріплює позиції на ринку і має великі перспективи розвитку. Це пояснюється тим, що ПП «Давос» – одне з небагатьох підприємств, що закуповує товар безпосередньо від виробників. Підприємство реалізує дуже широкий асортимент товарів вищої якості за цінами, які цікаві як роздрібним так і дрібно гуртовим покупцям. До того ж, ПП «Давос» має добре розвинену систему логістики, що дозволяє виробляти доставку товару "від 1 коробки" вчасно і з мінімальними кількісними і якісними втратами.

Керівництво ПП «Давос» в роботі підприємства поєднує врахування побажань споживачів, оперативне і гарантоване виконання їх вимог, постійне підвищення кваліфікації та професіоналізму своїх працівників з метою підвищення обсягів продажу продукції, отримання максимального прибутку та підвищення іміджу підприємства на вітчизняному ринку.

Функціонування підприємства можна охарактеризувати таким чином:

у наявності є дозволи та погодження від місцевої адміністрації, енергетиків, органів пожежної охорони, податкової інспекції та санепідемстанції;

раціональне місце розташування підприємства (було досліджено різні варіанти під'їзних шляхів, їх ступінь завантаженості, можливість їх взаємної заміни у разі необхідності);

широкий асортимент продукції, щебінь різних фракцій, цегла червона та біла, лаки, внутрішні та зовнішні фарби, декоративні штукатурки типу «короїд», «шуба» (внутрішні, зовнішні), сухі будівельні суміші (цемент, пісок, крейда, вапно, цементні шпаклівки та штукатурки, гіпсові шпаклівки та штукатурки, само вирівнювальна наливна підлога, вогнетривкі суміші, гідроізоляційні суміші, будівельні грунти).

У ПП «Давос» розроблена та використовується власна політика стосовно ефективної співпраці зі споживачами (табл. 1.1).

Таблиця 1.1

Напрямки політики ефективної співпраці ПП «Давос» зі споживачами

Напрямки політики	Сутність напрямку
I	Споживач – це найважливіша людина в бізнесі
II	Споживач залежить не від підприємства, а підприємство залежить від нього
III	Споживач не відволікає працівників підприємства від роботи, а є їх метою
IV	Споживач – це не та людина, з якою можна сперечатися
V	Споживач – це людина, яка звернулася до підприємства, тому що потребує певних товарів чи послуг. Завдання працівників надати їх йому з вигодою для нього і для підприємства
VI	Споживач – це не об'єкт статистики, а людина з певними емоціями та вподобаннями
VII	Споживач – це людина, без якої не буде ніякого бізнесу

Використовуючи такі гасла у своїй роботі, працівники підприємства усвідомлюють, що кожний споживач є важливим, тому докладають максимум зусиль для того, щоб задовольнити його потреби, незалежно від того, чи приходить він особисто, дзвонить або пише.

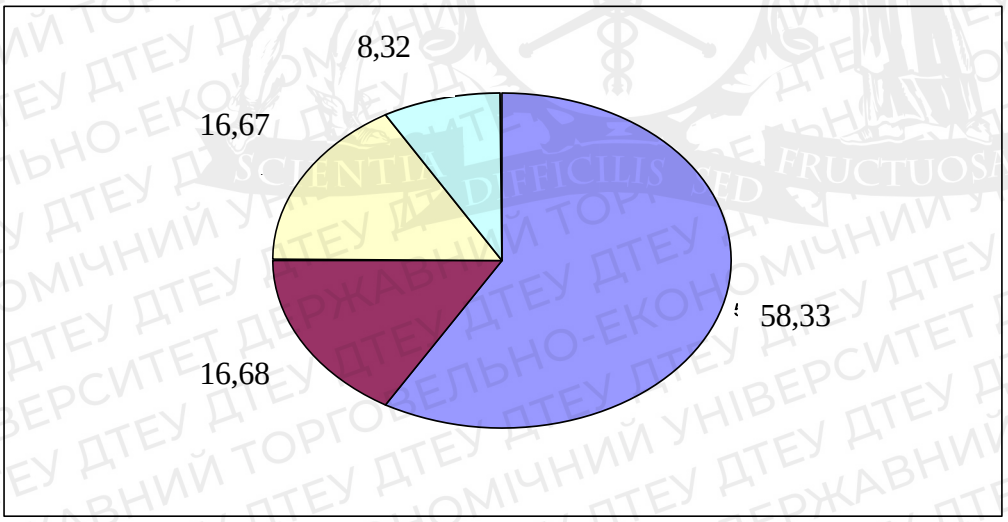
На вітчизняному ринку підприємство має позитивний імідж та довіру споживачів. У цілому, постійними споживачами є чисельні будівельні підприємства, архітектори, дизайнери, виконроби підприємств.

Характеристика основних постачальників продукції для ПП «Давос» представлена у табл. 1.2. Дані свідчать, що основні постачальники продукції підприємства розміщені у м. Харків, що є дуже зручним для підприємства. Постачання здійснюється, здебільшого, двічі на місяць. Частота постачання продукції для ПП «Давос» у відсотковому відношенні представлена на рис. 1.1. Дані свідчать, що у 58,33 % випадків постачання продукції здійснюється двічі на місяць; у 16,68 % – постачання продукції здійснюється раз на тиждень; у 16,67 % – постачання продукції здійснюється раз на квартал; у 8,32 % – постачання продукції здійснюється раз на місяць.

Таблиця 1.2

Характеристика основних постачальників продукції для ПП «Давос»

Назва підприємства	Місцезнаходження	Частота постачання продукції
1. ПРАТ «Усе для ремонту»	Харків	Двічі на місяць
2. ТОВ «Євроремонт»		Раз на тиждень
3. ТОВ «Інтер'єр»		Двічі на місяць
4. Будівельний супермаркет «Епіцентр»		Двічі на місяць
5. ПРАТ «Чік»		Двічі на місяць
6. Будівельний супермаркет «Будмен»		Раз на квартал
7. ПРАТ «Сам»		Раз на квартал
8. ТОВ «Майстер на усі руки»		Двічі на місяць
9. ТОВ «Еталон»		Раз на місяць
10. ПРАТ «Будкомплект»		Раз на тиждень
11. ПРАТ «Золоті руки»		Двічі на місяць
12. ПРАТ «Чемпіон»		Двічі на місяць



Умовні позначення:

58,33 % – % постачання продукції двічі на місяць;

16,68 % – % постачання продукції раз на тиждень;

16,67 % – % постачання продукції раз на квартал;

8,32 % – % постачання продукції раз на місяць.

Рис. 1.1. Частота постачання продукції для ПП «Давос»

Таким чином, підприємство за роки роботи зайняло гідне місце серед інших підприємств, що реалізують аналогічну продукцію. Основну ставку підприємство робить на якість продукції, тому що велика частина споживчої групи реалізованого товару припадає на будівництво соціальної сфери. Підприємство продовжує освоювати нові ринки і розширювати номенклатуру товарів. Встановлено, що у ПП «Давос» є переваги у ціновій політиці.

1.2. Опис предметної області

Ефективне формування товарних запасів є ключовим аспектом для забезпечення стійкості асортименту товарів, реалізації обґрунтованої цінової політики та підвищення рівня обслуговування покупців на підприємстві. Кожне підприємство повинно прагнути до оптимального рівня запасів для кожної товарної позиції, щоб забезпечити ефективне управління цими ресурсами.

Товарні запаси підприємств торгівлі постійно зазнають руху та оновлюються, з кінцевим етапом у їхньому циклі - споживанням. У сфері оптової торгівлі основною метою товарних запасів є задоволення потреб оптових покупців, тоді як для роздрібних підприємств важливо забезпечити сталу доступність товарів для кінцевих споживачів.

Налагоджена система управління запасами є важливою складовою для підтримки товарних запасів на оптимальному рівні. Оптимальний рівень запасів визначається відсутністю або мінімізацією надлишків та дефіциту.

Управління товарними запасами передбачає реалізацію таких операцій:

Нормування запасів: Розробка економічно обґрунтованих нормативів, таких як обсяг страхового запасу. Ці нормативи враховують різні аспекти, включаючи поточний запас, страховий запас, сезонне зберігання і т. д.

Оперативний облік і контроль: Здійснюється за допомогою спеціального

програмного забезпечення, наприклад, 1С Торгівля і Склад. Залишки товарів на початок і кінець періоду аналізуються і коригуються.

Регулювання: Підтримка запасів на певному рівні та їх зміна в залежності від попиту і періодів поставки. Надлишок і дефіцит товарів негативно впливають на комерційні результати підприємства, тому регулювання є важливою складовою управління запасами.

При розгляді ефективного управління запасами на підприємстві важливо збалансувати дві аспекти - позитивні та негативні сторони наявності запасів. Перша "чаша ваг" містить позитивні аспекти, такі як забезпечення надійності виробничого процесу і задоволення потреб покупців. Друга "чаша" концентрується на негативних аспектах, таких як витрати на утримання та відволікання капіталу від обороту через інвестування в запаси.

При розробці системи управління запасами важливо враховувати стратегічні пріоритети компанії та знаходити компроміс між ризиками та витратами, або ліквідністю та оборотністю.

Процес встановлення системи управління запасами на підприємстві може включати п'ять послідовних етапів:

- Визначення вартості запасів та їх характеристик: Цей етап включає в себе визначення об'ємних, часових та місцезнаходження параметрів запасів.

- ABC-аналіз: Виявлення ключових категорій запасів ("A", "B", "C") для оптимізації управління ними відповідно до їх важливості.

- Реєстрація існуючих методів та процедур: Визначення поточних методів та процедур управління запасами, а також встановлення критеріїв для їхньої оцінки.

- Порівняння існуючих методів з необхідними: Аналіз і налаштування системи інформаційного моніторингу запасів та визначення ефективності існуючої системи.

- Визначення кроків переходу: Розробка нової або удосконалення існуючої системи управління запасами та визначення кроків для впровадження цієї системи.

1.3. Огляд і аналіз існуючих аналогів, що реалізують функції предметної області

Технологія управління процесами на підприємстві ПП «Давос», представляє собою безліч програмних продуктів для управління бізнес процесами, а саме:

microsoft excel;

microsoft outlook;

1С: підприємство 7.1.

Облік деяких товарних позицій проводиться в облікових журналах.

Виходячи з цього визначено, що технології управління запасами потребує кардинальних змін, морально застаріла основна облікова система «1С: Підприємство 7.1.», яка не має змоги модернізації. Деякі програми потребують трудомісткої підтримки та налагодження, а саме microsoft excel та microsoft outlook

В цілому спостерігалась така ситуація на підприємстві:

відсутність цілісної автоматизації процесу управління;

низький рівень аналітичних систем та систем звітності;

недостатній рівень розвитку управління корпоративного рівня;

застаріла основна облікова система.

Проаналізувавши проблеми технології управління запасами, було вирішено удосконалити існуючу технологію шляхом створення комплексної інформаційної системи управління запасами в торгівлі, що дозволить усунути існуючі недоліки, комплексно автоматизувати бізнес-процеси торгівельного підприємства ПП «Давос», побудувати цілісний інформаційний простір, та забезпечити

додаткові вигоди.

Для вирішення поставлених завдань було проаналізовано ринок програмних продуктів та визначено, що для побудови якісної системи управління процесами підприємства слід звернути увагу на програмну продукцію лідерів ринку – «1С», що наведено в табл. 1.3.

Таблиця 1.3

Порівняльна характеристика програм для торгівлі

Задача	1С:Роздріб	1С:Управління торгівлею	1С:Управління невеликою фірмою	1С:Управління торговим підприємством
Ціноутворення, різні групи цін, врахування знижок	+	+	+	+
Роздрібні продажі	+	+	+	+
Гуртові продажі	–	+	+	+
Комісійна торгівля	–	+	–	+
Планування та управління асортиментом	–	–	–	–
Управління торговими представниками	–	+	–	+
Управління складом, запасами, закупками	+	+	+	+
Підтримка дисконтних програм	+	+	–	+
Облік по серійних номерах	–	+	–	+
Підтримка роботи з торговим обладнанням	+	+	+	+
Виробнича діяльність	–	–	+	+
Управління взаємовідносинами з клієнтами	–	+	–	+
Облік майна, основних засобів, нарахування амортизації	–	–	+	+
Банківські операції	–	+	+	+
Облік коштів	+	+	+	+
Кадровий облік працівників, табелі, графіки роботи, нарахування зарплати	–	–	+	+
Управління персоналом, планування роботи, облік особистих продажів продавців	+	–	+	–

Програмні продукти даної компанії мають різноманітні напрямки та функціонал. Виходячи з наведених порівняльних характеристик найбільш доцільно обрати для впровадження, програмний модуль «1С:Управління торговим

підприємством».

Програма «1С:Управління торговим підприємством 8 для України» призначена для автоматизації підприємств торгівельної сфери, а також сфери надання послуг (у т.ч. транспортних) та інших. Можливість доопрацювання стандартного функціоналу програми дозволяє в повній мірі підлаштувати її під бізнес-процеси конкретного підприємства.

«1С:Управління торговим підприємством 8» в одній програмі вирішує завдання:

бухгалтерського та податкового обліку (з підготовкою обов'язкової реглатментованої звітності);

оперативного обліку;

здійснювати управління торговою діяльністю;

управлінського обліку;

розрахунку зарплати та управління персоналом;

виробничої діяльності підприємства.

Основні функціональні можливості:

Управління торговою діяльністю.

Забезпечує контроль та аналіз всіх торгових операцій, що здійснює підприємство – як в цілому, так і по окремих відділеннях (супермаркетах, магазинах, роздрібних точках).

Включає в себе наступні ділянки обліку:

планування обсягів продажів (по підрозділах, групах товарів, окремих категоріях покупців, по підприємству в цілому);

управління замовленнями покупців, ведення клієнтської бази;

формування цін, систем знижок;

планування закупівель на основі аналізу потреб складу в товарах та планів продажів;

управління мережею роздрібних точок;

керування взаєморозрахунками з клієнтами та постачальниками;

підключення широкого асортименту торгового обладнання.

Програма складського обліку, забезпечує повний контроль та управління

запасами підприємства. Дозволяє здійснювати управління залишками ТМЦ (в т.ч. з контролем і обліком серій та термінів придатності), враховувати комплектації та

розукомплектації товарів, здійснювати функції ордерного обліку та резервування ТМЦ тощо.

Розрахувати економічний ефект від впровадження такої системи як «1С:Управління торговим підприємством» заздалегідь досить складно, проте:

з досвіду впровадження подібних проектів підвищення прибутку становить від 10% – 15%;

згідно зі статистикою товариства з управління виробничими запасами APICS, впровадження сучасної ERP-системи може забезпечити віддачу, наведену в табл. 1.4.

Таблиця 1.4

Планове зростання показників діяльності підприємств після впровадження сучасної ERP-системи на основі статистичних даних APICS

Показник	Діапазон значень	Середнє значення
1	2	3
Запаси та виробництво		
Зниження обсягів матеріальних запасів	12–30%	21%
Скорочення витрат на матеріальні ресурси	5–15%	9%
Зниження виробничих витрат	4–20%	8%
Скорочення операційних та адміністративних витрат	5–25%	11%
Збільшення обсягу продукції, що випускається	8–25%	15%
1	2	3
Оборотні кошти		
Зростання оборотності складських запасів	12–30%	19%
Збільшення оборотності грошових коштів	3–5%	4%
Ефективність та оперативність		
Скорочення термінів виконання замовлень	10–80%	30%
Зростання прибутку	7–30%	13%
Трудовитрати і звітність		
Скорочення трудовитрат в різних підрозділах	10–70%	30%
Прискорення отримання управлінської звітності	в 2–5 разів	в 3,3 рази
Прискорення підготовки регламентованої звітності	в 2–4 рази	в 2,8 рази

Відповідно зі словником APICS (American Production and Inventory Control Society), термін «ERP-система» (Enterprise Resource Planning – Управління ресурсами підприємства) може вживатися у двох значеннях.

По-перше, це – інформаційна система для ідентифікації і планування всіх

ресурсів підприємства, які необхідні для здійснення продажів, виробництва, закупівель і обліку в процесі виконання клієнтських замовлень. По-друге (в більш загальному контексті), це – методологія ефективного планування і управління всіма ресурсами підприємства, які необхідні для здійснення продажів, виробництва, закупівель і обліку при виконанні замовлень клієнтів у сферах виробництва, дистрибуції і надання послуг.

1.4. Специфікація вимог до системи.

Автоматизована система управління запасами для підприємства призначена для автоматизації процесу своєчасного і повного задоволення потреб в матеріалах, запасах, інструментах і т.д. при мінімальних витратах на їх доставку та зберігання на основі розкладів.

Підсистема управління запасними матеріалами, комплектуючими, складальними одиницями призначена для:

• На основі даних портфеля і статистики по засобах під час виробництва виробів, деталей, заготовок розраховувати необхідну кількість запасних виробів, деталей, заготовок;

Таблиця 1.5

Специфікація функціональних вимог

Ідентифікатор вимоги	Назва вимоги	Атрибути вимоги		
		Пріоритет	Складність	Контакт
1	Отримати заявку	рекомендоване	середня	Менеджери
2	Управління замовленнями	обов’язкове	висока	Менеджери
3	Перегляд замовлень	обов’язкове	висока	Менеджери
4	Вести статистику	обов’язкове	висока	Менеджери
5	Оперативний облік запасів	обов’язкове	висока	Менеджери
6	Ведення каталогів	обов’язкове	висока	Адміністратор

7	Реєстрація користувача	рекомендоване	висока	Адміністратор
8	Редагування інформації про користувача	обов’язкове	середня	Адміністратор
9	Видалення користувача	обов’язкове	середня	Адміністратор
10	Перелік Клієнтів	рекомендоване	висока	Менеджери
11	Перелік Складів	обов’язкове	висока	Менеджери

Специфікацію нефункціональних вимог наведено в таблиці 1.5.

Наведемо специфікацію суттєвих для проекту нефункціональних вимог:

1. Застосовність:

- мінімальний час для навчання звичайних і досвідчених користувачів;
- відповідність стандартам графічного інтерфейсу.

2. Надійність:

- постійна безвідмовна робота;
- пропускна здатність каналу зв’язку 100 Mb/s;
- забезпечення можливості віддаленого доступу до комп’ютера, на якому буде встановлена система;
- доступність – 5%.

3. Робочі характеристики:

- швидкість завантаження інтернет-ресурсу: 0,1 – 1 с;
- число операцій : 100 / 1 с;
- використання ресурсів: від 1 Gb, в залежності від кількості студентів.

4. Проектні обмеження:

- Операційна система Microsoft Windows 7/8/10/11;
- Microsoft SQL Server Management Studio;
-Visual Studio;

5. Вимоги до документації

- наявність інтерактивної довідки.

6. Інтерфейси:

-інтерфейс користувача – C# - інтерфейс.

Таблиця 1.6

Специфікація нефункціональних вимог

Ідентифікатор вимоги	Назва вимоги	Атрибути вимог		
		Пріоритет	Складність	Контакт
Застосовність				
1.1	Час, необхідний для навчання звичайних і досвідчених користувачів	Рекомендована	Низька	Адміністратор
1.2	Основні вимоги застосовності нової системи відносно інших систем, які знають користувачі	Опційна	Низька	Адміністратор
1.3	Вимоги по відповідальності стандартам графічного інтерфейсу користувача	Рекомендована	Низька	Адміністратор
Надійність				
2.1	Доступність	Обов’язкова	Середня	Адміністратор
2.2	Середній час безвідмовної роботи	Рекомендована	Середня	Адміністратор
2.3	Точність	Обов’язкова	Середня	Адміністратор
Робочі характеристики				
3.1	Використання ресурсів	Рекомендована	Середня	Адміністратор
3.2	Вимоги до технології програмування	Рекомендована	Середня	Адміністратор

1.5. **Постановка задачі**

Завданням формування та управління запасами є безперебійне задоволення потреб клієнтів, виконання планів продажів. Однак великі запаси вимагають значних фінансових ресурсів, які можуть бути ефективніше використані в інших сферах діяльності. Через це управління запасами повинно бути постійним процесом. Справне управління дозволяє вчасно виконувати зобов'язання та направляти вивільнені кошти у дохідні сфери, підвищуючи окупність інвестицій у запаси без втрати якості обслуговування. Оскільки попит на продукцію змінюється, а доступ до даних про потреби у кожній одиниці номенклатури обмежений, важко точно передбачити необхідність в товарах. Це часто призводить до надмірного "роздування" запасів, заморожування коштів і потреби в складних процедурах інвентаризації та управління запасами.

Використання автоматизованої системи управління запасами та бізнес-аналітики може вирішити ці проблеми, забезпечуючи регулярне збирання та обробку інформації. Оновлене управління запасами дозволяє:

- оптимізувати витрати, закуповуючи менше товарів "на випадок";
- підвищити рівень обслуговування шляхом забезпечення кращої доступності товарів;
- ефективніше використовувати активи компанії;
- гнучко реагувати на сезонні зміни попиту та потреби в конкретних продуктах;
- уникати накопичення дорогих запасів;
- зменшити суми списань;
- точно коригувати кількість товарів на складі при зміні асортименту або контрагентів;
- уникнути відсутності необхідного товару на складі.

Висновки за розділом 1

Було визначено що основні техніко-економічні показники діяльності ПП «Давос» зросли, що достатньо позитивно характеризує діяльність підприємства. Але баланс підприємства за два аналізовані роки жодного разу не був абсолютно ліквідним, оскільки визначені співвідношення не були дотримані. Щодо звітного періоду слід відзначити, що в даній моделі має місце нерівність А1 < П1, що свідчить про неплатоспроможність ПП «Давос» на момент складання балансу. Тобто на підприємстві недостатньо коштів для покриття найбільш термінових зобов'язань і найбільш ліквідних активів. Підприємству слід провести заходи, спрямовані на збільшення високоліквідних активів.

На основі проведеного вище аналізу товарних запасів можна констатувати той факт, що на ПП «Давос» мають місце зайві товарні запаси.

Головною причиною зайвих товарних запасів є неефективне управління товарними запасами, яке проявляється у відсутності обґрунтованого товарного нормативу запасів.

Технологію управління запасами, було вирішено удосконалити шляхом створення комплексної інформаційної системи управління, що дозволить усунути існуючі недоліки, комплексно автоматизувати бізнес-процеси торгівельного підприємства ПП «Давос», побудувати цілісний інформаційний простір, та забезпечити додаткові вигоди

2. Проектування архітектури системи та вибір необхідних апаратних та програмних компонентів.

2.1 Розроблення архітектури системи

Комп'ютерні мережі широко розповсюджені завдяки ідеї ефективного використання ресурсів. Висока пропускна здатність локальних мереж забезпечує швидкий доступ з одного вузла комп'ютерної мережі до ресурсів, що розташовані в інших вузлах. Для правильної роботи

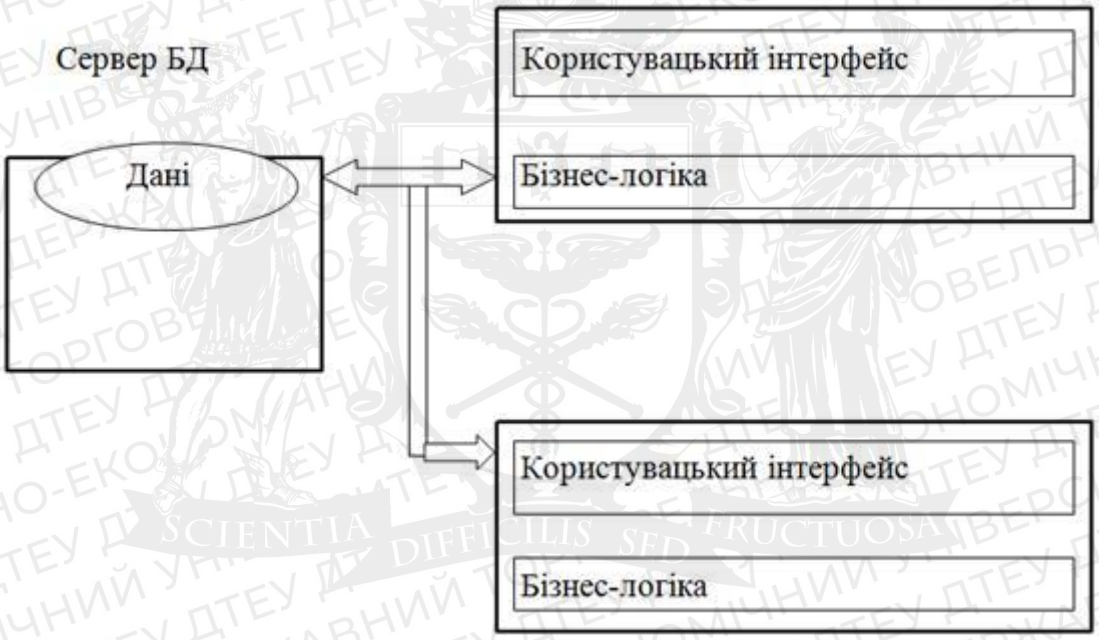


Рис. 2.1 «Трирівнева модель архітектури клієнт/сервер»

Сервер комп'ютерної мережі повинен мати ресурси, що відповідають його функціональному призначенню та потребам мережі. У зв'язку з архітектурою "клієнт-сервер" систем баз даних, ця концепція важлива через її спрощене та відносно економічне вирішення колективного доступу до баз даних в локальній мережі. Сервер баз даних фактично є Системою Управління Базами Даних (СУБД), яка підтримує всі необхідні функції, такі як визначення даних, обробка даних, захист і цілісність даних та інше. Сервер додатків включає різноманітні додатки, які виконуються "поверх" СУБД, включаючи додатки, розроблені користувачами, та вбудовані додатки. Елементами трирівневої архітектури є:

• Розподілена база даних, яка включає в себе таблиці локальних баз даних

на одному вузлі;

• Програми доступу до даних та частина прикладних програм на іншому вузлі (можливо, на сервері додатків);

• Клієнтські програми на клієнтських вузлах (можливо, тільки зовнішній інтерфейс). Сервер розподіленої бази даних (Distributed DataBase Server) - це операційна система, яка вирішує такі завдання:

• Управління іменами у розподіленому середовищі (глобальний словник даних);

• Оптимізація розподілених запитів;

• Управління розподіленими транзакціями.

Доступ до бази даних від прикладних програм або користувача здійснюється через звернення до клієнтської частини системи, а основним інтерфейсом між клієнтською і серверною частинами є мова SQL для баз даних. Структура системи управління запасами реалізована на основі багаторівневої архітектури, яку можна побачити на рисунку 2.2. На сучасному підприємстві використовується багато інформаційних систем для автоматизації бізнес-процесів. Для забезпечення роботи програми потрібна інтеграція з системою авторизації, де створюються користувачі. Розроблена система має функції збору різних видів інформації, і при наявності інших робочих систем може здійснювати імпорт даних у свою систему.

2.2 Проектування структури бази даних

В якості моделі даних для проектованої системи була обрана реляційна модель. Виходячи з обраної моделі даних, була спроектована за допомогою CASE-засобів Erwin схема логічної моделі даних (діаграма ERD – модель сутність-зв'язок). В результаті проведених досліджень предметної області було виявлено, що для роботи системи управління запасами на рівні підприємства необхідна наявність наступних сутностей, представлених в таблицях 2.1-2.8.

Таблиця 2.1

Сутність «Авторизація» та її атрибути

Атрибут	Опис
Логін	Логін користувача
Пароль	Пароль користувача

Таблиця 2.2

Сутність «Products» та її атрибути

Атрибут	Опис
ID	ID Унікальний код товару
Name	Назва Товару
Description	Помітка
ImagePath	Завантаження фото
BasePrice	Ціна
Quantity	Кількість
DiscountProcent	Знижка на товар в %
Category	Категорія
Location	Локація
Supplied	Поставщик
CreatedAT	Створення Товару
UpdatedAt	Редагування

Таблиця 2.3

Сутність «Locations» та її атрибути

Атрибут	Опис
ID	ID Унікальний код
Name	Назва
Description	Примітка
ImagePath	Завантаження Фото
CreatedAT	Створити
UpdateAT	Редагування
Addres	Адресса

Таблиця 2.4

Сутність «Orders» та її атрибути

Атрибут	Опис
ID	ID Унікальний код
CustomerId	ID Унікальний код Клієнта

LocationId	ID Унікальний код Локації складу
SellerId	ID Унікальний код Продавця
CreatedAt	Створення
UpdatedAt	Редагування

Таблиця 2.5

Сутність «Customers» та її атрибути

Атрибут	Опис
Id	ID Унікальний код
Name	Ім’я
PhoneNumber	Номер телефону
ImagePath	Звантаження фото
CreatedAt	Створити
UpdatedAt	Редагувати

Таблиця 2.6

Сутність «OrderProducts» та її атрибути

Атрибут	Опис
OrderProductId	ID Унікальний код Замовлення Товару
OrderId	ID Унікальний код Замовлення
ProductId	ID Унікальний код Товару
Quantity	Кількість
TotalPrice	Ціна

Таблиця 2.7

Сутність «Sellers» та її атрибути

Атрибут	Опис
ID	ID Унікальний код
Name	Ім’я
Description	Примітка
ImagePath	Завантаження Фото
Username	Призвіще
PasswordHash	Пароль Користувача
Salt	Нумерація
CreatedAt	Створення
UpdatedAt	Редагування

Таблиця 2.8

Сутність «Categories» та її атрибути

Атрибут	Опис
ID	ID Унікальний код
Name	Назва
Description	Примітка
ImagePath	Завантаження Фото
CreatedAt	Створення
UpdatedAt	Редагування

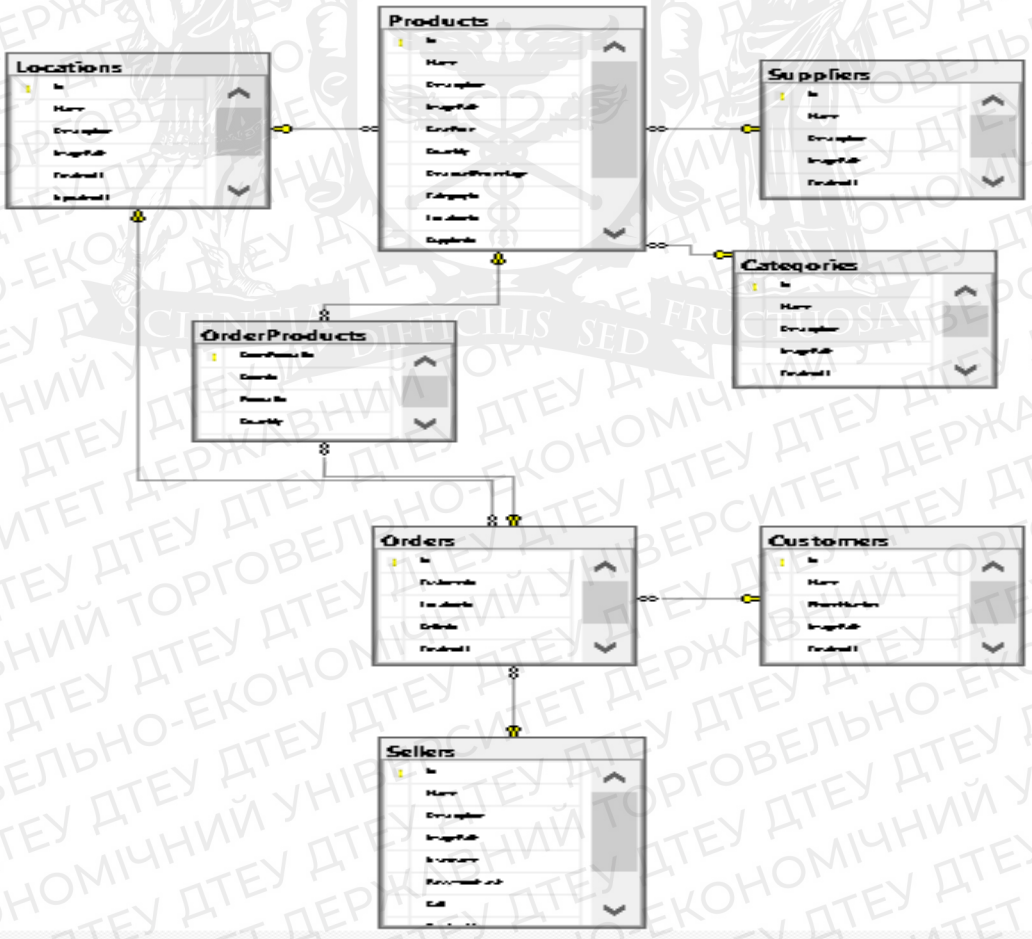


Рис. 2.2. Логічна структура бази даних

Висновки до розділу 2:

У даному розділі було проведено детальний аналіз та розроблено архітектуру програмного додатку, що спрямована на краще розуміння функцій його ключових компонентів. Основним результатом цього розділу є створення та опис структурної схеми, в якій визначені ключові компоненти: рівень клієнта , користувача та рівень даних.

Детально розглянута функціональна структура системи, включаючи основні елементи та модулі обробки даних. Це надає важливий інсайт у взаємодію різних частин системи та ролі кожного модулю у виконанні конкретних завдань.

Окрема увага приділена аналізу основних елементів бази даних та встановленню зв’язків між ними. Визначено ключові елементи, що складають структуру бази даних, та спроектовано її загальну структуру.

Отже, результати цього розділу становлять фундаментальний крок у розробці програмного додатку, надаючи чітку структуру та напрямок для подальших етапів розробки та впровадження.

3 РОЗДІЛ ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ УПРАВЛІННЯ ЗАПАСАМИ НА ПІДПРИЄМСТВІ

3.1 Програмна реалізація системи

Для реалізації системи управління запасами вибрано мову програмування C# при використанні програмного забезпечення Visual Studio 19. C# є об'єктно-орієнтованою мовою програмування, призначеною для розробки різноманітних застосунків на платформі Microsoft .NET. Однією з основних переваг C# є його інтеграція з .NET Framework, що забезпечує широкі можливості для розробки ефективних та масштабованих додатків.

C# також є компілюючою мовою програмування, і процес компіляції подібний до того, який було описано для мови Java. Код програми перетворюється у проміжний байт-код, який в подальшому виконується на віртуальній машині .NET (Common Language Runtime - CLR). Це дозволяє запускати програми на різних платформах, якщо на них встановлено відповідне середовище виконання .NET.

Мова C# ґрунтується на принципах об'єктно-орієнтованого програмування, і кожен її клас складається з властивостей та методів. Властивості визначають характеристики об'єкта, а методи - процедури та функції для роботи з об'єктом та його властивостями.

Система управління запасами буде побудована на основі структурної схеми, де ключові компоненти будуть рівень клієнта, користувача та рівень даних. Описана функціональна структура системи та її модулі обробки даних, а також спроектовано структуру бази даних для ефективного управління запасами на підприємстві.

При розробці прототипу системи управління запасами на підприємстві були спроектовані і реалізовані програмні класи, представлені на рисунку3.1.

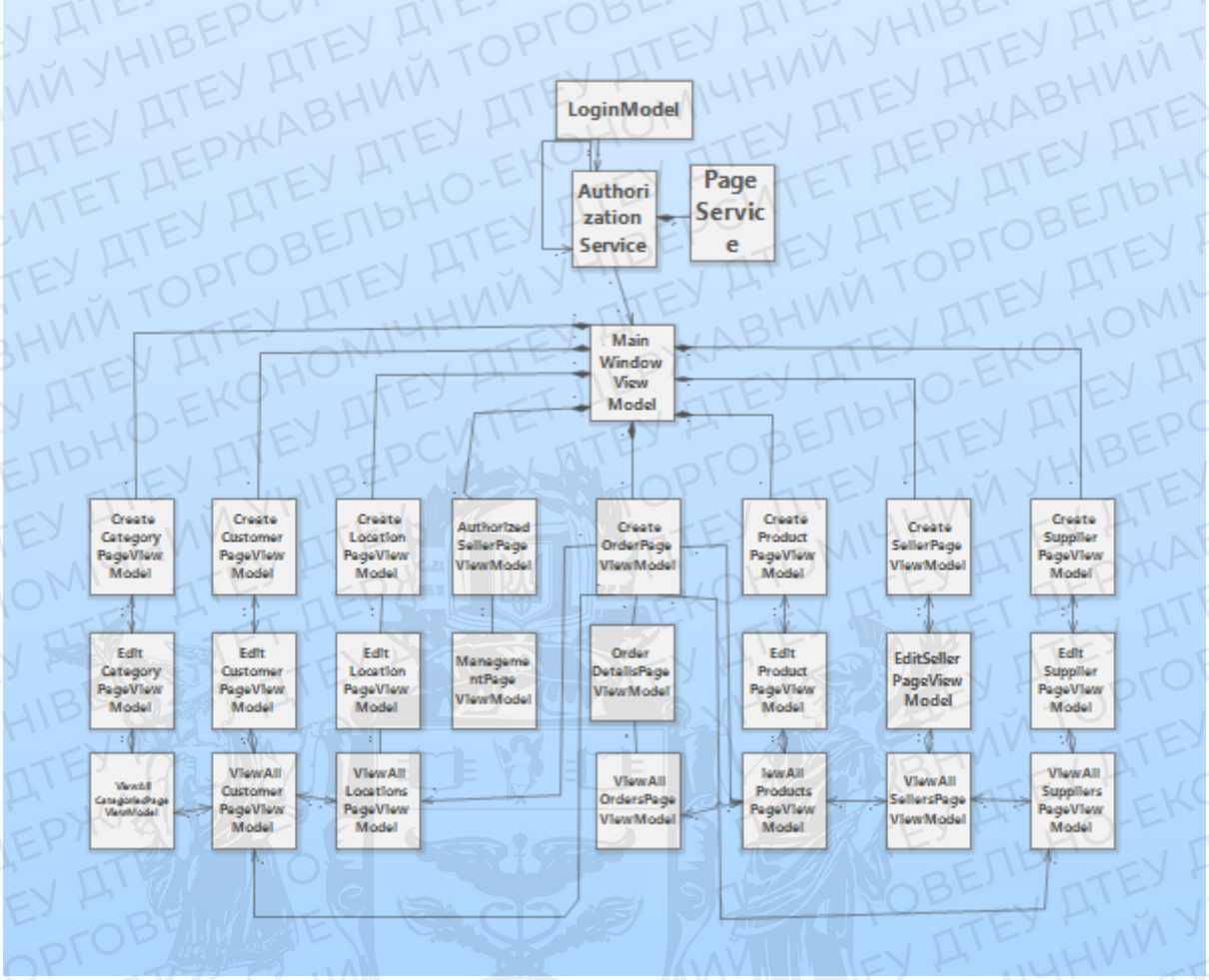


Рис. 3.1. Діаграма класів системи

3.2. Програмна реалізація бази даних

Фізична модель розробляється для СУБД - Microsoft SQL Server Management Studio. Розробка фізичної моделі даних грунтувалася на побудованої логічної моделі даних. Для розробки фізичної моделі даних використовувалося програмне забезпечення Microsoft SQL Server Management Studio.На рисунку 3.2 зображено шлях підключення бази данних під назвою “ShopManagerDB”.

```
services.AddSingleton<IConfiguration>(_ => new ConfigurationBuilder()
    .AddJsonFile("Configuration/appsettings.json", false, true)
    .Build());

// Adding the database context as a scoped service
services.AddDbContextFactory<ApplicationDbContext>(ob =>
{
    ob.UseSqlServer(
        "Server=(localdb)\\MSSQLLocalDB;Database=ShopManagerDb;Trusted_Connection=True;TrustServerCertificate=True;");
});
```

Рис. 3.2 Підключення Бази данних

Далі наводиться опис таблиць, полів, індексів таблиць і інших параметрів бази даних ShopManagerDB:

Имя столбца	Тип данных	Разрешить ...
Id	uniqueidentifier	☐
Name	nvarchar(100)	☐
Description	nvarchar(500)	☑
ImagePath	nvarchar(MAX)	☑
BasePrice	decimal(18, 2)	☐
Quantity	int	☐
DiscountPercentage	decimal(18, 2)	☐
CategoryId	uniqueidentifier	☐
LocationId	uniqueidentifier	☐
SupplierId	uniqueidentifier	☐
CreatedAt	datetime2(7)	☐
UpdatedAt	datetime2(7)	☑
		☐

Рис. 3.4 Таблиця Products

DESKTOP-GF1VIL8\L...b - dbo.Locations | DESKTOP-GF1VIL8\...dbo

Имя столбца	Тип данных	Разрешить ...
Id	uniqueidentifier	☐
Name	nvarchar(100)	☐
Description	nvarchar(500)	☑
ImagePath	nvarchar(MAX)	☑
CreatedAt	datetime2(7)	☐
UpdatedAt	datetime2(7)	☑
Address	nvarchar(100)	☐
		☐

Рис. 3.5 Таблиця Locations

Имя столбца	Тип данных	Разрешить ...
Id	uniqueidentifier	☐
Name	nvarchar(100)	☐
Description	nvarchar(500)	☑
ImagePath	nvarchar(MAX)	☑
CreatedAt	datetime2(7)	☐
UpdatedAt	datetime2(7)	☑
		☐

Рис. 3.6 Таблиця Suppliers

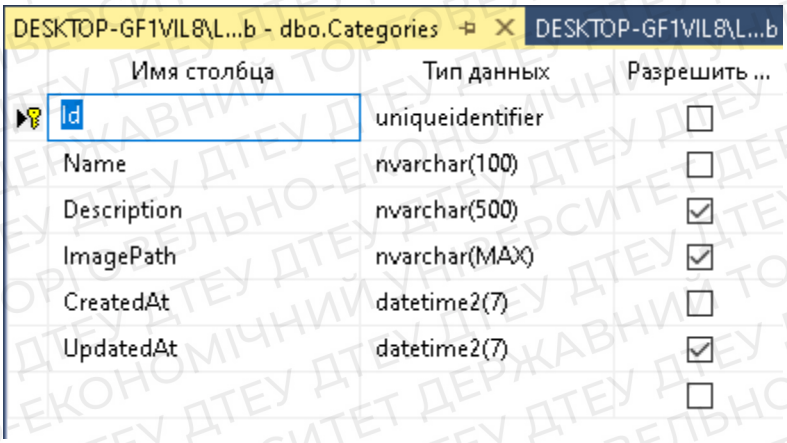


Рис. 3.7 Таблиця Categories

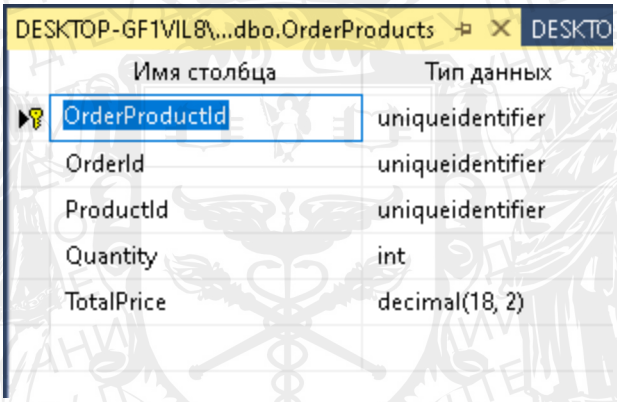


Рис. 3.8 Таблиця OrrderProducts

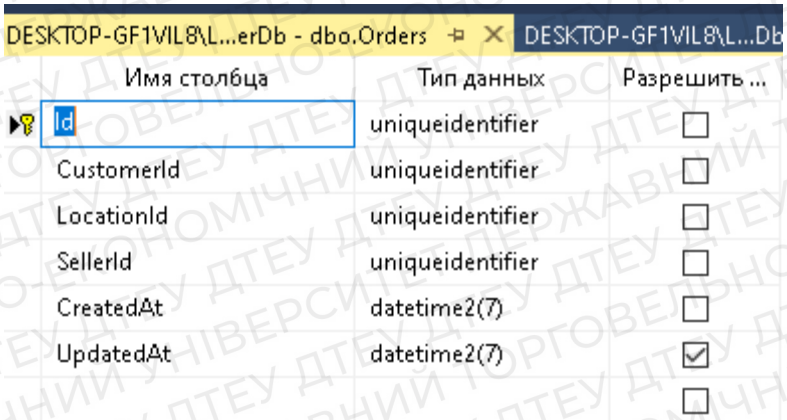


Рис. 3.9 Таблиця Orders

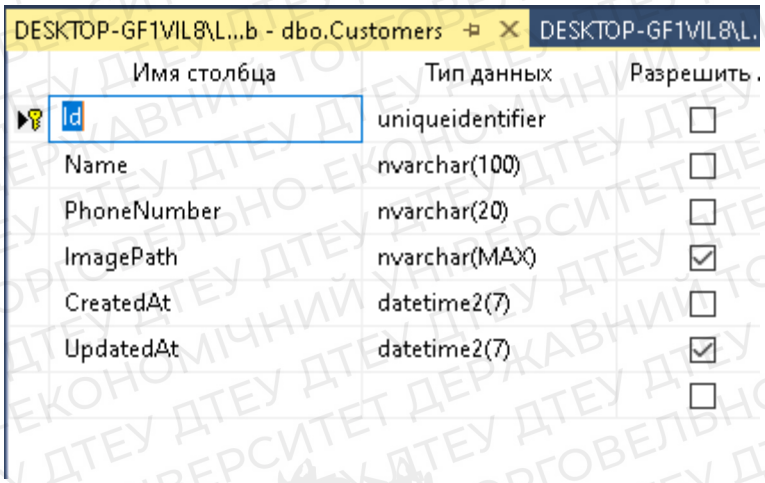


Рис. 3.10 Таблиця Customers

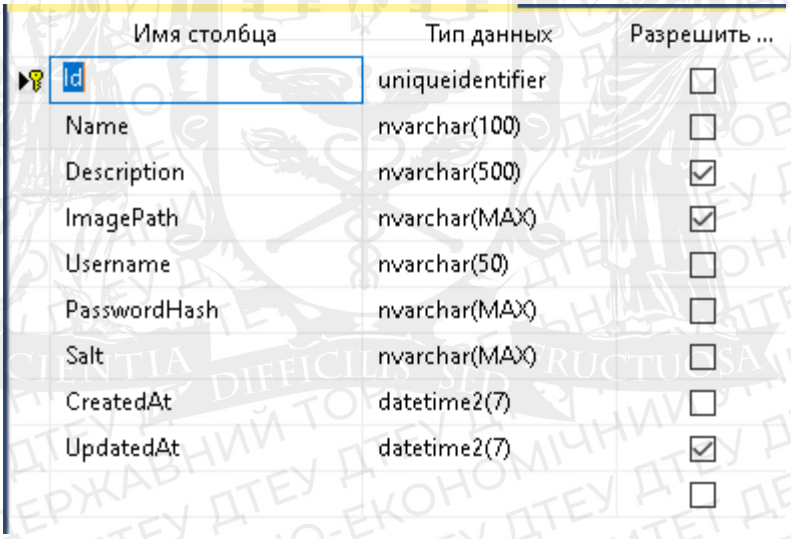


Рис. 3.11 Таблиця Sellers

Висновки до розділу 3:

У даному розділі було вивчено та обґрунтовано вибір технології та мови програмування для подальшої розробки системи управління запасами. Зазначено, що вибір технології та мови є стратегічним рішенням, враховуючи особливості завдання та потреби підприємства.

Мова програмування C# та технологія .NET Framework були обрані з урахуванням їхньої ефективності, об'єктно-орієнтованого підходу, інтеграції з .NET Framework, а також зручності розробки та сумісності з різними операційними системами.

Також було розглянуто засоби розробки бази даних, і вибір був зроблений на користь механізму збережених процедур. Це дозволяє оптимізувати взаємодію з базою даних та забезпечити високий рівень безпеки та ефективності операцій.

Далі, була проведена реалізація бази даних для програмної системи з використанням зазначених засобів. Цей етап є ключовим у розробці, оскільки належна реалізація бази даних забезпечить ефективне та надійне функціонування системи управління запасами.

Отже, результати цього розділу демонструють важливий крок у створенні програмної системи, підкреслюючи обґрунтованість вибору технології, мови програмування та ефективність реалізації бази даних.

РОЗДІЛ 4 ТЕСТУВАННЯ ТА ДОСЛІДНА ЕКСПЛУАТАЦІЯ

4.1 Тестування

В процесі реалізації системи управління запасами в торгівлі було проведено процедуру тестування .

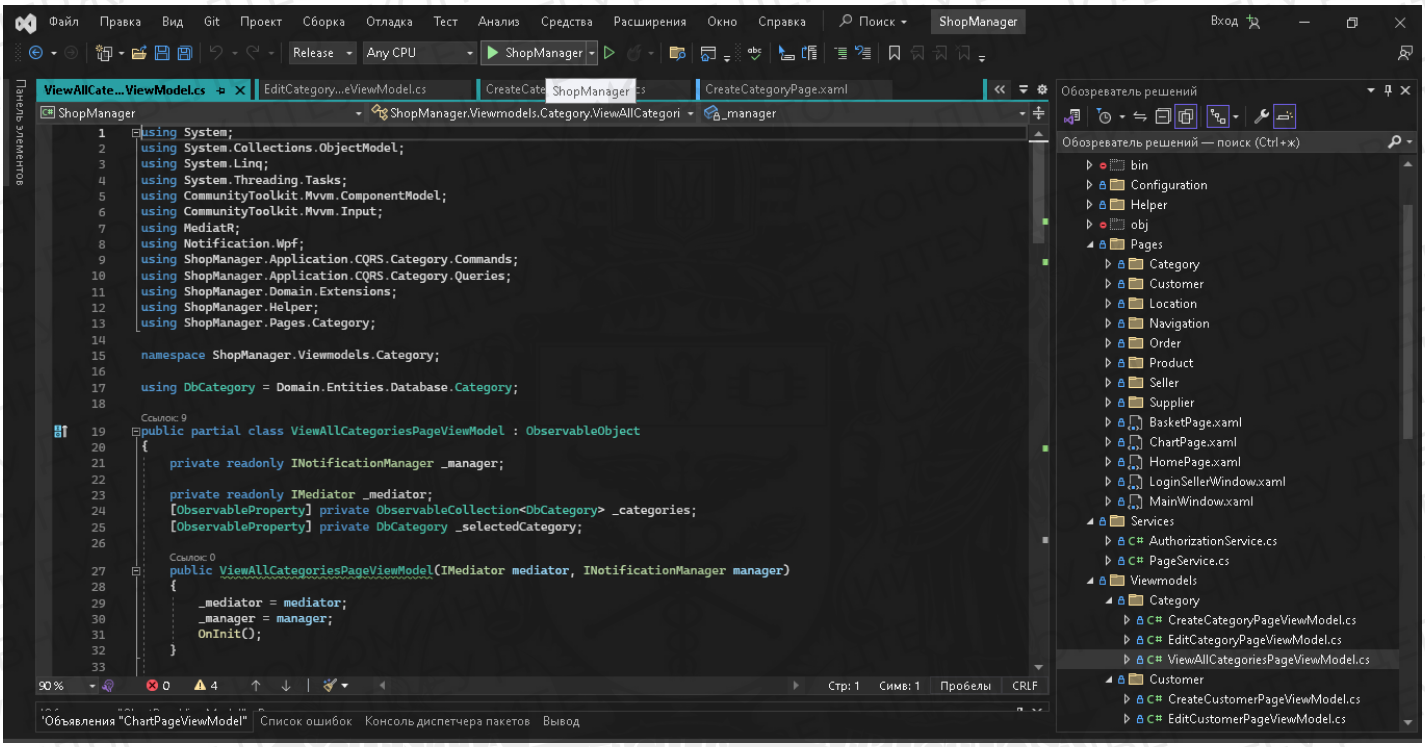


Рис. 4.1 Запуск системи в ПЗ Visual Studio

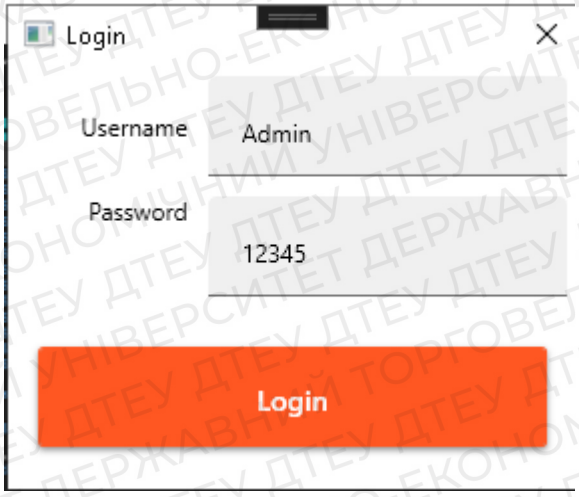


Рис. 4.2 Авторизація в систему

На рисунку 4.2 зображена сторінка Авторизаіїї в якій потрібно заповнити такі данні , як Username користувача та його Password , після підтвердити нажаттям на кнопку Login.

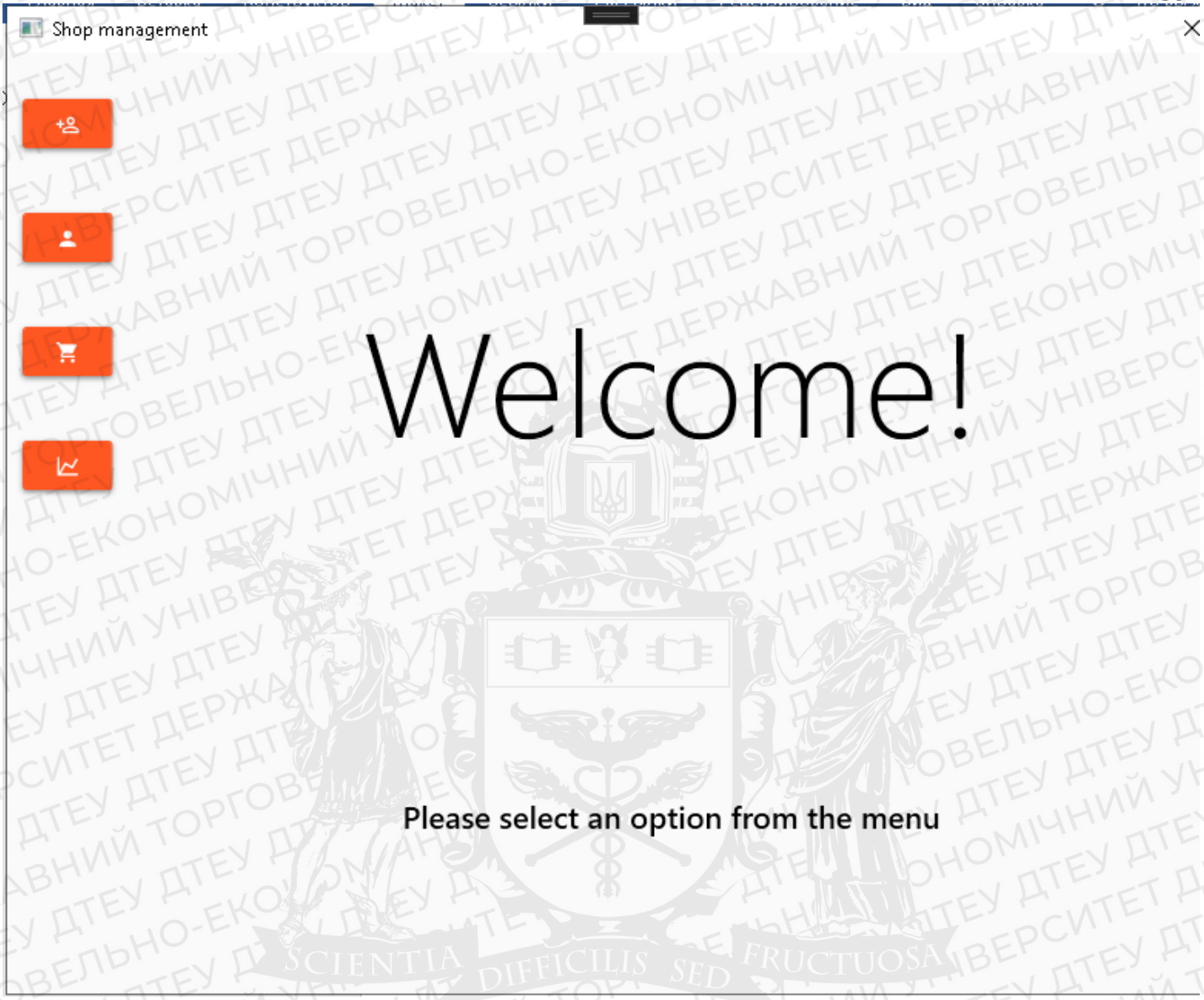


Рис. 4.3 Головне Меню

Після підтвердження сторінки авторизації відкривається головне меню. В головному меню нам доступно на вибір 4 категорії зліва:

1. Сторінка функціоналу для редагування та перегляду таких категорій як : Category, Products, Location , Sellers ,Suppliers , Customers .
2. Сторінка користувача , який пройшов аутентифікацію в систему.
3. Перегляд Orders.
4. Діаграма.

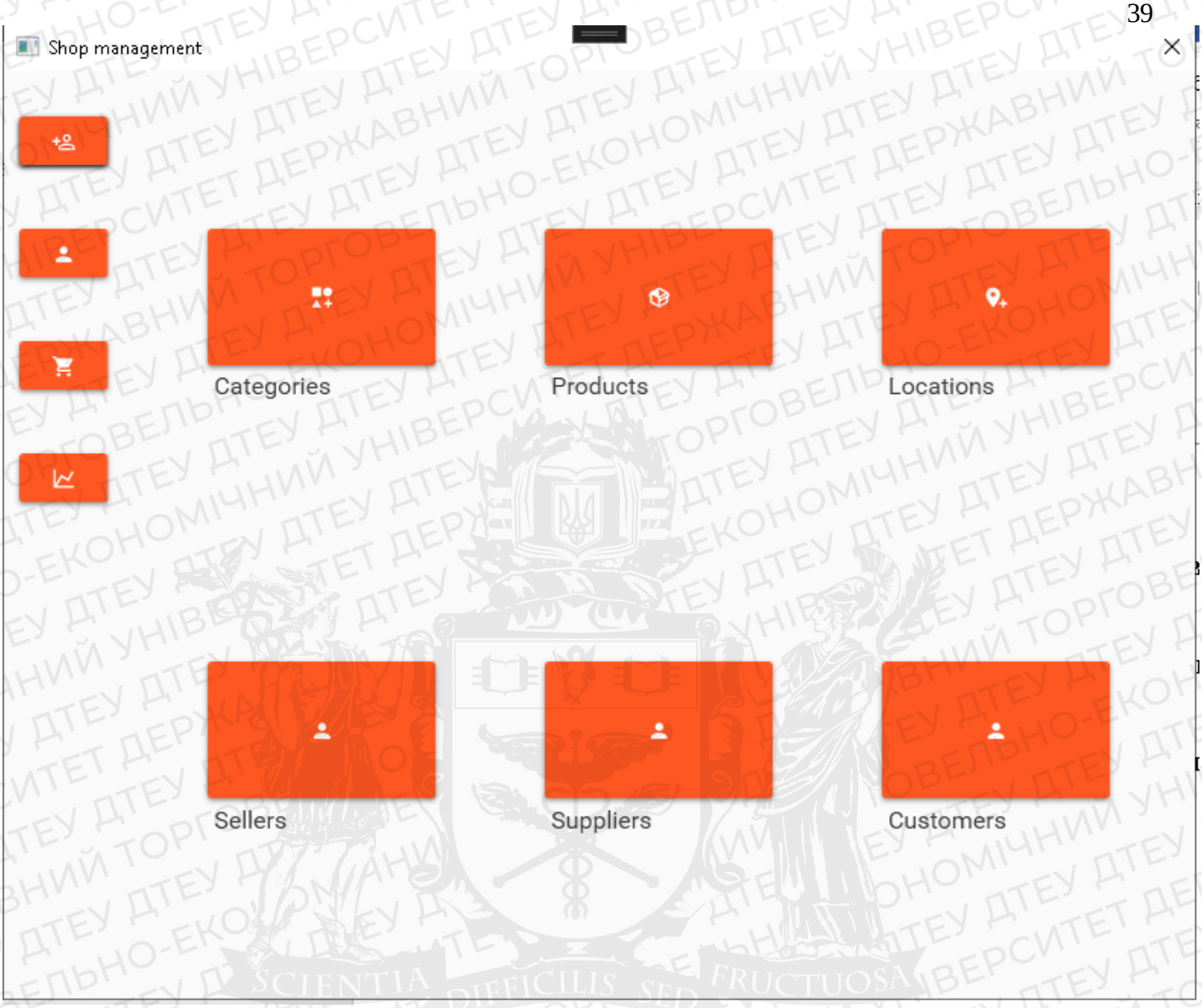


Рис. 4.4 Сторінка функціоналу

На рисунку 4.4 показана Сторінка функціоналу , на данній сторінці ми можемо обрати такі пункти як : Category, Products, Location , Sellers ,Suppliers , Customers.

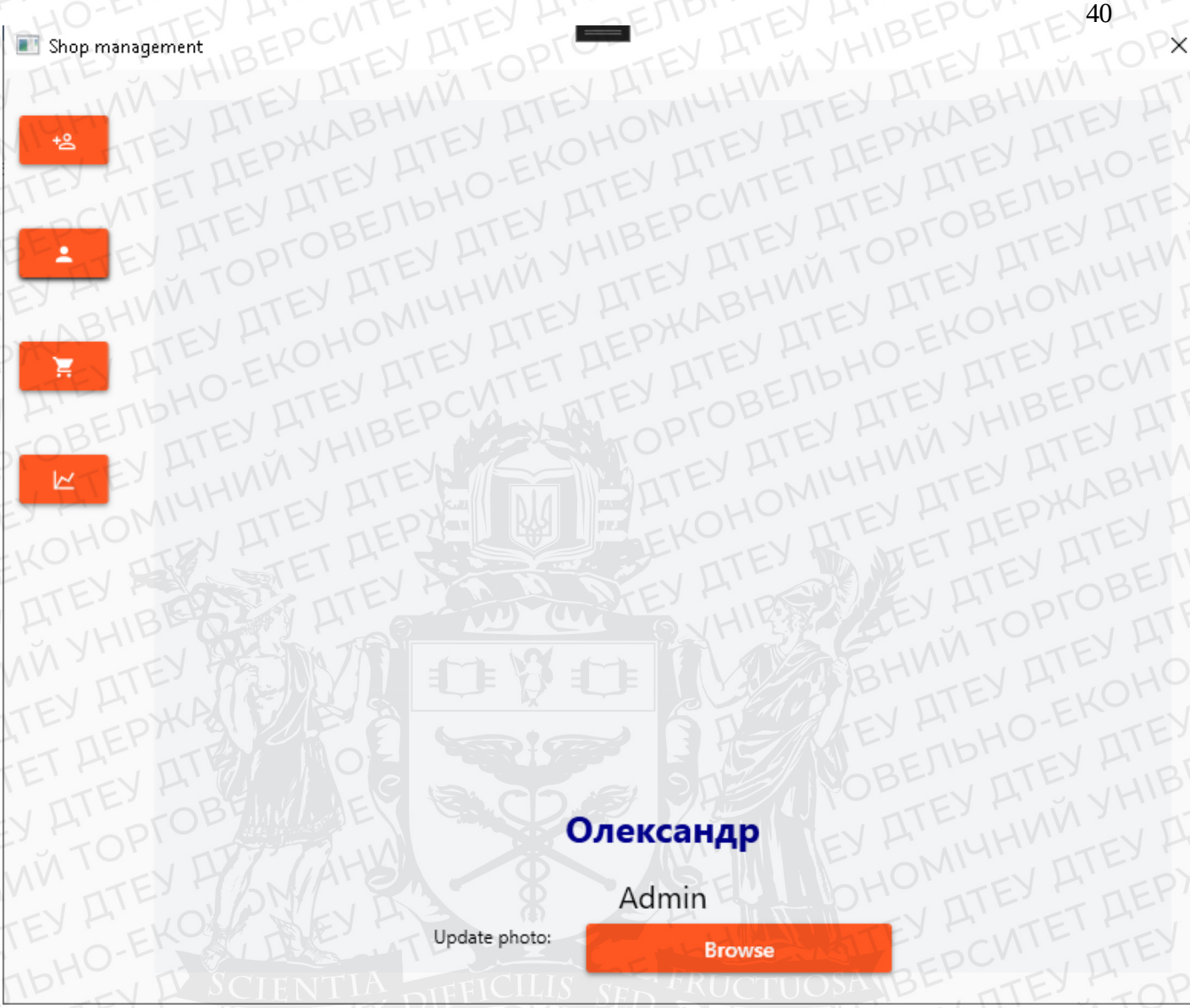


Рис. 4.5 Сторінка Коритувача

На рисунку 4.5 зображена Сторінка Користувача , а саме Name , Description, ми маємо можливість додати або змінити фото користувача натиснувши на кнопку Browse , після нам відкриється доступ до файлів на пристрої щоб обрати фотографію формату png , jpg, JFIF. Данну функцію ми можемо побачити на рисунку 4.6 .

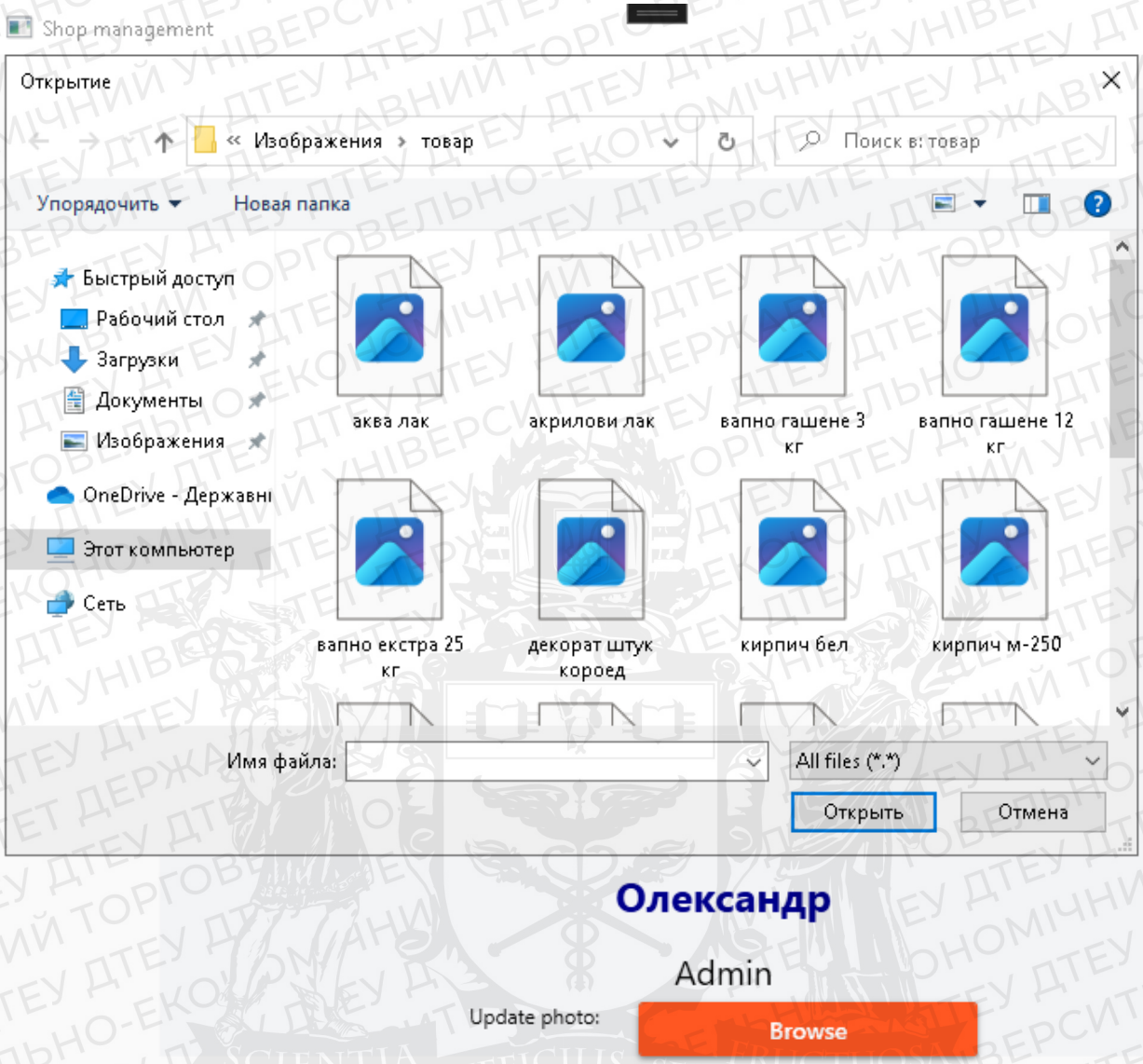


Рис. 4.6 Функція додавання фото Користувача

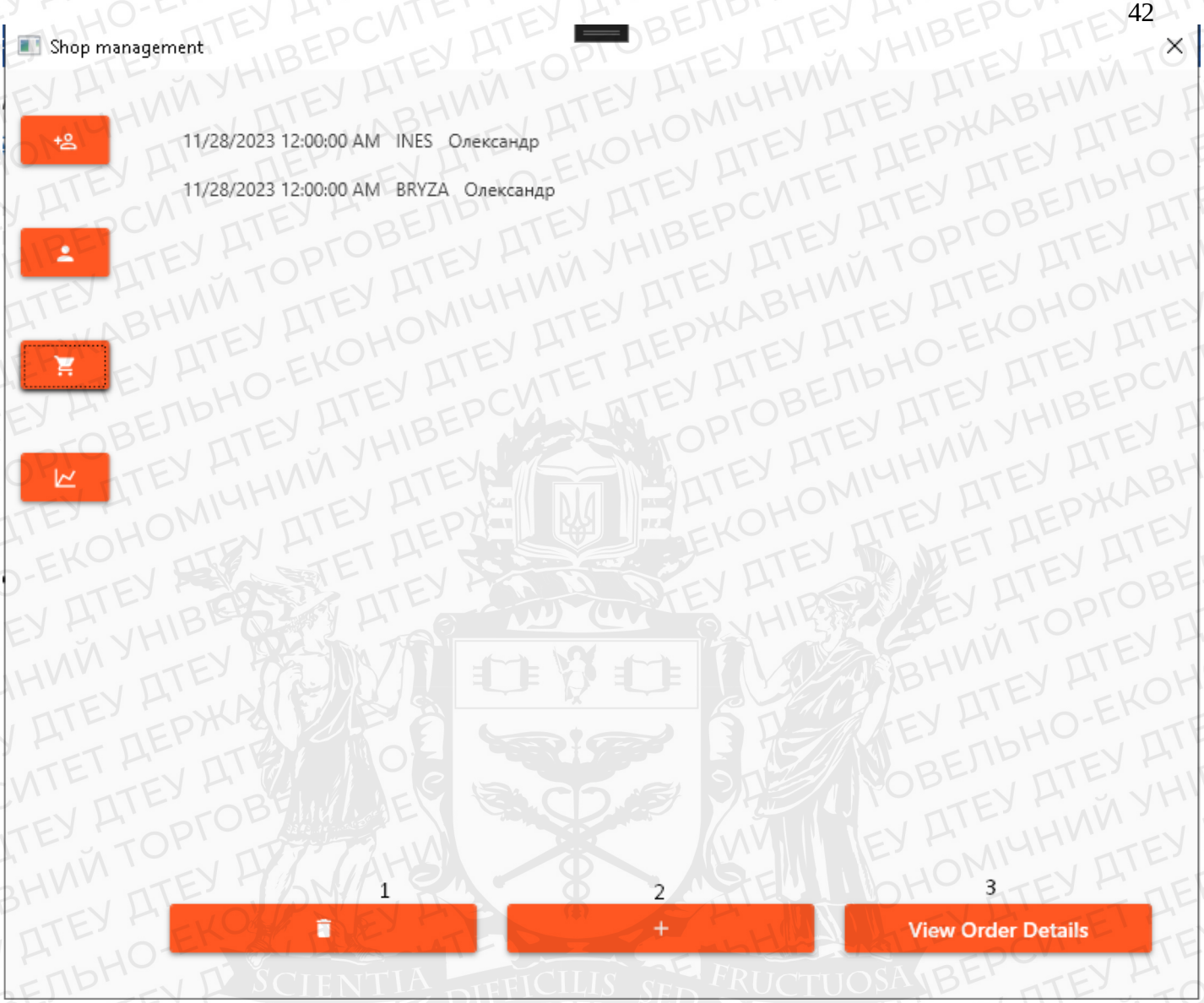


Рис 4.7 Сторінка Замовлення

На данній сторінці ми можемо побачити актуальні замовлення . дату їх створення , назва Складу для відправки . та ім’я менеджера , який створював замовлення . На вибір дано 3 кнопки :

1. Видалити замовлення
2. Створити замовлення
3. Деталі замовлення

Натиснувши кнопку 1 обране замовлення видалиться зі списку , натиснувши кнопку 2 нам відкриється вікно рисунок 4.8 на якому зображена сторінка для створення замовлення , натиснувши кнопку 3 ми побачимо всю інформацію обраного замовлення рисунок 4.9 .

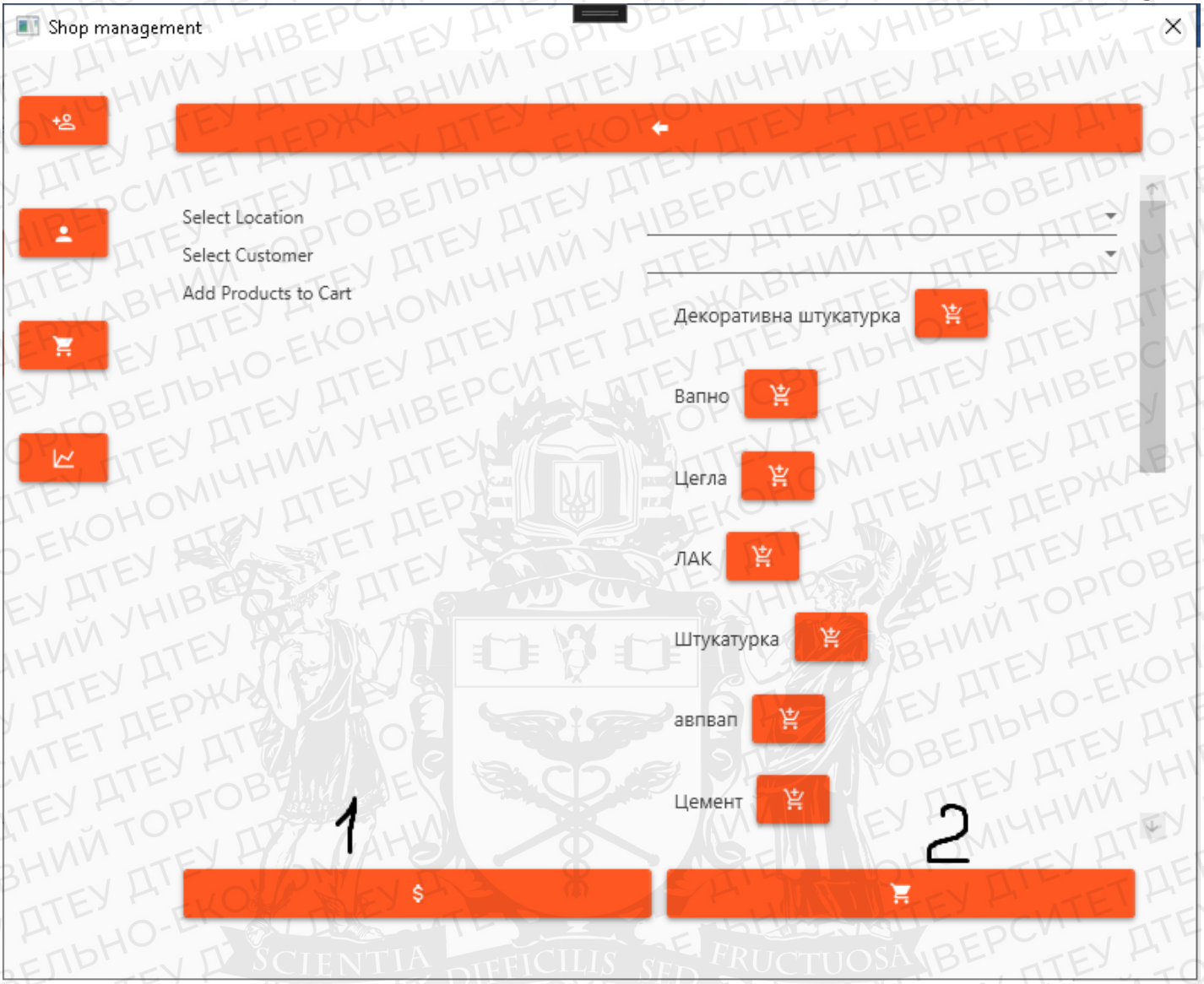


Рис. 4.8 Сторінка для створення замовлення

На данній сторінці створення замовлення нам доступно такі пункти як:

1. Select location
2. Select Customer
3. Add Products to Cart

Натиснувши на пункт Select location , на вибір буде дано список раніше заповнених Локацій рисунок 4.9. Натиснувши на пункт Select Customer на вибір буде дано список раніше заповнених Покупців рисунок 4.10 .З правої частини сторінки ми можемо обрати товар для додавання його в замовлення , при натисканні на товар ми побачимо повідомлення про успішне додавання його до замовлення рисунок 4.11 . Натиснувши на кнопку під номером 1 на рисунку 4.8 , ми побачимо повідомлення про згоду підтвердження замовлення рисунок 4.12 , для підтвердження потрібно натиснути на кнопку OK , після підтвердження

з’явиться повідомлення про успішне створення замовлення рисунок 4.13 . Натиснувши на кнопку 2 рисунок 4.8 , нам відкриється сторінка для перегляду доданих товарів до замовлення рисунок 4.14

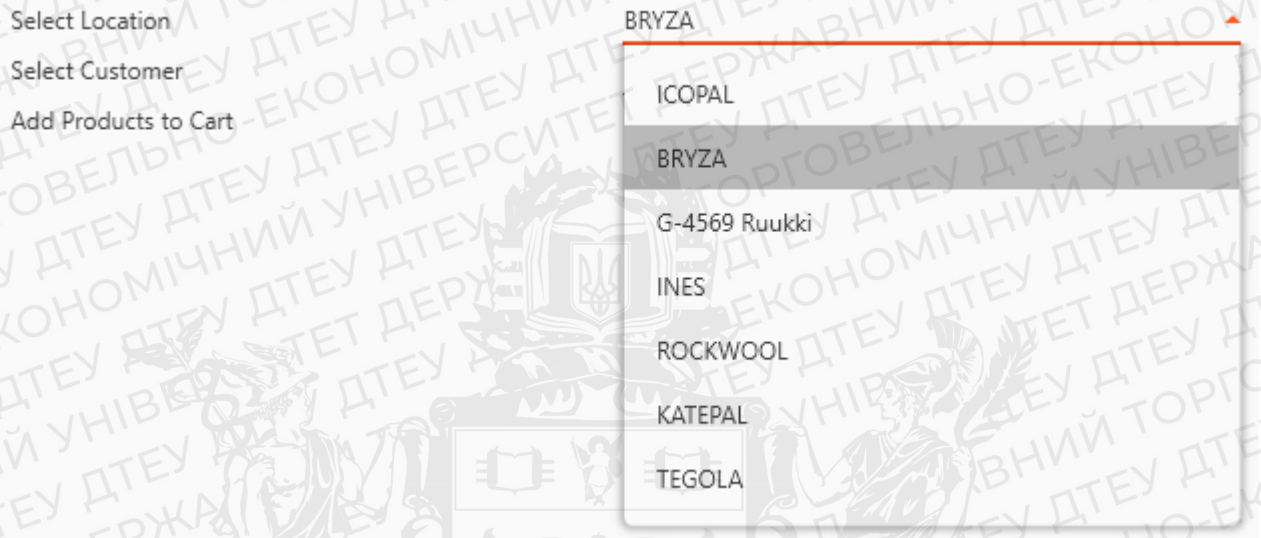


Рис 4.9 Список раніше заповнених Локацій

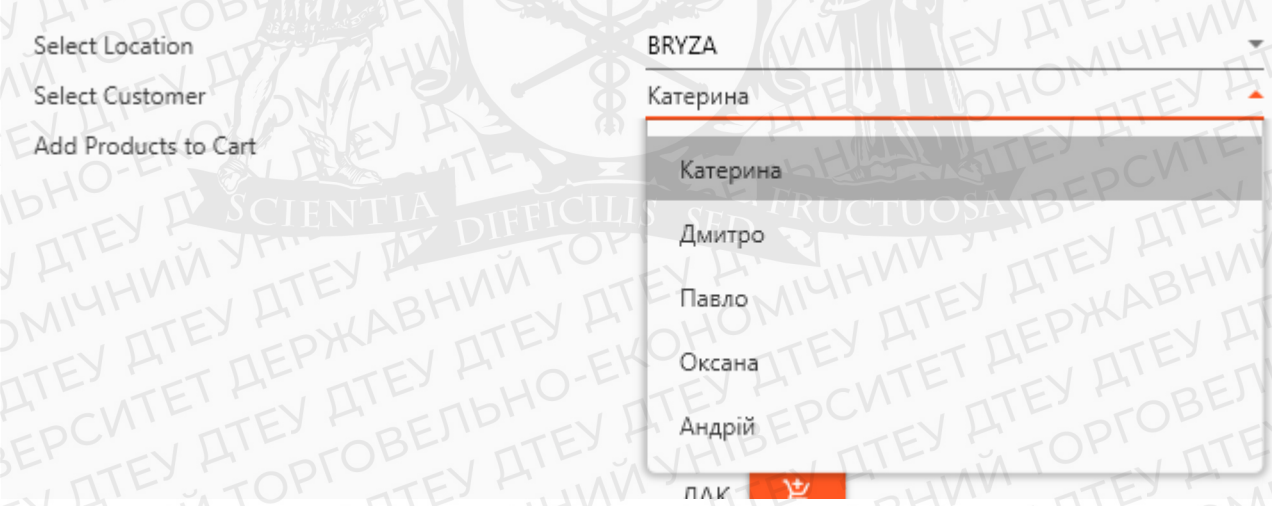


Рис 4.10 Список раніше заповнених Покупців

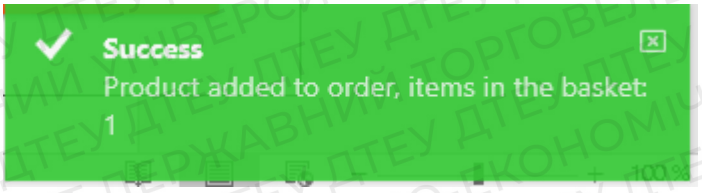


Рис 4.11 Повідомлення про успішне додавання товару до замовлення

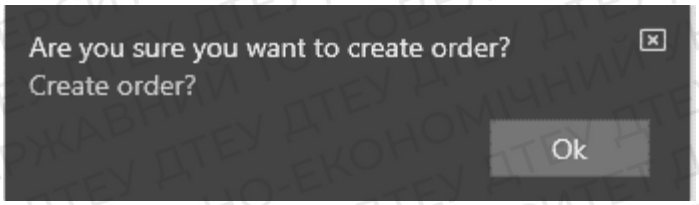


Рис 4.12 Повідомлення про згоду підтвердження замовлення

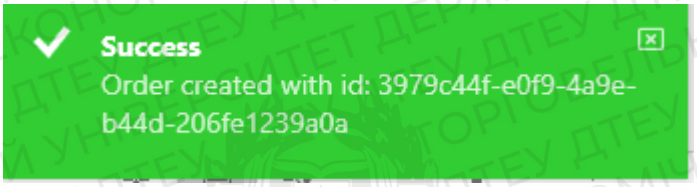


Рис 4.13 Повідомлення про успішне створення замовлення

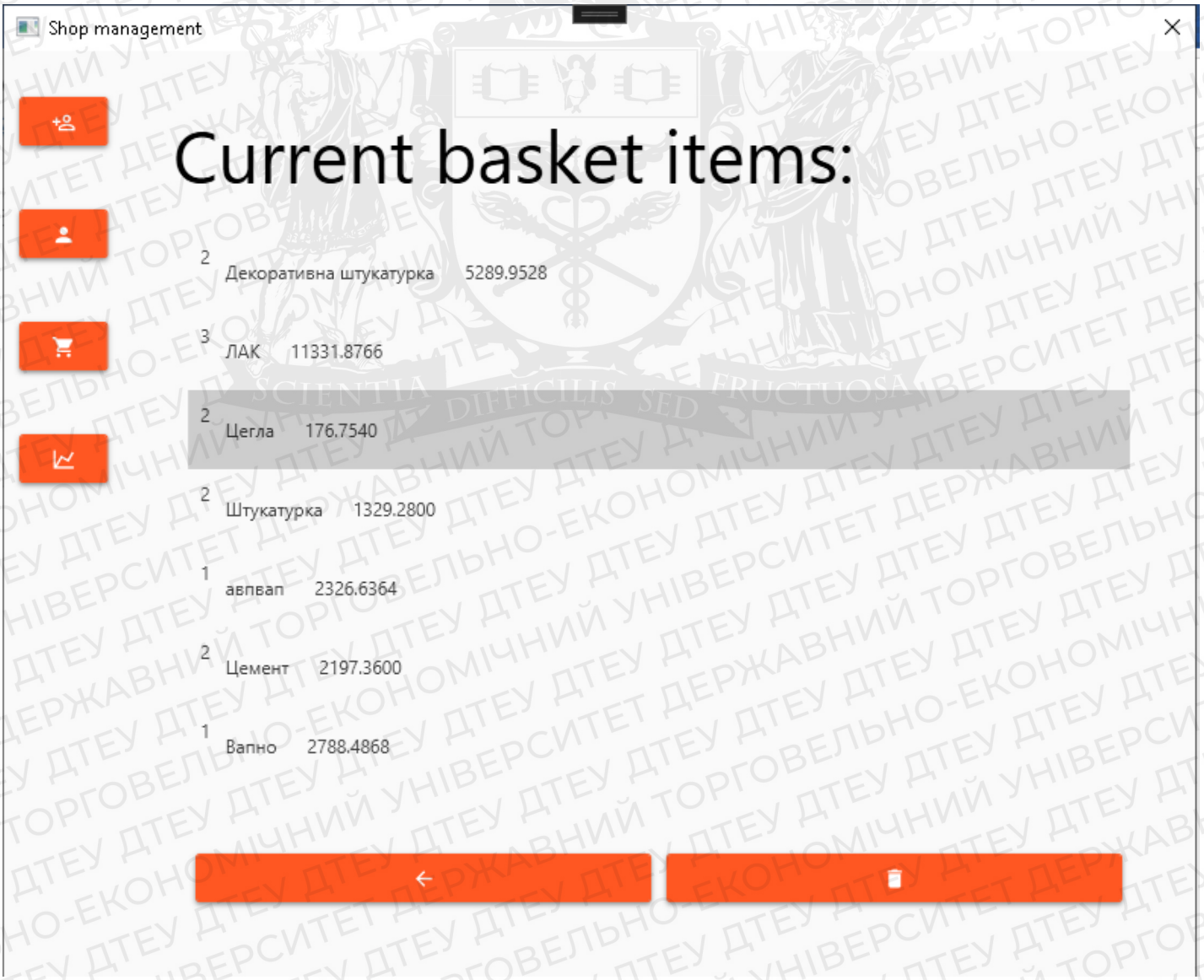


Рис. 4.14 Сторінка для перегляду обраних товарів

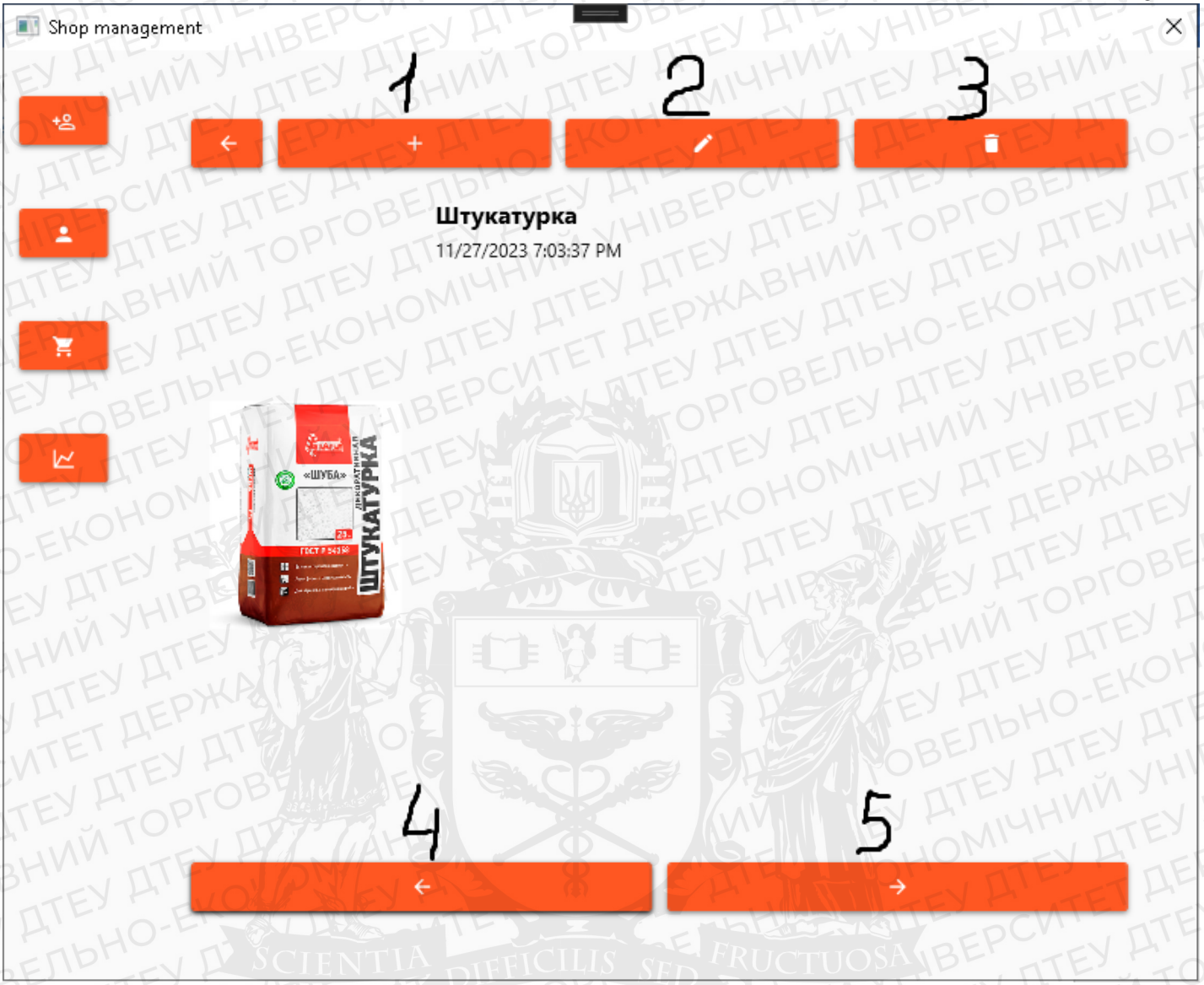


Рис. 4.15 Сторінка Category

На рисунку 4.15 зображена сторінка ‘Category’ на ній ми можемо бачити наступні кнопки для роботи :

1. Кнопка створення категорії
2. Кнопка для редагування обраної категорії
3. Видалення обраної категорії
4. Попередня категорія
5. Наступна категорія

При натисненні на кнопку 1 нам відкриється сторінка для створення нової категорії рисунок 4.16 , натиснувши кнопку 2 нам відкриється сторінка редагування обраної категорії рисунок 4.17 , при натисненні на кнопку 3 обрана категорія буде видалена, натиснувши кнопку 4 буде обрана попередня категорія (Окрім випадку якщо обрана нами не являється першою категорією в списку)

рисунок 4.18 , натиснувши кнопку 5 буде обрана наступна категорія (Окрім випадку якщо обрана нами не являється останньою категорією в списку) рисунок 4.19 .

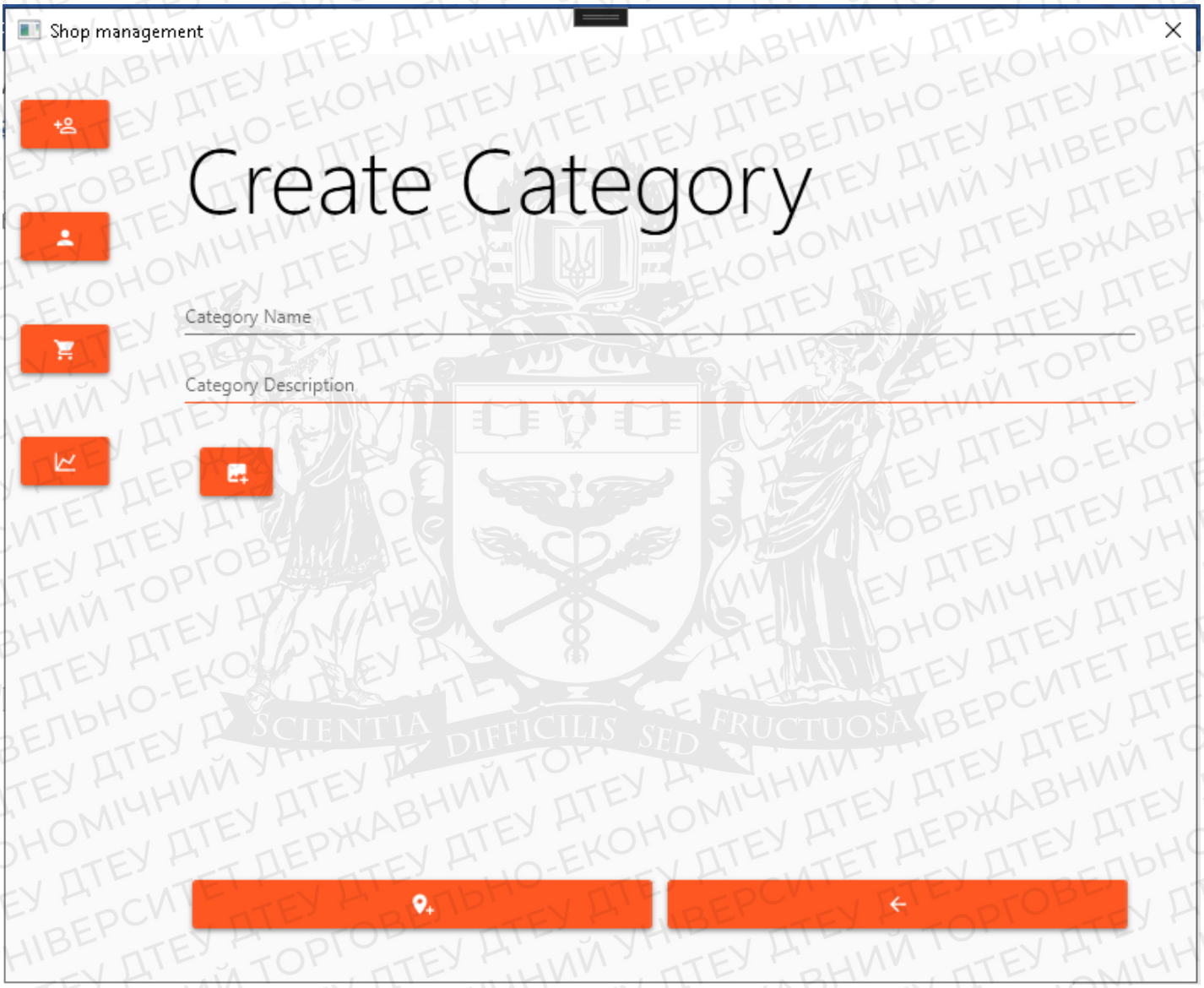


Рис. 4.16 Сторінка для створення нової категорії

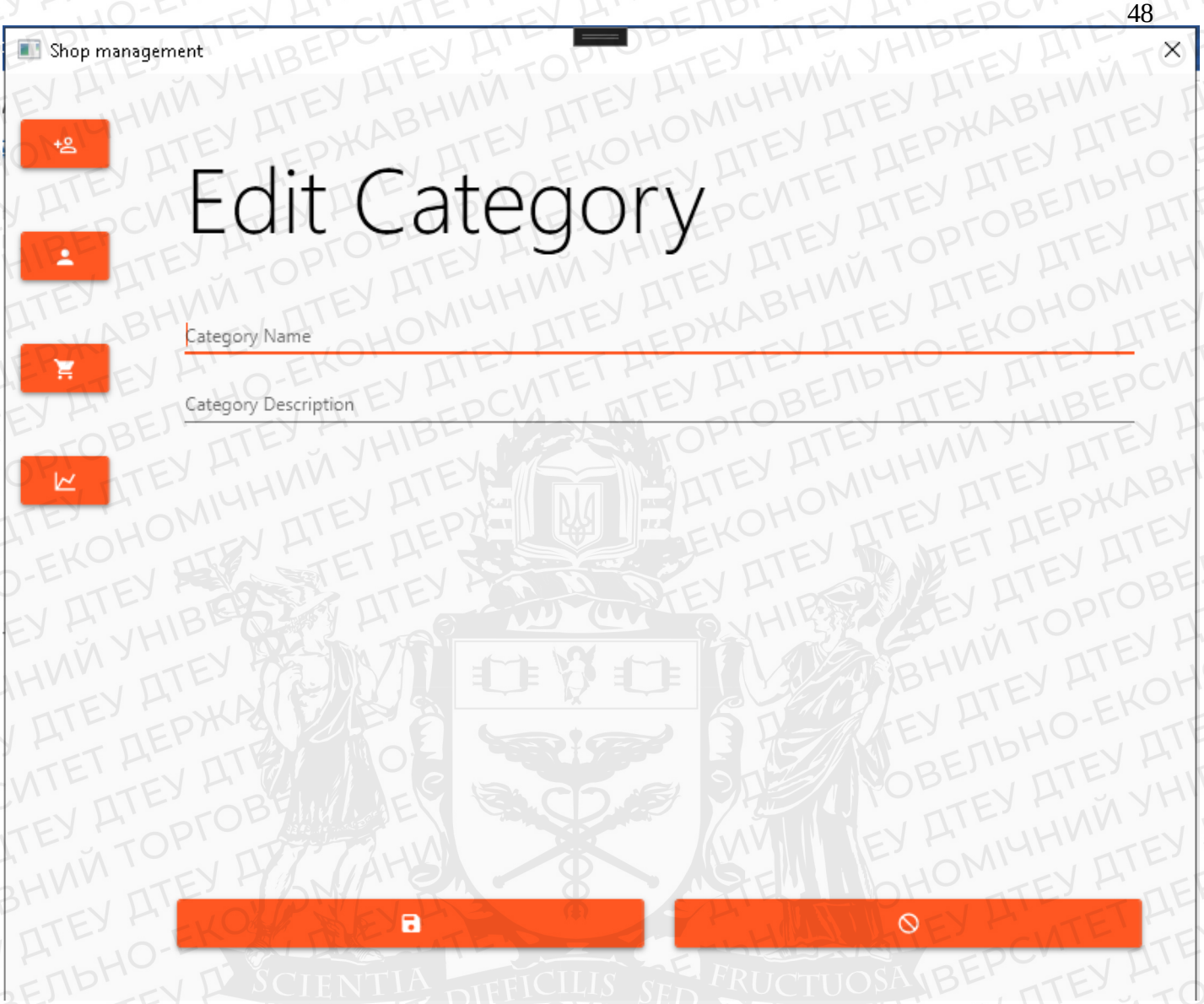


Рис. 4.17 Редагування обраної категорії

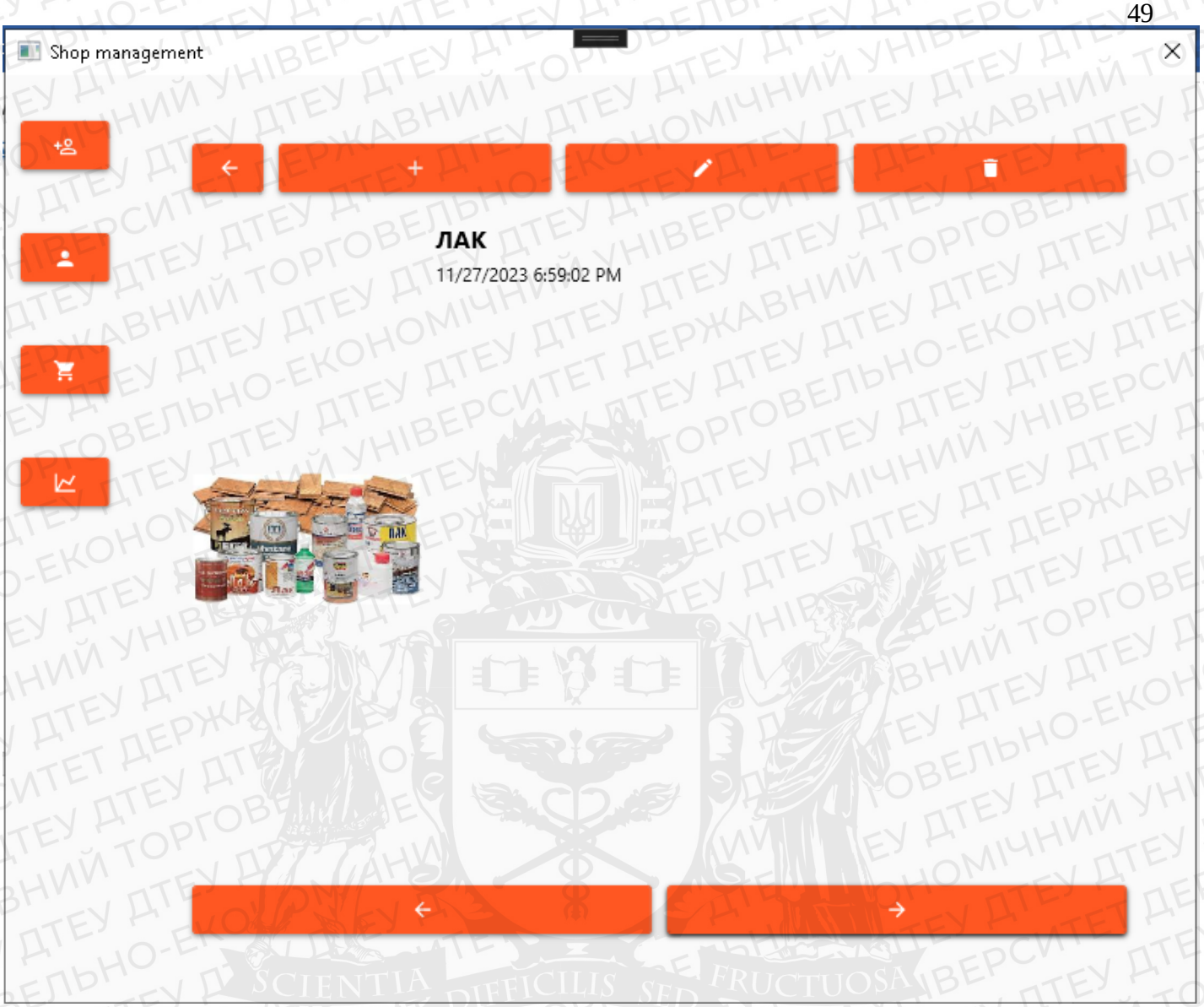


Рис. 4.18 Попередня категорія

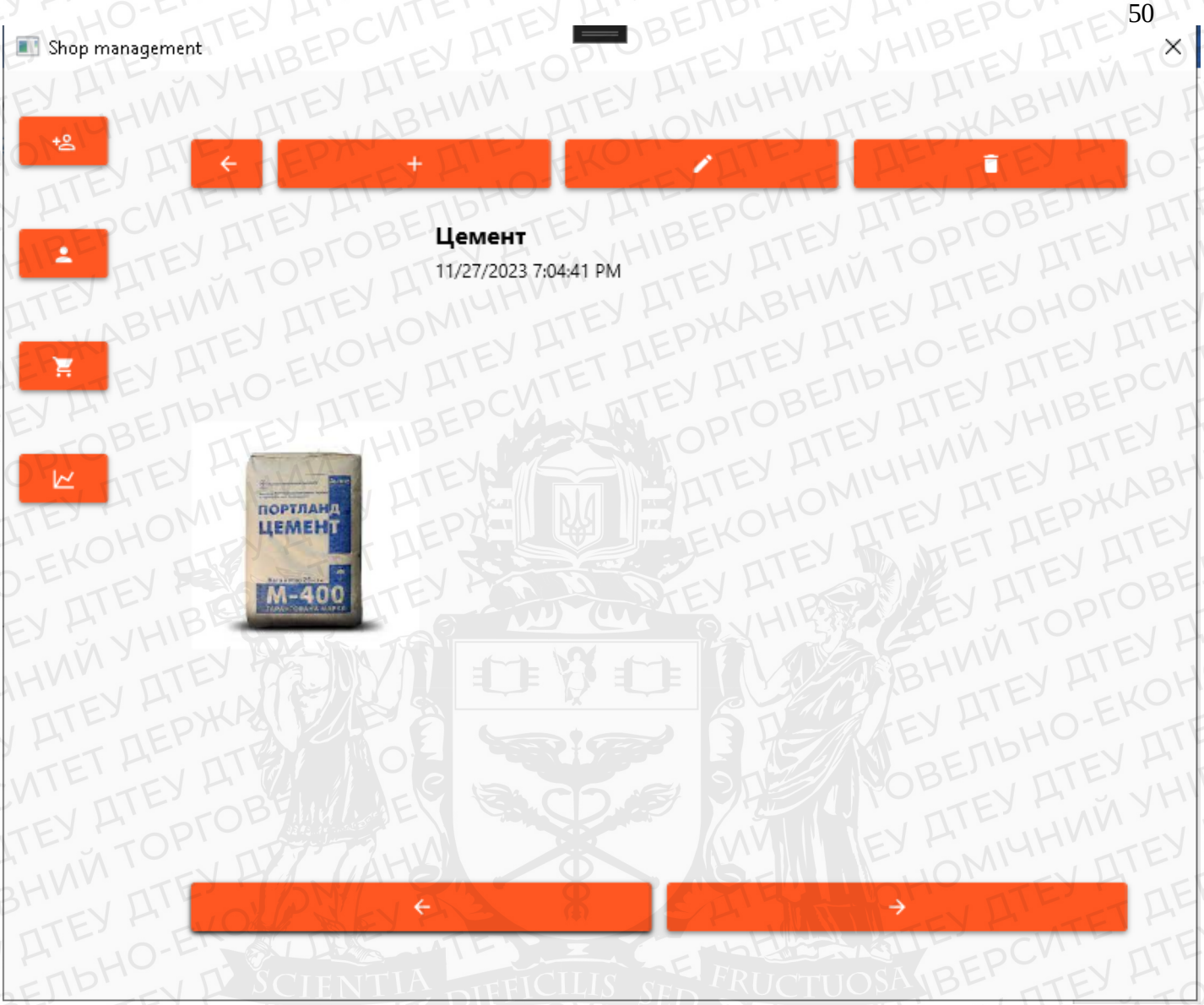


Рис. 4.19 Наступна категорія

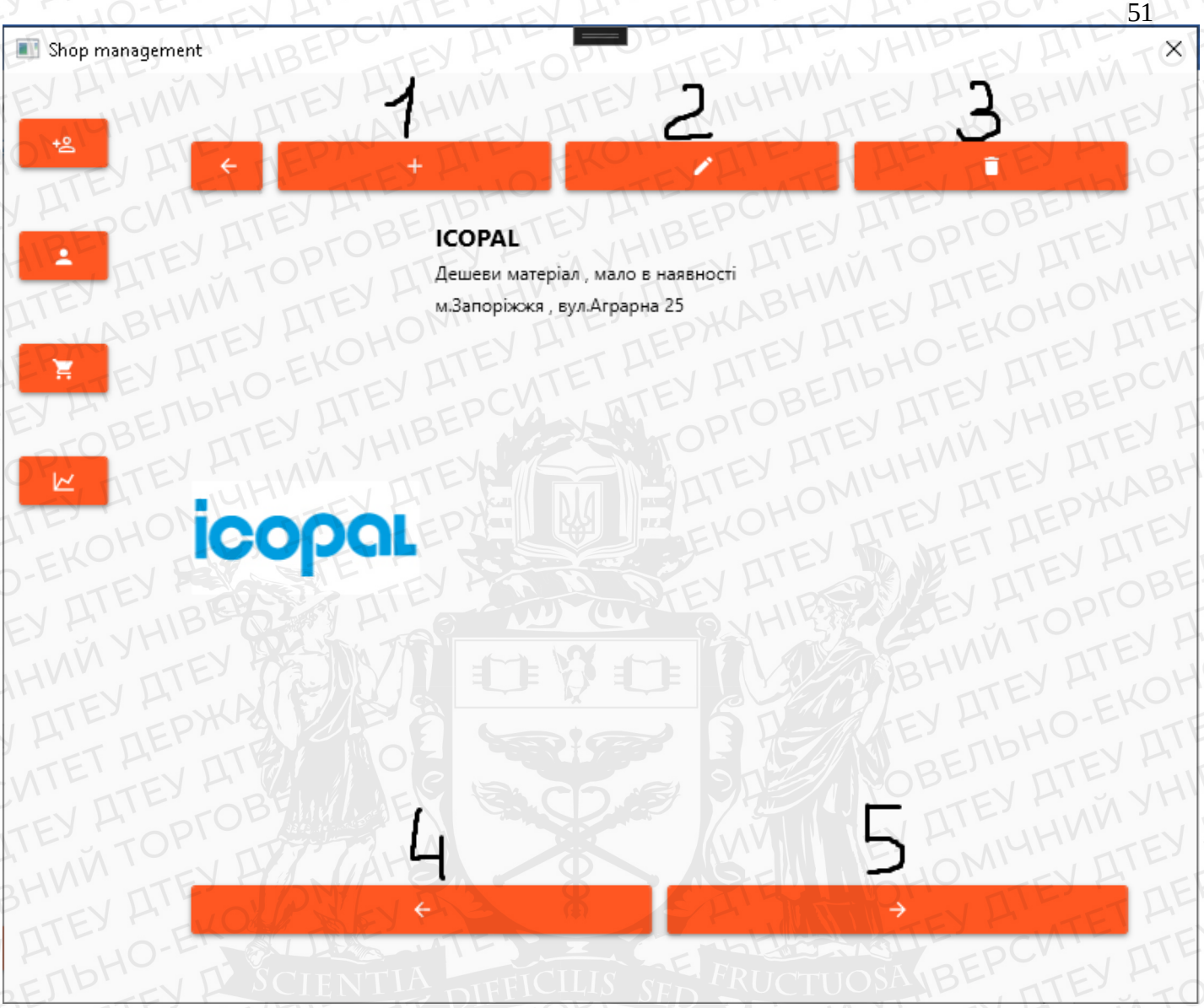


Рис. 4.20 Сторінка Locations

На рисунку 4.20 зображена сторінка ‘Locations’ на ній ми можемо бачити наступні кнопки для роботи :

1. Кнопка створення нової локації
2. Кнопка для редагування обраної локації
3. Видалення обраної локації
4. Попередня локація
5. Наступна локація

При натисненні на кнопку 1 нам відкриється сторінка для створення нової локації рисунок 4.21 , натиснувши кнопку 2 нам відкриється сторінка редагування обраної локації рисунок 4.22, при натисненні кнопки 3 обрана локація буде видалена , натиснувши кнопку 4 буде обрана попередня локація (Окрім випадку якщо обрана нами не являється першою в списку) рисунок 4.23 , натиснувши

кнопку 5 буде обрана наступна локація (Окрім випадку якщо обрана нами не являється останньою в списку) рисунок 4.24 .

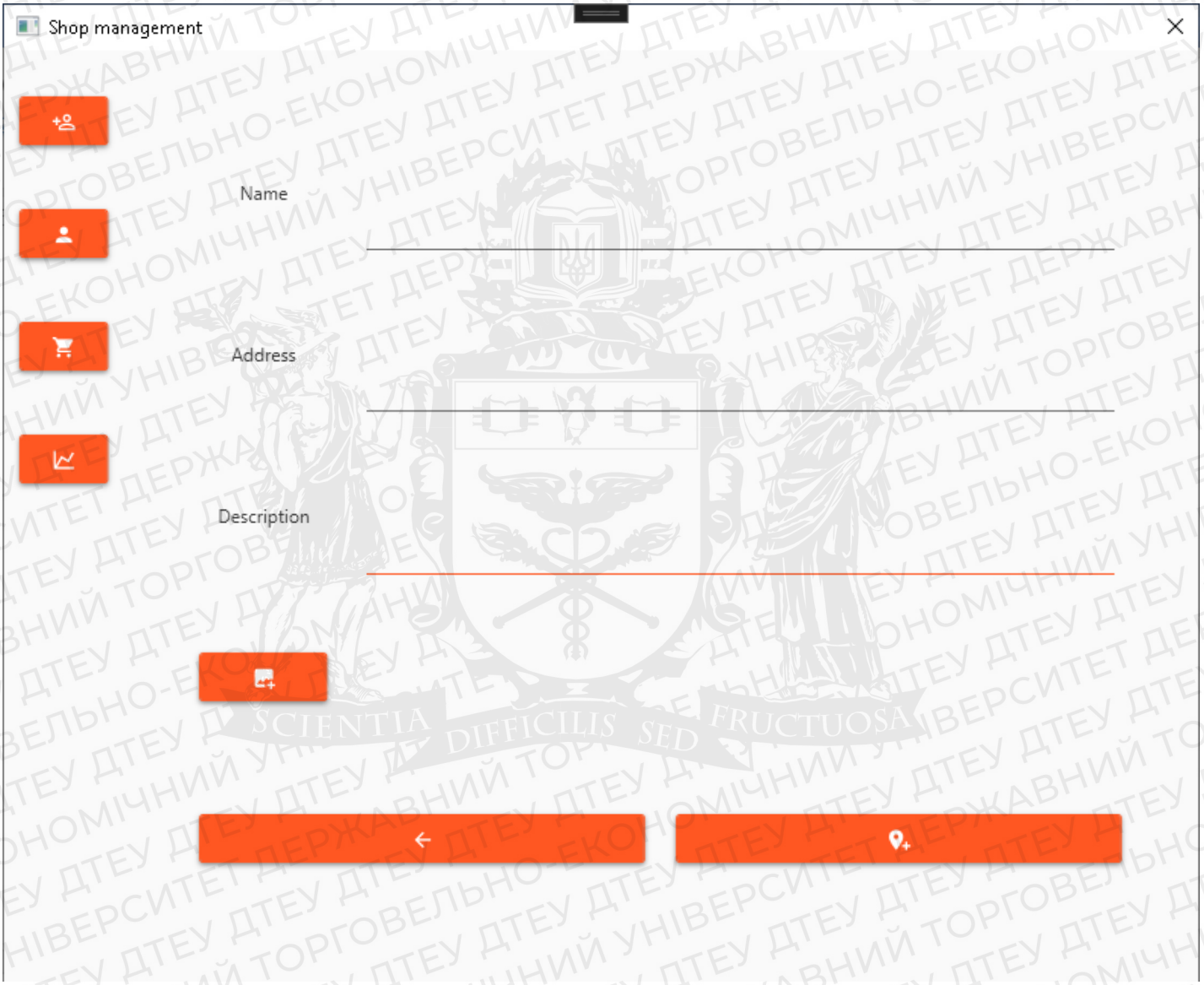


Рис. 4.21Сторінка створення Локації

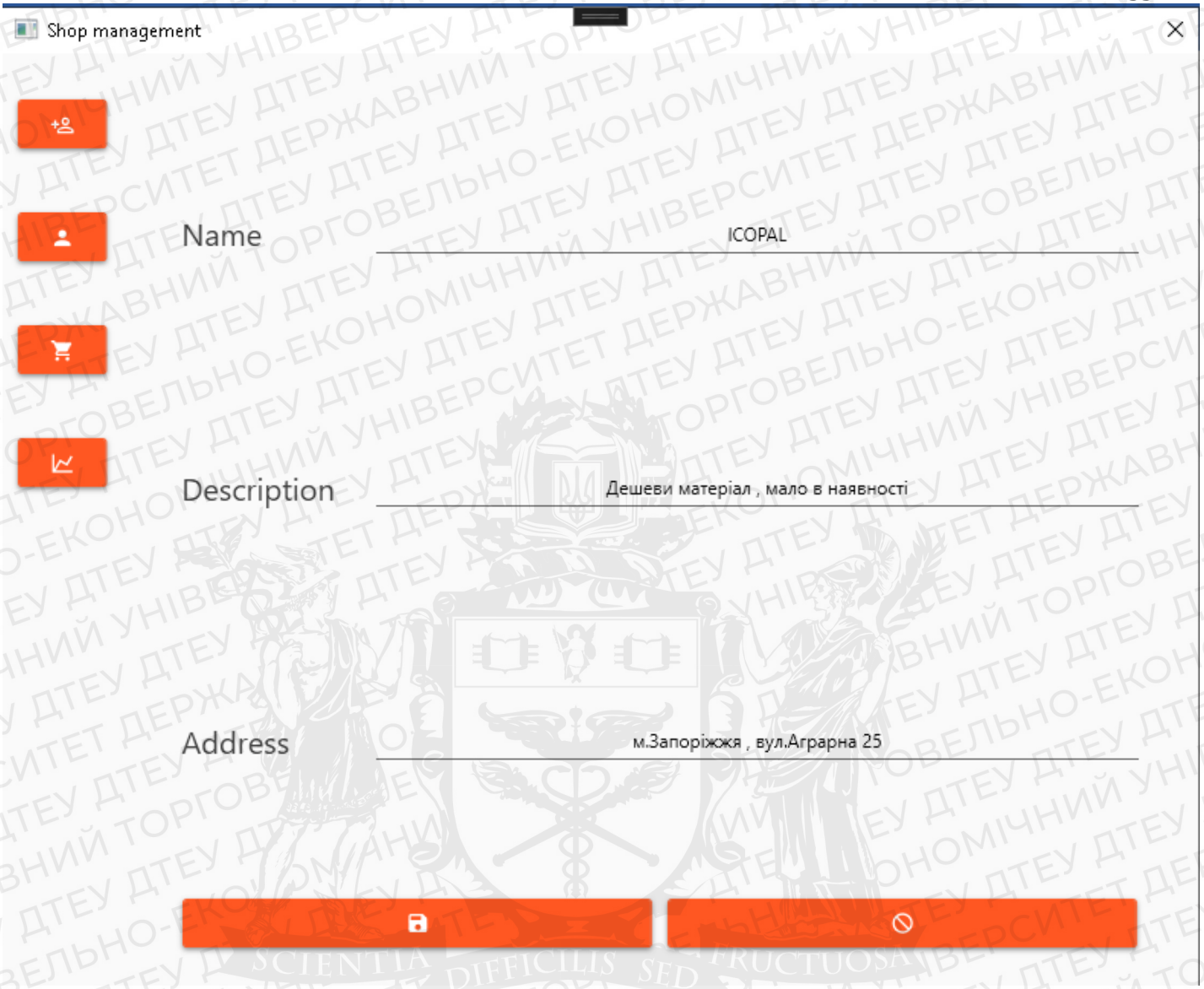


Рис. 4.22 Сторінка редагування обраної локації

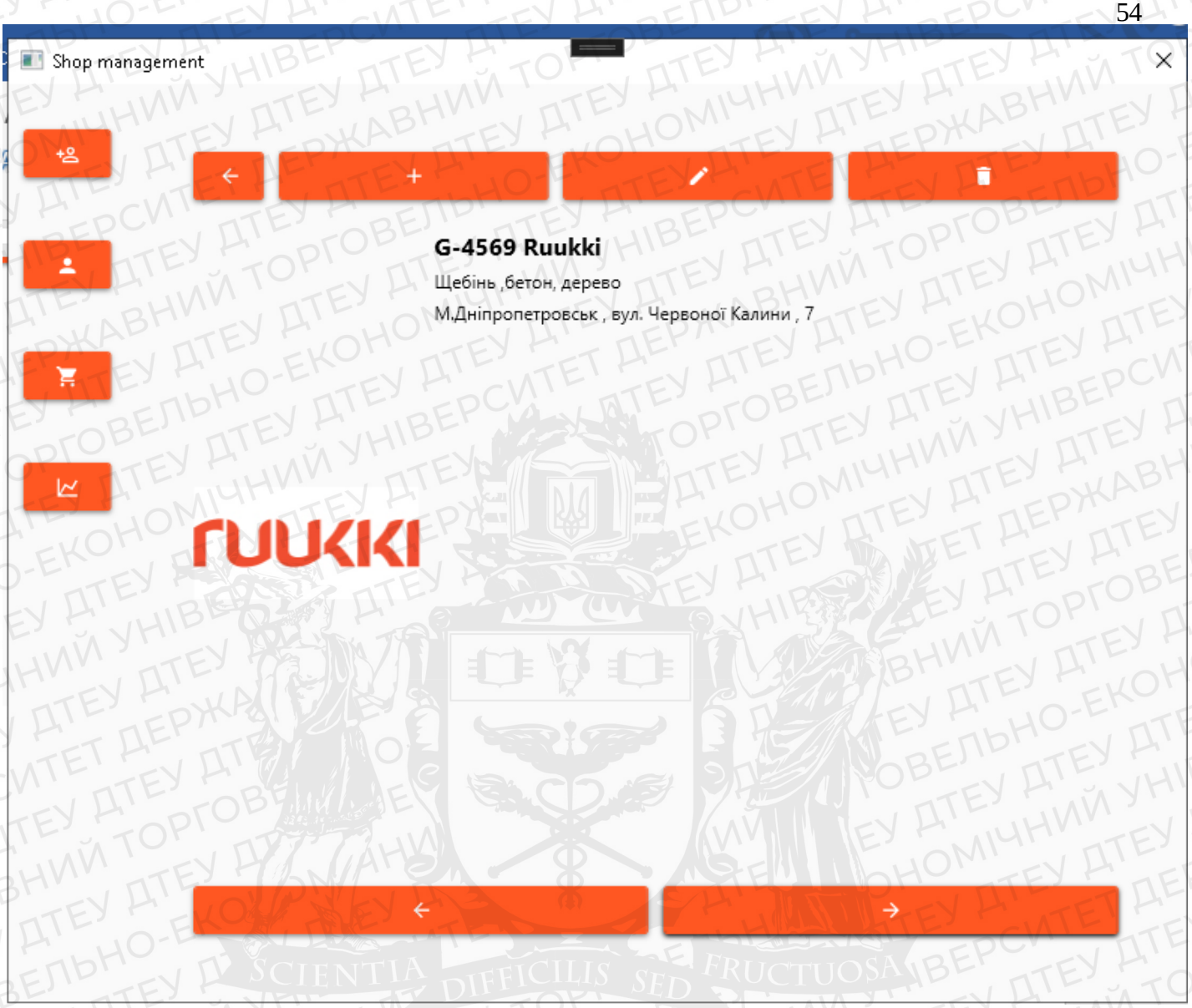


Рис. 4.23 Наступна локація

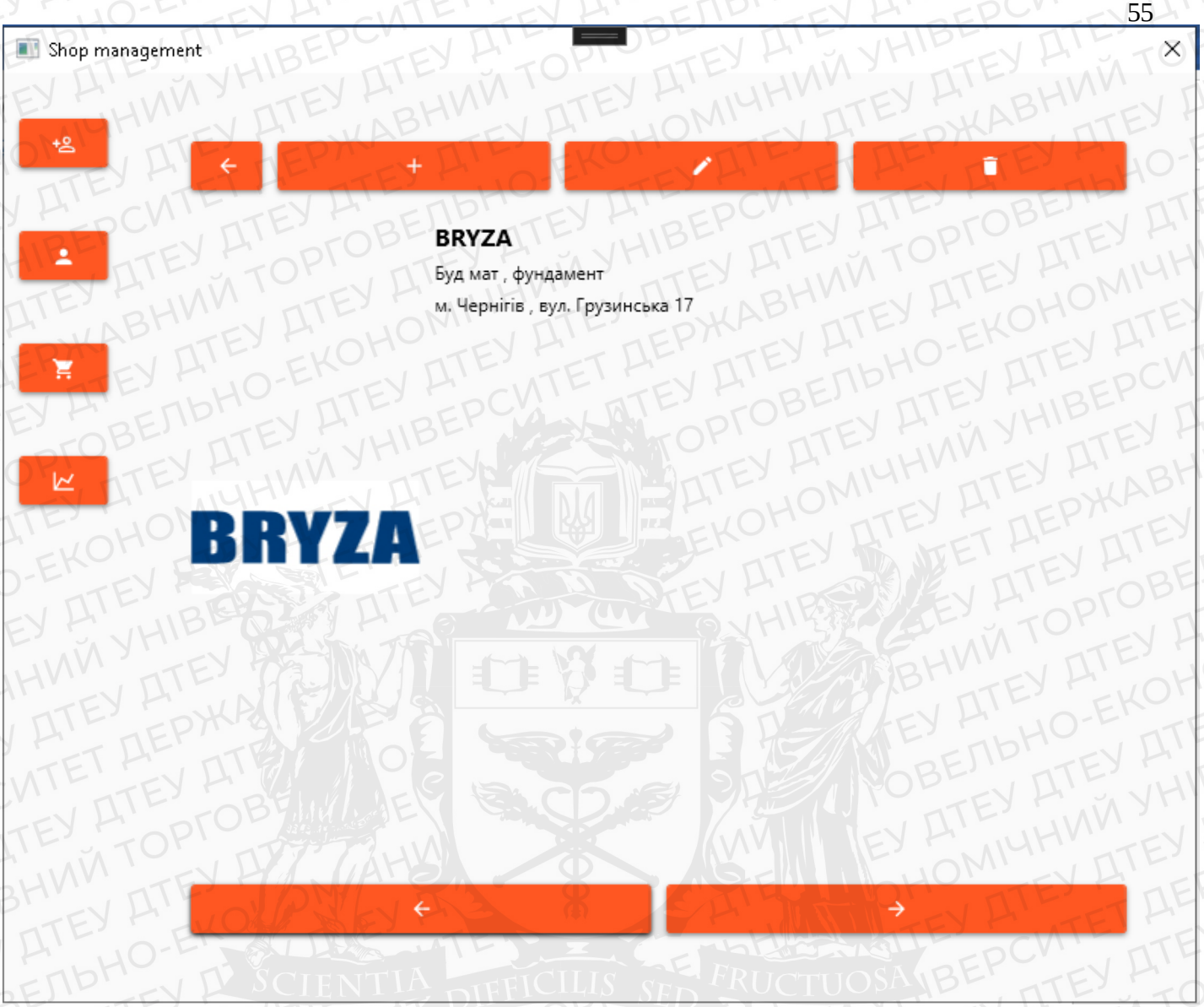


Рис. 4.24 попередня локація

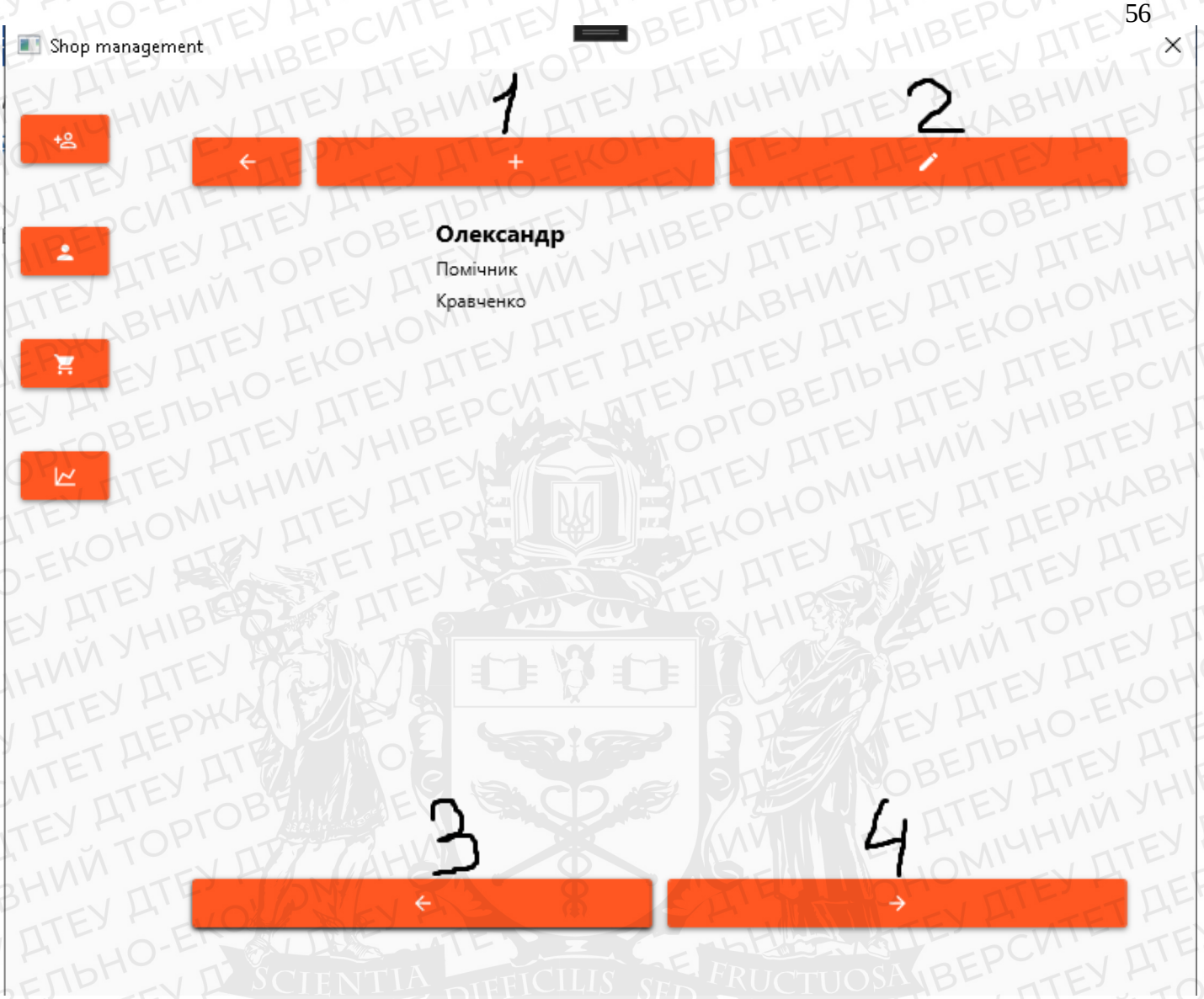


Рис. 4.25 Сторінка Sellers

На рисунку 4.25 зображена сторінка ‘Sellers` на ній ми можемо бачити наступні кнопки для роботи :

1. Кнопка створення нового продавця
2. Кнопка для редагування обраного продавця
3. Попередній продавець
4. Наступний продавець

При натисненні на кнопку 1 нам відкриється сторінка для створення нового продавця рисунок 4.26 , натиснувши кнопку 2 нам відкриється сторінка редагування обраного продавця рисунок 4.27, , натиснувши кнопку 3 буде обрана попередня локація (Окрім випадку якщо обрана нами не являється першою в списку) рисунок 4.28 , натиснувши кнопку 4 буде обрана наступна локація (Окрім випадку якщо обрана нами не являється останньою в списку) рисунок 4.29 .

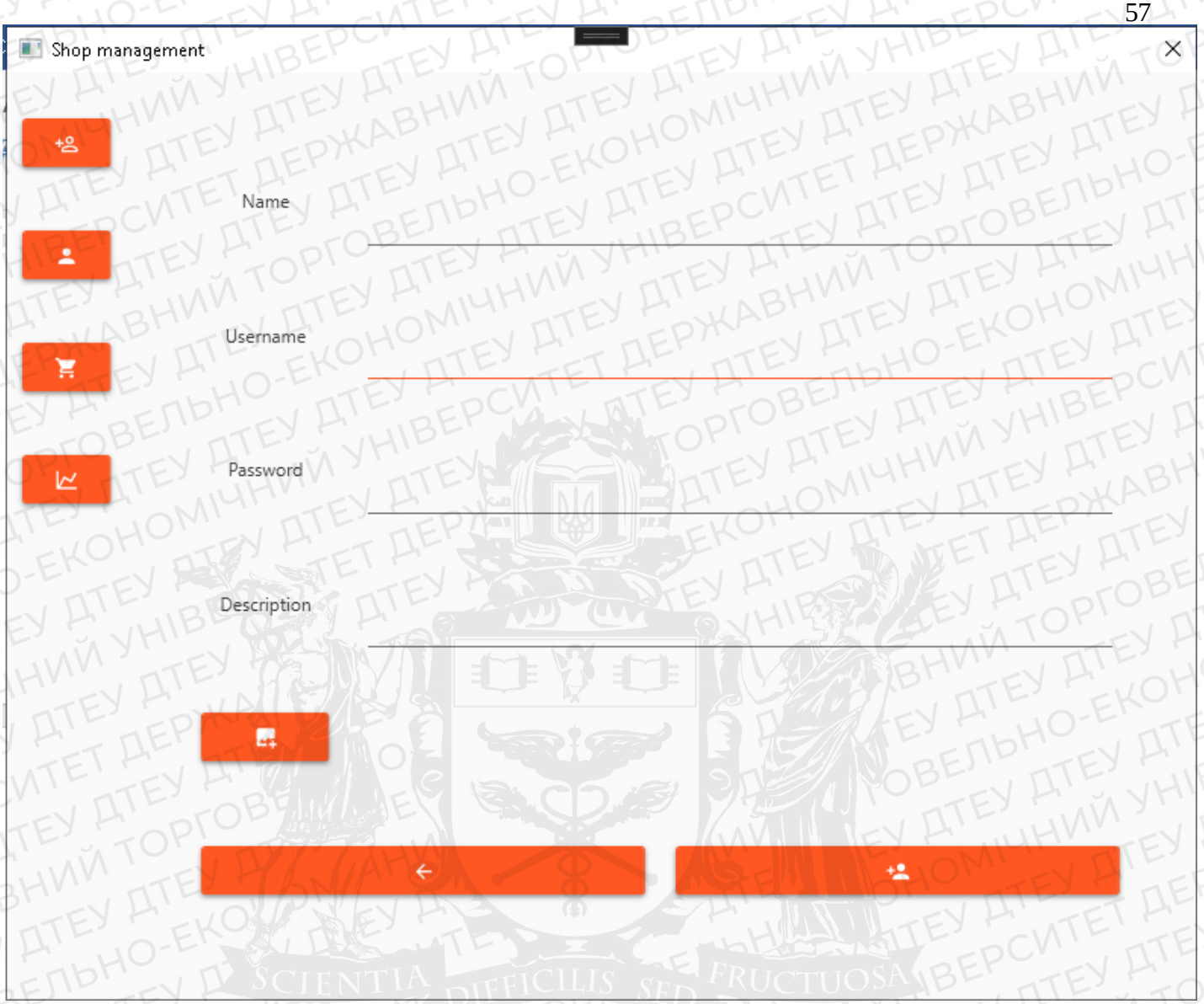


Рис. 4.26 Сторінка створення нового продавця

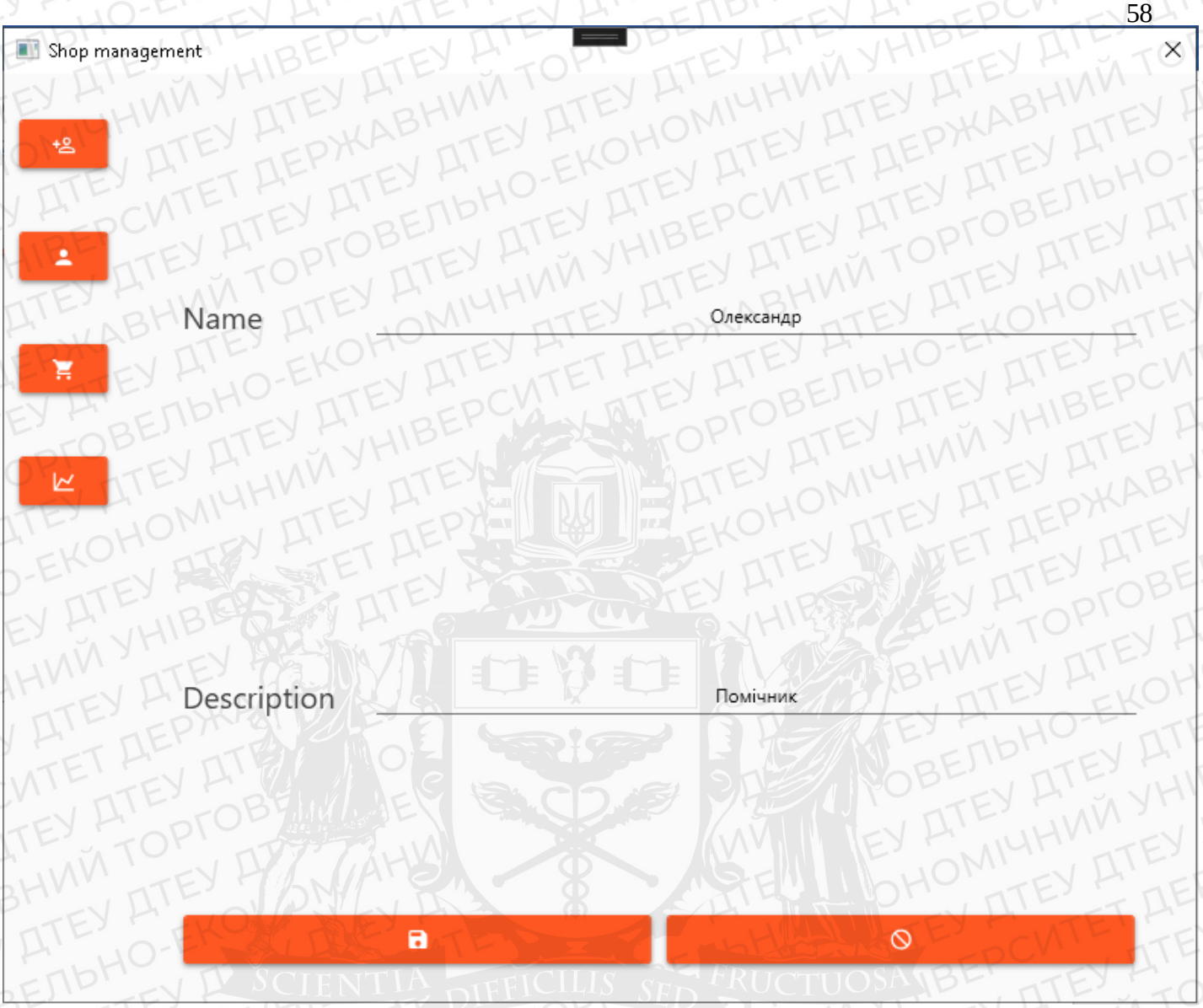


Рис. 4.27 Сторінка редагування продавця

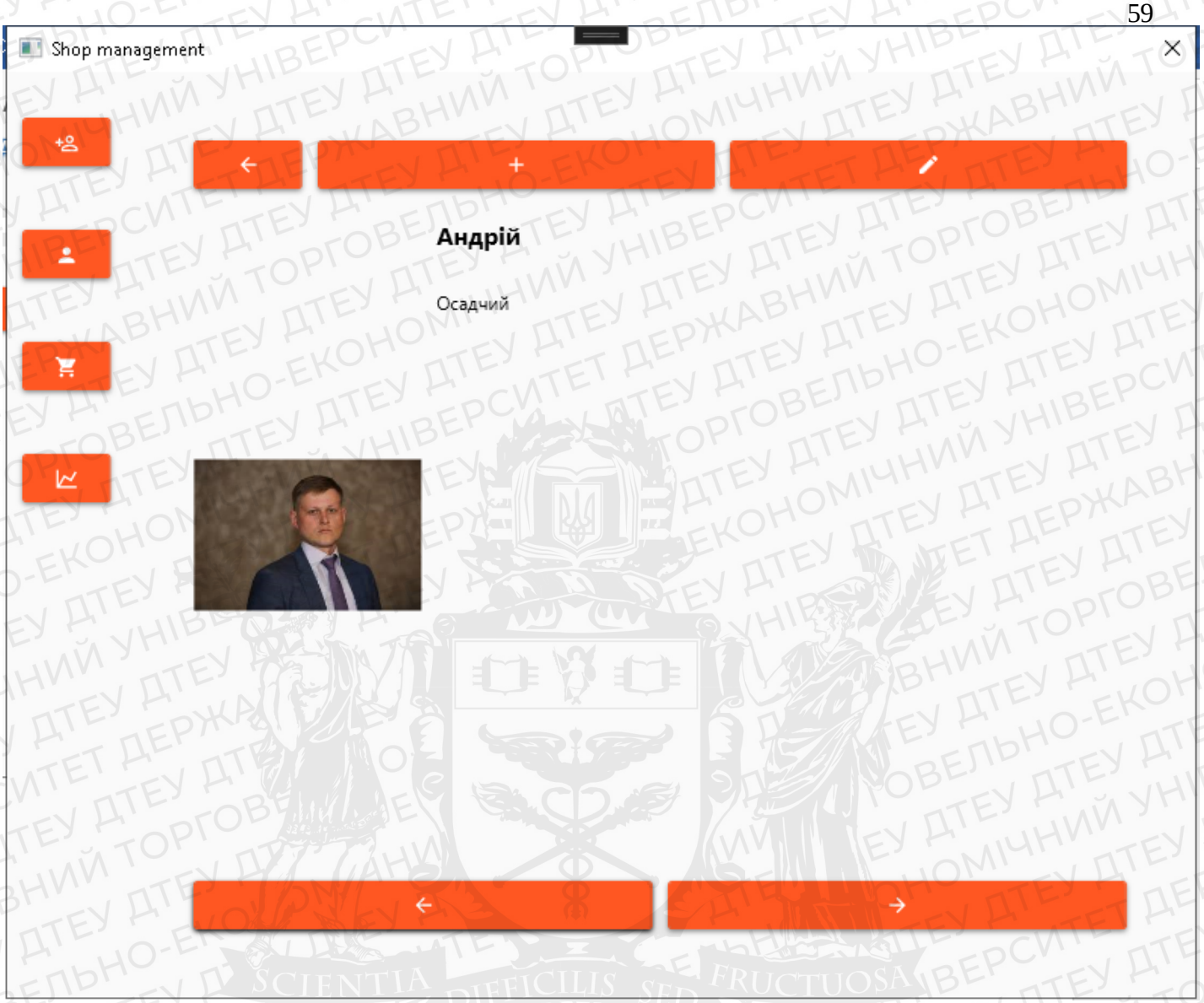


Рис. 4.28 Попередній продавець

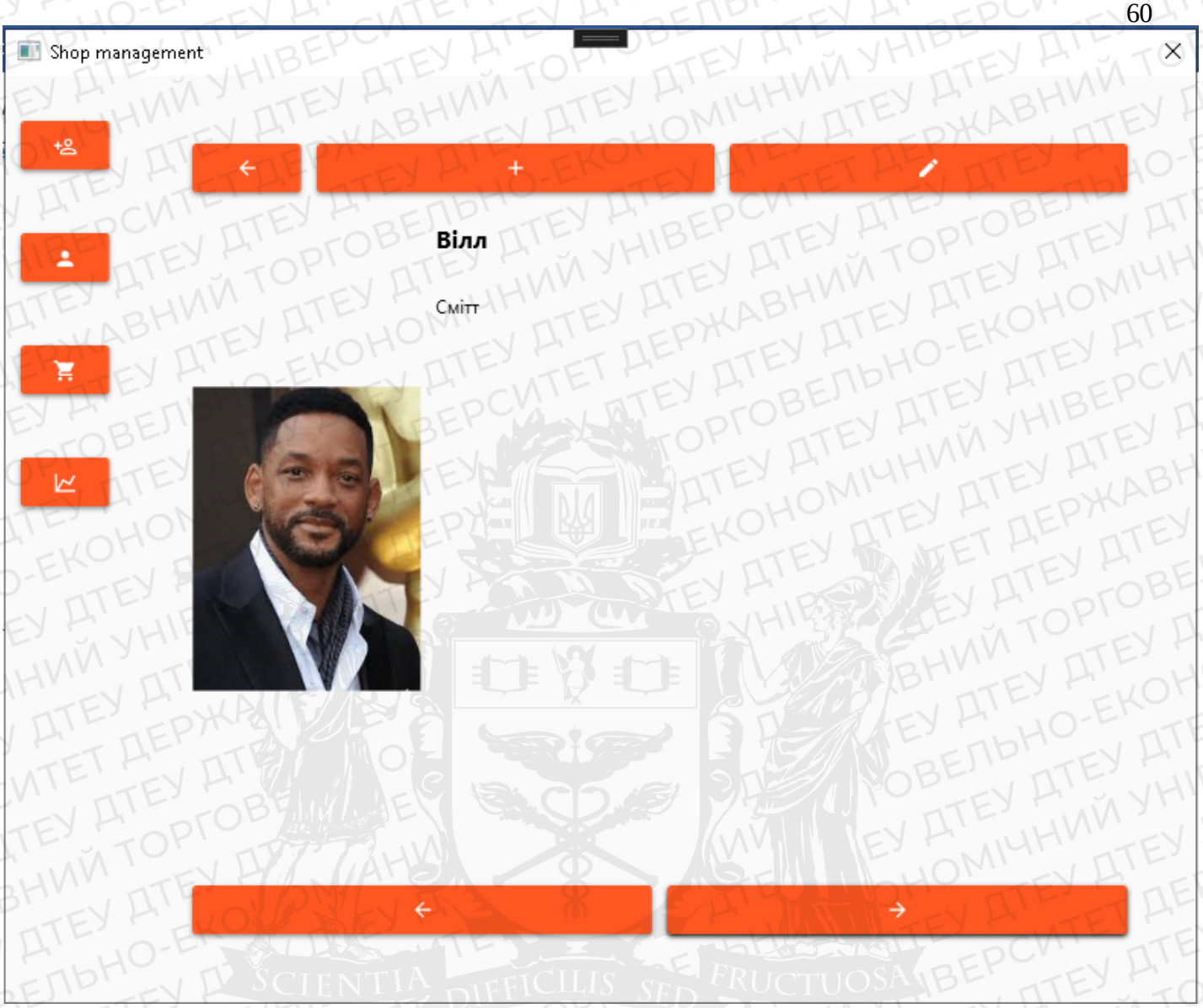


Рис. 4.29 Попередній продавець

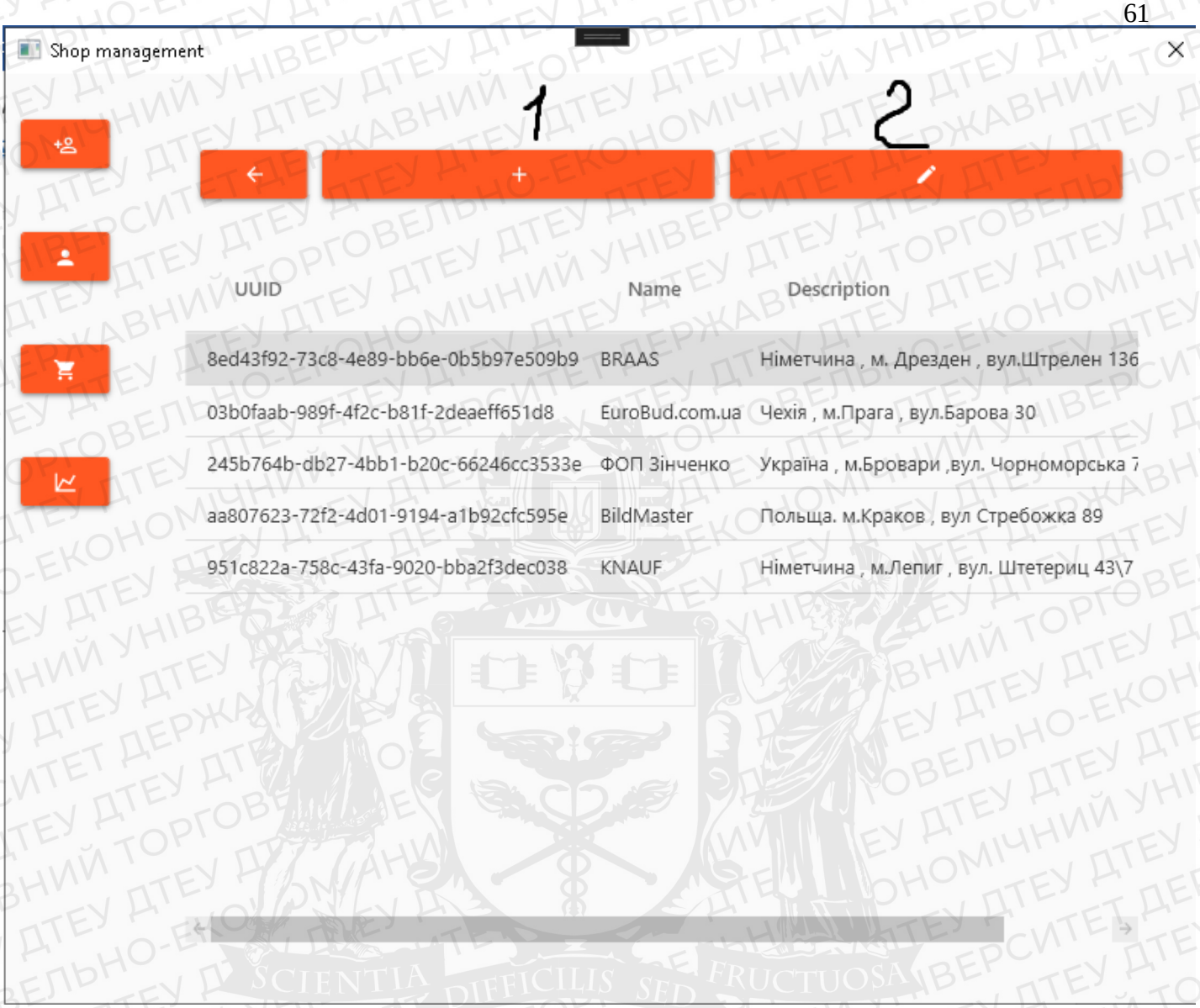


Рис . 4.30 Сторінка Suppliers

На сторінці ‘Suppliers ’ ми бачимо список постачальників товару , нам доступні кнопки 1 , додавання нового постачальника рисунок 4.31 , натиснувши кнопку 2 відкриєтеся сторінка редагування обраного постачальника рисунок 4.32 .

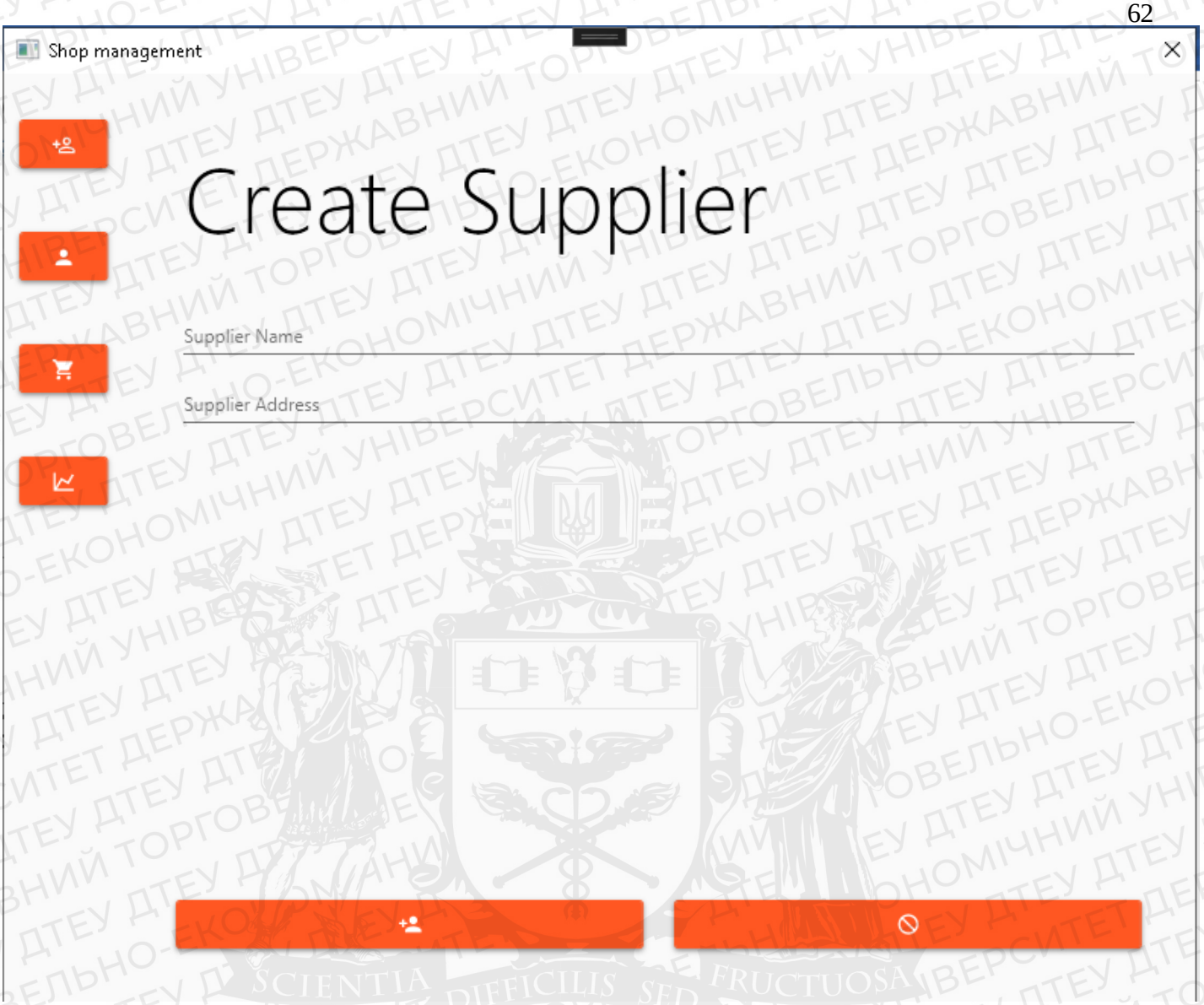


Рис. 4.31 Сторінка створення нового постачальника

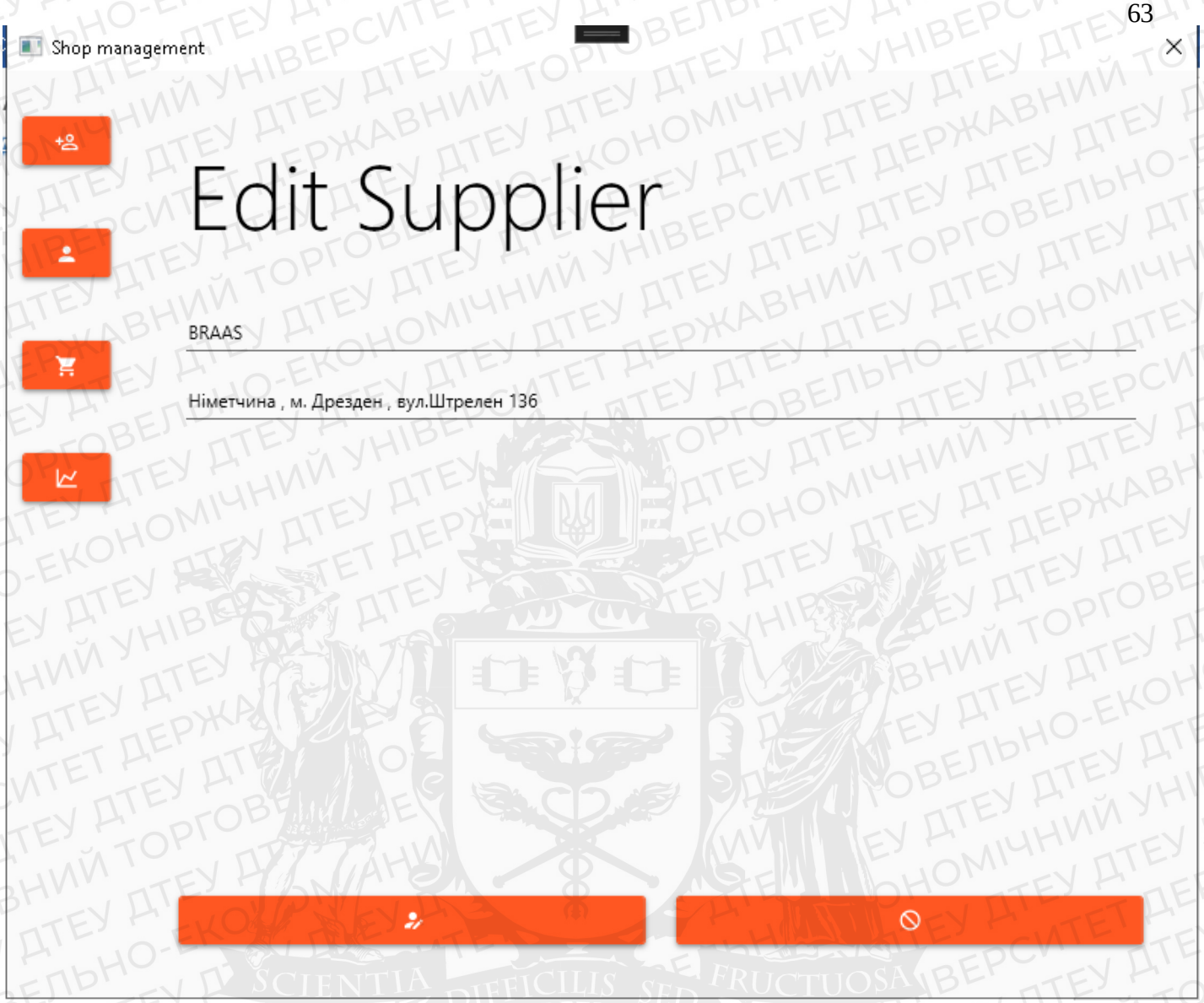


Рис. 4.32 Сторінка редагування постачальника

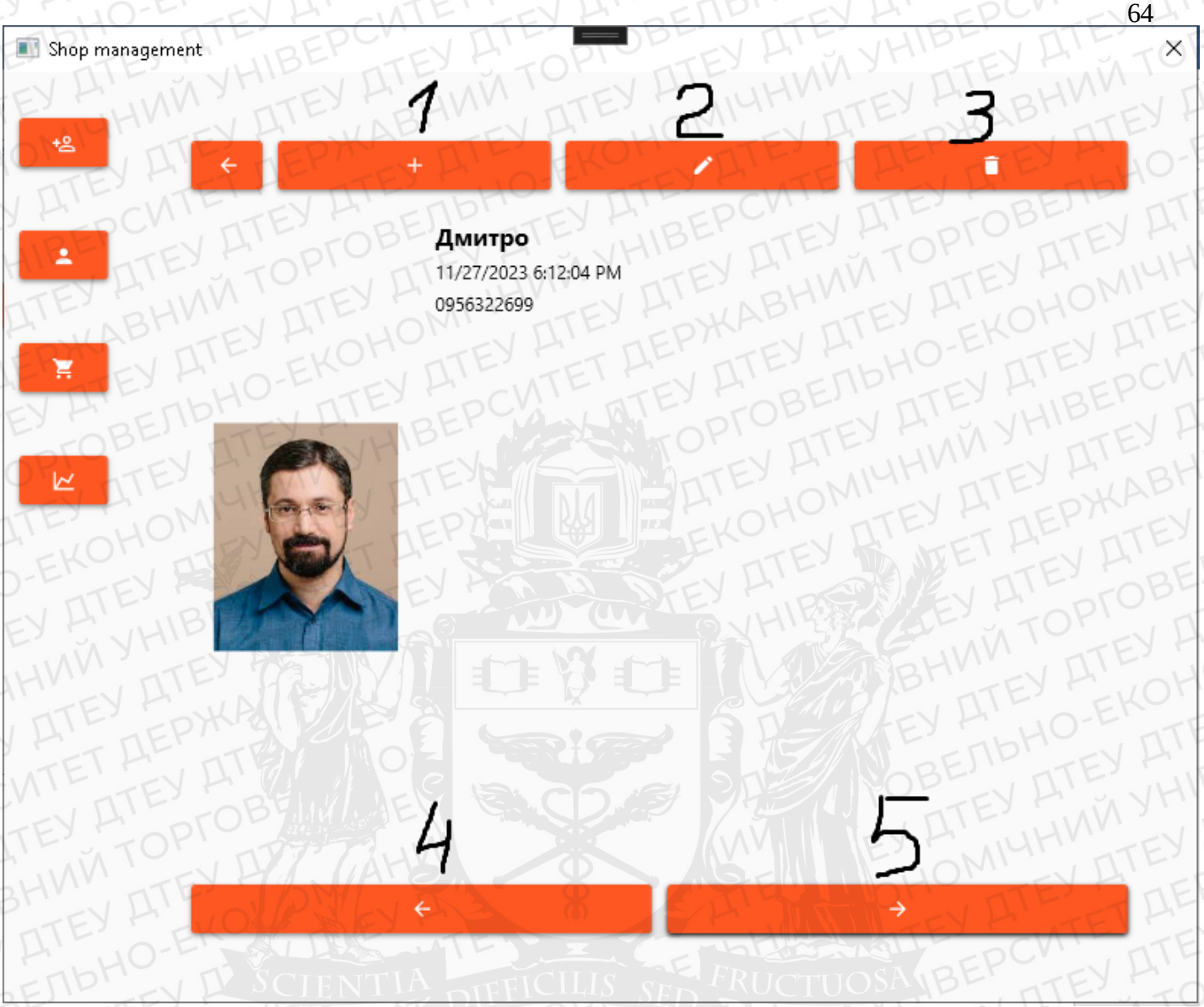


Рис .4.33 Сторінка Customers

На рисунку 4.25 зображена сторінка ‘Customers ` на ній ми можемо бачити наступні кнопки для роботи :

1. Кнопка створення нового покупця
2. Кнопка для редагування обраного покупця
3. Копка для видалення обраного покупця
4. Попередній покупець
5. Наступний покупець

При натисненні на кнопку 1 нам відкриється сторінка для створення нового продавця рисунок 4.34 , натиснувши кнопку 2 нам відкриється сторінка редагування обраного продавця рисунок 4.35, , натиснувши кнопку 3 буде видалений обраний покупець , натиснувши кнопку4 попередній покупець (Окрім випадку якщо обраний нами не являється першим в списку) рисунок 4.36 ,

натиснувши кнопку 5 буде обраний наступний покупець (Окрім випадку якщо обраний нами не являється останнім в списку) рисунок 4.37 .

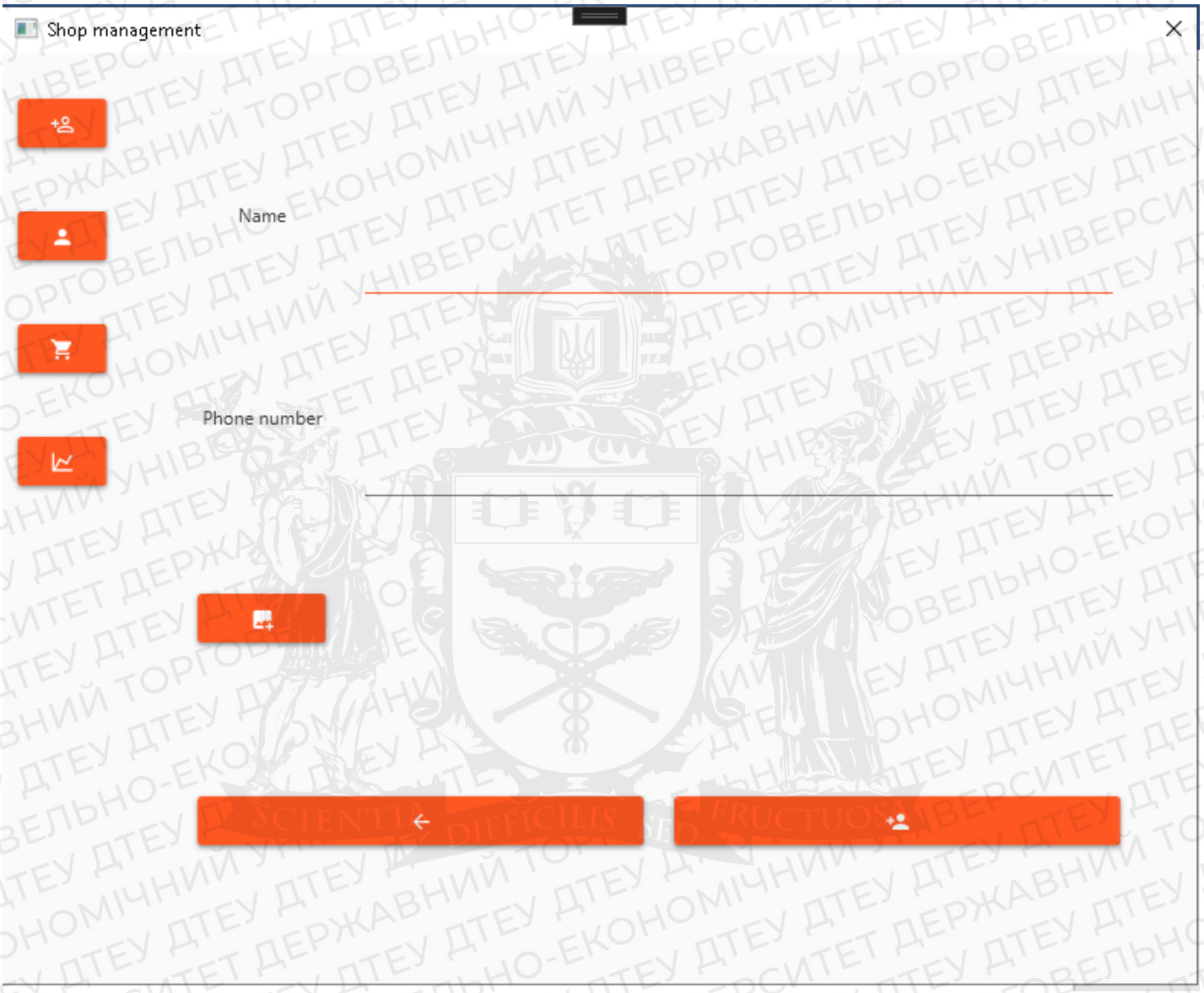


Рис. 4.34 Сторінка для створення покупця

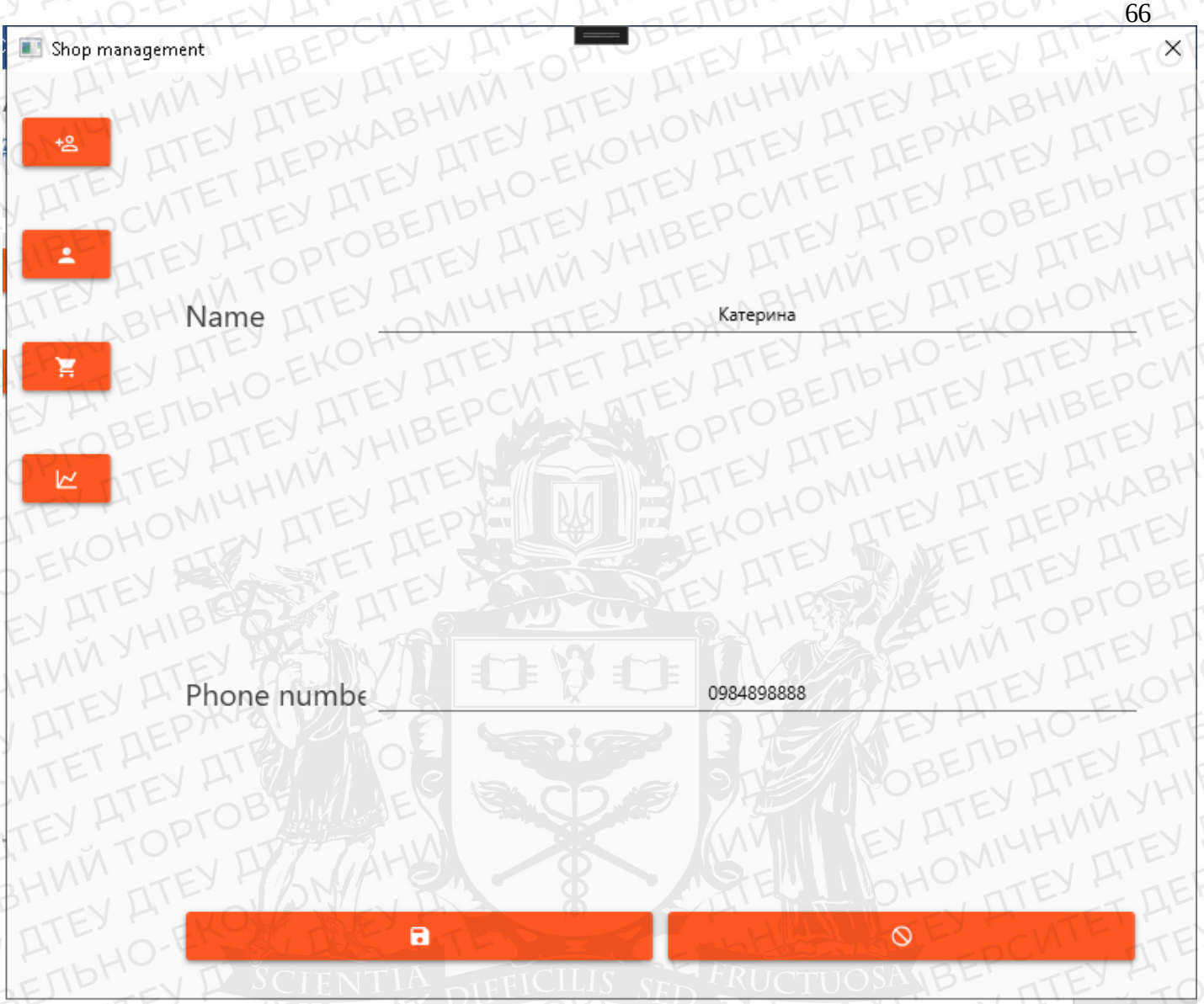


Рис. 4.35 сторінка редагування продавця

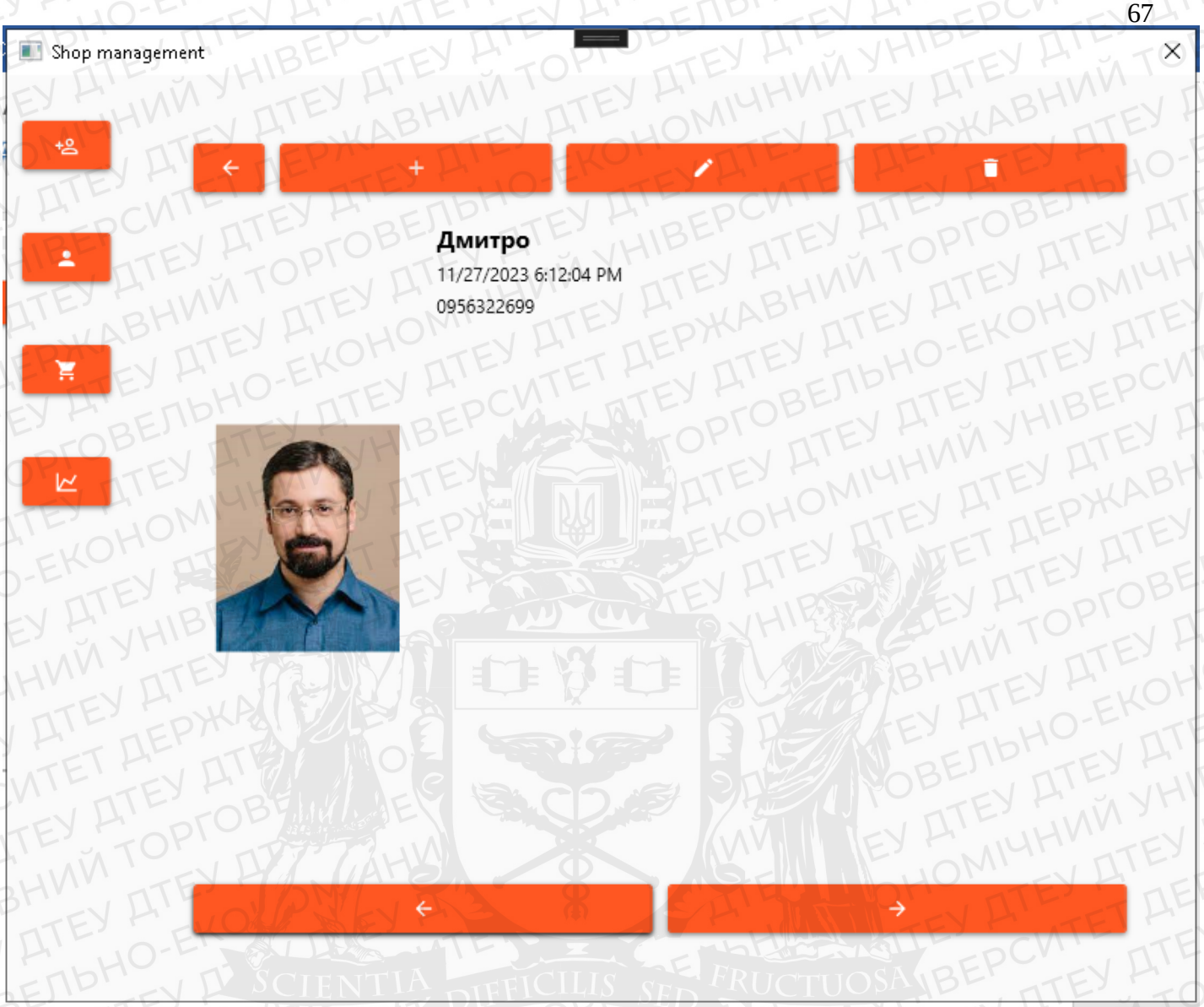


Рис .4.36 Наступний покупець

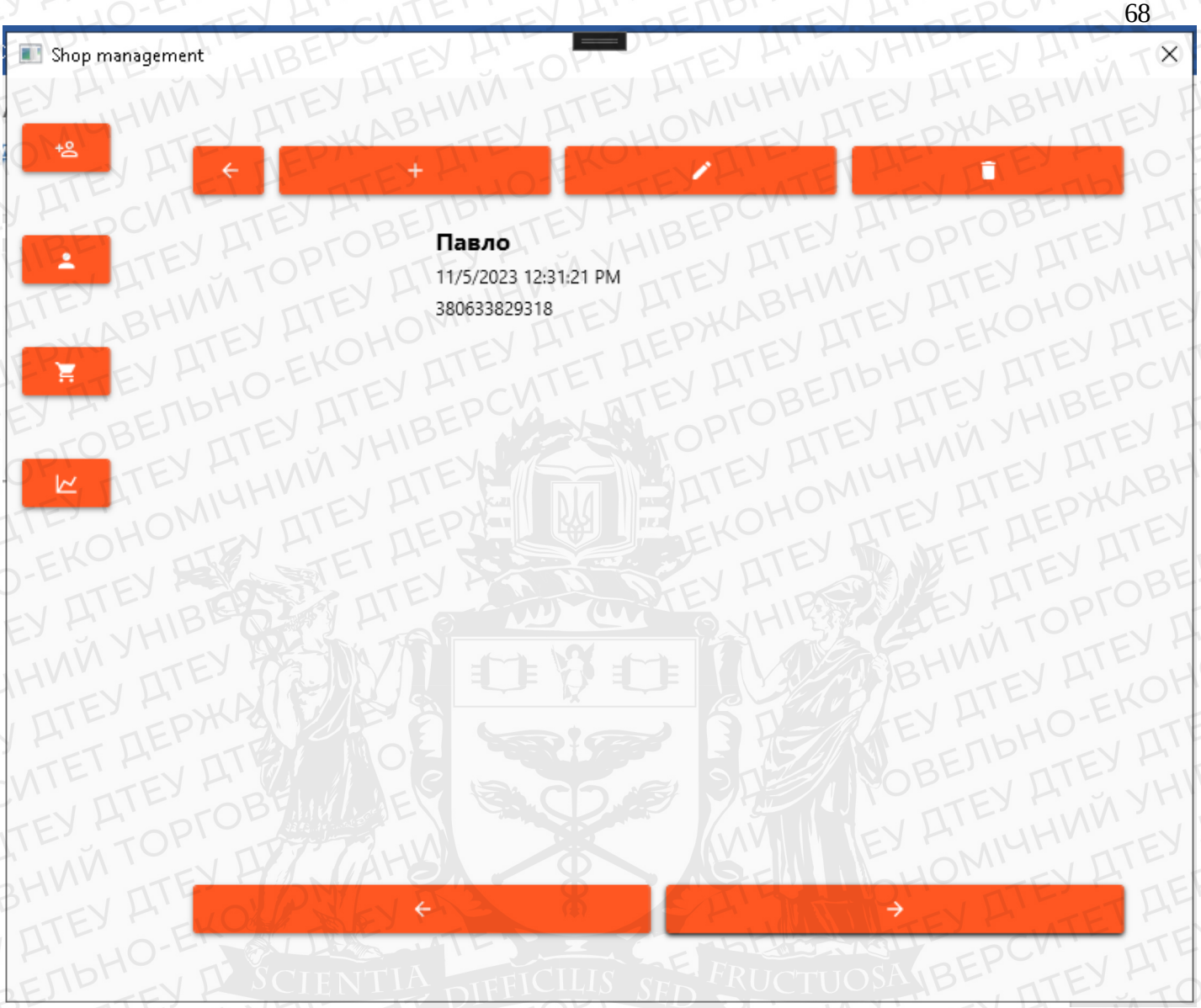


Рис. 4.37 попередній покупець

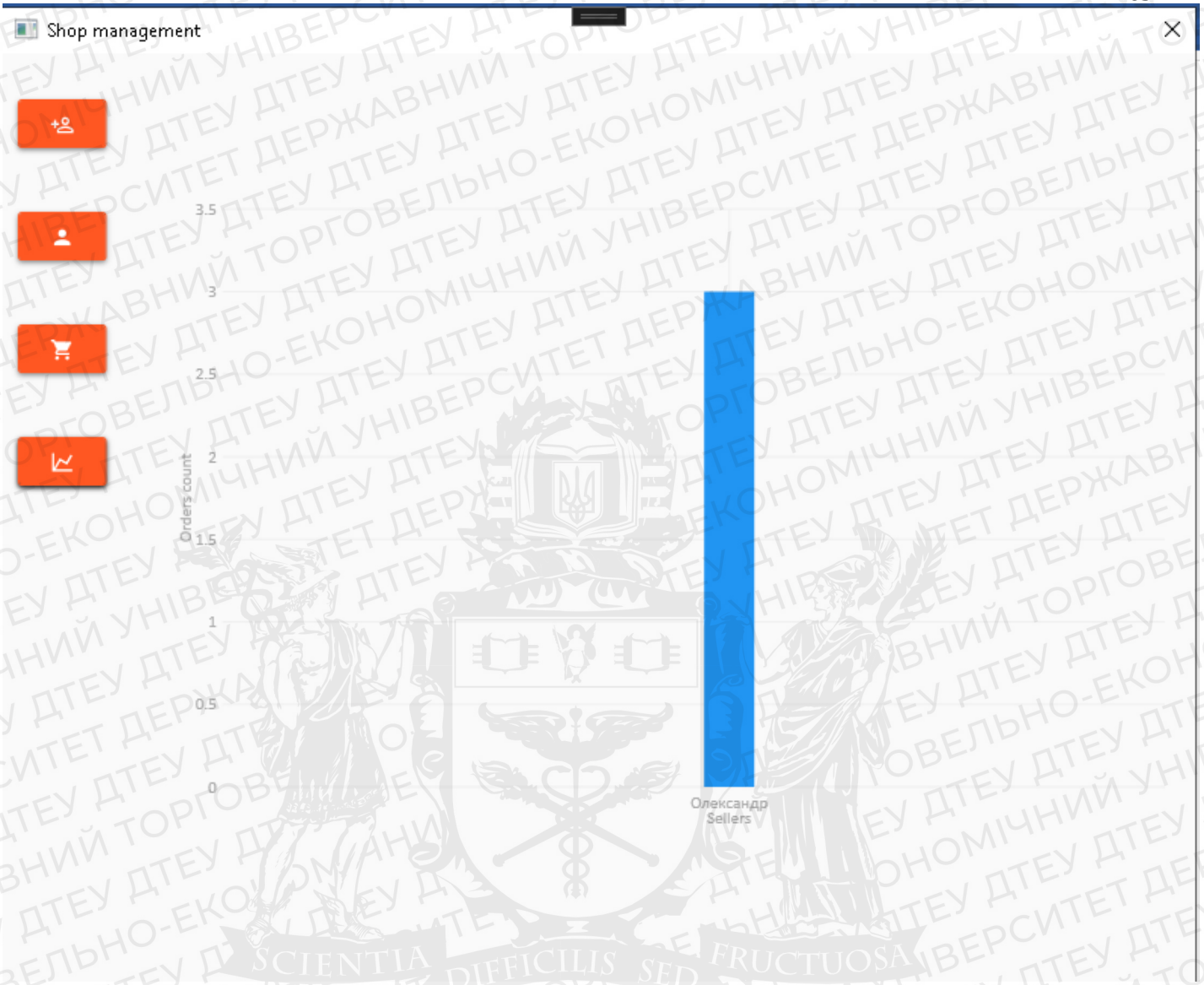


Рис. 4.38 Статистика по замовленням відповідними продавцями

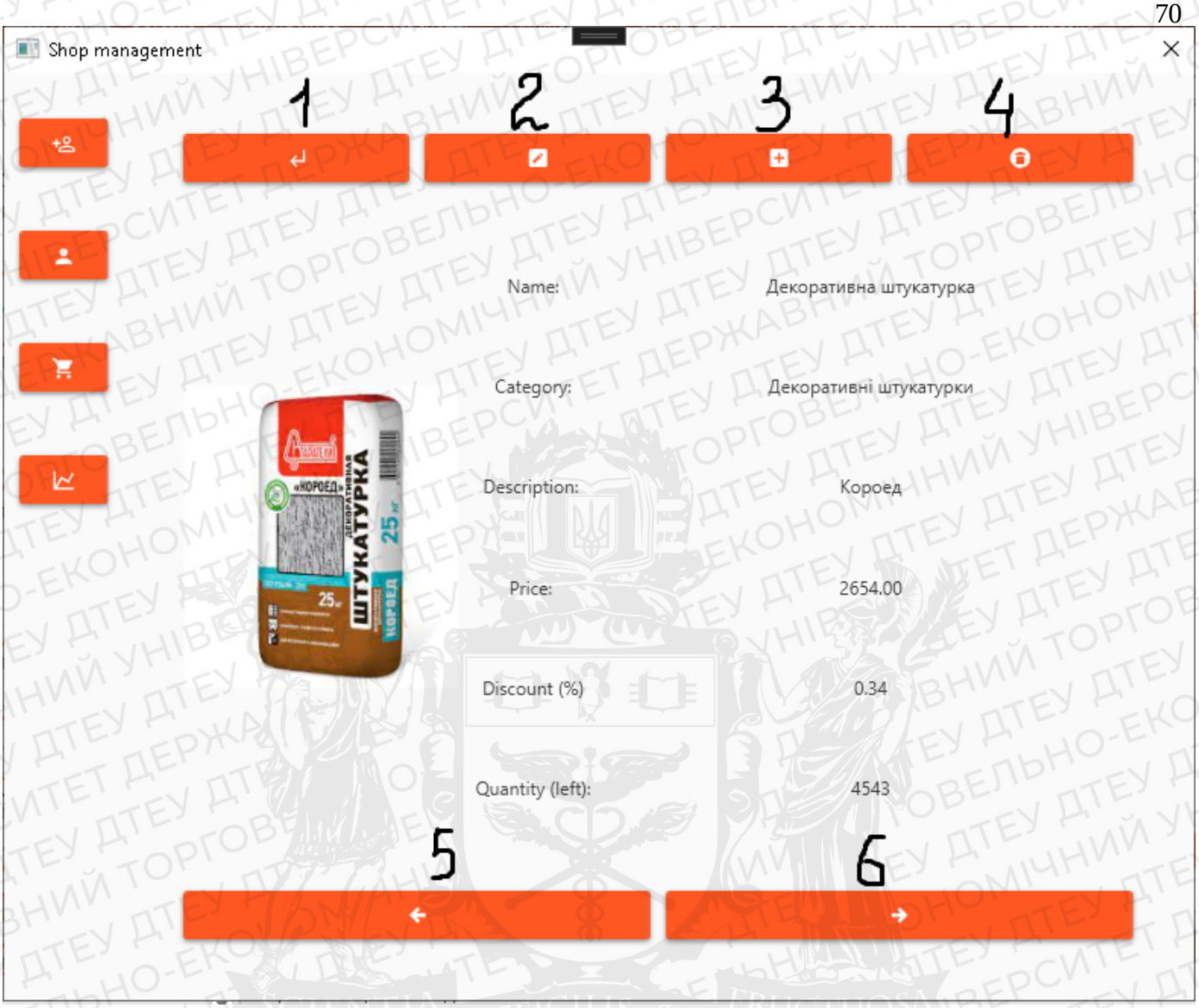


Рис. 4.39 Сторінка Product

На рисунку 4.39 зображена Сторінка product , на данній сторінці ми бачимо наступні кнопки:

1 Повернення до меню

2 Редагування товару

3 Сторінка створення нового товару

4 Видалення обраного товару

5 Попередній товар

6 наступний товар

Данна сторінка являється головною для управління запасами в торгівлі , менеджер зможе змінювати запаси відповідно від замовлень та залишків на складах.

При натисненні на кнопку 1 відкриється сторінка меню рисунок 4.3 ,

натиснувши кнопку 2 відкриється сторінка редагування обраного товару рисунок 4.40, натиснувши кнопку 3 відкриється сторінка створення нового товару рисунок 4.41, при натисненні на кнопку 4 обрана категорія буде видалена, натиснувши кнопку 5 буде обрана попередня категорія (Окрім випадку якщо обрана нами не являється першою категорією в списку) рисунок 4.42 , натиснувши кнопку 6 буде обрана наступна категорія (Окрім випадку якщо обрана нами не являється останньою категорією в списку) рисунок 4.43 .

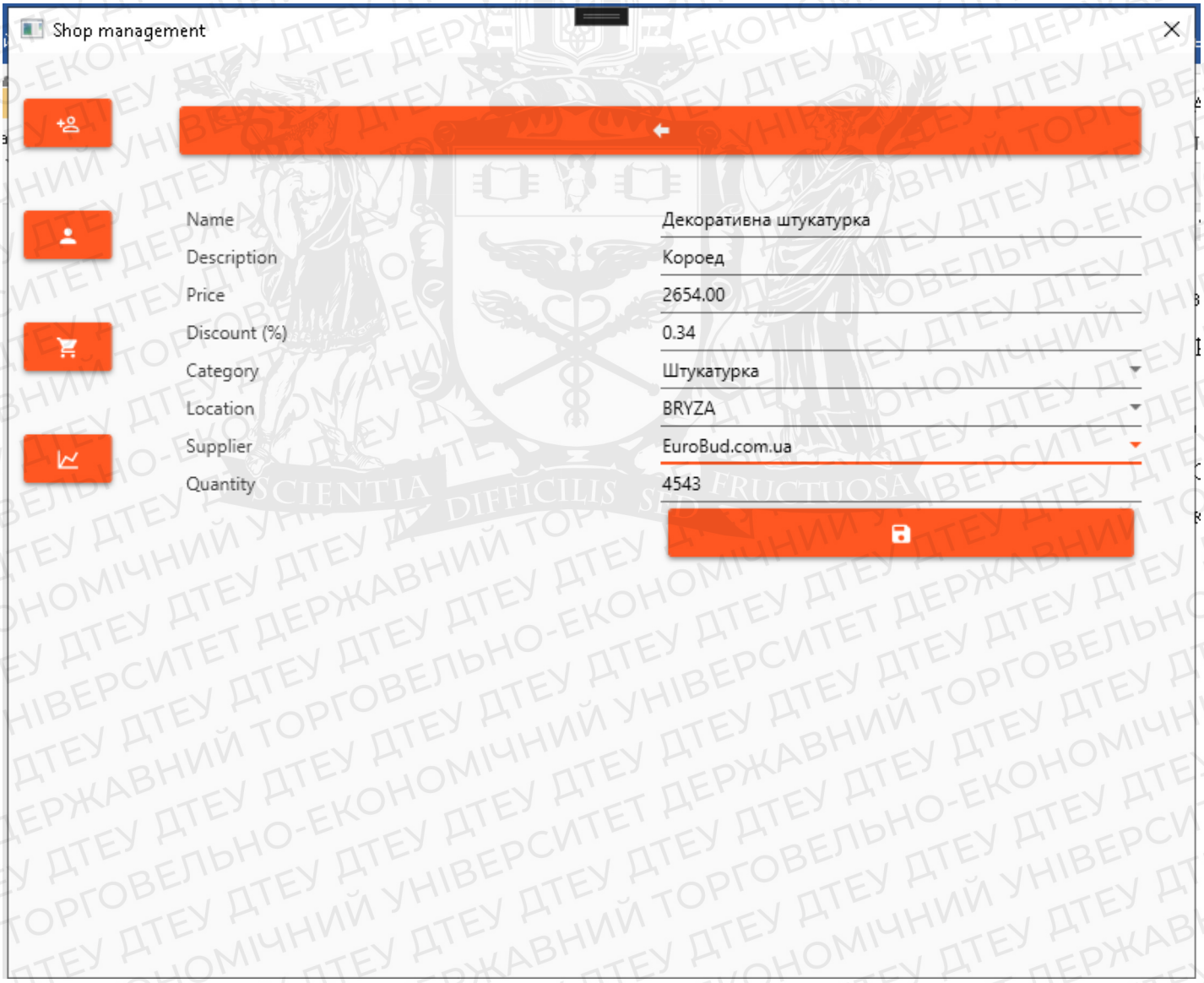


Рис. 4.40 Сторінка редагування обраного товару

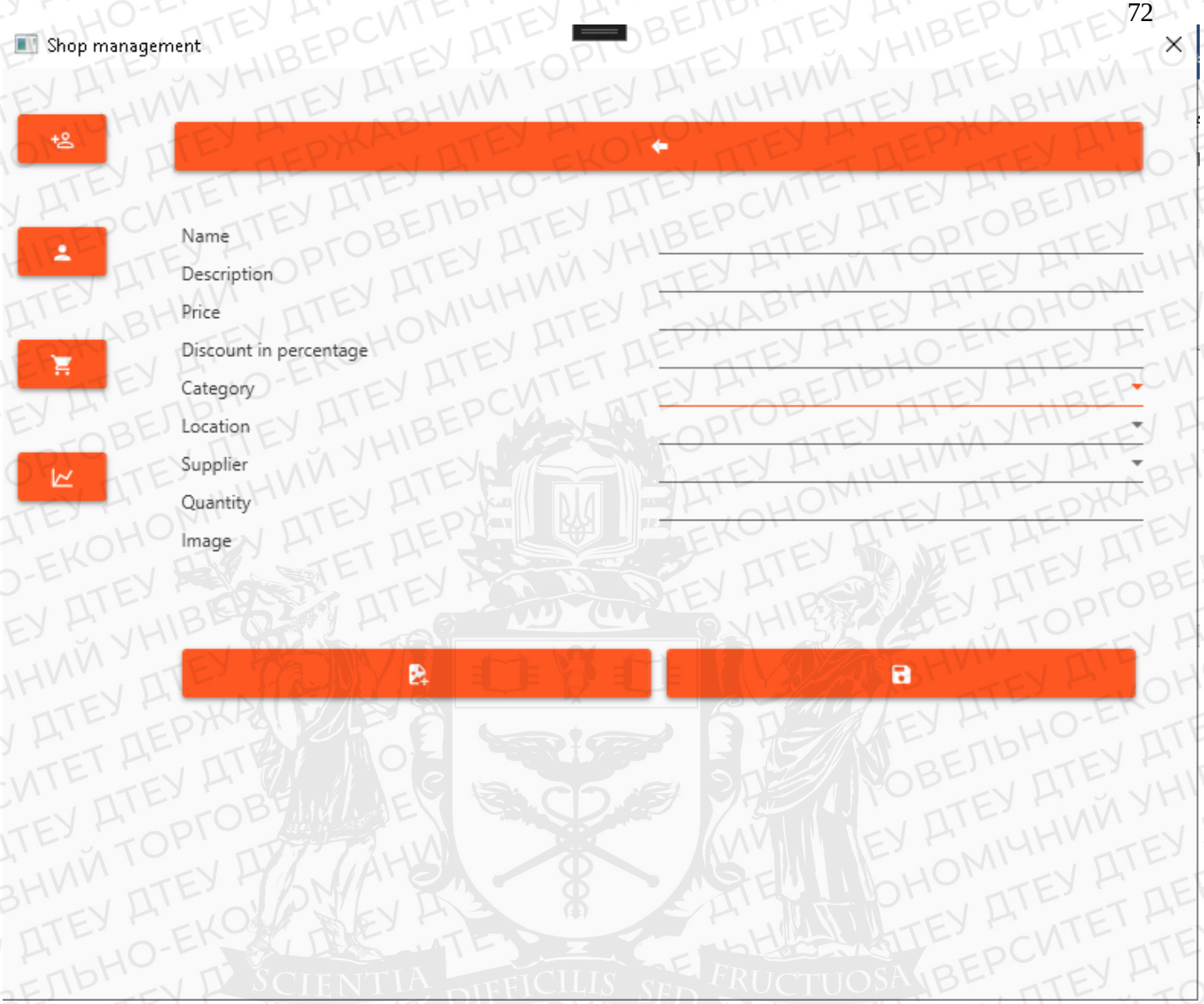


Рис.4.41 Сторінка створення нового товару

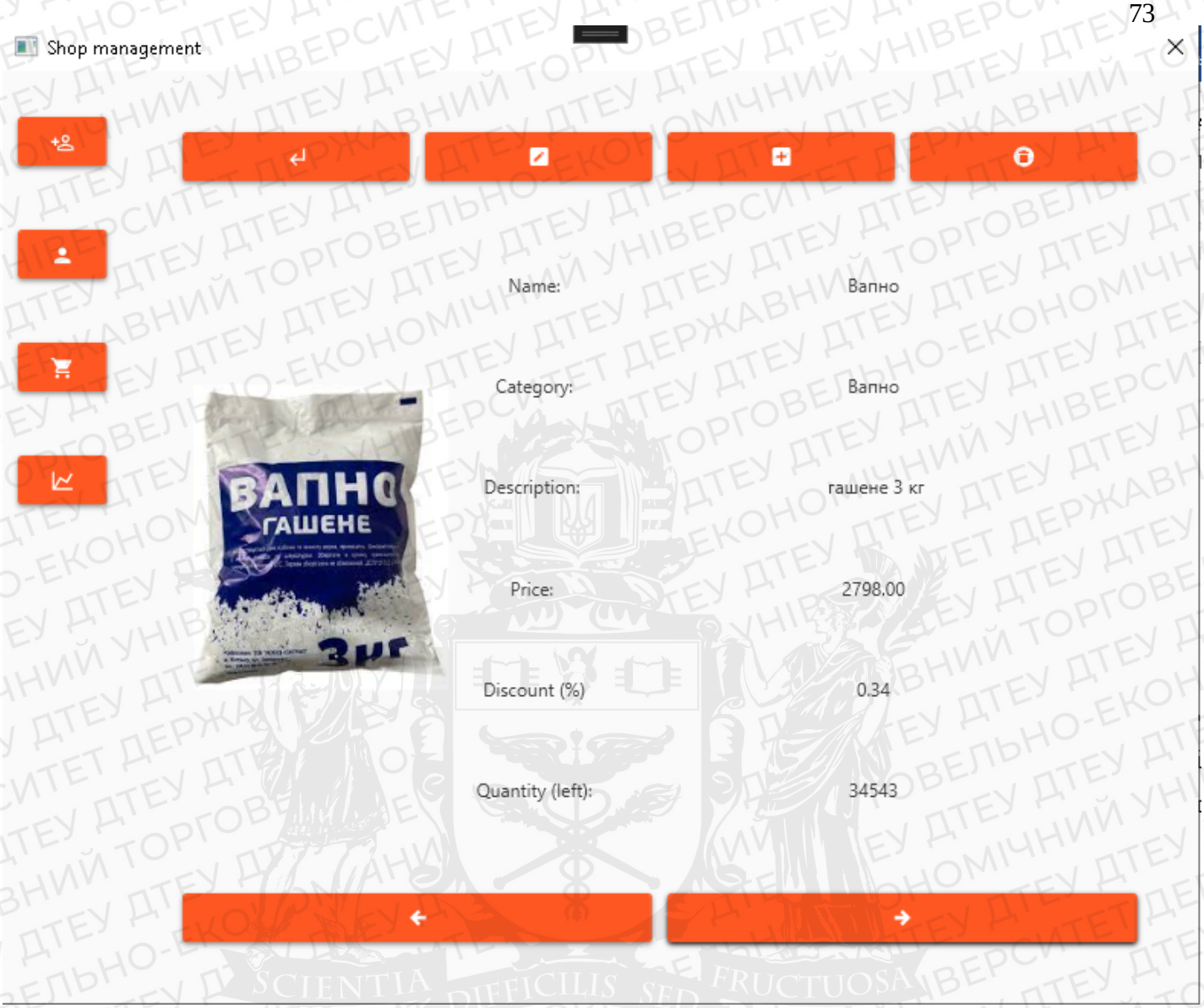


Рис 4.42 Попередній товар

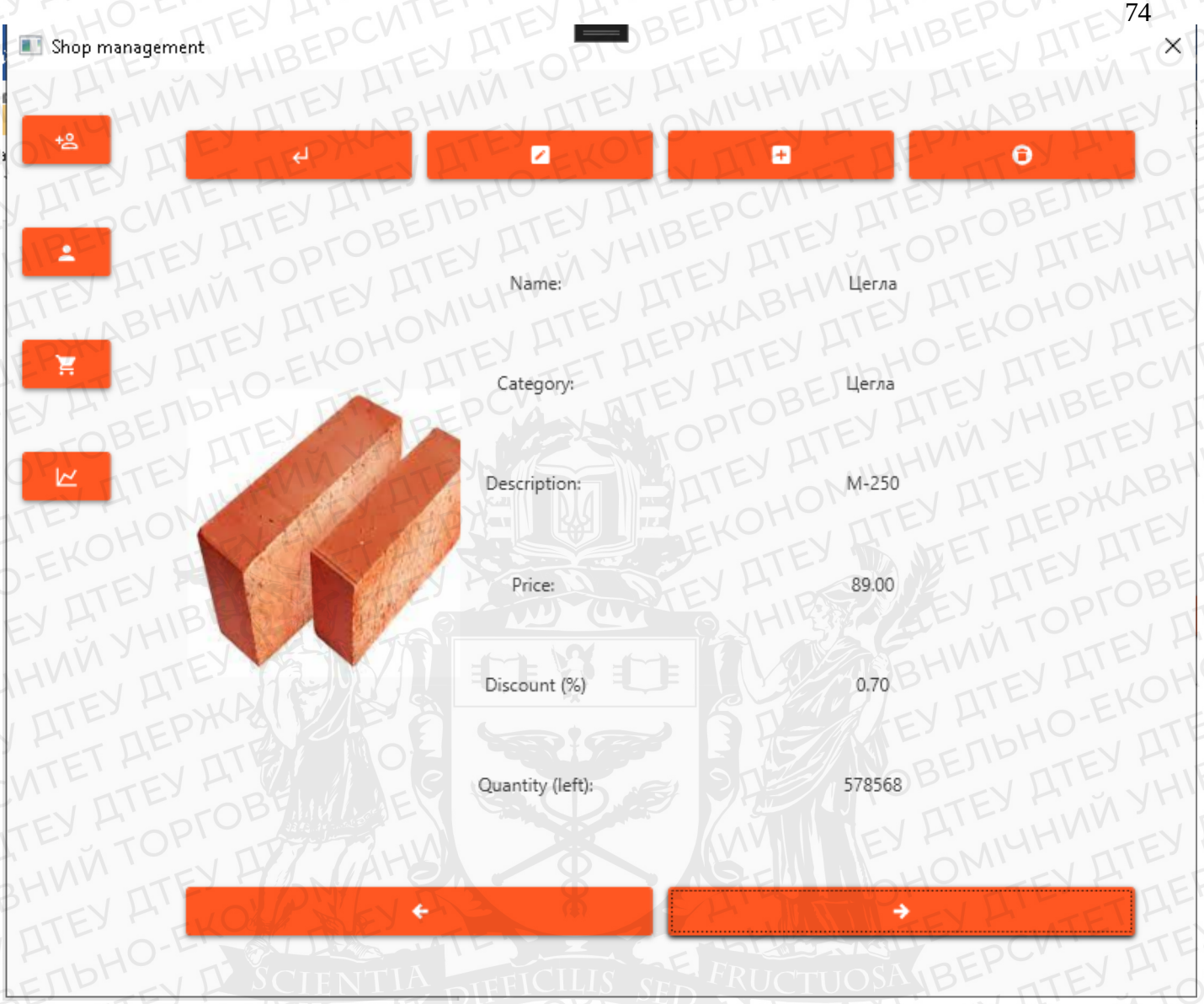


Рис. 4.43 Наступний товар

Висновки до розділу 4

Здійснено опис процедур тестування та їхніх результатів, було детально переглянуто програмне забезпечення , яке було розроблено , оцінено його функціонал та розглянуто можливості , а також вказані вимоги, дотримання яких необхідно для користування системою, описана інструкція користувача для роботи із системою.

ВИСНОВКИ

Технологію управління запасами, було вирішено удосконалити шляхом створення комплексної інформаційної системи управління підприємством, що дозволить усунути існуючі недоліки, комплексно автоматизувати бізнес-процеси торгівельного підприємства ПП «Давос», побудувати цілісний інформаційний простір, та забезпечити додаткові вигоди.

Було розглянуто теоретичні аспекти управління матеріальними запасами в торгівлі , як комплекс заходів, спрямований на забезпечення оптимальної кількості й видів фізичних ресурсів, необхідних для реалізації стратегічного плану організації.

Напрямком удосконалення системи управління запасами підприємства є оптимізація управління продукцією, співпраця з дистриб'юторами та оптимізація торгівельної системи ПП «Давос». У цьому контексті було досліджено технологію здійснення управління матеріальними запасами.

Відзначено, що ефективне управління запасами стає ключовим елементом успішної торговельної діяльності, особливо в умовах посиленої конкуренції та стрімкого розвитку ринків.

З метою покращення процесів управління запасами на підприємстві, була розроблена та впроваджена автоматизована система управління запасами в торгівлі з використанням мови програмування C# та MySql баз даних. Ця система розроблена з урахуванням конкретних потреб та особливостей торговельної сфери, щоб забезпечити максимальну ефективність в управлінні запасами та оптимізації витрат.

В результаті розробки та впровадження цієї системи очікується покращення управління запасами на підприємстві, зниження витрат та оптимізація робочих процесів, що в сукупності позитивно позначиться на ефективності торговельної діяльності та загальних фінансових показниках підприємства.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Азоев Г. Л. Конкурентные преимущества фирмы / Г. Л. Азоев, А.П. Челенков. – М. : Новости, 2000. – 256 с.

2. Алесинская Т. В. Основы логистики Общие вопросы логистического управления / Т. В. Алесинская. – Таганрог : Изд-во ТРТУ, 2005. – 121 с.

3. Алькема В. Г. Логістика: навчальний посібник / В. Г. Алькема, О. М. Сумець. – К. : ВД «Професіонал», 2008. – 272 с.

4. Аникин Б. А. Аутсорсинг : создание высокоэффективных и конкурентоспособных организаций : учебное пособие / Б. А. Аникин. – М. : ИНФРА–М, 2003. – 186 с.

5. Аникин Б. А. Логистика : учебное пособие. / Б. А. Аникин. – М. : ИНФРА–М, 2011. – 326 с.

6. Ансоф І. Стратегічне управління: підручник / І. Ансоф. – К. : ВД «Товариство», 2009. – 520 с.

7. Балабанова Л. В. Комерційна діяльність: маркетинг і логістика: навчальний посібник / Л. В. Балабанова, А.М. Германчук. – К. : Професіонал, 2004. – 143 с.

8. Банько В. Г. Логістика: навчальний посібник / В. Г. Банько. – К. : КНТ, 2007. – 435 с.

9. Белінський П. І. Менеджмент виробництва та операцій : навч. посібник / П. І. Белінський. – К. : Центр навчальної літератури, 2005. – 624 с.

10. Бєлявцев М. І. Маркетинг : навчальний посібник. / М. І. Бєлявцев, Л. М. Іваненко. – К. : ЦНЛ, 2005. – 328 с.

11. Бочкарев А. А. Планирование и моделирование цепи поставок : учебно-практическое пособие / А. А. Бочкарев. – М. : Издательство «Альфа-Пресс», 2008. – 192 с.

12. Вертегел А. В. Пути и факторы повышения конкурентоспособности предприятий в Украине / А. В. Вертегел, Е. В. Коваленко // Матеріали XIX

науково-технічної конференції студентів, магістрантів, аспірантів і викладачів ЗДІА, 22-25 квітня 2014 р. – Запоріжжя, 2014. – Т. IV. – С. 10.

13. Алекс Макки Введение в .NET 4.0 и Visual Studio 2010 для профессионалов = Introducing .NET 4.0: with Visual Studio 2010. — М.: «Вильямс», 2010. — С. 416. — ISBN 978-5-8459-1639-6 11.

14. Кен Хендерсон Професійне керівництво з SQL Server: структура та реалізація. — М.: Издательский дом «Вильямс», 2006. — С. 1056. ISBN 5- 8459-0912-0

15. Барсегян А. А., Куприянов М. С., Степаненко В. В., Холод И. И. Методы и модели анализа данных: OLAP и Data Mining. – СПб.: БХВ-Петербург, 2004. – 336 с.

16. Майо Д. Самоучитель Microsoft Visual Studio 2010 = Microsoft Visual Studio 2010: A Beginner’s Guide (A Beginners Guide). — С.: «БХВПетербург», 2010. — С. 464. — ISBN 978-5-9775-0609-0

17. О.І. Пурський, . П.Г. Демідов, Г.Т. Самойленко, А.В. Селіванова. МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ДО НАПИСАННЯ ВИПУСКНОГО КВАЛІФІКАЦІЙНОГО ПРОЕКТУ від 27 квітня 2020, протокол № 17

ДОДАТКИ

Додаток А

Класи ПЗ

AuthorizationService

```
using System.Threading.Tasks;
using MediatR;
using ShopManager.Application.CQRS.Seller.Login;
using ShopManager.Domain.Entities.Database;

namespace ShopManager.Services;

public interface IAuthorizationService
{
    public Seller? AuthorizedSeller { get; internal set; }
    bool IsAuthorized => AuthorizedSeller is not null;

    ValueTask<bool> AuthorizeSeller(string login, string password);
}

public sealed class AuthorizationService : IAuthorizationService
{
    private readonly IMediator mediator;

    public AuthorizationService(IMediator mediator)
    {
        this.mediator = mediator;
    }

    public async ValueTask<bool> AuthorizeSeller(string login, string password)
    {
        var command = new LoginSellerQuery(login, password);
        var isAuthorized = await mediator.Send(command);
        if (isAuthorized)
        {
            AuthorizedSeller = await mediator.Send(new GetSellerByLoginQuery(login));
            return true;
        }

        return false;
    }

    public Seller? AuthorizedSeller { get; set; }
        }
```

Клас PageService

```
using System;
using System.Runtime.CompilerServices;
using System.Windows;

namespace ShopManager.Services;

/// <summary>
///     Service for retrieving pages in a WPF application.
/// </summary>
public sealed class PageService : IPageService
{
    private readonly IServiceProvider serviceProvider;

    /// <summary>
    ///     Initializes a new instance of the <see cref="PageService" /> class.
    /// </summary>
    /// <param name="serviceProvider">The service provider used to retrieve pages.</param>
    public PageService(IServiceProvider serviceProvider)
    {
```

```
        this.serviceProvider = serviceProvider;
    }

    /// <inheritdoc />
    public FrameworkElement GetPage(Type pageType)
    {
        return Unsafe.As<FrameworkElement>(serviceProvider.GetService(pageType));
    }

    /// <inheritdoc />
    public FrameworkElement GetPage<TPage>() where TPage : FrameworkElement
    {
        return Unsafe.As<FrameworkElement>(serviceProvider.GetService(typeof(TPage)));
    }
}

/// <summary>
///     Interface for a service that retrieves pages in a WPF application.
/// </summary>
public interface IPageService
{
    /// <summary>
    ///     Gets a page of the specified type.
    /// </summary>
    /// <param name="pageType">The type of the page to retrieve.</param>
    /// <returns>The retrieved page as a <see cref="FrameworkElement" />.</returns>
    FrameworkElement GetPage(Type pageType);

    /// <summary>
    ///     Gets a page of the specified generic type.
    /// </summary>
    /// <typeparam name="TPage">The type of the page to retrieve.</typeparam>
    /// <returns>The retrieved page as a <see cref="FrameworkElement" />.</returns>
    FrameworkElement GetPage<TPage>() where TPage : FrameworkElement;
        }
```

Клас CreateCategoryPageViewModel

```
using System;
using System.IO;
using System.Threading.Tasks;
using System.Windows.Media.Imaging;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using MediatR;
using Microsoft.Win32;
using Notification.Wpf;
using ShopManager.Application.CQRS.Category.Commands;
using ShopManager.Helper;
using ShopManager.Pages.Category;

namespace ShopManager.Viewmodels.Category;

using DbCategory = Domain.Entities.Database.Category;

public partial class CreateCategoryPageViewModel : ObservableObject
{
    private readonly IMediator _mediator;
    private readonly INotificationManager _notificationManager;

    [ObservableProperty] private BitmapImage? _categoryImage;

    [ObservableProperty] private string? _description;

    [ObservableProperty] private string? _imagePath;

    [ObservableProperty] private string _name;

    public CreateCategoryPageViewModel(INotificationManager notificationManager, IMediator mediator)
    {
        _notificationManager = notificationManager;
```

```
    _mediator = mediator;
}

[RelayCommand]
private async Task OkButtonClicked()
{
    if (string.IsNullOrEmpty(Name))
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = "Please enter a name",
            Type = NotificationType.Error
        });
        return;
    }

    var createCategoryCommand = new CreateCategoryCommand(Name, Description, null);
    var category = await _mediator.Send(createCategoryCommand);
    if (CategoryImage != null)
    {
        var newImagePath = ImageHelper.GenerateImagePath(category);
        // Copy image to the path
        var oldImagePath = CategoryImage.UriSource.LocalPath;
        File.Copy(oldImagePath, newImagePath, true);
        var updateCategoryCommand = new UpdateCategoryCommand
        {
            CategoryId = category.Id,
            ImagePath = newImagePath
        };

        var updatedCategory = await _mediator.Send(updateCategoryCommand);
        if (updatedCategory.ImagePath != newImagePath)
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Failed to update category image",
                Type = NotificationType.Error
            });
            return;
        }
    }

    _notificationManager.Show(new NotificationContent
    {
        Title = "Success",
        Message = "Category created successfully",
        Type = NotificationType.Success
    });


    UiHelper.NavigateToPage(typeof(ViewAllCategoriesPage));
}

[RelayCommand]
private void OpenFileButtonClicked()
{
    OpenFileDialog openFileDialog = new()
    {
        Filter = "Image files (*.png;*.jpeg)|*.png;*.jpeg|All files (*.*)|*.*"
    };

    if (openFileDialog.ShowDialog() != true) return;

    var fileName = openFileDialog.FileName;
    if (fileName is null) return;

    try
    {
        var image = new BitmapImage();
        image.BeginInit();
```

```
            image.UriSource = new Uri(fileName);
            image.EndInit();
            CategoryImage = image;
        }
        catch (Exception)
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Please select a valid image",
                Type = NotificationType.Error
            });
        }
    }

    [RelayCommand]
    private void NavigateBack()
    {
        UiHelper.NavigateToPage(typeof(ViewAllCategoriesPage));
    }
    }
```

Клас EditCategoryPageViewModel

```
using System.Threading.Tasks;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using MediatR;
using Notification.Wpf;
using ShopManager.Application.CQRS.Category.Commands;
using ShopManager.Helper;
using ShopManager.Pages.Category;

namespace ShopManager.Viewmodels.Category;

using DbCategory = Domain.Entities.Database.Category;

public partial class EditCategoryPageViewModel : ObservableObject
{
    private readonly IMediator _mediator;
    private readonly INotificationManager _notificationManager;
    [ObservableProperty] private DbCategory _category;

    public EditCategoryPageViewModel(IMediator mediator, INotificationManager notificationManager)
    {
        _mediator = mediator;
        _notificationManager = notificationManager;
    }


    [RelayCommand]
    private async Task UpdateCategoryClicked()
    {
        if (string.IsNullOrEmpty(Category.Name))
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Category name cannot be empty",
                Type = NotificationType.Error
            });
            return;
        }

        var cmd = new UpdateCategoryCommand
        {
            CategoryId = Category.Id,
            Name = Category.Name,
            Description = Category.Description
        };
        var updatedCategory = await _mediator.Send(cmd);
        if (updatedCategory is null)
        {
```

```
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Category could not be updated",
                Type = NotificationType.Error
            });
            return;
        }

        _notificationManager.Show(new NotificationContent
        {
            Title = "Success",
            Message = "Category updated successfully",
            Type = NotificationType.Success
        });
        UiHelper.NavigateToPage(typeof(ViewAllCategoriesPage));
    }

    [RelayCommand]
    private void Cancel()
    {
        UiHelper.NavigateToPage(typeof(ViewAllCategoriesPage));
    }
    }
```

Клас ViewAllCategoriesPageViewModel

```
using System;
using System.Collections.ObjectModel;
using System.Linq;
using System.Threading.Tasks;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using MediatR;
using Notification.Wpf;
using ShopManager.Application.CQRS.Category.Commands;
using ShopManager.Application.CQRS.Category.Queries;
using ShopManager.Domain.Extensions;
using ShopManager.Helper;
using ShopManager.Pages.Category;

namespace ShopManager.Viewmodels.Category;

using DbCategory = Domain.Entities.Database.Category;

public partial class ViewAllCategoriesPageViewModel : ObservableObject
{
    private readonly INotificationManager _manager;

    private readonly IMediator _mediator;
    [ObservableProperty] private ObservableCollection<DbCategory> _categories;
    [ObservableProperty] private DbCategory _selectedCategory;

    public ViewAllCategoriesPageViewModel(IMediator mediator, INotificationManager manager)
    {
        _mediator = mediator;
        _manager = manager;
        OnInit();
    }

    private async void OnInit()
    {
        try
        {
            var categories = await _mediator.Send(new GetAllCategoriesQuery());
            Categories = categories.ToObservableCollection();
            SelectedCategory = Categories.FirstOrDefault();
        }
        catch (Exception)
        {
            _manager.Show(new NotificationContent
            {
                Title = "Error",
```

```
            Message = "Failed to load categories",
            Type = NotificationType.Error
        });
    }
}

[RelayCommand]
private void OpenEditCategoryPage()
{
    if (SelectedCategory is null)
    {
        _manager.Show(new NotificationContent
        {
            Title = "Error",
            Message = "Please select a customer to edit",
            Type = NotificationType.Error
        });
        return;
    }

    var pageType = typeof(EditCategoryPage);
    UiHelper.NavigateToPage(pageType);
    var currentPage = UiHelper.GetCurrentPage<EditCategoryPage>();
    currentPage.ViewModel.Category = SelectedCategory;
}


[RelayCommand]
private async Task DeleteCategory()
{
    if (SelectedCategory is null)
    {
        _manager.Show(new NotificationContent
        {
            Title = "Error",
            Message = "Please select a category to delete",
            Type = NotificationType.Error
        });
        return;
    }

    var result = await _mediator.Send(new DeleteCategoryCommand(SelectedCategory.Id));
    if (!result)
    {
        _manager.Show(new NotificationContent
        {
            Title = "Error",
            Message = "Failed to delete category",
            Type = NotificationType.Error
        });
        return;
    }

    Categories.Remove(SelectedCategory);
    SelectedCategory = Categories.FirstOrDefault();

    _manager.Show(new NotificationContent
    {
        Title = "Success",
        Message = "Successfully deleted category",
        Type = NotificationType.Success
    });
}

[RelayCommand]
private void NextCategory()
{
    if (SelectedCategory is null) return;

    var currentIndex = Categories.IndexOf(SelectedCategory);
    if (currentIndex < Categories.Count - 1) SelectedCategory = Categories[currentIndex + 1];
}
```

```
    [RelayCommand]
    private void PreviousCategory()
    {
        if (SelectedCategory == null) return;

        var currentIndex = Categories.IndexOf(SelectedCategory);
        if (currentIndex > 0) SelectedCategory = Categories[currentIndex - 1];
    }


    [RelayCommand]
    private void NavigateToPage(Type pageType)
    {
        UiHelper.NavigateToPage(pageType);
    }
    }
```

Клас CreateCustomerPageViewModel

```
using System;
using System.Diagnostics;
using System.IO;
using System.Threading.Tasks;
using System.Windows.Media.Imaging;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using MediatR;
using Microsoft.Win32;
using Notification.Wpf;
using ShopManager.Application.CQRS.Customer.Commands;
using ShopManager.Helper;
using ShopManager.Pages.Customer;

namespace ShopManager.Viewmodels.Customer;

public partial class CreateCustomerPageViewModel : ObservableObject
{
    private readonly IMediator _mediator;
    private readonly INotificationManager _notificationManager;
    [ObservableProperty] private BitmapImage _customerImage;
    [ObservableProperty] private string _name;
    [ObservableProperty] private string _phoneNumber;

    public CreateCustomerPageViewModel(INotificationManager notificationManager, IMediator mediator)
    {
        _notificationManager = notificationManager;
        _mediator = mediator;
    }

    [RelayCommand]
    private void NavigateBack()
    {
        UiHelper.NavigateToPage(typeof(ViewAllCustomerPage));
    }

    [RelayCommand]
    private void OpenFileButtonClicked()
    {
        OpenFileDialog openFileDialog = new()
        {
            Filter = "Image files (*.png;*.jpeg)|*.png;*.jpeg|All files (*.*)|*.*"
        };

        if (openFileDialog.ShowDialog() != true) return;

        var fileName = openFileDialog.FileName;
        if (fileName == null) return;

        try
        {
            var image = new BitmapImage();
            image.BeginInit();
```

```
            image.UriSource = new Uri(fileName);
            image.EndInit();
            CustomerImage = image;
        }
        catch (Exception ex)
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = $"Please select a valid image | {ex.Message}",
                Type = NotificationType.Error
            });
        }
    }

    [RelayCommand]
    private async Task OkButtonClicked()
    {
        // Check if name is empty
        if (string.IsNullOrEmpty(Name))
        {
            _notificationManager
                .Show(new NotificationContent
                {
                    Title = "Error",
                    Message = "Please enter a name",
                    Type = NotificationType.Error
                });
            return;
        }

        if (string.IsNullOrEmpty(PhoneNumber))
        {
            _notificationManager
                .Show(new NotificationContent
                {
                    Title = "Error",
                    Message = "Please enter an address",
                    Type = NotificationType.Error
                });
            return;
        }

        try
        {
            var createdLocation = await _mediator.Send(new CreateCustomerCommand
            {
                Name = Name,
                PhoneNumber = PhoneNumber,
                ImagePath = null
            });
            if (CustomerImage != null)
            {
                var newImagePath = ImageHelper.GenerateImagePath(createdLocation);
                // Copy image to the path
                var oldImagePath = CustomerImage.UriSource.LocalPath;
                File.Copy(oldImagePath, newImagePath, true);
                Debug.WriteLine(newImagePath);
                var updatedLocation = await _mediator.Send(new UpdateCustomerCommand
                {
                    CustomerId = createdLocation.Id,
                    ImagePath = newImagePath
                });

                if (updatedLocation.ImagePath != newImagePath)
                    throw new InvalidOperationException("Image path not updated");
            }

            _notificationManager.Show(new NotificationContent
            {
                Title = "Success",
                Message = "Customer created successfully",
```

```
            Type = NotificationType.Success
        });

        UiHelper.NavigateToPage(typeof(ViewAllCustomerPage));
    }
    catch (Exception ex)
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = $"An error occurred while creating the location: {ex.Message}",
            Type = NotificationType.Error
        });
    }
}
 }
```

Клас EditCustomerPageViewModel

```
using System;
using System.Threading.Tasks;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using MediatR;
using Notification.Wpf;
using ShopManager.Application.CQRS.Customer.Commands;
using ShopManager.Helper;
using ShopManager.Pages.Customer;
using DbCustomer = ShopManager.Domain.Entities.Database.Customer;

namespace ShopManager.Viewmodels.Customer;

public partial class EditCustomerPageViewModel : ObservableObject
{
    private readonly IMediator _mediator;

    private readonly INotificationManager _notificationManager;

    [ObservableProperty] private DbCustomer _customer;

    public EditCustomerPageViewModel(INotificationManager notificationManager, IMediator mediator)
    {
        _notificationManager = notificationManager;
        _mediator = mediator;
    }

    [RelayCommand]
    private void Cancel()
    {
        UiHelper.NavigateToPage(typeof(ViewAllCustomerPage));
    }

    [RelayCommand]
    private async Task Save()
    {
        if (string.IsNullOrEmpty(Customer.Name))
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Please enter a name",
                Type = NotificationType.Error
            });
            return;
        }

        if (string.IsNullOrEmpty(Customer.PhoneNumber))
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Please enter a phone number",
                Type = NotificationType.Error
            });
```

```
            return;
        }

        try
        {
            var updatedCustomer = await _mediator.Send(new UpdateCustomerCommand
            {
                CustomerId = Customer.Id,
                Name = Customer.Name,
                PhoneNumber = Customer.PhoneNumber
            });

            if (updatedCustomer is null)
            {
                _notificationManager.Show(new NotificationContent
                {
                    Title = "Error",
                    Message = "Customer not found",
                    Type = NotificationType.Error
                });
                return;
            }

            UiHelper.NavigateToPage(typeof(ViewAllCustomerPage));
        }
        catch (Exception ex)
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = ex.Message,
                Type = NotificationType.Error
            });
        }
    }
    }
```

Клас ViewAllCustomerPageViewModel :

```
using System;
using System.Collections.ObjectModel;
using System.Linq;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using MediatR;
using Notification.Wpf;
using ShopManager.Application.CQRS.Customer.Commands;
using ShopManager.Application.CQRS.Customer.Queries;
using ShopManager.Helper;
using ShopManager.Pages.Customer;
using DbCustomer = ShopManager.Domain.Entities.Database.Customer;

namespace ShopManager.Viewmodels.Customer;

public partial class ViewAllCustomerPageViewModel : ObservableObject
{
    private readonly IMediator _mediator;
    private readonly INotificationManager _notificationManager;
    [ObservableProperty] private ObservableCollection<DbCustomer> _customers;

    [ObservableProperty] private DbCustomer _selectedCustomer;

    public ViewAllCustomerPageViewModel(IMediator mediator, INotificationManager notificationManager)
    {
        _mediator = mediator;
        _notificationManager = notificationManager;
        OnInit();
    }

    private async void OnInit()
    {
```

```
    var customers = await _mediator.Send(new GetAllCustomerQuery());
    Customers = new ObservableCollection<DbCustomer>(customers);
    SelectedCustomer = Customers.FirstOrDefault();
}

[RelayCommand]
private void NextCustomer()
{
    if (SelectedCustomer is null) return;

    var currentIndex = Customers.IndexOf(SelectedCustomer);
    if (currentIndex < Customers.Count - 1) SelectedCustomer = Customers[currentIndex + 1];
}

[RelayCommand]
private void PreviousCustomer()
{
    if (SelectedCustomer == null) return;

    var currentIndex = Customers.IndexOf(SelectedCustomer);
    if (currentIndex > 0) SelectedCustomer = Customers[currentIndex - 1];
}

[RelayCommand]
private void DeleteCustomer()
{
    if (SelectedCustomer is null)
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = "Please select a customer to delete",
            Type = NotificationType.Error
        });
        return;
    }

    _notificationManager.ShowButtonWindow("Delete customer?", "Confirmation", () =>
    {
        _mediator.Send(new DeleteCustomerCommand
        {
            CustomerId = SelectedCustomer.Id
        });
        Customers.Remove(SelectedCustomer);
        SelectedCustomer = Customers.FirstOrDefault();
    }, "Confirm");
}

[RelayCommand]
private void OpenEditCustomerPage()
{
    if (SelectedCustomer is null)
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = "Please select a customer to edit",
            Type = NotificationType.Error
        });
        return;
    }

    var pageType = typeof(EditCustomerPage);
    UiHelper.NavigateToPage(pageType);
    var currentPage = UiHelper.GetCurrentPage<EditCustomerPage>();
    currentPage.ViewModel.Customer = SelectedCustomer;
}

[RelayCommand]
private void NavigateToPage(Type pageType)
{
    UiHelper.NavigateToPage(pageType);
```

```
    }
}
```

Клас CreateLocationPageViewModel

```
using System;
using System.Diagnostics;
using System.IO;
using System.Threading.Tasks;
using System.Windows.Media.Imaging;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using MediatR;
using Microsoft.Win32;
using Notification.Wpf;
using ShopManager.Application.CQRS.Location.Commands;
using ShopManager.Helper;
using ShopManager.Pages.Location;

namespace ShopManager.Viewmodels.Location;

public partial class CreateLocationPageViewModel : ObservableObject
{
    private readonly IMediator _mediator;
    private readonly INotificationManager _notificationManager;
    [ObservableProperty] private string _address;
    [ObservableProperty] private string? _description;

    [ObservableProperty] private BitmapImage _locationImage;
    [ObservableProperty] private string _name;

    public CreateLocationPageViewModel(INotificationManager notificationManager, IMediator mediator)
    {
        _notificationManager = notificationManager;
        _mediator = mediator;
    }

    [RelayCommand]
    private async Task OkButtonClicked()
    {
        // Check if name is empty
        if (string.IsNullOrEmpty(Name))
        {
            _notificationManager
                .Show(new NotificationContent
                {
                    Title = "Error",
                    Message = "Please enter a name",
                    Type = NotificationType.Error
                });
            return;
        }

        if (string.IsNullOrEmpty(Address))
        {
            _notificationManager
                .Show(new NotificationContent
                {
                    Title = "Error",
                    Message = "Please enter an address",
                    Type = NotificationType.Error
                });
            return;
        }

        try
        {
            var createdLocation = await _mediator.Send(new CreateLocationCommand
```

```
        {
            Address = Address,
            Name = Name,
            Description = Description,
            ImagePath = null
        });
        if (LocationImage != null)
        {
            var newImagePath = ImageHelper.GenerateImagePath(createdLocation);
            // Copy image to the path
            var oldImagePath = LocationImage.UriSource.LocalPath;
            File.Copy(oldImagePath, newImagePath, true);
            Debug.WriteLine(newImagePath);
            var updatedLocation = await _mediator.Send(new UpdateLocationCommand
            {
                LocationId = createdLocation.Id,
                ImagePath = newImagePath
            });

            if (updatedLocation.ImagePath != newImagePath)
                throw new InvalidOperationException("Image path not updated");
        }

        _notificationManager.Show(new NotificationContent
        {
            Title = "Success",
            Message = "Location created successfully",
            Type = NotificationType.Success
        });

        UiHelper.NavigateToPage(typeof(ViewAllLocationsPage));
    }
    catch (Exception ex)
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = $"An error occurred while creating the location: {ex.Message}",
            Type = NotificationType.Error
        });
    }
}

[RelayCommand]
private void NavigateBack()
{
    UiHelper.NavigateToPage(typeof(ViewAllLocationsPage));
}

[RelayCommand]
private void OpenFileButtonClicked()
{
    OpenFileDialog openFileDialog = new()
    {
        Filter = "Image files (*.png;*.jpeg)|*.png;*.jpeg|All files (*.*)|*.*"
    };

    if (openFileDialog.ShowDialog() != true) return;

    var fileName = openFileDialog.FileName;
    if (fileName == null) return;

    try
    {
        var image = new BitmapImage();
        image.BeginInit();
        image.UriSource = new Uri(fileName);
        image.EndInit();
        LocationImage = image;
    }
    catch (Exception)
    {
```

```
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Please select a valid image",
                Type = NotificationType.Error
            });
        }
    }
    }
```

Клас EditLocationPageViewModel

```
using System;
using System.Threading.Tasks;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using MediatR;
using Notification.Wpf;
using ShopManager.Application.CQRS.Location.Commands;
using ShopManager.Helper;
using ShopManager.Pages.Location;
// Измените на соответствующий namespace
using DbLocation = ShopManager.Domain.Entities.Database.Location; // Измените на соответствующий namespace

namespace ShopManager.ViewModels.Location;
// Измените на соответствующий namespace

public partial class EditLocationPageViewModel : ObservableObject
{
    private readonly IMediator _mediator;

    private readonly INotificationManager _notificationManager;

    [ObservableProperty] private DbLocation _location;

    public EditLocationPageViewModel(INotificationManager notificationManager, IMediator mediator)
    {
        _notificationManager = notificationManager;
        _mediator = mediator;
    }

    [RelayCommand]
    private void Cancel()
    {
        UiHelper.NavigateToPage(typeof(ViewAllLocationsPage));
    }

    [RelayCommand]
    private async Task Save()
    {
        if (string.IsNullOrEmpty(Location.Name))
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Name is required",
                Type = NotificationType.Error
            });
            return;
        }

        if (string.IsNullOrEmpty(Location.Address))
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Address is required",
                Type = NotificationType.Error
            });
            return;
        }
```

```
    try
    {
        var updatedLocation = await _mediator.Send(new UpdateLocationCommand() // Изменено на UpdateLocationCommand
        {
            Description = Location.Description,
            Name = Location.Name,
            LocationId = Location.Id // Изменено на LocationId
            // Добавьте другие необходимые поля
        });

        if (updatedLocation is null)
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Location not found", // Изменено на Location
                Type = NotificationType.Error
            });
            return;
        }

        UiHelper.NavigateToPage(typeof(ViewAllLocationsPage)); // Измените на соответствующую страницу
    }
    catch (Exception ex)
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = ex.Message,
            Type = NotificationType.Error
        });
    }
}
    }
```

Клас ViewAllLocationsPageViewModel

```
using System;
using System.Collections.ObjectModel;
using System.Linq;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using MediatR;
using Notification.Wpf;
using ShopManager.Application.CQRS.Location.Commands;
using ShopManager.Application.CQRS.Location.Queries;
using ShopManager.Helper;
using ShopManager.Pages.Location;
using DbLocation = ShopManager.Domain.Entities.Database.Location;

namespace ShopManager.Viewmodels.Location;

public partial class ViewAllLocationsPageViewModel : ObservableObject
{
    private readonly IMediator _mediator;
    private readonly INotificationManager _notificationManager;

    [ObservableProperty] private ObservableCollection<DbLocation> _locations;
    [ObservableProperty] private DbLocation _selectedLocation;

    public ViewAllLocationsPageViewModel(IMediator mediator, INotificationManager notificationManager)
    {
        _mediator = mediator;
        _notificationManager = notificationManager;
        OnInit();
    }

    private async void OnInit()
    {
        var locations = await _mediator.Send(new GetAllLocationsQuery());
```

```
    Locations = new ObservableCollection<DbLocation>(locations);
    SelectedLocation = Locations.FirstOrDefault();
}

[RelayCommand]
private void NextLocation()
{
    if (SelectedLocation is null) return;

    var currentIndex = Locations.IndexOf(SelectedLocation);
    if (currentIndex < Locations.Count - 1) SelectedLocation = Locations[currentIndex + 1];
}

[RelayCommand]
private void PreviousLocation()
{
    if (SelectedLocation != null)
    {
        var currentIndex = Locations.IndexOf(SelectedLocation);
        if (currentIndex > 0) SelectedLocation = Locations[currentIndex - 1];
    }
}

[RelayCommand]
private void DeleteLocation()
{
    if (SelectedLocation is null)
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = "Please select a location to delete",
            Type = NotificationType.Error
        });
        return;
    }

    _notificationManager.ShowButtonWindow("Delete location?", "Confirmation", () =>
    {
        _mediator.Send(new DeleteLocationCommand
        {
            LocationId = SelectedLocation.Id
        });
        Locations.Remove(SelectedLocation);
        SelectedLocation = Locations.FirstOrDefault();
    }, "Confirm");
}

[RelayCommand]
private void OpenEditLocationPage()
{
    if (SelectedLocation is null)
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = "Please select a location to edit",
            Type = NotificationType.Error
        });
        return;
    }

    var pageType = typeof(EditLocationPage);
    UiHelper.NavigateToPage(pageType);
    var currentPage = UiHelper.GetCurrentPage<EditLocationPage>();
    currentPage.ViewModel.Location = SelectedLocation;
}

[RelayCommand]
private void NavigateToPage(Type pageType)
{
    UiHelper.NavigateToPage(pageType);
```

```
}
    }
```

Клас AuthorizedSellerPageViewModel

```
using System;
using System.IO;
using System.Threading.Tasks;
using System.Windows;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using MediatR;
using Microsoft.Win32;
using Notification.Wpf;
using ShopManager.Helper;
using ShopManager.Services;
using DbSeller = ShopManager.Domain.Entities.Database.Seller;

namespace ShopManager.Viewmodels.Navigation;

public partial class AuthorizedSellerPageViewModel : ObservableObject
{
    private readonly IAuthorizationService authorizationService;
    private readonly IMediator mediator;
    private readonly INotificationManager notificationManager;

    public AuthorizedSellerPageViewModel(IAuthorizationService authorizationService, IMediator mediator,
        INotificationManager notificationManager)
    {
        this.authorizationService = authorizationService;
        this.mediator = mediator;
        this.notificationManager = notificationManager;
    }

    public DbSeller AuthorizedSeller => authorizationService.AuthorizedSeller!;

    [RelayCommand]
    private async Task BrowseImage()
    {
        try
        {
            OpenFileDialog openFileDialog = new()
            {
                Filter = "Image files (*.png;*.jpeg)|*.png;*.jpeg|All files (*.*)|*.*"
            };

            var dialogResult = openFileDialog.ShowDialog();

            if (!dialogResult.Value) return;

            var fileName = openFileDialog.FileName;
            if (fileName == null) return;

            // Check if image exists in path
            var imagePath = ImageHelper.GenerateImagePath(authorizationService.AuthorizedSeller);

            if (File.Exists(imagePath)) File.Delete(imagePath);


            // Copy image to path
            File.Copy(fileName, imagePath, true);
        }
        catch (Exception ex)
        {
            Clipboard.SetData(DataFormats.Text, ex.Message);
            notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = $"Error occured | {ex.Message}",
                Type = NotificationType.Error
            });
        }
    }
```

```
    }
```

Клас ManagementPageViewModel

```
using System;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using ShopManager.Helper;

namespace ShopManager.Viewmodels.Navigation;

public partial class ManagementPageViewModel : ObservableObject
{
   [RelayCommand]
   private void NavigateToPage(Type pageType)
   {
      UiHelper.NavigateToPage(pageType);
   }
      }
```

Клас CreateOrderPageViewModel

```
using System;
using System.Collections.ObjectModel;
using System.Linq;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using MediatR;
using Notification.Wpf;
using ShopManager.Application.CQRS.Basket;
using ShopManager.Application.CQRS.Customer.Queries;
using ShopManager.Application.CQRS.Location.Queries;
using ShopManager.Application.CQRS.Order.Commands;
using ShopManager.Application.CQRS.OrderProduct.Queries;
using ShopManager.Application.CQRS.Seller.Queries;
using ShopManager.Domain.Extensions;
using ShopManager.Helper;
using ShopManager.Pages;
using ShopManager.Pages.Order;
using ShopManager.Services;
using DbOrder = ShopManager.Domain.Entities.Database.Order;
using DbSeller = ShopManager.Domain.Entities.Database.Seller;
using DbCustomer = ShopManager.Domain.Entities.Database.Customer;
using DbProduct = ShopManager.Domain.Entities.Database.Product;
using DbLocation = ShopManager.Domain.Entities.Database.Location;
using DbOrderItem = ShopManager.Domain.Entities.Database.OrderProduct;

namespace ShopManager.Viewmodels.Order;

public partial class CreateOrderPageViewModel : ObservableObject
{
   private readonly IMediator _mediator;
   private readonly INotificationManager _notificationManager;
   private readonly IAuthorizationService _authorizationService;

   public CreateOrderPageViewModel(IMediator mediator, INotificationManager notificationManager, IAuthorizationService
authorizationService)
   {
      _mediator = mediator;
      _notificationManager = notificationManager;
      _authorizationService = authorizationService;
      OnInit();
   }


   private async void OnInit()
   {
      try
      {
         var customers = await _mediator.Send(new GetAllCustomerQuery());
```

```
            var locations = await _mediator.Send(new GetAllLocationsQuery());
            var products = await _mediator.Send(new GetAllOrderProductsQuery());

            Customers = customers.ToObservableCollection();
            Products = products.ToObservableCollection();
            Locations = locations.ToObservableCollection();
        }
        catch (Exception ex)
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = ex.Message,
                Type = NotificationType.Error
            });
        }
    }

    [RelayCommand]
    private void CompletePurchase()
    {
        if (!CheckForNull())
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Please select seller, customer and location",
                Type = NotificationType.Error
            });
            return;
        }

        _notificationManager.ShowButtonWindow("Create order?", "Are you sure you want to create order?",
            async () =>
            {
                var basketItems = await _mediator.Send(new GetAllProductsFromBasketQuery());
                if (!basketItems.Any())
                {
                    _notificationManager.Show(new NotificationContent
                    {
                        Title = "Error",
                        Message = "Basket is empty",
                        Type = NotificationType.Error
                    });
                    return;
                }

                var order = await _mediator.Send(new CreateOrderCommand(SelectedCustomer!.Id, _authorizationService.AuthorizedSeller.Id,
SelectedLocation!.Id, basketItems));
                _notificationManager.Show(new NotificationContent
                {
                    Title = "Success",
                    Message = $"Order created with id: {order.Id}",
                    Type = NotificationType.Success
                });
                // UiHelper.NavigateToPage(typeof(ViewAllOrdersPage));
            });
    }

    private bool CheckForNull()
    {
        return _authorizationService.AuthorizedSeller != null && SelectedCustomer != null && SelectedLocation != null;
    }

    [RelayCommand]
    private void NavigateBack()
    {
        UiHelper.NavigateToPage(typeof(ViewAllOrdersPage));
    }

    #region Collections
```

```
    [ObservableProperty] private ObservableCollection<DbSeller> _sellers;
    [ObservableProperty] private ObservableCollection<DbCustomer> _customers;
    [ObservableProperty] private ObservableCollection<DbProduct> _products;
    [ObservableProperty] private ObservableCollection<DbLocation> _locations;

    #endregion

    #region Selection properties

    [ObservableProperty] private DbCustomer? _selectedCustomer;
    [ObservableProperty] private DbProduct? _selectedProduct;
    [ObservableProperty] private DbLocation? _selectedLocation;
    [ObservableProperty] private ObservableCollection<DbProduct> _selectedProducts = new();


    [RelayCommand]
    private void OpenBasketPage() => UiHelper.NavigateToPage(typeof(BasketPage));

    [RelayCommand]
    private async void AddProductToOrder(DbProduct product)
    {
        var addToBasketCommand = new AddToBasketCommand(product);
        var itemsInTheBasket = await _mediator.Send(addToBasketCommand);
        _notificationManager.Show(new NotificationContent
        {
            Title = "Success",
            Message = $"Product added to order, items in the basket: {itemsInTheBasket}",
            Type = NotificationType.Success
        });
    }

    #endregion
    }
```

Клас OrderDetailsPageViewModel

```
using System.Collections.ObjectModel;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using LiveCharts;
using MediatR;
using ShopManager.Application.CQRS.Order.Queries;
using ShopManager.Helper;

namespace ShopManager.Viewmodels.Order;
using DbOrder = ShopManager.Domain.Entities.Database.Order;
public partial class OrderDetailsPageViewModel : ObservableObject
{
    [ObservableProperty]
    private GetOrderDetails.Model _orderDetails;


    [RelayCommand]
    private void NavigateBack() => UiHelper.NavigateToPage(typeof(Pages.Order.ViewAllOrdersPage));
}
```

Клас ViewAllOrdersPageViewModel

```
using System;
using System.Collections.ObjectModel;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
```

```
using MediatR;
using Microsoft.FeatureManagement;
using Notification.Wpf;
using Notification.Wpf.Controls;
using ShopManager.Application.CQRS.Order.Commands;
using ShopManager.Application.CQRS.Order.Queries;
using ShopManager.Configuration;
using ShopManager.Domain.Extensions;
using ShopManager.Helper;
using ShopManager.Pages.Order;

namespace ShopManager.Viewmodels.Order;

using DbOrder = ShopManager.Domain.Entities.Database.Order;

public partial class ViewAllOrdersPageViewModel : ObservableObject
{
    private readonly IFeatureManager featureManager;
    private readonly IMediator mediator;
    private readonly INotificationManager notificationManager;
    [ObservableProperty] private ObservableCollection<DbOrder> _orders;
    [ObservableProperty] private DbOrder? _selectedItem;

    public ViewAllOrdersPageViewModel(INotificationManager notificationManager,
        IMediator mediator, IFeatureManager featureManager)
    {
        this.notificationManager = notificationManager;
        this.mediator = mediator;
        this.featureManager = featureManager;
        OnInit();
    }

    [RelayCommand]
    private async void RemoveOrder()
    {
        if (!await featureManager.IsEnabledAsync(Features.Order.CanBeRemoved))
        {
            notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "You are not allowed to delete orders",
                Type = NotificationType.Error
            });
            return;
        }

        if (SelectedItem is null)
            return;
        notificationManager.ShowButtonWindow("OK",
            "Delete selected order?", async () =>
            {
                try
                {
                    var command = new DeleteOrderCommand()
                    {
                        OrderId = SelectedItem.Id
                    };
                    var res = await mediator.Send(command);
                    if (res)
                    {
                        notificationManager.Show(new NotificationContent
                        {
                            Title = "Success",
                            Message = "Order deleted",
                            Type = NotificationType.Success
                        });
                        Orders.Remove(SelectedItem);
                    }
                    else
                    {
                        notificationManager.Show(new NotificationContent
                        {
```

```
                Title = "Error",
                Message = "Order could not be deleted",
                Type = NotificationType.Error
            });
        }
    }
    catch (Exception e)
    {
        notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = e.Message,
            Type = NotificationType.Error
        });
    }
});
}

[RelayCommand]
private async Task OpenOrderDetails()
{
    if (SelectedItem is null)
    {
        notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = "No order selected",
            Type = NotificationType.Error
        });
        return;
    }

    var orderDetails = await mediator.Send(new GetOrderDetails.Query(SelectedItem.Id));
    UiHelper.NavigateToPage(typeof(OrderDetailsPage));
    UiHelper.GetCurrentPage<OrderDetailsPage>()
        .ViewModel.OrderDetails = orderDetails;
}

[RelayCommand]
private void CreateOrder()
{
    UiHelper.NavigateToPage(typeof(CreateOrderPage));
}

private async void OnInit()
{
    try
    {
        var orders = await mediator.Send(new GetAllOrdersQuery());
        Orders = orders.ToObservableCollection();
    }
    catch (Exception ex)
    {
        notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = ex.Message,
            Type = NotificationType.Error
        });
    }
}
    }
```

Клас CreateProductPageViewModel

```
using System;
using System.Collections.ObjectModel;
using System.IO;
using System.Threading.Tasks;
using System.Windows.Media.Imaging;
```

```
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using MediatR;
using Microsoft.Win32;
using Notification.Wpf;
using ShopManager.Application.CQRS.Category.Queries;
using ShopManager.Application.CQRS.Location.Queries;
using ShopManager.Application.CQRS.Product.Commands;
using ShopManager.Application.CQRS.Supplier.Queries;
using ShopManager.Domain.Extensions;
using ShopManager.Helper;
using ShopManager.Pages.Product;

namespace ShopManager.Viewmodels.Product;

using DbCategory = Domain.Entities.Database.Category;
using DbProduct = Domain.Entities.Database.Product;
using DbLocation = Domain.Entities.Database.Location;
using DbSupplier = Domain.Entities.Database.Supplier;

public partial class CreateProductPageViewModel : ObservableObject
{
    private readonly IMediator _mediator;
    private readonly INotificationManager _notificationManager;
    [ObservableProperty] private ObservableCollection<DbCategory> _categories;
    [ObservableProperty] private string? _description;
    [ObservableProperty] private string? _discountPercentage;
    [ObservableProperty] private ObservableCollection<DbLocation> _locations;
    [ObservableProperty] private string _name;
    [ObservableProperty] private string _price;
    [ObservableProperty] private BitmapImage _productImage;
    [ObservableProperty] private string _quantity;
    [ObservableProperty] private DbCategory? _selectedCategory;
    [ObservableProperty] private DbLocation? _selectedLocation;
    [ObservableProperty] private DbSupplier? _selectedSupplier;
    [ObservableProperty] private ObservableCollection<DbSupplier> _suppliers;

    public CreateProductPageViewModel(INotificationManager notificationManager, IMediator mediator)
    {
        _notificationManager = notificationManager;
        _mediator = mediator;
        OnInit();
    }

    [RelayCommand]
    private async Task CreateProduct()
    {
        if (string.IsNullOrEmpty(Name))
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Name is required",
                Type = NotificationType.Error
            });
            return;
        }

        if (string.IsNullOrEmpty(Price))
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Price is required",
                Type = NotificationType.Error
            });
            return;
        }

        if (string.IsNullOrEmpty(Quantity))
        {
            _notificationManager.Show(new NotificationContent
```

```
    {
        Title = "Error",
        Message = "Quantity is required",
        Type = NotificationType.Error
    });
    return;
}

if (SelectedCategory is null)
{
    _notificationManager.Show(new NotificationContent
    {
        Title = "Error",
        Message = "Please select a category",
        Type = NotificationType.Error
    });
    return;
}

if (SelectedLocation is null)
{
    _notificationManager.Show(new NotificationContent
    {
        Title = "Error",
        Message = "Please select a location",
        Type = NotificationType.Error
    });
    return;
}

if (SelectedSupplier is null)
{
    _notificationManager.Show(new NotificationContent
    {
        Title = "Error",
        Message = "Please select a supplier",
        Type = NotificationType.Error
    });
    return;
}

if (ProductImage is null)
{
    _notificationManager.Show(new NotificationContent
    {
        Title = "Error",
        Message = "Please select an image",
        Type = NotificationType.Error
    });
    return;
}

decimal price;
int quantity;
if (!decimal.TryParse(Price, out price))
{
    _notificationManager.Show(new NotificationContent
    {
        Title = "Error",
        Message = "Please enter a valid price",
        Type = NotificationType.Error
    });
    return;
}

if (!int.TryParse(Quantity, out quantity))
{
    _notificationManager.Show(new NotificationContent
    {
        Title = "Error",
        Message = "Please enter a valid quantity",
        Type = NotificationType.Error
```

```
        });
        return;
    }

    if (price <= 0)
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = "Price must be greater than 0",
            Type = NotificationType.Error
        });
        return;
    }

    if (quantity <= 0)
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = "Quantity must be greater than 0",
            Type = NotificationType.Error
        });
        return;
    }

    // Parse discount percentage
    decimal discountPercentage = 0;
    if (!string.IsNullOrEmpty(DiscountPercentage))
    {
        if (!decimal.TryParse(DiscountPercentage, out discountPercentage))
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Please enter a valid discount percentage",
                Type = NotificationType.Error
            });
            return;
        }

        if (discountPercentage is < 0 or > 100)
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Discount percentage must be between 0 and 100",
                Type = NotificationType.Error
            });
            return;
        }
    }

    var command = new CreateProductCommand
    {
        LocationId = SelectedLocation.Id,
        CategoryId = SelectedCategory.Id,
        SupplierId = SelectedSupplier.Id,
        Name = Name,
        Description = Description,
        Quantity = quantity,
        BasePrice = price,
        DiscountPercentage = discountPercentage / 100
    };

    var result = await _mediator.Send(command);
    if (result is null)
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = "Failed to create product",
```

```
            Type = NotificationType.Error
        });
        return;
    }

    if (ProductImage != null)
    {
        var newImagePath = ImageHelper.GenerateImagePath(result);
        // Copy image to the path
        var oldImagePath = ProductImage.UriSource.LocalPath;
        File.Copy(oldImagePath, newImagePath, true);
        var updatedLocation = await _mediator.Send(new UpdateProductImageCommand(result.Id, newImagePath));
        if (updatedLocation.ImagePath != newImagePath)
            throw new InvalidOperationException("Image path not updated");
    }

    _notificationManager.Show(new NotificationContent
    {
        Title = "Success",
        Message = "Successfully created product",
        Type = NotificationType.Success
    });

    UiHelper.NavigateToPage(typeof(ViewAllProductsPage));
}

[RelayCommand]
private void NavigateBack()
{
    UiHelper.NavigateToPage(typeof(ViewAllProductsPage));
}


[RelayCommand]
private void OpenImage()
{
    OpenFileDialog openFileDialog = new()
    {
        Filter = "Image files (*.png;*.jpeg)|*.png;*.jpeg|All files (*.*)|*.*"
    };

    if (openFileDialog.ShowDialog() != true) return;

    var fileName = openFileDialog.FileName;
    if (fileName == null) return;

    try
    {
        var image = new BitmapImage();
        image.BeginInit();
        image.UriSource = new Uri(fileName);
        image.EndInit();
        ProductImage = image;
    }
    catch (Exception)
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = "Please select a valid image",
            Type = NotificationType.Error
        });
    }
}

private async void OnInit()
{
    try
    {
        var categories = await _mediator.Send(new GetAllCategoriesQuery());
        Categories = categories.ToObservableCollection();
        var locations = await _mediator.Send(new GetAllLocationsQuery());
```

```
            Locations = locations.ToObservableCollection();
            var suppliers = await _mediator.Send(new GetAllSuppliersQuery());
            Suppliers = suppliers.ToObservableCollection();
        }
        catch (Exception e)
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = e.Message,
                Type = NotificationType.Error
            });
        }
    }
    }
```

Клас EditProductPageViewModel

```
using System;
using System.Collections.ObjectModel;
using System.Threading.Tasks;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using MediatR;
using Notification.Wpf;
using ShopManager.Application.CQRS.Category.Queries;
using ShopManager.Application.CQRS.Location.Queries;
using ShopManager.Application.CQRS.Product.Commands;
using ShopManager.Application.CQRS.Supplier.Queries;
using ShopManager.Domain.Extensions;
using ShopManager.Helper;
using ShopManager.Pages.Product;

namespace ShopManager.Viewmodels.Product;

using DbProduct = Domain.Entities.Database.Product;

public partial class EditProductPageViewModel : ObservableObject
{
    private readonly IMediator _mediator;
    private readonly INotificationManager _notificationManager;

    [ObservableProperty] private ObservableCollection<Domain.Entities.Database.Category> _categories;
    [ObservableProperty] private ObservableCollection<Domain.Entities.Database.Location> _locations;

    [ObservableProperty] private DbProduct _product;


    [ObservableProperty] private Domain.Entities.Database.Category _selectedCategory;
    [ObservableProperty] private Domain.Entities.Database.Location _selectedLocation;
    [ObservableProperty] private Domain.Entities.Database.Supplier _selectedSupplier;
    [ObservableProperty] private ObservableCollection<Domain.Entities.Database.Supplier> _suppliers;


    public EditProductPageViewModel(IMediator mediator, INotificationManager notificationManager)
    {
        _mediator = mediator;
        _notificationManager = notificationManager;
        OnInit();
    }


    [RelayCommand]
    private async Task SaveProduct()
    {
        await Task.Delay(1000);

        if (string.IsNullOrEmpty(Product.Name))
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Name is required",
```

```
            Type = NotificationType.Error
        });
        return;
    }

    if (string.IsNullOrEmpty(Product.BasePrice.ToString()))
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = "Price is required",
            Type = NotificationType.Error
        });
        return;
    }

    if (string.IsNullOrEmpty(Product.Quantity.ToString()))
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = "Quantity is required",
            Type = NotificationType.Error
        });
        return;
    }

    if (string.IsNullOrEmpty(Product.DiscountPercentage.ToString()))
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = "Discount percentage is required",
            Type = NotificationType.Error
        });
        return;
    }

    if (Product.BasePrice <= 0)
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = "Price must be greater than 0",
            Type = NotificationType.Error
        });
        return;
    }

    if (Product.Quantity <= 0)
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = "Quantity must be greater than 0",
            Type = NotificationType.Error
        });
        return;
    }

    if (Product.DiscountPercentage is < 0 or > 100)
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = "Discount percentage must be between 0 and 100",
            Type = NotificationType.Error
        });
        return;
    }

    if (SelectedCategory is null)
```

```
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = "Please select a category",
            Type = NotificationType.Error
        });
        return;
    }

    if (SelectedLocation is null)
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = "Please select a location",
            Type = NotificationType.Error
        });
        return;
    }

    if (SelectedSupplier is null)
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = "Please select a supplier",
            Type = NotificationType.Error
        });
        return;
    }

    try
    {
        var updateProductCommand = new UpdateProductCommand(Product);
        var updatedProduct = await _mediator.Send(updateProductCommand);
        if (updatedProduct is null)
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Failed to update product",
                Type = NotificationType.Error
            });

            return;
        }

        UiHelper.NavigateToPage(typeof(ViewAllProductsPage));
    }
    catch (Exception ex)
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = ex.Message,
            Type = NotificationType.Error
        });
    }
}


[RelayCommand]
private void NavigateBack()
{
    UiHelper.NavigateToPage(typeof(ViewAllProductsPage));
}


private async void OnInit()
{
    try
```

```
        {
            var categoriesTask = _mediator.Send(new GetAllCategoriesQuery());
            var locationsTask = _mediator.Send(new GetAllLocationsQuery());
            var suppliersTask = _mediator.Send(new GetAllSuppliersQuery());

            await Task.WhenAll(categoriesTask, locationsTask, suppliersTask);

            Categories = categoriesTask.Result.ToObservableCollection();
            Locations = locationsTask.Result.ToObservableCollection();
            Suppliers = suppliersTask.Result.ToObservableCollection();
        }
        catch (Exception ex)
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = ex.Message,
                Type = NotificationType.Error
            });
        }
    }
    }
```

Клас ViewAllProductsPageViewModel

```
using System.Collections.ObjectModel;
using System.Linq;
using System.Threading.Tasks;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using MediatR;
using Notification.Wpf;
using ShopManager.Application.CQRS.Product.Commands;
using ShopManager.Application.CQRS.Product.Queries;
using ShopManager.Domain.Extensions;
using ShopManager.Helper;
using ShopManager.Pages.Navigation;
using ShopManager.Pages.Product;
using DbProduct = ShopManager.Domain.Entities.Database.Product;
using DbLocation = ShopManager.Domain.Entities.Database.Location;
using DbCategory = ShopManager.Domain.Entities.Database.Category;

namespace ShopManager.Viewmodels.Product;

public partial class ViewAllProductsPageViewModel : ObservableObject
{
    private readonly IMediator _mediator;

    private readonly INotificationManager _notificationManager;
    [ObservableProperty] private ObservableCollection<DbProduct> _products;
    [ObservableProperty] private DbProduct _selectedProduct;

    public ViewAllProductsPageViewModel(INotificationManager notificationManager, IMediator mediator)
    {
        _notificationManager = notificationManager;
        _mediator = mediator;
        Initialize();
    }

    [RelayCommand]
    private async Task DeleteProduct()
    {
        if (SelectedProduct is null)
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Please select a product to delete",
                Type = NotificationType.Error
            });
            return;
        }
```

```
    _notificationManager.ShowButtonWindow("Are you sure you want to delete this product?", "Delete Product",
        async () =>
        {
            var result = await _mediator.Send(new DeleteProductCommand(SelectedProduct.Id));
            if (!result)
            {
                _notificationManager.Show(new NotificationContent
                {
                    Title = "Error",
                    Message = "Failed to delete product",
                    Type = NotificationType.Error
                });
                return;
            }

            Initialize();
        }, "Confirm delete");
    }

    [RelayCommand]
    private void EditProductPage()
    {
        if (SelectedProduct is null)
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Please select a product to edit",
                Type = NotificationType.Error
            });
            return;
        }

        UiHelper.NavigateToPage(typeof(EditProductPage));
        UiHelper.GetCurrentPage<EditProductPage>().ViewModel.Product = SelectedProduct;
    }

    [RelayCommand]
    private void NavigateBack()
    {
        UiHelper.NavigateToPage(typeof(ManagementPage));
    }

    [RelayCommand]
    private void CreateProduct()
    {
        UiHelper.NavigateToPage(typeof(CreateProductPage));
    }

    [RelayCommand]
    private void NextProduct()
    {
        if (SelectedProduct is null) return;

        var currentIndex = Products.IndexOf(SelectedProduct);
        if (currentIndex < Products.Count - 1) SelectedProduct = Products[currentIndex + 1];
    }

    [RelayCommand]
    private void PreviousProduct()
    {
        if (SelectedProduct is null) return;

        var currentIndex = Products.IndexOf(SelectedProduct);
        if (currentIndex > 0) SelectedProduct = Products[currentIndex - 1];
    }


    [RelayCommand]
    private void ClearFilters()
    {
        Initialize();
```

```
    }

    private async void Initialize()
    {
        var products = await _mediator.Send(new GetAllProductsQuery());
        Products = products.ToObservableCollection();

        SelectedProduct = Products.FirstOrDefault();
    }
    }
```

Клас CreateSellerPageViewModel

```
using System;
using System.Diagnostics;
using System.IO;
using System.Threading.Tasks;
using System.Windows.Media.Imaging;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using MediatR;
using Microsoft.Win32;
using Notification.Wpf;
using ShopManager.Application.CQRS.Seller.Commands;
using ShopManager.Helper;
using ShopManager.Pages.Seller;

namespace ShopManager.Viewmodels.Seller;

public partial class CreateSellerPageViewModel : ObservableObject
{
    private readonly IMediator _mediator;
    private readonly INotificationManager _notificationManager;
    [ObservableProperty] private string _description;
    [ObservableProperty] private string _name;
    [ObservableProperty] private string _password;

    [ObservableProperty] private BitmapImage _sellerImage;
    [ObservableProperty] private string _username;

    public CreateSellerPageViewModel(INotificationManager notificationManager, IMediator mediator)
    {
        _notificationManager = notificationManager;
        _mediator = mediator;
    }

    [RelayCommand]
    private async Task OkButtonClicked()
    {
        // Check if name is empty
        if (string.IsNullOrEmpty(Name))
        {
            _notificationManager
                .Show(new NotificationContent
                {
                    Title = "Error",
                    Message = "Please enter a name",
                    Type = NotificationType.Error
                });
            return;
        }

        try
        {
            var createdSeller =
                await _mediator.Send(new CreateSellerCommand(Name, Username, Password, Description, null));
            if (SellerImage != null)
            {
                var newImagePath = ImageHelper.GenerateImagePath(createdSeller);
                // Copy image to the path
                var oldImagePath = SellerImage.UriSource.LocalPath;
                File.Copy(oldImagePath, newImagePath, true);
                Debug.WriteLine(newImagePath);
```

```
            var updatedSeller = await _mediator.Send(new UpdateSellerCommand
            {
                SellerId = createdSeller.Id,
                ImagePath = newImagePath
            });

            if (updatedSeller.ImagePath != newImagePath)
                throw new InvalidOperationException("Image path not updated");
        }


        _notificationManager.Show(new NotificationContent
        {
            Title = "Success",
            Message = "Seller created successfully",
            Type = NotificationType.Success
        });

        UiHelper.NavigateToPage(typeof(ViewAllSellersPage));
    }

    catch (Exception ex)
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = $"An error occured while creating the seller: {ex.Message}",
            Type = NotificationType.Error
        });
    }
}

[RelayCommand]
private void NavigateBack()
{
    UiHelper.NavigateToPage(typeof(ViewAllSellersPage));
}

[RelayCommand]
private void OpenFileButtonClicked()
{
    OpenFileDialog openFileDialog = new()
    {
        Filter = "Image files (*.png;*.jpeg)|*.png;*.jpeg|All files (*.*)|*.*"
    };

    if (openFileDialog.ShowDialog() != true) return;

    var fileName = openFileDialog.FileName;
    if (fileName == null) return;

    try
    {
        var image = new BitmapImage();
        image.BeginInit();
        image.UriSource = new Uri(fileName);
        image.EndInit();
        SellerImage = image;
    }
    catch (Exception)
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = "Please select a valid image",
            Type = NotificationType.Error
        });
    }
}
    }
```

Клас EditSellerPageViewModel

```
using System;
using System.Threading.Tasks;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using MediatR;
using Notification.Wpf;
using ShopManager.Application.CQRS.Seller.Commands;
using ShopManager.Helper;
using ShopManager.Pages.Seller;
using DbSeller = ShopManager.Domain.Entities.Database.Seller;

namespace ShopManager.Viewmodels.Seller;

public partial class EditSellerPageViewModel : ObservableObject
{
    private readonly IMediator _mediator;
    private readonly INotificationManager _notificationManager;

    [ObservableProperty] private DbSeller _seller;


    public EditSellerPageViewModel(INotificationManager notificationManager, IMediator mediator)
    {
        _notificationManager = notificationManager;
        _mediator = mediator;
    }

    public DbSeller OriginalSeller { get; set; }

    [RelayCommand]
    private void Cancel()
    {
        UiHelper.NavigateToPage(typeof(ViewAllSellersPage));
    }

    [RelayCommand]
    private async Task Save()
    {
        if (string.IsNullOrEmpty(Seller.Name))
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Name is required",
                Type = NotificationType.Error
            });
            return;
        }

        if (string.IsNullOrEmpty(Seller.Username))
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Phone is required",
                Type = NotificationType.Error
            });
            return;
        }

        try
        {
            var updatedSeller = await _mediator.Send(new UpdateSellerCommand
            {
                Description = Seller.Description,
                Name = Seller.Name,
                SellerId = Seller.Id,
                Username = Seller.Username
            });

            if (updatedSeller is null)
            {
```

```
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Seller not found",
                Type = NotificationType.Error
            });
            return;
        }

        UiHelper.NavigateToPage(typeof(ViewAllSellersPage));
    }
    catch (Exception ex)
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = ex.Message,
            Type = NotificationType.Error
        });
    }
}
    }
```

Клас `public partial class ViewAllSellersPageViewModel : ObservableObject`

```
using System;
using System.Collections.ObjectModel;
using System.Linq;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using MediatR;
using Notification.Wpf;
using ShopManager.Application.CQRS.Seller.Queries;
using ShopManager.Helper;
using ShopManager.Pages.Seller;
using DbSeller = ShopManager.Domain.Entities.Database.Seller;

namespace ShopManager.Viewmodels.Seller;

public partial class ViewAllSellersPageViewModel : ObservableObject
{
    private readonly IMediator _mediator;
    private readonly INotificationManager _notificationManager;
    private int _currentIndex;

    [ObservableProperty] private DbSeller _selectedSeller;

    [ObservableProperty] private ObservableCollection<DbSeller> _sellers;

    public ViewAllSellersPageViewModel(IMediator mediator, INotificationManager notificationManager)
    {
        _mediator = mediator;
        _notificationManager = notificationManager;
        OnInit();
    }

    [RelayCommand]
    private void OpenEditSellerPage()
    {
        if (SelectedSeller is null)
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Please select a seller to edit",
                Type = NotificationType.Error
            });
            return;
        }

        NavigateToPage(typeof(EditSellerPage));
        var vm = UiHelper.GetCurrentPage<EditSellerPage>().ViewModel;
```

```
        vm.Seller = SelectedSeller;
    }

    [RelayCommand]
    private void NavigateToPage(Type pageType)
    {
        UiHelper.NavigateToPage(pageType);
    }

    [RelayCommand]
    private void NextSeller()
    {
        if (_currentIndex < Sellers.Count - 1)
        {
            _currentIndex++;
            SelectedSeller = Sellers[_currentIndex];
        }
    }

    [RelayCommand]
    private void PreviousSeller()
    {
        if (_currentIndex > 0)
        {
            _currentIndex--;
            SelectedSeller = Sellers[_currentIndex];
        }
    }

    private async void OnInit()
    {
        var sellers = await _mediator.Send(new GetAllSellersQuery());
        Sellers = new ObservableCollection<DbSeller>(sellers);
        _currentIndex = 0;
        if (Sellers.Any()) SelectedSeller = Sellers[_currentIndex];
    }
     }
```

Клас CreateSupplierPageViewModel

```
using System;
using System.Threading.Tasks;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using MediatR;
using Notification.Wpf;
using ShopManager.Application.CQRS.Supplier.Commands;
using ShopManager.Helper;
using ShopManager.Pages.Supplier;

namespace ShopManager.Viewmodels.Supplier;

public partial class CreateSupplierPageViewModel : ObservableObject
{
    private readonly IMediator _mediator;
    private readonly INotificationManager _notificationManager;
    [ObservableProperty] private string? _description;
    [ObservableProperty] private string? _name;

    public CreateSupplierPageViewModel(IMediator mediator, INotificationManager notificationManager)
    {
        _mediator = mediator;
        _notificationManager = notificationManager;
    }

    [RelayCommand]
    private void Cancel()
    {
        UiHelper.NavigateToPage(typeof(ViewAllSuppliersPage));
    }


    [RelayCommand]
```

```
    private async Task CreateSupplier()
    {
        if (string.IsNullOrEmpty(Name))
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Name is required",
                Type = NotificationType.Error
            });
            return;
        }

        var command = new CreateSupplierCommand
        {
            Name = Name!,
            Description = Description!
        };
        try
        {
            var supplier = await _mediator.Send(command);
            if (supplier is null)
            {
                _notificationManager.Show(new NotificationContent
                {
                    Title = "Error",
                    Message = "Supplier could not be created",
                    Type = NotificationType.Error
                });
                return;
            }

            _notificationManager.Show(new NotificationContent
            {
                Title = "Success",
                Message = $"Supplier {supplier.Name} created successfully",
                Type = NotificationType.Success
            });
            UiHelper.NavigateToPage(typeof(ViewAllSuppliersPage));
        }

        catch (Exception ex)
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = ex.Message,
                Type = NotificationType.Error
            });
        }
    }
```

Клас EditSupplierPageViewModel

```
using System;
using System.Threading.Tasks;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using MediatR;
using Notification.Wpf;
using ShopManager.Application.CQRS.Supplier.Commands;
using ShopManager.Helper;
using ShopManager.Pages.Supplier;
using DbSupplier = ShopManager.Domain.Entities.Database.Supplier;

namespace ShopManager.Viewmodels.Supplier;

public partial class EditSupplierPageViewModel : ObservableObject
{
    private readonly IMediator _mediator;
    private readonly INotificationManager _notificationManager;
    [ObservableProperty] private DbSupplier _supplier;
```

```
public EditSupplierPageViewModel(INotificationManager notificationManager, IMediator mediator)
{
    _notificationManager = notificationManager;
    _mediator = mediator;
}

[RelayCommand]
private async Task OkButtonClicked()
{
    if (string.IsNullOrEmpty(Supplier.Name))
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = "Name is required",
            Type = NotificationType.Error
        });
        return;
    }

    var command = new UpdateSupplierCommand
    {
        SupplierId = Supplier.Id,
        Name = Supplier.Name,
        Description = Supplier.Description
    };


    try
    {
        var result = await _mediator.Send(command);
        if (result is null)
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Supplier could not be updated",
                Type = NotificationType.Error
            });
            return;
        }

        _notificationManager.Show(new NotificationContent
        {
            Title = "Success",
            Message = "Supplier updated successfully",
            Type = NotificationType.Success
        });
        UiHelper.NavigateToPage(typeof(ViewAllSuppliersPage));
    }

    catch (Exception e)
    {
        _notificationManager.Show(new NotificationContent
        {
            Title = "Error",
            Message = $"Supplier could not be updated | {e.Message}",
            Type = NotificationType.Error
        });
    }
}

[RelayCommand]
private void Cancel()
{
    UiHelper.NavigateToPage(typeof(ViewAllSuppliersPage));
}
    }
```

Клас ViewAllSuppliersPageViewModel

```
using System;
```

```
using System.Collections.ObjectModel;
using System.Linq;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using MediatR;
using Notification.Wpf;
using ShopManager.Application.CQRS.Supplier.Commands;
using ShopManager.Application.CQRS.Supplier.Queries;
using ShopManager.Domain.Extensions;
using ShopManager.Helper;
using ShopManager.Pages.Supplier;
// Make sure you have a corresponding EditSupplierPage
using DbSupplier = ShopManager.Domain.Entities.Database.Supplier;

namespace ShopManager.Viewmodels.Supplier;

public partial class ViewAllSuppliersPageViewModel : ObservableObject
{
    private readonly IMediator _mediator;
    private readonly INotificationManager _notificationManager;
    private int _currentIndex;
    [ObservableProperty] private DbSupplier _selectedSupplier;

    [ObservableProperty] private ObservableCollection<DbSupplier> _suppliers;

    public ViewAllSuppliersPageViewModel(IMediator mediator, INotificationManager notificationManager)
    {
        _mediator = mediator;
        _notificationManager = notificationManager;
        OnInit();
    }

    [RelayCommand]
    private void DeleteSupplier()
    {
        if (SelectedSupplier is null)
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Please select a customer to delete",
                Type = NotificationType.Error
            });
            return;
        }

        _notificationManager.ShowButtonWindow("Delete supplier?", "Confirmation", async () =>
        {
            var removalResult = await _mediator.Send(new DeleteSupplierCommand(SelectedSupplier.Id));
            Suppliers.Remove(SelectedSupplier);
            SelectedSupplier = Suppliers.FirstOrDefault();
        }, "Confirm");
    }

    [RelayCommand]
    private void OpenEditSupplierPage()
    {
        if (SelectedSupplier is null)
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Please select a supplier to edit",
                Type = NotificationType.Error
            });
            return;
        }

        NavigateToPage(typeof(EditSupplierPage));
        var vm = UiHelper.GetCurrentPage<EditSupplierPage>().ViewModel;
        vm.Supplier = SelectedSupplier;
        // Assuming that the Supplier object also has a Clone method.
```

```
    }

    [RelayCommand]
    private void NavigateToPage(Type pageType)
    {
        UiHelper.NavigateToPage(pageType);
    }

    [RelayCommand]
    private void NextSupplier()
    {
        if (_currentIndex >= Suppliers.Count - 1) return;

        _currentIndex++;
        SelectedSupplier = Suppliers[_currentIndex];
    }

    [RelayCommand]
    private void PreviousSupplier()
    {
        if (_currentIndex <= 0) return;

        _currentIndex--;
        SelectedSupplier = Suppliers[_currentIndex];
    }

    private async void OnInit()
    {
        try
        {
            var result = await _mediator.Send(new GetAllSuppliersQuery());
            Suppliers = result.ToObservableCollection();
            _currentIndex = 0;
            if (Suppliers.Any()) SelectedSupplier = Suppliers[_currentIndex];
        }
        catch (Exception ex)
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = ex.Message,
                Type = NotificationType.Error
            });
        }
    }
     }
```

Клас BasketPageViewModel

```
using System;
using System.Collections.ObjectModel;
using System.Threading.Tasks;
using CommunityToolkit.Mvvm.ComponentModel;
using CommunityToolkit.Mvvm.Input;
using MediatR;
using Notification.Wpf;
using ShopManager.Application.CQRS.Basket;
using ShopManager.Helper;
using ShopManager.Pages.Order;

namespace ShopManager.Viewmodels;

using DbProduct = Domain.Entities.Database.Product;

public partial class BasketPageViewModel : ObservableObject
{
    [ObservableProperty] private ObservableCollection<DbProduct> _products;
    [ObservableProperty] private DbProduct? _selectedItem;
    private readonly INotificationManager _notificationManager;
    private readonly IMediator _mediator;

    public BasketPageViewModel(INotificationManager notificationManager, IMediator mediator)
    {
```

```
        _notificationManager = notificationManager;
        _mediator = mediator;
        OnInit();
    }

    private async void OnInit()
    {
        try
        {
            var basketProducts = await _mediator.Send(new GetAllProductsFromBasketQuery());
            Products = new ObservableCollection<DbProduct>(basketProducts);
        }
        catch (Exception ex)
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = ex.Message,
                Type = NotificationType.Error
            });
        }
    }

    [RelayCommand]
    private void NavigateBack() => UiHelper.NavigateToPage(typeof(CreateOrderPage));
    [RelayCommand]
    private async Task RemoveSelectedProductFromBasket()
    {
        if (SelectedItem is null)
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "No product selected",
                Type = NotificationType.Error
            });
            return;
        }

        var result = await _mediator.Send(new RemoveProductFromBasketCommand
        {
            ProductId = SelectedItem.Id,
        });

        if (result)
        {
            Products = new ObservableCollection<DbProduct>(await _mediator.Send(new GetAllProductsFromBasketQuery()));
        }
        else
        {
            _notificationManager.Show(new NotificationContent
            {
                Title = "Error",
                Message = "Could not remove product from basket",
                Type = NotificationType.Error
            });
        }
    }

    [RelayCommand] private void NavigateToPage(Type pageType) => UiHelper.NavigateToPage(pageType);
     }
```

Клас public partial class LoginSellerWindowViewModel : ObservableObject

```
using System;
using System.Linq;
using System.Threading.Tasks;
using System.Windows;
using CommunityToolkit.Mvvm.ComponentModel;
```

```
using CommunityToolkit.Mvvm.Input;
using MediatR;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.FeatureManagement;
using Notification.Wpf;
using ShopManager.Configuration;
using ShopManager.Pages;
using ShopManager.Services;

namespace ShopManager.Viewmodels;
public partial class LoginSellerWindowViewModel : ObservableObject
{
    private readonly IAuthorizationService authorizationService;
    private readonly IMediator mediator;
    private readonly INotificationManager notificationManager;
    private readonly IServiceProvider serviceProvider;

    [ObservableProperty] private string _password = string.Empty;
    [ObservableProperty] private string _username = string.Empty;

    public LoginSellerWindowViewModel(IMediator mediator, IServiceProvider serviceProvider,
        INotificationManager notificationManager, IFeatureManager featureManager, IAuthorizationService authorizationService)
    {
        this.mediator = mediator;
        this.serviceProvider = serviceProvider;
        this.notificationManager = notificationManager;
        this.authorizationService = authorizationService;
        CheckForAdminMode(featureManager);
    }

    private async void CheckForAdminMode(IFeatureManager featureManager)
    {
        if (!await featureManager.IsEnabledAsync(Features.Admin.IsEnabled)) return;
        Username = "Admin";
        Password = "12345";
    }

    [RelayCommand]
    private async Task Login()
    {
        if (string.IsNullOrEmpty(Username))
        {
            notificationManager.Show(new NotificationContent
            {
                Title = "Login failed",
                Message = "Username is empty",
                Type = NotificationType.Error
            });
            return;
        }

        if (string.IsNullOrEmpty(Password))
        {
            notificationManager.Show(new NotificationContent
            {
                Title = "Login failed",
                Message = "Password is empty",
                Type = NotificationType.Error
            });
            return;
        }

        var isUserRegistered = await authorizationService.AuthorizeSeller(Username, Password);
        if (!isUserRegistered)
        {
            notificationManager.Show(new NotificationContent
            {
                Title = "Login failed",
                Message = "Username or password is incorrect",
                Type = NotificationType.Error
            });
            return;
```

```
    }

    var mainWindow = serviceProvider.GetRequiredService<MainWindow>();
    mainWindow.Show();

    var windows = System.Windows.Application.Current.Windows;

    if (windows.Count > 1)
    {
        var loginWindow = windows.OfType<LoginSellerWindow>().FirstOrDefault();
        loginWindow?.Close();
    }
}
    }
```